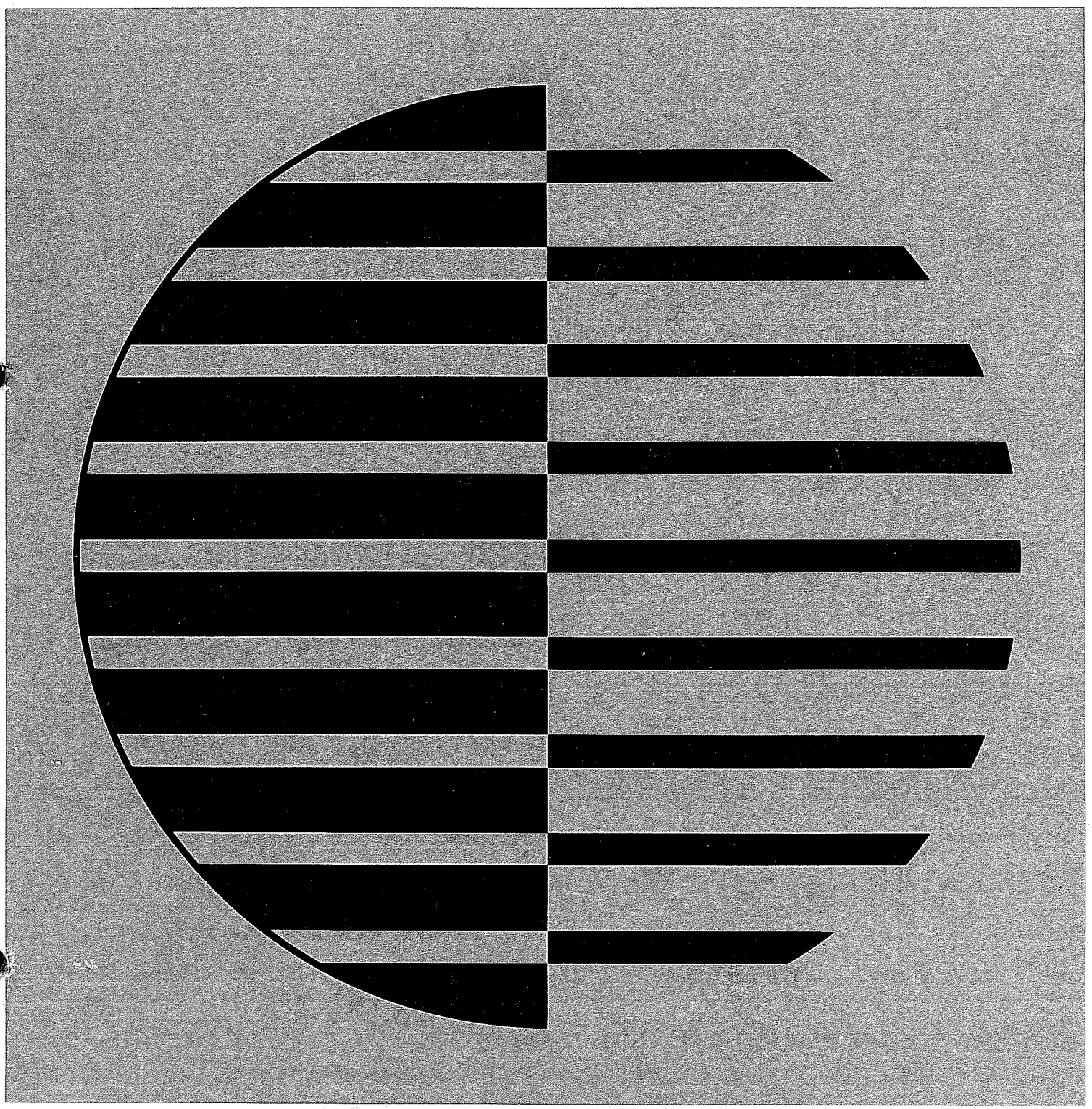
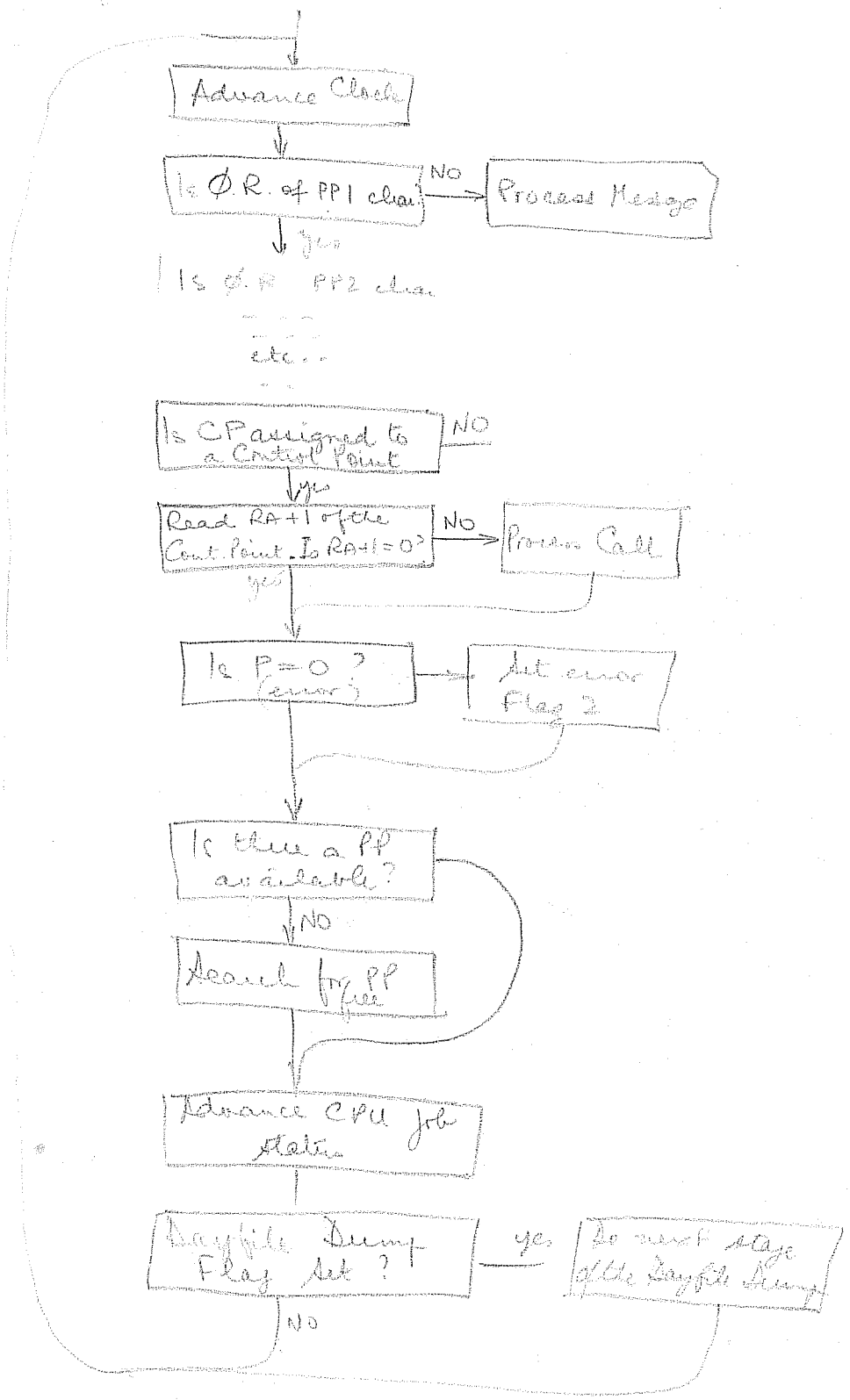
Richard J. Levingston

CONTROL DATA® 6000 SERIES COMPUTER SYSTEMS

Chippewa Operating System Reference Manual

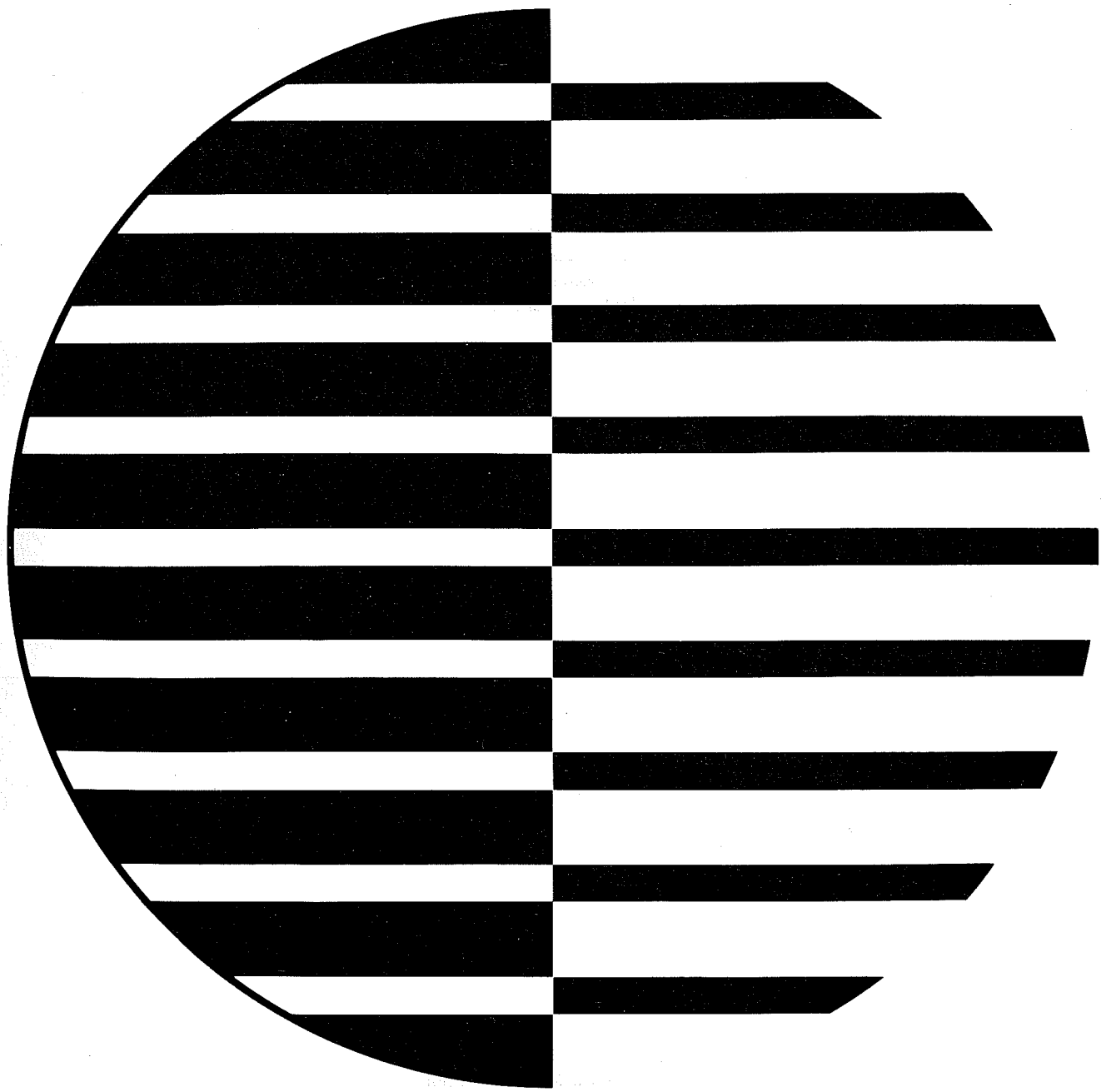


Monitor loop



CONTROL DATA® 6000 SERIES COMPUTER SYSTEMS

Chippewa Operating System Reference Manual



Additional copies of this manual may be obtained from the nearest Control Data Corporation Sales office listed on the back cover.

CONTROL DATA CORPORATION
Documentation Department
3145 PORTER DRIVE
PALO ALTO, CALIFORNIA

December 1965
Pub. No. 60134400

PREFACE

The Chippewa Operating System developed by the Chippewa Laboratories of the Control Data Corporation is designed for use with Control Data® 6000 Series Computers.

The Chippewa Operating System is a monitor system used for directing the sequencing of programs by Series 6000 computers. The system performs various input/output, compilation, and storage assignment tasks in a manner as to achieve the efficiency inherent in the concept of the Series 6000 computers.

Familiarity with 6000 hardware and related software manuals describing the ASCENT, ASPER, and FORTRAN languages is assumed. Information concerning the 6000 Series Computer/Programming Systems may be found in the following manuals.

	Publication Number
Control Data 6000 Series Computer Systems Reference Manual	60100000
Control Data 6600 Computer System/Programming System Reference Manual Volume 1 ASCENT (Assembly System Central Processor)	60101600
Control Data 6600 Computer System/Programming System Reference Manual Volume 2 ASPER (Assembly System Peripheral Processor)	60101700
Control Data 6000 Series Chippewa Operating System FORTRAN Reference Manual	60132700
Control Data 6600 Chippewa Operating System (Flow Diagrams)	60124500

CONTENTS

TABLES AND ILLUSTRATIONS

CHIPPEWA OPERATING SYSTEM TERMINOLOGY

byte	A 12-bit group of bits operated upon as a unit.
central library directory	A table containing information about all the central memory routines stored on the disk. (CLD)
channel status table	A table in resident central storage containing three words, the first 12 bytes of which are associated with the 12 input/output channels. (CST)
coded files	Alphanumeric files stored in packed display code for disk and magnetic tape.
control point	A number, 1 through 7, appearing on the display scope which is associated with status information about jobs in central memory.
common files	Common files are files that are not discarded upon job completion and may be picked up by other jobs.
dayfile	A file on disk that holds a running account of all control cards, equipment assignments, error diagnostics, central and peripheral processor time used, and I/O packages (e.g., PRINT, READ) used by the jobs in central memory.
dead start	The initial loading of the system tape by manual toggle switching after setting the panel switches.
display code	A code in which alphanumeric files are stored for console display purposes. Each line of alphanumeric characters begins at the first 12-bit byte of a central word and continues two characters per byte to the end of the line.
equipment number	A two-digit octal number which uniquely identifies the equipment for job assignments.
equipment status table	A table in central storage containing one-word entries for each physical equipment unit in the system. (EST)
file name table/ file status table	A common table structure containing the name and status of all coded files. (FNT/FST)
idle program	A program to which monitor transfers control when no other program is ready for execution.

input files	Job files which have not been assigned a control point (listed in the job backlog (H) display with associated priority). A stored form of a job on disk, or logical file of cards separated by a record separator or file separator card.
job display	The program that displays only data pertaining to the particular job. (DIS)
local files	Local files are files accessible at specified control points; local files may be initiated by a job and discarded at the end of a job.
output files	A list of job files which have not been printed (appearing on the job backlog (H) display). These jobs are processed on a priority basis. If equal priorities are encountered, the last one in is the first one out.
package	An I/O process which includes at least one routine, for example, Read Package, Print Package, or Job Display Package.
peripheral resident	Those PP programs that permanently reside in a peripheral processor.
pseudo-control point	A number, 0 or 8, which does not appear on the display scope; 0 is associated with the monitor program in PP0, and the display program in PP9; 8 is associated with the storage move program.
reference address	The address serving as a starting point for subsequent central resident address modification.
resident peripheral library	Peripheral programs that are stored in central memory. (RPL)
system display	The program that provides an overall status display for all currently running jobs. (DSD)
track reservation table	A table maintained in resident central storage for each disk file cabinet; each table contains 2048 bits. (TRT)
transient program	Any program that is called into a peripheral processor (PP) but does not permanently reside there (all PP programs other than monitor (MTR) in PP0, System Display (DSD) in PP9, and the peripheral resident).

INTRODUCTION 1

The Chippewa Operating System for the Control Data® Series 6000 uses advanced design features to achieve efficient multiprocessing of a wide variety of jobs. The Series-6000 Computers are composed of eleven processors. Ten of these processors perform peripheral and operating system functions; the eleventh, the central processor, performs computation and processing at very high speeds. The eleven processors have separate memories and operate concurrently under control of the Operating System. The large central memory is accessible to all eleven processors. The Operating System is in constant control of all jobs in process, assuring optimum use of I/O equipment and priority processing where required. It performs such functions as storage allocation, job scheduling, accounting, I/O control and operator communication.

1.1 SYSTEM COMPONENTS

The Operating System is initially loaded from the system tape (Appendix A) by means of keyed settings on the Dead Start panel. Components of the Operating System are distributed, during this operation, among the central memory, the ten peripheral memories, and the magnetic disk unit.

The central resident portion of the Operating System includes system control parameters and pointers, communication linkages, and frequently used programs and subroutines for both the central and the peripheral processors.

The peripheral resident portion consists of a resident program stored in each peripheral memory, the system monitor (MTR) assigned permanently to processor 0, and the system display program (DSD) assigned permanently to processor 9.

The remainder of the Operating System is stored on the magnetic disk unit or in central memory to be called as needed.

1.2 CENTRAL MEMORY

The central processor executes jobs of a computational or production nature including operational user programs and compilation and assembly of user source language programs. These programs are stored in central memory along with the data necessary for execution and control. Central memory is accessible to both central and peripheral processors and serves as the communication link between them.

A number of programs may be in operation concurrently in the central processor. Central resident contains status information for each running program which enables the Operating System to proceed in an orderly fashion in the control and sequencing of the programs.

1.3 PERIPHERAL PROCESSORS

The ten peripheral processors (PP) are used by the Operating System to perform all I/O functions required by the system or the operational programs. They also perform certain auxiliary system functions connected with job sequencing and control.

The PP's have identical resident programs which sense a location in central memory for a control word calling for some action. The resident program locates the required program in central memory or on the disk and loads it into its own memory for execution. The peripheral program may, itself, call additional peripheral programs into its memory, or call for action by the system monitor.

The PP's form a common pool available for assignment as needed by the system. Except for MTR, DSD and the resident programs, peripheral programs have no fixed processor assignments, and are loaded each time they are called. All processors operate concurrently to maintain a maximum I/O transfer rate.

1.3.1 MONITOR

The Operating System functions under the direction of the system monitor (MTR) which is permanently assigned to PP0. MTR repeatedly scans a set of locations in central memory which are set by transient peripheral programs to call for monitor action. MTR also senses the status of the running central program to determine program aborts, terminations, and requests for monitor action.

MTR is used in the assignment and release of all peripheral processors, data channels, disk storage and I/O equipment. It maintains constant surveillance of all processing in the central and peripheral processors.

1.3.2 SYSTEM DISPLAY

The system display program (DSD), permanently assigned to PP9, serves as the communication linkage between the system and the operator. Two console screens provide system monitoring information and displays of central memory, selectable by the operator. Through the console keyboard, the operator can modify central memory contents and request system functions.

1.4 DISK STORAGE

The magnetic disk unit contains the non-resident portion of the Operating System including both peripheral and central library programs. Central resident contains a peripheral library directory (PLD) and a central library directory (CLD) which define the disk location for each library program.

The disk also holds the job stack and the data files for the jobs in process. Output from job execution is collected on the disk for printing or punching by a peripheral program. In this sense, the disk serves as a large capacity buffer between the I/O devices and the processor complex.

1.5 OPERATION

Once the Operating System is loaded, all eleven processors execute idling programs while waiting to be called into operation. Processing is initiated when the operator selects the system input unit from the console keyboard; the system display program alerts the monitor to enter information in the system tables and call the requested program into operation in a PP. The input program loads job decks from the system input unit into disk storage. As each job is filed on the disk, the input program makes an entry in a file name/file status table (FNT/FST) in central memory from which jobs are selected on a priority basis.

As many as seven programs may be active in the central processor at one time. Each active program is assigned a control point area which contains all information necessary to control the program and resume operation after interrupts. MTR continually senses the status of each PP and the running central program. When a call for action is detected, MTR either performs the operation itself or passes the request to one of the other PP's.

MTR interrupts the running central program to pass control to the next active central program on a priority basis if I/O requests are made. When program completion is sensed, MTR assigns a PP to search the central memory FNT/FST for the next program to be assigned to the control point and loaded into central memory.

A program operating in the central processor may call peripheral library programs simply by entering the name of the required program in its reference address location, plus one (RA + 1). It can call directly any peripheral program whose name begins with a letter. Peripheral programs whose names begin with numbers are called only by other peripheral programs (section 2.4).

SYSTEM ORGANIZATION 2

Operating System storage is distributed among central memory, peripheral memories, and the magnetic disk unit. The most frequently used system tables and programs are stored in the resident areas of central and peripheral processor memories. The remaining Operating System programs are filed on the disk to be called as needed.

2.1 PP MEMORY

Each peripheral processor (PP) memory contains an identical resident program in locations 0100-0772. This resident program contains an idling routine which repeatedly examines a location in central resident associated with its PP. A message in this location is interpreted by the resident to perform some assigned function. The resident is charged with locating required peripheral library programs and loading them into its memory for execution.

The PP resident uses locations 0000-0017 for temporary storage. These locations may also be used by called peripheral programs; no continuity of information is required. Locations 0020-0074 are also available to called programs for temporary storage.

Locations 0075-0077 contain addresses of central memory locations which are associated with each PP. They must not be altered by a peripheral program. The contents of these locations in central memory are as follows:

0075 address of PP input register

0076 address of PP output register

0077 address of PP message buffer

The system monitor (MTR) is permanently assigned to PP0, beginning at location 1000. The system display program (DSD) is permanently assigned to PP9, beginning at location 1000.

PP memories 1-8 load transient programs as assigned by MTR from the peripheral libraries residing in central memory and on the disk. They are treated as overlays called by the resident PP programs. The basic transient programs, coded to begin execution at location 1000, are loaded beginning at location 0773. These programs can, themselves, call equipment driver overlays which are coded to begin at location 2000 and loaded at location 1773. Transient programs are not assigned to specific peripheral processors; they are called each time they are needed and then discarded (Figure 1).

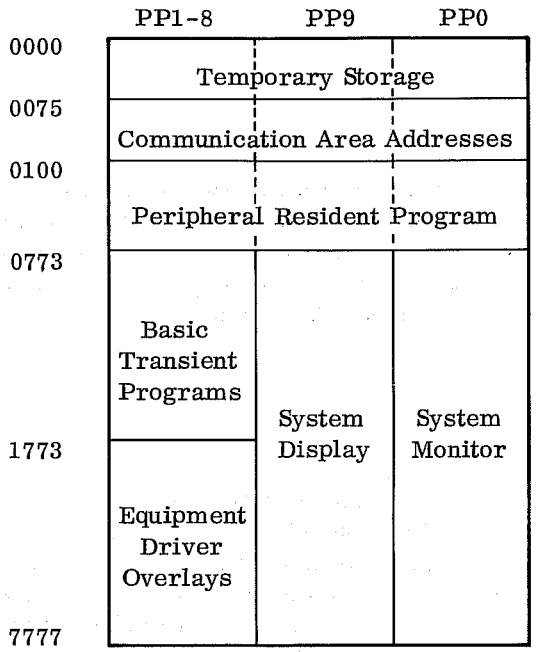


Figure 1. Peripheral Processor Memory Allocation

2.2 CENTRAL MEMORY

The lower portion of central memory from location 0000 to 1777 is occupied by system pointers, tables and communication areas (Figure 2). Storage locations for this area are fixed as follows:

Table 1. CONTENTS OF CENTRAL MEMORY STORAGE
(Locations $0000\text{-}1777_8$)

Storage Locations (octal)	Contents	
0000	Zero	
0001	RPL	(resident peripheral library pointer)
0002	PLD, limit	(peripheral library directory pointer)
0003	DFB, input,	output, limit (dayfile buffer pointer)
0004	FNT, limit	(file name/file status table pointer)
0005	EST, limit	(equipment status table pointer)
0006	RSL, limit	(resident subroutine library pointer)
0007	CLD, limit	(central library directory pointer)

Storage Locations (octal)	Contents
0010	TRT0 }
0011	TRT1 } track reservation table pointers
0012	TRT2 }
0013	Blank
0014	Monitor step control flag
0015-0017	Channel status table
0020-0022	Control point zero status information
0023	Idle time for central processor
0024	Idle time for peripheral processors
0025	Control point 0 recall input register
0026	Exchange jump address for simulator
0027	P address for simulator
0030-0037	Time and date line
0040-0052	Starting time counts
0053-0054	Blank
0055	Temporary storage for monitor
0056-0057	Control point stack indicators
0060-0067	PP1 communication area
0070-0077	PP2 communication area
0100-0107	PP3 communication area
0110-0117	PP4 communication area
0120-0127	PP5 communication area
0130-0137	PP6 communication area
0140-0147	PP7 communication area
0150-0157	PP8 communication area
0160-0167	PP9 communication area
0170-0177	PP0 communication area
0200-0377	Control point 1 area
0400-0577	Control point 2 area
0600-0777	Control point 3 area
1000-1177	Control point 4 area
1200-1377	Control point 5 area
1400-1577	Control point 6 area
1600-1777	Control point 7 area

Locations 002000-013777 are referred to by pointers in location 0001 through 0012. This area is occupied by the central resident program and by the system tables, library directories, and resident libraries. The remainder of central memory is available for program and data storage.

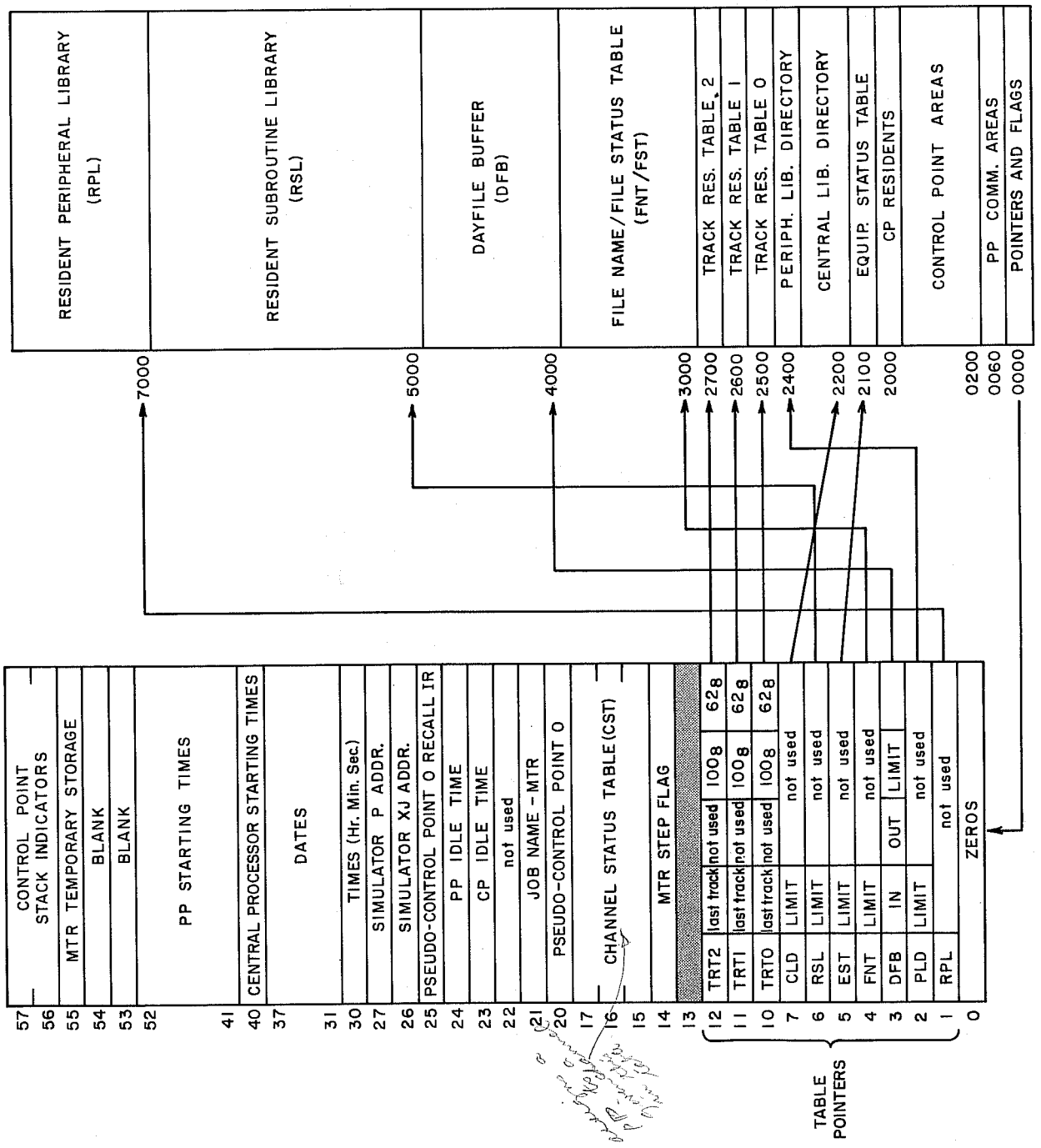


Figure 2. Resident Central Storage (typical)

2.3 DISK FILE

The magnetic disk file holds all programs associated with the Operating System which are not stored in peripheral or central memories. System programs stored on disk 0 consist of the peripheral processor library followed by a central processor library. Central resident contains directories for both disk file libraries defining the track and group number for each program (Appendix B).

All programs and data are stored on an input device (disk or tape units) prior to processing. Programs, alphanumeric data files and binary data files, regardless of content, are stored on the disk in a common structure. The basic unit of storage on the disk is a half-track consisting of either the odd or the even numbered sectors in a physical track. There are 2048 half-tracks in each disk cabinet and each half-track contains 64 sectors in the outer zone and 50 sectors in the inner zone. Each sector has a data capacity of 320 12-bit bytes (Figure 3).

A logical file on the disk is defined as a named half-track followed by any number of continuation half-tracks in a single cabinet. This named file must begin in the first sector of a half-track and may be of any length up to the capacity of the cabinet.

In addition to the 320 data bytes, each sector contains two control bytes which link the sectors and tracks making up a logical file. The first control byte designates the location of the next sector in the file. The second control byte contains the number of central processor words in the current sector. The first control byte is coded in one of two formats. If the next sector of the file is in the same half-track, the control byte contains the next sector number (0000-0077). If the next sector of the file is the beginning of a new half-track, the control byte is coded for the next half-track number as follows:

bits 0-2	head group number
bit 3	odd/even half-track
bits 4-10	track number
bit 11	always set

The control bytes for the last sector of a file are both zero. This sector is interpreted as a file mark and terminates the file.

Disk track availability is recorded in three track reservation tables (pointers, TRT0, 1, and 2) in central resident. Each table is associated with one of the three disk cabinets.

TRT0 = 1 disk cabinet

TRT1 = 2 disk cabinets

TRT2 = 3 disk cabinets

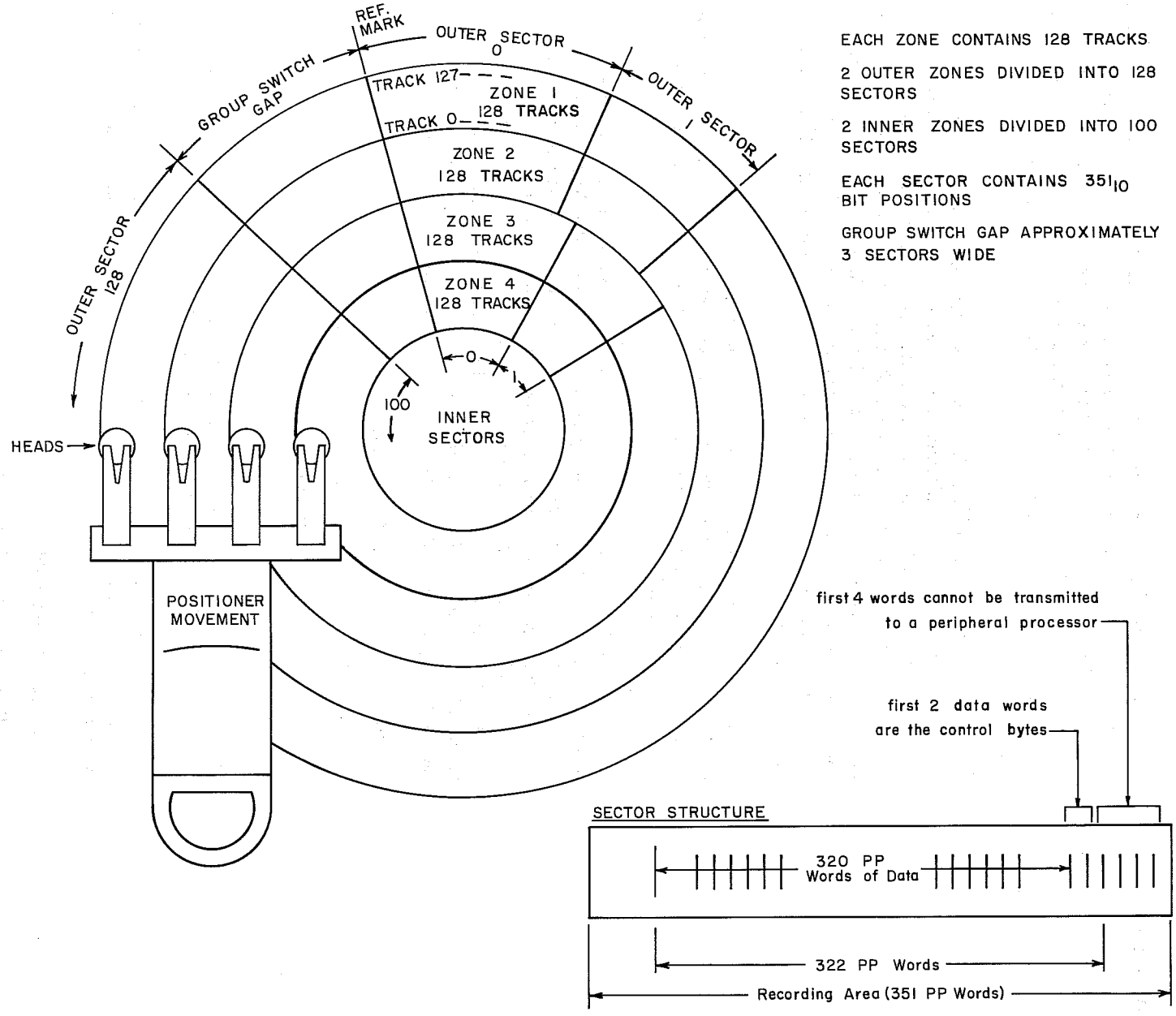


Figure 3. 6603 Disk Structure

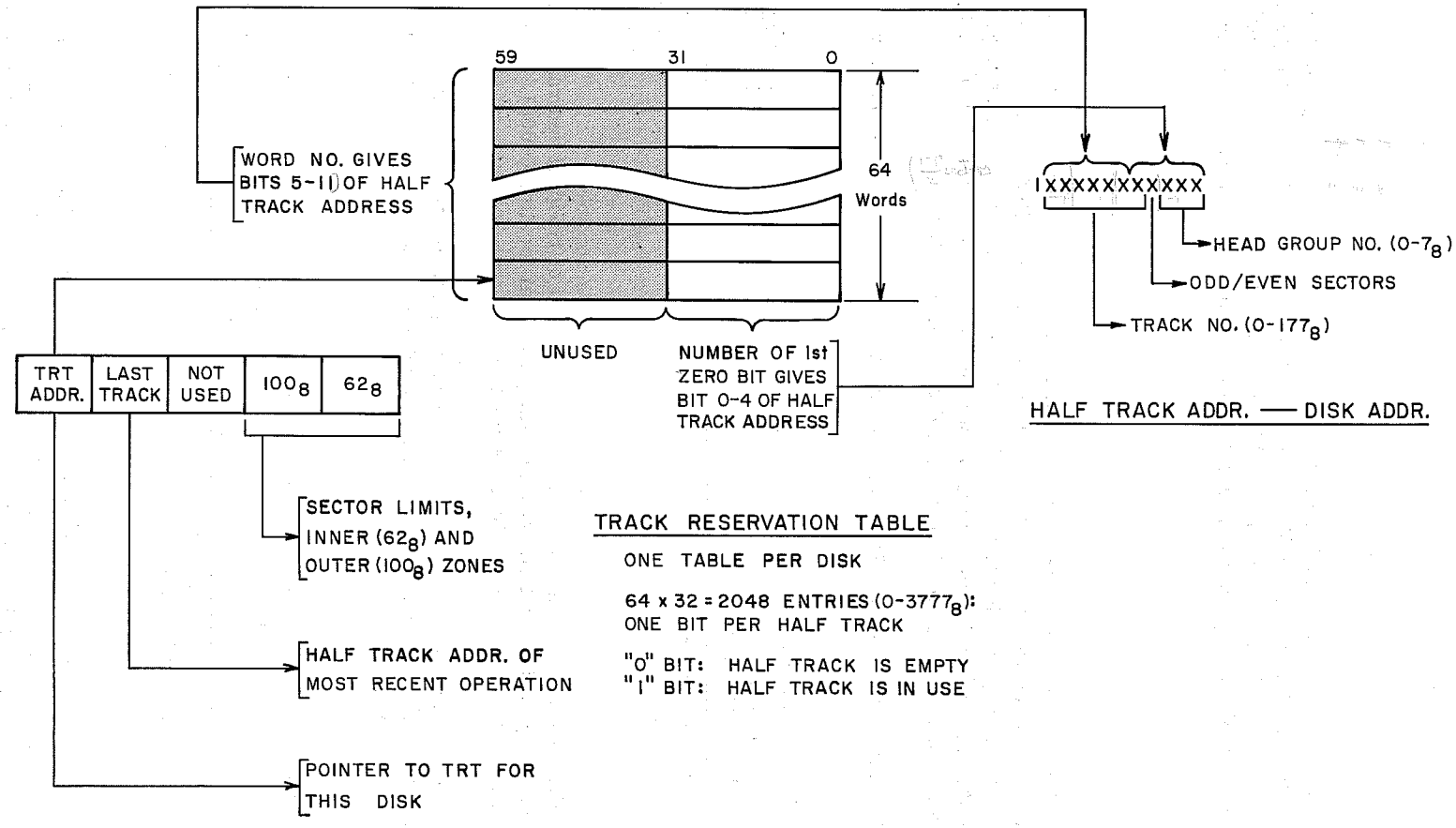


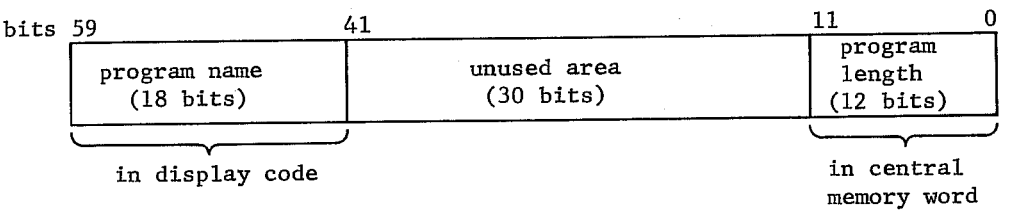
Figure 4. Track Reservation and Addressing

The full/empty status of each track in a cabinet is indicated by one of 2048 bits occupying the rightmost 32 bits of the 64 words in a table. The bit position in the word ($0-37_8$) corresponds to bits 0-4 of a track number. The word position in the table ($0-77_8$) corresponds to bits 5-10 of the track number (Figure 4).

2.4 PP PROGRAMS

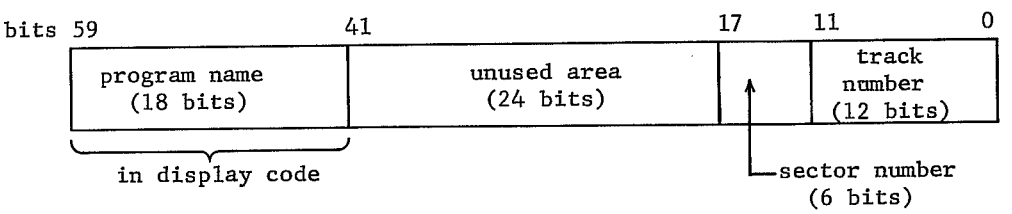
PP programs are stored either in the resident peripheral library (RPL) in central memory or in a magnetic disk file. When called, they are transferred to the appropriate PP for execution. PP programs whose names begin with a letter may be called by central programs; those whose names begin with a numeral are called only by other PP programs. (Section 3.1.)

The first word of each PP program contains the following information:



RPL programs are stored in a contiguous area of resident central memory beginning at the location shown by the pointer at location 000001. The last RPL program is followed by a zero word.

PP programs stored on disk 0 are located by the peripheral library directory (PLD) in the resident area of central memory. Each word of the directory defines one program:

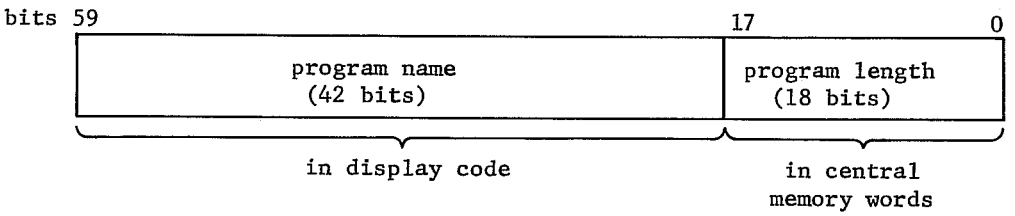


When a PP program is called by a central program, the system monitor (MTR) assigns this call to a PP, and the resident searches for the program in RPL. If it is not located in RPL, PLD is searched for the disk location of the program. When the program is found, it is loaded into the PP and executed. If a program is not found, a message appears at the associated control point and the job is discontinued. The existing output is printed automatically.

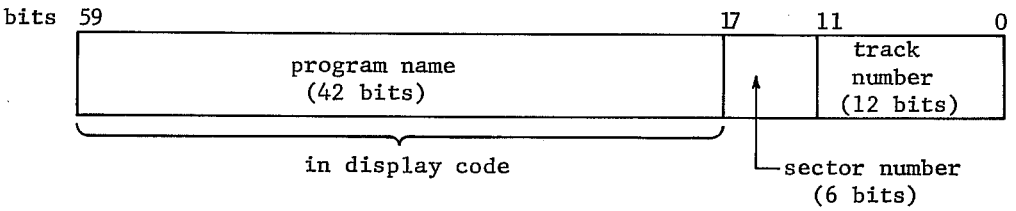
Appendix C lists suggestions for writing PP programs for the Chippewa Operating System, and the messages from the system peripheral programs which are entered into the dayfile are listed in section 8.6.

2.5 CENTRAL PROGRAMS

Central program subroutines which are most frequently used reside in central storage in the resident subroutine library (RSL). When a central program calls for these subroutines they are read directly from this area rather than from the disk. The first word of each program contains the following information:



The central library directory (CLD) locates central programs and subroutines stored in the disk 0 library. CLD, which is stored in central resident, consists of one-word entries in the following format:



SYSTEM CONTROL 3

All activities in the system are controlled and coordinated by the system monitor (MTR). The monitor is permanently assigned to PP0, where it operates continually to schedule the use of the other peripheral processors and the programs in the central processor. MTR functions on the basis of messages from other processors and, in turn, routes messages to the other processors for action.

3.1 PP COMMUNICATION AREAS

MTR communicates with the resident programs in the other nine peripheral processors through ten central memory areas. Each area is eight words in length, and each is associated with a particular processor.

The communication area is organized as follows:

Word 0 - processor input location

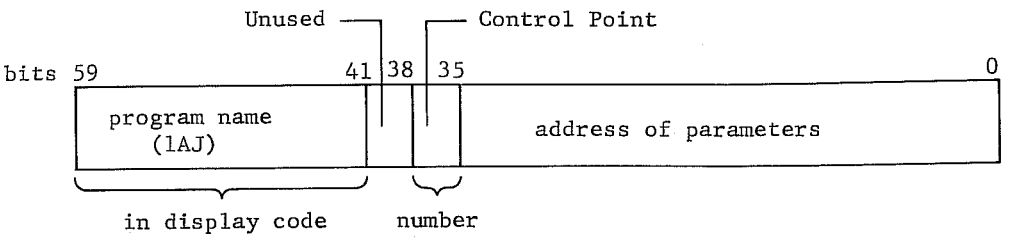
Word 1 - processor output location

Words 2-7 - message buffer

A PP idles in its resident program as long as its associated processor input location is cleared. The monitor enters a control word in a selected input location to call a transient peripheral program to that processor. The resident peripheral program senses the input location entry, locates the called program in the resident peripheral library or the peripheral library directory, and loads it into peripheral memory beginning at location 0773. Resident then jumps to 1000 to begin execution of the transient program.

The PP input register is not cleared by MTR until the transient program has completed its function, and control has returned to peripheral resident at location 0100.

The format of the input register control word is:



A peripheral program, by entering a value in its central memory output location, communicates a request to MTR. A request too long for a single location is continued in the 6-word message buffer. MTR continually scans the ten PP output locations. When a message is found, MTR jumps to the internal MTR subroutine to process the request. When the function has been performed, MTR clears the processor output register and continues scanning.

3.2 CONTROL POINT AREAS

The ability to exchange central memory programs depends upon the preservation of all critical parameters and register contents at the time of the exchange. 1600 octal central memory locations are reserved for this purpose. This area is divided into seven 200 octal word control point areas. Each area preserves the information necessary to enable an interrupted program to resume execution.

Relative octal locations of each control point area are:

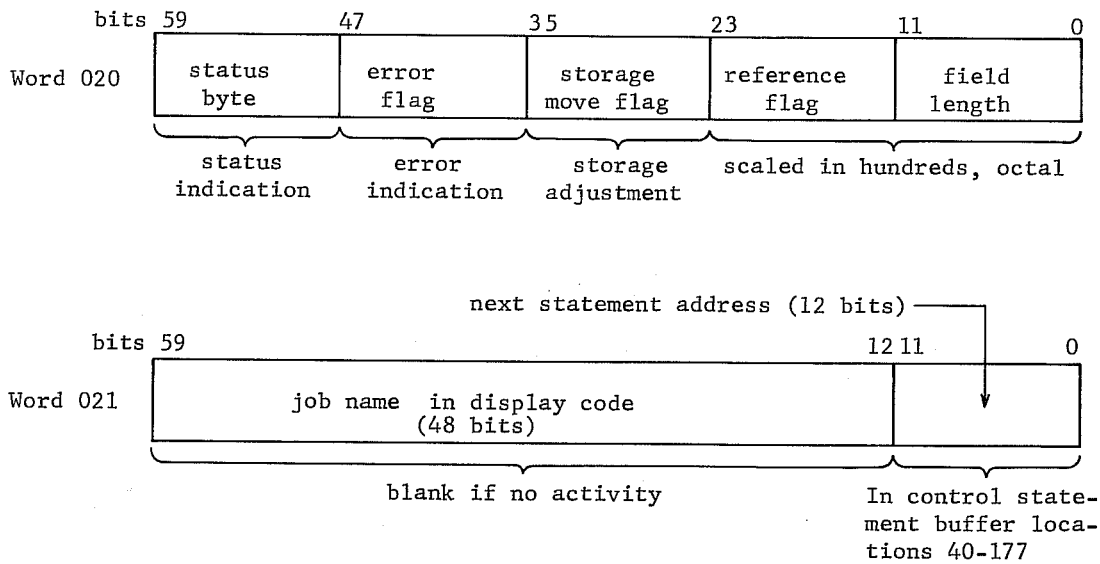
Locations	Control Point Areas	
000-017	Exchange Jump area information (Section 3.3)	
020-022	Status information	
023	Central Processor running time	seconds, milliseconds
024	PP running time	
025	PP recall input register	
026	Sense switches	
027	Equipment assignments	
030-037	Last dayfile message	
040-177	Control statement buffer	

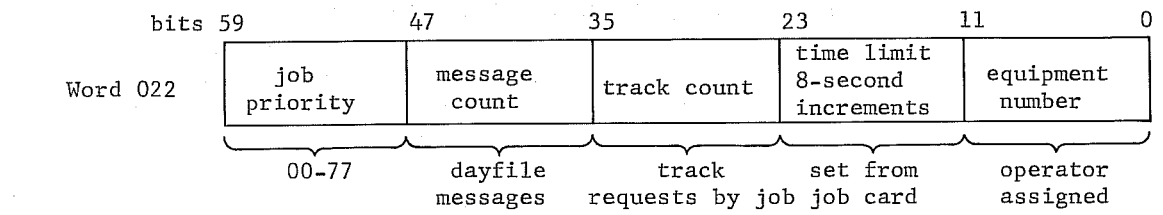
The seven control points, numbered 1 to 7 in central residence are used to record associated activities. There is also a pseudo-control point, numbered 0, to which the monitor program in PP0 and the display program in PP9 are attached; pseudo-control point 8 is used by a storage move program. Any other central or PP activity must be attached to a control point numbered 1 to 7.

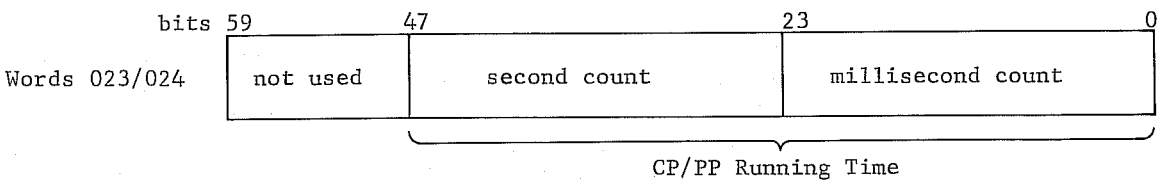
An activity attached to a control point can reserve an area of central memory and may request use of the central memory and of the central processor. PP's may also be attached temporarily to a control point. During normal operation, some control points, used for system operations such as the loading of jobs from a card reader, may never require the central processor.

The 200 octal word central resident area of a control point area (Figure 5) contains information such as the values of reference address (RA) and field length (FL) for its reserved area of central memory. Each type of information is assigned a particular location within the 200 octal word area, so that it can be picked up in a standard way by PP's.

The control point area includes the following information. Addresses are relative to the start of a 200 octal word control point area.







Word 025 is the PP recall input location which preserves an input register message refused because of unreadiness of equipment. It is periodically re-entered into a PP input register location by monitor until it is accepted.

Word 026 holds the simulated sense switch settings entered through control cards or the system display or job display consoles.

Word 027 holds a set of 1-bit indicators representing equipment assigned to this control point.

Words 030-037 contain the last dayfile message or console message associated with the job until replaced by a subsequent dayfile message. The dayfile is the peripheral program which retains the system history.

Words 040-177 hold all control statements in packed display code for the job assigned to this control point. As each statement is processed, the next statement address in word 021 is advanced. An area overflow is detected at load time and the job is rejected with the error message TOO MANY CONTROL CARDS. There is no set number of allowable control cards; this message refers to the packed total length of these cards.

The seven control points allow concurrent operations of up to eight PP programs plus one central processor program.

3.3 EXCHANGE JUMP AREA

Central processor operation is initiated or interrupted by an exchange jump command from MTR. MTR furnishes the central processor with the first word address of a 16-word area in central memory, the exchange jump area, which contains all necessary information about the program to be started or resumed in the central processor. This information and the structure of the 16-word block is shown in Figure 5.

The central processor enters the information about a new program into the appropriate registers and stores the current information of the interrupted program in the same 16-word block in central memory, thereby exchanging two programs. During this exchange, the normal functions of the central processor are not in effect.

All central processor reference addresses to central memory instructions or data are relative to the reference address (RA). The RA and field length (FL) define the central memory limits of a program (RA plus FL); field length is the total program length. The program address register (P) defines the location of a program step within the limits described. Each reference to memory is made to the address specified by P + RA. Therefore, relocation of a central memory program is easily performed by moving the program in memory and resetting RA to the new address in the exchange jump area.

When an exchange jump interrupts the central processor, several hardware steps insure that the interrupted program is left in a state for re-entry:

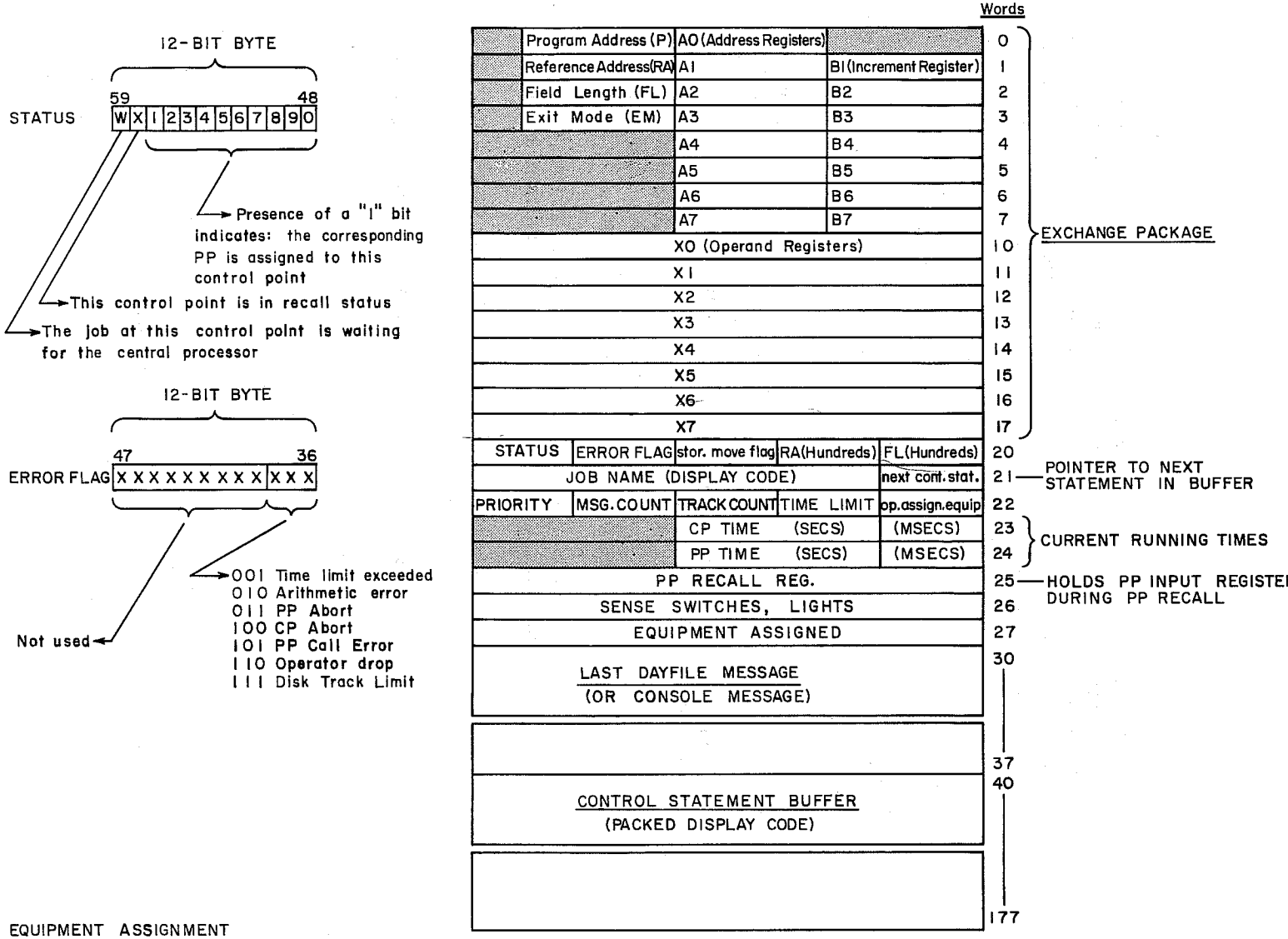
1. Instructions cease after all instructions from the current instruction word in the stack are issued.

2. P is set to the address of the next instruction word to be executed.

3. Issued instructions are executed.

4. The two programs are exchanged.

Each central memory program is assigned to a control point, within which, the first 16 words are reserved for the exchange jump area. When a central program is ready for execution, the initial values of P, RA and FL are entered into its exchange jump area by a PP. If this program's priority is greater than the currently operating central program, MTR initiates this program with an exchange jump command which interrupts the currently running program. If the program's priority is not greater, execution will not begin until the running program terminates or issues a recall request (RCL), or until its priority becomes higher than that of the running program.



EQUIPMENT ASSIGNMENT

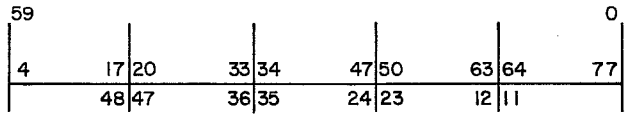


Figure 5. Control Point Areas and Exchange Jump Area

3.4 CONTROL POINT STACKING

When the control point makes use of the central processor, area 000-017 records exchange jump information. The most significant 12 bits of word 020 record the type of processor activity at the control point. The 12 bits are designated: W, X, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0.

A numbered bit is set to one only if the corresponding PP is working for the control point. W and X refer to the central processor. The status of a control point in regard to the central processor is A, B, C, D, E, F, G, W, X, or blank. The monitor maintains an ordered stack of control points between which it switches the central processor. The control point currently using the central processor is listed at the top of the stack and has status A.

W = 0, X = 0 Either the control point has no requirement for the central processor (blank status) or is listed in the stack and has a status from A to G.

W = 1, X = 0 The control point is in waiting (W) status and requires the central processor. W is cleared only when the control point is added to the stack, either when the stack is empty or when the control point's priority exceeds that of the control point with A status. In the latter case, the stack is pushed down, the control point with W status assumes A status, the control point which had A status assumes B, and so on.

W = 0, X = 1 The control point is in X status, awaiting recall. When the central processor is active for the control point with A status, the central program can temporarily relinquish control by entering the recall code RCL in RA + 1. It usually does this when awaiting completion of an input or output function by a PP. (Programs produced by the FORTRAN compiler contain instructions generated to set RCL when necessary). The stack is then pushed up and the status of the control point becomes X. The control point remains in X status until it is recalled and changed to W status, when it may again join the stack. The control point may be recalled by a PP which has completed a task, or by the monitor program, whichever happens first.

When a central program initiates a PP task by entering a code in RA + 1, it does not necessarily lose the central processor. A central program may buffer I/O so that it can operate simultaneously with the PP. However, by examining a word in central memory, a program can determine the progress of the input/output; and if it reaches a point where further progress is temporarily impossible, the central program may enter RCL in RA + 1. The control point then assumes X status.

When a PP has completed an I/O task, the monitor is alerted to recall the control point. If the control point is in X status, the monitor changes it to W; if not, its status is unchanged. When the status of a control point is changed from X to W, the monitor program performs a search for control point priority to determine if the control point should be listed in the stack.

When a control point has X or W status, the exchange jump area of its central program is stored at the same control point.

When a control point is listed in the stack, the exchange area of its central program is stored in the area of the control point above it in the stack. At the bottom of the stack is the idle program (a loop stop with P = 2) to which the monitor switches control when no other program is ready to execute.

The monitor program performs an exchange jump when the stack is pushed up or down. When the stack is to be pushed up the exchange is with the package in the area of the control point whose status is A. This will cause the program corresponding to B (or the idle program) to execute; the stack is then pushed up so that B becomes A. The former A leaves the stack to assume some other status (commonly X) and its registers are preserved in its own control point area.

When a program which had status W is to be introduced to the top of the stack, first the stack is pushed down, then an exchange jump is performed with the program from the new control point area, which assumes A status.

Example:

Control Point Number	Status	Contents of Exchange Jump Area Stored at Control Point
1	W	registers of program 1
2	B	registers of program 5
3	blank	(not requiring central processor)
4	A	registers of program 2
5	C	idle program
6	X	registers of program 6
7	blank	(not requiring central processor)

Since program 4 is active, it has no stored exchange jump area.

The monitor may remove a control point not at the top of the stack; but the control points listed above it must be pushed out of the stack into W status, and exchange jumps performed to bring the program below it (possibly idle) to the top of the stack.

Example:

To drop program 5 (C status) in the previous example:

Δ Exchange jump control point 4, and push up stack, giving W status to program removed (program 4).

† Exchange jump control point 2, and push up stack, giving W status to program removed.

†† Exchange jump control point 5, activating the idle program. Give control point 5 W, X or blank status.

The status of control points after each step is as follows:

Control Point Number	Original	After Δ	After †	After ††
1	W	W	W	W
2	B	A	W	W
3	blank	blank	blank	blank
4	A	W	W	W
5	C	B	A	?
6	X	X	X	X
7	blank	blank	blank	blank

The status of control point 5 depends on the reason that it was removed from the stack. A program dropped as a result of a console command, would assume blank status, and the monitor would reassemble the stack, according to the rules by which programs of W status may enter the stack. A program dropped as a result of a storage move request would assume a W status.

The monitor maintains in PP0 the list of control points in the stack, in PP0 words 60_8 down to 52_8. These words contain the addresses of the central resident areas corresponding to the control points which are in the stack. The address of a control point's central resident area is its number shifted up 7 bit positions (control point 1 at 200_8, control point 7 at 1600_8). Word 60_8 of PP0 lists the control point address for the control point with A status, and a zero word in 60_8 or below indicates the end of the stack.

Pseudo-control point 0 represents the idle program. Pseudo-control point 8 is for the storage move program. The central resident areas for control points 0 and 8 are 0 and 2000_8, respectively, but only a small part of each area is actually used for control point information; word 020_8 of the central resident contains control point zero status, word 021_8 contains control point zero job

name (MONITOR), word 2000_8 contains an exchange jump area for the storage move program.

The monitor also maintains the stack indicators in words $56\text{-}57_8$ of the central resident. The display program can pick them up to indicate the status of control points. Word 60_8 of PP0 always specifies the active control point. If it is zero, the idle program is running, if it is 2000_8 the storage move program is running.

3.5 STORAGE ALLOCATION AND MOVEMENT

Storage is allocated so that areas of central memory allocated to control points lie in the order of the control points, with no gaps between.

Example:

Assuming the central resident and library programs occupy from 0 to 13777_8 a possible arrangement is: (octal addresses)

Control Point Number	Reference Address (RA)	Field Length (FL)
1	14000	4000
2	20000	10000
3	30000	100000
4	130000	0
5	130000	150000
6	300000	0
7	300000	40000

The area of unoccupied central memory starts at RA + FL of control point 7, 340000.

A PP attached to a control point can request or release storage via the monitor program. This commonly takes place when a new job is brought to the control point with a different requirement than the previous job.

When storage allocated to a control point is to be changed, the monitor moves the storage that has been allocated to control points with larger numbers. The first step is to set storage move flags, the middle 12 bits of word 20_8 of the control point area.

A storage request to the monitor from a PP specifies the field length required and the monitor determines whether the present allocation must be changed. If a change is necessary, the monitor sets the storage move flag for control point 4 to the address of the requesting PP's output register, then it sets the storage move flags for control points 5, 6 and 7 and waits until PP activity for control points 4 to 7 is such that their central storage may be moved.

PP programs perform their own central memory relocation for a control point by reading RA and FL from the control point area. A program which occupies a PP for a long time must provide for intermittent relocation pauses. When a PP pauses for relocation, the monitor allows it to continue if the storage move flag for its control point is not set. If the flag is set, the pausing PP must wait until storage has been moved. (A PP does not continue until the monitor has cleared its output register).

When there is no PP activity at the control points with move flags, the monitor empties the stack by successive exchange jumps and push-ups until the idle program is executing. (Control points in the stack assume W status). It then prepares an exchange jump area in 2000_8 for the storage move program, and exchange jumps to the storage move program at 2020_8. Parameters of the storage move program preset in the exchange jump area are as follows:

B1 = RA + FL for control point requiring storage change

B2 = RA + FL for control point 7

B3 = storage increase or decrease (depending on sign) required by requesting control point

The storage move program transfers, forward or backward, the area between B1 and B2 and sets P = 0 upon completion, the monitor returns the idle program, updates RA and FL for the control points, clears the storage move flags, reassembles the stack by a priority search, and clears the output register of the PP which requested the change. A PP requesting storage must check RA and FL of its control point to determine that the change has been made.

The storage move program uses the central processor, but PP activity may continue at control points with a lesser number than that for which the change is being made.

3.6 PP RECALL

The recall facility of a PP is similar to that of the central processor. A PP recall register for each control point allows a PP to be freed if the program in the PP cannot be held up indefinitely; for example, if equipment needed is not ready. The program enters a code into the recall register of a control point and releases its PP. The code should be a PP call of the type normally put in a PP's input register. Later, the monitor will sense a request in the recall register and transfer it to the input register of a free PP. This device is particularly useful when a PP is needed, but cannot immediately proceed.

A PP may set a control point recall register then releases itself, becoming free for any other task. When monitor recalls a PP to the control point, it will use any free PP. If there are several conditions under which a program may choose to be recalled later, the recalled program can determine the stage reached by examining the control point area or by using parameters from its input register (which came from the control point recall register).

The recall facility is normally used when a single transient PP is the only activity at a control point.

3.7 ADVANCING CONTROL POINT STATUS

The main loop of the monitor program in PP0 examines the output register of each PP; and if non-zero, performs the requested function and clears the corresponding output register. If it is not convenient for the monitor to perform the function on this circuit of the main loop (if it must wait for some condition as in the storage move), the output register remains as is and the monitor tries again on the next main loop circuit.

When the monitor needs to know the control point number to which a PP output register request refers, it examines the PP input register. If the central processor is being used by a control point, the monitor examines RA + 1 for a request. If the monitor can meet the request, it does so and clears RA + 1 to inform the central program. If it cannot meet the request on this circuit of the main loop (there is no free PP) the monitor leaves RA + 1 and checks it again on the next main loop circuit. If the central processor is being used by a control point, the monitor also checks the value of P on every circuit of the main loop. P = 0 represents an error exit condition.

On every circuit of the main loop the monitor enters a routine called Advance Control Point Status. This routine promptly exits to the main loop if less than 64 ms. have elapsed since it was last entered. If 64 ms. have elapsed, this routine advances the status of one control point. Word 26_8 of PP0 contains a control point number, which the routine advances by 1, cyclically, in the range 1 to 7, then examines. The routine exits if the control point has no job name, or there is no free PP. Otherwise, it recalls the central processor PP to the

control point, or advances a job to its next phase. Most switching of the central processor results from other parts of the monitor main loop. This routine only performs those actions for which a slower time scale is both necessary and advantageous.

If the X flag for the central point is set, its status is changed to W and a priority search is made in case this control point is now entitled to the central processor. If the PP recall register is non-zero at the current control point, the routine transfers it to the input register of a free PP and performs some associated housekeeping tasks. Having tested for recall, the routine exits, if the status byte (W, X, 1, ..., 9, 0) is non-zero. It also exits if the storage move flag for the control point is set, or the control point is listed in the stack.

If no exit has been made so far, the job has finished all the central processor and peripheral processor activity associated with one phase. The routine exits to the monitor main loop.

3.8 CENTRAL/PP COMMUNICATION

Once each scan, MTR senses the second location (RA + 1) of the running central program. It is through this location that the central program passes requests to the monitor. These may be requests for PP action, or they may be notification of some significant state of the central program. The RA + 1 entries take the following forms:

Call Peripheral	PP program name in display code in upper 18 bits of RA + 1. Lower 36 bits contain the parameters for the peripheral program. MTR clears RA + 1 as soon as requested program is passed to PP for execution.
Recall	RCL in display code in upper 18 bits of RA + 1. MTR exchanges to next program waiting on a priority basis. This request should be used whenever the program cannot continue processing until an outside function is complete. All system peripheral programs recall the central program at completion of requested function.
Abort	ABT in display code in upper 18 bits of RA + 1. MTR calls a peripheral program to advance to next control statement.

MTR also performs the following operations with respect to central program:

Higher priority	MTR interrupts running program in favor of a higher priority program.
Arithmetic exit	MTR exchanges to next equal or lower priority program when central program address (P) becomes zero. Arithmetic exit mode flag is set.
Time limit	MTR exchanges to next central program on a priority basis. When the time limit of the running program is reached, a flag is set.

3.9 MONITOR/PP COMMUNICATION

MTR is preset to process a standard set of function requests from a PP routine. The PP normally enters a request through its resident program which places the request in its output location in central memory. MTR reads the output location and performs the request or assigns the request to another PP (Figure 6).

After entering a function request in the output location, the resident program delays while waiting for the function to be performed. When MTR clears the output register, the request has been completed. Any parameters which are to be returned, or error indications affecting the request are entered in the associated message buffer as indicated in the descriptions of the functions.

The error flag values for MTR Requests are:

1 Time limit

2 Arithmetic error

3 PP abort

4 Central processor abort

5 PP call error

6 Operator drop

7 Disk track limit

The descriptions of the function requests and the actions taken by MTR are given below.

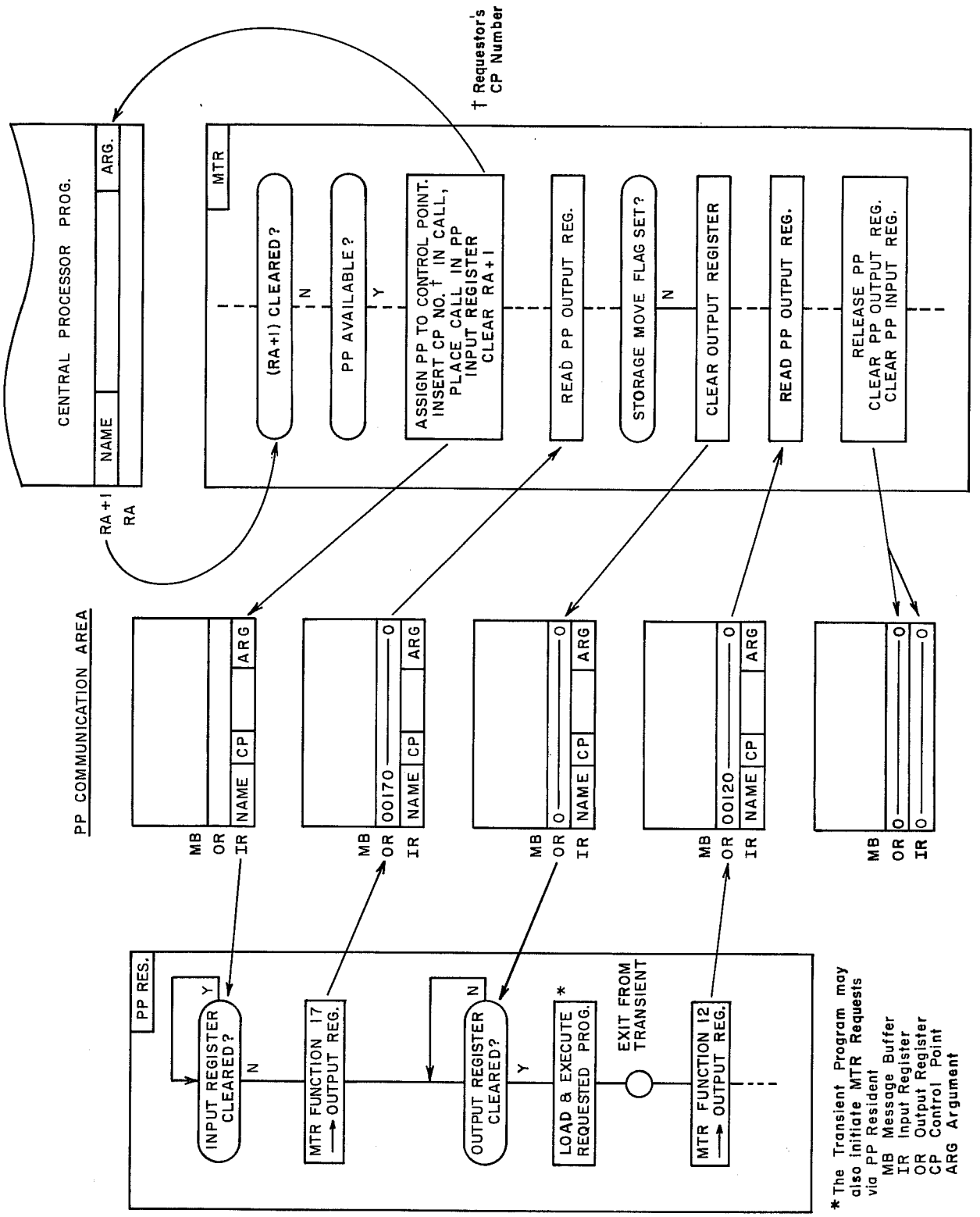


Figure 6. Monitor/PP Communication Area

3.9.1 MTR FUNCTION CODES

The output location contents shown are set by the calling PP.

01 0001, 0000, 0000, 0000, 0000

Process Dayfile Message

PP sets a message in its message buffer (words 2-7 of the communication area) before requesting this function.

MTR transfers the message into the assigned control point dayfile message area along with the time and job name. The message also goes to the dayfile buffer.

02 0002, 00nn, 0000, 0000, 0000

Request Channel nn

MTR assigns channel nn to the requesting processor as soon as it is available by storing the requesting PP number in the channel status table in central resident.

03 0003, 00nn, 0000, 0000, 0000

Drop Channel nn

MTR clears the assignment of channel nn in the channel status table. No check is made to assure that the PP requesting nn to be dropped was the PP originally requesting nn.

04 0004, 0000, 0000, 0000, 0000

Assign PP Time

MTR adds the current time minus PP starting time, in locations 040 - 052 of the central resident, to accumulated time in control point area. MTR also sets a new PP starting time (seconds and milliseconds).

MTR Function Codes	Output location contents set by calling PP
05	0005, 0000, 0000, 0000, 0000

Monitor Step Control

This control is initiated by a keyboard request. MTR sets an internal step control flag and at each subsequent request, MTR pauses for console keyboard input. A space from the keyboard causes MTR to process the request. A period from the keyboard causes MTR to process the request and clear the step control flag to resume high speed operation.

06 0006, 00nn, 0000, 0000, 0000

Request Disk Track

nn = 10, 11 or 12 corresponding to TRT pointers 0, 1 or 2. MTR checks the specified TRT; an available track number is put in the first byte of the processor message buffer, and the TRT is updated. If no track is available, byte 1 of the message buffer is cleared. When the request is for disk 0 (TRT0) the track count in the control point is updated. If the track count limit of 1000_8 is exceeded, error flag 7 is set in the control point area, the error message TRACK LIMIT appears, and the job is terminated.

07 0007, 00nn, tttt, 0000, 0000

Drop Disk Track

nn = 10, 11 or 12 corresponding to TRT 0, 1 or 2. MTR clears track assignment tttt from the specified TRT. If the disk 0 track is being dropped, the control point track count is reduced by 1. This function does not check correspondence between the control point being modified and the control point updated for this track.

10 0010, nnnn, 0000, 0000, 0000

Request Storage

Assigns nnnn hundred octal words of central storage to the control point of the requesting PP if neither of the following conflicts exist:

A PP is waiting for a previous storage request.

The amount of storage requested exceeds available central memory.

MTR Function Codes	Output location contents set by calling PP
10 (cont'd)	When assignment is made, MTR sets storage move flags at all control points above that of the requesting PP. When all activity through the control points ceases, their assigned storage areas are relocated to accommodate the storage request. The requesting PP must sense the field length parameter in its control point to verify the actual assignment. A pause function (17) should precede further requests for storage.
11	0011, 0000, 0000, 0000, 0000 Complete Dayfile The current dayfile buffer is dumped by the monitor. The buffer is added to system dayfile and job output file.
12	0012, 0000, 0000, 0000, 0000 Release PP MTR clears the PP control point assignment, computes PP running time, adds it to accumulated PP time, provides a new PP starting time for subsequent requests, and clears the PP input register.
13	0013, 0000, 0000, 0000, 0000 Abort Control Point The job associated with the requesting processor is terminated. The requesting processor is responsible for an explanatory message in the dayfile. The operation of this function is identical with function 12, except that error flag 3 is set for the abort function in the control point area.
14	0014, tttt, 0000, 0000, 0000 Time Limit A job time limit of tttt time increments is entered at the control point; each time increment is 8 seconds. Any previous time limit is superseded.

MTR Function Codes	Output location contents set by calling PP
15	0015, 0000, 0000, 0000, 0000

Request Central Processor

MTR sets the central waiting flag (W) at the control point and searches for job priorities to initiate central processor action. The request is ignored under the following conditions:

Error flag is set.

(RA + 1) = END request.

A central memory program has been assigned to this control point.

16 0016, 0000, 0000, 0000, 0000

Drop Central Processor

Execution of the central processor job at the PP control point is dropped.

This function is also used to release a control point assigned to PP I/O buffering; the PP is also dropped by this operation.

17 0017, 0000, 0000, 0000, 0000

Pause for Relocation

This function allows monitor to move central storage for the associated job. As long as the move flag at the control point is set, this function inhibits any further action of the requesting PP. When the move flag is cleared, the pause function is ended. (See function 10.)

The requesting processor should check the reference address for the control point after completion of this function to determine if central storage for its job has been moved.

0020, 0000, 0000, 0000, 0000

Request PP

This function requests initiation of another PP. The first word of the requesting PP message buffer contains the input location data for the new PP including the control point to which it should be assigned by MTR. The input location address of the new PP is placed in the first byte of the requesting PP message buffer. If no PP is available, a zero byte is returned.

MTR Function Codes	Output location contents set by calling PP
21	0021, 0000, 0000, 0000, 0000

Recall Central Processor

The central program associated with the requesting PP is restarted if the central recall flag (X) is set.

22	0022, nnnn, 0000, 0000, 0000

Request Equipment

MTR searches the equipment status table (EST) for an equipment of type nnnn and assigns it to the control point. EST is updated and MTR places the equipment number in the first byte of the PP message buffer. If the equipment is not available, a zero byte is returned.

23	0023, 00nn, 0000, 0000, 0000

Drop Equipment

MTR drops equipment number nn from the control point and updates EST to indicate this equipment is free for reassignment. There is no check by monitor to insure that the equipment number dropped from EST was assigned to this control point.

24	0024, 00nn, 0000, 0000, 0000

Request Priority

Assigns priority nn to the control point, and searches priorities to initiate central processor action.

25	0025, 000n, 0000, 0000, 0000

Assign Error Exit Mode

MTR drops central processor execution for the job at the control point and assigns the value n to the exit mode field in the control point exchange jump information area.

MTR Function Codes	Output location contents set by calling PP
26	Unassigned, ignored
27	0027, 0000, 0000, 0000, 0000

Toggle Simulator

A simulator of the central computer is called from the peripheral library or dropped as required. A call to an unavailable PP will cause the function to be ignored.

30 0030, 000n, 0000, 0000, 0000

Operator Drop

Drop the job at control point n and set error flag 6 at control point.

31 0031, 00nn, 0000, 0000, 0000

Equipment On

Clear the lockout bit for equipment nn in the equipment status table (EST).

32 0032, 00nn, 0000, 0000, 0000

Equipment Off

Set the lockout bit for equipment nn in the equipment status table (EST).

33 0033, 00nn, 000p, 0000, 0000

Assign Equipment

Assign equipment number nn to control point p and enter nn in the control point area for operator assignment. If equipment nn is busy, no assignment is made.

34–37 Unassigned, ignored.

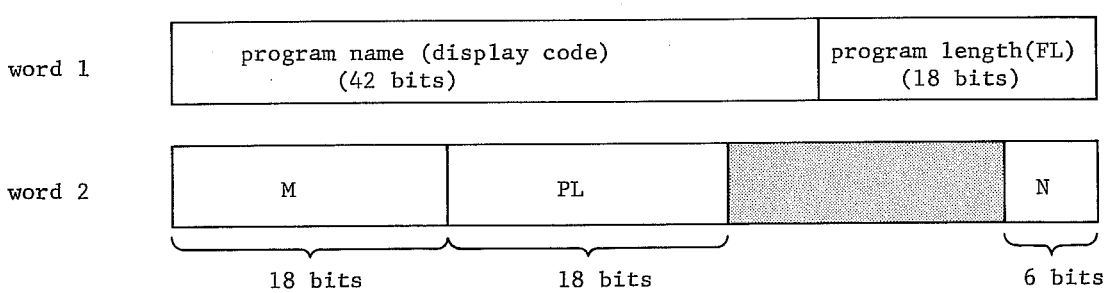
3.10 CALLING SEQUENCES

Programs and subroutines are classed as central processor programs or PP programs. Those most frequently used, of both classes, are stored in central memory; the remainder reside on the magnetic disk.

3.10.1 CENTRAL PROGRAMS

Central programs may be loaded from the resident subroutine library (RSL) which is an area of central memory, from the disk file indexed by the central library directory (CLD), from the INPUT disk file as a binary version of a previously compiled program, or from the named disk file as a binary version of a previously compiled program or a current compilation.

The first two words of the binary version of the program correspond to RA and RA + 1 which contain:



M number of words of executable instructions; does not include temporaries, constants, or variables

PL local length of program. If FL ≠ PL, PL + RA is address of next subroutine associated with this program

N number of parameters

The programs are loaded by a PP program which sets the exchange jump information (RA, FL, P) in the assigned control point.

P N + 2, the number of parameters plus 2

FL specified program field length from word 1

RA value given in word 20, byte 4 of assigned control point area

The first two words of the binary program are cleared before execution.

Central program parameters begin at RA + 2; they are file names in display code, left justified in the word. The set of file names is stored in the parameter area at the time the program is compiled. If parameters appear in control card which calls a central program for execution, these parameters replace the original values in the program parameter area. If no parameters are used, or if fewer appear on the control card than are specified in word 2, the original values are effective at run time. Embedded missing parameters are not allowed on the control card.

The list of parameters at the beginning of the program (starting at RA + 2) must be followed by a zero word. This is the entry line for the program. The first word for execution follows this entry line.

Coding for a Central Memory Program

Location	Tag	Contents	Remarks
RA		0 ⟷ 0	Program error exit location
RA+1		0 ⟷ 0	MTR communication
RA+2	A1	file name 1	Parameter 1
RA+3	A2	file name 2	Parameter 2
.	.	.	.
.	.	.	.
.	.	.	.
RA+n+1	An	file name n	Parameter n
RA+n+2		0 ⟷ 0	Program entry location
RA+n+3		Op.	First program instruction

Example:

Bits	59 ... 18	17 ... 0
RA	ONEIDAA	100000
RA+1	0 ⟷ 0	03
RA+2	INPUT	
RA+3	OUTPUT	
RA+4	TAPE2	
RA+5	0 ⟷ 0	
RA+6	(first program instruction)	

RA and RA+1 are cleared before execution begins

Program entry at RA+5, with program address = 5 and Field length = 100000

File names used in the program are, INPUT, OUTPUT, and TAPE2

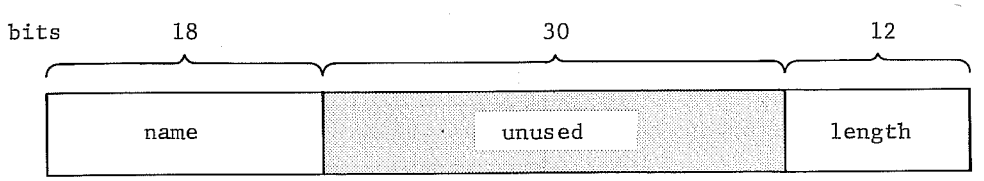
A central program may request MTR action by entering the name of a routine in display code left-justified in location RA+1. MTR periodically scans RA+1 of the running program for such requests. When RA+1 is non-zero, MTR passes the value to a PP for action. Requests such as input/output (CIO) are processed in this manner. When RA+1 is cleared, the running program may assume that the request has been honored, though not necessarily completed. Any parameters associated with the request must be put in the lower 36 bits of location RA+1 by the calling program. The format of the parameter list is dependent upon the program called.

Whenever a function request is given in RA+1 and the central program cannot continue processing until the function is complete, the program should give a recall (RCL) request after the function request has been honored. This allows another central program of lower or equal priority to continue execution, thereby obtaining more efficient use of the central processor. All system peripheral packages are written to recall the central program after the requested operation is complete.

A central program is normally terminated by entering END in display code in RA+1. The monitor will then call a PP routine to advance the job to the next control statement. Abnormal termination may be effected by entering ABT in display code in RA+1 or by executing an instruction which causes an error exit.

3.10.2 PP PROGRAMS

PP programs, used as system routines or central program peripheral routines, are stored in central memory in the resident peripheral library (RPL) or listed in the peripheral library directory (PLD) and stored in the disk file. The binary format is the same for both types. The first word of the program contains the program name and the number of central memory words occupied by the binary program.



Basic Programs

This central memory control word (five PP words) is not part of the executable code. The first word of the executable PP program is loaded at location 1000. The control word is loaded at PP location 0773-0777. Programs of this type are called basic programs. Loading should begin five PP locations before the actual transfer location since the first word of the program is part of the description and is not executable.

After loading, the PP resident routine transfers control to location 1000 to begin execution of the basic program. Basic programs may call overlay routines by searching RPL and PLD. Overlay routines are generally loaded beginning at address 1773 for execution beginning at location 2000. Each PP program and overlay must be coded to begin at a specific address and is loaded into PP memory at that address minus five. Basic programs begin at location 1000 since the PP resident routine will always load at 0773 and transfer control to location 1000.

A basic program may in turn call for overlays which it must load itself. They may be loaded anywhere in the PP memory but are usually loaded at location 2000 minus five. PP program names can be up to three characters long. If the program is to be callable by a central program, the first character must be a letter. If the PP program is a system peripheral routine (not callable by a central program), the first character must be a number.

When a basic program calls on an overlay, required information is usually passed through preassigned locations in the PP lower memory (locations 0000-0074).

System peripheral routine names beginning with the number 1 are loaded in PP memory starting at location 0773; those names beginning with 2 are loaded at 1773. To prevent destruction of the resident program, a peripheral routine must not start prior to location 1000; locations 0000 through 0074 may be used for temporary storage.

PP programs communicate with MTR through the input and output locations assigned to the particular PP in central memory. A central memory control point is assigned to each PP when it is executing a called program. This area contains all pertinent information about the central program requesting an operation, or about the buffer area for an I/O operation.

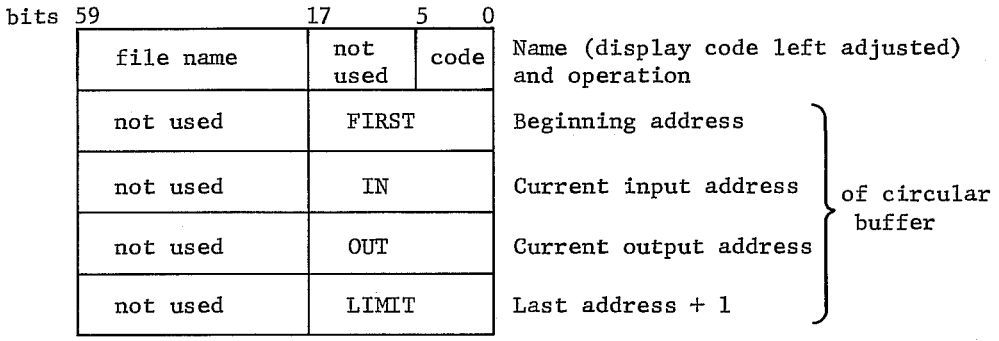
When the PP program needs an operation outside its realm, or a function to be supervised by MTR, it enters a request in its output location. Upon completing the request, MTR clears the PP output location. Information about a function is passed to MTR through the message buffer which follows the output location in central memory. Information is also returned to the requesting program in this area.

To terminate, the overlay generally returns to the calling PP program by exiting through the entry point of the overlay. To terminate a basic program, the program must clear the assigned PP input location (MTR function 12) and return control to PP resident by jumping to location 0100.

3.11 CIRCULAR BUFFER I/O

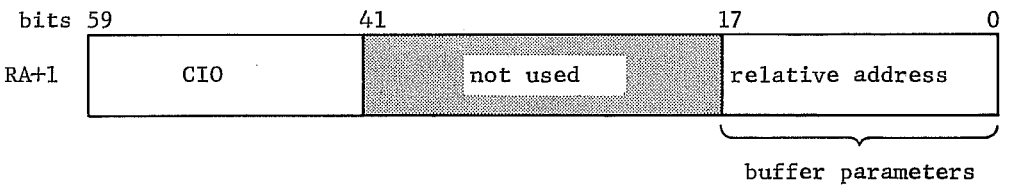
The circular buffer I/O (CIO) program may be called to a PP to perform input and output between a file and a circular central memory buffer. The user specifies a file name and operation code, plus information about the buffer, then CIO performs the operation.

Before calling CIO, circular buffer parameters must be set in five central memory words as follows:



The buffer and buffer parameter area must be within the field length of the job, and addresses are relative (address 0 for the job in absolute word RA).

A central program can call on CIO by entering in its word 1 (absolute RA+1) the code CIO and the (relative) address of the circular buffer parameters.



Example:

CBP is the symbolic address of the parameters:

```
SX6 = 031117B        CIO in display code
LX6  42              To top of X6
SX5 = CBP
IX6 = X6+X5          Add in address
SA6 = 1              Write to (RA+1)
```

PP calls in RA+1 of a running central program are detected by the monitor in PP0. The monitor finds a free PP to perform the task, then clears RA+1 to indicate that the task is started, not completed. The operation code in the first parameter word is even, and one is added to it by the PP when it has performed the operation. PP also updates IN and OUT in the parameter area, according to the function performed.

The central program continues after setting RA+1 but must not use RA+1 again until it is cleared by the monitor. This is programmed either by looping on non-zero RA+1 during the short time taken by the monitor to detect a call, or by checking that RA+1 is zero before making a further call.

The central program should inform the monitor with an RCL call if it is unable to proceed further for the time being, then the monitor will switch to another program. A central program may buffer input and output in order to proceed to a certain stage before being delayed.

Continuing the last example, if the central program chooses to wait until the PP has finished:

```
L1    SA1 = 1          Read (RA+1) to X1
      NZ   X1, L1      Wait till clear
      SA1 = CBP        Get first parameter word
      LX1  59          Determine if odd (PP finished)
      NG   OK          Continue to OK if PP finished
      SA6 = 220314B    RCL code
      LX6  42          To top of X6
      SA6 = 1          Recall code to (RA+1)
      JP   L1          Loop
OK                     Continue processing
```

The recall code causes control to be taken from the program. Control is resumed (subject to priority) in approximately 250 ms. or when a PP completing an operation tells the monitor to recall the corresponding central program. The monitor clears RA+1 after receiving the RCL. The loop at L1 holds up the program until the monitor switches control, but allows the program to continue when control is returned.

Most users need not concern themselves with CIO and RCL use since their FORTRAN programs are compiled to include the necessary calls. However, users programming input and output in machine instructions must remember to explicitly drop the central processor with a RCL when awaiting progress of a PP task. A PP call does not in itself cause the central processor to be switched; progress of a CIO operation is determined by examination of the parameter area.

The current data in a circular buffer starts at OUT and continues (possibly round the end of the buffer) to IN-1.

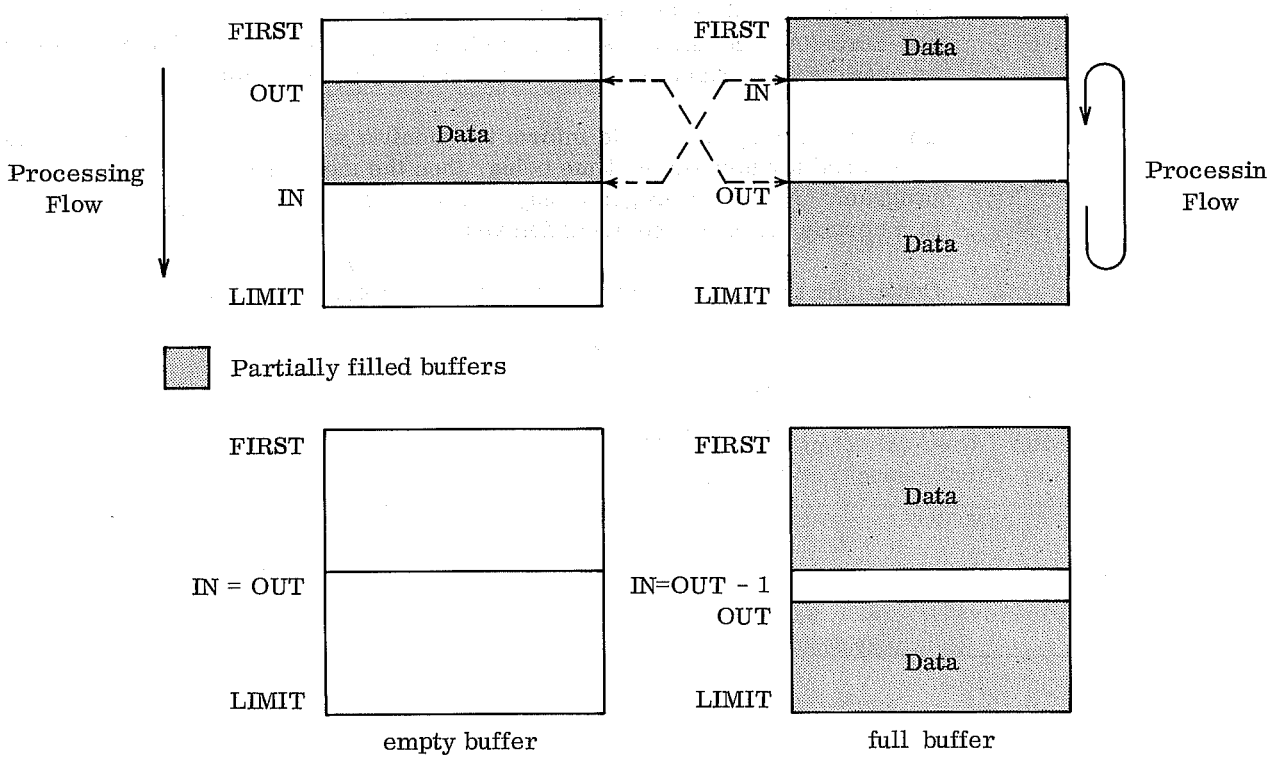


Figure 7. Circular Buffer I/O (CIO) Processing Flow

When a buffer is filled to capacity, the unused word between IN and OUT distinguishes it from an empty buffer for which IN = OUT. The capacity of a buffer is LIMIT - FIRST - 1. A buffer is generally initialized with IN = OUT = FIRST, then IN and OUT circle the buffer as data is inserted and extracted.

IN defines the address for insertion of data into a buffer. As data is inserted, IN is stepped round the buffer but never so as to catch up with OUT to avoid overstepping buffer capacity.

OUT defines the address for extraction of data from a buffer. As data is extracted, OUT is stepped round the buffer but never beyond IN since the buffer is empty by the time OUT = IN.

Commonly CIO, moving IN, reads data from a file to the buffer, and the data is extracted by a central program moving OUT; or a central program, moving IN, inserts data into the buffer, and the data is written to a file by CIO, moving OUT.

When CIO writes to a file, only data sufficient to write unit physical records for the medium of the file is extracted from the circular buffer. When writing to disk, for example, CIO takes only 64 word sectors from the buffer, unless an end-of-record or end-of-file is requested when a shorter sector of data can be written to empty the buffer. Similarly, on reading from a file, CIO will transfer another unit physical record to the buffer only if there is room for it. At the end of a read or write call, the positions of IN and OUT show how much data is left in the circular buffer.

The operation code given to CIO via the last 6 bits of the first buffer parameter word is also known as the buffer status since CIO returns a code to these 6 bits when a call has been completed. CIO is given an even code and adds one to the code after completing a called operation. For read operations, the code returned also indicates whether end-of-record or end-of-file terminated the read.

The two octal digits of the buffer status bits have the following meanings:

Value	First Digit	Second Digit
0	Not used	Request coded read
1	Buffer I/O	Coded read completed
2	End Record	Request binary read
3	File Mark	Binary read completed
4	Backspace	Request coded write
5	Rewind	Coded write completed
6	Rewind unload	Request binary write
7	Not used	Coded write completed

A command given to CIO is even, the first digit specifies the type of operation, and the second specifies the direction (read/write) and mode (coded/binary). For buffer I/O, as many physical records as possible are transferred between file and buffer. End record or End file are used in writing to empty the buffer and write end-of-logical-record or end-of-file for the particular medium.

One is added to the code to inform the program when a PP has finished a CIO call. For reading, the first digit may be changed by CIO to indicate that the transfer was terminated when end-of-logical-record (EOR) or end-of-file (EOF) was encountered.

A programmer should make certain that a buffer is emptied of all information to be read, before it is reused.

A backspace operation (40 = coded, 42 = binary) sets the parameters so data of the new file position can be extracted from OUT. The amount of information in the buffer (up to IN) depends upon the medium and the previous values of IN and OUT.

The CIO package is a PP library routine which determines the medium of a file, then calls in an overlay (listed below) to perform the operation:

2BD	Backspace disk	2RD	Read disk
2BT	Backspace tape	2RT	Read tape (or rewind)
2LP	Line printer	2WD	Write disk
2PC	Punch cards	2WT	Write tape
2RC	Read cards	2BP	Check legality of parameters

CIO is usually called by a central program, but a PP can set up a buffer and its parameters in central memory and call on another PP to perform a CIO operation. However, many PP packages which use circular central memory buffers call the required overlay directly into their own PP. The READ package (1LJ) transferring jobs from card to disk, for example, uses 2RC to read cards to a circular buffer, and 2WD to write them to disk.

3.11.1 BUFFER CODES

In the following discussion, the expressions in parentheses are the binary values which correspond to the buffer status octal digits, x may be either 0 or 1.

Input (001, 0x0)

File is read into circular buffer until buffer is filled (001, 0x1) or until end record (010, 0x1) or file mark (011, 0x1). The mode bit (binary/BCD) is ignored only when reading from disk or one-inch tape. A file mark response should never occur with data.

Output (001, 1x0)

Data in circular buffer is recorded on file for as many complete physical records as available data. No partial end record is made. (001, 1x1).

End Record (010, 1x0)

Data in circular buffer is recorded on file including a short physical record to mark end of logical record. The last physical record may be of zero length. IN=OUT=FIRST. (010, 1x1)

File Mark (011, 1x0)

a) Data in circular buffer; or

b) Last buffer status (001, 1xx)

   Data in circular buffer is recorded on file including a short physical record to mark end of logical record. Then a file mark is recorded. IN=OUT=FIRST (011, 1x1)

c) All others

   A file mark is recorded. IN=OUT=FIRST. (011, 1x1)

Backspace Binary (100, x10)

Backspace to end of last record. A file mark is considered a record in this case. IN=OUT=FIRST. (100, 011)

Backspace Coded (100, x00)

Backspace one coded line. A file mark is considered a coded line in this case. The last physical record will be left in the buffer beginning at FIRST. IN and OUT will be adjusted for a one line backspace. (100, 001)

Rewind (101, xx0)

Rewind the file. IN=OUT=FIRST. (101, 0x1)

Rewind Unload (110, xx0)

Rewind and unload the file. IN=OUT=FIRST. (110, 0x1)

3.11.2 OPERATION

All parameter values (IN, OUT, FIRST, LIMIT) must be set by the calling program. For input, CIO reads data into the buffer beginning at IN. CIO continues reading as long as there is storage available or until there is no more data. IN is advanced by one for each data word read. When the value of IN=OUT-1, the buffer is full and the operation complete.

If IN ≥ OUT, IN is advanced to LIMIT-1 and CIO automatically resets IN to FIRST and continues reading all in the same read operation until IN ≤ OUT-1. The maximum buffer capacity for any read is LIMIT-FIRST-1.

For output, CIO writes data from the buffer beginning at OUT. Only complete data blocks are written to the output file with no partial end record written unless the end of record or end of file function is given in the CIO call.

If EOR or EOF is not selected, CIO continues writing until there is not room in the buffer for a full block. If EOR or EOF is selected, CIO writes until OUT= IN and then sets IN=OUT=FIRST. OUT is advanced by one for each word written When OUT=LIMIT-1, CIO automatically resets OUT to FIRST and continues the write operation.

The central program must complete writing a record before requesting a backspace, rewind, or mode change but it need not complete writing a record before requesting a file mark.

The central program must complete reading a record before beginning output data to the buffer.

Binary tapes and coded one-inch tapes record 1000 octal word physical records in odd parity: no special control words are added. A zero length physical record is generated by recording a partial word (4 bytes). Coded one-inch tapes use packed display code with short word separators. Coded half-inch tapes record 120 character BCD code in even parity.

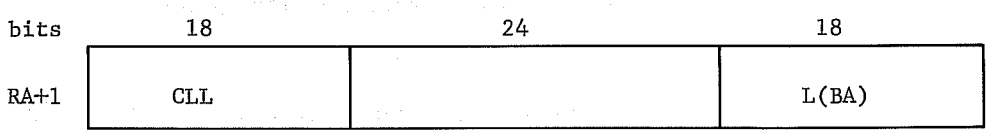
A one-inch tape with mixed binary and coded records presents problems if a backspace crosses a mode boundary. A problem exists in mode change from coded input to output on one-inch tape if the file is positioned between two lines of code. No problem exists if the file is positioned before or after a file mark. No problem exists in mode change on binary files.

A disk file cannot be substituted for a tape file if the file has multiple file marks or has data recorded after a file mark.

3.12 CALLING PROGRAMS

3.12.1 CALL CENTRAL OVERLAY

The CLL peripheral program loads one or more overlays into an area specified by a central memory program. This routine is called from a central program by placing the name CLL in display code in the upper 18 bits of RA+1 of the calling program. The low order 18 bits contain the relative address of the parameters (BA).



The location BA must be the first word of the following parameters, which contain:

BA		FWA
BA+1		LIMIT
BA+2	OVER1	FWA1
BA+3	OVER2	FWA2
BA+n+1	OVERn	FWAn
BA+n+2	(zero word)	

FWA	beginning address for first overlay
LIMIT	Limit address for group of overlays
FWAn	beginning address of OVERn (set by CLL)
OVERn	Name of overlay

CLL searches for the overlays in the order named. The order of search is:

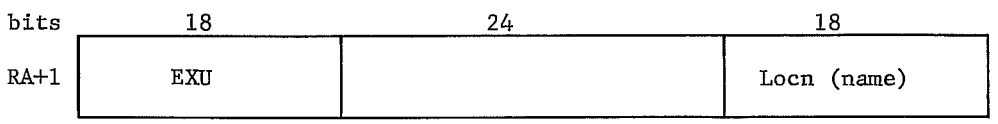
1. Resident Subroutine Library (RSL)
2. Control Library Directory (CLD)
3. Assigned job files

An overlay is loaded into central memory at the next available address beginning with FWA. This address is entered into BA+1+n of overlay n. CALL clears (BA) at the completion of the call.

If an overlay cannot be located, the address is left cleared in the BA region. If an overlay exceeds LIMIT, 777777 is entered as the value for FWA_n. The last overlay parameter must be followed by a cleared word.

3.12.2 CALL AND EXECUTE

EXU (call and execute) is a peripheral program which loads a called program to replace the calling program. The calling program is destroyed. The Monitor (MTR), in checking RA+1, calls EXU to overlay the calling program with the program identified at the address in the lower 18 bits of RA+1.



EXU is in display code

Locn (name) is the location of the name of the routine to be called and executed; name is in display code left justified.

3.12.3 DUMP STORAGE

This peripheral program may be called from a display console with a control card in any of the forms shown below: An octal jump is entered in the output file with the central storage address and one data word per line.

DMP.

dumps the exchange area into the output file.

DMP, 3400.

dumps from the reference address to the parameter address.

DMP (4000, 6000)

dumps from the first address specified to the second.

3.12.4 LOAD BINARY CORRECTIONS

This peripheral program may be called with a control card or from a display console. Binary corrections are read from the input file and entered in central storage. If a parameter is specified in the program call, binary cards are loaded beginning with that address; otherwise, loading begins at the reference address. Only one record is read from the input file. A call must be made for

each block of data to be loaded. This program may be called with either of the following formats:

LBC.

LBC. 2300.

3.12.5 LOAD OCTAL CORRECTIONS

This peripheral program may be called with a control card or at a display console. Octal corrections are read from the input file and entered in central storage. The octal cards used for these corrections must be in the following format:

```
23001 45020 04000 00042 00044
```

Address begins in column 1; leading zeros may be dropped in the address. The data word begins in column 7; spacing in the data word is not important but the word must contain 20 digits. Several formats may be used to call this program:

LOC.

reads all of the correction cards in the next input file record and modifies central storage accordingly.

LOC, 1000.

clears central storage from the reference address to the specified address. The correction cards are then read from the input file.

LOC (2022, 3465)

clears central storage from the first specified address to the second. The correction cards are then read from the input file. This program may be called to clear storage by providing an empty record in the input file.

3.12.6 PUNCH BINARY CARDS

This peripheral program, which may be called with a control card or at the display console, punches a deck of binary cards directly from central storage. Storage is not modified by this operation. The formats that may be used for the call are shown below:

PBC, 2000.

a binary deck is punched from the reference address to the specified address.

PBC (2000, 3000)

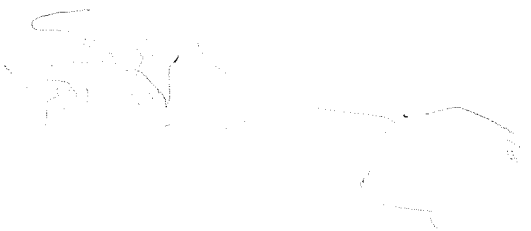
a binary deck is punched from the first specified address to the second.

PBC.

punches a binary deck using the first word in central storage as a control word for deck length. The deck always begins at the reference address and terminates one address less than that indicated in the lower 18 bits of the first word. This call may be used for punching any central or peripheral program in standard format.

3.12.7 DAYFILE MESSAGE

This peripheral program is called by entering MSG. in display code (left adjusted) in RA+1. The lowest order 18 bits of the call word indicate the central storage address for the parameter which consists of a sequence of words in display code to be entered in the dayfile. The time and job name are automatically inserted before the ensuing data. The data is terminated by a cleared byte.



EQUIPMENT USE AND FILE STRUCTURE 4

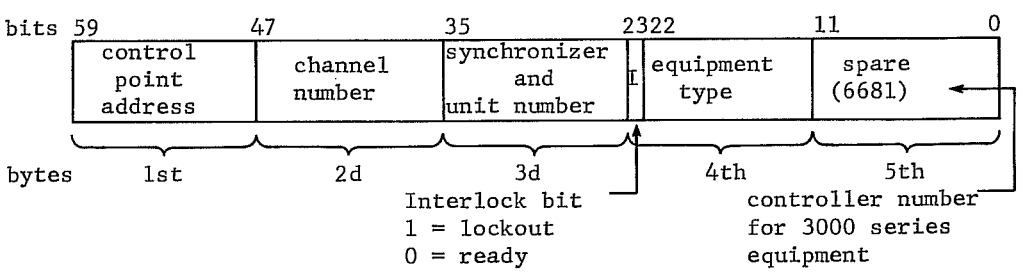
4.1 EQUIPMENT/CHANNEL AND FILE TABLES

The tables described in this section are the keys to all PP I/O requests. A request entered into a PP specifies an operation for a named file. With this information and the control point number, the PP program uses the tables to find which piece of I/O equipment it should drive and the attendant information concerning channel, synchronizer, unit number, and so forth.

4.1.1 EQUIPMENT STATUS TABLE

The Equipment Status Table (EST) in the central resident area indicates the status of available equipment. A two-digit octal number giving the relative address of a one-word table entry identifies each piece of equipment. The fourth byte of a table entry contains a two-character mnemonic type. A control point can request an equipment by number or type from the monitor, or the operator may be requested to assign a numbered equipment. With the exception of disk units, an equipment may be assigned to only one control point at a time.

Format for a one-word table entry:



An equipment status word contains the following information:

First byte — Address of control point (number times 200_8) to which equipment is currently assigned. Zero if equipment is not assigned or is a disk unit.

Second byte — Number of channel to which equipment is attached.

Third byte — Equipment synchronizer and unit number.

Fourth byte — Equipment type in display code:

DA	channel zero disk unit
DB	channel one disk unit
DC	channel two disk unit
DS	display console
CP	card punch
CR	card reader
LP	line printer
MT	607 (1/2") magnetic tape unit
WT	626 (1") magnetic tape unit
PL	plotter

Each code occupies only 11 bits; the 12th bit of the byte is an interlock for equipment which may not be used. This bit is one when the equipment is not available, zero when it is available. It may be turned on and off by a console command, so the operator can inform the system whether an equipment is up or down.

Fifth byte — This entry is not used.

The position of an equipment in the table is often assigned mnemonically; for example:

equipment 62 = tape on channel 6, unit 2

The equipment numbers may be assigned by the installation, and corresponding positions in the equipment status table set accordingly. Mnemonics are convenient for operators assigning tape units from the console. The quipment number is the same as the relative location of the equipment data in the equipment status table. The correspondence to tapes for a typical system is shown on the following page.

TABLE 2. EQUIPMENT/CHANNEL NUMBERS

Equipment	Channel	Type
00	0	Disk File
01	1	
02	2	
04	4	Card Reader (405B)
05	12	
06	13	Card Punch (415B)
07	13	Printer (501B)
10	10	Display Console
11	11	
30	3	Tape Transport (626B) (1")
31	3	
32	3	
33	3	
40	4	Tape Transport (607B) (1/2")
41	4	
42	4	
43	4	
50	5	
51	5	
52	5	
53	5	
60	6	Tape Transport (626B) (1")
61	6	
62	6	
63	6	
70	7	
71	7	
72	7	
73	7	

Word 027_8 of a control point area records the numbers of equipment (except disk) currently assigned to the control point. This allows ten equipment reservations per control point. For an object job, card reader and printer are not normally assigned explicitly, since cards will have already been stored on disk by READ and printer output will go to disk to be printed later by PRINT.

4.1.2 CHANNEL STATUS TABLE

A PP must not drive an equipment before reserving the appropriate channel. The channel number, determined from the equipment table entry should be reserved by a request to the monitor and released after use.

The monitor records the assignment of channels to PP's in the channel status table (CST) which occupies three words (0015 - 0017_8) of central resident. The first 12 bytes are associated with corresponding I/O channels. A cleared byte indicates the channel is not assigned; a PP number indicates a channel is assigned to that PP.

The diagram below indicates the assignment of each channel to a respective byte in central memory words 15, 16, and 17. Pseudo-channels 14 and 15 are used for access to file name and file status table entries. If assigned, the PP number is entered in display code.

Channels

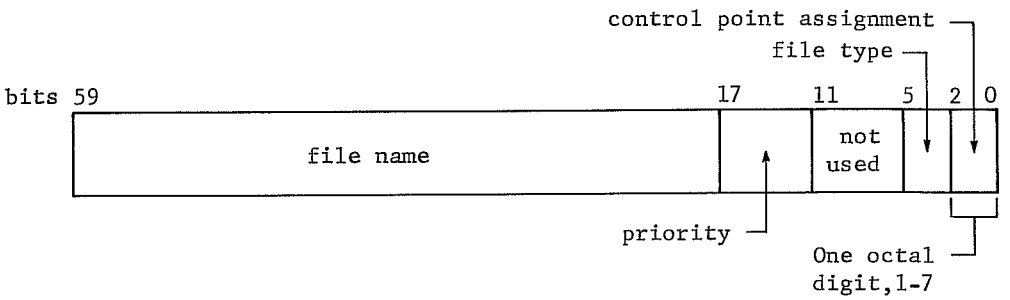
Central Memory words	bits 59	47	35	23	11	0
15	0	1	2	3	4	
16	5	6	7	10	11	
17	12	13	pseudo 14	pseudo 15		

4.1.3 FILE NAME/ STATUS TABLES

The name and status of all files are stored in central memory. The location of this table is stored in central memory location 0004. An area of the central resident contains a two word entry for each current file in the system. The first word of an entry belongs to the file name table, the second to the file status table.

FIRST WORD

A file name table word is set as follows:



A file name of up to seven alphanumeric characters, starting with a letter, is stored left justified in display code.

A priority is set for files of type input and output.

The file type bits distinguish four types of file:

0 INPUT file (stored form on disk of a stacked job)

1 OUTPUT file (stored form on disk of output awaiting printing)

2 COMMON file (may be passed on from one job to another)

3 LOCAL file (discarded at the end of a job)

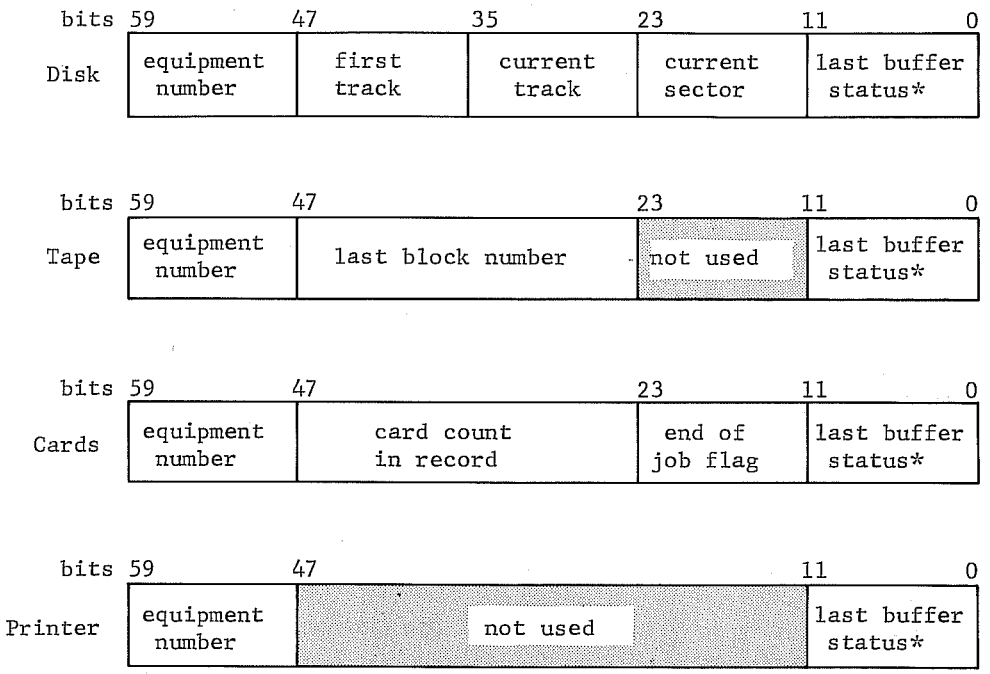
Files may be switched from local to common with the COMMON control card or from common to local with the RELEASE control card.

The above terms indicating file types are not external descriptions; they are internal designations which allow the system and user to maintain control over the use of the files. That is, a file which is classed as local may not be used by any job other than that to which it is local (same control point).

Control point assignment is the number of the control point to which the file is currently assigned. (Zero if unassigned).

SECOND WORD

The file status table entry corresponding to a file name table entry, contains in the top byte the physical equipment number (a pointer to an equipment status table entry). The remainder of the word depends upon the type of equipment.



The operations which may be performed on a file include read and write (coded or binary), backspace, write end record, and write end file mark. Files are stored serially; read and write refer to the next position of the file. Equivalence is preserved subject to the limitations of the equipment; for instance, a card or printer file may not be backspaced; also a file is considered to extend only as far as the last record written. A file on disk may have only one file mark after the last record; therefore if an object program is to preserve equivalence between tape and disk it should not write more than one file mark.

An operation on a named file is performed when file name, location of central memory buffers, and a code for the operation are specified (CIO, section 3.11). A PP can look up the name in the file name table and the equipment number from the corresponding file status table entry and perform the requested operation; the operation code is even. When the operation is complete, one is added to the code and it is entered in the last buffer status area of the file status table entry. This serves as an interlock so only one PP at a time uses the file (status is even when the file is active).

*Interlock: even, active; odd, inactive

4.2 FILE FORMAT

Files may 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 may operate on named files and the actual medium of a file can be specified on control cards; disk storage is assumed if no assignment is made.

Except for a tape file which may have more than one file mark, files consist of a single physical file divided into logical records. A logical record consists of a number of 60-bit words containing either coded or binary information. The form of storage and method of separating logical records depends upon the equipment.

Coded information is stored internally, in display code on either a disk sector or a magnetic tape record, with a maximum of 1000_8 central memory words. 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.

4.3 FILE NAMES

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.

File names must begin with an alphabetic character and may have a maximum of seven alphanumeric characters.

Three special file names, INPUT, OUTPUT, DAYFILE, are implied with each job. These names must not be used for temporary files and other I/O files.

INPUT - the file from which the job cards are read. Each job file is assigned the name appearing on the job card when loaded. As the job file is picked up for processing, the job file name is changed to INPUT.

OUTPUT - the file which ends in printer copy at the end of the job. During job processing, this file name (OUTPUT) is used to collect all records to be printed. When the job is completed, this file name is changed from OUTPUT to the name which appeared on the job card.

DAYFILE - a disk file which contains the system history (dayfile) messages for all jobs.

The total dayfile may be dumped with the following job deck structure:

```
JOBn,p,t,fl.
COMMON DAYFILE.
BKSP (DAYFILE)
ASSIGN52,A.
COPYBR (DAYFILE,A)
```

The dump will be on magnetic tape 52 in binary format with each line shifted one character position to the right and a leading blank added. Other formats may be selected with other copy routines. The dayfile is not updated while this job is being executed.

TAPEnn.

The characters, TAPE, are added by the FORTRAN compiler whenever a program I/O statement refers to unit nn. If unit nn is to be designated as other than a disk file, the name TAPEnn must be used in the control cards such as COMMON, REQUEST, etc.

Example:

```
READ(5,10) list
```

This FORTRAN statement would generate a reference to file TAPE5. The control card necessary for the file to be a tape would be:

```
ASSIGN MT, TAPE5.
```
or
```
ASSIGN 50, TAPE5.
```

```
REQUEST TAPES.
```
or with subsequent operator assignment.

4.4 CARD FILES

The format of a card file is as follows:

Column 1

7, 8, 9	End of logical record
6, 7, 8, 9	End of file
7, 9	Binary card (Figure 8)
7 and 9 not both in column 1	Coded card

A binary card can contain up to 15 central memory words starting at column 3. Column 1 also contains a central memory word count in rows 0, 1, 2 and 3 plus a check indicator in row 4. If row 4 of column 1 is zero, column 2 is used as a checksum for the card on input; if column 4 is one, no check is performed on input.

Column 78 and 79 of a binary card is not used, and column 80 contains a binary serial number. If a logical record is output on the card punch, each card has a checksum in column 2 and a serial number in column 80, which orders it within the logical record.

Coded cards are translated on input from Hollerith to display code, and packed 10 columns per central memory word. A central memory word whose lowest byte is zero marks the end of a coded card (it is a coded record) and the full length of the card is not stored if it has trailing blanks. This produces a compact form if coded cards are transferred to another medium.

4.5 DISK FILES

Storage for a disk file is reserved by the monitor in half-tracks as needed. A particular disk file is stored on a number of half-tracks within the same disk cabinet. A half-track consists of either the even or the odd numbered sectors of a track. With this format, data can be efficiently streamed between disk and central memory without the need for large buffers in peripheral memory.

Each sector of a file contains up to 64 central memory words of data, plus two control bytes. The first control byte identifies the location of the next file sector; it contains a sector number if the file is continued on the same half-track, a logical half-track number if the file is continued on a different track, or zero if there is no further information in the file.

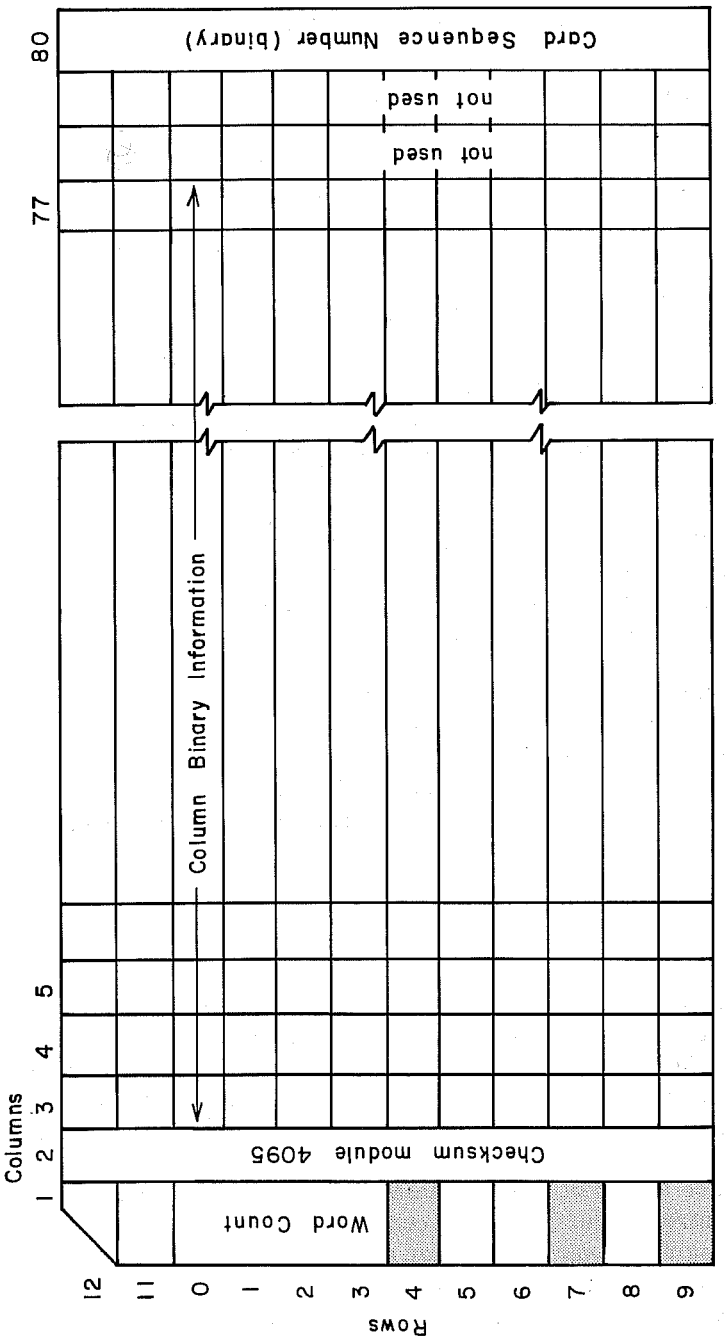


Figure 8. Standard Binary Card Format

A logical half-track number is specified as follows:

(1 bit)	Always set (indicates byte is a half-track number, and not a sector number)
(7 bits)	Physical track number
(1 bit)	Track of odd or even sectors
(3 bits)	Head group number

The second control byte specifies the number of central memory words of information in a sector. End-of-logical record is indicated if this number is less than 64. Both control bytes are zero for end-of-file.

Each disk file must start at the first sector of a logical half-track. When a half-track is full, continuation is to the first sector of another half-track. The forward linkage is constructed when writing and followed when reading. A sector with two zero control bytes is always generated after a write operation. The status word of a disk file records the current position. End-of-file can be generated explicitly by an operation code, or implicitly after a write operation.

A binary record on disk storage is the same as a logical record. A logical record may contain coded records, the end of each indicated by a central memory word whose lowest byte is zero. In this sense, a logical record of coded records is equivalent on disk storage to a binary record, and its use depends upon context.

4.6 BINARY AND CODED MODES

The code for an operation on a named file specifies whether it is to be performed in binary or coded mode. For some devices, the stored form of a file is not dependent upon the mode. When writing to disk, for example, a number of central memory words is transferred from a buffer, and there is no physical difference between a binary write and a coded write, as the distinction lies in the contents. This does not destroy equivalence between devices within their limitations, for instance, a printer always assumes coded information. For a program copying coded information from disk to 1/2" tape, the mode in reading from the disk to a central memory buffer is ignored; but the PP writing on tape will accept coded records (ending in a central memory word whose lowest byte is zero) from the buffer, and write each as a separate physical record on tape in BCD mode. A central program may extract a coded record from a named file with a coded input, and a search for a blank byte, since the PP software will have taken care of any special hardware action if the device was mode dependent.

4.7 MAGNETIC TAPE FILES

The stored form of a logical record is independent of the mode when written on 1" tape. A logical record is divided into 1000_8 word (central) physical blocks; a shorter block marks end-of-logical-record. If the length of a logical record is not a multiple of 1000_8 words the data is exactly contained in the physical blocks; if the length is a multiple of 1000_8 words; an additional shorter block of four 12-bit bytes is necessary.

A logical record can have zero length (corresponding to successive 7, 8, 9 cards).

Any coded records within a logical record end in a word whose lowest byte is zero. Thus, coded information sent to 1" tape is blocked, in display code.

Exactly the same information is written on 1/2" as on 1" tape in binary mode. A coded write to 1/2" tape causes BCD records of 120 characters to be written. Each coded record is translated from display code to BCD (IBM external code), padded with spaces to 120 characters, and written on tape with even parity. Upon input from 1/2" tape in coded mode, characters are translated from BCD to display code, trailing spaces are discarded, and the record is stored internally in the normal form.

End-of-file for 1/2" and 1" tapes is written as the usual file mark.

In reading 1" or binary 1/2" tape, blocks of less than 4 bytes are ignored as noise. Blocks of less than 6 bytes (12 characters) are ignored when reading 1/2" tape in coded (even parity) mode.

Disk storage cannot be substituted for tape if a program writes more than one end-of-file or writes information beyond end-of-file. Also there is no equivalent to end-of-logical record on 1/2" tape in coded mode.

PROGRAMMING LANGUAGES 5

5.1 ASSEMBLY LANGUAGES

The ASCENT assembler will produce Chippewa binary decks from ASCENT or ASPER coding. The language specifications are as described in the ASCENT and ASPER Reference Manuals with the extensions and restrictions explained in this section.

When ASCENT and ASPER coding is used:

FORTRAN statements may not be mixed with ASCENT coding.

Common and external symbols are not available.

System macros are not provided.

The ASCENT assembler can produce and modify COSY decks. COSY is a compressed symbolic deck containing all information, including comments, from the source deck; but it is from 1/10 to 1/5 the size of the source deck, depending on the number of comments. Two consecutive field separators terminate the variable field. The variable field may contain the arithmetic operators for multiplication and division. Programmer macros may be used in both ASCENT and ASPER. Macros may call macros.

A variable field definition (VFD) is available in ASCENT.

5.1.1 ASSEMBLY SYSTEM CALLS

Control card format for calling ASCENT assembly system to produce Chippewa binary decks:

ASCENT (L,X, PA, PC, PB, COSY)

L	If nonzero, a listing is written on the output file.
X	Inoperative; if nonzero, will cause a load-and-go file to be written.
PA	If nonzero, a Chippewa binary deck is written on P80C (Punch 80 columns).
PC	If nonzero, a COSY deck is written on P80C.

PB — If nonzero, a relocatable binary deck is written on P80C.

COSY — File from which COSY input may be read; this may be INPUT or COSY.

The nominal case is ASCENT (L, 0, 0, 0, 0, INPUT)

5.1.2 DECK STRUCTURE

The following are examples of deck structure for calling the assembler:

List Only

```
ASCENT (L)
```

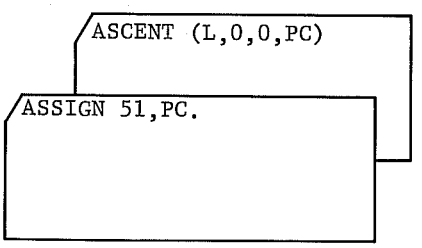
List and Punch a Chippewa Binary Deck

```
ASCENT (L,0,PA)
      or
ASCENT (L,,PA)
ASSIGN CP,P80C.
```

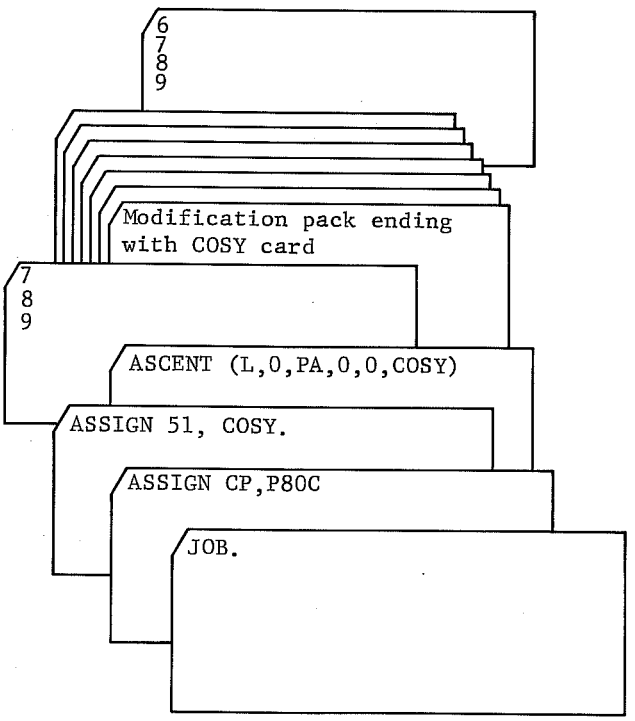
List and Punch a COSY Deck

```
ASCENT (L,0,0,PC)
      or
ASCENT (L,,,PC)
ASSIGN CP,P80C.
```

List and Write a COSY Deck on Tape 51



Insert Modifications Into a COSY Deck on Tape 51, List, and Punch a Chippewa Deck



5.1.3 ASCENT PSEUDO-OPERATIONS

	Name	Parameters	Description
	ASCENT		Sets assembly mode to ASCENT
	SUBRT	C	Causes relocation bits in Chippewa binary deck
	LIST	V1	Suppress listing if V1 $\neq$ 0
	SPACE	V1	Space V1 lines
	EJECT		Eject the page
	MACRO	Name, V1, V2...	Beginning of macro definition
	ENDM		End of macro definition
	IFF	V1, V2, V3	If V1 = 0, assemble next card if V2 $\neq$ V3
			If V1 $\neq$ 0, assemble next card if V2 = V3
	IFZ	V1, V2	Assemble succeeding V2 cards if V1 = 0
	IFN	V1, V2	Assemble succeeding V2 cards if V1 $\neq$ 0
	REPLACE	V1, V2	Replace cards from COSY V1 to V2. If V2 is omitted, replace card V1
	DELETE	V1, V2	Delete cards from COSY V1 to V2. If V2 is omitted, delete card V1
	INSERT	V1	Insert following source cards after COSY V1
	COSY		End of modifications and start of COSY deck
LOC†	EQU	V1	Assign V1 value to LOC
LOC	CON	V1, V2...	Assign values V1, V2, ... to LOC, LOC+1,...
LOC	BSS	V1	Assign a block of V1 core cells to LOC
LOC	BSSZ	V1	Assign a block of V1 core cells to LOC
LOC	BCD	nnCOMMENT	Assemble nn succeeding characters in BCD (nn must be 2 decimal digits)
LOC	BCD	*COMMENT*	Assemble characters within * in BCD
LOC	DPC	nnCOMMENT	Assemble nn succeeding characters in DPC (nn must be 2 decimal digits)
LOC	DPC	*COMMENT*	Assemble characters within * in DPC

†LOC indicates a symbolic identifier is permitted in the location field; where none is shown, this field is ignored by the assembler.

	Name	Parameters	Description
LOC	VFD		VFD card generates a 60 bit word. Field specifications are: Dnn/V1 generate nn bits of display code (nn must be a multiple of 6, the first char must be alphabetic) Nnn/V1 generate nn bits as an integer Ann/V1 generate nn bits as an address (if V1 is relocatable, nn must = 18 and the 18-bit byte must be positioned in bits 0 -17, 15 - 32, or 30 - 47). The sum of all nn's must be ≤ 60. If less than 60, the result will be left justified with 0 fill Examples: VFD D30/INPUT, N12/0, A18/NAME
	END		Indicates last card of assembly

5.1.4 ASPER PSEUDO-OPERATIONS

Name	Parameters	Description
ASPER		Sets assembly mode to ASPER.
ORG	V1	Sets location counter to value of V1
ORGR	V1	Sets location counter to value of V1

All other pseudo-ops are the same as for ASCENT except:

SUBRT	has no meaning
VFD	is not allowed

5.1.5 ERROR FLAGS

The left margin of the listing may have error flags as follows:

O	op-code error
U	undefined symbol in variable field
D	doubly defined symbol in variable or location field
V	VFD error
R	range error for ASPER jump instructions
F	error on CON, BCD, or DPC field

5.1.6 FORCING COMMANDS

The instruction following a RJ, JP or PS will be forced upper; and a plus in the location field will cause the instruction to be forced upper.

A minus in the location field will cause the instruction to be assembled in the next available portion of the 60 bit word regardless of the preceding instructions.

5.1.7 SCANNING RULES

A card may contain information from column 1 through column 72. Either a C in column 1 or a period in column 2 will indicate a remarks card. A period in or after column 11 sets off the remainder of the card as a remark; however, on a CON pseudo-op card or in a literal, a period is a decimal point.

The label field may start anywhere from column 2 through 5. The label ends in column 9. Column 10 is ignored.

The Op code field may start on or after column 11; it is terminated by a field separator which may be a blank, comma, or equal sign.

The variable field may begin after any number of field separators following the op code; the scan will attempt to locate the beginning of the variable field up to column 72. Field separators after the beginning of the variable fields delineate operands. Two consecutive field separators will terminate the scan and remarks may follow without any preceding period. The operand expressions in the variable field are evaluated in a left to right scan.

B26*5+A21 yields (B26 * 5) + A21

A21+B26*5 yields (A21 + B26) * 5

Parentheses are limited to literals and complex constants.

On a CON card the operand entries may be separated by blanks or commas, but two consecutive commas do not define a zero word.

CON 8.263E+5,26,27bA53*25bbCONSTANTS FOR J5

defines 4 words, a floating point number and 3 integers.

The numbers generated by literals are listed in the order of occurrence at the end of the program.

A dollar sign does not define the beginning of the op code field of another instruction.

5.1.8 PROGRAMMER-DEFINED MACROS

The programmer may define macros within an ASCENT or ASPER subprogram with a MACRO pseudo instruction in the following form:

Location	Opcode	Address
blank	MACRO	symbol, list

MACRO	pseudo op code
symbol	macro name
list	sequence of symbols and/or registers, separated by commas, which define the formal parameters of the macro

Macros are called by writing the name of the macro in the opcode field; and in the address field, the quantities to be substituted for the dummy parameters in the definition.

The following rules apply to ASCENT and ASPER use of macros:

1. The definition of a macro must precede the first executable instructions of the subprogram in which it is used.

2. Programmer-defined macros are local to the routine in which the definition appears.

3. A maximum of 100 macros is allowed per subprogram.

4. Macros may be nested to any depth; they may be used in the definition of other macros, provided they are themselves defined prior to use. Recursive definition (a macro used in its own definition) is not allowed.

5. Macro names may be any arrangement of letters and numbers which starts with a letter and contains no more than 8 characters.

6. The macro name must not be identical to a machine mnemonic code, a pseudo code, a system macro code, or any other programmer-defined macro in the same routine.

7. A maximum of 16 parameters are allowed in a macro parameter list.

8. The order and count must be the same for formal and actual parameters.

9. In ASCENT subprograms register names and operands in the formal parameter list may be changed by an actual parameter. For example, a parameter Bk may be changed to Ak or to an operand.

10. A zero actual parameter will cause insertion of a zero in the generated instruction if the formal parameter is in the address field, or a blank if the formal parameter is in the location field.

11. A symbol in the location field of a macro call will be assigned to the first word of the macro, and will override any symbol placed there as a parameter.

12. ENDM pseudo-op must be the last instruction in the macro definition.

ASCENT Example:

Location	Opcode	Address
	MACRO	ABC, D, B2, A2, BN, RESULT, X
D	SA1	B2
	SA2	BN+X
	FX6	X1*X2
	SA6	RESULT
	ENDM	
	MACRO	DEF, X4, AM, F, H, Z, L
	SAM	OP
K	ABC	E, B3, A3, Z, F, G
H	FX7	X6/X4
L	SA7	G
	ENDM	

Using the definitions above, a macro call of

	DEF	X5, A5, LOC1, LOC2, U*V+Q-10, 0

would generate the following set of instructions:

	SA5	OP
K	SA1	B3
	SA3	U*V+Q-12B+G
	FX6	X1*X2
	SA6	LOC1
LOC2	FX7	X6/X5
	SA7	G

ASPER Example:

Location	Opcode	Address
	MACRO	XYZ, OP, A, B, C
	LDM	A, B
	OP	C
	STM	A, B
	ENDM	

Using the definition above, a macro call of

LOC	XYZ	SBD, D1, D2, D3

would generate the following set of instructions:

LOC	LDM	D1, D2
	SBD	D3
	STM	D1, D2

5.1.9 RELOCATION RULES FOR SUBROUTINES

A symbol is any arrangement of letters and numbers which starts with a letter and contains up to 8 characters. A symbol is relocatable if it occurs in the label field of an instruction or pseudo-operation that defines a core location. A symbol is non-relocatable if it occurred on the label field of an EQU card whose operand is an integer. Operands that consist of expressions of non-relocatable symbols will not be relocated. Operands consisting of expressions that are mixtures of relocatable and non-relocatable symbols will not be relocated if a relocatable symbol is involved in a multiply (*) or divide (/) operation; or if an expression consists of the sum or difference of two or more relocatable symbols.

In the Chippewa system, the relocation bits are 16 and 17 of a 30-bit instruction. Bit 16 signifies common; bit 17 an ordinary symbol. Both 16 and 17 on or off signify a constant. Certain valid instructions although handled properly by the assembler may be loaded improperly.

Example:

SX7 220314B.

This instruction has bit 16 set and will be relocated in common.

The first two words of a subroutine define its name, length, and relocatable length for the loader. The required words may be generated by VFD pseudo-ops of the form:

```
VFD D24/NAME,N18/0,A18/END
VFD A18/RELOC,A18/END,N24/PARAMS
```

The symbol END should point to the last core location used by a routine. The symbol RELOC should point to the first constant used by a routine. Neither END nor RELOC may be non-relocatable; they should be defined in terms of relocatable symbols that have been multiplied by the integer, 1. A subroutine should therefore have the following format:

```
          ASCENT
PARAMS    EQU     N       N = number of parameters in the call
          SUBRT
          VFD     D24/NAME,N18/O,A18/END
          VFD     A18/RELOC,A18/END,N24/PARAMS
          BSS     PARAMS + 1
          ---
          (Subroutine Instructions)
          ---
RELOC     EQU     **1+1   start of constants
          ---
          (Subroutine Constants)
          ---
END       EQU     **1+1   end of constants
          END
```

5.1.10 MISCELLANEOUS DATA

The peripheral routines 2RC and 2PC have been modified for use with ASCENT. 2RC will input the entire 80 columns of a binary card if column 1 = 0005. 2PC will punch an 80 column card image if the file name is P80C. The ASCENT assembler requires a field length of 35000_8.

The tables have been defined on EQU cards and their lengths are easily changed by re-assembly. ASCENT table lengths are as follows:

Symbol	Length	Definition
AS25	1000	Maximum number of symbols
AS32	200	Maximum INSERT, DELTE, REPLACE cards
AS33	500*10	Insert table size
AS78	25	Number of literals
AS90	50	Number of macro names
AS91	400	Macro skeleton table length

5.2 FORTRAN AND ASSEMBLY LANGUAGE COMPILER

The Chippewa Operating System contains a compiler which is capable of processing programs written in the FORTRAN language, and programs written in a subset of the ASCENT Assembly Language. Format and required parameters:

RUN (m, fl, cl, bl, I, O)

m Mode of compilation

- G Compile and execute; no source or memory map listing
- S Compile, no execute, with source and memory map listing
- P Compile, no execute, punch complete program on binary cards, with source and memory map listing
- L Compile, no execute, list source and object code
- C Chain compile (in G-mode)
- B Batch program compile (in S-mode)
- I Compile, no execute, punch, object code and loader information only

fl	Object program field length (octal)
cl	Object program common length (octal)
bl	Object program I/O buffer length (octal)
I	File name for compiler input
O	File name for compiler listable output

If a parameter is entered as zero, or space, or the list of parameters is shorter than six, an assumed parameter is used by the compiler:

m	Assumed to be G
fl	Set to job field length (defined on job card)
cl	Set to length of common in main program
bl	Set to buffer length (installation parameter normally 2000_8)
I	Assumed to be INPUT (standard input file)
O	Assumed to be OUTPUT (standard output file)

If standard values are to be used for all parameters, the program call card may be shortened to:

```
RUN.
```

Compiler output, except in G mode, includes a reproduction of source statements, a map of variables and all error indications detected during compilation. If G mode is selected, all source output is suppressed unless errors are detected; in which case, output is the same as indicated for other modes. If L mode is selected, output will include an octal list of compiled instructions following the printing of each source statement for which object code is produced.

A binary copy of the correct object code compiled is always put on the disk as a binary file; the name of the program becomes the file name. This file may be called and executed repeatedly by name within this job file.

5.2.1 FORTRAN

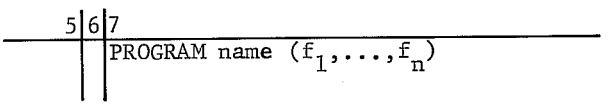
A full description of the FORTRAN language statements, card format, restrictions and error diagnostics is available in the Chippewa Operating System FORTRAN manual.

The compiler is capable of compiling programs written in FORTRAN IV, FORTRAN II and FORTRAN 63 specifications. FORTRAN IV is the implied compilation mode.

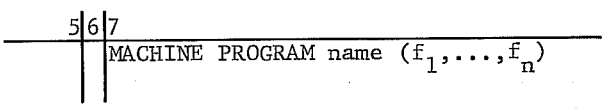
Header Cards

The first card of each program to be compiled and executed by the RUN compiler must be of the following format:

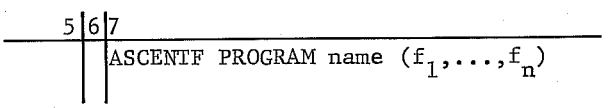
FORTRAN program statements



Central Chippewa assembly statements

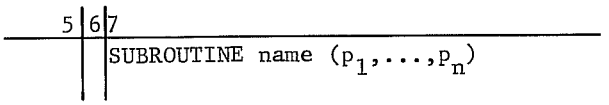


ASCENT assembly statements



f_i names of all input and output files required by the program and its subroutines.

Each subroutine written in the FORTRAN language must begin with the statement:



p_i parameters passed by the calling routine to the subroutine.

Each function written in the FORTRAN language to be compiled by RUN must begin with a statement equivalent to the subroutine statement except that the word SUBROUTINE is replaced by "type FUNCTION". Type which is optional, may be one of the following:

INTEGER

REAL

DOUBLE PRECISION or DOUBLE

COMPLEX

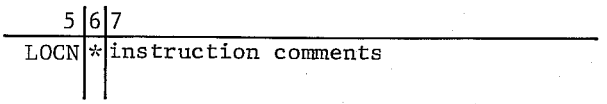
LOGICAL

5.3 MACHINE LANGUAGE

The compiler is capable of processing programs and subroutines written in the Chippewa assembly language. Such programs may be intermixed with regular FORTRAN programs and subroutines. A subprogram written in this language may be compiled separately by using a RUN program call card.

5.3.1 CARD FORMAT

The format for the assembly language statements is as follows:



LOCN	Location tag of 2 to 6 alphanumeric characters in columns 1-6. No embedded blanks. An asterisk in column 1 indicates a remark card.
*	Continuation mark, must be in column 6. Up to 19 continuation cards allowed for each statement; cannot have * in column 6 in the first card of a continuation statement.
instruction	May begin in column 7 or later and terminate before column 73 unless continued on the next card. No embedded blanks allowed.
comments	Comments may follow the instruction if separated by one blank.

5.3.2 CONSTANTS

Constants in standard FORTRAN notation may be specified in the assembly language. (Chippewa FORTRAN manual).

Constants may begin in column 7 or after and terminate before column 73 unless continued on the next card. Embedded blanks are not allowed. Octal constants have a B character suffix. A constant may have a symbol associated with it in the location field.

5.3.3 INSTRUCTION FORMATS

In the assembly language, operational registers are designated by single-character names as follows:

S = X0	0 = B0	A = A0
T = X1	I = B1	B = A1
U = X2	J = B2	C = A2
V = X3	K = B3	D = A3
W = X4	L = B4	E = A4
X = X5	M = B5	F = A5
Y = X6	N = B6	G = A6
Z = X7	O = B7	H = A7

The letter R specifies a return jump, and the letter P specifies all other jumps.

Let S represent any of the letters S-Z, I represent any of the letters I-O or the digit 0, and A represent any of the letters A-H. Let Q represent a positive integer less than 2^{16} or an alphanumeric tag of two-six characters. Then the forms of assembly-language instructions, grouped according to functional units required for execution, are as follows:

Symbolic Form	Machine Form	Example
0	00xxx	0
R=Q	01xxK	R=TAG
P=Q+I	02ixK	P=TAG+N
P=Q, S=0	030jK	P=TAG, T=0
P=Q, S/0	031jK	P=TAG, U/0
P=Q, S)0	032jK	P=TAG, V)0
P=Q, S(0	033jK	P=TAG, W(0
P=Q, S. I	034jK	P=TAG, X. I
P=Q, S. O	035jK	P=TAG, Y. O
P=Q, S. D	036jK	P=TAG, Z. D
P=Q, S. N	037jK	P=TAG, S. N
P=Q, I=I	04ijK	P=TAG, J=K

Symbolic Form	Machine Form	Example
P=Q, I/I	05ijK	P=TAG, L/M
P=Q, I)I	06ijK	P=TAG, N)O
P=Q, I(I	07ijK	P=TAG, I(0
S=S	10ijx	Y=V
S. L=S*S	11ijk	T. L=W*X
S. L=S+S	12ijk	V. L=Y+Z
S. L=S-S	13ijk	V. L=U-V
S=-S	14ijk	Z=-W
S. C=S*S	15ijk	W. C=T*U
S. C=S+S	15ijk	X. C=V+W
S. C=S-S	17ijk	Y. C=S-Z
S=S(Q)	20ijk	T=T(24)
S=S(-Q)	21ijk	U=U(-10)
S=S(I)	22ijk	V=X(N)
S=S(-I)	23ijk	W=W(-M)
S, I=S-	24ijk	X, J=W-
S, I=S+	25ijk	Y, K=Z+
S, I=S.	26ijk	Z, 0=U.
S=I, S.	27ijk	T=K, V.
S=*Q	43ijk	T=*12
S. N=S+S	30ijk	T. N=U+V
S. N=S-S	31ijk	U. N=X-Z
S. D=S+S	32ijk	V. D=V+V
S. D=S-S	33ijk	W. D=S-U
S. R=S+S	34ijk	X. R=Y+Z
S. R=S-S	35ijk	Y. R=Z-T
S. I=S+S	36ijk	T. I=U+X
S. I=S-S	37ijk	Z. I=Y-Z
S. N=S*S	40ijk	S. N=X*X
S. R=S*S	41ijk	Y. R=Y*Y
S. D=S*S	42ijk	T. D=T*U
S. N=S/S	44ijk	Y. N=T/X
S. R=S/S	45ijk	Z. R=X/W
$	46xxx	$
S=*S	47ixk	T=*W
S=(A+Q)	50ijK	T=(C+TAG)
S=(I+Q)	51ijK	U=(J+100)
S=(S+Q)	52ijK	Z=(T+30B)
S=(S+I)	53ijk	T=(T+J)
S=(A+I)	54ijk	U=(B+K)

Symbolic Form	Machine Form	Example
S=(A-I)	55ijk	V=(C-N)
S=(I+I)	56ijk	W=(M+N)
S=(I-I)	57ijk	X=(L-K)
I=A+Q	60ijK	J=H+TAG
I=I+Q	61ijK	K=L+10
I=S+Q	62ijK	L=L+55B
I=S+I	63ijk	M=T+J
I=A+I	64ijk	N=G+K
I=A-I	65ijk	O=A-L
I=I+I	66ijk	I=I+J
I=I-I	67ijk	J=K-M
S=A+Q	70ijK	T=G=TAG
S=I+Q	71ijK	U=K+5
S=S+Q	72ijK	V=L+15B
S=S+I	73ijk	W=X+J
S=A+I	74ijk	X=A+K
S=A-I	75ijk	Y=X-L
S=I+I	76ijk	Z=M+I
S=I-I	77ijk	S=N-O

Notes:

1. The arithmetic mode indicators L, C, N, D, R, and I may immediately follow a result register name; the period in these cases is optional.

```
TL=X*Y
UC=V+V
VN=T/W
WD=X+Y
XR=S-T
```

2. In the instructions 02 and 50-77 either term may be dropped, in which case a 0 designation is assembled.

```
P=K
P=TAG
T=(J)
J=15B
U=-K
```

3. In the instructions 50-54, 60-64, and 70-74 the terms may be interchanged unless Q is a constant.

```
T=(TAG+L)
M=K+B
U=L+X
```

4. In the instructions 50-52, 60-62, and 70-72 the plus sign may be replaced by a minus sign if Q is a constant.

```
X=(I-30)
J=K-55B
Y=-1'
```

5. In the instructions 51, 61, and 71 the right member may be an indicated sum or difference of a tag and a constant, in which case the constant must follow the tag.

```
W=(TAG-35)
K=TAG+1
U=TAG+100B
```

6. In the instruction 51 the parenthesized quantity may be a constant, represented in conventional FORTRAN form, only if the result register is to receive the machine version of that constant; in this case the address of the converted number is assembled into the instruction.

```
T=(-1.5E-6)
U=(47550516045547B)
```

7. If Q is to correspond to an octal integer, the digits in the number must be trailed by a B.

8. Alternate forms for certain instructions are:

S=S.S	11ijk
S=S$S	12ijk
S=-S.S	15ijk
S=-S$S	16ijk
S=S+S	36ijk
S=S-S	37ijk
S=S*S	40ijk
S=S/S	44ijk
A=A+Q	50ijk
A=S+Q	52ijk
A=S+I	53ijk
A=A+I	54ijk
A=A-I	55ijk
A=I+I	56ijk
A=I-I	57ijk

9. A plus sign in a location field will force the corresponding instruction to the high order positions of a new word.

10. The instruction 00 is assembled as a full zero word.

5.3.4 PSEUDO-OPERATIONS AND VARIATIONS

The following FORTRAN statements are permissible in a Chippewa language assembly:

COMMON

EQUIVALENCE

DIMENSION

EXTERNAL

DATA

These statements, written in the standard FORTRAN notation must appear after the PROGRAM or SUBROUTINE statement and before the assembly language cards. The variables in these statements may be referenced in subsequent instructions.

Six types of declaration statements are allowed. These provide constants with a symbol which may be referenced within the executable coding.

$CON(c_1=v_1, c_2=v_2, \ldots c_n=v_n)$

The constants on the right of the equal sign are assembled into a cell assigned to the symbol appearing on the left of the equal sign.

Example: CON(C1=25, C2=777B, C3=-6.54E-2)

$HOL(h_1=g_1, h_2=g_2, \ldots h_n=g_n)$

The ten-character groups, including spaces are converted to display code and assigned to a location identified by the symbols appearing to the left of the equal sign.

Example: HOL(H1=ABCDEFGHIJ, H2=1234567890)

$ABS(a_1=v_1, a_2=v_2, \ldots a_n=v_n)$

The unsigned values are assembled into the address field of the instructions containing the associated symbols.

Example: ABS(JJ=100, KK=100B, LL=7777B)

RES(r_1=w_1, r_2=w_2, ... r_n=w_n)

Local blocks are reserved for the number of words specified; the symbols is assigned to the first word at the block.

Example: RES(K1=10, K2=100B, K3=1000)

COM(c_1=v_1, c_2=v_2, ... c_n=v_n)

Blank common blocks is reserved for the number of words specified, the symbol is assigned to the first word of the block.

Example: COM(B1=1, B2=300, B3=205B)

SUB(s_1=r_1, s_2=r_2, ... s_3=r_3)

The subroutines named on the right of the equals operator are assembled into memory and tagged with the symbols appearing on the left.

Examples: SUB(SI=SIN, LG=LOG, OUT=OUTPTC)

The card format for declaration statements is similar to assembly language card format; they must appear in columns 7-72, and no embedded blanks are permitted; continuation cards and comments are handled in the same way.

The following alternate form of the above declaration statements is accepted. If the opening parenthesis is not used, any separation such as comma, slash, etc., may be used if the closing parenthesis is also dropped.

```
5 6 7
    CON v=c, v=c,...,c=c remark
```

5.3.5 PROGRAM ORGANIZATION

The card decks to be assembled must be in the following order:

1. Program or subroutine card
2. FORTRAN statements, if any
3. Declaration statements, if any
4. Instruction cards
5. Constant cards
6. END card

The instruction portion of the deck must begin with three zero lines of coding plus one zero line for each file name: $(f_1, \ldots, f_n)$ of the program card or each parameter $(p_1, \ldots, p_n)$ of the subroutine card:

	0	control word 1
	0	control word 2
A_1	0	parameter 1
	⋮	
A_n	0	implement 2
	0	exit/entry line

The first two zero lines correspond to the central information furnished by the compiler and the last zero line is the exit/entry line for a subroutine; unused by a main program. The first executable instruction follows this exit/entry line.

The constant portion of the deck must be separated from the instruction portion by a card with two periods in columns 7 and 8. Constants may appear in the instruction portion of the deck provided they are positive and less than 2^{54}.

END is a FORTRAN card and must have 7, 8, 9 punches in column 1.

A card with a period in column 1 may appear anywhere in the deck; it causes a page eject of the printed listing.

A card with an asterisk in column 1 may appear anywhere in the deck; it is treated as a remark card.

5.4 FORTRAN COMPILER/ ASCENT SUBSET

The FORTRAN compiler, RUN, is capable of processing programs or subroutines written in a subset of ASCENT assembly language. Such routines may be intermixed with regular FORTRAN programs or subroutines. A routine written in this language may be compiled independently by the RUN program call card described in Section 5.2.

5.4.1 HEADER CARD

Any routine written in this language to be processed by RUN must begin with a card of the following type:

Main Program

	7
	ASCENTF PROGRAM name (f_1,...,f_n)

Subroutine

	7
	ASCENTF SUBROUTINE name (p_1,...,p_n)

f_i are the file names of all I/O files used in this program and its subroutines.

p_i are the parameter names in their conventional FORTRAN usage.

5.4.2 CARD FORMAT

The card formats are as described in ASCENT programming manuals with the following restrictions:

An address field may contain an indicated sum of a symbol and constant, but not a sum or difference of two symbols.

Location symbols may start in column 1, but may not extend beyond column 6.

Instructions may start anywhere beyond column 6 but no card may contain more than one instruction.

A PS instruction causes assembly of a full zero word.

Double precision and complex literal constants are not accepted.

A minus sign is not allowed in a location field.

An asterisk is not allowed in an address field.

The instruction portion of the deck may contain BSS, BSSZ and EQU cards. The address field of such cards may contain only a single constant.

5.4.3 CONSTANTS

Constants may appear in the standard FORTRAN notation with the restrictions as designated in Section 5.3.2.

5.4.4 PSEUDO-OPERATIONS AND VARIATIONS

The allowable pseudo-operations and added statements and instructions are as described in Section 5.3.4.

5.4.5 PROGRAM ORGANIZATION

Each ASCENTF subprogram must be organized in the following manner:

1. Program and subroutine statement
2. FORTRAN statements, if any
3. ASCENT instructions
4. Constant cards
5. END card

The following definitions and restrictions must be adhered to in this organization.

The instruction portion of the deck must begin with three zero lines of coding plus one zero line for each file name (fi) or parameter p_i) of the header statement.

The constant portion of the deck may contain BSS, BSSZ, EQU, DPC, BCD and CON cards. The address field of these cards may contain only a single constant or 10-character display code string of the form *ABC... .J*. The DPC and BCD strings are converted to display code. BSS and BSSZ cards produce zeroed regions.

The constant portion of the deck must be separated from the instruction portion by a card with two periods punched in columns 7 and 8.

END is a FORTRAN card, with END punched in columns 7, 8, and 9.

All formats which have been described in Section 5.3.2, except the instruction formats, are acceptable to the compiler under ASCENTF.

5.4.6 COMPATIBILITY

Except for the initial card, double-period card, and separating instructions from constants, upward compatibility from the Chippewa system to SIPROS may be achieved by starting location symbols of no more than five characters in column 2 and starting instructions and pseudo-operations in column 11.

5.5 PERIPHERAL ASSEMBLY LANGUAGE

This assembler (PAS) is an alternate program which may be used instead of ASPER. It converts the peripheral processor symbolic language into PP absolute code. The assembled program is available as a binary card deck and a line printer listing.

The PP assembler may be called by the name PAS. on a program call card.

The assembler produces a side-by-side listing and binary cards. There is no provision for leaving a running version of the assembled program in a PP. Binary card punching is suppressed if an error is detected during the assembly process. The binary cards are punched in the Chippewa Laboratory standard binary card format:

Columns 1 and 2 contain a binary card identifier (7, 9 punches), the number of central memory words (60 bits) on the card (15_{10} maximum) and a card checksum. Columns 3-77 contain program information; column 80 is a card sequence number in binary.

An alternate form of punched cards may be obtained with a parameter on the PP program call card (transmitted via lower 18 bits of the input register).

```
3
PAS,1000.
```

This control card produces a full 80-column binary card with no checksum or other identification; however, these binary decks are not in a usable format for the Chippewa Operating system.

5.5.1 CARD FORMAT

The card format for this assembler is fixed field, except for the remarks and the DIS pseudo-operation.

Standard Format

Columns

1-4	Location field; maximum of four alphanumeric characters
5	Blank
6-8	Operation field; three-character mnemonic
9	Blank
10-13	Address field; maximum of four alphanumeric characters. There are no provisions for adding or subtracting constants.
14-19	Blank
20-80	Comments

One-word Instruction Format

with mnemonic operation

1 4	5	6 8	9	10 13	14 19	20 ← → 80
LOCN	b	OPN	b	ADDR	b ← → b	comment

with octal code

1 4	5	6 8	9	10 13	14 19	20 ← → 80
LOCN	b	bbb	b	NADR	b ← → b	comment

LOCN	Location tag	alphanumeric (optional)
OPN	Operation	mnemonic
ADDR	Address	octal constant (one or two octal digits)
		alphanumeric (four characters maximum)
NADR	Four octal digits denoting OPN and ADDR in absolute code	
b	Blank	

Two-word Instruction Format

with mnemonic operation

1 4	5	6 8	9	10 13	14 19	20 ← → 80
LOCN	b	OPN	b	ADDR	b ← → b	comment
LOCN	b	bbb	b	ADDR NADR	b ← → b	comment

with octal code

1 4	5	6 8	9	10 13	14 19	20 ← → 80
LOCN	b	bbb	b	NADR	b ← → b	comment
LOCN	b	bbb	b	ADDR NADR	b ← → b	comment

NADR Four octal digits denoting OPN and ADDR in absolute code; or in the second word, a four octal digit ADDR.

Exceptions:

1. The pseudo-operation REM which must begin in column 6 may use all of the area from columns 10-80 for comment.

2. A card with an asterisk in column 1 is treated as a remark and is free field after column 1.

3. The pseudo-operation DIS in columns 6-8 converts all characters from column 10-80 to standard packed display code.

5.5.2 CONSTANTS

Numeric constants used in a program must appear in columns 10-13 and are considered to be octal. All constants are right justified by the assembler. They have a range of 0000 to 7777 in a two-address instruction or a range of 00 to 77 in a one-address instruction. Negative constants are defined as the values from 40 to 77 and 4000 to 7777 in one-and two-address instructions. All others are positive octal constants. Each word of coding requires a separate card. If a constant contains a minus sign, it must appear in column 10.

5.5.3 PSEUDO-OPERATIONS

IDT — Identification of program; must be the first card.
Format: blank IDT name

ORG — Origin; beginning location, nnnn, of assembled program. Will be set to 1000_8 if no ORG specified.
Format: blank ORG nnnn

SBL — Set beginning location, nnnn, of binary punch output.
Format: blank SBL nnnn

BLR — Block reservation; reserve nnnn words beginning at location.
Format: locn BLR nnnn

REM — Remarks card. Asterisk in column 1 is equivalent.
Format: blank REM comment or * comment

EQU — Equates the location tag to a numeric constant, nnnn, or a predefined location to ltag, appearing in the address field.
Format: locn EQU ltag or locn EQU nnnn

DIS — Display code; converts the characters starting at column 10 to display code and packs them two to a word starting at this location. The word following packed display code is all zeros.
Format: locn DIS message

END — Defines the last card of the deck to be assembled.
Format: blank END

Table 3. PAS OPERATION CODES

Octal Opcode	Mnemonic	Address	Comments
00	PSN		Pass
01	LJM	md	Long jump to m + (d)
02	RJM	md	Return jump to m + (d)
03	UJN	d	Unconditional jump d
04	ZJN	d	Zero jump d
05	NJN	d	Nonzero jump d
06	PJN	d	Plus jump d
07	MJN	d	Minus jump d
10	SHN	d	Shift d
11	LMN	d	Logical difference d
12	LPN	d	Logical product d
13	SCN	d	Selective clear d
14	LDN	d	Load d
15	LCN	d	Load complement d
16	ADN	d	Add d
17	SBN	d	Subtract d
20	LDC	dm	Load dm
21	ADC	dm	Add dm
22	LPC	dm	Logical product dm
23	LMC	dm	Logical difference dm
24	PSN		Pass
25	PSN		Pass
26	EXN		Exchange jump
27	RPN		Read program address
30	LDD	d	Load (d)
31	ADD	d	Add (d)
32	SBD	d	Subtract (d)
33	LMD	d	Logical difference (d)
34	STD	d	Store (d)
35	RAD	d	Replace add (d)
36	AOD	d	Replace add one (d)
37	SOD	d	Replace subtract one (d)

Octal Opcode	Mnemonic	Address	Comments
40	LDI	d	Load ((d))
41	ADI	d	Add ((d))
42	SBI	d	Subtract ((d))
43	LMI	d	Logical difference ((d))
44	STI	d	Store ((d))
45	RAI	d	Replace add ((d))
46	AOI	d	Replace add one ((d))
47	SOI	d	Replace subtract one ((d))
50	LDM	md	Load (m + (d))
51	ADM	md	Add (m + (d))
52	SBM	md	Subtract (m + (d))
53	LMM	md	Logical difference (m + (d))
54	STM	md	Store (m + (d))
55	RAM	md	Replace add (m + (d))
56	AOM	md	Replace add one (m + (d))
57	SOM	md	Replace subtract one (m + (d))
60	CRD	d	Central read from (A) to d
61	CRM	md	Central read (d) words from (A) to m
62	CWD	d	Central write to (A) from d
63	CWM	md	Central write (d) words to (A) from m
64	AJM	md	Jump to m if channel d active
65	IJM	md	Jump to m if channel d inactive
66	FJM	md	Jump to m if channel d full
67	EJM	md	Jump to m if channel d empty
70	IAN	d	Input to A from channel d
71	IAM	md	Input (A) words to m from channel d
72	OAN	d	Output from A on channel d
73	OAM	md	Output (A) words from m on channel d
74	ACN	d	Activate channel d
75	DCN	d	Disconnect channel d
76	FAN	d	Function (A) on channel d
77	FNC	md	Function m on channel d

Notation	Interpretation
d	Implies d itself
(d)	The contents of d
((d))	The contents of the location specified by d
m	Implies m itself used as an address
m + (d)	The contents of d are added to m to form an operand (jump address)
(m + (d))	The contents of d are added to m to form the address of the operand
dm	An 18-bit quantity with d as the upper 6 bits and m as the lower 12 bits

5.5.4 ERROR FLAGS

Errors detected by the assembler are shown as a one-character error tag in the left margin of the assembled listing.

O Illegal operation code

U Undefined symbol in the address field

M Multiply defined location symbol

R Range error. The d portion is greater than 77_8; or on a jump instruction (03-07), the address symbol is more than 37_8 forward or backward.

5 Too many characters in operation code

K Constant error. (12-bit constants only)

D Direct address error (greater than 77_8)

A Absolute address error (greater than 77_8)

Q Improper use of EQU pseudo-op code (part of variable field not properly defined)

N Incorrect IDP pseudo-op code (first character is not alphabetic)

V Improper use of ORG pseudo-op code; a variable error where an ORG card attempts to set the value of the location backwards. The ORG card is ignored.

PAS Control Card Options

Punch even if errors are detected

Do not print a listing

CONTROL CARDS AND JOB PROCESSING 6

6.1 CONTROL CARDS

A job consists of one or more central programs which are executed with data files. Control cards are the first logical record; they identify the programs and their data files, and sequence program executions. The control cards specify how the job is to be processed; the operations performed upon the other records of a job file depend upon the control cards.

Each job must begin with a job card and end with a file separator card. All control cards must appear between the job card and the first record separator. The end of the control cards is signified by a 7,8,9 punch (end-of-record) card, or a 6,7,8,9 (end-of-file) card if the job consists of control cards only. No special multipunches are used for control cards and the information on a control card can start in column 1 (with no imbedded blanks). The card is free field thereafter. Cards terminate with a period or, if a parenthesized list appears, a closing parenthesis. The order of cards is described in section 7.

Except for program call cards and job cards, the control card formats are unique to the system and must not be used for the names of any programs.

6.1.1 JOB CARD

The first control card of a job must indicate the job name, priority, central processor time limit, and central memory requirement. Fields are separated by commas and the last field is terminated by a period.

```
name,priority,time limit,field length.
```

name — Alphanumeric job name; must begin with a letter and may be 1-7 characters.

If only a job name is specified, priority 1, time limit 1 minute, field length 40000_8 is assumed.

priority — 1 through 17 (octal).

The highest priority on disk is the next to be brought to a free control point for processing.

- The central processor is always given the highest priority control point that can use it.
- Completed job output files are printed in order of priority.

If a job which is completely compute-bound has the highest priority (17_8), and another job which does a great deal of I/O processing issues a recall instruction and is waiting, the operator may enter a greater priority to the I/O job (which will still allow the compute-bound program to retain the associated control point).

time limit — Total time limit for the job in seconds (central processing time); a maximum of 5 octal digits. The octal value in hundreds is approximately the time in minutes. The time limit is actually rounded up to a multiple of 10_8 by the system.

Time and space limits must suffice for the whole job, including all compilation and execution.

field length — Total field length of the job, in octal, maximum of 6 octal digits. The length cannot exceed 360,000. The field length (storage requirement) is rounded up to a multiple of 100_8 by the system.

If the RUN compiler is used and a listing requested, it prints out the amount of storage that was not needed, both for itself and the compiled program, so that future attempts may request a lesser amount. A common trial storage request for compilation is 100000_8 (32K). The standard amount, 40000_8, is sometimes sufficient for compilation, and usually adequate for utility jobs such as file copying.

Example:

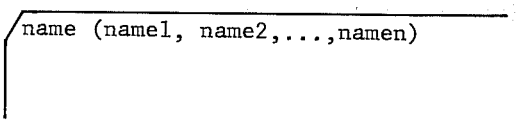
```
JOB765,3,600,40000.
```

Statements on a job card may use different separators, for example:

```
JOB765(2,350,100000)
```

Spaces may also be introduced, but are best omitted.

6.1.2 PROGRAM CALL CARD



name — Name of program being called, either a system program or user program.

name i — Names of all files referenced in this job, or parameters to the program. If a program being called has no parameters the line must terminate with a period. A closing parenthesis terminate a line with parameters.

Examples:

```
RUN(G,300000,100000,3000)
COPYCR(TAPE2, TAPE3,1000)
LINNEY.
```

6.2 EQUIPMENT ASSIGNMENT

Any file not specifically assigned on control cards is assigned by the system to storage on disk unit zero. A job need not request card reader and printer for normal input/output since its cards are already stored in the job input file on disk, and output for a printer is sent to the job output file on disk. Input and Output files are normally stored on disk zero.

The control cards of a job are processed in order, so any equipment assignment must be made before the corresponding file is referenced.

6.2.1 ASSIGN u, f

This control card assigns any available peripheral unit of type u to a file named f. The type u may be any of the equipment listed below or it may be an equipment number, in which case, operator action is not required.

DA	disk cabinet, channel 0	CR	card reader
DB	disk cabinet, channel 1	LP	line printer
DC	disk cabinet, channel 2	MT	607 magnetic tape (1/2")
DS	display console	WT	626 magnetic tape (1")
CP	card punch		

This must be the first appearance of the name f in the job file. The file name f is alphanumeric, begins with a letter, and is a maximum of seven characters long. Multiple file names are not allowed.

Examples:

ASSIGN50, TAPE6.	Assign equipment number 50 (channel 5, unit 0, 1/2" tape) to TAPE6. The job will be held up if this tape is presently assigned to another job.
ASSIGN CP, PUNCH.	Assign a card punch to the file named PUNCH. When the job writes to the file PUNCH, data will be punched on cards.
ASSIGN MT, TAPE2.	Operator to assign a 1/2" tape to file TAPE2.
ASSIGN WT, TAPE3.	Operator to assign a 1" tape to file TAPE3.

For MT and WT a message for the operator is displayed under the number of the job's control point:

WAITING FOR MT (or WT)

To assign tape 61 to control point 6, the operator would key 6.ASSIGN61.

ASSIGN01, TAPE9.	Use disk unit 1 to store file TAPE9.

For TAPE, the ASSIGN statement is intended for scratch tapes only; the operator may assign any free tape of the specified type.

A user supplied tape should be assigned with a REQUEST statement.

6.2.2 REQUEST f.

This control card requests the operator at the system display console to assign to this job the peripheral equipment specified by f. This must be the first appearance of the name f. The job waits for operator action before proceeding.

Example:

REQUEST TAPE 4.

Operator to assign an equipment for file TAPE 4.

In this case, the message REQUEST TAPE4. is displayed under the number of the job's control point, and the operator can key the number of the equipment on which the user's tape is mounted. For control point 4: 4.ASSIGN71.

The equipment number is related to an actual equipment through the equipment status table (Section 4.1.1).

The usual convention for the Chippewa System is to set unit numbers for 1/2" tapes to the lower numbers of a channel. The system may be loaded at dead start time from channel 5, unit 0, but this unit can be subsequently used by jobs.

6.3 COMMON FILES

Common files are files that are not discarded upon job completion. Normally, input files used by a running job (type local) are dropped; disk space is freed or equipment released. Output files are printed and discarded after printing. A job may declare a file to be common so as to make it available to other jobs. However, a job to which a common file is attached, can change it to local if discarding is desired.

6.3.1 COMMON f.

This control card has two effects:

1. If the file name, f, has common status in the FNT/FST and is not being used by another job, it is assigned to this job until dropped. If the file is being used by another job or does not have common status, this job must wait until the file is available.

2. If the file name, f, already appears as a local file name for the job, the file will be assigned common status in the FNT/FST and is available to any succeeding job after it is dropped by this job.

A file generated by a job may not be declared in a COMMON card until the job has been completed.

Example: COMMON BFILE.

6.3.2
RELEASE f.

With this control card, the common file named f currently assigned to this job will be dropped from common status and assigned local status in the FNT/FST.

Example:

RELEASE BFILE.

The common file, named BFILE, attached to this job is changed to type local so that it will be dropped at the end of the job.

6.4 SWITCH, MODE, EXIT

SWITCH n.

This control card sets pseudo sense switches for reference by a subsequent FORTRAN program; n = 1-6. The settings are preserved at the control point and copied to RA for the use by the central program. Switches may be changed by a console command.

Example:

SWITCH 6.

MODE n.

This control card may be used to change the arithmetic exit mode. n is a single octal digit (See Exchange Jump Information, Section 3.3.) The exit mode is set to zero unless otherwise specified.

Example:

MODE 3.

EXIT.

The EXIT card can be used to separate the control cards associated with the normal execution of a job from a group of control cards to be executed in the event of an error exit as listed below:

1 TIME LIMIT.	Job has used all the central processor time it requested.
2 ARITHMETIC ERROR.	Central processor error exit has occurred.
3 PPU ABORT.	PP has discovered an illegal request, e.g., illegal file name or request to write outside job field length.
4 CPU ABORT.	Central program has requested that the job be aborted.
5 PP CALL ERROR	Monitor has discovered an error in the format of a PP call entered in RA+1 by a central program (can occur if a program accidentally writes in RA+1, as can condition 3).
6 OPERATOR DROP.	Operator has requested the job be dropped.
7 DISK TRACE LIMIT.	No more room on a disk unit used by a job.

When one of these conditions occurs, an error flag (numbered as above) is set at the control point. In cases 1, 2, 5, 6, 7, a dayfile message is issued; and in case 3, the fault-finding PP issues a message (BUFFER ARGUMENT ERROR from CIO, or NOT IN PPLIB).

When an error flag is set, a search is made for the next EXIT control card; and if it is not found, the job is terminated. If an EXIT card is found, the error flag is cleared and succeeding control cards are processed. If an EXIT card is met and no error flag is set the job is terminated normally at that point.

Example:

MYJOB,1,400,100000.	Job card
ASSIGN WT, TAPE1.	Request scratch tape
RUN.	Compile and execute
EXIT.	
DMP.	Dump exchange package
DMP,1000.	Dump first 1000_8 words of store
7,8,9	End of control cards
(Program) 7,8,9	
(Data) 6,7,8,9	

The dumps are made only if an error condition occurs.

Record Separator

This card, consisting of a 7,8,9 punch in column 1, separates the different types of records (control cards, source language cards, data cards) within a job.

File Separator

This card, consisting of a 6,7,8,9 punch in column 1, must be the last card of each job deck. No job may use information beyond this card.

6.5 COMPILER AND PROGRAM CALLS

After the job card, any control card other than ASSIGN, REQUEST, COMMON, RELEASE, MODE, EXIT, or SWITCH, is a call for a central or PP program to be executed. A program may be in the library, or stored on a file used by the job. The control cards of a job are processed in order, and a number of programs may be executed in one job. A job is a set of programs using the same data files.

Parameters of a central program call follow on the same control card, for example:

RUN(P)	Compile, punch, and do not execute next record of INPUT file.
COPYBF(TAPE1,DISK2C)	Copy binary file from file TAPE1 to file DISK2C.
RUN.	Compile, execute, and do not list program on next logical record of input file. (Assumed by RUN if no compile mode parameter given.)

When RUN compiles a program, the program is written as a binary record on a disk file with the same name as the program (name taken from the program's first card). The program can thus be executed separately, for example:

RUN(S)	Compile, list, and do not execute program.
PG8C.	Execute program called PG8C.

A program call control card is interpreted as follows:

1. The names of files attached to the job's control point are searched for the program's name. If the program is found, it is read to central memory from the next record of the file.

2. The library of central programs on disk 0 is searched for the named program; if found, the program is read to central storage.

3. The library of peripheral programs on disk 0 is searched for the named program, and the program is assigned to a PP.

The parameters of a central program are entered beginning at RA+2, left adjusted in display code, before execution. Parameters are normally compiled into a program, and overridden only if new parameters are specified by a control card call. A peripheral program may have two numerical parameters of at most 6 octal digits. These are entered in the input register of the PP which executes the program.

The first card of a program compiled by RUN is not a control card. It is part of the program and must include a list of files used by the program, for example:

```
PROGRAM     SAM3     (INPUT, OUTPUT, TAPE1)
```

Such a statement supplies to RUN a list of files used by the program, which RUN enters from RA+2 as parameters in the binary form of the program. If the program is compiled and executed directly, those files will be used; but a separate call for the program can specify other files to be used instead. The binary form of SAM3 in which the parameters have been compiled could be used from the INPUT file:

```
JOB6,,,100000.
REQUEST FRED.
INPUT(,,FRED)
7,8,9
(SAM3 on binary cards)
7,8,9
(Data)
6,7,8,9
```

Here, any reference to TAPE1 in the source code of SAM3 would actually use FRED. As the first two file names were not overridden by the INPUT card, they would be used as in the source code.

The file names INPUT and OUTPUT are reserved for the job deck and output file for printing.

6.6 CENTRAL LIBRARY CALLS

The central library includes programs and subroutines. Subroutines are stored in central memory or on disk according to frequency of use; programs are always stored on disk. Except for the RUN compiler, a number of utility routines, and other frequently used programs may be added to the library.

The central utility routines include COPY, COPYCR, COPYCF, COPYBR, COPYBF, and COPYSBF for copying files or records between files in binary or coded form. BKSP and REWIND position a file, and VERIFY compares two files. Although resident in central library, these programs use little central processor time since they are used only to call for input/output; a job using them should be given high priority.

Example: Two binary files on magnetic tapes are to be copied to disk as a single file called TAPE9 for use by a FORTRAN program which uses the punch.

```
THEJOB,10,1000,200000.          Job card
REQUEST FIRST.                  Get first tape
REWIND(FIRST)
REQUEST SECOND                  Get second tape
REWIND(SECOND)
COPYBF(FIRST, TAPE9)            First tape to disk
REWIND(FIRST)
BKSP(TAPE9)                     Backspace over file mark
COPYBF(SECOND, TAPE9)           Second tape to disk
REWIND(SECOND)
REWIND(TAPE9)
ASSIGN CP, PUNCH.
RUN.
7,8,9
    PROGRAM H3 (INPUT, OUTPUT, PUNCH, TAPE9)
    (Rest of Program)
7,8,9
    (Data)
6,7,8,9
```

Since TAPE9 has not been assigned specifically, it goes to disk.

Example: To obtain a source listing of program SAM and write the binary version to tape for later use:

```
KEEPSAM,6,,100000.
RUN(S)                          List, and write SAM to disk
REQUEST SAMTAPE.
REWIND(SAMTAPE)
COPYBR(SAM, SAMTAPE)            Copy to tape
REWIND(SAMTAPE)
7,8,9
    PROGRAM SAM(INPUT, OUTPUT)
    .....
6,7,8,9
```

To use the binary tape version with data:

```
USESAM,1,1000,100000.
REQUEST SAMTAPE.
REWIND(SAMTAPE)
COPYBR(SAMTAPE, SAM)
REWIND(SAM)
SAM.
7,8,9
(data)
6,7,8,9
```

It is necessary to copy the tape to disk since a program call can be made only to a program stored on disk 0. The copying routines refer to logical records, which may extend over several physical records.

6.7 PERIPHERAL LIBRARY CALLS

Peripheral library programs, stored on disk 0, whose names begin with a letter can be called from control cards. They include:

CLL	Load one or more overlays into specified central area
EXU	Load a called program to replace the calling program
LBC	Load record of binary cards from input to central
LOC	Load octal corrections from input to central
PBC	Punch binary cards from central
DMP	Dump central to output file
DIS	Use a display console for this job
MSG	Dayfile messages
CIO	Processing of I/O requests

These routines are not often called by programmers; they are used mainly for system maintenance, and are more fully described in Section 3.12.

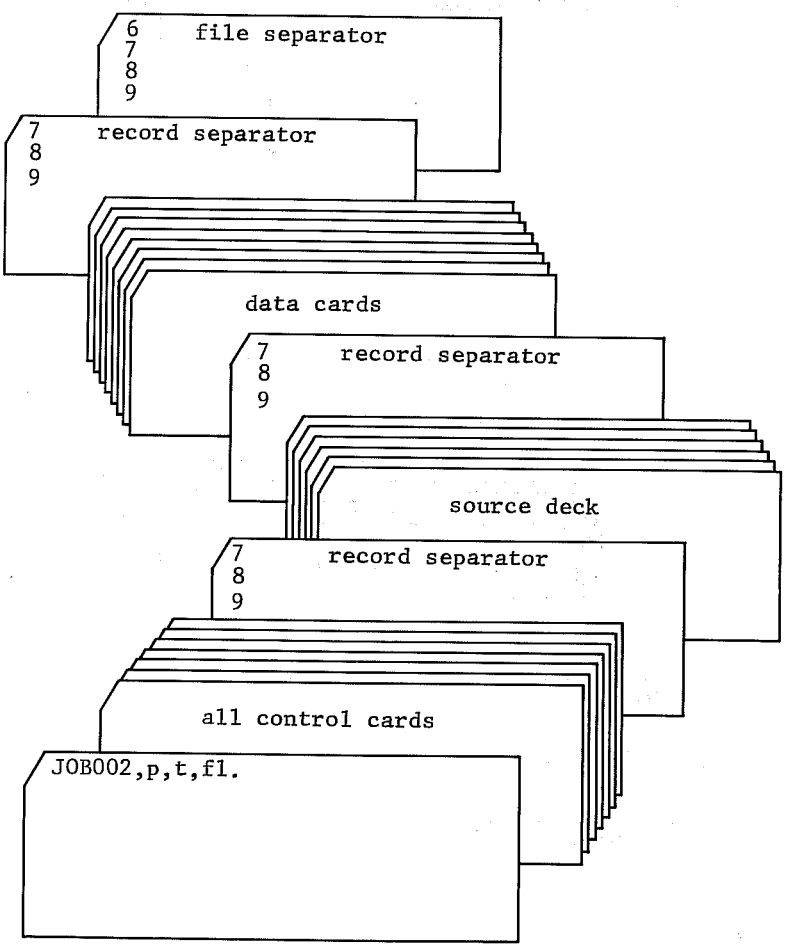
6.8 SYSTEM ACTION ON CONTROL CARDS

When a job is brought to a control point, the first record of the input file is copied to a 96-word buffer in central memory attached to the control point. If the control cards will not fit in this buffer, the message TOO MANY CONTROL CARDS appears in the dayfile and the job is terminated. Since cards are compactly stored 10 columns to a word without trailing blanks, this allows a large number of control cards (about 40) and the error usually arises when programmers have omitted the 7, 8, 9 card following the control cards.

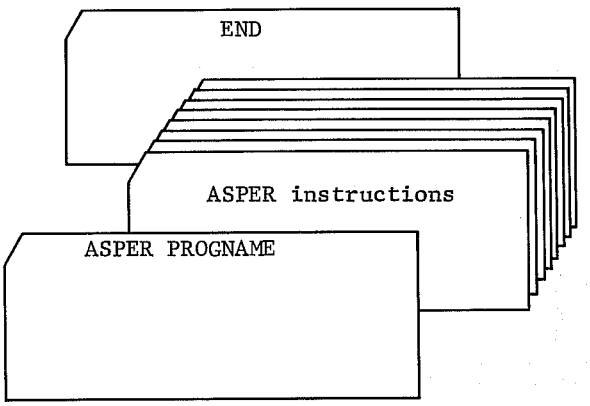
When a job is neither using nor awaiting the central processor or PP's, the monitor calls to a PP the package 1AJ to advance the job.

This condition may occur when an error flag is set at the control point (usually taken care of automatically by PP's), in which case 1AJ calls in an overlay 2EF to process the error flag. Otherwise the overlay 2TS is called to interpret the next control statement and act upon it. If there are no more control statements, the job is terminated.

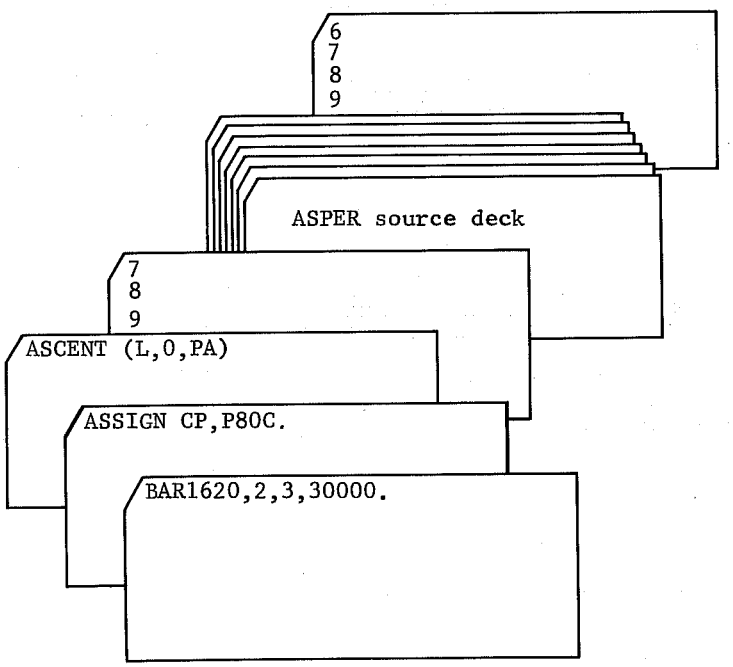
The following card deck indicates the arrangement of control cards to begin a job, separate job records, and terminate a job. The record separator which must be used between types of cards has 7, 8, 9 punches in column one. The file separator which terminates a job has 6, 7, 8, 9 punches in column 1.



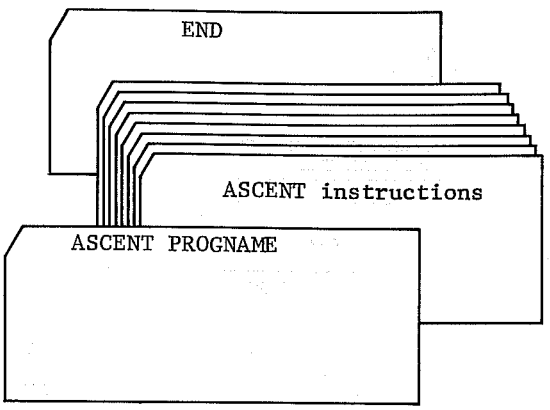
ASPER SOURCE DECK



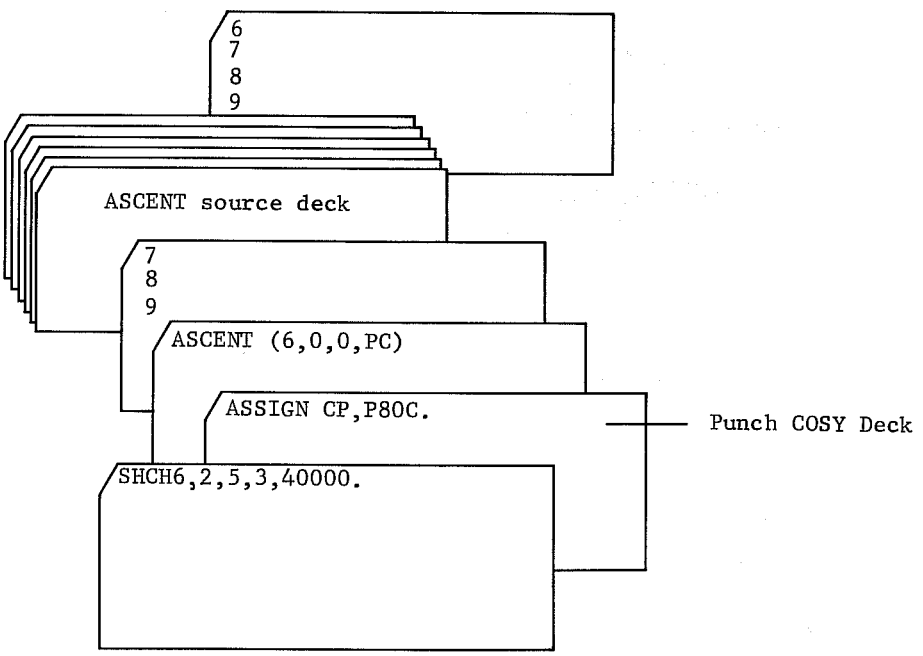
ASPER ASSEMBLY



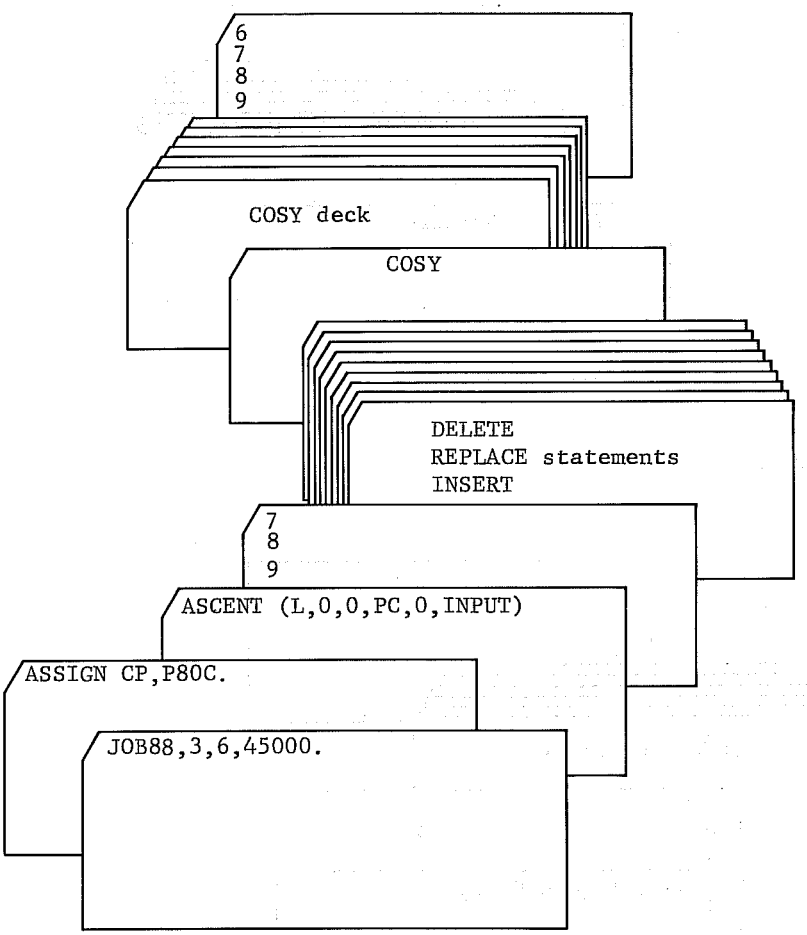
ASCENT SOURCE DECK



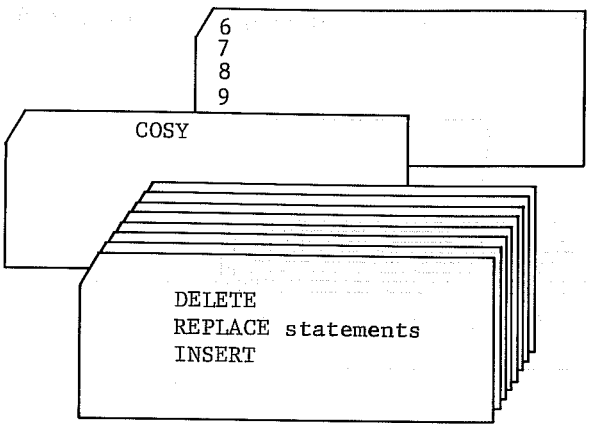
ASCENT ASSEMBLY

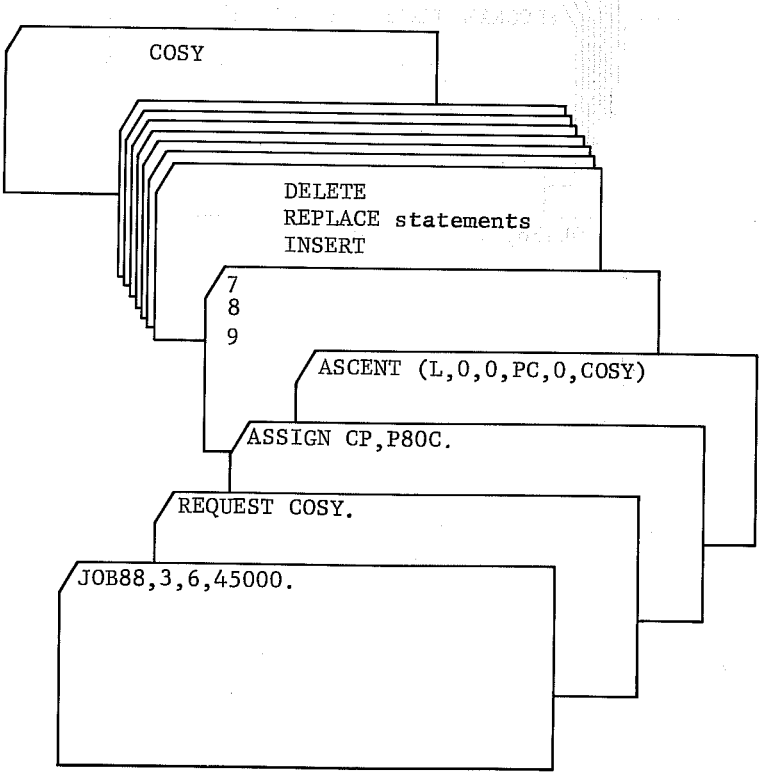


ASSEMBLY WITH COSY



COSY MODIFICATIONS

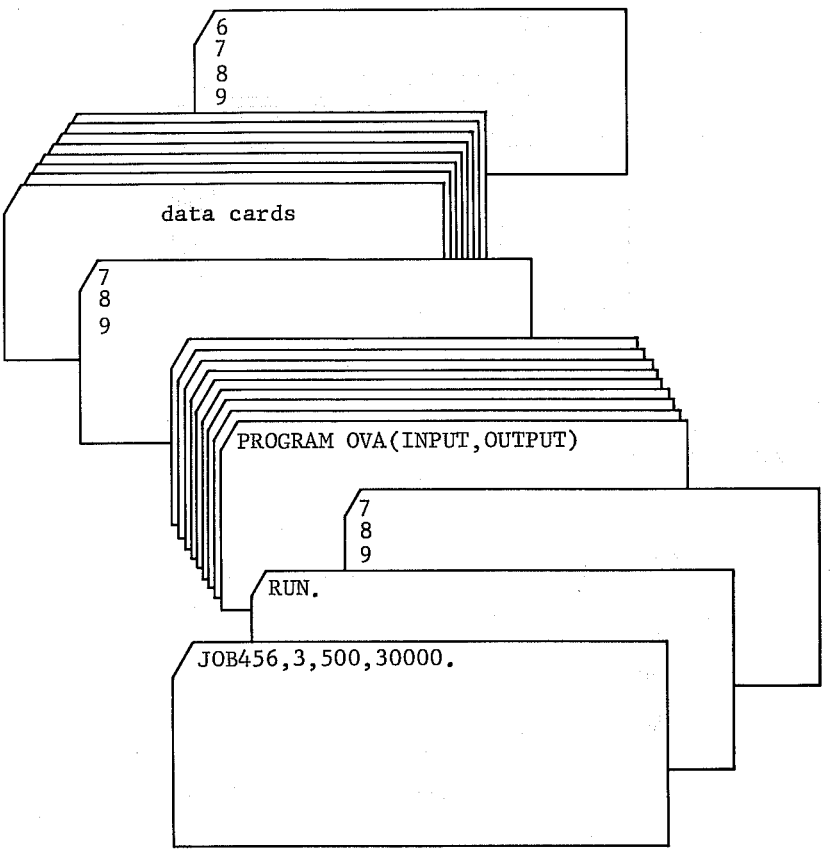




FORTRAN Load and Run

Job 1

INPUT and OUTPUT are the only I/O files used; no special control cards.



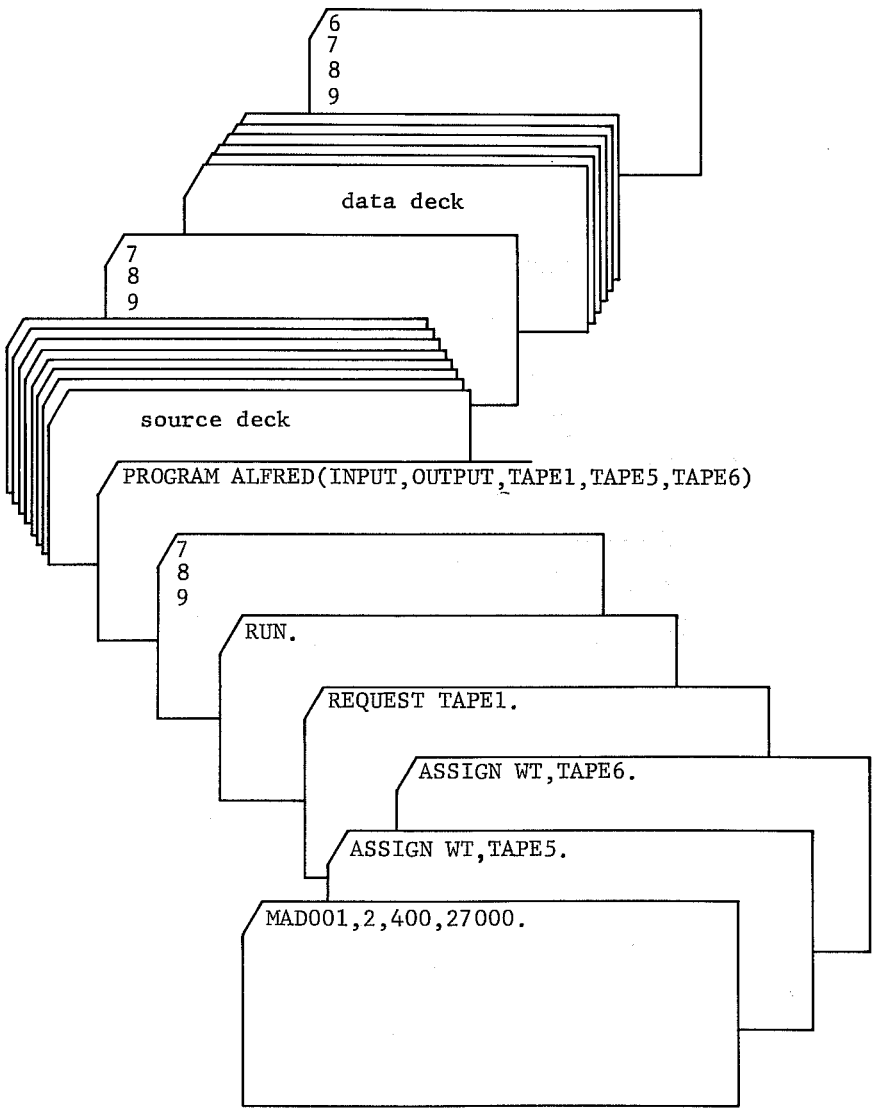
FORTRAN Load and Run

Job 2

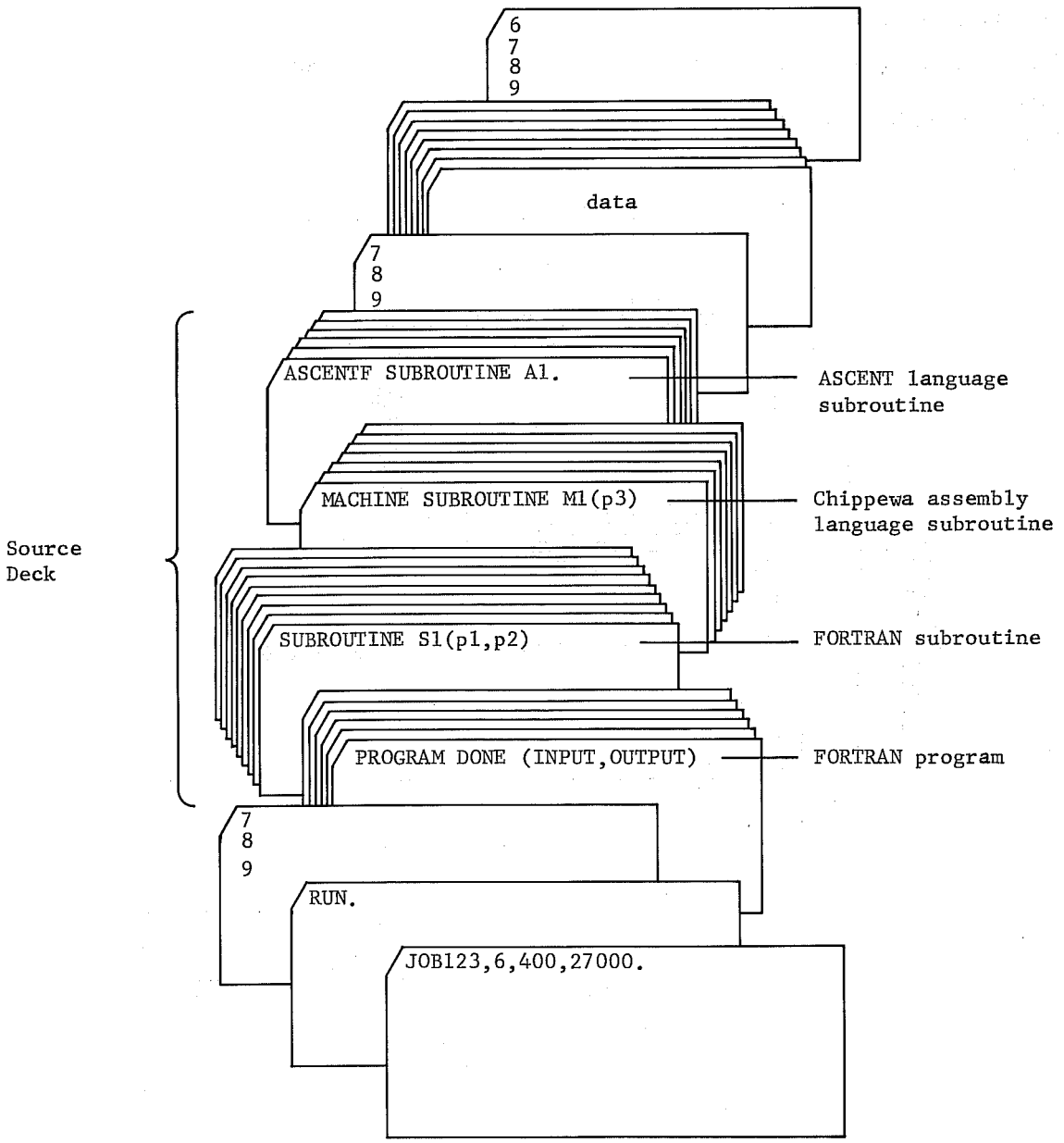
Three tape references:

TAPE1 — assumed input tape which operator loads on a particular unit

TAPE5
TAPE6 output scratch tapes drawn from tape pool

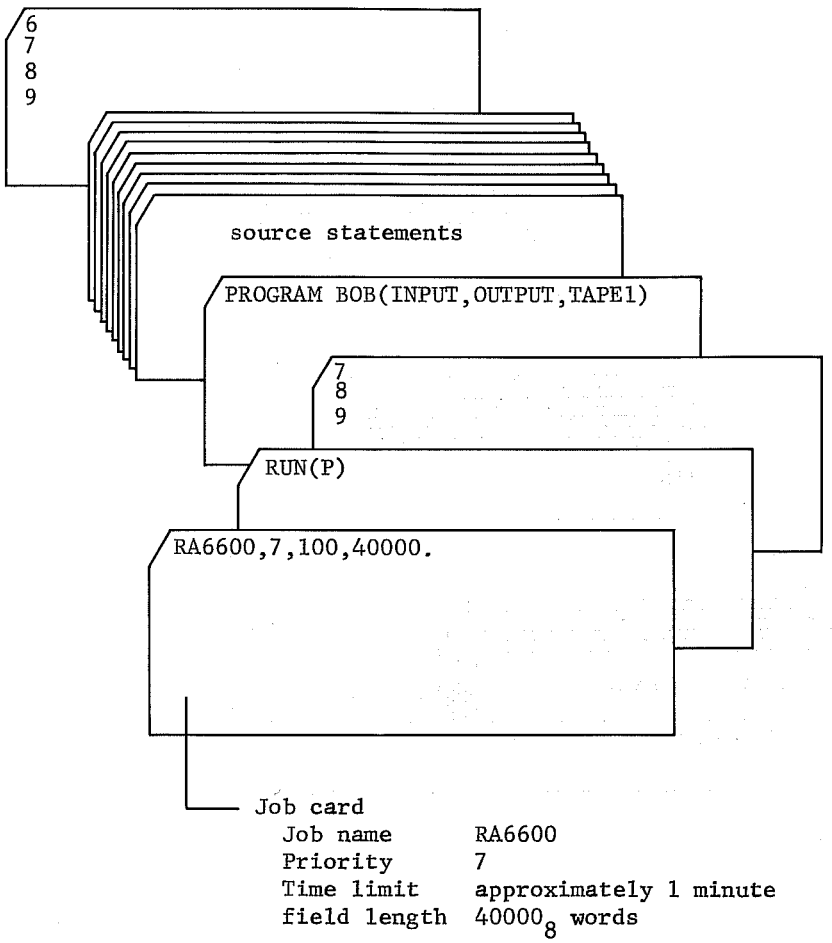


FORTRAN Compile and Execute with Mixed Deck



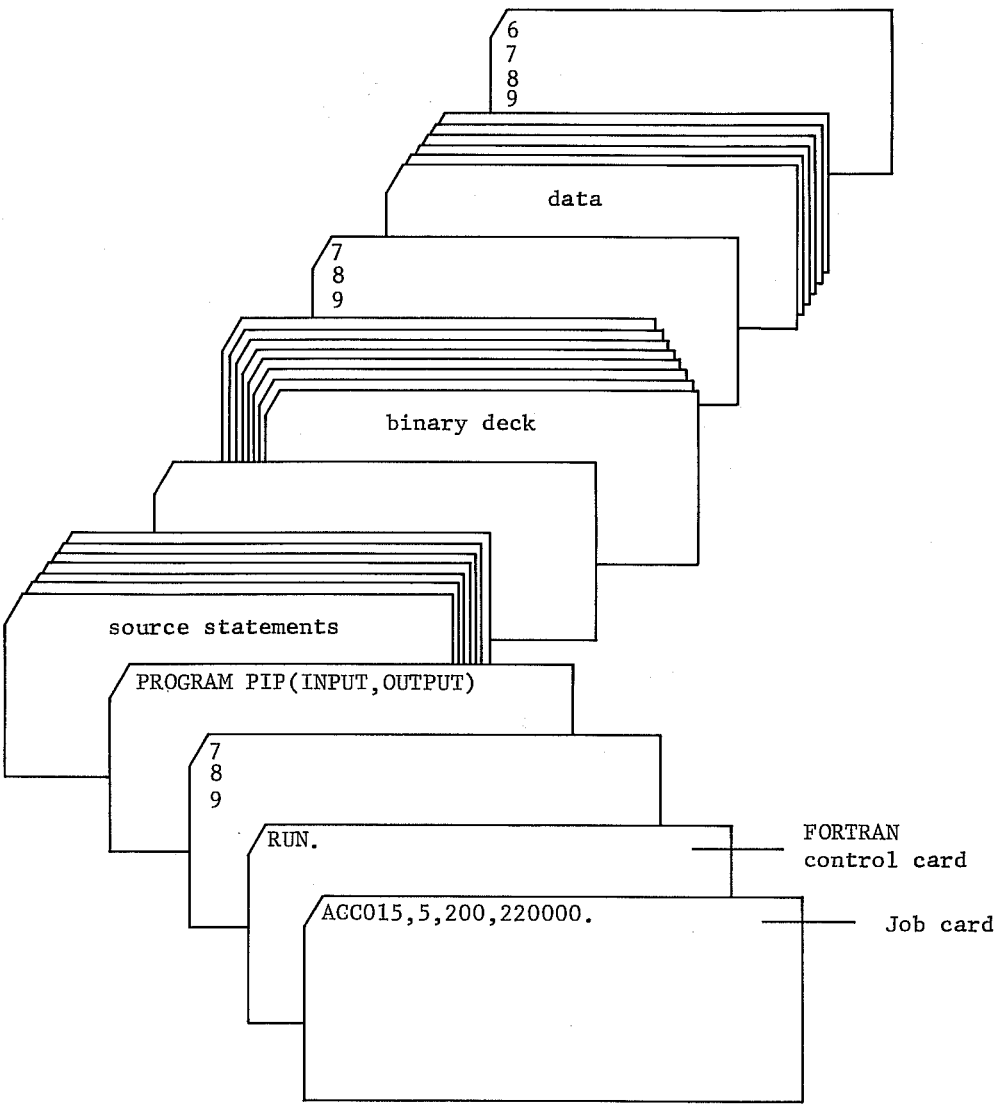
FORTRAN Compile and Produce Binary Cards

Compile program and produce binary cards, do not execute. Three files of I/O - INPUT, OUTPUT and TAPE1



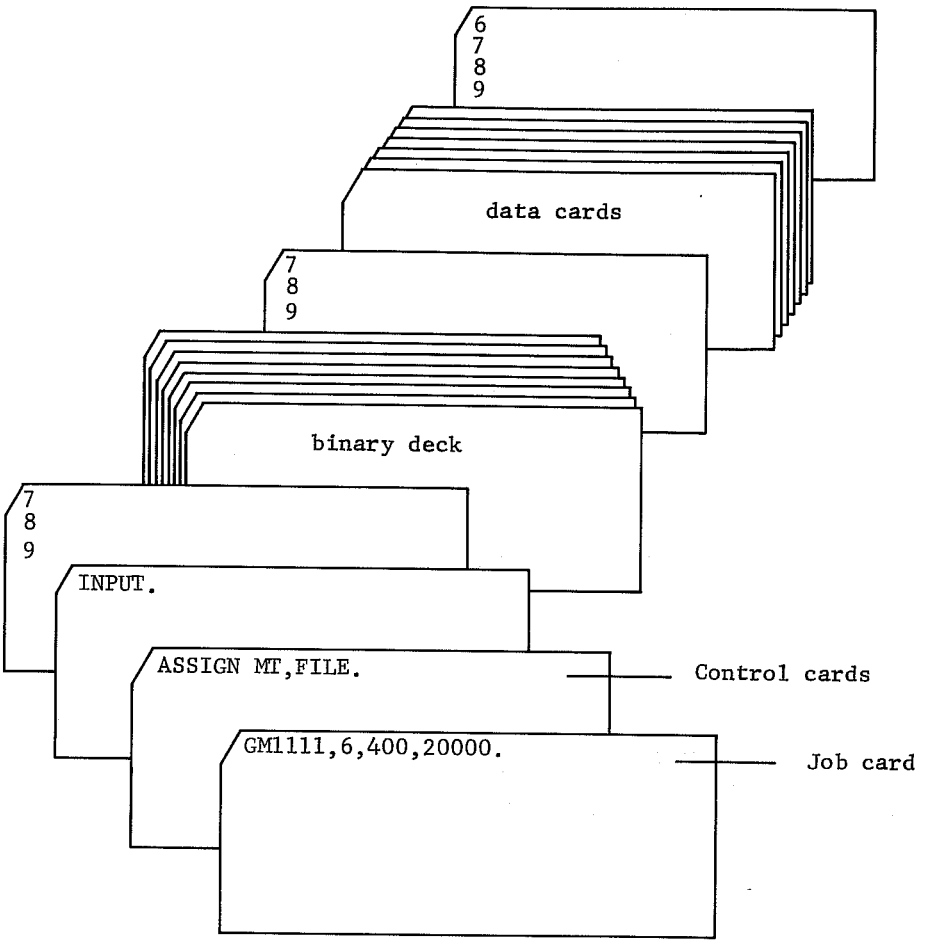
FORTRAN Compile and Execute (Plus a Prepunched Binary Deck of a Subroutine)

A binary deck to be loaded with a compiled routine must be preceded by a card with a plus (+) punch in column 1. This card indicates to the compiler that all succeeding cards to the end of record are binary cards to be loaded with the program just compiled. A minus (-) punch in column 1 may be used when no library subroutines are to be loaded.

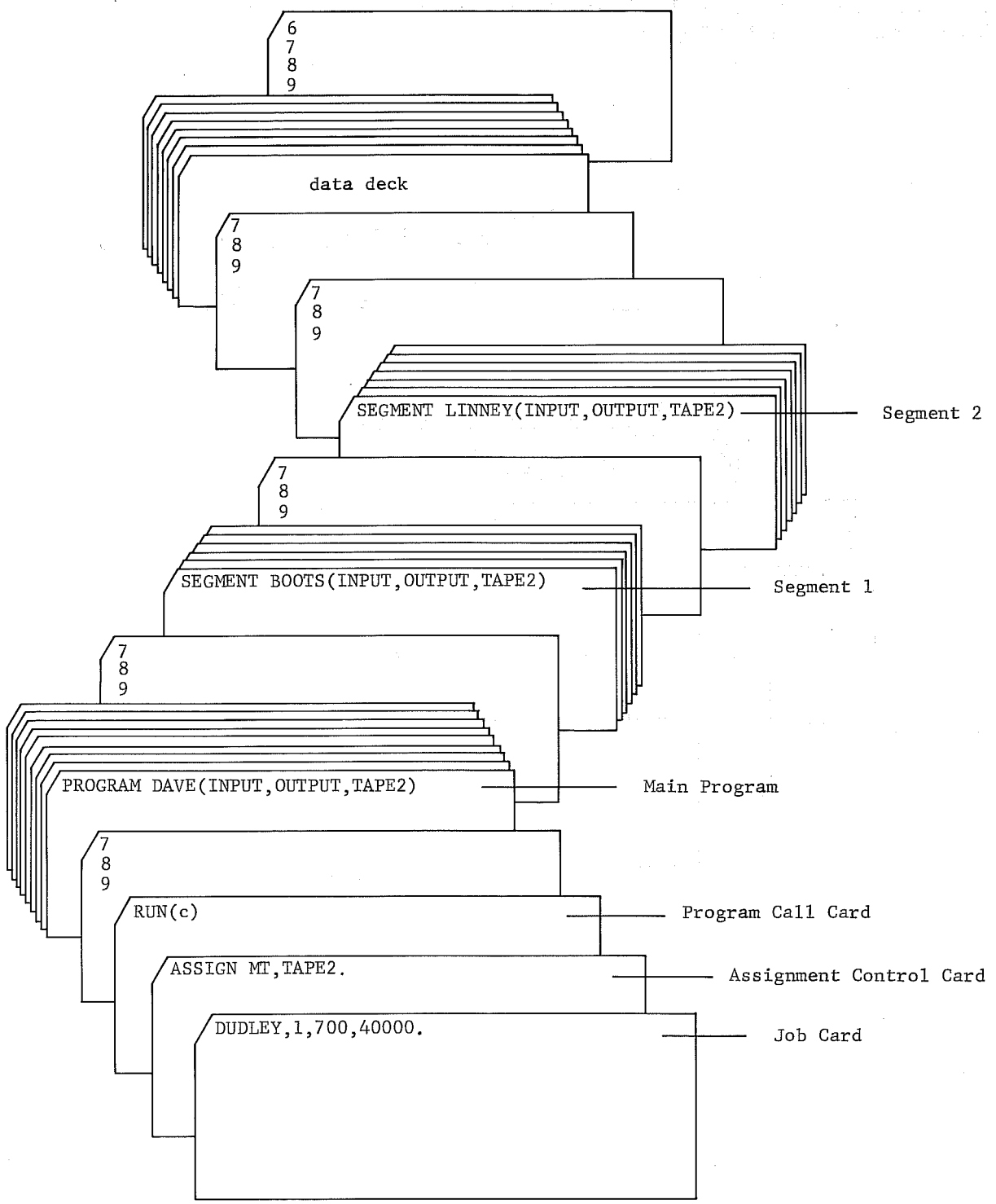


Load and Execute a Prepunched Binary Program

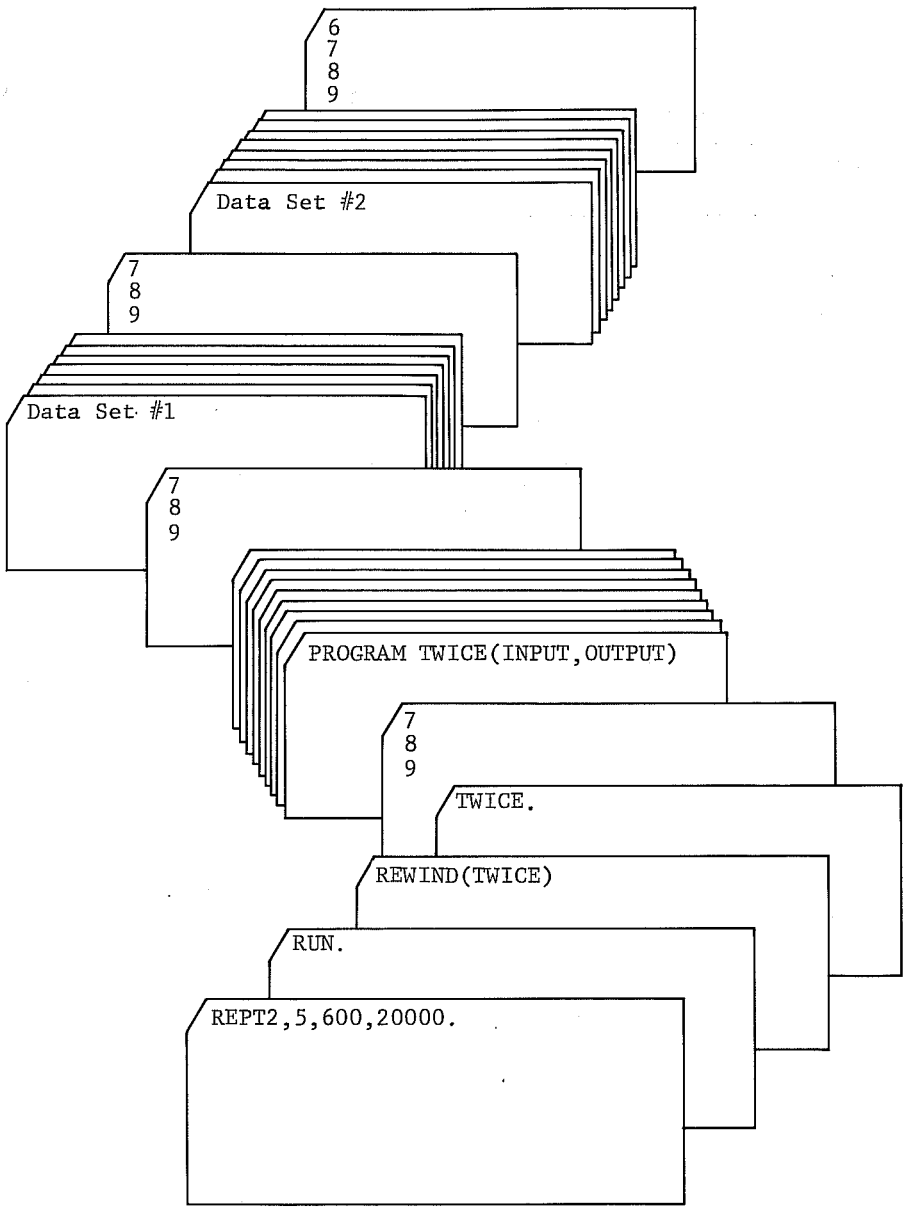
The binary cards in the input file following the record separator are loaded into central memory when the program call card INPUT is encountered.



FORTRAN Compile and Execute with Chaining



Compile Once and Execute Twice with Different Data Decks



DEAD START

USING 6000 PERIPHERALS

P		
0000		
1	1410	Load A with 10₈
2	7301	Output 8/10 Wds on Ch #1
3	0006	FWA = 0006
4	7501	Activate Ch #1
5	7113	Input on Ch #13
6	0000	FWA 0000
7	7713	Function code select
10	2060	TU #0 and Rewind
11	7713	Func. Read Binary
12	2020	
13	7413	Activate Ch #13.
14	7113	Input Ch #13.

USING 3000 PERIPH.

P	T.U	CARD READER	
0000			
1	7512		Disconnect Ch #
2	7712		Func. Ch #12
3	5000	(4000 CARDS)	Conn. Tape Func (405 BCD
4	7712		Func Ch #12
5	0010	(0001 CARD)	Rewind / Negate H→
6	7712		Func Ch #12
7	1400	(1500 CARDS)	Input to EOR / Input to 6660 deac
10	7412		Act. Ch #12
11	2001	(2000 CARDS)	Load A with 10000₈
12	0000	(0120 CARD)	
13	7112		Input Ch #12
14	0000		FWA 0000

When Pressing dead-start, program above will be obeyed. System tape must be on Ch #12.

SYSTEM/OPERATOR COMMUNICATION 8

8.1 DEAD START LOADING

When the dead start switch is toggled, the dead start loading process transmits a short program (up to 12 instructions) to the PP0 memory where it is executed. The dead start program in turn transmits a bootstrap program to another peripheral processor which brings in the system loader from the library tape and transfers control to it.

The system tape is completely loaded at dead start time and all data is transferred either to central memory or to the channel 0 disk file; the tape is not referenced again during operation.

Dead start panel settings are shown in Table 4.

8.2 PROCESSING MODES

After the system tape has been loaded via the dead start operation, no activity will be displayed at control points 1 to 7. After the system tape is rewound, a keyboard message at the console is needed to initiate a mode of job processing.

The Operating System provides two modes of operation - automatic and manual. With the Chippewa System, several programs can be running concurrently. The operator may override the automatic mode and manually modify, delete, or add new programs and change priorities as the need arises.

Table 4. DEAD START PANEL SETTINGS
(Bootstrap Loading of the System Tape
for 6000 Series Tape Units Only)

Memory†	Contents	Action Generated	Toggle Settings
01	1410	Load (A) with 10_8	001 100 001 000
02	730x	Output 10_8 words starting at location 6 on channel x (processor x will store these in its memory beginning at location 0)	111 011 000 xxx
03	0006		000 000 000 110
04	750x	Disconnect channel x (permits execution of program)	111 101 000 xxx
05	7113	Set to input mode (7770 words to location 0000 on channel 13)	111 001 001 011
06	0000		000 000 000 000
07	770x	Select rewind tape on channel x	111 111 000 xxx
10	2060		010 000 110 000
11	770x	Read up to 10,000 words in binary mode on channel x	111 111 000 xxx
12	2020		001 000 010 000
13	740x	Activate channel	111 100 000 xxx
14	710x	Set to input mode (channel x)	111 001 000 xxx
15	0000	Cleared during dead start	

† Locations at peripheral processor 0

Procedures for loading are described below.

1. Check to see that the dead start panel is set for the bootstrap loading of the system tape.

   a. Check that the toggle switches are at their proper settings; 1 bits up and 0 bits down.

   b. Toggle the DEAD START switch up and then down; the system tape is then loaded into the system.

2. Check the master display at the console; the control points should be displayed. If no display is apparent on the scope after the system tape has been loaded, adjust the INTENSITY knob on the panel directly under the display scopes. (Section 8.3 describes the visual displays that appear on the console scopes.)

8.2.1 AUTOMATIC MODE

After dead start loading, the master display routine at PP9 awaits input from the operator. A keyboard entry AUTO selects the mode for automatic processing. Under this mode, the monitor assumes input from punched cards and output to the printer. Monitor assigns the Read package to the first available PP (assigned to control point 1), and then assigns Print, to the next available PP (assigned to control point 2). The Read package reads, converts the information to display code, and forms input files on the disk until cards are depleted.

The monitor assigns the next package to all available control points, other than 7, which interrogates the jobs in the input file; the input files are executed according to the priority designated on the job card. As each job produces output, the records are packed and sent to the disk as printer output files. When a job is completed, output is printed through assignment of the Print package, in the order of job priorities.

In normal operation, job processing is initiated with the typein AUTO. which brings the following activities to control points.

Control Point Number	Job Name	PP Program	Activity
1	READ	1 LJ	Load jobs from card reader and store on disk.
2	PRINT	1 DJ(1 Printer) or 1 PJ(3 Printers)	Transfer job output from disk to printers.
3	NEXT	1 BJ	Search for job on disk to process at this control point.
4	NEXT	1 BJ	
5	NEXT	1 BJ	
6	NEXT	1 BJ	
7	-	-	No activity.

The jobs brought to control points are system jobs which process various phases of object jobs.

Example:

AUTO. is equivalent to keying:

1. READ.
2. PRINT.
3. NEXT.
4. NEXT.
5. NEXT.
6. NEXT.

Activities may also be dropped.

Example:

2. DROP. printing ceases, and output accumulates on the disk until PRINT or DUMP is introduced.

AUTO operation leaves control point 7 free, but an activity may be initiated at that point.

Example:

7. NEXT. enables control point 7 for object jobs.
7. LOAD. brings to control point 7 a package for loading jobs from magnetic tape to the disk job stack.

8.2.2 MANUAL MODE

Manual operation is similar to automatic except that each operation must be requested by the operator or the job itself. For example, after the dead start load has been executed, the operator may enter the control point number and the read statement, (2. READ.) on the keyboard; this will load the Read package in the next available PP and assign it to the designated control point.

If an input job stack is prepared off line, the operator enters the control point number and the load statement (3. LOAD.) on the keyboard, the Load package is loaded into the next available PP, the package requests the input unit number and continues.

To execute, the operator types n. NEXT. for each control point to be assigned a job. As each job is executed, output files are entered on the disk. No output processing will result until the operator enters the control point number and the print statement (4. PRINT.) An alternate method is to enter n. DUMP on the keyboard; the accumulated output files will be dumped on magnetic tape in the order of job priorities for printing off line. The operator may enter n. DROP. on the keyboard, which drops the job at the designated control point. After all input is read, the Read or Load routine may be dropped to allow another job to use the control point. This operation may be used in both automatic and manual mode.

8.3 CONSOLE AND DISPLAY SCOPES

Communication between the operator and the system is accomplished through the console keyboard entries and two or more console display scopes. The operator may introduce jobs, change priorities, and so forth. The operating system also allows the operator to examine selected portions of memory and keeps a permanent record of job history (dayfile) which can be called at any time.

The system communicates with the operator through the two visual tubes of the console display. Data is assembled and disassembled by individual routines in the operating system. At the request of the operator, all portions of job or system status may be displayed. Data entered at the console keyboard is also displayed on one of the scopes. A permanent record of system/console communication is retained in the dayfile and ultimately printed at the request of the operator.

The operator communicates with the system through the console keyboard (Figure 9). Keyboard typein messages are shown in the lower left-hand corner of the left display scope as they are being typed; the operator may check entries prior to initiating execution.

Each keyboard-initiated command to DSD or DIS is a single line. Backspacing causes the last character keyed to be blanked. Each command should end with the period followed by a carriage return, at which point the message is interpreted. If the command is acceptable, it is acted upon and the line on the screen is cleared; if not acceptable, FORMAT ERROR appears above the erroneous line. A blank typein can be used to clear an entire command.

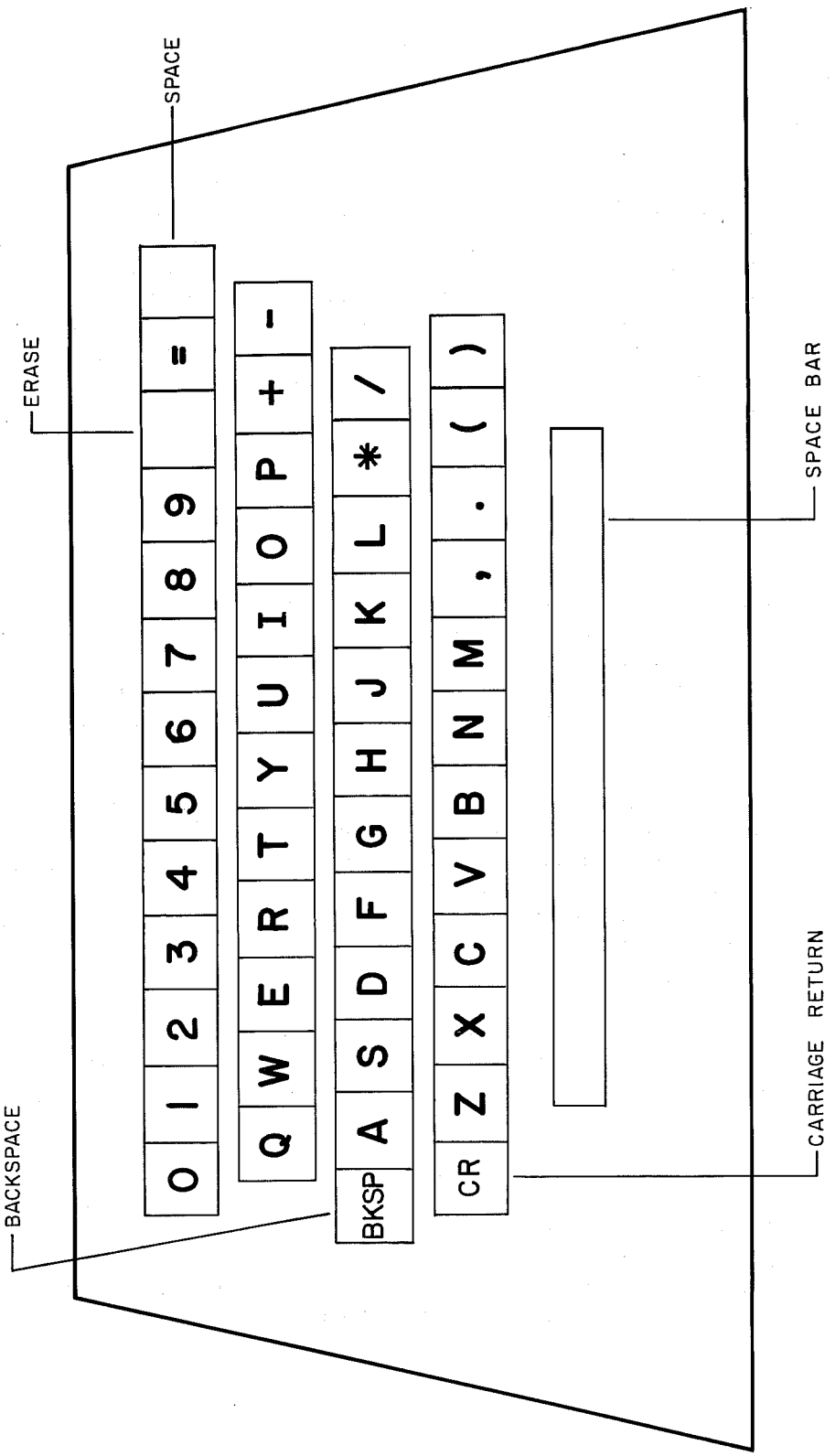


Figure 9. Console Keyboard

8.4 SYSTEM DISPLAY KEYBOARD ENTRIES

The keyboard, during the display of the overall system status, is used to initiate and control equipment assignment and job progress. The following table describes the keyboard codes and formats.

Table 5. SYSTEM DISPLAY KEYBOARD ENTRIES

Typein	Action Initiated
AUTO.	Used after dead start from system tape to initiate automatic job processing with card input and printer output.
STEP.	Selects a step mode for the operating system monitor in PP0. Requests from other PP's are processed when the keyboard space bar is pressed. High speed operation may be resumed by entering a period and depressing the carriage return key.
SIM.	Replaces the central processor with a simulator in a PP (the 007 location program); normally used in machine check out. Normal operation can be resumed by repeating SIM. Successive entries toggle the system between a simulated and a real central processor. The letter preceding central processor address on system display denotes the mode. P = xxxxxx real central processor S = xxxxxx simulated central processor
OFFxx.	Indicates to the system that equipment number xx must not be used, for example, during maintenance.
ONxx.	Returns equipment xx to the pool of available equipment.
TIME. 12.10.03, March 12, 1965.	Example of how to inform the system of clock time and date. The system uses this time (updated every second) for dayfile messages. If a TIME command is not given, the time since dead start is used.

Table 5. SYSTEM DISPLAY KEYBOARD ENTRIES (cont'd.)

Typein	Action Initiated
CONTROL POINT COMMANDS TO DSD	
Each of these commands is preceded by a control point number and a period. The numbers in the following examples are for illustrating format only.	
4. DROP.	Drop job at control point 4.
5. DIS.	Assign DIS package to control point 5. If there is no other console, DSD may relinquish the main console with the ASSIGN statement.
5. ASSIGN10.	Assign main console to control point 5.
1. READ.	Bring READ package to control point 1 for loading jobs from cards. This entry is needed if READ (usually initiated by AUTO) drops after reading a faulty job card.
2. PRINT.	Bring print job to control point 2; normally initiated by AUTO, but may be brought in separately.
7. LOAD.	Bring Load package to control point 7 to load jobs from tape to double file mark.
6. DUMP.	Bring Dump package to control point 6 to dump output files on tape.
(LOAD and DUMP request assignment of a tape unit).	
(READ, PRINT, LOAD, DUMP, and NEXT have no effect if the control point already has a job name).	
7. NEXT.	Bring job NEXT to control point 7 to look for a real job (on disk) for the control point.
6. GO.	Continue job at control point 6 if it has come to a pause (usually as a result of a FORTRAN pause statement, or if repeated tape transfers after parity failure is not effective).
5. ASSIGN53.	Assign equipment 53 to control point 5; normally used after a request for tape has been displayed, but any equipment may be assigned in this way.

Table 5. SYSTEM DISPLAY KEYBOARD ENTRIES (cont'd.)

Typein	Action Initiated
4. ONSW2.	Set sense switch 2 for FORTRAN program at control point 4. Settings are preserved at RA and at a word in the control point area.
4. OFFSW3.	Turn off sense switch 3 for control point 4.
DCN11.	Disconnect channel 11_8. (used by maintenance engineer)
FCN10.	Enter a zero function on channel 10_8. (used by maintenance engineer)

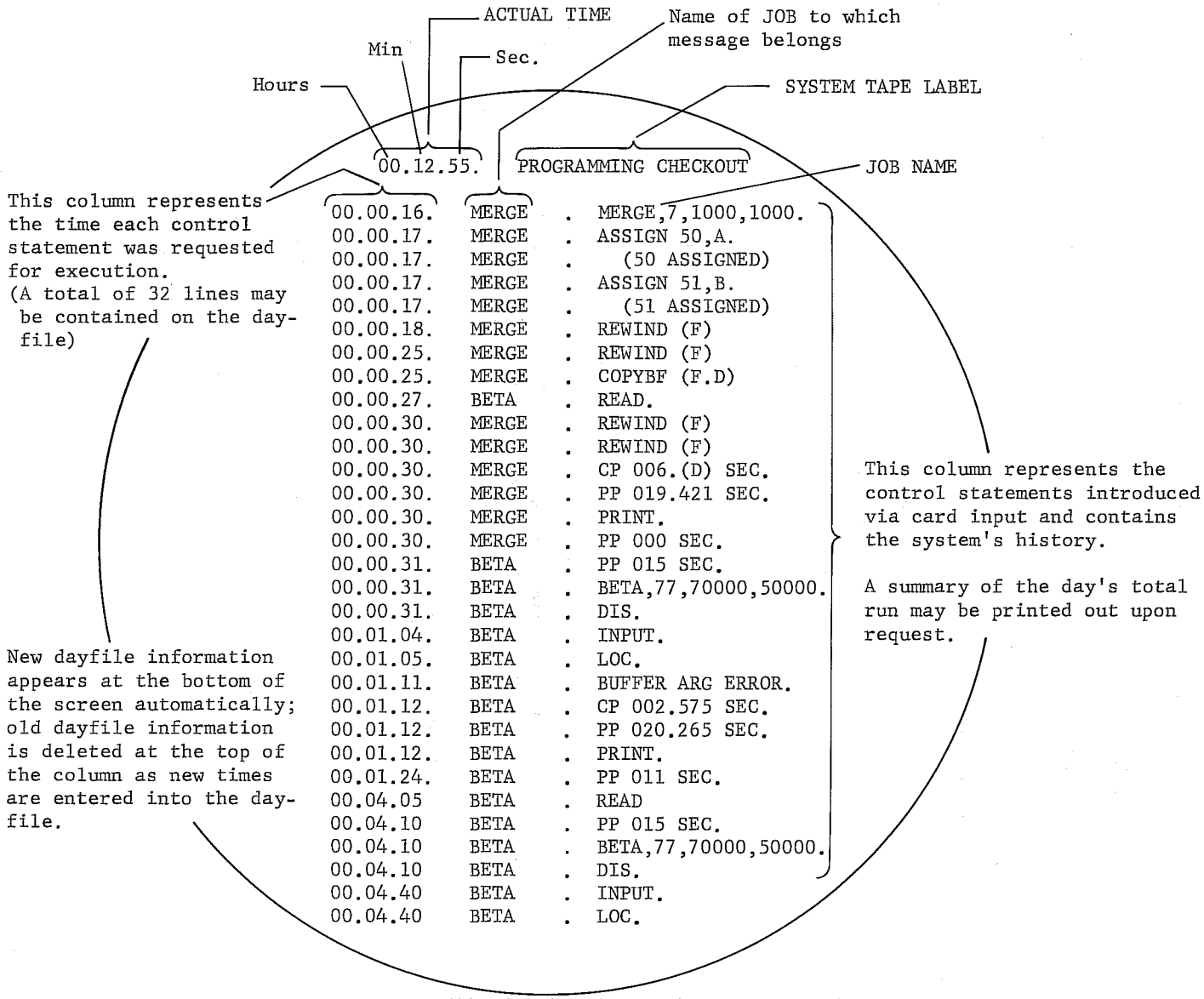
8.4.1 SYSTEM DISPLAY (DSD)

The system display program (DSD) is permanently located in PP9. DSD maintains a display on the two screens of the main console for all currently running jobs. The keyboard is used to initiate and control equipment assignment and job progress. During normal job processing, operators need not be concerned with the displays. However, the displays give an accurate picture of system progress; and, if needed, can be used for on-line debugging.

The console screens may be assigned any combination of two of the displays indicated in the table below. Figures 10, 12 and 13 illustrate typical views of system displays.

Table 6. SYSTEM DISPLAY CODES

Codes	Display
A	Dayfile
B	Job Status
C	Data Storage
D	Data Storage
E	Data Storage
F	Program Storage
G	Program Storage
H	Job Backlog



NOTE: Dayfile display data will appear on the printout at the end of each job automatically

Figure 10. Example of a Dayfile (A) Display

8.4.2 DAYFILE (A) DISPLAY

The dayfile consists of single line messages, each starting with the time and a job name:

```
02.14.33. TRIAL2C. RUN(S)
```

On the left is the actual time since dead start, or the real time if this was entered into the system by a TIME command to DSD.

Dayfile messages for a job are entered into the dayfile by PP programs as the job is processed or by central programs of the job. Each control card, including the job card, is listed at the time it is obeyed. The dayfile messages may be inspected as follows:

1) On a console screen (display A). The file is moved up the scope screen as messages are generated.

2) At the end of a job's printed output, all dayfile messages tagged with that job name are printed (Figure 11).

   Example:

10.28.01. SPECTRE. READ.	(Enters card reader)
10.28.12. SPECTRE. PP 011SEC.	(Time to read)
10.28.19. SPECTRE. SPECTRE,17,,100000.	(Job card, memory assigned)
10.28.19. SPECTRE. RUN.	(Control card, call RUN)
10.28.22. SPECTRE. END	(End of Compile, output by RUN)
10.28.22. SPECTRE. OP3	(Program ready to execute)
10.30.00. SPECTRE. CP 049.045SEC.	(Central processor time)
10.30.00. SPECTRE. PP 020.590SEC.	(PP time)

   The central processor time (in decimal) is that for the whole job, including compilation.

3) The dayfile is preserved on disk storage; its entire contents can be accessed for logging purposes (Figure 11).

Example:

By the job card sequence:

```
JBOND.
COMMON DAYFILE.
REWIND (DAYFILE)
COPYBR (DAYFILE, OUTPUT)
6,7,8,9
```

8.4.3 JOB STATUS (B) DISPLAY

For the Job Status (B) display (Figure 12), DSD provides the status of all control points, each in the following format:

```
N. JOBNAME PRIORITY, TIME LIMIT, RUNNING TIME S --------
    RA,          FL, EQUIPMENT NUMBERS.
LAST DAYFILE MESSAGE OR MESSAGE TO OPERATOR.
```

N is the control point number, S the central processor status (A, B, W, X etc.) or the job. If PP's are running at the control points, their numbers are entered in the first line. All other numbers of the display are in octal:

```
3. AJOB        7,          100,          37 X -2----7-
  30000, 40000, 51, 72.
RUN(S)
```

In this display, the job has used 37_8 seconds of central processor time so far, and is not using the central processor at present while PP's 2 and 7 are working.

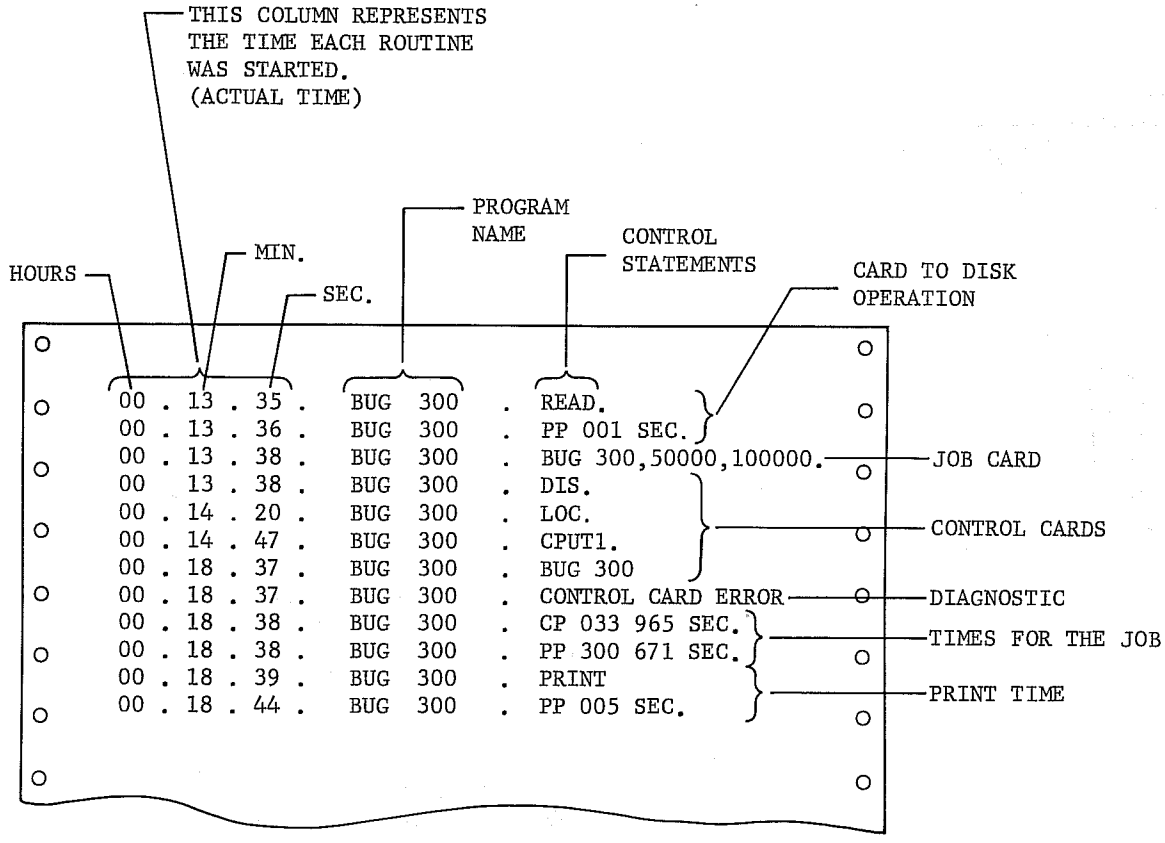


Figure 11. Example of Dayfile Printout

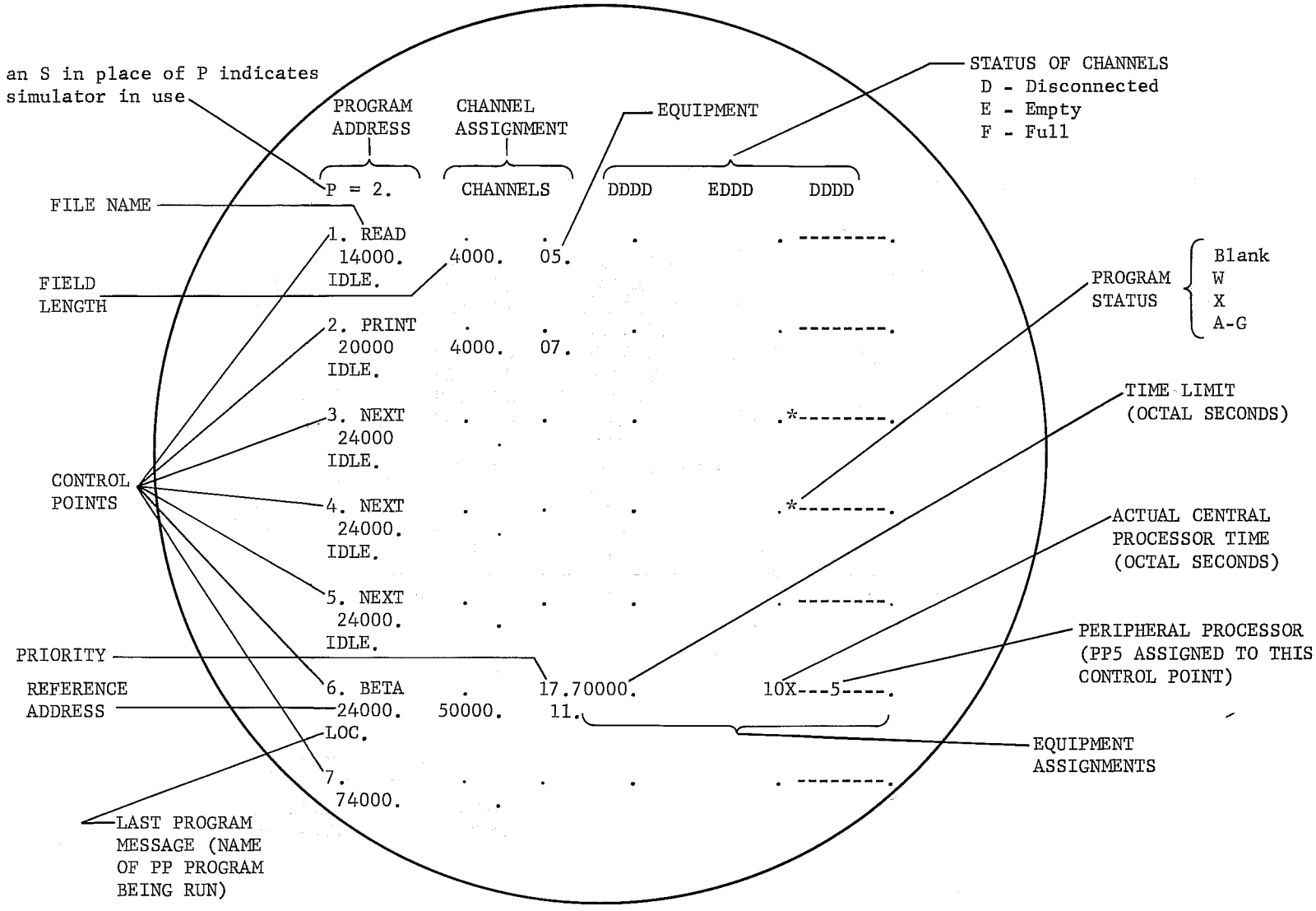


Figure 12. Job Status (B) Display

In the STATUS position, A is for the job currently using the central processor. B, C etc. are for jobs switched are for jobs switched away from the central processor because of insufficient priority. X is for a job involved in input/output, awaiting recall to the central processor. W is for a job waiting for the central processor that will not be admitted until its priority is higher than that of the A job. A job can proceed from X to A only via W status. A blank in the STATUS position indicates there is no need for the central processor at this stage of a job. A job name of NEXT indicates that a PP periodically searched the input file for a job to bring to the control point. A blank job name indicates there is no activity at the control point, which can be activated only by a command to DSD:

7.NEXT. - Look for a job for control point 7.

8.4.4 STORAGE DISPLAYS (C-G)

A storage display of type C through G (Figure 13) has 4 groups of central memory words, numbered 0 to 3. Each group consists of 8 central memory words, each displayed in octal on a separate line, preceded by its address. The address is absolute for DSD, relative for DIS. For data display (C, D, E), words are divided into 5 groups of 4 digits; for program display (F, G), words are divided into 4 groups of 5 digits.

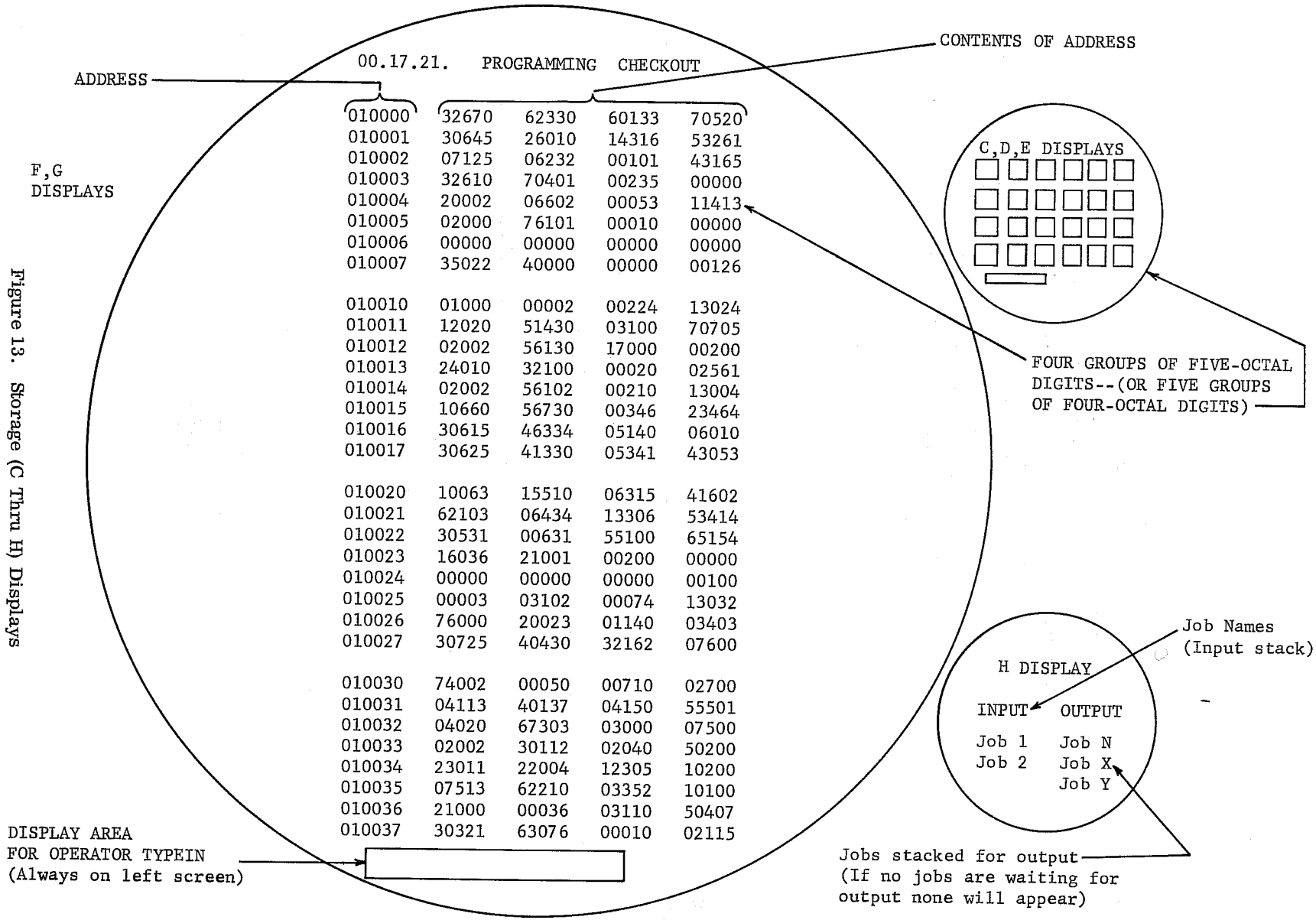
The address of an 8-word group to be displayed follows the display type and group number in a command:

D2,1020. Set words 001020-001027 in group 2 of D.

Group number 4 is used to set all four groups at once, to display 32 consecutive words:

D4,1020. Set words

	Group
001020-001027	0
001030-001037	1
001040-001047	2
001050-001057	3



010000	32670	62330	60133	70520
010001	30645	26010	14316	53261
010002	07125	06232	00101	43165
010003	32610	70401	00235	00000
010004	20002	06602	00053	11413
010005	02000	76101	00010	00000
010006	00000	00000	00000	00000
010007	35022	40000	00000	00126
010010	01000	00002	00224	13024
010011	12020	51430	03100	70705
010012	02002	56130	17000	00200
010013	24010	32100	00020	02561
010014	02002	56102	00210	13004
010015	10660	56730	00346	23464
010016	30615	46334	05140	06010
010017	30625	41330	05341	43053
010020	10063	15510	06315	41602
010021	62103	06434	13306	53414
010022	30531	00631	55100	65154
010023	16036	21001	00200	00000
010024	00000	00000	00000	00100
010025	00003	03102	00074	13032
010026	76000	20023	01140	03403
010027	30725	40430	32162	07600
010030	74002	00050	00710	02700
010031	04113	40137	04150	55501
010032	04020	67303	03000	07500
010033	02002	30112	02040	50200
010034	23011	22004	12305	10200
010035	07513	62210	03352	10100
010036	21000	00036	03110	50407
010037	30321	63076	00010	02115

Figure 13. Storage (C Thru H) Displays

Though only two of the A through H displays can be seen at a time, the settings for C through G are retained:

Successive commands:

CD.	Put C on left screen, D on right.
C4,400.	Look at words 400-437 on left screen.
D4,1000.	Look at words 1000-1037 on right screen.
AB.	Look at dayfile and job status.
CD.	Have another look at same words.

The three data displays (C, D, E) cannot be shown simultaneously, but they may be recalled.

A central memory word can be changed by a console command:

1022,2100 1000 0000 0111 2347.

Leading zeros can be dropped in both the address and data word, which are in octal.

8.4.5 JOB BACKLOG DISPLAY (H)

The H display under DSD shows a list of files of types input and output, giving the name and priority of each file. Input files are jobs waiting on disk; output files are the output of completed jobs, awaiting printing. The progress of any job not at a control point can be seen by using the H display.

8.5 JOB DISPLAY — DIS

A job display (DIS), similar to DSD, is used for information more relevant to a single job. Using DIS, the B display can demonstrate the exchange jump area of the job; central memory addresses relative to the job's reference address are used for data and program displays.

DIS can be called either from a control card (DIS.) or by a command to DSD. The job display package (DIS) may be called at any time during the execution of job. This package stops further automatic advance of the job control cards. The display covers only data pertaining to the particular job. The keyboard is used to advance the job control cards and to provide any two of the following displays in the same manner as for the DSD display.

The B display shows only the condition of the control point to which DIS is attached; it includes the next control statement, and a picture of the job's exchange package. The exchange package is displayed only while the job is in W, X or blank status. The operator may change priorities and suspend job execution with DIS.

Table 7. JOB DISPLAY CODES

Codes		
A	Dayfile	
B	Job Status	
C	Data Storage	5 groups of 4 octal digits per group
D	Data Storage	
E	Data Storage	
F	Program Storage	4 groups of 5 octal digits per group
G	Program Storage	

8.5.1 CHANGING PRIORITY

All jobs are assigned priorities from the job control card (0-17); a zero priority causes a job to be ignored. The operator may change the priority of any job through the keyboard, the range is $0\text{-}77_8$ with 77_8 being highest priority.

The operator may proceed as follows:

1. Type in: ENPR, dd.

   dd = a two-digit octal priority number

2. Press the carriage return key. If a priority change is attempted during a run, the program will stop (normal stop) and the operator may type in RCP. to resume central processor operation. A priority change may be made only when the job is assigned to a control point.

8.5.2 SUSPENDING EXECUTION

Suspending Job

To temporarily suspend execution of a job, the operator may type the following entries:

DCP. and carriage return

This will temporarily suspend the central processor and display the exchange jump area.

RCP. and carriage return

This will automatically request the central processor to begin execution at the next program address for a job suspended by a DCP. request.

Multiprocessing Termination

A control point may be associated with more than one piece of equipment. When this type of processing is encountered, subpoints are specified for the additional equipment used.

To terminate a single unit during multi-unit processing, the operator must proceed as follows:

1. Type in n. ENDx.

   n = control point

   x = subpoint

2. Press the carriage return key.

The following keyboard entries to DIS refer to the control point to which it is attached. Some of the entries cause the job to be switched away from the central processor. Execution can be resumed using RCP. or BKP. Numbers are in octal.

Table 8. ENTRIES FOR CHANGING JOB DISPLAY CONTENTS

ENP,12345.	Set P = 12345. (next instruction address, in exchange jump area).
ENA3,665000.	Set A3=665000 in exchange jump area.
ENB2,44.	Set B2=44 in exchange jump area.
ENX5,2223 4000 0000 0000 0200. (Spacing is unimportant)	Set X5=2223400000000000200 in exchange package.
ENEM,7.	Set Exit Mode = 7 in exchange jump area.
ENFL,10000.	Set FL=10000 in exchange jump area. (storage moved if necessary).
ENTL,200.	Set central processor time limit = 200_8 seconds.
ENPR,5.	Set job priority = 5.
DCP.	Drop central processor and display exchange jump area (in display B). When DIS is used, the exchange jump area is displayed in any case if the job does not have status A, B etc.
RCP.	Request central processor. This puts the job in W status, and it will use the central processor if its priority is sufficient. The register settings of the exchange jump area will be used.
BKP,44300.	Breakpoint to address 44300 in the program. Central processor execution begins at the current value of P and stops when P = 44300. DIS clears 44300 to stop the program at that point, and restores the original word when the stop occurs.
RNS.	Read and execute next control statement.
RSS.	Read next control statement and stop prior to execution.
ENS.xxxxxxxxxxxxxxx.	Allows the entry of any control statement xxxxxxxxxxxxxxx as if it had been entered on a control card. The statement can then be processed using RNS. or RSS.
GO.	Restarts a program which has paused.

Table 8. ENTRIES FOR CHANGING JOB DISPLAY CONTENTS (cont'd)

ONSW3.	Set sense switch 3 for the job.
OFFSW4.	Turn off sense switch 4 for the job.
HOLD.	DIS relinquishes the display console, but the job is held at the present status. A console must be reassigned to continue use of DIS.
DROP.	DIS is dropped and normal execution of the job is continued; it does not drop the job.
DMP(200,300)	Dump storage from 200 to 277 in the output file.
DMP(400)	Dump storage from the job reference address to 377.
DMP.	Dump exchange jump area to output file. (DMP formats are the same as if used on control cards).

8.6 DAYFILE MESSAGES

Dayfile messages that may be encountered during operation are listed below:

Routine	Message
1AJ	CP xxxx.xxx SECONDS.
1AJ	PP xxxx.xxx SECONDS.
1DJ	PRINT.
1LJ	READ.
1LJ	PP xxxx SEC.
1LT	LOAD.
1TD	DUMP.
2BP	BUFFER ARG. ERROR.
2BP	FNT LIMIT.
2BT	TAPE xx PARITY ERROR.
2EF	TIME LIMIT.
2EF	ARITH. ERROR.
2EF	PP CALL ERROR.

Routine	Message
2EF	OPERATOR DROP.
2EF	TRACK LIMIT.
2RD	DISK xx PARITY ERROR Gx Txxx Sxxx.
2RT	TAPE xx PARITY ERROR.
2SD	PP xxxx SEC.
2TJ	TOO MANY CONTROL CARDS.
2TJ	JOB CARD ERROR.
2TS	(xx ASSIGNED)
2TS	CONTROL CARD ERROR.
2WD	DISK x TRACK LIMIT.
2WT	TAPE xx WRITE PARITY ERROR.
DMP	DMP ARG ERROR.
EXU	PROGRAM NOT ON DISK.
EXU	PROGRAM TOO LONG.
LBC	LBC RANGE LIMIT.
LOC	LOC ARGUMENT ERROR.
MSG	MESSAGE FORMAT ERROR.
MSG	MESSAGE LIMIT.
PBC	PBC RANGE ERROR.
PPU	xx NOT IN PPLIB.
PPU	DISK xx PARITY ERROR Gx Txxx Sxxx.
DSD	FORMAT ERROR.
HLP	RPL OVERFLOW.

APPENDIX SECTION

SYSTEM TAPE AND LIBRARY MAINTENANCE A

SYSTEM TAPE

This tape contains all system library programs which are part of the Chippewa Operating System. The system tape is processed by the dead start loading described in section 8.1. When loading is initiated, the dead start panel instructions are transferred to PP0 and executed beginning at location 01. Processor 0 transfers words 6-15 to the processor on loading channel x, and resets itself to input mode. The instructions sent by PP0 to PPx are then executed in PPx.

SYSTEM TAPE ORGANIZATION

The first record of the system tape is transferred on data channel x to peripheral memory. The first record contains the peripheral resident program (common to all PP's) and a transient program which loads the remainder of the system tape. PPx transfers the peripheral resident program to the other nine processors.

Record 2(DSD) is transferred to PP9, the processor permanently assigned to system display.

Record 3 (MTR) is transferred to PP0, the processor permanently assigned to the system monitor. Record 4 is transferred to central memory to set the first 5000_8 words of central memory. PPx then disconnects all channels except x which starts execution in all ten processors. Records 5 and 6 (RSL and RPL) are single records containing central library and peripheral library programs, respectively; they are kept in reserved areas of central memory. The records from the end of RPL through a double 7, 8, 9 record are sent to a disk and entry is made in the peripheral library directory. The records from the double 7, 8, 9 to the zero length record are sent to a disk and entry is made in the central library directory.

Each record, following the zero length records, contains a central library subroutine or program to be stored on disk. The system loader builds an index of their positions in the central library directory in central memory. The system tape ends with a zero length record followed by a file mark.

The first word of a central or peripheral program contains its name and length (in central memory words). Several such programs constitute the single RSL or RPL record, but beyond these each program is a separate record.

Various peripheral routines with limited use, such as 2PC, DIS, DMP, etc., are punched separately and stored on disk. They enter the system via the peripheral library directory (PLD). All overlays called by "1" packages are in RPL because that is the only table searched for them. CIO uses 2PC, but coding to search PLD is included within CIO. PP resident always searches both RPL and PLD for a routine.

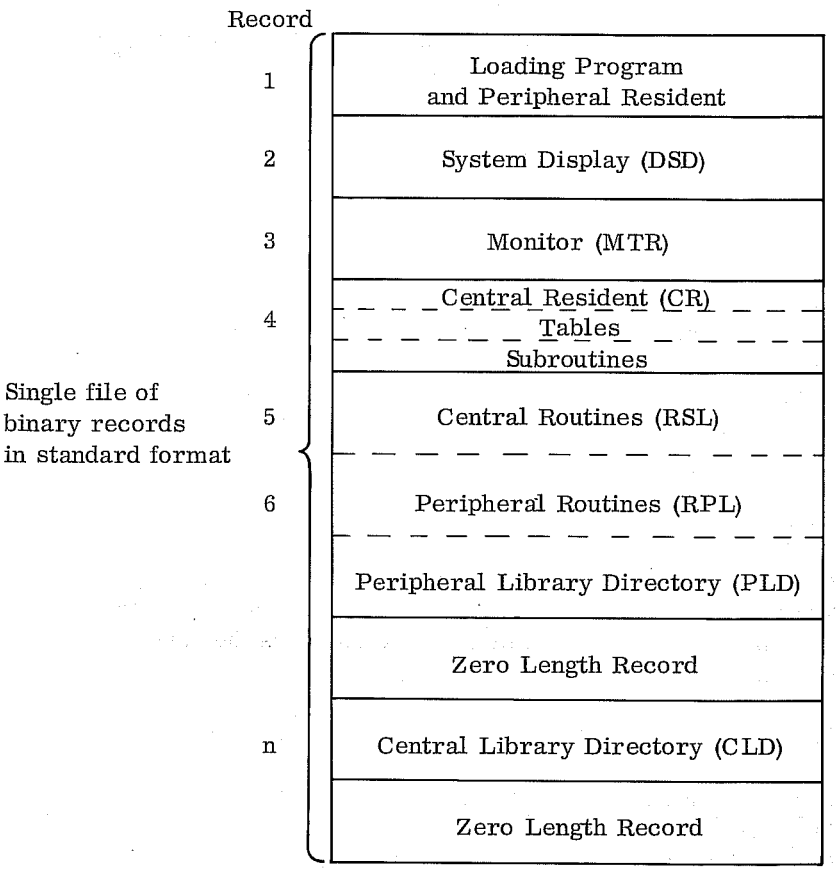


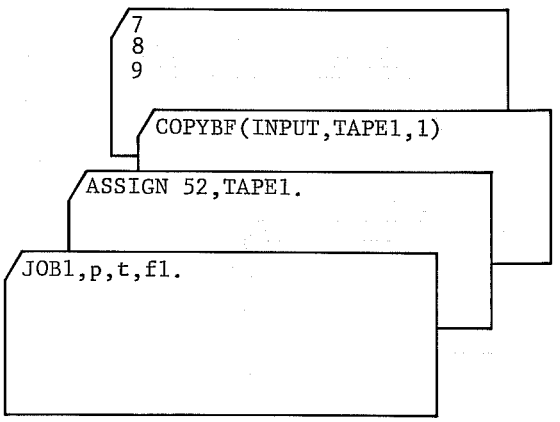
Figure A-1. System Tape Organization

PREPARATION AND FORMAT

Routines to be included in the system library are assembled as binary card decks. The input deck must be in the order in which it is to appear on the system tape. Each routine is followed by a record separator card (7, 8, 9 punches in column 1). All resident central processor routines and PP routines must be in the first group. An additional record separator (7, 8, 9) follows the last binary deck in the first group. This card produces a zero length record on the system tape when the full input deck is processed.

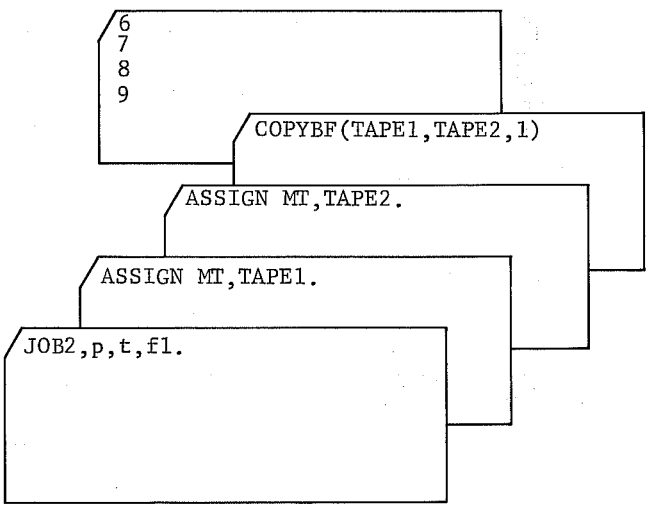
The record separator is followed by the binary card deck of all the central library routines, and again, each binary deck is followed by a record separator. The last central library routine must be followed by a file separator card.

The control cards for reading the cards are as follows:

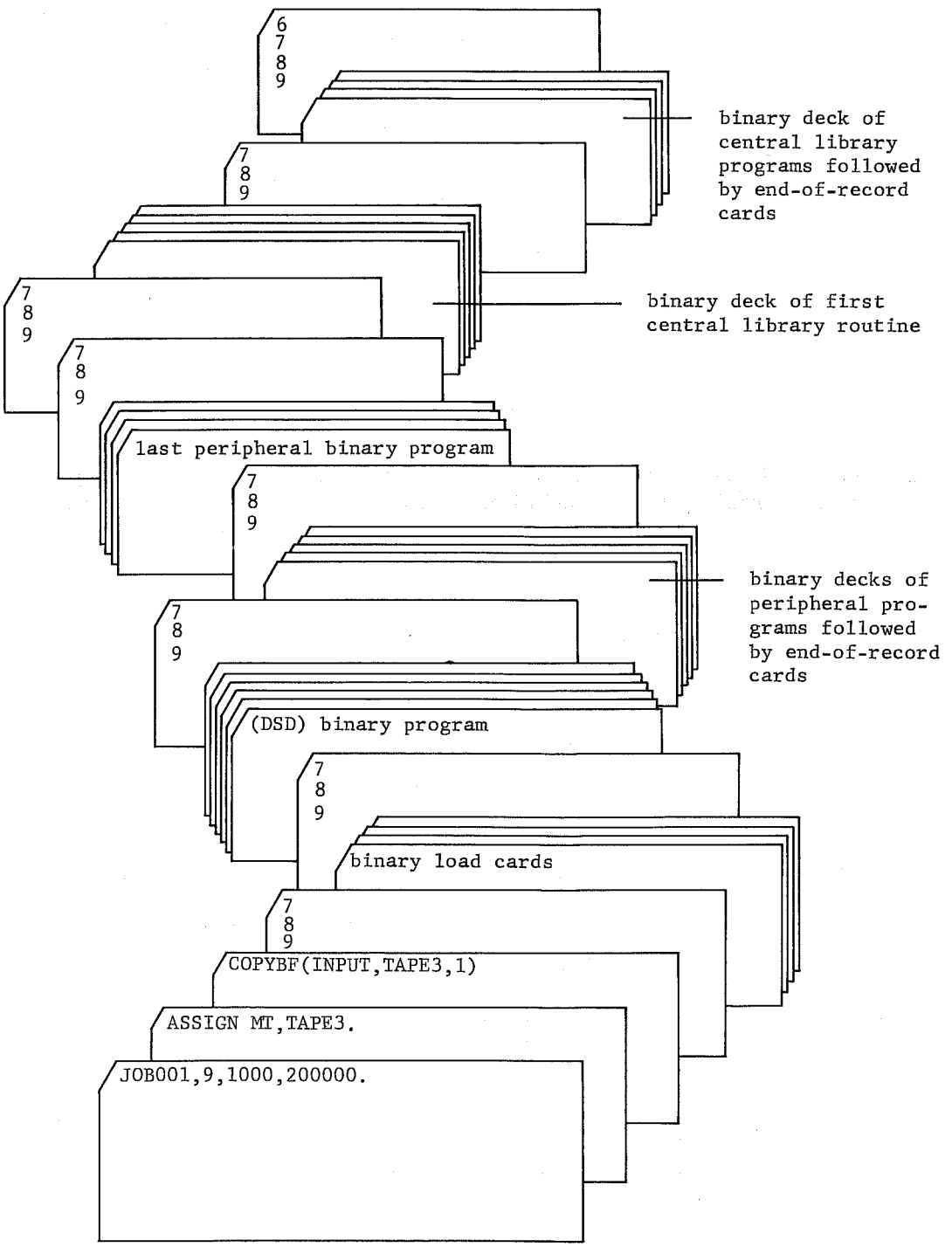


These cards, followed by the binary card decks will produce a systems tape on a half-inch magnetic tape, unit 52, reading from the INPUT file produced on the disk by the card read routine.

A copy of the system tape may be made with the following job deck with the original system tape as TAPE1 and the new tape as TAPE2:

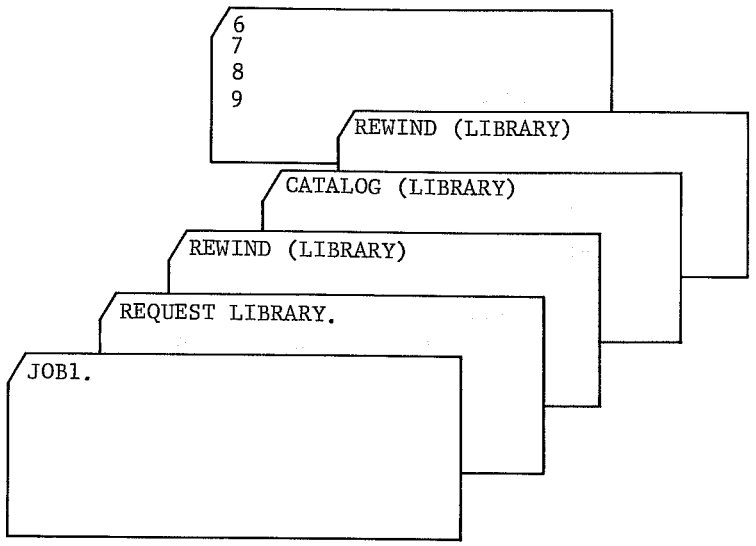


Deck Structure for Library Tape Preparation



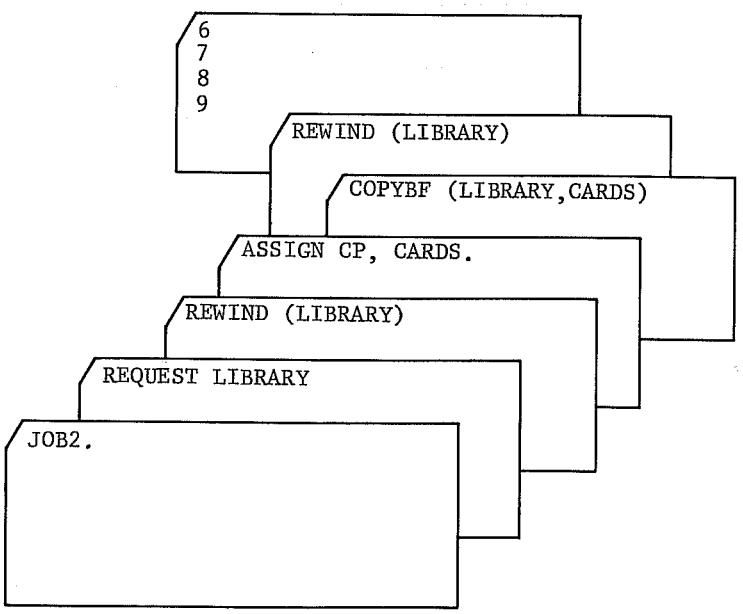
LIBRARY TAPE MAINTENANCE

A library program, CATALOG, is available to determine the names and lengths of programs in each record and produce a printed list to help maintenance. It is used as follows:



When the layout of a library tape is known, a new one may be produced tape-to-tape using COPYBR, COPYBF, LBC, LOC, RBR and WBR. A card-to-tape copy of system binary cards may also be made.

Selected binary records may be punched, or the entire tape may be punched out as follows:



Since PP and central library programs are always called for by name, the libraries may be extended by inserting new binary programs in the appropriate positions on the system tape. The modular structure of the system makes it easy to modify. Central memory space allocations determined by settings in the central resident, can be easily changed.

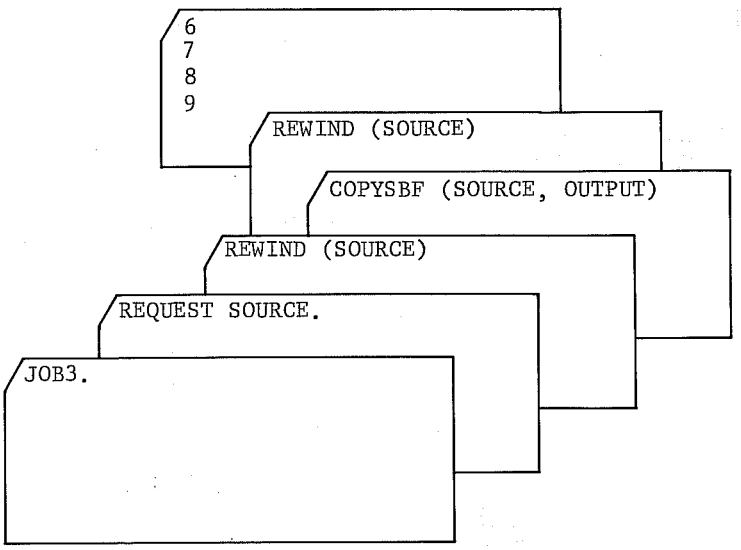
SYSTEM SOURCE CARDS

A separate tape contains the source cards.

First file	Peripheral programs
Second file	Central library programs
Third file	RUN compiler

This tape, written in binary, contains blocked coded cards. Each program is a separate logical record. It may be printed or punched with standard utility routines.

To list a whole file:



CENTRAL AND PERIPHERAL LIBRARY PROGRAMS B

The Resident Peripheral Library contains the following programs:

1AJ	Advance job	2RD	Read disk
1BJ	Begin job	2RT	Read tape
1DJ	Phase three print	2SD	Search day file
1LJ	Phase one card load	2TJ	Translate job card
2BD	Backspace disk	2TS	Translate statement
2BP	Process buffer parameters.	2WD	Write disk
2BT	Backspace tape	2WT	Write tape
2DF	Drop file	CIO	Circular I/O
2DT	Drop tracks	CLL	Call library
2EF	Error flag	EXU	Execute
2LP	Line printer	MSG	Message for day file
2RC	Read cards		

The Peripheral Library Directory lists the following programs and their disk file locations:

1LT	Phase one tape load
1TD	Phase three tape dump
2PC	Punch cards
DMP	Dump storage
LBC	Load binary cards
LOC	Load octal cards
PBC	Punch binary cards
DIS	Display job
SIM	CPU simulator
PAS	Assembler for PP code which executes in a PP

The Resident Central Library contains the following subroutines:

ALOG	Computes the natural logarithm of a floating point number. The result for a negative argument is indefinite.
ALOG10	Computes the common logarithm of a floating point number. The result for a negative argument is indefinite.
ATAN	Computes the arctangent of a floating point number.
COS	Computes the cosine of a floating point number.
DVCHK	Divide and check.
END	Causes end-of-file on all circular buffers for the program.
EXIT	Terminates a program as END. The word EXIT is entered in the day file.
EXP	Computes the floating point function E**X.
IBAIEX	Computes I**J where both I and J are fixed point.
IFENDF	Checks the status of a circular buffer for an end-of-file condition.
OVERFL	Result is set to one if overflow, and two if no overflow.
PAUSE	Loop on recall until operator action. The word PAUSE is entered in the day file.
RBAIEX	Computes A**I where A is floating point and I is fixed point.
RBAREX	Computes A**B where both A and B are floating point.
REMARK	Transmits a message to the system day file and displays it on the console display.
SIN	Computes the sine of the argument.
SQRT	Computes the square root of the argument. A negative argument results in an indefinite result.
SLITE	Sense lights. Turn off if zero. Turn on if non-zero.
SLITET	Respond with 1 if light was on, 2 if light was off. Turn off light.
SSWTCH	Respond with 1 if switch is down, 2 if switch is up.
START	Enter START in day file.
STOP	Enter STOP in day file and perform same function as END.

TAN	Compute the tangent of the argument.
TANH	Compute the hyperbolic tangent of the argument.
TIME	Enter the word TIME and Hollerith information in the day file.

The following central subroutines and programs are called from the disk by referring to the Central Library Directory (CLD):

FORTRAN Subroutines

BACKSP	Backspace medium
CHAIN	Loads a program from the disk file and executes it. Parameters passed from one program to another must be in the common region. All segments to be chained must be compiled with the same file names. The segment name is transferred to the day file and displayed prior to execution.
DISPLA	Displays a variable name and its numerical value as an integer if unnormalized, in floating point format if normalized.
ENDFIL	Write end-of-file.
INPUTB	Binary input
INPUTC	FORTRAN data input
OUTPTB	Binary output
OUTPTC	FORTRAN data output
RANF	Random number generator
REWINM	Rewind medium
XLOCF	Returns the memory location of a variable name.

Programs

APRAB	Assemble peripheral overlay
BKSP	Backspace medium
COPY	Copy to double file mark
COPYBF	Copy binary file

COPYBR	Copy binary record
COPYCF	Copy coded file
COPYCR	Copy coded record
COPYSBF	Copy shifted binary file
REWIND	Rewind medium
RETURN	Release equipment from control point assignment
RUN	Compile and run FORTRAN
UNLOAD	Rewind and unload medium
VERIFY	Verify two media

SUGGESTIONS FOR WRITING PP PROGRAMS C

The following miscellaneous data contains hints to aid the programmer in writing peripheral programs for use under control of MTR.

1. The control point reference found in the PP communication area input location is the control point number not the address. The control point address can be determined by shifting the control point number left 7 bits.

2. The reference address (RA) and field length (FL) in the control point are in octal hundreds.

3. A check must be made to determine whether the field length (FL) is large enough to contain the buffer parameter area (BA) and the buffer itself.

4. Periodic PAUSE functions (0017) should be issued throughout a PP program; once per record in I/O routines should be sufficient. When a NOT READY condition is encountered, PAUSE should be executed between each STATUS sense. In addition to allowing storage shifts, the PAUSE provides an abort mechanism.

5. An I/O channel must be reserved (0002) by the PP to prevent another PP from using the channel at the same time. This reservation is normally just for the duration of the actual passage of data. The reservation should be dropped during housekeeping periods between I/O records. It must be dropped in all cases when a PP program terminates. Care must be exercised to DROP. (0003) a channel only if it has been previously reserved by this PP.

6. There are two pseudo-channel reservations (14 and 15) which are used to interlock access to the File Name Table/File Status Table (FNT/FST). This prevents one PP from making a change in the table while another is reading it. (Reference central memory 00004)

7. Three tables, FNT/FST and EST (Reference central memory 00005), are the keys to all PP I/O requests.

   A request entered into a PP specifies an operation on a file name. With this information and the control point number, the PP program must use the tables to find which piece of I/O equipment it should drive and the attendant information concerning channel, synchronizer, unit number, etc.

8. When the disk is used, the programmer should be aware of the table entries concerning current disk position. (FNT/FST - word two, bytes 3, 4). If a file is assigned to the disk, the second word of the FNT/FST entry for that file contains the current track and current sector for that file: this may or may not be the current position for that disk. The current position for the disk always appears in byte 2 of the TRT pointer word for that disk. Determining the status eliminates unnecessary positioning functions. Status checking must be used by all disk handling routines. A positioning function, available for disk 0 in the idle loop (PP resident), is entered with the track number in A, and a channel is reserved. Do a return jump to location 0701.

9. A disk parity error will terminate the system and make an entry in the disk flaw table. Return jump to location 0201 with location 0006 preset to the track number, and location 0007 preset to the sector in which the parity error was detected.

10. Monitor function calls from PP programs are all handled by a subroutine in PP resident. Return jump to location 0761 with a request preset in locations 0011-0014 and the request number in A. If there are no parameters associated with the request, it is not necessary to clear 0011-0014. (See 0001, 0004, 0005, 0011, 0012, 0013, 0015, 0016, 0017, 0020, etc.) (Location 0661 may be used to wait for a cleared output register.)

11. Channel reservation function (0002) or drop channel function (0003) may be handled as in number 10 (RJ to 0761); or it may be handled directly by presetting A to the channel number and performing a return jump to 0741 for reserve (0002) or return jump to location 0751 for drop (0003) function.

12. Since PP's have access to every part of central memory, the programming interlocks must be adhered to in all cases. For this reason, a peripheral routine which is not completely debugged can cause strange results to totally unrelated programs. Therefore, debugging of PP programs should take place during a period which will not cause a hardship if a Dead Start is necessary.

13. A central processor program with priority zero will never be executed. This priority is used whenever a PP routine needs buffer area in central memory. It is used by Read and Print PP routines.

INDEX

CONTROL DATA
CORPORATION

COMMENT AND EVALUATION SHEET

6000 SERIES COMPUTER SYSTEMS
Chippewa Operating System Reference Manual
Pub. No. 60134400 December, 1965

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 REFERENCE.

FROM NAME: _____

BUSINESS ADDRESS: _____

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
Documentation Department
3145 PORTER DRIVE
PALO ALTO, CALIFORNIA

FOLD

FOLD

STAPLE

STAPLE

CONTROL DATA SALES OFFICES

ALAMAGORDO • ALBUQUERQUE • ATLANTA • AUSTIN, TEXAS • BILLINGS
BIRMINGHAM • BOSTON • BOULDER, COLORADO • CAPE CANAVERAL
CHICAGO • CINCINNATI • CLEVELAND • COLORADO SPRINGS • DALLAS
DAYTON • DENVER • DETROIT • DOWNEY, CALIFORNIA • GREENSBORO,
NORTH CAROLINA • HARTLAND, WISCONSIN • HONOLULU • HOUSTON
HUNTSVILLE • MIAMI • MONTEREY, CALIFORNIA • INDIANAPOLIS • KANSAS
CITY, KANSAS • LOS ANGELES • LAS VEGAS • MADISON, WISCONSIN
MINNEAPOLIS • NEWARK • NEW ORLEANS • NEW YORK CITY • OAKLAND
OMAHA • PALO ALTO • PHILADELPHIA • PHOENIX • PITTSBURGH
ROCHESTER, NEW YORK • SACRAMENTO • SALT LAKE CITY • SAN
BERNARDINO • SAN DIEGO • CONDADO, PUERTO RICO • SANTA BARBARA
SAN FRANCISCO • SEATTLE • ST. LOUIS • TULSA • WASHINGTON, D. C.

AMSTERDAM • ATHENS • BOMBAY • CANBERRA • DUSSELDORF • FRANK-
FURT • HAMBURG • JOHANNESBURG • LONDON • LUCERNE • MELBOURNE
MEXICO CITY • MILAN • MONTREAL • MUNICH • OSLO • OTTAWA • PARIS
TEL AVIV • STOCKHOLM • STUTTGART • SYDNEY • TOKYO (C. ITOH ELEC-
TRONIC COMPUTING SERVICE CO., LTD.) • TORONTO • ZURICH

CONTROL DATA
CORPORATION

8100 34th AVE. SO., MINNEAPOLIS, MINN. 55440

PUB. NO. 60134400

LITHO IN U. S. A.