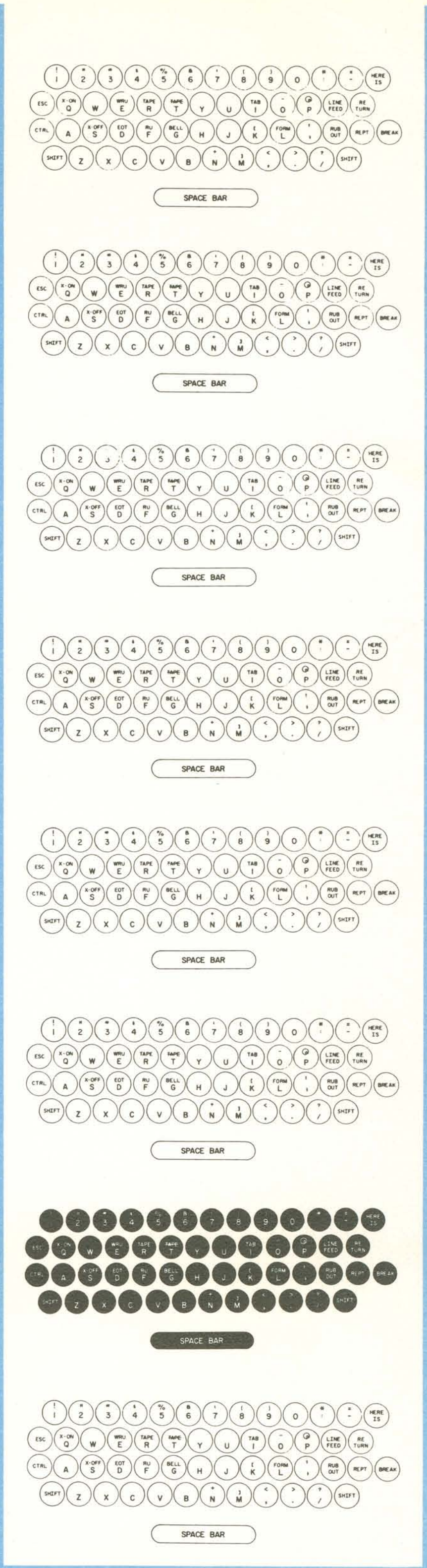
CONTROL DATA®
6000 SERIES TIME-SHARING

KRONOS

BATCH USER'S



REFERENCE MANUAL

CONTROL DATA
CORPORATION

CONTROL DATA®
6000 SERIES TIME-SHARING

KRONOS

BATCH USER'S

REFERENCE MANUAL

CONTROL DATA
CORPORATION

RECORD OF REVISIONS

Revision	Notes
A	Released 2-27-70

Address comments concerning this manual to:

Control Data Corporation
Business and Industrial Systems Division
Technical Publications
4201 North Lexington Avenue
St. Paul, Minnesota 55112

KRONOS
Batch User's Reference Manual
Publication Number 59150600

PREFACE

The KRONOS Time-Sharing System was developed by the Business and Industrial Systems Division of Control Data Corporation. KRONOS processes jobs from a maximum of 384 time-sharing terminals, central site batch jobs, and remote batch jobs.

This manual describes the external features of the KRONOS Operating System for the batch user. It does not contain a detailed internal description of the associated software nor a description of the time-sharing commands.

For further information concerning Control Data® 6000 Series computers and the KRONOS Time-Sharing System, consult the following manuals:

Title	Publication Number
Control Data 6400/6500/6600 Computer Systems Reference Manual	60100000
Time-Sharing FORTRAN Reference Manual	59150900
FORTRAN Translator (FTNTRAN) Reference Manual	59151000
EXPORT/IMPORT Reference Manual	59150500
BASIC Reference Manual	59150800
ALGOL Reference Manual	59151200
Text Editor (EDIT) Reference Manual	59150700
KRONOS Operating Guide	59151600
Control Data MODIFY File Editing System Reference Manual	59151100
Instant KRONOS	59152100
Time-Sharing User's Reference Manual	59151300
KRONOS Terminal User's Instant Manual	59152000

CONTENTS

CONTENTS (Cont'd)

CONTENTS (Cont'd)

APPENDICES

FIGURES

TABLES

SYSTEM DESCRIPTION 1

INTRODUCTION

The KRONOS Time-Sharing System coordinates multiple-user access to one CDC 6000 Series computer. Programs can be submitted from:

- Time-Sharing Terminal
- Central Site
- 200 User Terminal

The user can submit his job to KRONOS from a time-sharing terminal. The terminal responds as if it were a small computer console to which the user has sole access. However, the 6000 Series computer at the central site allocates only a small portion of its total time to process the requests of each individual terminal in succession. Hence, KRONOS is a time-sharing system.

A user can also enter his program at the computer center. He can then use all the equipment attached to the computer; i.e., card reader, punches, line printers, tapes, and other peripheral devices.

The user can also communicate with the 6000 Series computer from a 200 User Terminal at his own site. One or more jobs are collected into a job stack which is sent to the computer center over telephone lines. KRONOS processes the job stack in a manner similar to that of local batch processing. The system transmits the resulting output to the remote site.

KRONOS operation and performance depend directly on the system resources available for job processing. The job types, the number of time-sharing users, and the remote batch terminals determine the minimum hardware configuration necessary for system operation.

HARDWARE/SOFTWARE INTEGRATION

KRONOS uses the ten Peripheral Processor Units (PPUs) for system and input/output tasks, and the Central Processor Unit (CPU) to execute the user's jobs. Central Memory (CM) contains the user programs and a system software area called Central Memory Resident (CMR). When the term "system" appears in this manual, it is the equivalent of the KRONOS operating system. When lower case letters appear in format descriptions, they represent user-supplied values. Capital letters and punctuation marks† represent themselves.

† The exception is the ellipsis (...)

CENTRAL PROCESSOR UNIT

The CPU performs computational tasks but has no input/output capability. It communicates with the outside world through central memory. Under KRONOS, the CPU is used almost exclusively for program compilations, assemblies, and executions. The CPU program makes request of the system through the CPU Request register which is the Reference Address plus one (RA+1) of the current program.

PERIPHERAL PROCESSOR UNITS

The ten peripheral processors (PP0, PP1, . . . , PP9) are identical and perform a variety of tasks. However, the PPUs cannot assemble, compile, and execute user programs efficiently. Under KRONOS, the PPus are assigned various tasks such as system housekeeping, job processing, and input/output.

PPU number 0 contains the Monitor Program (MTR) that oversees or controls all other system activities. PP9, under the supervision of MTR, permanently drives the console typewriter and display scopes. The remaining PPUs, 1 through 8, are initially assigned to read their input registers (specific locations in central memory) over and over. To make a request, the monitor inserts a significant word into the input register of a PPU. When the PPU reads its input register, it obeys the request (or determines that it cannot do so), sets a drop PPU request in its output register to indicate to the monitor that the PPU has processed the request, and returns to its idling state. When idling, pool PPUs continually read their input registers. Thus, all requests to a PPU (other than PP0) are communicated through the input register of that PPU.

Each PPU (other than PP0) uses its output register (another location in central memory) for requests to the monitor and for completion status of the requests. The monitor periodically searches the PPU output registers for requests. The monitor zeros certain bits in the PPU output registers when the requests have been processed.

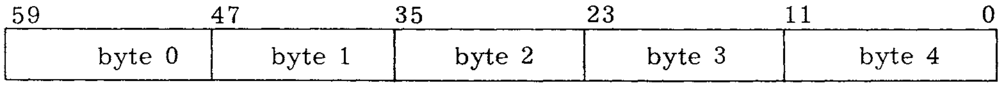
Although the primary task of a PPU is to act on requests from MTR, a PPU most occasionally request the cooperation of another PPU. PPUs request these additional PPUs through the monitor and must request permission from the monitor before using an input/output channel. Since each PPU is capable of connecting itself to any channel, only one PPU can use a specific channel at one time. To avoid two PPUs attempting to use the same channel (which would hang up both PPUs and the channel), the monitor maintains a list of channels and their status. Whenever a PPU requires a channel, it must first request the monitor to assign that channel for its exclusive use. When finished with the channel, the PPU specifies to the monitor that the channel is free.

CENTRAL MEMORY (60-BIT WORDS)

A number of programs can be executed concurrently under KRONOS. These programs are stored in central memory (user area) along with a set of necessary data for system operation (central memory resident). Central memory is accessible to both the CPU (within the field length of a given program) and the PPUs, and thus forms the communications link between the 11 processors and the system.

Central memory resident contains the library directory, system communication area, system tables, the CPU resident routine, and information about each job currently being executed. The user area contains the programs currently being executed for each job.

Central memory words are 60 bits in length and contain five 12-bit PPU memory words. These five PPU words, called bytes, are numbered 0 through 4:



Central Memory Word

MULTIPROGRAMMING

CONTROL POINTS

The system can execute several jobs simultaneously. During execution, these jobs are numbered (1-n). The index of each job is called a control point. When the system has selected a job for execution, it assigns the job to a control point. The control point number identifies and differentiates the job from the other jobs in execution. Each control point has a control point area in central memory that contains all of the information necessary for KRONOS to define and process the assigned job.

CONTROL POINT AREA

When a job is in central memory, the control point area to which it is assigned contains such information as job name, length, starting address in CM, elapsed time, assigned I/O equipment, and control statements. The control point area also contains a 16-word section called the exchange package. The exchange package contains all necessary information for starting or resuming a CM program — the contents of all registers used in executing a program.

Under KRONOS, control point 1 is dedicated to TELEX, the CPU executive for the REMOTE teletypewriter communication package. Control point n is reserved for the EXPORT/IMPORT executive.

CONTROL POINT NUMBER

In the control point system, a 5-bit number identifies a job in process. For example, a user's program requests the system to read a magnetic tape. This request generates several internal requests (which the PPUs pass back and forth). Each of these requests requires only five bits to identify the user's job.

FILES

A file is an organized collection of data. The file name identifies it to the user and to the system. KRONOS, the jobs it processes, and intermediate results are files or parts of files. A file consists of one or more logical records. Each logical record contains one or more Physical Record Units (PRUs) of data.

Files can be transferred from one device to another since equivalent formats are used for files on cards, printer, disk, and magnetic tape. The information in a file is stored serially. An object program can operate on named files and the actual medium of a file can be specified on control cards; disk storage is assumed if no medium is specified.

Coded information is stored internally in display code on either a disk sector or a magnetic tape record. The concept of logical records makes it possible to have equivalent forms of a file on several media, without losing the advantages of each form of storage. For example, several cards constituting a logical record can be transferred to an equivalent form on disk storage where they are blocked in sectors.

FILE NAMES

File names consist of one to seven alphanumeric characters.

Input and output operations of a central program involve a named file — a disk, magnetic tape, punched card, or printer file. The physical unit associated with a file name is controlled by the job control cards and is not a function of the central program coding directly. The operating system provides a common interface between the central program and the peripheral programs which drive the equipment.

The special files named:

- INPUT
- OUTPUT
- PUNCH
- PUNCHB
- P8

are supplied to each job by the system. The user should not try to assign these names to other temporary or I/O files.

INPUT FILE

The file named INPUT is the file from which the job can be read after it has been assigned to a control point. Before a job is at a control point, the job file is identified by the job name.

OUTPUT FILE

The file named OUTPUT contains the results which are to become printed copy when the job terminates. During job processing, the file OUTPUT collects records which are to be printed. When job processing is completed, the name of this file changes from OUTPUT to the job name.

PUNCH FILE

The file named PUNCH contains the data which is to become Hollerith punch card output when the job terminates. When job processing is completed, the name of this file changes from PUNCH to the job name.

PUNCHB FILE

The file named PUNCHB contains the data which is to become binary punch card output when the job terminates. When job processing is completed, the name of file PUNCHB also changes to the job name.

P8 FILE

The file named P8 contains the data which is to become 80-column binary punch card output when the job terminates. When job processing is completed, the name of file P8 also changes to the job name.

LOGICAL RECORDS

All files under the KRONOS system are organized into logical records. Input files are organized into logical records by the language translator or other program producing the output. Otherwise, the user must generate logical records.

Since the logical record is defined for each type of peripheral device, files retain their structure when transferred between devices. The physical format of a logical record is determined by the device on which the file resides. The physical record unit size is the smallest amount of information the system transfers during a single physical read or write operation for each device.

Logical records consist of one or more PRUs, the last of which is short- or zero-length. For mass storage and tape, a physical record unit that contains less than the normal amount of data marks the End-of-Record (EOR). Card data has a special EOR mark. An EOR is written on tape or mass storage as a zero-length PRU if the logical record data is an even multiple of the PRU size, or if a write operation was requested when no data was in the buffer. A zero-length PRU is a PRU containing no data.

DATA FORMATS

Data within the system can be binary or coded. Binary records can be of any length. Coded records are lines of display-coded characters. Binary data is in images of central memory but is blocked into physical records. The data block lengths are device-dependent.

Each line of coded data begins at the first byte of a word and continues two characters per byte to the end of a line. The line can be of any length, but should be of a size corresponding to the device to which the line will be written. The last CM word of the line is completed with at least one cleared byte. A cleared byte is the End-of-Line designation.

Coded data on mass storage or odd parity tape has the same format as binary data, and can be grouped into records. The following paragraphs describe data formats for each device.

MASS STORAGE

All data on a mass storage device is in 64 CM word PRUs. Each PRU is preceded by two bytes of control information which is available to the system only. Files on mass storage can be randomly accessed (logical records of mass storage can be read or written directly without reading or writing the entire file). Under KRONOS, a user can access a random file record by address or by record index. In Address mode, a user can access any random access file. In Index mode, a user can access only those random access files which have a list of record names or numbers as the first logical record. The addressing structure allows random access files to be temporarily stored on non-random devices without losing their random access characteristics.

After an initial random access read request for a file or End-of-Information has been written, the system treats a random access file like a sequential file. Subsequent read or write requests occur at the current file position.

The RPHR and WPHR macros can be used to randomly read or write a PRU of data.

KRONOS allows a random access file to be used as a sequential file. However, a user should not assume that a rewritten random access file is in any particular order.

ADDRESS MODE

The Address mode of random access allows the user to read or rewrite any group of PRUs. The user specifies the address of the first PRU to be read or rewritten. The length of the buffer or data in the buffer determines the number of PRUs to be read or rewritten. Therefore, the rewrite feature should be used carefully because it is possible to overwrite into the next block of data if a user requests a random write when more data is in the buffer than in the original block. Also, End-of-File (EOF) and End-of-Record write requests can cause control PRUs to overlap into following data.

To request Address mode random access processing, set the random access bit (r), set the first sector number in the random request field of the File Environment Table (FET) (Section 4), and set bit 17 of the random request field.

INDEX MODE

The Index mode of random access requires a table of record names or numbers with the associated first sector numbers. When an indexed random access file is to be saved, the system writes this table as the last logical record of the file.

A table of record names requires two words per record. A table of record number requires one word per record.

All random write requests on an indexed file occur at the End-of-Information mark. Therefore, when a record on an indexed file is rewritten, it can differ in size from the original record. The system writes the new record at the End-of-Information (EOI) mark and does not release the space reserved for the old record.

KRONOS provides system macros for Index mode random access (Section 4).

CODED PUNCHED CARDS

Coded cards are in Hollerith code, 80 characters per card. The system reads data from cards, converts it to display code, deletes trailing spaces, and packs it into a line of up to nine CM words. The system punches data for a card until an End-of-Line mark or the 80th character appears. If an End-of-Line mark does not appear after 80 characters, data is lost. Since the search for an End-of-Line mark terminates when 14 CM words have been checked, a maximum of 60 characters can be lost if this condition occurs.

CONVERSION

Conversion can be specified by the code card. This card has a 5-7-9 punch in column 1. Column 2 specifies the conversion.

Column 2	Conversion
blank	O26
9	O29 FORTRAN
8	O29 COBOL

Conversion is initially set to O26. A conversion code remains in effect until changed or until an End-of-Information mark is read, whichever occurs first.

BINARY PUNCHED CARDS

Binary cards contain:

- up to 15 words per card,
- a 7-9 punch in column 1,
- word count of card in column 1, rows 0, 1, 2, 3,
- an ignore checksum punch in column 1, row 4,
- the checksum modulo 4,095 in column 2,
- the binary sequence number in columns 79 and 80, and
- a blank in column 78.

An End-of Record mark has a 7-8-9 punch in column 1. An End-of-File mark has a 6-7-9 punch in column 1. An End-of-Information mark has a 6-7-8-9 punch in column 1.

PRINTED DATA

All printed data is in coded format. The system extracts data to be printed until an End-of-Line mark or the 136th character appears. If an End-of-Line mark does not appear after 136 characters, data is lost. Since the search for an End-of-Line mark terminates when 14 CM words have been checked, up to four characters can be lost.

CARRIAGE CONTROL

The system recognizes the first character in a line as the carriage control character. If the first character is a carriage control character (Table 1-1), it is not printed. When the system recognizes a carriage control character, it prints a line containing a maximum of 135 characters. Print mode is normally Auto Eject mode (single-space and bypass paper crease). This is the only carriage control command that remains in effect until changed. All other carriage control commands must be given for each line that they control.

TABLE 1-1. CARRIAGE CONTROL CHARACTERS

Character	Command
space	Single-space
1	Eject page before print
0	Skip one line before print (double space)
–	Skip two lines before print (triple space)
+	Suppress space before print
/	Suppress space after print
2	Skip to last line of form before print †
8	Skip to format channel 1 before print †
7	Skip to format channel 2 before print †
6	Skip to format channel 3 before print †
5	Skip to format channel 4 before print †
4	Skip to format channel 5 before print †
3	Skip to format channel 6 before print †
H	Skip to format channel 1 after print
G	Skip to format channel 2 after print
F	Skip to format channel 3 after print
E	Skip to format channel 4 after print
D	Skip to format channel 5 after print
C	Skip to format channel 6 after print
Q	Clear Auto Eject
R	Set Auto Eject
S	Select 6 lines
T	Select 8 lines

†No space after print

STANDARD FORMAT MAGNETIC TAPE

The standard KRONOS magnetic tape format is 7-track, 1/2-inch tape. The system writes to tape in odd parity; 512 CM word PRUs. An EOR mark is a short PRU (less than 512 words) and an EOF mark is a tape file mark. Data is in central memory image.

UNBLOCKED EXTERNAL BCD FORMAT MAGNETIC TAPE

Unblocked External BCD format is 7-track, line or card image 1/2-inch tape. The system writes to tape in even parity, 136 character PRUs unless the user specifies another PRU size on an ASSIGN or REQUEST card. Both EOR and EOF marks are represented by tape file marks. The system converts data to display code, deletes trailing spaces, and stores ten characters per CM word for a read operation. For a write operation, the system converts data to External BCD, adds trailing spaces if necessary, and writes the characters in a coded line, discarding characters that occur after the specified number.

BLOCKED EXTERNAL BCD FORMAT MAGNETIC TAPE

Blocked External BCD format is 7-track, 1/2-inch tape. The system writes to tape in even parity, 150 character PRUs unless the user specifies another PRU size on an ASSIGN or REQUEST card. Both EOR and EOF marks are represented by tape file marks. The system converts data to display code and stores ten characters per CM word for a read operation. If necessary, the system completes the last word with spaces. For a write operation, the system converts data to External BCD and writes the characters in a coded line, discarding the characters that occur after the specified number.

PERMANENT FILES

Under KRONOS the user has access to permanent files — files which cannot be lost by a deadstart operation. Permanent files can be accessed in any available system mode (Address or Index) once the file has been retrieved from permanent file storage and is available at the user's control point.

JOB PROCESSING 2

A job consists of one file of punched cards or card images. The first logical record of a job file contains the control cards that specify the job's processing requirements. The system processes jobs in three sequential but independent stages:

- Input
- Execution
- Output

Many jobs can be in the input and output stages of processing but only n jobs (one for each control point) can be in the execution stage. Each job must begin with a job card and end with an End-of-Information card. All other control cards follow the job card directly. The end of the control cards is signified by a 7-8-9 card (End-of-Record) or a 6-7-8-9 card (End-of-Information) if the job consists of control cards only.

JOB INPUT

The system reads an entire job from the card reader and stores it on mass storage in the input queue. A typical job file has three logical records: control cards, program cards, and data cards.

JOB EXECUTION

The system executes a job by:

- bringing the job to a control point,
- following the directives of the control cards, and
- accumulating data for the output stage of job processing.

A job is executed only when it is assigned to a control point. One job is assigned to each control point. When a control point becomes available, the system selects a job from the input queue and assigns it to the free control point.

After a job is assigned to a control point, KRONOS advances the job according to the job control cards. These control cards contain directives (such as LOAD or EXECUTE) that are interpreted and obeyed, one at a time, in the order they appear in the job file. For example, a compilation is achieved by a load and execute of the desired compiler program.

During execution, the system accumulates the output data (if there is any) in files on the system mass storage device.

JOB OUTPUT

When the system has obeyed the last control card for a job, it enters the files of accumulated output data into the output queue. The system selects files from the output queue and processes them according to file type (punched cards, printer listings, etc.).

CONTROL CARD TRANSLATION

KRONOS translates a control statement by:

1. Reading the statement from the control point control card buffer. If necessary, the system reloads control statements from the file INPUT.

2. Deleting all spaces between the beginning of the statement and the termination character (a period or a right parenthesis). KRONOS allows only standard FORTRAN characters to appear before the termination character, although other characters can appear in the comment field.

3. Searching the list of control card names and comparing them with the name of the card being processed. If the card name is on the list, the system processes the control statement according to the parameters. If the card name is not on the list, the system searches further.

4. Searching the File Name Table with the first characters (seven or less up to the separator character) for a file assigned to the control point with a name identical to the first characters of the control statement. If the system finds such a file on a mass storage device in absolute format, the system reads the file into central memory as a CPU program. If the file resides on a non-mass storage device, KRONOS aborts the job. If the file is in relocatable format on a mass storage device, the system transfers control to the loader (Section 5), extracts the arguments from the control statement, stores them in RA+2 through RA+(n+1), and requests the CPU to begin program execution.

5. Searching the Chippewa Library Directory for a program name that coincides with the card name. If the system finds such a program, it loads the program, stores the arguments in RA+2 through RA+(n+1), and requests the CPU to begin program execution.

6. Searching the SCOPE Library Directory for a program name that coincides with the card name. If the system finds such a program in absolute format, the system proceeds as for a Chippewa Library program (Step 5). If the program is in relocatable format, the system transfers control to the loader (Section 5).

7. Searching the Peripheral Library Directory if the statement name is three characters long and begins with a letter. If the system finds such a program, it constructs a call to a PPU using the name and arguments from the control statement.

If KRONOS cannot process the control statement during these steps, it declares the control statement illegal, issues a dayfile message, and aborts the control point job.

CONTROL CARDS 3

As introduced in Section 2, control cards direct job execution. KRONOS recognizes four types of control cards:

- Job
- Program Execution
- File Management
- Permanent File

All control cards, except the job card and program call cards, have two fields. The first field contains the card name, beginning in column 1. Card names described in this section are reserved for the system and cannot be used as program call names. The second field is optional; it can contain one or more parameters separated by commas. The two fields are separated by a separator character: + - " / = , (or $.

The parameter field is terminated by a period or right parenthesis. A terminator must be present, even though no parameters are specified.

The card names on job cards and program call cards contain the name of the job and the name of the program, respectively.

JOB CONTROL CARDS

JOB CARD

Control card format: jobname, Tt, CMfl, Pp. or
jobname, p, t, fl.

jobname Alphanumeric job name (one to seven characters) must begin with a letter. To assure unique job names, KRONOS replaces the last three characters with a system-generated value.

t Central processor time limit in octal seconds (maximum 77770_8). This time limit must suffice for compilation and execution of the job. If t is absent, KRONOS assumes t=100_8 (100_8 seconds ≈ 1 minute).

fl Total central memory field length of the job; a maximum of six octal digits. The system rounds the field length (storage requirement) to a multiple of 100_8. This field length cannot exceed:

- $360{,}000_8$ on a 131K machine
- $163{,}000_8$ on a 65K machine
- $61{,}000_8$ on a 32K machine

If fl is absent, KRONOS assumes $fl=50{,}000_8$.

p Priority level (octal) at which job enters the system; $1 \leq p \leq 17$. If p is absent, KRONOS assumes p=10.

The first control card for a job indicates the job name, priority, CPU time limit, and memory requirements. Commas separate the fields and a period terminates the job card. Blanks have no meaning. Fields other than jobname can appear in any order when identified by the leading characters in the field.

ACCOUNT CARD

Control card format: ACCOUNT, xxxxxxx, pswd.

xxxxxxx	user account number
pswd	user password

The second card in the control card record must be the ACCOUNT card. It specifies the user's account number. This account number is used in system bookkeeping as well as for user access to permanent files.

ONSW CARD

Control card format: ONSW $(n_1, n_2, \ldots, n_n)$

n_i Pseudo-sense switch number; $1 \leq n \leq 6$.

The ONSW control card sets pseudo-sense switches for reference by the user's program. The system stores the sense switch settings in the control point area and copies them to address RA of the user's area for use by the central program. The system operator can change these switch settings by console command.

OFFSW CARD

Control card format: OFFSW $(n_1, n_2, \ldots, n_n)$

n_i Pseudo-sense switch number; $1 \leq n \leq 6$.

The OFFSW control card clears pseudo-sense switches for reference by the user's program. The system stores the sense switch settings in the control point area and copies them to address RA of the user's area for use by the central program. The system operator can change these switch settings by console command.

MODE CARD

Control card format: MODE(n)

n	Exit Condition
0	Disable Exit mode; no selections made
1	Address is out of range because: ● Attempt was made to reference central memory or extended core storage outside established limits ● Word count in extended core storage communication instruction is negative ● Attempt was made to reference last 60-bit word (word 7) in relative address FL ECS
2	Operand out of range; floating-point arithmetic unit received an infinite operand
3	Address or operand is out of range
4	Indefinite operand; floating-point arithmetic unit received an indefinite operand
5	Indefinite operand or address is out of range
6	Indefinite operand or operand is out of range
7	Indefinite operand, operand is out of range, or address is out of range

The MODE control card selects the exit or stop conditions for the CPU program. When MTR executes an exchange jump, the exit selection code enters the CPU. The exit occurs as soon as the selected condition arises. If the user does not specify an error mode, the system assumes n=7.

COMMENT CARD

Control card format: COMMENT. comments or
*comments

comments Combination of characters the user wishes to display

The system displays the combination of characters following the period or asterisk on a COMMENT control card and enters them into the dayfile.

EXIT CARD

Control card format: EXIT.

The EXIT card separates the control cards for normal execution from those used when an error exit occurs. When the error exit condition occurs, the system searches the control card record for the next EXIT card. If the record does not contain an EXIT card, the system terminates the job. If the system finds an EXIT card, it clears the error condition and obeys the subsequent control cards. Appendix D describes the error messages.

MAP CARD

Control card format: MAP.

The MAP card sets the Loader Map flag for the control point. The loader will generate a core map for the job.

PARTIAL MAP CARD

Control card format: MAP(P)

The MAP card with a defined parameter sets the Partial Map flag for the control point. A full MAP card or NOMAP card clears the Partial Map flag.

NOMAP CARD

Control card format: NOMAP.

The NOMAP card clears the Loader Map flag for the control point. The loader will not generate a core map for the job.

REDUCEFL CARD

Control card format: REDUCEFL.

The REDUCEFL card clears the No Reduce FL flag for programs which normally reduce field length for the job.

NOREDUCE CARD

Control card format: NOREDUCE.

The NOREDUCE card sets the No Reduce FL flag for the job.

ROLLOUT CARD

Control card format: ROLLOUT.

The ROLLOUT card requests that the user's job be rolled out so that all memory assigned to the job (except the control point area) can be released.

SETPR CARD

Control card format: SETPR(p)

p priority level; $1 \leq p \leq 17$.

The SETPR control card allows the user to specify a new priority level for his job.

SETTL CARD

Control card format: SETTL(t)

t Central processor time limit in octal seconds (maximum 77770_8)

The SETTL control card permits the user to specify a new time limit for his job.

PROGRAM EXECUTION CONTROL CARDS

These control cards direct loading and execution of files. Program execution control cards have the general format:

name list comment

The card is a unit record of up to 80 characters, including freely interspersed spaces.

The name and list fields are required and comment is optional. The name is a string of one to seven alphanumeric characters. The comment is a string of Hollerith characters.

The list contains parameters to be used by the program being loaded. The contents of the list can vary greatly, depending on the requirements of the program being loaded. If parameters are not required, list is simply a period. Parameters can be enclosed in parentheses or preceded by a comma and terminated by a period. The list can contain as many parameters as fit on one card. The program being loaded dictates the form of the parameters to be used. The parameters can specify the type of information flow, input BCD output, binary output, special information, or the name of a file involved.

LOAD CARD

Control card format: LOAD (lfn_1) or
LOAD (lfn_1, lfn_2, . . . , lfn_n)

lfn_1 Name of file to be loaded

lfn_2-lfn_n Names of library files (other than the system library) from which to satisfy external references of lfn_1 $(0 \leq n \leq 50)$

The LOAD card directs the system to load the file lfn_1, and the programs from files lfn_2 through lfn_n required to satisfy external references occurring in lfn_1 into central memory.

Files lfn_2 through lfn_n must reside in mass storage. Loading begins from the current file position. Loading of lfn_1 terminates when the EOI mark or an empty record appears. The user can load segments and relocatable binary decks with the LOAD control card.

Use a LOAD card for each complete file required. The first record of the first file determines the kind of loading for subsequent LOAD cards.

SATISFY CARD

Control card format: SATISFY (lfn_1, lfn_2, . . . , lfn_n)

lfn_i Names of library files (other than the system library) from which to satisfy external references

For the SATISFY card, the system loads the programs from files lfn_1 through lfn_n required to satisfy external references in a file that the system has already loaded into central memory as a result of a LOAD card.

File lfn_i must reside in mass storage when a SATISFY card appears.

EXECUTE CARD

Control card format: EXECUTE (name, p_1, p_2, . . . , p_n)

name Program entry point where execution is to begin. If name is absent, the system uses the last transfer address (XFER) encountered

p_i List of parameters

The EXECUTE card causes the loader to complete program loading. This includes filling all unsatisfied references with entry points from the system library except where inhibited by segment parameters.

For segment or overlay operations, program execution begins in the first segment or the main overlay.

PROGRAM CALL CARD

Control card format: lfn (p_1, p_2, . . . , p_n)

lfn Program name and entry point where execution is to begin

p_i List of parameters

The system searches the File Name Table/File Status Table for a file named lfn. If it finds file lfn, it loads the file subprograms and bypasses any routines already loaded by LOAD cards. The system rewinds the file before loading begins. If the system does not find the file named lfn in the FNT/FST, it searches the system library and loads a subprogram with the same lfn name. When the system completes loading, and if it finds no fatal errors, it passes the parameters to the requested program and begins execution at the entry point named lfn.

Examples:

```
LOAD (ESTE)
EXECUTE.          is equivalent to: ESTE.
```

To replace one subprogram with a subprogram of the same name from another file, the sequence could take the form:

```
LOAD (HOST)
GUEST.
```

The subprograms will be loaded from the file HOST and the system bypasses the subprograms of the same name on GUEST.

FORTRAN CARD

Control card format: RUN (cm, fl, bl, if, of, rf, lc, as, cs)

cm — Compiler mode option; if omitted, the system assumes cm = G; if unrecognizable, the system assumes cm = S.

cm	Mode
G	Compile and execute; do not list unless explicit LIST cards appear in the deck
S	Compile with source list; do not execute
P	Compile with source list and punch deck on file PUNCHB; do not execute
L	Compile with source and object list; do not execute
M	Compile with source and object list; produce a punch deck on file PUNCHB; do not execute

fl — Object program field length (octal); if omitted, the system sets fl = field length at compile time.

bl — Object program I/O buffer length (octal); if omitted, the system sets bl = 2022_8

if — Name of file containing compiler input; if omitted, the system assumes if = INPUT

of — Name of file to receive computer output; if omitted, the system assumes of = OUTPUT

rf — Name of file to receive binary information; if omitted, the system assumes rf = LGO lc Line limit (octal) of the object program OUTPUT file; if omitted, the system assumes lc = 1000_8; if the line count exceeds the line limit, the system terminates the job

as ASA switch; if non-zero or non-blank, causes ASA I/O list/format interaction at execution time.

cs Cross reference switch; produces a cross reference listing

The RUN control card calls the FORTRAN compiler to the job control point.

COMPASS CARD

Control card format: COMPASS $(p_1, p_2, \ldots, p_n)$

p_i Parameters; can be in the following format:

a
a = fname
a = 0

Option	Meaning
A	Source is MODIFY compressed symbolic. If A is omitted, no A option is assumed.
B	Binary on file LGO. If no B parameter is supplied, this option is assumed.
B=0	No binary.
B=fname	Binary on file fname.
D	Generates binary even if there are assembly errors. If D is omitted, no D option is assumed.
LO	Select list options: CFGX.
LO=0	Normal list options. Options L and R are set.
LO=chars	Select list options according to the character string "chars".

Character	Lists
L	Normal listing
R	Reference information
G	Lines that result in code generation
A	Lines with → or ≠ marks removed
C	EJECT, SPACE, and TITLE control cards
D	Detailed code not normally listed
E	All iterations of duplicated code
F	Lines skipped by IF-type instructions
M	Lines generated by macro calls
S	Lines generated by systems macros
X	Lines generated by XTEXT instruction

I Input from file COMPILE.

I=fname Input from file fname.
When the I parameter is omitted, the system assumes I=INPUT.

L Long list on file OUTPUT. When the L parameter is omitted, this option is assumed.

L=0 No long list.

L=fname Long list on file fname.

O Short list on file OUTPUT. When the O parameter is omitted, this option is assumed.

O=0 No short list.

O=fname Short list on file fname.

P Select consecutive page numbering. When P is omitted, no P option is assumed.

S System text name SYSTEXT. When S is omitted, this option is assumed.

S=0 No system text.

S=sname System text name sname.

X External text from file OPL. When X is omitted, this option is assumed.

X=fname External text from file fname.

The COMPASS control card calls the COMPASS assembler to the job control point.

MODIFY CARD

Control card format: MODIFY (p_1, p_2, . . . , p_n)

p_i Parameters; can be in the following general formats:

a
a = fname
a = 0

Options	Specify
I	Use directive input from file INPUT
I=fname	Use directive input from file fname
I=0	Use no directive input

Options	Specify
P	Use file OPL for old program library
P=fname	Use file fname for old program library
P=0	Use no old program library file
C	Write compile output to file COMPILE
C=fname	Write compile output to file fname
C=0	Write no compile file
N	Write new program library on file NPL
N=fname	Write new program library on file fname
N=0	Write no new program library
S	Write source output on file SOURCE
S=fname	Write source output on file fname
S=0	Write no source output
L	List output on file OUTPUT
L=fname	List output on file fname
L=0	List no output information
LO	Select list options: ECTMWDS
LO=chars	Select list options according to the character string "chars":

Character	Lists
E	Errors
C	Directives other than INSERT, DELETE, RESTORE
T	Input text
M	Modifications made
W	Compile file directives
D	Deck status
S	Statistics
I	Inactive cards
A	Active cards

Any combination of characters selects the combination of list options.

A	Write compressed compile file
D	Ignore errors
F	Modify all decks
U	Modify only decks mentioned on DECK directives; F overrides U option
NR	Do not rewind compile file
X	Rewind input and output files, set A option, and call COMPASS assembler when modification is completed

X=prog	Rewind input and output files, set A option, and call the prog processing program when modification is completed
X=0	Do not call another processing program

The following options apply only if the X option is selected.

CB	Set assembler argument B=LGO
CB=name	Set assembler argument B=name
CB=0	Set assembler argument B=0
CL	Set assembler argument L=OUTPUT
CL=name	Set assembler argument L=name
CL=0	Set assembler argument L=0
CS	Set assembler argument S=SYSTEXT
CS=name	Set assembler argument S=name
CS=0	Set assembler argument S=0

With the MODIFY control card, the user calls the MODIFY program to his control point to edit a library file. The parameter list specifies additional information to MODIFY. The parameters can be in any order and are not mandatory. For any or all parameters omitted from the control card, the system uses a default value:

I = INPUT
P = OPL
C = COMPILE
N = 0
S = 0
L = OUTPUT

LO = ECTMWDS
A deselected
D deselected
F deselected
U deselected

NR deselected
X = 0
CB
CL = 0
CS

NOGO CARD

Control card format: NOGO.

When the loader encounters a NOGO card, it processes the loaded program in the same manner as for an EXECUTE card; however, it does not execute the program. This card permits program mapping, execution bypassing, and continuation of other portions of the job.

FILE MANAGEMENT CONTROL CARDS

The file management control cards permit the user to:

- request a file in a particular type of output,
- assign a file to a particular device,

- route a file to a particular destination,
- create a COMMON file,
- attach a COMMON file,
- change a LOCAL file to a COMMON file, and
- release a COMMON file.

If a file is not specifically assigned to a REQUEST or ASSIGN card, the system assigns it to disk storage. A job does not assign the card reader, printer, or punch for normal input/output from such things as compilations or assemblies since the system does this automatically. In addition, any file named OUTPUT, PUNCH, or PUNCHB will always be printed, punched, or punched binary by the system at job completion. The REQUEST card is used for large installations where the operator can best assign the available equipment when any peripheral device of a particular type is suitable. The ASSIGN card is for smaller installations and situations where only one particular peripheral device is suitable.

REQUEST CARD

Control card format: REQUEST (lfn, x_1, x_2, . . . , x_n)

lfn Logical file name; one to seven digits or letters. This is the name of the file to which equipment is to be assigned, and the name by which the user refers to the file within his program. A REQUEST card must have at least one parameter; the first parameter is the lfn.

x_i Any of the following:

LO	Density = 200 bpi
HI	Density = 556 bpi
HY	Density = 800 bpi
X	Process coded data in unblocked External BCD (136 characters/PRU unless otherwise specified). (Section 1)
B	Process coded data in blocked External BCD (150 characters/PRU unless otherwise specified). (Section 1)
C=xxxx	Set External BCD character count per PRU=xxxx; $1 \leq xxxx \leq 4{,}096$. Any value not in this range is ignored without comment and standard values are used

The user can fill his REQUEST card with x_i parameters. If parameter commands conflict, TSOS ignores all except the last parameter given.

The REQUEST card directs the system to display (at the system console) the user's request to assign a file to a peripheral device. The comment on the REQUEST card can specify the acceptable type of device. When the system operator assigns the peripheral equipment, job execution commences. Any equipment the user wishes to use must be requested on a REQUEST card or assigned on an ASSIGN card.

Because the system processes control cards in order, the REQUEST card must appear before the user references the file. If a parameter appears more than once or is illegal, the system issues a message and aborts the job.

ASSIGN CARD

Control card format: ASSIGN (nn, lfn, x_1, x_2, . . . , x_n)

nn Ordinal in equipment Status Table (EST) of peripheral device or equipment type:

Type	Equipment
CP	Card punch
CR	Card reader
DA	6603 disk
DB	6638 disk
DC	863 drum
DD	854 disk drive
DE	Extended core storage
DF	814 disk
DG	Disk drive group
LP	Line printer
MT	1/2-inch magnetic tape

lfn Name of file to be assigned

x_i Any of the following:

LO	Density = 200 bpi
HI	Density = 556 bpi
HY	Density = 800 bpi
X	Process coded data in unblocked External BCD (136 characters/PRU unless otherwise specified). (Section 1)
B	Process coded data in blocked External BCD (150 characters/PRU unless otherwise specified). (Section 1)
C=xxxx	Set External BCD character count per PRU=xxxx; $1 \leq xxxx \leq 4,096$. Any value not in this range is ignored without comment and standard values are used

The user can fill his ASSIGN card with x_i parameters. If parameter commands conflict, KRONOS ignores all except the last parameter given.

The ASSIGN card directs the system to assign equipment nn to file lfn. The system will refuse to assign the equipment if the user is not allowed to use the device or the file is already assigned to another device.

SETID CARD

Control card format: SETID ($\text{lfn}_1 = a_1$, $\text{lfn}_2 = a_2$, . . . , $\text{lfn}_n = a_n$)

lfn_i	Logical file name
a_i	New area code for file

The SETID card assigns a new area code or changes the area code for file lfn. The area code allows the user to route his file to a particular output device or device group because of proximity or other considerations.

COMMON CARD

Control card format: COMMON (lfn_1, lfn_2, . . . , lfn_n)

lfn_i	Name(s) of COMMON file(s)

If file lfn_i is a COMMON file in the FNT/FST and is not being used by another job, the system assigns it to this job. If the file is being used by another job or is not a COMMON file, the job must wait until the file is available.

If file lfn appears as a LOCAL file for the job, the system changes it to a COMMON file in the FNT/FST and it becomes available to any succeeding job after the current job terminates. However, if a COMMON file of the same name already exists, the system rejects the COMMON request and allows the job to continue.

If the file resides on a non-allocatable device such as magnetic tape, the equipment is reserved until the user releases the COMMON file.

RELEASE CARD

Control card format: RELEASE (lfn_1, lfn_2, . . . , lfn_n)

lfn_i	Logical file name

For the RELEASE card, the system changes COMMON file lfn_i, currently assigned to the job, to a LOCAL file in the FNT/FST. The file is then discarded at the end of the job unless the user program changes the file type by a subsequent file manipulation request.

LOCK CARD

Control card format: LOCK (lfn_1, lfn_2, . . . , lfn_n)

lfn_i Logical file name of a LOCAL file

With the LOCK card, the user can set the write lockout for local file lfn. Subsequently, the system allows only read operations on the file.

UNLOCK CARD

Control card format: UNLOCK (lfn_1, lfn_2, . . . , lfn_n)

lfn_i Logical file name of a LOCAL file

With the UNLOCK card, the user can clear the write lockout (rescind a LOCK command for local file lfn).

MACE CARD

Control card format: MACE.

The MACE card clears the SCOPE bit for loading of a file.

SCOPE CARD

Control card format: SCOPE.

The SCOPE card sets the SCOPE bit for loading of a file.

PERMANENT FILE CONTROL CARDS

SAVE CARD

Control card format: SAVE, $lfn_1 = pfn_1$, . . . , $lfn_n = pfn_n$

lfn_i Local file name; name of the file to be saved

pfn_i Permanent file name; name under which the file is to be stored on the permanent file device. (If omitted, $pfn_i = lfn_i$)

The SAVE control card permits the user to retain files in the permanent file system. As many files as can be specified on one control card can be retained with each SAVE command.

GET CARD

Control card format: GET, $\mathrm{lfn}_1 = \mathrm{pfn}_1$, . . . , $\mathrm{lfn}_n = \mathrm{pfn}_n$, ACCOUNT = xxxxxxx.

lfn_i Local file name; name given to the file when in use

pfn_i Permanent file name; name of the permanent file to be retrieved. (If omitted, $\mathrm{pfn}_i = \mathrm{lfn}_i$)

xxxxxxx The user number of another user; given when a user wishes to retrieve the file(s) of another user (provided permission has been granted)

The GET control card enables the user to retrieve a file from permanent file storage. As many files as can be specified on one control card can be retrieved with each GET command.

REPLACE CARD

Control card format: REPLACE, $\mathrm{lfn}_1 = \mathrm{pfn}_1$, . . . , $\mathrm{lfn}_n = \mathrm{pfn}_n$, ACCOUNT = xxxxxxx.

lfn_i Local file name; name of the new file that will be placed on the permanent file device

pfn_i Permanent file name; name of the file that is replaced (If omitted, $\mathrm{pfn}_i = \mathrm{lfn}_i$)

xxxxxxx The user number of another user; given when the user wishes to replace the file(s) of another user (provided permission has been granted)

The REPLACE control card permits the user to substitute a new file for an old file on the permanent file device. As many files as can be specified on one control card can be replaced with one REPLACE command.

PURGE CARD

Control card format: PURGE, pfn_1, . . . , pfn_n, ACCOUNT = xxxxxxx.

pfn_i Permanent file name; name of the file to be removed from the permanent file system

xxxxxxx The user number of another user; given when a user wishes to remove a file of another user from the permanent file system (provided permission has been granted)

The PURGE control card permits the user to remove a file from the permanent file device. As many files as can be specified on one control card can be purged with one PURGE card.

PERMIT CARD

Control card format: PERMIT, pfn, MODE = y, $accnum_1$, . . . , $accnum_n$.

pfn — Permanent file name; permanent file on which permission is to be granted.

y — Permission level. The possible permission levels are:

A — append only
E — execute only
N — none, removes previously granted permission
R — read and/or execute
W — write, read and/or execute

$accnum_i$ — The user numbers of persons to be granted access to file pfn

The PERMIT control card allows a user to grant file access to other users. As many users can be granted permission as user numbers can be contained on one control card.

APPEND CARD

Control card format: APPEND, lfn_1, pfn_1, ..., lfn_n, pfn_n, ACCOUNT = xxxxxxx.

lfn_i — Name of the local file to be appended to the permanent file

pfn_i — Name of the permanent file to which the local file is to be appended

xxxxxxx — The user number of another user; given when a user wishes to append the file(s) of another user

The APPEND control card permits the user to attach supplementary information to an existing file. The files lfn_i and pfn_i must always appear in pairs.

CATLIST CARD

Control card formats: CATLIST. or
CATLIST, F. or
CATLIST, xxxxxxx.

F — Specifies a full listing including the file length, creation date, and last access date of each permanent file

xxxxxxx — User number of another user; allows a user to obtain a listing of the files he can access in the catalog of user xxxxxxx.

The permanent file catalog control cards permit the user to obtain information about the files to which he has access. If a user has been granted access to the permanent files of user xxxxxxx (through the use of the PERMIT card), he can obtain a listing of the files to which he has access by specifying the user number xxxxxxx.

PROGRAM/SYSTEM COMMUNICATION 4

In addition to control cards, KRONOS provides a comprehensive set of system macros that the user's program can call to control input/output operations, request file manipulations, and communicate with the system at execution time. Input/output is accomplished via circular buffers.

During program execution, the system performs three types of operations as the result of macro calls:

- System operations; initiated by job action requests
- Input/output operations; initiated by file action requests
- File manipulations; initiated by file action requests

For file manipulation and input/output operations, the user must establish a File Environment Table (FET) for the pertinent file. The FET contains the circular buffer parameters for the operation. For FORTRAN or other compiler language input/output, the user should not be concerned about circular buffers and FET's as the compiler automatically provides these.

Moreover, the system does not require the FET to process job action requests.

This section describes circular buffers, File Environment Table, user/system communication subroutines, job action requests, and file action requests.

CIRCULAR BUFFERS

A circular buffer is a temporary storage area in central memory that contains data during input/output operations. It is called circular because routines that process input/output treat the last word of the buffer area as contiguous to the first word of the buffer area. The buffer parameters (FIRST, IN, OUT, and LIMIT) in the FET describe the circular buffer (see Figure 4-1).

FIRST

FIRST is the first word address of the buffer area. Routines that process I/O never change the value of FIRST.

LIMIT

LIMIT is the last word address plus one of the buffer area. Data is not stored in LIMIT. When LIMIT is reached, the next available address for storage is FIRST. Routines that process I/O likewise never change the value of LIMIT.

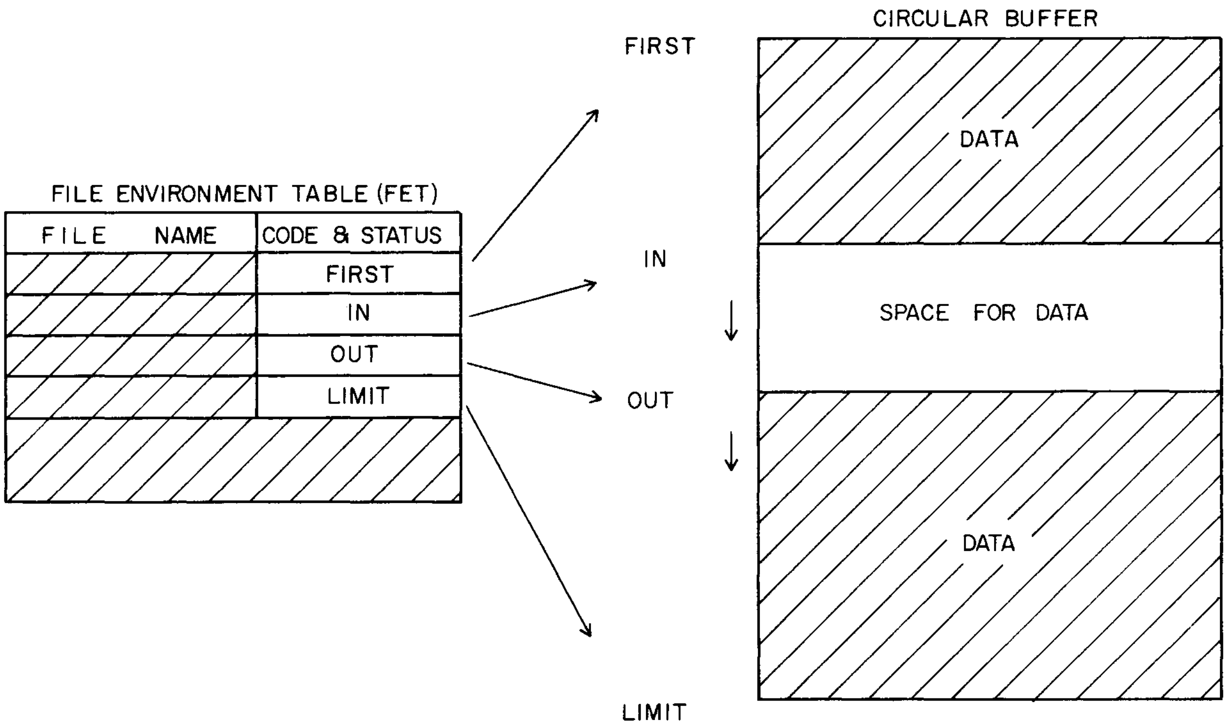


Figure 4-1. Circular Buffer

OUT

OUT is the next location to read to remove data from the circular buffer. Either the system or the calling program changes OUT depending on whether the operation is a write or a read (see Figure 4-2).

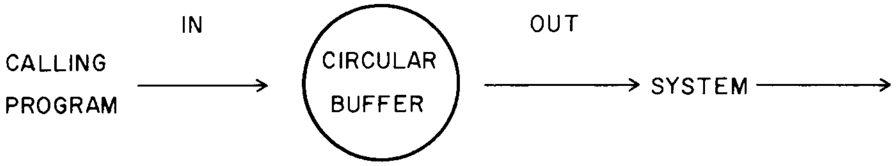


Figure 4-2. Write Operation

IN

IN is the next location to write data into the circular buffer. Either the system or the calling program changes IN depending on whether the operation is a read or a write (see Figure 4-3). When IN = OUT, the buffer is empty. When IN = OUT-1, the buffer is full.

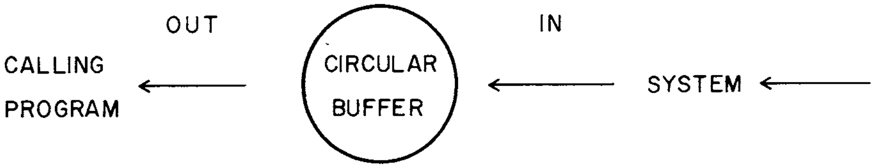
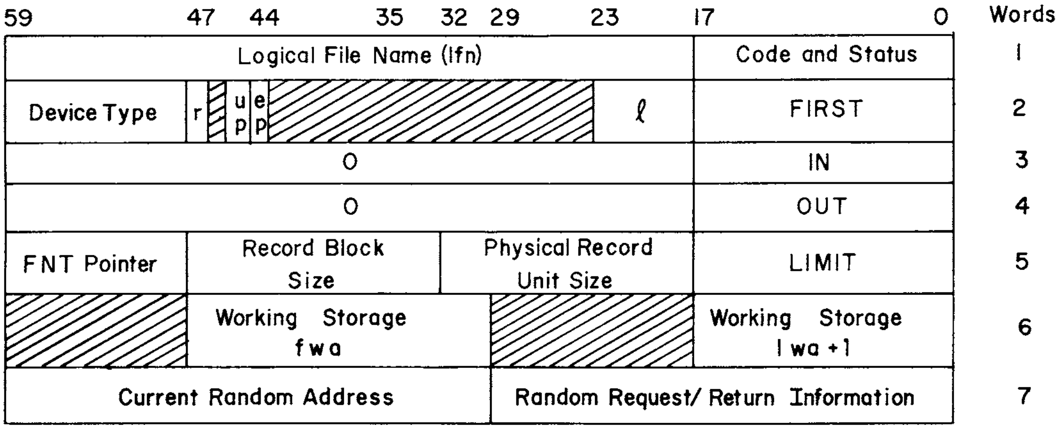


Figure 4-3. Read Operation

That is, one location is left empty in a full buffer to distinguish an empty buffer from a full buffer. A buffer is normally initialized with IN = OUT = FIRST, and IN and OUT circle the buffer as data is inserted and extracted.

FILE ENVIRONMENT TABLE

As introduced, the FET is a user-initiated communication area. The system and the user interrogate and update the FET during job execution. An FET within the user's field length must be initiated for each file of the user's job. The PPU I/O routine (CIO), CPU I/O subroutines, and the user's program access the system portion of the FET. A user section can be appended to the system FET to centralize other pertinent file information. The format of the system FET is shown in Figure 4-4.



59	47 44 35 32	29 23	17 0	Words
Logical File Name (lfn)			Code and Status	1
Device Type	r ▨ up ep ▨	▨ ℓ	FIRST	2
0			IN	3
0			OUT	4
FNT Pointer	Record Block Size	Physical Record Unit Size	LIMIT	5
▨	Working Storage fwa	▨	Working Storage lwa +1	6
Current Random Address		Random Request/ Return Information		7

Figure 4-4. System File Environment Table

The following lists describe the user and system information that appear in the system portion of an FET:

USER INFORMATION

Parameter	Word (Bits)	Description
device type	2 (48-59)	The system sets the device type in this byte when the user's program makes an OPEN request. Possible device types are: Code — Device 0401 — 6603 disk unit 0402 — 6638 disk unit 0403 — 836 drum 0404 — 854 disk drive 0405 — Extended core storage 0406 — 814 disk 0407 — Disk drive group 5524 — 1/2-inch magnetic tape 5420 — Line printer 4322 — Card reader 4320 — Card punch The device code is the display code for the device mnemonic with the most significant bit set for non-allocatable devices.
record block size	5 (33-47)	If the file resides in allocatable storage, the system returns the size of the device record block in this field at OPEN time. It is the number of physical record units in a record block. If the number of PRU's is not defined or is variable, the field is zero.
physical record unit	5 (18-32)	KRONOS returns the physical record unit size of the device to which the file is assigned in this field at OPEN time. It is the number of central memory words in a PRU.

SYSTEM INFORMATION

Parameter	Word (Bits)	Description
logical file name	1 (18-59)	The lfn field contains one to seven alphanumeric display code characters, left-justified; unused characters are zero-filled. This field is a common reference point for the CPU program and PPU I/O routines. The lfn parameter, declared in an FET creation macro, is also the location symbol associated with the first word of the FET. Thus, a reference to lfn in a file action request is a reference to the base address of the FET.

Parameter	Word (Bits)	Description
code and status	1 (0-17)	The code and status field communicates requested function codes and resulting status between the CPU program and the PPU I/O routines. The CPU program sets the request code in this field when it encounters a request for this file. The request codes are defined in the file action request descriptions and summarized in Appendix C. This field is initialized to one.
	1 (9-13)	Status information when request is complete; zero field indicates normal completion; nonzero field indicates abnormal completion (not necessarily an error).
	1 (1-8)	Request code for PPU call; these codes are defined in the file action request descriptions.
	1 (1)	File mode; 0 = coded, 1 = binary.
	1 (0)	Completion bit; 0 = busy or not complete, 1 = not busy or complete.
r	2 (47)	Random access bit; set to 1 if this is a random access file.
up	2 (45)	User processing bit; set to 1 to notify the calling program an End-of-Reel condition has been encountered during a 1/2-inch magnetic tape operation. If the up bit is set to 0, tape swapping proceeds automatically without notification to the calling program. The system completes the function being processed on the alternate tape reel. When the up bit is set to 1, and an End-of-Reel condition arises during a 1/2-inch magnetic tape operation, the system sets an End-of-Reel status (02_8) in bits 9-13 of the code and status field.
ep	2 (44)	Error processing bit; set to 1 to notify calling program of error conditions. If ep bit is set to 0, the operator must correct fatal errors when they arise.
ℓ	2 (18-23)	FET length; FET first word address +5+ℓ = last word address +1. The minimum FET length is five words (ℓ = 0). If the minimum FET is used, only the logical file name, code and status field, FIRST, IN, OUT, and LIMIT are relevant. KRONOS does not set or check any other field. A length of six words is used if a working storage area is needed for blocking/deblocking. A length of eight words is used if the r bit is set, indicating a random file.

Parameter	Word (Bits)	Description
FNT pointer	5 (48-59)	KRONOS sets the address of the corresponding FNT/FST entry for the file in this byte when it processes a file action request for the file. If the FET is of minimum length, the system does not set this pointer.
FIRST	2 (0-17)	The first word address of the buffer area; routines that process I/O never change the value of FIRST.
IN	3 (0-17)	The next location to write data into the circular buffer; either CIO or the calling program changes IN depending on whether the request is for a read or a write operation. When IN=OUT, the buffer is empty. When IN=OUT-1, the buffer is full; that is, one location remains empty in a full buffer to distinguish it from an empty buffer. The upper bits of location 3 of the FET are zeros to eliminate the need for masking.
OUT	4 (0-17)	The next location to read or remove data from the circular buffer; either CIO or the calling program change OUT depending on whether the request is for a write or a read operation. The upper bits of location 4 of the FET are zeros to eliminate the need for masking.
LIMIT	5 (0-17)	The last word address plus one of the buffer area; no data is stored at address LIMIT. When LIMIT is reached, the next available address for reading or writing is FIRST. Routines that process I/O never change the value of LIMIT.
working storage fwa	6 (30-47)	First word address of a working storage area from which data can be blocked into or deblocked from the circular buffer.
working storage lwa + 1	6 (0-17)	Last word address plus one of the working storage area (above).
current random address	7 (36-59)	Specifies file position if the file is a mass storage random access file.
random request/ return information	7 (0-35)	Used for communication between the random access functions and the PPU I/O routines; initially set to zero.

FET CREATION MACROS

System macros in the COMPASS language facilitate generation of the system FET. The following paragraphs describe FET creation macros.

CODED FILE (SEQUENTIAL)

lfn FILEC fwa, f, (WSA = addr_w, l_w), UPR, EPR, (FET=l_f)

BINARY FILE (SEQUENTIAL)

lfn FILEB fwa, f (WSA = addr_w, l_w), UPR, EPR, (FET=l_f)

CODED FILE (RANDOM)

lfn RFILEC fwa, f, (WSA = addr_w, l_w), (IND = addr_i, l_i), UPR, EPR, (FET=l_f)

BINARY FILE (RANDOM)

lfn RFILEB fwa, f, (WSA = addr_w, l_w), (IND = addr_i, l_i), UPR, EPR, (FET=l_f)

The last five subfields (WSA, IND, UPR, EPR, and FET) are order-independent (order is fixed within the subfield). Upper case characters designate actual subfield content; lower case characters indicate parameters to be supplied by the user. All parameters except lfn, fwa, and f are optional.

lfn	File name
fwa	Substituted in FIRST, IN, and OUT
f	fwa + f substituted in LIMIT
WSA	Working storage area parameters
addr_w	First word address of working storage area
l_w	$\text{addr}_w + l_w$ = last word address + 1 of working storage area
IND	Index buffer parameters
addr_i	First word address of index buffer
l_i	Length of index buffer
UPR	User desires processing at End-of-Reel
EPR	User desires to handle error conditions
FET=l_f	Length of FET if desired

Examples:

To create a minimum FET for the standard INPUT file:

```
INPUT FILEC BUFFER, LBUFFER
```

To create an FET for a binary random file:

```
FILEABC RFILEB BUFFER, LBUFFER, (IND = INDEX, LINDEX)
```

USER/SYSTEM COMMUNICATION SUBROUTINES

The user communication subroutines provide the linkage between user programs and the operating system. All file action requests and job action requests are processed by library subroutines, which the system loads with the user program within the field length of the job. The program communicates with the subroutines through macro requests and the FET. The subroutines communicate with KRONOS through hardware registers and by setting and checking RA + 1.

An exit from the system subroutine returns control to the object program at the point following the macro request, and does not save registers. However, the macro descriptions include the register usage for the convenience of the user. If the user specifies a macro parameter value that is the same as the contents of the register by which the parameter is passed, KRONOS does not generate code to set the register to the parameter value. Therefore, suitable choice of registers improves the generated code.

JOB ACTION REQUESTS

Job action requests use the A1, X1, A6, X6 registers unless otherwise specified in the descriptions. Job action requests are:

- MEMORY
- RECALL
- MESSAGE
- ENDRUN
- ABORT
- TIME
- RTIME
- CLOCK
- DATE
- SYSTEM
- CONSOLE
- ONSW
- OFFSW
- EREXIT
- OVERLAY
- ROLLOUT
- MODE
- SETPR
- SETTL

When a job action request parameter represents an address expression, the parameter can be:

- a register name,
- a relocatable address,
- an external symbol name, or
- an absolute address.

MEMORY FUNCTION

Using the MEMORY function, the user's program can determine or change the length of central memory assigned to the job control point. Before issuing the MEMORY request, the user should establish a one-word parameter list at location stat:

	59	29	0
stat	0 0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0	
	words		

Set non-zero on completion.

Macro format:

```
LOCATION  OPERATION  ADDRESS
          MEMORY     type,stat,r,words
```

type   CM or null

stat   Address of status word

r   If non-null, indicates recall status

words   Desired new field length

RECALL FUNCTION

Using the RECALL function, the user can relinquish the CPU until:

- an I/O process is completed, or
- the next time through the monitor loop,

depending on the form of the request. If the lfn parameter is included in the macro call, control does not return to the relinquishing program until the completion bit in the FET for file lfn is set. If the lfn is not included in the macro call, the user relinquishes the CPU only until the next time through the monitor loop.

Macro format:

```
LOCATION  OPERATION  ADDRESS
          RECALL     lfn
```

lfn   Logical file name (optional)

MESSAGE FUNCTION

Using the MESSAGE function, the user can display a message on the console scope and enter it in the dayfile.

The maximum message length is 40_{10} characters. The message ends at the first word with all zeros in bits 0-11 or at the 40th character, whichever comes first. Before issuing the MESSAGE function, the user should pack his message (in display code) from higher-numbered bits to lower-numbered bits in sequential locations starting with location addr.

Macro format:

	LOCATION	OPERATION	ADDRESS
1	2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17	18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
		MESSAGE	addr,x,r

addr Beginning address of list containing the message

x Message route

x = 0; message to system dayfile

x = 1; display on line 1

x = 2; display on line 2

x = 3; message to user's dayfile only

r If non-null, indicates recall status

ENDRUN FUNCTION

By issuing an ENDRUN function, the applications program requests normal termination of a run. KRONOS examines the control card record of the job deck and begins execution with the next unused control card. If there are no more control cards, or if the next card is an EXIT card, the system terminates the job.

Macro format:

	LOCATION	OPERATION	ADDRESS
1	2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17	18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
		ENDRUN	

ABORT FUNCTION

When an error such as an out-of-bounds memory reference occurs, the user can request the monitor to terminate the job abnormally by using the ABORT function. If the control card section of the job deck contains an EXIT card, the system continues processing the job with the control card immediately following the EXIT card.

Macro format:

	LOCATION	OPERATION	ADDRESS
1	2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17	18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
		ABORT	

TIME FUNCTION

For the TIME function, the system returns the central processor time used by the job in location stat:

	59	35	11 0
stat	undefined	seconds	milliseconds

Macro format:

	LOCATION	OPERATION	ADDRESS
1	2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17	18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
		TIME	stat

stat Address to receive the CPU time

RTIME FUNCTION

For the RTIME function, the system returns the real-time clock reading in location stat.

	59	35	11 0
stat	undefined	seconds	milliseconds

Macro format:

	LOCATION	OPERATION	ADDRESS
1	2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17	18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
		RTIME	stat

stat Address to receive the clock reading

CLOCK FUNCTION

For the CLOCK function, the system returns current reading of the system clock in location stat:

	59					0
stat	hh	●	mm	●	ss	●

Macro format:

	LOCATION	OPERATION	ADDRESS
1	2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17	18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
		CLOCK	stat

stat Address to receive the clock reading

DATE FUNCTION

For the DATE function, the system returns the current date typed by the operator in location stat:

	59					0
stat	mm	/	dd	/	yy	●

Macro format:

	LOCATION	OPERATION	ADDRESS
1	2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17	18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
		DATE	stat

stat Address to receive the clock reading

SYSTEM FUNCTION

With the SYSTEM function, the user can request the system to:

- recall the central processor,
- terminate the current CPU program,
- abort the control point job, and
- return time information.

The SYSTEM function requires the X2 register.

Macro format:

LOCATION	OPERATION	ADDRESS
	SYSTEM	req, r, p1, p2

req	Three character system request
r	If non-null, indicates recall status
p1	Bits 0-17 of the request
p2	Bits 18-35 of the request

Example: If a user wishes to dump the contents of location 1000 to 4000 octal and recall the CPU when the dump is completed, the SYSTEM request is:

```
SYSTEM DMP, R, 4000B, 1000B
```

CONSOLE FUNCTION

The CONSOLE function sets the K display control word to the area defined by addr.

Macro format:

LOCATION	OPERATION	ADDRESS
	CONSOLE	addr

addr   Area containing address to receive the display control card

59	35	17 0
input buffer address	right screen buffer address	left screen buffer address

ONSW FUNCTION

The ONSW function sets the sense switches corresponding to bits 0-5 in n.

Macro format:

LOCATION	OPERATION	ADDRESS
	ONSW	n

n Word containing sense switch references in bits 0-5.

OFFSW FUNCTION

The OFFSW function clears the sense switches corresponding to bits 0-5 in n.

Macro format:

LOCATION	OPERATION	ADDRESS
	OFFSW	n

n Word containing sense switch references in bits 0-5.

EREXIT FUNCTION

The EREXIT function sets the error exit return address equal to addr.

Macro format:

LOCATION	OPERATION	ADDRESS
	EREXIT	addr

addr Address set to error exit return address.

return (RA) =

59	47	29	23 0
	Address	Error Flag	

OVERLAY FUNCTION

The OVERLAY function loads an overlay from the system library or from file n. If it is not a 0,0 level overlay, the overlay entry address is returned in X1.

Macro format:

LOCATION	OPERATION	ADDRESS
	OVERLAY	n,1,system,fwa

n Name of either a file or an overlay, depending on whether the system parameter is present.

1 Name of overlay to be loaded from file n if the system parameter is not present.

system Specifies that the overlay is to be loaded from the system library. When omitted, the overlay is loaded from file n.

fwa Specifies the address at which the overlay is to be loaded. When omitted, the overlay is loaded at the overlay origin.

ROLLOUT FUNCTION

The ROLLOUT function rolls out the user's job and releases all memory assigned to the job (except the control point area).

Macro format:

LOCATION	OPERATION	ADDRESS
	ROLLOUT	

MODE FUNCTION

The MODE function requests exit mode n.

Macro format:

	LOCATION	OPERATION	ADDRESS
		MODE	n

n	Exit Condition
0	Disable Exit mode; no selections made
1	Address is out of range because: ● Attempt was made to reference central memory or extended core storage outside established limits ● Word count in extended core storage communication instruction is negative ● Attempt was made to reference last 60-bit word (word 7) in relative address FL ECS
2	Operand out of range; floating-point arithmetic unit received an infinite operand
3	Address or operand is out of range
4	Indefinite operand; floating-point arithmetic unit received an indefinite operand

SETPR FUNCTION

The SETPR function sets a new priority level for a job.

Macro format:

	LOCATION	OPERATION	ADDRESS
		SETPR	p

p Priority level. $1 \leq n \leq 17$.

SETTL FUNCTION

The SETTL function permits the user to specify a new time limit for a job.

Macro format:

LOCATION	OPERATION	ADDRESS
	SETTL	t

t Central processor time limit in octal seconds

FILE ACTION REQUESTS

File action requests are:

- RETURN
- EVICT
- ASSIGN
- COMMON
- LOCK
- RELEASE
- RENAME
- REQUEST
- SETID
- STATUS
- UNLOCK
- READ
- RPHR
- READC
- READH
- READS
- READW
- WRITE
- WRITER
- WRITEF
- WPHR
- WRITEC
- WRITEH
- WRITES
- WRITEW
- SKIPF
- SKIPFF
- SKIPEI
- BKSP
- BKSPRU
- SKIPB
- SKIPFB
- REWIND
- UNLOAD

A file action request results in a return jump to a system subroutine. Subsequent actions depend on the state of the file. The subroutine can post a request to KRONOS. After KRONOS accepts the request, control returns to the calling program if the recall bit r is equal to zero, or after KRONOS completes the request if r is equal to one. For macro specifying the optional final recall parameter, r is set to one when recall is present (the recall parameter is non-null in the macro call). File action requests use the A1, X1, A2, X2, A6, X6, A7, X7 registers except when otherwise specified. The logical file name is always passed by the X2 register.

When a file action request parameter represents an address expression, the parameter may be:

- a register name,
- a relocatable address,
- an external symbol name, or
- an absolute address.

The logical file name is the base address of the FET that controls the file for which the operation is being requested.

RETURN FUNCTION

The RETURN function returns file lfn to the system. If lfn is a COMMON file on which the system allows other than read operations, RETURN makes the file available to other users.

Macro format:

	LOCATION	OPERATION	ADDRESS
1	2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17	18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
		RETURN	lfn,r

lfn	Logical file name
r	If non-null, indicates recall status

EVICT FUNCTION

The EVICT function releases to the system all space occupied by a file on disk. It is then available for use by either the releasing program or other programs. If the file is a COMMON file on which the system allows other than read operations, this function returns the file to the system. It is then available to other users.

Macro format:

	LOCATION	OPERATION	ADDRESS
1	2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17	18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
		EVICT	lfn,r

lfn	Logical file name
r	If non-null, indicates recall status

ASSIGN FUNCTION

The ASSIGN function assigns common file lfn to the job.

Macro format:

	LOCATION	OPERATION	ADDRESS
1	2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17	18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
		ASSIGN	lfn

lfn Logical file name

COMMON FUNCTION

The COMMON function enters file lfn as a COMMON file available to any succeeding job after the current job terminates.

Macro format:

	LOCATION	OPERATION	ADDRESS
1	2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17	18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
		COMMON	lfn

lfn Logical file name

LOCK FUNCTION

The LOCK function sets the write lockout bit for the specified local file.

Macro format:

	LOCATION	OPERATION	ADDRESS
1	2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17	18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
		LOCK	lfn, r

lfn Logical file name

r If non-null, indicates recall status

RELEASE FUNCTION

The RELEASE function releases file lfn according to the type specified.

Macro format:

	LOCATION	OPERATION	ADDRESS
1	2–10	11–17	18–36
		RELEASE	lfn,type,r

lfn Logical file name

type Indicates the type of release. If omitted, the system assumes type COMMON.

Type	Action Taken
COMMON	sets file type to local
PRINT	releases file to PRINT queue
PUNCH	releases file to PUNCH queue
PUNCHB	releases file to PUNCHB queue
P8	releases file to P8 queue

r If non-null, indicates recall status

RENAME FUNCTION

The RENAME function replaces the name of file lfn with the new name specified.

Macro format:

	LOCATION	OPERATION	ADDRESS
1	2–10	11–17	18–36
		RENAME	lfn,name,r

lfn Logical file name

name New name to be given lfn. If name is not present, the new file name is taken from address lfn+6

r If non-null, indicates recall status

REQUEST FUNCTION

The REQUEST function displays a message which asks the operator to assign equipment to file lfn.

Macro format:

	LOCATION	OPERATION	ADDRESS
1	2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17	18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
		REQUEST	lfn

lfn	Logical file name

SETID FUNCTION

The SETID function sets the identification code for the specified file.

Macro format:

	LOCATION	OPERATION	ADDRESS
1	2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17	18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
		SETID	lfn,n,r

lfn	Logical file name
n	Identification code for file lfn
r	If non-null, indicates recall status

STATUS FUNCTION

The STATUS function returns the status of the specified file.

Macro format:

	LOCATION	OPERATION	ADDRESS
1	2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17	18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
		STATUS	lfn,random,r

lfn	Logical file name
random	If random is present, the current file position is returned to (lfn+6) If omitted, the current status is returned in bits 0-8 of (FET)
r	If non-null, indicates recall status

UNLOCK FUNCTION

The UNLOCK function clears the lockout bit for the specified file.

Macro format:

	LOCATION	OPERATION	ADDRESS
1	2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17	18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
		UNLOCK	lfn,r

lfn	Logical file name
r	If non-null, indicates recall status

TRANSFER DATA REQUESTS

READ FUNCTION

The READ function reads information into the circular buffer. If there is room in the circular buffer for at least one physical record unit, the system initiates reading and continues until:

- the buffer is full,
- an End-of-Record or End-of-File mark appears, or
- an End-of-Information mark appears.

This status information is returned as follows:

- X1 contains 0 if a transfer is complete
- X1 contains -1 if an EOF mark was encountered
- X1 contains the address of the last word transferred into the working buffer if an EOR mark was detected before the transfer was completed

Data is not transferred after an EOR or EOF mark is encountered.

Bit 1 in word 1 of the FET determines the mode.

Macro format:

LOCATION	OPERATION	ADDRESS
	READ	lfn,r

lfn Logical file name

r If non-null, indicates recall status

RPHR FUNCTION

The RPHR function reads the next physical record on the input device into the circular buffer. Bit 1 in word 1 of the FET determines the mode.

If the physical record is too big for the circular buffer, the system writes as many words as possible into the buffer.

The system returns an End-of-File response if the physical record is an EOF mark. The buffer will then be empty.

Macro format:

LOCATION	OPERATION	ADDRESS
	RPHR	lfn,r

lfn Logical file name

r If non-null, indicates recall status

READC FUNCTION

With the READC function, the user can transfer one coded line from a file to a working buffer. The system transfers data up to the next End-of-Line mark (12-bit zero byte) or until the working buffer is full. When READC exits, (X7)=(B6)=last word of data transferred.

Macro format:

LOCATION	OPERATION	ADDRESS
	READC	lfn,buf,n

lfn Logical file name

buf Address of first working storage word to receive the coded line

n Number of words in working storage area

READH FUNCTION

With the READH function, the user can transfer one coded line from a file to a working buffer and fill in trailing spaces. The system transfers data up to the next End-of-Line mark (12-bit zero byte) or until the working buffer is full. When READH exits, (X7)=(B6)= last word of data transferred.

Macro format:

LOCATION	OPERATION	ADDRESS
	READH	lfn,buf,n

lfn Logical file name

buf Address of first working storage word to receive the coded line

n Number of words in working storage area

READS FUNCTION

With the READS function, the user can transfer coded data from a file to a working string buffer. The system unpacks the data and stores it one character per word in the string buffer. Zero characters are stored as blanks (55). If the coded line terminates (12-bit zero byte appears) before the specified number of characters have been transferred, the system fills the remaining words with space codes. When READS exits, (X7) = (B6) = last word of data transferred.

Macro format:

LOCATION	OPERATION	ADDRESS
	READS	lfn,buf,n

lfn	Logical file name
buf	Address of the first working storage word to receive the character string
n	Number of characters to be transferred

READW FUNCTION

With the READW function, the user can fill a working buffer from the specified file. When READW exits, (X7) = (B6) = last word of data transferred.

Macro format:

LOCATION	OPERATION	ADDRESS
	READW	lfn,buf,n

lfn	Logical file name
buf	Base address of working buffer
n	Number of words in working buffer

WRITE FUNCTION

This function writes information from the circular buffer if there is sufficient information in the buffer to fill one or more physical record units. Writing continues until the buffer is empty, or the buffer contains insufficient data to fill a PRU. Bit 1 in word 1 of the FET determines the mode.

Macro format:

LOCATION	OPERATION	ADDRESS
	WRITE	lfn,r

lfn	Logical file name
r	If non-null, indicates recall status

WRITER FUNCTION

The WRITER function is like a WRITE function, except that the system writes a short or zero length PRU at the end of the data containing an End-of-Record mark.

Macro format:

LOCATION	OPERATION	ADDRESS
	WRITER	lfn,r

lfn	Logical file name
r	If non-null, indicates recall status

WRITEF FUNCTION

The WRITEF function directs the system to write a logical EOF mark. When the user issues a WRITEF function, the system first writes any data present in the buffer and terminates it with an EOR mark.

Macro format:

LOCATION	OPERATION	ADDRESS
	WRITEF	lfn,r

lfn	Logical file name
r	If non-null, indicates recall status

WPHR FUNCTION

The WPHR function directs the system to write the information in the circular buffer as a single PRU on the output device. Bit 1 in word 1 of the FET determines the mode.

If the buffer contains less than one PRU, a WPHR function effects no action.

If the buffer contains more than one PRU when the user issues this request, the system writes the first PRU and sets the IN and OUT pointers to show that words remain in the buffer.

Macro format:

	LOCATION	OPERATION	ADDRESS
1	2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17	18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
		WPHR	lfn,r

lfn	Logical file name
r	If non-null, indicates recall status

WRITEC FUNCTION

With the WRITEC function, the user can transfer one coded line from a working buffer to the specified file. The system transfers data up to the next End-of-Line mark (12-bit zero byte). When WRITEC exits, (X7) = (B6) = last word of data transferred.

Macro format:

	LOCATION	OPERATION	ADDRESS
1	2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17	18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
		WRITEC	lfn,buf

lfn	Logical file name
buf	Address of first word of working storage that contains the coded line

WRITEH FUNCTION

With the WRITEH function, the user can delete trailing spaces and transfer one coded line from a working buffer to the specified file. When WRITEH exits, (X7) = (B6) = last word of data transferred.

Macro format:

	LOCATION	OPERATION	ADDRESS
1	2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17	18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
		WRITEH	lfn, buf, n

lfn Logical file name

buf Address of first word of working storage that contains the coded line

n Number of words in working storage area

WRITES FUNCTION

With the WRITES function, the user can transfer data from a working string buffer to the specified file. The system deletes trailing space codes and packs the characters, ten per word, before transferring them to the file. When WRITES exits, (X7) = (B6) = last word of data transferred.

Macro format:

	LOCATION	OPERATION	ADDRESS
1	2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17	18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
		WRITES	lfn, buf, n

lfn Logical file name

buf Address of first word of working string buffer that contains the coded data

n Number of characters to be transferred

WRITEW FUNCTION

With the WRITEW function, the user can transfer a certain number of words from a working buffer to the specified file. When WRITEW exits, (X7) = (B6) = last word of data transferred.

Macro format:

	LOCATION	OPERATION	ADDRESS
1	2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17	18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
		WRITEW	lfn, buf, n

lfn Logical file name

buf Base address of working buffer

n Number of words to be transferred

POSITION FILE REQUESTS

SKIPF FUNCTION

The SKIPF function directs the system to bypass one or more logical records in the forward direction. The user's program can issue this request at any point in a logical record. The n parameter of the macro call specifies the number of logical records to skip.

If the system finds an End-of-Information mark before it satisfies the macro request, the system will leave the file at the EOI mark and return EOI status.

Macro format:

	LOCATION	OPERATION	ADDRESS
1	2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17	18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
		SKIPF	lfn,n,r

lfn Logical file name

n Number of logical records to skip; $1 \leq n \leq 777777_8$
If n is null, number is assumed to be one (passed via X1)

r If non-null, indicates recall status

SKIPFF FUNCTION

The SKIPFF function directs the system to bypass one or more files in the forward direction. The user's program can issue this request at any point in a file. The n parameter in the macro call specifies the number of files to skip.

If the system finds an End-of-Information mark before it satisfies the macro request, the system will leave the file at the EOI mark and return EOI status.

Macro format:

	LOCATION	OPERATION	ADDRESS
1	2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17	18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
		SKIPFF	lfn,n,r

lfn Logical file name

n Number of files to skip (passed via X1)

r If non-null, indicates recall status

SKIPEI FUNCTION

The SKIPEI function directs the system to bypass, in the forward direction, all information in the specified file to an End-of-Information mark.

Macro format:

	LOCATION	OPERATION	ADDRESS
1	2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17	18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
		SKIPEI	lfn,r

lfn	Logical file name

r	If non-null, indicates recall status

BKSP FUNCTION

The BKSP function directs the system to bypass one logical record in the reverse direction. The request can be issued at any point in a logical record. The system will not backspace past the beginning of the file.

Macro format:

	LOCATION	OPERATION	ADDRESS
1	2 3 4 5 6 7 8 9 10	11 12 13 14 15 16 17	18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
		BKSP	lfn,r

lfn	Logical file name

r	If non-null, indicates recall status

BKSPRU FUNCTION

The BKSPRU function directs the system to bypass one or more PRUs in the reverse direction. The request can be issued at any point in a logical record.

If n appears, n PRUs are bypassed. If n is null, one PRU is bypassed. The system will not backspace past the beginning of the file.

Macro format:

LOCATION	OPERATION	ADDRESS
	BKSPRU	lfn,n,r

lfn Logical file name

n Number of PRUs to skip; if n is null, the number is assumed to be one (passed via X1)

r If non-null, indicates recall status

SKIPB FUNCTION

The SKIPB function directs the system to bypass one or more logical records in the reverse direction. The user's program can issue this request at any point in a logical record. The macro parameter n specifies the number of logical records to skip. If the system encounters a Beginning-of-Information mark before it satisfies the macro request, the file remains positioned at the Beginning-of-Information and returns Beginning-of-Information status. The system will not backspace past the beginning of the file.

Macro format:

LOCATION	OPERATION	ADDRESS
	SKIPB	lfn,n,r

lfn Logical file name

n Number of logical records to skip, $1 \leq n \leq 377777_8$
If n is null, the number is assumed to be one (passed via X1)

r If non-null, indicates recall status

SKIPFB FUNCTION

The SKIPFB function directs the system to bypass one or more files in the reverse direction. The user's program can issue this request at any point in a file. The macro parameter n specifies the number of files to skip. The system will not backspace past the beginning of the file.

Macro format:

LOCATION	OPERATION	ADDRESS
	SKIPFB	lfn,n,r

lfn	Logical file name
n	Number of files to skip (passed via X1)
r	If non-null, indicates recall status

REWIND FUNCTION

For a REWIND function, the system positions the file at the beginning of the first data record or at the beginning of the current reel. A REWIND function for a rewound file has no effect.

Macro format:

LOCATION	OPERATION	ADDRESS
	REWIND	lfn,r

lfn	Logical file name
r	If non-null, indicates recall status

UNLOAD FUNCTION

The UNLOAD function causes the same system action as REWIND function. If the file resides on magnetic tape, the tape is unloaded.

Macro format:

LOCATION	OPERATION	ADDRESS
	UNLOAD	lfn,r

lfn	Logical file name
r	If non-null, indicates recall status

LOADER OPERATION 5

The KRONOS Loader provides high-speed transfer from input or storage devices to central memory. The loader can be called by control cards or from the text of an object program.

The loader can load and link subprograms assembled or compiled independently (in absolute or relocatable binary) to one another or to library subprograms. The loader issues diagnostic messages via the dayfile and prints memory maps when requested.

The loader can generate overlays which are written to a specified file in absolute format. The system then loads these overlays without the extra memory and time required to process relocation and linkage.

These operations are governed by control cards, loader requests from object programs, and the standard relocatable subprogram format.

The loader has the following features:

1. A control card request from the user's program initiates the loader call.
2. Loader requests can be used to:
    - Prepare overlays and write them to a defined file
    - Load relocatable program texts
    - Load overlays
3. The loader can load programs from more than one file (including the system library) for a single job.
4. During loading, the loader links all external reference and entry points.
5. When the loader completes loading, it fills all unsatisfied references from the system library, a user file, or by a diagnostic routine that traces calls to such references.
6. During loading, the loader creates a memory map of all programs for which the user requests such a map.
7. The loader completes loading when an EXECUTE or NOGO card appears. NOGO inhibits program execution and is used primarily to provide a map of the program.

Subsequent loading following the NOGO begins as if no programs had been loaded prior to the NOGO command. An EXECUTE card transfers control directly to the loaded programs.

8. The loader does not prohibit loading of overlays by normal jobs, or the converse. However, the user must be extremely careful in the allocation of core and in communication between component programs. The loader also permits loading of absolute "original" code but complex situations can arise if it is in overlay format.

9. The system reduces the job field length allocation for the loader when the loader completes loading unless a user's request (NOREDUCE control card) prohibits this action.

OVERLAYS

The loader provides the facility to subdivide a large task into portions called overlays, and write them in absolute format. These overlays are then loaded at execution time without a relocatable loading operation. The loader generates overlays by processing control cards (loader directives).

An ordered pair of octal numbers ($0\text{-}77_8$) identifies each overlay. The ordered pair is written generally as (m, n). An overlay can be at one of three levels:

- main (0, 0),
- primary (i, 0), or
- secondary (i, j).

A main overlay has a designation in which m and n are zero. A primary overlay has a designation in which n is zero and m is non-zero. A secondary overlay has a designation in which both m and n are non-zero.

Only one overlay at each level resides in central memory at one time.

The main overlay (0, 0) can remain in memory throughout the job. Loading any other main overlay will destroy a previously loaded one.

Primary overlays all begin at the same point immediately following the main overlay (0, 0). The loading of a primary overlay will destroy any other primary overlay. All overlays in a particular program must have unique identifiers.

The origin of secondary overlays immediately follow the associated primary overlay and they can be loaded only by their primary overlay or by the main overlay. Loading a secondary overlay destroys any previously loaded secondary overlay.

When the loader detects illegal overlays during preparation (because of erroneous identification or size), it aborts the job.

The following diagram shows the storage allocation in core storage resulting from several overlay loading operations:

Ordinal of Loading Request	First Overlay Number	Second Overlay Number	Contents of User's Area After this Overlay has been Loaded		
1	0	0	(0, 0)	Must be the first overlay loaded	
2	1	0	(0, 0)	(1, 0)	
3	1	1	(0, 0)	(1, 0)	(1, 1)
4	1	3	(0, 0)	(1, 0)	(1, 3)
5	2	0	(0, 0)	(2, 0)	
6	2	2	(0, 0)	(2, 0)	(2, 2)
7	2	1	(0, 0)	(2, 0)	(2, 1)
8	4	0	(0, 0)	(4, 0)	

LOADER DIRECTIVES

The system interprets the following control cards as directives for loader execution. The user can intersperse control cards with tables but cannot intersperse them with cards making up a table.

OVERLAY CARD

Control card format: OVERLAY (lfn, a, b, Cnnnnnn)

lfn — Name of the file for which this overlay is to be written. The first overlay card must have an lfn. Subsequent cards can omit lfn, indicating that the overlays are related and are to be written in the same lfn. A different lfn on subsequent cards results in generation of overlays to the file having the new name.

a — First overlay number, in octal.

b — Second overlay number, in octal.

Cnnnnnn — Indicates the number (nnnnnn, in octal) of words from the start of blank common storage for the loader to load the designated overlay. This parameter is optional. This feature allows the user to change the size of blank common storage at execution time. This parameter cannot be included in a main overlay directive. If this parameter is omitted, the loader uses the previously declared size for blank common storage.

OVERLAY DECKS

The data (relocatable binary decks) immediately following the OVERLAY card up to the next OVERLAY card or an End-of-File mark, make up the overlay deck. When the loader has loaded the overlay deck, the loader satisfies the undefined external references from the system library. It writes the overlay and the overlay identification as the next logical record in the file.

Each overlay has a unique entry point which is the last transfer address (XFER) that the loader encounters in the overlay subprograms when it is preparing the overlay. External references that the loader cannot satisfy, even from the system library, result in job termination after the loading operation terminates and the loader has produced maps for all of the overlays. The user can reference entry points in the main overlay from primary and secondary overlays. He can reference entry points in a primary overlay only from an associated secondary overlay.

OVERLAY FORMAT

Each overlay consists of a logical record in absolute format. The first word is the identification word. Subsequent words through end-of-logical record are data words.

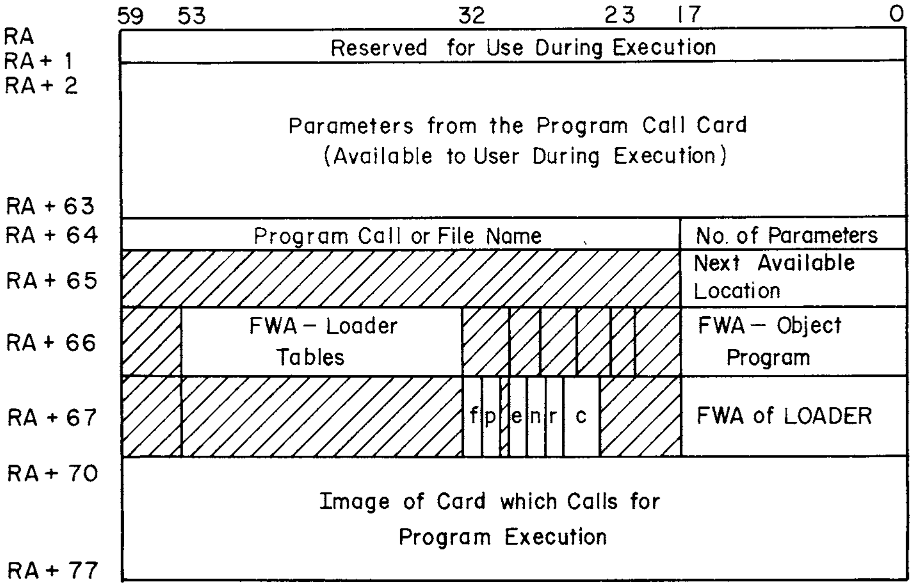
	59	47	41	33	17 0
First word:	5000	b	a	fwa	ea

a	First overlay number
b	Second overlay number
fwa	First word address of the overlay (the overlay is loaded starting at fwa)
ea	Address of the entry point to the overlay

MEMORY ALLOCATION

SYSTEM

The operating system requires storage space within the user's declared field length in contiguous memory locations. It automatically assigns the first 100_8 locations:


59
53
32
23
17
0
RA
RA+ 1
RA+ 2
Reserved for Use During Execution
Parameters from the Program Call Card
(Available to User During Execution)
RA + 63
RA + 64
Program Call or File Name
No. of Parameters
RA + 65
Next Available Location
RA + 66
FWA – Loader Tables
FWA – Object Program
RA + 67
f
p
e
n
r
c
FWA of LOADER
RA+ 70
Image of Card which Calls for Program Execution
RA+ 77

Parameter	Bit	Description
f	32	No Reduce FL bit; set if loader is not to reduce field length for the job.
p	31	Partial map bit; set if loader is to output a single line header instead of a complete map.
e	29	End-of-load bit; set if LDR has completed the requested load.
n	28	NOMAP flag; set if the loader is not to output a memory map.
r	27	RSS bit; set if operation is in RSS mode.
c	24-26	Code indicates last control card interpreted: 000 program call card 001 LOAD card 010 EXECUTE card 100 NOGO card

The system establishes loader tables at the high end of the user's field length. The user must provide space for the loader and the loader tables in his field length declaration. Blank common can overlay the loader tables. Conversely, if the user calls the loader again, they can overlay blank common. The user must assure that his field length is long enough to accommodate the loader tables and blank common if he wishes to preserve his data.

The system cannot guarantee that the applications program cannot destroy the loader tables.

USER

The loader assigns memory to subprograms and associated labeled common blocks as it encounters them. The loader preserves blank common relocation information until it has completed loading, at which time it allocates blank common following the last loaded program or labeled common block.

Declarations of blank common vary between subprograms, but the largest declaration determines the actual allocation.

MEMORY MAP

Following completion of the loading, the loader can produce a map of the user's area in the OUTPUT file. The map includes:

- Total length of all loaded programs and common blocks
- Length of the loader and its tables
- Names and locations of programs loaded, and the name of the file from which they are loaded
- Names and locations of common blocks
- Names and locations of entry points with a sublist of all programs referencing the entry point
- Unsatisfied external references

During generation of an overlay, the loader provides the record of a new overlay each time a call is made to the CPU loader program. The user can suppress the generation of this map by setting the NOMAP bit in the loader parameters to one.

DECK STRUCTURES 6

This section illustrates sample decks for:

- COMPASS language programs
- FORTRAN language programs

A card with a 7-8-9 punch in column 1 separates logical records. A card with a 6-7-9 punch in column 1 separates files. A card with a 6-7-8-9 punch in column 1 marks the End-of-Information. Figure 6-1 shows a basic job deck.

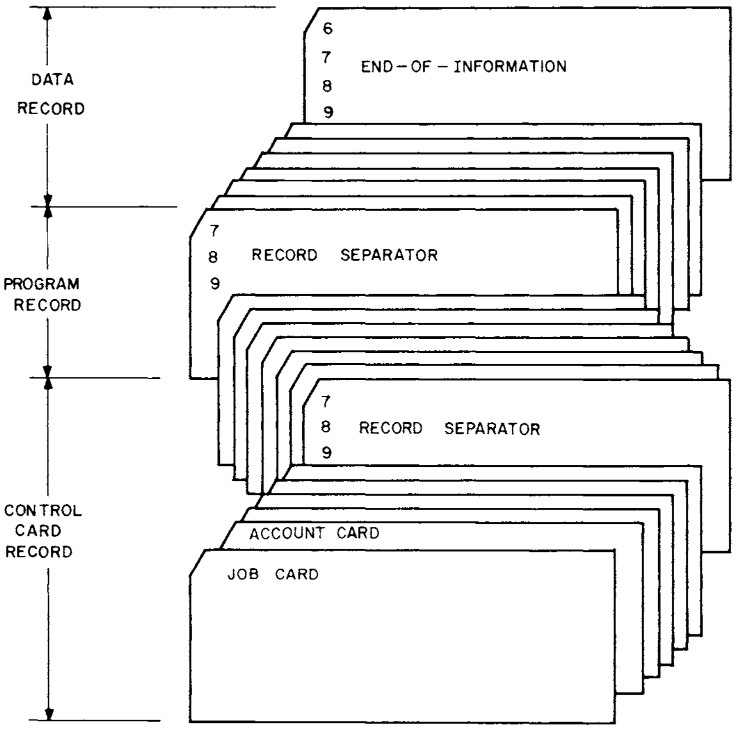


Figure 6-1. Basic Job Deck

COMPASS DECKS

Figures 6-2 and 6-3 illustrate typical COMPASS language decks. Figure 6-2 shows a deck for KRONOS to assemble with listing and binary output, and execute the subprogram using the included input data.

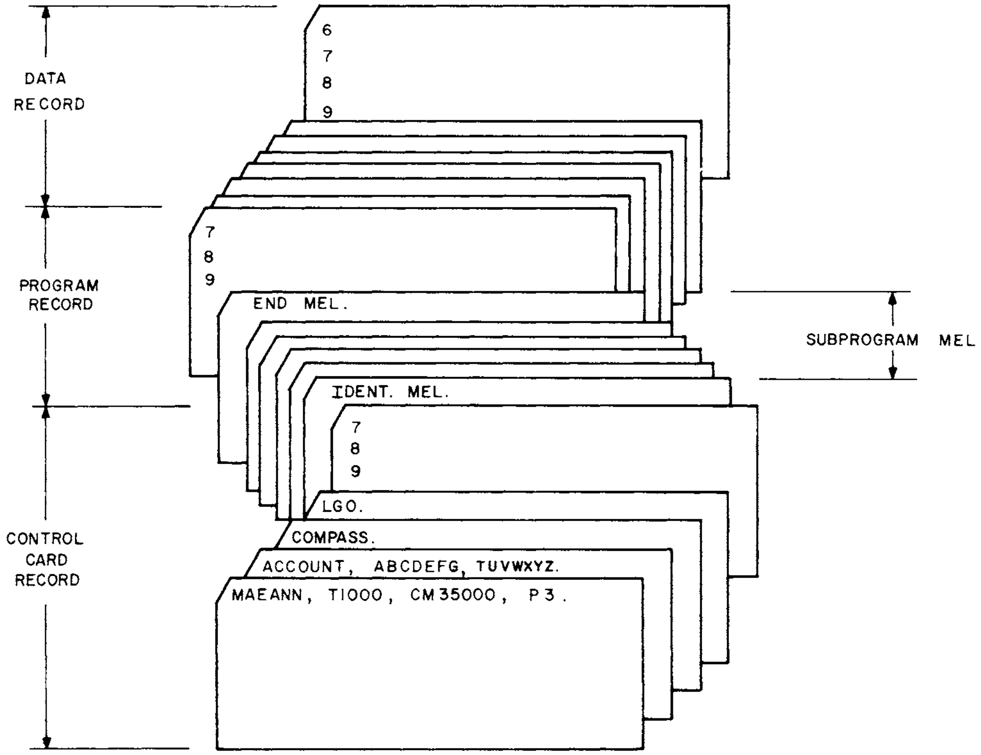


Figure 6-2. COMPASS Assemble and Execute Deck

Figure 6-3 shows a deck for KRONOS to assemble with listing and binary output, punch the binary output, and execute the first program.

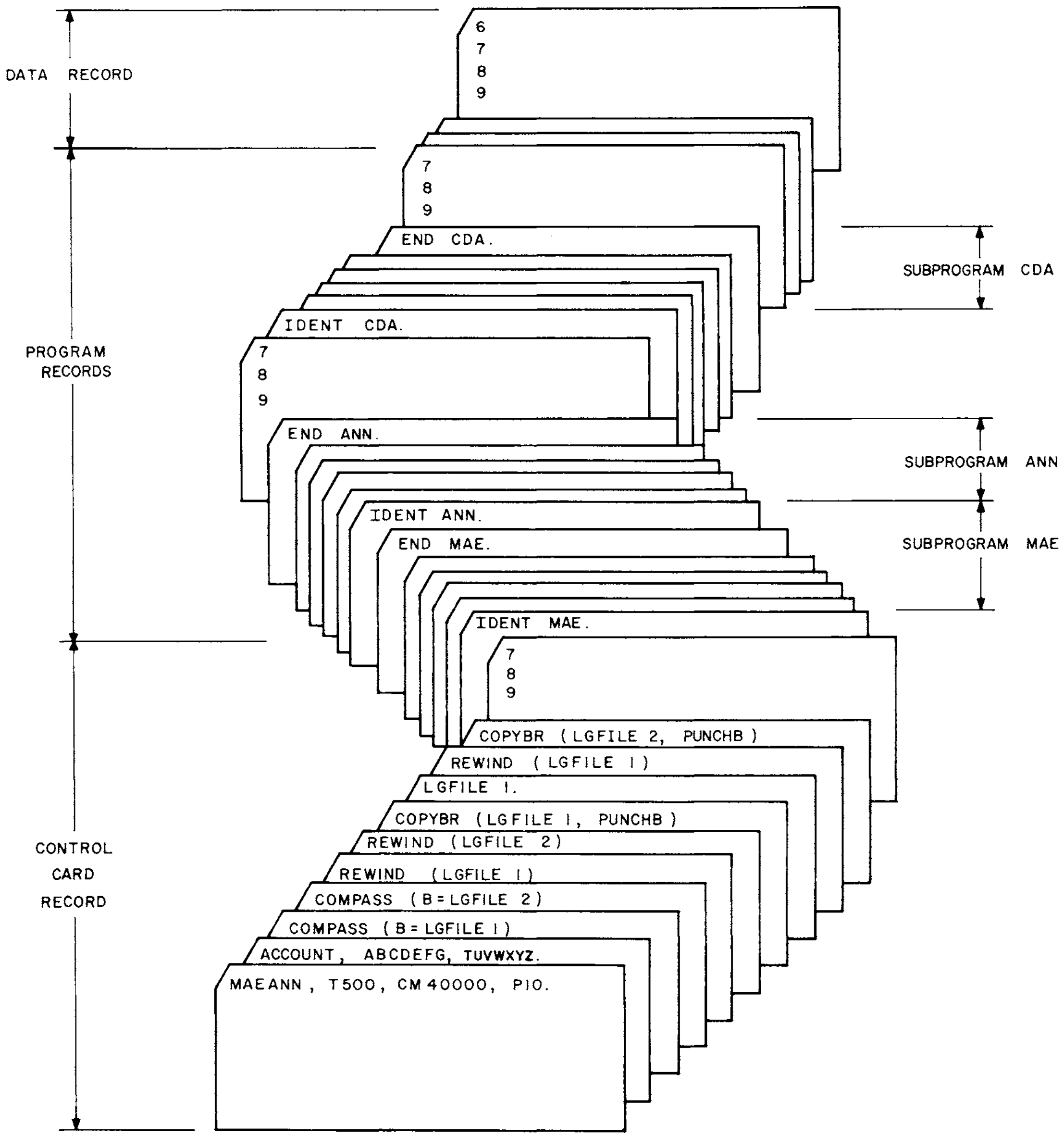


Figure 6-3. COMPASS Assemble, Execute and Punch Binary Output Deck

FORTRAN DECKS

Figures 6-4 through 6-6 illustrate various structures of FORTRAN language decks. These decks have various requirements for computation, user output, and overlay preparation.

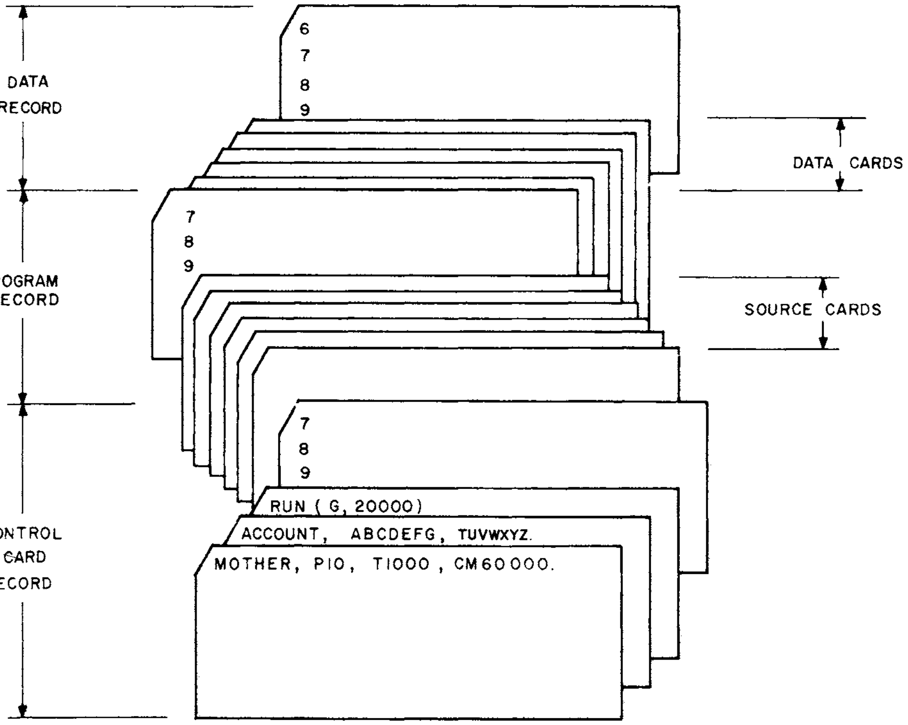
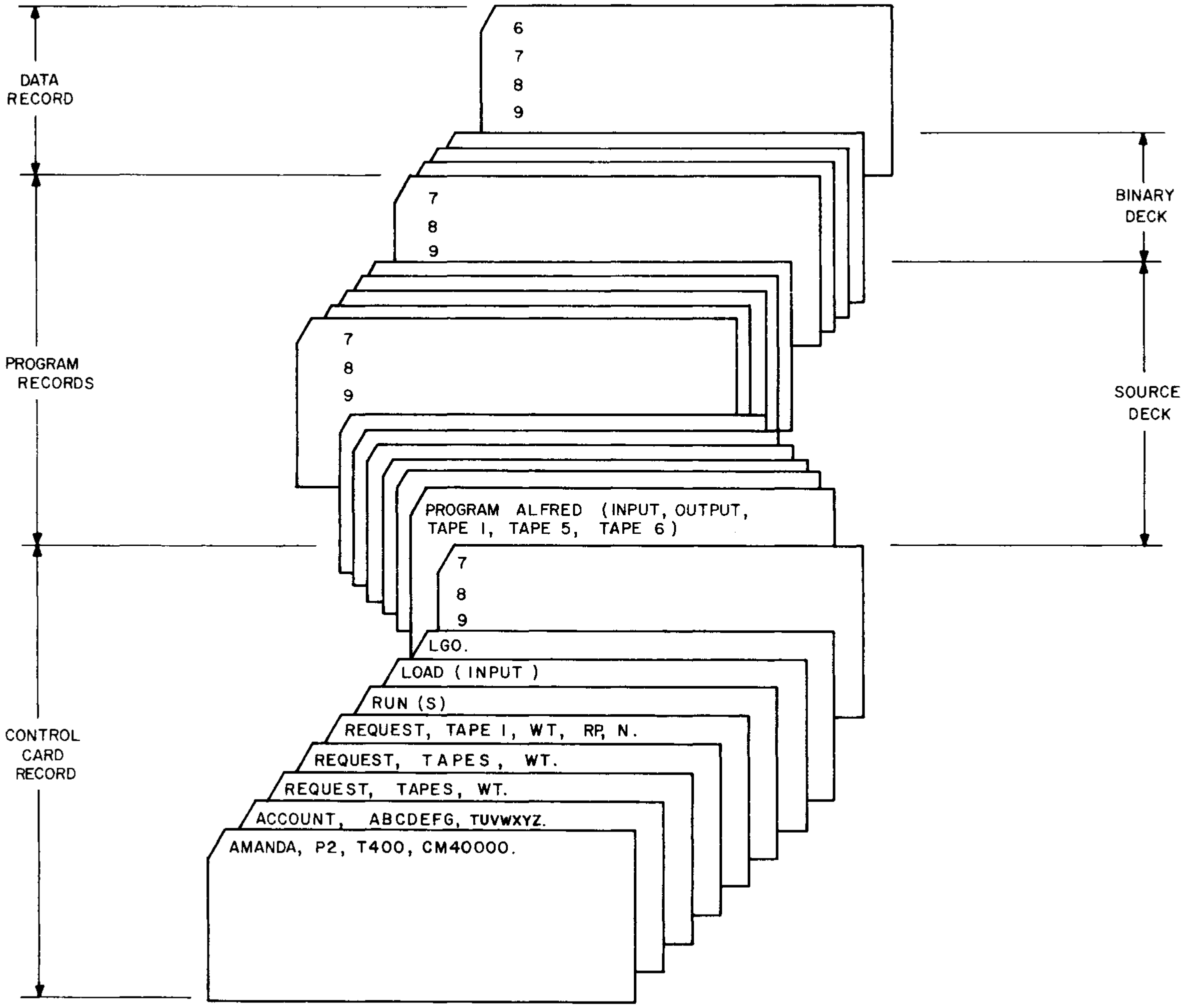


Figure 6-4. FORTRAN Compile and Execute Deck



```
PROGRAM ALFRED (INPUT, OUTPUT,
TAPE 1, TAPE 5, TAPE 6)
LGO.
LOAD (INPUT)
RUN (S)
REQUEST, TAPE 1, WT, RP, N.
REQUEST, TAPES, WT.
REQUEST, TAPES, WT.
ACCOUNT, ABCDEFG, TUVWXYZ.
AMANDA, P2, T400, CM40000.
```

Figure 6-5. FORTRAN Load and Run Deck

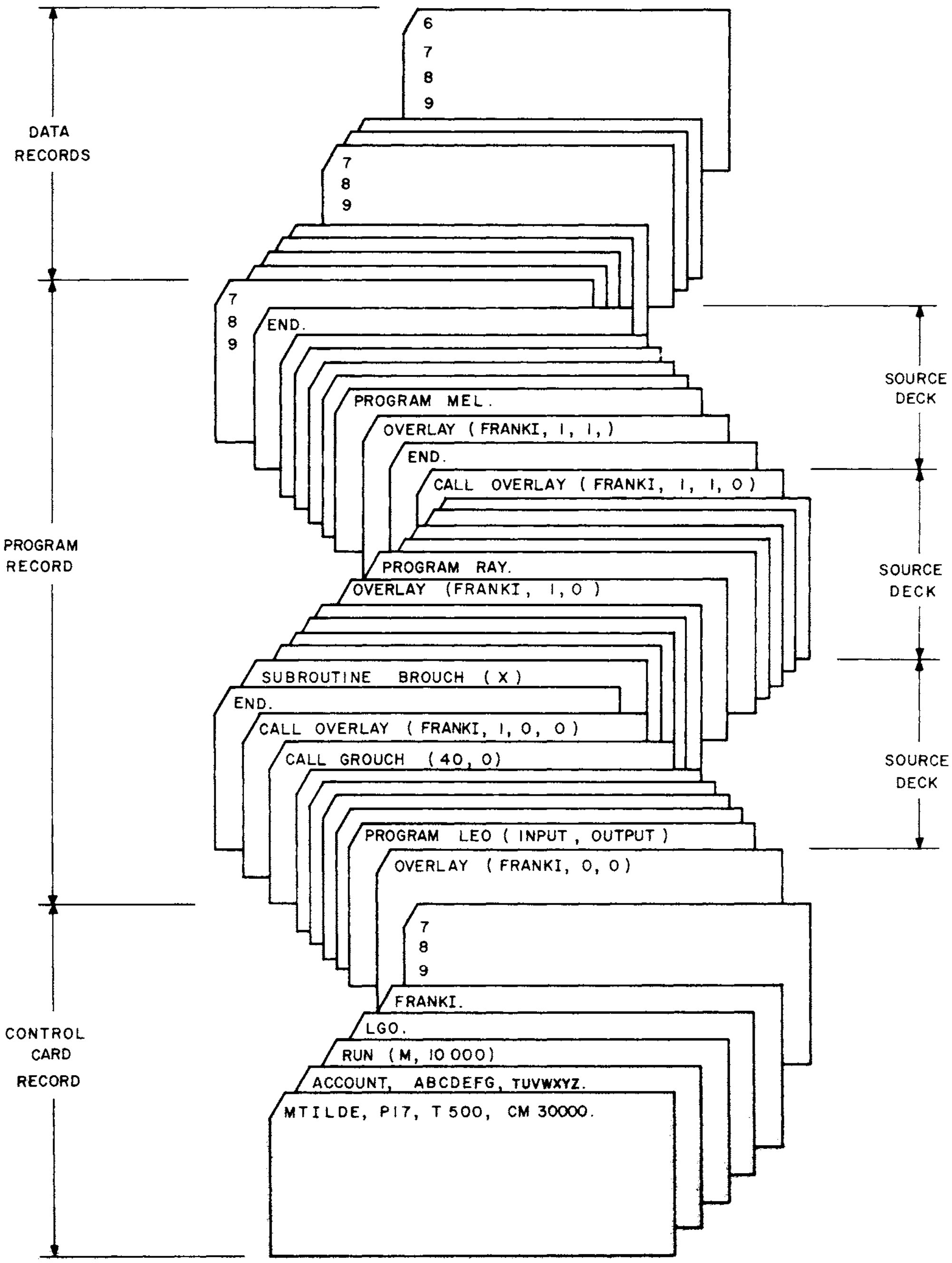


Figure 6-6. FORTRAN Overlay Preparation Deck

UTILITY PROGRAMS 7

The KRONOS library contains a set of PPU and CPU utility programs which the user can call with control cards. The system provides for card-to-tape, tape-to-tape, tape-to-print, card-to-central storage, central storage-to-punch, file editing, and general file manipulation operations. KRONOS performs utility operations on named files, each of which designates a specific peripheral device such as a card reader, tape unit, printer, card punch, or disk.

Before the first reference to any named file, the operator assigns a device to it with the ASSIGN command in response to a REQUEST control card, or by an ASSIGN card; otherwise, the system assigns the file to a disk unit. All files, except those on disk, specify a unique peripheral device and all references to the specific device are made through the file name.

Utility jobs conform to the normal deck structure (Section 6). If the user has only utility programs to run, the job card can specify a short field length. In all copy operations, the system automatically sets up the central memory buffer to use the entire field length of the job. Some operations between high-speed devices run faster with a large buffer.

The user's program should request the operator to assign equipment to all necessary files which do not reside on the disk. The system will rewind and position tapes upon request. The user can call each utility program by specifying its name, starting in column 1. Parameters for execution of the program appear in parentheses after the name.

LIBRARY ROUTINES

GENERATE USER LIBRARY FILE (LIBGEN)

Control card format: LIBGEN (L=library, N=name, I=input)

library — Name of library file; if undefined, the system assumes LIBRARY (file is not rewound).

name — Name of file identifier; this parameter is necessary if the user library file is to reside on the system library tape. If undefined, the system assumes ULIB.

input — Name of file containing input data; if undefined, system assumes LGO (file is rewound).

With the LIBGEN card, a user can generate a library file. The input file supplies the subprograms to be included in the file.

LIBGEN reads the input file and creates a directory of entry points. When a zero-length record or End-of-File mark appears, LIBGEN terminates the directory and rewinds the input file. LIBGEN then writes the directory and copies the input file to the library file. A zero-length record followed by a file mark terminates the library file.

EDIT USER LIBRARY FILE (LIBEDIT)

Control card format: LIBEDIT (p_1, p_2, . . . , p_n)

p_i Parameters; can be any of the following in any order:

Option	Specify
I = 0	Use no correction file input
I = fname	Use file fname for correction file input
OLD = 0	Use no old file
OLD = fname	Use file fname for old file
NEW = fname	Use file fname for new file
L = 0	Print no correction listing
L = 1	Print only the changes made
LO = fname	Use file fname as output file
LGO = 0	Generate no default correction file
LGO = fname	Use file fname for default correction file
C	Copy file NEW to OLD
V	Verify files OLD and NEW
R	Do not rewind OLD and NEW

By calling the LIBEDIT utility program, the user can edit a library file. Specific LIBEDIT directives, which appear in the program record, include:

*INSERT	*RENAME
*DELETE	*REPLACE
*IGNORE	*COPY
*ADD	*BEFORE

See the KRONOS external documentation for the current capabilities of these directives.

CONVERT UPDATE LIBRARY TO MODIFY LIBRARY (UPMOD)

Control card format: UPMOD (p_1, p_2, . . . , p_n)

p_i Parameters; can be any of the following in any order:

Option	Specify
P	UPDATE program library is on file OLDPL
P = fname	UPDATE program library is on file fname
N	MODIFY program library is on file OPL
N = fname	MODIFY program library is on file fname
M	MODIFY program library name is OPL
M = ℓname	MODIFY program library name is ℓname
F F=x	Convert to file mark

The UPMOD utility program converts program libraries that have been maintained by UPDATE to program libraries that can be maintained by MODIFY. Unless the F option is selected, UPMOD converts one logical record. For parameters omitted from the control card, the system assumes:

P = OLDPL
N = OPL
M = OPL

CATALOG FILE SET (CATALOG)

Control card format: CATALOG (files, N=n, L=list)

files	Name of file set to be cataloged
n	Number of files to catalog; if n=0, all files are cataloged until an empty file is found
list	Name of file to receive catalog information

The CATALOG card lists catalog information on file "catalog" for n files of the file set. For each file, CATALOG lists the:

- file name,
- current file number,
- record number,
- record type,
- record length, and
- checksums for the records.

The listing for each file begins on a new page.

COMPARE RECORDS (VERIFY)

Control card format: VERIFY ($file_1$, $file_2$, p_1, p_2, . . . , p_n)

$file_1$	Name of first file; if omitted, the system assumes TAPE1.
$file_2$	Name of second file; if omitted, the system assumes TAPE2.
p_i	Parameters; can be any of the following:

Option	Specify
N = x	Verify x files. If x = 0, terminate operation on the first empty file from either medium. If N is omitted, the system assumes N = 1.
E = x	List first x errors in comparison; if omitted, the system assumes E = 100.
L = fname	List output on file fname; if omitted, the system assumes L=OUTPUT.
A	Abort if errors occur. If omitted, the system will not abort.

The VERIFY program compares data on two files, word for word, in Binary mode. Whenever words on the two files do not match, VERIFY lists:

- the record number,
- the word number within the record, and
- the words from both media that do not match.

The E option can save the system much wasted effort if the files are drastically dissimilar.

COMPARE LIBRARY FILES (VFYLIB)

Control card format: VFYLIB (old, new, output, r)

old	Name of old library file; if omitted, the system assumes OLD
new	Name of new library file; if omitted, the system assumes NEW
output	Name of file to receive output; if omitted, the system assumes OUTPUT
r	If set, old and new are not rewound.

The VFYLIB program compares an old library file with a new one and lists:

- replacements,
- deletions,
- insertions, and
- changes in residence

on the output file.

EXTRACT EXTERNAL DOCUMENTATION (DOCEXT)

Control card format: DOCEXT (sfile, ℓfile)

sfile	Name of program source file; if omitted, the system assumes COMPILE
ℓfile	Name of file on which to list documentation; if omitted, the system assumes OUTPUT

The DOCEXT program extracts the external documentation — the lines that begin with three asterisks and all contiguous comment lines — from the source file and lists the documentation on the output file.

GENERATE SYSTEM SYMBOL CROSS REFERENCE (SYSREF)

Control card format: SYSREF (p_1, p_2, . . . ,p_n)

p_i Parameters; can be any of the following:

Option	Specify
P	Old program library input from file OPL.
P = fname	Old program library input from file fname
L	List output on file OUTPUT
L = fname	List output on file fname
S	System text from overlay SYSTEXT
S = ℓname	System text from overlay ℓname

The SYSREF program generates a cross reference list of system symbols used by decks on a specified old program library file.

COPY ROUTINES

COPY THROUGH EMPTY FILE (COPY)

Control card format: COPY ($file_1$, $file_2$, x)

$file_1$	Name of file to copy from
$file_2$	Name of file to receive copy
x	If present, rewind and verify both files

The COPY routine copies $file_1$ to $file_2$ through an empty file in $file_1$. It then backspaces $file_2$ over the last file mark. If the file names are omitted, the system assumes INPUT, OUTPUT.

This routine can be used to copy a tape even if the number of files on the tape is not known, providing the tape terminates with an empty file.

COPY BINARY FILE (COPYBF)

Control card format: COPYBF (file$_1$, file$_2$, n)

file$_1$ Name of file to copy from

file$_2$ Name of file to receive copy

n Number (decimal) of binary files to copy; if absent, the system assumes n=1

The COPYBF routine copies n binary files from file$_1$ to file$_2$. If the first and second parameters are omitted, the system assumes INPUT, OUTPUT.

COPY BINARY RECORD (COPYBR)

Control card format: COPYBR (file$_1$, file$_2$, n)

file$_1$ Name of file to copy from

file$_2$ Name of file to receive copy

n Number (decimal) of binary records to copy; if absent, the system assumes n=1

The COPYBR routine copies n binary records from file$_1$ to file$_2$. If the first and second parameters are omitted, the system assumes INPUT, OUTPUT.

The COPYBR operation terminates when COPYBR has read the required number of records (n).

COPY CODED FILE (COPYCF)

Control card format: COPYCF (file$_1$, file$_2$, n, fchar, ℓchar)

file$_1$ Name of file to copy from

file$_2$ Name of file to receive copy

n Number (decimal) of BCD files to copy; if absent, the system assumes n=1

fchar First character of each line to copy; if omitted, the system assumes fchar=1

ℓchar Last character of each line to copy; if omitted, the system assumes ℓchar=136

The COPYCF routine copies n coded (BCD) files from file$_1$ to file$_2$. If the first and second parameters are omitted, the system assumes INPUT, OUTPUT. If the first and second parameters are the same, the system skips n files.

COPY CODED RECORD (COPYCR)

Control card format: COPYCR ($file_1$, $file_2$, n, fchar, ℓchar)

$file_1$	Name of file to copy from
$file_2$	Name of file to receive copy
n	Number (decimal) of coded records to copy; if absent, the system assumes n = 1
fchar	First character of each line to copy; if omitted, the system assumes fchar = 1
ℓchar	Last character of each line to copy; it omitted, the system assumes ℓchar = 136

The COPYCR routine copies n logical records from $file_1$ to $file_2$. If the first and second parameters are omitted, the system assumes INPUT, OUTPUT. If the first and second parameters are the same, the system skips n records.

The COPYCR operation terminates when COPYCR has read the required number of records (n).

COPY SHIFTED BINARY FILE (COPYSBF)

Control card format: COPYSBF ($file_1$, $file_2$, n)

$file_1$	Name of file to copy from
$file_2$	Name of file to receive copy
n	Number (decimal) of coded records to copy; if absent, the system assumes n = 1

The COPYSBF routine copies n files of coded information from $file_1$ to $file_2$, shifting each line one character and adding a leading space. If the parameters are omitted, the system assumes INPUT, OUTPUT, 1.

This routine is used to format a print file where the first character of each line is not a control character and is to be printed. The added space character results in single line spacing and a page eject at the beginning of each record when the system prints the file.

COPY TO TERMINATOR (COPYX)

Control card format: COPYX ($file_1$, $file_2$, term, bksp)

$file_1$	Name of file to copy from; if undefined, system assumes INPUT
$file_2$	Name of file to receive copy; if undefined, system assumes OUTPUT
term	Terminator type: term = n; copy n records term = 00; copy to zero-length record term = cccccc; copy until record named cccccc appears If term is undefined, the system assumes term = 1

bksp	Backspace command:
	bksp = 0; no backspace
	bksp = 1; backspace $file_1$
	bksp = 2; backspace $file_2$
	bksp = 3; backspace both files
	If bksp is undefined, the system assumes bksp = 0

COPYX copies binary records to the specified terminator and then backspaces the file or files according to the parameters given.

FILE MANIPULATION ROUTINES

UNLOAD FILE (UNLOAD)

Control card format: UNLOAD ($file_1$, $file_2$, . . . , $file_n$)

The UNLOAD routine rewinds and unloads $file_1$, $file_2$, . . . , $file_n$ but does not dissociate the files from the control point.

REWIND FILE (REWIND)

Control card format: REWIND ($file_1$, $file_2$, . . . , $file_n$)

The REWIND routine rewinds $file_1$, $file_2$, . . . , $file_n$.

SKIP TO END-OF-INFORMATION (SKIPEI)

Control card format: SKIPEI ($file_1$)

$file_1$ Name of the file to be positioned; if omitted, the system assumes FILE.

The SKIPEI card directs the system to reposition $file_1$ to the End-of-Information symbol. This card can only be used with mass storage devices.

SKIP FILE FORWARD (SKIPF)

Control card format: SKIPF ($file_1$, n)

$file_1$ Name of file set to be positioned; if omitted, the system assumes FILE

n Number (decimal) of files to skip; if omitted the system assumes n = 1

The SKIPF card directs the system to bypass one or more files, in the forward direction to $file_1$.

SKIP FILE BACKWARD (SKIPFB)

Control card format: SKIPFB ($file_1$, n)

$file_1$ Name of file set to be positioned; if omitted, the system assumes FILE

n Number (decimal) of files to skip; if omitted, the system assumes n=1

The SKIPFB card directs the system to bypass one or more files, in the reverse direction, to $file_1$.

SKIP RECORD FORWARD (SKIPR)

Control card format: SKIPR ($file_1$, n)

$file_1$ Name of file to be positioned; if omitted, the system assumes FILE

n Number (decimal) of logical records to skip; if omitted, the system assumes n=1

The SKIPR card directs the system to position $file_1$ forward n logical records.

BACKSPACE LOGICAL RECORDS (BKSP)

Control card format: BKSP (file_1, n)

file_1 Name of file to be backspaced; if omitted, the system assumes FILE

n Number (decimal) of logical records to backspace; if omitted, the system assumes n=1

The BKSP routine backspaces file_1 n logical records. Backspacing terminates if file_1 becomes rewound.

RETURN FILES (RETURN)

Control card format: RETURN (file_1, file_2, . . . , file_n)

file_i Names of files to be returned

The RETURN card returns files to the system. If a file named is a COMMON file on which the system allows other than read operations, it becomes available to other users. Otherwise, the system disposes of the files according to file types.

INPUT/OUTPUT ROUTINES

LOAD BINARY CORRECTIONS (LBC)

Control card format: LBC, addr. or
LBC.

addr Address (octal) relative to RA at which binary load begins; if absent, LBC begins at RA

The user can call the LBC peripheral program with a control card. LBC reads binary corrections from the INPUT file and enters them in central storage. If addr is specified in the program call, binary cards are loaded beginning with that address; otherwise, loading begins at the reference address. LBC reads only one record from the INPUT file. The user must make an LBC call for each record of data to be loaded. This program is intended for loading cards punched by the PBC routine.

LOAD OCTAL CORRECTIONS (LOC)

Control card format: LOC (fwa, lwa) or
LOC (lwa) or
LOC.

fwa Address (octal) relative to RA at which octal load begins; if absent, LOC begins at RA

lwa Last word address plus one (octal) of the buffer area in which to load octal cards; if absent, LOC reads all correction cards in the next INPUT file record

The user can call the LOC peripheral program with a control card. LOC reads octal corrections from the INPUT file and enters them in central storage. The octal correction cards must be in the following format:

1	7
23001	45020 04000 00042 00044

Address begins in column 1; leading zeros can be omitted. The data word begins in column 7; spacing in the data word is not important but the word must contain 20 digits.

PUNCH BINARY CARDS (PBC)

Control card format: PBC (fwa, lwa) or
PBC (lwa) or
PBC.

fwa Address (octal) relative to RA at which the binary deck begins; if absent, PBC uses RA

lwa Last word address plus one (octal) of the binary deck; if absent, RA contains the number of words in the lower 18 bits

The PBC routine punches a deck of binary cards directly from central memory and does not modify central memory.

READ BINARY RECORD (RBR)

Control card format: RBR, n.

n Specifies fifth character of file name (1-7); first four characters are TAPE

The RBR routine loads one binary record from the file specified by the user. Loading begins at RA. RBR uses central memory locations FL-5 through FL-1 for buffer parameters and destroys the original contents of these locations.

WRITE BINARY RECORD (WBR)

Control card format: WBR, n.rl.

n Specifies fifth character of file name (1-7); first four characters are TAPE

rl Record length in words; if absent, length is taken from the lower 18 bits of RA

The WBR routine writes a binary record from central memory to the file specified by the user. WBR begins writing from RA. WBR uses central memory locations FL-5 through FL-1 for buffer parameters and destroys the original contents of these locations.

Example:

To write a program on tape after patching it:

```
REQUEST TAPE5.
REQUEST TAPE2.
REWIND (TAPE5).
REWIND (TAPE2).
RBR, 5.
LOC.
WBR, 2.
```

WRITE FILE MARKS (WRITEF)

Control card format: WRITEF (file$_1$, n)

file$_1$ Specifies the file to be written on

n Number of file marks; if omitted, the system assumes n=1

The WRITEF card instructs the system to write n file marks on a file.

WRITE EMPTY RECORDS (WRITER)

Control card format: WRITER (file$_1$, n)

file$_1$ Specifies the file to be written on

n Number of blank records; if omitted, the system assumes n=1

The WRITER card instructs the system to write n empty records on a file.

REQUEST FIELD LENGTH (RFL)

Control card format: RFL, nfl

nfl	New field length (octal)

The RFL routine changes the field length for the execution of a program. This routine is also used internally by the RUN compiler. For a short 5000_8 word program, storage would be used most efficiently by calling RFL.

DUMP STORAGE (DMP)

Control card format: DMP (fwa, lwa) or
DMP (lwa) or
DMP.

fwa First word address of memory to be dumped; if five digits are prefixed by a 4, fwa is absolute first word address; otherwise, fwa is relative to RA. If fwa is absent, Dump mode depends on the presence or absence of lwa

lwa Last word address plus one of memory to be dumped; if five digits are prefixed by a 4, lwa is absolute last word address plus one; otherwise, lwa is relative to RA. If lwa alone is present, DMP assumes fwa = RA. If neither fwa or lwa is present, DMP dumps the exchange package and 100 locations before and after the stop location

The user can call the DMP peripheral program with a control card. The DMP routine dumps central memory according to the DMP call parameters.

I/O CODES

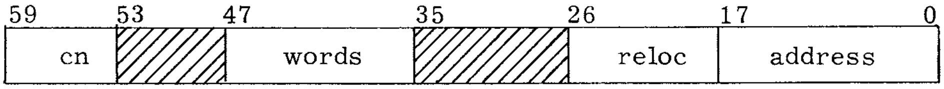
Code	Macro	Order
00	RPHR	Read one PRU from file
04	WPHR	Write one PRU to file
10	READ	Read to CM
14	WRITE	Write until short PRU
24	WRITER	Write to End-of-Record
30	SKIPFF	Skip forward n logical files
34	WRITE	Write a file mark
40	BKSP	Backspace one logical record
44	BKSPRU	Backspace n logical PRUs
50	REWIND	Rewind file
60	UNLOAD	Unload tape file
70	RETURN	Return file to system
114	EVICT	Release allocatable storage
130	SKIPFB	Skip backwards n logical files
200	—	Read to End-of-Information
240	SKIPF	Skip forward n logical records
640	SKIPB	Skip backwards n logical records

RELOCATABLE SUBROUTINE FORMATS B

In order to be externally compatible with SCOPE versions of COMPASS and other language translators, KRONOS subscribes to the SCOPE relocatable subroutine format. Hence, the logical record of output (subroutine) consists of an indefinite number of tables. Each table in this appendix is a subdivision of a logical record.

IDENTIFICATION WORD

The first word of each table identifies the table to the system. That is, it indicates the kind of information that the table contains. The format of the identification word is:



The parameters of the identification word are:

TABLE B-1. IDENTIFICATION WORD PARAMETERS

Code Number (cn)	Table	reloc	address	words
34	Program Identification and Length	not used	0	Number of words in table (not counting identification word)
36	Entry Point	not used	not used	
40	Text	reloc=0, relative to RA reloc=1, relative to program origin reloc=3-77$_8$, relative to labeled common block M, where M is in position LR-2 of LCT	load address	
42	Fill	0	0	
43	Replication	not used	not used	
44	Link	not used	0	
46	Transfer	not used	not used	
77	Prefix	not used	not used	

PROGRAM IDENTIFICATION AND LENGTH (PIDL) TABLE

	59	17 0
word 1	name of subprogram	pl

name of subprogram May be 7 display code blank characters

pl Program length

	59	17 0
words 2 - wc	name of common block	bl

wc Word count

name of common block May be 7 display code blank characters

bl Block length

If wc = 1, no common references appear in the program. The subprogram length is relevant only in the first Program Identification and Length Table. All PIDL tables must appear prior to the display of any other tables associated with a given subprogram. The names of common blocks cannot be duplicated in a PIDL table. The list of common block names is called the Local Common Table (LCT). Since the relocation of addresses relative to common blocks is designated by the positions in the LCT, the order of the common block names is significant. (The first word in the LCT is referred to as position 1.)

ENTRY POINT (ENTR) TABLE

The Entry Point (ENTR) Table contains the names of the entry points to the subprogram and its associated labeled common blocks. The ENTR Table must immediately follow the PIDL tables.

	59	26	17 0
words 1 - wc	name of entry point		▨
	▨	rl	loc

rl Relocation of the address specified by LOC

0 absolute, relative to RA (no relocation)

1 program address

$3\text{-}77_8$ relative to common block M, where M is in position rl-2 of LCT (M must not refer to blank common)

loc Address of data word to be modified

TEXT TABLES

Text tables, which can appear in any order and any number, contain subprogram data and relocation information for the data. Each table includes an origin for the data, the data itself, and relocation indicators.

The relocation word (first word), which is a series of 4-bit bytes, describes relocation of the three possible memory address references in a 60-bit data word. Relocation is determined relative to either the program origin or the complement of the program origin (negative relocation). The value and relocation for each byte is as follows:

000X	no relocation
10XX	upper address, program relocation
11XX	upper address, negative relocation
010X	middle address, program relocation
011X	middle address, negative relocation
1X10	lower address, program relocation
1X11	lower address, negative relocation
0010	same as 1X10
0011	same as 1X11

These designations permit independent and simultaneous relocation of both upper and lower addresses.

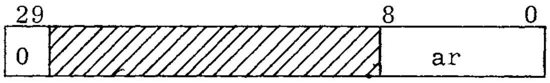
Data words (words 2-wc, wc $\leq 20_8$) are loaded consecutively beginning at the load address L, and their addresses are relocated as specified by the corresponding byte in the relocation word.

All addresses are relocated absolutely or relative to the program origin. They are never relocated relative to a labeled common block.

FILL TABLE

Words 1-wc are partitioned into sets of 30-bit contiguous bytes consisting of one control byte followed by an indefinite number of data bytes. The last byte can be zero. The control byte holds information concerning each of the subsequent data bytes until another control byte is encountered.

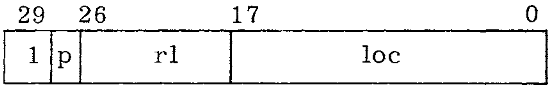
A zero byte is treated as a control byte in the format:



ar — Relocation of the value in address position of word specified in succeeding data bytes:

0	absolute, relative to RA (no relocation)
1	program relocation
2	negative relocation
3-77_8	relative to common block M, where M is in position ar-2 of LCT

The data byte format is:



p — Position within word of address specified by rl and loc

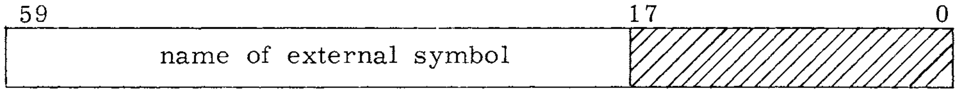
10	upper
01	middle
00	lower

rl — Relocation of address specified by loc

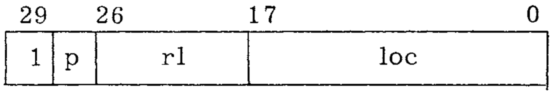
loc — Address of data word to be modified

LINK TABLE

Word format:



Names of external symbols (7 characters) must begin with a character for which the display code representation has a high order bit equal to zero. The data bytes have the form:



p — Position within the word of the reference to the external symbol:

10	upper
01	middle
00	lower

rl Relocation of address specified by loc

0 absolute, relative to RA

1 program relocation

3-77_8 relative to common block M where M is in position rl-2 of LCT

loc Address of the word containing the reference to the external symbol

REPLICATION TABLE

Each entry in the Replication Table is in the following format:

59	41	26	17 0
i		sr	s
c	b	dr	d

i Increment d by this value before each data block is repeated; first repetition of block is at d, second at d+1, etc. The data block (B-long) with origin at s is repeated c times beginning at d the first time, and beginning at the previous origin plus i thereafter. If i=0, i is interpreted as b.

sr Relocation of the address specified by s:

0 absolute, relative to RA

1 program relocation

3-77_8 relative to common block M where M is in position sr-2 of LCT; must not refer to blank common

s Initial address of the source data; must be non-zero

c Number of times data block is to be repeated; if c=0, c is interpreted as 1

b Size of data block; if b=0, b is interpreted as 1

dr Relocation of address specified by d

d Initial address for destination of data; if d=0, d is interpreted as s+b

TRANSFER TABLE

Each entry in the Transfer Table is in the following format:

59	17 0
entry point name	▨

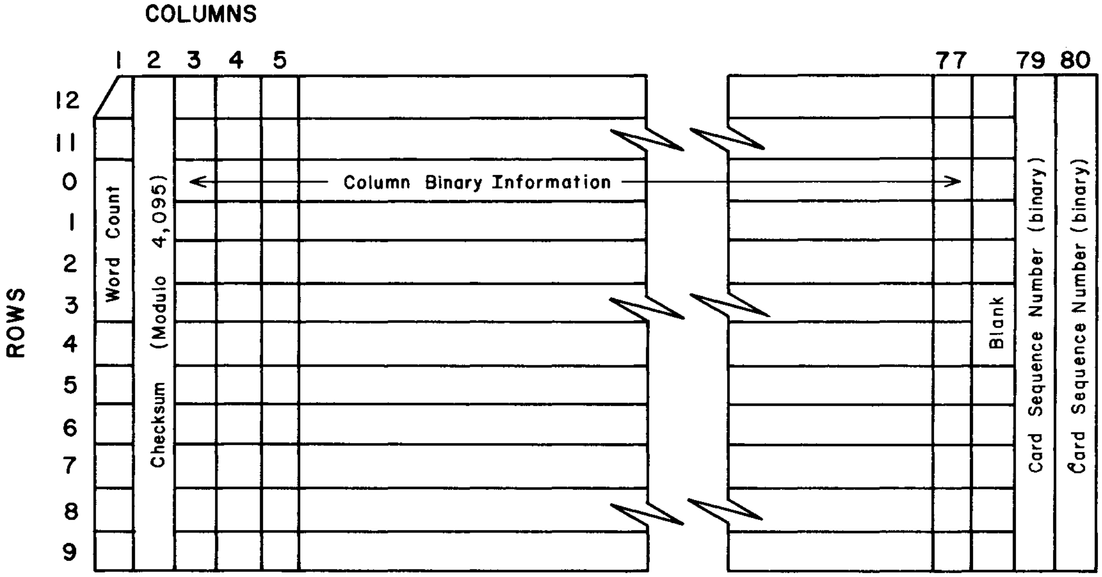
This table indicates the end of a subroutine and a pointer address. The entry point name need not be in the subprogram. If name is blank, there is no named transfer.

CARD FORMAT C

COLUMN 1

Column 1 indicates the kind of data that might appear on a card:

Punch	Represents
7, 8, 9	End-of-Record mark
6, 7, 9	End-of-File mark
5, 7, 9	Change code conversion
6, 7, 8, 9	End-of-Information mark
7, 9	Binary card
Not 7 and 9	Coded card


COLUMNS
1
2
3
4
5
77
79
80
ROWS
12
11
0
1
2
3
4
5
6
7
8
9
Word Count
Checksum (Modulo 4,095)
Column Binary Information
Blank
Card Sequence Number (binary)
Card Sequence Number (binary)

BINARY CARD

A binary card can contain up to 15 central memory words of data starting with column 3. Column 1 also contains a central memory word count in rows 0, 1, 2, and 3 plus a checksum indicator in row 4. If row 4 of column 1 is unpunched, column 2 can be used as a checksum for the card. If row 4 is punched, column 2 is not an accurate checksum for the card. Column 78 is blank. Columns 79 and 80 contain the card sequence number in binary.

CODED CARDS

Coded cards are in Hollerith code of up to 80 characters per card. When the system reads coded cards, it converts the data to display code and packs it 10 columns per CM word and deletes trailing blanks. Code conversion can be specified.

When the system punches coded cards, it extracts data from a line to an End-of-File mark or to the 80th character converting from display code to Hollerith.

ERROR DIAGNOSTIC MESSAGES D

Message	Explanation
ARITH. ERROR.	The system monitor has detected an arithmetic error condition in the last executing CPU program.
CHARACTER LIMIT ERROR.	The number of the first character to copy cannot be greater than the number of the last character; nor can the last character number be greater than 150.
CLL. ARG. ERROR. xxxxxx.	The argument xxxxxx is (a) equal to or greater than the field length minus two, or (b) the upper address of the argument is equal to or greater than the field length, or (c) the first address is equal to or greater than the upper boundary address.
CLO ARG. ERROR.	The CLO argument address is greater than or equal to the field length.
CPxx. COMPARE ERROR.	Card punch xx has mispunched a card. Operator action is not required since the card punch offsets the bad card and the card following it, and repunches both of these cards.
CRxx, BINARY CARD ERROR.	The card reader xx has detected a binary checksum error on the last card read.
DISK x PARITY ERROR Tyyy Gz Swww.	While loading from the disk, a 6603 or 6638 driver has detected an unrecoverable parity error on disk x, track yyy, group z, and sector www.
DMP ARG. ERROR.	The user has requested a dump of an area in central memory, the limits of which are outside the user field length.
DRUM x PARITY ERROR Uy Gzz Awwwww.	A 3436/863 driver has detected an unrecoverable parity error on unit y, of drum system x, at address wwwww, of head group zz.
ECS ABORT Axxxxxxx.	An ECS abort occurred while a program was referencing the data beginning at address xxxxxxx.
END OF FILE.	An End-of-File mark was found before a copy operation was complete.
EQxx TRACK LIMIT.	All tracks on device xx are reserved. Hence, none are available for processing of the user file.
ERROR IN ARGUMENTS.	Check the control card syntax as described in Sections 3 and 7.

Message	Explanation
FATAL LOAD ERROR xx.	KRONOS has detected a fatal load error, number xx, during the current load operation. The system aborts the job.
ffffff NOT ON SYSTEM.	File fffff does not reside on the system mass storage device.
FILE ALREADY ASSIGNED.	The control statement has requested a file which is already assigned to the job.
FILE NAME CONFLICT.	Check the preceding control card. Two names are identical which KRONOS does not allow.
FILE NAME IN USE.	The file name on a COMMON or RESERVE request is currently used by a COMMON file.
FILE NOT CREATED.	A control statement has attempted to declare a non-existent file as COMMON.
FILE NOT IN MASS STORAGE.	The file requested for program execution does not reside on a mass storage device.
FILE PROTECTED.	An attempt was made to release a "read only" file.
FL TOO SHORT FOR CATALOG.	Provide at least $6,200_8$ locations for the CATALOG program.
FL TOO SHORT FOR COPY.	Provide at least $4,6000_8$ locations for the COPY operation.
FL TOO SHORT FOR DOCEXT.	Provide at least $6,200_8$ locations for the DOCEXT program.
FL TOO SHORT FOR LOADER.	There is not enough memory space in the user's field length for LOADER.
FL TOO SHORT FOR MODIFY.	Provide at least $24,000_8$ locations for MODIFY operations.
FL TOO SHORT FOR VERIFY.	Provide at least $6,300_8$ locations for the VERIFY program.
FORMAT ERROR ON CONTROL CARD.	A system routine has found a format error on a control card. The bad card precedes this message in the dayfile.
ILLEGAL CHARACTER NUMBER.	The character count on a control card must be a numeric character.
ILLEGAL CONTROL CARD.	The control statement could not be identified.
ILLEGAL EQUIPMENT.	A control card has attempted to assign or request a non-existent peripheral device to a user file.
ILLEGAL FILE COUNT.	The file count on a control card must be a numeric character.
ILLEGAL PRIORITY.	Do not request a priority of less than one or greater than 17.

Message	Explanation
ILLEGAL RECORD COUNT.	The record count on a control card must be a numeric character.
ILLEGAL RECORD FORMAT.	Records on the source file from which to generate user libraries must be in relocatable form.
IMPROPER PACKAGE LABEL.	The called PPU package name is in improper format for insertion in RPL.
INVALID PARAMETER IN PROGRAM.	An illegal call to the ALGOL routine finder (ALG) has been generated.
JOB CARD ERROR.	KRONOS has detected an error on a job card.
LBC ARG. ERROR.	The LBC argument address is greater than the field length.
LIBRARY OVERFLOW.	The system cannot make another entry in the library directory.
LOADER NOT IN LIBRARY.	A program has written other information in the system library area and destroyed the loader.
LOADER TEXT BAD.	The routine LOD was unable to properly load the system loader.
LOC ARG. ERROR.	In an attempt to load octal corrections to central memory, the operator has done one of two things: a. addressed an area outside his field length, or b. his FIRST address is greater than his LIMIT address.
MSG ARG. ERROR.	A dayfile message specifies an address that is greater than or equal to the field length of the user's control point.
MTxx, READ PARITY ERROR.	The system has detected a parity error after eight re-read attempts. The operator should either type n.GO (in which case the record containing the parity error is lost) or type n.DROP.
MTxx, WPE RECOVERED.	The system has recovered a write parity error located on magnetic tape unit xx. The job continues.
MTxx, WPE UNRECOVERED.	The system has not recovered a write parity error (located on magnetic tape unit xx) after eight rewrite attempts and one Skip Bad Spot function. The operator should type either n.GO or n.DROP.
NO DATA IN FILE.	The file requested for execution contains no data.
OPE ARG. ERROR	The address of the File Environment Table in the call to Open File Routine (OPE) is greater than or equal to the FET field length minus five.

Message	Explanation
OPERATOR DROP.	The system monitor has set the Operator Drop error flag.
PBC ARG. ERROR.	The first address specified in calling PBC is greater than the terminal address.
PBC RANGE ERROR.	The terminal address (second parameter) is outside the user field length while in a call to the Punch Binary Card (PBC) routine.
PP CALL ERROR.	The system monitor has detected an error in the last CPU request for PPU action.
PROGRAM NOT AVAILABLE.	The system was unable to find the specified program in any of the system tables.
PROGRAM NOT ON MASS STORAGE.	The program does not reside on a mass storage device.
PROGRAM TOO LONG.	The program does not fit in the storage area available.
RBR ARG. ERROR.	The fifth character of the file name, called by the Read Binary Record routine, is greater than seven.
RECORD LENGTH TOO LONG.	The requested record length is greater than or equal to the field length minus five (buffer parameter area)
RECORD TOO LONG.	The length of the record is greater than the job length.
REQ ARG. ERROR.	Check the syntax for the REQUEST control card in Section 3.
REQ. ILLEGAL EQUIPMENT REQUEST.	Control card bears a request for a non-existent type of equipment.
REQ, ILLEGAL FILE NAME.	File name required to process REQUEST card is not left-justified alphanumeric characters with zero fill.
RFL ARG. ERROR.	The quantity of central memory requested exceeds that available, or the reference address is beyond the terminal address for the user's job.
ROLLIN FILE BAD.	The system has encountered an error in the format of the job file while rolling in the job.
RPL ARG. ERROR.	The package entered into the resident peripheral library is greater than or equal to the area in central memory allotted for it.
TIME LIMIT.	The job has exceeded the requested time limit. KRONOS aborts the job.
TOO MANY ARGUMENTS.	The number of arguments on the control statement is greater than the number allowed by the program.

Message	Explanation
TRACK LIMIT.	The system monitor has set the Track Limit error flag.
UNIDENTIFIED PROGRAM.	A program was not a PPU, a SCOPE, nor a Chippewa program.
WBR ARG. ERROR.	The fifth character of the file name is greater than seven.
xxx TOO LONG.	The PPU package xxx exceeds 1, 314 central memory words (the maximum PPU package size in length).
xxx NOT IN PP LIB.	KRONOS has assigned program xxx to a PPU for execution, but program xxx is not in the system library. Typically, this happens when the user puts garbage in location RA+1.
814 xx PARITY ERROR Tyyy Szzzz.	An 814xx (each half of the 814 is considered as one device) has detected an unrecoverable parity error at sector zzzz of track yyy.
854 xx PARITY ERROR Cyyy Szzz.	A 3234/853/854 driver has detected an unrecoverable parity error at sector zzz of cylinder yyy.

CONTROL CARDS E

Control Card	Function
ACCOUNT, xxxxxxx.	Specifies the user's account number
APPEND, lfn$_1$, . . . , lfn$_n$, ACCOUNT = xxxxxxx.	Appends information to an existing permanent file
ASSIGN (nn, lfn, x)	Assigns peripheral equipment
BKSP (file$_1$, n)	Backspaces n logical records
CATALOG (filea, num, fileb)	Makes a catalog for filea
CATLIST. or CATLIST, F. or CATLIST, xxxxxxx.	Lists a catalog of the permanent files
COMMON (lfn)	Creates or attaches a COMMON file
COMMENT. comments	Enters the dayfile message "comments"
COMPASS (p_1, p_2, . . . , p_n)	Calls COMPASS assembler
COPY (file$_1$, file$_2$)	Copies to an empty file
COPYBF (file$_1$, file$_2$, n)	Copies n binary files
COPYBR (file$_1$, file$_2$, n)	Copies n binary records
COPYCF (file$_1$, file$_2$, n)	Copies n BCD files
COPYCR (file$_1$, file$_2$, n)	Copies n BCD records
COPYSBF (file$_1$, file$_2$, n)	Copies n shifted binary files
COPYX (file$_1$, file$_2$, term, bksp)	Copies binary records to terminator and backspaces
DMP (fwa, lwa) or DMP (lwa) or DMP.	Dumps storage
DOCEXT (sfile, ℓfile)	Calls the DOCEXT program which extracts the external documentation
EXECUTE (name, p_1, p_2, . . . , p_n)	Executes starting at entry point "name"
EXIT.	Control cards to obey in case of error follow

Control Card	Function
GET, lfn_1 = pfn_1, . . . , lfn_n = pfn_n, ACCOUNT = xxxxxxx.	Retrieves a file from permanent storage
LBC, addr. or LBC.	Loads binary corrections
LIBEDIT (p_1, p_2, . . ., p_n)	Calls the LIBEDIT program
LIBGEN (library, name, input)	Generates a user library file
LOAD (lfn_1) or (lfn_1, lfn_2, . . . , lfn_n)	Loads lfn_1, satisfies references from lfn_2 - lfn_n
LOC (fwa, lwa) or LOC (lwa) or LOC.	Loads octal corrections
LOCK (lfn_1, . . ., lfn_n)	Sets the write lockout for file lfn_1
MACE.	Clears the SCOPE bit for loading a file
MAP.	Sets Map flag
MAP (P)	Sets Partial Map flag
MODE (n)	Sets error mode = n
MODIFY (p_1, p_2, . . . , p_n)	Calls MODIFY
NOGO.	Finish loading process, do not execute
NOMAP.	Clears Map flag
NOREDUCE.	Sets No Reduce FL flag
OFFSW (n_1, . . . , n_n)	Clears pseudo-sense switches
ONSW (n_1, . . . , n_n)	Sets pseudo-sense switches
OVERLAY (lfn, a, b, Cnnnnnn)	Initiates an overlay for file lfn
PBC (fwa, lwa) or PBC (lwa) or PBC.	Punches binary cards
PERMIT, pfn, MODE=y, $accnum_1$, . . . , $accnum_n$.	Grants permanent file access to other users
PURGE, pfn_1, . . . , pfn_n, ACCOUNT = xxxxxxx.	Purges a permanent file
RBR, n.	Reads n binary records
REDUCEFL.	Clears No Reduce FL flag

Control Card	Function
RELEASE (lfn)	Changes status of a file from COMMON to LOCAL
REPLACE, lfn_1 = pfn_1, ..., lfn_n = pfn_n, ACCOUNT = xxxxxxx.	Substitutes a new file for an old permanent file
REQUEST (lfn, x)	Requests an operator to assign equipment
RESERVE (lfn)	Changes status of a file to COMMON
RETURN ($file_1$, $file_2$, ..., $file_n$)	Returns files to the system
REWIND ($file_1$, $file_2$, ..., $file_n$)	Rewinds files
RFL, nfl.	Changes the field length
ROLLOUT.	Rolls out user's job
RUN (cm, fl, bl, if, of, rf, lc, as, cs)	Calls FORTRAN compiler
SATISFY (lfn_1, lfn_2, ..., lfn_n)	Satisfies external references from lfn_1
SAVE, lfn_1 = pfn_1, ..., lfn_n = pfn_n.	Retains files in the permanent file system
SCOPE.	Sets SCOPE bit for loading a file
SETID (lfn_1 = a_1, ..., lfn_n = a_n)	Assigns area code for file lfn_1
SETPR (p)	Sets priority level
SETTL (t)	Sets time limit
SKIPEI ($file_1$)	Skips to End-of-Information
SKIPF ($file_1$, n)	Skips n files forward
SKIPFB ($file_1$, n)	Skips n files backward
SKIPR ($file_1$, n)	Skips n records forward
SYSREF (p_1, p_2, ..., p_n)	Generates a list of system symbols
UNLOAD ($file_1$, $file_2$, ..., $file_n$)	Unloads and rewinds files
UPMOD (p_1, p_2, ..., p_n)	Converts program libraries maintained by UPDATE to libraries that can be maintained by MODIFY
VERIFY ($file_1$, $file_2$, p_1, p_2, ..., p_n)	Compares data in two files

Control Card	Function
VFYLIB (old, new, output, r)	Compares old library file with a new one
WBR, n, rl.	Writes n binary records
WRITEF (file_1, n)	Writes n file marks on a file
WRITER (file, n)	Writes n empty records on a file

CHARACTER SET F

Char.	Display	Hollerith	EXT BCD	INT BCD	Char.	Display	Hollerith	EXT BCD	INT BCD
A	01	12-1	61	21	6	41	6	06	06
B	02	12-2	62	22	7	42	7	07	07
C	03	12-3	63	23	8	43	8	10	10
D	04	12-4	64	24	9	44	9	11	11
E	05	12-5	65	25	+	45	12	60	20
F	06	12-6	66	26	−	46	11	40	40
G	07	12-7	67	27	*	47	11-8-4	54	54
H	10	12-8	70	30	/	50	0-1	21	61
I	11	12-9	71	31	(	51	0-8-4	34	74
J	12	11-1	41	41	)	52	12-8-4	74	34
K	13	11-2	42	42	$	53	11-8-3	53	53
L	14	11-3	43	43	=	54	8-3	13	13
M	15	11-4	44	44	space	55	space	20	60
N	16	11-5	45	45	,	56	0-8-3	33	73
O	17	11-6	46	46	.	57	12-8-3	73	33
P	20	11-7	47	47	≡	60	0-8-6	36	76
Q	21	11-8	50	50	[	61	8-7	17	17
R	22	11-9	51	51	]	62	0-8-2	32	72
S	23	0-2	22	62	:	63	8-2	00	12
T	24	0-3	23	63	≠	64	8-4	14	14
U	25	0-4	24	64	→	65	0-8-5	35	75
V	26	0-5	25	65	∧	66	11-0	52†	52
W	27	0-6	26	66	∨	67	0-8-7	37	77
X	30	0-7	27	67	↑	70	11-8-5	55	55
Y	31	0-8	30	70	↓	71	11-8-6	56	56
Z	32	0-9	31	71	<	72	12-0	72††	32
0	33	0	12	00	>	73	11-8-7	57	57
1	34	1	01	01	≤	74	8-5	15	15
2	35	2	02	02	≥	75	12-8-5	75	35
3	36	3	03	03	¬	76	12-8-6	76	36
4	37	4	04	04	;	77	12-8-7	77	37
5	40	5	05	05	%	00	8-6	16	16

† 11-0 and 11-8-2 are equivalent
†† 12-0 and 12-8-2 are equivalent

INDEX

INDEX (Cont'd)

INDEX (Cont'd)

COMMENT SHEET

MANUAL TITLE CONTROL DATA® KRONOS Batch User's Reference Manual

PUBLICATION NO. 59150600 REVISION A

FROM: NAME:

BUSINESS ADDRESS:

COMMENTS:

This form is not intended to be used as an order blank. Your evaluation of this manual will be welcomed by Control Data Corporation. Any errors, suggested additions or deletions, or general comments may be made below. Please include page number references and fill in publication revision level as shown by the last entry on the Record of Revision page at the front of the manual. Customer engineers are urged to use the TAR.

CUT ALONG LINE

NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

FOLD ON DOTTED LINES AND STAPLE

STAPLE

STAPLE

FOLD

FOLD

FIRST CLASS
PERMIT NO. 8241

MINNEAPOLIS, MINN.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION
4201 NORTH LEXINGTON AVENUE
ST. PAUL, MINNESOTA 55112

ATTN: BISD PUBLICATIONS

CUT ALONG LINE

FOLD

FOLD

CONTROL DATA
CORPORATION

CORPORATE HEADQUARTERS, 8100 34th AVE. SO., MINNEAPOLIS, MINN, 55440
SALES OFFICES AND SERVICE CENTERS IN MAJOR CITIES THROUGHOUT THE WORLD