Burroughs

B 5500

Information
Processing Systems

MASTER CONTROL PROGRAM
REFERENCE MANUAL

Burroughs

B 5500

INFORMATION PROCESSING SYSTEMS

MASTER CONTROL PROGRAM

REFERENCE MANUAL

Burroughs Corporation
Detroit, Michigan 48232

Printed in U. S. America 6-69 1042462

Correspondence regarding this document should be forwarded using the Remarks Form at the back of the manual, or may be addressed directly to Systems Documentation, Sales Technical Services, Burroughs Corporation, 6071 Second Avenue, Detroit, Michigan 48232.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

INTRODUCTION

Frequently, a lack of adequate information results in much time being spent resolving software problems. This manual is an aid in solving these problems. It includes assorted items of information about the Master Control Program (MCP) Tables, segments, links, words, and related material, which, together with a listing of the MCP, will make it easier to define software problems.

Specifically, this reference manual is divided into nine sections, which focus on the MCP Tables, the Disk, MCP classification and Organization of Core Storage, Logging, MCP Operational Tables, Binary Cards, Library Maintenance, and Interrupt Handling. In addition, there are four appendices which supplement the text.

SECTION 1
MASTER CONTROL PROGRAM TABLES

GENERAL.

An MCP is a modular supervisory computer program which takes over repetitive functions, some being logically complex, to make computer programmers and operations more productive and efficient. The MCP provides the overall coordination and control processing that is so important to total production through the maximum use of all B 5500 components. Operator intervention is nearly eliminated because complete management of the system is assumed by the MCP, a comprehensive operating system that provides simultaneous Input/Output (I/O) operations and multiprocessing. By controlling the sequence of processing, initiating all I/O operations, and providing automatic handling procedures to meet virtually all processing conditions, the MCP can obtain maximum use of the system components at all times. Since so many functions are performed under this centralized control, changes in schedule, system configuration, and program sizes can be readily accommodated. Thus, greater overall production and efficiency is achieved.

All versions of MCP handle the primary functions of control programs: loading, interrupts, I/O control, selection and initiation of program I/O error conditions, system log, storage allocation, overlay, and multiprogramming. The MCP is composed of tables (i.e., arrays) and of procedures with an outer block which coordinates their operation. Section 1 focuses on the tables which compose the MCP.

PROGRAM REFERENCE TABLE (PRT) FOR THE MCP.

The PRT contains the locations reserved for variables, data descriptors, and program descriptors which give information about data arrays and other program information. These locations are likely to change in future MCP's. Brackets [] indicate a descriptor. Otherwise, the variable is an operand.

Word	Contents
RRRMECH	Mask word used by STATUS to check I/O devices.
[SLATE]	Descriptor pointing to SLATE array.
NSLATE	Pointer to last entry which was started from SLATE.
LSLATE	Pointer to last entry placed in the SLATE.
AVAIL	Contains the address of the stopper for available storage links; its value is the highest available address -1.
MSTART	Contains the address of the first area of storage after the end of the MCP SAVE Procedures and the outer block code.
MEND	Pointer to last storage link in memory.
TOGLE	Word containing the following mask bits: HP2TOG, STATUSBIT, SHEETFREE, STACKUSE, STOREDY, USERSPACEREADY, HOLDFREE, NSECONDREADY, ABORTABLE, BUMPTUTIME, KEYBOARDREADY, NOBACKTALK, QTRDY, INTFREE, SPOENDULLOG, REMOTELOGFREE.
[BED]	Descriptor pointing to BED array.
P1MIX	Mix index of normal state job for which work is being done by Processor 1 in either normal or control state.
P2MIX	Mix index of the normal state job for which work is being done by Processor 2 in normal state.
DATE	Contains current date (YYDDD -- BCL).
CLOCK	Contains the number of time interval interrupts processed since Halt/Load (H/L) multiplied by 64.
XCLOCK	External clock which is set by system operator and tells the time of day (maintained in 60ths of a second).

Word	Contents
READY	Contains the contents of the ready register on the last read.
[PRT]	Descriptor pointing to PRT array.
[JAR]	Descriptor pointing to Jobs Actually Running (JAR) array.
[INTRNSC]	Descriptor pointing to the INTRNSC array.
INTSIZE	Used to determine row size for each mix index in INTABLE.
[INTABLE]	Descriptor pointing to the INTABLE array.
[SHEET]	Descriptor pointing to SHEET array.
JOBNUM	Pointer to the last entry in the BED. The number of entries in the BED is equal to JOBNUM (in decimal) DIV 2 + 1.
[PRYOR]	Descriptor pointing to the PRYOR array, which is a table containing priorities for each mix index.
NOPROCESSTOG	<0 if normal state processing is allowed.
[NFO]	Descriptor pointing to the NFO array.
[ISTACK]	Descriptor pointing to the independent stack.
[PROCTIME]	Descriptor pointing to the PROCTIME array. PROCTIME[1] contains processor time for job with mix index = 1.
[IOTIME]	Descriptor pointing to the IOTIME array. IOTIME [1] contains I/O time for job with mix index = 1.
[CHANNEL]	Descriptor pointing to the CHANNEL array. CHANNEL[1] contains logical unit of last descriptor sent out on channel 1.
[FINALQUE]	Descriptor pointing to the FINALQUE array.
[LOCATQUE]	Descriptor pointing to the LOCATQUE array.

Word	Contents
IOQUEAVAIL	Pointer to the first available space in IOQUE.
[IOQUE]	Descriptor pointing to the IOQUE array.
[UNIT]	Descriptor pointing to the UNIT array.
[TINU]	Descriptor pointing to the TINU array.
[WAITQUE]	A QUEUE of units for which there are I/O requests but no I/O channel is available.
NEXTWAIT	Pointer into WAITQUE at next available slot.
FIRSTWAIT	Pointer at next unit to be used when a channel becomes available.
[LABELTABLE]	Descriptor pointing to the LABELTABLE array.
[MULTITABLE]	Descriptor pointing to the MULTITABLE array.
[RDCTABLE]	Descriptor pointing to the RDCTABLE array.
[PRNTABLE]	Descriptor pointing to the PRNTABLE array.
ILL	The head of the queue through which all data communication output passes.
INQCT	Counter of unprocessed data communications interrupts.
PINGO	Used to link tanked MCP input messages together.
READQ	The head of the queue of all "sought" terminal/buffers.
RRNCOUNT	Count of Read-Ready-Normal data communications interrupts.
[TRANSACTION]	Descriptor pointing to the TRANSACTION array.
LEFTOFF	Used by OLAY for overlaying core.
MESSAGEHOLDER	Used by SPOUT and MESSAGEWRITER to link SPO messages.
BYPASS	Used by ENTERUSERFILE to locate the ends of the regular and bypass directories.

Word	Contents
NEXTSLOT	Variable used to indicate the next available position for file entry in the disk directory.
DISKBOTTOM	Variable set to the highest address of the directory.
[DBARRAY]	Descriptor pointing to the DBARRAY. The array used with the DB feature of the DEBUGGING option.
DBADR	Variable used to contain the disk address for the DB feature.
DIRECTORYFREE	Variable used to interlock the directory.
IOMASK	Variable used as a mask to sleep until the complete I/O action is finished.
SAVEWORD	Used to indicate which "in-use" device is to be saved.
CORE	Used by SELECTRUN to determine if a job should be introduced into the mix.
KEYBOARDCOUNTER	Count of unprocessed keyboard requests.
NUMESS	Initialized to -100; counts +1 for each SPOUT message waiting to go to the SPO.
STATIONMESSAGEHOLDER	Beginning of link-list of messages waiting to go to remotes.
[STATION]	Descriptor pointing to the STATION array.
[FS]	Descriptor pointing to the FS array.
TUMAX	Number of columns in the STATION array.
[ATTACHED]	Descriptor pointing to the ATTACHED array.
[USERCODE]	Descriptor pointing to the USERCODE array.
LOOKQ	Used to link user codes, masks, and times for remotes.

Word	Contents
[UNITCODE]	Descriptor pointing to the UNITCODE array.
MCP	Variable containing the user identification of the privileged user.
MIXMASK	Mask for legal input mix messages from remotes.
INFOMASK1 INFOMASK2	Masks for legal input (from remotes) keyboard messages not requiring a mix index.
CCMASK1 CCMASK2	Masks for legal control card reserved words from remote stations.
OPTION	Contains the OPTION word.
[USERDISK]	Descriptor pointing to the USERDISK array.

OPTION WORD.

Word stored in MCP PRT to set and reset options. The OPTION word is also stored as the first word in DIRECTORYTOP. The OPTION word can be set or reset via the COLD START Routine or via the keyboard.

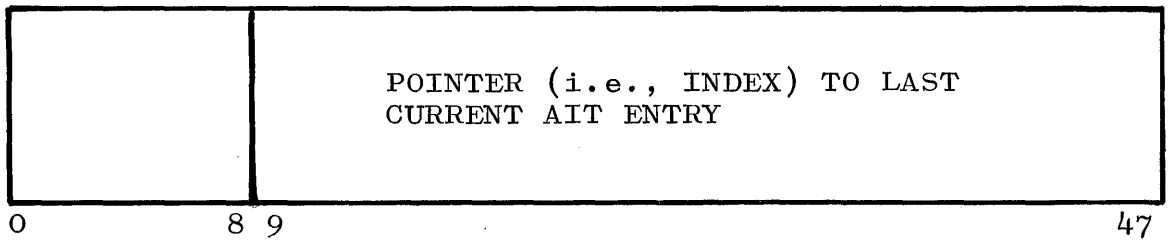
Option	Field	Keyboard Mnemonic
USEDRA	[47:1]	DRA
USEDRB	[46:1]	DRB
BOJMESS	[45:1]	BOJ
EOJMESS	[44:1]	EOJ
OPNMESS	[43:1]	OPEN
TERMGO	[42:1]	TERMNATE
GIVEDATE	[41:1]	DATE
GIVETIME	[40:1]	TIME
SAMEBREAKTAPE	[39:1]	ONEBREAK
AUTOPRINT	[38:1]	AUTOPRNT
CLEARWRS	[37:1]	CLEARWRS
DISCONDC or NOTIFYOP	[36:1]	DISCONDC
COPNMESS	[35:1]	COMPLFILE
CLOSEMESS	[34:1]	CLOSE
None	[33:1]	ERRORMSG
RETMSG	[32:1]	RET

Option	Field	Keyboard Mnemonic
LIBMSG	[31:1]	LIBMSG
SCHEDMSG	[30:1]	SCHEDMSG
SECMSG	[29:1]	SECMSG
DSKTOG	[28:1]	DSKTOG
RELTOG	[27:1]	RELTOG
PBDREL	[26:1]	PBDREL
CHECKLINK	[25:1]	CHECK
DSKMSG	[24:1]	DISKMSG
DKLOG	[23:1]	DISKLOG (TSS only)
LIBERR	[22:1]	LIBERR (TSS only)
USEPBD	[21:1]	PBDONLY
SVPBT	[20:1]	SAVEPBT
MOD3IOS	[2:1]	(Cannot be accessed through keyboard.)

ARRAY INFORMATION TABLE (AIT).

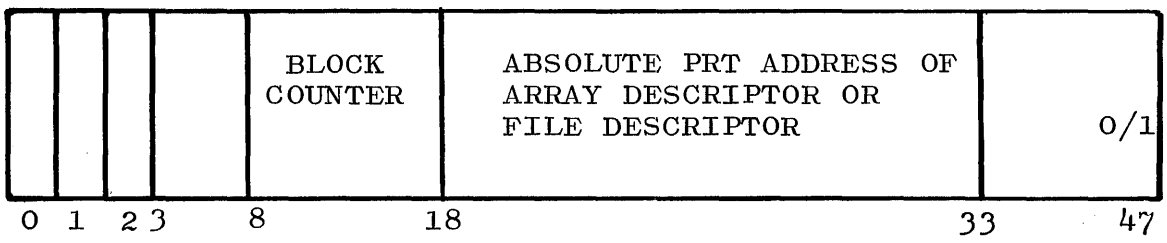
One AIT is associated with each program.

WORD 0



Field	Contents
[0:9]	0
[19:39]	Index to last current AIT entry. (INTEGER).

REMAINING WORDS



Field	Contents
[0:1]	Flag bit.
[1:2]	0 = array. 1 = run time error entry. 2 = file.
[3:5]	Number of dimensions.
[8:10]	Block counter (i.e., nesting depth when file or array is declared).
[18:15]	Absolute address of file or array descriptor or of Run Time Errors* (RTE) cell containing label word.
[33:15]	Save indicator (1=SAVE) for arrays or for RTE.

LABEL WORD

The operand constructed by GOTOSOLVER.

Field	Contents
[CF]	R-relative address of the label descriptor.
[FF]	Proper F Register setting. If 0, then outer block.
[8:10]	Block Counter setting.

The operand is usually contained in the stack or in the PRT (for RTE). It is used to represent action labels or formal labels.

STATION TABLE FORMAT.

The Station Table Format is a Dump debugging aid. It is formatted as follows:

Field	Contents
[0:1]	Flag bit (off).
[1:1]	Output in process by STATIONMESSAGEWRITER.

*Error type:

1 = integer overflow.
2 = exponent overflow.
4 = invalid index.
8 = divide by zero.
16 = flag bit.

Field	Contents
[2:1]	SPO Console input request flag (Bit 32 should be on also).
[3:1]	Not used.
[4:4]	TU index into STATION for next control station. If not a control station, its own index.
[8:1]	Not used.
*[9:4]	TU address for this word.
*[13:1]	DTCU Translator bypassed: Translate=1, omit translate=0. (Translation: ASCII to BCL or Baudot to BCL.)
*[14:4]	Buffer address for this word.
[18:4]	Buffer index into STATION for next control station. If not a control station, its own index.
[22:1]	Station busy.
*[23:1]	Adapter sensed "abnormal" condition.
*[24:1]	Read-Ready Buffer.
*[25:1]	Group mark or IFAL ending: Group mark=0, IFAL=1.
[26:1]	Break.
*[27:1]	Write ready: 1=Write without group mark ending. (Additional write required to clear buffer.) 0=Group mark finish write.
*[28:1]	Input error.
[29:1]	Write in-process.
*[30:1]	Station not ready.
[31:1]	Mix messages not desired flag (1=no mix message, 0=output mix messages.)
[32:1]	SPO Console flag. (When this bit is on, all input is treated as if it had originated at the SPO.)

*Indicates hardware-defined fields.

Field	Contents
[33:1]	Not used.
[34:1]	Message Delete action required.
[35:3]	Not used.
[38:5]	Exclusive user's mix index. (=31 if station is a SPO Console)
[43:1]	Tanked input.
[44:1]	Tanked MCP input being entered.
[45:1]	Station assigned to a job.
[46:1]	Station logged-in.
[47:1]	Not used.

SECTION 2

THE DISK

GENERAL.

A disk file or system memory is a prerequisite to the use of the MCP. The disk file is used by the MCP as an auxiliary storage area. Therefore, it is necessary to be acquainted with its organization.

Disk storage is divided into two categories: system disk and user disk. System disk is the disk area reserved for:

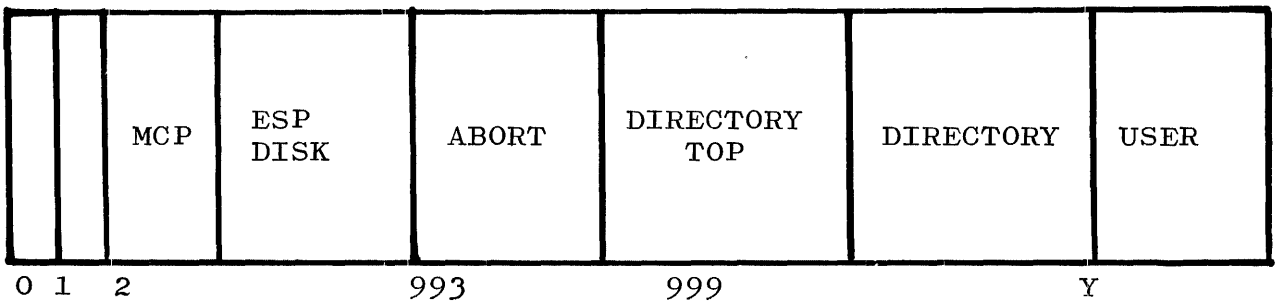
a. The DF MCP program and tables.
b. The disk directory.
c. The available-disk list, and other DF MCP uses.

User disk is the area used for remaining facilities. Data files, scratch files, and library programs, including the B 5500 problem oriented compilers, may be stored in the user disk area. This section presents that disk information which is applicable to the B 5500 MCP Tables.

DISK LAYOUT.

An area on disk to be used for a particular file must be explicitly reserved for that file. A program must specify the amount of disk required for a file. The DF MCP allows a single file to occupy from one to twenty separate areas on disk. The number of areas and their size is specified by the program that creates the file. The fact that a file is stored in more than one area does not in any way affect the way it is referenced by a program. Regardless of the number of areas used, a program always addresses a file as though it were one continuous string of records.

The disk layout is dealt with in the following manner:



Segment	Contents
0	Not used.
1	Copy of H/L Button Card.
MCP starts at segment 2	Storage of operating system.
ESPDISK	Used by the MCP for scratch pad.
ABORT	Location of the Abort Table (Segments 993 => 998).
DIRECTORYTOP	Contains parameters for MCP. Segment 999 on disk.

Word	Field	Contents
0		OPTION word.
1		DATE (in BCL).
2		Number of electronic storage units.
4		Highest address of directory (Y).
5		Last number used for control deck.
6		First control deck queued (location in directory).

Segment		Contents
DIRECTORYTOP (cont)		
Word	Field	Contents
7		Last control deck queued (location in directory).
8		Next number available for printer backup disk.
9		Multiprocessing core factor.
10 => 15		Specify which data communications stations are similar to the SPO.
	[10].[0:16]	Not used.
	.[16:16]	TU 1, buffers 0 => 15.
	.[32:16]	TU 2, buffers 0 => 15.
	[11].[0:16]	TU 3, buffers 0 => 15.
	.[16:16]	TU 4, buffers 0 => 15.
	.[32:16]	TU 5, buffers 0 => 15.
	[12].[0:16]	TU 6, buffers 0 => 15.
	.[16:16]	TU 7, buffers 0 => 15.
	.[32:16]	TU 8, buffers 0 => 15.
	[13].[0:16]	TU 9, buffers 0 => 15.
	.[16:16]	TU 10, buffers 0 => 15.
	.[32:16]	TU 11, buffers 0 => 15.
	[14].[0:16]	TU 12, buffers 0 => 15.
	.[16:16]	TU 13, buffers 0 => 15.
	.[32:16]	TU 14, buffers 0 => 15.
	[15].[0:16]	TU 15, buffers 0 => 15.
	.[16:32]	Not used.
		Entries are made in words 10 => 15 by the procedure MARKSPOSTA.

Segment	Contents
DIRECTORYTOP (cont)	

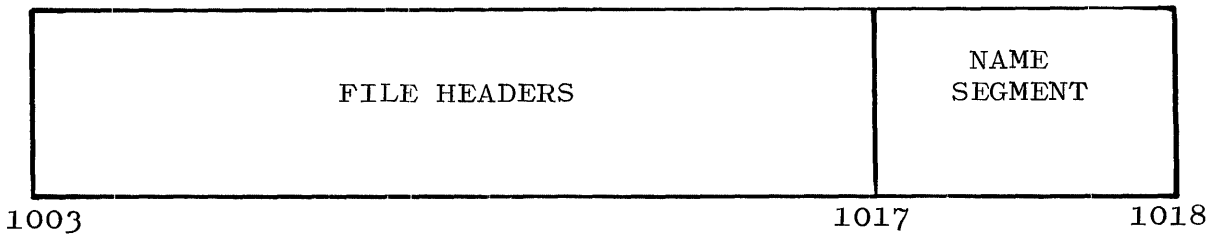
Word	Field	Contents
		The Boolean variable HOLDFREE is used to interlock the DIRECTORYTOP.
16		Q value for data communications input.
17 => 28		Reserved for expansion.
29		Contains USERDISKSL during restarts and breakouts.

DIRECTORY	Area used by MCP to maintain directory of entire in-use disk.
USER	Area used for storage of system log, compilers, and all other user files.

DISK DIRECTORY.

The MCP maintains, on disk, a disk directory which provides information about all permanent files on the disk. Each directory section is composed of 16 segments which contain information for up to 15 files. These sections are allocated as needed in the disk directory area specified by the user.

The 16th segment in a section contains the names (i.e., file identification) of each file defined in that section. In the first name position, @114 indicates last entry, and @14 means available entry. The preceding 15 segments are referred to as file headers.



NAME SEGMENT

Contains up to 15 pairs of names for each file defined in this section. The names are in the same order as the files in the preceding 15 segments. There are two words for each file header.

FILE HEADERS

Word	Field	Contents
0	[0:15]	Record length.
	[15:15]	Block length.
	[30:12]	Records per block.
	[42:6]	Segments per block.
1		Row length in segments.
2		File creator or user code. = 0 Free file. > 0 Public file, sole user file, private file. < 0 Security file.
3	[1:1]	If 1, then file is new format.*
	[2:10]	Save factor (binary).
	[12:18]	Date of last access (binary).
	[30:18]	Creation date (binary).
4	[1:1]	= 1 if file interlock.
	[2:7]	Used by library maintenance.
	[9:1]	If 1, then file is new format.
	[10:1]	If 1, then file is a program.
	[11:25]	Reserved for expansion - bits set to zero.
	[36:6]	File type (used by Time Sharing System). = 0, Unknown = 1, BASIC = 2, ALGOL = 3, COBOL

*Effective with Mark VIII, change 41, page 3, Systems Note No. 229, June 21, 1968.

Word	Field	Contents
4 (cont)		= 4, FORTRAN = 5, TSPOL = 6, XALGOL = 7, SEQ = 8, DATA = 9, LOCK (Security).
	[42:6]	Open count.
5	[1:1]	= 1, PRIVATE file.
	[1:48]	= 0, SOLE user file or SECURITY file (see H[2].[1:1]). = 12, INFO file if H[6] = 12, and H[2] ≠ 0; PUBLIC file if H[6] = 0 and H[2] = 0; FREE file if H[6] = 12 and H[2] = 0.
6	[6:42]	File ID if SECURITY file (see H[2] and H[5]).
7		Number of logical records (EOF pointer).
8		Number of segments per row, as specified in file declaration.
9		Number of rows, as specified in file declaration.
10 through 29		Binary disk addresses of rows (zero if not assigned).

The layout of the disk below the directory is presented in figure 2-1.

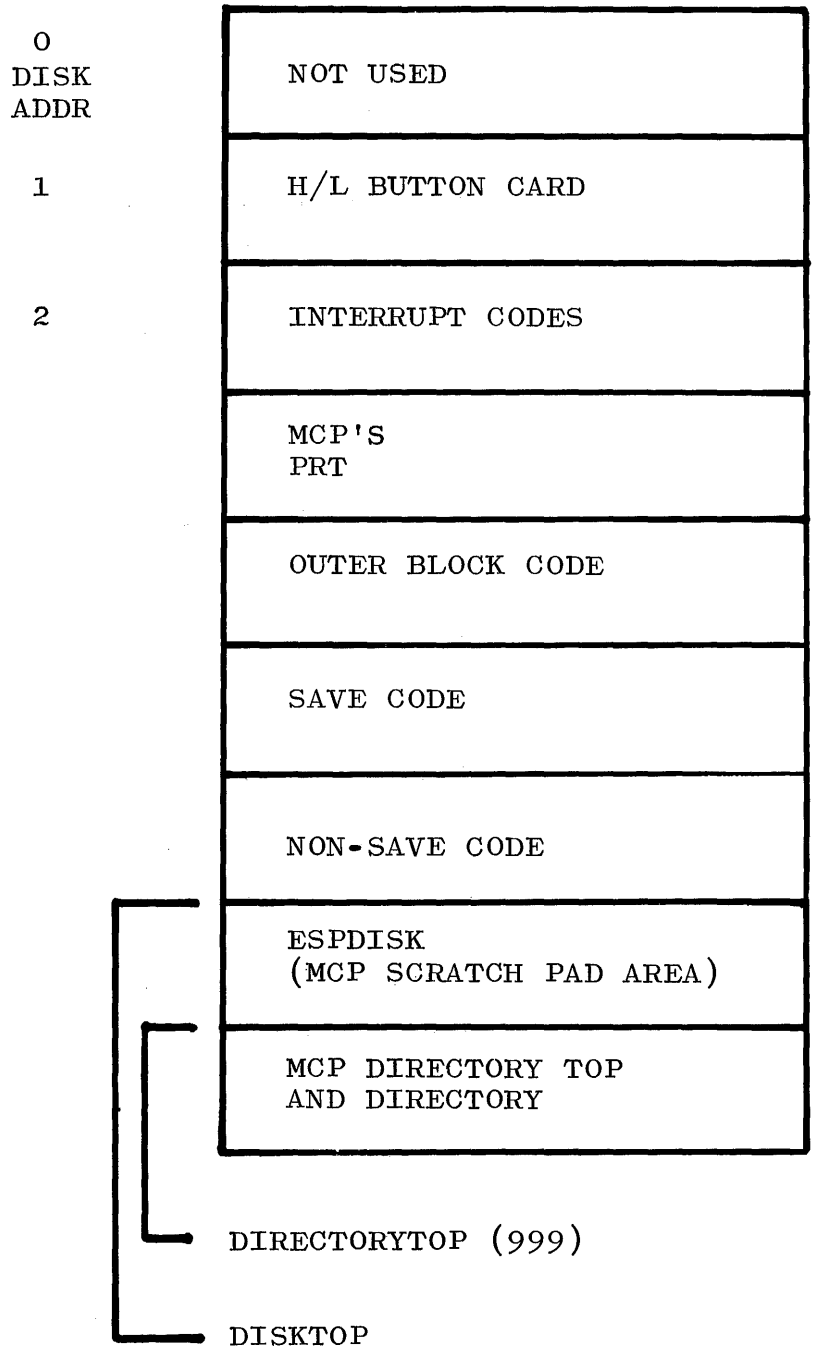


Figure 2-1. Layout of Disk Below the Directory

The MCP scratch pad area begins at the first segment beyond the last segment used for non-save code. Initially, ESPDISKAVAIL is set = 0, and DISKTOP is set to the address of the first usable segment in the scratch pad area. DISKTOP will contain the address of the first usable segment in the contiguous scratch pad area. If DISKTOP + 1 ever becomes equal to the address of DIRECTORYTOP, then a PUNT message of NO MCP DISK will be initiated and the system will halt.

NOTE

GETESPDISK will return the address of an available segment in the MCP scratch area as an operand.

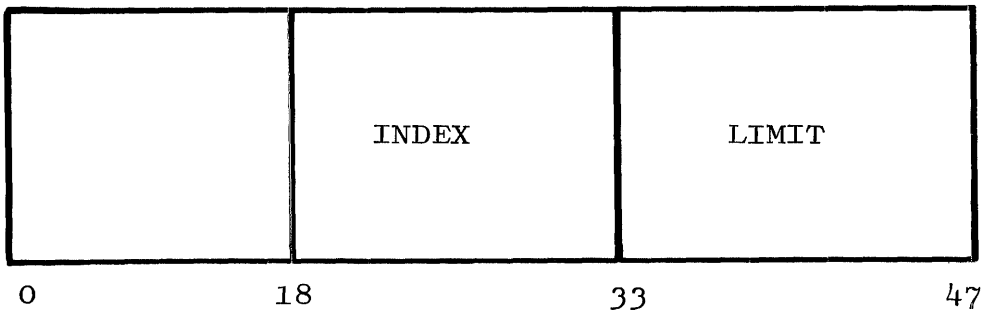
FORGETESPDISK (segment address) will return the address specified by "segment address", which must be an operand, to an available scratch area.

DALOC.

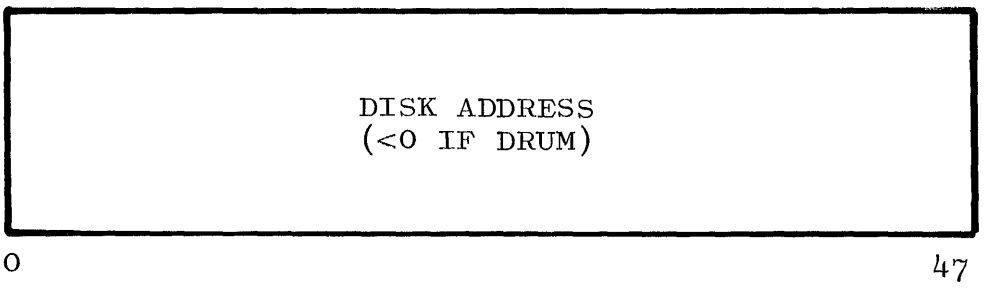
DALOC is a two dimensional array used to manage overlay storage both on drums and disk. It has MIXMAX + 1 rows, each of which is initially nine words long, and is expandable as required.

DALOC INX P1MIX points to the DALOC Row for P1MIX which has the following construction:

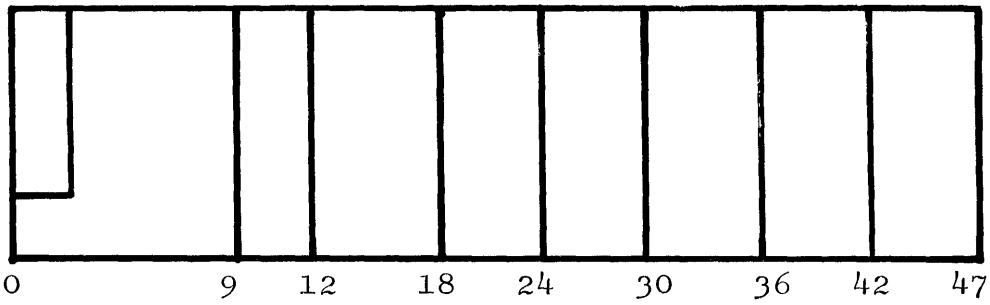
WORD 0



WORD 1



WORD 2



NOTE

Words 3 and 4, 5 and 6, 7 and 8, etc. are identical in construction and use to words 1 and 2.

Word 0	Field	Contents
INDEX	[18:15]	Pointer to the first evenly numbered word (2, 4, 6, etc.) which may be used to locate some overlay storage.
LIMIT	[33:15]	Pointer to the last (largest) evenly numbered word which is being used for this P1MIX.

Word 1

Contains the disk address of the base of a 500 segment section of overlay storage. One such section is made available at SELECT RUN time. If this word is negative, the overlay storage referenced is on the drum.

Word 2	Field	Contents
	[2:7]	Contains the next relative address available within the sub-section indicated by the following field (9:3).
	[9:3]	Indicates which of the following sub-sections (100 segments each) is active (0 through 4).
	[18:6]	Sub-section number 1.
	[24:6]	Sub-section number 2.
	[30:6]	Sub-section number 3.
	[36:6]	Sub-section number 4.
	[42:6]	Sub-section number 5.
		Each sub-section controls 100 segments of overlay storage. The number in each field indicates the number of times the system has allocated space from the applicable 100 segments of a sub-section.

When an area referenced by a descriptor has been overlaid, bits 33:6 of the descriptor contain a value used to locate the odd-numbered word in the DALOC Row for this mix index, which contains the base disk address of the 500 segment section in which the information has been placed. Bits 39:9 of the descriptor contain the offset, which, when added to the base, gives the absolute disk address of the information.

When a previously overlaid area is made present again, these two fields are transferred to the F field of the descriptor. This will assure that subsequent overlays of this data will return to the same place on disk.

If we say DESC is defined as the descriptor, then the disk address to

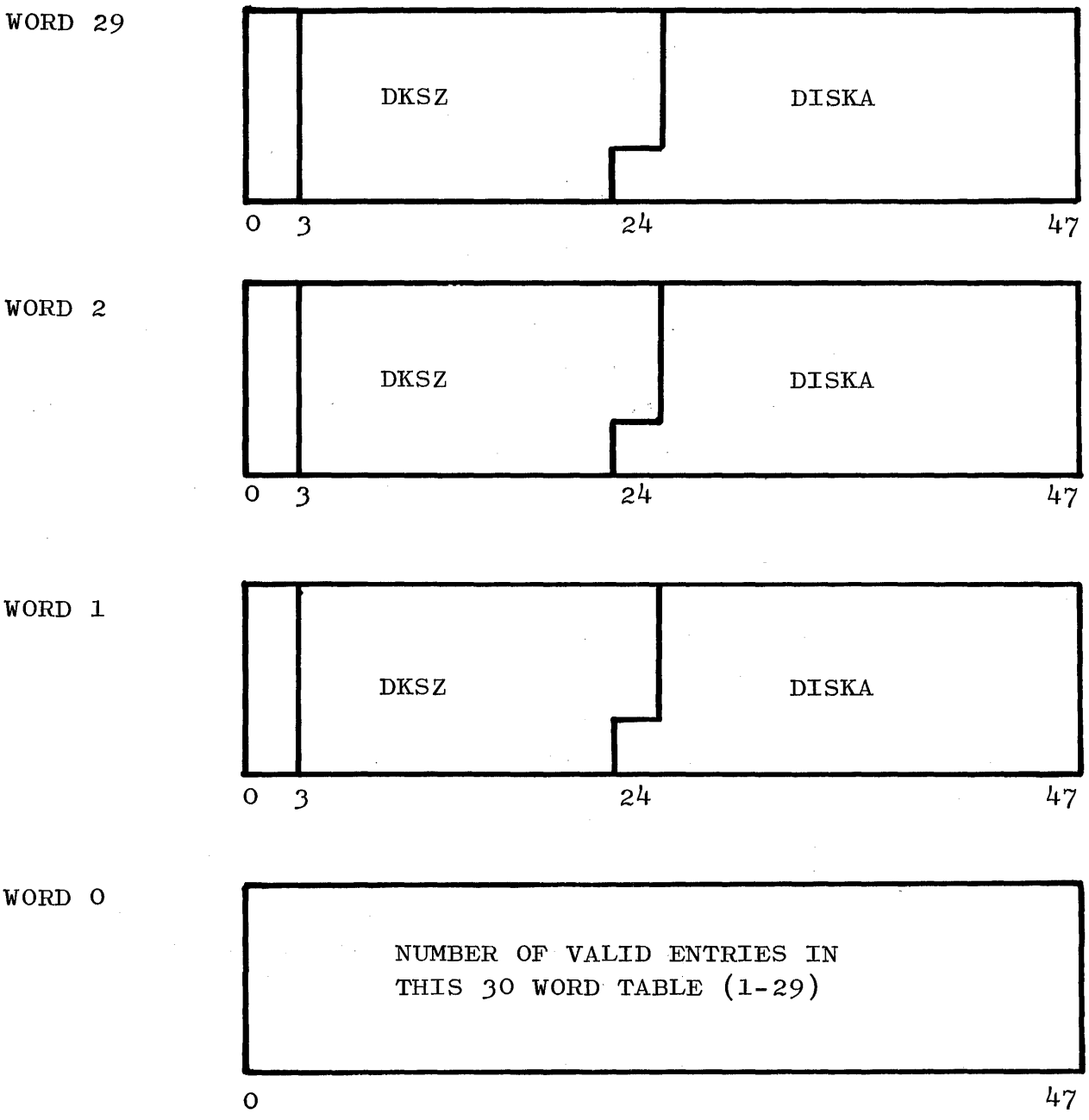
which we must re-overlay is calculated as

DALOC[P1MIX,DESC.[33:6]x2-1]+DESC.[39:9]

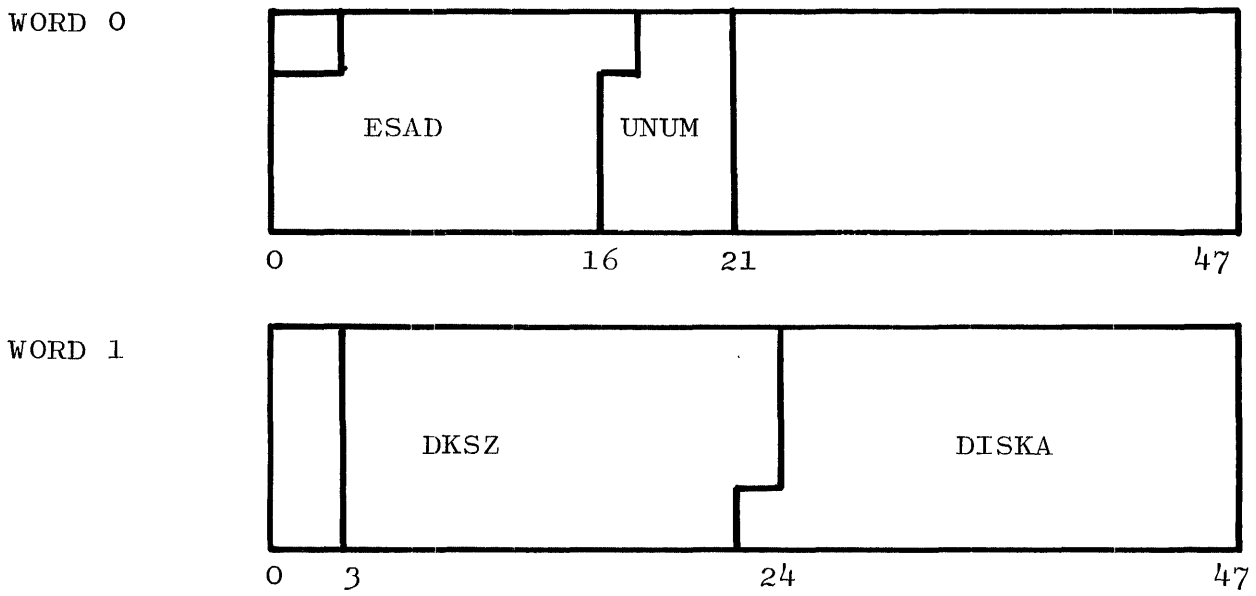
AVAILABLE-DISK TABLE.

The Available-Disk Table is a two part, Fine/Coarse Table. While the Fine Table is maintained on disk in ESPDISK, the Coarse Table is kept in core memory.

The format of the Fine Table in ESPDISK is as follows:



The format of the Coarse Table in core memory is as follows:



The Coarse Table is expandable in two word increments, as additional Fine Table entries are required.

Word	Field	Contents
DISKA	[23:25]	Absolute disk address of a section of available disk.
DKSZ	[3:20]	Size of the available area starting at DISKA.

In the Coarse Table, there are copies of the entry for the largest area in the associated Fine Table.

ESAD	[1:15]	Address of the associated Fine Table in ESPDISK.
UNUM	[16:5]	Number of valid entries in the Fine Table. UNUM must agree with word 0 of the Fine Table.

SECTION 3
MCP CLASSIFICATION AND ORGANIZATION OF CORE STORAGE

GENERAL.

Certainly, if core storage is to be put to use efficiently, it must be classified and organized. Basically, storage is organized through the use of memory links.

The MCP classifies core areas containing information which must remain in place as non-overlayable storage. For example, the MCP has routines and tables that must frequently be used when handling interrupt conditions and other control functions. The space that would be momentarily gained by overlaying such information would not be worth the time required to make the information present when needed again.

There is also a need for certain kinds of object program information to remain in fixed locations while a program is being processed. This requirement holds for all information which will be referenced by the MCP through the use of absolute addresses; for example, control fields which contain absolute addresses of program segments.

Overlayable storage refers to information in core storage that must be present when needed. It is often the case that all information pertaining to a program cannot be in core at the same time. This is most often the case when programming for operating systems with less than maximum core. However, the majority of the information related to object programs, and most information in the MCP, may be used relatively infrequently. With respect to such information, the major factor determining its necessity to be present in core is that it must be present when needed.

Since the B 5500 programs are stored on disk during the time they are processing, individual program segments are read into core as they are needed. If the area used by the program segment is to be overlaid, there is an exact copy of it on disk. The MCP has only to mark the segment absent in appropriate places, and the area it

occupied can be used for other segments. If the segment is needed again, it can be read into core from disk.

Available storage is storage currently not in use. Such storage can be assigned as needed. Section 3 deals with the memory links which are used by the MCP.

MEMORY LINKS.

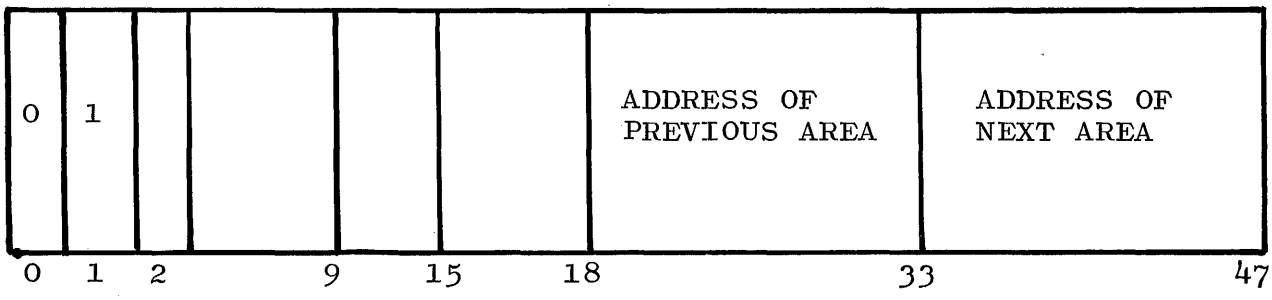
Memory links are used by the MCP to keep track of the assignment of core areas. There is an available memory link in every unassigned area. A memory link for available storage occupies three words. These words provide the following information:

a. They specify that the area is available.
b. They specify the size of the area.
c. They provide the address of the following available area.

When core storage is classified and organized for the first time after a H/L, the MCP performs operations to determine what memory modules are available on the system in a contiguous area from memory address 0. Links are set up so that the areas in those modules which are not present are never assigned, and, consequently, never addressed. Permanent MCP program segments related to initialization routines may be in core after initialization, but they are overlayable; all other core is marked available.

Figures 3-1, 3-2, and 3-3 present information necessary to deal successfully with memory links.

WORD 1

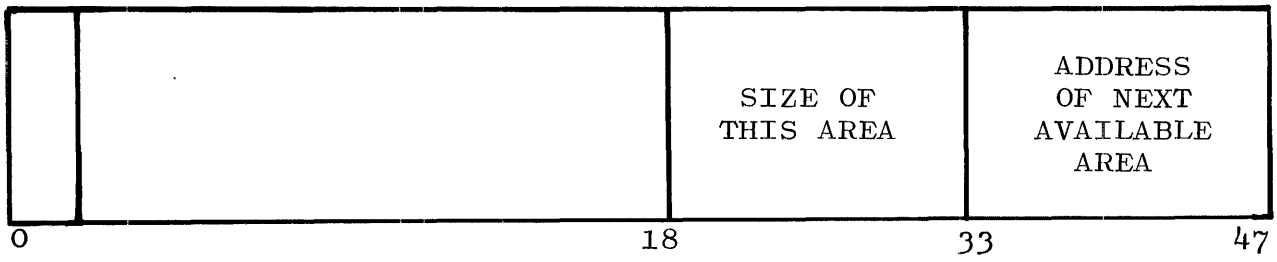


Field	Field Value	Contents
[0:1]	0	Flag bit.
[1:1]	1	Availability.
[2:1]	NA*	
[3:6]	NA	
[9:6]	NA	
[15:3]	NA	
[18:15]	Address	Address of first word of link for previous area.
[33:15]	Address	Address of first word of link for next area.

*NA=Not Applicable

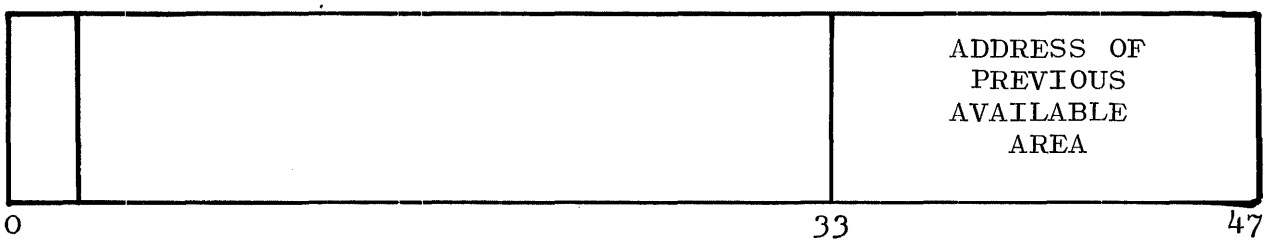
Figure 3-1. Links for Available Area (3 Words) Field Values and Contents (Sheet 1 of 2)

WORD 2



Field	Field Value	Contents
[0:18]	0	Zeros required by LLL operator.
[18:15]	Varies	Size of available area.
[33:15]	Address	Address of second word in link for next available area.

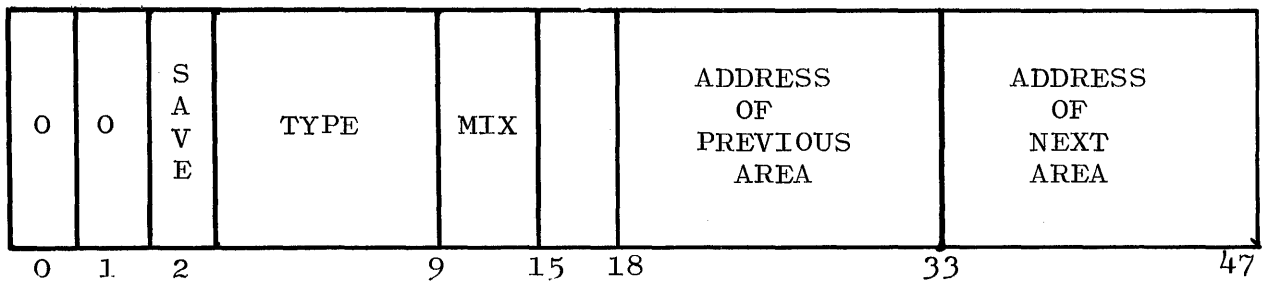
WORD 3



Field	Field Value	Contents
[0:33]	0	None.
[33:15]	Address	Address of second word in link for previous available area.

Figure 3-1. Links for Available Area (3 Words) Field Values and Contents (Sheet 2 of 2)

WORD 1

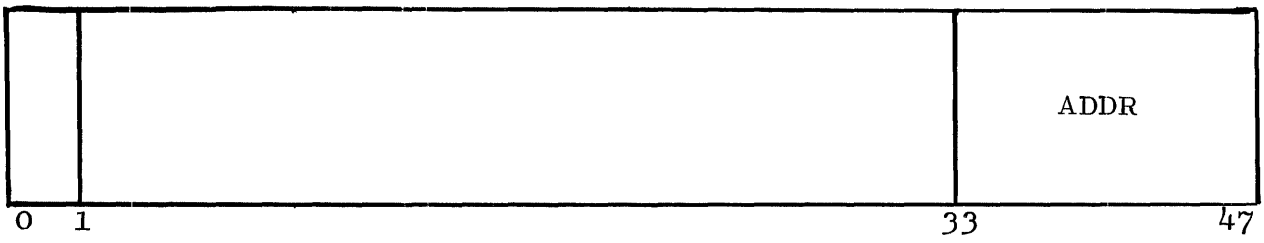


Field	Field Value	Contents
[0:1]	0	Flag bit.
[1:1]	0	Availability.
[2:1]	0/1	Save (1 = save area).
[3:6]	0-7	Type of area: 0 = miscellaneous data. 1 = program. 2 = data. 3 = I/O buffers. 4 = ALGOL FIB. 5 = MCP data communications area. 6 = COBOL FIB. 7 = intrinsic segment.
[9:6]	0 - MIXMAX	Mix index of program using area. Intrinsics and MCP always have MIX field = 0. Re-entrant program code has the mix of the first job.
[15:3]	NA	None.

Figure 3-2. Link for In-Use Area (2 Words) Field Values and Contents (Sheet 1 of 2)

Field	Field Value	Contents
[18:15]	Address	Address of first word of link for previous area.
[33:15]	Address	Address of first word for next area.

WORD 2



Field	Field Value	Contents
[0:33]	Varies	Type 2 data-overlay address on disk. Size of program segments and intrinsics.
[33:15]	Address	If data or file tank, address of array descriptor. If object program, segment number. If it is a data communications' buffer area, READQ or ILL link word. If MCP program segment, PRT address. If intrinsic, segment number for the job which first made it present.

Figure 3-2. Link for In-Use Area (2 Words) Field Values and Contents (Sheet 2 of 2)

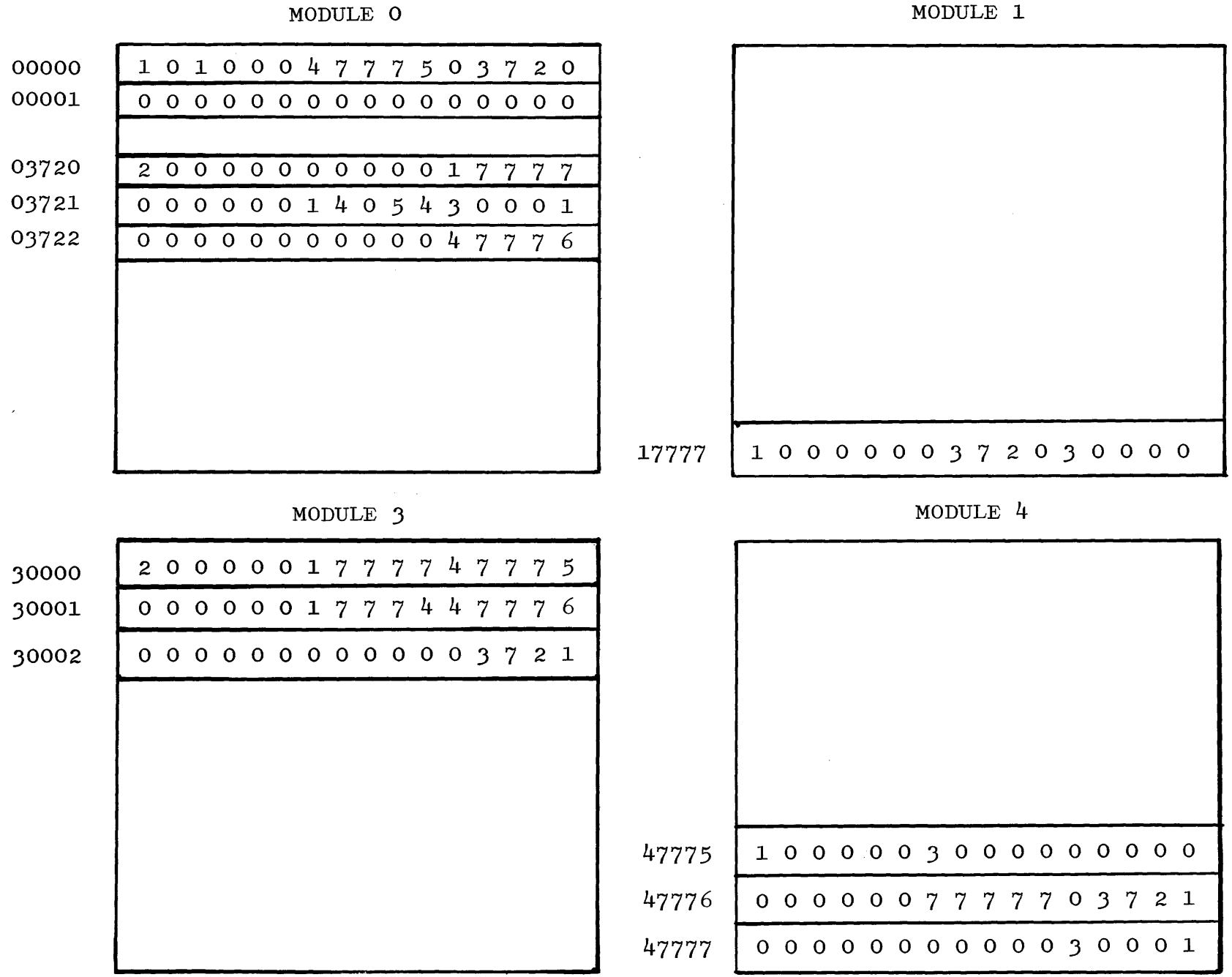


Figure 3-3. Core Memory at H/L Time: Modules 0, 1, 3, and 4 On Line

SECTION 4
LOGGING

GENERAL.

MCP maintains a computer log recording the system time and other information concerning the program. The maintenance of the log is performed automatically by the MCP. Records are written on the log file in the order in which the information becomes available. The log information is written in a file on user disk called SYSTEM/LOG.

The first record in the log is used by the MCP. The value of the first field in this record specifies the number of records written in the log. The value of the second field specifies the record capacity of the log. The third and fourth fields are used in conjunction with the warning messages supplied by the MCP which specify when the log is half full or full. The fifth field contains the word "DSKLOG."

The MCP writes several types of records for every job. These are:

a. BOJ and EOJ records.
b. File records.
c. Idle time and H/L records.

A program to print the log is provided. However, any program can read the log file. Consequently, each installation can provide its own log printing program, and format the output as desired. Section 4 is a detailed presentation of the logging procedures and formats for the B 5500 Systems.

ABORT TABLE.

The Abort Table is kept by the MCP to log-off abort jobs. It is used by NSECOND in termination. The Table is located at DIRECTORYTOP - 6.

Word	Contents
0	XCLOCK

Word	Contents
1	DATE
2	"ABORT"

The next three entries are repeated for each job in the mix. If the mix number is not assigned, the entries are zeroed.

Relative Location	Contents
3xMIX[1]	Process time.
3xMIX + 1	I/O time.
3xMIX + 2	IDLETIME (from the JAR).
3xMIX + 90	First name of object program.
3xMIX + 91	Second name of object program.
3xMIX + 92	[1:23] start time, and [24:24] pointer to location of control card in ESPDISK to be written into the SYSTEM/LOG.

FORMAT OF THE SYSTEM/LOG.

A program to print the log is provided by Burroughs Corporation. However, any program can read the log file. Consequently, each installation can provide its own log printing program, and format the output as desired.

SYSTEM/LOG SPECIFICATIONS.

Log information for programs run on a B 5500 System is written in a file on user disk. The log file occupies one area on disk, and has the ⟨file identification prefix⟩ SYSTEM and the ⟨file identification⟩ LOG. It is the user's responsibility to provide this file.

The file SYSTEM/LOG is blocked. There are six logical records per physical record. The logical records are five words (i.e., 40 characters) in length; the physical records are 30 words in length.

1. Mix index.

LOG ENTRY SPECIFICATIONS.

Entries in the log can be considered to fall into one of four categories:

a. Compile-and-Go entries,
b. Compile-Only entries,
c. Execute entries, and
d. Disk log entries.

With respect to these categories, the following rules determine how a program is entered in the log.

a. If a Compile-and-Go run is made and the program being compiled contains no syntax errors, the log information for both the compiler and the object program is listed in a Compile-and-Go entry.

b. If a Compile-and-Go run is made and the program being compiled contains syntax errors, or if a Compile-for-Syntax run is made, or if a Compile-to-Library is made, the log information for the compiler is listed in a Compile-Only entry.

c. If an Execute run (i.e., library call-out) is made, the log information for the object program is listed in an Execute entry.

d. If the DISKLOG compile time option for the MCP is set, disk files will be logged at the time the files are removed from the disk (e.g., after a CC REMOVE) under the following conditions:

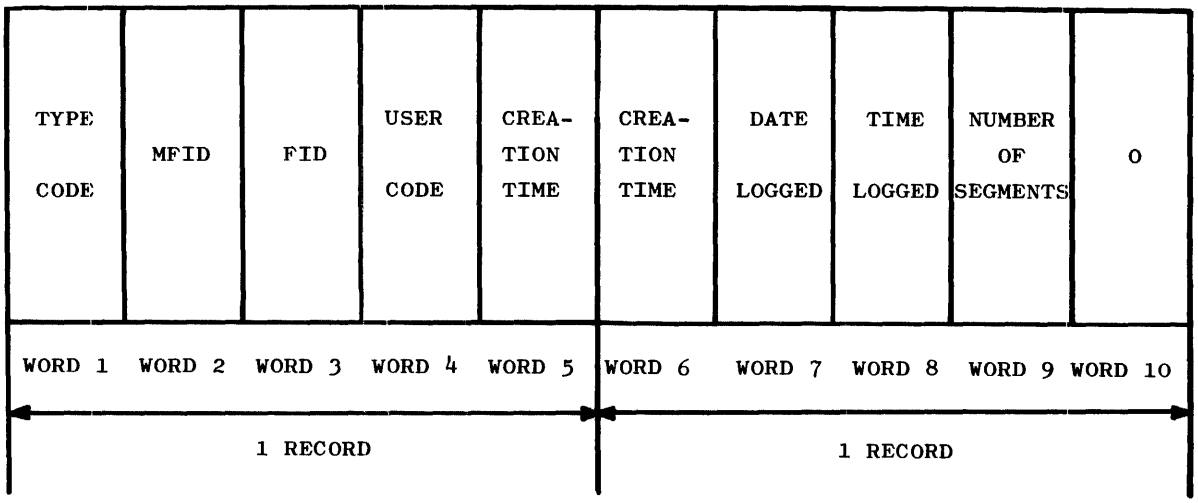
   1) When a scratch file is CLOSEd.

   2) When a file is CLOSEd after obtaining more space.

   3) When a file is LOADed from a library tape with the same ⟨multi-file identification⟩ / ⟨file identification⟩ as a file on disk.

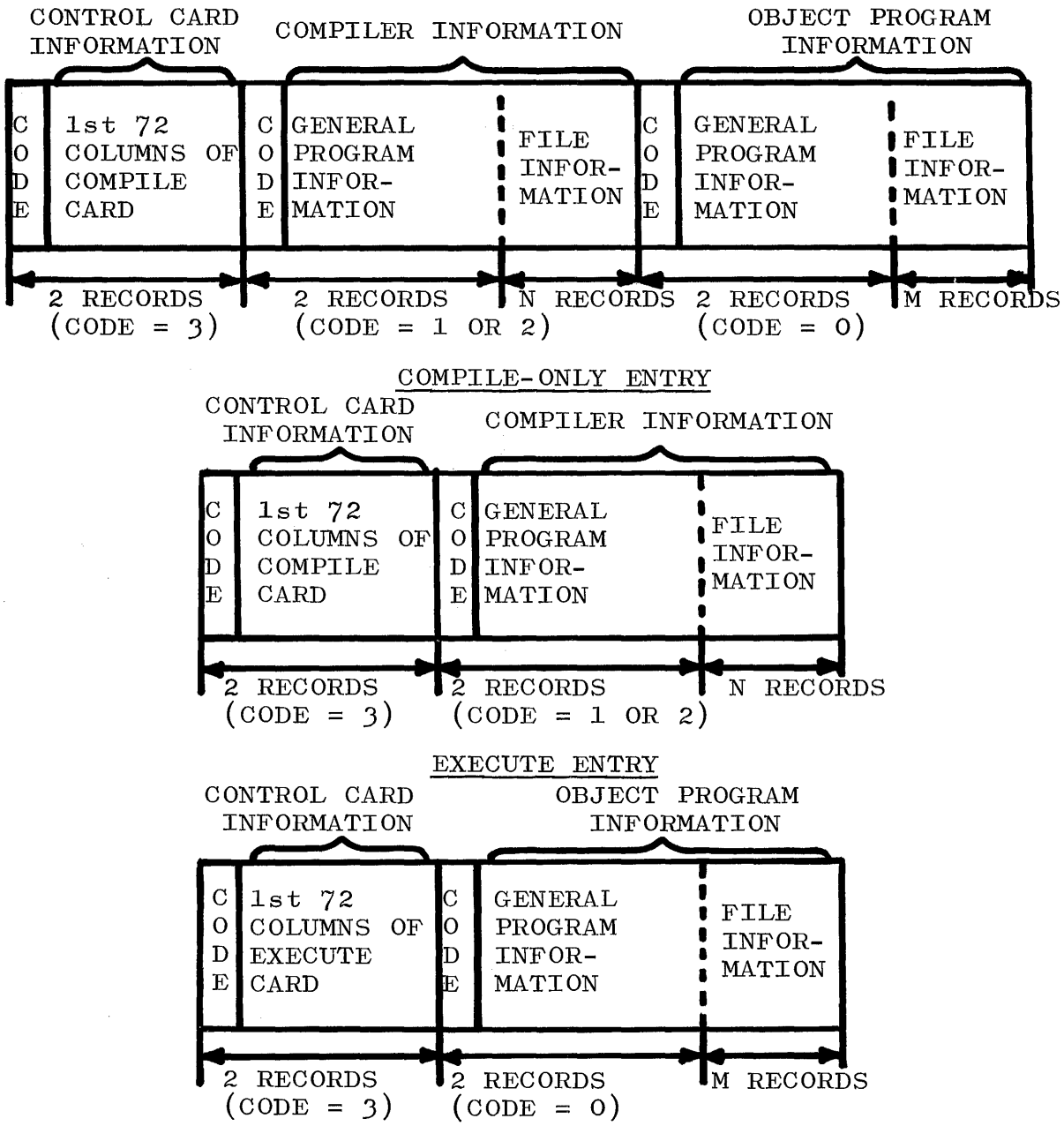
4) When the operator enters a log-out istruction, LNDK. The LNDK message logs out all disk files and resets their creation date (H[3].[30:18]) and the creation time (H[1].[25:23]).

Figures 4-1 and 4-2 present the disk file and the general format for log entries. The first log entry starts in the record with relative address 1.



Entry	Description
Type Code	=8
MFID	First Name of File (7 CHR)
FID	Second Name of File (7 CHR)
User Code	(7 CHR)
Creation Date	In Form YYDDD (BCD)
Creation Time	1/60 Second (OCT)
Date Logged	In Form YYDDD (BCD)
Time Logged	1/60 Second (OCT)
Number of Segments	(OCT)
0	Reserved for Expansion

Figure 4-1. Disk File Log Entry



NOTE

N = Number of files declared by compiler.

M = Number of files declared by object program.

Figure 4-2. General Format of each of the Three Types of Log Entries

CODE WORD.
As shown in Figure 4-2, each log entry contains:

a. Control card information, and
b. Compiler and/or object program information.

The code word preceding each group of information denotes the type of information. Information preceded by a 1 pertains to the ALGOL Compiler; information preceded by a 2 pertains to the COBOL Compiler; and information preceded by a 0 pertains to an object program. Code 4 denotes the end-of-log information.

CONTROL CARD INFORMATION.
Control card information is contained in the first two records of a log entry, starting at the second word of the first record. This information is a copy of the contents of the first 72 columns of the COMPILE Card or EXECUTE Card that caused the particular run to be scheduled.

The word immediately preceding the control card information is a code with the integer value 3.

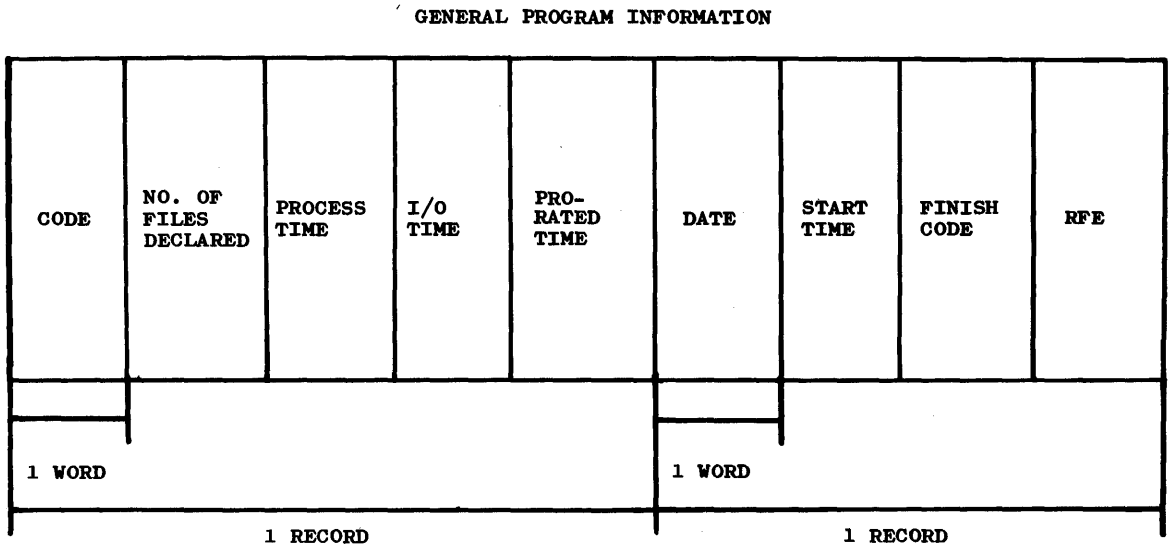
COMPILER AND OBJECT PROGRAM INFORMATION.
Compiler information and object program information have identical formats. Therefore, the format for this information will be discussed under the general name "program information".

Program information falls into two categories:

a. General program information, and
b. File information.

The general program information is contained in two records. The file information requires a variable number of records, depending upon the number of files declared by the program. There is one record required in the log for each file declared by the program. Each record of file information, however, has the same format.

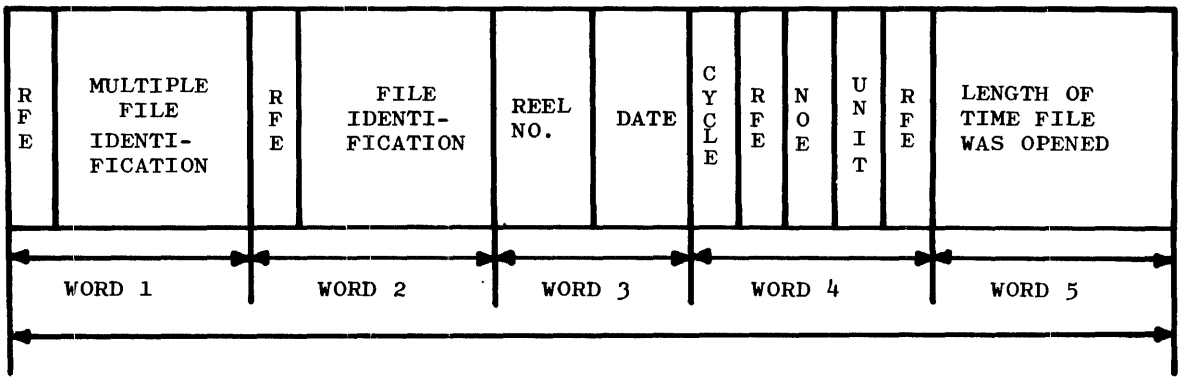
Figure 4-3 shows the format of general program information. Figure 4-4 shows the format of one record of file information.



Entry	Description
CODE	INTEGER: 1 = ALGOL, 2 = COBOL, 3 = obj. prog., 5 = printer backup, 6 = FORTRAN, 8 = disk log
NO. OF FILES OPENED	INTEGER
PROCESS TIME	INTEGER: Time in 60ths of a second.
I/O TIME	INTEGER: Time in 60ths of a second.
DATE	BCL: YYDDD format* (e.g., 65046).
START TIME	INTEGER: 60ths of a second since H/L time.
STOP TIME	INTEGER: 60ths of a second since H/L time.
FINISH CODE	INTEGER: 0 = EOJ, 1 = SYNTAX ERROR, 2 = DS-ed.
RFE	(Reserved for Expansion).

*The YYDDD format provides that the YY characters specify the last two digits of the year, and the DDD characters specify the number of the day of the year.

Figure 4-3. Format of General Program Information in a Log Entry, including the Code Word.



Entry	Description	
MULTIPLE FILE IDENTIFICATION	BCL:	Located in second through eighth characters of WORD 1.
FILE IDENTIFICATION	BCL:	Located in second through eighth characters of WORD 2.
REEL NO.	BCL:	Located in first through third characters of WORD 3.
DATE	BCL:	Located in fourth through eighth characters of WORD 3.
CYCLE	BCL:	Located in first and second characters of WORD 4.
NOE (Number of errors while handling file)	BINARY:	Located in fifth and sixth characters of WORD 4.
UNIT	BINARY:	Located in seventh character of WORD 4. (See list below for meanings of the values of UNIT.)
LENGTH OF TIME FILE WAS OPENED	INTEGER:	Time in 60th of a second.
RFE	(Reserved for expansion.)	

Figure 4-4. Format of One File Information Record

The values of UNIT specify what unit was used by the subject file. The values are now defined.

Value	I/O Unit
0	Not opened
1	MTA
2	MTB
3	MTC
4	MTD
5	MTE
6	MTF
7	MTH
8	MTJ
9	MTK
10	MTL
11	MTM
12	MTN
13	MTP
14	MTR
15	MTS
16	MTT
17	DRA
18	DRB
19	DKA
20	DKB
21	LPA
22	LPB
23	CPA
24	CRA
25	CRB
26	SPO
27	PPA
28	PRA
29	PPB
30	PRB
31	DCA

SPECIAL RECORDS AND LOG INITIALIZATION.
Additional information concerning log maintenance for the MCP includes the following:

RECORD ZERO. The first record in SYSTEM/LOG (i.e., the record with relative address 0) is used by the MCP when making log entries. The value of the first word in record zero specifies the number of records written in the log. The value of the second word specifies the record capacity of the log. The third and fourth words are used in conjunction with the warning messages supplied by the MCP which signify when the log is half-full and full. The fifth word contains, in BCL, "DISKLOG".

RECORD N + 1. The first word of the record immediately following the last log entry contains a code with the value 4. This record denotes the end-of-log information, and it is not included in the value contained in the first word record of record zero.

INITIALIZING THE LOG. If a user program wishes to initialize the log (i.e., set up the log so that the MCP considers the log empty), the following actions are performed:

a. The first, third, and fourth words in record zero must be set to zero.

b. The first word in record 1 must be set to 4.

FORMAT OF THE REMOTE/LOG.
The topic of formatting the Remote Log is dealt with by giving the necessary specifications for the log. The log entry specifications are presented as six types. In addition, the two part partition of the file REMOTE/LOG is discussed with the file maintenance procedures and with the WR keyboard input message.

REMOTE LOG SPECIFICATIONS.
The remote log information for the data communications' facilities is written in a file on the user disk. The file has the ⟨file identification prefix⟩ REMOTE and the ⟨file identification⟩ LOG.

The file REMOTE/LOG is blocked and is confined to one area on the disk. There are five logical records per physical record. A logical record is five words in length or 40 characters; a physical record is 30 words in length. It is the user's responsibility to provide this file. Logging for data communications is bypassed if the system does not provide a REMOTE/LOG file.

LOG ENTRY SPECIFICATIONS.
Entries in the Remote Log are of six types:

Type 1	Log-out entry.
Type 2	Log-in entry.
Type 3	Control card entry of less than 32 characters.
Type 4	Control card entry of 32 characters or more, but not greater than 72 characters.
Type 5	Job statistics entry.
Type 6	Abort information entry.

Types 1, 2, and 3 each require one logical record in the log.
Types 4, 5, and 6 require two logical records per entry.

TYPE 1 LOG-OUT ENTRY.
The following information is entered into the file REMOTE/LOG when a data communications' station logs out.

1 Record

Word 0	[9:9]	Station number ([9:4]=TU,[14:4]=BUF).
	[42:6]	Code = 1.
Word 1		User identification (as specified by the File Security System).
Word 2		Current date (YYDDD-BCL).
Word 4		Unused.

TYPE 2 LOG-IN ENTRY.

The MCP enters the following information in the file REMOTE/LOG when a data communications' station logs in.

1 Record

Word 0	[9:9]	Station number ([9:4]=TU,[14:4]=BUF).
	[42:6]	Code = 2.
Word 1		User identification (as specified by the File Security System).
Word 2		Current date (YYDDD-BCL).
Word 3		Time of day at log-in.
Word 4		Unused.

TYPE 3 CONTROL CARD ENTRY (31 CHARACTERS OR LESS).

The MCP enters the following information (or type 4 information) in the file REMOTE/LOG when a job is selected to run. Every RUN or EXECUTE from a remote station is logged.

1 Record

Word 0	[9:9]	Station number ([9:4]=TU,[14:4]=BUF).
	[18:24]	RUN NUMBER*.
	[42:6]	Code = 3.
Word 1 through Word 4		Contents of control card.

TYPE 4 CONTROL CARD ENTRY (32 CHARACTERS AND UP TO 72 CHARACTERS).

The MCP enters the following information (or type 3 information) in the file REMOTE/LOG when a job is selected to run. Every RUN or EXECUTE from a remote station is logged.

2 Records

Word 0	[9:9]	Station number ([9:4]=TU,[14:4]=BUF).
	[18:24]	RUN NUMBER*.
	[42:6]	Code = 4.
Word 1 through Word 9		Contents of control card.

*Entries in the file REMOTE/LOG corresponding to entries in the file SYSTEM/LOG have the same RUN NUMBER, where a job's RUN NUMBER is defined to be its start time (in 60ths of a second) as specified in the System Log.

TYPE 5 JOB STATISTICS.

The MCP enters the following information in the file REMOTE/LOG when a station detaches from a job.

Word	Field	Contents
0	[2:1]	1 if this station attached by entering an EXECUTE or RUN card; 0 if attached by READ, SEEK or WRITE.
	[9:9]	Station number ([9:4]=TU, [14:4]=BUF).
	[18:24]	RUN NUMBER (as specified in the types 3 or 4 entries).
	[42:6]	Code = 5.
1		User code.
2		First name of the object program (seven characters).
3		Second name of the object program (seven characters).
4		Processor time in 60th of a second; i.e., processor time used for this station, out of total used by job.
5		Pro-rated time in 60th of a second; i.e., pro-rated time used by this station, out of total used by job.
6		I/O time in 60th of a second; i.e., I/O time used by this station, out of total used by job.
7	[3:21]	Start date; i.e., date when job attached to this station (in binary).

Word	Field	Contents
7 (cont)	[27:21]	Stop date; i.e., date when job detached from station (in binary).
8		Attach time; i.e., time when job attached to station.
9		Detach time; i.e., time when job detached from station.

TYPE 6 ABORT INFORMATION ENTRY.
The form of a type 6 entry is:

Word	Field	Contents
0	[2:1]	1 if this station attached by entering an EXECUTE or RUN Card; 0 if attached by READ, SEEK or WRITE.
	[9:9]	Station number ([9:4]=TU, [14:4]=BUF).
	[18:24]	Run number.
	[42:6]	Code = 6.
1		User code.
2		First name of object program.
3		Second name of object program.
4		Processor time in 60ths of a second; i.e., processor time used by this station out of total used by job.
5		Pro-rated time in 60ths of a second; i.e., pro-rated time used by this station out of total used by job.
6		I/O time in 60ths of a second; i.e., I/O

Word	Field	Contents
6 (cont)		time used by this station out of total used by job.
7	[3:21]	Start date; i.e., date when job attached to this station (in binary).
	[27:21]	Stop date; i.e., date of last H/L (in binary).
8		Attach time; i.e., time when job attached to station.
9		Detach time; i.e., time at last H/L.

CREATION OF REMOTE LOG ENTRIES.
As indicated above, log-in, log-out, and control card entries are made at the time at which they occur. This is possible since the information contained in those entries is immediately available.

The information contained within a job statistics' entry is accumulated during the time that a remote terminal is attached to a program. The entry is recorded in the Remote Log at the time a program and remote terminal become detached from one another.

It is the responsibility of the object program to dictate which remote station is to be charged for any particular "slice" of a program's processor, I/O, and pro-rated time. The task involved in specifying the station to be charged is, however, an easy one. The procedure involved in slicing times is as follows.

The MCP maintains a table, called USERSTA, which contains one location for each program in the mix. The contents of a given program's location in this table is the station address of the remote station presently specified to be charged for the time used by that program.

When a program enters the mix, its location in the USERSTA Table is set to the address of station 0/0, a non-existent remote terminal. The times assigned to station 0/0 are those which the program

does not assign to any given station; i.e., they are unassigned time. Then from that time until the address in that program's USERSTA location changes, station 0/0 is charged for all processor, I/O, and pro-rated times charged to the program. When the address in the program's USERSTA location changes, the remote terminal whose address is then specified begins being charged for the times assigned to the program, etc.

The way in which a program designates the address to be placed in USERSTA (i.e., the way in which a program designates the station to be charged) is to perform either a passive or active interrogate statement referencing the station. In ALGOL, this involves a statement of the form STATUS (TUBUF,0) or STATUS (TUBUF,1). In COBOL, it involves a statement such as MOVE FILENAME FROM TU, BUF TO STATUSWORD or MOVE FILENAME FROM TU, BUF AFTER CHECK TO STATUS-WORD. Each time such an interrogate is performed, the MCP checks to see if the terminal buffer address currently in the program's USERSTA location is different from the one specified in the interrogate statement. If it is, the old station is charged with all times since the previous change in USERSTA and the new station is established as the new recipient of time.

It should be noted that, if a program wishes to designate certain times as being unassigned (i.e., assigned to station 0/0), it should perform a passive interrogate on station 0/0.

Whenever a station is detached from a program, a job statistics entry is recorded in the log. The entry, of course, contains all the times which were allotted to the station in the manner described above.

The file REMOTE/LOG is partitioned in two parts. If n is an integer specifying the number of 30 word segments used by the file, then the first n-ABRTLNGTH segments are reserved for remote terminal log-entries. The record capacity of this area in logical records is 6 x (n-ABRTLNGTH). The abort information is written in the remaining ABRTLNGTH segments of the file. The parameter ABRTLNGTH

(MCP Sequence No. 00908000) specifies the number of segments used in maintaining abort information and must not exceed 34. A remote terminal requires an entry in the Abort Table for each program to which it is attached; the maximum number of entries accommodated is 3 x ABRTLNGTH -1. In the event that a H/L is necessary, the abort area of the file REMOTE/LOG is checked to determine if any remote terminals were attached prior to the H/L sequence. Abort information is formatted as a type 6 entry, and placed in the first (n-ABRTLNGTH) segments.

FILE MAINTENANCE PROCEDURES.

To retain information for the file REMOTE/LOG, a FILE CARD group should appear in the Cold Start Deck (see Burroughs B 5500 Electronic Information Processing System Operation Manual, Form 1024916, pages 3-9 through 3-11).

The first record of the file REMOTE/LOG (i.e., the record with relative address 0) describes the remainder of the file. Contents of record 0 are:

Record 0 File REMOTE/LOG	Word 0	Value of word equals the number of logical records written in the file REMOTE/LOG.
	Word 1	Value of word equals the record capacity (in logical records) of the file REMOTE/LOG.
	Word 2 through Word 4	Reserved for system use.

A user program must initialize word 0 of the file REMOTE/LOG to 0 and word 1 to the record capacity of the file. For example, if the FILE Card in the FILE CARD group of the Cold Start Deck has the form

```
FILE REMOTE/LOG,1x1000
```

then a user program must initialize Record 0, Word 0 to 0 and Record 0, Word 1 to 6000.

The B 5500 operator is notified when the log is half-full and when the log is full. Should the log become full, wrap around will occur. If the log is not present, the operator will be notified the first time the log is accessed.

Operator notification is via the SPO and the messages are:

#REMOTE/LOG FULL

This message is typed when the log is full. Wrap-around will occur the next time the log is accessed.

#DUMP REMOTE/LOG

This message is typed when the log is half-full.

#NULL REMOTE/LOG

This message is typed the first time the remote log is accessed and is not present.

THE WR KEYBOARD INPUT MESSAGE.

If the file with ⟨file identification prefix⟩ REMOTE and ⟨file identification⟩ LOG is not on disk and the operator enters a WR keyboard input message, then 1000 segments are obtained for the file REMOTE/LOG and it is entered in the disk directory. The first 1000-ABRTLNGTH segments are reserved for log-entries; the record capacity in logical records of this area equals 6 x (1000-ABRTLNGTH). The remaining segments are reserved for information pertinent to remote terminals currently attached to programs for abort logging if necessary. An entry is made in this section of the file for each remote terminal attached to a job. The maximum number of such entries is 3 x (ABRTLNGTH-1). The message

#REMOTE LOG ON DISK

is typed out on the SPO when the REMOTE/LOG has been placed on disk and initialized.

SECTION 5
I/O CONTROL

GENERAL.

The I/O Control functions of the MCP are logically divided into two parts:

a. Processing of I/O requests made by the object program, and,

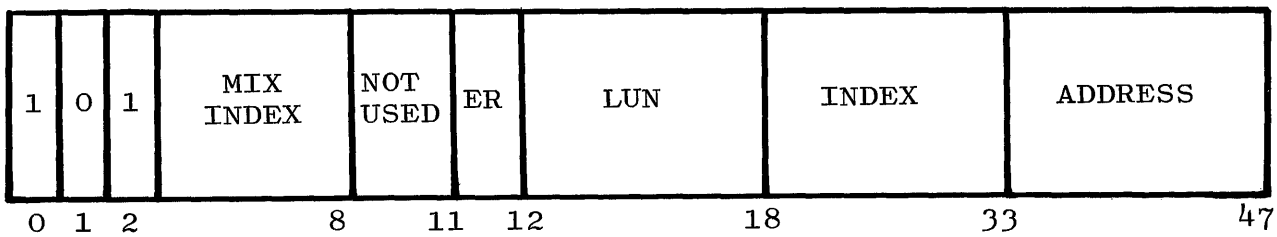
b. Processing of situations resulting from a hardware action; i.e., an independent interrupt.

The first covers opening the files, reading and writing files, with special regard to problems offered by the disk files, and closing the files. The second describes the processing of I/O results, hardware errors, and error routines. Section 5 focuses on the detailed MCP Table information which is applicable to I/O Control.

I/O-QUEUE (LOCATQUE, UNIT).

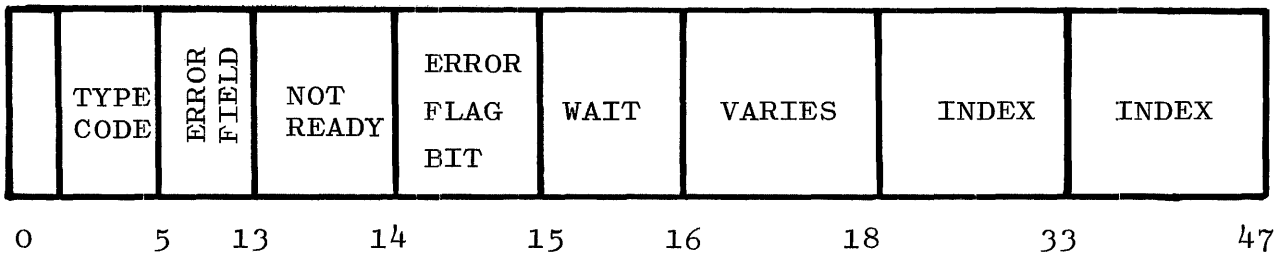
IOQUE, FINALQUE, and LOCATQUE together with UNIT forms the I/O-QUEUE. An I/O request for logical unit U requires three words of space in the I/O-QUEUE. If the request occupies position S in the I/O-QUEUE, then IOQUE(S) contains the I/O descriptor for the request. FINALQUE(S) contains the I/O descriptor skeleton to be used at I/O complete time to rebuild the original I/O descriptor. LOCATQUE(S) points to the location of the I/O descriptor at the time of request. The spaces not used in the I/O-QUEUE are linked together through IOQUE. The first available entry is pointed to by IOQUEAVAIL.

All entries in LOCATQUE have the following format:



Word	Field	Contents
5	[0:3]	Descriptor identification bits.
MIX INDEX	[3:5]	MIX INDEX of program which requested the I/O operation.
	[8:3]	Not used.
ER	[11:1]	Error retry in process.
LUN	[12:6]	Logical unit number of unit on which the I/O is to be executed.
INDEX	[18:15]	Index into I/O-QUEUE of next I/O request on this unit. @77777 if no additional requests occur.
ADDRESS	[33:15]	Address of the I/O descriptor used for this request at time request was made. If buffering is being used by the object program, the descriptors are rotated and the I/O descriptor may not remain in its original location.

All entries in UNIT have the following format (this is a SAVE array):



Word	Field	Contents
0	[0:1]	Flag bit.
Type code	[1:4]	0 = card reader. 1 = line printer. 2 = magnetic tape. 3 = drum. 4 = disk. 5 = SPO. 6 = card punch. 8 = paper tape punch. 9 = paper tape reader. 10= data communications.
Error field	[5:8]	Error field of last I/O done on this unit.
0/1	[13:1]	Not ready bit 0 = unit ready. 1 = unit not ready.
0/1	[14:1]	Error flag bit 0 = no errors. 1 = errors.
0/1	[15:1]	Waiting for I/O channel 1 = I/O awaiting an I/O channel.
Varies	[16:2]	I/O in process bits 00 = unit not in process. 11 = unit in process. 01 = for line printer only. I/O complete but awaiting printer finish.
INDEX	[18:15]	Index of first I/O request for which service is not complete. @77777 if

Word	Field	Contents
INDEX (cont)		none. This is the first entry in the QUEUE.
	[33:15]	Index of last I/O request (entry in the QUEUE) for which service is not complete.

INPUT OUTPUT ASSIGNMENT TABLE.

The I/O Assignment Table is presented as table 5-1. The discussion which follows the table focuses on the logical unit numbers for each I/O unit.

LOGICAL UNIT NUMBERS.

The MCP associates one unique logical unit number with each I/O unit, which is different from the hardware unit number. The logical unit numbers assigned the I/O units were determined by the format of the result of the Read-Ready-Register (RRR) operator. The result of the RRR operation is stored in the field [17:31]. Numbering from right to left, bit [47:1] is numbered 0, and bit [22:1] is numbered 25.

Table 5-1
I/O Assignment Table

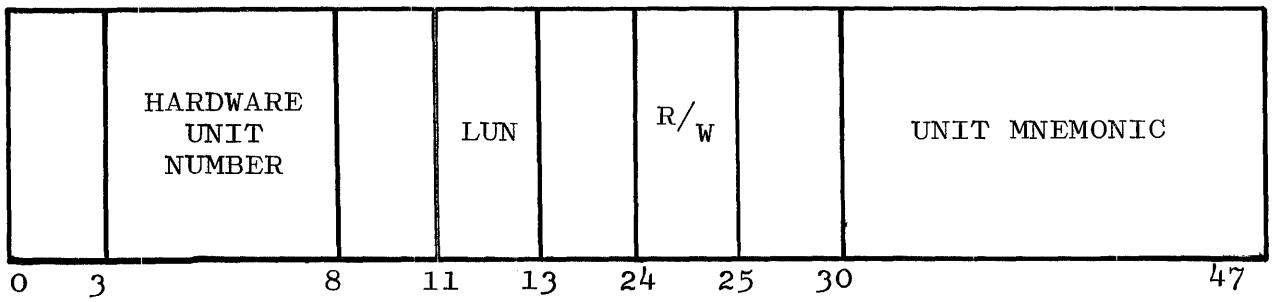
UNIT	LOGICAL UNIT NUMBER	RRRMECH BIT
MTA	0	47
MTB	1	46
MTC	2	45
MTD	3	44
MTE	4	43
MTF	5	42
MTH	6	41
MTJ	7	40
MTK	8	39

Table 5-1 (cont)

I/O Assignment Table

UNIT	LOGICAL UNIT NUMBER	RRRMECH BIT
MTL	9	38
MTM	10	37
MTN	11	36
MTP	12	35
MTR	13	34
MTS	14	33
MTT	15	32
DRA	16	31
DRB	17	30
DKA	18	29
DKB	19	28
LPA	20	27
LPB	21	26
CPA	22	25
CRA	23	24
CRB	24	23
SPO	25	22
PPA	26	21
PRA	27	20
PRB	28	19
PPB	29	18
DCA	30	17
XXX (Used by ZIP)	31	
CDA	32	
CDB	33	
CDC	34	
CDD	35	
MTX	36	
CDE through CDZ, excluding CDI and CDO, and CD2 through CD9.	37 through 64	

TINU is an array used by the I/O routines and provides information about the I/O units. The entries in TINU are ordered according to logical unit number. Information about the unit with logical unit number, LUN, is in TINU [LUN]. All entries in TINU have the following format:



Word	Field	Contents
0	[0:3]	Not used.
	[3:5]	Unit number recognized by hardware.
0	[8:5]	Not used.
	[11:17]	This field contains the logical unit number indicator, which has the following characteristics. The expression (0&TINU [LUN] . [5:11:7] /@1000000000000) will produce a result with all zeros, except in the bit location corresponding to the RRR result bit location designated for the unit represented by TINU [LUN].
	[18:12]	Error count for unit while open.
Unit mnemonic	[30:18]	Three character abbreviation for the unit represented by TINU [LUN]. For example, CRA.

LABELTABLE, MULTITABLE, and RDCTABLE contain label information by logical unit number.

The LABELTABLE is the primary table in the group. The entry of a unit into this table specifies one of the following:

a. The unit is NOT READY.

b. The unit is READY and contains a file that can be used for output (e.g., a line printer file, or a magnetic tape file with a write-ring).

c. The unit is READY and contains an input file not in-use (the LABELTABLE entry in this case would include the file identification of the input file), or

d. The file is READY but in-use.

The MULTITABLE contains the multiple file identification of the file, if any, on the unit represented by the table entry. The RDC Table contains the reel number, purge date, and cycle number of the file, if any, on the unit represented by the table entry. Information in the LABELTABLE, the MULTITABLE, and the RDCTABLE is obtained from the standard labels on the files, if the files are so labeled. Otherwise, the information can be supplied through the use of Label Equation Cards or operator messages. The STATUS Procedure has the primary responsibility of maintaining these tables.

LABELTABLE [I] contains the file identification for logical unit I. MULTITABLE [I] contains the corresponding multi-file identification. RDCTABLE [I] contains the corresponding reel number ([14:10]), creation date ([24:17]), and cycle ([41:7]). If UNIT I is assigned to a program, RDCTABLE [I], [8:6] contains a mix index. Special entries into the LABELTABLE include:

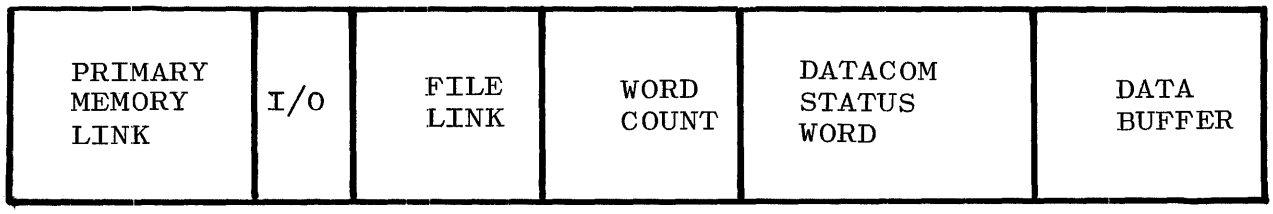
Entry	Contents
@114	Unit not ready.
-@14	Unit in use by system.
@214	Unit is RW/L or saved.
@314	Unit contains an unlabeled tape.
+	Unit available.

Entry	Contents
-	Unit in use.
0	Scratch.

For units 0 through 15:

PRNTABLE [I] contains a 1 in [30:18] if the file is labeled, and, if assigned to a program, the address of the top I/O descriptor in [15:15]. PRNTABLE [I] . [1:1] is 1 if the unit has a write ring. UNIT[S] contains the F field pointing to the first I/O in IOQUE. IOQUE[S] (waiting I/O descriptors) contains the F field pointing to next I/O. If none, @77777 and C field points to UNIT[S]. IOQUEAVAIL points to the first open space in IOQUE; each then points to the next.
FINALQUE[S] (skeleton descriptors) contains the result expected.
LOCATQUE[S] contains the location of the top I/O descriptor.

DATA COMMUNICATION BUFFER



NOTE

With respect to File Link, [FF] = link to next buffer, and [CF] = address of top IOD.

NON-DATA COMMUNICATION BUFFER

PRIMARY MEMORY LINK	SECONDARY MEMORY LINK	FILE LINK [FF] [CF]	WORD COUNT	DATA BUFFER

NOTE

For input, READQ link; for output, ILL link. [FF] = points to previous entry. [CF] = points to next entry - the Data-com status word is the first word of the Data Buffer.

FILE PARAMETER BLOCK (FPB) - ADDRESSED BY R+3.

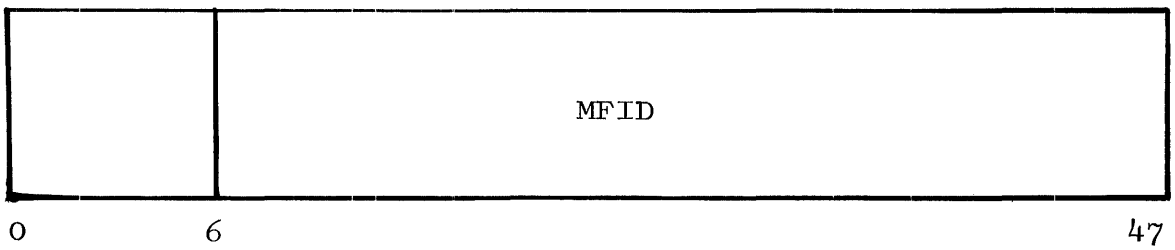
Each program has an FPB, which is created when a program is compiled. It is later modified by the SELECTION Routine during the "fix-up" before a program is initiated. The FPB for a program has an entry for every file to be used by the program.

When a file is declared in a program, that is, when the source program associates the file identifier with a file name and file handling techniques, the compiler assigns the file identifier a file number. This file number, rather than the file identifier, is then used in all references made to the corresponding file by the object program. For each file member, and in file number order, there is an entry in the program's FPB. Each entry in the FPB contains the file identifier, the multiple file identification, and the file identification for the particular number. The location and size of the FPB are placed in an entry of the program's zero

segment. When the SELECTION Procedure is performing "fix-up" operations, it uses this information to obtain the FPB. The FPB must be used at this time to process Label Equation Cards, if any.

Label Equation Cards are special program parameter cards that can be used at run time to associate a file name with a file identifier used in the source language representation of a program. Each Label Equation Card contains the file identifier concerned and the equation information. The equation information includes the multiple file identification and the file identification to be associated with the file identifier. When SELECTION obtains a program's FPB, it also obtains all Label Equation Cards for the program, if any. Then the file identifiers in the FPB entries are compared with the file identifiers on Label Equation Cards. If a match is found, information in the FPB is replaced with the corresponding information from the Label Equation Card. It is in this way that file names associated with files represented by file identifiers can be decided at run time. After all Label Equation Cards for a program have been handled, SELECTION modifies the FPB again by removing the file identifier entries, which are no longer required. Then a descriptor containing the address of the compacted FPB is placed in a specified location in the object program's PRT. Using this description and a file number, the object program is able to make all necessary references to FPB entries.

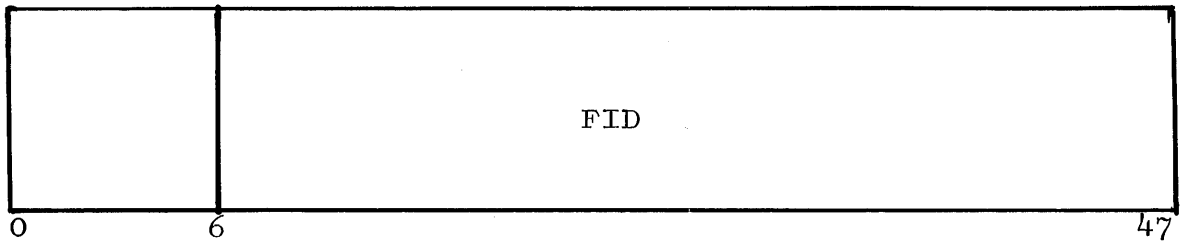
WORD 1



Word	Field	Contents
0	[0:6]	Not used.

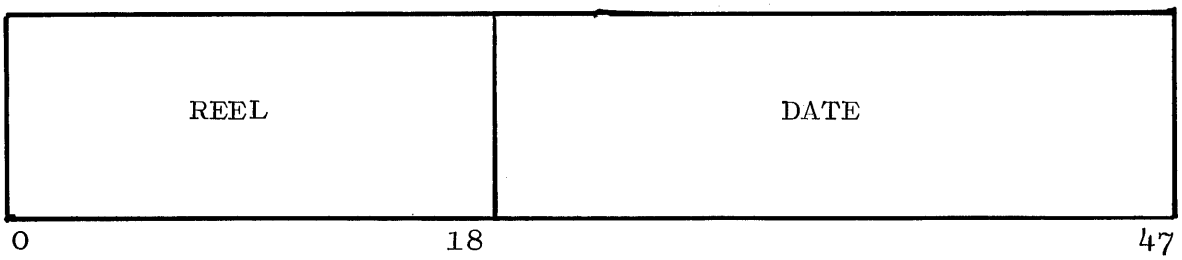
Word	Field	Contents
MFID	[6:42]	Seven characters multi-file identification.

WORD 2



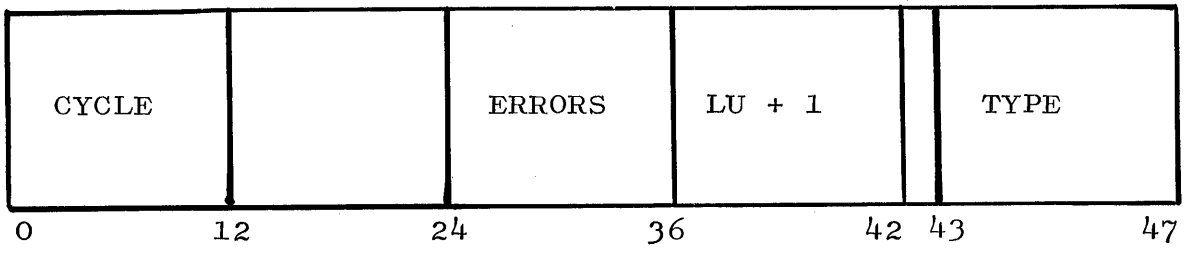
Word	Field	Contents
0	[0:6]	Not used.
FID	[6:42]	Seven character file identification.

WORD 3



Word	Field	Contents
REEL	[0:18]	Reel number in three character alpha.
DATE	[18:30]	Creation date in five characters.

WORD 4



Word	Field	Contents
Cycle	[0:12]	Cycle number (two characters).
Error	[24:12]	Total number of errors for this file.
LU + 1	[36:6]	Logical unit number plus one. Zero indicates unit not assigned.
Forms	[42:1]	1 = types of forms messages.
Type	[43:5]	0 = CP/CR. 1 = LP only. 2 = MT. 3 = DG (designated). 4 = LP/PBT. 5 = specified unit (unlabeled). 6 = PBT only. 7 = PT. 8 = PT unlabeled. 9 = MT unlabeled. 10 = disk. 11 = SPO. 12 = disk serial. 13 = disk update. 14 = data communications. 15 = PBD only. 16 = PBT/PBD. 17 = LP/PBD. 18 = LP/PBT/PBD. 19 = REMOTE

WORD 5

Word	Field	Contents
File open	[1:1]	1 = file is open.
	[2:46]	I/O time on this unit.

FILE INFORMATION BLOCK (FIB).

At run time, there is one FIB generated for each file to be used by a program. An FIB is generated by an object program at each

program point corresponding to a file declaration in the source language representation of the program. Initially, the FIB contains only the information about file handling techniques provided in the source program. When a file is put to use, I/O routines use a file's FIB to store information pertinent to the file such as block counts, record counts, etc. At the point when a file's FIB is created, a buffer descriptor area, containing an I/O descriptor for each buffer area to be used for the file, is also created.

Word	Field	Contents
0		Beginning file.
1		Beginning reel.
2		Ending file.

} USE Routines (see note).

NOTE

Field	Contents
[1:11]	Starting index, BEFORE Routine.
[12:12]	Ending index, BEFORE Routine.
[24:12]	Starting index, AFTER Routine.
[36:12]	Ending index, AFTER Routine.

3		Ending reel.
4	[1:1]	1 = USE Routines present.
	[2:1]	1 = labels omitted.
	[3:2]	EOR rerun: 00 = no, 01 = output tape, 10 = scratch tape.
	[5:1]	1 = optional.
	[6:1]	1 = no I/O part.
	[7:1]	1 = sort file.
	[8:4]	Internal type code. 0 = CR 1 = LP

Word	Field	Contents
	2 = MT	
	3 = DR	
	4 = DK	
	5 = SPO	
	6 = SPO	
	7 = PBT	
	8 = PP	
	9 = PR	
	10 = DC	
	11 = CD	
	12 = PBD	
	[12:1]	0 = bits [13:11] are the file number. 1 = bits [13:11] are the FPB index.
	[13:11]	See above.
	[24:1]	1 = release unit at CLOSE.
	[25:2]	Disposition of file. 00 = rewind. 01 = no rewind. 10 = RW/LK. 11 = RW and release.
	[27:3]	Access mode. 0 = serial. 1 = random. 2 = update.
	[30:18]	Save factor.
5	[1:1]	Used by File Security to indicate (in ALGOL and COBOL I/O ERROR Routines) that we do not have a parity, but an invalid user condition.
	[18:15]	Used by PRNPST/DISK to contain a count of the number of writes used.

Word	Field	Contents
	[40:1]	1 = at end of file.
	[41:1]	1 = CLOSED, unit retained.
	[42:1]	1 = CLOSED, unit released.
	[43:1]	1 = input.
	[44:1]	1 = reverse.
	[45:1]	Not used.
	[46:2]	0 = unblocked. 1 = TECH A. 2 = TECH B. 3 = TECH C.
6		Block count.
7		Record count.
8	[3:15]	Relative PRT location of descriptor for hash totals.
	[15:10]	Number of rows. } for disk files.
	[33:15]	Size of rows. } for disk files.
9	[1:1]	Block already checked (COBOL).
	[3:45]	Rerun control (number of records).
10		Rerun control counter.
11		Number of records per block.
12		Number of records in current block.
13	[1:9]	Number of buffers requested.
	[10:9]	Number of buffers assigned.
	[19:1]	1 = bad key.
	[20:1]	1 = seek given.
	[21:1]	1 = read (first operation) for COBOL only.

Word	Field	Contents
	[22:1]	1 = open.
	[23:1]	1 = write block back.
	[24:1]	0 = alpha (mode).
	[25:1]	1 = reverse (direction).
	[26:1]	1 = memory inhibit (for input).
	[27:1]	1 = input.
	[28:10]	Current reel number.
	[38:1]	1 = forms.
	[39:5]	External type code.
	[44:3]	Not used.
	[47:1]	1 = COBOL.
14		Descriptor for disk file header in core. If file is open, 30 words. (See note.)
15		Error use input index. Error use input end index.
	[24:6]	Logical unit number.
	[30:10]	Special select counter.
	[40:8]	Block count.
16*		Copy of current original I/O descriptor.
17		Number of words left in the buffer.
18	[3:15]	Buffer size.
	[18:15]	TECH C buffer length.
	[33:15]	Maximum record length.
19*		Final I/O descriptor for program release (FINALQUE).

* With flag bit off.

NOTE

FIB[14] has a special use for printer backup files.

[18:15] Pointer to current 18-word psuedo-buffer; i.e., area where next buffer load will go (count backwards).

[33:15] Pointer to last available pseudo-buffer.

When FIB[14].[FF] = FIB[14].[CF], then a PBIO must be done.

FIB[5].[18:15] contains a count of the number of writes (number of pseudo-buffers) used in this file. Used to put in I/O descriptor for use at print time to catch parities, discrepancies, etc.

FILE TANK.

ALGOL (Addressed by a descriptor located in the file's PRT cell.)

Word	Contents	
0	Pointer to label (parity action label).	not used by COBOL
1	Pointer to label (EOF action label).	
2	Pointer to FIB[0].	
3	Pointer to read-in label if input.	
	Pointer to build label if output.	
4	Pointer to top I/O descriptor.	
5	Top I/O descriptor.	
6	Remaining I/O descriptors.	
.	.	
.	.	
.	.	
N		

COBOL

Words 2 through N are located in the PRT for COBOL object programs. Words 0 and 1 are not present.

LABEL EQUATION TABLE (USED BY SHEET [13]).

Entries in the Label Equation Table are:

Word	Field	Contents
0		Multi-file identification (seven characters).
1		File identification (seven characters).
2	[0:18]	Reel number (BCL three characters).
	[18:30]	Creation date (BCL five characters).
3	[0:12]	Cycle (BCL two characters).
	[42:1]	Forms message required bit.
	[43:5]	File types: 0 = card punch or card reader. 1 = line printer. 2 = labeled magnetic tape. 3 = specific unit. 4 = line printer or printer backup tape. 5 = unlabeled specific unit. 6 = printer backup tape. 7 = paper tape. 8 = unlabeled paper tape. 9 = unlabeled magnetic tape. 10 = random disk. 11 = SPO. 12 = serial disk. 13 = update disk.

Word	Field	Contents
		14 = data communications.
		15 = printer backup disk.
		16 = printer backup tape or printer backup disk.
		17 = line printer or printer backup disk.
		18 = line printer, printer backup tape or printer backup disk.
		19 = remote.
4	[0:6]	Number of characters in the internal file name.
	[6:42]	First seven characters of the internal file name.
5-11		Remainder of the internal file name (as required).
12		
13		
14		Equals ? if this is the last entry; otherwise, entries 14 through 25 are the same as above for the next file.
26		
27		
28		
29		Disk address of the next Label Equation entry. If there is no other entry, it equals 0.

SECTION 6
MCP OPERATIONAL TABLES

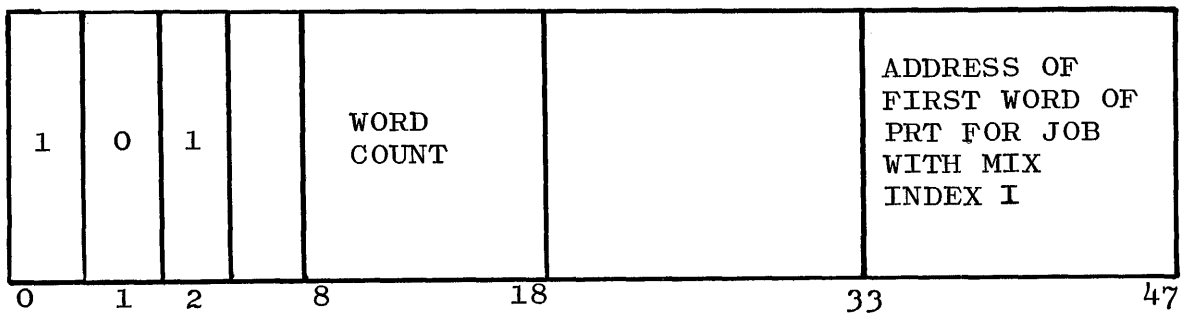
GENERAL.

The MCP must have certain information about the object programs it is running and the equipment it is controlling. This information is stored in various tables and is updated during execution time. This section deals with the tables and procedures which sustain the operation of the MCP.

PRT[*,*].

The PRT is a two dimensional array. The rows of the PRT array are the PRT's of the object programs in the current mix. The rows of the PRT array are ordered according to the MIX indexes of the programs in the mix. Access can be made to the PRT of a given program by accessing the PRT array with a row subscript equal to the program's MIX index; e.g., PRT [MNDX, 7] references the eighth word in the PRT of the program which has the MIX index MNDX.

Word [I, *] of the PRT array is:



Field	Contents
[0:3]	Identification.
[3:5]	
[8:10]	Size of PRT.
[18:15]	
[33:15]	Address of first word in PRT of object program (R+0).

Format of Object Program's PRT:

Cell	Contents	Description
R+0	"EEEEEEEE"	Used by MCP to denote beginning of PRT.
1		Used by ANALYSIS for branch to non-present label.
2	5 000....0	"Memory" for normal state.
3	FPB	Descriptor pointing to FILE PARAMETER BLOCK.
4	SD	Descriptor pointing to SEGMENT DICTIONARY.
5	BC	Descriptor pointing to BLOCK CONTROL intrinsics.
6	AIT	Descriptor pointing to ARRAY INFORMATION TABLE.
7	MSCW	Mark Stack Control Word.
10	INCW	Initiate Control Word.
11	COM/PRL	Location to store constants for the Communicate and Program Release operators.
12		Data descriptor pointing to R+0. F field points to location of stack bottom.
13	SIZEERROR/ OWN ARRAY TABLE	Descriptor pointing to OAT in ALGOL. COBOL [FF] points to the PRT cell reserved for SIZE ERROR indicator.
14	ALGOLWRITE/ COBOLFCR	Program descriptor pointing to write intrinsics for ALGOL, and to FCR for COBOL.

Cell	Contents	Description
15	ALGOLREAD	Program descriptor pointing to read intrinsics for ALGOL.
16	ALGOLSELECT/ COBOLREAD	READ/WRITE descriptor pointing to select descriptor for ALGOL.
17	0	ZERO.
20	BLOCKCTR	Block level counter (starts at 1 with outer-most block of symbolic programs).
21	JUNK	Temporary storage location for use by software.
22	EXITR	Character mode descriptor which references the first syllable of the program; i.e., the outermost block which is generated by the compiler.
23	LISTRTN	Used to obtain next element of a list.
24		Program descriptor of block number 2; i.e., the block which corresponds to the outermost block of the symbolic program.
25	ERROR COUNT	Storage location used by compiler to store the error count. First PRT location assigned by compiler.
26	SAVE TIME	Length of time to save object code.

Cells 22 through 25 are used in this context by ALGOL.

The PRT contents of a FORTRAN object program are:

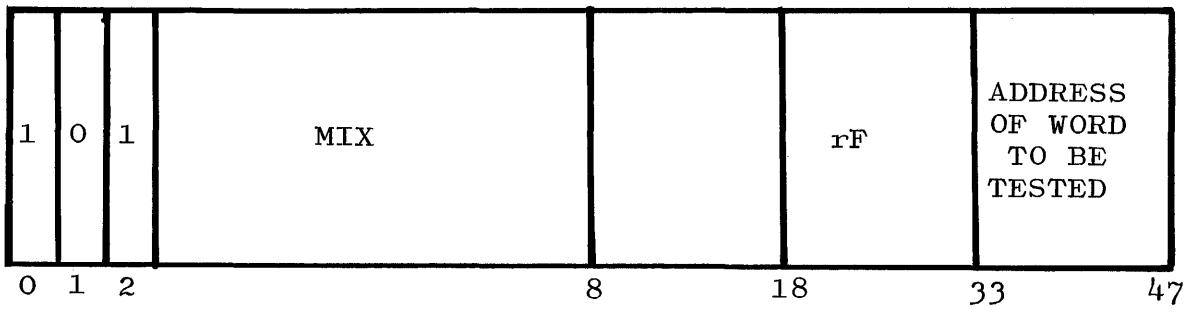
Cell	Contents
R + 0	EEEEEEEE
1	Used by .LABEL.
2	500000000...0
3	FPB
4	SD
5	BC
6	AIT
7	MSCW
10	INCW
11	COM/PRL
12	R + 0, stack
13	OWNARRAY description
14	ALGOL WRITE
15	ALGOL READ
16	ALGOL FILE CONTROL
17	0
20	BLOCKCTR
21	JUNK
22	BASENSIZE
23	LISTRTN
24	CLASN
25	HOLTOG
26	Powers of ten
27	21 word ARRAY for any formatted output and for use by ZIP.
30	ERR
31	SQRT
32	ARSIN
33	EXP
34	SIN
35	ALOG
36	TAN
37	ATAN
40	GAMMA
41	DATAN
42	DCOS

Cell	Contents
R + 43	DSIN
44	ATAN2
45	CABS
46	DMOD
47	DEXP
50	DSQRT

BED [*].

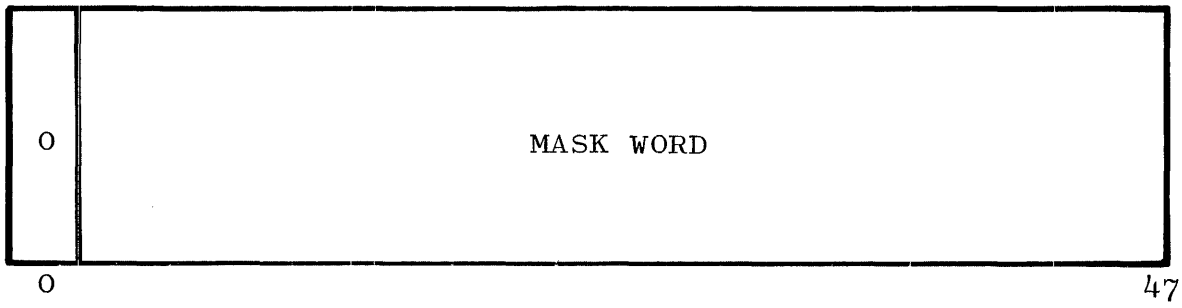
The BED array, the SLEEP, and COMPLEXSLEEP procedures are used to suspend the processing of an object program until a certain condition exists. Two word entries into the BED are made through use of the SLEEP Routine. The last entry in the BED is pointed to by JOBNUM. The BED is also used by the NOTHINGTODO Routine to restart jobs which have been temporarily suspended. Entries made by the SLEEP Routine are:

WORD 1



Contents	Field	Description
5	[0:3]	Descriptor identification bits.
MIX	[3:5]	MIX INDEX of suspended program.
0	[8:10]	Size field.
rF	[18:15]	F Register setting for suspended program.
Address	[33:15]	Address of word to be tested to determine if the necessary condition is satisfied.

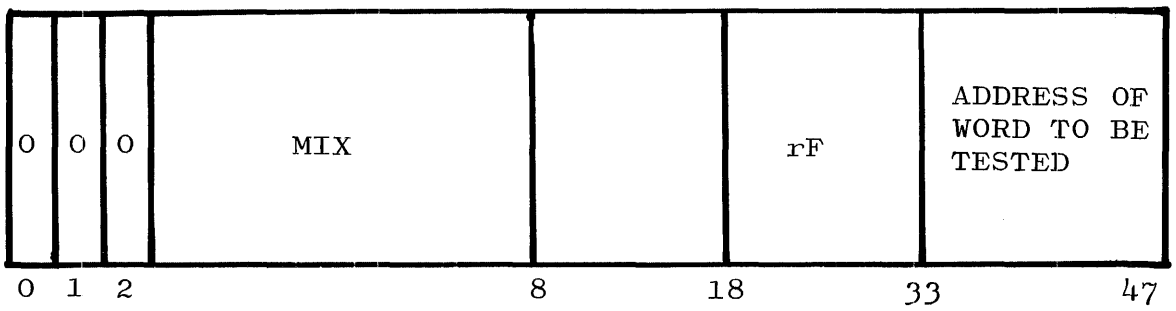
WORD 2



Word	Field	Contents
0	[0:1]	Flag bit (cannot be used for mask bit).
MASK	[1:47]	Contains ones in bit positions which indicate when the needed condition is present. All other bits are set to zero.

Entries made via the COMPLEXSLEEP Routine are:

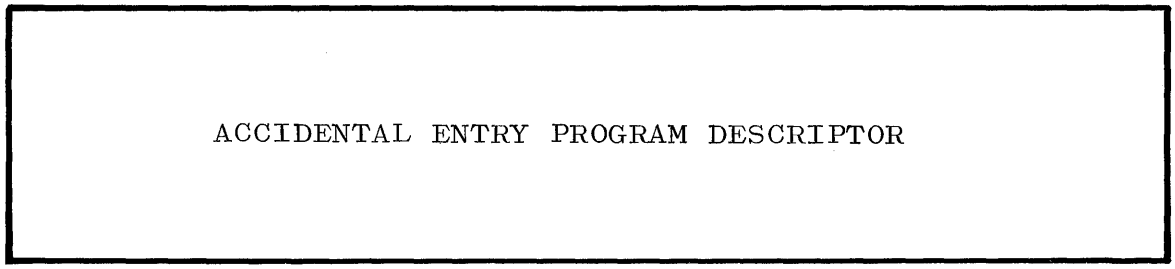
WORD 1



Word	Field	Contents
0	[0:3]	Operand identification bits.
MIX	[3:5]	MIX INDEX of suspended program.
0	[8:10]	Size field.
rF	[18:15]	F Register setting for suspended program.

Word	Field	Contents
Address	[33:15]	Value to be tested against the result from the procedure called by accessing word 2.

WORD 2



Word	Field	Contents
Program Descriptor	[0:48]	Program descriptor which, when accessed, will return a value of 1 if the suspended program can be reactivated. It returns a value of 0 (zero) if it cannot be reactivated.

As conditions dictate, NOTHINGTODO searches the BED to determine if a program can be reactivated. Essentially, the following statements indicate how the test is made.

```
NT1 := Index of entry to be tested;
NT2 := BED [NT1]   ;
NT3 := BED [NT1 + 1]  ;
IF NOT (NT2 AND NT3) ≠ NOT 0 THEN START JOB;
```

BED is ordered by priority.

JOBS ACTUALLY RUNNING: (JAR) [*,*].

The SELECTION routine will fill the JAR from the SHEET when enough space is available to run a job. Entries in the JAR are ordered by mix index and are:

Word	Field	Contents
0		Object program's first name (seven characters). If this is a compiler, this entry is < 0.
1		Object program's second name (seven characters). If this job is in the process of being DS-ed, this entry is < 0.
2	[1:1]	After SELECTION, if this program was compiled using COBOL = 1.
	[1:2]	During SELECTION, as follows: 0 = normal. 2 = job has been XS-ed. 3 = job has been ES-ed.
	[8:10]	0 = go job (from Compile-and-Go). 1 = compiler (Compile-and-Go). 2 = execute job. 3 = compiler (syntax check - set to 2 later). 4 = compiler (Compile-to-Library). 5 = run job. 99 = aborted job (from Initialize). 1023 = syntax errors.
	[18:15]	Skeleton disk address (if JAR [2] . [8:10] = 1, 2, or 4) for the skeleton SHEET for GO part.
	[33:15]	Priority.
3	[8:10]	Scheduled identification for this job.
	[33:15]	Estimated processor time.
4		Estimated I/O time.
	[1:23]	Starting date for the log (binary).
5	[24:24]	Starting time for the log.
6	[18:15]	Size of log information in ESPDISK.

Word	Field	Contents
6	[33:15]	Location of the first record of the log information in ESPDISK. If [2] . [8:40] = 0, then this is the compile log information.
7		Idle time.
8		Length of each row of the code file.
9		Number of rows.
10-29		Disk address for each row of the code file.

The code for a given program may be located by using the JAR entries beginning at JAR [10]. The Segment Dictionary for any given normal state program contains a disk address in the [33:15] field which is the address of that segment, relative (by disk segment) to the JAR [10] entry. If any given relative address exceeds the JAR [8] length, then the next row (JAR [11], JAR [12], etc.) is automatically chosen for the location of the code on the disk. The following formula may be used to locate a given segment of code on the disk for a given program:

Assume RD = the relative disk address from Segment Dictionary entry [33:15] field.

DISK SEGMENT ADDRESS = (JAR [PlMIX, (RD DIV JAR [PlMIX, 8]) + 10]) + (RD MOD JAR [PlMIX, 8])

Mix indexes which are inactive are indicated by a zero entry in JAR [MIX]. If a breakout has been done, JAR [10] = 0, and the Segment Dictionary addresses point to the copied code file in backup storage.

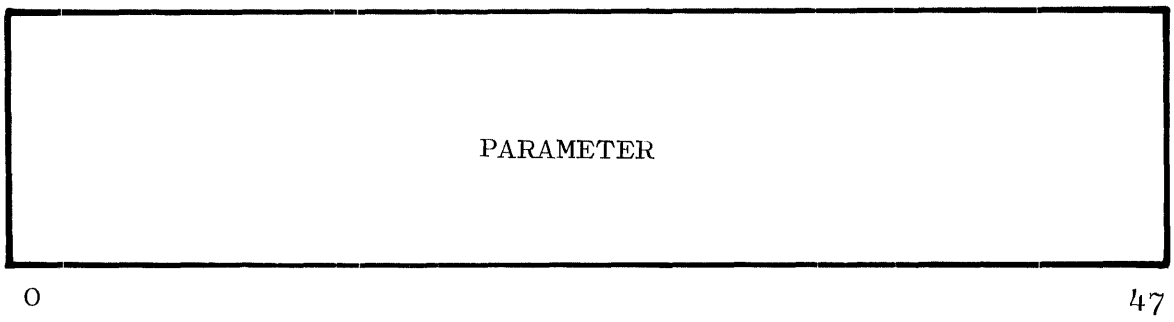
SLATE [*].

The SLATE is a queue of requests to run independent MCP routines whose functions are not directly related to object programs; e.g., STATUS, CONTROLCARD, SELECTION, and RUN.

MCP routines which desire to run independent routines cause entries to be made in the SLATE by calling the INDEPENDENTRUNNER Routine and passing the address of the program descriptor for that routine and a parameter for the routine. INDEPENDENTRUNNER then makes the two necessary entries into the SLATE. The first word of an entry is a parameter to the routine. The second word of an entry is the PRT address of the routine. NSLATE and LSLATE are pointers into the SLATE. NSLATE points at the last entry which was started, and LSLATE points at the last entry placed in the SLATE.

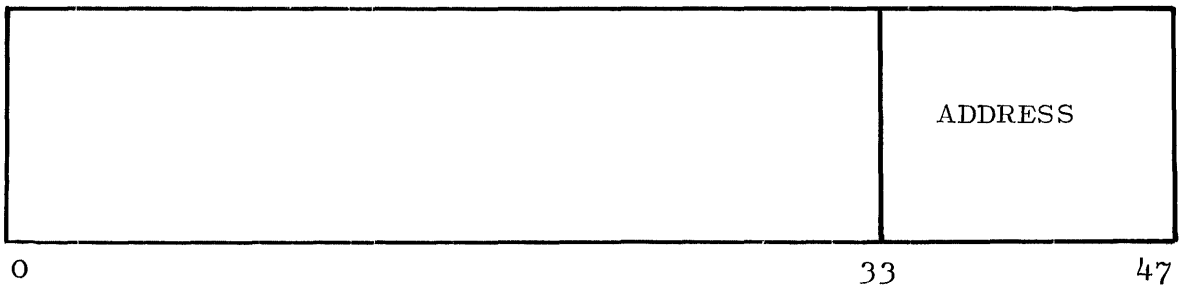
Routines noted in the SLATE are called out by the NOTHINGTODO Routine on a first-in, first-run basis. All entries in the SLATE have the format:

WORD 1



[0:48] varies according to routine. The parameter is for the Independent Routine.

WORD 2



If word 2 is negative ([1:1] = 1), this program was compiled by COBOL.

Word	Field	Contents
0	[0:33]	
Address	[33:15]	Address points to the program descriptor of the Independent Routine.

SHEET [*].

The SHEET provides information to the SELECTION Routine to introduce jobs into the mix. Entries in this table are made by the CONTROL-CARD Routine. Entries in the SHEET are:

Word	Field	Contents
0		Object program's first name (seven characters). If this is a compiler, this word is < 0.
1		Object program's second name (seven characters).
2	[1:2]	0 = normal, waiting. 2 = job has been XS-ed. 3 = job has been ES-ed.
	[8:10]	0 = go job (from Compile-and-Go). 1 = compiler (for Compile-and-Go - set to 2 later). 2 = execute job. 3 = compiler (for syntax check). 4 = compiler (for Compile-to-Library). 5 = run job.
	[18:15]	Skeleton disk address (if SHEET [2] . [8:10] = 1, 2, or 4).
	[33:15]	Priority (same as SHEET [18]).
3	[8:10]	Schedule identification for this job.
	[33:15]	Estimated processor time.
4		Estimated I/O time.

Word	Field	Contents
	[1:23]	Starting date for the log (binary).
5	[24:24]	Starting time for the log.
6	[1:1]	1 = new format for Label Equation Cards.
	[18:15]	FPB Information.
	[33:15]	Location of the first part of the log.
7		
8		
9		
10		
11		
12		Stack size.
13		Disk address of Label Equation entries applicable to this entry only.
14		
15		Disk address of Label Equation entries presented when program was compiled, and applicable to all executions of this job.
16		Estimated processor time.
17		Estimated I/O time.
18		Priority.
19		Common value.
20		Estimated core requirement.
21		Stack size
22		Time to save program (on Compile-to-Library).
23	[9:9]	Remote station address, if any, otherwise, 0.

Word	Field	Contents
	[31:17]	Time this job was entered in the SHEET (for TS message).
24		User code.
25		
26		
27		
28		
29		Disk address for next SHEET entry (= 0, if this is the last entry).

The word, field, and contents for the format of segment zero for the programs is:

Word	Field	Contents
0		Relative location of the Segment Dictionary.
1		Size of the Segment Dictionary.
2		Relative location of the PRT. If < 0 then job compiled by COBOL.
3		Size of the PRT.
4		Relative location of the File Parameter Block.
5		Size of the File Parameter Block.
6		Starting segment number. [1:1] = 1 if new format, else 0.
7	[18:15]	Core requirement/64.
	[33:15]	Number of files.
8		
9		
10		
11		
12		

Word	Field	Contents
13		
14		
15		Disk address of Label Equation entries presented when program was compiled and applicable to all executions.
16		Estimated processor time (from compilation).
17		Estimated I/O time (from compilation).
18		Priority (from compilation).
19		Common value (from compilation).
20		Estimated core requirements (from compilation).
21		Stack size (from compilation).
22		
23		
24		
25		
26		
27		
28		
29		

SEGMENT DICTIONARY AND RELATED PRT CELLS AS CREATED BY A COMPILER. Each program has a Segment Dictionary containing one entry for every program segment in the program, and one word for every intrinsic used. The first word in the Segment Dictionary is referenced as word zero. The entry for any particular segment is located in the

Segment Dictionary word that corresponds to that segment's number; e.g., the entry for segment 3 would be in the fourth word of the Segment Dictionary.

A Segment Dictionary entry as created by a compiler contains the following information, except for entries for intrinsics.

a. The relative address of the segment within the program file on disk. Relative address zero is reserved for a special segment which contains such information as a pointer to the PRT, a pointer to the Segment Dictionary, a pointer to the program parameter block, etc.

b. The size of the segment.

c. An index into the PRT of the first program descriptor that references the segment.

d. A flag specifying whether or not the segment is a type 2 segment.

Entries for intrinsics provide no segment size and have the intrinsics number in lieu of the relative disk address. Otherwise, they are the same.

Although each Segment Dictionary entry may have one or more program descriptors in the PRT, some have none; e.g., fill segments. The program descriptor entries in the PRT, as created by a compiler, contain the following:

a. The relative address within the segment pertinent to the program descriptor.

b. The index into the Segment Dictionary of the entry for the segment to which the program descriptor pertains. This index is equal to the number of the segment.

c. A link (index) to the next program descriptor which addresses the same segment.

d. A stop bit if the program descriptor entry is the last one pertaining to the segment.

FIELDS AND THEIR VALUES FOR THE SEGMENT DICTIONARY AND THE RELATED PRT CELLS.

This topic is dealt with by focusing on the fields and the values of, first, the Segment Dictionary, and, second, the PRT. The fields and their values for the Segment Dictionary are:

Field	Field Value
[0:1]	
[1:1]	1 for type 2 segments (DATA), otherwise, 0.
[2:1]	1 for intrinsics, otherwise, 0.
[3:1]	Reentrant bit.
[4:4]	
[8:10]	Link to program descriptor in PRT (links to first entry in link-list).
[18:15]	If program segment, size of segment; if present, its location in core. For intrinsics, it is meaningless.
[33:15]	Disk address of segment or intrinsic number (relative).

The fields and their values for the related PRT Cells are:

Field	Field Value
[0:4]	Non-present program descriptor bits.
[4:2]	Mode and argument bits.
[6:1]	Stop bit showing end of link list.
[7:11]	If stop bit is on to indicate that it is the last entry, this field contains the index into the Segment Dictionary. Otherwise, it is a link (index) to the next program descriptor that references the segment.

Field	Field Value
[18:15]	Index into Segment Dictionary of the entry pertaining to the segment.
[33:15]	Relative address within the program segment pertinent to the program descriptor.

FORMAT OF FIRST 30 WORDS (1 DISK SEGMENT) OF ALL PROGRAM FILES.
The first 30 words, starting at relative address 0 (zero), of all program files must have the following format:

Word	Field	Contents
0		Location of Segment Dictionary.
1		Size of Segment Dictionary.
2		Location of PRT (0 if compiled using COBOL).
3		Size of PRT.
4		Location of File Parameter Block.
5		Size of File Parameter Block.
6		Starting segment number.
7	[33:15]	Number of files.
	[18:15]	Core requirement/64.
8-29		Not used.

NOTE

The locations noted above are specified according to their relative address within the program file. Sizes are expressed in terms of number of words.

METHOD FOR DECLARING ARRAY SPACE.
The call on the DF MCP to declare array space is nearly identical to the call made when using the MD MCP. With the exception that a different literal value is used to specify the type of storage,

the same parameters are required in the stack. However, when the DF MCP is called, an operand call on a block control intrinsic program descriptor is used rather than a communicate operator.

The following parameters are required in the stack:

a. Mark Stack Control Word.

b. Descriptors pointing to the array descriptors for each array being declared.

c. Sizes of the array dimensions.

d. Number of dimensions.

e. Number of arrays being declared.

f. TYPE of storage.

With these parameters in the stack, an operand call on the block intrinsic program descriptor will cause the array space setup. The values for TYPE are defined as follows:

0 = Regular array space (overlayable).
1 = SAVE array space (non-overlayable).
2 = OWN array space.
3 = SAVE and OWN array space.

NFO.
NFO contains the following for each active mix index and is used for reconstructing the PRT for stack overflow conditions. NDX represents the number of entries per job in the NFO Table.

NFO [(MIX-1) times NDX] = File Parameter Block data descriptor.

NFO [(MIX-1) times NDX+1] = Segment Dictionary name descriptor.

NFO [(MIX-1) times NDX+2] = Location of bottom of stack (word containing all B's).

NFO [(MIX-1) times NDX] . [1:17] = Clock time at BOJ.

NFO [(MIX-1) times NDX] . [18:15] = Core estimate DIV 64.

NFO [(MIX-1) times NDX] . [33:15] = Location of stackbottom.

LOOKQ.

LOOKQ is a variable in the MCP's PRT which will point at the first entry for a logged-in user of a remote device. Each entry is ten words long, including the memory link. Each entry links to the next and will eventually point back to LOOKQ. The format of the LOOKQ word is:

Field	Contents
[0:9]	0
[9:9]	@777
[18:15]	Address of secondary link word for last entry.
[33:15]	Address of secondary link word for first entry.

LOOKQ entry:

Word	Field	Contents
0		Memory link word for save memory type.
1		Secondary link word.
	[0:9]	0
	[9:9]	Terminal and buffer number.
	[18:15]	Address of next entry.
	[33:15]	Address of previous entry.
2		User code.
3		CCMASK1

Word	Field	Contents
4		CCMASK2
5		INFOMASK1
6		INFOMASK2
7		MIXMASK
8		Time when user logged in (sec/60).
9		Not used.

MESSAGEHOLDER.
The MESSAGEHOLDER is dealt with as follows:

Field	Contents
[18:15]	Points at the last buffer added to the SPO queue. The first word of each message starting at MESSAGE-HOLDER [33:15] is a memory link address of the next message in the SPO queue that is to be printed. The last message in the SPO queue will contain zeroes in the first word.
[33:15]	Points at the SPO message that is currently being printed or the next SPO message to be printed.

A maximum of 100 messages of varied length (length depends on the routine that calls SPOUT) may be placed in the SPO queue. NUMESS + 100 equals the number of messages left in the SPO queue. If NUMESS equals -100, then the SPO queue is empty. The messages must contain a group mark (←). If not, when SPOUT is called, the memory following the message will be printed and destroyed.

INQUIRY: ARRAY DCB [16] AND THE ORR WORD.
DCB is a table used by the data communications' handling procedures. Initially, all words in DCB = 0. There are two pointer words used in conjunction with DCB. These pointer words are NEXTINQ and CURRINQ. NEXTINQ points at the word in DCB that will be used when handling the next Inquiry Request Interrupt. CURRINQ points at the

word in DCB that will be used when handling the next COM9; i.e., the next FILL ⟨array row⟩ WITH INQUIRY statement.

HANDLING AN INQUIRY REQUEST INTERRUPT.
When an Inquiry Request interrupt occurs, DCB [NEXTINQ] is tested to see if it equals zero. If it is zero, a buffer area is obtained and its address is placed in DCB [NEXTINQ] . [33:15]. Then a read is performed to handle the interrupt, and the number of words in the message is placed in DCB [NEXTINQ] . [18:15]. If DCB [NEXTINQ] were not zero, it would already be set-up with the address and size of an available buffer area.

If after the read is performed, the result descriptor shows that input was received, DCB [NEXTINQ] . [1:1] is set to 1, DCB [NEXTINQ] . [14:4] is set to the terminal unit number of the unit that provided the message, and the ORR word (see below) is set to note that the TU is "output ready" or "output possible."

If the result descriptor shows an "output ready" condition (i.e., ready for another line of a message), DCB and NEXTINQ are left as is, and the ORR word is set to indicate the "output ready" condition.

HANDLING A FILL WITH INQUIRY.
When a communicate indicates that an inquiry message is requested, DCB [CURRINQ] is tested for a value less than zero; i.e., tested to see if DCB [CURRINQ] . [1:1] = 1. If DCB [CURRINQ] is less than zero, the message from the buffer area addressed by DCB [CURRINQ] . [33:15] is supplied to the requestor, together with the TU number in DCB [CURRINQ] . [14:4]. CURRINQ is then incremented to the next location. The space for buffer area addressed by the previous CURRINQ word is returned. If DCB [CURRINQ] is not less than zero, the requestor is put to COMPLEXSLEEP waiting on DCB [CURRINQ] < 0.

THE ORR WORD.
The ORR word indicates the "output ready" status and "output possible" status of all TU's. A unit is "output ready" and "output

possible" if the TU is waiting for a message. If it is handling one line of output and will be coming back for another, it is "output possible," but not "output ready." The following tests provide "output ready" and "output possible" information.

IF (TWO (TU) AND ORR) ≠ 0 THEN OUTPUT READY

IF (TWO (TU + 15) AND ORR) ≠ THEN OUTPUT POSSIBLE

NOTE

TWO is a function such that TWO (X) = 2 * X.

SECTION 7
BINARY CARDS

GENERAL.

Once after every Halt/Load (H/L) operation, the initial operations call the INITIALIZE Routine into action. The Routine reads from disk into core the information which was entered into the system through the MCP load deck and stored on disk by the Cold Start Routine. This information is placed in certain MCP PRT variables. The Routine initializes and updates the tables used by MCP (PRT, SHEET, etc.), and performs the first organization and classification of core storage. It also creates the Available-Disk Table.

After these operations, the MCP prints the H/L messages on the SPO. During initialization, the field used to maintain the ready or not ready status of the peripheral units is set to indicate that all units are in not ready status. The first execution of the N-Second Routine sets the indicators properly.

This section on Binary Cards presents the H/L Button Card, the ESPOL Transfer and Load Cards, and the Initialization Code brought in by the H/L Card.

H/L CARD.

Information in the log is not lost due to H/L operations. When the log becomes half full, a message is typed to notify the operator. When the log is almost full, an MCP routine is fired up which changes the name of the SYSTEM/LOG and which initializes a new SYSTEM/LOG. The new name that is given to SYSTEM/LOG is ⟨M⟩ ⟨D⟩ ⟨C⟩/SYSLOG where ⟨M⟩ = a two digit number representing the month of the year, ⟨D⟩ = a two digit number representing the day of the month, and ⟨C⟩ = a three digit number that is incremented each time the name changing routine is invoked. A keyboard message which gives the new name is written after the name has been changed. For example,

**** NEW LOG FILE IS 1230007/SYSLOG

Table 7-1 presents the H/L Button Card, and figure 7-1 presents a flow chart of the H/L Card.

Table 7-1

H/L Button Card

20	0441				Mark Stack
		3410			Literal 702
			0360		Literal 74
				4231	Branch Forward Unconditional (40-0)
21	7500				Word Mode
		0000			Program
			0000		Descriptor
				0023	
22	0211				Interrogate Interrupt
		0014			Literal 3
			4131		Branch Backward Unconditional (22-0)
				0435	Exit
23	7012				Operand Call F-2
		7007			Descriptor Call F-1
			0421		B Store Destructive
				0014	Literal 3
24	4411				Initiate I/O
		0054			Literal 13
			4131		Branch Backward Unconditional (22-0)
				4155	Dial A 41

Table 7-1 (cont)

H/L Button Card

25	6461				Dial B 64
		1065			Transfer Bits 10
			0000		Literal 0
				0425	B Not Equal to A
26	0074				Literal 17
		0131			Branch Backward Conditional (22-3)
			0064		Literal 15
				4131	Branch Backward Unconditional (23-3)
27	0000				Literal 0
		0062			Operand Call 14
			0064		Literal 15
				4131	Branch Backward Unconditional (24-3)
30	0000				Literal 0
		0066			Operand Call 15
			0104		Literal 21
				4131	Branch Backward Unconditional (24-3)
31	0000				Literal 0
		0072			Operand Call 16
			0124		Literal 25
				4131	Branch Backward Unconditional (24-3)

Table 7-1 (cont)

H/L Button Card

32	0000	Literal 0
	0076	Operand Call 17
	0144	Literal 31
	4131	Branch Backward Unconditional (24-3)
33	5140	Disk File Read Descriptor
	0000	7 Segments
	4070	from address specified
	0137	in 0137
34	5140	Disk File Read Descriptor
	0000	$77_{(8)}$ Segments
	4770	from address specified
	0461	in 0461
35	5140	Disk File Read Descriptor
	0000	$77_{(8)}$ Segments
	4770	from address specified
	4223	in 4223
36	7700	Character Mode
	0000	Program
	0000	Descriptor
	0037	

Table 7-1 (cont)

H/L Button Card

37	0153				Recall Source Address F-1
		0204			Recall Destination Address F-2
			0405		Transfer Words 04
				0000	Exit Character Mode
40	0167				Descriptor Call 35
		0106			Operand Call 21
			0441		Mark Stack
				0440	Literal 9
41	0163				Descriptor Call 34
		0106			Operand Call 21
			0441		Mark Stack
				0010	Literal 2
42	0157				Descriptor Call 33
		0106			Operand Call 21
			0441		Mark Stack
				0660	Literal 14
43	0600				Literal 140
		0172			Operand Call 36
			0520		Literal 124
				4131	Branch Backward Unconditional (17-0)

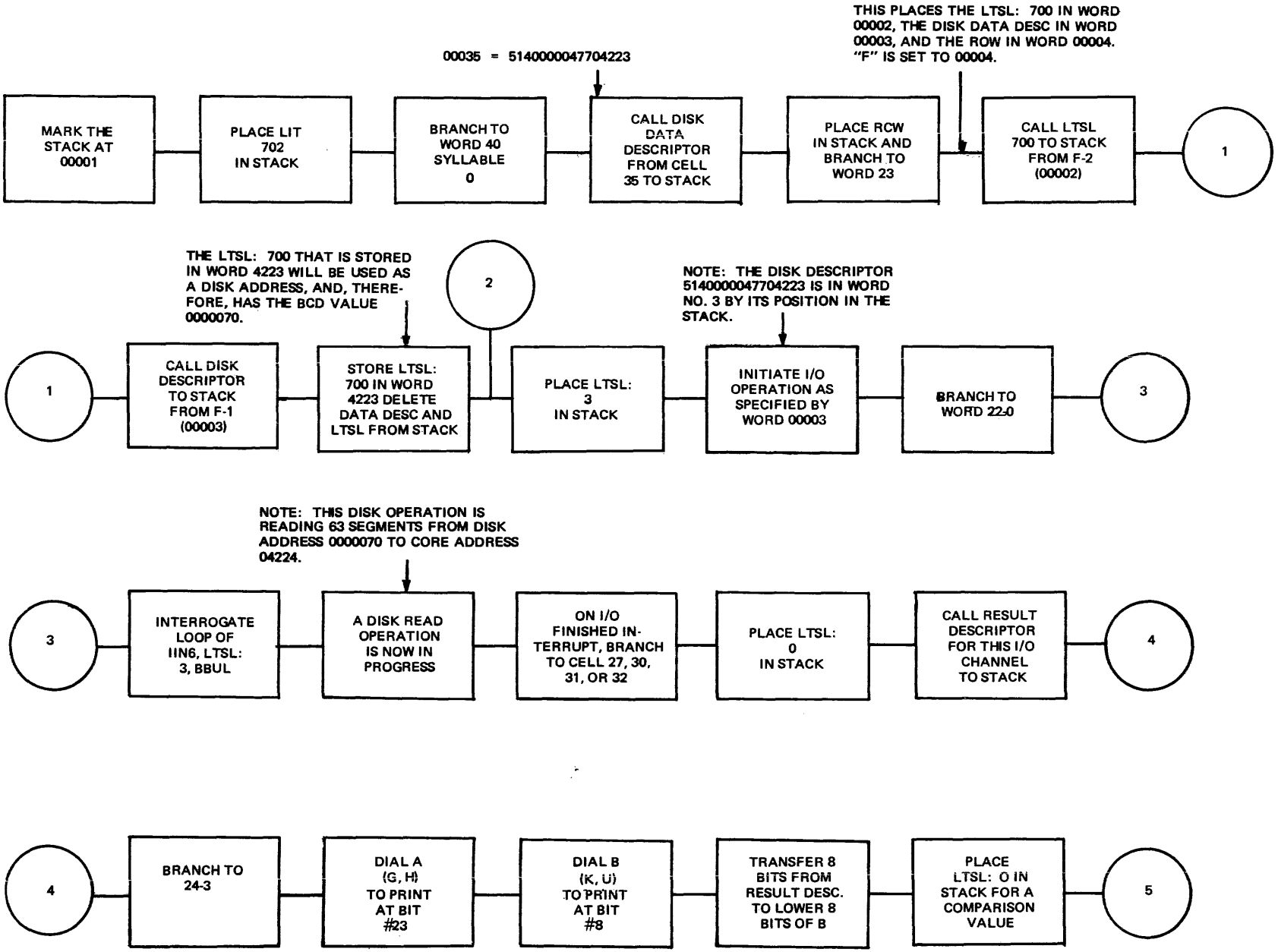


Figure 7-1. Detailed Flowchart of the H/L Card (Sheet 1 of 2)

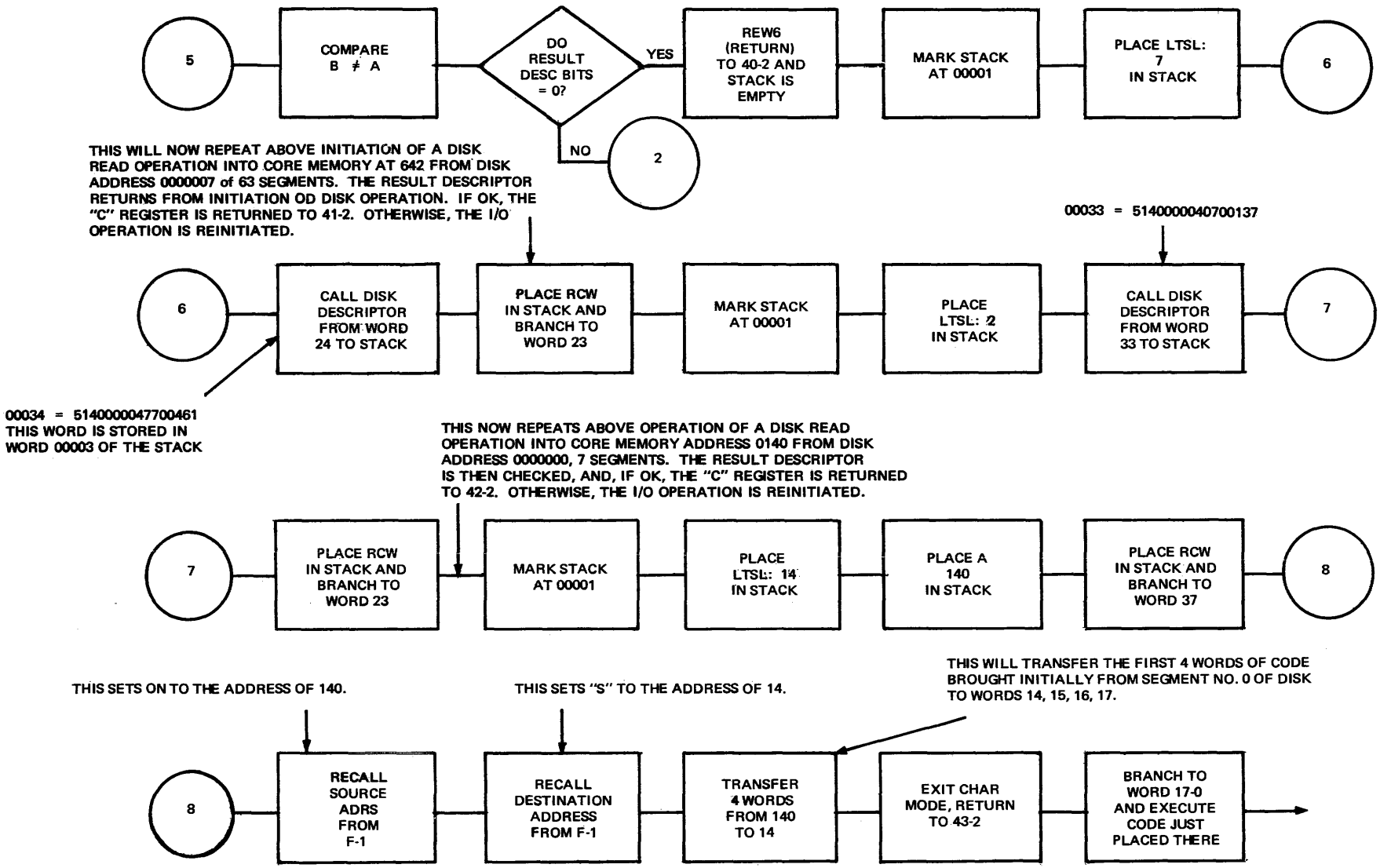


Figure 7-1. Detailed Flowchart of the H/L Card (Sheet 2 of 2)

ESPOL TRANSFER CARD.

This is an alphanumeric card, which is the final card in the MCP Load Deck and the COLD START Deck. Table 7-2 presents this card.

Table 7-2

ESPOL Transfer Card

11	7500				Word Mode
		0000			Program
			0000		Descriptor
				0012	
12	0004				Literal 1
		5355			Dial A 53 (C Field)
			3061		Dial B 30 (F Field)
				1765	Transfer Bits 15_{10}
13	7006				Operand Call F-1
		0004			Literal 1
			0421		B Store Destructive
				0435	Exit
14	7700				Character Mode
		0000			Program
			0000		Descriptor
				0015	

Table 7-2 (cont)

ESPOL Transfer Card

15	0253				Recall Source Address F-2	The 3969 words starting at 00160 are relocated.
		0104			Recall Destination Address F-1	
			7752		Begin Loop 63_{10}	
				7705	Transfer Words 63_{10}	
16	0051				End Loop	
		0000			Exit Character Mode	
			0441		Mark Stack	
				0046	Operand Call 11	
17	0441				Mark Stack	
		0700			Literal 160	
			0100		Literal 20	
				0062	Operand Call 14	
20	0040				Literal 10	
		4131			Branch Backward Unconditional (16-2)	
			0000			
				0000		

NOTE

20 is overlaid by character mode transfer in 15-3.

ESPOL LOAD CARD.

This card is a binary card, which is the first card of the MCP Load Deck or the COLD START Deck. The card is presented in table 7-3.

Table 7-3

ESPOL Load Card

20	0104				Literal 21
		4411			INITIATE I/O
			0020		Literal 4
				4231	Branch Forward Unconditional (22-0)
21	5240				Card Read Descriptor
		1200			Alpha 12_8 Words
			4000		CRA
				0044	
22	4455				Dial A 44 (Bit 20)
		0211			Interrogate Interrupt
			0020		Literal 4
				4131	Branch Backward Unconditional (22-0)
23	7700				Character Mode
		0000			Program Descriptor
			0000		
				0024	

Table 7-3 (cont)

ESPOL Load Card

24	0453	Recall Source Address F-4
	0304	Recall Destination Address F-3
	0243	Call Repeat Field F-2
	0005	Transfer Words
25	0000	Exit Character Mode
	0065	Transfer bits 00
	0100	Literal 20
	4131	Branch Backward Unconditional (22-0)
26	0110	Literal 22
	4131	Branch Backward Unconditional (22-0)
	0055	Dial A 00
	0055	Dial A 00
27	0000	Literal 0
	0062	Operand Call 14
	0050	Literal 12
	4231	Branch Forward Unconditional (32-2)
30	0000	Literal 0
	0066	Operand Call 15
	0030	Literal 6
	4231	Branch Forward Unconditional (32-2)

Table 7-3 (cont)

ESPOL Load Card

31	0000				Literal 0
		0072			Operand Call 16
			0010		Literal 2
				4231	Branch Forward Unconditional (32-2)
32	0000				Literal 0
		0076			Operand Call 17
			7561		Dial B 75
				0165	Transfer Bits 01
33	0010				Literal 2
		0231			Branch Forward Conditional (34-0)
			0010		Literal 2
				4131	Branch Backward Unconditional (33-2)
34	0004				Literal 1
		0107			Descriptor Call 21
			2025		Duplicate
				0044	Literal 11
35	0106				Operand Call 21
		2025			Duplicate
			3355		Dial A 33
				4061	Dial B 40

Table 7-3 (cont)

ESPOL Load Card

36	2565				Transfer Bits 25
		2025			Duplicate
			2265		Transfer Bits 22
				2025	Duplicate
37	1765				Transfer Bits 17
		2025			Duplicate
			1465		Transfer Bits 14
				5355	Dial A 53
40	5361				Dial B 53
		1765			Transfer Bits 17
			0000		Literal 0
				0044	Literal 11
41	0106				Operand Call 21
		2025			Duplicate
			1555		Dial A 15
				2261	Dial B 22
42	0165				Transfer Bits 01
		2255			Dial A 22
			7261		Dial B 72
				0465	Transfer Bits 04

Table 7-3 (cont)

ESPOL Load Card

43	0441	Mark Stack
	0116	Operand Call 23
	0500	Literal 120
	4131	Branch Backward Unconditional (20-0)

INITIALIZATION.

The B 5500 System is initiated when the machine operation performs an H/L operation by pressing the HALT switch, then the LOAD switch. The operation automatically caused Processor one to go into control state and a portion of code to be read into the first locations of core memory in module zero. Control is then automatically transferred to core address 16 and the system is in operation. Initial operations cause the INITIALIZE Procedure and permanent segments of the DC MCP to be read from disk into core. The DC MCP then performs various initialization functions, including performing the first organization and classification of core storage, and creating the Available-Disk Table. Table 7-4 presents the format for the Initialization Code brought in by the H/L Card.

Table 7-4

Initialization Code Brought in by H/L Card

14	7700	Character Mode
	0000	Program
	0000	Descriptor
	0015	

Table 7-4 (cont)

Initialization Code Brought in by H/L Card

15	0253				Recall Source Address F-2	Starting at 00160, the 3969 words are relocated beginning at 00020.
		0104			Recall Destination address F-1	
			7752		Begin Loop 63_{10}	
				7705	Transfer Words 63_{10}	
16	0051				End Loop	
		0000			Exit Character Mode	
			0000			
				0000		
17	0441				Mark Stack	
		0700			Literal 160_8	
			0100		Literal 20_8	
				0062	Operand Call 14_8	

NOTE

Enter at 17-0 from branch command 43-3 of the H/L Card.

Operating Conditions:

a. Timer can be on.
b. Printer Finished or Keyboard Request will stop the program.
c. Will work on any I/O Channel.

SECTION 8
LIBRARY MAINTENANCE

GENERAL.

Procedures are available to maintain the user program library, construct requested library entries, and to update the associated tables. Section 8 deals with library maintenance action by providing a detailed depiction of the format for a library tape and the format of a library maintenance segment used for load information.

FORMAT OF A LIBRARY TAPE.

The contents for each physical record determines the format of a library tape. Figure 8-1 presents this information in a sequential manner, providing a place for everything from tape label to tape mark.

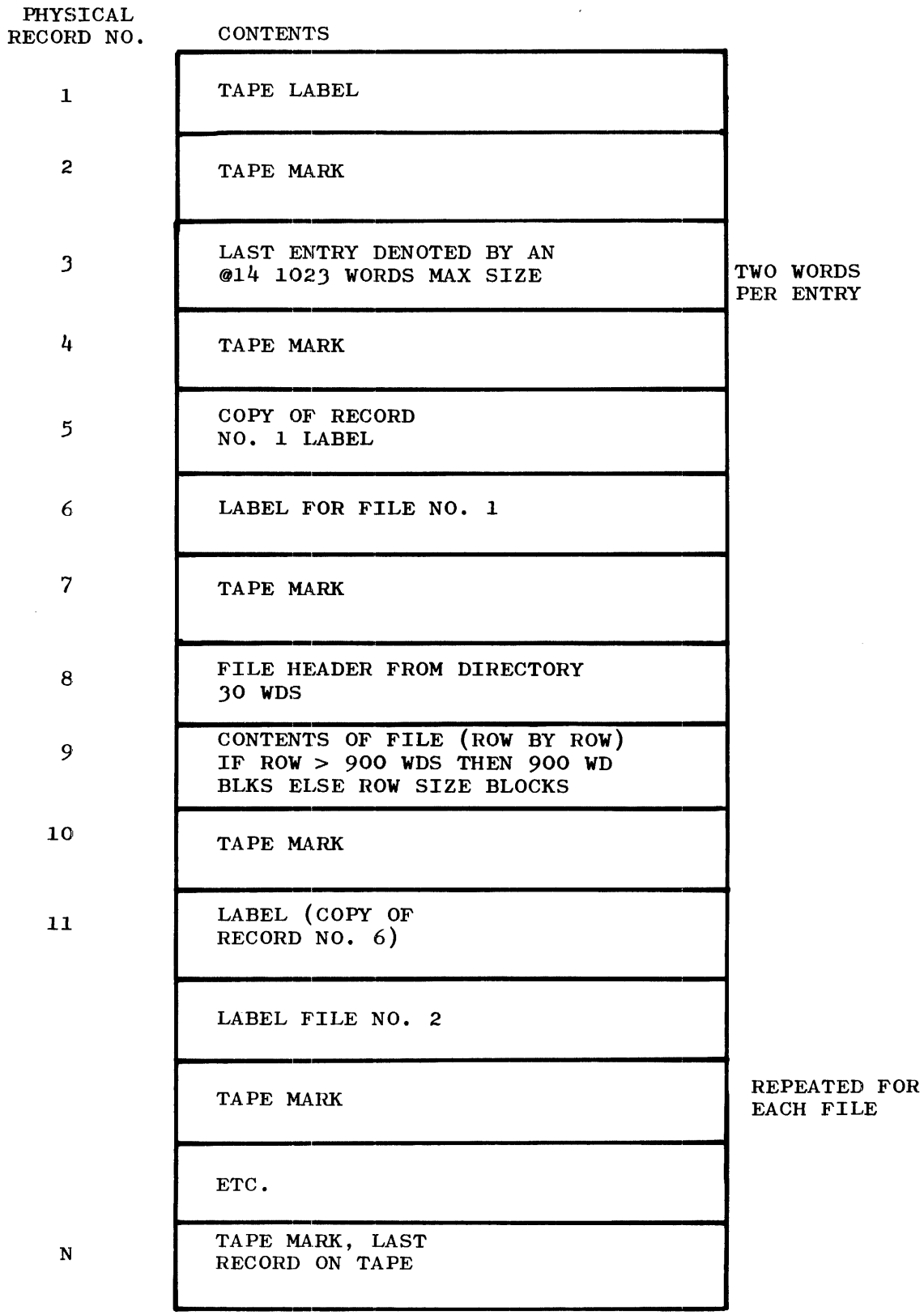


Figure 8-1. Format of a Library Tape

FORMAT OF LIBRARY MAINTENANCE SEGMENT FOR LOAD INFORMATION (SHEET ENTRY).

The SHEET entry for load information requires a specific format for each library maintenance segment. Figure 8-2 provides the format for a segment to illustrate the arrangement of words and programs.

[0].[2:6] = UNITNO = 23, 24 or $\geq$ 32.

.[8:1] = 1, DUMP EXPIRED FILES.

.[FF] = T, no, indicating LOAD, DUMP, etc. (see MCP procedure RESWDS).

[1] TAPE LABEL

[2] MULTI-FILE ID

[3] FILE ID

.

.

.

.

.

.

[27] MULTI-FILE ID

[28] FILE ID

[29] ESPDISK ADDRESS LINK

Figure 8-2. Format of the Library Maintenance Segment for Load Information

SECTION 9

INTERRUPT HANDLING

GENERAL.

Interrupts are initiated by the hardware itself when the computer is operating in normal state and certain conditions are encountered. The MCP also provides facilities that allow programs to have rerun points. If a program requests a breakout, all processing of object programs is halted. Subsequently, all of memory and overlay storage is written on magnetic tape; then, in the case of a breakout, object programs are re-initiated and continue processing.

When a program is to be restarted at a rerun point, no programs may be on the system. Also, all files related to the program(s) to be restarted must be in place on the units where they were at breakout time. At such a time, a restart request will be handled by reading the restart information and restoring core to the condition that existed when the breakout occurred. Then, overlay storage is restored.

Finally, only that program is restarted. Other programs, which may have been in process when the breakout occurred and which are reflected in the restored memory and overlay storage, are terminated. The BREAKSTART Procedure is the primary procedure used to perform breakouts and restarts. Section 9 focuses on the handling of a Presence Bit Interrupt in the operation of B 5500 Hardware and Software.

PRESENCE BIT INTERRUPT ACTION.

When a Presence Bit Interrupt is detected, control is transferred to the Presence Bit Routine. The fact that a Presence Bit Interrupt occurred means that a program has executed a syllable that caused an attempt to access information described by a descriptor with a zero presence bit. The following action takes place:

a. Presence Bit Interrupt is set in Central Control by the attempt of a normal state program to access a non-present data descriptor. This is a descriptor with bit [2:1] = 0.

b. This being a syllable-dependent interrupt, it is sensed at SECL (Syllable Execution Complete Level) time. This causes an SFIL (Store for Interrupt Level) operator to be placed into the T Register.

c. The B Register is pushed down.

d. The A Register is pushed down.

e. If you are in character mode, build and push down an ILCW (Interrupt Loop Control Word).

f. Build and push down an ICW (Interrupt Control Word).

g. Build and push down an IRCW (Interrupt Return Control Word).

h. Build an INCW (Initiate Control Word) and place it in the object (normal state) program's PRT at R + 10 (octal).

i. Force an INI (Interrogate Interrupt) operator into the T Register.

j. Transfer to either Cell 55 (octal) or Cell 67 (octal), depending on whether this was a Presence Bit on P1 or P2. Set the R Register to zero. Set the S Register to 100 (octal).

k. Place an 18 (decimal) in the TOS (Top of Stack) at Cell 101 (octal).

l. Transfer to the MCP Outer Block Label P1PROCESS.

m. Set the S Register to point at the IRCW stored in the object (Normal state) program's stack at step g, above.

n. Set the F Register to zero.

o. Branch forward as many syllables as indicated by the number placed in the top of stack at 101 (octal in k, above).

p. In this case, we will end up at the call MAKEPRESENT (ANALYSIS). Note that ANALYSIS is a typed (REAL) procedure. Before entering MAKEPRESENT, we actually enter ANALYSIS, returning with a value to be passed as a parameter to MAKEPRESENT.

APPENDIX A
MCP COMMUNICATES

To use the communicate operator, a normal state program first places a parameter in its stack. The word at the top of the stack is then stored in the cell addressed by R + 9. The Communication Interrupt Bit is set and the MCP routine that handles this interrupt first locates R + 9 of the program that caused the interrupt. Then, according to this code value, the MCP transfers control to the section of the MCP designed to handle a communicate interrupt with that code. The operator is treated as a NOOP in control state. The following is a list of the codes used by the communicate operator:

Code	Description
0	Invalid End-of-Job in COBOL or FORTRAN.
1	TIME (⟨variable⟩) function in ALGOL. DATA (⟨data name⟩) function in COBOL.
2	SLEEP call (wait) two parameters passed.
3	Return specific array (pass array name and dimensions).
4	ZIP WITH or PERFORM WITH array row or file name.
5	Normal End-of-Job. Calls COM5 which calls SIGNOFF.
6	WHEN function (pass number of seconds).
7	Fill array row.
8	ZIP ⟨program-id⟩ or PERFORM ⟨program-id⟩.
9	Fill with inquiry. (Not applicable to data communications systems.)
10	Block Exit (ALGOL storage return).
11	ALGOL I/O Function. Pass parameter as follows:

Code	Description
	0 = file open. 1 = parity message. 2 = End-of-File message. 3 = End-of-Tape message. 4 = data communications. 5 = return disk. 6 = close file. 7 = process RER. 8 = select error. 9 = space. 10 = refill. 11 = read label. 12 = IOREQ. 13 = rotate buffers.
12	BREAK or RERUN.
13	COBOL I/O Functions (OPEN and CLOSE only).
14	Invert overlayable status of an array row.
15	DISPLAY.
16	ACCEPT.
17	COBOL I/O errors (called from COBOLFCR or Sort Intrinsic).
18	Inquiry Write (Not applicable to data communications.)
19	Printer Backup Routine (PRNPBT).
20	Tape swap for tape sort.
21	Get Space for sort.
22	Return space from sort.
23	Load Control (LDCNTRL).
24	Return one row of a disk file.
25	Return old copy of OWN array.
26	Invalid arguments to intrinsics (LN, SQRT, etc.). ALGOL only.
27	COBOL data communications interrogate.
28	ALGOL data communications interrogate.
29	Miscellaneous errors (error terminate).
30	Directory search statement and label equation.
31	ALGOL DELAY function.

Code	Description
32	Data communications seeks, detaches, and interrogates.
33	FORTRAN PAUSE statement.
34	FORTRAN error terminate.

APPENDIX B
STANDARD B 5500 LABEL RECORD

Word	Character (word)	Character (record)	Field Description
1	1-8	1-8	Must contain bLABELbb.
2	1	9	Must be zero.
2	2-8	10-16	Multi-file identification.
3	1	17	Must be zero.
3	2-8	18-24	File identification.
4	1-3	25-27	Reel-Number (within file).
4	4-8	28-32	Date-Written (creation date).
5	1-2	33-34	Cycle-Number (to distinguish between identical runs on the same day).
5	3-7	35-39	Purge-Date (date this file can be destroyed).
5	8	40	Sentinel (1 = End-of-Reel, 0 = End-of-File).
6	1-5	41-45	Block Count.
6-7	6-8/1-4	46-52	Record Count.
7	5	53	Memory-Dump-Key (1 = memory dump follows label).
7-8	6-8/1-2	54-58	Physical Tape Number.

The remainder of the information contained in the label record varies for ALGOL and COBOL files as follows:

ALGOL FILES

Word	Character (word)	Character (record)	Field Description
8	3	59	Blocking Indicator: blocked = 1 for ⟨fixed logical⟩ ⟨fixed physical⟩ = 3 for ⟨fixed physical⟩ ⟨fixed logical⟩ not blocked = 0
8	4-8	60-64	Buffer Size (number of words).
9	1-5	65-69	Maximum Record Size (number of words).
9	6-8	70-72	Zeroes.

COBOL FILES

Word	Character (word)	Character (record)	Field Description
8	3-8	59-64	Reserved for File-Control-Routine - not currently being used.
9-?	1-?	65-??	Users Portion - may be of any format desired by the user and may be up to 8,120 characters in length for tape files, up to 16 characters in length for card file, and up to 56 characters in length for printer files.

APPENDIX C
CCMASK1 - CCMASK2 - MIXMASK - INFOMASK

CCMASKs are used to check the validity of a control card entered via data communications. The variables CCMASK are used if a special mask is not provided by REMOTE/USERS. The CCMASK Table shows the card column used to set the bit in REMOTE/USERS, the associated bit with each control function, and those bits which are set in the standard mask. The MIXMASK is used in a similar way to check those input messages which may or must have a mix number preceding them. INFOMASK1 and INFOMASK2 are used to check the validity of those input messages which do not include a mix index. The three mask tables are now presented.

The format for the CCMASK Card is as follows:

Column	Word	CCMASK1 Bit	Standard Mask
1-23	Not used	0-22	---
24	INFO	23	no
25	USE	24	no
26	RELEASE	25	no
27	FREE	26	no
28	PUBLIC	27	no
29	USER	28	no
30	RUN	29	yes
31	COMPILE	30	yes
32	EXECUTE	31	yes
33	DUMP	32	no
34	UNLOAD	33	no
35	ADD	34	no
36	LOAD	35	no
37	REMOVE	36	no
38	CHANGE	37	no
39	UNIT	38	no
40	END	39	yes
41	DATA	40	no
42	LABEL	41	no
43	FILE	42	yes
44	EXPIRED	43	no
45	Not used	44	---
46	Not used	45	---
47	Not used	46	---
48	Not used	47	---

Column	Word	CCMASK2 Bit	Standard Mask
49	Not used	0	---
50	Not used	1	---
51	PROCESS	2	yes
52	IO	3	yes
53	PRIORITY	4	no
54	COMMON	5	yes
55	CORE	6	no
56	STACK	7	no
57	SAVE	8	yes
58	Not used	9	---
59	Not used	10	---
60	Not used	11	---
61	ALGOL	12	yes
62	XALGOL	13	no
63	FORTRAN	14	yes
64	LONALG	15	yes
65	BASIC	16	no
66	Not used	17	---
67	WITH	18	no
68	COBOL	19	yes
69	LIBRARY	20	no
70	SYNTAX	21	no
71	FROM	22	no
72	TO	23	no

The format of the MIXMASK Card is as follows:

Column	Word	MIXMASK Bit	Standard Mask
1	Not used	0	---
2	DS	1	no
3	IL	2	no
4	OU	3	no
5	OK	4	no
6	FM	5	no
7	AX	6	no
8	FR	7	no
9	OF	8	no
10	TI	9	yes
11	WY	10	yes
12	RM	11	no
13	UL	12	no
14	ST	13	no
15	IN	14	no
16	OT	15	yes
17	QT	16	no
18	PR	17	no
19	PS	18	no

Column	Word	MIXMASK Bit	Standard Mask
20	XS	19	no
21	ES	20	no
22	SM	21	yes
23	HR	22	yes
24	CT	23	no
25	XT	24	no
26	TL	25	no
27	SS	26	no
28	WU	27	no
29	WA	28	no
30	HM	29	no
31	CU	30	no

The format of the INFOMASK Card is as follows:

Column	Word	INFORMASK1 Bit	Standard Mask
1	Not used	0	---
2	PG	1	no
3	MX	2	yes
4	DD	3	no
5	RW	4	no
6	PD	5	yes
7	DB	6	no
8	DP	7	no
9	DT	8	no
10	DS	9	no
11	PT	10	no
12	RS	11	no
13	EI	12	yes
14	CC	13	yes
15	PB	14	no
16	RY	15	no
17	TR	16	no
18	OL	17	yes
19	LN	18	no
20	WD	19	yes
21	WT	20	yes
22	LR	21	no
23	RO	22	no
24	SO	23	no
25	TO	24	yes
26	SV	25	no
27	LD	26	no
28	CD	27	yes
29	RD	28	no
30	RN	29	no
31	ED	30	no
32	CI	31	no
33	TF	32	yes

Column	Word	INFOMASK1 Bit	Standard Mask
34	SF	33	no
35	TS	34	yes
36	RR	35	no
37	QV	36	no
38	EX	37	yes
39	PI	38	yes
40	LO	39	yes
41	LI	40	yes
42	SS	41	yes
43	SM	42	yes
44	HM	43	yes
45	TC	44	yes
46	ZZ	45	yes
47	BO	46	yes
48	WP	47	no

Column	Word	INFORMASK2 Bit	Standard Mask
49	Not used	0	---
50	WU	1	no
51	LF	2	no
52	LC	3	no
53	LS	4	no
54	XI	5	no
55	WR	6	no
56	WM	7	yes
57	BK	8	no
58	BS	9	no
59	US	10	no
60	SC	11	no
61	CL	12	no
62	QT	13	no
63	WI	14	no
64	CU	15	no

APPENDIX D

USASCII X3.4 - 1967 STANDARD CODE

Bits b_7 b_6 b_5 →					0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b_4	b_3	b_2	b_1	Row ↓ Column →	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	8	6	F	V	f	v
0	1	1	1	7	BEL	ETB	/	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	'		L	\	l	¦
1	1	0	1	13	CR	GS	-	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	—	o	DEL

NUL Null

SOH Start of Heading (CC)

STX Start of Text (CC)

ETX End of Text (CC)

EOT End of Transmission (CC)

ENQ Enquiry (CC)

ACK Acknowledge (CC)

BEL Bell (audible or atten- signal)

BS Backspace (FE)

HT Horizontal Tabulation (punched card skip) (FE)

DLE Data Link Escape (CC)

DC1 Device Control 1

DC2 Device Control 2

DC3 Device Control 3

DC4 Device Control (stop)

NAK Negative Acknowledge (CC)

SYN Synchronous Idle (CC)

ETB End of Transmission Block (CC)

CAN Cancel

EM End of Medium

LF	Line Feed (FE)	SS	Start of Special Sequence
VT	Vertical Tabulation (FE)	ESC	Escape
FF	Form Feed (FE)	FS	File Separator (IS)
CR	Carriage Return (FE)	GS	Group Separator (IS)
SO	Shift Out	RS	Record Separator (IS)
SI	Shift In	US	Unit Separator (IS)
DEL	Delete*		

NOTE

(CC) Communication Control.

(FE) Format Effector.

(IS) Information Separator.

The numbers in the left hand vertical column are the count contained in DA1F through DA4F or DB1F through DB4F. The numbers in the top horizontal row are the count contained in Dn5F through Dn6F.

	0	1	2	3
0	•	P / DLE	bl.	0
1	A	Q / DC1	DISCON. ≥	1
2	B / STX	R / DC2	"	2
3	C / ETX	S / DC3	#	3
4	D / EOT	T / DC4	$	4
5	E / ENQ	U / NAK	%	5
6	F / ACK	V / SYN	&	6
7	G	W	≤	7
8	H	X	(	8
9	I	Y	)	9
10	J	Z	*	:
11	K	[	+	;
12	L	MULTI x	,	<
13	M	]	-	=
14	N / SO	↑	.	>
15	O / SI	← QM	/	?

*In the strict sense, DLE is not a control character.

INDEX

BURROUGHS CORPORATION
DATA PROCESSING PUBLICATIONS
REMARKS FORM

TITLE: _____ FORM: _____

_____ DATE: _____

_____

CHECK TYPE OF SUGGESTION:

☐ ADDITION ☐ DELETION ☐ REVISION ☐ ERROR

GENERAL COMMENTS AND/OR SUGGESTIONS FOR IMPROVEMENT OF PUBLICATION:

FROM: NAME _____ DATE _____

TITLE _____

COMPANY _____

ADDRESS _____

_____

cut along ted line

STAPLE

FOLD DOWN SECOND FOLD DOWN

Postage Will Be Paid by Addressee

No Postage Stamp Necessary If Mailed in the United States

BUSINESS REPLY MAIL
First Class Permit No. 817, Detroit, Mich. 48232

Burroughs Corporation
6071 Second Avenue
Detroit, Michigan 48232

attn: Sales Technical Services
Systems Documentation

FOLD UP FIRST FOLD UP

1042462 6-69 Printed in U. S. America