Burroughs

B 1700 SYSTEMS

Master Control Program (MCP)

REFERENCE MANUAL

PRICED ITEM

Burroughs

B 1700 SYSTEMS
MASTER CONTROL PROGRAM
(MCP)

REFERENCE MANUAL

Burroughs Corporation
Detroit, Michigan 48232

PRICED ITEM

Printed in U.S. America 8-75 1088010

Correspondence regarding this document should be forwarded using the Remarks Form at the back of the manual, or may be addressed directly to Publications Department, Technical Information Organization, TIO-West, Burroughs Corporation, 9451 Telstar Avenue, El Monte, California 91731.

TABLE OF CONTENTS (Cont)

INTRODUCTION

The Master Control Program for the B 1700 is responsible for managing the demands and resources of the system. The concepts and functions of the MCP cannot be viewed as a series of step-by-step procedures, but rather as a series of continuous processes, each capable of functioning independently, but occurring simultaneously.

The purpose of the B 1700 Master Control Program Reference Manual is to provide an insight into the operating environment of the B 1700, and to segregate the internal components of the Master Control Program (MCP) in order to analyze their relationship in respect to the overall management of the system.

The information contained in this manual reflects System Software Release Mark IV.0.

OVERVIEW OF THE MASTER CONTROL PROGRAM

The primary function of the Master Control Program (MCP) is to optimize the productivity of the B 1700 computer system. External intervention is held to the absolute minimum, and maximum throughput is achieved by incorporating into the MCP the primary tasks of I/O control, file handling, multiprogramming, interrupts, memory allocation, user programs, and operator interface.

The MCP is mainly disk resident. It consists of many routines and functions, which, when combined, form a single program. Because of the size of the MCP, well over a million bits, only those segments of the MCP that are needed at a specific time are brought into memory.

The MCP is written in SDL (Software Development Language) and, therefore, the SDL Interpreter or portions of it are always active.

The MCP is an operating system. It manages the demands and resources of the B 1700 system and reduces programming effort by providing a "family" of commonly needed functions and services. The MCP requires strict adherence to program structures and procedures in order that there may be communication between different programs, interpreters, and the MCP itself.

COLDSTART, the function performed to initiate the system, structures the system disk as required by the MCP, as shown in figure 1. Clear/Start, the function performed to begin processing, structures memory as required by the MCP, as shown in figure 2. The first duty of the MCP after a Clear/Start is to perform some initialization of its own, after which, it enters its "main loop" and is susceptible to control input.

LIST OF APPLICABLE B 1700 PUBLICATIONS

The following is a list of publications relative to the B 1700 and referenced in this manual:

Publication Title	Form No.
B 1700 System Software Operational Guide	1068731
B 1700 System Reference Manual	1057155

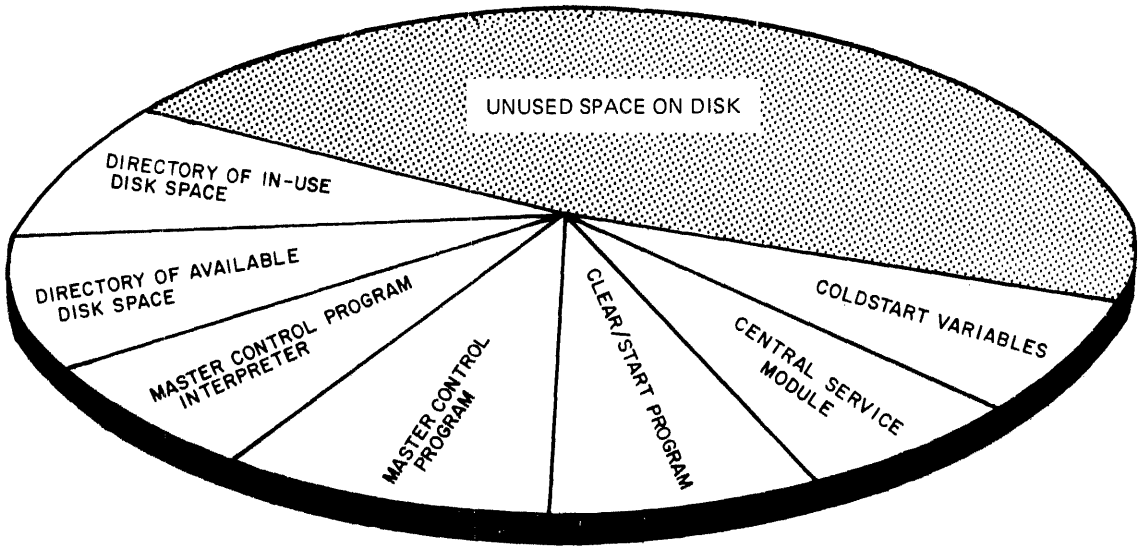


Figure 1. System Disk at COLDSTART

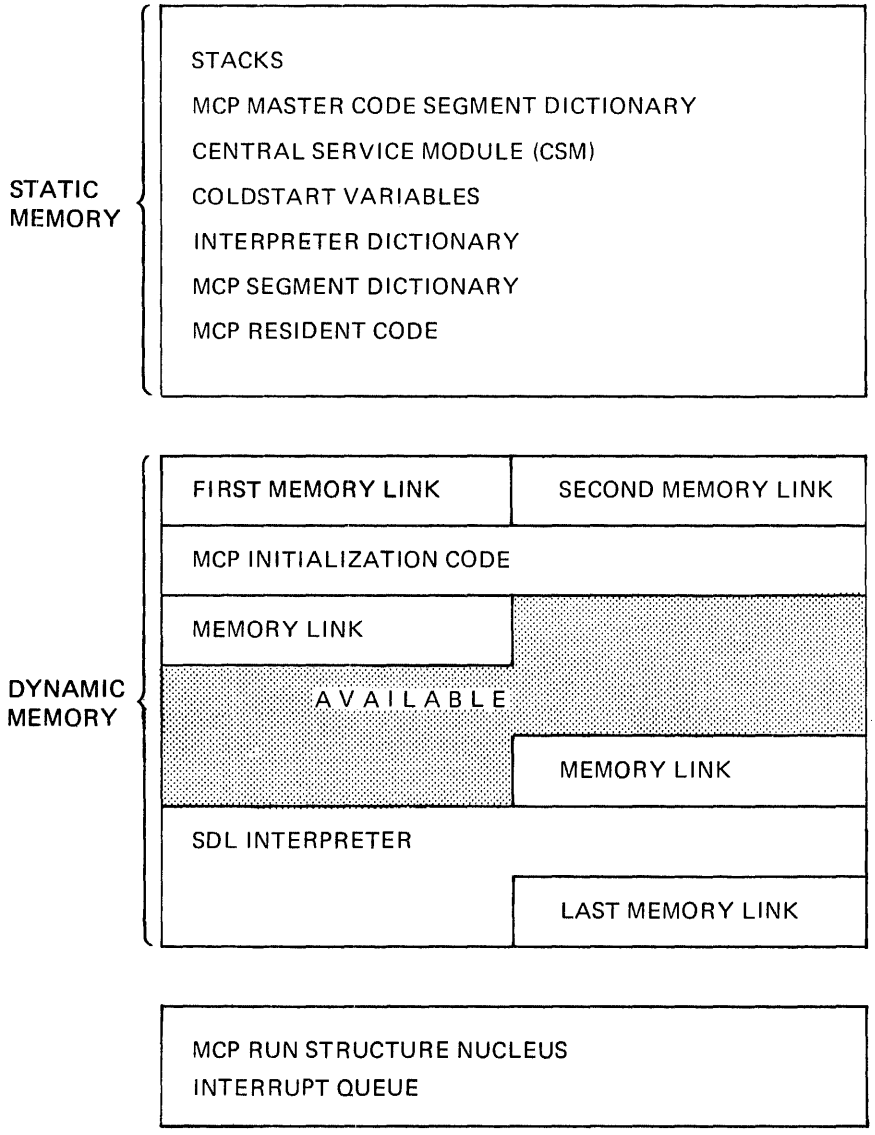


Figure 2. B 1700 Memory After Clear/Start

SECTION 1
MCP - PROCESSOR INTERFACE

MEMORY MANAGEMENT

There are two different memories within the B 1700 processor: Main Memory, and Control Memory (B 1720 series only).

Control Memory

The MCP and control memory are not directly related and have independent functions relating to the system. The purpose of control memory is to house the microinstructions. The larger the control memory, the faster the system throughput, since control memory is up to four times faster than main memory. If a set of microinstructions exceeds the available control memory, the MCP stores the overflow in main memory. Control memory is available for the B 1720 series of systems only. For a further discussion concerning control memory and microinstructions refer to the B 1700 Systems Reference Manual, Form No. 1057155.

Main Memory

Main memory is available to all programs, including the MCP. It is addressable to the individual bit and has variable operational lengths from 1 to 24 bits. Main memory has no physical *word* or *byte* boundaries which are visible to the rest of the system. Main memory is shared by all programs requiring it, and is divided into variable-size areas. Figure 1-1 illustrates main memory format.

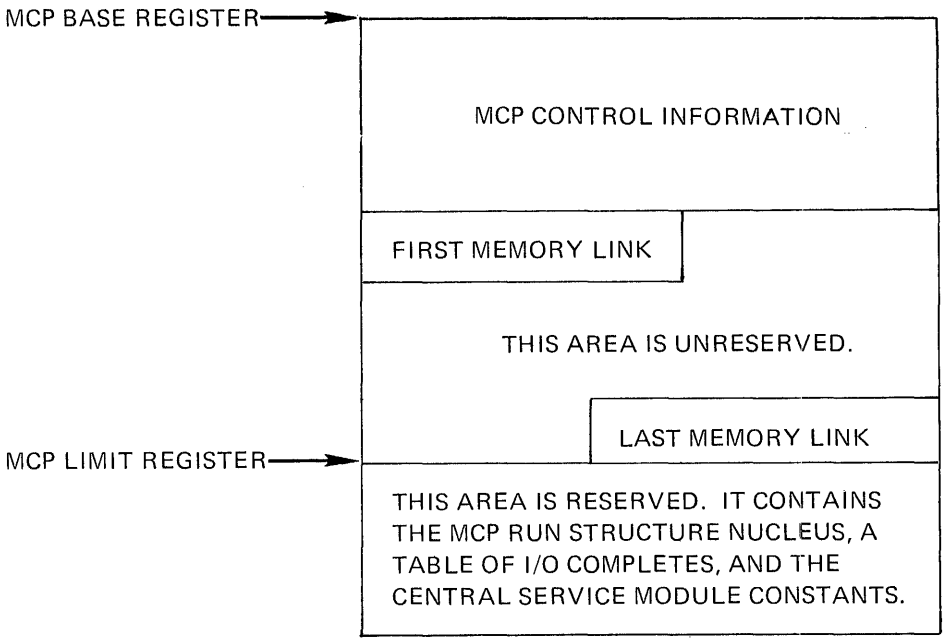


Figure 1-1. Main Memory

Memory Links

The memory link, vital to the functional operation of the MCP, is a string of bits containing information delimiting the area that the MCP allocates. Memory links contain information such as the size of the area allocated, an indication as to what program has control of the block, its usage, and a forward and backward reference to adjacent memory links. Table 1-1 describes a memory link format.

Table 1-1. Memory Link

Field Name	Type	Length	Description
01 MEMORY.LINK	Bit	175	
02 ML.BACK	Bit	24	Points to the preceding memory link. When this is the first memory link, ML.BACK contains @FFFFFF@.
02 ML.FRONT	Bit	24	Points to the next memory link. When this is the last memory link, ML.FRONT contains @FFFFFF@.
02 ML.SIZE	Bit	24	Contains the number of bits from the end of the memory link to the beginning of the next memory link.
02 ML.GROUP	Bit	55	
03 ML.POINTER	Bit	24	Points to the system descriptor that refers to the information stored in this memory area.
03 ML.MIX	Bit	6	Contains the program mix number using this area. The MCP mix number is always zero.
03 ML.SAVE	Bit	1	Indicates whether this field can (ML.SAVE=0) or cannot (ML.SAVE=1) be overlaid.
03 ML.TYPE	Bit	24	Defines the type of data contained within the area. When ML.TYPE contains a 2, MEMORY.LINK is 175 bits; otherwise, it is 163 bits. The following is a list of area types: Type — Code Code — 0 Data — 1 Available — 2 Run structure — 3 MCP (temporary) — 4 User file — 5 Code segment dictionary — 6 Interpreter — 7 Dictionary master — 8 Queue directory — 9 Message buffer — 10 Message list — 11

Table 1-1. Memory Link (Cont)

Field Name	Type	Length	Description
			Type / Code Ready to be made available 12 Data segment 13
02 ML.AVL	Bit	48	
03 ML.F.AVL	Bit	24	Points to the next "available" memory link. When this is the last link, ML.F.AVL contains @FFFFFF@.
03 ML.B.AVL	Bit	24	Points to the preceding "available" memory link. When this is the first link, ML.B.AVL contains @FFFFFF@.
02 ML.DISK Remaps ML.AVL	Bit	36	The ML.DISK field, comprised of the first 36 bits of the ML.AVL field, is the disk address associated with the information stored in the memory block described by this memory link.

Memory Areas

Main memory space is classified into two types: available and in-use. Any area not currently being used is marked "available." All other areas are considered by the MCP to be "in-use."

Obtaining Memory Space

Memory space is obtained by the MCP using the GETSPACE procedure. The various MCP routines that reference GETSPACE pass parameters such as the memory size requirement, mix number of the requesting program, and the address and type of data to be placed into the memory area.

GETSPACE then scans memory links, starting at the first memory link, searching for available area. Once available space is found (ML.TYPE = 2), a memory link associated with that area is constructed and the GETSPACE routine is exited, returning that memory address to the routine that requested the space. If no space is available, (ML.TYPE $\neq$ 2), GETSPACE returns @FFFFFF@. The MCP, depending on the situation, can overlay areas to obtain a large enough contiguous area of memory.

Available areas (ML.TYPE = 2) can be used by any program, including the MCP. In most cases the available area is larger than the actual space needed. Since it is, however, inefficient to allocate more memory than required, the MCP allocates only the required space and then delimits the area with memory links. Figure 1-2 depicts a typical memory map.

First and Last Memory Links

The MCP maintains the address of the first and last memory links. These links, referred to as dummy links, make it possible to scan total memory by delimiting the scanning search technique.

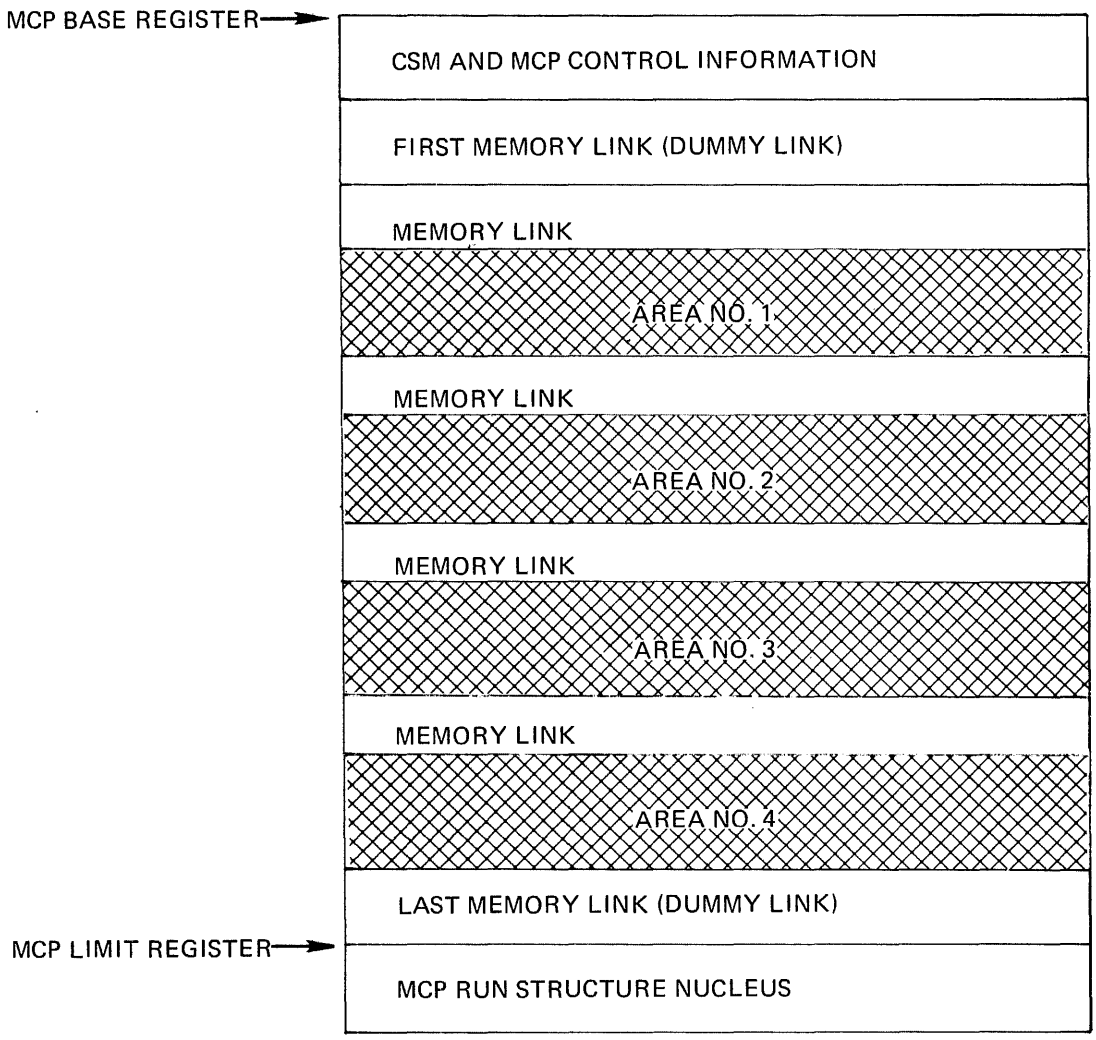


Figure 1-2. Typical Memory Map

Overlayable/Non-Overlayable Data

The MCP, when unable to find sufficient area to allocate during its first search through memory, then seeks for those areas that can be overlaid, which in most cases are areas that already have copies on disk, such as code segments. All program code is overlayable because it has copies on disk. When a code segment is overlaid, the starting disk address, contained in ML.POINTER, is returned to the respective Code Segment Dictionary entry. That dictionary entry is then marked "not present," indicating the code is not residing in memory. The memory link field, ML.TYPE, is then marked "available" and the area is free to be used. When data other than code has to be overlaid, a copy is written to disk before the area is made available if a copy does not already exist. The MCP does have the ability to combine contiguous overlaid areas in order to provide the requested space.

INTERRUPTS

A B 1700 interrupt is discovered by the interpreter and handled by the Central Service Module (CSM) or the Master Control Program. Interrupts on the B 1700 are not serviced by hardware functions but by system software. This function is termed a "soft" interrupt; therefore, the term "interrupt" implies a "soft" interrupt unless otherwise specified.

B 1700 interrupts are classified into two types: external and program dependent. Interrupts, per se, are discovered by interpreters during their fetch cycle (the process of decoding an S-op).

When discussed in conjunction with the MCP, interrupts can seem ambiguous in that the MCP never "sees" processor interrupts as such. They are intercepted by the interpreters and, if necessary, translated into communicates for the MCP in the following categories.

a. Program-generated communicates

b. Interpreter-generated communicates.

c. CSM-generated communicates.

The CSM-generated communicates include:

a. Pointers to I/O descriptors representing I/O interrupts.

b. Entries representing Code Segment Dictionary addresses for needed MCP segments.

c. Run Structure Nucleus Communicate Message Pointers of normal state object programs indicating one of the following:

1. Interpreter segment not present.

2. Code segment not present.

3. CSM segment not present.

4. Data segment not present.

5. Hi-priority I/O complete.

6. Trace print.

Table 1-2 describes the types of interrupts and the methods in which they are handled.

Table 1-2. Interrupt-Handling on the B 1700

Type of Interrupt	MCP Control		User Program Control	
	Interpreter Function	CSM Function	Interpreter Function	CSM Function
SERVICE REQUEST	Passes control to CSM.	Yields control to I/O driver; then returns to interpreter.	Passes control to CSM.	Yields control to I/O driver; then returns to interpreter.
TIMER	Passes control to CSM.	Increments Timer and returns control to MCP.	Passes control to CSM.	Increments Timer. If overflow occurs, return is to the MCP; if not, return is to the interpreter.
I/O INTERRUPT	Passes control to CSM.	Stacks the I/O request and returns control to MCP.	Passes control to CSM.	Stacks the I/O request and returns to the MCP.

Table 1-2. Interrupt-Handling on the B 1700 (Cont)

Type of Interrupt	MCP Control		User Program Control	
	Interpreter Function	CSM Function	Interpreter Function	CSM Function
MEMORY PARITY	Causes halt and display.		Builds the message and returns to MCP.	
OUT OF BOUNDS	Causes halt and display.		Builds the message and returns to MCP.	
CONSOLE HALT	Causes halt and display.		Causes halt and display.	
PROGRAM DEPENDENT	Causes halt and display.		Builds the message and returns to MCP.	

Program Communicates

All information in a program communicate is contained within the memory area allocated to the program with the communicate message pointer of the program referencing the address and size of the communicate message. This differs from interrupt communicates in that the interrupt message is contained within the communicate itself. Table 1-3 describes the type of program communicates (RS.ITYPE = 01) referenced by the message pointer in the Run Structure Nucleus.

Table 1-3. Program Communicates

Type Code	Description	Type Code	Description
0	Undefined	12	Access File Information Block
1	Read	13	Data overlay
2	Write	14	Access disk file reader
3	Seek	15	Undefined
4	Sorter control	16	Undefined
5	Sorter read	17	Undefined
6	Undefined	18	Undefined
7	Undefined	19	Undefined
8	Open file	20	Terminate (End-of-Job)
9	Close file	21	Undefined
10	Position file	22	Time or date
11	Access File Parameter Block	23	Undefined

Table 1-3. Program Communicates (Cont)

Type Code	Description	Type Code	Description
24	Snooze	35	Freeze/Unfreeze
25	ZIP	36	Compile card information
26	Accept	37	Dynamic Memory Base
27	Display	38	Memory Dump
28	USE return	39	Undefined
29	Sort handler	40	Undefined
30	Trace	41	Data Comm Queue; Data Comm Write
31	Undefined	42	Data Comm Wait
32	COBOL abnormal end	43	Undefined
33	Sort End-of-Job	44	Program call
34	Undefined	45	Stack size change

Program-Dependent Interrupts

Program-dependent interrupts are generated in response to conditions detected by the interpreter of the program, but need not be explicitly issued by the object program. The interpreter builds the communicate message and places it into the communicate message pointer field of the Run Structure Nucleus and then returns control to the MCP. Table 1-4 describes the types of program-dependent interrupts (RS.ITYPE=00). The format of the communicate message is as follows:

	RS.COMMUNICATE.MSG.PTR			
Description:	Interrupt Identifier	Type	Not Used	Address (If Required)
Field Length, in Bits:	2	6	16	24
Identifier:	RS.ITYPE	RS.INMBR	RS.ILENGTH	RS.IADDRESS

Table 1-4. Program-Dependent Interrupts

RS.ITYPE	Definition	RS.ITYPE	Definition
0	Undefined	19	Exponent overflow
1	Evaluation/Program Pointer stack overflow	20	Exponent underflow
2	Control stack overflow	21	Expression out of range
3	Name/Value stack overflow	22	Superfluous exit
4	Remap size error	23	Out of memory space
5	Invalid parameter	24	Invalid link
6	Invalid substring	25	Type error
7	Invalid subscript	26	Integer overflow
8	Invalid return	27	Message transfer data not present
9	Invlaid case	28	Message transfer invalid data template
10	Divide by zero	29-56	Not used
11	Invalid index	57	Sizechange cleanup
12	Read out-of-bounds, memory parity	58	Interpreter segment not present
13	Invalid operator	59	Hi-priority Reader/Sorter complete
14	Invalid parameter to value descriptor	60	Put in ready queue (response to timer overflow or I/O Interrupt)
15	Convert error		
16	Stack overflow	61	Trace print
17	Uninitialized data item	62	Code segment not present
18	Write out-of-bounds	63	Data segment not present

NOTE

On types 58, 62, and 63, the address of the system descriptor causing the "not present" interrupt is in the address part of the communicate message.

All program-dependent interrupt types less than 57 cause the program involved to discontinue processing. This action is relayed by the MCP to the console printer, with a message stating the problem, the current value of the next instruction pointer (NIP), and termination information. If the TERM option is set, the MCP automatically terminates the program without operator intervention.

External Interrupts

External interrupts are those not having any connection with a particular program or job being processed at the time the interrupt occurs. Below are those interrupts that can be classified as external.

a. Timer.

b. I/O service request.

c. Memory parity error.

d. Memory out-of-bounds condition.

e. I/O interrupt. (Different from item b, above, because of apparent violation of base and limit registers.)

Each of the above conditions generates an interrupt bit and is recognized by the examination of the appropriate bits in the CC or CD registers. (Refer to the B 1700 System Reference Manual, Form No. 1057155.) In addition to the individual interrupt bits developed, the XYST register has a bit referenced as the INT (INTERRUPT OR), which indicates that one or more of the interrupt conditions are true. The CSM first tests the INT bit of the XYST register; if INT is true, the CSM then examines other bits to determine specifically what caused the interrupt. If the CSM finds the INT bit false, no further checking is needed.

Interrupt and Communicate Servicing

The handling of an interrupt is accomplished by the outermost loop of the MCP, which gains control and performs the following functions:

a. Handles unprocessed I/O interrupts via the NUTHIN.TDO and IOCOMPLETE procedures.

b. Updates the system clock.

c. Checks the ACTIVE SCHEDULE for programs waiting to be executed.

d. Empties the Communicate queue by calling the program handler (IH procedure).

e. Reinstates programs in the Ready queue.

MCP Non-Processing Mode

The MCP remains in its "outer loop" while waiting for processing. Basically the outer loop performs the following:

a. If the system clock is greater than the next timer overflow interval, N.SECOND is called.

b. The MCP then enters a routine called NUTHIN.TDO and passes it a zero, causing any interrupts in the Interrupt queue to be processed by the IOCOMPLETE routine.

c. The MCP then checks for an entry in the active schedule; if there is, the FIREPROG routine attempts to execute the program.

d. If there is an entry in the Communicate queue, the MCP processes the communicate (IH procedure).

e. If there is a program in the Ready queue, the MCP attempts to reinstate the program. When the routine, M.REINSTATE, relinquishes control, the MCP processes the communicate (in the IH procedure) caused by M.REINSTATE when it relinquished control.

f. The above functions perform in a continuous manner, forming the "outer loop."

SYSTEM INITIALIZATION

Before processing can begin, the system must be initialized. This consists of loading the Master Control Program, Central Service Module, and the SDL Interpreter. A Clear/Start then readies the system for operation.

Procedures for COLDSTART

COLDSTART is the name of the routine used to initialize a B 1700 system. There are four basic steps to COLDSTART:

a. Constructing and initializing the Disk Directory and available tables on the system disk.

b. Loading the MCP to the system disk.

c. Loading the SDL Interpreter and the CSM for the B 1710 and B 1720 systems, as well as the programs SYSTEM/INIT, SYSTEM/LOAD.DUMP, FILE/LOADER, and SYSTEM/MEM.DUMP.

d. Displaying a message to the operator to perform a Clear/Start. COLDSTART creates a table that contains the variables needed by Clear/Start for processing control.

Table 1-5 describes the COLDSTART variable table used by the MCP.

Table 1-5. COLDSTART Variables

Field Name	Type	Length	Description
01 COLD.START.VARIABLES	Bit	664	
02 CLEAR.START.FLAGS			The following flags are used at Clear/Start to initialize the system. The flags are pointers to the files needed, and being used, during the initialization.
03 CS.TRACT	Bit	4	CS.INTERP, CS.MCP, CS.GISMO, and CS.INIT are local indices set by Clear/Start to indicate which file within a particular category was selected.
03 CS.INTERP	Bit	4	
03 CS.MCP	Bit	4	
03 CS.GISMO (CSM)	Bit	4	
03 CS.INIT	Bit	4	On system disk resides a directory from which Clear/Start selects the files needed to initialize the system. The directory format is an array of 40 36-bit disk addresses (one segment) followed by 36 30-character name entries (six segments). Following this directory are the files referenced by the above pointers. Following are the files referenced by CS.INIT.

Table 1-5. COLDSTART Variables (Cont)

Field Name	Type	Length	Description
			Code File 0 = Standard system initializer 1 = Entry system (MCPI) initializer 2 = Experimental system initializer 3 = B 1710 series CSM 4 = B 1720 series CSM 5 = 6 = Entry system (MCPI) 7 = B 1710 series trace CSM 8 = B 1720 series trace CSM 9 = 10 = Entry system (MCPI) trace CSM 11 = Experimental CSM 12 = B 1710 series MCP interpreter 13 = B 1720 series MCP interpreter 14 = Entry system (MCPI) interpreter 15 = B 1710 series MCP trace interpreter 16 = B 1720 series MCP trace interpreter 17 = Entry system (MCPI) trace interpreter 18 = Experimental MCP interpreter 19 = Standard MCP

Table 1-5. COLDSTART Variables (Cont)

Field Name	Type	Length	Description
			20 = Entry MCP (MCPI) 21 = Trace MCP 22 = Entry trace MCP (MCPI) 23 = Experimental MCP 24 = Stand-alone memory dump 25 = Stand-alone entry (MCPI) memory dump 26 = Stand-alone disk dump 27 = Stand-alone SDL program 28 = Stand-alone I/O debug 29 = Loader for stand-alone SDL program 30 = Experimental stand-alone program (MIL)
02 NAME.TABLE	Bit	36	Contains the beginning disk address of the files listed under CS.INIT.
02 INTERP.DIC.ENTRIES	Bit	24	Contains the number of entries in the Interpreter Dictionary.
02 CS.SIZE	Bit	24	Contains the length of the COLDSTART variable table.
02 DUMP.FILE	Bit	24	Points to the disk address containing a memory dump (SYSTEM/DUMPFILE).
02 PAGE.SIZE	Bit	24	Contains a page size in Control Memory. This is referenced when interpreters are being swapped in and out of Control Memory.
02 MPF.TABLE	Bit	36	Contains a disk address pointing to the first entry in the multiple pack file table. When more than one multiple pack file is present, forward/backward links connect the additional multiple pack file table entries.

Table 1-5. COLDSTART Variables (Cont)

Field Name	Type	Length	Description
02 LOG.MIX.INFO	Bit	36	References the time, number of disk accesses, the log address, and mix number of each job on the system. For a Clear/Start the MCP uses this data to update the log.
02 DISK.AVAIL	Bit	36	Contains the disk address of the Working Available table.
02 DISK/DIRECTORY	Bit	36	Contains the disk address of the Disk Directory.
02 TEMP.TABLE	Bit	36	Contains the disk address of the Temporary Available table.
02 SYSTEM.DRIVES	Bit	16	Contains 16 one-bit fields identifying the system disk drives present on the system. There is a maximum of 16 system disk drives.
03 SYSTEM.DRIVE	Bit	1	Elementary field of the above structure.
02 AVL.TABLE.DISP	Bit	64	This field contains 16 four-bit fields identifying the 10-segment portion of the Temporary Available table assigned to each system drive on the system. For example: If drive 0 (DPA) and drive 3 (DPD) are system drives, the SYSTEM. DRIVES bit configuration would be 1001000000000000. The AVL.TABLE.DISP configuration would be 0001000000000000. In this example, DPD would be the second Temporary Available table constructed for the system drive.
02 SY.DAY	Bit	5	Contains the current day.
02 SY.MONTH	Bit	4	Contains the current month.
02 SY.YEAR	Bit	7	Contains the current year.
02 SY.JDAY	Bit	9	Contains the current day (in Julian date format).
02 SY.TIME	Bit	21	Contains the time in hours, minutes, seconds, and tenths of seconds.

Table 1-5. COLDSTART Variables (Cont)

Field Name	Type	Length	Description
03 SY.HOUR	Bit	5	Contains the hour: 0-23.
03 SY.MIN	Bit	6	Contains the minute: 0-59.
03 SY.SEC	Bit	6	Contains the second: 0-59.
03 SY.10THSEC	Bit	4	Contains the tenth of second: 0-9.
02 SY.12HOUR	Bit	5	Contains the hour: 0-12.
02 SY.DAYNAME	Char.	9	Contains the day of the week.
02 SY.MERIDIAN	Char.	2	Contains AM or PM.
02 SYSTEM.OPTIONS	Bit	80	The following options contained under SYSTEM.OPTIONS, Lexic Level 03, (except those listed below) are available and remain in a reset condition until set. Those which are preset at COLDSTART are as follows: TIME DATE BOJ EOJ TERM DUMP
03 LOG.OPTION	Bit	1	Provides the space on the system disk for the log.
03 CHARGE.OPTION	Bit	1	Requires that all programs scheduled for execution have a CHARGE number.
03 LIB.OPTION	Bit	1	Causes the MCP to display the file-identifier for any resultant action performed on the file.
03 OPEN.OPTION	Bit	1	Causes the message, "file-identifier OPENED," to be displayed at each file open.
03 TERM.OPTION	Bit	1	Instructs the MCP to automatically discontinue (DS) processing a program when an irrecoverable error has occurred. (This option eliminates the possibility of obtaining a memory dump.)

Table 1-5. COLDSTART Variables (Cont)

Field Name	Type	Length	Description
03 TIME.OPTION	Bit	1	Causes the message "TR PLEASE" to be displayed during a Clear/ Start. The operator is required to enter the time before processing can begin.
03 DATE.OPTION	Bit	1	Causes the message "DR PLEASE" to be displayed during a Clear/ Start. The operator is required to enter the date before processing can begin.
03 CLOSE.OPTION	Bit	1	Causes the message, "file-identifier CLOSED," to be displayed at each file close.
03 PBT.OPTION	Bit	1	Causes an output file assigned to a printer or card punch to be diverted to tape backup if the designated output device is not available and providing the program allows tape backup.
03 PBD.OPTION	Bit	1	Causes file backup to go to disk under the same parameters as the PBT.OPTION.
03 BOJ.OPTION	Bit	1	Causes a BOJ message to be displayed when a program begins execution.
03 EOJ.OPTION	Bit	1	Causes an EOJ message to be displayed when a program reaches EOJ.
03 SCHM.OPTION	Bit	1	Causes the MCP to display a message when a program is placed in the waiting schedule.
03 LAB.OPTION	Bit	1	Causes the tape label to be displayed each time a BOT (beginning-of-tape mark) is read.
03 RMOV.OPTION	Bit	1	Automatically causes the removal of the "old file" in a duplicate file situation. This is similar to the "RM" console message.
03 DUMP.OPTION	Bit	1	Creates a system dump file. If the option is reset, the SYSTEM/ DUMPFILE is removed from the Disk Directory.

Table 1-5. COLDSTART Variables (Cont)

Field Name	Type	Length	Description
03 ZIP.OPTION	Bit	1	Displays on the console printer all ZIP statements that are communicated to the MCP.
03 MEM.OPTION	Bit	1	Causes the MCP to display a message whenever there is no available memory.
03 SWO1.OPTION	Bit	1	Used for systems software debugging.
03 SWO2.OPTION	Bit	1	Used for systems software debugging.
03 SWO3.OPTION	Bit	1	Used for systems software debugging.

Disk COLDSTART Variables

Because of the constant demand for memory space, the infrequently used portions of the COLDSTART variables are stored on the systems disk and are read into memory as needed. Table 1-6 contains a list of these variables.

Table 1-6. Disk COLDSTART Variables

Field Name	Type	Length	Description
01 DISK.CS.VARIABLES	Bit	420	
02 MASTER.IOAT	Bit	36	Contains the disk address of the I/O Assignment Table.
02 MASTER.DISK.AVAIL	Bit	36	Contains the disk address of the Master Available Table.
02 NEXT.LOG.RECORD	Bit	36	Contains the disk address of the next available log record.
02 LOG.SIZE	Bit	24	Contains the remaining size of the current area of the log.
02 NEXT.ELOG	Bit	36	Contains the disk address of the Elog (engineering log).
02 ELOG.SIZE	Bit	24	Contains the size remaining in the current area of the Elog.
02 JOB.NO	Bit	24	Contains the job number that is to be assigned to the next program executed.

Table 1-6. Disk COLDSTART Variables (Cont)

Field Name	Type	Length	Description
02 PBD.NO	Bit	24	Contains the next number to be assigned to a printer or punch backup file.
02 DUMP.NO	Bit	24	Contains the next number to be assigned to a dump file.
02 CTLDCK.NO	Bit	24	Contains the next number to be assigned to a pseudo file.
02 LOG.NO	Bit	24	Contains the next number to be assigned to a log file.
02 Q.DISK	Bit	36	Contains the disk address of work areas used by the MCP; for example, program rollouts, disk log, and temporary FPBs.
02 TRACE.FPB	Bit	36	Contains the disk address of the FPB trace skeleton.
02 CTLDCK.DIRECTORY	Bit	36	Contains the disk address of the pseudo reader directory.
02 PBD.BLCKS.AREA	Bit	24	Contains the blocking factor for disk backup files.
02 LOG.LAST.AREA	Bit	1	Indicates the last area of the log is being used.

Clear/Start

The Clear/Start program structures memory and prepares the system for program execution. A Clear/Start is performed for the following conditions:

a. System power-ups.

b. Unscheduled halts.

c. Apparent MCP or Interpreter loops.

d. Operating environment changes.

The Clear/Start procedure performs the following functions:

a. Writes correct parity throughout memory.

b. Loads selected operating environments from the Name Table. (To be discussed later in this section.)

c. Creates a "run structure" with which the MCP may operate.

d. Passes control to the MCP.

Clear/Start can be performed on any system and with either MCP I or MCP II.

Name Table

The Name Table, built during COLDSTART, resides on disk and identifies systems software/firmware used in the operational environment of the system. The Name Table allows recovery from an experimental mode of operation to the standard mode. The COLDSTART procedure loads and identifies the required systems software/firmware to begin operations on whatever hardware is available. After a Clear/Start, a systems pack may be moved from one system to another, requiring only another Clear/Start to begin processing. COLDSTART loads the following systems software/firmware:

a. Standard MCP.

b. SDL Interpreter for the B 1710 series.

c. SDL Interpreter for the B 1720 series.

d. CSM for the B 1710 series.

e. CSM for the B 1720 series.

f. System Initializer.

g. SYSTEM/LOAD.DUMP

h. FILE/LOADER

i. SYSTEM/MEM.DUMP

A Clear/Start is required to effect any environment change. The *changed* systems software/firmware becomes the new basis for operation, and remains in effect until changed. However, limited environmental switching can be done on a temporary basis during the Clear/Start procedure. Table 1-7 contains the Name Table entries.

Table 1-7. Systems Software/Firmware

Name Table Entry Number	System Mnemonic	Description
0	N	Standard System Initializer
1	NE	Entry System Initializer.
2	NX	Experimental System Initializer.
3	G1	B 1710 Central Service Module.
4	G2	B 1720 Central Service Module.
5	G3	Reserved.
6	GE	Entry Central Service Module.

Table 1-7. Systems Software/Firmware (Cont)

Name Table Entry Number	System Mnemonic	Description
7	G1T	B 1710 Trace Central Service Module.
8	G2T	B 1720 Trace Central Service Module.
9	G3T	Reserved.
10	GET	Entry Trace Central Service Module.
11	GX	Experimental Central Service Module.
12	I1	B 1710 MCP Interpreter.
13	I2	B 1720 MCP Interpreter.
14	IE	Entry MCP Interpreter.
15	IIT	B 1710 MCP Trace Interpreter.
16	I2T	B 1720 MCP Trace Interpreter.
17	IET	Entry MCP Trace Interpreter.
18	IX	Experimental MCP Interpreter.
19	M	Standard MCP II.
20	ME	Entry MCP (MCPI).
21	MT	Trace MCP.
22	MET	Entry Trace MCP.
23	MX	Experimental MCP.
24	SD	Stand-Alone Memory Dump (required for any system dumps).
25	SDE	Stand-Alone Entry Memory Dump.
26	SDD	Stand-Alone Disk Dump.
27	SDL	Stand-Alone SDL Program.
28	SIO	Stand-Alone I/O Debug.
29	SL	Loader for Stand-Alone SDL Program (needed to load the SDL interpreter).
30	SX	Stand-Alone Program (MIL).

SECTION 2
MCP - OBJECT PROGRAM COMMUNICATION

INTRODUCTION

The B 1700 system, whether operating serially (MCPI) or in a multiprogramming environment (MCPII), processes object programs generated by B 1700 compilers. This section discusses object programs, compilers, code file (object code) structures, and object code execution.

COMPILATION

A compiler is a special-purpose computer program which accepts source statements in a language for which the compiler was written, and translates those source statements into object code to be stored on disk either temporarily or permanently. The compilation process requires specific functions to be performed by the compiler and the Master Control Program. Certain control information about the program is stored on disk with the program object code.

Compilation Types

The COMPILE control statement designates the type of compilation to be performed:

a. COMPILE . . . (and GO)

b. COMPILE . . . [TO] LIBRARY

c. COMPILE . . . SAVE

d. COMPILE . . . [FOR] SYNTAX

Refer to the COMPILE statement in the B 1700 System Software Operational Guide, Form No. 1068731, for additional information concerning the COMPILE statement.

MCP-Compiler Recognition

The COMPILE statement is a request to the MCP to schedule a particular program having special parameters for execution. The COMPILE statement is scanned for the type of compilation to be performed.

OBJECT PROGRAMS

Compiler output is an object program. Object programs are stored on disk, either temporarily or permanently, in executable format. Object program structure consists of various components, some of which are optional depending upon requirements of each language. The following terms pertain to object programs:

Term	Description
Object program	Output of a compiler containing a Program Parameter Block, File Parameter Blocks, and object code.
Object code	The executable code in an object program.
S-code	The object code that is interpreted by an S-machine (discussed later in this section).
Object program file	A disk copy of the object program.

An object program file may have one or more of the following components:

a. Program Parameter Block (required).

b. Scratchpad area (required).

c. File Parameter Blocks.

d. S-code (required).

e. Data Segment Dictionary.

f. Code Segment Dictionary (required).

g. Data Segments.

The Program Parameter Block (PPB) and the scratchpad area (an MCP/object program work area) reside in the first two disk segments of all code files, as illustrated in figure 2-1.

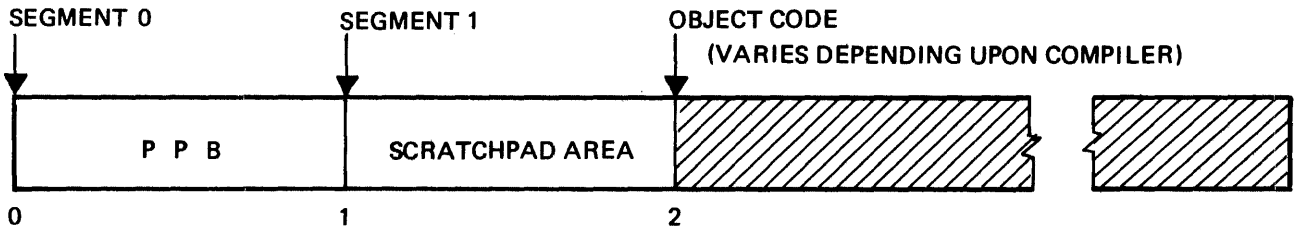


Figure 2-1. MCP/Object Program Work Area Example

Program Parameter Block (PPB)

The first disk segment (relative zero) of all code files generated by a compiler is the Program Parameter Block (PPB). It contains information and reference addresses needed in the processing of an object program, and is classified into two parts: general information, and log information.

General information is supplied by the compiler during compilation and contains the data required by the MCP to schedule a program for execution. At execution time, the MCP expands the PPB from one segment to three, supplying additional information needed for execution, for the system log, and for run-time control. Table 2-1 describes the Program Parameter Block format.

Table 2-1. Program Parameter Block

Field Name	Type	Length	Description
01 PROGRAM.PARAMETER.BLOCK	Bit	2332	
02 PROGRAM.NAME	Char	30	Contains the program-identifier.
03 PROG.CURRENT.DIRECTORY	Char	10	Contains the program pack-identifier.
03 PROG.NAME.FIRST	Char	10	Contains the program family-name.
03 PROG.NAME.SECOND	Char	10	Contains the program file-identifier. NOTE For a compilation, the log copy of the PPB contains the program name of the compiler.
02 PROG.INTRINSIC	Char	20	Contains the intrinsic file-identifier.
03 PROG.INTRINSIC.DIRECTORY	Char	10	Contains the intrinsic pack-identifier.
03 PROG.INTRINSIC.NAME	Char	10	Contains the intrinsic family-identifier.
02 PROG.INTERP.NAME	Char	30	Contains the interpreter identifier.
03 PROG.INTERP.DIRECTORY	Char	10	Contains the interpreter pack-identifier.
03 PROG.INTERP.NAME.FIRST	Char	10	Contains the interpreter family-name.
03 PROG.INTERP.NAME.SECOND	Char	10	Contains the interpreter file-identifier.
02 PROG.PRIORITY	Bit	4	Contains the priority of the program. (Compiler default = 4.)
02 PROG.BEGINNING	Bit	32	Contains the first instruction pointer.
02 PROG.STATIC.CORE	Bit	24	During compilation the compiler calculates the amount of memory to be allocated that immediately follows the base register of the program. During execution if there are data segments present, this field is the same size that is assigned in the first entry of the Data Dictionary when the program was compiled. If a disk address is

Table 2-1. Program Parameter Block (Cont)

Field Name	Type	Length	Description
			present in the dictionary entry, the memory allocated starts from that disk address.
02 PROG.DYNAMIC.CORE	Bit	24	Contains the size of dynamic memory.
02 PROG.TOTAL.CORE	Bit	24	Contains the smallest amount of memory required to execute the program. PROG.TOTAL.CORE consists of the following components: PROG.STATIC.CORE, plus PROG.DYNAMIC,CORE, plus DATA DICTIONARY SIZE, plus the FIB DICTIONARY SIZE.
02 PROG.BIGGEST.SEG	Bit	24	Contains the size of the largest code segment.
02 PROG.DATA.DIC	Bit	64	Contains the Data Dictionary descriptor.
02 PROG.SEG.DIC	Bit	64	Contains the Segment Dictionary descriptor. NOTE A normal descriptor contains the position, length, and the type of data that is referenced.
02 PROG.FPB.ADDRESS	Bit	24	Contains the relative disk address of the first FPB in the program file (relative to the base of the code file). FPBs are contiguous.
02 PROG.FILES	Bit	8	Contains the total number of files in program. The maximum is 225.
02 PROG.CHARGE.NUMBER	Bit	24	Contains the user-assigned charge number of the program.
02 PROG.ONLY.SEG	Bit	10	Reserved for the SDL overlay handler.
02 PROG.FREEZER	Bit	1	Prohibits the program to be rolled-out of memory.
02 PROG.LINKS	Bit	1	Indicates that memory links are needed in the dynamic memory area.

Table 2-1. Program Parameter Block (Cont)

Field Name	Type	Length	Description
02 PROG.TRACE	Bit	8	Enable a trace to begin with the first executable statement.
02 PROG.SCHED.PRIORITY	Bit	4	Contains the program scheduling priority.
02 PROG.VIRTUAL.DISK	Bit	24	Contains the number of disk segments for data overlays. If the field contains zeros and data overlays are required, 1000 disk segments are assigned.
02 PROG.IPB	Bit	24	Contains the relative file location of the interpreter parameter block.
02 FILLER	Bit	92	
NOTE The entries above this point are compiler-generated; those entries below this point are MCP-generated.			
02 PROG.PROG.PTR	Bit	36	Contains the absolute disk address of the PPB.
02 PROG.EXECUTE.TYPE	Bit	4	The execution type codes consist of the following: 1 = Execute 2 = Compile and Go 3 = Compile for Syntax 4 = Compile to Library 5 = Compile and Save 6 = "Go" part of Compile and Go 7 = "Go" part of Compile and Save
02 PROG.EOJ.TYPE	Bit	4	Contains the condition of program termination. 0 = Normal end-of-job 1 = DS or DP 2 = Error condition in program 3 = Aborted

Table 2-1. Program Parameter Block (Cont)

Field Name	Type	Length	Description
02 PROG.GENERATOR.NAME	Char	30	Contains the compiler-identifier.
03 PROG.GENERATOR. DIRECTORY	Char	10	Contains the compiler pack-identifier.
03 PROG.GENERATOR.NAME. FIRST	Char	10	Contains the family-name.
03 PROG.GENERATOR.NAME. SECOND	Char	10	Contains the compiler file-identifier.
02 PROG.DATE.COMPILED	Bit	36	Contains the date and time program was compiled.
02 PROG.SCHED.LINK	Bit	36	Contains the disk address of the next item in the schedule.
02 PROG.SCHED.PR.COPY	Bit	4	Contains the scheduled priority.
02 PROG.SCHED.SIZE	Bit	24	Contains the total amount of memory required for the scheduled job.
02 PROG.JOB.NUMBER	Bit	24	Contains the job number of this program.
02 PROG.FLAGS	Bit	4	Contains MCP internal control flags.
03 FILLER	Bit	1	Reserved.
03 PROG.DONT.REENTER	Bit	1	Indicates that the code segment dictionary is non-standard.
03 PROG.SORT	Bit	1	Indicates this is a sort program.
03 PROG.WAIT.OPERATOR	Bit	1	Indicates this program is waiting for response to an "FS" or "RS" message.
02 PROG.EX.AFTER.NAME	Char	30	Contains points that link conditionally executed programs such as EXECUTE.THEN or EXECUTE.AFTER to their predecessor.
03 PROG.EX.AFTER.DIRECTORY	Char	10	Contains the pack-identifier for the EXECUTE.AFTER program.
03 PROG.EX.AFTER.NAME FIRST	Char	10	Contains the family-name for the EXECUTE.AFTER program.
03 PROG.EX.AFTER. NAME SECOND	Char	10	Contains the file-identifier for the EXECUTE.AFTER program.

Table 2-1. Program Parameter Block (Cont)

Field Name	Type	Length	Description
02 PROG.SORT.DATA	Bit	36	Contains the disk address of the sort parameters.
02 PROG.SORT.SPAD	Bit	36	Contains disk address of the sort area.
02 PROG.MY.MIX	Bit	24	Contains the mix-index number of the program.
02 PROG.HIERARCHY	Bit	8	Contains nesting Lexic Level for the program procedure being used.
02 PROG.PRIOR.MIX	Bit	24	Contains the calling program mix-number.
02 PROG.SCHED.DATE	Bit	36	Contains the date and time the program is scheduled for execution.
02 PROG.BOJ.DATE	Bit	36	Contains the date and time of the program BOJ.
02 PROG.EOJ.DATE	Bit	36	Contains the date and time of the program EOJ.
02 PROG.PROCESS.TIME	Bit	24	Contains the total processor time of the program.
02 PROG.OBJ.NAME	Char	30	Indicates the compilation is completed.
03 PROG.OBJ.DIRECTORY	Char	10	Contains the object program pack-identifier.
03 PROG.OBJ.NAME.FIRST	Char	10	Contains the object program family-name.
03 PROG.OBJ.NAME.SECOND	Char	10	Contains the object program file-identifier.
02 PROG.PSEUDO.READER	Bit	24	Contains the beginning address of the pseudo reader table.
02 PROG.PORT.CHAN	Bit	7	Contains the port and channel of the disk unit on which the object program resides.

Scratchpad

The scratchpad is an area generated by the compiler that is utilized by the interpreters during program execution. The initial scratchpad resides in the second disk segment of all code files, immediately following the Program Parameter Block (PPB). It is the responsibility of the compiler to generate the initial scratchpad settings. The contents of the scratchpad are dependent on the S-language represented by the S-machine state during processing. When a program is first permitted to execute, the scratchpad in the

processor is loaded from the scratchpad image of the object program on disk. When a program relinquishes control, the interpreter stores the scratchpad settings and restores them when the program is reinstated. Figure 2-2 contains a skeleton scratchpad format.

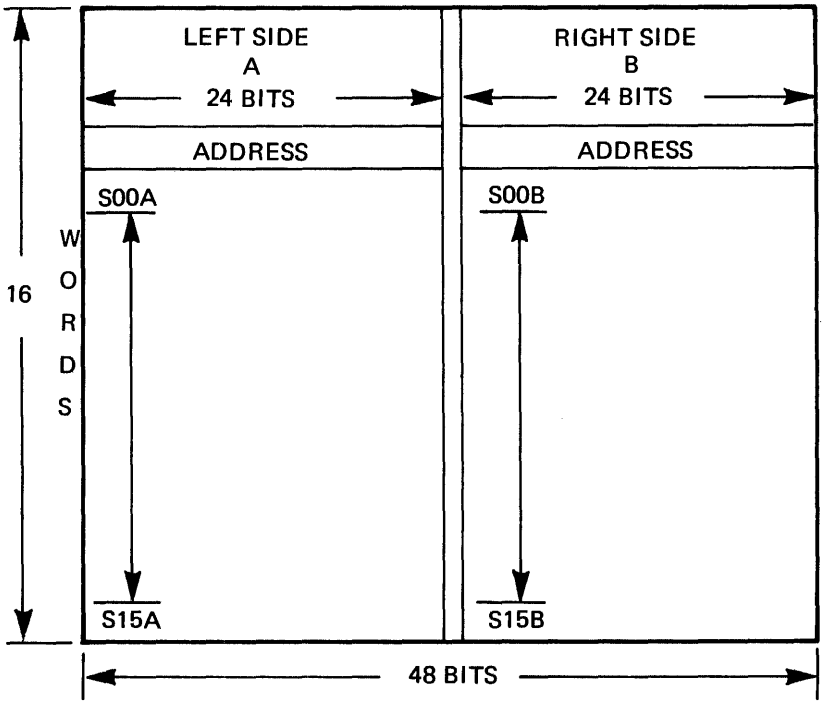


Figure 2-2. Skeleton Scratchpad

Scratchpad, as addressed by an interpreter, may be referenced as sixteen 48-bit words, or thirty-two 24-bit words. Refer to the B 1700 System Reference Manual, Form No. 1057155, for a detailed explanation of the scratchpad and its register-related functions.

The initial scratchpad settings and the storing and restoring of the processor state of a program are the two major functions of the interpreter concerning scratchpad. Since the MCP is also a code file, the only difference between its scratchpad and that of an object program is the internal settings; the format and size are the same.

The MCP at program beginning-of-job copies the initial scratchpad settings from disk into the field RS.M.MACHINE contained in the Run Structure Nucleus of the program. When the program receives control and begins processing, the interpreter loads portions of the RS.M.MACHINE into the processor scratchpad. When the program relinquishes control, the interpreter stores the appropriate scratchpad settings back into RS.M.MACHINE. The interpreter reloads the scratchpad settings the next time the program gains control.

The S-machine is the processor structure of an object program. The instructions of the object program while in the processor are called S-operators or S-ops. The combined set of these S-operators is called the S-code. The interpreter fetches these S-operators, determines the operation to be performed, and executes a series of microinstructions to achieve the desired result. The various S-machines, being interpreted by sets of microinstructions, enable the B 1700 to execute different language structures in the processor at the same time.

Figure 2-3 shows the operational flow of a program and its S-machine, which consists of the CSM/MCP interface, data and associated addressing, and the S-code with its associated fetch routine.

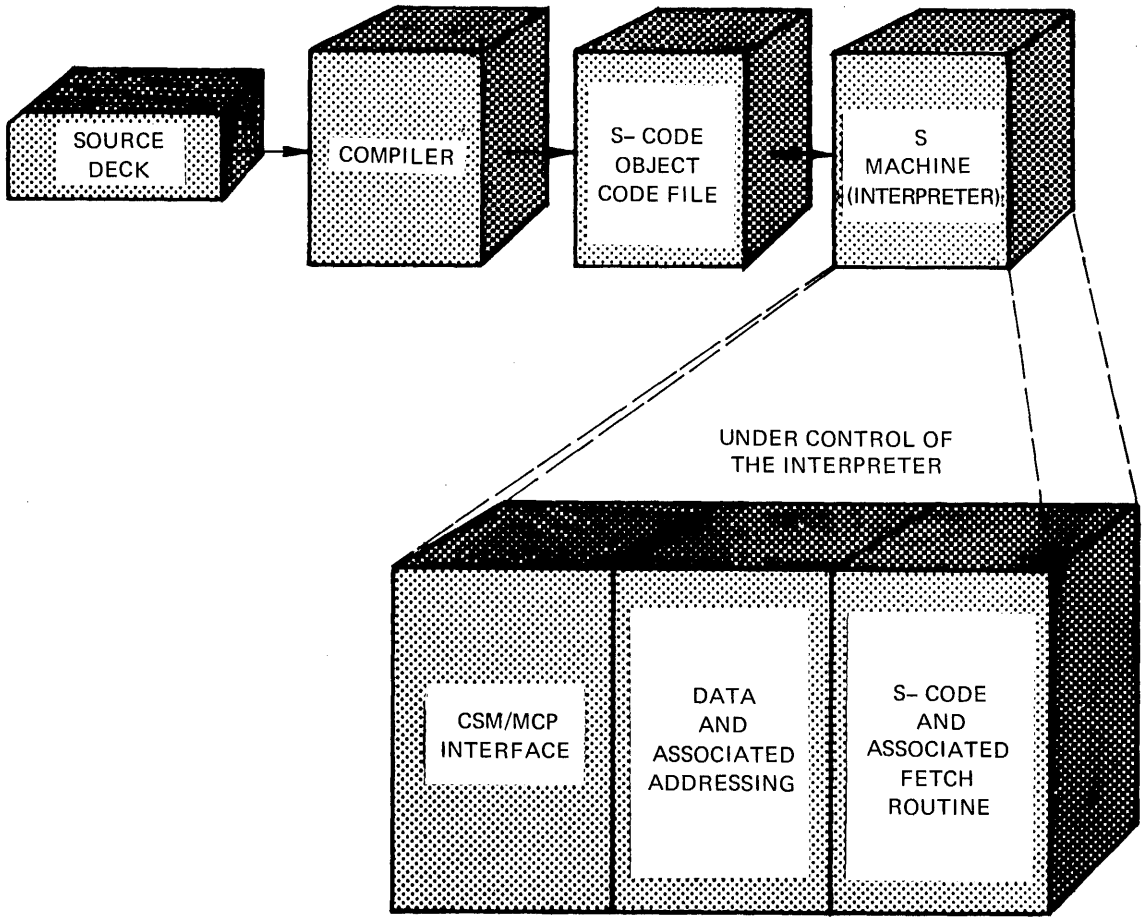


Figure 2-3. Stages of Program Flow on the B 1700 and S-Machine

S-Code

The compiler, by scanning a program's source language statements, generates object code. Compilers formulate the object code into logical divisions, called segments, for referencing by the MCP during execution. This can be done either at the programmer's discretion or by criteria within the compiler itself. In SDL, the segment dictionaries themselves are segmented (called pages).

The object code is addressed by page number, segment number, and displacement value. The page and segment numbers are indices into the Code Segment Dictionary (explained below). Displacement is a pointer to the code relative to the base of the segment. The base address is contained in the dictionary descriptor.

Code Segment Dictionary

The compiler builds a Code Segment Dictionary for each object program that references the addresses and lengths of each code segment. There is an entry in the dictionary for each code segment.

The Code Segment Dictionary provides the MCP with the address and location, whether in memory or on disk, of each code segment used by a program. The program references its code symbolically by the page, segment, and displacement value. The MCP, through a field in the Run Structure Nucleus, locates the Code Segment Dictionary, consisting of system descriptors (see table 2-2). If the code segment desired is in memory, it is at the address stored in SY.CORE. If the code segment is on disk, SY.ADDRESS contains the disk address. The length of the code segment is given in SY.LENGTH.

Table 2-2. System Descriptor

Field Name	Type	Length	Description
01 SYSTEM.DESCRIPTOR			
02 SY.MEDIA	Bit	2	Indicates whether the data being referenced is on disk or in memory.
02 SY.LOCK	Bit	1	Not implemented.
02 SY.IN.PROCESS	Bit	1	Indicates whether there is an I/O in process for the information represented by this descriptor. If so, the field, SY.CORE, contains a reference pointer to the I/O descriptor.
02 SY.INITIAL	Bit	1	Indicates that the field, SY.ADDRESS, is a disk address and is read-only data. If the operation is a write, then the MCP finds disk space and replaces the address in SY.ADDRESS.
02 SY.FILE	Bit	1	Indicates that this descriptor references a file whose user count is to be decremented when this descriptor is processed.
02 FILLER	Bit	10	Reserved.
02 SY.TYPE	Bit	4	Contains a type code indicating the kind of data type represented by this descriptor. Type — Description 0 — Bit 1 — Digit 2 — Character 3 — Normal descriptor 4 — Disk segment 5 — System descriptor 6 — System intrinsic
02 SY.ADDRESS	Bit	36	Contains the address whether on disk or in memory of the information referenced by this descriptor.

Table 2-2. System Descriptor (Cont)

Field Name	Type	Length	Description
03 FILLER	Bit	12	Contains the port, channel, and unit number if this is a disk address.
03 SY.CORE	Bit	24	Contains the address of the data (memory or disk).
02 SY.LENGTH	Bit	24	Contains the number of units or items referenced by this descriptor as determined by the field, SY.TYPE.

Data Dictionary

The Data Dictionary of an object program references the address and length of the data described in the data definitions of the program. If the data is not segmented, the Data Dictionary has only one entry; otherwise, there is an entry for each data segment declared by the programmer, as is the case with COBOL.

A program may need to process more data than is possible to have present in memory for the lifetime of the run. The Data Dictionary gives the programmer, or in some cases the compiler, the ability to group data into "blocks," some of which can be in memory or on disk at any point during the execution of the program. When a program references a data item and it is not present in memory, the MCP brings into memory the data segment of that item. This operation can require the writing out of other data segments in order to make memory space available. The field PROG.DYNAMIC.CORE in the PPB specifies the size of the memory area to be reserved for program data segments. All data segments of the program compete for this space.

The first entry in the Data Dictionary describes the static memory space of the program. If the length field of this entry is non-zero, the MCP initializes the static memory from that entry at BOJ.

The COBOL compiler, for example, initializes WORKING STORAGE data as the program is compiled, storing the initialized data in the object code file on disk. When initialized data is stored on disk, the relative disk address of the data segment is stored in the directory entry that describes the data. The initialize bit, SY.INITIAL of the dictionary entry, is set to 1 at BOJ, indicating to the MCP that it must obtain a new disk area to contain the data segment the first time it is overlaid.

File Parameter Block (FPB)

Compilers build a File Parameter Block (FPB) for each file declared in an object program. The length of each FPB is one disk segment. The FPB defines the file and its characteristics. At BOJ, the MCP stores both the File Parameter Block and the Program Parameter Block in the log area on disk (regardless of the LOG option setting) for reference during execution. Table 2-3 describes the FPB format.

NOTE

The disk space used to store the PPB and the FPBs is returned at EOJ if the LOG option has not been set.

Table 2-3. File Parameter Block

Field Name	Type	Length	Description
01 FPB	Bit	1088	
02 FPB.FILE.NAME	Char	10	Contains the internal file name.
02 FPB.NAMES	Char	30	Contains the external file name.
03 FPB.PACK.ID	Char	10	Contains the disk pack-identifier.
03 FPB.MULTI.FILE.ID	Char	10	Contains the family-name.
03 FPB.FILE.ID	Char	10	Contains the file-identifier.
02 FPB.HDWR	Bit	6	Contains the hardware-type code.
02 FPB.MODE	Bit	4	Contains the data recording mode.
03 FPB.EVEN.PARITY	Bit	1	Contains the parity: 1, even; 0, odd.
03 FPB.CODE.TYPE	Bit	3	000 = EBCDIC 010 = BCL 001 = ASCII 011 = Binary
02 FPB.BUFFERS	Bit	24	Contains the number of buffers requested.
02 FPB.BACKUP	Bit	2	Contains the type of file backup: 00 = Either tape or disk 01 = Tape only 10 = Disk only 11 = Either tape or disk.
02 FPB.BACKUP.OK	Bit	1	Indicates that file backup is permitted.
02 FPB.HDWR.OK	Bit	1	Indicates that sending to hardware permitted.
02 FPB.BOOLEANS	Bit	24	Contains control flags and data concerning this file.
03 FPB.FORMS	Bit	1	Indicates the output file requires special forms.
03 FPB.OPTIONAL	Bit	1	Indicates this is an optional file.
03 FPB.VARIABLE	Bit	1	Indicates this file contains variable-length records.
03 FPB.LOCK	Bit	1	Indicates that this file is to be "locked" at termination time.

Table 2-3. File Parameter Block (Cont)

Field Name	Type	Length	Description
03 FPB.COBOL	Bit	1	Indicates an implied OPEN not allowed.
03 FPB.EOP	Bit	1	Indicates the presence of an end-of-page routine.
03 FPB.DEFAULT	Bit	1	Indicates the MCP is to assign the block size, record size, and other default attributes for input disk or tape files according to the disk file header or tape label.
03 FPB. PSEUDO	Bit	1	Indicates that this is a pseudo reader file.
03 FPB.RMT.KEY	Bit	1	Indicates key field has been assigned for the Network Definition Language (NDL) data communications.
03 FPB.NO.LABEL	Bit	1	Overrides the file label and uses the unit name (FPB.UNIT.NAME).
03 FPB.WORK.FILE	Bit	1	Indicates the presence of a program work file. (The job-number is included in the family-name.)
02 FPB.RECORD.SIZE	Bit	24	Contains the record size in bits.
02 FPB.RECORD.PER.BLOCK	Bit	24	Contains the number of records per physical record (block).
02 FPB.MAX.BLOCK.SIZE	Bit	24	Contains the maximum physical record size, in bits, for variable length records, and is not consulted if FPB.VARIABLE is 0.
02 FPB.ADVERB	Bit	12	When this is an implied OPEN, this field contains the type of open.
02 FPB.LABEL	Bit	48	Contains the descriptor pointing to the user label area in memory.
03 FPB.LABEL.LENGTH	Bit	24	Contains the length of the label area in bits.
03 FPB.LABEL.ADDRESS	Bit	24	Contains the address of the label area.
02 FPB.LABEL.USE	Bit	32	Contains the segment and displacement addresses of the label routine.

Table 2-3. File Parameter Block (Cont)

Field Name	Type	Length	Description
02 FPB.LABEL.TYPE	Bit	4	Contains the type of file label: 0 = Burroughs Standard; 1 = unlabeled.
02 FPB.SAVE	Bit	24	Contains the file retention save factor for magnetic tape and disk.
02 FPB.REEL	Bit	24	Contains the reel number.
02 FPB.SERIAL	Bit	24	Contains the serial number.
02 FPB.USE.ROUTINE	Bit	32	Contains the segment and displacement of the first instruction in the USE routine.
02 FPB.USE.AREA	Bit	48	Contains the 24-bit length and the 24-bit address referencing the work area of the USE routine.
02 FPB.SR.STATION	Bit	4	Contains the read station identifier for the Reader/Sorter.
02 FPB.ACCESS	Bit	4	Contains the file access code: 0 = serial; 1 = random.
02 FPB.AREAS	Bit	24	Contains the maximum number of disk areas allowed for this file.
02 FPB.EU.DRIVE	Bit	4	Contains the specified electronics unit (EU) or the drive number on which the file resides when FPB.INC.EU or FPB.SPECIAL.EU is equal to 1.
02 FPB.ALL.AT.OPEN	Bit	1	Allocates all areas of file at open time.
02 FPB.CYL.BOUNDARY	Bit	1	Allocates the file areas on cylinder boundaries (disk pack or cartridge only).
02 FPB.MULTI.PACK.FILE	Bit	1	Enables the file to be multi-pack.
02 FPB.SPECIAL.EU	Bit	1	Indicates the file resides on a specified EU or drive specified by FPB.EU.DRIVE.
02 FPB.INC.EU	Bit	1	Increments the EU or drive for each data area.
02 FILLER	Bit	3	Reserved.
02 FPB.REPETITIONS	Bit	8	Contains the number of copies for a backup file.

Table 2-3. File Parameter Block (Cont)

Field Name	Type	Length	Description
02 FPB.OPEN	Bit	24	Contains the date of the last file open.
02 FPB.1ST.OPEN	Bit	24	Contains the date the file was first opened.
02 FPB.RECORD.COUNT	Bit	24	Contains the number of records currently accessed.
02 FPB.BLOCK.COUNT	Bit	24	Contains the number of blocks currently accessed.
02 FPB.NO.OPEN.AND.CLOSE	Bit	16	Contains the number of opens and closes on this file.
02 FPB.CUMULATIVE	Bit	24	Contains the total time that the file was open.
02 FPB.ERRORS	Bit	24	Contains the number of irrecoverable read/write errors.
02 FPB.MCPDATA	Bit	36	Contains the file header disk address address if FPB.MCPINTERNAL indicates the MCP has created a special internal file.
02 FPB.MCPINTERNAL	Bit	1	Indicates the MCP has created an internal file.
02 FPB.BACKUP.ALREADY	Bit	1	Indicates that this is already a backup file; therefore prohibits the backing up of the file again.
02 FPB.NEW.FORMAT	Bit	24	Contains the literal @FFFFFF@.
02 FPB.PSEUDO.PDR	Bit	24	Contains the disk address of the pseudo reader for this file.

File Information Block (FIB)

A File Information Block (FIB) is an MCP table residing in memory containing information concerning a file. There is a FIB for every file that is processed. It is created from information in the associated File Parameter Block, and is used during the processing of the file. For example, the record size and blocking factor are two of the parameters that the FIB receives from the FPB.

Other information maintained by the MCP in the FIB consists of the input/output mode and the current status of the file, as well as counters and data reference pointers. Table 2-4 describes the File Information Block format.

Table 2-4. File Information Block

Field Name	Type	Length	Description
01 FILE.INFORMATION.BLOCK	Bit	782	
02 FIB.TYPE	Bit	6	Contains the hardware unit on which which the file resides.
02 FIB.BOOLEANS	Bit	28	The following fields are used by the MCP, identifying the characteristics of the file and its various internal attributes.
03 FIB.OPEN	Bit	1	
03 FIB.INPUT	Bit	1	
03 FIB.OUTPUT	Bit	1	
03 FIB.PSEUDO	Bit	1	
03 FIB.REVERSE	Bit	1	
03 FIB.VARIABLE	Bit	1	
03 FIB.READ.LOCK	Bit	1	
03 FIB.DISK	Bit	1	
03 FIB.TAPE	Bit	1	
03 FIB.DISK.PACK	Bit	1	
03 FIB.LABELED	Bit	1	
03 FIB.CLUSTER	Bit	1	
03 FIB.BACKUP	Bit	1	
03 FIB.96	Bit	1	
03 FIB.COBOL	Bit	1	
03 FIB.EOT	Bit	1	
03 FIB.STOP.IO	Bit	1	
03 FIB.CYL.ALLOC	Bit	1	
03 FIB.MPF	Bit	1	
03 FIB.NEWFILE	Bit	1	
03 FIB.NEWAREA	Bit	1	
03 FIB.SPECIAL.EU	Bit	1	

Table 2-4. File Information Block (Cont)

Field Name	Type	Length	Description
03 FIB.INC.EU	Bit	1	
03 FIB.MCPINTERNAL	Bit	1	
03 FIB.WAITNEWAREA	Bit	1	
03 FILLER	Bit	2	
02 FIB.RETRY.COUNT	Bit	12	Contains the number of read/ write retries.
02 FIB.UNIT	Bit	12	Contains the physical location of the unit.
03 FIB.CHANNEL	Bit	7	Contains the port and channel of the unit.
03 FILLER	Bit	1	Reserved.
03 FIB.UNIT.NO	Bit	4	Contains the hardware unit number.
02 FIB.UNIT.STATUS	Bit	24	Contains the memory address of the Input/Output Available Table (IOAT) entry.
02 FIB.FPB	Bit	36	Contains the FPB disk address of this file.
02 FIB.RS	Bit	24	Contains the memory address of the limit register of the Run Structure Nucleus.
02 FIB.RECORD	Bit	48	Contains a dummy (skeleton) descriptor created at file open and used for read/writes.
03 FILLER	Bit	8	Reserved.
03 FIB.RECORD.SIZE	Bit	16	Contains the length of the next data record to be read.
03 FIB.ALPHA.SIZE	Bit	13	Contains the record size, in bits, multiplied by 8.
03 FIB.RECORD.ADDR	Bit	24	Contains the address of the next logical record to be processed in the buffer.
02 FIB.CURRENT	Bit	24	Contains the memory address of the current buffer.
02 FIB.BUFFER	Fixed	24	Contains the size of the buffer referenced in FIB.CURRENT.

Table 2-4. File Information Block (Cont)

Field Name	Type	Length	Description
02 FIB.BLOCK.SIZE	Bit	24	Contains the maximum block size of the file.
02 FIB.RECORD.COUNT	Bit	24	Contains the number of logical read/writes on the file.
02 FIB.BLOCK.COUNT	Bit	24	Contains the number of physical read/writes on the file. NOTE The MCP physically read/writes in blocks of data, not individual records. This is distinguished by the term physical read/writes, as opposed to logical read/writes.
02 FIB.HEADER	Bit	24	Contains memory address of the file header.
02 FIB.KEY	Bit	24	Contains the current key used by the MCP for accessing random files.
02 FIB.RECORDS.BLOCK	Bit	24	Contains the number of records per block of the file.
02 FIB.CLOSE.TYPE	Bit	12	Contains the type of close to be performed.
02 FIB.RECORDS.AREA	Bit	24	Contains the number of records per disk area.
02 FIB.BLOCKS.AREA	Bit	24	Contains the block address.
02 FIB.BPA.COUNT	Bit	24	Contains the number of blocks per area.
02 FIB.AREAS	Fixed	24	Contains the number of areas for the file.
02 FIB.AREA.NUMBER	Fixed	24	Contains the current area number during serial type accessing.
02 FIB.SEGS	Bit	24	Contains the number of disk segments per block.
02 FIB.EOF	Bit	24	Contains the reference address of the end-of-file (logical record).
02 FIB.BEOF	Bit	24	Contains pointer to the block end-of-file (physical record).

Table 2-4. File Information Block (Cont)

Field Name	Type	Length	Description
02 FIB.ACCESS	Bit	4	Contains the file access type: 0 = Serial 1 = Random
02 FIB.CHANNEL.INFO	Bit	24	Contains a memory address pointing to Channel table entry.
02 FIB.EU.DRIVE	Bit	4	Contains the special EU (unit) number or drive number.
02 FIB.DISK.ADDRESS	Bit	36	Contains the current disk address of the file.
03 FIB.DISK.PC	Bit	7	Contains the current port and channel of the file.
03 FILLER	Bit	1	Reserved.
03 FIB.DISK.EU	Bit	4	Contains the drive number of the file.
03 FIB.DISK.SG	Bit	24	Contains the current disk address of the file.
02 FIB.USE.ROUTINE	Bit	32	Contains the segment and displacement of the first instruction in the MICR USE.ROUTINE of the MCP.
02 FIB.USE.AREA	Bit	48	Contains the length and address of the MICR USE.ROUTINE work area of the MCP.
02 FIB.MAX.RECORDS	Bit	24	Contains the maximum number of records the file may contain.
02 FIB.PSEUDO.RDR	Bit	24	Contains the memory address of the pseudo reader for this file.

PROGRAM OPERATION

The object program provides the MCP with information supplied by the programmer and the compiler that is used to manage program execution. At BOJ, the MCP utilizes this information to provide memory space and to build a program run structure.

Run Structure

The run structure of a program consists of an area in memory, bound by a base register and limit register, a Run Structure Nucleus immediately following the limit register, and the Data and FIB Dictionaries. The program S-code (object code) can reside anywhere in memory and is not considered part of the run structure.

The area between the base and limit registers is used for program data consisting of both overlayable and non-overlayable data. For example, a COBOL program run structure contains the following information between its base and limit registers:

a. Edit table.

b. COP (current operand) table.

c. Special registers.

d. Data name monitor symbols.

e. Work area for intermediate results.

f. Overlayable data area.

g. Perform stack.

h. Alter table.

The format of information contained between the base and limit registers is dictated by the S-machine for that program. Therefore, the run structure for COBOL is different than that of SDL. Figure 2-4 illustrates the basic format of a program run structure, and figure 2-5 shows a program run structure as it appears in memory.

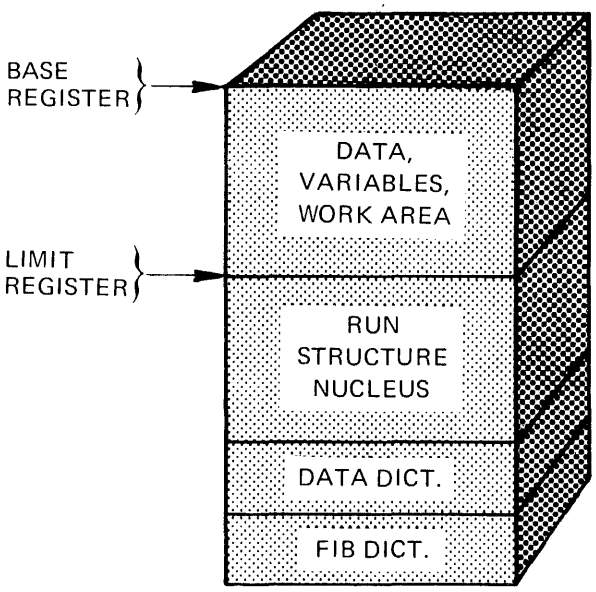


Figure 2-4. Program Run Structure

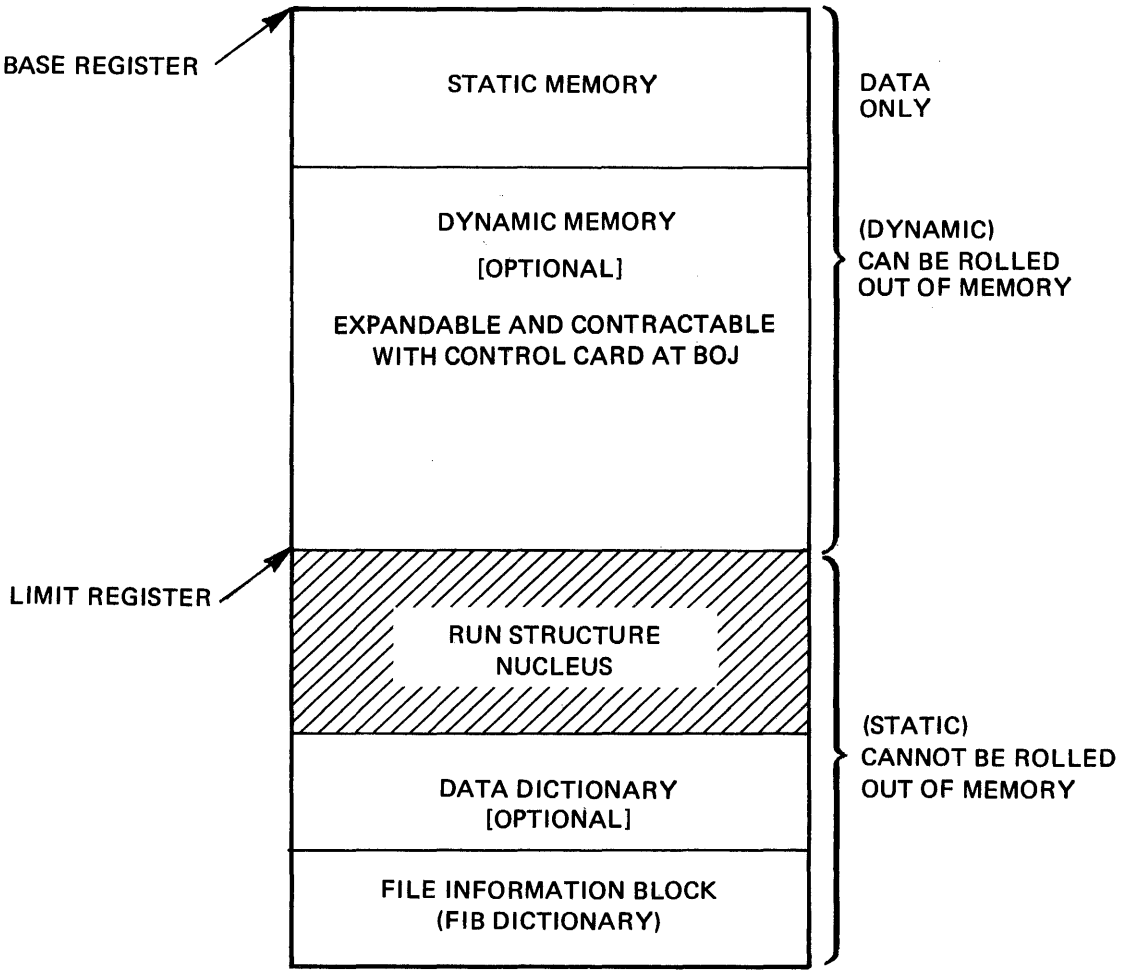


Figure 2-5. Program Run Structure Memory Layout

Run Structure Nucleus

The Run Structure Nucleus (RS.NUCLEUS) immediately follows the limit register. Each program, including the Master Control Program, has an RS.NUCLEUS. The RS.NUCLEUS contains information needed by the MCP and interpreters to execute programs. All communication between the MCP and object programs originates within a communicate message area in the RS.NUCLEUS of an object program.

Run Structure Nucleus Message Pointer

Each interpreter formulates communicate constructs that are received by the object program in its Run Structure Nucleus message pointer. The communicate message pointer, located immediately after the limit register, is 48 bits in length, and consists of the type of message plus its size and address. The location addressed by the communicate message pointer is a designated address within the data space of the program (between the base and limit registers) in which messages are sent to the MCP from the object program. The communicate message pointer has the following format.

Field Length, in Bits:	RS.COMMUNICATE.MSG.PTR			
	2	6	16	24
Description:	Type of Communicate or Interrupt	Type of Program-Dependent Interrupt	Length, in Bits, of the Communicate Message	Address of Communicate Message

The first field (see RS.ITYPE in table 2-6) distinguishes between program communicates and interrupts. The following codes identify the type of communciate or interrupt:

00 - Program-dependent or external interrupts.

10 - Undefined interrupt.

01 - Program communicate.

11 - Reserved for the MCP (used to call the TERMINATE.FILE.CLEANUP procedure for the program).

Run Structure Nucleus Reinstate Message Pointer

Messages from the MCP to the object program are placed in the Run Structure Nucleus reinstate message pointer of the program. The reinstate message pointer is a 48-bit descriptor, divided into two 24-bit fields. It signifies whether an exception condition has or has not taken place during the communicate, and indicates the type of exception condition.

The MCP sets the first 24 bits of the Run Structure Nucleus message pointer (RS.REINSTATE.MSG.PTR) to hexadecimal 000018 following a communicate request from an object program. The low-order 24 bits indicate the results of the communicate. Table 2-5 describes the exception conditions of a communicate message.

Table 2-5. RS.REINSTATE.MSG.PTR Exception Conditions

Type Code	Result of Communicate
000000	No errors, valid communicate.
000001	I/O requested on an unopened file.
000002	Read requested on an output file.
000003	Write requested on an input file.
000004	Seek requested on a serial file.
000005	Disk record count exceeded.
000006	Attempt made to read an unassigned disk area.
000007	No available user disk.
000008	Disk key exceeds the End-of-File pointer for a random file.

Run Structure Nucleus Format

Table 2-6 describes a Run Structure Nucleus format as it appears to the MCP. The RS.NUCLEUS resides in memory immediately following the limit register of its related program.

Table 2-6. Run Structure Nucleus

Field Name	Type	Length	Description
01 RS.NUCLEUS	Bit	2100	
02 RS.COMMUNICATE.MSG.PTR	Bit	48	All communication from the program to the MCP takes place through the communicate message pointer. Each interpreter has access to the program communicate message pointer.
03 RS.ITYPE	Bit	2	Contains the type of message: communicate or interrupt.
03 RS.INMBR	Bit	6	Contains type code for a program-dependent interrupt.
03 RS.ILENGTH	Bit	16	Contains the length field of a program communicate.
03 RS.IADDRESS	Bit	24	Points to the memory address containing the communicate message.
02 RS.COMMUNICATE.LR	Bit	24	Points to the memory address of the limit register with which this program communicates. Normally, this is the address of the MCP limit register.
02 RS.REINSTATE.MSG.PTR	Bit	48	Contains a communicate message from the MCP to the program. However, if no exceptions are encountered, the field contains zero.
02 RS.MY.BASE	Bit	24	Contains the address pointing to the base register of this program.
02 RS.MY.LIMIT	Bit	24	Contains the address pointing to the limit register of this program.
02 RS.MCP.BIT	Bit	1	Indicates this is the MCP Run Structure Nucleus.
02 RS.NIP	Bit	32	Contains a pointer to the next instruction.

Table 2-6. Run Structure Nucleus (Cont)

Field Name	Type	Length	Description
02 RS.SEG.DIC.PTR	Bit	24	Contains a pointer to the memory address of the Code Segment Dictionary.
02 RS.DATA.DIC	Bit	24	Contains a pointer to the memory address of the Data Dictionary.
02 RS.INTERP.ID	Bit	5	Identifies the interpreter being used by referencing its entry in the Interpreter Dictionary.
02 RS.INTRINSICS.LOC	Bit	10	Contains the page and segment displacement addresses of the SDL virtual memory intrinsic.
02 RS.M.MACHINE	Bit	768	Contains the scratchpad (processor or S-machine state of the program) at program roll-out.
02 RS.PRIORITY	Bit	24	Contains the assigned priority of the program which includes the main priority assigned, the fractional portion, and the internally controlled priority assigned by the MCP.
03 RS.PRIORITY.INTEGER	Bit	4	Contains the assigned priority of the program. Default is 4.
03 RS.INSTANT.PRIORITY	Bit	4	Not used.
03 RS.PRIORITY.FRACTION	Bit	16	Not used.
02 RS.NUMBER.FILES	Bit	8	Contains the total number of files declared by the program.
02 RS.BOOLEANS	Bit	24	Contains a series of flags used in conjunction with the RS.MCP.USE field indicating that the program is waiting for operator intervention signified by the bits set in the following fields.
03 RS.IL	Bit	1	Console message
03 RS.UL	Bit	1	Console message.
03 RS.OF	Bit	1	Console message.
03 RS.FR	Bit	1	Console message.

Table 2-6. Run Structure Nucleus (Cont)

Field Name	Type	Length	Description
03 RS.FM	Bit	1	Console message.
03 RS.OU	Bit	1	Console message.
03 RS.OK	Bit	1	Console message.
03 RS.RM	Bit	1	Console message.
03 FILLER	Bit	15	
02 RS.TYPE	Bit	16	Contains the disk copy address of this program.
02 FILLER	Bit	4	
02 RS.AUX1	Bit	24	Used as a temporary storage area for MCP communicates.
02 RS.Q.IDENT	Bit	24	Identifies the queue to which the program belongs.
02 RS.LAST.OVLY	Bit	24	Contains a pointer to the last program data overlay.
02 RS.DATA.OVERLAYS	Bit	24	Points to the dynamic memory address of the program.
02 RS.LAST.LINK	Bit	24	Contains a pointer to the last memory link within the dynamic memory space.
02 RS.WAIT.EVENT	Bit	24	Contains the address of the MCP event routine that caused the program to be placed in a "wait" condition. It usually references the memory address of the result descriptor field within the I/O descriptor for which the program is waiting an I/O completion.
02 RS.SER.NO	Bit	24	Contains the disk pack serial number for multiple pack files.
02 FILLER	Bit	4	
02 RS.OVLY.DISK.PTR	Bit	12	Contains an index into the data overlay area on disk.
02 RS.OVLY.DISK.SIZE	Bit	12	Contains the total data overlay space available on disk.

Table 2-6. Run Structure Nucleus (Cont)

Field Name	Type	Length	Description
02 RS.MIX.NMBR	Bit	24	Contains the mix-index number of this program.
02 RS. FIB.DIC	Bit	24	Points to the memory address of the File Information Block (FIB).
02 RS.TRACE	Bit	24	Points to the memory address of the trace print line.
02 RS.TRACE.FIB	Bit	24	Points to the memory address of the FIB during the trace mode of operation.
02 RS.TRACE.BITS	Bit	8	Contains flags denoting the type of trace being performed.
02 RS.MEDIA	Bit	1	Indicates whether the program is on disk or residing in memory.
02 RS.LENGTH	Bit	24	Contains the length of the program run structure.
02 RS.Q.LINK	Bit	24	Points to the memory address of the Run Structure Nucleus of the next program.
02 RS.STATUS	Bit	24	Checks the status of a program. Example: When MCP recognizes that a "WY" console message has been entered, RS.STATUS is scanned, giving condition of the program. Status codes and their descriptions are:

Status Code	Description
0	Executing.
1	No file.
2	No user disk.
3	Duplicate library.
4	Duplicate input file.
5	Possible duplicate file – multi-pack files.
6	Waiting for hardware device.
7	Program stopped.

Table 2-6. Run Structure Nucleus (Cont)

Field Name	Type	Length	Description
	Status Code		Description
	8		Waiting I/O complete.
	9		Waiting a data communication message.
	10		Waiting overlay.
	11		Waiting keyboard input.
	12		Hardware not ready.
	13		Waiting operator action.
	14		Waiting to close a file.
	15		Waiting DS or DP.
	16		Waiting for a continuation pack.
	17		File not on disk.
	18		Locked file open.
	19		Waiting core transfer.
	20		Program in a wait status.
	21		No memory, waiting in communicate queue.
	22		No memory, waiting in ready queue.
	23		Stopped in communicate queue.
	24		Stopped in ready queue.
	25		Waiting in communicate queue.
	26		Waiting in ready queue.
	29		Terminating.
	30		Data in ready queue.
	31		Data in communicate queue.

Table 2-6. Run Structure Nucleus (Cont)

Field Name	Type	Length	Description
02 RS.JOB.NUMBER	Bit	24	Contains the job number of this program.
02 RS.DISK.USE	Bit	20	Contains the total number of disk accesses for program overlays.
02 RS.TIME	Bit	20	Contains the total processor elapsed time for this program.
02 RS.PROG.PTR	Bit	36	Contains the disk address of the code file.
02 RS.PAUSE	Bit	24	Contains the time, in tenths of a second, that this program is not being processed.
02 RS.DONT.REENTER	Bit	1	Indicates that the program is not allowed to share its Code Segment Dictionary.
02 RS.SD.PTR.FLAG	Bit	1	Indicates that the field RS.SEG.DIC.PTR. points either to the memory address of the Code Segment Dictionary or to the memory address of the dictionary container.
02 RS.LINKS	Bit	1	Indicates that memory links are present.
02 RS.ABORT	Bit	1	Indicates that the program is ready to be discontinued.
02 RS.DISK	Bit	36	Contains an address pointing to the location of the code when rolled-out of memory.
02 RS.RECAUSE	Bit	1	Ensures that when the program is returned to memory for processing that it enters the CAUSE routine of the MCP.
02 RS.IN.MOTION	Bit	1	Indicates that the program is in the process of being rolled-out or rolled-in.
02 RS.HIERARCHY	Bit	8	Contains the nesting level of the program initiated by a program "CALL". There is a maximum of 255.
02 RS.APPARATION	Bit	1	Indicates that this program, while executing, referenced another program and is now waiting for the "called" program to terminate.

Table 2-6. Run Structure Nucleus (Cont)

Field Name	Type	Length	Description
02 RS.SIZECHANGE	Bit	1	Indicates that the scratchpad area is being programmatically modified in size.
02 RS.WAIT.MIX	Bit	1	Indicates that the field, RS.PAUSE, contains the mix-index of the program for which this program is waiting completion.
02 RS.WAIT.IS.JN	Bit	1	Indicates that the field, RS.PAUSE, contains the job-number of the program for which this program is waiting completion.
02 RS.OUT.MEM	Bit	1	Indicates that the program has stopped executing and is waiting memory space to restart.
02 RS.OUT.ABS	Bit	1	Indicates that the program has been stopped by the operator.
02.RS.FREEZE	Bit	1	Indicates that the program cannot be rolled-out of memory.
02 RS.PSEUDO.READER	Bit	24	Contains a pointer to the memory address of the current pseudo reader.
02 FILLER	Bit	24	
02 RX.OVLY.DESC	Bit	248	Contains the I/O descriptor used for code and data overlays.
02 RS.MCP.USE	Bit	1	Indicates that the event referenced by the field, RS.BOOLEANS, must occur before the program can continue processing.
02 RS.FPB	Bit	36	Points to the disk address of the file being opened.
03 FILLER	Bit	12	
03 RS.UNIT.INDEX	Bit	24	Contains an index referencing an entry in the Input/Output Available Table (IOAT).
02 RS.OVLY.DISK.BASE	Bit	36	Points to the base address of the program data overlays on disk.
02 RS.LOG.PTR	Bit	36	Points to the disk address of the current PPB and FPBs.

Object Program In Memory at BOJ

Figure 2-6 contains a simplified program run structure consisting of a programs various pointers and their relationship to each other.

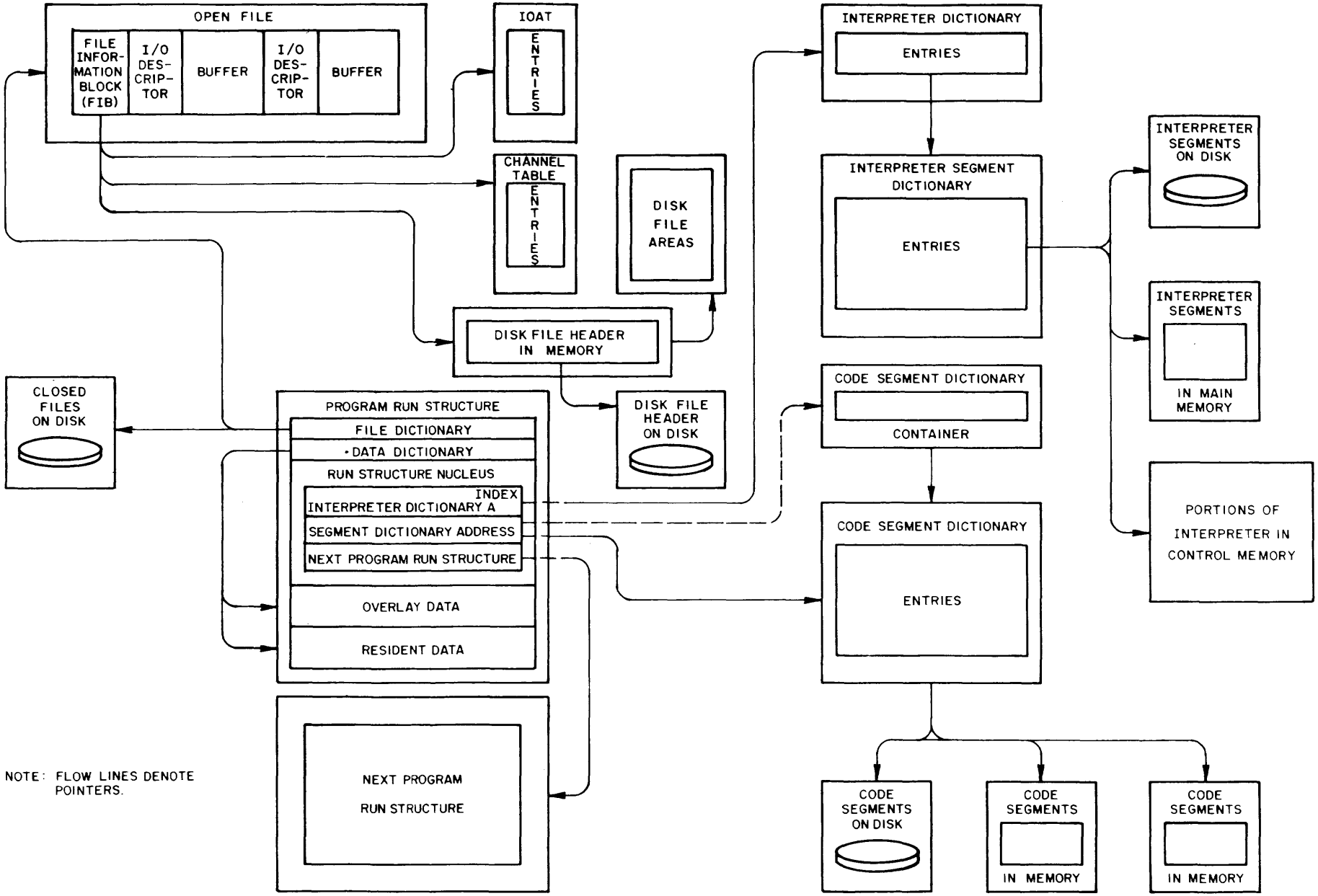


Figure 2-6. Simplified Program Run Structure

SECTION 3
INPUT/OUTPUT

INTRODUCTION

Input/Output (I/O) is the transfer of data between peripherals and the central processing unit. The I/O components of the B 1700, referred to as the I/O subsystem, consist of the Central Service Module (CSM), MCP, interpreters, and the object programs. Each peripheral has its own I/O channel which simultaneously inputs/outputs data with other channels.

The MCP supports the I/O subsystem by performing the following functions:

a. Manages all peripherals on the system.

b. Initiates I/O for all programs.

c. Reinstates the program after I/O.

d. Builds a log of I/O activities.

e. Handles automatic error recovery procedures.

The programmer, by generating I/O source statements such as *read* and *write* causes the compiler to formulate object code instructions containing the specific information needed to perform the I/O. Once the object program is in memory and is being executed, the interpreter communicates I/O results received from the CSM to the MCP through the communicate message pointer contained in the program Run Structure Nucleus.

The MCP prepares the data needed by the CSM to initiate the I/O. By constructing a description of the I/O and providing a buffer for the I/O data, the MCP passes the I/O information to the CSM. The CSM attempts to service the I/O while the MCP continues processing. Once data transfer is completed, control is eventually returned to the object program for the next instruction.

During the interpreter fetch loop, an interrogation is made for any I/O needing service. If I/O servicing is required, the CSM determines what I/O device or devices need servicing and performs the required data transfers.

I/O DESCRIPTOR

I/O descriptors are constructed by the MCP upon receiving an I/O request issued by an object program. An I/O descriptor contains the control information needed to achieve the I/O operation. I/O descriptors with multiple buffers are chained together, with each buffer having an associated descriptor.

After initiating the first I/O descriptor in the I/O buffer chain, the CSM examines the I/O result which is located in the first 24 bits of the I/O descriptor. When the result of that I/O is received, the CSM checks the I/O status of the next I/O descriptor and attempts to initiate that I/O request. While the CSM is linking between I/O descriptors, processing continues.

NOTE

If an error condition exists on any I/O operation, that particular I/O chain is broken and the MCP is notified.

The differences concerning the information contained within I/O descriptors are due to both the hardware device selected and the nature of the I/O.

I/O Status

After an I/O operation is complete and the ending buffer address of the data is stored, the status of the I/O is placed in the I/O result field in the I/O descriptor. The result status is marked either complete or complete with exceptions.

Table 3-1 describes the format of an I/O descriptor. For additional information concerning I/O descriptors refer to the B 1700 System Reference Manual, Form No. 1057155.

Table 3-1. I/O Descriptor

Field Name	Type	Length	Description
01 IO.DESCRIPTOR	Bit	248	
02 IO.RESULT	Bit	24	Contains the I/O result status of an attempted I/O operation.
03 IO.COMPLETE	Bit	1	Used by the I/O control, indicating a successful I/O has been completed.
03 IO.EXCEPTION	Bit	1	Indicates whether an exception condition has been issued by the I/O control. The IO.EXCEPTION and the IO.COMPLETE fields enable the MCP to monitor the status of the I/O at any time. The settings of both fields that indicate I/O status are:

IO.COMPLETE Field	IO.EXCEPTION Field	Description
0	0	Indicates that the I/O descriptor is not in use. The MCP writes zeroes to this area.
0	1	Indicates that the I/O request is in process.
1	0	Indicates that the I/O is complete with no exceptions.
1	1	Indicates that the I/O is complete with exceptions.

03 FILLER	Bit	10	
03 IO.DEST	Bit	3	Contains the port identification for this I/O descriptor.

Table 3-1. I/O Descriptor (Cont)

Field Name	Type	Length	Description
03 IO.INTERRUPT	Bit	1	Causes the CSM to store an interrupt message in the INTERRUPT queue when the I/O is complete.
03 IO.HI.INT	Bit	1	Indicates that the MCP is not requesting a high-priority interrupt at operation complete (MICR Reader/Sorter).
02 IO.LINK	Bit	24	Contains an address pointing to the next I/O descriptor in the chain. After storing the I/O status information, and after returning any requested interrupt message, the CSM (in the absence of an error condition) fetches the address contained in IO.LINK referencing the next I/O descriptor.
02 IO.OP	Bit	24	Contains the type of the I/O operation. It consists of the operation code, any variants, and the unit number of the I/O.
02 IO.BEGIN	Bit	24	Contains an address pointing to the beginning data location in the I/O buffer.
02 IO.END	Bit	24	Contains an address pointing to the end of the I/O buffer.
02 IO.DISK.ADDRESS	Bit	24	Contains the beginning disk segment address of the I/O operation. When the I/O is magnetic tape, IO.DISK.ADDRESS contains the address of the tape unit Lock descriptor.
The entries above this point are common for all I/O descriptors.			
02 IO.MCP.IO	Bit	24	Indicates how the MCP is to handle an I/O request when control is regained from an interrupt situation.
02 IO.FIB	Bit	24	Addresses the File Information Block (FIB) of the file requesting the I/O.

Table 3-1. I/O Descriptor (Cont)

Field Name	Type	Length	Description
02 IO.FIB.LINK	Bit	24	Addresses the next I/O descriptor in this I/O chain.
02 IO.BACK.LINK	Bit	24	Addresses the previous I/O descriptor in the I/O chain
02 IO.PORT.CHAN	Bit	7	Contains the physical hardware location (port and channel) for the I/O request.
03 IO.PORT	Bit	3	
03 IO.CHANNEL	Bit	4	

MAGNETIC TAPE I/O SUBSYSTEM

Because of the serial nature in processing magnetic tape, there is only one chain of I/O descriptors for the entire magnetic tape I/O subsystem. Similar to disk I/O operation, the I/O descriptors are executed in logical sequence. To help ensure the serial processing of magnetic tape, a "lock" type descriptor is used for control.

Lock Descriptor

Table 3-2 describes the Lock Descriptor format.

Table 3-2. Lock Descriptor

Field Name	Type	Length
01 LOCK.DESCRIPTOR	Bit	144
02 LOCK.RESULT	Bit	24
02 LOCK.LINK	Bit	24
02 LOCK.OP	Bit	24
02 LOCK.LINK.FIRST	Bit	24
02 LOCK.LINK.NEXT	Bit	24
02 LOCK.LINK.PREVIOUS	Bit	24

There is one lock descriptor in the magnetic tape I/O subsystem chain for each tape unit. All I/O descriptors for a particular unit are linked, directly or indirectly, to the lock descriptor for that unit. The LOCK.LINK field of the lock descriptor always contains either the memory address of the lock descriptor for the next unit in the magnetic tape I/O subsystem, or the address of a "pause" type descriptor. There is one "pause" descriptor for the entire magnetic tape I/O subsystem. The LOCK.LINK.FIRST field of the lock descriptor always contains the memory address of the first physical record for the unit. This field is initialized by the MCP when the file is opened. LOCK.LINK.NEXT contains the memory address of the next logical record to be executed on the unit.

NOTE

The physical record is the first record in a data buffer. The logical record is the record within a buffer that is currently being processed.

LOCK.LINK.NEXT is initialized by the MCP, but is maintained by the CSM.

The function of a lock descriptor can be explained using the following example where two programs, A and B, are using one magnetic tape control with three available units. The complexity of the magnetic tape I/O subsystem, however, does not increase, regardless of the number of controls or units.

Example:

Figure 3-1 demonstrates program A and program B utilizing two units under one control, and the I/O descriptor chain that is manufactured by the MCP for the I/O operations.

In this example, Program A opens unit 1 as input and has three buffers. The MCP obtains memory space and links the three I/O descriptors to the lock descriptor for unit 1. The three I/O descriptors are denoted in figure 3-1 as READ-1, READ-2, and READ-3.

Program B opens unit 2 as output and has two assigned buffers. The MCP obtains memory space and links the two I/O descriptors to the lock descriptor for unit 2. The two I/O descriptors are denoted in figure 3-1 as WRITE-1 and WRITE-2.

Unit 3, not currently being used, has the mandatory lock descriptor present.

At the time of the open by program A, the magnetic tape control is idling (pause type descriptor). When the pause is recognized as complete (result of the open), the CSM exits the pause descriptor and reads the lock descriptor for unit 1. The I/O complete bit in the result descriptor field of a lock descriptor is always 0, causing the CSM to always read the lock descriptor operation code.

The CSM recognizing a lock descriptor, swaps 01 into the first two bits of the result descriptor field, LOCK.RESULT, and examines the two bits that were received. If they are 01, indicating I/O in progress, the CSM exits immediately to the next lock descriptor in the chain. This prevents two controls from attempting to execute descriptors for the same unit at the same time. In this example, however, there is only one control but this technique of checking lock descriptors provides a fast and simple method for the MCP to lock out an entire unit.

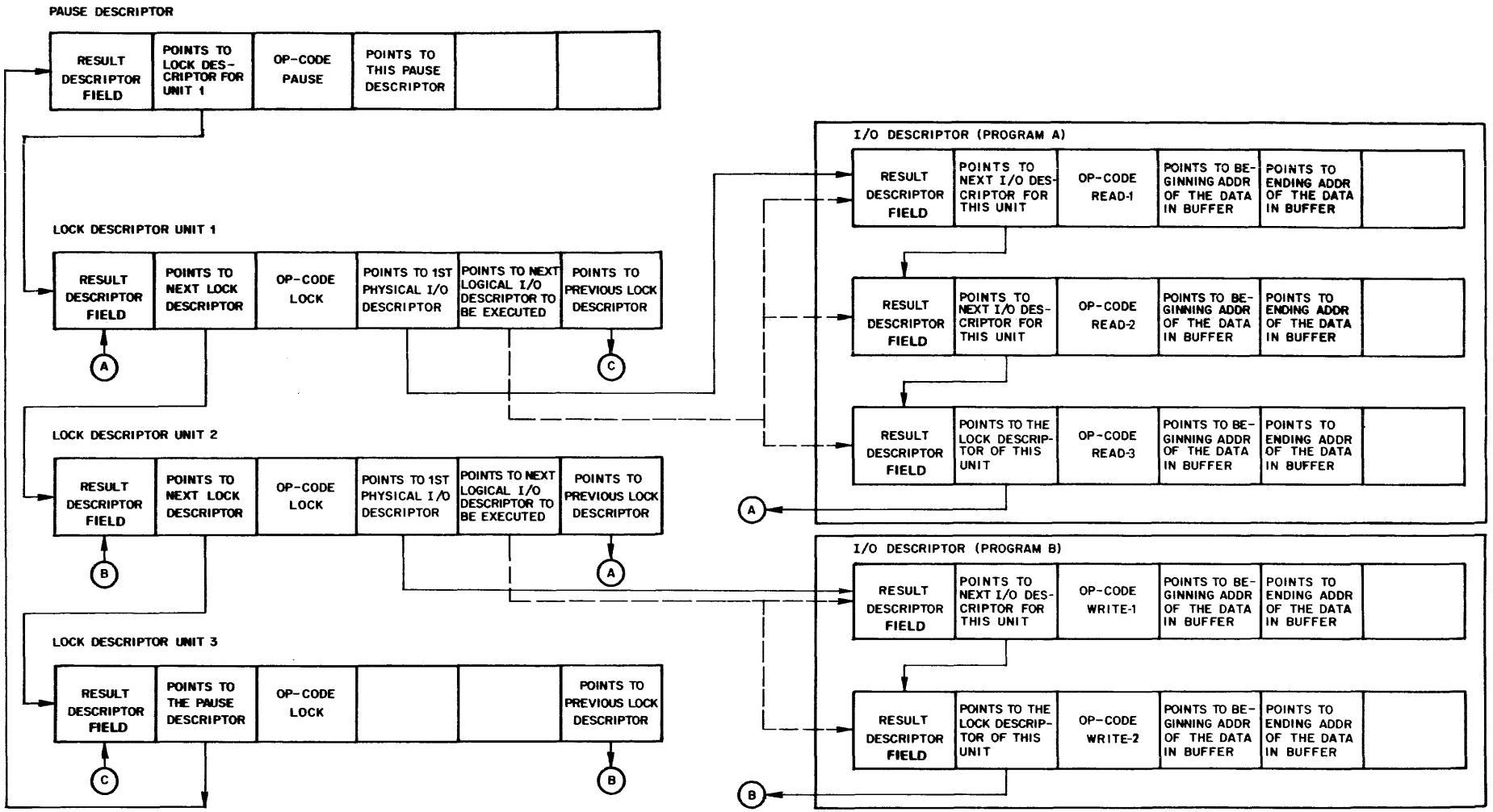


Figure 3-1. Magnetic Tape I/O Subsystem Descriptor

NOTES:

1. SOLID LINES DENOTE POINTERS SET UP BY MCP ROUTINES (OPEN, CLOSE).
2. BROKEN LINES DENOTE POINTERS MODIFIED BY I/O CONTROL OPERATIONS.
3. BLANK BOXES DENOTE "NOT USED".

CHANNEL TABLE

The MCP Value stack (refer to B 1700 System Reference Manual, Form No. 1057155) contains a set of eight contiguous 24-bit fields (0-7). Each field contains either a zero or the Channel table memory address corresponding to that port.

If the Value stack field contains any dispatches directed to that port (e.g., a message from the MCP notifying the CSM that there is an I/O for that port and channel), the dispatch routine in the CSM reports a missing device condition to the MCP. If the field contains a memory address, the I/O is attempted. Each Channel table is an area in memory containing 16 entries, each 48 bits in length. Table 3-3 describes the Channel table.

Table 3-3. Channel Table

Field Name	Type	Length	Description
01 CHANNEL.TABLE			
02 CHANNEL.BUSY	Bit	1	Indicates the I/O device has processing activity.
02 CHANNEL.PENDING	Bit	1	Indicates the I/O is waiting execution.
02 CHANNEL.EXCEPTION	Bit	1	Indicates the device is idle because of an exception condition. All dispatches to this device are ignored as long as this bit is 1.
02 CHANNEL.PAUSE	Bit	1	Indicates whether there is to be an automatic dispatch at Timer interrupt using the reference address stored in the field, CHANNEL.REF.ADDR.
02 CHANNEL.OVERRIDE	Bit	1	Overrides bits 0, 1, and 2 of the table and sends a dispatch to the control regardless of the status or condition of the I/O. Bits 0, 1, and 2 of the table are set to 0 at the time of a dispatch.
02 CHANNEL.EXCHANGE	Bit	1	Indicates whether this entry is or is not part of an exchange, or is or is not the last entry in the exchange.
02 CHANNEL.OLD.MODE	Bit	1	Indicates whether the device is to be processed by "enhanced" I/O or the previous subsystem.

Table 3-3. Channel Table (Cont)

Field Name	Type	Length	Description
NOTE "Enhanced" I/O is a temporary term used to distinguish between the newer version of the I/O subsystem and the previous or older I/O subsystem to which there has been significant programmatic changes.			
02 CHANNEL.INTEGRITY	Bit	1	Indicates the MCP has determined what kind of a device is on the channel.
02 CHANNEL.LOCK.CNTR	Bit	4	Contains counters for the lock descriptor chains.
02 CHANNEL.TYPE	Bit	4	Contains a type code used for the MCP Dump Analyzer program. Type 0 = serial device Type 1 = disk Type 2 = tape Type 3 = cassette Type 4 = MFCU
02 CHANNEL.LAST	Bit	1	Indicates that this device is the last entry in the Channel table for this port. This aids the CSM at Timer interrupt for sparsely populated channel tables.
02 CHANNEL.EXCHANGE.PC	Bit	7	Stores the "linked" port and channels.
03 CHANNEL.EXCHANGE.P	Bit	3	Contains the port.
03 CHANNEL.EXCHANGE.C	Bit	4	Contains the channel.
02 CHANNEL.REF.ADDR	Bit	24	Contains the beginning reference address when the control is activated after a Timer interrupt. (Refer to CHANNEL.PAUSE)

Channel Table Operation

The first two bits of a Channel table entry (CHANNEL.BUSY and CHANNEL.PENDING) are used together as described below.

a. When the CSM is at a point where it is idle, it swaps 01 into the first two bits of the Channel table. If a 00, 01, or 10 is received, the device goes idle (waits for I/O to complete) and writes a 00. If it receives a 11, it writes a 10 and continues executing.

b. When the CSM initiates a device from a dispatch, it swaps in a 11. If a 00 is received, it dispatches the I/O; if it receives a 10 or 11, it exits, doing nothing; if it receives a 01, it continues to read the first two bits until it receives a 00.

The CHANNEL.EXCEPTION field in the Channel table is used by those devices that go idle on an I/O exception condition. The MCP resets this bit before redispatching the I/O descriptor that caused the exception. (Tape and disk devices do not go idle on an exception condition.)

If the MCP forces a dispatch on a "busy" control (e.g., console printer in a test and wait condition), it sets the CHANNEL.OVERRIDE bit in the Channel table prior to the dispatch.

The result status field in the I/O descriptor is initialized by the MCP prior to a dispatch, and is updated by the I/O subsystem while the I/O operation is in progress. The result status field is then replaced by the result descriptor field when the I/O is complete.

Listed below are the result status field bit assignments prior to an I/O completion (bits are indexed from left to right, 0 to 23):

Bit	Description
0	Always contains a 0.
1	Set to 1 by the CSM during processing.
2	Reserved for expansion of the I/O subsystem.
3	Reserved for expansion of the I/O subsystem.
4	Reserved for expansion of the I/O subsystem.
5	Indicates a locked state.
6	Indicates a disk device (uses explicit pause and link on exception or I/O complete).
7	Indicates a tape device (Lock descriptor mechanism).
8	Indicates reader/sorter.
9	Indicates data communications.
10	Reserved.
11	Indicates this is an exchange.
12-14	Contains the destination port.
15	Indicates an interrupt request.
16	Indicates an interrupt request (high-priority).
17-19	Contains the port to which to return the interrupt (usually 0, indicating the processor).
20-23	Contains the channel number of the device.

Except when the CHANNEL.OLD.MODE field in the Channel table indicates the "old" I/O mode, a dispatch is directed by the MCP to the CSM for each result status field of an I/O descriptor that is initialized. If the result status field is not set to 1 during the I/O operation, bits 0 and 1 are set to 0 immediately prior to a dispatch.

After the dispatch and prior to I/O complete, the only two bits the MCP can modify in the result status field are bits 15 and 16. The modification is done as a replacement operation and is done only once.

For those devices that are processed by the "old" I/O mode, such as Data Comm and the Reader/Sorter, the CHANNEL.OVERRIDE field in the channel table is set prior to initiating the dispatch.

PORTS

In the Value stack of the MCP, there is a set of eight contiguous 24-bit fields that correspond to ports 0 through 7 respectively. Each field contains either a 0 or the memory address of the Channel table that corresponds to that port.

When a port field contains zeros, indicating that there is not a channel associated with the port, dispatches directed to the referenced port cause the dispatch routine in the CSM to wait in anticipation of a "missing device condition." A "missing device condition" in any other situation halts the system.

I/O devices connected to the processor on port 0 are assumed to be on port 7.

EXCHANGES

Exchanges are logical switching matrices interfacing a control or controls to one ore more electronic units that direct the flow of information to and from those units. For example: Suppose there are number port and channel combinations corresponding to *n* working controls for an exchange. One of those combinations according to the I/O descriptor is selected as the primary port and channel for that exchange. The primary port and channel is the only port and channel combination allowed in the result status field of any I/O descriptor for that exchange.

The Channel table entry corresponding to the primary port and channel for the exchange sets the CHANNEL.EXCHANGE bit. The port and channel bits in the Channel table then point to the next port and channel for the same exchange. If the new port and channel is the last port and channel combination for the exchange, then CHANNEL.EXCHANGE in the Channel table is not set, and its port and channel field in the Channel table points back to the primary port and channel. Otherwise, CHANNEL.EXCHANGE is set and the port-channel field points to the next port and channel for that exchange.

Dispatches are sent only to the primary port and channel. The CSM searches for an idle control on an exchange when attempting to service a dispatch.

Any dispatches directed to a device whose Channel table entry has CHANNEL.PAUSE set will cause the CSM to use the address in the CHANNEL.REF.ADDR as the dispatch address, as opposed to the address sent with the dispatch. This action occurs because a magnetic tape I/O subsystem must be initiated through lock descriptors.

INPUT/OUTPUT ASSIGNMENT TABLE (IOAT)

The Input/Output Assignment Table (IOAT) is a memory resident table, constructed, and used exclusively by the MCP. The purpose of the IOAT is to retain information concerning the status and availability of all peripherals on the system.

Table 3-4 describes an IOAT entry format.

Table 3-4. Input/Output Assignment Table

Field Name	Type	Length	Description
01 IOAT	Bit	492	
02 UNIT.INITIAL	Bit	66	Contains the physical location of this peripheral.
03 UNIT.HDWR	Bit	6	Contains the hardware identification number.

Table 3-4. Input/Output Assignment Table (Cont)

Field Name	Type	Length	Description
03 UNIT.PCD	Bit	12	Contains the port and channel of the peripheral.
04 UNIT.PORT.CHANNEL	Bit	7	Contains the port and channel.
05 UNIT.PORT	Bit	3	Contains the port.
05 UNIT.CHANNEL	Bit	4	Contains the channel.
04 FILLER	Bit	1	Reserved.
04 UNIT.UNIT	Bit	4	Contains the unit number of this port and channel.
03 UNIT.NAME	Char	6	Contains the unit mnemonic identification.
02 UNIT.LABEL.ADDRESS	Bit	36	Points to the unit label location. When the entry represents a disk unit, it points to the Disk Pack Information table.
03 FILLER	Bit	24	
03 UNIT.PACK.INFO	Bit	24	Contains @FFFFFF@ if this entry is for a system pack.
02 UNIT.RS	Bit	24	Contains the address pointing to the limit register of the program to which this entry is assigned.
02 UNIT.FLAGS	Bit	24	A set of indicators that continually monitor the operating condition of the peripheral represented by this entry in the IOAT. The following fields are used for the monitoring purpose.
03 UNIT.AVAILABLE	Bit	1	
03 UNIT.AVAILABLE.INPUT	Bit	1	
03 UNIT.AVAILABLE.OUTPUT	Bit	1	
03 UNIT.WAIT.FOR.NOT.READY	Bit	1	
03 UNIT.TEST.AND.WAIT	Bit	1	
03 UNIT.SAVED	Bit	1	
03 UNIT.REWINDING	Bit	1	
03 UNIT.EOF.SENSED	Bit	1	

Table 3-4. Input/Output Assignment Table (Cont)

Field Name	Type	Length	Description
03 UNIT.LOCKED	Bit	1	
03 UNIT.LABEL.SENSED	Bit	1	
03 UNIT.PRINT.BACKUP	Bit	1	
03 UNIT.PURGE	Bit	1	
03 UNIT.LOCK.AT.TERM	Bit	1	
03 UNIT.TO.BE.SAVED	Bit	1	
03 UNIT.FLUSH	Bit	1	Indicates the unit is to read to End-of-File. (Example: Card Reader.)
03 UNIT.TAPEF	Bit	1	
03 UNIT.DISKF	Bit	1	
03 UNIT.STOPPED	Bit	1	
03 UNIT.TRANSLATE	Bit	1	
03 UNIT.CTRL.CARD.USING	Bit	1	
03 UNIT.REMOTE.JOB	Bit	1	
03 UNIT.CLOSED	Bit	1	
03 UNIT.FILLER	Bit	2	
02 UNIT.STATUS	Bit	15	Contains the first 15 bits of the I/O result descriptor field.
02 UNIT.JOB.NUMBER	Bit	24	Contains the job number assigned to this entry.
02 UNIT.FIB.ADDRESS	Bit	24	Points to the File Information Block (FIB) when this entry is associated with a file.
02 UNIT.LABEL.TYPE	Bit	2	Contains the label type code: 00 = Omitted 01 = Burroughs standard 02 = ANSI 03 = Installation
02 FILLER	Bit	24	Reserved.
02 UNIT.TEST.DESC	Bit	248	Contains the test I/O descriptor used to interrogate the status of the peripheral (e.g., Is it ready? Is it busy?).

SECTION 4
MCP DISK STRUCTURES

INTRODUCTION

A significant aspect of the B 1700 design is disk management, which is the responsibility of the MCP. There are four areas of disk management which the MCP controls:

a. Directory maintenance.

b. Disk allocation.

c. File assignment.

d. Record addressing.

A disk pack or cartridge must be initialized by the program Disk Initializer before it can be used on the system. At the time of initialization, the user assigns a serial number, a disk identifier, the type of pack or cartridge (system, unrestricted, restricted, or interchange), the date of initialization, and comments if desired.

Head-per-track disk is pre-initialized; no user initialization is necessary.

The basic function of initializing a disk is to assign an address for each segment. This address is the means by which the MCP accesses disk segments. After the assigning of the addresses, the Disk Initializer generates the following: a Master Available Table, skeletal Disk Directory, Working Available Table, Temporary Available Table, and a label identifying the pack or cartridge.

DISK PACK/CARTRIDGE CHARACTERISTICS

The disk pack and disk cartridge are data storage devices that can be moved on or off line. Both the disk pack and disk cartridge are divided into segments (sectors), tracks, and cylinders which organize the storage of data and identify its locations. Figures 4-1 and 4-2 illustrate the basic characteristics of the read/write mechanism and the recording surfaces.

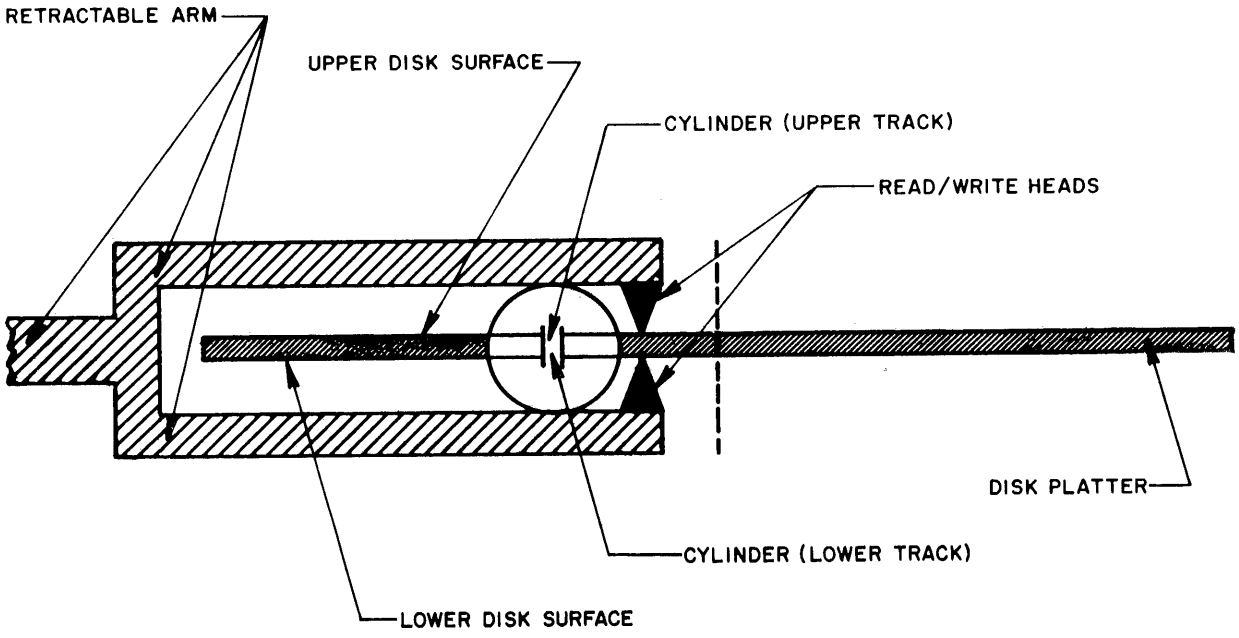


Figure 4-1. Disk Read/Write Mechanism

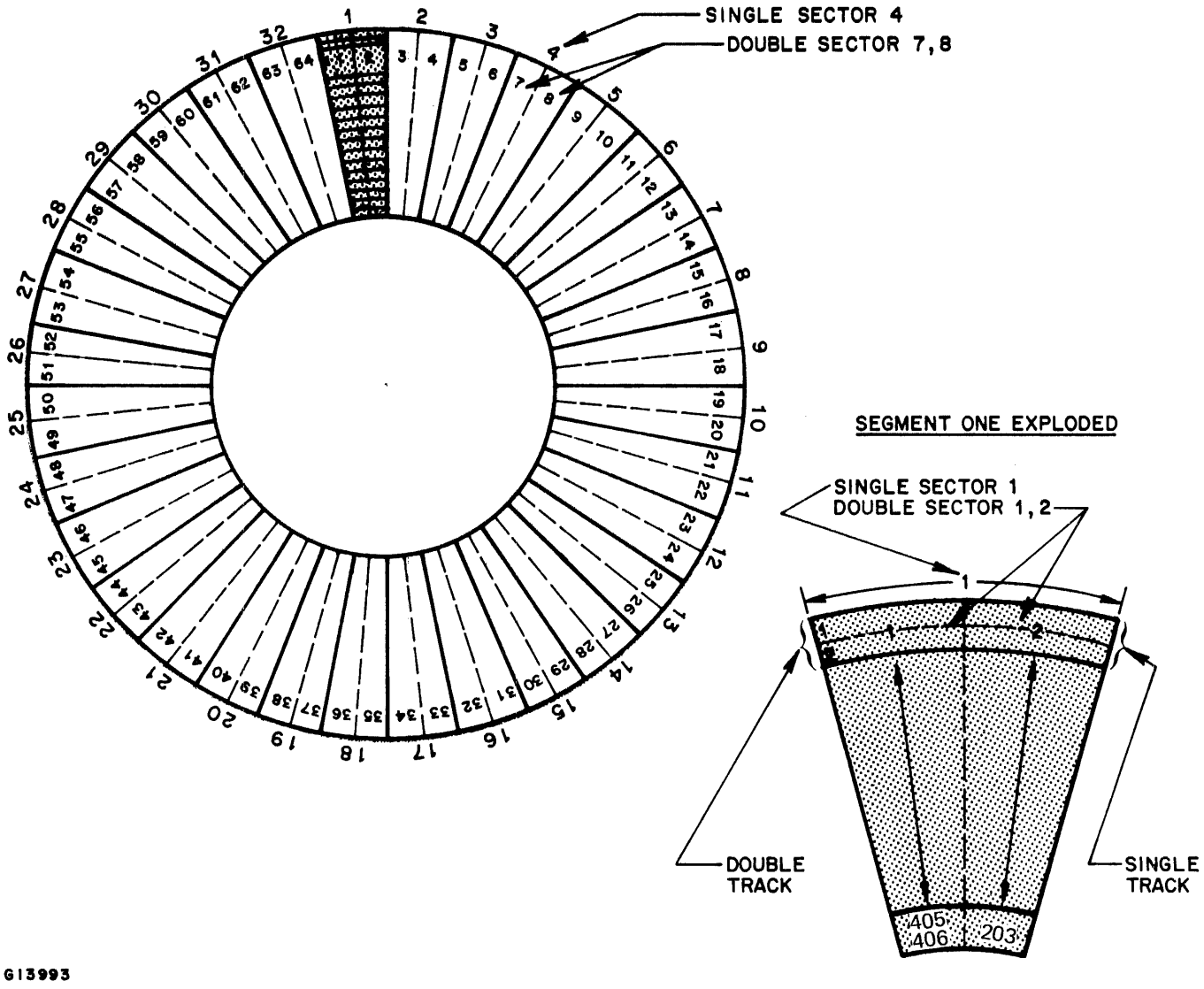


Figure 4-2. Disk Recording Surface

The Data storage format consists of a circular track, divided into sectors called segments with each segment containing 180 bytes of storage, as shown in figure 4-2. A disk platter has two surfaces, each capable of storing data. All tracks that lie in the same vertical plane constitute a cylinder.

DISK FILE IDENTIFIERS

The B 1700 programmer addresses disk files by their symbolic names. The physical disk address of a file is the responsibility of the MCP and not the user program. The file-identifier has three possible components:

a. Disk pack/cartridge-identifier.

b. Main directory identifier (family-name).

c. Sub-directory identifier (file-identifier).

For a detailed description of disk file names and their usage concerning system operation, refer to the B 1700 System Software Operational Guide, Form No. 1068731.

DISK DIRECTORIES

Files are either temporary or permanent. A temporary file is a disk file that is created by programmatic action and whose existence is only in relationship to the creating program. It becomes non-existent when the creating program terminates or when the file is closed without a lock. A permanent file is one that can be accessed at any time by any program and remains in the Disk Directory until removed.

At COLDSTART, the MCP reserves 16 contiguous segments of systems disk for the initial Disk Directory; each segment contains 11 entries, giving a possible total of 176 directory entries. If more space is required, the MCP increases the Disk Directory size by one segment for each additional 11 entries. Figure 4-3 shows a simplified Disk Directory with a main directory/sub-directory file entry. Table 4-1 describes the Disk Directory format.

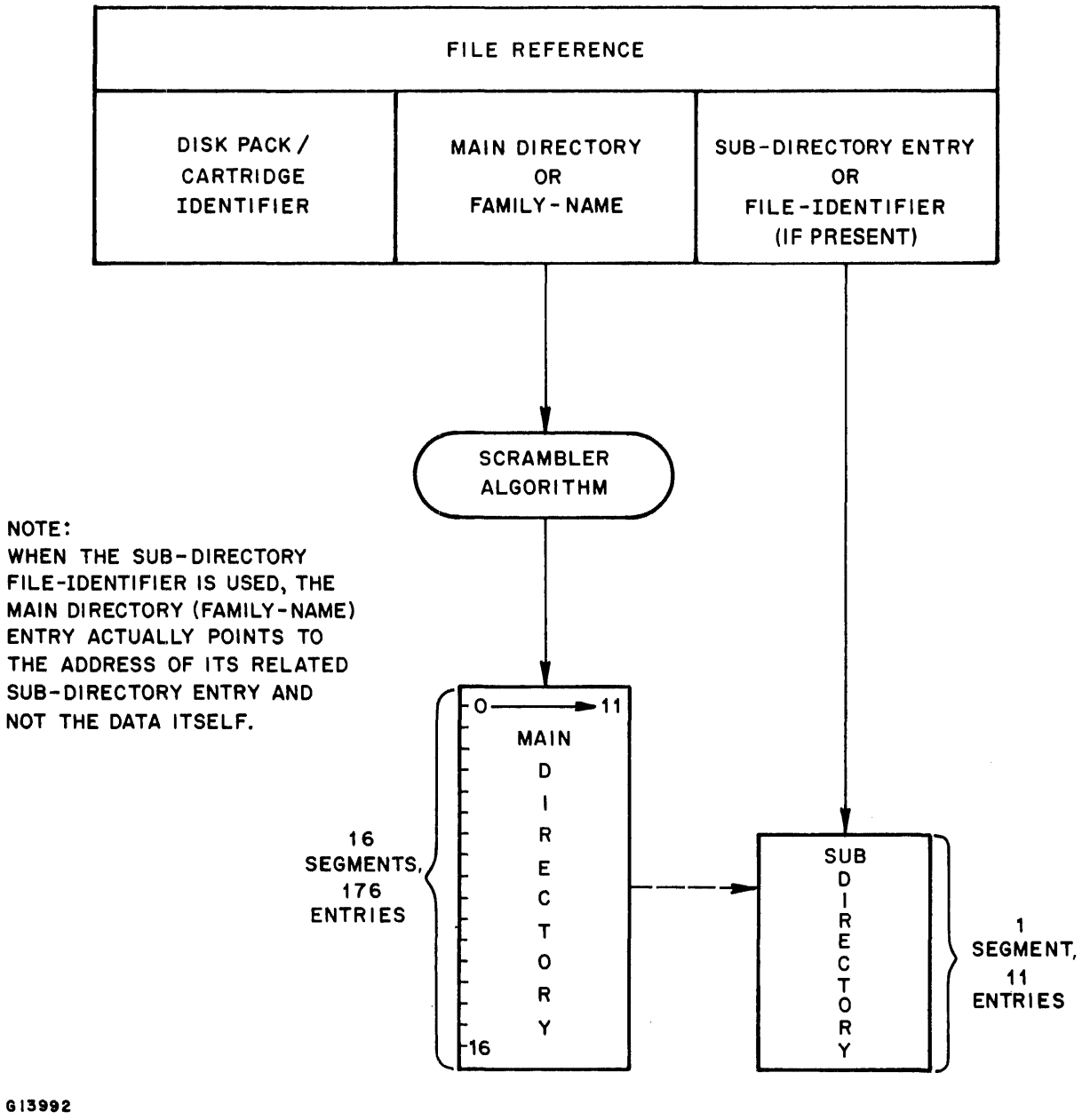


Figure 4-3. Disk Directory Filing Technique

Table 4-1. Disk Directory

Field Name	Type	Length	Description
01 DIRECTORY	Bit	1440	
02 DISK.SUCCESSOR	Bit	36	Contains a disk address that points to the next directory entry, and is referred to as the forward link. If this entry contains zero, it is the last entry of the directory.
02 DISK.PREDECESSOR	Bit	36	Contains a disk address that points to the previous directory entry, and is referred to the backward link. If this entry contains zero, it is the first entry of the directory.
02 DISK.SELF	Bit	36	Contains the disk address of this entry in the directory.
02 FILLER	Bit	12	
02 DISK.NAME	Char	10	Contains the family-name of this entry.
02 DISK.ADDRESS	Bit	36	Points to the file header of this entry; if there is a sub-directory entry, it points to the related sub-directory.
02 DISK.FILE.TYPE	Bit	4	Describes the file access type referenced by this directory entry, as follows: Type — Description 1 Log, Elog 2 Sub-directory 3 Control deck 4 Print back-up 5 Punch back-up 6 Dump file 7 Interpreter 8 Code file 9 Data file 10 Undefined 11 Variable data
02 FILLER	Bit	1200	Contains space for 10 available entries.

Main Directories

The MCP enters a file into the main directory by a technique whereby the family-name of the file is randomized into a single reference digit of 0 to 15. This digit is used as an index into the directory where all files having the same index reside. This method reduces directory search time by a factor of 16. If, while scanning the directory, the MCP finds the dictionary is full, it expands that directory by one segment, updating the field DISK.SUCCESSOR to point to the new segment. If the MCP is simply searching for a file, the family-names are compared until the file is located.

Sub-Directories

Files that are identified by a first or primary name and also by a second name have sub-directory entries. When a file has an identifier such as A/B or A/C, the main directory field, DISK.ADDRESS, does not reference the location of the file but contains the address of its related sub-directory. The DISK.ADDRESS field in the sub-directory then points to the file header. The DISK.FILE.TYPE field in the main directory contains a 2 when there is a sub-directory associated with the file. The format of main directories and sub-directories is the same. Figure 4-4 illustrates an example of disk directory entries.

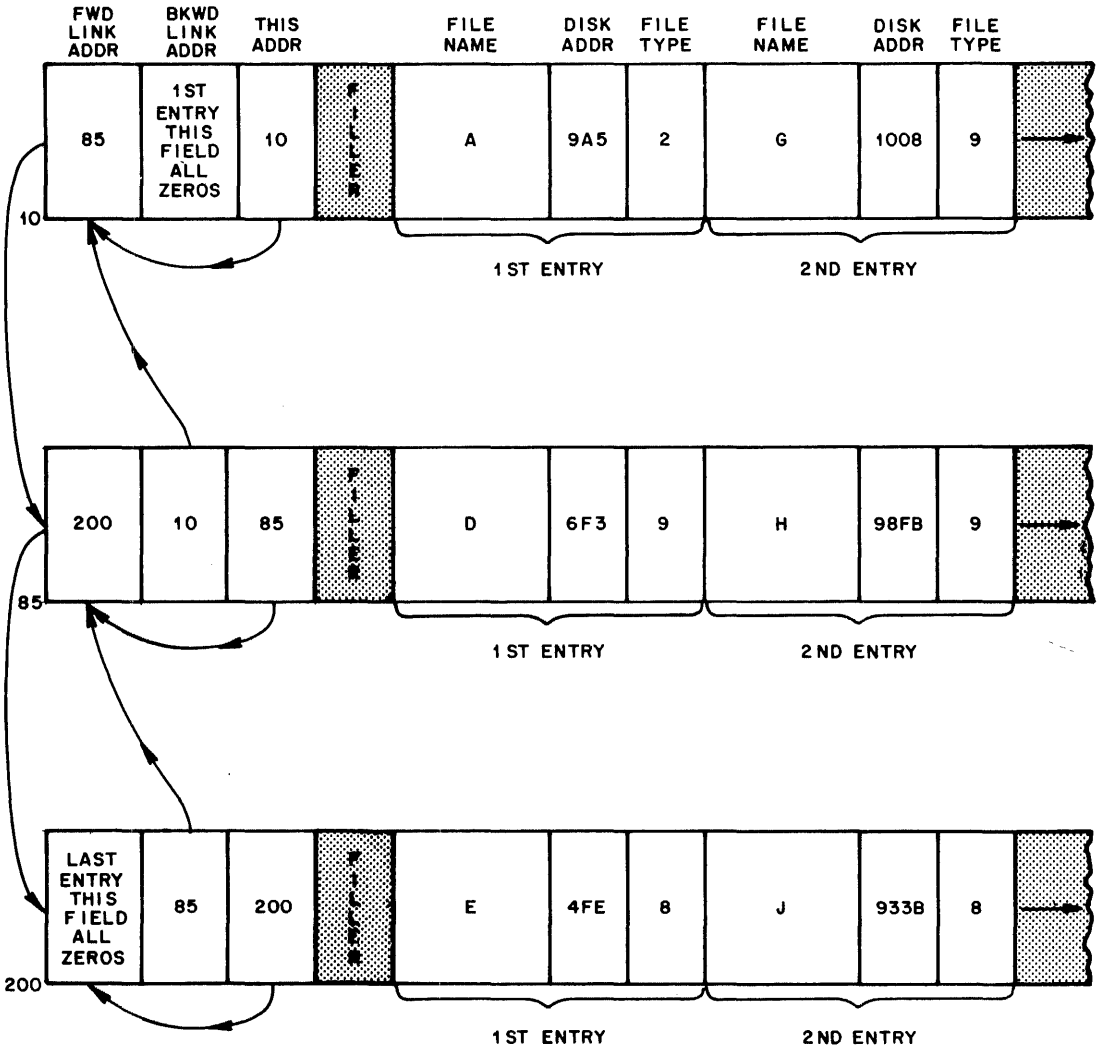


Figure 4-4. Disk Directory Entries

DISK FILE HEADER

The disk file header contains the physical characteristics of the disk file and the absolute addresses of each area allocated to the file. Table 4-2 describes the format of a disk file header table entry.

NOTE

Disk file headers are outside a program run structure. Therefore a program memory dump does not reflect the disk file headers. The disk file headers can be obtained by dumping total memory and executing MCPII/ANALYZER.

Table 4-2. Disk File Header

Field Name	Type	Length	Description
01 DISK.FILE.HEADER	Bit	1440	
02 DFH.CORE.ADDR	Bit	24	Points to the memory address of the disk file header when the header is in memory.
02 DFH.SELF	Bit	36	Contains the disk address of this disk file header.
02 DFH.NO.USERS	Bit	8	Contains the number of programs that are referencing the file but not necessarily accessing the file.
02 DFH.USERS.OPEN.OUT	Bit	4	Contains a counter signifying the number of programs that are currently accessing the file.
02 DFH.OPEN.TYPE	Bit	4	Contains the type of file open: 1000 = Lockout 0010 = Input 0001 = Output 0100 = Lock
02 DFH.FILE.TYPE	Bit	4	Contains the type of file.
02 DFH.PERMANENT	Bit	4	Contains the file status. 0000 = Temporary File 0001 = Permanent File 1111 = System File
02 DFH.HDR.SIZE	Bit	24	Contains the disk file header size.
02 DFH.RECORD.SIZE	Bit	24	Contains the record size.
02 DFH.RECDS.BLOCK	Bit	24	Contains the number of records per block (blocking factor).

Table 4-2. Disk File Header (Cont)

Field Name	Type	Length	Description
02 DFH.BLOCKS.AREA	Bit	24	Contains the number of blocks per area.
02 DFH.SEGS.AREA	Bit	24	Contains the number of segments per area.
02 DFH.AREA.RQST	Bit	12	Contains the number of areas requested for the file.
02 DFH.AREA.CTR	Bit	12	Contains the number of areas used.
02 DFH.EOF.POINTER	Bit	24	Input File: Contains the End-of-File address. Output File: Contains the number of records output. Random File: Contains the maximum key value.
02 DFH.BPS.NO	Bit	24	Contains the base pack serial numbers of a multiple pack file.
02 DFH.BLOCK.COUNT	Bit	24	Contains the number of processed blocks.
02 DFH.MPF	Bit	4	Indicates this is a multiple pack file.
02 DFH.DBM.LINK	Bit	24	Points to the next file header (data management).
02 DFH.DBM.DFH.NO	Bit	24	Contains a reference number used as the internal identifier for file headers using data management.
02 DFH.USER.INFO	Bit	24	Not implemented.
02 DFH.SAVE.FACTOR	Bit	12	Contains the number of days a file is to be retained beyond its last access date.
02 DFH.CREATION.DATE	Bit	16	Contains the date of file creation.
02 DFH.ACCESS.DATE	Bit	16	Contains the date that the file was last opened.
02 DFH.MPF.ADDR	Bit	36	Contains the disk address of the Multiple Pack Information table.
02 FILLER	Bit	60	Reserved.

Table 4-2. Disk File Header (Cont)

Field Name	Type	Length	Description
02 DFH.AREA.ADDRESS	Bit	36	Contains the beginning disk address of the first data location. There is a maximum of 105 areas available.
03 DFH.UNIT	Bit	12	Contains the port, channel, flag indicator, and the electronics unit of this disk unit.
04 DFH.PORT	Bit	3	Contains the port identification.
04 DFH.CHAN	Bit	4	Contains the channel identification.
04 DFH.SER.NO.FLAG	Bit	1	Denotes whether the DFH.ADDR is the actual data address or the pack serial number of a multiple pack file.
04 DFH.EU	Bit	4	If this file is a multiple pack, DFH.EH contains either a @C@ for a continuation or @B@ for a base pack. If this is not a multiple pack file, DFH.EU contains the disk drive EU number.
03 DFH.ADDR	Bit	24	Contains either the disk pack serial number or the address of the disk file header area on the disk.

DISK LABEL

Each pack or cartridge, whether system or removable, contains a label describing its physical characteristics. Sectors 0 to 63 are reserved for the disk label and various MCP tables, and are not used for any other purpose. A disk label is constructed at the time of initialization, gathering data from two sources:

a. The information provided by the Disk Initializer control card.

b. The information provided by the Disk Initializer program itself.

Table 4-3 describes the disk label format.

Table 4-3. Disk Pack Label

Field Name	Type	Length	Description
01 PACK.LABEL	Char	180	
02 PL.VOL1	Char	4	Contains the literal "VOL1".
02 PL.SERIAL.NO	Char	6	Contains the disk pack serial number supplied by the control card.
02 PL.ACCESS.CODE	Char	1	Not implemented
02 PL.ID	Char	17	Contains the disk pack identifiers.
03 PL.NAME	Char	10	Contains the user-assigned name of the disk pack that is on the control card.
03 PL.FILLER	Char	7	Used for label compatibility with other electronic units.
02 PL.SYSTEM.INTERCHANGE	Char	2	Enables the pack to be formatted in such a way as to make it compatible and interchangeable with other Burroughs Systems. (Not implemented.)
02 PL.CODE	Char	1	Contains a 0 when the pack is a "scratch pack." NOTE: A "scratch pack" is defined as a pack that has been purged or initialized. A pack that simply has had its files removed is not considered a "scratch pack."
02 PL.FILLER	Char	6	Reserved.
02 PL.OWNER.ID	Char	14	Contains the disk pack name supplied by initialization control card.
02 PL.TYPE	Char	1	Contains the usage code of the disk pack: R = Restricted S = System U = Unrestricted
02 PL.CONTINUE	Char	1	Contains a "C" if this is a continuation pack.

Table 4-3. Disk Pack Label (Cont)

Field Name	Type	Length	Description
02 PL.FILLER	Char	26	Reserved.
02 PL.INT	Char	1	Always contains a blank.
02 PL.VOL2	Char	4	Contains the literal "VOL2".
02 PL.DATE.INITIALIZED	Char	5	Contains the date the pack was initialized, in Julian format, as supplied by the initialization control card.
02 PL.INIT.SYSTEM	Char	6	Contains the literal 17MC33.
02 PL.DISK.DIRECTORY	Char	8	Contains the decimal disk address of the Disk Directory.
02 PL.MASTER.AVAIL	Char	8	Contains the decimal disk address of the Master Available table.
02 PL.DISK.AVAILABLE	Char	8	Contains the decimal disk address of the Working Available table.
02 PL.INTEGRITY	Char	1	Indicates either no files are open, or one or more files are open.
02 PL.ERROR.COUNT	Char	6	Contains a counter containing the number of read, write or parity errors. The counter is imcremented by 1 each time the MCP attempts to recover from an error. The MCP attempts 20 read/writes before informing the operator of the error. This count is displayed on the Console Printer each time the pack is mounted and made Ready. If the count increases at a rapid rate, the pack and drive should be inspected.
02 PL.SECTORS.XD	Char	6	Contains the number of sectors removed by an "XD" console input message.
02 PL.TEMP.TABLE	Char	8	Contains the decimal disk address of the Temporary Available table.
02 PL.PCD	Char	3	Contains the port, channel, and drive on which this pack is mounted.

Table 4-3. Disk Pack Label (Cont)

Field Name	Type	Length	Description
02 PL.ASSIGNED.TO.BPS	Char	6	Contains the serial number of the base pack to which this pack is assigned.
02 PL.FILLER	Char	30	Reserved.

PACK INFORMATION TABLE

When a disk pack is on-line, the MCP builds a Pack Information table describing its characteristics and maintains its current status while the disk pack is on the system. Table 4-4 describes the Pack Information table format.

Table 4-4. Pack Information Table

Field Name	Type	Length	Description
01 PACK.INFOR	Char	373	
02 P.NAME	Char	10	Contains the disk pack name assigned at disk initialization.
02 P.SERIAL.NO	Bit	24	Contains the disk pack serial number assigned at disk initialization.
02 P.DISK.DIRECTORY	Bit	36	Contains the disk address of the Disk Directory.
02 P.DISK.AVAILABLE	Bit	36	Contains the disk address of the Disk Available table.
02 P.TEMP.TABLE	Bit	36	Contains the disk address of the Temporary Available table.
02 P.UNIT.NAME	Bit	6	Contains the EU mnemonic identifiers of the pack residence. For example, DPA or DPB.
02 P.PCD	Bit	12	Contains the hardware location of the pack.
03 P.PORT.CHAN	Bit	7	Contains the port and channel location of the pack.
03 FILLER	Bit	1	Reserved.
03 P.DRIVE.NO	Bit	4	Contains the drive number of the pack.
02 P.NO.USERS	Bit	8	Contains the number of users currently accessing this pack.

Table 4-4. Pack Information Table (Cont)

Field Name	Type	Length	Description
02 P.NO.MPF.USERS	Bit	8	Contains the total number of multiple pack files open on this pack.
02 P.TO.BE.POWERED.DOWN	Bit	1	Informs the MCP that the pack is to be powered down.
02 P.RESTRICTED	Bit	3	Identifies the type of pack for the MCP: 0 = System resource pack 1 = Restricted 2 = Unrestricted User 3 = Interchange
02 P.CONTINUE	Bit	1	Indicates this is a continuation pack for a multiple pack file.
02 P.SCRATCH	Bit	1	Indicates this is a scratch pack.
02 P.FULL	Bit	1	Indicates no disk space is currently available.
02 FILLER	Bit	1	Reserved.
02 P.ASSIGNED.TO.BPS	Bit	24	Contains the base pack serial number of a continuation pack.
02 P.BACK.LINK	Bit	24	Points to the previous Pack Information Table entry (backward link).
02 P.LINK	Bit	24	Points to the next Pack Information Table entry (forward link).

DISK FILE ALLOCATION

The allocation of disk space is the responsibility of the MCP. When a request is made for space, the MCP searches the Working Available table until an area is found that is equal to or greater than the space required. There are times when contiguous space is not available. The MCP then notifies the system operator of the insufficient space. This requires operator intervention either to provide the space required or to discontinue the program. The MCP cannot assign two or more non-contiguous areas to a space request.

FILE LOOK-UP

When searching for a file, the MCP uses the sequence illustrated in figure 4-5.

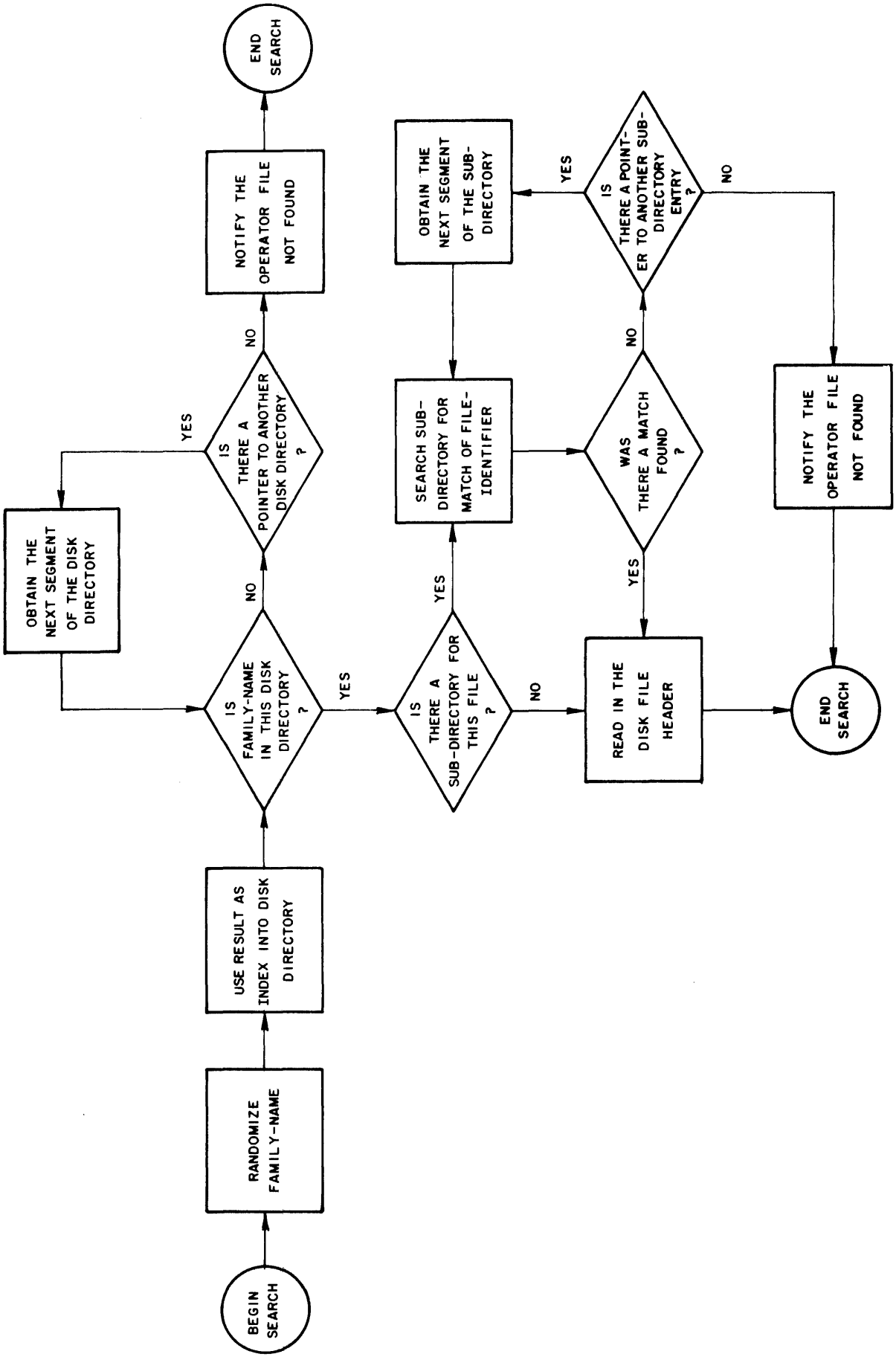


Figure 4-5. Disk File Search

DISK FILE CONSTRUCTION

Files on disk are constructed and addressed either sequentially or randomly.

Sequential File

In a sequential file, the records of data are stored on the disk in a consecutive manner based upon a sequentially controlled program key. These records are made available to a program sequentially from record to record until End-of-File. For example, to locate the 992nd record in a 1000-record file, all preceding 991 records must be read and bypassed prior to accessing the 992nd record.

Random File

The random file is a storage technique by which the program must only pass the relative record number desired, referred to as the key, to the MCP. The MCP then takes the key and, by calculation, arrives at the disk address. This method allows the selection of any record in the file by a single read/write operation.

AVAILABLE TABLES

The MCP maintains three tables which reference all available disk space.

a. Master Available table.

b. Working Available table.

c. Temporary Available table.

Master Available Table

The Master Available table, constructed at disk initialization, reflects all areas of the disk that can be utilized for the storing of data. The following criteria determine the physical acceptability of the pack transition to be used on the system.

a. Addresses 0 through 63 must be free of read/write errors.

b. The maximum number of read/write errors for addresses greater than 63 cannot be exceed 71.

The Master Available table consists of three contiguous disk segments. It references and maintains the total available disk area in disk address sequence.

The Master Available table for head-per-track is created by COLDSTART.

Working Available Table

The Working Available table (referred to as the Available table) contains the addresses and sizes of all disk areas not allocated either to a user program or to the MCP. The Working Available table is in the same format as the Master Available table, having a length of 10 segments and expanded when necessary. When entries are made in the Disk Directory, the space reflected in the disk file header is removed from the Working Available table. Note that the entries in the Disk Directory plus those in the Working Available table can total an amount less than the total amount of disk space specified in the Master Available table. This occurs when the MCP temporarily obtains disk space without creating a Disk Directory entry.

Temporary Available Table

The Temporary Available table references those areas on disk that are being used but are not entered in the Disk Directory or the Working Available table. Areas that are "temporary" are those that are currently being used by a program or the MCP. When a file is made permanent, the space is removed

from the Temporary Available table and entered into the Disk Directory. Some examples of the disk space appearing in the Temporary Available table, but not in the Disk Directory or the Working Available table, are the following:

a. Space allocated for a new disk file.

b. Program roll-out areas.

c. Program work files.

The Temporary Available table uses five segments, expanded when necessary. New entries are placed in the first available position. Table 4-5 describes the Master, Working, and Temporary Available table format.

Table 4-5. Master, Working, and Temporary Available Tables

Field Name	Type	Length	Description
01 DISK.AVAILABLE.TABLE	Bit	1440	
02 AVL.FOR.LINK	Bit	36	Contains the disk address of the next available segment. If this is the last entry in the table, this field contains zeros (forward link).
02 AVL.BACK.LINK	Bit	36	Contains the disk address of the previous entry in the table. When this is the first entry in the table, this field contains zeros (backward link).
02 AVL.SELF	Bit	36	Contains the disk address of this entry.
02 FILLER	Bit	4	Reserved.
02 AVL.BLOCK	Bit	60	Reserved.
03 AVL.ADDRESS	Bit	36	Contains the disk address of the next available disk area.
03 AVL.LENGTH	Bit	24	Contains the length of the area which AVL.ADDRESS references. There is a maximum of 22 entries per segment.

MULTIPLE PACK FILES

A multiple pack file is a file that may be contained on one or more removable disk packs. The file originates on a "base" pack and can continue to a maximum of 15 packs. These packs are called "continuation" packs. A multiple pack file can have only one base pack, and must be on-line for all opens and closes performed on the file.

The base pack can contain either single and multiple pack type files but cannot contain continuation files associated with another base pack. The disk pack header on the base pack retains all information concerning the file, which includes the addresses that are assigned to that file and the serial number of each continuation pack.

When a multiple pack file requires additional space to store data, the MCP searches for another continuation pack associated with the same base pack. If a continuation pack is not available, the MCP searches for a scratch pack of the same processing type (restricted, unrestricted) as the base pack. If one is available, that pack is assigned as a continuation pack for that file.

Multiple Pack File Information Table

When an input multiple pack file is opened, the file header is read into memory from the base pack; with output files, the header is constructed by the MCP in memory. The MCP then obtains space on the system pack for a Multiple Pack File Information table, containing the information associated with that multiple pack file. The table contains base pack information, and a copy of the disk file header. Table 4-6 describes the Multiple Pack File Information table format.

Table 4-6. Multiple Pack File Information Table

Field Name	Type	Length	Description
01 MPF.INFO.TABLE	Bit	1392	
02 MPF.FORWARD	Bit	36	Points to the next multiple pack file entry.
02 MPF.BACKWARD	Bit	36	Points to the previous multiple pack file entry.
02 MPF.SELF	Bit	36	Points to this multiple pack file entry.
02 MPF.NAME	Char	30	Contains the multiple pack file-identifier.
02 MPF.HEADER.SIZE	Bit	24	Contains the size of composite file header.
02 MPF.HEADER.ADDRESS	Bit	24	Points to the composite file header in memory.
02 MPF.BPS.NO	Bit	24	Contains the base pack serial number.
02 MPF.OPEN.TYPE	Bit	4	Contains the type of file opened. (Same as the DFH.OPEN.TYPE in the Disk File Header table.)
02 MPF.NEW.FILE	Bit	1	Indicates that this is a newly created disk file.
02 MPF.NEW.AREA	Bit	1	Indicates that this is a newly created disk area.

Table 4-6. Multiple Pack File Information Table (Cont)

Field Name	Type	Length	Description
02 MPF.CS	Bit	1	Indicates that a Clear/Start was or was not performed since this entry was created.
02 FILLER	Bit	1	
02 MPF.BASE.PACK.TYPE	Bit	4	Contains the processing pack type. 1 = Restricted 2 = Unrestricted
02 MPF.ARRAY			Used to record all continuation packs that are on-line.
03 MPF.ONLINE			Contains a maximum of 16 items in this array.
04 MPF.SERIAL.NO	Bit	24	Contains the serial number of the pack.
04 MPF.HDR.DSK	Bit	36	Contains the disk address of the disk file header.

MCP Composite Header

The MCP builds a composite header in memory for each multiple pack file opened. Each address in the header contains one of two items:

a. If the pack is on-line, it contains the absolute disk address of that area.

b. If the pack is off-line, it contains the serial number of the pack on which the area resides.

cut along dotted line

cut along dotted line

BURROUGHS CORPORATION
DATA PROCESSING PUBLICATIONS
REMARKS FORM

TITLE: B 1700 MASTER CONTROL PROGRAM (MCP) Reference Manual

FORM: 1088010
DATE: 8-75

CHECK TYPE OF SUGGESTION:

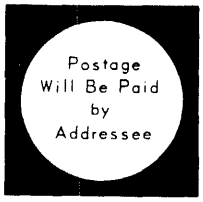
☐ ADDITION ☐ DELETION ☐ REVISION ☐ ERROR

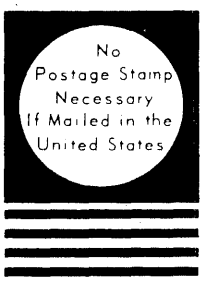
GENERAL COMMENTS AND/OR SUGGESTIONS FOR IMPROVEMENT OF PUBLICATION:

FROM: NAME _____ DATE _____
TITLE _____
COMPANY _____
ADDRESS _____
_____

STAPLE

FOLD DOWN SECOND FOLD DOWN





BUSINESS REPLY MAIL
First Class Permit No. 1009; El Monte, CA. 91731

Burroughs Corporation
P. O. Box 142
El Monte, CA. 91734

attn: Publications Department
Technical Information Organization, TIO – West

FOLD UP FIRST FOLD UP

1088010 8-75 Printed in U.S. America