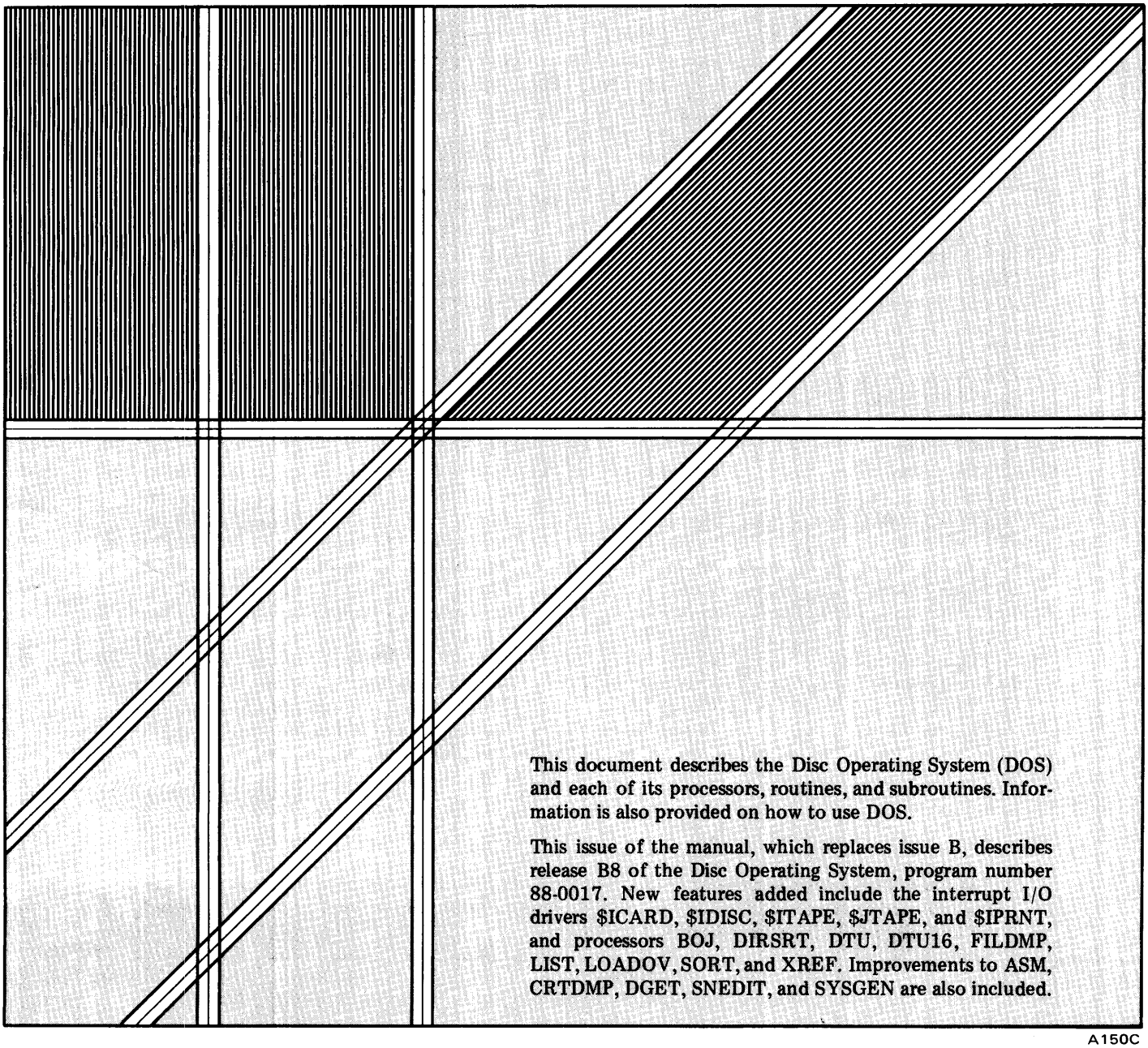
Four-Phase Systems, Inc.
10420 North Tantau Avenue
Cupertino, California 95014

Document Number SIV/70-50-1C
Stock Number 37C
Issue A: 15 August 1971
Issue B: 1 April 1972
Issue C: 15 April 1973

Price: $3.50

System IV/70

Disc Operating System (DOS) Reference Manual



This document describes the Disc Operating System (DOS) and each of its processors, routines, and subroutines. Information is also provided on how to use DOS.

This issue of the manual, which replaces issue B, describes release B8 of the Disc Operating System, program number 88-0017. New features added include the interrupt I/O drivers $ICARD, $IDISC, $ITAPE, $JTAPE, and $IPRNT, and processors BOJ, DIRSRT, DTU, DTU16, FILDMP, LIST, LOADOV, SORT, and XREF. Improvements to ASM, CRTDMP, DGET, SNEDIT, and SYSGEN are also included.

A150C

LIST OF EFFECTIVE PAGES

THIS PUBLICATION CONTAINS 142 PAGES CONSISTING OF THE FOLLOWING:

* The asterisk indicates pages changed, added, or deleted by the current change. The portion of the text affected by the current change (or the portion of the text changed for this issue) is indicated by a vertical line in the outer margins of the page.

A337A

Contents

Introduction

DOS Routines and Processors

Appendices

Glossary

Illustrations

INTRODUCTION

GENERAL

The Disc Operating System (DOS) is a collection of programs and routines that provide an easy-to-use interface between the user and the System IV/70 computer with disc drive system. DOS allows the user to write, assemble, and execute programs using as input either a card reader, the disc file, or a video/keyboard terminal, and to obtain output using either the display, a printer, or some other system output device. Using the routines provided by DOS, the user may perform a complete range of data-file oriented processes, including storage, retrieval, and processing of data, with a minimum of effort. These routines provide support for the keyboard, disc, and card reader as input devices, and for the disc, printer, and video display as output devices. The disc is considered to be the principal mass storage device for the system, but provision is also made for supporting magnetic tape mass storage. Four-Phase supported software for use with magnetic tape systems is provided†. A typical DOS configuration is shown in Figure 1.

The principal control program under DOS is the DOS MONITR, which controls allocating of space on the disc, inputting and outputting of data, and related functions. MONITR controls the disc directory, a table which associates file names, addresses, and other attributes. Since the directory contains all the information required to control the disc, and its misuse can destroy the contents of a cartridge, only MONITR and its associated routines are allowed to directly access the directory. In addition, MONITR possesses

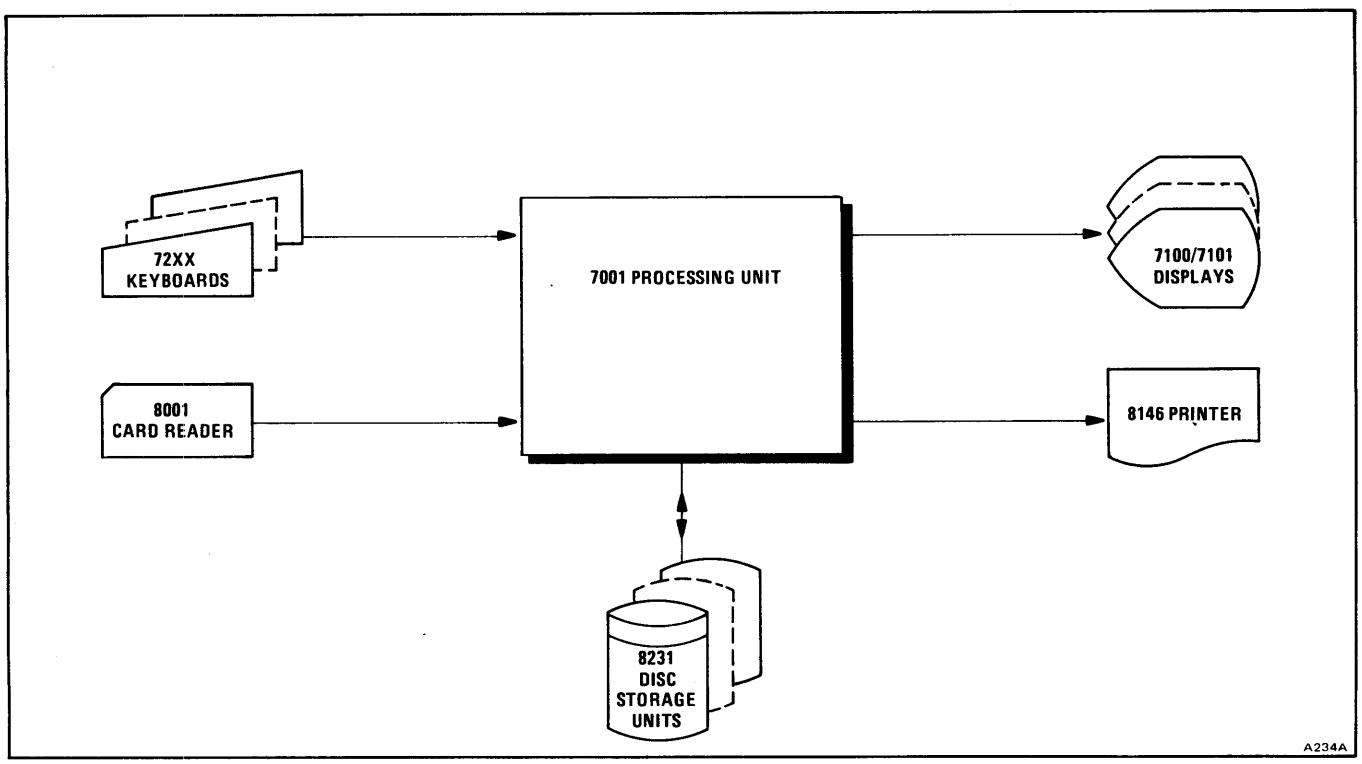


Figure 1. A Typical System IV/70 DOS Configuration

† See "Peripheral Unit Programming Manual", document SIV/70—40—1; also see $ITAPE and $JTAPE in this manual.

utility routines that control access to the disc and other peripherals, and perform other standard system functions. These routines can be used readily by system users. MONITR also controls the access to various system processors, including the assembler and relocatable loader. These two processors are used to process assembly language source files and relocatable code to produce object code which the computer can execute. Similarly, the DOS routines possess the logic required to process DOS-language control and parameter statements.

DOS typically operates under control of one video/keyboard terminal, loading and executing programs in a serial, stacked job, batch manner, with no thought given to time sharing within DOS, although user programs may time share the system in any desired manner. DOS performs all its I/O operations in a polling manner with the computer's interrupt structure disabled, although user programs will regularly use the interrupt structure.

In general, there is no requirement for any part of the operating system to be in memory while the user's program is operating; a brief routine, EXIT, is provided to return control to MONITR when the user program is finished. However, if the user program can reside above 03400 in memory, it can have access to MONITR routines for various purposes, if these are called correctly at load time. In general, linkage between DOS routines and user programs is performed using locally undefined symbols (virtuals) and ENTRY directives in the routines where the symbols are defined. Program libraries in relocatable form are provided to help resolve these linkages.

System processors and other named programs and files on the disc are available in absolute form and are accessed directly using control statements. Examples of such processors are the assembler and relocatable loader and disc copy, dump, and edit programs. The *control statements* calling these processors specify program names and parameters, such as input and output files and other options.

Programs for execution under DOS are either memory loads or control files. A *memory load* (load module) is a program in absolute binary form that is executable; all inter-routine cross references, or linkages, have been resolved. It is thus executed in a load-and-go manner, without further system intervention unless the program calls for it. A *control file* is a source file consisting of control statements that call and execute programs, routines, and control files as a single thread; i.e., the user may construct any linear sequence of system and/or user programs for execution using control files. One control file may call another; such calls may be stacked up to a maximum of 10 levels. The processor LIST is provided to enable listing of control files without need for executing them.

One tool provided for the user's convenience is the directory dump (DIRDMP), a program that prints the disc directory; another is DRFND, which searches the directory and provides the disc location and attributes of a file, given its name. Another tool is the allocator, which assists the user in placing his files on the disc efficiently. Other programs and routines allow the user to update files, to input or output data, to modify the directory, and to return storage whose usage is complete to an availability table. These programs operate under control of the batch monitor, MONITR. The monitor and its attendant routines are designed to be operable with user programs, if desired. In general, programs that reside above MONITR so that they can operate with DOS routines, sacrifice the use of the interrupt system and the video display areas located in the portion of memory where MONITR is stored.

Four-Phase supported system and applications programs are generally designed to be used with DOS, in that they can be called and executed as memory loads, relocatable files, or control files. These programs include the CODE assembler and its relocatable loader, SORT, the 2260/2848 and 3270 Simulators, and the Video Display Library. These programs are designed to furnish their own input/output operations and most will operate as memory loads.

Programs and routines are furnished as disc files in any or all of three forms: source, relocatable, and absolute. Programs in source form are expressed in the job control language (JCL) of DOS (e.g., "// LOADOV", see "Using the Disc Operating System") or the CODE assembler language; they must be processed respectively by DOS to be executed or by the Assembler to produce relocatable files. Relocatable files are routines that may possess ENTRY statements and/or unresolved virtuals to enable them to be linked to other routines when processed by the relocatable loader to produce absolute files which may be executed by the computer.

Various programs are furnished in different forms as required for appropriate operation; the description of each program notes the forms in which the program is provided. All system programs are provided in source form for back-up and documentation purposes, and little-used or unsupported programs are also provided in this form if the user wishes to assemble and load them. Files in this form may also be edited and deleted readily; in general the source files will be kept on a back-up or master disc and deleted from working files. A control procedure, DELSYS, is provided for this purpose.

Programs provided in relocatable form are intended to be linked to other programs by the relocatable loader; many such programs are provided in SYSLIB and other system libraries. Such programs are linked to using a BRM or BAL instruction within the assembly language form of the user program. Data files with virtuals and/or entries can also be processed in relocatable form.

Absolute programs include processor files, data files, and any user-defined programs such as application programs; they are invoked from the disc using DOS language control statements. Many of these are simply "load-and-go" programs that take control and execute when loaded. For an overall picture of the operation of DOS, see Figure 2.

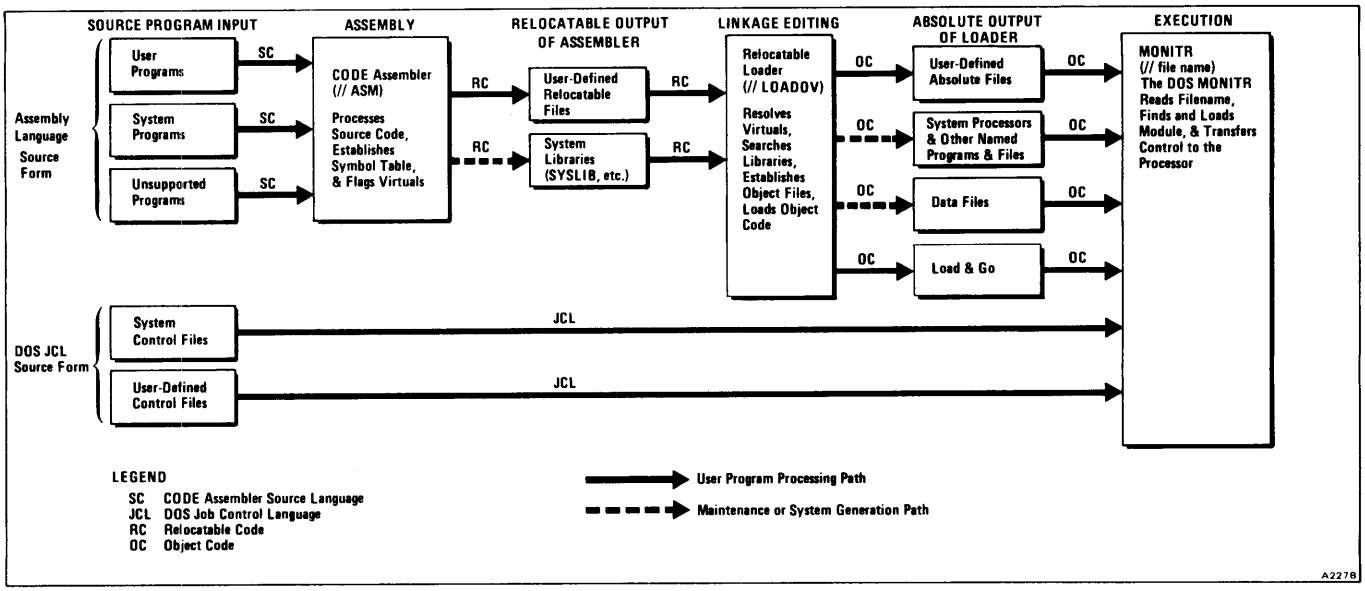


Figure 2. Overall DOS Operation

USING THE DISC OPERATING SYSTEM

When the disc system is bootstrapped into the computer, DOS will automatically set the first video/keyboard terminal as the system input device; i.e., control input will come from the first keyboard. The system input routine SYSIN will then accept control statements in DOS Language from this keyboard. *Control statements* are of two types: processor statements and parameter statements. A *processor statement* consists of the name of a file on disc 0, which may be a program or a control file. The user may invoke any such file using a single processor statement; the format is:

```
//bzzzzzz
```

where b is a blank and zzzzzz is a file name of six characters or less in the 64-character ASCII subset (see Appendix D), with the first character alphabetic. The name must appear in columns 3 through 9; characters in columns 10 through 80 are treated as comments.

A processor statement may be modified by one or more parameter statements, depending on its particular requirements. Typically, *parameter statements* specify input or output file names and devices, and other options for the processor statements they accompany. The format for most parameter statements is a slash followed by one or more fields separated by commas. A period causes the rest of the statement to be ignored. All blanks are ignored. Each field is of the form "KEYWORD = VALUE @ DRIVE". The keyword is identified only by its first letter, so that abbreviations may be used. If the value is a name, only the first 6 characters will be used. Drive 0 is used if no drive is specified. If a drive is specified, it must be by a number in the range 0-7. Typical parameter statements are:

```
/INPUT = TEMP.
/OUTPUT = RELOC@1.
```

The subfields "=", "VALUE", and "@ DRIVE" are optional. Thus, some parameter statements have a format consisting of a slash followed only by a name of six characters or less. These statements are generally used to specify binary (yes/no) options.

Note that the character set of the DOS Language is the 64-character subset of ASCII, as opposed to the 120-character superset of ASCII which is the repertoire of the video/keyboard display system. See Appendix D for a list of the character sets. DOS is limited to 64 characters so that punched cards and standard printers may be used without modification or confusion. However, on a system with an 8147 or 8152 Printer, the full 96 characters can be entered from the keyboard and printed. The assembler will also accept lower-case data through the DCA directive. Location LUCONV in MONITR (0035 absolute) should be zero to suppress conversion of lower case to upper case.

Using the appropriate control statements, the user may select various options:

a. To execute a previously stored program that is a memory load or control file, enter a processor statement using the name of the program, e.g., // COPY or // DIRDMP. If entered from the keyboard, the cursor return marks the end of the statement; if entered on cards, the end of the card is the punctuation. The processor statement must be followed by any required parameter statements. Note that the user's program must read (using OPTION) and use the parameter statements specified for any processor statements he creates.

b. To execute a program that is stored on the disc in relocatable form, use the following control statements:

```
// LOADOV
/I=yyyyyy.
//
```

Where yyyyyy is the name of the relocatable program and // is used to end the parameter statements for LOADOV. "// LOADOV" specifies the relocatable loader; the older // LOAD may also be used.

c. To assemble a program that is stored on the disc in source form, use the following control statements:

```
// ASM
/I=xxxxxx, O=yyyyyy.
//
```

Where xxxxxx is the name of the source program and yyyyyy is the name given to the relocatable object program produced by the assembler. "// ASM" specifies the assembler.

d. To create a memory load from a relocatable program stored on the disc, use the following control statements:

```
// LOADOV
/I=yyyyyy, O=zzzzzz.
//
```

Where yyyyyy is the name of the reloctable program and zzzzzz is the name given to the memory load. Up to 15 relocatable files may be loaded and linked using the relocatable loaders; see the descriptions of the processors LOAD and LOADOV for details.

e. To obtain an index of the system programs and routines supplied with the disc, type "// DIRDMP" followed by "//b". The disc directory will be listed on the system output device. This directory contains the names and related information about the files on the disc. See the description of the processor DIRDMP for an explanation of the list.

f. To transfer control from the first keyboard to another system input device, type a processor statement with the appropriate name, e.g., "// CARDS" for the card reader. Such an option may be used to load programs, data, or other information from the system input device. To return control from the card reader to the keyboard, put // KYBRD at the end of a card deck.

g. To load a source program onto the disc, use the following card deck:

```
// SINDSK
/O=xxxxxx.
      .
      .
      .
source program
      .
      .
      .
//                                  (End of file card)
blank card
```

Where xxxxxx is the name given to the source program. "// SINDSK" is the disc input program; it copies the input stream from the system input device to the disc.

h. To load a control file onto the disc from cards, use the following card deck:

```
// SINDSK
/O=xxxxxx, END=**, FLAG=C.
//
     .
     .
     .
Processor and parameter statements.
     .
     .
     .
//                                  (End of file card)
**
blank card
```

The "END=**" statement establishes an end-of-file mark different from the usual system "// " and prevents the processor statements in the control file from being executed. The control file is ended by "**". The processor statement(s) may call control files as well as absolute programs. Up to 10 such control files may be stacked. The control file may be invoked by using "// xxxxxx" where xxxxxx is the name given to the control file.

i. To assemble, load, and execute a program from cards, use the following card deck:

```
// SINDSK
/O=TEMP.
      .
      .
      .
source program
      .
      .
      .
// ASM
/I=TEMP, O=TEMP.
// LOADOV
/I=TEMP.
//
blank card
```

This sequence will assemble, load, and execute. Note that the files are assigned to temporary storage using "TEMP"; this is encouraged since JOB will return this scratch to the disc availability table. The four letters "TEMP" may have one or two more characters attached to create numerous scratch files; JOB only looks for the first four letters when performing garbage collection. Examples: TEMPA, TEMP01.

SYSTEM PROCESSORS

System processors are programs that are provided in absolute form and are callable using DOS language control statements. Some system processors are provided as independent programs that function with MONITR in memory and others overlay MONITR when loaded and function without its being resident. The system processors are described in the following paragraphs.

Disc Resident Monitor (MONITR)

MONITR uses the routines SYSIN and SYSOUT. SYSIN accepts input from one of the system input devices (cards, disc, or keyboard) and passes it to MONITR. MONITR uses the routine SYSOUT to print the input, a record at a time, on the system output device. Then MONITR processes the record. When a statement with "//b" in bytes 1-3 is encountered, the contents of bytes 4-9 is treated as a name and looked up in the disc directory. If the name is that of a memory load, the memory load is loaded and executed. If the name is that of a source file, that file is used as SYSIN (the system input file) until an end-of-file (//b or other EOF as defined in certain processors) is encountered. This process is shown in more detail in the flowchart under "MONITR" in this document.

Along with SYSIN and SYSOUT, MONITR also contains the routines OPTION (read parameter statements), $DISC (perform disc I/O), DRFND (find a disc file in the directory), DRMOD and DRUPD (modify the directory), and ALLOC and ALLOC1 (allocate disc files). These routines are used by MONITR but may also be accessed by user programs. See "MONITR" and the descriptions of the other routines for details.

The routine EXIT (furnished in relocatable form) may be used by any program to return control to MONITR.

Assembler Program (ASM)

This is the CODE assembler described in Section 8 of the "Computer Reference Manual", Four-Phase document SIV/70—11—1. MONITR is overlayed by this processor.

Return Deleted Sectors (BOJ, JOB)

BOJ and JOB function as the DOS garbage collectors returning deleted sectors to the disc availability table. If the name of a file has been blanked out (e.g., by DRUPD) or if the file has a name whose first four letters are TEMP, JOB will delete the file. BOJ returns sectors that are missed by JOB. These processors overlay MONITR.

Card-To-Disc File Restoration (CDDC)

CDDC is used for restoring destoryed disc files from back-up cards and for creating contiguous disc files. CDDC resides with MONITR and uses its facilities.

Disc-To-Disc Copier (COPY)

COPY is used to copy any disc file from one location to another, either on the same or separate discs. COPY resides with MONITR and uses its facilities.

Disc Dubber (COPY01)

COPY01 copies the complete contents of one cartridge to another. This processor overlays MONITR.

Display and Edit Disc or Memory Files (CRTDMP)

CRTDMP is used to display files from memory or the disc on a video screen and to edit them from the keyboard if required. This processor overlays MONITR.

Print the Disc Directory (DIRDMP)

DIRDMP is used to print the disc directory on the system output device. It resides with MONITR and uses the facilities of that program.

Modify the Disc Directory (DIRMOD)

DIRMOD furnishes the user with a means for manipulating file names and values inside the disc directory. DIRMOD resides with MONITR and uses its facilities.

Sort the Disc Directory (DIRSRT)

DIRSRT sorts the entries in the disc directory into ASCII sequence; it may then be printed using DIRDMP. DIRSRT resides with MONITR.

Print a Disc File (FILDMP, LIST)

FILDMP allows a chained or contiguous disc file to be printed in octal, with ASCII interpretation. LIST allows control files to be printed without being executed. These processors reside with MONITR.

Load Relocatable Files (LOAD, LOADOV)

These are two versions of the relocatable loader described in Section 8 of the "Computer Reference Manual", Four-Phase document SIV/70—11—1. The loaders overlay MONITR. Use of the improved loader, LOADOV, is encouraged.

Read Octal Cards (OCTAL)

OCTAL enables cards to be read in the standard Four-Phase octal format. OCTAL overlays the monitor.

Copy Input to Disc (SINDSK)

SINDSK copies SYSIN, the system input stream, onto specified disc files. SINDSK resides with MONITR and uses its facilities.

Edit Disc Files (SNEDIT)

SNEDIT allows the user to number and edit card images on the disc. SNEDIT resides with MONITR and uses its facilities.

Sort Chained Files (SORT)

SORT sorts chained files (e.g., files created by SINDSK) into any specified collating sequence.

Cross Reference ASM Symbols (XREF)

XREF creates a sorted file of all symbols in an ASM file and prints the sorted file. The file also contains the locations of each symbol keyed to SNEDIT-assigned sequence numbers.

DOS MONITR ROUTINES

The DOS MONITR contains a number of widely used routines. The user can take advantage of these routines as long as the program is loaded above MONITR. To use these routines, include /I = SYSTEM in the load step (LOADOV) and use the calling sequence given in the section of this manual for the desired routine. Note that /I = SYSTEM is not required for SYSIN, SYSOUT, SYSJCT, $DISC, and OPTION since these routines are in the system relocatable library (SYSLIB). However, the MONITR versions of SYSIN and OPTION are required if these routines are to process disc control files.

System Input (SYSIN)

This input routine works normally with MONITR. SYSIN reads 27-word records from the specified system input device (for example, card images). SYSIN may be used by user programs, if called with the resident monitor or through the relocatable library SYSLIB. Note, however, that if SYSIN is to be used without MONITR, the logic required for processing disc control files will not be present and the user program will not be able to access the disc through DOS; the keyboard and card reader will be serviced properly, however. The keyboard input under SYSIN performs simple editing: a nonblinking, destructive cursor is displayed on the screen and the ← and → keys and the space bar can move the cursor left and right.

System Output (SYSOUT)

This routine will list a record on the current system output device, normally the printer. The maximum length of a record for SYSOUT is the line length of the printer in use. To eject a page, the subroutine SYSJCT is provided.

Disc I/O ($DISC)

This routine will read and write sectors and perform error checking with the disc. $DISC runs using status, not the interrupt system of the computer.

Locate Disc Files (DRFND)

Given the name of a disc file, DRFND will search the disc directory and return the location and other information concerning the file. The format of the directory entries is shown in Figure 3.

Disc Directory Modifier (DRMOD)

This routine is used only after a file is located using DRFND; its function is to change the name and values of the entry last found by DRFND. A special calling sequence is provided for overwriting protected files.

File Creating (DRUPD)

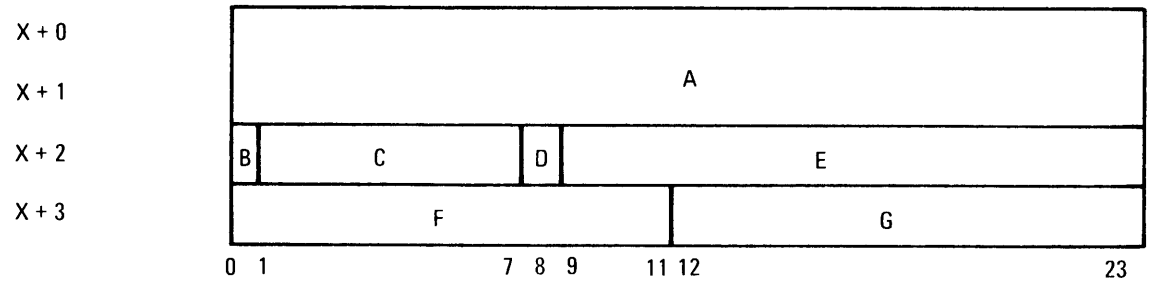
DRUPD is used to update files on the disc and create new files. If old files are found with the same name as the new file, the names of the old files will be blanked and the files will be returned to the disc availability table the next time JOB is called. If the protect bit is on, the old file will not be deleted and the new file will not be created, unless a special calling sequence is used.

Parameter Reading (OPTION)

OPTION reads and lists parameter statements for any processor that requires them. Each parameter field is of the form "KEYWORD=VALUE" or "KEYWORD", as explained above under "Using the Disc Operating System". OPTION may be used by user programs if called under MONITR or in relocatable form under SYSLIB. However, if the SYSLIB version is used only the card reader and the keyboard will be handled correctly.

Allocating Sectors (ALLOC, ALLOC1)

ALLOC and ALLOC1 are used to allocate a specified number of sectors on the disc. ALLOC1 is used when chained files are created; ALLOC will allocate from 1 to 199*16 contiguous sectors. These routines use $DISC to read the disc availability table; they return the starting sector address for use by the calling program.



Word	Bits	Symbol	Meaning
X + 0, X + 1	0-23 0-23	A	Six character ASCII name with all parity bits 0.
X + 2	0	B	1 = protected 0 = automatic deletion rule applies, i.e., a file is deleted whenever a new file of the same name is created.
	1-7	C	Flag byte. Seven bits of information available to application programs.
	8	D	Format bit. 0 = contiguous 1 = chained
	9-23	E	Contiguous files: E = 0 if data, otherwise E is load address in octal of memory load. Chained files: E = ending sector in octal; 06177 (3199) maximum.
X + 3	0-11	F	Sector count -1 in octal, F≤3199.
	12-23	G	Starting sector in octal; 06177 (3199) maximum.

A131B

Figure 3. Format of Directory Entries

SYSTEM RELOCATABLE LIBRARY ROUTINES

Various relocatable files are provided for the user's convenience in putting together programs. These files are provided in system libraries, which may be accessed using the relocatable loader. Relocatable libraries may be chained together using EOP statements as explained below. The principal system library is SYSLIB, which contains many widely used routines. SYSLIB is the default library used in resolving virtuals if no other library is specified. Also furnished are CHLL, CHRR, CHML, and CHMR, which provide sequential access to byte-files on disc; the different libraries are used depending on desired byte alignments and whether one or multiple byte-files are to be open at the same time. The routines DGET, DPUT, MDGET, and MDPUT reside in these four libraries; see the descriptions of the four routines for further information. Another library is R17-Q, which contains interrupt driven I-O drivers for card reader, magnetic tape, disc, and printer. All five of these libraries link to SYSLIB if further virtuals remain to be resolved when their processing is complete. COBLIB, the COBOL object time library, links to R17-Q. SYSLIB is designed to be updated as programs for the user's convenience are added.

The user may also create additional libraries in source form, assemble them, and refer to them using the /L parameter under // LOADOV program. The library may be any named file with ENTRY statements for the resolution of virtuals; the last statement should be an EOP statement in assembly language form. If a label is attached to an EOP, the loader will search for a file with the name specified in the label, so that libraries may be chained, and a user library can link to SYSLIB. Note that in any library with numerous relocatable files and virtuals, all occurrences of a given symbol as a virtual must proceed the occurrence of the symbol as an ENTRY. Example:

```
// LOADOV
/INPUT = IN.
/LIBRARY = LIB.
//
```

These files would be constructed as follows:

```
IN      (text)
        .
        .
        .
        END

LIB     (text)
        .
        .
        .
        END
SYSLIB  EOP

SYSLIB  (text)
        .
        .
        .
        END
        EOP
```

SYSLIB furnishes its own EOP statement. The EOP statement is used only in linking libraries together. If a library is to be fetched under // LOAD or // LOADOV, and its name is not specified with a /L parameter, the name of the library must appear in the *label* field of an EOP assembly language statement.

Only one /L parameter may be given with each // LOAD or // LOADOV statement. Note also that if all virtuals are properly resolved using a given library, further libraries will not be searched even if specified. The files shown above could also be constructed using the following procedure:

```
// LOADOV
/I = IN.
//
```

These files would be constructed as follows:

```
IN      (text)
        .
        .
        .
        END
LIB     EOP

LIB     (text)
        .
        .
        .
        END
        EOP

SYSLIB  (text)              (default if required)
        .
        .
        .
        END
        EOP
```

If this method is used, the user must avoid the following situation:

```
// LOADOV
/I = F1, I = F2.
//
```

Where F1 ends with F3 EOP, and F3 ends with EOP. If the library F3 resolves all the virtuals in F1, then F2 will not be loaded. The two ways of avoiding the problem are to restrict the use of EOP's to library files or to keep one or more unresolved virtuals to the last input file.

Current SYSLIB and R17-Q routines include:

Call Monitor (EXIT)

This routine may be used by a program to return control to MONITR. Depending on the calling sequence used, EXIT may simply return control to MONITR or it may specify the next processor.

Disc I/O ($DISC, $IDISC)

These routines will read and write sectors and perform error checking with the disc. They are both available in source and relocatable form. $DISC runs using status, not the interrupt system of the computer, whereas $IDISC uses the interrupt system.

Other Interrupt I/O Drivers ($ICARD, $IPRNT, $ITAPE, $JTAPE)

Interrupt driven I/O utilities for card reader, printer, and magnetic tape are furnished in source and relocatable form.

Character Manipulation Instruction Tables

The character manipulation instructions are described in Section 5 of the "Computer Reference Manual", Four-Phase document SIV/70—11—1. The tables are described and listed there. The entries to these tables are designed to be left as virtuals in user programs and fetched as required from SYSLIB.

Memory Dump ($DUMP)

$DUMP provides for dumping any portion or all of memory onto the system output device. The calling sequence is:

```
BRM     $DUMP
 PZE    FWA         Address of first word to be dumped
 PZE    LWA         Address of last word to be dumped
```

Parameter Reading (OPTION)

This is the OPTION routine described under "DOS Routines". For further details, see "OPTION" section of this manual.

System Input (SYSIN)

This is the SYSIN routine described under "DOS Routines". For further details, see "SYSIN" section of this manual.

System Output (SYSOUT, SYSJCT)

These are the SYSOUT and SYSJCT routine described under "DOS Routines". For further details, see "SYSOUT" section of this manual.

Card Input (CARDIN)

This is the card read routine described and listed in Section 8001 of "Peripheral Unit Programming", Four-Phase document SIV/70—40—1. The calling sequence is:

```
BAL     CARDIN
 DCN    n  (n=0, pack binary; =1, unpack binary; =2, pack ASCII; =3, unpack ASCII)
 DCN    BUFFER ADDRESS
        RETURN
```

Note that this routine does not furnish a line feed after each card image as does the SYSIN routine; i.e., it makes no provision for formatting card images for the printer or other output device. It also does not remove parity from ASCII mode card images.

PROGRAMS FURNISHED IN SOURCE FORM ONLY

Aside from the source versions of system processors and routines furnished for back-up purposes, certain little-used and nonsystem-oriented programs are furnished as assembly language source files. They may be assembled and loaded as the user requires. The source files may be deleted from working cartridges using the system processor DELSYS. The programs furnished in source form only are:

2260 Simulator

The remote or local version of the simulator is furnished with some DOS systems. For usage of the simulator see "2260/2848 Simulator Reference Manual", Four-Phase document SIV/70—55—1.

Video Display Library

The video display library is documented in "Video Display Library Reference Manual", Four-Phase document SIV/70—54—1. It is intended that users assemble this system as a library to be used in relocatable form with other routines. "SYSLIB EOP" is placed at the end of the source program for this purpose; linkage to SYSLIB for further resolution of virtuals is thus automatic.

Hollerith/ASCII/EBCDIC Conversion Routines

Routines for converting between Hollerith, ASCII, and EBCDIC are furnished in source form as P67-A. The user may assemble and use these; see the "COBOL Programmer's Guide", document SIV/70—45—1 issue A or later.

Disc Files to Cards (DCCD)

DCCD is furnished in source form only so that the user may attach his own punch driver using SNEDIT. See "DCCD" in this document.

CORRECTION AND DUPLICATION OF CARTRIDGES

Various means are provided for correcting errors and duplicating cartridges (see Figure 4), although duplication can only be performed efficiently using two disc drives. With two drives, duplication can be performed quickly and readily using system processor COPY01, which simply transfers the complete contents of the cartridge on drive 0 to the cartridge on drive 1. Similarly, processor COPY provides an easy means for transferring a file from one cartridge to another.

For performing corrections, the system processor SNEDIT is provided (which can perform insert, delete, replacement, and merge editing on any files with card-image records), and all system master discs are provided with complete source files of all supported system programs. Authorized Program Analysis Request (APAR) corrections to programs supported under DOS will be distributed in a form suitable for use with SNEDIT. It is anticipated that such corrections will be performed by the customer (or a Four-Phase Systems Branch Office) on the customer's master cartridge, then transferred to his working cartridges. Note, however, that if widely shared programs, such as those in SYSLIB, are changed, this may necessitate the customer's reassembling and/or reloading working programs with linkages to the system programs changed.

This method will not suffice for changes to MONITR, the system resident monitor program, which accesses the directory and which has numerous entry points, any one of which might be moved by a programming change. Each entry point is accessed by many different system routines, so that any change in the monitor can be expected to have consequences throughout the system. Whenever a change is made in MONITR, therefore, it is necessary to assemble and load the system processors and routines again. For this purpose, the two control files MKSYS and SYSGEN are provided, plus the control file DELSYS, which removes unneeded source and relocatable files from working cartridges.

When executed upon a master cartridge, the control file MKSYS assembles all the source files of the system (i.e., all the files documented in this manual), then employs the relocatable loader to produce all system absolute files, including new absolute files for MONITR (NEWMON). A new version of the relocatable file SYSTEM (NEWSYS) is also created at this time. This file includes all the pointers into the new MONITR and is required to give the user access to the MONITR entry points. Note that after MKSYS is executed upon a master disc cartridge, numerous relocatable and absolute files will be added. If MKSYS is run with NOPRNT selected, no listing will be printed allowing the program to be executed more rapidly.

Next the control file SYSGEN is executed; this processor deletes all the system files on the working cartridge, then copies, selected relocatable files (e.g., the libraries and SYSTEM), the absolute files (and, optionally the system source files) to the second cartridge. Note that SYSGEN assumes that the second cartridge has been formatted and has a working disc directory and availability table on Sectors 06-017 of cylinder 0. If a new cartridge is being loaded, it will be necessary to first run the disc diagnostic on it to establish the header formats, then to execute COPY01 from a working cartridge to establish the availability table and the directory. Note that once MKSYS has been performed, SYSGEN may be used to dub any number of working cartridges without destroying any working files that may be on the working cartridges. Once MKSYS has been performed upon a master cartridge, the cartridge will have more absolute files added, which are copied to the working cartridge each time; these appear on the directory dump of the master cartridge with a prefixed asterisk. When a new master is created, good practice dictates first making a new cartridge using COPY01, then applying corrections, then applying MKSYS and SYSGEN to make new working cartridges. The old master should then be saved for backup.

If a cartridge contains numerous system source and relocatable files, it may be desirable to remove them to free disc space. The DELSYS processor deletes all the system-source and all system-relocatable files except the libraries, then deletes itself, then calls JOB to return all the deleted files to the availability table. DELSYS should be used only on working cartridges, not on system or backup cartridges.†

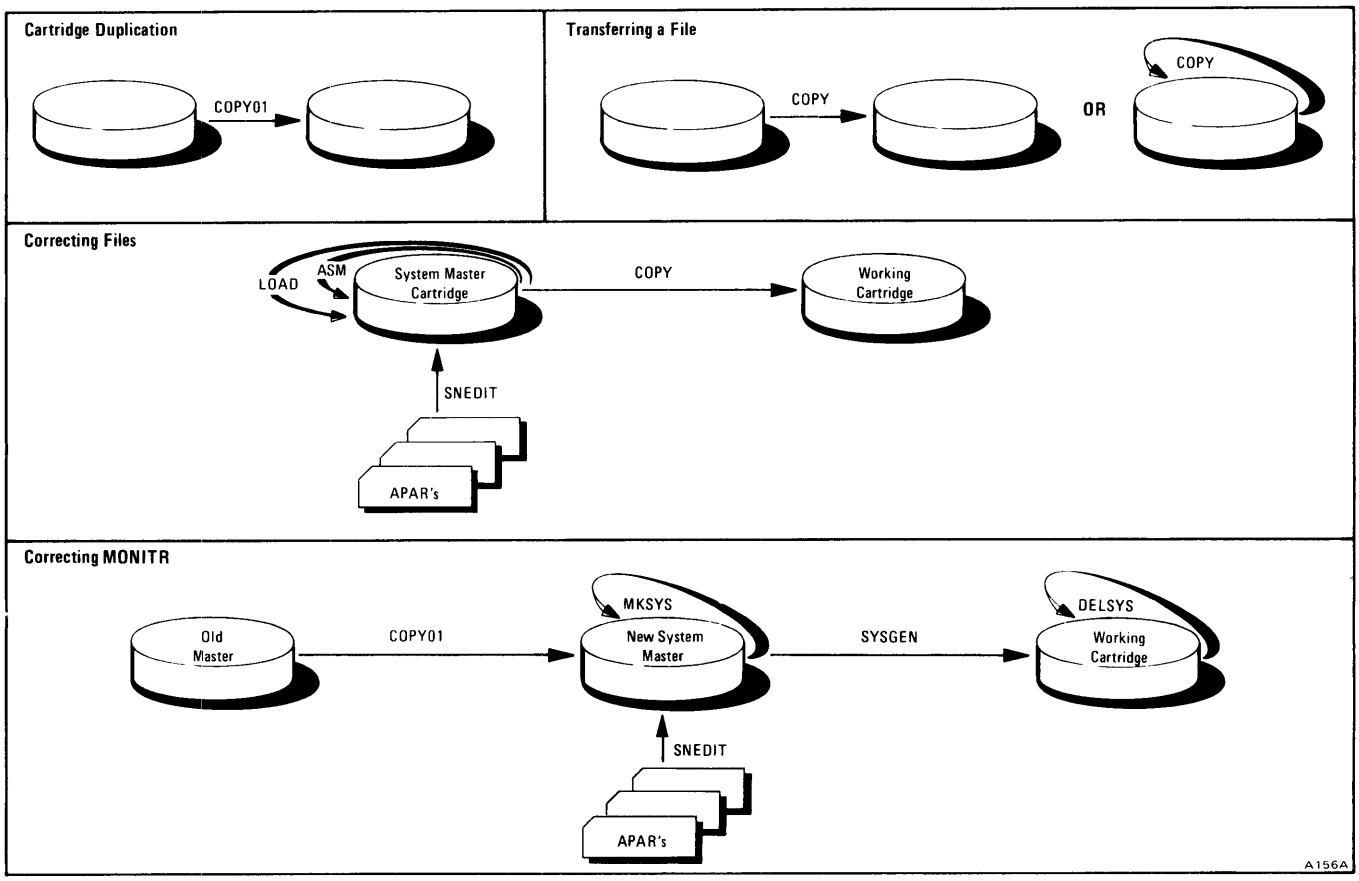


Figure 4. Correction & Duplication of Cartridges

† The processor LIST is provided for obtaining listings of control files such as MKSYS, SYSGEN, and DELSYS without the need for executing them. The user may alter the control files using SNEDIT.

MINOR PROCESSORS

Minor processors may be included in job streams to interrupt the course of processing, change the selection of SYSIN or SYSOUT devices, include comment statements, etc. These processors are in the source file P17-P and the relocatable file R17-P. In absolute form they are called by the names given below. They are as follows:

// CARDS	Entered from the keyboard, specifies the card reader as the system input device.
// KYBRD	Entered from the card reader, specifies the first keyboard as the system input device.
// P8131	Entered in the job stream, selects the 8131 Printer as the SYSOUT device.
// P8143	Entered in the job stream, selects the 8143 Printer as the SYSOUT device.
// P8145	Entered in the job stream, selects the 8145 Printer as the SYSOUT device.
// P8141	Entered in the job stream, selects the 8141 or 8142 Printer as the SYSOUT device.
// P8146	Entered in the job stream, selects the 8146, 8147, 8151, or 8152 Printer as the SYSOUT device.
// NOPRNT	Entered in the job stream, disables SYSOUT.
// PAUSE	Halts execution of the job stream temporarily. To resume execution, move the AUTO/MANUAL switch to MANUAL, then to AUTO.
// EJECT	Causes the SYSOUT printer to eject a page.
/.	Used to insert comments into a parameter stream.
//	A // followed by seven spaces and then a comment is used to insert comments into a control statement stream. Note that if this method is used after an ASM or LOADOV stream, the card will be treated as a //b, and the message lost.

If the incorrect printer is selected (e.g., through use of an incorrect // P statement), it may be impossible to communicate with DOS the next time the system is bootstrapped. If this happens, "// SYSTEM IV/70 DISC OPERATING SYSTEM 88-0017-XX" (where XX is the latest release code) will appear on the first screen and it will be impossible to get a cursor. Go to MANUAL mode and look at location 036 (LPOUT). (If necessary, press SYSTEM RESET, then STEP to get into MANUAL mode.) Load the correct value into LPOUT — see Appendix C for acceptable values of LPOUT — then execute a branch to location 1 (72000001) to start the system. Note that it will also be impossible to communicate with the system if the printer is turned off (unless NOPRNT is selected).

In SYSLIB, a default value of 0 (P8145) for LPOUT is provided. In installations with another type of printer, it will be necessary to change this value. Execute the following.

```
// SNEDIT
/INPUT = P17-9, OUTPUT = P17-9.
#092900, 092900 (see a current P17-9 listing for LPOUT location)
LPOUT  DCN      (appropriate LPOUT value)
//
// ASM
/INPUT = P17-9, OUTPUT = SYSLIB.
//
```

DOS MONITR ROUTINE $DISC

Purpose: To provide seek, read, and write operations for the 8231 Cartridge Disc Unit. Many drives and controllers may be handled by $DISC. However, it is distributed to handle 4 drives on one controller on channel 2.

Requirements: One or more 8231 Disc Units and 0250 words of memory.

Usage: Calling sequence:

```
BAL     $DISC
 PZE    REQTAB
 BXX    reject
```

Where REQTAB is the location of a four word table which specifies the I/O operation requested. The reject return is taken when the request is invalid or an unrecoverable error condition arises. The first word at REQTAB is altered by $DISC and indicates the status of the request upon return.

See attached IO Request Table.

Special Features: $DISC maintains an internal device status table (DST) for each 8231 drive on the system. Several 8231 drives and even multiple 8230 controllers can be handled by $DISC. Multiple tracks can be read by a single call to $DISC.

Method: The DOS Monitor version runs off status only. If a second call is made to $DISC after a previous seek was initiated for the same drive, $DISC will wait until the first seek is complete before processing the new request.

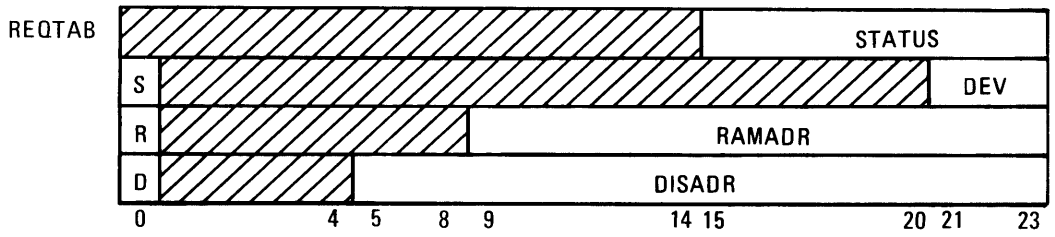
Restrictions: $DISC is not reentrant, so that it should not be called from an interrupt routine unless a software lock is provided or the appropriate interrupt level is disarmed when used at a lower level or in the background. Redirect logic is not provided in the current version.

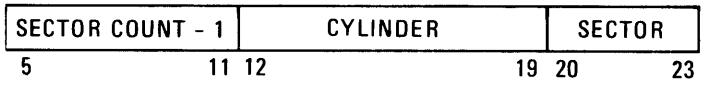
Errors: The reject return is taken by $DISC if the request as specified in REQTAB is invalid (e.g., device not ready, DEV too large, illegal disc address, reentrant call) or a permanent error develops (e.g., solid cyclic data error, or other status error which is illegal).

Register Usage: All registers are preserved.

Disc Versions: In source form, $DISC is furnished as part of MONITR (P17-2B), the bootstrap loader program (P17-1), and in various processors. In relocatable form it is called from SYSLIB; user programs will access it there. It is not normally accessed separately in absolute form.

IO Request Table:



Word	Bits	Symbol	Meaning
REQTAB	23-0	STATUS†	Bit 23 = Not Ready Bit 22 = Busy Bit 21 = Cyclic Redundancy Error Bit 20 = Too Late Bit 19 = Header Error Bit 18 = Head Out of Range Bit 17 = Seek Incomplete Bit 16 = Illegal Request Bit 15 = Incorrect Length Bits 0-14 = Not Used
REQTAB+1	23-21	DEV‡	Index to drive 0 ⩽ device ⩽ 7
	20-1	—	Not used
	0	S‡	1 Do seek only 0 Do data transfer (with seek if necessary)
REQTAB+2	23-9	RAMADR‡	Location in memory of data buffer
	8-1	—	Not used
	0	R‡	1 Normal return only when status = 0 0 Normal return after operation initiated or stacked
REQTAB+3	23-5	DISADR‡	Disc address request word 
	4-1	—	Not used
	0	D‡	1 write data on disc 0 read data from disc

†Furnished by $DISC

‡Furnished by user program

The reject return is taken if any of the input values are out of range. Note that cylinders 200 through 202 are illegal.

A098E

DOS LIBRARY ROUTINE $DUMP

Purpose: To dump memory onto SYSOUT.

Requirements: The 8231 Disc Unit, 0264 words of memory, and a SYSOUT printer.

Usage: Calling sequence:

```
BRM     $DUMP
 PZE    LOWER          (The lowest memory location dumped)
 PZE    UPPER          (The highest memory location dumped)
(NORMAL RETURN)
```

On return, all interrupts are disarmed.

To dump a running program put the computer into MANUAL mode, enter a BRM to the address of $DUMP† into TIR, press step *once*, then return to AUTO mode. The LOWER and UPPER constants will be the last such used, or 0 and $DUMP if they have not been altered. Note that $DUMP must be resident (i.e., when $DUMP is left as a virtual in the source program, it will be resolved by SYSLIB).

Method: $DUMP uses SYSOUT to print memory. The location LUCONV (see Appendix C) is checked; lower case symbols will be printed if the appropriate printer is attached and LUCONV = 0. The location LPOUT is used to determine which printer is attached. Note that if MONITR is not resident and $DUMP is used relocatable, the user may furnish values for these variables or accept the SYSLIB default values of LUCONV ≠ 0 and LPOUT = 0.

Special Features: A line is printed containing the six variable working registers (RP, RA, RB, X1, X2, X3 in order). Then memory is printed, eight words per line, with the starting address. Lines of all zeros are suppressed. Each line is printed both in octal and in ASCII.

Subroutines Used: The DOS routine SYSOUT.

Register Usage: $DUMP saves all registers and condition codes.

Disc Versions: $DUMP is furnished in source for as part of P17-9 and the relocatable file $DUMP called from SYSLIB.

† The location $DUMP can be obtained from the load map for the program, if the recommended procedure is followed — i.e., if $DUMP is called from SYSLIB.

DOS LIBRARY ROUTINE $ICARD

Purpose: To pick and read a card under the System IV/70 interrupt system.

Requirements: The 8001 Card Reader and 0260 words of memory. The R17-Q routines $IOPEN and $ICLOS.

Usage: Each time a card is required, execute:

```
BAL     $ICARD
 PZE    BUFFER ADDRESS
```

Where BUFFER ADDRESS is the location of the 27-word user buffer. $ICARD will pick and read a card, converting Hollerith to ASCII and stripping off parity bits; the card image will be read into the program's internal buffer, then transferrred to the user buffer with an end of record (LF, ASCII 012) placed in character position 81.

Before $ICARD or any other interrupt routine in this group is executed ($ICARD, $IPRNT, $JPRNT, $IDISC, $ITAPE, $JTAPE), execute

```
BAL     $IOPEN
```

to set up the controls for time-critical processing. After interrupt processing is completed (all I/O operations complete), execute

```
BAL     $ICLOS
```

to terminate operations and clear all interrupts. Once $IOPEN has been executed, it should not be executed again until after $ICLOS is executed — i.e., only one execution of $IOPEN should take place in any normal sequence of operations.

Method: $ICARD and the other interrupt routines use time-critical queuing to keep their operations out of one another's way. $ICARD reads the card image into its internal buffer, then checks for an end of file (//b) in columns 1-3. The program then puts the card image into the user buffer. If the end of file was not found, the program requests another pick before returning to the user; otherwise it simply returns to the user. $ICARD uses a self-generated internal user table similar to the ones used by $IDISC and $ITAPE.

Restrictions: In general, this and all interrupt I/O routines are incompatible with MONITR — crucial low memory locations are overlayed.

Register Usage: All registers are saved.

Disc Versions: $ICARD is part of COBLIB (the COBOL object time program library) and R17-Q (the interrupt I/O driver library). Both these libraries link to SYSLIB. $ICARD is available in source form as part of P17-Qx (where x is the current revision).

DOS LIBRARY ROUTINE $IDISC

Purpose: To provide interrupt-driven seek, read, and write operations for the 8231 Cartridge Disc Unit.

Requirements: One or more 8231 Disc Units and 0636 words of memory. The R17-Q routines $IOPEN and $ICLOS.

Usage: Calling sequence:

```
BAL     $IDISC
 PZE    REQTAB
```

Where REQTAB is the location of a five word table that specifies the I/O operation requested. The first word at REQTAB is altered by $IDISC and indicates the status of the request upon return. See the attached IO Request Table.

Before $IDISC or any other interrupt routine in this group is executed ($ICARD, $IPRNT, $JPRNT, $IDISC, $ITAPE, $JTAPE), execute

```
BAL     $IOPEN
```

to set up the controls for time-critical processing. After interrupt processing is completed (all I/O operations complete), execute

```
BAL     $ICLOS
```

to terminate operations and clear all interrupts. Once $IOPEN has been executed, it should not be executed again until after $ICLOS is executed — i.e., only one execution of $IOPEN should take place in any normal sequence of operations.

A queue of request tables is maintained for each drive on the system (each table with a different name), so that the user may make requests to $IDISC without regard for the problem of reentrancy.

Method: $IDISC and the other interrupt routines use time-critical queuing to keep their operations out of one another's way. $IDISC does not perform double buffering.

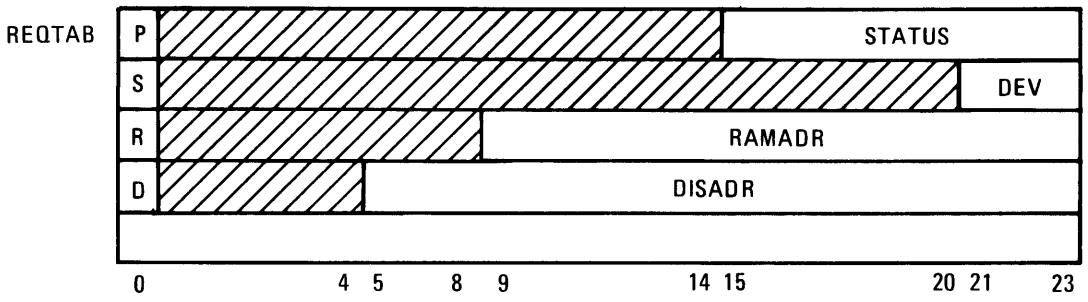
Restrictions: This and all interrupt I/O routines are incompatible with MONITR because crucial low memory locations are overlayed.

Errors: $IDISC posts errors it cannot correct in the status word at REQTAB + 0; the user must handle correction of these conditions. Redirect logic is not provided in the current version.

Register Usage: All registers are saved.

Disc Versions: $IDISC is part of COBLIB (the COBOL object time library) and R17-Q (the interrupt I/O driver library). Both these libraries link to SYSLIB. The program is also available in source form as part of P17-Qx (where x is the current revision).

IO Request Table:



Word	Bits	Symbol	Meaning
REQTAB	23-0	STATUS†	Bit 23 = Not Ready Bit 22 = Busy Bit 21 = Cyclic Redundancy Error Bit 20 = Too Late Bit 19 = Header Error Bit 18 = Head Out of Range Bit 17 = Seek Incomplete Bit 16 = Illegal Request Bit 15 = Incorrect Length Bits 1-14 = Not Used
		P†	Bit 0 = Status posted, operation complete
REQTAB+1	23-21	DEV‡	Index to drive 0 ⩽ device ⩽ 7
	20-1	—	Not used
	0	S‡	1 Do seek only 0 Do data transfer (with seek if necessary)
REQTAB+2	23-9	RAMADR‡	Location in memory of data buffer
	8-1	—	Not used
	0	R‡	1 Normal return only when status = 0 0 Normal return after operation initiated or stacked
REQTAB+3	23-5	DISADR‡	Disc address request word SECTOR COUNT – 1 | CYLINDER | SECTOR 5 11 12 19 20 23
	4-1	—	Not used
	0	D‡	1 write data on disc 0 read data from disc
REQTAB+4	23-0	—	BSS 1 — used by queuing software

†Furnished by $IDISC

‡Furnished by user program

The reject return is taken (status bits 0, 16 = 1) if any of the input values are out of range, Note that cylinders 200 through 202 are illegal.

A363A

DOS LIBRARY ROUTINES $IOPEN & $ICLOS

Purpose: To perform housekeeping tasks required for starting and shutting down interrupt I/O operations.

Requirements: The 8231 Disc Unit and 042 words of memory.

Usage: Before any of the interrupt I/O routines $ICARD, $IPRNT, $JPRNT, $IDISC, $ITAPE, or $JTAPE can be executed, execute:

```
BAL     $IOPEN
```

This routine should be performed only once before $ICLOS is executed. It initializes certain cells that surround the interrupt IOID locations.

After all interrupt operations are completed, use:

```
BAL     $ICLOS
```

This routine loops, checking to make sure that all the interrupt processors have finished their work, then clears out all interrupt levels and returns to the calling program.

Register Usage: Only RA, RB, X1 are saved. X2 and X3 not used.

Disc Versions: $IOPEN and $ICLOS are part of COBLIB (the COBOL object time library) and R17-Q (the interrupt I/O driver library). Both these libraries link to SYSLIB. The program is also available in source form as part of P17-Qx (where x is the current revision).

DOS LIBRARY ROUTINES $IPRNT, $JPRNT

Purpose: To print a line ($IPRNT) or eject a page ($JPRNT) on a Four-Phase supported system printer using the System IV/70 interrupt system.

Requirements: A system printer and 0502 words of memory. The R17-Q routines $IOPEN and $ICLOS.

Usage: Each time a line is to be printed, execute:

```
BAL     $IPRNT
 PZE    BUFFER ADDRESS
```

Where BUFFER ADDRESS is the location of the line buffer. You may furnish a line feed (ASCII 012) or form feed (ASCII 015) as appropriate to terminate the line, or you may allow the system end of record mark (012) that is furnished by system input routines to terminate the line. If neither of these procedures is followed, $IPRNT will place a line feed in the last character position of the line (one past the last printable position). See "SYSOUT" for line lengths. When $IPRNT is called, it moves the print line to its own internal buffer, and enqueues a print request, then returns to the calling program if the request cannot be serviced immediately.

To eject the current page, use the calling sequence:

```
BAL     $JPRNT
```

The program will send a form feed (ASCII 014) to the printer.

Before $IPRNT or $JPRNT or any other interrupt routine in this group is executed ($ICARD, $IPRNT, $JPRNT, $IDISC, $ITAPE, $JTAPE), execute

```
BAL     $IOPEN
```

to set up the controls for time-critical processing. After interrupt processing is completed (all I/O operations complete), execute

```
BAL     $ICLOS
```

to terminate operations and clear all interrupts. Once $IOPEN has been executed, it should not be executed again until after $ICLOS is executed — i.e., only one execution of $IOPEN should take place in any normal sequence of operations.

Method: Symbolic location LPOUT is a virtual in this routine; the user may resolve it in accordance with the table shown in Appendix C or allow the default value of 0 (8145 printer) from SYSLIB to be furnished. The routine interprets this value and selects the printer driver accordingly.

Special Features: This program functions exactly like an interrupt SYSOUT and may be used to replace SYSOUT anywhere that functions linked using SYSTEM are not in use. The method is to edit the source to change the constant that selects ENTRY statements for $IPRNT, $JPRNT or SYSOUT, SYSJCT. Execute the following to convert $IPRNT to SYSOUT:

```
// SNEDIT
/I=P17-Qx, O=P17-Qx.
#139100,139100
SYSNAM  EQU     1
//
// ASM
/I=P17-Qx, O=R17-Q.
```

Restrictions: This routine overlays crucial low memory locations in MONITR; in general, MONITR and interrupt routines are incompatible.

Register Usage: All registers saved.

Disc Versions: $IPRNT and $JPRNT are part of COBLIB (the COBOL object time library) and R17-Q (the interrupt I/O driver library). Both these libraries link to SYSLIB. The program is also available in source form as part of P17-Qx (where x is the current revision).

DOS LIBRARY ROUTINE $ITAPE

Purpose: To provide interrupt driven I/O operations for the 8511/8512 Magnetic Tape Unit.

Requirements: One or more 8511/8512 Magnetic Tape Units and 0551 words of memory. The R17-Q routines $IOPEN and $ICLOS.

Usage: Calling sequence:

```
BAL     $ITAPE
 PZE    REQTB
```

Where REQTB is the location of a five word table that specifies the I/O operation requested. The first word at REQTB is altered by $ITAPE and indicates the status of the request upon return. See the attached IO Request Table.

Before $ITAPE or any other interrupt routine in this group is executed ($ICARD, $IPRNT, $JPRNT, $IDISC, $ITAPE, $JTAPE), execute

```
BAL     $IOPEN
```

to set up the controls for time-critical processing. After interrupt processing is completed (all I/O operations complete), execute

```
BAL     $ICLOS
```

to terminate operations and clear all interrupts. Once $IOPEN has been executed, it should not be executed again until after $ICLOS is executed — i.e., only one execution of $IOPEN should take place in any normal sequence of operations.

A queue of request tables is maintained for each drive on the system (each table with a different name), so that the user may make requests to $ITAPE without regard for the problem of reentrancy.

Method: $ITAPE and the other interrupt routines use time-critical queuing to keep their operations out of one another's way. $ITAPE does not perform double buffering. For further details, see "Peripheral Unit Programming Manual", document SIV/70—40—1.

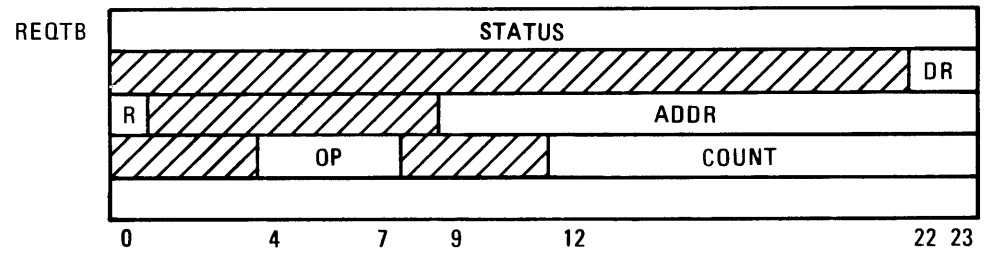
Restrictions: This and all interrupt I/O routines are incompatible with MONITR because crucial low memory locations are overlayed.

Errors: $ITAPE posts errors it cannot correct in the status word at REQTB + 0; the user must handle correction of these conditions.

Register Usage: Only RA, X1, X2, X3 are saved.

Disc Versions: $ITAPE is part of COBLIB (the COBOL object time library) and R17-Q (the interrupt I/O driver library). Both these libraries link to SYSLIB. The program is also available in source form as part of P17-Qx (where x is the current version).

IO Request Table:



Word	Bits	Symbol	Meaning
REQTB	23-0	STATUS†	Bit 23, Drive not ready Bit 22, Busy (error condition) Bit 21, Parity error Bit 20, Write Protect Bit 19, Beginning of Tape (BOT) Bit 18, End of Tape (EOT) Bit 17, Data Lost (too late) Bit 16, File mark Bits 15-13, Not used Bits 12 and 11, Byte boundary on a read = 00, Last data word full, operation complete = 01, Last word has 1 byte left justified = 10, Last word has 2 bytes left justified = 11, Last word full, operation complete Bit 10, Short count Bit 9, Device address out of range (see REQTB+2)‡ Bit 8, Incorrect operation type (see REQTB+3)‡ Bit 7, Number of words transmitted is wrong‡ Bit 6, Lost interrupt Bits 5-1, Not used Bit 0 = 1, Status posted, operation complete‡ = 0, Operation incomplete‡
REQTB+1	23-22 21-0	DR‡	Drive index: 00, 01, 10, or 11 Not used
REQTB+2	23-9 8-1 0	ADDR‡ R‡	Memory address of tape data buffer Not used Bit 0 = 1, Normal return only when operation complete = 0, Return when operation queued or initiated
REQTB+3	23-12 11-8 7-4 3-0	COUNT‡ OP‡	Read or Write count in bytes Not used Operation Type 000 = Read; interrupts 001 = Write; interrupts 002 = Write file mark 003 = Erase 004 = Skip record 005 = Backspace record 006 = Skip file 007 = Backspace file 010 = Rewind 011 = Reset 012 = Read backward; interrupts 013 = Read backward; lockup 014 = Read; lockup 015 = Write; lockup Not used
REQTB+4	23-0	—	BSS 1 — Used by time critical processing in $ITAPE

†Furnished by $ITAPE ‡Furnished by user program ‡Generated by $ITAPE; other bits come from controller and are posted by $ITAPE.

A207C

DOS LIBRARY ROUTINE $JTAPE

Purpose: To provide interrupt driven I/O operations for the 8513 Magnetic Tape Unit.

Requirements: One or more 8513 Magnetic Tape Units and 0562 words of memory. The R17-Q routines $IOPEN and $CLOS.

Usage: Calling sequence:

```
BAL     $JTAPE
 PZE    REQTB
```

Where REQTB is the location of a five word table that specifies the I/O operation requested. The first word at REQTB is altered by $JTAPE and indicates the status of the request upon return. See the attached IO Request Table.

Before $JTAPE or any other interrupt routine in this group is executed ($ICARD, $IPRNT, $JPRNT, $IDISC, $ITAPE, $JTAPE), execute

```
BAL     $IOPEN
```

to set up the controls for time-critical processing. After interrupt processing is completed (all I/O operations complete), execute

```
BAL     $ICLOS
```

to terminate operations and clear all interrupts. Once $IOPEN has been executed, it should not be executed again until after $ICLOS is executed — i.e., only one execution of $IOPEN should take place in any normal sequence of operations.

A queue of request tables is maintained for each drive on the system (each table with a different name), so that the user may make requests to $JTAPE without regard for the problem of reentrancy.

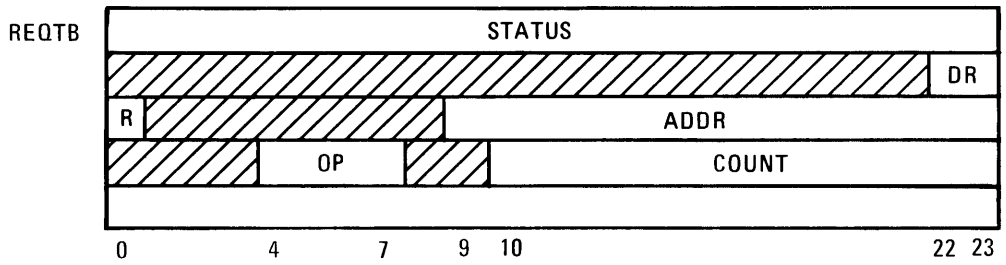
Method: $JTAPE and the other interrupt routines use time-critical queuing to keep their operations out of one another's way. $JTAPE does not perform double buffering. For further details, see "Peripheral Unit Programming Manual", document SIV/70—40—1.

Restrictions: This and all interrupt I/O routines are incompatible with MONITR because crucial low memory locations are overlayed.

Errors: $JTAPE posts errors it cannot correct in the status word at REQTB + 0; the user must handle correction of these conditions.

Disc Versions: $JTAPE is part of COBLIB (the COBOL object time library) and R17-Q (the interrupt I/O driver library). Both these libraries link to SYSLIB. The program is also available in source form as part of P17-Qx (where x is the current version).

IO Request Table:



Word	Bits	Symbol	Meaning
REQTB	23-0	STATUS†	Bit 23, Drive not ready Bit 22, Busy Bit 21, Hard Error Bit 20, Write Protect Bit 15, Transfer required Bit 14, Not used Bit 13, Rewinding Bits 12 and 11, Byte boundary on a read = 00, Last data word full, operation complete = 01, Last word has 1 byte left justified = 10, Last word has 2 bytes left justified = 11, Last word full, operation complete Bit 10, Short count Bit 9, Reject Bit 8, Corrected parity Bit 7, 1600 ID Bit 6, Operation complete Bit 5, Device address out of range (see REQTB+2)‡ Bit 4, Incorrect operation type (see REQTB+3)‡ Bit 3, Number of words transmitted is wrong‡ Bit 2, Lost interrupt Bit 1, Not used Bit 0 = 1, Status posted, operation complete‡ = 0, Operation incomplete‡ Bit 19, Beginning of Tape (BOT) Bit 18, End of Tape (EOT) Bit 17, Too Late Bit 16, File mark
REQTB+1	23-22 21-0	DR‡	Drive index: 00, 01, 10, or 11 Not used
REQTB+2	23-9 8-1 0	ADDR‡ R‡	Memory address of tape data buffer Not used Bit 0 = 1, Normal return only when operation complete = 0, Return when operation queued or initiated
REQTB+3	23-10 9-8 7-4 3-0	COUNT‡ OP‡ —	Read or Write count in bytes Not used Operation Type 000 = Read 001 = Write 002 = File mark 003 = Erase 004 = Skip record 005 = Backspace record 006 = Skip file 007 = Backspace file 010 = Rewind 011 = Reset Not used
REQTB+4	23-0	—	BSS 1 — Used by time critical processing in $JTAPE

†Furnished by $JTAPE ‡Furnished by user program ‡Generated by $JTAPE; other bits come from controller and are posted by $JTAPE.

A364A

DOS MONITR ROUTINE ALLOC

Purpose: Allocate a specified number of contiguous sectors from the disc.

Requirements: The 8231 Disc Unit, the DOS MONITR routines $DISC and ALLOCF, and the system disc buffer BUF.

Usage: Calling sequence:

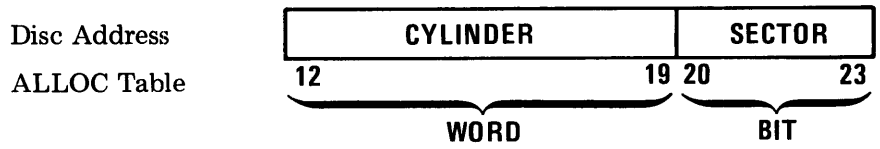
```
BAL     ALLOC
 PZE    DRIVE INDEX
 PZE    N
 BSS    1 (returned sector address)
 BRA    not enough room on disc (reject)
(NORMAL RETURN)
```

Where N is the desired number of sectors. The sector address returned has (N-1) in bits 0-11, so that it has the correct format for a read to the disc (if N≤16), or for the fourth word in a disc directory entry. If N available contiguous sectors cannot be found, the reject return is taken.

A recommended method for using ALLOC is to leave MONITR resident. This is accomplished by origining the user program at or above 03400 and by placing /INPUT = SYSTEM first in the parameter string at load time.

Method: ALLOC uses sector 6 on each cartridge, which contains a table of available sectors. In each word of the table, 16 bits are used to represent the availability of each of the 16 sectors in a cylinder. Bit 0 of word 0 of the table represents sector 0, cylinder 0. If a bit is on, the represented sector is available. If a bit is off, the represented sector is nonexistent, in cylinder 0, in use, or in a file whose name is blank. For the fixed disc file allocations, see "MONITR".

A 12 bit disc address is mapped into the allocation table as follows:



Subroutines Used: $DISC, ALLOCF.

Register Usage: Only X1-X3 are saved.

Disc Versions: ALLOC is furnished in source form as part of file P17-7B, in relocatable form with R17-7B (the file SYSTEM must be resident), and in absolute form as part of MONITR.

DOS MONITR ROUTINE ALLOC1

Purpose: Efficiently allocate single sectors of the disc for chained format files.

Requirements: The 8231 Disc Unit, the DOS MONITR routines $DISC and ALLOCF, and the system disc buffer BUF.

Usage: Each time a sector is desired, use:

```
BAL     ALLOC1
 PZE    DRIVE INDEX
 PZE    SECTOR (used on input and output - see "Method" below)
 BRA    not enough room on disc
```

ALLOC1 must be closed after all sectors are allocated to copy the updated allocation table back to the disc. Use:

```
BAL     ALLOCF
```

A recommended method for using ALLOC1 is to leave MONITR resident. This is accomplished by origining the user program at or above 03400 and by placing /INPUT = SYSTEM first in the parameter string at load time.

Timing: ALLOC1 performs a disc access when (1) the first sector is allocated, (2) a sector on a different drive is allocated, (3) ALLOC is used, (4) any directory operation is done, or (5) SYSIN reads a record from a control procedure. ALLOCF requires at most one access.

Method: ALLOC1 uses the table of available sectors, as described for ALLOC. ALLOC1 takes the first available sector in the cylinder of SECTOR or the nearest higher cylinder (wrapping around after cylinder 199) and returns the disc address in SECTOR. This technique is used to minimize access arm movements for chained files.

Subroutines Used: The DOS routine $DISC.

Register Usage: Only X1-X3 are saved.

Disc Versions: ALLOC1 is furnished in source form as part of file P17-7B, in relocatable form with R17-7B (the file SYSTEM must be resident), and in absolute form as part of MONITR.

DOS MONITR PROCESSOR ASM

Purpose: ASM assembles CODE language statements as described in the "System IV/70 Computer Reference Manual", document SIV/70—11—1.

Requirements: The 8231 Disc Unit, 8K words of memory, and the DOS MONITR.

Usage: Enter the control statements:

```
// ASM
/INPUT = name_i @ drive_i.
/RELOC = name_r 2 drive_r.
//
```

Into SYSIN. The input source file "name$_i$" on drive "drive$_i$" will be assembled, producing a listing on SYSOUT and a relocatable file name "name$_r$" on drive "drive$_r$". Note that the system will recognize either /RELOC or /OUTPUT for the output parameter; however it is recommended that only one parameter name or the other be used in a given job stream.

Messages: A symbol table overflow is the only condition that will cause an error halt. The following messages may be printed by the assembler:

THE RELOCATABLE OUTPUT CANNOT BE CATALOGUED ON THE DISC	(this is the error return from DRUPD)
THE SPECIFIED OUTPUT IS TEMPORARILY NOT SUPPORTED	(no /O or /R parameter detected: no default option is offered)
THE INPUT FILE IS NOT IN THE DIRECTORY	
THE FOLLOWING DATA CARD WAS FOUND IN THE PARAMETER STEAM	(followed by a line feed and the card in error)
THE CARD READER IS TEMPORARILY NOT SUPPORTED	(no /I parameter detected: the default option is not currently available)
POST-PROCESSOR NOT ON DISC	
ASSEMBLER NOT ON DISC	
1 UNRECOVERABLE DISC ERROR	(Disc error on rewinding input for second pass)
2 UNRECOVERABLE DISC ERROR	(Linkage error on first sector of input file)
3 UNRECOVERABLE DISC ERROR	(Disc error encountered in DPUT)
4 UNRECOVERABLE DISC ERROR	(Disc error in attempting to read postprocessor)

5 UNRECOVERABLE DISC ERROR	(Linkage error encountered in DGET)
6 UNRECOVERABLE DISC ERROR	(Disc error encountered in DGET)
1 NOT ENOUGH ROOM ON DISC	(No-more-sectors exit from ALLOC1 in DPUT)
2 NOT ENOUGH ROOM ON DISC	(No-more-sectors exit from ALLOC1 in OPEN)

Subroutines Used: See "MONITR".

Disc Versions: The assembler exists in three parts. ASM, the preprocessor, is furnished as source file P17-CB and absolute file ASM. DASM, the main processor, is called by ASM and exists as the source files P3-1G, P3-2, P3-3B, P3-4B, P3-5, P3-6, P3-7, and absolute file DASM. ASMX, the postprocessor, is called by DASM and is furnished as source file P17-E and absolute file ASMX.

DOS MONITR PROCESSOR BOJ

Purpose: To return sectors to the availability table that are missed by JOB.

Requirements: The 8231 Disc Unit, 6K words of memory, and the DOS MONITR.

Usage: Enter the following control statement into SYSIN:

```
// BOJ
```

Method: BOJ compares the disc directory on each disc that is ready with the allocation table on that disc to determine which sectors are actually in use in named files; it then returns all unused sectors to the allocation table if possible, checks for sectors used but not allocated, etc. This is in contrast to JOB, which merely checks for deleted (blank name) and TEMP files and returns their sectors to the availability table.

Normally BOJ is not required — JOB performs adequately unless certain errors have occurred. However, if a disc is nearly full, a processor that attempts to build a file can cause sectors to be lost if it overflows the disc and fails to return the allocated sectors to the availability table and also fails to name the file because of taking its overflow exit. BOJ will recover sectors lost in this way and in other ways.

Restrictions: BOJ assumes that all disc drives that are ready have DOS cartridges on them. BOJ will destroy sectors $6\text{-}15_{10}$ on non-DOS formatted cartridges.

Messages: For each disc processed, BOJ will print the following messages:

ON DRIVE M	(printed for each drive that is ready)
A SECTORS ARE IN USE	(always printed)
B SECTORS ABOVE ARE COUNTED TWICE	(not printed if zero: includes ALIASes and errors)
C SECTORS ARE AVAILABLE	(always printed)
D SECTORS WERE RECLAIMED	(always printed)
E SECTORS WERE RECLAIMABLE BUT FOR DISC ERRORS	(not printed if zero)
F SECTORS WERE IN USE BUT NOT ALLOCATED	(not printed if zero)
SECTOR XXXXX, ON DRIVE Y, HAS AN UNRECOVERABLE ERROR	(printed for each bad sector)
ERROR IN SECTOR INHIBITS ALLOCATOR REWRITE	(Usually indicates a sector linkage error in a chained file. To recover, the user determines which file contains the bad sector, deletes the file, and executes JOB followed by BOJ again.)

Subroutines Used: See "MONITR".

Disc Versions: BOJ is furnished as source file P17LB and as absolute program BOJ.

DOS MONITR PROCESSOR CDDC (CARD TO DISC)

Purpose: Reconstruct disc files from backup card decks and create contiguous disc files.

Requirements: The 8231 Disc Unit, 4K words of memory, the DOS MONITR, and a SYSOUT printer.

Usage: Enter the control statement:

```
// CDDC
```

Followed by sets of parameter statements and data decks.

Each group of parameter statements is as follows:

```
/OUTPUT = name @ drive.                  (required)
/TYPE = OCTAL or SOURCE or RELOC.        (required)
/SECTORS = number. (default = 1)         (optional)
/PROTECT.                                (optional)
/END = code. (default = // )             (optional)
/MONITR.                                 (optional)
/CLEAR.                                  (optional)
```

The /OUTPUT parameter is always required and specifies the name of the file to be created and the drive on which the file will be placed. The /TYPE parameter is always required, and designates input format and output file format. The first letter of the value may be used (i.e., O, S, or R). Octal files are contiguous; source and relocatable files are in chained format. If input is a source deck, the end card can be specified via the /END option. The three nonblank characters following the "=" will be used in checking card columns 1-3 to terminate input. Default is "//b". If input is octal, the file size in terms of the number of sectors to be allocated must be specified. Regardless of the size of the octal deck (too small or too large), the file will be allocated into the number of contiguous sectors specified in the "/SECTORS = number" parameter. Unused sectors are set to zero.

The /PROTECT parameter flags the output file as being protected. Note: if a protected file already exists on the disc under the name given in the "/OUTPUT = name" parameter, no new output file will be created. If an old file which is protected is to be replaced under the same name, the protect flag must first be deleted using DIRMOD.

If a disc is being rebuilt from scratch, cylinder 0 is rebuilt by using /MONITR and /CLEAR. /CLEAR can only be used if /MONITR is specified. /MONITR can only be used if /TYPE=OCTAL is specified. /MONITR causes the octal deck to be loaded onto the disc in sectors 0-5 and entered into the directory with a load address of 0001 and sector count of 020, starting at sector 0. /CLEAR causes the directory to be cleared of all entries, and the allocator tables reset to indicate the full disc cartridge is available. Thus /CLEAR should only be specified on the first set of control cards for a given disc cartridge. These options should be used with caution as they can blow a cartridge.

The directory load address for octal decks is taken to be the first origin in the deck. Note that successive octal correction cards will be correctly applied. CDDC is also used to allocate contiguous disc space (e.g., initializing files for use by COBOL programs). Such files are either data files or executable ("bootstrap") files. A data file has a load point and transfer point (defined by the origin and transfer fields on an octal card) of 0; a bootstrap file has a load point and transfer point of 1. These are defined using a single octal card with only a sequence number, origin field, and transfer field (see "Octal Data Format" below). This program operates as the functional opposite of the DCCD program.

Example: To generate a bootstrap file for a checkpoint dump of a program 30 sectors in length.

```
// CDDC
/OUTPUT = BOOTFL @ 0, TYPE = 0, SECTORS = 30.
BTF00000&0000001-0000001 (zeros for a data file)
//
```

Octal Data Format: The octal cards are organized as follows (see the illustration "Format of Octal Cards"): The first field of each card contains program identification (any characters in columns 1-3) and the card sequence number (octal digits in columns 4-8). The sequence must start with zero and each number must be one more than the preceding number; after the last number (077777), the sequence repeats with 00000.

The remaining nine fields can be any of four types: origin, transfer, checksum, or data depending upon the character punched in the first column of the field.

Type of Field	Identifying Character
Data	Any number 0 through 9 (8 and 9 are not generally used since they are converted to 0 and 1, respectively).
Origin	+ (Any ASCII characters with codes less than 60 except 55 are acceptable to the octal loader but are not commonly used.)
Transfer	- (055)
Checksum	Any other characters.

Most of the fields on the cards are *data fields*, which represent object program coding. Each data field thus may contain an instruction.

An *origin field* defines the starting address of an area into which the following data fields are to be loaded. An origin field is always the second field on card 1. Thereafter in the deck, origin fields and data fields can be mixed in any order.

The *transfer field* specifies the address to which control is transferred after all the object code is loaded. The last field in the deck is a transfer field. Any fields following the transfer field on the card are skipped by the octal loader but may contain comments.

The *checksum field* is used by the octal loader to check the validity of the data fields. The checksum field must be the last field on a card. The checksum will be calculated starting from the most recent preceding checksum field to the present one; if no checksum field is present, the check will not be made.

Octal Patches: An octal patch card has the same format as the cards in the octal deck. At least one origin field must appear on the first patch card, the sequence number of each card must be correct (greater than the number on the preceding card), and a transfer field to start the program must appear on the last card of the patch. The patch must be placed after the card containing the checksum field.

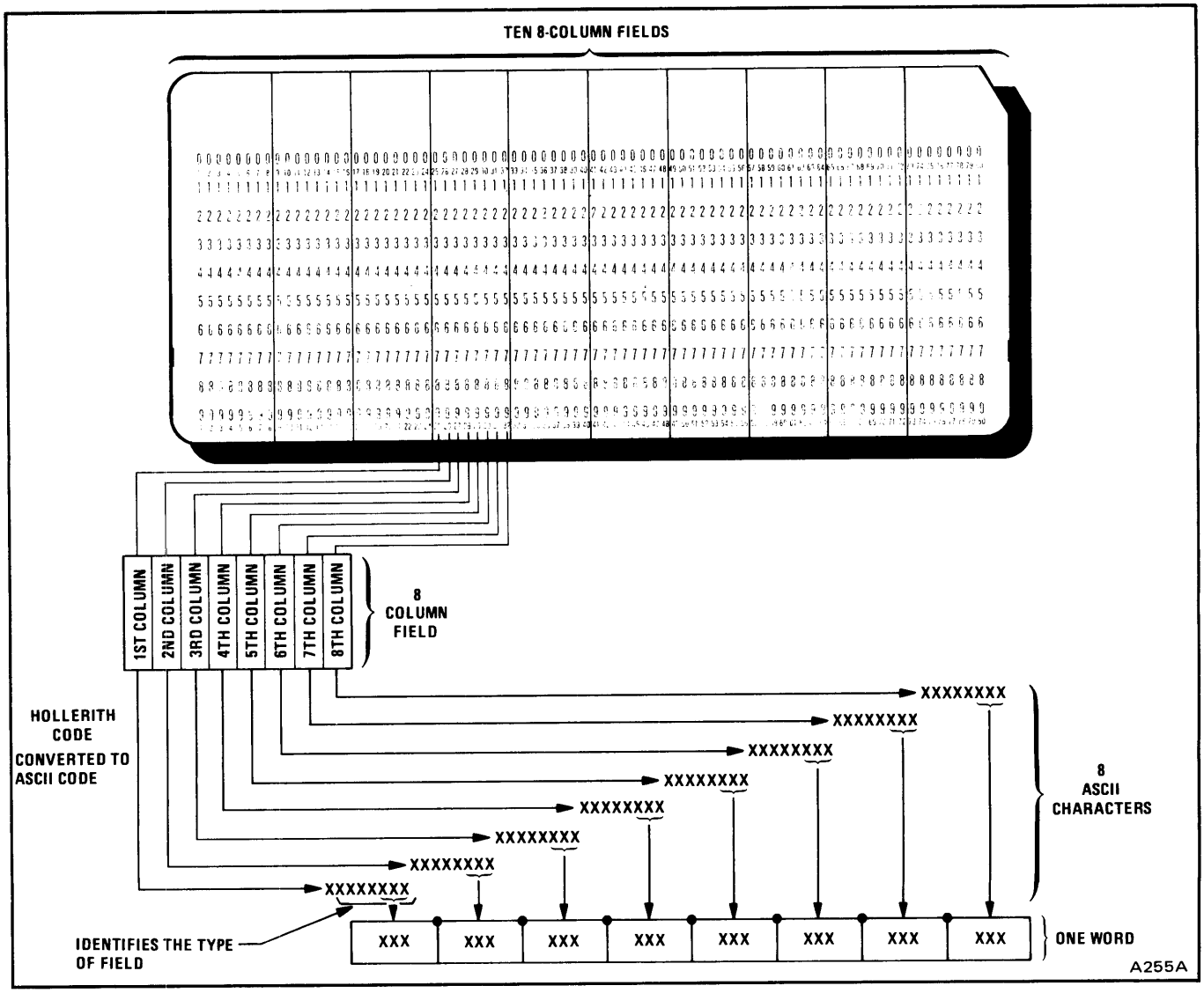
Messages: CDDC can print the following error messages:

```
NAME FIELD ERROR!                       (no output name given)
SEQUENCE ERROR                          (on relocatable or octal files,
                                            cards out of sequence)

CHECK SUM ERROR                         (on relocatable or octal files)
STORE OUT OF BOUNDS                     (on octal files, data origined
                                            too high or too low on
                                            SECTORS parameter)


FATAL DISK ERROR OR OUT OF ROOM!
OUTPUT FILE PROTECTED!
NOT ENOUGH ROOM ON DISK (FATAL)!
DIRECTORY FULL OR FATAL DISK ERROR!
PARAMETER CARD ERROR!
```

Subroutines Used: See "MONITR". Also DPUT, $DUMP, EXIT.

Disc Versions: CDDC is furnished as source file P57 and as absolute program CDDC.

Format of Octal Cards:



DOS MONITR PROCESSOR COPY

Purpose: To copy files from drive to drive.

Requirements: The 8231 Disc Unit, 4K words of memory and the DOS MONITR.

Usage: Enter the control statement:

```
// COPY
```

Followed by groups of parameter statements, separated by blank statements (blank card or cursor return), followed by //b into SYSIN. Each group of parameter statements is as follows:

```
/INPUT = name_i @ drive_i.
/OUTPUT = name_o @ drive_o.
/MONITR.                                    (optional)
/CLEAR.                                     (optional)
```

File "name$_i$" will be copied from drive$_i$ to drive$_o$, and the output file will be named "name$_o$". Drive$_i$ may equal drive$_o$.

/MONITR may be used only during system generation. The input file will be copied to the output drive, but the file will be stored in sectors 0-5. This option should be used with caution, since it can blow the cartridge, /CLEAR is only used with /MONITR. If /CLEAR is used, all files will be cleared from the disc directory and allocation tables before the output file is catalogued.

The COPY processor first copies the specified file (the /INPUT parameter), then updates the directory. If the output file is protected, the copy of the input file will be named "TEMP.A". If a file named "TEMP.A" already exists and is protected, then the name "TEMP.B" will be used. If necessary, names up to "TEMP.Z" will be used. If files named "TEMP.A" through "TEMP.Z" are all protected, then the sectors the copied input file occupies will be lost, until BOJ is run.

Note that COPY allows copying a file whose name is the same as one already on the output cartridge. The second file named will be kept and the first file will be deleted. However, if the first file is protected, the second file will be given a "TEMP.x" name.

Messages: The COPY program can print the following error messages:

```
THE INPUT FILE NAME IS ILLEGAL!
THE OUTPUT FILE NAME IS ILLEGAL!
THE INPUT FILE CANNOT BE FOUND!
THERE IS NOT ENOUGH ROOM FOR THE
   OUTPUT FILE!
THERE IS AN UNRECOVERABLE DISC ERROR
   IN THE INPUT FILE!
THERE IS AN UNRECOVERABLE DISC ERROR
   IN THE OUTPUT FILE!
```

THE MONITR FILE IS NOT IN BOOTSTRAP FORMAT!	(MONITR must be in a format suitable for a bootstrap load on the System IV/70)
THE OUTPUT FILE COULD NOT BE CATALOGUED!	(this is an error return from DRUPD)

Subroutines Used: See "MONITR".

Disc Versions: COPY is furnished as source file P17-NB and as absolute program COPY.

DOS PROCESSOR COPY01

Purpose: To copy a disc cartridge from drive 0 to drive 1.

Requirements: Two 8231 Disc Units (designated drives 0 and 1), 6K words of memory and the DOS MONITR.

Usage: Mount the disc to be copied on drive 0 and the scratch disc on drive 1. The scratch disc must be formatted with the correct headers. Bootstrap the system. At the first keyboard, enter the following control statement:

```
// COPY01
```

The program will halt and display on the first screen:

```
CLEAR HALT TO COPY DISC 0 to DISC 1.
```

If the operator moves AUTO/MANUAL to MANUAL and back to AUTO, the contents of disc 0 will be transferred to disc 1. Note that if this procedure is being used to create working discs and it is desired to use DELSYS to clear out source files, the newly created cartridge must be moved to drive 0 for DELSYS to operate.

Subroutines Used: See "MONITR".

Messages: COPY01 can display the following messages on the screen:

XXXXX	(this is the sector address of the sector currently being copied)
INPUT PACK IS BAD	(to try again, move AUTO/MANUAL to MANUAL, then back to AUTO)
OUTPUT PACK IS BAD	(probably a header error. The disc diagnostic should be run on the output cartridge)
COMPLETE	(successful dub. To dub another cartridge put the new cartridge on drive 1, then move AUTO/MANUAL to MANUAL, then back to AUTO twice)

Disc Versions: COPY01 is furnished as source file P51 and as absolute program COPY01.

DOS PROCESSOR CRTDMP

Purpose: To provide a disc-to-video or memory-to-video octal display with editing and rewrite capabilities.

Requirements: The 8231 Disc and 4K words of memory.

Usage: First initialize the console keys as follows: for an 80/81 character-per-line screen, set switch 0 up; for a 48 character-per-line screen, set switch 0 down. Then place the disc drive index (in octal) in switches 21-23. The drive index can be changed for output only; the program must be restarted to change the drive.

Enter the following control statement into SYSIN:

```
// CRTDMP
```

CRTDMP will be loaded into memory at absolute location 01240.

The rest of the initialization can proceed from the keyboard. For a memory-to-screen dump, press function key F1; for disc-to-screen, F2. Next type a 5-digit octal memory or disc address. The memory address is simply the first address to be dumped. The disc address is determined as follows:

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
000	CCC	CCC	CCS	SSS

Where CCCCCCCC is the 8-bit cylinder address and SSSS is the four-bit sector address; see "$DISC".

As soon as the address is supplied, a 64-word (one quarter sector) dump, in octal, will appear on the screen with the cursor positioned at the first character. The following keys can be used to manipulate the cursor and change the display:

- Cursor up (Roll up) — Positions cursor to preceding line.
- Cursor down (Roll down) — Positions cursor to next line.
- Cursor space (Insert) — Positions cursor to next character.
- Cursor backspace (Delete) — Positions cursor to preceding character.
- EOM — Signifies that the user is finished with this segment. Causes the display screen to be cleared and then waits for a new format (key F1 or F2) and a new address.
- Shifted EOM — Signifies that the user has altered the segment and it is to be rewritten. Caution should be exercised so that data is not inadvertently destroyed. After rewriting data, the function is the same as EOM.
- ATTN — Signifies that user desires to select a new address. Note that format cannot be changed using the ATTN key. Resets the pointer to the start of the new sector.
- Shifted cursor up (Roll ↑) — Causes the next 64 word block of memory or the disc to be displayed.
- Key 0-7 or space — The octal character is entered at the cursor position and the cursor is advanced. Data entered in the blank portion of the screen or in the location address will be ignored.

- Shifted cursor down (Roll ↓) — Executes BAL EXIT to return to MONITR.
- F11 — Print the current sector. [LPOUT] = 5 is furnished but may be changed.

Method: The areas specified by the user are read into memory and converted into an ASCII octal-equivalent display, then transferred to the first screen.

Restrictions: The software discards the first character read from the keyboard because it may be left over from a previous operation; the user should press the first function key twice.

Subroutines Used: The DOS routine $DISC.

Disc Versions: CRTDMP is furnished as source file P42 and as absolute program CRTDMP.

DOS PROCESSOR DCCD

Purpose: Convert disc files to card decks.

Requirements: The 8231 Disc Unit, 4K words of memory, the DOS MONITR routine $DISC, DOS SYSLIB routine EXIT, and library CHLL where DGET resides, and a user provided card punch driver.

Usage: Enter the control statement:

```
// DCCD
```

Followed by groups of parameter statements separated by blank statements (blank card or cursor return), followed by //b into SYSIN. Each group of parameter statements consists of:

```
/INPUT = name @ drive.                    (required)
/OUTPUT = name @ drive.                   (optional)
/ALL = parameter.                         (optional)
```

The /INPUT parameter is always required, and specifies the input file for conversion to cards. The /OUTPUT parameter is optional, and if omitted, the default option is taken; i.e., output is appended to the end of the previous output file (Disc or Card Punch). The initial default option for output is the punch, and output can be directed to the punch by using the output parameter with no file name.

The /ALL parameter is used with /INPUT = b. If /ALL is given with no "= parameter", the complete disc will be punched. If /ALL = S, all source files will be punched. If /ALL = R, the relocatable files will be punched, in hexadecimal form. If /ALL = O, the object code will be punched as octal cards.

Data files and core load files (contiguous) are converted to octal card format. Source files generate source decks, and relocatable files are punched in the format of octal cards, but the conversions is to hexadecimal.

This program operates as the functional opposite of the CDDC program.

At present, this program is furnished only in source form so that the user may furnish his own output punch driver. No punch device is currently supported by Four-Phase Systems. The procedure for assembling this program with a user supplied punch driver is as follows: obtain a listing of DCCD using the assembler. Note that at card image 119300 the location PUNCH is found; the comments explain the calling sequence for the Punch routine:

```
BAL     PUNCH
 PZE    MODE                    (address of punch mode code, dependent
                                  on card punch used)
 PZE    ADDRESS OF CARD BUFFER
```

Edit the user punch driver into this routine using SNEDIT; a convenient location for insertion of the driver is after card image 121000.

Messages: DCCD can print the following messages:

```
OUTPUT FILE PROTECTED!
DIR FULL OR DISK ERROR!        (general disc error)
CONTROL CARD ERROR!
FILE NOT IN DIRECTORY!
DPUTI DISK ERROR!              (disc error in DPUTI only)
```

Subroutines Used: See "MONITR". Also $DUMP, EXIT, DGET.

Disc Versions: DCCD is furnished as source file P58.

DOS PROCESSOR DELSYS

Purpose: To remove all system source files and relocatable files (except the library files) from working cartridges.

Requirements: The 8231 Disc Unit, 4K words of memory, and the DOS MONITR.

Usage: Enter the following control statement into SYSIN:

```
// DELSYS
```

The computer will halt and display the following message on SYSOUT:

```
// PAUSE    LAST CHANCE TO CHANGE MIND BEFORE DELETING FILES
```

To delete the files, at the computer console move the AUTO/MANUAL switch to MANUAL, then to AUTO. DELSYS also deletes itself.

Restrictions: Any files, other than the system relocatable and absolute files, which are desired for use on this cartridge, should be assembled (and loaded if required) before DELSYS is executed. For information on the system core (i.e., MONITR, assembler, loader, processors and libraries), see the descriptions of the programs or the directory dump of a DELSYSed cartridge.

Subroutines Used: See "MONITR". Also DIRMOD, JOB.

Disc Versions: DELSYS is furnished in source form as a control file.

DOS LIBRARY ROUTINE DGET

Purpose: Read chained sequential disc files a byte at a time.

Requirements: The 8231 Disc Unit, 0130 words of memory, a user supplied disc buffer, and library file CHLL or CHRR where DGET resides.

Usage: Before any bytes can be input, the file must be opened. Use:

```
BAL     DGETI
 PZE    DRIVE INDEX
 PZE    VALUE
 PZE    BUFFER ADDRESS
 BRA    EOF
```

To open a chained sequential file for input. VALUE is the address of a double word containing the file pointers, as returned by DRFND. BUFFER ADDRESS is the address of a 256 word buffer that DGET may use to read sectors. DGET will automatically read the next sector if required. EOF is the address of the next routine to be executed. When DGET is called and there are no more bytes in the file, the EOF parameter will be executed.

Whenever a byte is desired, use

```
BAL     DGET
```

The byte will be left or right justified in RA, depending on whether

```
/LIBRARY = CHLL.      or
/LIBRARY = CHRR.
```

respectively is included in the load step (see "LOADOV"). Unused bytes will be zero.

DGET files do not need to be closed.

Restrictions: DGET may have at most one file open at a time. If DGET is used for reading files written by SINDSK, characters that were repeated on the input will appear compressed. The compression involves three characters: first the compressed character, second ESC (033), then an 8-bit count of the number of times the compressed character is repeated. Software using DGET may be required to perform decompression. See "SINDSK".

Errors: DGET will take the EOF exit if there is an unrecoverable disc error, or if the pointers in the chained format sector header are in error. The value of [RA] indicate which exit has been taken:

[RA]	Reason
01000	End of File
0400	File not a chained file
0200	Linkage error
040	Disc error

Subroutines Used: $DISC is used to read sectors.

Register Usage: Only X1-X3 are saved.

Disc Versions: DGET is furnished in the source files P1710B (CHLL) and P1711B (CHRR), and in relocatable form in the relocatable libraries CHLL and CHRR.

DOS MONITR PROCESSOR DIRDMP

Purpose: Print out the disc directory on SYSOUT.

Requirements: The 8231 Disc Unit, 0646 words of memory, the DOS MONITR, and a SYSOUT printer.

Usage: Enter the control statement:

```
// DIRDMP
/DRIVE = n.          (optional)
//                   (if DRIVE is used)
```

Into SYSIN, followed by carriage return. (If no /D is used, give two carriage returns.) The directory for drive n will be printed. If no drive is specified, the directories for all drives on the system will be printed. See "Directory Entry Format".

Method: A new page is started for each new directory. X + 0 and X + 1 (the file name) are printed as a 6-character name. The printer spaces, prints fields B through D, spaces, and prints fields E through G with spaces between. B = P is protected, b if not. C = R if relocatable, b is source or absolute, C if control, A if the file name is an alias. D = 0 if contiguous, 1 if chained. E = octal address. F = sector count in octal.† G = octal address.

Error Message: If a nonready or nonexistent drive is requested, DIRDMP will print

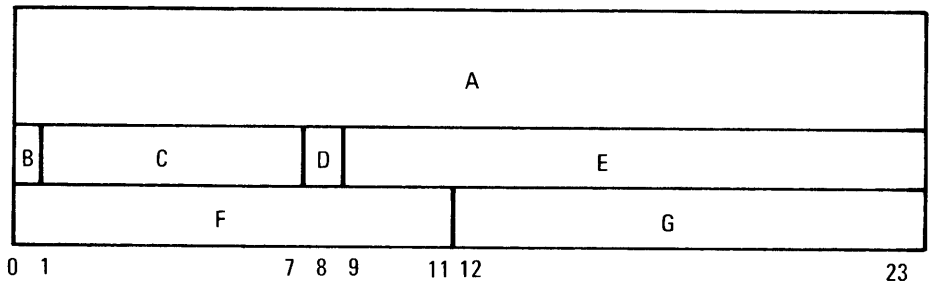
```
"DRIVE NUMBER n IS NOT READY!"
```

Subroutines Used: See "MONITR".

Disc Versions: DIRDMP is furnished as source file P17-J and as absolute program DIRDMP.

† The count -1 is kept in the table; 1 is added for printing.

Directory Entry Format:



Word	Bits	Symbol	Meaning
X + 0, X + 1	0-23 0-23	A	Six character ASCII name with all parity bits 0.
X + 2	0	B	1 = protected 0 = automatic deletion rule applies, i.e., a file is deleted whenever a new file of the same name is created.
	1-7	C	Flag byte. Seven bits of information available to application programs.
	8	D	Format bit. 0 = contiguous 1 = chained
	9-23	E	Contiguous files: E = 0 if data, otherwise E is load address in octal of memory load. Chained files: E = ending sector in octal; 06177 (3199) maximum.
X + 3	0-11	F	Sector count -1 in octal, F≤3199.
	12-23	G	Starting sector in octal; 06177 (3199) maximum.

A131B

DOS MONITR PROCESSOR DIRMOD

Purpose: To aid in maintaining the disc directory. One use is to delete old files. DIRMOD is called directly from SYSIN; DRUPD and DRMOD are called from processors and application programs.

Requirements: The 8231 Disc Unit, 0333 words of memory, and a SYSOUT printer.

Usage: Enter the control statement:

```
// DIRMOD
```

Followed by groups of parameter statements separated by blank statements (blank card or cursor return), followed by //b into SYSIN. Each group of parameter statements is as follows:

```
/INPUT = name_i @ drive_i.          (required)
/OUTPUT = name_o @ drive_o.         (optional)
/ALIAS = name_a.                    (optional)
/PROTECT.                           (optional)
/Q.                                 (optional)
/FLAG = x.                          (optional)
/LOAD = number.                     (optional)
```

The /INPUT parameter is always required, and specifies the file entry to be changed. If /ALIAS is used, the /INPUT directory entry will be unchanged and a second name for "name$_i$" will be created. /ALIAS overrides any /OUTPUT parameter.

If no /OUTPUT parameter is used, the /INPUT name will not be modified. If "/OUTPUT = " with no name is used, then "name$_i$" will be deleted. If "/OUTPUT = name$_o$" is used, the file will be renamed "name$_o$". The other parameters are independent of /ALIAS and /OUTPUT. /PROTECT sets the protect flag. /Q resets the protect flag and allows protected files to be changed. /FLAG sets the flag type to "X". /LOAD changes the load address to "number".

If a file with the same name as an existing file is created, the old file is deleted unless it is protected, in which case, an error message will occur.

Restrictions: /ALIAS should be used with caution. If one ALIAS is deleted, the file will be deleted, but the other ALIASes in the directory will not be updated.

Messages: DIRMOD can print the following error messages:

```
THE /I PARAMETER IS INVALID
THE /I FILE IS NOT IN THE DIRECTORY
THE /L PARAMETER IS NOT A NUMBER
THE /A PARAMETER IS INVALID
THERE WAS AN UNRECOVERABLE DISK ERROR
```

```
THE /I FILE IS PROTECTED
THE /L PARAMETER CAN'T BE USED ON CHAINED FILES
THE DIRECTORY IS FULL
THE /O FILE IS PROTECTED
THE /A ENTRY IS PROTECTED
```

Detection of any of these errors will terminate DIRMOD and return control to MONITR.

Subroutines Used: See "MONITR".

Disc Versions: DIRMOD is furnished as source file P17-K and as absolute program DIRMOD.

DOS PROCESSOR DIRSRT

Purpose: To sort the disc directory entries into ASCII sequence.

Requirements: An 8231 Disc Unit, 8K words of memory, and the DOS MONITR.

Usage: Enter the control statements:

```
// DIRSRT
/DRIVE = n.            (optional)
//
```

The directory on drive n will be sorted into ascending ASCII sequence and written back onto the disc. Then, if DIRDMP is evoked, the directory will be printed in this sequence. If new files are created, they will be added at the end of the directory. If no drive index is given, 0 will be assumed.

Method: Bubble sort.

Disc Versions: DIRSRT is furnished as source file P17-R and as absolute program DIRSRT.

DOS LIBRARY ROUTINE DPUT

Purpose: Create chained sequential disc files a byte at a time.

Requirements: The 8231 Disc Unit, 0200 words of memory, a user supplied disc buffer, and library file CHLL or CHRR where DPUT resides.

Usage: Before any bytes may be output, the file must be opened. Use:

```
BAL     DPUTI                          Initialization Routine
 PZE    DRIVE INDEX
 PZE    BUFFER ADDRESS
 PZE    SECTOR ADDRESS
 BRA    ERROR
```

To open a chained sequential file for output. BUFFER ADDRESS is the address of a 256 word buffer which DPUT may use to build sectors. When the buffer fills up, it is automatically written to the disc, and the buffer is reinitialized. SECTOR ADDRESS is a sector address. DPUT will place the file as close to SECTOR as possible, as described in ALLOC1. ERROR is the address of an error routine.

To output a byte from RA, use:

```
BAL     DPUT
```

It will take the left or right byte of RA depending on whether

```
/LIBRARY = CHLL.      or
/LIBRARY = CHRR.
```

respectively is included in the load step (see "LOADOV"). Unused bytes will be ignored.

After all bytes are output, the DPUT file must be closed. To close the file, use:

```
BAL     DPUTF                          Closing Routine
 PZE    VALUE
```

Where VALUE is the address of a double word in which DPUTF will store the pointers to the file. The pointers will be in the same format as used in the directory (see DIRDMP), with the flag byte and the protect bit both zero. For format of chained files, see illustration "Format of Chained Files on Disc".

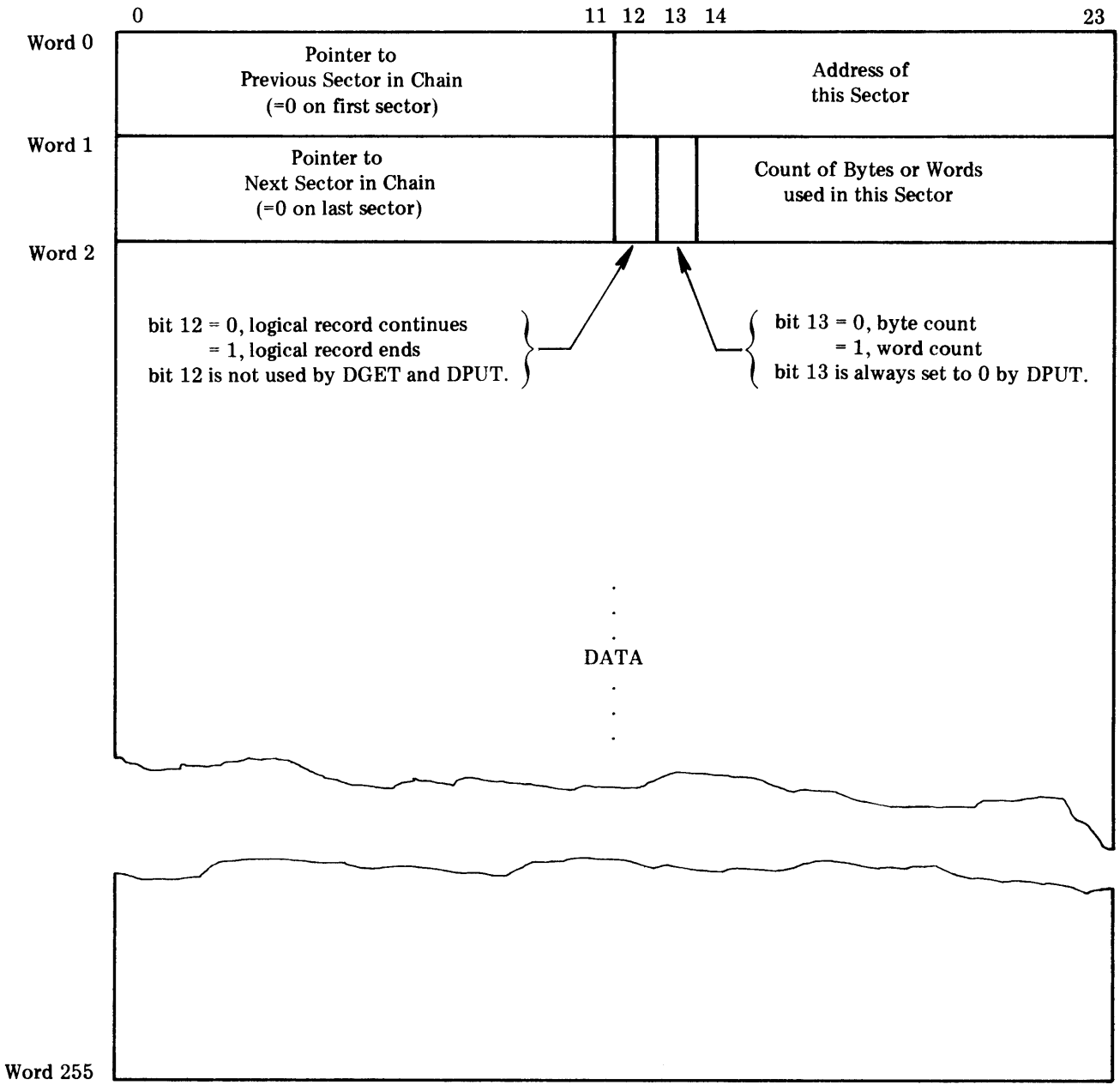
Restrictions: DPUT may have at most one file open at a time.

Errors: DPUT will take the ERROR exit if an unrecoverable disc error occurs, or if there are no available sectors in the specified cartridge. Any program that uses DPUT must be prepared for this condition and should return sectors from files that could not be completed to the availability table for future use or the sectors will be lost, even if JOB is run. However, use of BOJ will recover these sectors.

Subroutines Used: $DISC, ALLOC1, ALLOCF.

Register Usage: Only X1-X3 are saved.

Disc Versions: DPUT is furnished in source form as part of files P1710B (CHLL) and P1711B (CHRR) and in relocatable form in libraries CHLL and CHRR.



Format of Chained Files on Disc

DOS MONITR ROUTINE DRFND

Purpose: Find the disc location and attributes of a file, given its 6 character name.

Requirements: The 8231 Disc Unit, the DOS MONITR and its routine $DISC, and the system buffer BUF.

Usage: Calling sequence:

```
BAL     DRFND
 PZE    DRIVE INDEX
 PZE    NAME
 PZE    VALUE
 BRA    not in table or disc error
```

Where NAME is the location of a double word containing the 6 character name (the system standard name format is 6 characters in the 64 character ASCII subset, left justified with blank fill). VALUE is the location of a double word where the value fields of the directory entry NAME (i.e., words X + 2 and X + 3) will be stored if the search is successful. See "Directory Entry Format" under DIRDMP.

To load a memory load and execute, construct a request table (see REQTAB under $DISC), using [x + 3] in the directory entry as [REQTAB + 3] and [x + 2] as [REQTAB + 2]. The no error return from $DISC would be a branch to the load address: [x + 2]. Next execute BAL $DISC, and the program will load and go.

A recommended method for using DRFND is to leave MONITR resident. This is accomplished by origining the user program at or above 03400 and by placing /I = SYSTEM first in the parameter string at load time.

Method: DRFND sequentially searches the disc directory (see DIRDMP), until either the name is found, a name of zero is found (i.e., the end of the working part of the directory), or the complete directory has been processed. For fixed format usage on the disc, see MONITR.

Subroutines Used: The DOS routine $DISC.

Register Usage: Only X1-X3 are saved.

Disc Versions: DRFND is furnished in source form as part of file P17-7B and in absolute form as part of MONITR.

DOS MONITR ROUTINE DRMOD

Purpose: Modify the disc directory. Note that DRUPD should usually be used for the standard "automatic deletion rule" for storing files on the disc, rather than using DRMOD directly.

Requirements: The 8231 Disc Unit, DOS MONITR routines DRFND and $DISC, and the system buffer BUF.

Usage: DRMOD is used only after DRFND has been successful. DRMOD changes both the name and the value of the entry last found by DRFND to those specified by the calling sequence. The calling sequence is:

```
BAL     DRMOD
 PZE    NAME
 PZE    VALUE
 BRA    protected entry
 BRA    disc error or parameter error
```

Where NAME and VALUE are symbolic names pointing to double words containing the desired new name and new value (see DRFND). If a new name is used, the old file is not deleted.

A recommended method for using DRMOD is to leave MONITR resident. This is accomplished by origining the user program at or above 03400 and by placing /INPUT = SYSTEM first in the parameter string at load time.

Restrictions: DRFND must have been successful on its last call before DRMOD can succeed. DRMOD exists and is used principally as a subroutine under DRUPD.

Errors: DRMOD takes its error exit in any of these three cases:

- DRFND was not successful or not called.
- An unrecoverable disc error occurs.
- The protect bit (bit 0 of the third word) is on.

The third error may be overridden by using the following calling sequence:

```
BAL     DRMOD
 PZE    NAME
 PZE    VALUE
 NOP
 BRA    error
```

Subroutines Used: The DOS routine $DISC.

Register Usage: Only X1-X3 are saved.

Disc Versions: DRMOD is furnished in source form as part of file P17-7B and in absolute form as part of MONITR.

DOS MONITR ROUTINE DRUPD

Purpose: Create a new file, deleting any old files of the same name.

Requirements: The 8231 Disc Unit, DOS MONITR routines DRFND, DRMOD, and $DISC, and the system buffer BUF.

Usage: Calling sequence:

```
BAL     DRUPD
 PZE    DRIVE INDEX
 PZE    NAME
 PZE    VALUE
 BRA    PROTECT
 BRA    DIRECTORY FULL OR DISC ERROR
```

Where NAME and VALUE are symbolic names pointing to double words containing the desired new name and new value (see DRFND).

A recommended method for using DRUPD is to leave MONITR resident. This is accomplished by origining the user program at or above 03400 and by placing /INPUT = SYSTEM first in the parameter string at load time.

Method: If any old entry with the same name exists, the name field is replaced with all blanks. This special reserved name signifies to the program JOB that the disc file specified by the value field may be returned to the disc availability table. Then DRUPD replaces the next available entry whose name field is binary zeros with the name and value specified. The following calling sequence will allow the automatic deletion rule to be used on protected files:

```
BAL     DRUPD
 PZE    DRIVE INDEX
 PZE    NAME
 PZE    VALUE
 NOP
 BRA    ERROR
```

Subroutines Used: DRFND, DRMOD, $DISC.

Register Usage: Only X1-X3 are saved.

Disc Versions: DRUPD is furnished in source form as part of file P17-7B and in absolute form as part of MONITR.

DOS PROCESSOR DTU

Purpose: To selectively dump disc to tape in such a way that the output tape, when bootstrapped, will selectively dump tape back to disc.

Requirements: The 8231 Disc Unit, an 8511/8512 Magnetic Tape Unit, and 4K words of memory.

Usage: Enter the control statement:

```
// DTU
```

Followed by a group of parameter statements into SYSIN; these must be entered through the keyboard. Each group, except the last group, is followed by a data statement and a message data statement. The last group is followed by a // and a message statement. The parameter statements are as follows:

```
/INPUT = DISC @ drive_i.
/OUTPUT = TAPE @ deck_o.
/LOW = number_l.                    (optional)
/HIGH = number_h.                   (optional)
// or data statement
message statement
```

The disc drive unit is "drive$_i$". The tape deck is "deck$_o$", which must be 0 for the first group of parameter statements. The entries "number$_l$" and "number$_h$" are sector addresses that denote the range of sectors to be dumped. The default values of "number$_l$" and "number$_h$" are 0 and 06177 respectively. A blank statement is conventionally used to terminate each parameter string except the last, which requires a "//b"; however, a comment may be used as a separator. In summary, the above parameter statements tell DTU to dump sectors "number$_l$" to "number$_h$" from "drive$_i$" to "deck$_o$". After the // or data statement is entered, DTU will halt and display the parameters for verification. The halt must be cleared (AUTO/MANUAL down, then up) before the message statement can be entered. This message statement is displayed on the screen when the tape is bootstrapped.

The output tape produced will contain a program and data from the dumped sectors for every group of parameter statements. This program is in bootstrap format and will dump the sectors back onto disc. When the system is bootstrapping from tape, the console keys should be at 37705221. Bootstrapping will load the programs onto the same sectors of the same cartridges on the same drives from which they were dumped.

Messages: DTU can print the following error messages:

```
WRONG LENGTH RECORD
UNEQUAL SECTOR ADDRESSES
OUTPUT DISC ERROR
INVALID OPTION
NOT READY, READY TAPE DECK
TAPE ERROR
NO INPUT SPECIFIED
NO OUTPUT SPECIFIED
```

```
INVALID INPUT-OUTPUT COMBINATION
INVALID DECK #
INVALID DRIVE #
INPUT DISC ERROR
INVALID SECTOR RANGE
INVALID SECTOR ADDRESS
COMPLETE, BUT NO TAPE MARK SENSED
COMPLETE, BUT NO TAPE MARK OR SECOND FILE SENSED
LOST INTERRUPT
```

Subroutines Used: $DISC, $TAPE, OPTION, EXIT, SYSIN.

Disc Versions: DTU is furnished in source form as P62A1, STAPE, P62A3, and P62LIB, and in absolute form as DTU Control file C62AL assemblies and loads DTU.

DOS PROCESSOR DTU16

Purpose: To selectively dump disc to tape in such a way that the output tape, when bootstrapped, will selectively dump tape back to disc.

Requirements: The 8231 Disc Unit, an 8513 Magnetic Tape Unit, and 4K words of memory.

Usage: Enter the control statement:

```
// DTU16
```

Followed by a group of parameter statements into SYSIN; these must be entered through the keyboard. Each group, except the last group, is followed by a data statement and a message data statement. The last group is followed by a // and a message statement. The parameter statements are as follows:

```
/INPUT = DISC @ drivei.
/OUTPUT = TAPE @ deck0.
/LOW = numberl.                    (optional)
/HIGH = numberh.                   (optional)
// or data statement
message statement
```

The disc drive unit is "$drive_i$". The tape deck is "$deck_o$", which must be 0 for the first group of parameter statements. The entries "$number_l$" and "$number_h$" are sector addresses that denote the range of sectors to be dumped. The default values of "$number_l$" and "$number_h$" are 0 and 06177 respectively. A blank statement is conventionally used to terminate each parameter string except the last, which requires a "//b"; however, a comment may be used as a separator. In summary, the above parameter statements tell DTU16 to dump sectors "$number_l$" to "$number_h$" from "$drive_i$" to "$deck_o$". After the // or data statement is entered, DTU16 will halt and display the parameters for verification. The halt must be cleared (AUTO/MANUAL down, then up) before the message statement can be entered. This message statement is displayed on the screen when the tape is bootstrapped.

The output tape produced will contain a program and data from the dumped sectors for every group of parameter statements. This program is in bootstrap format and will dump the sectors back onto disc. When the system is bootstrapping from tape, the console keys should be at 37705221. Bootstrapping will load the programs onto the same sectors of the same cartridges on the same drives from which they were dumped.

Messages: DTU16 can print the following error messages:

```
WRONG LENGTH RECORD
UNEQUAL SECTOR ADDRESSES
OUTPUT DISC ERROR
INVALID OPTION
NOT READY, READY TAPE DECK
TAPE ERROR
NO INPUT SPECIFIED
NO OUTPUT SPECIFIED
```

```
INVALID INPUT-OUTPUT COMBINATION
INVALID DECK #
INVALID DRIVE #
INPUT DISC ERROR
INVALID SECTOR RANGE
INVALID SECTOR ADDRESS
COMPLETE, BUT NO TAPE MARK SENSED
COMPLETE, BUT NO TAPE MARK OR SECOND FILE SENSED
LOST INTERRUPT
```

Subroutines Used: $DISC, $TAPE, OPTION, EXIT, SYSIN.

Disc Versions: DTU16 is furnished in source form as P62BD, P62B$, P62BB and P62LIB, and in absolute form as DTU16. Control file C62BL assemblies and loads DTU16.

DOS LIBRARY ROUTINE EXIT

Purpose: EXIT returns control to the DOS program MONITR in a standard and orderly way.

Requirements: The 8231 Disc Unit and 045 words of memory. EXIT resides in SYSLIB.

Usage: To return control to MONITR without specifying the next processor, use:

```
BAL     EXIT
 DCN    0
```

To return control to MONITR, effectively forcing a "// NEXT" statement into the SYSIN stream, use:

```
        BAL     EXIT
         DCN    NAME
         .
         .
         .
        FORCE   0
NAME    DCA     'NEXT'
```

Special Features: If MONITR is in memory and SBUF contains // NEXT, then:

```
BRA     MONITR
```

Is equivalent to the second calling sequence given above. LDRHI and LDRLO contain specialized versions of EXIT.

Method: See the flow chart under MONITR. EXIT contains its own disc routine.

Restrictions: EXIT must be above 0400 for correct operation.

Disc Versions: EXIT is furnished in source form as part of file P17-9 and in relocatable form in SYSLIB.

DOS PROCESSOR FILDMP

Purpose: To dump chained or contiguous disc files to the system printer.

Requirements: The 8231 Disc Unit, 4K words of memory, a SYSOUT printer, and the DOS MONITR.

Usage: Enter the control statements:

```
// FILDMP
/INPUT = name @ drive.  or      }        (one required)
/INPUT = disc address @ drive.  }
/EJECT.                                  (optional)
//
```

Into SYSIN. The input file "name" on drive "drive" or the sector "disc address" on "drive" will be printed. If the file or sector is a chained file it will be printed in octal, 16 bytes to a line, followed by an ASCII interpretation. If it is a contiguous file, it will be printed eight octal words per line, followed by an ASCII interpretation. The program checks the format bit in the directory entry to determine whether the file is chained or contiguous and proceeds accordingly. If /E is specified, the page will be ejected before printing starts.

Restrictions: FILDMP may be used to list source files, but the interpretation will not be clear — the preferred method is to use LIST.

Disc Versions: FILDMP is furnished as source file P17-S and as absolute program FILDMP.

DOS MONITR PROCESSOR JOB

Purpose: JOB returns all sector storage which has been deleted to the disc availability table.

Requirements: The 8231 Disc Unit, 6K words of memory, and the DOS MONITR.

Usage: Enter the following control statement into SYSIN:

```
// JOB
```

Method: JOB will automatically delete and return all files whose name is blank or starts with the four letters TEMP. JOB also rebuilds the directory and compresses it by moving entries into the space where entries have been deleted.

Restrictions: JOB assumes that all disc drives that are ready have DOS cartridges on them. JOB will destroy sectors $6\text{-}15_{10}$ on non-DOS formatted cartridges.

Messages: If JOB executes successfully, it will print on SYSOUT:

```
MMMMM SECTORS WERE RETURNED TO DRIVE D, MAKING THE TOTAL NUMBER
   OF SECTORS IN THE DISC POOL NNNNN
```

Where MMMMM is the number of deleted sectors, D is the drive involved, and NNNNN is the number of sectors available for use.

If an unrecoverable disc error occurs, JOB will print on SYSOUT:

```
SECTOR NNNNN ON DRIVE D HAS AN UNRECOVERABLE ERROR
```

Where NNNNN is the sector and D is the drive where the error was detected.

Subroutines Used: See "MONITR".

Disc Versions: JOB is furnished as source file P17-LB and as absolute program JOB.

DOS PROCESSOR LIST

Purpose: To list control files and other source files.

Requirements: The 8231 Disc Unit, 4K words of memory, and the DOS MONITR.

Usage: Enter the following control statements:

```
// LIST
/INPUT = file @ drive. }        may be repeated as many times
blank card             }        as necessary.
//
```

Into SYSIN. The source file thus specified will be listed on SYSOUT. Unlike FILDMP, LIST performs decompression and treats ends of file (//b) statements as data. Thus, control files will be printed with no interpretation.

LIST handles source records up to 132 bytes in length. Longer records are truncated.

Disc Versions: LIST is furnished as source file P17-T and as absolute program LIST.

DOS MONITR PROCESSOR LOAD

Purpose: Load relocatable files produced by ASM, and either load-and-go, catalogue, or punch on SYSPCH the resulting absolute binary memory load.

Requirements: The 8231 Disc Unit, operates in 6K words of memory (the loader itself uses 02162 words), and the DOS MONITR.

Usage: Enter the control statements:

```
// LOAD
/INPUT = name1 @ drive1.          (required)
/INPUT = name2 @ drive2.          (optional)
         .
         .
         .
/INPUT = namen @ driven.          (optional up to 15 inputs)
/POSITION OF LOADER = LO.         (optional)
/LIBRARY = libry.                 (optional)
/U ERRORS OK.                     (optional)
/OUTPUT = nameo @ driveo.         (optional)
//
```

Into SYSIN. The relocatable files "name$_i$" will be loaded in the order specified from "drive$_i$". After all input files are loaded, "libry" or the default SYSLIB on drive zero will be used to resolve any remaining virtuals. A library may be any named relocatable file created by the user; libraries may be chained using EOP source statements (see "System Relocatable Files"). The load map will then be printed. If any virtuals remain, the loader will return to MONITR unless /U ERRORS OK was specified. If the load is satisfactory, the load module will be catalogued as "name$_o$" on "drive$_o$" as specified in the /O parameter. If there is no output name specified, the load module will be punched on SYSPCH, if any. (Note that at present no system punch is supported, but provision is made for linking to a user supplied punch.) If no /OUTPUT parameter at all occurs, the load module will be immediately executed.

Note that programs that load and go take the starting address for execution from the last END card that contains an operand. Programs that load and store, however, must be executed using a branch to the starting location placed at the program load point. Programs to be loaded under DOS are not allowed to overlay a part of MONITR: thus the load address must be greater than 0410 except in the case of a bootstrap load-and-go program. Bootstrap programs load at 1.

The last entry in SYSLIB is:

```
         ENTRY   :START
:START   BRA     :BEGIN
         END
```

The user may make :BEGIN the starting point for a program merely be defining it and making it an ENTRY; and making :START a virtual (e.g., DCN :START) so it will be picked up from SYSLIB.

Special Features: If /POSITION OF LOAD = LO is included, a version of the loader will be used which resides in low memory, and loads programs from the top of memory downwards. This option is mainly used to load processors (such as COBOL object programs that will generate screen sections in low memory on execution) into high memory where the default loader resides.

Restrictions: At most 15 "/INPUT = name$_i$" parameter statements may be used. The loader cannot load routines which load into the memory that it or the load map resides in.

Messages: For Error halts in the Loader see Appendix B. The following messages may be printed by the Loader:

```
THE FOLLOWING DATA CARD WILL BE LOST (followed by the card image
                                       that will be lost: occurs
                                       if a data card has ter-
                                       minated the input to OPTION)
XXXXXX IS NOT IN THE DIRECTORY       (where XXXXXX is the input file
                                       name)
NO INPUT FILE WAS SPECIFIED
ONE INPUT FILE WAS A NUMBER
XXXXXX IS NOT IN THE CHAINED FORMAT  (where XXXXXX is the file name)
TOO MANY INPUTS. ONLY THE FIRST FILES WILL BE LOADED.
LIBRARY NOT ON DRIVE 0 - NOT SUPPORTED
INPUT NOT RELOCATABLE
```

Subroutines Used: See "MONITR".

Disc Versions: LOAD is divided into two parts. LOAD, the preprocessor, is furnished as source file P17-F and as absolute program LOAD. LDRHI and LDRLO, the loader programs, are called by LOAD (see P parameter above) and are both furnished as the source files P12-1B and P12-2B which together can be assembled to form the loader; they are also furnished as the absolute files LDRHI and LDRLO.

DOS MONITR PROCESSOR LOADOV

Purpose: To load relocatable files and either load-and-go or catalogue the resulting absolute binary memory load. Overlay capability is included.

Requirements: The 8231 Disc Unit, 8K words of memory, and the DOS MONITR.

Usage: Enter the control statements:

```
// LOADOV
/INPUT = file1 @ drive1.                (required)
/INPUT = file2 @ drive2.                (optional up to 15 inputs)
/POSITION OF LOADER = LO.               (optional; default high)
/LIBRARY = library.                     (optional; default SYSLIB)
/UNRESOLVED VIRTUALS OK.                (optional; default "not ok")
/OUTPUT = file3 @ drive3.               (optional in main, illegal in
                                          higher modules)
/MODULE = name.                         (not in main, required in
                                          higher modules)
/ROOT = name.                           (not in main)
/BOTTOM OF MODULE = name or addr.       (optional)
/TOP OF MODULE = name or addr.          (optional)
/EXECUTE = name or addr.                (optional)
blank card
.
.
.
//
```

Into SYSIN. The input file (files) will be processed by the relocatable loader in an attempt to generate an executable output module. If successful, the module will be catalogued as "file3" on "drive3" as specified in the /O parameter. If no /O parameter is specified, the module will be immediately executed.

The /U and /P parameters are the same as for the DOS MONITR processor LOAD: If there are any unresolved virtuals, the loader will return to MONITR unless /UNRESOLVED VIRTUALS OK was specified. If /POSITION OF LOADER = LO is included, a version of the loader will be used that resides in low memory, and loads programs from the top of memory downwards. This option is mainly used to load certain screen-oriented programs into high memory (where the default loader resides). Specifically, the option is used for programs (such as COBOL object programs) that generate screen areas in low memory during their execution.

If more than one set of parameters is given (blank cards are separators), the loader will generate a tree-structure of modules, using the first parameter string to generate the main-root module and the second and further modules for the branches. The maximum number of /I files for each module is 15, but the number of branch modules that can be generated is only limited by what the disc or discs can hold. The /O parameter applies only to the main-root module; the name thus specified will be taken as the root name and the name of the output disc file. If no /O parameter is given, TEMP.. is assumed, and // TEMP.. will be inserted into SYSIN to cause a load-and-go.

If the /L parameter is given, the specified library will be used for resolving virtuals in the current module only — the process of resolving virtuals using libraries will begin only after all the modules in common paths have been scanned for ENTRYs that satisfy the virtuals in this module.

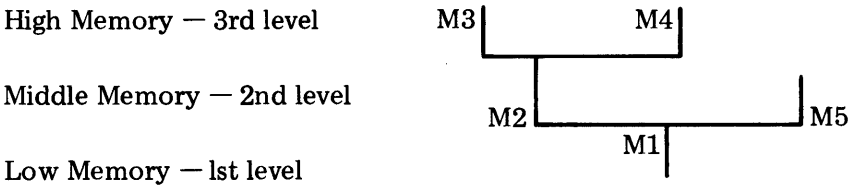
The /M parameter allows the user to specify the name by which the module will be called; the method for calling is described below under "Method". A module name must be compatible with both symbol names under ASM and option parameter strings. Thus, it must be six or fewer characters in length, may not begin with a number, and may not contain the symbols + - * / @ & . comma or blank.

The name associated with the /R parameter is the name of the module located just below the current module — see "Method" for the meaning of "below" in this context. This root module must already have been specified. If no name is given with ROOT, the default is to the main module.

The /B and /T parameters allow the user to specify symbolic or absolute locations for the loading of the module.

The /E parameter causes a branch to the starting point to be appended to the module, and allows a variety of methods for specifying this starting point. If no /E parameter is given, no branch is added. If /E is given without a value (no "= name or addr"), the loader takes the symbolic or absolute location for starting from the operand field of the last END statement in this module whose operand field was nonblank. If a symbolic name or an address is given in the /E statement, the branch address is the point thus specified.

Method: The order in which the modules are specified in the control statements for LOADOV determines the order of their interdependency and the order in which they can be executed. The fundamental approach is to build and execute programs as tree structures. One or more levels of load modules are constructed, each level of which is subordinate to the one below and may have other levels (above) subordinate to it. Of course, there is a bottom or main-root module, which must be single and has all the other levels subordinate to it. In this structure, a "path" is defined as a module or group of modules that are connected in a one-dimensional manner and contain, within the path, no parallel elements or gaps. For example, in the following diagram M1-M2-M4 is a path and so is M1-M5, but M5-M4 is not. A module with no modules subordinate to it (highest on the tree in its execution chain) is called a leaf. In the example, M1 is the root for all the modules, and M2 is a root for M3 and M4; M3, M4, and M5 are leaves.



M1 is the main-root module. Modules M2 and M5 are called into memory from disc by M1. The method for calling is discussed below. Since M2 and M5 cannot be in memory simultaneously, they are treated on parallel paths. Because M1 is the lowest level module that calls them from memory, they are both placed on the same level, one level up from M1, which is called their root. Similarly M2 is the root module for M3 and M4. M2 is resident when either M3 or M4 is resident. The option card syntax communicates the shape of the tree-structure using the order of the option cards and the /R parameter.

The main module is always resident. Higher level modules may also be resident, including all the modules in the path currently being executed. In general, modules from different paths will not be resident at the same time.

A lower-level module takes precedence over a higher-level module in the sequence for resolving virtuals, with all the load modules in a path taking precedence over any libraries. Each path must be exhausted before another can be started. Thus, in the example, all the modules connected with M2 must be loaded before M5, or a different tree-structure will be generated. The required order of loading for the example is:

```
// LOADOV
/INPUT = I1.
/OUTPUT = M1.
(blank card)
/INPUT = I2.
/MODULE = M2.
/ROOT = M1.
(blank card)
/INPUT = I3.
/MODULE = M3.
/ROOT = M2.
(blank card)
/INPUT = I4.
/MODULE = M4.
/ROOT = M2.
(blank card)
/INPUT = I5.
/MODULE = M5.
/ROOT.
//
```

$OVRLY is a SYSLIB routine that allows modules higher than the main-root module (including overlays) to be called by object programs. Following is the calling sequence and library routine in a root module to allow the module (/MODULE = NAME) on a higher level to be called. When the main-root module is bootstrapped into memory by MONITR, X3 is initialized with a value (the sector address of the main-root module), which must be stored in $DBASE before $OVRLY is executed and before the user program changes X3. Note that "NAME" must be defined as an ENTRY in a lower-level module in the same execution chain as the module. NAME defines a word-pair (BSS 2) on an even boundary (FORCE 0) where the loader will place MZE Memory Address, DCN Disc Address. Note that this procedure merely calls the specified modules into memory; no provision is made for passing control between modules. This is a user function.

```
*CALLING SEQUENCE FOR $OVRLY TO BE USED BY ROOT MODULE
        ST3    $DBASE   MUST BE EXECUTED BEFORE X3 IS MODIFIED
        .
        .
        .
        BAL    $OVRLY   AFTER MONITR LOADS THE MAIN MODULE
        DCN    NAME     LOAD THE OVERLAY MODULE'S NAME
        BRA    REJECT   ERROR EXIT
```

```
*ROUTINE FETCHED FROM SYSLIB
        ENTRY $DBASE
        ENTRY $OVRLY
$OVRLY  ST23  RET23    KEEP INDEX REGS
        LD23* 0,X2     SET UP BY LOADER
        AD3   $DBASE   RELOCATE DISC ADDRESS
        ST23  REQ23
        BAL   $DISC    LOAD THE MODULE
         DCN  REQ0
         DEC  RET23    TAKE THE REJECT ADDRESS
        LD23  RET23
        BRA   2,X2     RETURN TO CALLER
$DBASE  BSS   1
        FORCE 0
REQ0    BSS   1
        DCN   0        ALWAYS DRIVE 0
REQ23   BSS   2
RET23   BSS   2
        END
```

Messages: LOADOV prints a load map for each module. The load map contains the following information: name of module; the symbolic name, address, and status of each symbol defined by an ENTRY statement in the module; the minimum and maximum locations allocated for the module in memory; any error messages; and the number of sectors allocated for the module. The status of a symbol is: U = undefined, D = doubly defined, N = not referenced, L = fetched from a library.

LOADOV contains no error halts. The following error messages may be printed:

```
** UNRECOVERABLE DISC ERROR ON SECTOR XXXXX DRIVE Y!
** MODULE ENTRY UNDEFINED!
** MODULE ENTRY IS NOT A DOUBLEWORD!
** VIRTUAL CHAIN ERROR!
** THERE IS NO LIBRARY FILE NAMED XXXXXX!
** THE LIBRARY FILE XXXXXX IS NOT RELOCATABLE!
** NOT ENOUGH CONTIGUOUS DISC SPACE!
** CANNOT BE CATALOGUED!
** LIBRARY MUST BE ON DRIVE X!
** INVALID INPUT NAME!
** XXXXXX IS NOT IN THE DIRECTORY!
** XXXXXX IS NOT RELOCATABLE!
** THERE ARE NO VALID INPUT FILES IN THIS MODULE!
** LOAD STEP ABORTED BEFORE OUTPUT FILE CREATED!
** OVERFLOW!
** TOO MANY INPUTS!
** UNRESOLVED VIRTUALS!
** CAUTION: THIS FILE IS NOT DOS LOADABLE!
** XXXXXX ISN'T DEFINED IN YYYYYY'S ANCESTORS!
** NO MODULE NAME WAS SPECIFIED!
```

```
** NO OUTPUT PARAMETER IS ALLOWED!
** NO SUCH ROOT MODULE EXISTS!
** ROOT DEFAULTS TO THE MAIN MODULE!
** INTERNAL ERROR
** NTH /I FILE HAS DISC ERROR
** NTH /I FILE HAS LINKAGE ERROR
** XXXXXX FILE HAS DISC ERROR     } (XXXXXX is a library name)
** XXXXXX FILE HAS LINKAGE ERROR  }
```

Restrictions: LOADOV will not load programs that ORG executable code or data (memory allocated using BSS directives does not count) between 2 and 0411 (inclusive).

Disc Versions: LOADOV is furnished as source files P12ABW, P12BB, P12CB, P12DB, and P12EB, which assemble and load (/P=LO) to form the executable program. It is also furnished in executable form as LOADOV.

DOS LIBRARY ROUTINE MDGET

Purpose: To read chained sequential disc files a byte at a time. Any number of files can be in process of being read at one time, and a file may be read by more than one reader at a time.

Requirements: The 8231 Disc Unit, 0120 (CHML) or 0122 (CHMR) words of memory, a 256-word user-supplied disc buffer, a nine-word file table to be used by MDGET, and library file CHML or CHMR where MDGET resides.

Usage: The user must supply two items before opening his file using MDGETI:

- The double word VALUE containing the file pointers must be obtained using DRFND and stored into TABLE + 4 and TABLE + 5. See TABLE format below.
- A branch instruction to be executed when an end-of-file or a disc error occurs must be placed in TABLE + 8.

Before any bytes can be input, the file must be opened using:

```
BAL     MDGETI
 PZE    DRIVE INDEX
 PZE    BUFFER ADDRESS
 PZE    TABLE ADDRESS
```

MDGETI initializes a file reader which uses TABLE to store file pointers and parameters and sets the file pointers to the beginning of the file. BUFFER ADDRESS is the address of a 256-word buffer which MDGET will use to read the disc sectors. TABLE ADDRESS is the address of a nine-word table used by MDGET to keep the variables required by the file reader. TABLE format:

```
        FORCE   0
TABLE   DCN     LR0T
+1      DCN     Buffer Pointer for LCR instruction
+2      DCN     STATUS
+3      DCN     DRIVE INDEX
+4      DCN     BUFFER ADDRESS
+5      DCN     DISC ADDRESS OF NEXT SECTOR
+6      DCN     DISC ADDRESS OF THIS SECTOR
+7      DCN     BYTE COUNT
+8      BRA     EOF, DISC ERROR, OR NOT CHAINED FILE.
```

When the file is open, TABLE + 2, 3, 4, and 5 are the request table used by the $DISC routine. TABLE + 5 is the forward pointer in the chained sector format (see "DPUT"). Note that except for the pointers at TABLE + 4 and 5 and the instruction at TABLE + 8, all the information in the table is furnished by MDGET and may be used or ignored by the user program.

Whenever a byte is required, use:

```
BAL     MDGET
 PZE    TABLE ADDRESS
```

The byte will appear left or right justified in RA depending on whether

```
/LIBRARY = CHML.        or
/LIBRARY = CHMR.
```

respectively is included in the load step (see "LOADOV"). Unused bytes will be zero.

MDGET files need not be closed.

MDGET may be used on several files simultaneously. Also, the same file may be read using more than one TABLE, with the effect that a file may be read using multiple pointers.

Restrictions: Each file reader must have its own table. A file may not be read while it is being written using DPUT or MDPUT.

If MDGET is used for reading files written by SINDSK, characters that were repeated on the input will appear compressed. The compression involves three characters: first the compressed character, then ESC (033), then an 8-bit count of the number of times the compressed character is repeated. Software using MDGET may be required to perform decompression. See "SINDSK".

Errors: MDGET will take the error exit if an EOF is encountered, if there is an unrecoverable disc error, or if the pointers in the chained format are in error.

Subroutines Used: $DISC is used to read all sectors. The user must call DRFND himself to get the file pointers.

Register Usage: Only X1, X2, X3 are saved.

Disc Versions: MDGET is furnished in the source files P17-12 (CHML) and P17-13 (CHMR) and in the relocatable libraries CHML and CHMR.

DOS LIBRARY ROUTINE MDPUT

Purpose: To create chained sequential disc files a byte at a time. Multiple files may be created during a single time frame.

Requirements: The 8231 Disc Unit, 0264 (CHML) or 0266 (CHMR) words of memory, a 256-word user-supplied disc buffer, a 10-word file table to be used by MDPUT, and library file CHML or CHMR where MDPUT resides.

Usage: Before any bytes may be output, the file must be opened using:

```
BAL     MDPUTI
 PZE    DRIVE INDEX
 PZE    BUFFER ADDRESS
 PZE    TABLE ADDRESS
```

BUFFER ADDRESS is the address of a 256-word buffer which MDPUT may use to build sectors. MDPUT will locate the file on the disc as close to a user supplied sector address as possible, as described in ALLOC1. TABLE ADDRESS is the address of a 10-word user supplied table (prefixed by FORCE 0), which MDPUT uses for file pointers and parameters. The user must supply two items: 1) an instruction in TABLE + 8, which will be used in case of an error condition; 2) the starting sector address in TABLE + 9. Table format:

```
        FORCE   0
TABLE   DCN     SR0T
+1      DCN     Buffer pointer for SCR instruction
+2      DCN     STATUS
+3      DCN     DRIVE INDEX
+4      DCN     BUFFER ADDRESS
+5      DCN     DISC ADDRESS of sector in Memory
+6      DCN     DISC ADDRESS of Previous Sector
+7      DCN     BYTE COUNT
+8      BRA     DISC ERROR OR DISC FULL
+9      DCN     SECTOR COUNT in upper 12 bits AND STARTING SECTOR
                ADDRESS in lower 12 bits
```

When the file is open, TABLE + 2, 3, 4, and 5 are the request table used by the $DISC routine. TABLE + 6 is the backwards pointer in the chained sector format (see "DPUT"). Note that except for TABLE + 8 and the starting sector address in TABLE + 9 all the information in the table is furnished by MDPUT and may be used or ignored by the user program. To output a byte from RA, use:

```
BAL     MDPUT
 PZE    TABLE ADDRESS
```

The byte will be taken from the left or right byte of RA depending on whether

```
/LIBRARY = CHML.          or
/LIBRARY = CHMR.
```

respectively is included in the load step (see "LOADOV"). Unused bytes are ignored.

After all bytes are output, the MDPUT file must be closed. To close use:

```
BAL     MDPUTF
 PZE    TABLE ADDRESS
```

On completion of MDPUTF, the locations TABLE + 4 and TABLE + 5 will contain the VALUE double word contains the pointers to the file. The pointers will be in the format used in the directory (see "DIRDMP") with the flag byte and protect bit both zero. If the file is to be reused, the user will enter it into the directory using DRUPD. For the format of chained files, see illustration "Format of Chained Files on Disc" under "DPUT".

Restrictions: An output file may not be used as input until it has been closed using MDPUTF and entered into the directory using DRUPD. A file once closed may not be reopened by MDPUTI. Each open output file must have its own TABLE.

Errors: MDPUT will take the error exit (TABLE + 8) on the following conditions:

- No available sectors on the specified drive. Any program using MDPUT must be prepared for this condition and must return sectors from a partially unfilled file to the availability table for future use or these sectors will be lost even if JOB is run; however, use of BOJ will recover these sectors.
- Disc read error in attempting to read previous sector.
- Disc write error in attempting to rewrite previous sector.

Subroutines Used: $DISC, ALLOC1, ALLOCF.

Register Usage: Only X1, X2, X3 are saved.

Disc Versions: MDPUT is furnished in source for as part of files P17-12 (CHML) and P17-13 (CHMR) and in relocatable form in libraries CHML and CHMR.

DOS PROCESSOR MKSYS

Purpose: To produce all the system relocatable and absolute files for transfer to other cartridges by SYSGEN.

Requirements: The 8231 Disc Unit, 6K words of memory, the assembler and loader, and the DOS MONITR.

Usage: With a system master cartridge loaded on drive zero and all applicable APAR's performed, enter the following control statement into SYSIN:

```
// MKSYS
```

Method: MKSYS employs the four control files: MKPROC (construct the system processors), MKASM (construct the assembler), MKLDR (construct the relocatable loader) and MKMON (construct MONITR). Each of these files calls the assembler and assembles its source files into relocatable files. Then the relocatable loader is called to process the relocatable files (including those for MONITR, the assembler, and the loader) into absolute files. The system library files are kept in relocatable form. These files are then ready for processing by SYSGEN.

Subroutines Used: See "MONITR". Also ASM, DIRMOD, LOADOV.

Disc Versions: MKSYS is furnished as a control file.

DOS PROCESSOR MONITR

Purpose: MONITR is the disc resident batch monitor for the disc operating system.

Requirements: The 8231 Disc Unit, 4K words of memory for MONITR and processors (MONITR resides in a maximum of 03400 locations), one display/keyboard, a SYSOUT printer, and the 8001 Card Reader (optional, alternate SYSIN device).

Usage: To originally start the system, bootstrap the disc. "// SYSTEM IV/70 DOS B8" will be printed. The first display/keyboard terminal will be the system input device, and a nonblinking cursor will be displayed. To make the card reader the system input device, type // CARDS on the keyboard.

To use MONITR routines in a processor, do the following:

a. Use a relocatable main program, with the start address the first address of the program.

b. Let the MONITR routine references be virtuals.

c. In the load step, use:

```
// LOADOV
/INPUT = SYSTEM.
/INPUT = name.
/OUTPUT = processor.
//
```

Special Features: No part of MONITR need be in memory with an application program. The routine EXIT is supplied for linking back to MONITR. See "Fixed DOS Cartridge Allocations" for the disc location of MONITR. Processors which reside above MONITR may use the MONITR routines, all of which are left in memory. In particular, the routines SYSIN, SYSOUT, SYSJCT, OPTION, $DISC, DRFND, DRMOD, DRUPD, ALLOC, ALLOC1, ALLOCF and the character tables are designed to be used in this fashion if desired. Memory 0140 — 0177 is used for the system working buffer, SBUF, so that it appears on the first display unit, regardless of format options. Whenever MONITR loads a file into memory, it places the first sector address of the file into X3.

Method: OPTION reads SYSIN, listing on SYSOUT. When a statement with "// " (slash, slash, blank) in bytes 1-3 is encountered, the name in bytes 4-9 is looked up in the disc directory. If the name is that of a memory load, the memory load is loaded and executed.† If the name is that of a source file, that file is used as SYSIN until an end-of-file is encountered.

The first 64 words of sector 0 contain numerous system variables and constants and the pointers into SYSTCK, the stack of processors awaiting execution. See Appendix C for important variables in sector 0. When MONITR exits to a processor, these values are saved for future use. The routine SV0 performs this operation. System users wishing to change the system variables and place the new values into sector 0 on the disc may do so by executing BRM SV0 in their code. SV0 is furnished as part of MONITR.

† On systems with 96K bytes of memory, it is possible for files longer than all of memory to wrap around and wipe out MONITR. Therefore, the file length is checked, and shortened if greater than 32K words.

See the attached flow chart for more detail.

Restrictions: The maximum level of disc control procedures using disc control procedures is 10.

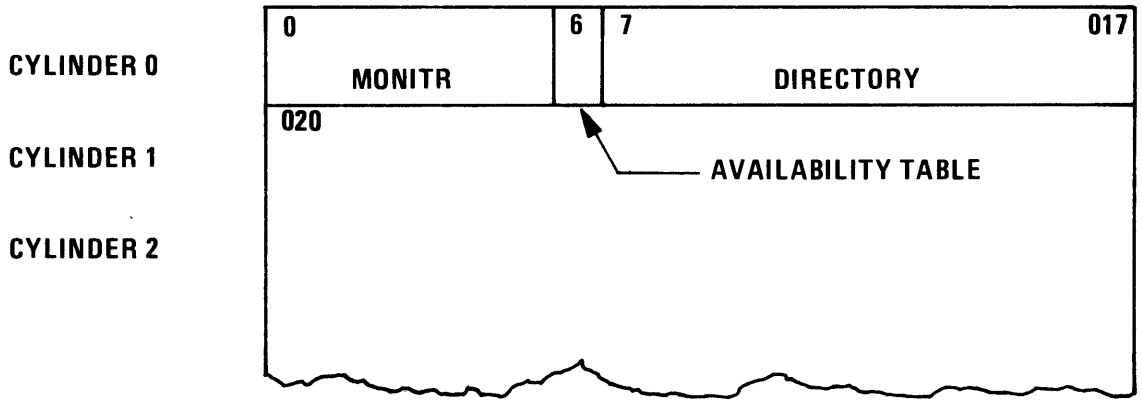
Messages: MONITR can print the following warnings:

1. **name IS RELOCATABLE!
2. **name OVERFLOWS SYSIN!
3. **name NOT IN PACK 0!
4. **name IS DATA FILE!
5. **name OVERLAPS MONITR! (1< starting address ≤ 0410)
6. **name OVERFLOWS MEMORY BUT WILL BE LOADED!

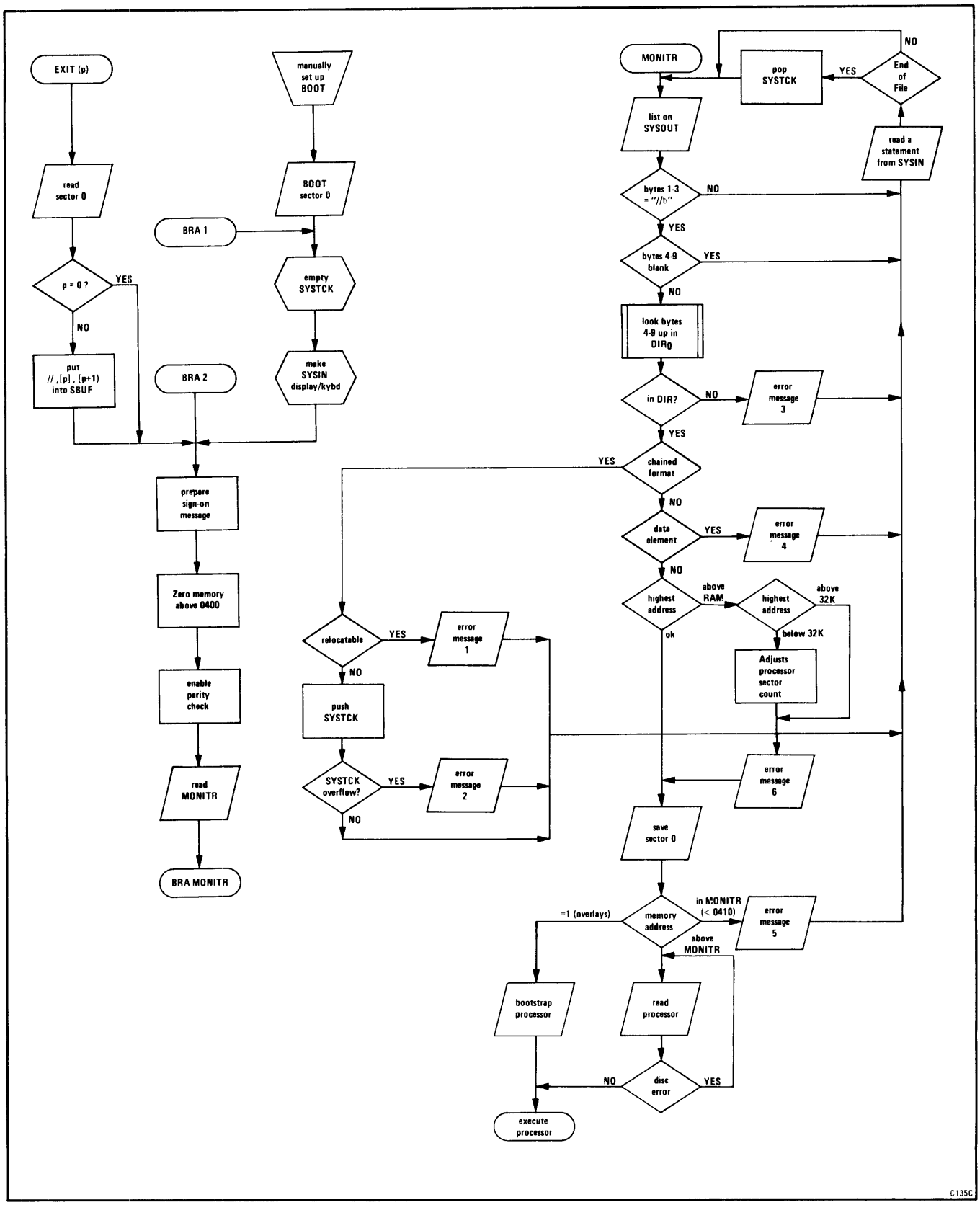
Where "name" is the name from the last // statement. Except in case 6, MONITR ignores the control statement. Note that data cards are not considered an error, so that MONITR can list a deck or disc file containing no control cards, without any special listing routines. See SYSIN for a description of the display/keyboard operating procedures.

Disc Versions: MONITR is furnished as source files P17-2B (monitor control), P17-1B (bootstrap sector), P17-3 (SYSIN), P17-4 (SYSOUT), P17-6 (OPTION), and P17-7B (the allocator and directory manipulation routines). It is also furnished as absolute program MONITR, which is only called by EXIT or the bootstrap instruction.

Fixed DOS Cartridge Allocations:



Flow Chart:


EXIT (p)
read sector 0
p = 0?
YES
NO
put //,[p],[p+1) into SBUF
manually set up BOOT
BOOT sector 0
BRA 1
empty SYSTCK
make SYSIN display/kybd
BRA 2
prepare sign-on message
Zero memory above 0400
enable parity check
read MONITR
BRA MONITR
MONITR
pop SYSTCK
YES
End of File
NO
list on SYSOUT
read a statement from SYSIN
bytes 1-3 = "//b"
NO
YES
bytes 4-9 blank
YES
NO
look bytes 4-9 up in DIR0
in DIR?
NO
error message 3
YES
chained format
YES
NO
data element
YES
error message 4
NO
highest address
above RAM
highest address
above 32K
ok
below 32K
Adjusts processor sector count
relocatable
YES
error message 1
NO
push SYSTCK
SYSTCK overflow?
YES
error message 2
NO
error message 6
save sector 0
memory address
=1 (overlays)
in MONITR (< 0410)
error message 5
above MONITR
bootstrap processor
read processor
disc error
NO
YES
execute processor
C135C

DOS PROCESSOR OCTAL

Purpose: To load programs from octal decks.

Requirements: The 8231 Disc Unit, 0200 words of memory, and an 8001 Card Reader.

Usage: Enter into SYSIN the following control statement:

```
// OCTAL
```

The octal loader will be loaded into the top of memory and control will be transferred to it. To proceed, load the cards into the card reader and start it. If it is desired to load this program into another part of memory, use DIRMOD and the L parameter to change the load address.

Special Features: The program zeros out the contents of memory and enables the checking of memory parity.

Disc Versions: OCTAL is furnished as source file P40 and as absolute program OCTAL.

DOS MONITR ROUTINE OPTION

Purpose: OPTION reads and lists parameter setting records for various processors.

Requirements: The 8231 Disc Unit, 0212 words of memory, and the DOS MONITR. If OPTION is not used with the DOS MONITR, it cannot be used for processes requiring DOS language statements; i.e., for processes requiring access to the allocator or disc directory.

Usage: Calling sequence:

```
BAL     OPTION
 DCN    N
 DCN    LTRS
 DCN    VALS
 DCN    DRV
 BRA    control card terminated input
        data card terminated input
```

Where LTRS is the location of the first word of an array of N letters, each left adjusted with zero fill, and VALS is the location of the first word of an array of N double words to receive the values of their corresponding letters. VALS should be on an even boundary. DRV is the location of the first word of an array of N words to receive the drive index of their corresponding letters.

A recommended method for using OPTION is to leave MONITR resident. This is accomplished by origining the user program at or above 03400 and by placing /INPUT = SYSTEM first in the parameter string at load time.

Method: OPTION first zeros locations VALS and DRV. OPTION then reads SYSIN. If bytes 1-3 are "//b", then OPTION takes the control card exit.† If byte 1 is not "/", then OPTION takes the data card exit.† Otherwise, OPTION scans the card, ignoring blanks. The scan is terminated by a period or the end of the record. A comma ends the current parameter field, and restarts the scan.

Most parameter fields are of the form "keyword = value @ drive". The first letter is looked up in LTRS. If the letter is not in LTRS, the field is skipped. If it is in LTRS, the value is placed in the double word of VALS corresponding to the LTRS entry. If the first character of the value is not numeric, the value is stored as characters, left justified with blank fill. If the first character of the value is numeric, the value is stored as a binary number in the second word, with the first word set to all blanks. If the first digit is zero, the number is treated as an octal number; otherwise, as a decimal number. The @ option does not apply with numeric "value". The first nonnumeric and nonblank character stops the numeric conversion. Note that a parameter can be constructed without "=" or "value"; this allows for construction of binary options, such as the "P" and "Q" parameters under DIRMOD. The contents of VALS will be set to all blanks if this option is used. If an "@" is encountered, the next character encountered is converted to binary, if numeric, and stored in DRV. The rest of the field is skipped. The illustration "Option Parameter Inputs and Outputs" summarizes the actions taken with various parameter combinations.

† OPTION can be fetched in two ways: from MONITR (I = SYSTEM at load time) or by default from SYSLIB. If the MONITR version is used, OPTION executes BRM SV0 to save system variables everytime the control card or data card exit is taken. The SYSLIB version does not call SV0. See "MONITR".

Option Parameter Inputs and Outputs:

Inputs To Option				Outputs From Option		
keyword	=	value	@drive	[VALS]	[VALS+1]	[DRV]
b	b	b	b	0	0	0
P	b	b	b	b	b	0
P	=	b	b	b	b	0
P	=	b	drive no.	b	b	drive no.
P	=	number	b	b	number	0
P	=	number	drive no.	b	number	0
P	=	alpha	b	alpha	alpha	0
P	=	alpha	drive no.	alpha	alpha	drive no.
b = blank or word of blanks; P = parameter; alpha = alphabetic.						

When the record is completely processed, it is listed on SYSOUT, the next record is read from SYSIN, and processing continues.

Restrictions: The keywords for parameters must be distinguishable by their first letters. Note that the scan goes to column 80. Thus if a control file is being run from a sequenced disc file, the sequence numbers are read by option, unless there is a ".". Thus "/DRIVE = 0 001120" is drive number 1120. Therefore, the use of periods at the ends of option parameters is encouraged.

Messages: The following are sample OPTION input records.

```
/INPUT = FILE1@2.
/I = FILE1, O = FILE2. THE REST OF THE CARD IS IGNORED
/ABSOLUTE.
```

OPTION reads the following records as equivalent:

```
/INPUT = S1.
/I  NP  U T = S 1   .
/, INPUT =  S1.
/I = S1
/I = S2, I = S1, .I = S3
```

Subroutines Used: See "MONITR".

Register Usage: None saved. OPTION assumes that SYSOUT saves X3.

Disc Versions: OPTION is furnished two forms: part of MONITR and relocatable. As part of MONITR it is furnished as the source file P17-6 and in absolute form. As an independent relocatable file it is furnished in source form as part of P17-9 and in relocatable form as part of SYSLIB.

DOS MONITR PROCESSOR SINDSK

Purpose: SINDSK copies the system input stream (SYSIN) to the disc for the assembler and to make procedure files.

Requirements: The 8231 Disc Unit, 0670 words of memory, and the DOS MONITR.

Usage: Enter the control statements:

```
// SINSDK
/OUTPUT = name @ drive.          (required)
/PROTECT.                        (optional)
/FLAG = x.                       (optional)
/END = **.
//                               (if END is used)
```

Followed by the records to be copied, followed by some // statement into SYSIN. The output will be catalogued under "name" on drive "drive", will have a flag of "X" and will be protected if /PROTECT was specified. SYSIN furnishes a line feed (012 ASCII) at the end of each record; this is also the system standard record marker.

If the records to be copied contain DOS control statements, the /END option should be used. The three characters after /END = will be used to end the records. /END = ** is suggested as a standard.

Method: SINDSK compresses data before writing it to the disc. If any character is repeated more than four times, it will be compressed. The sequence of compressed data is:

- The character that was repeated on the input.
- Escape character (033).
- Count of times the character was repeated (8 bit binary). The maximum number of characters that can be expressed in this manner is 256. Note that DPUT and MDPUT do not perform this compression and DGET and MDGET do not decompress data automatically.

Messages: SINDSK can print the following error messages:

```
** DIRECTORY - DISC ERROR
** DIRECTORY - FULL
** XXXXXX PROTECTED
** OUT OF SPACE ON DISC OR /O FILE DISC ERROR
** /O FILE IS EITHER MISSING OR INVALID
```

Detection of any of these errors causes SINDSK to terminate and transfer control to MONITR.

Subroutines Used: See "MONITR".

Disc Versions: SINDSK is furnished as source file P17-BC and as absolute program SINDSK.

DOS MONITR PROCESSOR SNEDIT

Purpose: To edit source or date files on the disc. Capability is provided to delete or insert records and to borrow records from one file and insert them into another.

Requirements: The 8231 Disc Unit, 4K words of memory, the DOS MONITR, and a SYSOUT printer.

Usage: Enter into SYSIN the following control statements:

```
// SNEDIT
/INPUT = file1.
/OUTPUT = file2.
/PROTECT.                    (optional)
/FLAG = x.                   (optional)
/END = &&.                   (optional)
/ALTER = $.                  (optional)
/RENUMBER.                   (optional)
/DELTA = N.                  (optional)
```

Followed by card files or other records to be edited as described below, followed by //ƀ. Input will be taken from file 1 and output placed on file 2; if the file names are the same the old file will be destroyed,† but if they are different, the old file will be preserved. If the /P parameter is used, the protect bit will be set for the new file, regardless of whether it was set for the old file; if the protect bit was set for the old file, it will be set for the new file also. Processor DIRMOD should be used for deleting protection. The /F parameter allows the user to specify a flag character in the newly created directory entry; see "DIRDMP". If no /F parameter is given, no flag will be written. The /E parameter is used to establish a new end code so that //ƀ and / control statements can be embedded in control strings. The /A parameter is used for intermediate processing also; the value of the A parameter replaces the # used to indicate a record number. && is the suggested convention for E and $ for A. The first time a file is processed by SNEDIT, the user must specify the /R parameter so that the file will be numbered using columns 75-80 of each record; on subsequent editing passes it will be renumbered or not depending on whether the /R parameter is specified. The standard (default) increment for numbering the files is an increment of ten, but the /D parameter is provided to allow any increment up to six digits in length.

To edit a protected file, copy the file to a TEMP file name using SNEDIT and performing the editing at the same time, then remove the protection from the old file using the Q parameter under DIRMOD. Next, create a new protected file with the TEMP file and the old protected name using SNEDIT with the P parameter selected, and the old file will be returned to the disc pool; the TEMP file will be returned when JOB is executed.

† However, if the file names are the same and the old file is protected, SNEDIT will not operate and an error message will be generated.

The following examples illustrate various applications of SNEDIT. The editor operates on numbered records. In the examples, TFn is file number n, CARDn is card image (i.e., record) number n, and & is the same as +. All the files can be restricted as to drive by using @ DRIVEn parameters.

Example 1. To number the records on a file, enter the following control records into SYSIN:

```
// SNEDIT
/INPUT = TF1, OUTPUT = TF2, RENUMBER, DELTA = N.
//
// TF2
```

This causes the card-length records on TF1 to be numbered with an increment of N and transferred to TF2, then prints TF2.

Example 2. To insert new records into a file whose records are numbered, specify the record *in front of which* the new records are to go using record numbers prefixed by # in the following manner:

```
// SNEDIT
/INPUT = TF2, OUTPUT = TF3, RENUMBER.
#1
CARD A ....
CARD B ....
//
// TF3
```

This causes CARD A and CARD B to be inserted in front of the record numbered 1. The new file is then renumbered and transferred to TF3 and TF3 is printed. If the RENUMBER parameter is specified, TF3 will be renumbered. If not, the records will retain the numbering given to TF2; if more files are added than the original numbering of TF2 can absorb, the old higher-numbered files will be renumbered to the minimum degree required using the new DELTA.

Example 3. To delete records, give the record numbers in pairs, separated by a comma. (If a single number instead of a pair is given, the record will not be deleted.) The numbers must be in columns 1-72. The records will be deleted *inclusively* and renumbered:

```
// SNEDIT
/INPUT = TF2, OUTPUT = TF3.
#4, 6
#9,    9
//
```

This causes the records numbered 4, 5, 6, and 9 to be deleted.

Example 4. To merge two numbered files (TF2 into TF4), it is necessary to use END and ALTER parameters to create an edited intermediate command file. Then when the intermediate file is executed, it will be merged with the second input file automatically. The following records are used:

```
// SNEDIT
/INPUT = TF2, OUTPUT = TF5, END = &&, ALTER = $.
$ 1
// SNEDIT
/INPUT = TF4, OUTPUT = TF6.
#N
$ R
//
&&
// TF5
//
// TF6
//
```

This establishes a new end code (&&) so that the //b and / control statements will be made part of the intermediate file (TF5) rather than executed immediately. The $ alter code is also established to allow # statements to be made part of TF5 rather than executed immediately. This allows the user to define the limits of the scan within the file to be inserted (TF2), so that this file may be edited. $1 forces the first record of TF2 to be taken (for example, if $4 were specified, the first three records of TF2 would be discarded before insertion). The insertion will occur just before record number N of TF4; if N is two numbers (for example, #4, 6 as in Example 3) the records thus numbered in TF4 will be deleted and TF2 will be put in their place. The largest number card that will be taken from TF2 is R-1. (R may be chosen to be larger than the highest numbered record to be merged, or it may be adjusted to discard records from the end of the insert file.)

First the file TF5 is created, containing the control file between $1 and $R, and records 1 through R-1 of file TF2. Next TF5 is executed, thereby executing the control file, causing TF2 to be inserted before record N in TF4 and creating TF6. Last, TF6 is printed.

Messages: SNEDIT can print the following error messages:

```
** /I FILE NOT CHAINED
** /I FILE LINKAGE ERROR
** /I FILE DISC ERROR
** DIRECTORY - DISC ERROR
** DIRECTORY FULL
** XXXXXX PROTECTED  (XXXXXX is file name)
** DELTA MUST BE GREATER THAN ZERO
** OUT OF SPACE ON DISC OR /O FILE DISC ERROR
** /I FILE NOT IN DIRECTORY
** SERIAL COUNT OVERFLOW
** /O FILE IS EITHER MISSING OR INVALID
** CARD ABOVE IN ERROR
```

Subroutines Used: See "MONITR". Also DGET.

Disc Versions: SNEDIT is furnished as source file P17-BC and as absolute program SNEDIT.

DOS PROCESSOR SORT

Purpose: To sort DOS SINDSK files.

Requirements: One or more 8231 Disc Units, 8K words of memory, and a SYSOUT printer.

Usage: Enter the control statements:

```
// SORT
/INPUT = name1 @ drive1.                    (required)
/OUTPUT = name2 @ drive2.                   (required)
/FIRST = first column of key.               (optional, default 1)
/LAST = last column of key.                 (required)
/MAX = max record size.                     (required)
/COM = name3 @ drive3.                      (optional)
/WORK = no. of sectors in each work file.   (optional)
```

Into SYSIN. The chained file "name$_1$" will be sorted into ascending ASCII collating sequence and written to "name$_2$". The F and L parameters specify the number and location of the continuous string of bytes within the record to be used as the sort key. The /M parameter specifies the length (in bytes) of the largest record in the file. The /C option allows the user to provide a key comparison routine. "name$_3$" must be a memory load file containing a comparison routine that has been loaded uisng the /P = LO option. The comparison routine must obey the following conventions:

- A pointer to the entry point is contained in word 1 of the module. Word 2 is zero or points to an optional last routine.
- Upon entry, X1 and X3 point to the two records to be sequenced. The records begin at 1,X1 and 1,X3 respectively. Word 0 of each record contains a record length (words) in the low-order 16 bits. The remaining words of the record contain the data.
- If the record pointed to by X1 is greater than (sorts behind) the other record, return should be made to 0,X2. Otherwise return should be made to 1,X2.
- Index registers should be restored prior to return.
- The COM routine is entered via BAL, not BRM. Therefore, the first location of the routine must be executable.

If the /W option is not supplied, SORT will assume twice the number of sectors in the original input file. If the sort does not work, and the message "FILE LIMITS EXCEEDED" is printed, try again with W twice this size.

If a user last routine is not given, the sorted records will merely be written to the output file. If the user last routine is given, it must obey the following conventions:

- On entry to the user last routine, X1 points to the first location of the record under consideration. The user may manipulate the record in any way desired, but should restore the pointer in X1 before exiting. The length of the record may not be changed.

- The last routine may determine that the record is or is not to be included in the output file. If the record is not to be included, the return should be made to 0,X2. If the record is to be included, the return should be made to 1,X2.
- The last routine must restore all index registers.
- The last routine is called via BAL, not BRM. Therefore the first location of the routine must be executable.

Special Features: SORT will attempt to use all disc drives that are in ready status. SORT work files are optionally allocated among available drives. This means that all disc cartridges must be DOS compatible; i.e., must contain an allocation table in sector 6.

Restrictions: "name1" and "name2" must be chained files. Each record in "name1" must be terminated by a line feed (ASCII 012). If the records were read in by a system routine (e.g., SINDSK) the line feeds will be furnished. SINDSK type character compression may appear in "name1" and will be used in writing "name2".

Messages: SORT prints a self-explanatory fatal error message in the event of an unrecoverable error.

Processing Method: SORT has two phases:

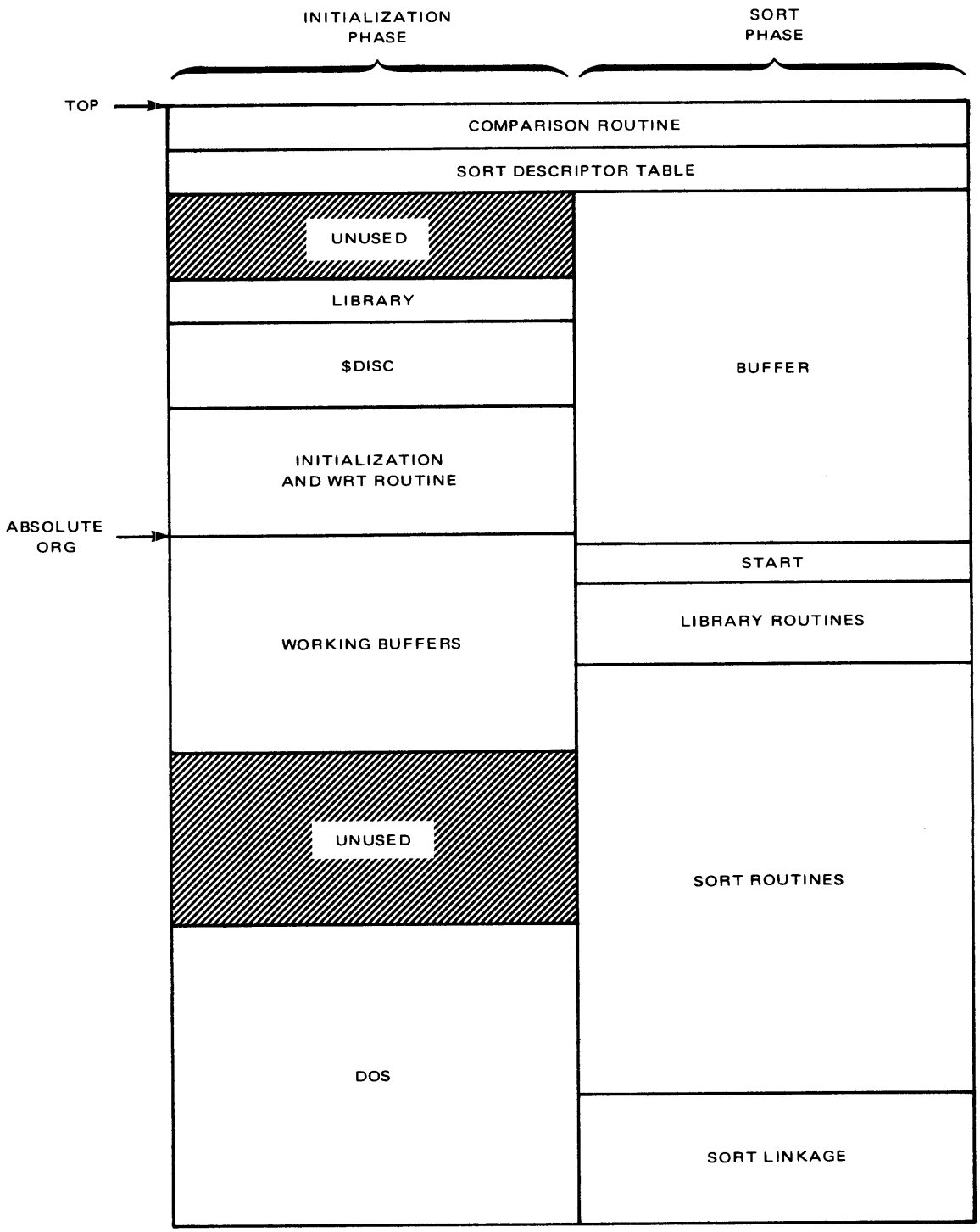
- Initialization Phase. The control cards are read by OPTION and the information is stored in the Sort Descriptor Table. The input file is opened and copied to sort work file number 4. Device allocation for work files is performed by checking all ready drives for available space. This phase is ORG'ed in such a fashion that it can read in the next phase by using a private copy of $DISC.
- Sort Phase. The input file on file 4 is sorted, and the final merge pass output is written to the output file.

Sort Descriptor Table:

Memory size
LPOUT contents
Name of next item in control stream (SBUF+1, +2)
Address of comparison routine
Maximum record size
Device no. for file 1
Device no. for file 2
Device no. for file 3/4
Output file name and drive
Input file name and drive
Number of sectors in each work file
Start disk address of file 4

Disc Versions: SORT is available in source form as P44BB3, P60-1x through P60-9x, P6011x (where x is the current revision letter) and CHLLIS (a private interrupt driven version of CHLL) and in executable form as the absolute programs SORT and SORTPH.

Sort Memory Organization:



DOS PROCESSOR SYSGEN, SYSGN2

Purpose: To transfer the system files (created by MKSYS) to new cartridges.

Requirements: Two 8231 Disc Units, 6K words of memory, the relocatable loader, and the DOS MONITR.

Usage: With a master cartridge, on which MKSYS has been executed, loaded on drive 0 and a properly formatted cartridge on drive 1, enter the following control statement into SYSIN:

```
// SYSGEN
```

The system source and object files on drive 1 are deleted. Then the system object files created by MKSYS are copied onto drive 1.† The program then halts, displaying the message // PAUSE CLEAR HALT TO COPY SOURCE, ELSE BOOTSTRAP. If the halt is cleared (AUTO/MANUAL down, then up) the system source files will be copied. Otherwise, the system may be bootstrapped so that no source files will appear on the cartridge on drive 1.

Subroutines Used: See "MONITR". Also LOADOV, COPY, DIRMOD.

Disc Versions: SYSGEN is furnished as a control file.

† Note that the processor SYSGN2 is also furnished for transferring the files from drive 0 to drive 2.

DOS MONITR ROUTINE SYSIN

Purpose: Obtain a record from the current system input device.

Requirements: The 8231 Disc Unit and 0254 words of memory if DOS MONITR is not used. SYSIN can function under DOS MONITR or as a relocatable file under SYSLIB; in either case the hardware requirements are the same as those for MONITR.

Usage: Calling sequence:

```
BAL     SYSIN
 PZE    SBUF
```

SBUF is the address of the 27 word buffer to be used. If the keyboard is a possible input device for the application, the standard buffer of 0140 — 0177 should be used. The input record is returned to SBUF. The first 80 characters are in the 64 or 96 character ASCII subset. The 81st character is 012, the ASCII LF (or NL — new line) symbol.† When a routine is done with SYSIN, the routine should write memory 0001 — 0100 onto sector 0 to record the current input device and position. BRM SV0 will accomplish this.

The display/keyboard input has a simple editor. The key "delete" removes the character to the left of the cursor. The key "erase" blanks the current line. Either the key "cursor return" or the key "EOM" will enter the current line into the system input stream. Other controls are ignored. Keys are displayed as entered, with a conversion from lower to upper case if [LUCONV] ≠ 0. The cursor is a steady solid frame, and is a destructive cursor.

If SYSIN is taken from SYSLIB, the flags LUCONV and CINPUT are left as virtuals. If not defined by the user program, they will be taken from SYSLIB with default values ≠ 0.

Method: SYSIN has a base device which is switchable between the card reader and the first display/keyboard terminal. If [CINPUT] ≠ 0, then the base system input device is the card reader; otherwise, it is the first display/keyboard terminal. SYSTCK is a stack of disc procedure files. If the stack is not empty, the disc is used as the system input device; if it is empty, the base device is used. If a disc file is the current SYSIN, then the sector, word, and byte address are decoded from [[SYSTCK]] as follows: the sector address is in bits 12-23, the byte address is in bits 10-11, and the word address is in bits 2-9.

SYSIN uses the chained sequential access method. Each record is of variable length, ended with the ASCII character LF. The ASCII character ESC is used to compress duplicate characters. The 3 character sequence "character, ESC, binary count" represents (count + 1) occurrences of "character". All other ASCII controls are ignored. Lower case letters are converted to upper case if [LUCONV] ≠ 0. Characters above 0177 are ignored.

Cards are read using the controller's packed ASCII mode. Parity bits are removed, and a LF is inserted in position 81.

† Furnished by SYSIN. If a record is longer than 80 characters, SYSIN will truncate it to 80; 80 characters is the system standard source record length.

This program assumes that the constants in the Character Tables (P17-5) are in the order specified in the "System IV/70 Computer Reference Manual" document number SIV/70—11—1. This is the order in which the tables are furnished in SYSLIB. Caution must be exercised in changing these tables for special applications; one method is to use different labels on altered Character Tables.

Restrictions: The version of SYSIN in SYSLIB cannot handle disc procedure files.

Errors and Messages: For any card reader error, replace the card last picked and ready the reader. SYSIN will not read the last card in a deck; if the hopper runs out and the last card should be read, it must be repositioned with the rest of the deck behind it. Usually, the last card should be a blank card.

SYSIN can print the following message:

```
*** DISK ERROR IN PROCEDURE FILE ***
```

Subroutines Used: See "MONITR".

Register Usage: Only X1, X2, X3 are saved.

Disc Versions: SYSIN is furnished in two forms: part of MONITR and relocatable. As part of MONITR it is furnished as the source file P17-3 and in absolute form. As an independent relocatable file it is furnished in source form as part of P17-9 and relocatable as part of SYSLIB.

DOS MONITR ROUTINES SYSOUT, SYSJCT

Purpose: List a record on the current system output device.

Requirements: The 8231 Disc Unit, 0211 words of memory, and the chosen printer. The choice is made at run time. The printers currently supported are the 8131, 8142, 8143, 8145, 8146, and 8151.

Usage: To list a record, use the calling sequence:

```
BAL     SYSOUT
 DCN    BUF
```

Where BUF is the starting location of a string of 6-level ASCII subset characters terminated by any ASCII control. The ASCII controls LF (012) and FF (014) cause a line feed and a form feed respectively. The action on other controls is device dependent. If the string is longer than the maximum for the device, print lines may be lost. The maximums are 80 for the 8143, 120 for the 8142, and 132 for the 8131, 8145, 8146, and 8151. Print lines $\leqslant$ 80 characters in length will always be correctly handled.

To eject the current page, use the calling sequence:

```
BAL     SYSJCT
```

SYSOUT will send a form feed (ASCII 014) to the printer driver routine.

If SYSOUT is taken from SYSLIB, the flag LPOUT is left as a virtual. If not defined by the user program, it will be taken from SYSLIB with the default value 0.

Method: The contents of LPOUT select the printer.

LPOUT = 0 for the 8145
= 1 for the 8143
= 2 for the 8142
= 3 for the 8131
= 4 for no print (NOPRNT)
= 5 for the 8146 and 8151

This program assumes that the constants in the Character Tables (P17-5) are in the order specified in the "System IV/70 Computer Reference Manual" document number SIV/70—11—1. This is the order in which the tables are furnished in SYSLIB. Caution must be exercised in changing these tables for special applications; one method is to use different labels on altered Character Tables.

Register Usage: Only X3 is saved.

Disc Versions: SYSOUT is furnished in two versions: part of MONITR and relocatable. As part of MONITR it is furnished as the source file P17-4 and in absolute form. As an independent relocatable file it is furnished in source form as part of P17-9 and relocatable as part of SYSLIB.

DOS PROCESSOR XREF

Purpose: To cross reference all symbols in an ASM source file.

Requirements: The 8231 Disc Unit, 8K words of memory, a SYSOUT printer, the DOS MONITR, SORT, and SORTPH.

Usage: Enter the following control statements into SYSIN:

```
// XREF
/INPUT = name @ drive.  }          may be repeated as many
blank card              }          times as necessary
//
```

name must be an ASM source file with sequence numbers, as assigned by SNEDIT, in columns 75-80. A blank card must always follow each input card. For each input file, a new output file and listing are created; the temporary name TEMPGR is used for the working files.

Method: XREF creates a file with one record for each occurrence of an identifier, then calls SORT to sort this file and uses the sorted file to print a cross reference table. If the disc or disc directory is full, an error message will be printed and control will return to the SYSIN input stream.

Restrictions: With more than one disc drive in the system, there must be a disc cartridge mounted on drive zero.

Errors: A "D" will be printed in front of the second and any further definitions of a symbol.

Disc Versions: XREF is furnished as source files P681, and P682, and control file C68. The absolute files are XREF and A682. Only XREF is called by the user. Note that P681 uses subroutine INCARD (source file P683).

APPENDIX A

SUMMARY OF DOS FILES

DOS FILENAMES

File	Source	Relocatable	Absolute
The Monitor files are			
Bootstrap sector	P17-1B	R17-1B	Called only by bootstrap instruction
Monitor control	P17-2B	R17-2B	MONITR (called by EXIT, bootstrap, or BRA MONITR)
SYSIN	P17-3	R17-3	Not called individually
SYSOUT	P17-4	R17-4	Not called individually
Character tables	P17-5	R17-5	— — —
OPTION	P17-6	R17-6	Not called individually
Directory & Allocator routines (DRFND, DRMOD, DRUPD, ALLOC, ALLOC1, ALLOCF)	P17-7B	R17-7B	Not called individually
NEWSYS Creator	P17-8	SYSTEM	— — —
System Relocatable Library	P17-9	SYSLIB	— — —
Chained Access Programs	P1710B, P1711B, P17-12, P17-13	CHLL, CHRR, CHML, CHMR	— — — — — —
The DOS processors are			
DASM (processor)	P3-1G, P3-2, P3-3B, P3-4B, P3-5, P3-6, P3-7	R3-1G, R3-2, R3-3B, R3-4B, R3-5, R3-6, R3-7	DASM (called by ASM only)
Relocatable Loader	P12-1B, P12-2B	R12-1B, R12-2B	LDRHI, LDRLO
LOADOV	P12ABW, P12BB, P12CB, P12DB, P12EB	R12ABW, R12BB, R12CB, R12DB, R12EB	LOADOV
SINDSK	P17-BC	R17-BC	SINDSK
SNEDIT	P17-BC	R17-BC	SNEDIT
ASM (preprocessor)	P17-CB	R17-CB	ASM
ASMX (postprocessor)	P17-E	R17-E	ASMX (called by DASM only)
LOAD (preprocessor)	P17-F	R17-F	LOAD
DIRDMP	P17-J	R17-J	DIRDMP
DIRMOD	P17-K	R17-K	DIRMOD
JOB	P17-LB	R17-LB	JOB
BOJ	P17LB	R17LB	BOJ
COPY files	P17-NB	R17-NB	COPY
Minor Processors	P17-P	R17-P	See "Minor Processors"
DIRSRT	P17-R	R17-R	DIRSRT
FILDMP	P17-S	R17-S	FILDMP
LIST	P17-T	R17-T	LIST
OCTAL Loader	P40	R40	OCTAL
CRTDMP	P42	R42	CRTDMP
Copy disc (COPY01)	P51	COPYDD	COPY01
CDDC	P57	R57	CDDC
SORT	P60-1C, P60-2A, P60-3A, P60-4A, P60-5A, P60-6A, P60-7A, P60-8A, P60-9A, P44BB3, P6011A, CHLLIS	R60-1A, R60-2A, R60-3A, R60-4A, R60-5A, R60-6A, R60-7A, R60-8A, R60-9A, R44BB1, R6011A, CHLLI	SORT, SORTPH
XREF	P681, P682, P683	R681, R682, R683	XREF & A682

DOS FILE STRUCTURE

MONITR Processors

ASM
BOJ
CDDC
COPY
DIRDMP
DIRMOD
JOB
LOAD
LOADOV
MONITR
SINDSK
SNEDIT

DOS MONITR Routines

$DISC
ALLOC
ALLOC1
ALLOCF
DRFND
DRMOD
DRUPD
OPTION
SYSIN
SYSJCT
SYSOUT
Character Manipulation Tables

SYSLIB Routines

$DISC
$DUMP
CARDIN
EXIT
OPTION
SYSIN
SYSJCT
SYSOUT
Character Manipulation Tables

System Libraries (System Relocatable Files)

CHLL
CHML
CHMR
CHRR
COBLIB
R17-Q
SYSLIB

Library Routines

$DUMP
$ICARD
$IDISC
$IOPEN, $ICLOS
$IPRNT, $JPRNT
$ITAPE
$JTAPE
DGET
DPUT
EXIT
MDGET
MDPUT

Control Files

C68
DELSYS
MKSYS
SYSGEN, SYSGN2

Processors

COPY01
CRTDRMP
DIRSRT
DTU
DTU16
FILDMP
LIST
OCTAL
SORT
XREF

Minor Processors

CARDS
EJECT
KYBRD
NOPRNT
PAUSE
P8131
P8141
P8143
P8145
P8146

Programs Furnished In Source Form Only

Video Display Library
DCCD

Unsupported Programs Used With DOS

DEMO48 (Processor)
COPYMD (Processor)
GENCTR (Processor)
TRACE (Relocatable)

APPENDIX B

HALTS IN THE RELOCATABLE LOADER AND ASSEMBLER

LOADER HALTS

Whenever the relocatable loader LOAD is used to load programs, a *load map* of resolved virtuals will be printed on SYSOUT. The first item in the load map will be $LOAD, a reference location within the loader. For different loads using the same configuration, the location of $LOAD will not change. The error halts within the loader are all located relative to $LOAD; thus, if a halt occurs within the loader, get the contents of RP and compare them to the known location of $LOAD, then refer to the following table. Note that LOADOV has no halts.

Symbolic Location	Relative to $LOAD†	Reason	Action on Restart (AUTO-MANUAL-AUTO)
NROOM	–062	No room on disc, allocation table full.	Return to MONITR, no load map.
DRFAIL	–060	Directory error: – Output file protected – Disc error – Permanent error – Directory full	Load map created, but no output file.
LERR	–042	Invalid pointer on chained virtuals.	Load map created, but no output file.
NFILE	–040	No file exists for library entry.	Loading stops, load map created, but no output file generated.
OVERFL+4	–020	Invalid location, overflows loader or out of bounds for program, new load map entry overlays program or loader.	Comment given only once; data is lost, but load continues.

† These values are for LDRHI; subtract 1 for LDRLO.

ASSEMBLY HALT

The only condition that will cause the assembler to halt is a symbol table overflow. On restart, the program will proceed.

APPENDIX C

COMMONLY USED LOW MEMORY LOCATIONS IN MONITR

The first 0100 locations of the bootstrap section of MONITR are considered to be frozen and may be accessed by the programmer as absolute locations. These absolute locations are carried over from earlier releases. In addition, SBUF, the system card image buffer, is considered to be frozen at location 0140 because of the strategic location for both 48 and 81 character-per-line video display systems. Locations 0173-0177 were added to the absolute locations at release B06.

Note that user programs that employ SYSIN and/or SYSOUT in environments where more than one SYSIN or SYSOUT device is in use must have reference to certain of these locations. Specifically, CINPUT, LUCONV, and LPOUT furnish information required in multiple input and output situations. If SYSIN or SYSOUT are being used without MONITR, it will be necessary to read sector 0 of the disc if the status of these locations is required.

Commonly used locations are:

Symbolic Location	Absolute Location	Contents/Significance
BT10-1	0001	Bootstrap entry point and restart location. — Resets SYSTCK (system input stack) to empty. — Resets CINPUT (SYSIN input flag) to take input from keyboard. — Zeros out rest of memory. — Reads rest of MONITR from disc. — Enables memory parity checking. — Transfers control to MONITR.
BT10	0002	Same as 0001 except that SYSTCK and CINPUT are not changed.
MEMORY	0033	Highest memory address +1.
CINPUT	0034	SYSIN input flag: 0 = keyboard, nonzero = cards.
LUCONV	0035	Lower case conversion for SYSIN. = 0 No conversion of lower case to upper case. ≠0 Convert lower case to upper case on input.
LPOUT	0036	SYSOUT Printer index: 0 = 8145 1 = 8143 2 = 8142 3 = 8131 4 = No print 5 = 8146 and 8151

Symbolic Location	Absolute Location	Contents/Significance
SYSTCK	0041	SYSIN input stack pointer for disc procedure files.
SBUF	0140	System input buffer used by SYSIN. Eighty characters long with a line feed in the 81st position. When the system is bootstrapped this buffer will contain "// SYSTEM IV/70 DISC OPERATING SYSTEM 88-0017- XX", where XX is the current release code.
SBUF+1	0141	Used with SBUF+2 to obtain name on current control statement.
SBUF+2	0142	Used with SBUF+1 to obtain name on current control statement.
SBUF+033 to 037	0173-0177	System communication area. This area is used by COBOL to communicate between modules. These locations can also be used by user programs that reside with MONITR in memory, but a call to EXIT or rebootstrapping the program will destroy the information.
$HEAD	0200	Routine used for disc I/O to read or write headers only. Calling sequence same as $DISC.

$DISC	0204	Routine used for disc I/O except for reading headers. Calling sequence:
		`BAL $DISC Linkage`
		`PZE REQTAB Address of disc I/O request table`
		`HLT $ Disc error`

APPENDIX D

CHARACTER SETS

In the usual course of events, the DOS language is expressed in the 64-character ASCII as shown in Figure D–1; however the full 96-character set can be used if the cell LUCONV in MONITR is zero. This is particularly useful if the output printer is the 8147 or 8152 Printer (see Figure D–4).

The System IV/70 Computer operates on a character set (see Figure D–2) that contains the other character sets as a subset. The special characters shown in Figure D–2 can be generated by the IV/70 keyboard and displayed on the video screen, but if MONITR is called upon to process these characters, they will probably be ignored. Figures D–3 and D–4 show the character sets for the character printers and line printers.

Third Octal Digit	First & Second Octal Digits															
	00	01	02	03	04	05	06	07	10	11	12	13	14	15	16	17
0	NUL	BS	DLE	CAN	SP	(	0	8	@	H	P	X	`	h	p	x
1	SOH	HT	DC1	EM	!	)	1	9	A	I	Q	Y	a	i	q	y
2	STX	LF	DC2	SUB	"	*	2	:	B	J	R	Z	b	j	r	z
3	ETX	VT	DC3	ESC	#	+	3	;	C	K	S	[	c	k	s	{
4	EOT	FF	DC4	FS	$	,	4	<	D	L	T	\	d	l	t	|
5	ENQ	CR	NAK	GS	%	-	5	=	E	M	U	]	e	m	u	}
6	ACK	SO	SYN	RS	&	.	6	>	F	N	V	^	f	n	v	~
7	BEL	SI	ETB	US	'	/	7	?	G	O	W	_	g	o	w	DEL

64-Character ASCII subset, recognized by printers and generated by the card reader. Note that codes 133 ([), 134 (/), 135 (]), and 136 (^) are displayed as ÷, ×, \|, and ↑ respectively on the 7100/7101 video displays. Also note that codes 133, 134, 135, and 136 are generated by the card reader (with the parity bit set as necessary to produce even parity) from the hole patterns produced by keys ¢, 0-2-8, ¬, and | on the 029 Keypunch.

A108B

Figure D–1. ASCII Code Set

Third Octal Digit	First & Second Octal Digits															
	00	01	02	03	04	05	06	07	10	11	12	13	14	15	16	17
0	•	⌗†	|	□	SP	(	0	8	@	H	P	X	`	h	p	x
1	△†	←	[	↘ †	!	)	1	9	A	I	Q	Y	a	i	q	y
2	♭†	\	\	■	"	*	2	:	B	J	R	Z	b	j	r	z
3	¢	/	—	°	#	+	3	;	C	K	S	÷	c	k	s	{
4	◢	£	√†	◄	$	,	4	<	D	L	T	×	d	l	t	¦
5	♯†	■	–	▌	%	-	5	=	E	M	U	|	e	m	u	}
6	♯†	¬	^	►	&	.	6	>	F	N	V	↑	f	n	v	~
7	▫†	|	]	\	'	/	7	?	G	O	W	_	g	o	w	///

†These symbols are currently displayed but not supported. Other symbols may be substituted on later models.

A107B

Figure D—2. IV/70 Display Characters

Third Octal Digit	First & Second Octal Digits															
	00	01	02	03	04	05	06	07	10	11	12	13	14	15	16	17
0	NUL				SP	(	0	8	@	H	P	X	`	h	p	x
1					!	)	1	9	A	I	Q	Y	a	i	q	y
2		LF			"	*	2	:	B	J	R	Z	b	j	r	z
3				ESC	#	+	3	;	C	K	S	[	c	k	s	{
4		FF			$	,	4	<	D	L	T	\	d	l	t	¦
5		CR			%	-	5	=	E	M	U	]	e	m	u	}
6					&	.	6	>	F	N	V	^	f	n	v	~
7	BEL				'	/	7	?	G	O	W	_	g	o	w	

NUL, BEL, LF, and CR are recognized by Teletype Printers (8100 Controller). LF, FF, CR and DEL are recognized by the 8131 Printer. DEL (0377) is used as the null character with this printer.

64-Character ASCII subset, recognized by all character printers. Note that codes 133 ([), 134 (\), 135 (]), and 136 (^) are displayed as ÷, ×, |, and ↑ respectively on the 7100/7101 video displays.

31 additional characters that may be recognized by some printers.

A087D

Figure D—3. Character Printer Code Set

Third Octal Digit	First & Second Octal Digits															
	00	01	02	03	04	05	06	07	10	11	12	13	14	15	16	17
0					SP	(	0	8	@	H	P	X	`	h	p	x
1					!	)	1	9	A	I	Q	Y	a	i	q	y
2					"	*	2	:	B	J	R	Z	b	j	r	z
3					#	+	3	;	C	K	S	[	c	k	s	{
4					$	,	4	<	D	L	T	\	d	l	t	|
5					%	-	5	=	E	M	U	]	e	m	u	}
6					&	.	6	>	F	N	V	↑	f	n	v	¬
7					'	/	7	?	G	O	W	←	g	o	w	□

Control and form feed characters. See text for details

64-Character ASCII subset, recognized by all line printers

32 additional codes recognized by 8147 and 8152 printers

A230B

Figure D—4. Line Printer Code Set

GLOSSARY

Automatic Deletion Rule. The rule that files are deleted whenever a new file with the same name is created. The system works in this mode, except that files may be protected from this rule by a /PROTECT function.

Chained File. A file whose sectors may be scattered throughout the disc. Each sector has links in each direction, so that the sectors may be accessed sequentially, either forwards or backwards.

Closed. A file is closed when all the information needed to access the file by name is on the disc. DRUPD performs the function of closing files.

Contiguous File. A file all of whose sectors are contiguous in the disc sector address space. The sectors do not contain pointers and are organized only in terms of their sequence on the disc.

Control File. A *chained file* containing control statements, separated by CR. Also called a *procedure file.*

Data File. Any *contiguous file*, especially those so marked with a load address of 0.

Disc Pool. The set of sectors immediately available for use in creating new files. Sectors from deleted files are put into the disc pool by // JOB. Lost sectors are reclaimed by // BOJ.

File. A set of sectors on one disc, whose name and defining parameters are in the disc directory. Files may be either contiguous or chained.

Memory Load. A *contiguous file* which contains a program in absolute binary. Also called a load module.

Open. A file is open when all the information needed to access data sectors is in memory. DRFND performs the function of opening files.

Relocatable File. A *chained file* containing CODE Assembler relocatable output. These files are created by // ASM or by // COBOL, named with the /RELOC or /OUTPUT parameter.

Source File. A *chained file* containing CODE Assembler or COBOL source statements or DOS Language control statements, separated by CR.

USER'S COMMENTS

System IV/70 Disc Operating System (DOS)
Reference Manual
SIV/70—50—1C

Your comments will be considered for improving future documentation. Please give specific page and line references if appropriate.

- Which of the following best describes your occupation:

  ☐ Programmer
  ☐ Systems Analyst/Designer
  ☐ Engineer
  ☐ Operator
  ☐ Instructor
  ☐ Student
  ☐ Manager
  ☐ Customer Engineer
  ☐ Other _____

- In what ways do you use this document?

  ☐ Reference Manual
  ☐ In a class
  ☐ Self Study
  ☐ Introduction to the Subject
  ☐ Introduction to this System
  ☐ Other _____

- Comments/Criticisms

Thank you for your assistance. No postage required if mailed in the USA.

SIV/70—50—1C

fold fold

FIRST CLASS
Permit No. 194
Cupertino,
California

BUSINESS REPLY MAIL
No Postage Necessary if Mailed in the United States

Postage will be Paid by . . .

FOUR-PHASE SYSTEMS
10420 North Tantau Ave.
Cupertino, Calif. 95014

Attention: Technical Publications

fold fold

Cut Along Here

Staple Here

SIV/70—50—1C

System IV/70

Disc Operating System (DOS)

Reference Manual

SIV/7[illegible]50-1C

10420 North Tantau Avenue, Cupertino, California 95014 • (408) 255-0900