BURROUGHS
B5700
INFORMATION PROCESSING SYSTEMS
MASTER CONTROL PROGRAM
REFERENCE MANUAL

BURROUGHS CORPORATION
BUSINESS MACHINES GROUP
LARGE SYSTEMS SUPPORT
DETROIT, MICHIGAN

COPYRIGHT (C) 1969, 1971, 1972 BURROUGHS CORPORATION

AA119117

BURROUGHS CORPORATION BELIEVES THE INFORMATION IN THIS MANUAL TO BE ACCURATE AND RELIABLE, AND MUCH CARE HAS BEEN TAKEN IN ITS PREPARATION. HOWEVER, THE CORPORATION CANNOT ACCEPT ANY RESPONSIBILITY, FINANCIAL OR OTHERWISE, FOR ANY CONSEQUENCES ARISING OUT OF USE OF THIS MATERIAL. THE INFORMATION CONTAINED HEREIN IS SUBJECT TO CHANGE. REVISIONS MAY BE ISSUED TO ADVISE OF SUCH CHANGES AND/OR ADDITIONS.

CORRESPONDENCE REGARDING THIS DOCUMENT SHOULD BE FORWARDED TO MANAGER, LARGE SYSTEMS SUPPORT, SALES TECHNICAL SERVICES, BURROUGHS CORPORATION, 6071 SECOND AVENUE, DETROIT, MICHIGAN 48232.

PREFACE

THIS MANUAL IS PROVIDED TO REDUCE THE AMOUNT OF TIME REQUIRED TO SOLVE SYSTEM PROBLEMS. IT HAS PROVED TO BE OF MUST BENEFIT WHEN USED IN CONJUNCTION WITH A LISTING AND CROSS REFERENCE OF THE MCP.

THIS DOCUMENT CONTAINS INFORMATION CONCERNING PRIMARY SYSTEM FUNCTIONS, GENERAL SYSTEM OPERATION, AND DETAIL INFORMATION CONCERNING THE MCP. ALL INFORMATION IS IN ALPHABETIC SEQUENCE TO FACILLITATE REFERENCE.

INTRODUCTION

THE MASTER CONTROL PROGRAM (MCP) IS A MODULAR SUPERVISORY COMPUTER PROGRAM WHICH TAKES OVER REPETITIVE FUNCTIONS, SOME BEING LOGICALLY COMPLEX, TO MAKE COMPUTER PROGRAMMERS AND OPERATORS 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 B5500/B5700 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.

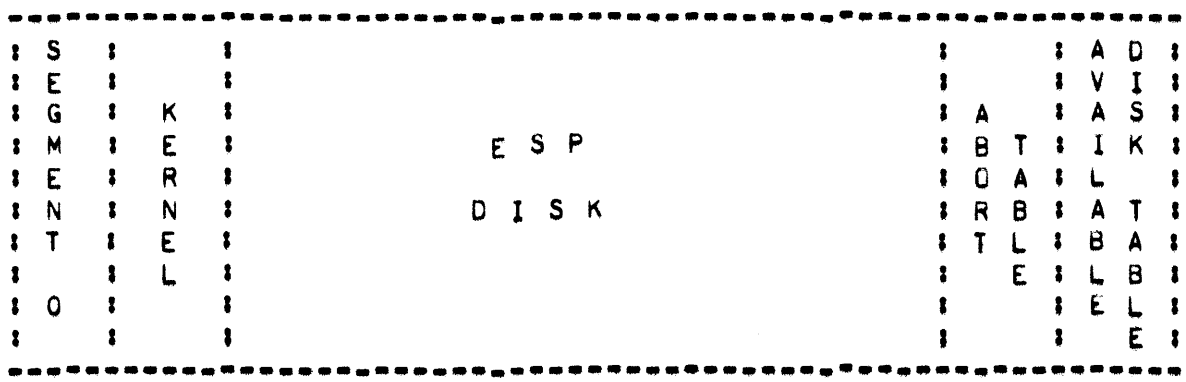
ALL VERSIONS OF MCP HANDLE THE PRIMARY FUNCTIONS OF CONTROL PROGRAMS: LOADING, INTERRUPTS, I/O CONTROL, SELECTION AND INITIATION OF PROGRAM I/O, ERROR CONDITIONS, SYSTEM LOG, STORAGE ALLOCATION, OVERLAY, AND MULTIPROGRAMMING. THE MCP IS COMPOSED OF TABLES (I.E., ARRAYS) AND OF PROCEDURES WITH AN OUTER BLOCK WHICH COORDINATES THEIR OPERATION.

ABORT TABLE

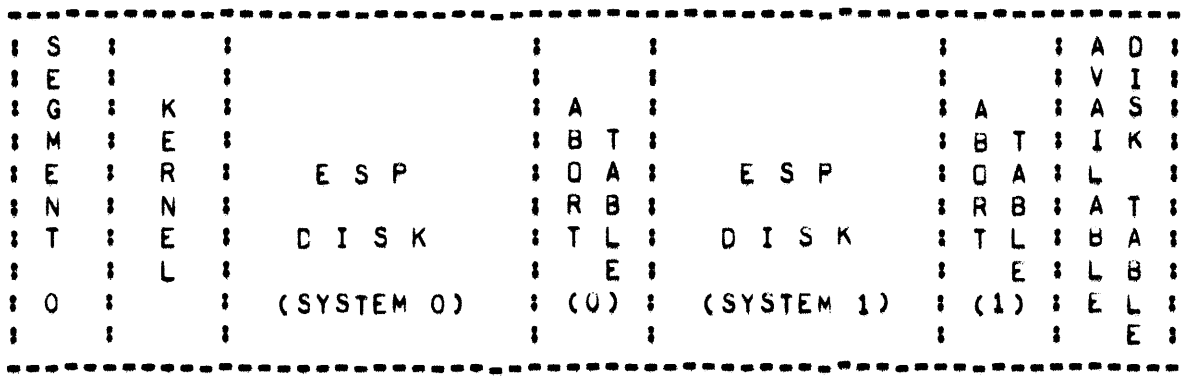
THE ABORT TABLE IS USED TO CREATE SYSTEM LOG ENTRIES FOR JOBS THAT WERE RUNNING WHEN A SYSTEM HALT OCCURRED. THE ABORT TABLE IS SIX SEGMENTS LONG AND IS LOCATED BETWEEN THE END OF ESP DISK AND THE AVAILABLE DISK TABLE. THE ABORT TABLE IS SIX SEGMENTS (180 WORDS) LONG.

ON SHAREDISK SYSTEMS, EACH SYSTEM HAS ITS OWN ESP DISK AND ABORT TABLE. THE AREA BETWEEN THE "KERNEL" AND THE AVAILABLE DISK TABLE - 6 IS EVENLY DIVIDED FOR EACH SYSTEM.

LOCATION OF ABORT TABLE ON NON-SHAREDSK SYSTEM



LOCATION OF ABORT TABLE ON SHAREDISK SYSTEM (2 SYSTEMS)



- ABORT TABLE -

FORMAT:

FIRST 3 SEGMENTS:

WORD	CONTENTS
0	XCLOCK
1	DATE
2	"ABORT"

NEXT THREE ENTRIES ARE REPEATED FOR EACH JOB IN THE MIX. ENTRIES ARE ZEROED IF MIX NUMBER NOT ASSIGNED.

RELATIVE LOC.	CONTENTS
3×MIX	PROCESS TIME
3×MIX + 1	I-O TIME
3×MIX + 2	"IDLETIME" (FROM THE JAR)

SECOND 3 SEGMENTS:

RELATIVE LOC.	CONTENTS
3×MIX + 90	FIRST NAME OF OBJECT PROGRAM
3×MIX + 91	SECOND NAME OF OBJECT PROGRAM
3×MIX + 92	[1:23] START TIME [24:24] POINTER TO CONTROL CARD IN "ESPDISK"
3×MIX + 180	USERCODE

- ABORTFILE, ABORTSPEC, ACTDATE, ACTUALIOERR -

ABORTFILE

CONTAINS CURRENT RECORD NUMBER IN USE IN THE ABORT PORTION OF THE REMOTE/LOG.

ABORTSPEC

USED BY REMOTELOGGER AND INITIALIZEDCFILE ROUTINES TO PROCESS THE REMOTE ABORT LOG.

ACTDATE

CONTAINS THE CURRENT DATE IN THE FORM MMDDYY.

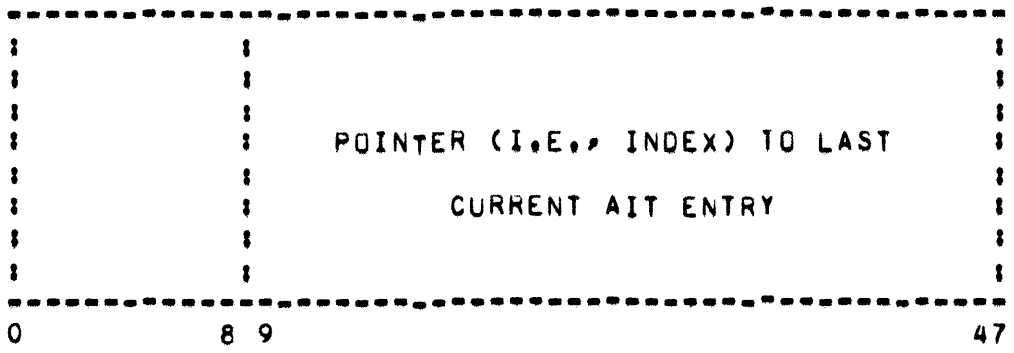
ACTUALIOERR

ACTUALIOERR(U,R) IS A PROCEDURE WHICH HANDLES I/O ERRORS ON LOGICAL UNIT "U" WHICH HAD A RESULT DESCRIPTOR "R", AND CLEARS THEM, TYPICALLY WITH A SPO MESSAGE.

ARRAY INFORMATION TABLE

ONE "AIT" IS ASSOCIATED WITH EACH "ALGOL" PROGRAM THAT DECLARES ONE OR MORE FILES OR ARRAYS.

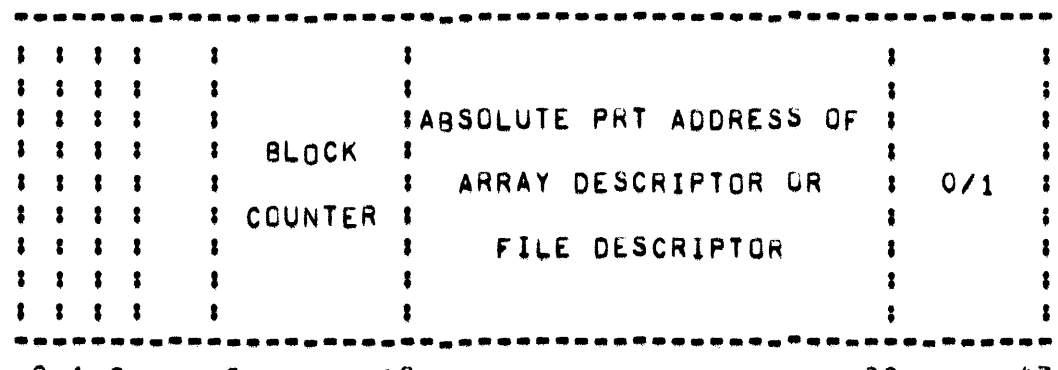
WORD 0



FIELD	CONTENTS	DESCRIPTION
[0:9]	0	
[19:39]	INTEGER	INDEX TO LAST CURRENT "AIT" ENTRY.

- ARRAY INFORMATION TABLE -

REMAINING WORDS



FIELD	CONTENTS	DESCRIPTION
[0:1]	0	FLAG BIT
[1:2]	0,1,2	0 = ARRAY, 1 = RUN-TIME ERROR ENTRY 2 = FILE 3 = TASK ARRAY
[3:5]	0-31	NUMBER OF DIMENSIONS
[8:10]	0-1023	BLOCK COUNTER (I.E., NESTING DEPTH WHEN FILE OR ARRAY IS DECLARED)
[18:15]	ADDRESS	ABSOLUTE ADDRESS OF FILE OR ARRAY DESCRIPTOR OR OF "RTE" CELL CONTAINING LABEL
[33:15]	0/1	SAVE INDICATOR (1=SAVE) FOR ARRAYS
	1,2,4,8,16	FOR RTE ERROR TYPE 1- INTEGER OVERFLOW 2- EXPONENT OVERFLOW 4- INVALID INDEX 8- DIVIDE BY ZERO 16- FLAG BIT

ENTRIES ARE PLACED IN THE "AIT" BY LEXICOGRAPHICAL OR BLOCK LEVEL (ENTRY.[8:10]). THE MCP PROCEDURE "ASR" SCANS THE "AIT" FROM THE BACK TO THE FRONT AND PROVIDES FOR THE RETURN OF SPACE FOR ARRAYS AND FILES.

ARRAY SPACE DECLARATION

THE FOLLOWING PARAMETERS ARE REQUIRED IN THE STACK:

1. MKSCW.
2. DESCRIPTORS POINTING TO THE ARRAY DESCRIPTORS FOR EACH ARRAY BEING DECLARED.
3. SIZES OF THE ARRAY DIMENSIONS.
4. NUMBER OF DIMENSIONS.
5. NUMBER OF DESCRIPTORS.
6. "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).

3 "SAVE" AND "OWN" ARRAY SPACE.

- ATTACHED, AUXCODE, AUXDATA -

ATTACHED

DESCRIPTOR POINTING TO ATTACHED ARRAY, WHICH CONTAINS INFORMATION BY MIX NUMBER CONCERNING WHICH B487 TU/BUFF-S ARE ASSOCIATED WITH JOBS.

E = 3:9
D = 12:9
C = 21:9
B = 32:9
A = 39:9

AUXCODE

DESCRIPTOR POINTING TO AUXCODE ARRAY USED TO KEEP TRACK OF HOW MUCH AUXILIARY MEMORY IS IN USE BY MIX INDEX FOR CODE SEGMENTS.

AUXDATA

DESCRIPTOR POINTING TO AUXDATA ARRAY USED TO KEEP TRACK OF HOW MUCH AUXILIARY MEMORY IS IN USE BY MIX INDEX FOR DATA SEGMENTS.

AUXILIARY MEMORY AND RESOURCE ALLOCATION

ABSTRACT

A METHOD FOR UTILIZING THE INHERENT ADVANTAGES OF AN AUXILIARY CORE MEMORY STORAGE UNIT IS INTRODUCED AND DISCUSSED, AND DIFFERENCES BETWEEN THIS METHOD AND TRADITIONAL ALLOCATION METHODS FOR DISK STORAGE ARE ANALYZED. THIS DOCUMENT PRESENTS THE SYNTAX AND SEMANTICS OF CERTAIN NEW CONSTRUCTS IN COBOL AND ALGOL WHICH PERMIT THE PROGRAMMER TO ASSIST THE OPERATING SYSTEM IN MANAGEMENT OF MAIN AND AUXILIARY MEMORY.

THIS DOCUMENT IS EXPECTED TO SERVE AS A PROGRAMMING LANGUAGE SPECIFICATION FOR APPLICATIONS PROGRAMMERS, AND AS A SYSTEM DESIGN SPECIFICATION AND REFERENCE DOCUMENT FOR SOFTWARE PERSONNEL INVOLVED IN MAINTAINING OR MODIFYING THE B5500/B5700 MCP. FAMILIARITY WITH EITHER COBOL OR ALGOL IS ASSUMED, AND, FOR THE MCP REFERENCE PORTION, A GOOD GENERAL UNDERSTANDING OF THE MCP FUNCTIONS INVOLVED.

INTRODUCTION

THE BURROUGHS B5500/B5700 INFORMATION PROCESSING SYSTEM IS A DISK-FILE ORIENTED VIRTUAL MEMORY COMPUTER SYSTEM UTILIZING THE TECHNIQUES OF MULTIPROGRAMMING, MULTIPROCESSING AND RE-ENTRANT CODE STRUCTURES TO FACILITATE PARALLEL EXECUTION OF PROCESSOR AND INPUT-OUTPUT FUNCTIONS. HOWEVER, IT BEGAN LIFE AS THE BURROUGHS B5000 - A DRUM ORIENTED SYSTEM. TO THIS PARENTAGE IT OWES THE TECHNIQUES OF OVERLAY DRUM AND DISK ALLOCATION.

WHEN THE DISK-BASED OPERATION SYSTEM FOR THE B5500 WAS RELEASED, ALLOCATION OF OVERLAY DRUM STORAGE WAS ACCOMPLISHED IN EXACTLY THE SAME MANNER AS ALLOCATION OF OVERLAY DISK. THE OPERATIVE PHILOSOPHY, HOWEVER, WAS THAT DRUMS WOULD NOT NORMALLY BE USED FOR OVERLAY STORAGE. THEY WOULD CONTINUE TO CONTAIN THE DRUM SYSTEM SOFTWARE AND BE USED UNDER THE DRUM OPERATING SYSTEM AND BE RETURNED TO BURROUGHS UPON SUCCESSFUL CONVERSION OF ALL PROGRAMS TO THE DISK ENVIRONMENT.

WITH THE DEVELOPMENT OF B6500 MEMORY AS AN AUXILIARY MEMORY STORE FOR THE B5500 CAME THE REALIZATION THAT MORE SOPHISTICATED ALLOCATION AND DEALLOCATION TECHNIQUES MUST BE DEVELOPED. THE DISSIMILARITIES OF "CORE/DRUM" AUXILIARY MEMORY AND DISK STORAGE ARE LEGION: CAPACITY, TRANSFER RATE AND LATENCY ARE SOME OF THESE. THE HIGH COST-PER-BIT OF THE AUXILIARY MEMORY NECESSITATED AN ATTEMPT TO

UTILIZE IT AS MUCH AS POSSIBLE.

THE DESIGN OF DISK STORAGE ALLOCATION HAD AS ITS GOALS THE USE OF MINIMAL CORE STORAGE TO CONTROL THE MAXIMAL AMOUNT OF OVERLAY DISK, AND THIS OBJECTIVE WAS ACHIEVED. HOWEVER, A TRADE-OFF OF SOME OTHER DESIRABLE CHARACTERISTIC WAS INEVITABLE. THE DECISION WAS MADE TO BE RELATIVELY PROFLIGATE IN THE USE OF QUANTITIES OF DISK, AND TO AVOID GARBAGE COLLECTION ENTIRELY, IN THE INTERESTS OF SPEED AND MINIMAL CORE REQUIREMENTS.

TO EFFECTIVELY UTILIZE THE AUXILIARY MEMORY STORAGE CURRENTLY AVAILABLE, THE FOLLOWING DESIGN PARAMETERS WERE ESTABLISHED:

- THE ALLOCATION AND DEALLOCATION OF AUXILIARY MEMORY MUST BE ACCOMPLISHED WITHIN SEVERE TIME CONSTRAINTS TO TAKE ADVANTAGE OF THE INCREASED TRANSFER RATE AND NEGLIGIBLE LATENCY OF AUXILIARY MEMORY.

- THE ALLOCATION/RETURN ALGORITHMS MUST OPERATE IN VERY SMALL INCREMENTS OF STORAGE, IN ORDER TO MAXIMIZE THE USE OF THE AUXILIARY MEMORY AS AN OVERLAY STORE. SUBJECT TO THIS CONSTRAINT THE AMOUNT OF MAIN MEMORY USED MUST BE MINIMIZED.

- THE AUXILIARY MEMORY OVERLAY SYSTEM MUST OPERATE WITH AS LITTLE DISRUPTION AS POSSIBLE TO THE EXISTING ALGORITHMS RESPONSIBLE FOR THE DISK OVERLAY SYSTEM.

A SYSTEM HAS BEEN DESIGNED AND IMPLEMENTED WHICH SUCCESSFULLY MEETS THESE CRITERIA. AUXILIARY MEMORY IS ALLOCATED FROM (AND RETURNED TO) BIT TABLES IN MAIN MEMORY. THESE TABLES ARE ORGANIZED INTO CORRESPONDING COARSE AND FINE TABLES, SO THAT ALLOCATION OF AREAS CAN BE SWIFTLY CARRIED OUT FOR BOTH LARGE AND SMALL AREAS. AN ADDITIONAL DESIGN CONSIDERATION WAS THE USE OF COUNTERS TO ENABLE SWIFT DETERMINATION OF WHETHER AN AREA OF SUFFICIENT SIZE EXISTED IN THE APPROPRIATE TABLES WITHOUT PERFORMING AN ELABORATE AND EXHAUSTIVE SEARCH. AS A RESULT OF THIS DECISION, THE "SEARCH" TIME IS MATERIALLY REDUCED FOR ALL CASES, AT THE COST OF A VERY SLIGHT INCREASE IN TIME IN THOSE CASES WHERE AN AREA IS ACTUALLY ALLOCATED.

THE AUXILIARY MEMORY IS ALLOCATED IN "CHUNKS" OF SIXTEEN WORDS. ELEVEN BITS ARE THEREFORE SUFFICIENT TO CONTAIN A REFERENCE TO THE DESIRED MEMORY ADDRESS (FROM 0 THROUGH 32752); A SINGLE BIT IS REQUIRED TO SPECIFY WHICH AUXILIARY MEMORY UNIT IS TO BE USED, AND THREE BITS MAY BE USED AS A FLAG TO INDICATE THAT THIS IS AN AUXILIARY MEMORY REFERENCE, RATHER THAN AN OVERLAY DISK REFERENCE. THE RESULT OF THIS IS A DECREASE IN THE MAXIMUM AMOUNT OF OVERLAY STORAGE AVAILABLE FOR A JOB FROM A THEORETICAL LIMIT OF 945,000 WORDS PER PROGRAM TO A THEORETICAL LIMIT OF 890,536 WORDS PER PROGRAM. WHILE IT IS THEORETICALLY POSSIBLE TO WRITE A PROGRAM WHICH WOULD RUN UNDER THE PREVIOUS SYSTEM AND FAIL UNDER THE CURRENT

SYSTEM (FOR LACK OF OVERLAY SPACE). IMPROVEMENTS IN THE ALLOCATION ALGORITHM INVOLVED REDUCE THIS POSSIBILITY TO SUCH AN EXTENT THAT IT CAN BE IGNORED.

- AUXILIARY MEMORY AND RESOURCE ALLOCATION -

GENERAL DIFFERENCES

IN ORDER TO PROVIDE THE PROGRAMMER WITH THE ABILITY TO ASSIST IN MANAGING MEMORY AND OVERLAY STORAGE, THE SYNTAX OF THE ALGOL, COBOL, AND XALGOL LANGUAGES HAS BEEN EXTENDED. TWO MAJOR FEATURES HAVE BEEN ADDED, "READ-ONLY" ARRAYS AND THE ABILITY TO ALLOW PROGRAM CODE TO BE PLACED ON AUXILIARY MEMORY. A "READ-ONLY" ARRAY IS WRITTEN TO OVERLAY STORAGE (AUXMEM OR DISK) THE FIRST TIME IT IS OVERLAID. SUBSEQUENT OVERLAYS USE THE INFORMATION WHICH WAS WRITTEN ON THE FIRST OVERLAY, THUS SAVING THE TIME REQUIRED FOR THE WRITE.

PLEASE NOTE THAT THE "READ-ONLY" ARRAY FEATURE IS AVAILABLE FOR BOTH AUXMEM AND NON-AUXMEM SYSTEMS.

TWO SPO OPTIONS HAVE BEEN CREATED WHICH FURTHER GOVERN AUXILIARY MEMORY USAGE. IF THE OPTION "CODEOLAY" IS SET WHEN A PROGRAM IS INITIALIZING (GOING TO BOJ), ALL PROGRAM DESCRIPTORS WILL BE MARKED SO THAT PROGRAM SEGMENTS WILL BE WRITTEN TO AUXMEM WHEN OVERLAID IF SPACE IS AVAILABLE. THE OPTION "DATAOLAY", IF SET WHEN A PROGRAM IS INITIALIZING, ALLOWS DATA TO BE OVERLAID TO AUXMEM.

COBOL DIFFERENCES

THE COBOL LANGUAGE HAS, IN ITS STANDARDIZED FORM, PROVIDED A MECHANISM FOR THE PROGRAMMER TO SPECIFY THE PRIORITY OF SECTIONS OF HIS CODING IN THE PROCEDURE DIVISION. THIS SYNTAX HAS NOT BEEN MODIFIED, BUT A NEW SEMANTIC MEANING HAS BEEN GIVEN TO THE CONSTRUCT:

```
<SECTION-NAME> SECTION 1.
(OR <SECTION-NAME> SECTION 01.).
```

THE NEW MEANING ATTACHED TO THIS CONSTRUCT (WHICH WAS FORMERLY IGNORED BY THE COMPILER) IS THAT ALL PARAGRAPHS WITHIN THIS SECTION ARE TO BE MARKED SO THAT THEY WILL BE OVERLAID TO AUXILIARY MEMORY, RATHER THAN DISK, IF THIS IS POSSIBLE AT THE TIME THEY ARE OVERLAID. IF NO AUXILIARY MEMORY IS AVAILABLE, THE REQUEST MAY STILL BE FILLED AT THE NEXT OVERLAY.

SINCE A PROGRAM SEGMENT, ONCE WRITTEN TO AUXILIARY MEMORY, NORMALLY REMAINS THERE UNTIL THE JOB (OR JOBS IN THE CASE OF RE-ENTRANT JOBS) USING IT IS FINISHED, A MECHANISM MUST BE SUPPLIED TO ALLOW THE PROGRAMMER TO RELEASE THOSE PROGRAM SEGMENTS - AND DATA AREAS - WHICH ARE NO LONGER REQUIRED OR DESIRED TO BE RESIDENT ON AUXILIARY MEMORY. THIS MECHANISM IS IMPLEMENTED THROUGH THE NEW VERB, FORGET.

THE SYNTAX OF THE FORGET VERB IS AS FOLLOWS:

```
<MEMORY MANAGEMENT SENTENCE> ::=FORGET <IDENTIFIER LIST>
<IDENTIFIER LIST>            ::=<IDENTIFIER>/<IDENTIFIER>  (,)
                                <IDENTIFIER LIST>
<IDENTIFIER>                 ::=<DATA  NAME>/<PROCEDURE NAME> /
                                <SECTION NAME>
```

THE SEMANTICS OF THE FORGET SENTENCE REQUIRE THAT CERTAIN CONVENTIONS BE OBSERVED. PRIMARILY, THIS IS A RESTRICTION WHICH ALLOWS ONLY 01-LEVEL DATA TO BE RETURNED, SINCE THE ACTION TAKEN IS TO RETURN AN ENTIRE ARRAY ROW.

WHEN AN 01-LEVEL DATA NAME IS THE OBJECT OF A FORGET VERB, THE MEMORY AREA (IF ANY) CURRENTLY ASSIGNED TO THE CORRESPONDING ARRAY ROW IS RETURNED, AND ANY DISK OR AUXILIARY MEMORY STORAGE ASSIGNED AS AN OVERLAY AREA FOR THE ROW IS RETURNED. THE DESCRIPTOR IS MARKED AS NOT PRESENT, NEVER ACCESSED. NOTE THAT THIS IMPLIES THAT ANY "VALUE IS" CLAUSE WHICH SET UP CONSTANT CONTENTS FOR THE ARRAY IS NOT RE-EXECUTED, AND, AT NEXT ACCESS, EACH CHARACTER IN THE ARRAY ROW WILL BE ZERO.

WHEN A PROCEDURE-NAME IS THE OBJECT OF A FORGET VERB, THE CODE

SEGMENT ASSOCIATED WITH THAT PROCEDURE IS OVERLAID, AND ANY AUXILIARY MEMORY ASSOCIATED WITH THIS CODE SEGMENT IS RETURNED. THE SEGMENT DICTIONARY IS LEFT MARKED SO THAT IF THE CODE SEGMENT IS SUBSEQUENTLY MADE PRESENT AND OVERLAID IT WILL ONCE AGAIN BE OVERLAID TO AUXILIARY MEMORY.

ALGOL DIFFERENCES

IN ORDER TO PROVIDE THE ALGOL PROGRAMMER WITH THE ABILITY TO ASSIST IN MANAGING MEMORY AND OVERLAY STORAGE, A FEW ADDITIONAL STATEMENTS AND A NEW DECLARATION HAVE BEEN ADDED TO BURROUGHS EXTENDED ALGOL FOR THE B5500.

THE ADDITIONAL DECLARATION NOW ALLOWED IS THE "SAVE" PROCEDURE DECLARATION. THIS IS ACCOMPLISHED BY PREFIXING THE RESERVED WORD "SAVE" TO A PROCEDURE (OR FUNCTION) DECLARATION. CORRESPONDENCE OF DECLARATIONS IS CHECKED BETWEEN FORWARD AND ACTUAL DECLARATIONS; IT IS THE RESPONSIBILITY OF THE PROGRAMMER TO ENSURE THAT FORWARD DECLARATIONS REFLECT THE STATUS OF THE ACTUAL DECLARATION. (A SYNTAX ERROR WILL BE PRODUCED ON A MISMATCH OF THIS NATURE, HOWEVER). THE SEMANTICS OF ALGOL "SAVE" PROCEDURES IS THE SAME AS THAT OF COBOL 01-PRIORITY SECTIONS: THE SEGMENT DICTIONARY ENTRY FOR THE CODE SEGMENT (OR SEGMENTS, DEPENDING UPON BLOCK STRUCTURE) IS MARKED TO INDICATE THAT, THE FIRST TIME IT IS OVERLAID, IT SHOULD BE WRITTEN OUT TO AUXILIARY MEMORY, AND THAT SUBSEQUENT PRESENCE-BIT ACTION WILL CAUSE IT TO BE READ IN FROM AUXILIARY MEMORY RATHER THAN FROM DISK.

TO ALLOW THE PROGRAMMER THE ABILITY TO RETURN AUXILIARY MEMORY OVERLAY STORAGE WHICH IS NO LONGER REQUIRED, THE RELEASE STATEMENT HAS BEEN EXPANDED TO INCLUDE THE FOLLOWING NEW SYNTAX:

```
<MEMORY MANAGEMENT SENTENCE> ::=RELEASE(<MEMORY ELEMENT>)
<MEMORY ELEMENT>             ::=<ARRAY ROW DESIGNATOR>/
                                <PROCEDURE DESIGNATOR>/ <FUNCTION
                                DESIGNATOR>
```

FOR EITHER AN ARRAY ROW OR A CODE SEGMENT, THE ACTIONS TAKEN ARE AS FOLLOWS: ANY MEMORY IN USE FOR THE DATA OR CODE SEGMENT IS RETURNED, ANY AUXILIARY MEMORY IN USE FOR THE DATA OR CODE SEGMENT IS RETURNED, AND, IN THE CASE OF DATA, ANY OVERLAY DISK IN USE FOR THE ARRAY ROW IS RETURNED. FOR DATA, THE DESCRIPTOR IS MARKED AS ABSENT, NEVER ACCESSED; THIS WILL CAUSE A NEW ROW OF THE SAME SIZE TO BE ALLOCATED AND ZEROED OUT SHOULD THIS ARRAY ROW SUBSEQUENTLY BE REFERENCED. FOR PROGRAM SEGMENTS, THE SEGMENT DICTIONARY IS LEFT MARKED SO THAT, IF THE SEGMENT IS LATER MADE PRESENT AND THEN OVERLAID, AN ATTEMPT WILL BE MADE TO ALLOCATE AUXILIARY MEMORY FOR ITS STORAGE.

AN ADDITIONAL FACILITY HAS BEEN PROVIDED IN ALGOL TO ALLOW THE PROGRAMMER TO UTILIZE DYNAMIC READ-ONLY ARRAYS. THE TERM "DYNAMIC READ-ONLY ARRAY" SEEMS TO BE IN CONFLICT WITH ITSELF: THE MEANING IS THAT AN ARRAY CAN BE MADE READ-ONLY, AND LATER ALTERED (WHICH, OF COURSE, DESTROYS ITS READ-ONLY NATURE) THROUGH PROGRAM CONTROL.

A NEW STATEMENT, THE UNLOCK STATEMENT, AND A MODIFICATION OF AN EXISTING STATEMENT, THE LOCK STATEMENT, HAVE BEEN ADDED TO BURROUGHS EXTENDED ALGOL. THE SYNTAX OF THESE STATEMENTS IS AS FOLLOWS:

```
<READ-ONLY CONTROL STATEMENT>          ::=LOCK(<ARRAY ROW
                                    DESIGNATOR>) / UNLOCK(<ARRAY ROW
                                    DESIGNATOR>)
```

THE SEMANTICS OF THESE STATEMENTS CONTROL THE MCP ACTION TO BE TAKEN UPON OVERLAY OF THE DESIGNATED ARRAY ROW. WHEN AN ARRAY ROW IS LOCKED, THE DESCRIPTOR IS MARKED TO INDICATE THAT IT HAS BEEN MADE READ-ONLY BY THE PROGRAMMER. WHEN THE ARRAY ROW IS NEXT OVERLAID, NORMAL OVERLAY ACTION OCCURS AND THE DESCRIPTOR IS MARKED TO INDICATE THAT THE DISK CONTAINS A VALID COPY OF THIS READ-ONLY DATA. ON SUBSEQUENT OVERLAY OF THE ARRAY ROW, THE DATA IS NOT WRITTEN TO DISK, AS A VALID COPY ALREADY EXISTS THERE. NOTE THAT IT IS SPECIFICALLY PROHIBITED TO CHANGE AN ELEMENT OF A READ-ONLY ARRAY. ERRATIC RESULTS MAY BE PREDICTED FROM THE VIOLATION OF THIS PROHIBITION. IN ORDER TO CHANGE THE CONTENTS OF A READ-ONLY ARRAY, THE PROGRAMMER MUST EXPLICITLY UNLOCK THE DESIRED ROW BEFORE CHANGING IT. IF DESIRED, THE ROW MAY THEN AGAIN BY LOCKED.

MCP REFERENCE DOCUMENT

NARRATIVE DESCRIPTION

THE MCP ALLOCATES AUXILIARY MEMORY FROM COARSE AND FINE "BIT TABLES". ONE SUCH TABLE IS CONSTRUCTED FOR EACH AVAILABLE AUXILIARY MEMORY UNIT AT HALT/LOAD TIME. IF AN AUXILIARY MEMORY UNIT IS PLACED ON-LINE AFTER THE SYSTEM HAS BEEN HALT/LOADED, IT WILL BECOME AVAILABLE FOR USE AFTER ALL JOBS IN THE MIX GO TO END OF JOB. THE BIT TABLES ARE INDEXED THROUGH THE MCP ARRAY, CTABLE[*]. THIS TABLE CONTAINS DESCRIPTORS POINTING TO THE ACTUAL COARSE AND FINE TABLES.

A CTABLE ENTRY IS EITHER 0 (IF THE APPROPRIATE AUXILIARY MEMORY UNIT IS NOT AVAILABLE), OR IS A DESCRIPTOR POINTING TO THE COARSE TABLE OR FINE TABLE FOR THAT UNIT.

THE STRUCTURE OF WORDS IN THE FINE TABLES AND COARSE TABLES IS THE SAME. THIS STRUCTURE IS:

BITS	CONTENTS
0:3	0
3:6	LONG RUN COUNT
9:7	BIT VECTOR OVERLAP FROM PREVIOUS WORD
16:32	BIT VECTOR FOR THIS WORD

THE LONG RUN COUNT IS THE COUNT OF THE NUMBER OF CONTIGUOUS "ONE" BITS IN THE [9:39] FIELD OF THIS WORD. THIS RUN COUNT IS SUBJECT TO A MAXIMUM OF EIGHT, SINCE THIS IS THE MAXIMUM LENGTH OF INTEREST TO THE ALLOCATION ALGORITHMS. THIS RUN LENGTH IS DETERMINED EACH TIME AUXILIARY MEMORY STORAGE IS ALLOCATED FROM OR RETURNED TO THIS WORD.

THE [9:7] FIELD IS USED TO CONTAIN INFORMATION FROM THE [41:7] FIELD OF THE PRECEDING WORD. SINCE NO PART OF THE ALLOCATION ALGORITHM EVER SEARCHES FOR MORE THAN EIGHT CONTIGUOUS BITS, THIS SEVEN-BIT OVERLAP IS SUFFICIENT TO ALLOW ALL SEARCHES TO BE MADE UPON A SINGLE WORD, WITHOUT CONSIDERATIONS OF WORD BOUNDARIES.

THE DISTINCTION BETWEEN FINE SEARCH AND COARSE SEARCH IS MADE PURELY ON THE BASIS OF SIZE. IF THE SIZE OF THE AREA REQUIRED IS LESS THAN 128 WORDS, A FINE SEARCH IS MADE. IF IT IS 128 WORDS OR MORE, A COARSE SEARCH IS MADE. COARSE SEARCHES ARE MADE FROM THE "FRONT" OF THE TABLE (UNIT A PRECEDING UNIT B), AND FINE SEARCHES ARE MADE FROM THE "BACK" OF THE TABLE. THIS IS A MINIMAL SORT OF INSURANCE AGAINST CHECKERBOARDING.

EACH CODE SEGMENT OR DATA SEGMENT OVERLAID TO AUXILIARY MEMORY IS

PROCEDED BY A ONE-WORD IDENTIFICATION, COMMONLY TERMED A "LINK", SINCE IT IS CONSTRUCTED IN THE SECONDARY MEMORY LINK OF THE CORE AREA INVOLVED. THE FORMAT OF THESE LINKS IS AS FOLLOWS:

BITS	MEANING
1:1	= 1, CODE SEGMENT
	= 0, DATA SEGMENT
3:15	DISK ADDRESS (FOR CODE SEGMENTS ONLY)
18:15	SIZE OF AREA (IN WORDS)
33:15	MIX INDEX OF MOTHER JOB

THESE DATA ARE USED IN THE "CASUALTY RECOVERY" PROCESS DETAILED BELOW, AND TO RESTORE THE PROPER DISK ADDRESS WHEN A CODE SEGMENT IS RELEASED FROM AUXILIARY MEMORY.

THE NORMAL DISK-BASED OVERLAY SYSTEM ALLOCATES A LARGE AMOUNT OF CONTIGUOUS DISK (15,000 WORDS) TO A PROGRAM AS REQUIRED, AND THEN SUB-ALLOCATES SECTIONS OF THIS AREA TO MEET INDIVIDUAL REQUESTS. ONE OF THE NOTABLE ADVANTAGES OF THIS TECHNIQUE IS THAT NO "GARBAGE COLLECTION" IS REQUIRED. AT END OF JOB, ALL OF THIS DISK CAN BE RETURNED TO THE AVAILABLE DISK TABLES FROM THE MEMORY-RESIDENT ARRAY OF ADDRESSES ASSOCIATED WITH THE TERMINATING JOB. SINCE AUXILIARY MEMORY IS ASSIGNED IN VERY SMALL CHUNKS (AS SMALL AS 16 WORDS AT A TIME), A GARBAGE COLLECTION MECHANISM IS NECESSARY TO RETURN THOSE AUXILIARY MEMORY SEGMENTS WHICH ARE IN USE AT END OF JOB. MOST NORMAL DATA SEGMENTS WILL HAVE BEEN THROUGH THE BLOCK EXIT (ASR) MECHANISM, BUT DATA (SUCH AS FORMATS) WHICH LEAVE ENTRIES IN THE SEGMENT DICTIONARY, AND GLOBAL ARRAYS SUCH AS THE AIT AND DAT WILL NOT HAVE BEEN RELEASED. "COM5" TAKES CARE OF THIS ACTION. IF, HOWEVER, ADDITIONAL AUXILIARY MEMORY IS FOUND TO BE ASSIGNED TO THE JOB, A ROUTINE CALLED "AUXILIARYMEMORYCASUALTYRECOVERY" IS INVOKED TO RECOVER THE AUXILIARY MEMORY. THIS IS ACCOMPLISHED THROUGH COMPLEMENTING THE AVAILABLE AUXILIARY MEMORY TABLE (THUS CREATING AN "IN-USE" TABLE) AND, USING THE LINK WORDS, READING THROUGH THE AUXILIARY MEMORY AND RETURNING DATA AREAS ASSIGNED TO THE APPROPRIATE JOB.

TECHNICAL INFORMATION

THE ALLOCATION AND RETURN OF AUXILIARY MEMORY IS EFFECTED THROUGH SIX NEW MCP ROUTINES, AND THROUGH MINOR AND MAJOR REVISIONS OF PRE-EXISTING ROUTINES. THE DESIGN AND CODING OF THESE ROUTINES IS DISCUSSED IN DETAIL IN THIS SECTION. IT SHOULD BE BORNE IN MIND, HOWEVER, THAT IN THE LAST RESORT THE BEST AND MOST COMPLETE REFERENCE DOCUMENT IS STILL THE MCP LISTING ITSELF.

"FILLORKILL" IS A "SAVE" PROCEDURE WHICH IS ANCILLARY TO THE ALLOCATION AND DEALLLOCATION ROUTINES. ITS PRIMARY FUNCTION IS TO PERFORM THE BIT-TWIDDLING NECESSARY TO REMOVE SPACE FROM (OR RETURN SPACE TO) A TABLE. THE STRUCTURE OF COARSE TABLE AND FINE TABLE ENTRIES IS EXACTLY THE SAME; THUS "FILLORKILL" IS USED TO HANDLE THEM BOTH. ITS PARAMETERS ARE:

1) THE DESCRIPTOR FOR THE ARRAY TO BE MANIPULATED,
2) THE STARTING BIT INDEX,
3) THE SIZE OF AREA (IN NUMBER OF BITS), AND
4) A CODE TO SPECIFY WHETHER THIS AREA IS TO BE ALLOCATED OR RETURNED.

THE MECHANISM USED IS TO CALCULATE STARTING AND ENDING WORD INDICES, AND BIT INDICES WITHIN THESE WORDS, FROM THE BIT INDEX AND SIZE PARAMETERS. AN INITIAL MASK IS CREATED AND, WHILE THE WORD INDICES ARE UNEQUAL, SUBROUTINE "WHATEVERTURNSYOUON" IS CALLED. THIS SUBROUTINE PERFORMS THE APPROPRIATE MASKING OPERATION ON THE CURRENT WORD, COMPUTES THE NEW MAXIMUM RUN LENGTH AND STORES THE RESULT AWAY. IT THEN INCREMENTS THE WORD INDEX INTO THE ARRAY, AND CONSTRUCTS THE MASK FOR THE NEXT WORD. THIS NEXT MASK IS COMPOSED OF THE [41:7] FIELD OF THE CURRENT MASK DIALED INTO THE [9:7] FIELD OF THE NEW MASK, WHICH ALSO HAS THE BITS IN THE [16:32] FIELD ALL SET TO ONE. THIS PROVIDES FOR THE PROPER MARKING OF THE "OVERLAP" AREA IN THE SUCCEEDING WORD. WHEN THE CURRENT WORD INDEX IS EQUAL TO THE ENDING WORD INDEX - WHICH MAY, OF COURSE, OCCUR IMMEDIATELY - THE MASK IS MODIFIED SO THAT ONLY THOSE BITS AFFECTED IN THE FINAL WORD WILL BE MODIFIED. THE SUBROUTINE IS AGAIN CALLED, AND, IF THIS LAST ENTRY WILL MODIFY THE "OVERLAP" PORTION OF THE FOLLOWING WORD, THE MASK IS ADJUSTED ([16:32] FIELD SET TO ZERO), AND THE SUBROUTINE CALLED A FINAL TIME. "FILLORKILL" EXITS, HAVING RETURNED OR ALLOCATED THE REQUIRED AREA AND UPDATING THE RUN COUNTS OF EACH AFFECTED WORD.

"AUXILIARYSPACE" IS A SAVE INTEGER PROCEDURE WHICH SEARCHES THE FINE AND COARSE TABLES TO ALLOCATE AUXILIARY MEMORY. THE SEARCHES, AND THE SUBSEQUENT ALLOCATION PROCESS, ARE CARRIED OUT BY THE SUBROUTINES "FINESEARCH" AND "COARSESEARCH". THE ALGORITHMS USED BY THESE SUBROUTINES ARE STRONGLY RELATED, DIFFERING PRIMARILY IN THE DIRECTION IN WHICH THE SEARCH TAKES PLACE. "AUXILIARYSPACE" HAS A

SINGLE PARAMETER, "SIZE", WHICH IS THE LENGTH (IN WORDS) OF THE DATA OR CODE SEGMENT FOR WHICH AUXILIARY MEMORY IS TO BE ALLOCATED. "AUXILIARYSPACE" CONVERTS THIS TO THE NUMBER OF 16-WORD CHUNKS REQUIRED, AND CALLS THE SUBROUTINE "SEARCH", WHICH CALLS THE APPROPRIATE SEARCH ROUTINE: "COARSESEARCH" TO FIND EIGHT OR MORE CHUNKS (128 WORDS OR MORE), OR "FINESEARCH" FOR SEVEN OR LESS CHUNKS.

"FINESEARCH", AS PREVIOUSLY NOTED, SEARCHES THE GIVEN FINE TABLE FROM BACK TO FRONT. STARTING WITH WORD 63 OF THE TABLE, IT COMPARES THE PARAMETER "SIZE" (WHICH IS NOW IN TERMS OF 16-WORD CHUNKS) WITH THE [3:6] FIELD OF THE TABLE ENTRY. (NOTE THAT THE MAXIMUM VALUE OF "SIZE" IS SEVEN, OTHERWISE WE WOULD BE IN "COARSESEARCH"). WHEN A WORD IS FOUND WHOSE "LONG RUN" FIELD IS GREATER THAN OR EQUAL TO "SIZE", THE ROUTINE ESTABLISHES WHERE WITHIN THE WORD THE RIGHTMOST SUCH RUN EXISTS. TO DO THIS, A MASK IS CONSTRUCTED, CONSISTING OF ONE BIT IN THE [48-SIZE:SIZE] FIELD OF THE MASK WORD. THE SUBROUTINE ENTERS A LOOP WHICH "AND-S " THIS MASK AND THE FINE TABLE ENTRY AND COMPARES THE RESULT TO THE MASK. WHEN THE RESULT IS EQUAL TO THE MASK, THE APPROPRIATE BITS HAVE BEEN FOUND. UNTIL THE LOW-ORDER MATCH IS FOUND, THE MASK IS SLID ONE BIT TO THE LEFT BY ADDING IT TO ITSELF. NOTE THAT THIS TECHNIQUE IS USEFUL ONLY IF THE WORD BEING TESTED HAS SIGNIFICANT BITS ONLY IN THE MANTISSA PORTION, AS ALL COARSE TABLE AND FINE TABLE ENTRIES DO. IT IS ALSO IMPORTANT TO POINT OUT THAT, IF THE "LONG RUN" COUNT IS NOT ACCURATE, THE SYSTEM COULD LOOP INDEFINITELY--THE SEARCH AT THIS POINT IS GEARED TO A DETERMINATION OF WHERE THE DESIRED RUN IS, NOT TO DETERMINING WHETHER IT EXISTS.

ASSUMING THAT ALL IS WELL WITH THE WORLD (OR AT LEAST WITH THE FINE TABLE), THE DESIRED NUMBER OF BITS ARE ALLOCATED FROM THE FINE TABLE BY A CALL UPON "FILLORKILL". THE NECESSARY INDICES ARE DEVELOPED FROM THE WORD INDEX INTO THE FINE TABLE AND THE COUNT OF THE NUMBER OF SHIFTS OF THE MASK. A CHECK IS MADE TO DETERMINE WHETHER THE FINE BITS ALLOCATED ARE ENCOMPASSED BY ONE OR BY TWO COARSE BITS, AND THE COARSE TABLE BIT(S) MODIFIED BY ANOTHER CALL ON "FILLORKILL". IT IS IMPORTANT TO NOTE THAT THIS ALLOCATION FROM THE COARSE TABLE MAY BE REDUNDANT -- THE RELEVANT BIT OR BITS MAY HAVE ALREADY BEEN ALLLOCATED; IT IS FASTER TO PERFORM THIS REDUNDANT ALLOCATION THAN TO SPECIFICALLY GUARD AGAINST IT. THE SUBROUTINE STORES ITS RESULT IN "CFRONT" (A PSEUDONYM FOR "AUXILIARYSPACE" ITSELF) AND EXITS.

"COARSESEARCH" USES AN ALGORITHM STRONGLY RELATED TO THAT USED IN "FINESEARCH". THE MAJOR DIFFERENCE IS THAT COARSESEARCH SEARCHES FROM FRONT TO BACK AND LEFT TO RIGHT AND THAT, WHILE THE FINE SEARCH MAY ALLOCATE BLINDLY FROM THE COARSE TABLE, "COARSESEARCH" USES THE FINE TABLE AS A TOOL TO ALLOCATE AS FEW COARSE BITS AS POSSIBLE. "COARSESEARCH" USES THE LOCAL VARIABLE "CBITS" (COARSE BITS) IN THE SAME MANNER AS "FINESEARCH" USES "SIZE". STARTING FROM THE ZERO-TH WORD OF THE COARSE TABLE, IT COMPARES CBITS TO THE "LONG RUN" ([3:6]) FIELD OF EACH ENTRY. WHEN ONE IS FOUND WHOSE LONG RUN IS GREATER THAN OR EQUAL TO THIS SIZE THE SUBROUTINE CONSTRUCTS A MASK,

CONSISTING OF ONES IN THE [9:SIZE] FIELD, WITH WHICH IT SUCCESSIVELY TESTS FIELDS OF THE COARSE TABLE ENTRY IN THE SAME MANNER AS "FINESEARCH". IN THE CASE OF NO MATCH, HOWEVER, THE MASK IS SHIFTED RIGHT (RATHER THAN LEFT) THROUGH USE OF A FIELD ISOLATE.

WHEN THE DESIRED NUMBER OF COARSE BITS ARE FOUND, SEVEN OF THE EIGHT FINE BITS CORRESPONDING TO THE COARSE BIT WHICH IMMEDIATELY PRECEDES THE SELECTED RUN ARE EXAMINED. BECAUSE OF THE OVERLAP ACROSS WORDS AND THE FACT THAT WE HAVE SEARCHED FROM LEFT-TO-RIGHT WITHIN THE CHOSEN WORD, WE KNOW THAT THE PRECEDING BIT IN THE COARSE TABLE IS OFF, AND CONSEQUENTLY, THAT AT LEAST ONE OF THE EIGHT FINE BITS CORRESPONDING TO THIS COARSE BIT MUST BE OFF. WE USE THIS FACT TO TRY TO SLIDE THE ALLOCATION OF THIS (RELATIVELY LARGE) AREA TO THE FRONT OF THE AUXILIARY MEMORY AS MUCH AS POSSIBLE. IT IS ALSO POSSIBLE TO "SAVE" A COARSE BIT BY THIS SLIDING OF THE AREA, IF THERE ARE SUFFICIENT CONTIGUOUS AVAILABLE FINE BITS. ONCE THIS HAS BEEN DETERMINED, THE COARSE AND FINE BITS INVOLVED ARE ALLOCATED THROUGH SUCCESSIVE CALLS ON "FILLORKILL", THE PSEUDO-ADDRESS OF THE ALLOCATED AREA IS LEFT IN "CFRONT", AND THE SUBROUTINE EXITS.

YOU MAY NOTICE THAT NO MENTION HAS BEEN MADE OF THE ACTION TAKEN WHEN NO WORD IN THE APPROPRIATE BIT TABLE FITS THE CRITERIA REQUIRED-- THAT IS, WHEN SUFFICIENT SPACE DOES NOT EXIST ON THE PARTICULAR AUXILIARY MEMORY UNIT. IN SUCH A CASE, THE CALLED SUBROUTINE MERELY EXITS WITHOUT CHANGING THE VALUE OF "CFRONT".

THE DRIVER SUBROUTINE, "SEARCH", USES A LOCAL VARIABLE, "INDEX", TO SET UP LOCAL ARRAYS TO POINT TO THE CORRECT FINE AND COARSE TABLES. IT IS ALSO USED TO DETERMINE WHETHER THE DESIGNATED AUXILIARY MEMORY UNIT IS AVAILABLE FOR USE. IF NOT, IT EXITS WITHOUT FURTHER ACTION. IF THE UNIT DESIGNATED BY "INDEX" IS AVAILABLE FOR USE, "SEARCH" CALLS "FINESEARCH" OR "COARSESEARCH" AND EXITS.

THE BODY OF "AUXILIARYSPACE" MERELY DETERMINES THE NUMBER OF BITS TO BE FOUND, WHICH OF THE TWO COARSE OR FINE TABLES IS TO BE SEARCHED FIRST, AND CALLS "SEARCH". IF, WHEN IT RETURNS, "CFRONT" IS NEGATIVE, NO SPACE HAS BEEN ALLOCATED. IT THEN CALLS "SEARCH" AGAIN, TO CHECK THE OTHER PAIR OF TABLES. IF, AT THIS POINT, "CFRONT" IS STILL NEGATIVE, THE DESIRED SPACE COULD NOT BE ALLOCATED AND A VALUE OF ZERO IS RETURNED TO THE OUTSIDE WORLD. IF SPACE WAS ALLOCATED, THE [33:3] FIELD OF THE PSEUDO-ADDRESS IS SET TO 7 (TO INDICATE THAT THIS IS AN AUXILIARY MEMORY ADDRESS) AND THE [34:1] BIT IS SET TO INDICATE WHICH AUXILIARY MEMORY UNIT IS INVOLVED. THIS VALUE, A PSEUDO-ADDRESS WHICH CAN BE INTERPRETED BY "ACTUALOVERLAYADDRESS", IS THEN RETURNED AS THE RESULT OF THE FUNCTION.

"FORGETAUXILIARYSPACE" IS A SAVE PROCEDURE WHICH RETURNS AUXILIARY MEMORY SPACE TO THE BIT TABLES. ITS TWO PARAMETERS ARE "SIZE", THE SIZE (IN WORDS) OF THE AREA TO BE RETURNED, AND "LOC", THE PSEUDO-ADDRESS USED TO ACCESS THE AREA. THE PROCEDURE PERFORMS THE SAME

TRANSFORMATION ALGORITHM OF SIZE AS DOES "AUXILIARYSPACE", THAT IS, ((SIZE + 1) + 15) DIV 16, WHICH SIMPLIFIES TO SIZE.[38:6]+1.

THE FIRST ACTION OF THE PROCEDURE IS TO CALL "FILLORKILL" TO RETURN THE NECESSARY BITS TO THE FINE TABLE. IN THE COURSE OF THIS PROCEDURE CALL, THE LOCAL VARIABLES "FINETABLE" AND "TABLE" ARE INITIALIZED AND THE PARAMETERS, "LOC" AND "SIZE", ARE MODIFIED. THE PROCEDURE THEN ESTABLISHES THE FIRST AND LAST COARSE BIT INDICES CORRESPONDING TO THE RETURNED FINE BITS, AND CALLS SUBROUTINE "NOTALLTHERE" TO DETERMINE WHETHER THESE "BOUNDARY" COARSE BITS SHOULD BE TURNED ON. IF, AT THE END OF THESE CHECKS, AT LEAST ONE COARSE BIT IS TO BE RETURNED TO THE TABLE, "FILLORKILL" IS AGAIN CALLED TO RETURN IT.

"AUXILIARYMEMORYCASUALTYRECOVERY", DISCUSSED PREVIOUSLY IN THE NARRATIVE SECTION, IS A PROCEDURE WHICH IS CALLED TO "GARBAGE COLLECT" THE AUXILIARY MEMORY SYSTEM IF A JOB IS ABNORMALLY TERMINATED. SINCE THE RELEVANT DATA DESCRIPTORS WHICH CONTAIN REFERENCES TO AUXILIARY MEMORY MAY NOT EXIST (IN THE CASE OF DOPE VECTORS FORGOTTEN BY "TERMINALMESSAGE") OR MAY BE DIFFICULT TO FIND (IN THE STACK OR PRT), THE PROCEDURE USES THE COMPLEMENT OF THE AVAILABLE AUXILIARY MEMORY BIT TABLES TO DETERMINE IN-USE AREAS. THE LINK WORD FOR A GIVEN IN-USE AREA IS READ INTO MEMORY, AND, IF IT REPRESENTS DATA STORAGE FOR THE TERMINATED JOB, IS RETURNED TO THE AVAILABLE TABLE. IN ANY EVENT, THE SIZE OF THE LINK WORD IS USED TO INDEX PAST THAT AREA AND CHECK THE LINK WORD OF THE NEXT IN-USE AREA.

"AUXDATA" AND "AUXCODE" ARE ARRAYS, INDEXED BY MIX INDEX, WHICH ACCUMULATE THE AMOUNT OF AUXILIARY MEMORY USED FOR DATA OVERLAY AND CODE OVERLAY, RESPECTIVELY. FOR RE-ENTRANT JOBS WHICH USE THE "CODE OVERLAY TO AUXILIARY MEMORY" CONSTRUCTS (OR FOR ANY JOB IF THE CODEOLAY OPTION IS SET), THE "MOTHER" MIX INDEX IS USED. THE VALUE IN AUXCODE IS PASSED ALONG THE LINE OF SUCCESSIVE "MOTHER" JOBS, ONLY BEING RETURNED WHEN THE LAST MEMBER OF A RE-ENTRANT FAMILY TERMINATES. DATA, OF COURSE, IS KEPT SEPARATELY, SINCE IT IS NOT RE-ENTRANT. AUXCODE IS COUNTED UP BY THE OLAY PROCEDURE WHEN SPACE ON AUXILIARY MEMORY IS ASSIGNED, AND COUNTED DOWN BY EITHER "COM5" (AT THE END OF THE LAST JOB USING THIS CODE) OR BY "COMMUNICATE" WHEN THE SPACE IS EXPLICITLY RETURNED BY THE PROGRAMS. AUXDATA IS COUNTED UP BY THE "DISKSPACE" PROCEDURE, WHEN IT SUCCESSFULLY ALLOCATES AUXILIARY MEMORY FOR DATA STORAGE. IT IS COUNTED DOWN BY "COMMUNICATE", FOR EXPLICIT PROGRAMMATIC RELEASE OF THE SPACE, BY "DISKRTN" FOR IMPLICIT RELEASES (BLOCK EXIT, FOR EXAMPLE), AND BY "COM5" AND THE CASUALTY RECOVERY CODE. THESE ARRAYS ARE ALSO USED BY THE AUXPRINT PROCEDURE, WHICH, IN RESPONSE TO THE "AU" OR "<MIX> AU" KEYBOARD INPUT, PRINTS THE AMOUNT OF AUXILIARY MEMORY IN USE FOR THE GIVEN JOB (OR FOR ALL JOBS).

THE "DISKIO" PROCEDURE HAS BEEN MODIFIED TO HANDLE AUXILIARY MEMORY ADDRESSES IN A MORE STRAIGHT-FORWARD FASHION. IF A "DISK" ADDRESS

PASSED TO "DISKIO" IS NEGATIVE, IT IS INTERPRETED AS AN AUXILIARY MEMORY ADDRESS AND THE APPROPRIATE DESCRIPTOR IS BUILT. IN ADDTION, TO PROVIDE FOR THE LINK WORD, THE NUMBER OF WORDS TO BE READ OR WRITTEN IS INCREASED BY ONE, AND THE CORE ADDRESS IS NOT CHANGED, SO THE THE FIRST WORD READ INTO OR WRITTEN FROM WILL BE A SECONDARY MEMORY LINK (IN GENERAL).

THE OVERLAY DISK ALLOCATION AND RETURN ROUTINES HAVE BEEN SLIGHTLY MODIFIED TO INCORPORATE THE AUXILIARY MEMORY ALLOCATION SUBSYSTEM. "GETMOREOLAYDISK" HAS BEEN MODIFIED TO ACQUIRE A MAXIMUM OF 55 CHUNKS OF OVERLAY DISK PER JOB (THE PREVIOUS LIMIT WAS 63 CHUNKS). "DISKSPACE" HAS BEEN MODIFIED BY REMOVAL OF ALL CODE WHICH HAD BEEN CONCERNED WITH DRUMS, AND BY THE INCLUSION OF A CALL ON "AUXILIARYSPACE". "DISKRTN" HAS BEEN CHANGED TO CORRECTLY RETURN AUXILIARY MEMORY. THIS CHANGE REQUIRED AN ADDITIONAL PARAMETER, "SIZE", WHICH "DISKRTN" PASSES ALONG TO "FORGETAUXILIARYSPACE". "DISKSPACE" AND "DISKRTN", AS PREVIOUSLY MENTIONED, HAVE BEEN MODIFIED TO KEEP TRACK OF THE AMOUNT OF AUXILIARY MEMORY USED (IN THE ARRAY AUXDATA).

THE OLAY ALGORITHM HAS BEEN MODIFIED TO PERMIT THE USE OF THE AUXILIARY MEMORY FOR OVERLAY OF CODE SEGMENTS, AND TO ALLOW FOR READ-ONLY ARRAYS. READ-ONLY ARRAYS ARE DISTINGUISHED BY THE FACT THAT THE "UNIT" FIELD ([3:5]) OF THE MOTHER DESCRIPTOR IS NON-ZERO; IT IS EITHER 24 OR 28. NEITHER OF THESE VALUES IS USED FOR AN ACTUAL INPUT/OUTPUT UNIT, SO THEY ARE READILY RECOGNIZED AS BEING DATA DESCRIPTORS. WHEN OVERLAYING A READ-ONLY ARRAY, THE DATA IS WRITTEN (TO DISK OR TO AUXILIARY MEMORY) ONLY IF THE [3:5] FIELD IS 24. IF IT IS 28, THE DATA HAS ALREADY BEEN WRITTEN TO DISK. AFTER THE DATA HAS BEEN WRITTEN OUT, THE [3:5] FIELD IS SET TO 28 TO INHIBIT FURTHER OVERLAY OUTPUT ACTION.

THE MAJOR CHANGE TO THE OVERLAY ALGORITHM, HOWEVER, INVOLVES THE FACILITY TO OVERLAY CODE SEGMENTS TO AUXILIARY MEMORY. ONE OF THE CHANGES MADE ELIMINATES THE MUTIPLE-USAGE OF CERTAIN STACK CELLS; WHEN THE OVERLAY ROUTINE WAS WRITTEN, SOME STACK LOCATIONS WERE ASSIGNED DIFFERENT MEANINGS FOR DATA OVERLAY THAN FOR CODE OVERLAY, BECAUSE NO OUTPUT WAS INVOLVED IN CODE OVERLAY. THIS, OF COURSE, HAS BEEN CHANGED. WHEN A CODE SEGMENT IS TO BE OVERLAID, THE SEGMENT DICTIONARY ENTRY IS EXAMINED. IF THE [4:2] FIELD IS EQUAL TO 2, THE SEGMENT IS TO BE OVERLAID TO AUXILIARY MEMORY, BUT HAS NOT YET BEEN WRITTEN OUT. A CALL IS MADE ON "AUXILIARYSPACE" TO ACQUIRE THE REQUISITE OVERLAY AREA. IF SUFFICIENT SPACE IS NOT AVAILABLE ON AUXILIARY MEMORY, THE SEGMENT DICTIONARY IS LEFT MARKED AS BEFORE, AND THE OVERLAY PROCEEDS AS WITH A NORMAL CODE SEGMENT. IF THE AUXILIARY SPACE WAS ASSIGNED, HOWEVER, THE SECONDARY MEMORY LINK IS PREPARED IN THE FORM PREVIOUSLY DESCRIBED: SIGN BIT ON TO INDICATE A CODE SEGMENT, SIZE OF THE CODE SEGMENT IN THE [18:15] FIELD, DISK ADDRESS OF THE CODE SEGMENT (IN THE PROGRAM FILE) IN THE [3:15] FIELD, AND PRIMARY (MOTHER) MIX INDEX IN THE [33:15] FIELD. THE SEGMENT DICTIONARY ENTRY IS MARKED BY TURNING ON THE [5:1] BIT, AND

THE AUXILIARY MEMORY PSEUDO-ADDRESS IS PUT INTO THE [33:15] FIELD. AT THIS POINT THE AUXCODE ARRAY IS COUNTED UP, AND THE PSEUDO-ADDRESS CONVERTED TO AN AUXILIARY MEMORY ADDRESS WHICH CAN BE INTERPRETED BY "DISKIO". IF I/O QUEUE SPACE IS AVAILABLE, "DISKIO" IS CALLED TO ENTER THIS REQUEST INTO THE QUEUE. OVERLAY THEN PROCEEDS AS NORMAL. WHEN ALL DESCRIPTORS AND CONTROL WORDS RELATING TO THIS SEGMENT HAVE BEEN MARKED IN ALL PRT-S AND STACKS OF JOBS USING THE SEGMENT, A CHECK IS MADE TO DETERMINE IF THE SEGMENT IS TO BE OUTPUT ON THIS OVERLAY. IF IT IS, THEN IF THE OUTPUT WAS NOT PREVIOUSLY INITIATED IT IS INITIATED AT THIS TIME; IN ANY EVENT, OLAY WAITS FOR THE COMPLETION OF THE OUTPUT.

AS PREVIOUSLY MENTIONED, "COM5" HAS BEEN MODIFIED TO RETURN CERTAIN DATA STORAGE ON AUXILIARY MEMORY AT END OF JOB. THIS DATA INCLUDES THE AIT (ARRAY INFORMATION TABLE) ARRAY, THE OAT (OWN ARRAY TABLE) ARRAY, AND FORMATS, CASE STATEMENT BRANCH TABLES, AND OTHER DATA SEGMENTS WHICH HAVE SEGMENT DICTIONARY ENTRIES. ENTRIES WHICH HAVE SEGMENT DICTIONARY ENTRIES ARE RETURNED IN THE SAME LOOP WHICH RETURNS CODE SEGMENTS. CODE SPACE ON AUXILIARY MEMORY IS RETURNED BY "COM5" IN A LOOP WHICH CHECKS EACH SEGMENT DICTIONARY ENTRY FOR THE [5:1] BIT (WHICH INDICATES AUXILIARY MEMORY) AND RETURNS THE STORAGE TO THE AVAILABLE TABLE. "AUXCODE" IS USED AS A COUNTER -- WHEN IT REACHES ZERO, ALL THE CODE STORAGE HAS BEEN RETURNED AND THE LOOP IS ENDED.

COMMUNICATE NUMBER 38 HAS BEEN ADDED AS THE MECHANISM THROUGH WHICH PROGRAMMATIC RETURN OF AUXILIARY MEMORY TAKES PLACE. WHETHER DATA STORAGE OR CODE STORAGE MAY BE RETURNED. IF CODE STORAGE IS TO BE RETURNED, A COPY OF THE PROGRAM DESCRIPTOR FOR THE SEGMENT IS PASSED. THE "COMMUNICATE" PROCEDURE DETERMINES THE SEGMENT NUMBER CORRESPONDING TO THIS DESCRIPTOR (IN THE SAME MANNER AS DOES THE "MAKEPRESENT" ROUTINE) AND CHECKS THE [4:2] FIELD OF THE SEGMENT DICTIONARY ENTRY. IF THIS IS ZERO, IT IS A NORMAL CODE SEGMENT AND NO OVERT ACTION TAKES PLACE. IF IT IS NON-ZERO, THEN IT IS EITHER TO BE OVERLAID TO AUXILIARY MEMORY OR IT HAS ALREADY BEEN OVERLAID TO AUXILIARY MEMORY. IN EITHER CASE, THE ALGORITHM WAITS FOR THE [3:1] BIT OF THE SEGMENT DICTIONARY TO BE ZERO--THIS IS THE RE-ENTRANT CODE INTERLOCK BIT. IT THEN TURNS ON THE INTERLOCK -- TO SUSPEND PRESENCE BIT ACTION ON THE SEGMENT-- AND WAITS FOR THE "STOREBY" TOGGLE TO INDICATE THAT THE MEMORY LINK SYSTEM IS INTACT. IF THE CODE SEGMENT IS PRESENT, THE SECONDARY MEMORY LINK WHICH PRECEDES THE DATA IS SAVED, AND OLAY CALLED TO OVERLAY THE AREA (IF POSSIBLE). CARE HAS BEEN TAKEN TO TURN OFF THE [4:1] BIT OF THE SEGMENT DICTIONARY ENTRY SO THAT NO OUTPUT IS DONE. IF THE SEGMENT IS NOT PRESENT, BUT IS ON AUXILIARY MEMORY, A DISK READ IS INITIATED TO READ IN THE LINK WORD OF THE AREA. THE LINK WORD IS THEN USED TO RECALL THE ORIGINAL DISK ADDRESS OF THE SEGMENT, AND THIS IS PUT INTO THE SEGMENT DICTIONARY ENTRY. THE AUXILIARY MEMORY IS RETURNED, AND AUXCODE COUNTED DOWN. FINALLY, THE [3:1] BIT OF THE SEGMENT DICTIONARY ENTRY IS SET TO 0 (TO INDICATE THAT IT IS UNLOCKED), AND THE [4:2] FIELD IS SET BACK TO 2.

IN THE CASE OF DATA SEGMENTS, PROBLEMS OF RE-ENTRANCY DO NOT EXIST, SINCE (AT THE MOMENT) PROGRAMS DO NOT SHARE DATA STRUCTURES. THE APPROPRIATE "MOTHER" DESCRIPTOR FOR THE ARRAY ROW IS EXAMINED AND, IF IT IS PRESENT, IS MARKED TO SHOW THAT IT IS "READ-ONLY" AND A VALID COPY EXISTS ON DISK. THE OLAY ROUTINE IS CALLED TO MARK ALL COPIES OF THE DESCRIPTOR AS BEING ABSENT; NO OUTPUT WILL OCCUR BECAUSE THE MOTHER DESCRIPTOR INDICATES THAT IS IS NOT NEEDED. IF ANY AUXILIARY MEMORY OR OVERLAY DISK HAS BEEN ASSIGNED TO THE ARRAY ROW, IT IS RETURNED. THE MOTHER DESCRIPTOR IS MARKED TO SHOW THE ROW AS NOT PRESENT, NEVER ACCESSED. ANY FURTHER ACCESS WILL CAUSE A NEW ROW TO BE ALLOCATED AND INITIALIZED.

CONCLUSION

THE GUIDING PHILOSOPHY BEHIND THE DEVELOPMENT OF THESE MODIFICATIONS TO THE BURROUGHS MCP FOR THE B5500/B5700 HAS BEEN TO TREAT AUXILIARY MEMORY AS A SIGNIFICANT OVERLAY DEVICE IN ITS OWN RIGHT. ALTHOUGH THE MAXIMUM CONFIGURATION IS SMALL (65,536 WORDS) COMPARED TO THAT OF THE HEAD-PER-TRACK DISK, A GREAT DEAL OF CARE HAS BEEN TAKEN TO MAXIMIZE THE USEFULNESS OF AUXILIARY MEMORY. THROUGH USING TWO ADDITIONAL PARALLEL ACCESS PATHS, AND BECAUSE OF THE ESSENTIALLY NONEXISTANT LATENCY OF AUXILIARY MEMORY, IT IS BELIEVED THAT SUBSTANTIAL IMPROVEMENTS IN OPERATION EFFICIENCY WILL BE ACHIEVED.

AVAIL

CONTAINS THE ADDRESS OF THE STOPPER FOR AVAILABLE STORAGE LINKS. ITS VALUE IS (HIGHEST AVAILABLE ADDRESS)-1.

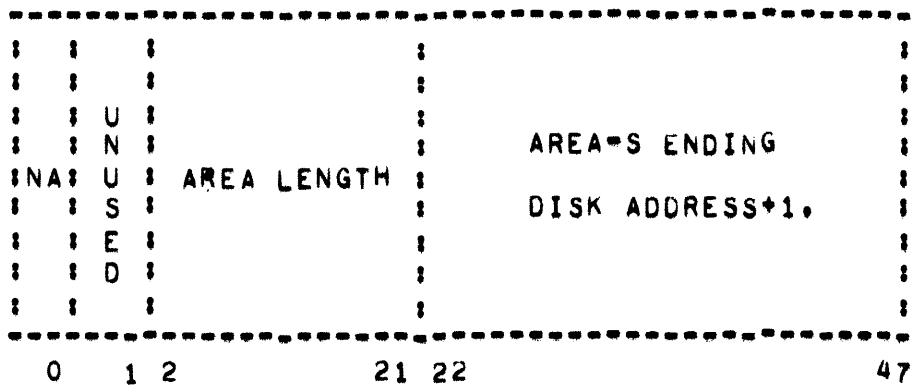
AVAILABLE DISK TABLE

THE AVAILABLE TABLE MAINTAINS AN ACCOUNT OF THE AVAILABLE PORTIONS OF DISK, I.E., THOSE SEGMENTS OF DISK NOT IN USE AND NOT PART OF A PERMANENT FILE. THE AVAILABLE TABLE IS COMPRISED OF 2 SUBTABLES:

SUBTABLE 1:

THIS IS THE DISK RESIDENT PORTION OF THE AVAILABLE TABLE. AN ENTRY IN THIS SUBTABLE DESCRIBES ONE CONTIGUOUS AREA OF AVAILABLE DISK. THE .[2:20] FIELD OF THE ENTRY CONTAINS THE NUMBER OF SEGMENTS IN THE AVAILABLE AREA, AND THE .[22:26] FIELD CONTAINS THE ENDING DISK ADDRESS OF THE AVAILABLE AREA, +1. THE ENTRIES ARE ARRANGED IN INCREASING ORDER OF THE .[22:26] FIELD. THEY ARE GROUPED ACCORDING TO EU: THOSE ENTRIES CORRESPONDING TO EU#0 ARE AT THE BOTTOM OF THE SUBTABLE, FOLLOWED BY SOME EMPTY WORDS (SEPARATION WORDS), THEN THOSE ENTRIES CORRESPONDING TO EU#1, FOLLOWED BY SOME SEPARATION WORDS, ETC. SUBTABLE 1 STARTS AT DISK ADDRESS USERDISKBOTTOM, AND CURRENTLY MAY BE AS LARGE AS 134 SEGMENTS. THE SPECIFIC ENTRY BREAKDOWN IS:

ENTRY I:



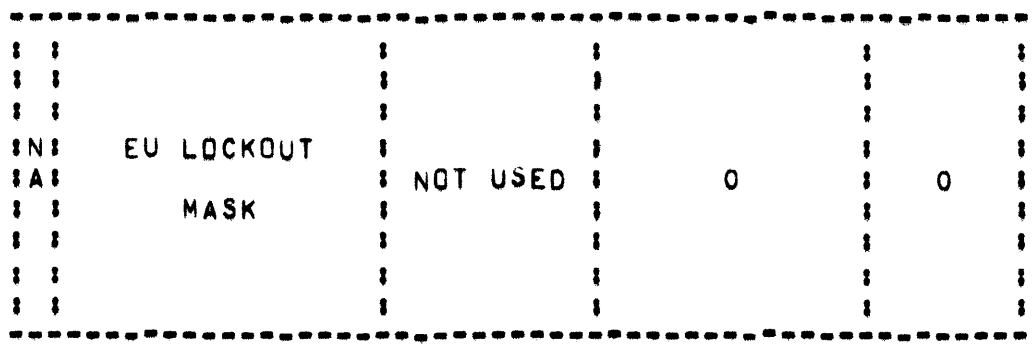
FIELD	CONTENTS	DESCRIPTION
.[1:1]		NOT USED
.[2:20]	AREA LENGTH	# SEGMENTS IN AVAILABLE AREA
.[22:26]	DISK ADDRESS	AREA-S ENDING DISK ADDRESS+1.

- AVAILABLE DISK TABLE -

SUBTABLE 2:

THIS IS THE CORE RESIDENT PORTION OF THE AVAILABLE TABLE. IT PROVIDES FOR RAPID ACCESS TO SUBTABLE 1. A TYPICAL ENTRY CONTAINS AN OFFSET TO THE START OF THE ENTRY-S GROUP, THE GROUP SIZE, THE MAXIMUM AVAILABLE AREA FOR THAT GROUP (EU), AND THE SPEED OF THE EU. ASSOCIATED WITH THIS ENTRY IS A "HALF ENTRY" CONTAINING LOCKOUT INFORMATION. THE SPECIFIC SUBTABLE CONTENTS ARE:

WORD 0:

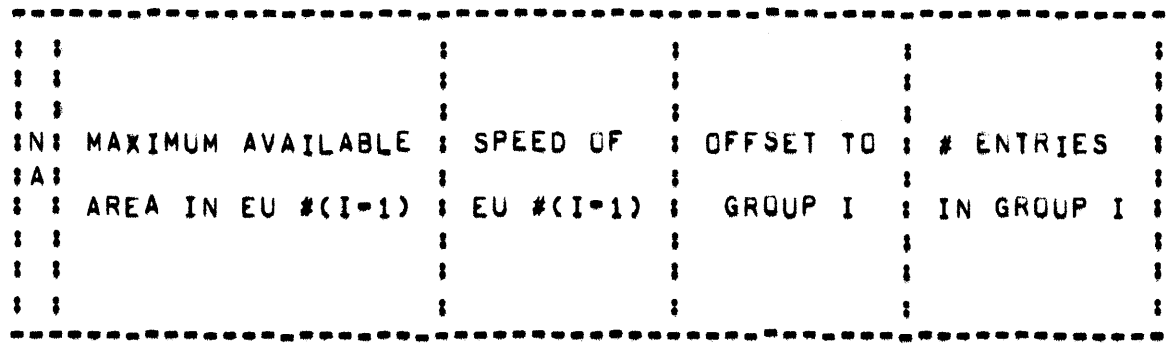


FIELD	CONTENTS	DESCRIPTION
.[1:20]	BIT MASK	EU LOCKOUT MASK; IF ANY PART OF AN EU IS LOCKED OUT, AT H-L TIME BIT #20-N IS SET, WHERE N IS THE EU # (0,1,2,...,19). A MISSING MOD IS NOT TREATED AS A LOCKED OUT MOD.
.[21:5]		NOT USED
.[26:12]	0	NECESSARY FOR PROGRAM LOGIC
.[38:10]	0	NECESSARY FOR PROGRAM LOGIC

- AVAILABLE DISK TABLE -

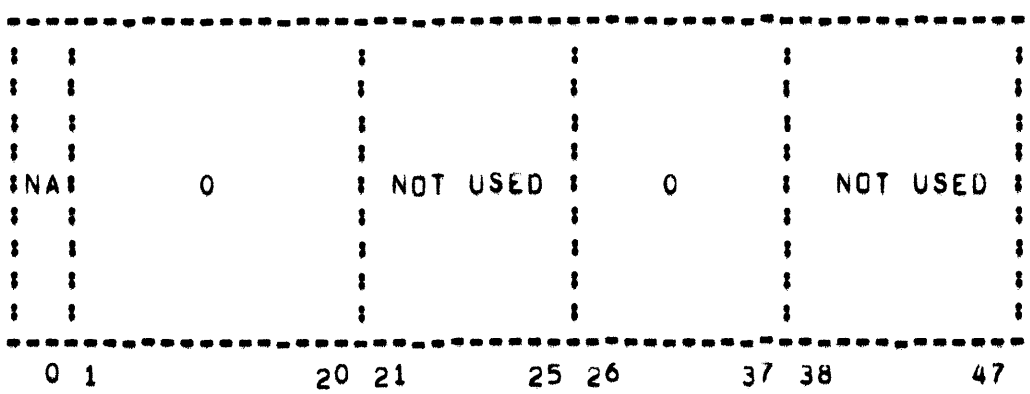
WORD I: (I=1,2,...,N WHERE N = TOTAL NUMBER OF EU'S ATTACHED TO SYSTEM, INCLUDING THOSE LOCKED OUT AND/OR ON DKB).

(NOTE THAT "GROUP I" PERTAINS TO THE COLLECTION OF THOSE SUBTABLE 1 ENTRIES DESCRIBING THE AVAILABLE AREAS IN EU #(I-1))



FIELD	CONTENTS	DESCRIPTION
.[1:20]	MAX NUMBER SEGMENTS	MAXIMUM NUMBER OF CONTIGUOUS SEGMENTS AVAILABLE IN EU #(I-1)
.[21:5]	SPEED	SPEED OF EU #(I-1). CURRENTLY, SPEED=1 CORRESPONDS TO 20 MIL DISK, SPEED=2 TO 40 MIL DISK.
.[26:12]	OFFSET (# WORDS)	OFFSET IS WORDS FROM THE START OF SUBTABLE 1 TO THE START OF GROUP I.
.[38:10]	# WORDS IN GROUP I	NUMBER OF ENTRIES IN GROUP I (I.E., # DISTINCT AVAILABLE AREAS IN EU #(I-1)).

WORD N+1: (N = N DESCRIBED ABOVE)



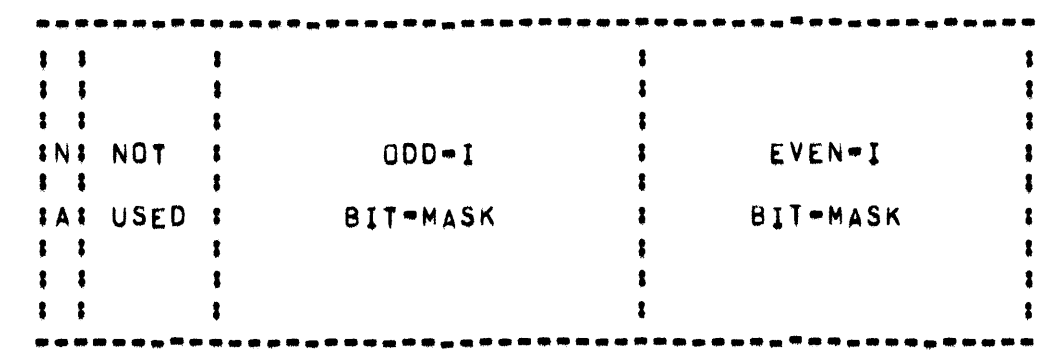
THIS WORD IS NECESSARY FOR PROGRAM LOGIC.

FIELDS ,[1:20] AND ,[26:12] MUST BE 0. FIELDS ,[21:5] AND ,[38:10] ARE NOT USED.

- AVAILABLE DISK TABLE -

WORD J: (J = N+2,N+3,...,N+2+[N/2])

THESE WORDS PROVIDE DETAILED INFORMATION DESCRIBING THE LOCKOUT CONDITION OF EACH SUBMOD (THERE ARE 4 SUBMODS PER MOD OF DISK, AND UP TO 5 MODS PER EU). IN THE FOLLOWING, I IS THE EU # (I=0,1,2,...,UP TO 19).



FIELD	CONTENTS	DESCRIPTION
.[1:7]		NOT USED
.[8:20]	BIT=MASK	IF I=1,3,5,..., THEN THIS FIELD OF WORD J=N+2+[I/2] DESCRIBES THE 20 SUBMODS FOR EU #I; SEE BELOW FOR DETAILS.
.[28:20]	BIT=MASK	IF I=0,2,4,..., THEN THIS FIELD OF WORD J=N+2+[I/2] DESCRIBES THE 20 SUBMODS FOR EU #I; SEE BELOW FOR DETAILS.

THESE MASKS ARE CONSTRUCTED AT H/L TIME. THEY ARE USED BY FORGETUSERDISK TO DETERMINE IF AN AREA ON DISK IS WHOLLY, OR IN PART, LOCKED OUT. EACH PHYSICAL EU HAS 20 LOCKOUT SWITCHES: 4 SWITCHES FOR EACH OF THE POSSIBLE 5 MODS. THERE IS ONE BIT-MASK PER EU, AND 20 BITS PER MASK, ONE BIT FOR EACH SWITCH. THE LOWER ORDER BIT CORRESPONDS TO THE LEFT MOST SWITCH (THE SWITCHES NUMBER FROM THE LEFT ON THE PHYSICAL EU SWITCH PANEL: 1,2,...,20). IF A SWITCH IS ON (LOCKED OUT), THEN THE BIT IS SET. A NON-EXISTENT MOD IS ASSUMED TO EXIST, BUT TO BE FULL (HAVE NO AVAILABLE DISK). FOR EXAMPLE, ASSUME EU #7 HAS 3 MODS, AND THAT SWITCH #4 IS ON. SUPPOSE THERE ARE 9 EU"S. THEN N=9, AND J=9+2+[7/2]=11+3=14. SINCE I=7, I IS ODD AND EU #7 HAS ITS 20 BIT MASK LOCATED AT WORD 14 (NUMBERING FROM 0,1,2,...,14), FIELD .[8:20]. WITHIN THIS MASK, THE 4TH LOWER ORDER BIT IS SET, I.E., BIT 24 IN WORD 14. THIS INDICATES TO

- AVAILABLE DISK TABLE -

FORGETUSERDISK THAT, FOR 20 MIL DISK, ADDRESSES 7030000-7039999 ARE LOCKED OUT, AND FOR 40 MIL DISK, THAT ADDRESSES 7030000-7039999 AND 7130000-7139999 ARE LOCKED OUT. NOTE THAT WORD 0, SUBTABLE 1, BIT 13 IS SET.

- AVAILABLE DISK TABLE -

AVAILABLE DISK TABLE (SHAREDISK)

WITH 2 EXCEPTIONS, THE SHARED DISK AVAILABLE TABLE IS THE SAME AS THE STANDARD AVAILABLE TABLE. THESE EXCEPTIONS ARE:

1. SUBTABLE 2 IS DISK RESIDENT, AND RESIDES NEXT TO, AND JUST BELOW, SUBTABLE 1.

2. SUBTABLE 2, WORD 0, FIELD ,[38:10] HAS THE VALUE N+2+[N/2], WHERE N IS THE TOTAL NUMBER OF EU'S. THIS VALUE IS NECESSARY FOR PROGRAM LOGIC.

VARIABLES RELATED TO THE AVAILABLE TABLE

1. NEUP, REAL SIMPLE VARIABLE. NEUP.[CF] CONTAINS THE NUMBER OF EU-S ON DKA, AND NEUP.[FF] CONTAINS THE TOTAL NUMBER OF EU-S ON THE SYSTEM.

2. EUIO, REAL ARRAY, SIZE = EUIOFFSET+NEUP.[FF]. WORD I, I= EUIOFFSET, EUIOFFSET+1,...,NEUP.[FF]-1, CONTAINS A TAPERING-FUNCTION MEASUREMENT OF THE ELAPSED I/O TIME FOR EU #(I-EUIOFFSET), AS COMPUTED FROM THE LAST H/L. EUIOFFSET IS DEFINED TO BE 4, THE MAXIMUM POSSIBLE NUMBER OF I/O CHANNELS PER SYSTEM. THE FIRST EUIOFFSET WORDS ARE USED TO MEASURE THE ELAPSED I/O TIME ON EACH EU.

3. PEUIO, REAL ARRAY, SIZE = NEUP.[FF]. WORD I, I=0,1,...,NEUP.[FF]-1, CONTAINS THE ELAPSED TIME FOR EU #I, AS MEASURED FROM THE LAST N-SECOND.

4. (SHAREDISK ONLY): AVS, INTEGER SIMPLE VARIABLE. AVS IS THE NUMBER OF WORDS TO READ THE NEXT TIME THE AVAILABLE TABLE IS READ IN FROM DISK. IT IS RE-COMPUTED FROM TIME-TO-TIME.

5. (NON-SHAREDISK ONLY): AVTABLE, REAL ARRAY, SIZE = NEUP.[FF]+2+ [NEUP.[FF]/2]. THIS IS SUBTABLE 1.

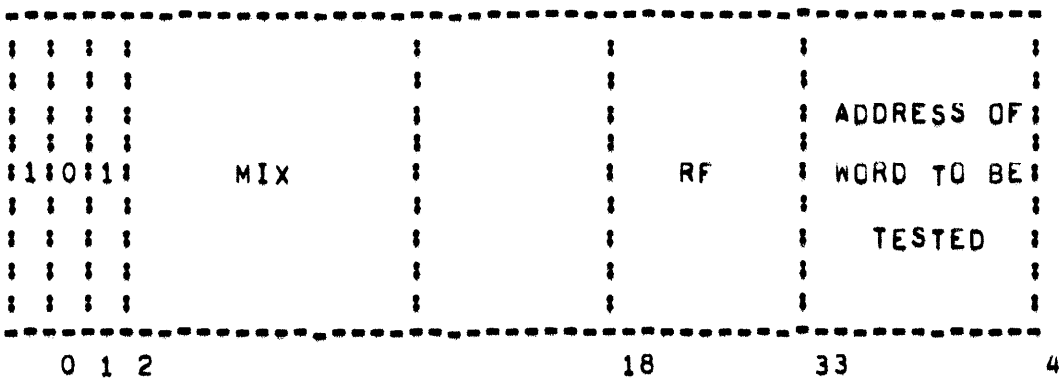
BACKWATER

CONTAINS THE TAIL OF THE LIST USED BY EGGTIMER FACILITY.

BED[*]

THE "BED" ARRAY, THE "SLEEP", "COMPLEXSLEEP", "SNOOZE" AND "COMPLEXSNOOZE" 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 THE USE OF THE "SNOOZE" ROUTINE. THE LAST ENTRY IN THE "BED" IS POINTED TO BY "JOBNUM". THE BED IS ORDERED BY THE PRIORITY OF THE PROCESS PROVIDED AS A PARAMETER BY THE "SNOOZE" AND "COMPLEXSNOOZE" ROUTINES. NORMALLY, THE VALUE IS PRIORITY[P1MIX], HOWEVER, THE VALUE MAY BE ZERO FOR THE HIGHEST PRIORITY TO 1023 FOR THE LOWEST PRIORITY. THE "BED" IS ALSO USED BY THE "NOTHINGTODO" ROUTINE TO RESTART JOBS WHICH HAVE BEEN TEMPORARILY SUSPENDED. ENTRIES MADE BY THE "SLEEP" OR "SNOOZE" ROUTINE:

WORD 1



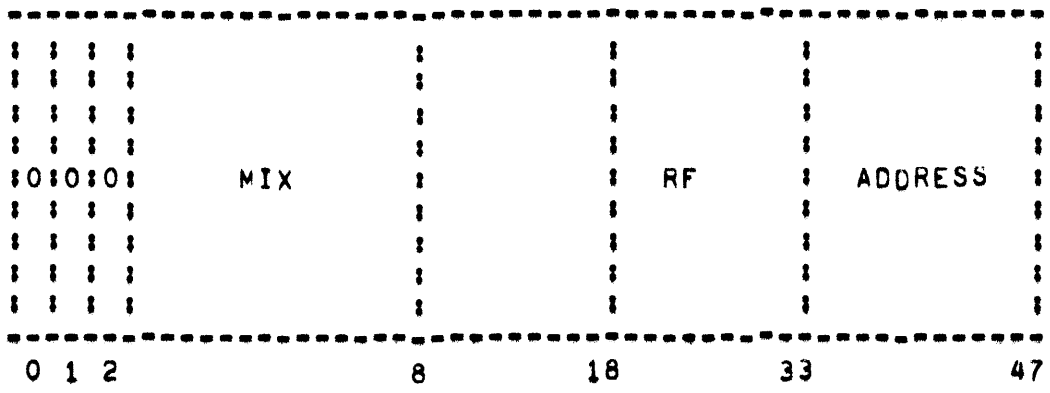
FIELD	CONTENTS	DESCRIPTION
[0:3]	5	DESCRIPTOR IDENTIFICATION BITS.
[3:5]	MIX	"MIX INDEX" OF SUSPENDED PROGRAM.
[8:10]	0	SIZE FIELD.
[18:15]	RF	F REGISTER SETTING FOR SUSPENDED PROGRAM.
[33:15]	ADDRESS	ADDRESS OF WORD TO BE TESTED TO DETERMINE IF THE NECESSARY CONDITION IS SATISIFIED

WORD 2

FIELD	CONTENTS	DESCRIPTION
[0:1]	0	FLAG BIT (CANNOT BE USED FOR MASK BIT).
[1:47]	MASK	CONTAINS 1-S IN BIT POSITIONS WHICH INDICATE WHEN THE NEEDED CONDITION IS PRESENT. ALL OTHER BITS ARE SET TO ZERO.

ENTRIES MADE VIA THE "COMPLEXSLEEP" OR "COMPLEXSNOOZE" ROUTINE:

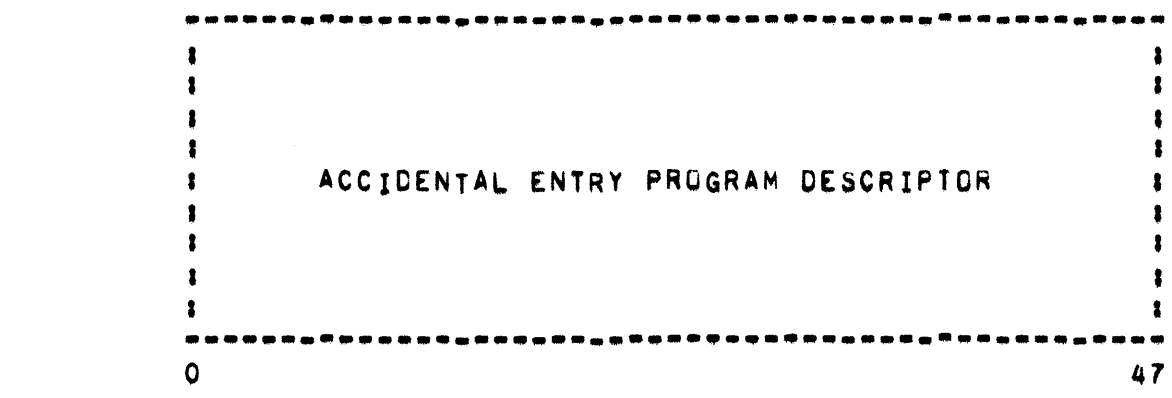
WORD 1



FIELD	CONTENTS	DESCRIPTION
[0:3]	0	OPERAND IDENTIFICATION BITS.
[3:5]	MIX	"MIX INDEX" OF SUSPENDED PROGRAM.

[8:10]	0	SIZE FIELD.
[18:15]	RF	F REGISTER SETTING FOR SUSPENDED PROGRAM.
[33:15]	ADDRESS	VALUE TO BE TESTED AGAINST THE RESULT FROM THE PROCEDURE CALLED BY ACCESSING WORD 2.

WORD 2



FIELD	CONTENTS	DESCRIPTION
[0:48]	PROGRAM DESCRIPTOR	ACCIDENTAL ENTRY PROGRAM DESCRIPTOR WHICH WHEN ACCESSED WILL RETURN A VALUE OF 1 IF THE SUSPENDED PROGRAM CAN BE RE-ACTIVATED OR 0 (ZERO) IF IT CANNOT BE REACTIVATED.

AS CONDITIONS DICTATE, "NOTHINGTODO" SEARCHES THE "BED" TO DETERMINE IF A PROGRAM CAN BE REACTIVATED. THE FOLLOWING STATEMENTS INDICATE HOW THE TEST IS MADE.

```
P(64,STS); % SET STACK TO 100
FOR NT1:=0 STEP 2 UNTIL JOBNUM DO %SCAN BED ENTRIES
BEGIN P(INI);% TEST FOR INTERRUPTS
      NT2:=BED[NT1];% 1ST WORD OF ENTRY
      IF P(NT3:=BED[NT1+1],DUP,XCH,DEL,NOT) THEN% TEST FLAG BIT
                                                 % IN WORD 1
           BEGIN P1MIX:=[NT2].MIXF;% ACCIDENTAL ENTRY EVALUATION
                 P([NT2],[18:15],STS);% RESTORE S REGISTER FOR
                                      % EXPRESSION
                 NT3:=NT3;% CAUSES ACC. ENTRY & RETURN A ZERO OR
                          % ONE
                 P1MIX:=0;
                 P(64,STS);% SET STACK TO 100
           END;
      IF NOT(NT2 AND NT3) NEQ NOT 0 THEN% TEST FOR AWAKENING
           BEGIN P1MIX:=[NT2],MIXF;% WILL WAKE HIM UP
                 PROCTIME[P1MIX]:=PROCTIME[P1MIX]%
                      -P(RTR)-CLOCK;% UPDATE PROCESSOR TIME
                 IF JOBNUM NEQ NT1 THEN% IF NOT 1ST ENTRY IN BED
                 STREAM(N:=JOBNUM-NT1,A:=[BED[N1+2]],
                 B:=[BED[NT1]]);
                 BEGIN S:=A; DS:=N WDS END;% SLIDE OTHER ENTRIES
                                           % BACK
           JOBNUM:=JOBNUM-2;% DECRIMENT NUMBER OF ENTRIES IN BED
           SECONDCIR:=0;
           PRYOR[P1MIX].[FF]:=0;
           P([NT2],STF,XIT);% EXIT SNOOZE CALL TO ORIGINAL
                            % PROCESS
END;       END;
```

THE "BED" IS ORDERED IN PRIORITY, AS PROVIDED BY "SNOOZE" AND "COMPLEXSNOOZE", THUS THE SEARCH AT NOTHINGTODO PROCEEDS FROM THE HIGHEST (0) TO THE LOWEST (1022) PRIORITY.

BREAKOUT/RESTART

INTRODUCTION

THE B5700 MCP SUPPORTS AN EXTENSIVE "BREAKOUT/RESTART" FACILITY WHICH IS ACTIVE WHEN THE COMPILE-TIME OPTION "BREAKOUT" IS SET TRUE. IT ALLOWS A PROGRAMMER IN ALGOL OR COBOL TO SPECIFY THE LOCATION OF PROGRAM BREAKPOINTS WHICH, SUBJECT TO CERTAIN RESTRICTIONS, ALLOW THE RECOVERY OF THAT PORTION OF THE EXECUTION OF A JOB PRIOR TO THAT POINT IN THE EVENT OF OF ANY FORM OF SYSTEM FAILURE.

THIS PROCESS IS DESCRIBED IN TERMS OF THE CREATION (BREAKOUT) AND USE (RESTART) OF THE BREAKPOINTS.

BREAKOUT

GENERAL SPECIFICATIONS

ALGOL PROGRAMS CREATE A PERMANENT BREAK FILE ON DISK WHEN A BREAK STATEMENT IS EXECUTED. COBOL PROGRAMS ACT SIMILARLY WHEN A RERUN STATEMENT IS EXECUTED OR A RERUN CLAUSE INVOKED WITH ONE EXCEPTION: A "COBOL REEL RERUN" CLAUSE SPECIFIES THAT THE BREAK FILE IS PLACED ON THE BEGINNING OF AN OUTPUT REEL OF A TAPE FILE. IN THIS INSTANCE, A TEMPORARY BREAK FILE IS CREATED ON DISK, IS COPIED TO THE TAPE, AND THE DISK FILE IS DESTROYED ON COMPLETION OR TERMINATION OF THE COPY.

TAPE, DISK, AND PSEUDO-READER FILES AND LINE PRINTER FILES ARE PERMITTED TO BE OPEN WHEN A BREAK IS DONE. SORT FILES, CARD READER, SPO, CARD PUNCH, PAPER PUNCH, AND DATA COMMUNICATION FILES PRECLUDE BREAKING. IF BREAK IS ATTEMPTED WHILE ANY OF THE FORBIDDEN FILES IS OPEN, THE BREAK ATTEMPT IS IGNORED AND THE OPERATOR IS TOLD WHICH FILE INTERFERED WITH THE BREAK.

BREAKS ARE NUMBERED CYCLICALLY FROM 00 TO 99. THE NUMBERING OF BREAK FILES FOLLOWING A RESTART IS THE SAME AS WOULD OCCUR IF NO RESTART WAS DONE. THUS, IF A BREAK FILE 06 WAS USED TO RESTART A JOB, THE NEXT BREAK FILE CREATED BY THE FILE WOULD BE 07.

PROVISION HAS BEEN MADE FOR THE HANDLING OF WRITE ERRORS WHEN THE REEL OPTION IS EVOKED, FORCING THE BACKUP FILE TO TAPE. IF AN ERROR OCCURS THE OPERATOR WILL BE NOTIFIED OF A "BADUMP", THE REEL IS SWITCHED, AND A NEW BREAK FILE IS BUILT USING THE SAME BREAK NUMBER.

BREAK FILES ARE PROGRAM FILES NAMED ACCORDING TO THEIR BREAK NUMBER AND THE PROGRAM THAT BUILT THEM. THUS, BREAK FILES CREATED WHEN EXECUTING THE PROGRAM "BREAKER/A" WOULD BE NAMED "BREAKER/BREAKNN" WHERE EACH "NN" IS A BREAK NUMBER. THE FILES HAVE THIRTY WORDS PER RECORD, AND THE RECORDS ARE ALLOCATED IN 500-RECORD CHUNKS.

BREAK FILES DIFFER FROM OTHER PROGRAM FILES ALTHOUGH SEGMENT ZERO IS QUITE SIMILAR. THEY INCLUDE A COPY OF THE JOB-S CURRENT OVERLAY DISK, CODE, AND INTRINSICS THE JOB MIGHT USE. THE BREAK FILE ALSO HAS A MAP FOR EXACTLY REPLACING SOME OF THE JOB-S CORE AREAS, PRIMARILY THE NON-OVERLAYABLE AREAS.

BREAKOUT USUALLY TAKES ABOUT FIVE SECONDS BUILDING A TWO-CHUNK BREAK FILE, DURING WHICH TIME THE SYSTEM IS DISK BOUND. IT USES ABOUT 1500 WORDS OF CORE STORAGE.

- BREAKOUT/RESTART -

BREAKING DOES NOT AFFECT THE CONTENTS OF FILES. BOTH TEMPORARY AND PERMANENT DISK FILES ARE ENTERED IN THE DIRECTORY WHEN A BREAK OCCURS TO FACILITATE RESTART. TAPE FILES ARE MARKED TO BE SAVED.

ALL LABEL EQUATION INFORMATION IS RETAINED AND IS USED TO RECOGNIZE FILES AT RESTART.

THE BREAKOUT/RESTART FACILITY IS AVAILABLE ON SYSTEMS WHICH HAVE AUXILIARY MEMORY (OR DRUM). WHEN A PROGRAM WHICH IS USING AUXMEM DOES A BREAK, BOTH CODE AND DATA REFERENCES TO AUXMEM ARE CHANGED TO DISK. THE DATA ON AUXMEM IS COPIED TO DISK, AND THE BREAK IS PERFORMED.

SPECIFIC COMPONENTS

A PROGRAM WHICH DESIRES TO CREATE A BREAKOUT FILE PLACES PARAMETERS IN ITS STACK AND EXECUTES A COMMUNICATE INSTRUCTION. THE LAST PARAMETER IN THE STACK MUST SPECIFY COMMUNICATE NUMBER TWELVE. THE NEXT PARAMETER SPECIFIES WHETHER OR NOT THE BREAKOUT FILE IS TO BE WRITTEN ON TAPE. THE NEXT PARAMETER GIVES THE LOGICAL UNIT NUMBER OF THE BREAKOUT TAPE, IF ANY.

THE "COMMUNICATE" ROUTINE IN THE MCP CALLS COM12 WHICH IS A TYPELESS, PARAMETERLESS PROCEDURE WHICH ACTS AS THE "DRIVER" FOR THE BREAKOUT PROCESS. COM12 FIRST CALLS THE BREAK ROUTINE. THE BREAK ROUTINE IS THE HEART OF THE BREAKOUT FACILITY. IT IS A BOOLEAN PROCEDURE WHICH HANDLES THE CHECKING OF TYPES OF FILES TO DETERMINE IF THE BREAKOUT MAY BE DONE AT ALL. IT OUTWAITS OR DELAYS FILE ACTIVITY, RECORD POSITION AND ACCESS MODE OF THE FILES, AND ENSURES FILE INTEGRITY BOTH ON BREAKOUT AND RESTART. IF IT DECIDES THAT THE BREAKOUT CANNOT BE DONE, IT STOPS ITS OPERATION, SPOUTS THE MESSAGE "--CAN-T BREAK...", AND RETURNS WITH VALUE TRUE WHICH CAUSES COM12 TO EXIT.

THE PROCEDURE DECLARATION FOR BREAK IS AS FOLLOWS:

```
BOOLEAN PROCEDURE BREAK(TK); VALUE TK; INTEGER TK;
```

WHERE PARAMETER "TK" CONTAINS THE MEMORY LINK OF AN AREA OF MEMORY TO BE CHECKED FOR POSSIBLY BEING A FILE TANK, FILE BUFFER, ETC..

IF THE RESULT OF BREAK IS FALSE, COM12 THEN CALLS BREAKOUT TO PREPARE A TEMPORARY BREAKOUT DISK FILE IN THE PROPER FORMAT FOR RESTART, AND RETURNS THE ADDRESS OF THE DISK FILE HEADER -1, SUITABLE FOR ENTERUSERFILE, IN CASE COM12 MUST ENTER THE FILE PERMANENTLY. THE BREAKOUT ROUTINE ITSELF IS TYPED REAL AND IS PARAMETERLESS.

COM12 AT THIS POINT SPOUTS A MESSAGE OF THE FORM:

"PRIORITY:PREFIX/SUFFIX=MIX:BREAKNN BUILT".

THEN IT CHECKS R4. IF R4 IS NOT EQUAL TO 4, IT CALLS ENTERUSERFILE PASSING AS PARAMETERS THE PREFIX OF THE PROGRAM, THE BREAK FILE SUFFIX "BREAKNN", AND THE ADDRESS RETURNED BY BREAKOUT. OTHERWISE, IT ATTEMPTS TO WRITE ONTO THE TAPE DESIGNATED BY R5 THE TEMPORARY BREAKOUT DISK FILE BY CALLING BADUMP. THE PROCEDURE DECLARATION FOR BADUMP IS:

```
BOOLEAN PROCEDURE BADUMP(R,HZ); VALUE R,HZ; REAL R,HZ;
```

WHERE PARAMETER "R" DESIGNATES THE LOGICAL UNIT NUMBER OF THE TAPE AND "HZ" IS THE ADDRESS OF THE DISK FILE HEADER -1. THE RESULT OF BADUMP IS TRUE IF THE TAPE COULD NOT BE WRITTEN PROPERLY. THIS RESULT IS SENT BACK TO THE CALLING PROGRAM THROUGH R4.

COM12 THEN RELEASES ALL BREAKOUT SCRATCH STORAGE. IF IT WAS CALLED FROM AN EI KEYIN, IT DS-ES THE PROGRAM.

RESTART

GENERAL SPECIFICATIONS

IF THE REEL RERUN CLAUSE WAS SPECIFIED, THE BREAK FILE MUST BE LOADED FROM TAPE TO DISK. THIS IS ACCOMPLISHED BY THE "RS" OPERATOR MESSAGE WHICH CHECKS TO DETERMINE IF THE UNIT MENTIONED IN THE MESSAGE IS A LABELED, WRITE-ENABLED TAPE WITH A BREAK FILE COPY. IF IT IS THE SYSTEM WILL LOAD THE COPY UNLESS THERE IS ALREADY A PERMANENT DISK FILE WITH THE SAME NAME AS THAT OF THE COPY. AFTER LOADING, THE UNIT IS LEFT MARKED AND POSITIONED FOR THE PENDING RESTART. IF THE LOAD TRY FAILED, THE UNIT IS SET NOT-IN-USE AND LOCKED. NOTE THAT THE "RS" MESSAGE ONLY CAUSES THE RELOADING OF THE BREAK FILES. IT DOES NOT INITIATE RESTARTS.

FILES RE-OPENED WHILE RESTARTING MUST CORRESPOND CLOSELY TO THOSE OPEN WHEN THE BREAK FILE WAS BUILT. THE RE-OPENING FILES ARE THEREFORE THOROUGHLY CHECKED BEFORE ACCEPTANCE. A FILE-S UNIT TYPE MAY NOT BE CHANGED BETWEEN BREAK AND REOPENING. FILE REOPENING IS EXPLAINED IN THE FOLLOWING SECTIONS.

TO RESTART, ONE EXECUTES THE APPROPRIATE BREAKOUT FILE. MAPPED AREAS MUST BE REPLACED EXACTLY WHERE THEY WERE AT BREAKOUT. THIS REPLACEMENT IS CONSIDERED TO BE THE RESTARTING PROCESS. WHEN REPLACEMENT IS COMPLETE, THE OPERATOR IS NOTIFIED THAT THE JOB HAS RESTARTED, AND THE SYSTEM BEGINS REOPENING THE JOB-S FILES. REPLACEMENT SIDE EFFECTS ARE PRESENTED BELOW. RESTARTS MUST WAIT FOR PARTICULAR AREAS TO BECOME AVAILABLE, THEREFORE RESTARTS SHOULD BE PERFORMED WITH NO JOBS IN THE MIX, PREFERABLY IMMEDIATELY AFTER A HALT/LOAD.

IF THE JOB ORIGINALLY UTILIZED AULILIARY MEMORY (OR DRUM), ALL PROGRAM REFERENCES TO AUXMEM WERE CHANGED TO OVERLAY DISK SO THAT AUXILIARY MEMORY DOES NOT HAVE TO BE ON-LINE FOR A RESTART.

ALL CONTROL CARDS USED IN ORIGINALLY INITIATING THE PROGRAM ARE USED TO RESTART THAT PROGRAM. STACK, FILE, AND COMMON PROGRAM PARAMETER RECORDS ARE IGNORED BY THE RESTARTING PROCESS.

ONCE A RESTART JOB HAS BEEN INITIATED AND UNTIL THE RESTARTING PROCESS IS COMPLETE, THE SYSTEM IS OCCUPIED REPLACING OR REBUILDING THE JOB-S CORE AREAS. IT IS POSSIBLE THAT THE ENTIRE REQUIRED ADDRESS RANGE IS NOT AVAILABLE, THAT SOME IN USE AREA IS INTERFERING WITH THE REPLACEMENT. SIMPLY WAITING RESOLVES SOME INTERFERENCE. SOME AREAS MUST BE MOVED OR DENIED SPACE DURING RESTARTS AS IT IS NOT POSSIBLE TO WAIT SO THAT THE INTERFERENCE WILL CLEAR UP.

THERE ARE VARIOUS CONSEQUENCES OF REPLACEMENT. THE RESTART WILL BE AUTOMATICALLY "ES"-ED IF THE MEMORY CONFIGURATION DIFFERS FROM THAT AT BREAK. THE "IN", "OT", AND "ST" OPERATOR MESSAGES ARE NOT VALID WHEN APPLIED TO RESTARTING JOBS, NOR IS "CI" ALLOWED WHILE A JOB IS RESTARTING.

THE OPERATOR MAY MONITOR A JOB-S PROGRESS IN RESTARTING. IF HE REQUESTS "WY" ON THE RESTARTING JOB, HE WILL BE TOLD IF THE RESTART IS WAITING DUE TO INTERFERING AREAS. RARELY, HOWEVER, SHOULD THE MCP OR THE RESTARTING JOB ITSELF CAUSE INTERFERENCE.

A RESTART JOB THAT DOES NOT HAVE AN EXCESSIVE AMOUNT OF OVERLAY DISK USUALLY TAKES ABOUT FOUR SECONDS RESTARTING. NONE OF ITS STORAGE IS OVERLAYABLE. OF ITSELF, A RESTARTING JOB USES LESS CORE STORAGE THAN IT DID BREAKING. HOWEVER, WHEN RESOLVING INTERFERENCES INVOLVES MOVING MANY AREAS, A NO MEM SITUATION MAY OCCUR. IN THIS EVENT, MOVEMENT IS TEMPORARILY ABANDONED, AND THE SYSTEM RECOVERS, THOUGH A HALT/LOAD AND A NEW RESTART ATTEMPT MAY BE REQUIRED.

AFTER A JOB HAS RESTARTED, ITS FILES ARE REOPENED. THE SYSTEM FINDS FILES, CHECKS THEM AGAINST DATA SAVED AT BREAK, AND THEN REPOSITIONS THEM FORWARD ACCORDINGLY. IF A FILE CHECKED IS SOMEHOW UNSUITABLE, THE OPERATOR IS TOLD WHAT WAS WRONG, AND THE SYSTEM TRIES FINDING THE CORRECT FILE.

ONLY TAPE FILES ARE ACTUALLY SPACED. THE RESTART JOB IS TERMINATED IF AN IRRECOVERABLE PARITY OR OTHER PROBLEM OCCURS. POSITIONING OTHER FILES IS IGNORED. THEIR PROPER POSITIONING DERIVES FROM INTERNAL REFERENCES LEFT INTACT.

NOTE THAT FILE CHANGES MADE AFTER A BREAK ARE NOT CONSIDERED WHILE REOPENING THE FILE. FOR INSTANCE, ERRORS MAY RESULT IF A RECORD IS ADDED, DELETED, OR ALTERED. SUCH ERRORS NEED NOT APPEAR IMMEDIATELY AFTER THE BREAKPOINT.

A JOB-S TAPE, DISK, PSEUDO-READER, PRINTER, PRINTER BACK UP TAPE, AND PRINTER BACK UP DISK FILES MAY BE OPEN AT BREAK. REOPENING THESE TYPE FILES IS CONSIDERED BELOW.

NOTE THE CHECKS WHICH APPLY. TO BREAK AT ALL, OTHER TYPE FILES AND SORT FILES MUST HAVE BEEN CLOSED. THEY ARE IGNORED WHILE THE SYSTEM REOPENS FILES.

PRINTER FILES NEED NO ATTENTION WHILE BREAKING. AT RESTART, PRINTING CONTINUES WITH WHICHEVER INFORMATION LOGICALLY FOLLOWED THE BREAK. THE UNIT SELECTED IS NOT POSITIONED WITHIN THE PAGE OR PHYSICALLY LABELED. BACKUP FILES RESTART SIMILARLY, BUT WITH NEW FILES. ONLY TYPE CORRESPONDENCE IS CHECKED.

DISK FILES ARE PERMANENT-OR-NOT ACCORDING TO WHETHER THEY ARE

MENTIONED IN THE DIRECTORY AT BREAK. TEMPORARY FILES BECOME PERMANENT. THEIR SAVE FACTOR IS THEN SET TO SIXTY-THREE IF IT WAS PREVIOUSLY ZERO. THE DIRECTORY'S INFORMATION ABOUT PERMANENT FILES IN USE MAY BE INACCURATE, THEREFORE, IT IS UPDATED EACH BREAK.

AT RESTART, DISK FILES ARE CHECKED FOR BLOCKING, RECORD SIZE, NUMBER AND USE OF ROWS, ALLOWED NUMBER OF ROWS, AND END-OF-FILE SUITABILITY.

PSEUDO READERS ARE CHECKED FOR TYPE CORRESPONDENCE ONLY. THE SYSTEM'S NORMAL CHECKS DETECT SHORT CONTROL-DECKS, AND THE OTHER DISK FILE CHECKS ARE NOT RELEVANT. HOWEVER, CONTROL DECKS HAVE INTERNAL CONTROL CARDS LINKAGES, REFERENCES TO WHICH REOPENING DOES NOT CHECK. FURTHER, THE SYSTEM CAN NOT FIND A SUB-DECK UNLESS IT IS CURRENTLY IN A PSEUDO-READER.

TAPE FILES (NOT PRINTER OR PUNCH BACK UP) HAVE A PHYSICAL BLOCK COUNT RECORDED EACH BREAK. FINDING AN INPUT TAPE FILE MAY INVOLVE SEARCHING A MULTIFILE REEL. REOPENING TAPE FILES ARE FOUND AS THOUGH THEY ARE INPUT FILES - ONCE CHECKED AND ACCEPTED THEY ARE SPACED FORWARD ACCORDING TO THE DIFFERENCE BETWEEN CURRENT AND BREAK COUNTS. THUS, A TAPE IS EFFECTIVELY POSITIONED RELATIVE TO ITS REEL'S BEGINNING. AS ACCEPTED, TAPES ARE POSITIONED IN PARALLEL.

IF AN OUTPUT TAPE REEL WAS "RS"-ED TO LOAD THE BREAK JUST RESTARTED, THEN THE TAPE WAS CHECKED BY THE "RS" HANDLER, AND ITS UNIT WAS LEFT NOT-IN-USE, POSITIONED AND MARKED TO PREVENT JOBS WHICH ARE NOT RESTARTS FROM FINDING IT. THE UNIT IS THEREFORE CHECKED FOR MARKS THE "RS" HANDLER LEFT. IF THE UNIT IS UNMARKED, A TAPE IS FOUND, CHECKED LIKE OTHER TAPES, AND ALSO CHECKED FOR A BREAK FILE COPY.

OTHER TAPES (NOT BACK UP) ARE CHECKED FOR TYPE, LABEL, FORMAT, DUMP, AND WRITE-ENABLE CORRESPONDENCE; TAPE FORMAT CORRESPONDENCE IS WRONG IF THE FILE REQUESTED IS FOUND AFTER THE BLOCK BROKEN. TAPE LABEL CORRESPONDENCE IS WRONG IF THE BROKEN FILE HAD (LACKED) A LABEL AND THE FILE FOUND LACKED (HAD) ONE. DUMP CORRESPONDENCE IS WRONG IF THE FILE FOUND LACKS A NEEDED BREAK FILE COPY. WRITE-ENABLE CORRESPONDENCE IS WRONG IS THE FILE NEEDS (LACKS) A WRITE-RING.

SPECIFIC COMPONENTS

AS NOTED EARLIER, A PROGRAM IS RESTARTED BY THE OPERATOR OR USER EXECUTING THE APPROPRIATE BREAKOUT FILE. THIS IS TRUE EVEN IF THE FILE WAS ON TAPE, SINCE THE "RS <UNIT>" COMMAND SIMPLY PLACES THE BREAKOUT TAPE FILE ON DISK, AND DOES NOT INITIATE THE RESTART. THUS SELECTION IS THE ENTRY POINT FOR RESTARTING.

SELECTION CHECKS EVERY FILE WHICH IS EXECUTED FOR BEING A BREAKOUT FILE. A BREAKOUT FILE-S SEGMENT ZERO IS SIMILAR TO THAT OF A NORMAL PROGRAM CODE FILE IN THAT MANY FIELDS ARE SHARED. THE BITS IN THE HEADER SIGNIFYING THAT THE FILE CONTAINS EXECUTABLE CODE ARE SET (HEADER[4].[9:2]=3) FOR A BREAKOUT FILE. SELECTION RECOGNIZES A BREAKOUT FILE BY THE [2:1] OF SEGMENT ZERO BEING SET. IF THIS BIT IS SET AND BREAKOUT IS SET FALSE, THE FILE IS CONSIDERED NON-EXECUTABLE. IF THE BIT IS SET AND BREAKOUT IS SET TRUE, THE CURRENT MEMORY CONFIGURATION IS CHECKED AGAINST THAT STORED IN THE FILE IN TERMS OF MEMASK, MSTART, AND ADDRESS OF NFO. IF ANY DISCREPANCIES ARE FOUND, THE RESTARTING JOB IS ES-ED IN A SPECIAL MANNER BY CALLING FORK ON RSDSED. OTHERWISE RESTART IS CALLED WITH FIRST PARAMETER A POINTER TO SEGMENT ZERO OF THE FILE IN CORE AND SECOND PARAMETER A POINTER TO A SCRATCH CORE AREA. AFTER RESTART IS CALLED, SELECTION SETS RESTARTING TO THE NEGATIVE OF THE MIX NUMBER WHICH THE RESTARTING JOB HAS BEEN ASSIGNED TO FLAG THAT A JOB IS BEING RESTARTED AND TO IDENTIFY THAT JOB.

THE PROCEDURE DECLARATION OF RESTART IS AS FOLLOWS:

```
PROCEDURE RESTART(S,T); VALUE T; ARRAY S[*]; REAL T;
```

WHERE PARAMETER "S" POINTS TO SEGMENT ZERO FROM SELECTION AND PARAMETER "T" POINTS TO A SCRATCH CORE AREA TO BE USED FOR RESTART STACK. SINCE SELECTION HAS NOT SET UP THE JOB COMPLETELY, IT CANNOT BE NORMALLY TERMINATED UNTIL BOJ. THE RESTARTING VALUE OF THE NEGATIVE OF THE MIX INDEX AND CODE IN THE PROPER PLACES IN THE MCP ENFORCE THIS RULE.

THE RESTART ROUTINE BUILDS AN ARRAY ROW (ML) REFLECTING THE STATUS OF THE SET OF MEMORY LINKS AT BREAKOUT TIME, SETS UP REPLY[P1MIX] SUCH THAT THE OPERATOR MAY MONITOR THE PROGRESS OF RESTARTING WITH WY, MAY CLEAR CERTAIN CONDITIONS WITH OK, AND MAY DISCONTINUE THE RESTART WITH DS, AND SETS UP GRSD[P1MIX] INITIALLY AS POINTING TO THE ML ARRAY ROW. THEN RESTART CALLS FORK ON KERNEL AND EXITS TO SELECTION.

THE PROCEDURE DECLARATION FOR FORK IS AS FOLLOWS:

- BREAKOUT/RESTART -

```
SAVE PROCEDURE FORK(R,S,Q,MK); VALUE R,S,Q,MK;
     REAL R,S,MK; NAME Q;
```

WHERE PARAMETERS ARE INTERPRETED AS FOLLOWS:

PARAMETER	MEANING
R	ROUTINE TO BE CALLED
S	NEW STACK FOR ROUTINE
Q	ADDRESS TO BE TESTED (LIKE SNOOZE)
MK	MASK TO BE USED (LIKE SNOOZE)

THIS ROUTINE IS USED TO PASS P1MIX FROM ONE STACK TO STACK. IN CALLING FORK, ONE ASSURES THAT THE CONDITION NOT (M[Q]AND MK)NEQ(NOT 0) IS NOT IN FORCE. FORK WILL MAKE "R" PRESENT IF NECESSARY. SINCE OLAY CHECKS STACKS TO DECIDE A ROUTINE IS USELESS, FORK DOES NOT LOSE CONTROL ONCE "R" IS PRESENT. THE CALLING ROUTINE MUST, AFTER FORKING, RELINQUISH P1MIX BEFORE CONTROL LOSS AND ASSURE THE CONDITION WILL BE IN FORCE. FURTHER, IF "R" EXITS AFTER AWAKING IT WILL HANG HERE. THE METHOD FORK USES TO PERFORM ITS WORK IS TO FAKE A SNOOZE ENTRY FROM THE BEGINNING OF THE ROUTINE TO BE CALLED. THEN, WHENEVER THE CONDITION NOT(M[Q] AND MK)NEQ(NOT 0) IS SATISFIED, OUTER BLOCK CODE CAUSES CONTROL TO BE TRANSFERED TO THE ROUTINE WITH THE PROPER P1MIX. AS NOTED ABOVE, THE ROUTINE MAY NOT PERFORM A NORMAL EXIT BUT MUST BRANCH SOMEWHERE OR FORK TO SOME ROUTINE.

ONE ROUTINE WHICH IS FORK"ED IS KERNEL WHICH CONTROLS THE RESTART PROCESS. IT CALLS OTHER ROUTINES TO REOPEN FILES, MOVE MEMORY CONTENTS AROUND, ETC.. IT ITERATES UNTIL IT CAN ARRANGE MEMORY CONTENTS SUCH THAT THEY ARE COMPATIBLE WITH THOSE DESCRIBED IN THE ML ARRAY ROW POINTED TO BY GRSD[P1MIX]. SOME OF THE MORE INTERESTING CASES OCCUR WHEN THE KERNEL DISCOVERS THAT IT HAS TO MOVE ITSELF, SOME OF ITS OTHER ROUTINES, OR ITS STACK. CODE THROUGHOUT THE MCP SUPPORTS THIS MOVEMENT EITHER PASSIVELY OR ACTIVELY. IT SUPPORTS THE MOVEMENT PASSIVELY BY NEVER WATCHING THE JAR, CIDTABLE, NFO, AND OTHER MAJOR GLOBAL ARRAYS, AND LOSING CONTROL. IT SUPPORTS THE MOVEMENT ACTIVELY IN CERTAIN ROUTINES, SUCH AS READEMFROMDISK, WITH SPECIAL CODE TO CHECK FOR AN ACTIVE KERNEL AND TO FACILITATE ITS ACTIVITIES THROUGH SPECIAL ACTION. AS MENTIONED EARLIER, RESTART FORKS THIS ROUTINE TO ARRANGE PLACING THE AREAS ML DESCRIBES, (GRSD[P1MIX] DESCRIBES ML), WITH THAT GOAL. IT MAKES SUCCESSIVE PASSES AT MEMORY, PLACING EACH AREA AS SOON AS POSSIBLE THEN TRYING TO MOVE WHATEVER INTERFERED.

THE MOVE ROUTINES ARE MODULAR, SOME ARE REQUIRED, AND SOME NOT. EACH TIES RESTART CLOSER TO THE SYSTEM AND RENDERS MAINTAINENCE MORE

DIFFICULT. UNNECESSARY ROUTINES SHOULD BE JUDGED FOR THIS CONVENIENCE OVER THAT OF RESTART. THE KERNEL-S ITERATION IS NOT STRICTLY CONVERGENT, BUT MAY BE ENDED AT THE OPERATOR-S DISCRETION WITH A "DS" MESSAGE. HE MAY INITIATE A DIALOG TO MONITOR THE KERNEL-S PROGRESS WITH A "WY" MESSAGE.

AS KERNEL PROCEEDS, IT PERIODICALLY CHECKS TO SEE IF THE OPERATOR HAS, THROUGH KEYIN, CHANGED REPLY. IF SO, KERNEL SPOUTS A STATUS MESSAGE FOR A "WY" AND BRANCHES TO RSDSED IN CASE OF A "DS". "OK" IS NOT A VALID KEYIN TO THE KERNEL ITSELF, BUT IS TO REOPEN, WHICH IS CALLED BY THE RESTART, WHICH IS CALLED BY KERNEL , AND SPECIFIES THE RETRY OF REOPENING THE FILE WHICH CAUSED A "--WRONG..." OR "--WRONG FILE..." MESSAGE. KERNEL ITSELF MOVES MOST MEMORY AREAS AROUND TO AVOID MEMORY AREA CONFLICTS, HOWEVER, THE KERNEL CANNOT MOVE ITSELF OR ITS OWN STACK. IN ORDER TO DO THIS, AND IN SEVERAL OTHER SPECIFIC CASES, IT CALLS MOVETHEREST.

RSDSED IS A PROCEDURE WHICH IS UNTYPED AND PARAMETERLESS. IT IS EITHER "FORK"-ED BY SELECTION OR BRANCHED TO BY THE KERNEL. AT THE TIME WHEN THIS PROCEDURE IS EVOKED, THE JOB LACKS THE PREREQUISITES FOR NORMAL TERMINATION (STACK, PRT, ETC.). THEREFORE, THE PROCEDURE ATTEMPTS TO ENSURE THAT NORMAL TERMINATION IS NOT APPLIED.

REOPEN IS A PROCEDURE WHICH IS ALSO UNTYPED AND PARAMETERLESS. IT IS USED TO POSITION TAPE FILES CONTAINING "BREAK" INFORMATION. STANDARD ROUTINES ARE USED TO DETERMINE THE UNIT THE DESIRED TAPE IS MOUNTED ON. REOPEN THEN CHECKS THE BREAK INFORMATION AT THE START OF THE REEL TO ENSURE THE VALIDITY OF THE TAPE. ONCE THIS HAS BEEN ACCOMPLISHED, ALL REFERENCES TO THE FILE WHICH MAY HAVE BEEN ALTERED DURING THIS PROCESS ARE CORRECTED. ONE EXCEPTION IS THAT ODD FILES MARKED "CLOSED/RETAINED" AT BREAKOUT ARE NOW MARKED AS "CLOSED/RELEASED". REOPEN ALSO APPROPRIATELY MARKS MEMORY LINKS SO THAT TERMINATE AND FILECLOSE AY CLOSE THE FILE.

AT THE VERY END OF ITS SUCCESSFUL OPERATIONS, KERNEL BRANCHES TO RSUFFIX IN ORDER TO COMPLETE THE RESTART. RSUFFIX COMPLETES THE RELEVANT OPERATIONS OF SELECTION NOT PERFORMED WHEN RESTART WAS CALLED, REINITIALIZES OVERLAY TABLES AND DISK STORAGE, RESETS COBOL SAME AREAS TO GIVE INVALID ADDRESS TO THE PROGRAM IF IT REFERENCES THEM, REINITIALIZES FSROW, INTABLE, FPB, SEGMENT DICTIONARY, AND MAKES THE NECESSARY PROGRAM SEGMENTS PRESENT AGAIN. IT THEN FILLS IN ANY MEMORY AREAS NOT PREVIOUSLY FILLED IN. IT SPOUTS THE MESSAGE "...RESTARTED" AND REOPENS ALL FILES OPEN AT BREAKOUT TIME. ALL FINISHED, IT BRANCHES TO RETURN IN THE OUTER BLOCK TO BEGIN NORMAL-STATE PROCESSING.

THE ONLY MAJOR RESTARTING ROUTINE WHICH HAS NOT BEEN DESCRIBED PREVIOUSLY HERE IS REELER. WHEN KEYIN DISCOVERS THAT THE OPERATOR HAS ENTERED "RS", IT CALLS REELER WITH A PARAMETER WHICH IS A POINTER TO THE LOCATION IN MEMORY OF THE CHARACTER AFTER THE CHARACTERS RS IN THE MESSAGE ENTERED BY THE OPERATOR. REELER

DECODES THE INFORMATION, VERIFIES IT, AND IF IT IS CORRECT THUS FAR, CHECKS THE DESIGNATED UNIT FOR CONTAINING A BREAKOUT FILE. IF SO, IT COPIES IN THE FILE AND RECREATES THE DISK FILE USED TO CREATE THE TAPE FILE, THEN ENTERS IT IN THE DIRECTORY UNDER THE NAME UNDER WHICH IT WOULD HAVE BEEN ENTERED ORIGINALLY. IT DOES NOT AUTOMATICALLY EXECUTE THE BREAKOUT FILE.

SYSTEM EFFECTS

BREAKOUT-RESTART WAS DESIGNED TO ELIMINATE ITS CHARACTERISTIC OPERATOR MAINTENANCE REQUIREMENT. WITH THIS END IN VIEW, THE DESIGN CONFINES ITS ATTENTION TO THE JOB PRINCIPALLY INVOLVED, THEREBY MARKEDLY REDUCING TANGENTIAL INTERACTIONS WITH THE SYSTEM AND OTHER JOBS. TWO OF SUCH INTERACTIONS LEFT WARRANT ATTENTION.

NOTE THAT BREAKOUT SAVE ALL INTRINSICS THE JOB MAY USE. THIS ENSURES THE INTEGRITY OF REFERENCES THERETO, BUT USERS MAY OCCASIONALLY RESTART WITH FUNCTIONALLY DATED INTRINSICS, AN ERROR. TO AVOID POTENTIAL PROBLEMS, THE SAME MCP AND INTRINSICS USED AT BREAKOUT TIME MUST BE USED AT RESTART TIME.

BREAKOUT MESSAGES

"--CAN-T BREAK DATA FILE DESIGNATOR RDC: JOB SPECIFIER".
THE JOB TRIED BREAKING WITH A FILE OF UNSUITABLE TYPE OPEN. THE BREAK TRY IS IGNORED.

"PRIORITY: JOB SPECIFIER=MIX: BREAK NUMBER BUILT".
THE SPECIFIED JOB JUST BROKE, CREATING THE BREAK FILE PROGRAM NAME/BREAK NUMBER. THE BREAK FILE IS THEN MOVED TO AN OUTPUT TAPE, IF NECESSARY.

"PRIORITY: JOB SPECIFIER=MIX: BADUMP ON UNIT".
THE SYSTEM COULD NOT COPY A BREAK FILE ONTO THE TAPE MENTIONED. IT WILL TRY AGAIN ON A NEW REEL.

RESTART KEYIN

THE RS MESSAGE ALLOWS THE OPERATOR TO ADD A BREAK FILE TO THE DIRECTORY, THE BREAK FILE HAVING BEEN COPIED TO A COBOL JOB-S OUTPUT TAPE. THE RS MESSAGE FORMAT IS:

"RS UNIT",

THE RESPONSES ARE:

"RS UNIT INV KBD". UNIT IS NOT A TAPE.

"UNIT NOTE".
NOTE IS ONE OF NOT READY, IN USE, SCRATCH, WRITE LOCK, OR NO DUMP.

",PROGRAM NAME/ BREAK BREAK NUMBER NOT ADDED, DUP LIB RS UNIT."
THERE IS ALREADY A DISK FILE WITH THE NAMES MENTIONED. THE BREAK FILE ON THE UNIT IS NOT LOADED.

"UNIT ERROR IN DUMP"
THE TAPE-DISK BREAK FILE COPY WENT AWRY. THE BREAK FILE IS NOT LOADED.

"PROGRAM NAME / BREAK NUMBER ADDED, TAPE POSITIONED: UNIT."
THE RS UNIT WAS SUCCESSFUL. THE UNIT IS LEFT POSITIONED AND MARKED FOR THE PENDING RESTART OF THE LOADED BREAK FILE.

RESTART MESSAGES

"JOB SPECIFIER-MIX GONE TIME."
THE JOB WAS A RESTART WHICH WAS ES-ED OF DS-ED BEFORE HAVING RESTARTED.

"PRIORITY: JOB SPECIFIER-MIX RESTARTED."
THE DESIGNATED RESTART JOB HAS COMPLETED REPLACING CORE STORAGE AND NOW HAS A NORMAL JOB STRUCTURE. THE JOB WILL BEGIN REOPENING FILES NEXT.

"PRIORITY: JOB SPECIFIER-MIX RESTART IS STATE",
STATE IS WAITING OR MOVING.
THE OPERATOR REQUESTED MIX WY BEFORE THE JOB RESTARTED. THIS RESPONSE TELLS WHAT RESTART IS DOING.

"--MIX: MIX,,,,,,MIX IN THE WAY",
THE MIXES ARE USING CORE AREAS NEEDED TO REPLACE THE RESTARTING JOB-S STORAGE.

FILE REOPENING MESSAGES

"--WRONG FILE DATA FILE DESIGNATOR RDC: JOB SPECIFIER",
REOPENING INVOLVES CHECKING FILES FOR COMPATIBILITY WITH THOSE IN USE AT BREAKOUT. THE DESIGNATED FILE IS NOT SUFFICIENTLY COMPATIBLE AND THE NEXT MESSAGE HINTS WHY. THE OPERATOR MAY RESPOND WITH AN "OK", "WY", OR "DS" REPLY, "OK" INITIATING A RECHECK.

"--WRONG HINT, ,,,,, HINT."
THE HINTS ARE AMONG THE FOLLOWING:

WRITE STATE THE FILE-S WRITE RING IS IN THE WRONG PLACE (IN THE BOX OR ON THE TAPE).

LABEL	THE FILE LACKS (NEEDS) A LABEL.
TYPE	A FILE ON DISK (TAPE, LINE PRINTER) AT BREAK MUST BE ON DISK (TAPE, LINE PRINTER) AT RESTART. FOR EXAMPLE, A PRINTER FILE WAS BROKEN AND THE OPERATOR TRIED REOPENING IT AS A PRINTER BACK UP DISK.
ROWS USED	DISK FILES HAVE UP TO TWENTY ROWS. THE ONE FOUND DOES NOT HAVE THE RIGHT NUMBER OF ROWS.
NO. OF ROWS	THE DISK FILE FOUND HAS THE WRONG NUMBER OF ROWS ALLOWED.
EOF	IT IS TOO SHORT.
ROW LENGTH	ITS ROWS ARE THE WRONG SIZE.
FORMAT	IF IT IS A DISK FILE, ITS BLOCKING OR RECORD LENGTH IS WRONG. IF IT IS A TAPE FILE, IT WAS FOUND BEYOND WHERE IT WAS IN BREAKOUT.
SECURITY	A FILE SECURITY ERROR OCCURRED.
NO DUMP	THE TAPE FILE LACKS A BREAK FILE COPY.

BUFFER (NORMAL)

WORD	FIELD	CONTENTS
-1		SIZE=1 OF (N)
0		LOCATQUE SKELETON
		(FIB[10],[3:15] POINTS HERE)
	[0:3]	=5
	[3:5]	=MUST BE ZERO
	[8:10]	= LOGICAL UNIT NUMBER
	[FF]	= POINTER TO WORD TWO OF NEXT BUFFER
	[CF]	= ADDRESS OF TOP IO DESCRIPTOR
1		WORDCOUNT FOR ALL BUT DISK
		ALTERNATE DISK ADDRESS
2		DESCRIPTOR POINTS HERE
		ADDRESS FOR DISK
N-1		AS PER WORD 1 FOR REVERSE FILES
N		AS PER WORD 0 EXCEPT FF POINTS
		AT (N-2) OF NEXT BUFFER.

* BUFFER (PRINTER BACK UP) *

BUFFER (PRINTER BACK UP)

WORD	FIELD	CONTENTS
-1		LOCATQUE WORD (COPY)
0		LOCATQUE WORD (FIB [10],[3:15] POINTS HERE)
1		5 AREAS OF 18 WORDS EACH
1-17		PRINT RECORD
18	[3:30]	IO DESCRIPTOR
	[CF]	RECORD NUMBER (IN FIB[5],[FF], TOO)

BUILDLABEL

BUILDLABEL(LABLE,MID,FID,REEL,CDATE,CYCLE,PFACT,PTN,BLKODE,BSIZE,RSIZE) IS A PROCEDURE WHICH CONSTRUCTS A STANDARD TAPE LABEL IN "LABLE" WITH THE SPECIFIED PARAMETERS.

BYPASS

USED BY ENTERUSERFILE TO LOCATE THE ENDS OF THE REGULAR AND BYPASS DIRECTORIES.

BYPASS.[18:15] POINTS AT END OF REGULAR DIRECTORY.

BYPASS.[33:15] POINTS AT END OF BYPASS DIRECTORY.

CCMASK1

CCMASK1 IS A MASK FOR DEFAULT ALLOWABLE CONTROL CARD RESERVED WORDS FROM REMOTES AND IS DEFINED AS FOLLOWS:

WORD	CCMASK1 BIT	STANDARD MASK
NOT USED	0-22	---
UNLOCK	23	NO
USE	24	NO
LOCK	25	NO
FREE	26	NO
PUBLIC	27	NO
USER	28	NO
RUN	29	YES
COMPILE	30	YES
EXECUTE	31	YES
DUMP	32	NO
UNLOAD	33	NO
ADD	34	NO
LOAD	35	NO
REMOVE	36	NO
CHANGE	37	NO
UNIT	38	NO
END	39	YES
DATA	40	NO
LABEL	41	NO
SET	42	NO
RESET	43	NO
NOT USED	44	---
NOT USED	45	---
NOT USED	46	---
FILE	47	YES

CCMASK2

CCMASK2 IS A MASK FOR DEFAULT ALLOWABLE CONTROL CARD RESERVED WORDS FROM REMOTES AND IS DEFINED AS FOLLOWS:

WORD	CCMASK2 BIT	STANDARD MASK
EXPIRED	0	NO
ACCESSED	1	NO
PROCESS	2	YES
IO	3	YES
PRIORITY	4	NO
COMMON	5	YES
CORE	6	NO
STACK	7	NO
SAVE	8	YES
NOT USED	9	---
NOT USED	10	---
NOT USED	11	---
ALGOL	12	YES
XALGOL	13	NO
FORTRAN	14	YES
TSPOL	15	YES
BASIC	16	YES
COBOL68	17	YES
WITH	18	NO
COBOL	19	YES
LIBRARY	20	NO
SYNTAX	21	NO
FROM	22	NO
TO	23	NO

CCTOG

COUNTER FOR THE NUMBER OF USERS OF THE RSWDS ARRAY. WHEN CCTOG BECOMES NONPOSITIVE, THE MEMORY LINK BEFORE RESWDS IS CHANGED TO MARK THE AREA AVAILABLE.

CHANNEL

DESCRIPTOR POINTING TO THE CHANNEL ARRAY. CHANNEL[I] CONTAINS LOGICAL UNIT OF LAST DESCRIPTOR SENT OUT ON CHANNEL I. CHANNEL[0] WILL CONTAIN THE LOGICAL UNIT NUMBER THAT WAS ATTEMPTED BUT TERMINATED AS A RESULT OF AN I/O BUSY CONDITION.

CHECK

CHECK IS A KEYBOARD OPTION ENABLED BY SETTING THE COMPILE-TIME MODULE CHECKLINKS. IF CHECK IS TRUE, ALL MEMORY LINKS WILL BE CHECKED EACH TIME A "GETSPACE" IS PERFORMED. IF A "BAD" LINK IS FOUND, THE MESSAGE "INVALID LINK" WILL BE DISPLAYED ON THE SPO AND THE SYSTEM WILL ENTER A "DO UNTIL FALSE" LOOP. THE BAD LINK SHOULD THEN BE IN THE TOP OF THE STACK.

- CIDROW FORMAT -

CIDROW FORMAT

```
                        94 WORDS LONG
-1                      INDEX INTO CIDTABLE (LOGICAL UNIT NUMBER-32)
0                       ADDRESS OF TOP BUFFER
1                       ADDRESS OF ALTERNATE BUFFER
2                       DECK FILE ID
3                       RECORD COUNT
            [2:1]       RESERVED BREAKOUT RESTART
4                       RECORD NUMBER OF NEXT CONTROL CARD
5                       WORDS USED IN CURRENT BLOCK
6                       LINK TO NEXT DECK ON DISK
7-29                    FILE HEADER
30                      RESULT OF LAST DISK I/O ON FIRST BUFFER
31                      DISK ADDRESS
32-61                   FIRST RECORD BUFFER
62                      RESULT OF LAST I/O ON SECOND BUFFER
63                      DISK ADDRESS
64-93                   SECOND RECORD BUFFER
```

THE MCP HANDLES THE PSEUDO-READER INPUT IN A PING-PONG FASHION, ALTERNATING THE USE OF THE FIRST AND SECOND BUFFERS, HOPEFULLY USING ONE FOR PROCESSING AND THE OTHER FOR INPUT.

THE ROUTINE "READFROMDISK" IS USED FOR ALL ACCESSES TO PSEUDOREADERS, ONCE THEY HAVE BEEN OPENED. ITS PROCEDURE DECLARATION IS:

```
 BOOLEAN PROCEDURE READFROMDISK(H,IB);
      VALUE H,IB;
      ARRAY H,IB;
```

WHERE "H" IS A POINTER TO CIDROW FOR THIS PSEUDOREADER-S LOGICAL UNIT NUMBER AND "IB" IS A POINTER TO THE AREA IN WHICH TO PLACE THE NEXT CARD IMAGE.

"READFROMDISK" RETURNS A VALUE OF "TRUE" IF "IB" CONTAINS A CONTROL CARD IMAGE AND "FALSE" IF "IB" CONTAINS A NON-CONTROL CARD IMAGE.

CLOCK

CONTAINS NUMBER OF TIME INTERVAL INTERRUPTS PROCESSED SINCE LAST H/L TIMES 64.

COMMUNICATES

TO USE THE COMMUNICATE OPERATOR, A NORMAL STATE PROGRAM FIRST PLACES PARAMETERS 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 ACCESSES (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 NO-OP IN CONTROL STATE.

IF THE VALUE FOUND IN (R+9) BY COMMUNICATE IS NEGATIVE, THE PROGRAM WILL BE DS-ED WITH THE MESSAGE "-INVALID COM". IF THE VALUE IS TOO LARGE OR NOT INTEGERIZED, UNDEFINED RESULTS, INCLUDING SYSTEM HANGS, MAY BE EXPECTED. NEGATIVE COMMUNICATE PARAMETER NUMBERS ARE USED BY THE TIME SHARING SYSTEM.

TO SHORTEN THIS DISCUSSON, LET US CONSIDER THE STACK SETUP AT THE TIME OF THE COM OPERATOR.

```
RCW             (COMMUNICATE)
MSCW            (COMMUNICATE)
IRCW            ((R+9) REGISTER POINTS AT IRCW)
ICW
A REGISTER
B REGISTER
PARAMETER 1
PARAMETER 2
PARAMETER 3
PARAMETER 4
PARAMETER 5
.
.
.
.
.
```

PARAMETER 1 IS REFERENCED AS (F-4), PARAMETER 2 IS REFERENCED AS (F-5), ETC.. THUS THERE IS A CONSISTENT NOTATION USED IN THE MCP REFERENCING THE PARAMETERS OF A COM, AS FOLLOWS:

- COMMUNICATES -

RELATIVE ADDRESS	PARAMETER	REAL	INTEGER	ARRAY	NAME
F-4	1	R4	I4	A4	N4
F-5	2	R5	I5	A5	N5
F-6	3	R6	I6	A6	N6
F-7	4	R7	I7	A7	N7
F-8	5	R8	I8	A8	N8

COM CODE	DESCRIPTION
0	"-INVALID EOJ" ERROR TERMINATION. IT IS NORMALLY USED IN COBOL, FORTRAN, AND TSPOL.
1	TIME FUNCTION. I4 CODE — VALUE RETURNED IN I4 LSS -1 — I4 -1 — USER CODE 0 — DATE (YYDDD) 1 — TIME OF DAY IN SIXTIETHS OF SECONDS 3 — IO TIME IN SIXTIETHS OF SECONDS 4 — MACHINE TIME VALUE 0-64 5 — DATE (MMDDYY) 6 — NAME OF DAY OF WEEK MINUS SYLLABLE "DAY" GTR 6 — I4
2	WAIT CALL FROM ALGOL, UNTIL CALL FROM COBOL. IT INVOKES SLEEP([MIA5]],R4) THEN RETURNS TO THE CALLING PROGRAM WHEN SLEEP RETURNS.
3	RETURNS SPECIFIC ARRAY AND AIT STORAGE. THE POINTER TO THE ARRAY DESCRIPTOR MUST IN N4, AND THE NUMBER OF DIMENSIONS TO BE RETURNED MUST BE IN R5.
4	ZIP WITH, IN ALGOL, AND PERFORM WITH, IN COBOL, OR ARRAY ROW OR FILE NAME. THE ARRAY OR FILE DESCRIPTOR MUST BE IN A4, THE SIZE FIELD OF A FILE DESCRIPTOR IS ZERO AND THAT OF AN ARRAY DESCRIPTOR IS NONZERO, SO THIS IS USED FOR DISCRIMINATION PURPOSES.
5	END OF JOB. IT HAS NO PARAMETERS. IT MUST BE EXECUTED WHEN THE BLOCKCOUNTER = 1 OR PROPER ACTION WILL NOT BE TAKEN.
6	WHEN FUNCTION. I4 CONTAINS THE NUMBER OF SECONDS TO WAIT BEFORE REACTIVATING THE PROGRAM.

7 FILL ARRAY ROW.

I4 CONTAINS INDEX OF THE FILL SEGMENT IN THE SEGMENT DICTIONARY, AND A5 CONTAINS THE ARRAY ROW DESCRIPTOR FOR THE ARRAY TO RECEIVE THE FILL INFORMATION. THE MINIMUM OF THE NUMBER OF WORDS IN THE FILL AND IN THE ARRAY ROW IS MOVED.

8 ZIP PERFORM WITH TWO PARAMETERS IN ALGOL OR COBOL.

R4 CONTAINS THE SUFFIX OF THE JOB TO BE EXECUTED AND R5 CONTAINS THE PREFIX.

9 FILL ARRAY ROW WITH INQUIRY.

(NOT APPLICABLE TO B407 DATA COMMUNICATIONS II SYSTEM.)

10 BLOCK EXIT.

THIS COMMUNICATE RETURNS BLOCK STORAGE. IT HAS NO PARAMETERS.

11 ALGOL IO FUNCTIONS INTERFACE.

R4 PARAM VALUE	FUNCTION
0	FILEOPEN(0,A5,[CF])
1	"-PAR NO LABEL..."
2	"-EOF NO LABEL..."
3	"-EOT NO LABEL..."
4	GETUSERDISK...
5	DATA COMM II SEEK
6	FILECLOSE(NFLAG(A5))
7	"RER NO LABEL..."
8	"-SELECT ERROR..."
9	ALGOL SPACE STATEMENT
10	REFILL BUFFERS
11	READ NEXT LABEL ON MULTI-FILE TAPE
12	IOREQUEST...
13	ROTATE BUFFERS
14	DATA COMM II READ
15	DATA COMM II READ SEEK
16	DATA COMM II BUFFER RELEASE
17	DATA COMM II WRITE
18	DATA COMM II BUFFER LOCATE
19	DATA COMM II COBOL WRITE
20	SAVE UNIT WITH LOGICAL UNIT NUMBER R6

THEN GIVE "-RER NO LABEL..."

OTHER VALUES OF R4 WILL POSSIBLY CAUSE SYSTEM HANGS.

12 BREAK IN ALGOL,
RERUN IN COBOL.

THIS COMMUNICATE IS INACTIVE UNLESS MCP COMPILE-TIME OPTION BREAKOUT IS SET TRUE.

R4 = 4 MEANS COBOL REEL RERUN, OTHER VALUES MEAN NORMAL BREAK OR RERUN.

R5 = LOGICAL UNIT NUMBER OF TAPE FOR COBOL REEL RERUN, OR ZERO.

13 COBOL IO FUNCTIONS INTERFACE.

THIS COMMUNICATE CALLS COM13, WHICH DEFINES THE PARAMETERS REQUIRED AND IS NORMALLY USED FOR COBOL OPEN AND CLOSE ONLY.

14 INVERT OVERLAYABLE STATUS OF AN ARRAY ROW.

N4 POINTS TO THE DESCRIPTOR OF THE DESIRED ARRAY ROW.

15 DISPLAY FOR COBOL, FILE TYPE 11 OUTPUT FOR ALGOL, OUTPUT TO LABEL-EQUATED SPO FOR OTHERS.

A4 POINTS TO ONE WORD BEFORE THE INFORMATION TO BE DISPLAYED ON THE SPO; UP TO 20 WORDS MAY BE DISPLAYED; THE INFORMATION MUST BE TERMINATED WITH A GROUP MARK.

16 ACCEPT FOR COBOL, FILE TYPE 11 INPUT FOR ALGOL, INPUT FROM LABEL-EQUATED SPO FOR OTHERS.

A4 POINTS TO TWO WORDS BEFORE THE INFORMATION TO BE DISPLAYED. THE PROGRAM THEN WAITS FOR THE OPERATOR TO ENTER A "AX" RESPONSE. THE CORE AREA MUST BE AT LEAST 10 WORDS LONG OR INVALID LINK MAY RESULT.

17 IO ERROR MESSAGES CALLED FROM COBOLFCR AND SORT INTRINSICS.

A5 POINTS TO THE BASE OF THE FIB OF THE SUBJECT FILE. IF FIB[5].[1:1] THEN THE PROGRAM IS DS-ED WITH THE MESSAGE "-- INVALID USER...". OTHERWISE R4 CONTAINS THE ERROR MESSAGE NUMBER TO BE PRINTED IN THE FORM "-IO ERR NN..."; R6 CONTAINS A FLAG THAT THE PROGRAM IS (=1) OR IS NOT (=0) TO BE DS-ED; R8 CONTAINS THE REEL NUMBER TO BE PRINTED IN THE MESSAGE; AND R7 CONTAINS THE DATE. THE ROUTINE FILEMESS IS CALLED TO SPOUT THE MESSAGE AND DS THE JOB.

A LIST OF THE I/O ERROR NUMBERS AND THEIR MEANINGS CAN BE FOUND IN THE SECTION ON "I/O ERROR MESSGES".

18 INQUIRY WRITE.

(NOT APPLICABLE TO B487 DATA COMMUNICATIONS II SYSTEM.)

19 PRINT OR PUNCH BACK-UP - A PBT OR PBD.

THIS IS A SPECIAL PURPOSE COMMUNICATE WHICH IS NORMALLY USED ONLY BY PRNPBT DISK. R4 IS AS FOLLOWS:

R4.[18:15] = OUTPUT PRINT OR PUNCH LOGICAL UNIT NUMBER.

IF R4.[33:15] LSS @37777 THEN R4.[33:15] = PBD NUMBER IN OCTAL 8. OTHERWISE, R4.[42:6] = INPUT PBT OR PUT LOGICAL UNIT NUMBER.

PRT CELL @25 OF THE CALLING PROGRAM MUST BE A SIMPLE VARIABLE, INITIALLY ZERO. NO LIST-DIRECTED IO MAY HAVE BEEN PERFORMED BY THE CALLING PROGRAM PREVIOUS TO PERFORMING THE COMMUNICATE.

THE FIRST FIVE WORDS IN THE FPB ARE USED FOR THE INPUT FILE AND THE NEXT FIVE WORDS ARE USED FOR THE OUTPUT FILE. THE ROUTINE CALLED BY THIS COMMUNICATE IS COMM.

20 TAPE SWAP FOR TAPE SORT.

THIS IS A SPECIAL PURPOSE COMMUNICATE USED ONLY FOR TAPE SORTING.

21 SORT STORAGE ASSIGNMENT.

THIS IS A SPECIAL PURPOSE COMMUNICATE USED BY THE SORT TO ASSIGN CORE STORAGE.

THE SORT ROUTINE IS REQUESTING THE ASSIGNMENT OF R6 AREAS EACH R5 WORDS LONG. UPON EXIT, N4 POINTS TO AN AREA CONTAINING (R5*R5) WORDS WHICH DESCRIBES HOW THE SPACE WAS ACTUALLY ALLOCATED FOR THE REQUEST.

THE CONQUER ROUTINE IS USED TO ACTUALLY GET THE SPACE REQUESTED. IT ATTEMPTS TO GET THE SPACE ALL IN ONE AREA. FAILING THIS, IT SUBDIVIDES THE REQUEST UNTIL IT HAS EITHER GOTTEN THE SPACE OR HAS SUBDIVIDED INTO MORE THAN R6 AREAS; IN THE LATER CASE, IT SLEEPS UNTIL CLOCK CHANGES AND ATTEMPTS TO REPROCESS THE ORIGINAL REQUEST, REPEATING UNTIL IT HAS THE SPACE OR IS DS-ED.

22 SORT STORAGE RETURN.

THIS COMMUNICATE RETURNS THE STORAGE GOTTEN BY COMMUNICATE 21.

N4 POINTS TO THE AREA DESCRIBED UNDER COMMUNICATE 21 WHICH SPECIFIES HOW THE STORAGE WAS ALLOCATED TO THE SORT ON REQUEST.

23 LOAD CONTROL.

THIS IS A SPECIAL PURPOSE COMMUNICATE NORMALLY USED ONLY BY LDCNTRL DISK. IT HAS NO PARAMETERS.

PRT CELL @25 OF THE CALLING PROGRAM MUST BE A SIMPLE VARIABLE, INITIALLYZERO. THE FIRST FIVE WORDS OF THE FPB ARE USED FOR THE INPUT FILE AND THE NEXT FIVE WORDS ARE USED FOR THE OUTPUT FILE.

THE ROUTINE CALLED BY THIS COMMUNICATE IS COM23.

24 RETURN ONE ROW OF A DISK FILE.

A4 POINTS TO THE HEADER OF THE DISK FILE IN CORE.

R5 IS THE INDEX (RANGE 10 TO 29) OF THE ROW TO BE RETURNED IN THE HEADER.

THE COMMUNICATE RETURNS THE A4[8] DISK SEGMENTS BEGINNING AT A4[R5] TO THE AVAILABLE DISK TABLE AND ZEROES A4[R5].

25 RETURN OLD COPY OF OWN ARRAY.

A5 IS A COPY OF THE DESCRIPTOR OF THE OWN ARRAY.

R4 IS THE NUMBER OF DIMENSIONS IN THE OWN ARRAY. THE ORIGINAL DESCRIPTOR IS MODIFIED TO APPEAR AS IT DID WHEN THE PROGRAM BEGAN TO RUN.

26 INVALID ARGUMENTS TO MATH INTRINSICS.

THIS COMMUNICATE IS CALLED WITH A CODE NUMBER IN I4 WHICH SPECIFIES WHICH TYPE OF INVALID ARGUMENT ERROR HAS BEEN ENCOUNTERED AND THUS, WHAT MESSAGE WITH WHICH TO DS THE PROGRAM.

I4	TERMINAL MESSAGE
0	"-NEGTV ARGMNT LN..."
1	"-NEGTV ARGMNT LN..."
2	"-ZERO ARGMNT LN..."

3 "-MAXN ARGMNT EXP..."
4 "-MAXN ARGMNT SIN..."
5 "-MAXN ARGMNT COS..."

OTHER VALUES OF I4 MAY CAUSE SYSTEM HANGS, PRIMARILY INVALID LINKS.

27 COBOL DATACOMM II INTERROGATE.

THIS IS THE MCP PART OF THE COBOL CONSTRUCT:

MOVE FILENAME FROM TU, BUFF [AFTER CHECK] TO STATUSWORD.

PARAMETER	MEANING
A4	POINTER TO STATUSWORD
R5.[1:1]=1	FLAG FOR ACTIVE INTERROGATE
R5.[44:4]	BUFF
R6	TU
A7	POINTER TO PREVIOUS STATUSWORD USED IN CASE TU=0.

28 ALGOL DATACOMM II INTERROGATE.

THIS IS THE MCP PART OF THE ALGOL FUNCTION STATUS(TUBUF, OPTION) WHERE TUBUF=0&TU[9:44:4]&BUF[14:44:4].

R4 REPRESENTS OPTION AND R5 PRESENTS TUBUF, AS ABOVE.

PARAMETER	VALUE	MEANING
R4	0	PASSIVE INTERROGATE
R4	1	ACTIVE INTERROGATE
R4	3	NEGATE USERCODE[P1MIX]
R4	4	MAKE USERCODE[P1MIX] POSITIVE
R5, R4=0	0-0(=TU-BUF)	PASSIVE INTERROGATE ON 0-0 CHARGES TIME TO STATION 0-0 AND RETURN 0
R5, R4=0 OR 1	TU-BUF	ACTIVE-PASSIVE INTERROGATE ON TU-BUF CHARGES TIME TO STATION TU-BUF AND RETURNS STATUS OF STATION TU-BUF

29 DS OR SPOUT MESSAGE ABOUT PROGRAM.

THIS COMMUNICATE PROVIDES FOR DS-ING AN OBJECT PROGRAM OR SPOUTING A MESSAGE ABOUT A PROGRAM. R4 IS USED TO SPECIFY THE MESSAGE REQUIRED. R5 IS SET TO TRUE SPECIFIES P1MIX IS TO BE DS-ED.

VALUE OF R4	MESSAGE
0	-DEC ERR = NO. DISK ROWS = NNNN
1	-DEC ERR = ARRAY DIMENSION = NNNN
2	(SAME AS R4 = 0)
3	-MAXN ARGMNT EXP...

OTHER VALUES OF R4 WILL PROBABLY CAUSE SYSTEM HANGS, DUE TO INVALID LINKS.

30 ALGOL SEARCH AND COBOL RECORD FILE STATUS.

THIS COMMUNICATE SEARCHES THE DISK DIRECTORY AND RETURNS DATA IN ARRAY.

A4 POINTS TO THE ARRAY AREA. THE ARRAY MUST BE AT LEAST 7 WORDS LONG.
N5 POINTS TO THE TOP IO DESCRIPTOR OF THE FILE.
A4[0] IS USER-TYPE OR NOT-PRESENT FLAG.
A4[1] IS <MFID>.
A4[2] IS <FID>.
WHERE: NOT-PRESENT FLAG IS -1
INVALID USER FLAG IS 0
PRIMARY USER FLAG IS 7 (LM, INPUT, AND OUTPUT BITS)
SECONDARY USER FLAG IS 3 (INPUT AND OUTPUT BITS)
TERTIARY USER FLAG IS (INPUT BIT ONLY)

IF PRIMARY, SECONDARY, OR TERTIARY USER:
A4[3] IS RECORD LENGTH
A4[4] IS BLOCK LENGTH
A4[5] IS END OF FILE POINTER
A4[6] IS OPEN COUNT:
 ----[43:5] FOR SYSTEM A
 ----[38:5] FOR SYSTEM B
 ----[33:5] FOR SYSTEM C
 ----[28:5] FOR SYSTEM D

ALSO, IF ARRAY IS LONGER THAN 9 WORDS:
A4[7] IS FILE TYPE
A4[3] IS HEADER[8]
A4[9] IS HEADER[1]

31 ALGOL DELAY FUNCTION.

THIS COMMUNICATE PROVIDES FOR A WAIT CALL WITH A SPECIFIED TIMEOUT AND RETURNS AN INDICATION OF WHETHER CONDITION WAS SATISFIED OR NOT.

I4 CONTAINS THE NUMBER OF SECONDS TO WAIT BEFORE TIMEOUT

ACTION IS TAKEN.

R5 CONTAINS THE MASK WHICH IS TO BE COMPARED AGAINST THE WORD POINTED TO BY A6.

I4 WILL BE SET TO 1 ON RETURN IF THE MASK CONDITION WAS SATISFIED AND 0 IF TIMEOUT CONDITION WAS SATISFIED.

A6.[CF] MUST BE GREATER THAN 511 OR "=INVALD ADDRESS" ERROR TERMINATION WILL RESULT.

32 B487 DATA COMMUNICATIONS SEEKS, DETACHES, AND INTERROGATES.

THIS IS A GENERAL PURPOSE COMMUNICATE USED FOR SEVERAL CONSTRUCTS EACH IN ALGOL AND COBOL.

PARAMETER	COBOL STACK	ALGOL STACK
A4	ARA FID	ARA FID
R5	BUF	TUBUF
R6	TU	0
R7	FUNC	FUNC

R7	R6	R5	A4	ACTION
0	0	TUB	FID	ALGOL SEEK
0	TU	BUF	FID	COBOL SEEK
2	0	TUB	0	ALGOL DETACH
2	TU	BUF	0	COBOL DETACH
5	TU	BUF	ARA	COBOL RECORD DATA STATUS
1	0	0	ARA	ALGOL STATUS(A[*])
1	0	TUB	0	ALGOL STATUS(STA)
4	0	0	FID	ALGOL STATUS(FID)

33 FORTRAN PAUSE STATEMENT.

THIS COMMUNICATE COMPRISES THE MCP PORTION OF THE FORTRAN PAUSE STATEMENT.

I4 CONTAINS THE PAUSE CODE IN THE LAST SIX CHARACTERS. THE MESSAGE

"PAUSE #XXXXXX FOR ..." IS TYPED, THEN THE PROGRAM IS STOPPED, AS IF THE OPERATOR HAD ENTERED "ST". THE OPERATOR THEN HAS THE CHOICE OF ENTERING OK TO ALLOW THE JOB TO CONTINUE OR DS TO KILL THE PROGRAM.

34 GENERALIZED ERROR TERMINATION.

THIS COMMUNICATE IS NORMALLY USED WHENEVER AN INTRINSIC MUST TERMINATE WITH A "CUSTOM BUILT" ERROR MESSAGE. THE INTRINSIC BUILDS THE MESSAGE IN A SAVE AREA.

R5 POINTS TO AN AREA OF CORE CONTAINING THE MESSAGE TO BE USED IN THE ERROR TERMINATION. THE MESSAGE SHOULD START WITH A HYPHEN, END WITH A GROUP MARK, AND BE NO LONGER THAN 80 CHARACTERS. FAILURE TO SATISFY THESE REQUIREMENTS MAY RESULT IN SYSTEM HANGS DUE TO INVALD LINK.

35 LIBMAIN COMMUNICATE.

THIS COMMUNICATE SHOULD BE USED ONLY BY LIBMAIN/DISK.

PARAMETERS COME FROM COMMON AND ESPDISK.

36 B487 DATA COMM II FILE TYPE REMOTE I-O INTERFACE.

THIS COMMUNICATE IS THE INTERFACE BETWEEN THE MCP AND THE INTRINSICS FOR ALL FILE TYPE REMOTE I-O.

PARAM	WRITE	READ
R4	1	0
A5	I-O DESCRIPTOR	I-O DESCRIPTOR
R6	CHANNEL SKIP IN [CF] LINE SKIP IN [FF]	0
R7	TIMEOUT IN SIXTIETHS OF SECONDS	TIMEOUT IN SIXTIETHS OF SECONDS
A8	POINTER TO FILE TANK	POINTER TO FILE TANK
R9	RECORD SIZE IN [CF] SUPPRESS TRANSFER IF [2:1]	RECORD SIZE

THE RESULT IS RETURNED IN R9, WITH INTERPRETATION AS FOLLOWS:

VALUE	INTERPRETATION
0	EOT-EOF BRANCH IF POSSIBLE
1	NO ERROR
2	PAR BRANCH IF POSSIBLE

37 CHAIN STATEMENT IN ALGOL AND FORTRAN, PERFORM AFTER END IN COBOL.

THIS COMMUNICATE IS THE MCP PORTION OF THE CHAIN STATEMENT WHICH ALLOWS A PROGRAM TO SPECIFY THE NAME OF A PROGRAM TO BE EXECUTED AFTER THE CURRENTLY-EXECUTING PROGRAM TERMINATES.

R4 CONTAINS THE SUFFIX OF THE NAME OF THE PROGRAM AND R5 CONTAINS THE PREFIX.

A SEGMENT OF ESPDISK IS GOTTEN, THE PREFIX AND SUFFIX AND TERMINAL UNIT/BUFFER NUMBER OF THE PROGRAM ARE WRITTEN INTO THE SEGMENT, AND THE SEGMENT NUMBER IS PLACED IN JAR[PIMIX, 9].[FF].

AT COMS TIME, THAT ELEMENT OF THE JAR IS CHECKED, AND, IF THE JOB TERMINATED NORMALLY, THE JOB SPECIFIED BY THE CHAIN STATEMENT IS EXECUTED.

38 RETURN MAIN AND AUXILIARY STORAGE FOR CODE OR DATA SEGMENT.

THIS COMMUNICATE ALLOWS THE PROGRAMMER TO SPECIFY THAT AN ARRAY OR PROGRAM SEGMENT IS TO BE DELETED FROM MAIN AND AUXILIARY MEMORY, JUST AS IF IT HAD NEVER BEEN REFERENCED.

A4 POINTS TO THE SEGMENT TO BE RETURNED.

THE AIT IS ALSO UPDATED.

39 ARRAY RETURN FOR BASIC.

THIS COMMUNICATE IS USED BY THE BASIC INTRINSICS TO RETURN ARRAY STORAGE.

N4 POINTS TO THE ARRAY DESCRIPTOR AND R5 CONTAINS THE NUMBER OF DIMENSIONS.

THE AIT IS ALSO UPDATED.

40 ON LINE MAINTENANCE INTERFACE.

THIS COMMUNICATE INTERFACES THE ON LINE MAINTENANCE PROGRAM

WITH THE MCP IN CERTAIN AREAS.

IF R5 = 0, R5 IS SET TO USERSTA[P1MIX]; OTHERWISE DKBUSINESS IS CALLED INDEPENDENTLY, THE COMMUNICATE WAITS FOR IT TO FINISH, THEN RETURNS CERTAIN INFORMATION IN R5, RELATED TO THE PERFORMANCE OF DKBUSINESS.

41 IOREQUEST.

THIS COMMUNICATE ALLOWS A PROGRAM TO CALL IOREQUEST DIRECTLY.

THE ENTIRE ACTION OF THE COMMUNICATE IS AS FOLLOWS:

IOREQUEST(R7,R6,FLAG(R5))

AND THEN TO RETURN TO THE CALLING PROGRAM.

COMPLEXSNOOZE

THE PROCEDURE DECLARATION FOR COMPLEXSNOOZE IS:

```
SAVE PROCEDURE COMPLEXSNOOZE(PRI,CODE);
     VALUE PRI; REAL PRI,CODE;
     BEGIN SNOOZE(PRI,1,P(,CODE,LOD));END
```

THE PARAMETERS ARE INTERPRETED AS FOLLOWS:

PRI	PRIORITY WITH WHICH PROCESS IS TO BE PLACED INTO THE BED.
CODE	EXPRESSION TO BE EVALUATED WHICH WILL DETERMINE WHEN THE CALLING PROCESS MAY BE REACTIVATED BY RETURNING A "1" OR WHEN IT SHOULD REMAIN SUSPENDED BY RETURNING A "0".

ESPOL COMPILES THE NAME-CALL EXPRESSION'S CODE INTO AN ACCIDENTAL ENTRY AT EACH CALL ON COMPLEXSNOOZE. COMPLEXSNOOZE THEN PASSES THIS SPECIAL DESCRIPTOR TO SNOOZE, WHICH PLACES IT IN THE BED AS THE MASK, AND A "1" IN THE ADDRESS FIELD. SPECIAL CODE IN THE OUTER BLOCK OF THE MCP CAUSES THE EXPRESSION TO BE REEVALUATED EACH TIME THE PROCESS IS A CANDIDATE FOR AWAKENING.

COM11

THE ALGOL READ/WRITE ENTRY POINT TO CONTROL STATE. MCP SEQUENCE NO. 14623000.

STACK:

```
+12
+11
+10 FIB
+ 9 FPB
+ 8 I
+ 7 LOC
+ 6 INFO
+ 5 A
+ 4 S
+ 3 F
+ 2 T
+ 1 B
  0 RCW
- 1 MSCW
- 2 IRCW
- 3 ICW
- 4 CODE
      0 = OPEN          6 = CLOSE
      1 = PARITY        7 = RDATA
      2 = EOF           8 = SELERR
      3 = EOT           9 = SPACE
      4 = DISKSPACE    10 = REFILL
      5 = SEEKDC
- 5 TANK OR PHYL OR HEADER OR TANG
- 6 ROW OR STA OR FINAL
- 7 RESULT OR TIMEOUT

                    WRITE     READ
                    -----     ----
      RESULT 0 =    ----      NOINPUT
             1 =    OK
             2 =    BREAK     ABNORM

- 8
- 9
```

- COM13, COM19, COM23, COM5 -

COM13

THE COBOL READ/WRITE ENTRY POINT TO CONTROL STATE.

COM19

THE BUSINESS END OF PRNPBT/DISK. COM19 PRINTS OR PUNCHES FILES FROM DISK AND TAPE.

COM23

THE BUSINESS END OF THE PROGRAM LDCNTRL/DISK. COM23 PLACES DECKS IN THE PSEUDO-READERS.

COM5

REMOVES A PROGRAM FROM THE MIX. CALLED FOR NORMAL EOJ AND ALL ERROR CONDITIONS. RETURNS STORAGE, CLOSES FILES, ETC. IN ADDITION, STARTS "GO" PART OF COMPILE AND GO JOBS.

* CORE, COUNTARRY, CTABLE *

CORE

USED BY SELECTRUN TO DETERMINE IF A JOB SHOULD BE INTRODUCED INTO THE MIX.

[4:14] MULTIPROCESSING FACTOR (x100)
[18:15] SUM OF CORE ESTIMATES FOR ALL JOBS NOW ACTIVE IN THE MIX (DIV 64)
[33:15] AMOUNT OF CORE MEMORY INITIALLY AVAILABLE FOR PROCESSING NORMAL STATE JOBS (DIV 64).

COUNTARRY

DESCRIPTOR POINTING TO COUNTARRAY ARRAY WHICH CONTAINS STATISTICS COUNTERS BY MIX INDEX.

CTABLE

DESCRIPTOR POINTING TO CTABLE ARRAY.

DALOC

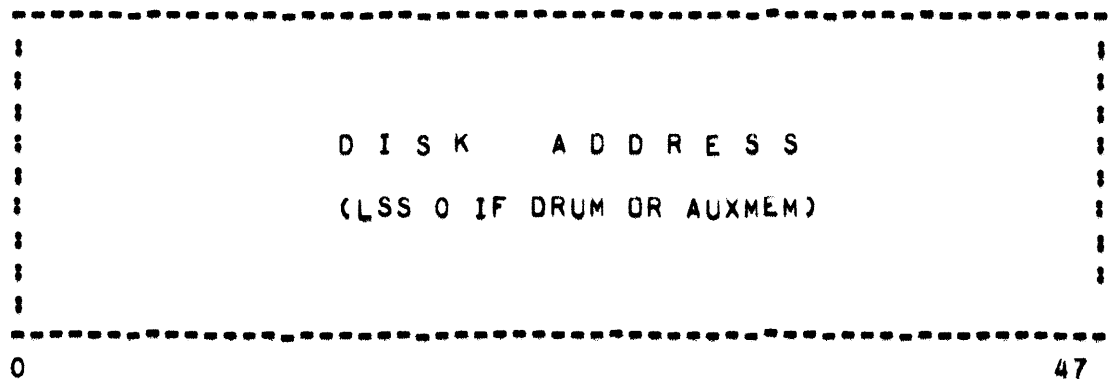
DALOC IS A TWO DIMENSIONAL ARRAY USED TO MANAGE OVERLAY STORAGE BOTH ON DRUMS AND DISK. IT HAS MIXMAX + 1 ROWS, EACH OF WHICH IS INITIALLY NINE WORDS LONG, AND IS EXPANDABLE AS REQUIRED.

EACH MIX HAS ONE DALOC ROW TO KEEP TRACK OF ITS BACKUP DISK. THE FIRST TABLE ENTRY BEGINS IN WORD 1. EACH TWO-WORD ENTRY KEEPS TRACK OF A 500 SEGMENT CHUNK WHICH IS DIVIDED INTO SUBSECTIONS OF 100 SEGMENTS EACH. DALOC INX P1MIX POINTS INTO THE APPROPRIATE DALOC ROW FOR P1MIX.

WORD 0

WORD	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.

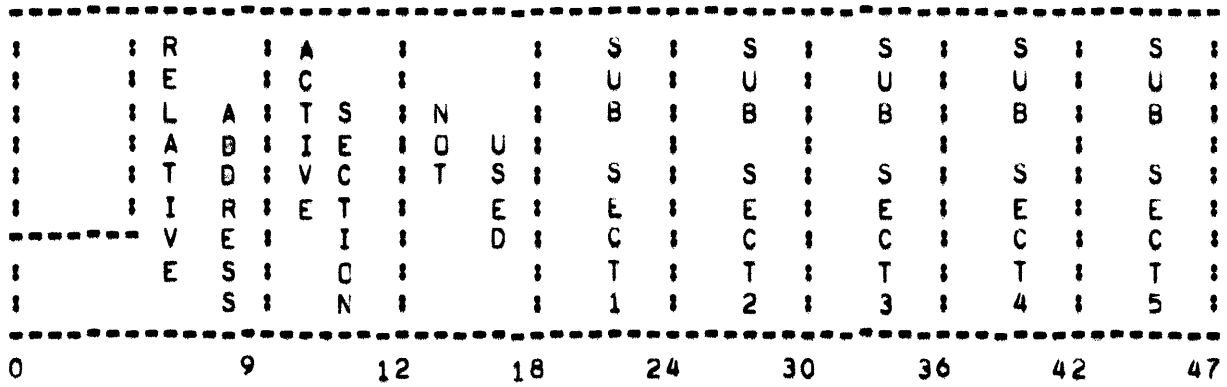
WORD 1



WORD 1

CONTAINS THE DISK ADDRESS OF THE BASE OF A 500 SEGMENT SECTION OF OVERLAY STORAGE WHICH IS MADE AVAILABLE AT SELECTRUN 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 INDCATED BY THE [9:3] FIELD.
[9:3]	INDCATES WHICH OF THE FOLLOWING SUB-SECTIONS (100 SEGMENTS EACH) IS ACTIVE (0-4, 5 INDICATES ALL SUB-SECTIUNS ARE FULL).
[18:6]	SUB-SECTION NO. 1
[24:6]	SUB-SECTION NO. 2

[30:6] SUB-SECTION NO. 3
[36:6] SUB-SECTION NO. 4
[42:6] SUB-SECTION NO. 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 "DESC" IS DEFINED AS THE DESCRIPTOR, THEN THE DISK ADDRESS TO WHICH THE DATA MUST BE RE-OVERLAID IS CALCULATED AS:

```
DALOC[⍎1MIX,DESC.[33:6]×2-1]+DESC.[39:9]
```

DATE

CONTAINS CURRENT DATE (YYDDD IN BCL)

DBADR

VARIABLE USED TO CONTAIN THE DISK ADDRESS FOR THE DB FEATURE OF THE DEBUGGING MODULE.

DBARRAY

DESCRIPTOR POINTING TO THE DBARRAY, THE ARRAY USED WITH THE DB FEATURE OF THE DEBUGGING MODULE

DCQARA

DESCRIPTOR POINTING TO THE DCQARA ARRAY

DCQPTSTACK

POINTS TO CURRENT STACK OF DCQPT ROUTINE WHICH HANDLES SOME B487 CONDITIONS.

DC19Q

DC19Q HOLDS INFORMATION NEEDED BY THE MCP FOR FILE TYPE 19 DATACOM I/O AND RJE WHEN USED WITH THE B487 HARDWARE. THE VARIABLE DC19Q IS THE HEAD OF THE QUEUE WHICH IS FORWARD LINKED THROUGH THE "C" FIELDS ([33:15]) AND BACKWARD LINKED THROUGH THE "F" FIELDS ([18:15]) OF THE FIRST WORD IN EACH ENTRY. IF DC19Q IS POSITIVE, THE QUEUE IS INTERLOCKED WHILE A NEW ENTRY IS BEING ADDED. THE FORMAT OF EACH ENTRY IS:

WORD	FIELD	CONTENTS
0		QUEUE WORD
	[4:5]	MIX
	[9:9]	TU,STA
	[18:15]	LINK TO LAST ENTRY
	[33:15]	LINK TO NEXT ENTRY
1		READ QUEUE LINK
	[1:1]	TANKING IN PROCESS
	[8:10]	I=O BUFFER SIZE
	[18:15]	LAST ENTRY IN THE QUEUE (TANKED INPUT GOES HERE IF [1:1] = 0)
	[33:15]	FIRST ENTRY IN THE QUEUE (LOOKS HERE WHEN PROGRAM REQUESTS READ)
2		OUTPUT RESULT DESCRIPTOR 0 = GOOD 1 = BREAK 2 = EOT
3		BUFFER COUNT (IF = 0 THEN WAIT UNTIL GTR 0)
4		TOTAL NUMBER OF BUFFERS ASSIGNED (=4 FOR RJE LINES)

READ QUEUE ENTRY: (NOTE: SEE WORD 1 ABOVE):

WORD	FIELD	CONTENTS
0		(SECOND MEMORY LINK) IS QUEUE WORD
	[1:1]	INPUT MESSAGE COMPLETED BIT
	[2:1]	PARITY BRANCH FLAG
	[3:1]	END BRANCH FLAG
	[8:10]	NUMBER OF WORDS IN MESSGE
	[18:15]	LINK TO LAST ENTRY
	[33:15]	LINK TO NEXT ENTRY

- DC19Q -

1		PLACE HOLDER
	[15:15]	COUNT OF WORDS LEFT IN THIS AREA (IF THIS WERE TO GO NEGATIVE, DISASTER WOULD OCCUR, THE ERROR FLAG WOULD BE SET)
	[30:18]	STREAM ADDRESS OF WHERE TO MOVE NEXT B487 BUFFER CHUNK
2		FIRST WORD OF MESSAGE AREA

NEW ENTRIES ARE PLACED INTO DC19Q BY NINTEENREADER AND COM36 ON INPUT AND STATIONMESSAGEWRITER ON OUTPUT. ENTRIES ARE REMOVED BY COM36 WHEN USED BY A PROGRAM AND BY KEYIN ROUTINES WHEN "BK" IS ENTERED FOR A GIVEN MIX INDEX.

DEBUGGING FACILITIES

GENERAL.

WHEN THE DCMCP IS COMPILED WITH DEBUGGING = TRUE, THE FOLLOWING FALICITIES ARE PROVIDED.

A. MEMORY DUMP UNDER MCP CONTROL (ALSO AVAILABLE WITH DUMP= TRUE).
B. SPO ACCESS TO CORE MEMORY.
C. SPO ACCESS TO DISK.
D. TRACE.

ANY OF THESE FEATURES MAY BE INITIATED PROGRAMMATICALLY (IN CONTROL STATE ONLY), AND ALL EXCEPT TRACE MAY BE INITIATED THROUGH SPO INPUT MESSAGES. TRACED OUTPUT IS GENERATED THROUGH A SPO INPUT MESSAGE.

MEMORY DUMP.

A MEMORY DUMP TO PRINTER MAY BE INITIATED BY ENTERING A "DPLP" THROUGH THE KEYBOARD, OR BY CALLING THE PROCEDURE PRINTCORE FROM WITHIN THE MCP. THIS PROCEDURE HALTS ALL NORMAL STATE PROCESSING, AND PRINTS THE CONTENTS OF ALL ASSIGNED MEMORY AREAS ON A PRINTER. THE FORMAT OF THE PRINT-OUT IS SUCH THAT A DOUBLE SPACE APPEARS BETWEEN AREAS. FOR AVAILABLE AREAS, ONLY THE LINKS ARE PRINTED. AT THE COMPLETION OF THE DUMP, NORMAL PROCESSING IS RESUMED. A SIMILAR DUMP TO TAPE MAY BE PERFORMED BY ENTERING DPMT OR BY CALLING DUMPCORE FROM WITHIN THE MCP.

SPO ACCESS TO MEMORY.

THIS FEATURE, DDT, MAY BE INITIATED BY ENTERING DD THROUGH THE SPO, OR BY EXECUTING THE FOLLOWING STATEMENT IN THE MCP:

```
POLISH(DT,DEL)
```

FURTHER, ONE OF THE ABILITIES OF THIS FEATURE IS TO INSERT A CALL ON THE DDT ROUTINE INTO A CODE STRING.

DEBUGGING STATEMENTS

THE DDT ROUTINE FIRST TYPES A MESSAGE ON THE SPO, CONSISTING OF "C:L = FOLLOWED BY THE ADDRESS OF SYLLABLE FROM WHICH IT WAS CALLED. THEN THE KEYBOARD IS READIED AND THE OPERATOR MAY ENTER DEBUGGING STATEMENTS. THE SYNTAX OF THE DEBUGGING STATEMENT IS AS FOLLOWS:

```
<DEBUGGING STATEMENT> ::= <MODE PART> <STATEMENT> <END OF INPUT>
<STATEMENT> ::= <CALL STATEMENT> / <DELETE CALL STATEMENT> /
     <PRINT SYMBOLS STATEMENT> / <DISPLAY STATEMENT> /
     <REPLACE STATEMENT> / <DEFINE STATEMENT> / <EXIT>
<EXIT> ::= ;
<MODE PART> ::= <EMPTY> / ≥ <MODE>
<MODE> ::= A / D / O
<CALL STATEMENT> ::= $ <EXPRESSION>
<DELETE CALL STATEMENT> ::= @ <ADDRESS>
<ADDRESS> ::= <EXPRESSION> / <EXPRESSION> : <DIGIT>
<PRINT SYMBOLS STATEMENT> ::= % <CODE>
<CODE> ::= <DECIMAL DIGIT>
<DISPLAY STATEMENT> ::= <EXPRESSION> <SIZE>
<SIZE> ::= <EMPTY> / ; <EXPRESSION>
<REPLACE STATEMENT> ::= <EXPRESSION> = <VALUE>
<VALUE> ::= <OCTAL NUMBER> / . <DECIMAL NUMBER> / <ALPHA STRING>
<DEFINE STATEMENT> ::= <EXPRESSION> : <CODE> <SYMBOL>
<EXPRESSION> ::= <GENERAL EXPRESSION> / <OPERATOR> <GENERAL
     EXPRESSION>
<GENERAL EXPRESSION> ::= <PRIMARY> / <PRIMARY> <OPERATOR>
     <GENERAL EXPRESSION>
<PRIMARY> ::= / / = / * / <SYMBOL> / <OCTAL NUMBER> /
     <DECIMAL NUMBER> / [<EXPRESSION>]
<OPERATOR> ::= + / -
<OCTAL NUMBER> ::= <OCTAL DIGIT> / <OCTAL NUMBER> <OCTAL DIGIT>
<OCTAL DIGIT> ::= 0 / 1 / 2 / 3 / 4 / 5 / 6 / 7
<DECIMAL NUMBER> ::= <DECIMAL DIGIT> / <DECIMAL NUMBER> <DECIMAL DIGIT>
<DECIMAL DIGIT> ::= <OCTAL DIGIT> / 8 / 9
<SYMBOL> ::= <ALPHA STRING>
<ALPHA STRING> ::= <LETTER> / <ALPHA STRING> <LETTER>
<LETTER> ::= A / B / C ... X / Y / Z
```

THE SEMANTICS OF THE VARIOUS DEBUGGING STATEMENTS ARE AS FOLLOWS:

DEBUGGING STATEMENT.

A DEBUGGING STATEMENT CONSISTS OF A STATEMENT, OPTIONALLY PRECEDED BY A MODE PART, AND TERMINATED BY END OF INPUT.

MODE PART.

THE OUTPUT OF THE DDT FEATURE IS <ADDRESS> = , FOLLOWED BY THE CONTENT OF THE WORD AT THE MEMORY LOCATION GIVEN BY ADDRESS. THIS WORD WILL BE TYPED IN OCTAL, ALPHANUMERIC, OR DECIMAL, DEPENDING ON THE MODE SWITCH. THIS MODE SWITCH IS SET TO OCTAL, INITIALLY, BUT MAY BE CHANGED BY ENTERING A MODE PART. THE MODES ARE OBVIOUS.

CALL STATEMENT.

A CALL STATEMENT CAUSES A CALL ON THE DDT ROUTINE TO BE INSERTED INTO THE CODE STRING IN PLACE OF THE FIRST LITC SYLLABLE LOCATED AFTER THE ADDRESS GIVEN IN THE STATEMENT. THE VALUE OF THE LITC SYLLABLE IS SAVED, AND RETURNED TO THE ROUTINE SO THAT ITS LOGIC WILL NOT BE DESTROYED. A MESSAGE IS TYPED GIVING THE WORD AND SYLLABLE OF THE INSERTED CALL SO THAT IT MAY BE LOCATED LATER.

DELETE CALL STATEMENT.

THE DELETE CALL STATEMENT REMOVES FROM THE CODE STRING, AT THE SPECIFIED ADDRESS, A CALL ON THE DDT ROUTINE, AD REPLACES IT WITH LITC SYLLABLE PREVIOUSLY THERE.

DISPLAY STATEMENT.

THE DISPLAY STATEMENT CAUSES A WORD (OR WORDS) IN CORE MEMORY TO BE TYPED ON THE SPO. THE ADDRESS OF THE WORD (OR FIRST WORD) IS GIVEN, AND, IF MORE THAN ONE WORD IS DESIRED, THE NUMBE OF WORDS TO BE TYPED IS SPECIFIED, PRECEDED BY A SEMICOLON.

REPLACE STATEMENT.

THE REPLACE STATEMENT CAUSES THE VALUE TO BE THE RIGHT OF = TO BE STORED INTO THE ADDRESS GIVEN BY THE EXPRESSION.

DEFINE STATEMENT.

THE DEFINE STATEMENT CAUSES THE SYMBOL ON THE RIGHT TO BE ASSIGNED THE SPECIFIED CODE AND TO HAVE THE VALUE OF THE EXPRESSION ON THE LEFT.

PRINT SYMBOLS STATEMENT.

THE PRINT SYMBOLS STATEMENT CAUSES, FOR ALL SYMBOLS WITH THE SPECIFIED CODE, A LINE TO BE TYPED CONTAINING THE SYMBOL AND THE CONTENTS OF THE WORD ADDRESSED BY THE PREVIOUSLY DEFINED VALUE OF THAT SYMBOL. SOME SYMBOLS ARE KNOWN WITHOUT DEFINITION. F IS DEFINED AS THE CONTENTS OF THE F-REGISTER JUST PRIOR TO ENTERING THE DDT PROCEDURE, AND THE UNIT MNEMONIC OF EACH MAGNETIC TAPE UNIT IS DEFINED AS THE ADDRESS OF THE LABELTABLE ENTRY FOR THE CORRESPONDING UNIT.

EXIT.

A SEMICOLON CAUSES THE DDT ROUTINE TO EXIT AND RETURN TO HIS CALLER.

EXPRESSION.

AN EXPRESSION CONSISTS OF ONE OR MORE PRIMARIES SEPARATED BY THE OPERATORS + AND -. THESE OPERATORS HAVE THEIR NORMAL MEANINGS OF ARITHMETIC ADDITION AND SUBTRACTION, RESPECTIVELY.

AN EXPRESSION OF THE FORM <OPERATOR> <GENERAL EXPRESSION> IS EQUIVALENT TO THE GENERAL EXPRESSION *<OPERATOR> <GENERAL EXPRESSION>.

PRIMARIES.

THE VARIOUS PRIMARIES ARE:

/	THE VALUE OF THIS PRIMARY IS THE F FIELD ([18:15]) OF THE LAST WORD DISPLAYED.
=	THE VALUE OF THIS PRIMARY IS THE C FIELD ([33:15]) OF THE LAST WORD DISPLAYED.
*	THE VALUE OF THIS PRIMARY IS ADDRESS OF THE LAST WORD DISPLAYED.
<SYMBOL>	THE VALUE OF THIS PRIMARY IS THE VALUE PREVIOUSLY (OR INTRINSICALLY) DEFINED FOR THE SYMBOL.
<OCTAL NUMBER>	THE VALUE OF THESE PRIMARIES IS THE VALUE

<DECIMAL NUMBER> OF THE NUMBER.

[<EXPRESSION>] THE VALUE OF THIS EXPRESSION IS THE C FIELD OF THE WORD IN MEMORY ADDRESSED BY THE EXPRESSION WITHIN THE BRACKETS PAIR. THE PRIMARY = IS EXACTLY EQUIVALENT TO THE PRIMARY [*].

CONSOLE ACCESS TO DISK.

THE DISKBUG FEATURE MAY BE INITIATED BY ENTERING DB THROUGH THE KEYBOARD OR BY INVOKING THE PROCEDURE DISKBUG FROM WITHIN THE MCP.

UPON INVOCATION, THIS FEATURE IDENTIFIES ITSELF BY TYPING THE MESSAGE DISKBUG ON THE SPO, AND CALLS DDT TO ALLOW THE OPERATOR TO SET THE VARIABLE DBADR TO THE DISK ADDRESS OF THE SEGMENT TO WHICH ACCESS IS DESIRED. DISBUG WILL READ THE DESIRED SEGMENT INTO THE ARRAY DBARRAY, AND AGAIN CALL DDT. THE OPERATOR MAY THEN INSPECT DBARRAY, AND CHANGE DBADR. THIS PROCESS REPEATS UNTIL THE OPERATOR SETS DBADR TO ZERO. THE ADDRESSES OF DBADR AND DBARRAY MAY BE DETERMINED FROM A PRT LISTING OF PRT INDEX OF THE CURRENT MCP.

TRACE.

THE TRACE FEATURE CONTAINS TWO SECTIONS. THE FIRST STORES THE DATA TO BE TRACED IN CORE AND ON DISK, AND THE SECOND PRINTS THE STORED DATA.

STORING.

TO USE THE TRACE FEATURE, IT IS NECESSARY TO COMPILE INTO THE MCP CALLS ON THE PROCEDURE TRACE, PASSING THE DATA TO BE STORED, AND THE FORMAT IN WHICH THE DATA IS PRINTED. THE TRACE PROCEDURE ATTEMPTS TO DETERMINE THE SEGMENT AND RELATIVE ADDRESS OF THE CALL, AND STORES THE DATA, FORMAT, MIX INDEX, SEGMENT, AND RELATIVE ADDRESS, USING THE DISK AREA DEFINED IN THE MCP. THIS DISK AREA MUST ALSO BE DEFINED IN THE COLD/START DECK AS THE SAME ADDRESS AND SIZE AS IN THE MCP. IF TRACAREASIZE IS EXCEEDED, WRAPAROUND OCCURS WITHIN THE DISK FILE. IF SHOULD BE NOTED THAT TWO CONDITIONS EXIST WHICH MAY CAUSE ERRORS IN THE TRACE. IF THE TRACE CALL APPEARS IN AN INDEPENDENT ROUTINE OR IN THE OUTER BLOCK OF THE MCP, THE SEGMENT AND ADDRESS CANNOT BE PROPERLY DETERMINED. THE SECOND CONDITION IS THAT ALTHOUGH THE DISK STORAGE IS BLOCKED, (DATA IS STORED IN CORE UNTIL A BLOCK IS ACCUMULATED, AND THEN IS WRITTEN ONTO DISK) AND BUFFERED (TWO AREAS OF CORE ARE USED FOR ACCUMULATING DATA TO BE WRITTEN), NO CHECK IS MADE THAT THE DISK WRITE HAS BEEN COMPLETED FROM A BUFFER BEFORE REUSING IT. THIS PROBLEM MAY BE EASED BY

EITHER CHANGING THE DEFINED TRACESIZE WHICH IS THE SIZE OF THE TRACE BUFFERS, OR BY ENSURING THAT TRACE CALLS DO NOT OCCUR WITH SUFFICIENT FREQUENCY TO DESTROY PREVIOUS DATA.

PRINTING THE TRACED DATA.

THE ROUTINE TO PRINT THE ACCUMULATED TRACE DATA IS INITIATED BY ENTERING PT THROUGH THE KEYBOARD. THIS CAUSES THE DATA TO BE PRINTED ON LPA, IN LAST-IN, FIRST-OUT ORDER, FOLLOWED BY A MEMORY DUMP. DISKBUG IS THEN INITIATED.

THE TRACE OUTPUT CONTAINS ONE ITEM PER LINE, CONSISTING OF THE PRT ADDRESS OF THE PROCEDURE CALLING TRACE, THE RELATIVE ADDRESS WITHIN THAT PROCEDURE OF THE TRACE CALL, THE MIX INDEX, AND THE DATA BEING TRACED IN THE FORMAT SPECIFIED BY THE TRACE CALL. THE POSSIBLE FORMATS AE (WITH THEIR CORRESPONDING CODES):

0 HALF-WORD OCTAL: THE DATA IS FORMATTED AS TWO 8-DIGIT OCTAL NUMBERS REPRESENTING THE TWO HALVES OF THE DATA WORD.

1 ALPHA: THE DATA IS FORMATTED AS EIGHT BCL CHARACTERS.

2 DECIMAL: THE DATA IS FORMATTED AS AN 8-DIGIT DECIMAL INTEGER.

3 OCTAL: THE DATA IS FORMATTED AS THREE OCTAL NUMBERS, ONE OF SIX DIGITS, AND TWO OF FIVE DIGITS, REPRESENTING THE FIELDS [0:18],[18:15] AND [33:15] OF THE DATA WORD.

PRINTING OF THE TRACED DATA NOT OCCUR AT THE SAME TIME THAT THE TRACE IS BEING PERFORMED, THE OPERATOR MUST NOT ENTER "PT" TO PRINT THE TRACE WHILE THE POSSIBILITY EXISTS THAT A TRACE WILL BE PERFORMED.

DECKREMOVER

DECKREMOVER(B) IS A PROCEDURE WHICH IS CALLED BY "KEYIN" TO REMOVE PSEUDO-DECKS USING "REMOVEDECK".

DIRECT

DESCRIPTOR POINTING TO DIRECT ARRAY, WHICH CONTAINS THE RESERVED WORDS ACCEPTED BY THE CONTROLCARD ROUTINE ON CONTROL CARDS.

DIRECTORY - BYPASS

THE BYPASS DIRECTORY IS A MECHANISM TO REDUCE THE AMOUNT OF TIME REQUIRED TO OBTAIN THE DETAIL HEADER RECORD OF A FILE FROM THE MAIN DIRECTORY. ON NON-SHAREDISK SYSTEMS, THE BYPASS DIRECTORY IS BUILT DURING EACH HALT/LOAD. ON SHAREDISK SYSTEMS THE BYPASS DIRECTORY IS BUILT DURING THE HALT/LOAD WHEN A SEARCH OF DISK SEGMENT ZERO REVEALS THAT NO OTHER SYSTEMS HAVE INTRINSIC FILES SPECIFIED.

EACH LOGICAL BLOCK IN THIS DIRECTORY IS 60 WORDS (2 SEGMENTS) LONG AND CAN CONTAIN ENTRIES FOR UP TO 20 FILES. THE FORMAT OF AN ENTRY IS:

BLOCK[I]	EITHER <MFID> OF THE FILE OR AN @14 WHICH INDCATES THAT THE POSITION IS AVAILABLE FOR A NEW ENTRY.
BLOCK[I+1]	<FID>
BLOCK[I+2].[33:15]	ADDRESS OF THE HEADER RECORD IN THE MAIN DIRECTORY.

ONE ADDITIONAL FIELD IS USED IN THE 2ND WORD OF EACH PHYSICAL BLOCK.

BLOCK[2].[18:15]	ADDRESS OF THE NEXT PHYSICAL BLOCK IN THE BYPASS DIRECTORY WHICH IS AN ENTENSION OF THE ENTRIES IN THIS BLOCK.

THE MCP USES A SCRAMBLE TECHNIQUE WHICH USES THE <MFID> AND <FID> OF THE FILE NAME TO ACCESS ONE OF THE 60 WORD BLOCKS. ONCE THIS BLOCK IS OBTAINED, A SEQUENTIAL SEARCH IS MADE TO DETERMINE IF THE REQUIRED <MFID> AND <FID> IS PRESENT. IF THE NAMES ARE NOT PRESENT, THE "F" FIELD OF THE 2ND WORD OF THE BLOCK IS EXAMINED, AND IF IT IS NOT ZERO, THE NEXT "CHAINED" 60 WORD BLOCK IS BROUGHT IN AND THE PROCESS IS REPEATED.

THE "CHAIN" IS NECESSARY AS THE SCRAMBLE MAY PRODUCE A GREAT NUMBER OF "HITS" ON THE SAME BLOCK IN THE BYPASS DIRECTORY.

THE BYPASS DIRECTORY IS INITIALIZED BY SETTING THE <MFID> OF ALL ENTRIES TO @14. THE END OF THE BYPASS DIRECTORY IS POINTED TO BY THE MCP VARIABLE BYPASS.[33:15]. AS A 60 WORD AREA IS FILLED, AN ADDITIONAL 60 WORD AREA IS OBTAINED ON THE END OF THE BYPASS DIRECTORY, BYPASS.[33:15] IS DECRIMENTED, AND A CHAIN IS ESTABLISHED IN BLOCK[2].[18:15].

ENTRIES ARE REMOVED FROM THE BYPASS DIRECTORY BY PLACING AN @14 IN THE <MFID> OF AN ENTRY. NO CONSOLIDATION OF "OVERFLOW" AREAS WHICH HAVE RESULTED FROM MULTIPLE "HITS" IS ATTEMPTED.

- DIRECTORY - BYPASS -

THE BYPASS DIRECTORY IS LOCATED STARTING AT "DIRECT - 2" AND GROWS TOWARDS THE MAIN DIRECTORY. IT IS THEREFORE OF THE UPMOST IMPORTANCE TO DECLARE SUFFICIENT SPACE FOR BOTH THE MAIN AND BYPASS DIRECTORIES AND TO HAVE AN ACCURATE FILE DESCRIPTION IN THE COLD START DECK TO FACILITATE ANALYSIS.

BYPASS.[18:15] POINTS AT THE END OF THE MAIN DIRECTORY. IT IS POSSIBLE FOR THE BYPASS DIRECTORY AND MAIN DIRECTORY TO GROW TOWARDS EACH OTHER SO THAT AN OVERLAP CONDITION EXISTS (BYPASS.[18:15] = BYPASS.[33:15]). IF THIS OCCURS, THE MESSAGE "DISK TOO CHECKERBOARDED" WILL BE OUTPUT ON THE SPO AND A HALT/LOAD MUST BE DONE. IF A HALT/LOAD IS IMPOSSIBLE DUE TO THE SIZE OF THE DIRECTORIES, A FALSE "END OF DIRECTORY" MARK CAN BE WRITTEN IN THE MAIN DIRECTORY LOCALLY THROUGH AN I/O CHANNEL AND THE USER MAY BE ABLE TO DUMP MOST OF HIS FILES. THIS PROCEDURE IS RECOMMENDED ONLY WHEN ALL ELSE FAILS.

DIRECTORY - FILE HEADER

```
H[0],[0:15]       RECORD LENGTH
    ,[15:15]      BLOCK LENGTH
    ,[30:12]      RECORD/BLOCK
    ,[42:6]       SEGMENTS/BLOCK
H[1],[6:18]       CREATION DATE FOR LOGGING (WHEN ON DISK)
    ,[25:23]      CREATION TIME FOR LOGGING (WHEN ON DISK)
    ,[1:47]       NUMBER OF LOGICAL RECORDS PER ROW (WHEN IN CORE)
H[2],[0:48]       =0  FREE FILE
    ,[1:1]        =0  SOLE USER, PUBLIC OR PRIVATE FILE
    ,[1:1]        =1  SECURITY FILE
    ,[6:42]       PRIMARY USER-S CODE
H[3],[1:1]        =1  NEW FILE HEADER FORMAT
    ,[2:10]       SAVE FACTOR (BINARY)
    ,[12:18]      DATE OF LAST ACCESS (BINARY)
    ,[30:18]      CREATION DATE (BINARY)
H[4],[1:1]        =1  FILE IS BEING LOADED OR NAME IS BEING CHANGED
    ,[2:1]        =1  FILE IS OPENED BY AN EXCLUSIVE USER
    ,[3:1]        =1  A PROGRAM IS WAITING TO USE THE FILE
    ,[4:2]        SYSTEM NUMBER OF EXCLUSIVE USER
    ,[3:6]        ≠0  MIX INDEX OF LIBMAIN/DISK
    ,[6:1]        USED BY AUTOPRINT TO MARK A PBD FILE
    ,[7:1]        USED TO MARK PSEUDO DECKS THAT WERE CREATED ON
    ,[8:1]        MARKS AN MC-ED COMPILER
                  A TIME-SHARING SYSTEM BY A ZIP WITH FILE-ID
    ,[9:2]        =2  FILE IS DATA
                  =3  FILE IS PROGRAM
                  =0  DON"T KNOW IF DATA OR PROGRAM
    ,[11:1]       FILE ACCESSED BIT
    ,[12:4]       SYSTEM FILE TOGGLES
    ,[16:5]       OPEN COUNT 2 FOR SYSTEM 0 (A)
    ,[21:5]       OPEN COUNT 2 FOR SYSTEM 1 (B)
    ,[26:5]       OPEN COUNT 2 FOR SYSTEM 2 (C)
    ,[31:5]       OPEN COUNT 2 FOR SYSTEM 3 (D)
    ,[36:6]       =0  TYPE IS UNKNOWN
                  =1  BASIC
                  =2  ALGOL
                  =3  COBOL
                  =4  FORTRAN
                  =5  TSPOL
                  =6  XALGOL
                  =7  SEQ
                  =8  DATA
                  =9  LOCK
                  =10 INFO
```

- DIRECTORY - FILE HEADER -

```
    .[42:6]       NOT USED
H[5].[0:48]       =0  SOLE USER FILE
                  = 12 IF PUBLIC FILE
                  N/A IF FREE OR SECURITY FILE
                  =<MFID> IF A PBD FILE (SEE MCP 37046630)
                  =OUTPUT FILE NAME IF A SCHEDULE FILE
    .[1:1]        = 1 PRIVATE FILE
    .[6:42]       = <MFID> OF SECURITY FILE FOR A PRIVATE FILE
H[6]              = 12 IF INFO FILE
                  = 0 IF SOLE USER OR PUBLIC FILE
                  =  REEL  NUMBER  IF  PBD  FILE  (I.E.  001 FOR PBD/
                  0006001).
                  = "AFTER" TIME OF SCHEDULE TASK (TSS ONLY)
                  =  NOT  0  &  SCHEDNUM[CTC] IF SCHEDULE OUTPUT FILE
                  (TSS ONLY)
    .[1:1]        =  1 ON A FILENNN/SCHEDUL WHICH IS BEING TERMINATED
                  (TSS ONLY).
    .[6:42]       = <FID> OF SECURITY FILE IF PRIVATE FILE
    .[9:9]        =TU/BUF IF A DECK FROM RJE
    .[36:9]       =TU/BUF IF PUD OR PBD FROM RJE
    .[33:15]      = DISK ADDRESS OF NEXT CONTROL DECK
H[7]              NUMBER OF LOGICAL RECORDS (EOF POINTER)
H[8]              NUMBER OF SEGMENTS PER ROW
H[9].[1:1]        TOGGLE 1 FOR SYSTEM 0 (A)
    .[2:1]        TOGGLE 1 FOR SYSTEM 1 (B)
    .[3:1]        TOGGLE 1 FOR SYSTEM 2 (C)
    .[4:1]        TOGGLE 1 FOR SYSTEM 3 (D)
    .[5:1]        TOGGLE 2 FOR SYSTEM 0 (A)
    .[6:1]        TOGGLE 2 FOR SYSTEM 1 (B)
    .[7:1]        TOGGLE 2 FOR SYSTEM 2 (C)
    .[8:1]        TOGGLE 2 FOR SYSTEM 3 (D)
    .[9:5]        OPEN COUNT 1 FOR SYSTEM 0 (A)
    .[14:5]       OPEN COUNT 1 FOR SYSTEM 1 (B)
    .[19:5]       OPEN COUNT 1 FOR SYSTEM 2 (C)
    .[24:5]       OPEN COUNT 1 FOR SYSTEM 3 (D)
    .[29:14]      NOT USED
    .[43:5]       MAXIMUM NUMBER OF ROWS
H[10]-H[29]       DISK ADDRESSES OF ROWS (0 IF NOT ASSIGNED)
```

OPEN COUNTS AND TOGGLES

TOGGLE 1	TOGGLE 2	OPEN COUNT 1	OPEN COUNT 2
0	0	INPUT ONLY	INPUT
0	1(OUTPUT)	NOT USED	INPUT
1	0	SHARED	INPUT

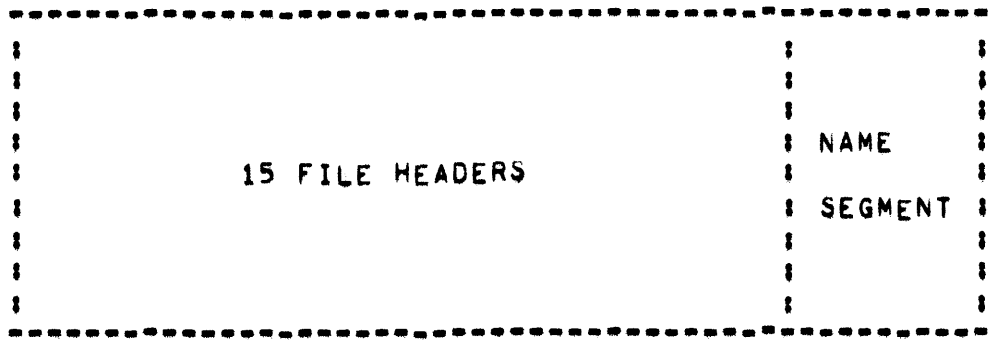
DIRECTORY - MAIN

THE "MCP" MAINTAINS A DIRECTORY ON DISK WHICH CONTAINS INFORMATION PERTAINING TO ALL PERMANENT FILES ON USER DISK. THE DIRECTORY CONSISTS OF TWO SECTIONS, THE MAIN DIRECTORY AND THE BYPASS DIRECTORY.

THE MAIN DIRECTORY STARTS AT DIRECTORYTOP + 4 AND CONTINUES TO THE VALUE SPECIFIED IN THE [18:15] FIELD OF THE MCP VARIABLE BYPASS. EACH LOGICAL SECTION OF THE MAIN DIRECTORY CONSISTS OF 16 SEGMENTS. THE FIRST 15 SEGMENTS CONTAIN DETAILED INFORMATION REGARDING FILES SPECIFIED IN THE 16TH SEGMENT. THE 16TH SEGMENT THUS CONTAINS THE <MFID> AND <FID> FOR ALL OF THE PRECEDING 15 FILES. THE MCP SCANS THE 16TH SEGMENT FROM BACK TO FRONT TO THAT S[28] AND S[29] ARE ACCESSED (AND FILLED) BEFORE S[26] AND S[27].

IF THE <MFID> OF AN ENTRY IS EQUAL TO @14 THEN THAT "SLOT" IN THE DIRECTORY IS AVAILABLE FOR INSERTION OF A NEW FILE. THE END OF THE MAIN DIRECTORY IS INDCATED BY A <MFID> EQUIVALENT TO @114. BYPASS. [18:15] WOULD THUS POINT AT THE NAME SEGMENT THAT CONTAINED THE @114.

A LINKING STRUCTURE EXISTS FOR AVAILABLE ENTRIES IN THE MAIN DIRECTORY. THE <FID> OF A "NAME SEGMENT" ENTRY THAT HAS A <MFID> OF @14 IS THE RELATIVE ADDRESS OF THE NEXT AVAILABLE ENTRY. THE RELATIVE ADDRESS OF THE MOST RECENT AVAILABLE ENTRY IS CONTAINED IN THE MCP VARIABLE "NEXTSLOT".

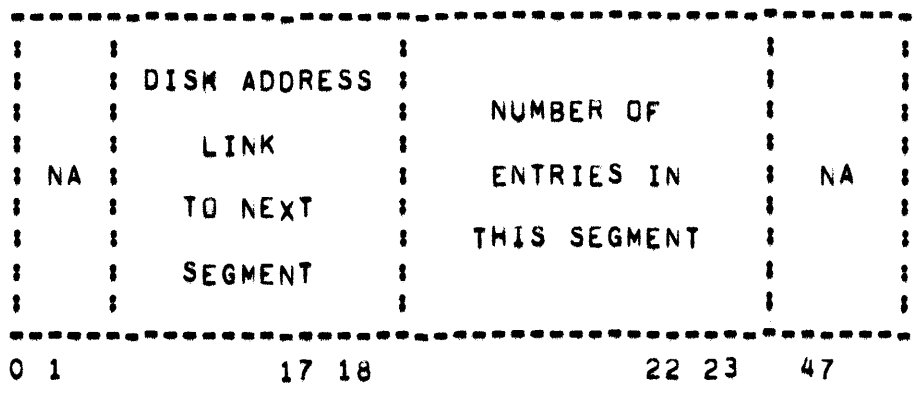


DIRECTORY - SCRATCH (SHAREDISK)

A SHARED DISK MCP MAINTAINS A RECORD OF CURRENT SCRATCH FILE ROWS. THIS RECORD IS REQUIRED TO RELEASE SCRATCH FILES BELONGING TO THE SYSTEM WHEN THE SYSTEM GOES DOWN. THERE ARE TWO TABLES CONTAINING THIS INFORMATION.

TABLE 1: DISK RESIDENT PORTION OF THE SCRATCH DIRECTORY. THIS TABLE CONSISTS OF 30 WORD SEGMENTS SCATTERED THROUGHOUT THE SYSTEM=S ESP DISK AND LINKED TOGETHER VIA A LINK PORTION OF THE ZEROTH WORD OF EACH SEGMENT. THE FIRST WORD (WORD 0) OF EACH SEGMENT CONTAINS A COUNT OF THE MEANINGFUL ENTRIES IN THE SEGMENT (1≤,≤29), AND A LINK TO THE NEXT SEGMENT. THE INITIAL LINK, I.E., THE LINK TO THE FIRST SEGMENT OF TABLE 1, IS STORED IN WORD # SYSNO (0,1,2, OR 3) IN A SEGMENT LOCATED AT DIRECTORYTOP+1, IN THE ,[1:17] FIELD (SYSNO IS THE SYSTEM NUMBER). WHEN A SHARED SYSTEM GOES DOWN, THE OPERATOR CAN, FROM ANOTHER UP=AND=RUNNING SYSTEM, "CLEAR" THE DOWN SYSTEM. THIS CLEARING WILL, AMONG OTHER ACTIONS, OBTAIN THIS INITIAL LINK AND PROCEED TO LINK FROM ONE SEGMENT TO ANOTHER, FREEING UP (RETURNING TO AVAILABLE STATUS) ALL OF THE SCRATCH DISK IN USE BY THE DOWN SYSTEM (SEE WORD I BELOW). THE LAST LINK IS ZERO. THE SEGMENT CONTENTS ARE:

WORD 0:

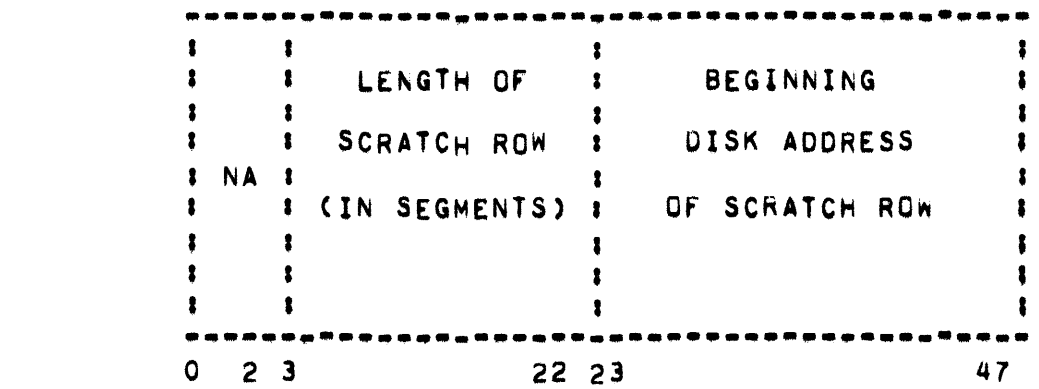


FIELD	FIELD VALUE	FUNCTION
[1:17]	DISK ADDRESS	PROVIDES LINK TO NEXT SEGMENT OR ZERO IF NO NEXT SEGMENT.
[18:5]	NUMBER ENTRIES (1≤,≤29)	NUMBER OF ENTRIES IN THE SEGMENT.
[23:25]	NA	

THIS WORD IS NOT ACTUALLY USED BY THE SYSTEM; HOWEVER IT IS MAINTAINED BY THE SYSTEM TO ALLOW ANOTHER SYSTEM TO LINK THROUGH THE (DOWN) SYSTEM-S TABLE 1 SCRATCH INFORMATION (SEE ABOVE).

WORD I:

(I=1,2,...,N WHERE N = [18:5] OF WORD 0).
EACH ENTRY (WORD) DESCRIBES ONE ROW OF A SCRATCH FILE.



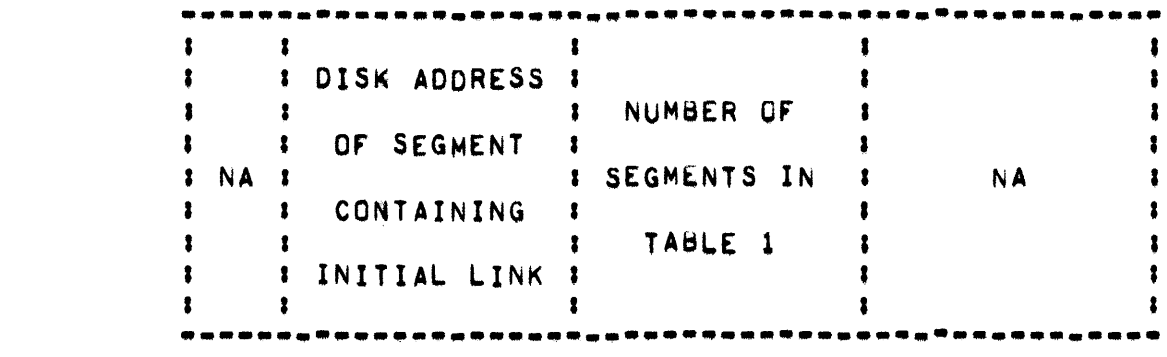
FIELD	FIELD VALUE	FUNCTION
[0:3]	NA	
[3:20]	NUMBER SEGMENTS	NUMBER OF SEGMENTS IN THE SCRATCH FILE ROW.
[23:25]	DISK ADDRESS	DISK ADDRESS OF FIRST SEGMENT OF THE SCRATCH FILE ROW.

EACH ENTRY (WORD) IS ADDRESS ([23:25]) KEYED. THE ENTRIES ARE UNORDERED WITHIN THE SEGMENT, BUT THE SEGMENTS ARE ORDERED BY MAXIMUM KEY: MAX (SEGMENT[J])<MIN(SEGMENT[J+1])

- DIRECTORY - SCRATCH (SHAREDISK) -

TABLE 2: CORE RESIDENT PORTION OF THE SCRATCH DIRECTORY. THIS TABLE IS USED TO PROVIDE RAPID ACCESS TO THE PROPER SEGMENT OF TABLE 1. IT IS CALLED THE "SCRATCHVEC".

WORD 0:

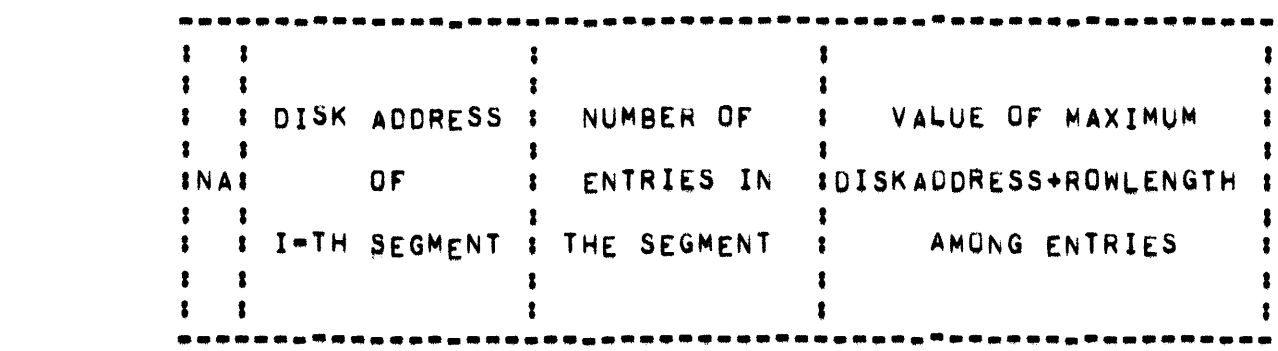


FIELD	FIELD VALUE	FUNCTION
[1:17]	DISK ADDRESS	DISK ADDRESS OF THAT SEGMENT WHOSE "SYSNO" WORD CONTAINS THE DISK ADDRESS LINK TO THE FIRST SEGMENT OF TABLE 1 (SEE TABLE 1 DESCRIPTION ABOVE).
[18:10]	NUMBER SEGMENTS	NUMBER OF SEGMENTS IN TABLE 1.

- DIRECTORY - SCRATCH (SHAREDISK) -

WORD I: (I=1,2,3,...,M WHERE M IS THE NUMBER OF SEGMENTS IN TABLE 1. M IS (WORD 0).[18:10]).

THE I-TH WORD OF TABLE 2 CHARACTERIZES THE I-TH SEGMENT OF TABLE 1 BY STORING THE SEGMENT-S DISK ADDRESS, ITS NUMBER OF ENTRIES, AND THE VALUE OF THE HIGHEST DISKADDRESS+ROWLENGTH (.[3:20]+.[23:25]) CONTAINED AMONG THE ENTRIES OF THE SEGMENT. THE PROPER SEGMENT TO WHICH AN ENTRY IS TO BE ADDED OR DELETED IS FOUND BY LOCATING THE FIRST HIGHEST DISKADDRESS+ROWLENGTH GREATER THAN THE ENTRY-S DISK ADDRESS. THIS SEARCH IS DONE IN CORE AND REDUCES THE NUMBER OF DISK I/O S.



FIELD	FIELD VALUE	FUNCTION
[1:17]	DISK ADDRESS	DISK ADDRESS OF I-TH SEGMENT OF TABLE 1
[18:5]	NUMBER ENTRIES	NUMBER OF ENTRIES IN THE I-TH SEGMENT
[23:25]	DISK ADDRESS	VALUE OF THE HIGHEST DISKADDRESS+ROWLENGTH AMONG THE SEGMENT-S ENTRIES

USE OF THE SCRATCH DIRECTORY:

IF A SYSTEM STOPS RUNNING, ANOTHER SYSTEM OBTAINS THE INITIAL DISK ADDRESS LINK TO THE DOWN SYSTEM-S TABLE 1. IT THEN PROCEEDS SEQUENTIALLY FROM ONE SEGMENT TO ANOTHER, FREEING UP ALL OF THE DOWN SYSTEM-S SCRATCH FILE ROWS.

DIRECTORYFREE

VARIABLE USED TO INTERLOCK THE DIRECTORY.

DIRECTORYTOP

DIRECTORYTOP CONTAINS INFORMATION WHICH IS USED IN SYSTEM INITIALZATION DURING A HALT/LOAD OPERATION. DIRECTORYTOP IS ONE SEGMENT LONG AND HAS THE FOLLOWING FORMAT:

D[0]	OPTION WORD
D[1]	DATE (BCL)
D[2]	NUMBER OF ELECTRONIC UNITS (CONTENTS OF "ESU" CARD FROM COLD OR COOL START DECK).
D[4]	VALUE OF DIRECT (FROM COLD OR COOL START DECK).
D[8]	NEXT NUMBER AVAILABLE FOR PRINTER BACK UP DISK OR PUNCH BACK UP.
D[9]	CORE, THE MULTIPROCESSING FACTOR.
D[10]-D[15]	SPECIFY WHICH DATA COMMUNICATIONS STATIONS HAVE BEEN MADE REMOTE SUPERVISORY PRINTERS.

```
D[10].[0:16]  NOT USED
     .[16:16] TU1, BUFFERS 0-15
     .[32:16] TU2, BUFFERS 0-15
D[11].[0:16]  TU3, BUFFERS 0-15
     .[16:16] TU4, BUFFERS 0-15
     .[32:16] TU5, BUFFERS 0-15
D[12].[0:16]  TU6, BUFFERS 0-15
     .[16:16] TU7, BUFFERS 0-15
     .[32:16] TU8, BUFFERS 0-15
D[13].[0:16]  TU9, BUFFERS 0-15
     .[16:16] TU10, BUFFERS 0-15
     .[32:16] TU11, BUFFERS 0-15
D[14].[0:16]  TU12, BUFFERS 0-15
     .[16:16] TU13, BUFFERS 0-15
     .[32:16] TU14, BUFFERS 0-15
D[15].[0:16]  TU15, BUFFERS 0-15
     .[16:32] NOT USED
```

D[16]	"Q" OR TIME OUT VALUE FOR DATA COMMUNICATIONS INPUT AFTER WHICH A "BLAST READ" WILL BE PERFORMED BY THE MCP.
D[17]	SPECIFIES WHICH REMOTE UNITS ARE SUPERVISORY PRINTERS.

D[18]	SPECIFIES TIME OF DAY (XCLOCK).
D[19]	VALUE OF FENCE (TSSMCP).
D [20].[8:10]	SERIAL NUMBER FOR THE <MFID> OF THE SYSTEM LOG FILE CREATED WHEN A "LN" IS ENTERED THROUGH THE KEYBOARD (TSSMCP).
.[30:18]	LOCATION OF THE NEXT SEGMENT TO BE WRITTEN IN THE TSS LOG.
D[21]	STATUS OF SCHEDULE LINES (TSSMCP).
D[22].[28:10]	SERIAL NUMBER FOR THE <MFID> OF THE REMOTE LOG FILE CREATED WHEN AN "LR" IS ENTERED THROUGH THE KEYBOARD (BATCH SYSTEM).
.[38:10]	SERIAL NUMBER OF THE <MFID> OF THE MAINTENANCE LOG FILE CREATED WHEN AN "LNML" IS ENTERED THROUGH THE KEYBOARD.
D[24]-D[25]	DISK CONFIGURATION - DKA. EACH CHARACTER POSITION REPRESENTS AN EU. SU 0 = .[1:1], SU1=.[2:1], ETC. BITS 28-47 REPRESENT THE SPEED OF A EU. 1=40 MIL 0=20 MIL EU 0=BIT 47.
D[26]-D[27]	SAME AS D[24]-D[25] EXCEPT FOR DKB IF NO DFX IS PRESENT.
D[28]	DISK ADDRESS OF DIRECTORYTOP.

THE BOOLEAN VARIABLE "HOLDFREE" WHEN TRUE INDICATES THAT DIRECTORYTOP IS BEING MODFIED.

DIRECTORYTOP - AUXILIARY

EACH SYSTEM IN A SHAREDISK ENVIRONMENT HAS ITS OWN COPY OF DIRECTORYTOP. THE FORMAT OF THE AUXILIARY DIRECTORYTOP ENTRIES IS THE SAME AS THAT DEFINED FOR DIRECTORYTOP. THE PLACEMENT OF THE AUXILIARY COPIES IS ILLUSTRATED IN THE FOLLOWING TABLE:

AVAILABLE DISK STACK	AUX DIRECTORYTOP (SYS 3)	AUX DIRECTORYTOP (SYS 2)	AUX DIRECTORYTOP (SYS 1)	MAIN DIRECTORYTOP (SYS 0)

- DIRECTORYTOP + 1 - DIRECTORYTOP + 3 -

DIRECTORYTOP + 1 (SHAREDISK)

WORD	DESCRIPTION
0	SCRATCHVEC FOR SYS A
1	SCRATCHVEC FOR SYS B
2	SCRATCHVEC FOR SYS C
3	SCRATCHVEC FOR SYS D

DIRECTORYTOP + 2 (SHAREDISK)

WORD	DESCRIPTION
0	HOLDER
1	BYPASS
2	NEXTSLOT

DIRECTORYTOP + 3 (SHAREDISK)

WORD	DESCRIPTION
0	"LASTCDNUM"
1	"FIRSTDECK"
2	"LASTDECK"

DISK ORGANIZATION

INTRODUCTION

THE B5700 IS A DISK ORIENTED SYSTEM WHICH UTILIZES HEAD-PER-TRACK DISK FOR BOTH OPERATING SYSTEM FUNCTIONS AND STORAGE OF USER FILES.

DISK STORAGE CAN BE SEPARATED INTO TWO SECTIONS, FOR DISCUSSION PURPOSES, WHICH ARE SYSTEM DISK AND USER DISK. SYSTEM DISK CONSISTS OF A SMALL AMOUNT OF DISK, LOCATED ON ELECTRONICS UNIT ZERO, STORAGE UNIT ZERO, RESERVED FOR USE BY THE OPERATING SYSTEM(S). PART OF THIS AREA IS THE DISK DIRECTORY, WHICH IS MAINTIAINED BY THE MASTER CONTROL PROGRAM, AND CONTAINS INFORMATION CONCERNING ALL PERMANENT FILES ON USER DISK.

ON THE B5700, THE OPERATING SYSTEM CONTROLS CREATION, PLACEMENT, AND ACCESSING OF ALL DISK FILES. A USER PROGRAM REFERS TO A FILE THROUGH USE OF A SEVEN CHARACTER <MULTIFILE ID> AND A SEVEN CHARACTER <FILE ID> OF THE FORM <MFID>/<FID>. THE PROGRAMMER CAN SPECIFY RECORD SIZE, BLOCK SIZE, THE NUMBER OF RECORDS FOR EACH ROW (PAGE SIZE), AND THE MAXIMUM NUMBER OF ROWS. THE MCP WILL ACQUIRE EACH ROW AS IT IS NEEDED, TO A MAXIMUM OF 20 ROWS, AND WILL CREATE AND MANTAIN ENTRIES IN THE DISK DIRECTORY PERTAINING TO THE FILE. REFERENCE IS MADE TO A SPECIFIC RECORD WITHIN A FILE IN TERMS OF LOGICAL RECORD POSITION FROM THE START OF THE FILE. AS THE MCP MAINTAINS DISK, NO PROGRAMMATIC REFERENCES ARE MADE TO ABSOLUTE DISK ADDRESSES.

THE ACTUAL PHYSICAL LOCATION OF AREAS IN SYSTEM DISK ABOVE THE START OF ESP DISK IS DETERMINED FROM THE "DIRECT" AND "DRCTRYTP" CARDS IN THE COLD/START DECK AND FROM MCP DEFINES. SYSTEM DISK CONTAINS THE FOLLOWING:

CONTENTS	STARTING DISK ADDRESS
SEGMENT ZERO	0
HALT LOAD KERNEL	1
ESP DISK & ABORT TABLE	50
AVAILABLE DISK TABLE	HOLDLIST - DISKAVAILTABLEMAX (DISKAVAILTABLEMAX IS DEFINED AT MCP SEQUENCE 00165500)
HOLDLIST	DCP DISK - (HOLDMAX+29) DIV 30 (HOLDMAX IS DEFINED AT MCP SEQUENCE 00418200)
DCP DISK	DRCTRYTP - 6

EUIOHOLDER	DRCTRYTP - 5
LOG SEGMENT	DRCTRYTP - 4
SYS D DIRECTORYTOP	DRCTRYTP - 3
SYS C DIRECTORYTOP	DRCTRYTP - 2
SYS B DIRECTORYTOP	DRCTRYTP - 1
SYS A DIRECTORYTOP	DRCTRYTP
SCRATCH DIRECTORY HDR.	DRCTRYTP + 1
DIRECTORY INTERLOCK	DRCTRYTP + 2
CONTROL DECK INTERLOCK	DRCTRYTP + 3
MAIN DISK DIRECTORY	DRCTRYTP + 4
BYPASS DIRECTORY	BYPASS,[33:15] (THE BYPASS DIRECTORY ACTUALLY STARTS AT DIRECT - 4 AND GROWS TOWARDS THE MAIN DIRECTORY)

DRCTRYTP - 6, DCP DISK, IS USED ONLY IF "DCP" IS SET TRUE ON THE TIME SHARING SYSTEM.

DRCTRYTP - 4 THROUGH DRCTRYTP - 1, AND DRCTRYTP +1 THROUGH DRCTRYTP + 3 ARE USED BY SHAREDISK SYSTEMS ONLY.

USER DISK OCCUPIES THE REMAINDER OF DISK STARTING AT DIRECT + 4.

SEGMENT ZERO (DISK)

DISK SEGMENT ZERO CONTAINS THE NAMES OF THE MCP AND INTRINSIC FILES FOR ALL SYSTEMS. THE "KERNEL" READS THIS INFORMATION DURING A HALT/LOAD OPERATION AND LOADS THE SPECIFIED MCP FILE. THE MCP READS SEGMENT ZERO TO DETERMINE THE APPROPRIATE INTRINSICS FILE.

INFORMATION IS ENTERED IN THE SEGMENT THROUGH THE "CM" AND "CI" KEYBOARD MESSAGES AND BY THE "COOL START" AND "COLD START" PROGRAMS.

HALT/LOAD "KERNEL"

THE "KERNEL" IS BROUGHT INTO MAIN MEMORY EITHER BY THE HALT/LOAD OPERATION IF THE "KERNEL" IS ON DISK OR THROUGH THE CARD READER USING THE "CARD LOAD SELECT" HARDWARE FEATURE. IT READS DISK SEGMENT ZERO TO DETERMINE WHAT MCP IS TO BE LOADED, GOES TO THE ABSOLUTE ADDRESS SPECIFIED BY THE APPROPRIATE ENTRY, AND LOADS THE SPECIFIED MCP.

ESP DISK

EXECUTIVE SCRATCH PAD DISK IS USED BY THE MCP FOR TEMPORARY STORAGE.

ON SHAREDISK SYSTEMS THIS AREA ALSO CONTAINS THE SCRATCH DIRECTORY.

SCRATCH DIRECTORY (SHAREDISK)

A SHARED DISK "MCP" MAINTAINS A RECORD OF CURRENT SCRATCH FILE ROWS. THIS RECORD IS REQUIRED TO RELEASE SCRATCH FILES BELONGING TO THE SYSTEM WHEN THE SYSTEM GOES DOWN.

IF A SYSTEM STOPS RUNNING, ANOTHER SYSTEM OBTAINS THE INITIAL DISK ADDRESS LINK TO THE DOWN SYSTEM'S TABLE 1. IT THEN PROCEEDS SEQUENTIALLY FROM ONE SEGMENT TO ANOTHER, FREEING UP ALL OF THE DOWN SYSTEM'S SCRATCH FILE ROWS.

ABORT TABLE

THE ABORT TABLE IS USED TO CREATE SYSTEM LOG ENTRIES FOR JOBS THAT WERE RUNNING WHEN A SYSTEM HALT OCCURRED. THE ABORT TABLE IS SIX SEGMENTS LONG AND IS LOCATED BETWEEN THE END OF ESP DISK AND THE AVAILABLE DISK TABLE. THE ABORT TABLE IS SIX SEGMENTS (180 WORDS) LONG.

ON SHAREDISK SYSTEMS, EACH SYSTEM HAS ITS OWN ESP DISK AND ABORT TABLE. THE AREA BETWEEN THE "KERNEL" AND THE AVAILABLE DISK TABLE - 6 IS EVENLY DIVIDED FOR EACH SYSTEM.

AVAILABLE DISK TABLE

THE AVAILABLE TABLE MAINTAINS AN ACCOUNT OF THE AVAILABLE PORTIONS OF DISK, I.E., THOSE SEGMENTS OF DISK NOT IN USE AND NOT PART OF A PERMANENT FILE.

DISK DIRECTORY

THE DISK DIRECTORY CONSISTS OF TWO SECTIONS, THE MAIN DIRECTORY AND THE BYPASS DIRECTORY. THE MAIN DISK DIRECTORY CONTAINS THE <MFID>/<FID> OF EACH DISK FILE AND 30 WORDS (1 SEGMENT) OF DETAIL INFORMATION. THE BYPASS DIRECTORY IS LOCATED AT THE END OF THE MAIN DIRECTORY AND IS USED TO PROVIDE FAST ACCESS TO THE MAIN DIRECTORY ENTRIES.

A DETAILED EXPLANATION OF EACH OF THE AREAS APPEARS IN OTHER PARTS OF THIS DOCUMENT.

DISKADDRESS

DISKADDRESS(MID,FID,A,H,IO) IS A PROCEDURE WHICH OBTAINS THE NEXT DISK ADDRESS FOR THE FILE WHOSE LAST ADDRESS WAS "A", WITH HEADER "H", "IO" SPECIFYING AN I/O FILE.

DISKBOTTOM

VARIABLE SET TO THE HIGHEST ADDRESS OF THE DIRECTORY.

DISKERROR

DISKERROR(PARAM) IS A PROCEDURE WHICH OUTPUTS A DISK ERROR MESSAGE OF THE FORM:

```
DKX  DA=XXXXXXXX I=XXXXXXXXXXXXXXXX R=XXXXXXXXXXXXXXXX RT=X IO=X M=X
```

DISKIO

DISKIO(LOCIOD,CORE,SIZE,DISK) IS A PROCEDURE WHICH INITIATES, BUT DOES NOT WAIT FOR COMPLETION OF DISK I/O OPERATIONS FROM ADDRESS "DISK" (BINARY) OF SIZE "WORDS", INTO ADDRESS "CORE"+1, RETURNING THE DESCRIPTOR AT "LOCIOD".

DISKOUNT

COUNTER USED IN DISK FILE EXCHANGE CONFIGURATIONS TO KEEP TRACK OF THE NUMBER OF CONTROLS IN USE.

DISKWAITIME

DESCRIPTOR POINTING TO DISKWAITIME ARRAY WHICH CONTAINS DISK WAIT TIME TOTALS BY MIX INDEX FOR STATISTICS CODE.

EGGCLK

COUNTER USED TO RECORD THE NUMBER OF TIMES EGGTIMER FACILITY HAS BEEN CALLED FROM OUTER BLOCK CODE.

EGGENTER

EGGREMOVE

EGGSELECT

EGGENTER, EGGSELECT, EGGREMOVE AND CODE AT THE TIMER INTERRUPT AREA PROVIDE AN EGGTIMER FACILITY WHICH CAN BE USED TO EVOKE AN ACTION AFTER A GIVEN PERIOD OF TIME. ONE OF TWO ACTIONS CAN BE CAUSED:

THE CONTENTS OF A MEMORY LOCATION CAN BE MODIFIED
A ONE-PARAMETER PROCEDURE CAN BE CALLED

IF THE FIRST ACTION IS DESIRED, A CALL ON EGGENTER SHOULD BE PERFORMED PASSING A NEGATIVE VALUE IN "WHATODO" - THE [33:15] FIELD OF "WHATODO" MUST CONTAIN THE ADDRESS OF THE LOCATION TO BE MODIFIED. THEN AFTER "SECONDS" SECONDS, THE VALUE OF "PARAMETER" WILL BE OR-ED INTO THAT LOCATION.

IF THE SECOND ACTION IS DESIRED, "WHATODO" MUST CONTAIN ONLY THE ADDRESS OF THE PROCEDURE TO BE CALLED - THE ADDRESS MUST BE IN THE [33:15] FIELD OF "WHATODO". AFTER "SECONDS" SECONDS, THE PROCEDURE WILL BE CALLED PASSING "PARAMETER" AS ITS PARAMETER.

NOTE: THE PROCEDURE IS CALLED, NOT RUN, THROUGH USE OF INDEPENDENTRUNNER.

EGGSELECT IS RUN AT TIMER, WHEN REQUIRED.

EGGREMOVE IS CALLED BY EGGSELECT TO REMOVE ENTRIES WHICH MAY HAVE BEEN ACTED UPON. IF EARLY REMOVAL IS DESIRED, EGGREMOVE MAY BE CALLED DIRECTLY. WHEN THIS IS DONE, "WHATODO" AND "PARAMETER" MUST BE IDENTICAL TO THOSE IN THE EGGENTER CALL WHICH MADE THE ENTRY AND "SPACE" MUST BE ZERO.

THIS FACILITY USES A LINKED-LIST WITH FOUR-WORD NODES, COMPOSED AS FOLLOWS:

WORD	FIELD	CONTENTS
0	[18:15]	TIME TO END DELAY (IN SECONDS)
	[33:15]	LINK TO NEXT ENTRY
1	[33:15]	LINK TO LAST ENTRY
2		WHATODO (SEE DESCRIPTIONS ABOVE)
3		PARAMETER (SEE DESCRIPTIONS ABOVE)

- EGGENTER, EGGREMOVE, EGGSELECT -

THE PROCEDURE DECLARATIONS FOR EGGENTER,EGGSELECT, AND EGGREMOVE ARE AS FOLLOWS:

```
      PROCEDURE EGGENTER(WHATODO,PARAMETER,SECONDS);

                VALUE WHATODO,PARAMETER,SECONDS;
                REAL WHATODO,PARAMETER;
                INTEGER SECONDS;


      PROCEDURE EGGSELECT(PARAMETER);

                VALUE PARAMETER;
                REAL PARAMETER;


      PROCEDURE EGGREMOVE(WHATODO,PARAMETER,SPACE);

                VALUE WHATODO,PARAMETER,SPACE;
                REAL  WHATODO,PARAMETER,SPACE;
```

THE OUTER BLOCK CODE FOR THE EGGTIMER FACILITY IS AS FOLLOWS:

```
 IF M[WATER] NEQ WATER THEN
 BEGIN% EGGTIMER
      EGGCLK:=EGGCLK-1;
      IF EGGSELECT THEN
           IF P(WATER,0&NOT(EGGCLK)[CTF],LLL,DEL) NEQ WATER
           THEN BEGIN EGGSELECTSTOPPED:=FALSE;
                      INDEPENDENTRUNNER(P(.,EGGSELECT),0);
                END;
 END EGGTIMER;
```

A SAMPLE CALL ON EGGENTER IS AS FOLLOWS:

```
      EGGENTER(P(,INQUPT),NFLAG(RD)&0[CTC],4);
```

THIS WOULD HAVE CAUSED THE ROUTINE INQUPT TO BE CALLED AFTER AT LEAST 4 SECONDS WITH THE PARAMETER (NFLAG(RD)&0[CTC]).

ENDOFDECK

ENDOFDECK(R,TUSTA) IS THE NORMAL MEANS BY WHICH A DECK IS REMOVED FROM PSEUDO-READER "R" BY CALLING "REMOVETHEDECK".

ENTERCONTROLDECK

ENTERCONTROLDECK(H) IS A PROCEDURE WHICH MAKES THE APPROPRIATE DISK FILE HEADER ENTRY IN THE DIRECTORY FOR HEADER "H", CREATES THE PROPER "CHAIN" FROM THE LAST CONTROL DECK HEADER, AND UPDATES "LASTCDNUM".

ENTERUSERFILE

ENTERUSERFILE(A,B,L) IS A PROCEDURE WHICH ENTERS A PERMANENT FILE, "A"/"B" WITH HEADER "L" IN THE DISK DIRECTORY.

ESPBIT

THE ESPBIT PROCEDURE IS RESPONSIBLE FOR BRINGING MCP SEGMENTS INTO CORE. THIS PROCEDURE, RATHER THAN MAKEPRESENT, MUST BE USED FOR MCP SEGMENTS AS THE PRESENSE BIT INTERRUPT FEATURE IS NOT AVAILABLE WHILE OPERATING IN CONTROL STATE.

THE ESPOL COMPILER CREATES A TYPE OF "SEGMENT DICTIONARY" WHICH CONTAINS THE RELATIVE DISK ADDRESS OF ALL "OVERLAYABLE" PROCEDURES. THE OPERATION OF ESPBIT IS DEPENDENT ON THE PROCEDURE DESCRIPTORS FOR OVERLAYABLE PROCEDURES IN THE MCP-S PRT.

ESPBIT OBTAINS THE "C" AND "L" VALUES FROM THE RCW, PLACED IN THE STACK WHEN THE PROCEDURE WAS CALLED, TO DETERMINE THE SYLLABLE THAT CAUSED THE PROCEDURE CALL. FROM THIS THE PROCEDURE DESCRIPTOR IS LOCATED. USING THE INFORMATION IN THE DESCRIPTOR, ESPBIT THEN PLACES THE PRT ADDRESS OF THE PROCEDURE IN THE MEMORY LINK FOR USE BY OLAY, AND PLACES THE CORE ADDRESS OF THE PROCEDURE IN THE PROCEDURE DESCRIPTOR. A BRANCH IS THEN MADE TO THE APPROPRIATE PROCEDURE AS IF ESPBIT HAD NEVER INTERVENED.

ESPCOUNT

CONTAINS A COUNTER OF THE NUMBER OF ESPDISK SEGMENTS REMAINING.

ESPDISKBOTTOM

CONTAINS THE LOWEST ADDRESS OF ESPDISK.

ESPDISKTOP

CONTAINS THE HIGHEST ADDRESS OF ESPDISK.

ESPTAB

CONTAINS A POINTER TO A BIT TABLE USED IN ALLOCATION OF ESPDISK.

EUIO

CONTAINS THE I/O TIME USED BY A GIVEN EU. THIS INFORMATION IS USED IN AN ATTEMPT TO REDUCE EU ACCESS CONFLICT.

EUQ

DESCRIPTOR POINTING TO EUQ ARRAY, WHICH IS USED FOR HANDLING A DISK FILE EXCHANGE CONFIGURATION.

EUW

CONTAINS INFORMATION RELATED TO THE EUQ ARRAY USED IN A DISK FILE EXCHANGE CONFIGURATION.

FETCH

FETCH(UNITNO,CARDLOC,SOURCE) OBTAINS THE NEXT IN A SERIES OF CONTROL "CARDS", WHICH MAY ACTUALLY BEAR LITTLE RELATION TO 80-COLUMN RECORDS, ON LOGICAL UNIT "UNITNO" WITH "SOURCE" AND "CARDLOC" ALTERNATING AS THE ORIGIN OF INFORMATION. "SOURCE" IS SET TO THE BEGINNING OF THE CARD AND THE LAST WORD OF THE CARD IS SET EQUAL TO A PERIOD (.).

FILE INFORMATION BLOCK

AT RUN TIME, THERE IS ONE FIB GENERATED FOR EACH FILE TO BE USED BY A PROGRAM. A 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.

(FOR ALGOL PLACED ABOVE TANK)

WORD	FIELD	CONTENTS
0		COBOL (DOD) BEGINNING FILE USE ROUTINES
	[1:11]	STARTING INDEX BEFORE ROUTINE
	[12:12]	ENDING INDEX BEFORE ROUTINE
	[24:12]	STARTING INDEX AFTER ROUTINE
	[36:12]	ENDING INDEX AFTER ROUTINE
		COBOL68 BEGINNING USE ROUTINES
	[1:11]	BEFORE BEGINNING FILE USE ROUTINE
	[12:12]	AFTER BEGINNING FILE USE ROUTINE
	[24:12]	BEFORE BEGINNING REEL USE ROUTINE
	[36:12]	AFTER BEGINNING REEL USE ROUTINE
1		DISK FILES - LOWER BOUND RECORD NUMBER (LSUBU)
		COBOL (DOD) BEGINNING REEL USE ROUTINES
	[1:11]	STARTING INDEX BEFORE ROUTINE
	[12:12]	ENDING INDEX BEFORE ROUTINE
	[24:12]	STARTING INDEX AFTER ROUTINE
	[36:12]	ENDING INDEX AFTER ROUTINE
		ALGOL - BEGINNING REEL NUMBER
2		COBOL (DOD) ENDING FILE USE ROUTINES
	[1:11]	STARTING INDEX BEFORE ROUTINE
	[12:12]	ENDING INDEX BEFORE ROUTINE
	[24:12]	STARTING INDEX AFTER ROUTINE
	[36:12]	ENDING INDEX AFTER ROUTINE

- FILE INFORMATION BLOCK -

```
                    COBOL68 ENDING USE ROUTINES
          [1:1]     BEFORE ENDING FILE USE ROUTINE
          [12:12]   AFTER ENDING FILE USE ROUTINE
          [24:12]   BEFORE ENDING REEL USE ROUTINE
          [36:12]   AFTER ENDING REEL USE ROUTINE

3                   DISK FILES - UPPER BOUND RECORD NUMBER (LSUBU)

                    COBOL (DOD) ENDING REEL USE ROUTINES
          [1:11]    STARTING INDEX BEFORE ROUTINE
          [12:12]   ENDING INDEX BEFORE ROUTINE
          [24:12]   STARTING INDEX AFTER ROUTINE
          [36:12]   ENDING INDEX AFTER ROUTINE

                    COBOL68 - PRT ADDRESS FOR PRINTER
                    LINKAGE COUNTER

4         [1:1]     1 = USE ROUTINES PRESENT
          [2:1]     1 = LABELS OMITTED
          [3:2]     "EOR" RERUN
                    0 = NO
                    1 = OUTPUT TAPE
                    2 = SCRATCH TAPE
          [5:1]     1 = OPTIONAL
          [6:1]     1=NO IN-OUT PART
          [7:1]     1= SORT FILE
          [8:4]     INTERNAL TYPE CODE
          [12:1]    0 = BITS [13:11] IS FILE NUMBER
                    1 = BITS [13:11] IS "FPB" INDEX
          [13:11]   FILE NUMBER FOR FPB INDEX
          [24:1]    1 = RELEASE UNIT AT "CLOSE"
          [25:2]    DISPOSITION OF FILE
                    0 = REWIND
                    1 = NO REWIND
                    2 = REWIND AND LOCK
                    3 = REWIND AND RELEASE
          [27:3]    ACCESS MODE
                    0 = SERIAL
                    1 = RANDOM
                    2 = UPDATE
          [30:18]   SAVE FACTOR

5         [1:1]     INDICATES AN INVALID USER AS
                    OPPOSED TO A PARITY
          [2:1]     MUST NOT BE USED
          [6:1]     NONSTANDARD LABELS
          [7:1]     COBOL (BLOCK LOCK)
          [8:4]     ALGOL OR FORTRAN FILE ATTRIBUTE
          [9:2]     OTHERUSE
                    0 = CANTUSE
```

- FILE INFORMATION BLOCK -

```
                    1 = INPUT
                    2 = OUTPUT
                    3 = I-O
          [11:2]    MYUSE
                    0 = CANTUSE
                    1 = INPUT
                    2 = OUTPUT
                    3 = I-O
          [13:3]    SHARED DISK
                    =0 OPEN SHARED
                    =1 OPEN INPUT
                    =2 OPEN OUTPUT
                    =3 OPEN WRITE LOCK
                    =4 OPEN EXCLUSIVE
          [16:1]    NON-STNADARD LABEL
          [17:1]    LABEL EQUATED FROM DISK
          [18:15]   PRINT RECORD NUMBER (PB ONLY)
          [38:1]    1 = CURRENTREEL NEQ FIRSTREEL
          [39:1]    1 = REEL OPTIONAL AND ABSENT
                    (COBOL ONLY)
          [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]    PARITY LAST I-O
          [46:2]    0 = UNBLOCKED
                    1 = TECH A
                    2 = TECH B
                    3 = TECH C

6                   BLOCK COUNT

7                   RECORD COUNT

8         [1:1]     HASH TOTALS = 0
          [2:1]     INDICATES [25:23] IS
                    IN SEGMENTS
          [18:2]    NOT USED
          [20:5]    NUMBER OF ROWS (DISK FILES ONLY)
          [25:23]   SIZE OF ROWS (DISK FILES ONLY)
          [33:15]   RELATIVE "PRT" LOCATION OF DESCRIPTOR
                    FOR HASH TOTALS (TAPE FILES ONLY)

9                   RERUN CONTROL (NUMBER OF RECORDS)
          [2:1]     BLOCK CHECKED (COBOL)
                    TRUE IF BOUNDED
          [3:45]    RERUN CONTROL(USED BY BREAKOUT)

10                  RESERVED FOR BREAKOUT-RESTART
          [1:1]     WRITABLE TAPE AT BREAKOUT
```

- FILE INFORMATION BLOCK -

	[2:1]	NOT USED
	[3:15]	HEAD OF BUFFER RING
	[18:30]	TRANSACTION
11		NUMBER OF RECORDS PER BLOCK
12		NUMBER OF RECORDS IN CURRENT BLOCK
13	[1:9]	NUMBER OF BUFFERS REQUESTED MUST =1 FOR PB FILES =2 FOR REMOTE FILES
	[10:9]	NUMBER OF BUFFERS ASSIGNED MUST =2 FOR REMOTE FILES
	[19:1]	1 = BAD KEY
	[20:1]	1 = SEEK GIVEN
	[21:1]	1 = READ (1ST OPERATION) WAS LAST PREVIOUS OPERATION
	[22:1]	1 = OPEN UPDATE
	[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]	USED BY COBOLIO AND COBOLFCR
	[47:1]	1 = COBOL
14		DESCRIPTOR FOR DISK FILE HEADER IN CORE. INPUT NUMBER OF BUFFERS REQUESTED (REMOTE FILES ONLY)
	[3:15]	HEADER ADDRESS (PBD ONLY)
	[18:15]	BASE ADDRESS OF NEXT RECORD SLOT IN BLOCK (PB ONLY)
	[33:15]	BASE ADDRESS OF CURRENT RECORD BLOCK (PB ONLY)
15	[1:23]	COBOL: PARITY ERROR USE ROUTINE
	[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	[1:1]	1 = IGNORE BUFFERS IN OPEN-CLOSE (SORT MANAGES THEM)
	[2:1]	COBOL - FILE TYPE CHANGED FROM DISK TO TAPE
	[3:15]	BUFFER SIZE

* FILE INFORMATION BLOCK *

	[18:15]	"TECH C" BUFFER LENGTH
	[33:15]	MAXIMUM RECORD LENGTH
*19		FINAL I-O DESCRIPTOR FOR PROGRAM RELEASE (FINALQUE)
20		COBOL68 <MFID>
21		COBOL68 <FID>

* UNFLAGGED

INTERNAL TYPE CODES (USED IN FIB[4].[8:4]) "KIND"

0 = CR
1 = LP
2 = MT
3 = DR
4 = DK
5 = SPO
6 = CP
7 = PBT
8 = PP
9 = PR
10 = DC
11 = CD
12 = PBD
13 = REMOTE

EXTERNAL TYPE CODES (USED IN FIB[13].[39:5]) "TYPE"

0 = CP/CR
1 = LP
2 = MT
3 = DESIGNATED
4 = LP/PBT
5 = SPECIFIED (MUST BE UNLABELED)
6 = PBT
7 = PT
8 = PT UNLABELED
9 = MT UNLABELED
10 = DISK RANDOM
11 = SPO
12 = DISK SERIAL
13 = DISK UPDATE
14 = DATACOM
15 = PBD
16 = PBT/PBD
17 = LP/PBD

- FILE INFORMATION BLOCK -

18 = LP/PBT/PBD
19 = REMOTE
20 = PUT (PUNCH BACK-UP TAPE)
21 = CP/PUT
22 = PUD (PUNCH BACK-UP DISK)
23 = CP/PUD
24 = PUT/PUD
25 = CP/PUT/PUD

FILE PARAMETER BLOCK

EACH PROGRAM HAS A COMPILER FORMAT FPB WHICH IS CREATED WHEN THE PROGRAM IS COMPILED. IT IS LATER USED (ALONG WITH OTHER INFORMATION) BY THE SELECTION ROUTINE DURING THE "FIX-UP" BEFORE A PROGRAM IS INITIATED TO CONSTRUCT THE INITIAL PROGRAM FPB. THE PROGRAM FPB HAS AN ENTRY FOR EVERY TO BE USED BY THE PROGRAM.

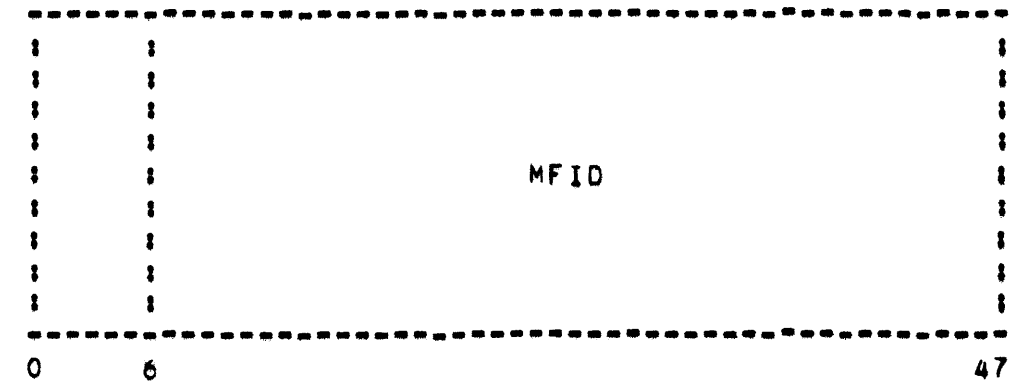
WHEN A FILE IS DECLARED IN A PROGRAM, THAT IS, WHEN THE SOURCE PROGRAM ASSOCIATES THE FILE IDENTIFIER WITH A FILE NAME AND FILE HANDLING TECHNIQUES, THE COMPILER ASSIGNS THE FILE IDENTIFIER A FILE NUMBER. THIS FILE NUMBER, RATHER THAN THE FILE IDENTIFIER, IS THEN USED IN ALL REFERENCES MADE TO THE CORRESPONDING FILE BY THE OBJECT PROGRAM. FOR EACH FILE MEMBER, AND IN FILE NUMBER ORDER, THERE IS AN ENTRY IN THE COMPILER FORMAT FPB. EACH ENTRY IN THE COMPILER FPB CONTAINS THE FILE IDENTIFIER, THE MULTIPLE FILE IDENTIFICATION, AND THE FILE IDENTIFICATION FOR THE PARTICULAR NUMBER. THE LOCATION AND SIZE OF THE COMPILER FPB ARE PLACED IN AN ENTRY OF THE PROGRAM-S SEGMENT ZERO. WHEN THE SELECTION PROCEDURE IS PERFORMING "FIX-UP" OPERATIONS, IT USES THIS INFORMATION TO CONSTRUCT THE FPB. THE COMPILER 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 IDENTFIER 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 COMPILER FPB, IT ALSO OBTAINS ALL LABEL EQUATION CARDS FOR THE PROGRAM, IF ANY. THEN THE FILE IDENTIFIERS IN THE PROGRAM-S FPB ENTRIES ARE COMPARED WITH THE FILE IDENTIFIERS ON THE LABEL EQUATION CARDS. IF A MATCH IS FOUND, INFORMATION IN THE COMPILER 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 COMPILER FPB AGAIN BY REMOVING THE FILE IDENTIFIER ENTRIES, WHICH ARE NO LONGER REQUIRED. THEN A DESCRIPTOR CONTAINING THE ADDRESS OF THE REBUILT FPB IS PLACED IN (R+3) IN THE OBJECT PRGRAM-S PRT. USING THIS DESCRIPTION AND A FILE NUMBER, THE OBJECT PROGRAM IS ABLE TO MAKE ALL NECESSARY REFERENCES TO FPB ENTRIES.

- FILE PARAMETER BLOCK -

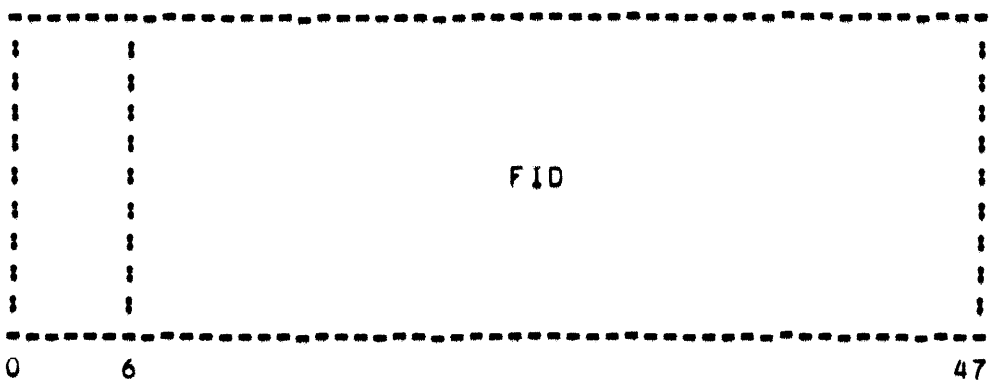
(FPB) (ADDRESSED BY R+3)

WORD 1



FIELD	CONTENTS	DESCRIPTION
[0:6]	0	NOT USED
[6:42]	MFID	SEVEN CHARACTERS MULTI-FILE IDENTIFICATION

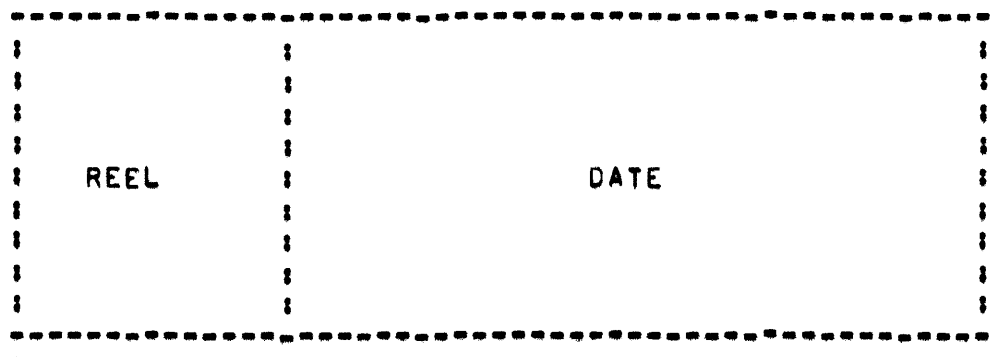
WORD 2



FIELD	CONTENTS	DESCRIPTION
[0:6]	0	NOT USED
[6:42]	FID	SEVEN CHARACTER FILE IDENTIFICATION

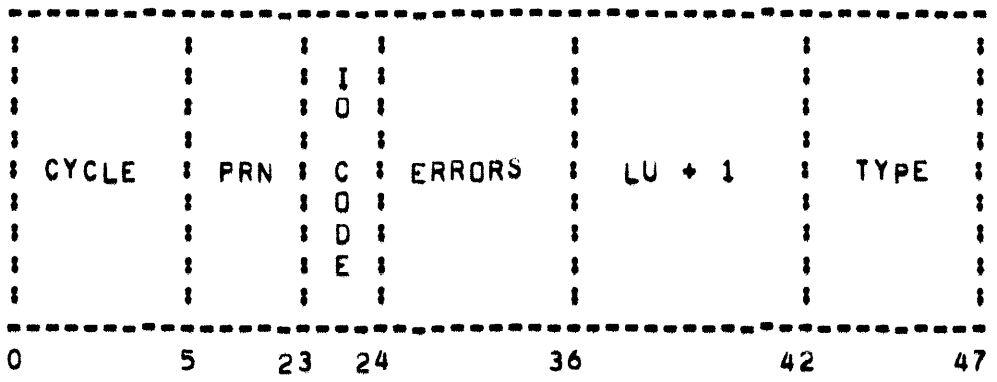
- FILE PARAMETER BLOCK -

WORD3



FIELD	CONTENTS	DESCRIPTION
[0:18]	REEL	REEL NUMBER IN THREE CHARACTER ALPHA
[18:30]	DATE	CREATION DATE IN FIVE CHARACTERS

WORD 4



FIELD	CONTENTS	DESCRIPTION
[1:5]	CYCLE	CYCLE NUMBER (BINARY)
[6:17]	PRN	PHYSICAL REEL NUMBER
[23:1]	IC CODE	0=IF INPUT, 1=IF OUTPUT
[24:12]	ERROR	NUMBER OF ERRORS
[36:6]	LU+1	LOGICAL UNIT NUMBER PLUS ONE
		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 = PPB
29 = PPB
30 = PRB
31 = DCA
32 = MCP

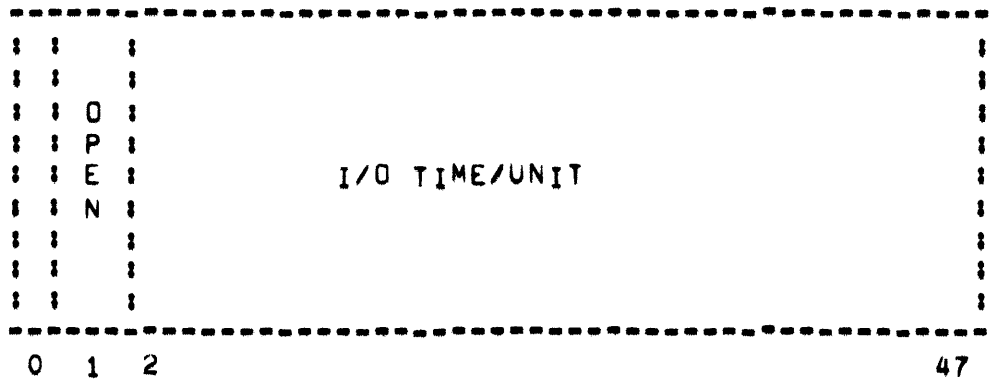
[42:1] FORMS — ZERO INDICATES UNIT NOT ASSIGNED

[42:5] TYPE — 0 = CP-CR

1 = LP
2 = MT
3 = DG-DESIGNATED
4 = LP
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 = DATE COMMUNICATION
15 = PBD ONLY
16 = PBT/PBD
17 = LP/PBD
18 = LP/PBT/PBD
19 = REMOTE DATACOM
20 = PUT (PUNCH BACK-UP TAPE)
21 = CP/PUT
22 = PUD (PUNCH BACK-UP DISK)
23 = CP/PUD
24 = PUT/PUD
25 = CP/PUT/PUD

WORD 5



FIELD	CONTENTS	DESCRIPTION
[1:1]	1	FILE OPEN
[2:46]		I-O TIME PER UNIT

- FILE PARAMETER BLOCK -

FPB GENERATED BY THE COMPILER

CHARACTER POSITION

1,2	FILE NUMBER (12 BIT BINARY) STARTS WITH 1.
3	ALGOL FILE TYPE
4-10	MFID
11-17	FID
18	LENGTH OF INTERNAL ID (6 BIT BINARY)
19-N	INTERNAL ID
N+1...	STARTS OVER AGAIN OR 2 ZEROES TO QUIT.

FILE TANK

ALGOL (ADDRESSED BY A DESCRIPTOR IN THE FILE-S PRT).

WORD	CONTENTS
-1	NUMBER OF BUFFERS - 1
0	LOCATQUE SKELETON, ALSO TEMPORARY STORAGE FOR ALGOL PARITY LABEL AND FORTRAN ERROR LABEL
1	ZERO, ALSO TEMPORARY STORAGE FOR ALGOL AND FORTRAN EOF LABEL
2	POINTER TO "FIB[0]"
3	POINTER TO READ-IN LABEL IF INPUT POINTER TO COMPILER LABEL IF OUTPUT
4	POINTER TO TOP I-O DESCRIPTOR
5	TOP I-O DESCRIPTOR (FILE DESCRIPTOR POINTS HERE)
6	REMAINING I-O DESCRIPTORS
.	.
.	.
.	.
N	.

THE FIB IS TO BE PLACED ABOVE THE TANK FOR ALL JOBS WITH THE EXCEPTION OF DOD COBOL WHICH HAS THE TANK IN ITS PRT.

COBOL

WORDS -1 THROUGH 1 DO NOT EXIST FOR COBOL PROGRAMS. THE PRT LOCATIONS FOR THOSE ENTRIES ARE LISTED UNDER THE "SELECT" CLAUSE FOR EACH FILE. THE ENTRIES START AT PRT @72.

FILE SECURITY

LEVELS OF SECURITY.

THE FILE SECURITY SYSTEM HAS BEEN DESIGNED TO PROHIBIT UNAUTHORIZED USERS FROM HAVING ACCESS TO THE SYSTEM OR TO ANY FILES BELONGING TO AUTHORIZED USERS. THE SIMPLER, MOST USED, PORTION OF THE SYSTEM RELATES TO VARIOUS LEVELS OF ACCESS TO THIS FILES ONE USER MAY GRANT TO OTHERS THROUGH THE SOLE USE OF CONTROL CARDS APPLIED TO PREVIOUSLY-SAVED DISK FILES. THE MORE SOPHISTICATED PORTION OF THE SYSTEM IS BASED ON THE CONCEPT THAT FILES MAY BE MADE PRIVATE BY A SECURITY FILE WHICH CONTAINS A LIST OF AUTHORIZED USERS AND PROGRAMS WHICH MAY ACCESS THE FILE. ONLY THE CREATOR OF THE FILE MAY ESTABLISH AND MAINTAIN THE SECURITY FILE ASSOCIATED WITH HIS FILE. A USER TRYING TO USE A PRIVATE FILE WILL BE CHECKED BY THE MCP AGAINST THE LIST BEFORE THE REQUESTOR IS ALLOWED TO ACCESS ANY RECORDS CONTAINED IN THE FILE. IF NEITHER THE REQUESTOR"S USER CODE NOR THE PROGRAM SPECIFIER OF THE PROGRAM BEING EXECUTED ARE CONTAINED IN THE SECURITY FILE FOR THE FILE BEING OPENED, THEN PROGRAM CONTROL WILL BE TRANSFERRED TO EITHER THE PARITY ACTION LABEL (FOR ALGOL OR FORTRAN PROGRAMS) OR TO THE USE ROUTINE (FOR COBOL PROGRAMS). THE ABSENCE OF AN ERROR HANDLING LABEL OR ROUTINE WILL RESULT IN THE TERMINATION OF THE PROGRAM.

ONCE THE REQUESTOR IS DEFINED AS BEING A PRIVILEGED (UNLIMITED ACCESS), PRIMARY (UNLIMITED ACCESS TO FILES CREATED BY THIS USER), SECONDARY (ACCESS TO THE FILE FOR INPUT OR OUTPUT), OR TERTIARY USER (ACCESS FOR INPUT ONLY), THE FILE WILL BE MADE AVAILABLE TO THE REQUESTOR AND THE MCP WILL NOTE THE MANNER IN WHICH THIS REQUESTOR MAY ACCESS THE FILE. ACCESS TO THE FILE IN ANY UNAUTHORIZED MANNER WILL RESULT IN EITHER TRANSFER OF CONTROL TO THE ERROR HANDLING LABEL OR ROUTINE, OR TERMINATION OF THE PROGRAM.

IDENTIFIED USER.

A USER MAY BE IDENTIFIED BY EITHER LOGGING-IN FROM A REMOTE STATION OR ENTERING A USER CONTROL CARD. AN IDENTIFIED USER MAY OPEN A PRIVATE FILE FOR INPUT IF HE IS A PRIVILEGED, PRIMARY, SECONDARY, OR TERTIARY USER. AN IDENTIFIED USER MAY OPEN A PRIVATE FILE FOR INPUT AND OUTPUT IF HE IS A PRIVILEGED, PRIMARY, OR SECONDARY USER. AN IDENTIFIED USER MAY MAINTAIN A PRIVATE FILE (I.E., PERFORM LIBRARY OR SECURITY FILE MAINTENANCE) IF HE IS EITHER THE PRIVILEGED OR PRIMARY USER. A USER IS CONSIDERED THE PRIVILEGED USER IF HIS USER CODE IS THE FIRST ENTRY IN THE REMOTE/USERS FILE.

USER AND FILE CLASSIFICATION.

<PRIVILEGED USER> :=[A USER WHO HAS UNLIMITED ACCESS]

<PRIMARY USER> :=[THE CREATOR OF THE FILE. I.E., THE USER WHO CAUSED THE FILE TO BE ENTERED INTO THE DISK DIRECTORY]

<SECONDARY USER> :=[A USER DESIGNATED AS BEING ABLE TO ACCESS A FILE FOR INPUT OR OUTPUT]

<TERTIARY USER> :=[A USER DESIGNATED AS BEING ABLE TO ACCESS A FILE FOR INPUT ONLY]

<SECURITY FILE> :=[THE FILE CONTAINING USER CODES FOR THE SECONDARY AND/OR TERTIARY USERS]

<FREE FILE> :=[A FILE OPEN TO ALL USERS FOR INPUT, OUTPUT, AND LIBRARY MAINTENANCE, AND TO THE PRIVILEGED USER ONLY FOR SECURITY MAINTENANCE]

<PUBLIC FILE> :=[A FILE OPEN TO ALL USERS EXCEPT FOR LIBRARY OR SECURITY FILE MAINTENANCE]

<UNLOCKED FILE> :=[A FILE OPEN TO ALL USERS FOR INPUT ONLY, EXCEPT FOR PRIVILEGED USER, WHICH HAS ANY TYPE OF ACCESS]

<PRIVATE FILE> :=[A FILE WITH AN ASSOCIATED SECURITY FILE]

<SOLE-USER FILE> :=[A FILE WHOSE ONLY VALID USER IS ITS PRIMARY USER]

MCP ACTIONS.

THE MCP MAINTAINS A TABLE OF USER CODES CALLED THE USERCODE TABLE FOR EACH ACTIVE MIX INDEX. THIS TABLE CONTAINS THE CONTENTS OF THE USER CODE FROM EITHER THE "LI" MESSAGE OR THE "USER" CONTROL CARD. THE TABLE IS USED FOR THE PURPOSE OF FILE PROTECTION WHENEVER A DISK FILE IS OPENED OR CLOSED WITH LOCK OR PURGE.

IN ADDITION TO THE USERCODE TABLE, THE MCP MAINTAINS A TABLE CONTAINING USER CODES ASSOCIATED WITH EACH ACTIVE TERMINAL BUFFER. THE DATA COMMUNICATIONS INTERROGATE FUNTION IS CAPABLE OF UPDATING THE USERCODE TABLE WITH THE USER CODE ASSOCIATED WITH THE TERMINAL BUFFER SPECIFIED. THIS FACILITY IS NECESSARY AS TO ALLOW A PROGRAM WHICH MAY HANDLE MORE THAN ONE USER TO CREATE AND ACCESS FILES FOR ANY OF THOSE USERS.

MCP ACTIONS FOR FILES OPENED INPUT OR OUTPUT AND ALREADY IN THE DISK DIRECTORY:

1. ALL FREE AND PUBLIC FILES WILL BE MADE AVAILABLE TO ANY USER.

2. SOLE-USER FILES ARE AVAILABLE ONLY TO THE PRIMARY USER OF THE FILE.

3. ACCESSING ANY PRIVATE FILE WILL REQUIRE THAT AN ENTRY IN THE ASSOCIATED SECURITY FILE BE EQUIVALENT TO THE REQUESTING JOB'S ENTRY IN THE USERCODE TABLE OR TO THE PROGRAM SPECIFIER OF THE REQUESTING PROGRAM. IF NO VERIFICATION CAN BE MADE, TRANSFER WILL BE MADE TO THE PARITY ACTION LABEL (ALGOL OR FORTRAN) OR THE USE ROUTINE (COBOL), IF ONE IS PRESENT. OTHERWISE, THE PROGRAM WILL BE TERMINATED.

MCP ACTIONS FOR OUTPUT FILES CLOSED WITH LOCK AND NOT ALREADY IN THE DISK DIRECTORY.

1. IF THE USERCODE TABLE ENTRY IS EMPTY, THE FILE WILL BE MADE A FREE FILE.

2. OTHERWISE, THE FILE WILL BE ENTERED INTO THE DISK DIRECTORY AS A SOLE-USER FILE.

FOR ZIP (<PROGRAM SPECIFIER>), THE MCP WILL APPEND THE USER CODE FROM THE USERCODE TABLE TO THE CONTROL CARD INFORMATION. FOR THE USE OF THE GENERALIZED ZIP, THE MCP WILL INSERT THE USER CODE FROM THE USERCODE TABLE INTO THE ARRAY OR THE FILE HEADER, WHICHEVER IS APPROPRIATE.

USER CONTROL CARD.

A USER CONTROL CARD IS USED TO ENTER A USER CODE FROM AN INPUT SOURCE OTHER THAN A REMOTE STATION. USER CARDS RECEIVED FROM A REMOTE STATION ARE IGNORED. THE USER CODE FROM A USER CARD IS USED TO INITIALIZE A JOB-S ENTRY IN AN MCP TABLE OF USER CODES WHEN A JOB IS SELECTED TO RUN. A USER CARD MUST PRECEDE OTHER CONTROL CARDS AND, IF THERE IS MORE THAN ONE USER CARD PER DECK, ONLY THE FIRST CARD WILL HAVE ANY SIGNIFICANCE. IF NO USER CARD IS ENTERED, A JOB-S USER TABLE ENTRY WILL BE INITIALIZED TO ZERO.

THE FOLLOWING INFORMATION MUST APPEAR ON A USER CONTROL CARD:

```
?USER=<USER CODE>
```

EXAMPLES:

```
?USER=BATMAN
?USER=SUPERMAN
```

LIBRARY AND SECURITY FILE MAINTENANCE

LIBRARY MAINTENANCE FUNCTIONS REQUIRE THE PRESENCE OF A USER CODE FOR MAINTENANCE OF ANY NON-FREE FILE. THE USER CODE MUST BE INTRODUCED VIA A USER CONTROL CARD OR "LI" MESSAGE WHICH MUST PRECEDE THE LIBRARY MAINTENANCE CONTROL CARDS. NO LIBRARY MAINTENANCE WILL BE DONE ON NON-FREE FILES UNLESS THE USER CODE IS EQUIVALENT TO EITHER THE PRIVILEGED USER OF THE FILE. FIVE SECURITY FILE MAINTENANCE FUNCTIONS ARE PROVIDED TO MAINTAIN SECURITY FILES AND PRIVATE FILES AND ARE AS FOLLOWS:

?USE <SECURITY FILE SPECIFIER> ON <FILE SPECIFIER>

THE <FILE SPECIFIER> WILL BE MADE INTO A PRIVATE FILE USING THE <SECURITY FILE SPECIFIER> AS ITS SECURITY FILE. THE <SECURITY FILE SPECIFIER> NAMES A FILE WHICH MUST HAVE THE SECURITY FILE FORMAT AND MUST BE A SOLE-USER FILE. THIS FILE MAY OR MAY NOT BE A SECURITY FOR OTHER FILES AT THIS TIME. THE <FILE SPECIFIER> MUST BE A SOLE-USER FILE AND THE USER CODE OF THE REQUESTOR MUST BE EQUIVALENT TO THE PRIMARY USER OR THE USER OF THE FILE.

?LOCK <FILE SPECIFIER>

THE <FILE SPECIFIER> WILL BE MADE INTO A SOLE-USER FILE. THE <FILE SPECIFIER> MUST BE EITHER A PRIVATE OR PUBLIC OR UNLOCKED FILE AND THE USER CODE OF THE REQUESTOR MUST BE EQUIVALENT TO EITHER THE PRIVILEGED USER OR THE PRIMARY USER OF THE FILE. IN ADDITION, THE PRIVILEGED USER MAY LOCK A FREE FILE. LOCKED FILES ALLOW NO ACCESS TO OTHER USERS.

?PUBLIC <FILE SPECIFIER>

THE <FILE SPECIFIER> WILL BE MADE INTO A PUBLIC FILE. THE <FILE SPECIFIER> MUST BE A PRIVATE OR SOLE-USER OR UNLOCK FILE, AND THE USER CODE OF THE REQUESTOR MUST BE EQUIVALENT TO EITHER THE PRIVILEGED USER OF THE PRIMARY USER OF THE FILE. PUBLIC FILES ALLOW SECONDARY ACCESS TO OTHER USERS.

?UNLOCK <FILE SPECIFIER>

THE <FILE SPECIFIER> WILL BE MADE INTO AN UNLOCKED FILE. THE <FILE SPECIFIER> MUST BE A PRIVATE OR SOLE-USER OR PUBLIC FILE, AND THE USER CODE OF THE REQUESTOR MUST BE EQUIVALENT TO EITHER THE PRIVILEGED USER OR THE PRIMARY USER OF THE FILE. UNLOCKED FILES ALLOW TERTIARY USE TO OTHER USERS.

?FREE <FILE SPECIFIER>

THE <FILE SPECIFIER> WILL BE MADE INTO A FREE FILE. THE <FILE SPECIFIER> MUST BE EITHER A SOLE-USER, PRIVATE OR PUBLIC FILE, AND THE USER CODE OF THE REQUESTOR MUST BE EQUIVALENT TO EITHER THE PRIVILEGED USER OR THE PRIMARY USER OF THE FILE. FREE FILES ALLOW ANY ACCESS TO OTHER USERS.

EXAMPLES:

```
?USER=BOSS; USE SECURE/BLOCK ON ALGOL/*,DIRCTRY/DISK;END.
?USER=BOSS; LOCK COBOL/DISK,LOGOUT/DISK;END.
?USER=CHARLEY; PUBLIC CHARLEYS/FILES;END.
?USER=AXXUUVV; UNLOCK MY/FILE;END.
?USER=1234567; FREE PUBLIC/FILES;END.
```

THE FOLLOWING MESSAGES MAY BE OUTPUT AS A RESULT OF THE PRECEDING SECURITY MAINTENANCE CONTROL CARDS:

-INVALID USER<FILE DESIGNATOR>:<JOB SPECIFIER>,<TERMINAL REFERENCE>

THIS MESSAGE INDICATES THAT AN OBJECT PROGRAM HAS ATTEMPTED AN INPUT OR OUTPUT OPERATION ON DISK FILE FOR WHICH IT WAS NOT A VALID USER, AND THE OBJECT PROGRAM DID NOT SPECIFY ANY ACTION FOR SUCH A CONDITION. CONSEQUENTLY, PROCESSING OF THE PROGRAM WAS DISCONTINUED.

<FILE DESIGNATOR> SECURITY MAINT IGNORED

THIS MESSAGE INDICATES THAT AN ATTEMPT WAS MADE TO PERFORM SECURITY FILE MAINTENANCE ON A DISK FILE THAT WAS IN A STATE SUCH THAT THE REQUIRED MAINTENANCE COULD NOT BE COMPLETED.

<USER CODE> INVALID USER OF <PROGRAM SPECIFIER>

THIS MESSAGE INDICATES THAT AN ATTEMPT WAS MADE TO ACCESS A PROGRAM FILE BY A DESIGNATED USER WHO WAS NOT A VALID USER FOR THE FILE.

<FILE SPECIFIER> SECURED WITH <FILE SPECIFIER>

THIS MESSAGE IS TYPED AFTER THE MCP HAS PERFORMED AN OPERATION SPECIFIED BY A USE CONTROL CARD, PROVIDING THAT THE SECMSG OPTION IS SET.

<FILE SPECIFIER> RELEASED FROM <FILE SPECIFIER>

THIS MESSAGE IS TYPED AFTER THE MCP PERFORMED AN OPERATION SPECIFIED BY A LOCK CONTROL CARD, PROVIDING THAT THE SECMSG OPTION IS SET.

<FILE SPECIFIER> FREE FILE

THIS MESSAGE IS TYPED AFTER THE MCP HAS PERFORMED AN OPERATION SPECIFIED BY A FREE CONTROL CARD, PROVIDING THAT THE SECMSG OPTION IS SET.

<FILE SPECIFIER> PUBLIC FILE

THIS MESSAGE IS TYPED AFTER THE MCP HAS PERFORMED AN OPERATION SPECIFIED BY A PUBLIC CONTROL CARD, PROVIDING THAT THE SECMSG OPTION (OPTN 29) IS SET.

<FILE SPECIFIER> UNLOCK FILE

- FILE SECURITY -

THIS MESSAGE IS TYPED AFTER THE MCP HAS PERFORMED AN OPERATION SPECIFIED BY AN UNLOCK CONTROL CARD, PROVIDING THAT THE SECMSG OPTION IS SET.

FORMAT OF SECURITY FILE ENTRIES

THERE ARE TWO DIFFERENT TYPES OF ENTRIES IN A SECURITY FILE. THE FIRST TYPE CONSISTS OF ONE WORD AND IS USED TO SPECIFY A USER CODE, DELETED ENTRY, OR THE LAST ENTRY. THE SECOND TYPE CONSISTS OF TWO CONTIGUOUS WORDS AND IS USED FOR A PROGRAM SPECIFIER. THE TWO WORD ENTRY MUST NOT BE OVERFLOW A LOGICAL RECORD SO THAT THE <MFID> IS IN THE FIRST RECORD AND <FID> IS IN THE SECOND. THE FORMAT OF THE FILE IS AS FOLLOWS:

FIELD [0:6]	FIELD [6:42]=CONTENTS
@00	USER CODE OF A SECONDARY USER.
@20	USER CODE OF A TERTIARY USER.
@03	<MFID> OF A PROGRAM WHICH MAY ACCESS THE FILE FOR INPUT OR OUTPUT IN THE SAME MANNER AS A SECONDARY USER. (1ST WORD OF A 2 WORD ENTRY.)
@02	<MFID> OF A PROGRAM WHICH MAY ACCESS THE FILE FOR INPUT ONLY IN THE SAME MANNER AS A TERTIARY USER. (1ST WORD OF 2 WORD ENTRY.)
@00	@14=INDICATES A DELETED ENTRY.
@00	@114=INDICATES THE LAST ENTRY.

THE SECOND WORD OF A TWO WORD ENTRY HAS THE FOLLOWING FORMAT:

FIELD [0:46]	FIELD [6:42]=CONTENTS
@00	<FID> OF A PROGRAM SPECIFIER.

DISK FILE HEADER FORMAT - FILE SECURITY

THE OPERATING SYSTEM PLACES A SERIES OF ENTRIES IN THE DISK FILE HEADER RECORD WHICH INDICATE THE TYPE OF SECURITY FILE, THE USER CODE OF THE PRIVILEGED USER (IF ANY), AND THE <MFID> AND <FID> OF THE SECURITY FILE (IF ANY). THE FORMAT OF THE APPLICABLE PORTION OF THE DISK FILE HEADER RECORD IS AS FOLLOWS:

TYPE	HEADER[2]	HEADER[5]	HEADER[6]
FREE FILE	=0	[0:42]=0 [43:6]=@14	[0:42]=0 [43:6]=@14
INFO FILE	[1:1]=0 [6:42]=PRIMARY USER"S USER CODE	[0:42]=0 [43:6]=@14	[0:42]=0 [43:6=@14
PRIVATE FILE	[1:1]=0 [6:42]=PRIMARY USER"S USER CODE	[1:1]=1 [6:42]=<MFID> OF SECURITY FILE	[1:1]=0 [6:42]=<FID> OF SECURITY FILE
PUBLIC FILE	[1:1]=0 [6:42]=PRIMARY USER"S USER CODE	[0:42]=0 [43:6]=@14	=0
SECURITY FILE	[1:1]=1 [6:42]=PRIMARY USER"S USER CODE	=0	=0
SOLE USER FILE	[1:1]=0 [6:42]=PRIMARY USER"S USER CODE	=0	=0
UNLOCK FILE	[1:1]=0 [6:42]=PRIMARY USER"S USER CODE	[0:42]=0 [43:6]=@14	[0:42]=0 [43:6]=@14

REMOTE USER FILE SECURITY

USER CODE AND AUTHENTICATION CODE.

THE USER CODE IS USED TO VERIFY THAT A JOB IN THE SYSTEM HAS BEEN AUTHORIZED TO USE A PARTICULAR DISK FILE. CHECKING FOR AN AUTHORIZED USER IS DONE BY THE MCP AT FILE OPEN OR CLOSE TIME. A USER CODE MAY BE PRESENTED TO THE SYSTEM VIA A USER CONTROL CARD AT THE CENTRAL SITE CARDREADER, OR VIA THE LOG-IN PROCEDURE FOR REMOTE STATIONS. THE MCP CHECKS TO MAKE SURE THAT ONLY AUTHORIZED USERS ARE PERMITTED TO USE THE SYSTEM.

THE BASIC IDEA OF THE FILE SECURITY SYSTEM IS THAT THE CREATOR OF A FILE MUST SPECIFY WHICH USERS ARE AUTHORIZED TO USE HIS FILE. THIS IMPLIES THAT A PERSON MUST DIVULGE HIS USER CODE TO ANYONE WHO HAS CREATED A FILE HE WANTS TO USE. THEREFORE, ONE'S USER CODE CANNOT BE KEPT COMPLETELY CONFIDENTIAL. FOR THIS REASON A PERSON AT A REMOTE STATION MAY BE REQUIRED TO TYPE IN A LOG-IN MESSAGE WHICH CONTAINS HIS REMOTE-USER IDENTIFICATION CONSISTING OF THIS PARTICULAR AUTHENTICATION CODE AS WELL AS HIS USER CODE. THE AUTHENTICATION CODE WOULD THEN BE USED BY THE MCP TO VERIFY THAT THE PERSON ATTEMPTING TO GET ACCESS TO THE SYSTEM IS IN FACT AUTHORIZED TO USE THE SPECIFIED USER CODE. THIS AUTHENTICATION CODE NEED NEVER BE DIVULGED TO ANYONE ELSE AND CAN BE USED TO PROVIDE A SIGNIFICANT INCREASE IN THE SECURITY OF THE SYSTEM.

USER AND AUTHENTICATION CODES ARE RESTRICTED IN SIZE TO SEVEN ALPHANUMERIC CHARACTERS. ANY CODE LONGER THAN SEVEN CHARACTERS WILL BE TRUNCATED TO SEVEN CHARACTERS.

EXAMPLES

BOND
CHARLEY
GOLDFINGER
4UIWILL

LOG-IN PROCEDURE FOR DATA COMMUNICATIONS.

BEFORE A REMOTE USER CAN UTILIZE THE DCMCP SYSTEM CAPABILITIES, HE IS REQUIRED TO LOG-IN TO THE SYSTEM BY ENTERING AN "LI" MESSAGE. THIS MESSAGE WILL CONTAIN A REMOTE USER IDENTIFICATION CONSISTING OF AN IDENTIFICATION CODE AND, AT THE OPTION OF THE INSTALLATION, AN AUTHENTICATION CODE WHICH IS NOT EMPTY.

THE MCP WILL CHECK THE REMOTE USER IDENTIFICATION CONTAINED IN THE LOG-IN MESSAGE AGAINST THE FILE OF AUTHORIZED USER CODES CALLED REMOTE/USERS. IF THE CODE ENTERED IS NOT IN THE FILE THE USER WILL BE SO INFORMED BY THE MCP AND WILL BE PREVENTED FROM USING THE SYSTEM UNTIL A CORRECT LOG-IN MESSAGE IS TYPED IN. IF THE CODE ENTERED IS IN THE LIST OF AUTHORIZED USERS, THE MCP WILL LOG THE STATION IN, RECORD ITS LOG-IN TIME, AND CONSIDER THE STATION TO BE A VALID USER. ALSO, THIS USER'S STATION WILL BE CONSIDERED TO HAVE SPO CAPABILITIES.

A REMOTE STATION HAVING SPO CAPABILITIES CAN ENTER SYSTEM KEYBOARD INPUT MESSAGES, INCLUDING CONTROL CARD INFORMATION. IF A REMOTE USER IS PROPERLY LOGGED-IN, THREE MASKS WILL BE ASSIGNED TO HIS STATION. THESE MASKS CAN BE USED TO LIMIT BOTH THE CONTROL CARD AND KEYBOARD INPUT MESSAGES THAT THIS USER IS ALLOWED TO ENTER AT HIS STATION, IF THE MASKS HAVE BEEN ADDED TO THE "REMOTE/USERS" FILE.

IF NO LIST OF AUTHORIZED USERS HAS BEEN SUPPLIED BY THE PARTICULAR INSTALLATION, THEN A USER WILL BE LOGGED-IN REGARDLESS OF THE CODE ENTERED, BUT NO LOG-IN TIME WILL BE RECORDED. IN THIS CASE, THE STANDARD MASKS DEFINED IN THE MCP WILL BE USED.

THE LOG-IN MESSAGE HAS THE FOLLOWING FORM:

```
? LI <SEPARATOR> <USER CODE> <SEPARATOR> <AUTHENTICATION CODE>
```

EXAMPLES:

```
? LI : 007 : M
? LI  BY CHARLEY
```

REMOTE/USERS DISK FILE

A FILE CALLED REMOTE/USERS SHOULD BE MAINTAINED ON DISK TO CONTAIN THE LIST OF AUTHORIZED USERS. (THE ONLY VALID USERS OF THIS FILE MUST BE THE DESIGNATED PRIVILEGED USER AND THE MCP.) THIS FILE CAN BE CREATED AND MAINTAINED BY THE PRIVILEGED USER WITH A PROGRAM, AN EXAMPLE OF WHICH IS PROVIDED BY BURROUGHS. THIS PROGRAM IS CALLED UPDATE/USERS AND IS AVAILABLE ON COMPLETE SYSTEM RELEASES.

TO ENSURE THAT THE REMOTE/USERS FILE IS SECURED UNDER THE PROPER USER CODE, THE FOLLOWING CONTROL CARD SHOULD BE PRESENTED TO THE MCP WHENEVER A NEW REMOTE/USERS IS PLACED ON DISK:

```
CC USER = <PRIVILEGED USER CODE>; LOCK REMOTE/USERS; END.
```

BEFORE A NEW PRIVILEGED USER CODE WILL BECOME EFFECTIVE AFTER A NEW REMOTE/USERS FILE IS LOADED, A HALT/LOAD MUST BE PERFORMED.

A RECORD IN THE REMOTE/USERS FILE WILL CONTAIN AT LEAST THE FOLLOWING INFORMATION:

A. USER IDENTIFICATION CODE.
B. CONTROL CARD MASK.
C. INFORMATION KEYBOARD INPUT MESSAGE MASK.
D. MIX-RELATED KEYBOARD INPUT MESSAGE MASK.

AT THE OPTION OF EACH INSTALLATION, AN AUTHENTICATION CODE AND/OR COMMENTS CAN ALSO BE INCLUDED IN A RECORD. THE RECORD SIZE MUST BE CONSTANT THROUGHOUT THE FILE AND MUST BE EITHER 6, 10, 15, OR 30 WORDS. RECORDS THAT ARE DELETED FROM THE FILE MUST HAVE A USER CODE OF @14. THE LAST ENTRY MUST HAVE A USER CODE OF @114.

IF AN INSTALLATION IS USING THE REMOTE JOB ENTRY FEATURE, THE FIRST ENTRIES IN THE REMOTE/USERS FILE WILL BE AS FOLLOWS:

ENTRY 0 FOR EITHER NORMAL OR RJE

WORD	CONTENTS
0	MCP NAME
1	CCMASK1 FOR USER
2	CCMASK2 FOR USER
3	INFOMASK1 FOR USER
4	INFOMASK2 FOR USER
5	MIXMASK FOR USER
	(THE FOLLOWING ENTRIES ARE OPTIONAL)
6	IF LSS 0, THEN AUTHENTICATION CODE REQUIRED

	IF GTR 0, THEN AVAILABLE TO USER
7-TO-RECORD-SIZE	AVAILABLE TO USER

ENTRY 1

WORD	CONTENTS
0	IF = @214 THEN RJE LIST OTHERWISE START OF NORMAL ENTRIES
1-RECSIZE	IF GTR 1, WILL CONTAIN O&TU (RJE LINE [9:4]) &BUF (RJE LINE [14:4])

ANY ADDITIONAL ENTRIES, WHICH HAVE WORD 0 = @214 BEFORE A NORMAL ENTRY IS ENCOUNTERED, ARE RJE LINES.

THE NORMAL USERCODE LIST HAS THE FOLLOWING FORMAT:

FOR VALID ENTRIES

WORD	CONTENTS
0	USER IDENTIFICATION CODE
1	CCMASK1 FOR USER
2	CCMASK2 FOR USER
3	INFOMASK1 FOR USER
4	INFOMASK2 FOR USER
5	MIXMASK FOR USER (THE FOLLOWING ENTRIES ARE OPTIONAL)
6	IF LSS 0, THEN AUTHENICATION CODE REQUIRED IF GTR 0, THEN AVAILABLE TO USER
7-TO-RECORD SIZE	AVAILABLE TO USER

FOR DELETED ENTRIES:

WORD	CONTENTS
0	@14
1-TO-RECORD SIZE	IRRELEVANT

FOR LAST RECORD IN FILE:

WORD	CONTENTS

- FILE SECURITY -

0	@114
1-TO-RECORD SIZE	IRRELEVANT

CCMASK1, CCMASK2, MIXMASK, AND INFOMASK

ARE USED TO CHECK THE VALIDITY OF A CONTROL CARD ENTERED VIA DATA COMMUNICATIONS. THEY ARE USED IF A SPECIAL MASK IS NOT PROVIDED BY REMOTE/USES. THE CCMASK TABLE SHOWS THE CONTROL CARD WORDS CORRESPONDING TO THE BITS IN REMOTE/USERS, 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 PRECEDINGTHEM. INFOMASK1 AND INFOMASK2 ARE USED TO CHECK THE VALIDITY OF THOSE INPUT MESSAGES WHICH DO NOT INCLUDE A MIX INDEX. THE THREE MASK WORDS 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	UNLOCK	23	NO
25	USE	24	NO
26	LOCK	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	SET	42	NO
44	RESET	43	NO
45	NOT USED	44	---
46	NOT USED	45	---
47	NOT USED	46	---
48	FILE	47	YES

COLUMN	WORD	CCMASK2 BIT	STANDARD MASK
49	EXPIRED	0	NO
50	ACCESSD	1	NO
51	PROCESS	2	YES
52	IC	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	TSPOL	15	YES
65	BASIC	16	YES
66	COBOL68	17	YES
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
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
32	BK	31	YES
33	AU	32	NO

THE FORMAT OF THE INFOMASK1 CARD IS AS FOLLOWS:

COLUMN	WORD	INFOMASK1 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	SC	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
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	INFOMASK2 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
65	XD	16	NO
66	SY	17	NO
67	SL	18	NO
68	SI	19	NO
69	AU	20	NO
70	DC	21	NO

FILECLOSE

FILECLOSE(A) IS A PROCEDURE WHICH CAUSES A FILE WITH TOP I/O DESCRIPTOR "A" TO BE CLOSED BY DOING A CALL ON REALFILECLOSE(A).

FILEOPEN

FILEOPEN(XTRA,ALPHA) IS A PROCEDURE WHICH CAUSES A FILE WITH TOP I/O DESCRIPTOR "ALPHA" TO BE OPENED.

FINALQUE

DESCRIPTOR POINTING TO THE FINALQUE ARRAY, WHICH CONTAINS A SKELETON DESCRIPTOR USED TO RETURN THE RESULT DESCRIPTOR TO THE CALLER.

FINDINPUT

FINDINPUT(MID,FID,REEL,CDATE,CYCLE,COBOL,UL,OF,MODE,FN) IS A PROCEDURE WHICH LOCATES THE INPUT FILE INDICATED BY THE PARAMETERS. "UL" IS TRUE FOR UNLABELED FILES AND "OF" IS TRUE FOR OPTIONAL FILES (COBOL). THE PROCEDURE IS TYPED REAL AND RETURNS THE LOGICAL UNIT NUMBER.

FINDOUTPUT

FINDOUTPUT(MID,FID,REEL,CDATE,CYCLE,TYPE,FORMS,KIND) IS A PROCEDURE WHICH LOCATES THE OUTPUT DEVICE TO BE USED FOR THE SPECIFIED FILE. "TYPE" AND "KIND" ARE INTERNAL CODES FOR UNIT TYPES. THE PROCEDURE IS TYPES REAL AND RETURNS THE LOGICAL UNIT NUMBER AND, FOR DISK FILES, THE HEADER.

FIRSTDECK

POINTS TO THE HEADER OF THE FIRST PSEUDO DECK IN THE QUEUE.

FIRSTWAIT

POINTER AT NEXT UNIT TO BE ACTIVATED WHEN AN I/O CHANNEL BECOMES AVAILABLE. THIS IS USED IN CONJUNCTION WITH THE WAITQUE.

FORGETESPDISK

FORGETESPDISK(SEGMENT) IS A PROCEDURE WHICH RETURNS AN ESP DISK SEGMENT TO THE AVAILABLE DISK TABLE.

FORGETSPACE

FORGETSPACE(LOC) IS A PROCEDURE WHICH RETURNS STORAGE AT "LOC"-2 TO THE AVAILABLE CHAIN. LINKS IN FRONT AND BEHIND THE AREA ARE AUTOMATICALLY CHECKED FOR VALIDITY. ERROR CONDITIONS MAY RESULT IN AN "INVALID LINK" EVEN THOUGH THE MEMORY DUMP INDICATES NO "BAD LINKS". AVAILABLE AREAS ARE CONSOLIDATED SO THAT 2 CONSECUTIVE AVAILABLE AREAS NEVER OCCUR.

FORGETUSERDISK

FORGETUSERDISK(A,N) IS A PROCEDURE WHICH RETURNS USER DISK TO THE AVAILABLE DISK TABLE. "A" IS THE ABSOLUTE DISK SEGMENT ADDRESS OF AN AREA "N" SEGMENTS LONG. "N" HAS THE ADDITIONAL MEANING:

```
IF N LSS 0 THEN MAKE A SCRATCHDIRECTORY DELETION.
IF N GTR 0 THEN  DON"T MAKE A SCRATCHDIRECTORY DELETION.
IF N  =  0 THEN IMMEDIATELY GO AWAY.
```

FORMESS

FORMESS(BUFF,H) IS A PROCEDURE WHERE "BUFF" CONTAINS THE UNIT MNEMONIC. IF THE UNIT IS IN USE, THE CORRESPONDING MESSAGE IS SPOUTED. IF THE UNIT IS NOT IN USE, LABELTABLE[U] IS SET TO -@114 AND MULTITABLE[U] IS SET TO ZERO. THEN, IF "H" IS FALSE, THE BIT FOR THE UNIT IS TURNED OFF IN "READY", "RRRMECH", AND "SAVEWORD". IF "H" IS TRUE, THEN THE BIT FOR THE UNIT IS TURNED ON IN THE THREE WORDS.

FS

FS IS THE FILE SECURITY ARRAY WHICH IS FOUR WORDS LONG AND IS UTILIZED FOR DISK FILE SECURITY. THE ARRAY IS INDEXED BY MIX AND FNUM DIV 5. TWO BITS ARE UTILIZED FOR EACH DISK FILE.

[42:6] = DISK FILE 1 - FIRST FILE
[44:2] = DISK FILE 2

THESE TWO BITS CONTAIN THE NOT FUNCTION OF HEADER[CF].[31:2]. THREE PROCEDURES ACCESS THE ARRAY, SECURITYCHECK, PROGRAMRELEASE, AND REOPEN. SECURITYCHECK RETURNS THE FOLLOWING VALUES:

0 = NO USER
2 = INPUT
3 = I/O
7 = I/O LIBMAIN DISK

PROGRAMRELEASE USES THIS ARRAY TO DETERMINE IF I/O SHOULD BE DONE. IF THE FILE STATUS INHIBITS WRITING, THE PROGRAM RELEASE IS NOT PERFORMED.

REOPEN UTILIZES THE ARRAY TO VERIFY THAT THE DISK FILE IN QUESTION HAS THE SAME SECURITY AS THAT WHICH EXISTED WHEN THE PROGRAM WAS ORIGINALLY RUN.

GET

"GET" IS A PROCEDURE WHICH MANIPULATES THE STATION ARRAY. "GET" IS CALLED WITH THE TU/BUFF NUMBER IN THE [9:9] FIELD OF ITS PARAMETER AND RETURNS THE CORRESPONDING STATION ARRAY ENTRY, WHICH IS COMMONLY KNOWN AS A STATUS WORD.

GETESPDISK

GETESPDISK IS A PROCEDURE WHICH OBTAINS A FREE SEGMENT OF DISK BETWEEN THE "KERNEL" AND THE "ABORT TABLE".

GETSPACE

GETSPACE(SIZE,TYPE,SAVEF) IS A PROCEDURE WHICH OBTAINS STORAGE OF "SIZE" ACTUAL WORDS, PLUS 2 WORDS FOR THE LINK, AND RETURNS THE ADDRESS OF THE FIRST LINK WORD, WHICH IS MARKED WITH THE "TYPE" OF STORAGE, THE CURRENT VALUE OF P1MIX, AND THE "SAVEF" FUNCTION. THE SECOND LINK WORD IS ZEROED, BUT NOT THE REST OF THE AREA.

GETUSERDISK

GETUSERDISK IS DEFINED TO BE A CALL ON THE TYPED PROCEDURE PETUSERDISK(N,T). "N" IS THE NUMBER OF SEGMENTS REQUESTED AND "T" IS THE EU NUMBER OR SPEED NUMBER. PETUSERDISK WILL REUTRN "-1", "0" OR THE ABSOLUTE DISK SEGMENT ADDRESS OF THE RESULTANT AREA. SEE T.[2:1] FOR THE "-1" AND N.[22:1] FOR THE "0".

T GTR 0	T IS A PREFERRED SPEED #: T=1,2,3,4...OR 31.
T LSS 0	-T IS A PREFERRED EU #: T=-1,-2,-3,-4...OR -20.
T EQL 0	DON"T CARE ABOUT SPEED # OR EU #. USE EU WITH LEAST EU I/O.

T.[2:1]	0 = IF CAN"T GET PREFERRED SPEED # OR EU #, TREAT AS T = 0 (ABOVE).
	1 = IF CAN"T GET PREFERRED SPEED # OR EU # , RETURN A "-1".

N GTR 0	MAKE A SCRATCHDIRECTORY ENTRY.
N LSS 0	DON"T MAKE A SCRATCHDIRECTORY ENTRY.
N EQL 0	IMMEDIATELY RETURN WITH A "0".
N.[2:1]	0 = IF CAN"T FIND ANY USER DISK AND T.[2:1]=0, NO-USER-DISK. 1 = IF CAN"T FIND ANY USER DISK, AND T.[2:1] = 0, RETURN "0".

GRSD

DESCRIPTOR POINTING TO GRSD ARRAY, WHICH IS USED FOR SEGMENT DICTIONARY INFORMATION ON BREAKOUT AND FOR MEMORY LINK INFORMATION ON RESTART.

[1:1]	EI
[8:5]	OLD UNIT + 1 (AFTER REELER ONLY)
[13:5]	NEW UNIT + 1 (AFTER REELER ONLY)
[18:7]	BREAK NUMBER I.E., 00, 01, ETC.
[25:8]	SELECTION FILE COUNT
[33:15]	OVERLAY DISK ADDRESS OF RSD, THE RESTART SEGMENT DICTIONARY.

HOLDER

CONTAINS INFORMATION ABOUT HOLDLIST.
.[18:15] = SIZE OF HOLDLIST.
.[33:15] = DISK ADDRESS OF HOLDLIST.

HOLDLIST

THE HOLDLIST CONTAINS A ONE WORD ENTRY FOR EACH PROCESS WHICH HAS BEEN SUSPENDED TO WAIT FOR ACCESS TO A FILE WHICH IS CURRENTLY IN USE. THE CONTENTS OF THIS WORD ARE:

[2:2]	SYSTEM NUMBER OF SUSPENDED PROCESS
[10:8]	MIX NUMBER OF SUSPENDED PROCESS (TSS ONLY)
[18:15]	DISK ADDRESS OF FILE HEADER
[33:15]	ADDRESS OF A WORD WHICH IS SET TO 1 WHEN THE FILE BECOMES AVAILABLE.

ILL

THE HEAD OF THE QUEUE THROUGH WHICH ALL DATA COMMUNICATION OUTPUT PASSES.

INDENPENDENTRUNNER

THE PROCEDURE DECLARATION FOR "INDENPENDENTRUNNER" IS AS FOLLOWS:

```
SAVE PROCEDURE INDEPENDENTRUNNER(ROUTINE,PARAMETER);

     VALUE ROUTINE,PARAMETER;
     ARRAY PARAMETER[*];
     REAL  ROUTINE;
```

THE INTERPRETATION OF THE PARAMETERS IS SELF-EXPLANATORY.

THE OUTER BLOCK MCP CODE WHICH ACTIVATES THE INDEPENDENT ROUTINES PLACED IN THE SLATE BY "INDEPENDENTRUNNER" IS AS FOLLOWS:

```
 IF NSLATE NEQ LSLATE THEN % SOMETHING IN THE STACK
 IF STACKUSE THEN % NOBODY USING THE INDEPENDENT STACK
 BEGIN TOGLE:=TOGLE AND NOT STACKMASK;%MARK STACK IN USE
      P(ISTACK,STS);% SET "S" TO BASE OF STACK
      P(0,STF);% SET "F" TO ZERO FOR OLAY ROUTINE-S USE
      NSLATE:=NSLATE+2 AND SLATEND;% ADVANCE NSLATE CIRCULARLY
                                   % (SLATEND=POWER OF 2 - 1)
      SECONDCTR:=0;
      NT4:=SLATE[NSLATE+1];% ESPBIT USES NT4 FOR ABSENT ROUTINES
      P(MKS,NT4,DIB 0,LOD,SLATE[NSLATE],COC);% CALL RUNNER
      GO TO NOTHINGTODO;% AFTER RUNNER FINISHES, GO TO
                        % NOTHINGTODO
 END;
```

TO PREVENT INTERLOCKING THE INDEPENDENT STACK "ISTACK" FOR LONG PERIODS OF TIME, A NON-TRIVIAL ROUTINE WHICH IS ALWAYS CALLED INDEPENDENTLY MUST CAUSE ITS STACK TO BE MOVED. THIS IS MOST ALWAYS DONE IN THE FOLLOWING MANNER:

THE PROCEDURE "DKBUSINESS" HANDLES VARIOUS SPECIAL-PURPOSE FUNCTIONS RELATED TO THE DISK DIRECTORY. IT IS CALLED INDEPENDENTLY FROM "KEYIN" AND "COMMUNICATE" WITH THE PARAMETER INDICATING WHAT IS TO BE DONE. A SAMPLE CALL IS:

```
INDEPENDENTRUNNER(P(,DKBUSINESS),0);
```

DKBUSINESS IS SET UP TO MOVE ITS STACK. THE PROCEDURE DECLARATION AND FRAGMENTS OF THE ROUTINE ARE AS FOLLOWS:

```
      PROCEDURE DKBUSINESS(BUFF); VALUE BUFF; REAL BUFF;

      BEGIN
           REAL RCW=+10,              % NOTE MANNER OF DECLARING
                MID=RCW+1,            % LOCAL VARIABLES AS
                FID=MID+1,            % BEING SPECIFICALLY (F+)-
                A=FID+1,              % RELATIVE SO THAT ESPOL WILL
                B=A+1;                % NOT GENERATE LIT 0-S FOR EACH

      P(GETSPACE(128,12,0)+1,STS,BUFF,RCW,0,RDS,0,XCH,CFX,STF);
      STACKUSE:=TRUE;
      P(0,0,0,BUFF,DUP); BUFF:=P.[15:15]-1; P(0,0);
```

THE FIRST LINE OF CODE GOT SPACE FOR THE NEW STACK, MOVED ITS PARAMETER AND RETURN CONTROL WORD TO THE NEW STACK, AND POINTED THE "S" AND "F" REGISTERS TO THE NEW STACK.

THE NEXT LINE RELEASED THE INDEPENDENT STACK.

THE THIRD LINE INITIALIZED AND GOT STACK SPACE FOR ALL LOCAL PARAMETERS.

```
      BUFF:=0; KILL([BUFF]); END % OF DKBUSINESS
```

THE PRECEDEING LINE OF CODE CLEARS PARAMETER TO ZERO TO AVOID ANY UNNECESSARY SIDE EFFECTS, THEN CALLS KILL PASSING AS PARAMETER A DESCRIPTOR CALL ON THE PARAMETER OF THE INDEPENDENT RUNNER. KILL SETS ITS "F" REGISTER TO @100 AND CALLS "FORGETSPACE" ON THE (ADDRESS IN ITS PARAMETER) -2 TO RETURN THE INDEPENDENT STACK. IT THEN BRANCHES TO NOTHINGTODO.

INFOMASK1

MASKS FOR DEFAULT ALLOWABLE KEYBOARD INPUT MESSAGES NOT REQUIRING A MIX INDEX FROM REMOTES.

WORD	INFOMASK1 BIT	STANDARD MASK
NOT USED	0	---
PG	1	NO
MX	2	YES
DD	3	NO
RW	4	NO
PD	5	YES
DB	6	NO
DP	7	NO
DT	8	NO
DS	9	NO
PT	10	NO
RS	11	NO
EI	12	YES
CC	13	YES
PB	14	NO
RY	15	NO
TR	16	NO
OL	17	YES
LN	18	NO
WD	19	YES
WT	20	YES
LR	21	NO
RO	22	NO
SC	23	NO
TO	24	YES
SV	25	NO
LD	26	NO
CD	27	YES
RD	28	NO
RN	29	NO
ED	30	NO
CI	31	NO
TF	32	YES
SF	33	NO
TS	34	YES
RR	35	NO
QV	36	NO
EX	37	YES
PI	38	YES

- INFOMASK1 -

LO	39	YES
LI	40	YES
SS	41	YES
SM	42	YES
HM	43	YES
TC	44	YES
ZZ	45	YES
BO	46	YES
WP	47	NO

INFOMASK2

MASKS FOR DEFAULT ALLOWABLE KEYBOARD INPUT MESSAGES NOT REQUIRING A MIX INDEX FROM REMOTES.

WORD	INFOMASK2 BIT	STANDARD MASK
NOT USED	0	---
WU	1	NO
LF	2	NO
LC	3	NO
LS	4	NO
XI	5	NO
WR	6	NO
WM	7	YES
BK	8	NO
BS	9	NO
US	10	NO
SC	11	NO
CL	12	NO
QT	13	NO
WI	14	NO
CU	15	NO
XD	16	NO
SY	17	NO
SL	18	NO
SI	19	NO
AU	20	NO
OC	21	NO

INITIALIZE

INITIALIZE IS A PROCEDURE WHICH SETS UP THE SYSTEM AT HALT/LOAD TIME. IN ADDITION TO INITIALIZING MASKS AND CONSTANTS, INITIALIZE HAS THE TASK OF ALLOCATING MCP TABLES (USING FIX), AND BUILDING THE AVAILABLE DISK TABLE UTILIZING THE INFORMATION IN THE MAIN DISK DIRECTORY. THE HALT/LOAD MESSAGE IS FORMATTED HERE, AND THE SYSTEM BEGINS BY INTERROGATING ITS PERIPHERAL UNITS.

INTERRUPT

INTERRUPT(TYPE) IS A PROCEDURE WHICH ANALYZES A PROCESSOR-DEPENDENT INTERRUPT, AND TAKES APPROPRIATE ACTION FOR "TYPE" (USUALLY TERMINATION).

INQCT

COUNTER OF UNPROCESSED DATA COMMUNICATIONS INTERRUPTS.

INQUIRY

ARRAY DCB[16] AND THE ORR WORD

"DCB IS A TABLE USED BY THE DATA COMMUNICATION 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]". THEN "CURRINQ" IS 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, IT IS "OUTPUT POSSIBLE", BUT NOT "OUTPUT READY", IF IT IS HANDLING ONE LINE OF OUTPUT AND WILL BE COMING BACK FOR ANOTHER.) 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.

INTABLE

DESCRIPTOR POINTING TO THE INTABLE ARRAY, WHICH CONTAINS INFORMATION CONCERNING USE OF INTRINSICS BY MIX INDEX.

INTER-PROGRAM COMMUNICATION
(TASKING, EVENTS, LOCKS, AND SOFTWARE INTERRUPTS)

AN OBJECT PROGRAM WHICH EITHER CONTAINS OR IS INVOKED BY A PROCESS, CALL, OR RUN/EXECUTE STATEMENT, OR MANIPULATES LOCKS (COBOL) OR EVENTS (ALGOL), WILL BE FLAGGED IN SEGMENT ZERO (WORD 2 [3:1]=1) OF ITS CODE FILE AS HAVING A TASK ARRAY.

THE FORMAT OF THE TASK ARRAY (MYSELF AT PRT @26) IS AS FOLLOWS:

WORD	DESCRIPTION
TSKA[0]	= TASKVALUE: PROVIDED FOR USER
TSKA[1]	= 7 CHR <MFID> OF CODE FILE
TSKA[2]	= 7 CHR <FID> OF CODE FILE
TSKA[3]	= STATUS: 1 = SCHEDULED 2 = ACTIVE -1= TERMINATED (DS-ED OR EOJ) -2= INITIATION ATTEMPTED BUT FAILED
TSKA[4]	= STACKNO: MIX INDEX IF RUNNING SCHEDULE-ID IF SCHEDULED
TSKA[5]	HEAD OF LIST OF LOCK-ITEMS IN CONTROL OR QUEUED
TSKA[6]	= TYPE: 0 = ASYNCHRONOUS DEPENDENT (PROCESS) 1 = SYNCHRONOUS DEPENDENT (CALL) 2 = INDEPENDENT (RUN OR EXECUTE)
TSKA[7]	CALL STATE 0 = INITIAL 1 = EXIT PROGRAM - EXIT PROGRAM RETURN HERE 2 = CONTINUED OR RE-CALLED
TSKA[8]	[1:1]=1 IF JUST EXECUTED INTERRUPTER INTRINSIC AND SFINTQ IS NON-EMPTY [2:1]=1 IF SFINTQ IS NON-EMPTY [3:1]=1 IF INTERRUPTER INTRINSIC IS RUNNING [4:1]=1 SFINTQ INTERLOCK BIT (ON TO START) [FF] = ABSOLUTE ADDRESS OF OLD IRCW [CF] = HEAD OF LIST OF DECLARED INTERRUPTS

SEGMENT 0 FOR IPC PROGRAM FILES

WORD	DESCRIPTION
S[2].[2:1]	=1 IF THERE ARE DECLARED INTERRUPTS
S[2].[3:1]	=1 FOR AN IPC PROGRAM FILE (EITHER FOR INVOKING OR INVOKED)

S[2].[4:1] =1 FOR AN INVOKED IPC PROGRAM FILE

NOTE: S[2].[2:3] = JAR[2].[5:3].

JAR[2].[6:1]=1 INDICATES TO COMS THAT THIS JOB MAY HAVE DEPENDENT TASK DESCENDENTS TO BE DS-ED OR ES-ED AND LOCK QUEUES TO BE CLEANED UP WHEN IT TERMINATES.

S[8] NUMBER OF TASK PARAMETERS TO BE RECEIVED (= N BELOW)

S[9] DISK ADDRESS OF PARAMETER DESCRIPTION SEGMENT

FORMAT OF ENTRY IN PARAMETER DESCRIPTION SEGMENT (BEGINNING IN WORD 1)

[18:15] : TYPE =
- 0 = TASK ARRAY -NAME
- 1 = EVENT-LOCK -NAME
- 2 = PRT CELL -NAME
- 3 = PRT CELL -VALUE
- 4 = (SAVE) ARRAY -NAME
- 5 = ARRAY -VALUE

(ONLY 1-DIMENSIONAL ARRAYS CAN BE PASSED AS TASK PARAMETERS).

[8:10] SIZE OF ARRAY FOR TYPES 4 AND 5, ELSE 0.

[33:15] PRT LOCATION FOR TYPES 0-4, FOR TYPE 5: RELATIVE DISK ADDRESS OF TYPE-2 SEGMENT.

FORMAT OF INTERRUPT (IN PRT)

UPPER WORD (LINK WORD)

[1:1] 1 = IF INTERRUPT IS DISALLOWED.

[FF] ABSOLUTE ADDRESS OF NEXT INTERRUPTON EVENTS ATTACH LIST OR OF THE EVENT IF THIS INTERRUPT IS THE LAST ON THE LIST.

[CF] RELATIVE PRT ADDRESS OF NEXT DECLARED INTERRUPT FOR THIS PROCESS.

LOWER WORD: PROCEDURE DESCRIPTOR FOR INTERRUPT.

FORMAT OF EVENT (IN OBJECT PROGRAM-S PRT)

"ORIGINAL" EVENT-ITEM (AT ABSOLUTE ADDR ABSEVT):

[1:1] EVENT INTERLOCK BIT (ON TO START)

[5:1] 1 = DISTINGUISHES THE EVENT FROM ATTACHED INTERRUPTS

[18:15] ABSOLUTE ADDRESS OF FIRST INTERRUPT ON ATTACH LIST

[47:1] HAPPEN BIT

"COPY" EVENT-ITEM (RECEIVED AS A PARAMETER):

[33:15] ABSOLUTE ADDRESS OF ORIGINAL EVENT

FORMAT OF LOCK (IN OBJECT PROGRAM-S PRT)

[1:1] 1 = LOCKED(LOCK BIT, ORIGINAL ONLY)

[2:1] 1 = IN CONTROL (CONTROL BIT)

[3:1] 1 = ORIGINAL LOCK-ITEM (ORIGINAL BIT)
0 = A COPY

[4:1] QUEUE INTERLOCK (ORIGINAL ONLY)

[8:10] MIX INDEX OF PROGRAM IN CONTROL (ORIGINAL). ALSO RELATIVE PRT ADDRESS USED TO LINK ALL LOCK-ITEMS IN CONTROL OR IN WAIT QUEUES (COPY).

[18:15] POINTER TO NEXT PROCESS IN WAIT QUEUE, ELSE 0

[33:15] POINTER TO HEAD OF QUEUE (ORIGINAL). ALSO POINTER TO ORIGINAL LOCK-ITEM (COPY).

INTERRUPT HANDLING

INTRODUCTION

AS A PROGRAM PROCESSES AFTER BEING INITIATED, IT MAY SOON REQUIRE ADDITIONAL PROGRAM SEGMENTS AND/OR DATA WHICH WERE NOT PROVIDED BY THE SELECTION PROCEDURE. THE PRINCIPAL SOURCE OF INFORMATION FOR A PROGRAM IS ITS PRT. ALL SIMPLE VARIABLES, OTHER THAN THOSE DECLARED LOCAL TO PROCEDURES, HAVE PRT LOCATIONS; THEREFORE, THEY ARE ALWAYS PRESENT IN CORE WHILE A PROGRAM IS PROCESSING. PROGRAM SEGMENTS AND DATA SEMENTS ARE NOT ALWAYS PRESENT. HOWEVER, EACH PROGRAM SEGMENT AND DATA SEGMENT HAS A DESCRIPTOR RELATED TO IT; EITHER A PROGRAM DESCRIPTOR (E.G., LABEL DESCRIPTOR, PROCEDURE DESCRIPTOR) OR A DATA DESCRIPTOR. THESE DESCRIPTORS ARE LOCATED IN THE PRT. WHEN A PROGRAM ACCESSES A DESCRIPTOR, THE PRESENCE BIT OF THE DESCRIPTOR WILL, OR COURSE, DENOTE THE PRESENCE OR ABSENCE OF THE INFORMATION DESCRIBED. IF A DESCRIPTOR IS ACCESSED AND ITS PRESENCE BIT IS ZERO, THE PRESENCE BIT INTERRUPT WILL BE SET, CONTROL WORDS WILL BE GENERATED AND PUT IN PLACE, AND SUBSEQUENTLY CONTROL IN PROCESSOR 1 WILL BE TRANSFERRED TO THE PRESENCE BIT INTERRUPT LOCATION.

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. WHEN THIS SITUATION OCCURS, THE CONTROL WORDS FOR THE INTERRUPT CONTAINS SETTINGS FOR RC AND RL THAT ADDRESS THE SYLLABLE FOLLOWING THE SYLLABLE THAT CAUSED THE INTERRUPT.

TO INVESTIGATE THE INTERRUPT CONDITION, THE PRESENCE BIT ROUTINE FIRST LOCATES THE PRT OF THE INTERRUPTED PROGRAM THROUGH USE OF THE PRT ARRAY AND P1MIX. THE INITIATE CONTROL WORD FROM THE PRT IS THEN USED TO LOCATE THE OTHER CONTROL WORDS AT THE TOP OF THE PROGRAM-S STACK. THROUGH USE OF THE REGISTER SETTINGS IN THE CONTROL WORDS, PRESENCE BIT LOCATES THE SYLLABLE THAT CAUSED THE INTERRUPT AND, SUBSEQUENTLY, THE ADDRESS OF THE DESCRIPTOR WITH A ZERO PRESENCE BIT.

TO REMEDY THE SITUATION CAUSED BY THE DESCRIPTOR WITH A ZERO PRESENCE BIT, THE PRESENCE BIT ROUTINE FIRST ADJUSTS THE REGISTER SETTINGS IN THE CONTROL WORDS SO THEY WILL REFLECT THE CONDITION THAT EXISTED BEFORE THE SYLLABLE THAT CAUSED THE INTERRUPT WAS EXECUTED. THEN THE INFORMATION REQUIRED BY THE PROGRAM IS MADE PRESENT.

METHODS OF MAKING INFORMATION PRESENT

THE OCCURRENCE OF A PRESENCE BIT INTERRUPT ONLY INDICATES THAT A PROGRAM REQUIRES INFORMATION. THE KIND OF INFORMATION REQUIRED IS INDICATED BY THE DESCRIPTOR THAT WAS ACCESSED. THE METHOD IN WHICH THE INFORMATION IS PROVIDED IS DETERMINED BY THE KIND OF INFORMATION REQUIRED.

MAKING THE DATA PRESENT FOR THE FIRST TIME.

WHEN A DATA DESCRIPTOR WITH A ZERO PRESENCE BIT IS ACCESSED FOR THE FIRST TIME, IT CONTAINS A CODE NUMBER IN BITS [33:15] - THE ADDRESS FIELD. THE FACT THAT THIS CODE IS PRESENT DENOTES THAT NO DATA HAS YET BEEN RELATED TO THE DESCRIPTOR. WHEN THIS IS SO, AN AREA IN CORE MUST BE PROVIDED AND ASSIGNED TO THE DESCRIPTOR. THE SIZE FIELD IN THE ACCESSED DESCRIPTOR SPECIFIES THE KIND OF STORAGE. (E. G., IF THE CODE IS 0 OR 2, OVERLAYABLE STORAGE WILL BE ASSIGNED; IF THE CODE IS 1, NON-OVERLAYABLE STORAGE WILL BE ASSIGNED.) TO OBTAIN THE REQUIRED AREA IN CORE, THE PRESENCEBIT ROUTINE CALLS THE GETSPACE PROCEDURE PASSING PARAMETERS THAT SPECIFY: (1) THE SIZE OF THE AREA REQUIRED (2) THAT THE AREA IS FOR DATA, AND (3) THE CODE INDICATING IF THE AREA IS OVERLAYABLE OR NON-OVERLAYABLE. GETSPACE PROVIDES THE AREA AND RETURNS ITS ADDRESS TO THE PRESENCEBIT ROUTINE. PRESENCEBIT THEN INITIALIZES THE ASSIGNED AREA WITH ZEROES, IF THE CODE IS 0 OR 1, OR ONES, IF THE CODE IS 2, PLACES THE ADDRESS OF THE AREA-S DESCRIPTOR IN THE AREA-S MEMORY LINK FOR USE IF THE AREA IS AT SOME TIME OVERLAID, PLACES THE ADDRESS OF THE AREA IN THE DATA DESCRIPTOR, AND SETS THE DESCRIPTOR-S PRESENCE BIT TO 1.

MAKING OVERLAID DATA PRESENT.

WHEN A DATA DESCRIPTOR WITH A ZERO PRESENCE BIT IS ACCESSED, ITS ADDRESS FIELD MAY CONTAIN AN OVERLAY STORAGE ADDRESS WHICH INDICATES THAT THE DESIRED INFORMATION HAS BEEN OVERLAID AND MUST BE READ IN FROM THE OVERLAY STORAGE AREA. AS IN THE ABOVE CASE, PRESENCEBIT MUST CALL GETSPACE TO OBTAIN AN AREA INTO WHICH THE DESIRED INFORMATION CAN BE READ. THEN THE ADDRESS OF THE AREA-S DATA DESCRIPTOR IS PLACED IN THE AREA-S MEMORY LINK FOR USE IF THE AREA IS EVER OVERLAID, THE ADDRESS OF THE AREA IS PLACED IN THE AREA-S DESCRIPTOR, AND THE DESCRIPTOR-S PRESENCE BIT IS SET TO 1. PRESENCEBIT THEN CALLS ON THE DISKIO PROCEDURE WHICH INITIATES THE INPUT OPERATION FROM THE SPECIFIED ADDRESS AND RETURNS CONTROL TO THE PRESENCEBIT ROUTINE. PRESENSEBIT THEN MARKS THE OVERLAY DISK AREA AVAILABLE. . AT THIS POINT, PRESENCEBIT CANNOT CONTINUE UNTIL THE INPUT OPERATION IS COMPLETE. THEREFORE, PRESENCEBIT CALLS THE SLEEP PROCEDURE SPECIFYING NOT TO RETURN CONTROL UNTIL THE I/O OPERATION IS COMPLETED. THE CONTROL SECTION OF THE MCP THEN TAKES CONTROL AND PRESENCEBIT IS TERMPORARILY SUSPENDED. WHEN CONTROL IS

RETURNED TO PRESENCEBIT, THE DESIRED INFORMATION IS IN THE AREA NOW ADDRESSED BY THE DESCRIPTOR.

MAKING PROGRAM SEGMENTS PRESENT.

WHEN A PROGRAM DESCRIPTOR WITH A ZERO PRESENCE BIT IS ACCESSED, THE DESIRED PROGRAM SEGMENT MUST BE READ IN FROM DISK. PROGRAM DESCRIPTORS, HOWEVER, MAY ADDRESS POINTS WITHIN A PROGRAM SEGMENT (E. G., A LABEL DESCRIPTOR ADDRESSES A POINT IN A SEGMENT TO WHICH CONTROL MAY BE TRANSFERRED). CONSEQUENTLY, THE ADDRESS FIELD OF A PROGRAM DESCRIPTOR WITH A ZERO PRESENCE BIT CONTAINS THE RELATIVE ADDRESS OF THE WORD WITHIN THE SEGMENT THAT THE DESCRIPTOR MUST ADDRESS. TO DETERMINE A SEGMENT-S DISK ADDRESS, PRESENCEBIT MUST EXAMINE THE SEGMENT-S ENTRY IN THE PROGRAM-S SEGMENT DICTIONARY. THEREFORE, THE PROGRAM DESCRIPTOR CONTAINS, IN ADDITION TO THE RELATIVE ADDRESS, AN INDEX WHICH CAN BE USED TO LOCATE THE SEGMENT-S ENTRY IN THE PROGRAM-S SEGMENT DICTIONARY. USING THIS INDEX AND THE ADDRESS OF THE SEGMENT DICTIONARY WHICH WAS PLACED IN THE PROGRAM-S PRT BY SELECTION, THE SEGMENT DICTIONARY IS LOCATED AND THE SIZE AND DISK ADDRESS OF THE SEGMENT ARE OBTAINED. THEN, THE PRESENCEBIT CALLS GETSPACE TO OBTAIN AN AREA INTO WHICH THE SEGMENT CAN BE READ, PLACES THE ADDRESS OF THAT AREA IN THE SEGMENT-S ENTRY IN SEGMENT DICTIONARY, AND CALLS DISKIO TO INITIATE THE I/O TO READ THE SEGMENT INTO THE SPECIFIED AREA. SUBSEQUENTLY, THE INDEX VALUE FOR THE SEGMENT-S ENTRY IN THE SEGMENT DICTIONARY IS PLACED IN THE AREA-S MEMORY LINK, ALL PROGRAM DESCRIPTORS ADDRESSING THE AREA ARE GIVEN THEIR ABSOLUTE ADDRESSES (I.E. EACH DESCRIPTOR-S ADDRESS FIELD IS SET TO THE VALUE OBTAINED BE ADDING ITS RELATIVE ADDRESS TO THE SEGMENT-S BASE ADDRESS), AND THE PRESENCE BITS OF THOSE DESCRIPTORS ARE SET TO ONE. PRESENCEBIT THEN CALLS THE SLEEP PROCEDURE SPECIFYING NOT TO RETURN CONTROL UNTIL THE DISK I/O IS COMPLETED. THE CONTROL SECTION THEN TAKES CONTROL AND PRESENCEBIT IS TEMPORARILY SUSPENDED. WHEN CONTROL IS RETURNED, THE DESIRED INFORMATION IS IN THE AREA NOW ADDRESSED BY THE PROGRAM DESCRIPTOR (S).

AFTER INFORMATION HAS BEEN MADE PRESENT AND THE CONCERNED DESCRIPTOR IS PROPERLY ADJUSTED, PRESENCEBIT TRANSFERS CONTROL TO INITIATE. THEN, DUE TO THE ADJUSTMENT MADE IN THE PROGRAM-S INTERRUPT WORDS, THE PROGRAM RESUMES CONTROL AT THE POINT WHERE IT WILL AGAIN ATTEMPT TO ADDRESS THE DESIRED INFORMATION, THIS TIME SUCCESSFULLY.

CONTROL SECTION OF MCP

THE CONTROL SECTION OF THEM MCP PERFORMS FOUR PRINCIPAL FUNCTIONS: (1) IT INTERROGATES INTERRUPTS, (2) IT CHECKS FOR CHANGES IN THE STATUS OF PERIPHERAL UNITS, (3) IT PROVIDES A MEANS BY WHICH AN MCP ROUTINE AND/OR PROCEDURE CAN REQUEST THAT AN INDEPENDENT PROCEDURE BE CALLED, AND (4) IT PROVIDES A MEANS BY WHICH AN MCP PROCEDURE CAN SUSPEND ITS PROCESSING UNTIL A NECESSARY CONDITION EXISTS. TO PERFORM THESE FUNCTIONS, THE CONTROL SECTION INCLUDES THE

INDEPENDENT RUNNER PROCEDURE AND THE SLATE ARRAY, THE SLEEP PROCEDURE AND THE BED ARRAY, AND THE NOTHINGTODO ROUTINE.

THERE ARE TWO IMPORTANT FACTORS INVOLVED WITH THE MECHANICS OF THE MCP CONTROL SECTION. ONE IS THAT A MCP PROCEDURE CAN RESERVE A CORE AREA FOR USE AS A PRIVATE STACK. IN SOME CASES, A PROCEDURE CAN USE THE STACK AREA OF THE NORMAL STATE PROGRAM WHOSE INTERRUPT IT IS HANDLING; IN OTHER CASES, A PROCEDURE MUST CALL GETSPACE AND OBTAIN A NON-OVERLAYABLE AREA. IN EITHER CASE, THE PROCEDURE CAN SET RS TO THE DESIRED AREA THROUGH USE OF THE STS OPERATOR THAT CAUSES RS TO BE SET TO A SPECIFIED ADDRESS. THE SECOND FACTOR INVOLVED WITH THE MECHANICS OF THE CONTROL SECTION IS THE B5700 PROCEDURE HANDLING TECHNIQUES; NAMELY, THE HARDWARE-S GENERATION AND USE OF CONTROL WORDS FOR PROCEDURE ENTRY AND EXIT.

THE INDEPENDENTRUNNER AND THE SLATE ARRAY ARE USED WHEN AN MCP PROCEDURE OR ROUTINE WISHES TO REQUEST THAT AN INDEPENDENT PROCEDURE BE CALLED. A PROCEDURE IS TERMED AN "INDEPENDENT PROCEDURE" IF IT IS NOT DIRECTLY ASSOCIATED WITH A PARTICULAR NORMAL STATE PROGRAM. FOR EXAMPLE, THE PROCEDURES STATUS, CONTROL CARD, SELECTION, AND RUN ARE INDEPENDENT PROCEDURES.

TO REQUEST THAT AN INDEPENDENT PROCEDURE BE CALLED, A CALL IS MADE ON THE INDEPENDENTRUNNER PROCEDURE SPECIFYING THE PROCEDURE TO BE CALLED AND A PARAMETER FOR THAT PROCEDURE. INDEPENDENTRUNNER THEN MAKES AN ENTRY IN THE SLATE ARRAY SPECIFYING THE GIVEN INFORMATION AND THEN RETURNS CONTROL TO THE REQUESTING PROCEDURE.

THE SLEEP PROCEDURE AND THE BED ARRAY ARE USED WHEN AN MCP PROCEDURE WISHES TO SUSPEND ITS PROCESSING UNTIL A CERTAIN CONDITION EXISTS. AN EXAMPLE OF WHEN A PROCEDURE MUST SUSPEND ITSELF IS AFTER IT HAS INITIATED AN I/O AND CANNOT CONTINUE UNTIL THE I/O HAS BEEN COMPLETED. TO SUSPEND ITSELF TEMPORARILY, A PROCEDURE CALLS THE SLEEP PROCEDURE AND PASSES TWO PARAMETERS; ONE PARAMETER IS THE ADDRESS OF A WORD TO BE TESTED, AND THE OTHER IS A MASK WORD THAT SPECIFIES WHICH BIT(S) IN THE DESIGNATED WORD SHOULD BE TESTED. THE SLEEP PROCEDURE THEN MAKES AN ENTRY IN THE BED ARRAY. THE INFORMATION IN THE BED ENTRY INCLUDES: (1) THE ADDRESS OF THE WORD SPECIFIED AS THE "TEST WORD", (2) THE MASK TO BE USED WITH THE TEST WORD, (3) THE CURRENT VALUE OF P1MIX WHICH PROVIDES THE MIX INDEX OF THE PROGRAM THAT CAUSED THE SUSPENDED PROCEDURE TO BE CALLED, AND (4) THE ADDRESS OF THE RETURN CONTROL WORD OF THE SLEEP PROCEDURE. THE SLEEP PROCEDURE OBTAINS THIS VALUE BY READING THE F-REGISTER. AFTER MAKING THE ENTRY IN THE BED ARRAY, SLEEP TRANSFERS CONTROL TO THE NOTHINGTODO ROUTINE. IT IS IMPORTANT TO NOTE THAT CONTROL IS TRANSFERRED TO NOTHINGTODO BY BRANCHING ON A LABEL DESCRIPTOR. CONSEQUENTLY, THE CONTROL WORDS GENERATED WHEN SLEEP WAS CALLED ARE LEFT IN THE PRIVATE STACK OF THE PROCEDURE THAT CALLED SLEEP. IT SHOULD ALSO BE NOTED THAT THE ENTRY MADE IN THE BED CONTAINS THE ADDRESS OF THE RETURN CONTROL WORD OF SLEEP.

NOTHINGTODO ROUTINE

THE NOTHINGTODO ROUTINE IS THE PRINCIPAL POINT OF CONTROL IN THE CONTROL SECTION. WHEN NOTHINGTODO RECEIVES CONTROL, IT FIRST PERFORMS AN INTERROGATE INTERRUPT OPERATION. THEN IF THERE ARE NO INTERRUPTS, IT CHECKS FOR AN ENTRY IN SLATE. IF THERE IS AN ENTRY IN THE SLATE AND IT IS THEN POSSIBLE TO CALL AN INDEPENDENT PROCEDURE, NOTHINGTODO SETS RS TO A STACK AREA FOR THE INDEPENDENT PROCEDURE AND CAUSES IT TO BE CALLED. IF THERE ARE NO ENTRIES IN THE SLATE, NOTHINGTODO INVESTIGATES BED. EACH TIME BEFORE EXAMINING A BED ENTRY, NOTHINGTODO PERFORMS AN INTERROGATE INTERRUPT OPERATION; THEN, IF THERE ARE NO INTERRUPTS, IT SELECTS AN ENTRY FROM THE BED. FROM A BED ENTRY, NOTHINGTODO GETS THE ADDRESS OF THE WORD TO BE TESTED AND THEN OBTAINS THE WORD. THE TEST WORD IS THEN MASKED WITH THE MASK WORD PROVIDED IN THE BED ENTRY. A MASK IS A WORD CONTAINING ZERO IN EVERY BIT POSITION OTHER THAN THE POSITIONS OF BITS TO BE TESTED. THE MASKING OPERATION IS A LOGICAL AND OPERATION PERFORMED ON THE TEST WORD AND THE MASK. THE LOGICAL AND OPERATION GENERATES A RESULT WORD WITH 1-S IN THE BIT POSITIONS IN WHICH BOTH THE TEST WORD AND THE MASK HAVE 1-S . IF THE MASKING OPERATION PRODUCED NEGATIVE RESULTS (I.E., THE CONDITION REQUIRED BY THE SUSPENDED PROGRAM STILL DID NOT EXIST), ANOTHER ENTRY WOULD BE TESTED. IF NO BED ENTRY PROVIDES POSITIVE RESULTS, NOTHINGTODO PERFORMS INTERROGATE INTERRUPT OPERATIONS AND CHECKS FOR CHANGES IN THE STATUS OF PERIPHERAL UNITS. IF THE MASKING OPERATION PRODUCED POSITIVE RESULTS(I.E., IF THE CONDITION REQUIRED BY THE SUSPENDED PROGRAM THEN EXISTED) THE NOTHINGTODO ROUTINE WOULD REMOVE THAT BED ENTRY AND RETURN CONTROL TO THE SUSPENDED PROCEDURE.

TO RETURN CONTROL TO A SUSPENDED PROCEDURE, NOTHINGTODO FIRST SETS P1MIX TO THE MIX INDEX VALUE IN THE BED ENTRY. THEN THE F-REGISTER IS SET TO THE ADDRESS OF THE RETURN CONTROL WORD OF THE SLEEP PROCEDURE. THIS CONTROL WORD IS, OF COURSE, STILL IN THE PRIVATE STACK OF THE PROCEDURE THAT SUSPENDED ITSELF BY CALLING SLEEP. THEN THE NOTHINGTODO ROUTINE CAUSES AN EXIT OPERATION TO BE PERFORMED, JUST AS WOULD BE PERFORMED TO EXIT A PROCEDURE. THE EXIT OPERATION IS HANDLED BY THE HARDWARE. SINCE THE F REGISTER AT THAT TIME CONTAINS THE ADDRESS OF THE RETURN CONTROL WORD FOR THE SLEEP PROCEDURE, THE EXIT OPERATION RETURNS CONTROL TO THE SUSPENDED PROCEDURE JUST AS THOUGH THE SLEEP PROCEDURE HAD CAUSED THE EXIT OPERATION.

AN EXAMPLE OF WHEN A PROCEDURE MUST TEMPORARILY SUSPEND ITSELF IS AFTER IT HAS INITIATED AN I/O AND CANNOT CONTINUE PROCESSING UNTIL THE I/O HAS BEEN COMPLETED. SPECIFICALLY, CONSIDER THE CASE NOTED ABOVE WHERE PRESENCEBIT CALLED DISKIO TO INITIATE A READ TO OBTAIN INFORMATION TO BE MADE PRESENT FOR A PROGRAM. IN THAT CASE, THE PARAMETERS THAT PRESENCEBIT WOULD PASS TO SLEEP WOULD BE: (1) THE ADDRESS OF THE I/O DESCRIPTOR USED TO PERFORM THE I/O OPERATION, AND (2) A MASK THAT SPECIFIED THAT THE I/O COMPLETE BIT IN THE I/O DESCRIPTOR WAS TO BE TESTED.

THE I/O COMPLETE BIT IS THE BIT THAT AN MCP I/O ROUTINE SETS TO 1 WHEN THE I/O OPERATION, DESCRIBED BY THE I/O DESCRIPTOR, HAS BEEN SUCCESSFULLY COMPLETED. WITH THIS INFORMATION AND P1MIX AND THE RESULT OBTAINED BY READING THE F REGISTER, SLEEP WOULD MAKE A BED ENTRY AND TRANSFER CONTROL TO NOTHINGTODO. SUBSEQUENTLY, THE INITIATED I/O WOULD BE COMPLETED BY THE I/O HARDWARE AND A I/O FINISHED INTERRUPT WOULD BE SET. THEN THE INTERRUPT CONDITION WOULD BE DETECTED AND CONTROL WOULD BE TRANSFERRED TO THE IOFINISH PROCEDURE. THE IOFINISH PROCEDURE WOULD THEN PERFORM ITS OPERATIONS AND SET THE PERTINENT I/O COMPLETE BIT TO 1. SUBSEQUENTLY, THE NOTHINGTODO ROUTINE WOULD TAKE CONTROL AND TEST ENTRIES IN BED. WHEN THE ENTRY MADE FOR PRESENCEBIT WAS DETECTED IN NOTHINGTODO, AS NOTED ABOVE, PRESENCEBIT WOULD BE REACTIVATED.

A NOTE ON PARALLEL PROCESSING

AS IS NOTED IN THE PARAGRAPH DESCRIBING THE INITIATE ROUTINE, PROGRAMS ARE INITIATED ON PROCESSOR 2 IN FAVOR OF PROCESSOR 1. THIS IS DONE BECAUSE PROCESSOR 1 HANDLES CONTROL STATE OPERATIONS, AND PROCESSOR 2 CAN ONLY IDLE IF NO NORMAL STATE PROGRAM IS INITIATED ON IT. TO OBTAIN MAXIMUM USE OF PROCESSOR 2, INTERRUPTS MUST BE HANDLED IN SUCH A WAY THAT PROCESSOR 1 NEED NEVER IDLE IF A NORMAL STATE PROGRAM IS READY TO BE INITIATED. CONSEQUENTLY, INTERRUPTS ARE HANDLED ON A PRIORITY BASIS. THAT IS, EACH INTERRUPT HAS A PRIORITY AND THOSE OF HIGHER PRIORITY ARE HANDLED FIRST. PROCESSOR 1 INTERRUPTS HAVE A HIGHER PRIORITY THAN PROCESSOR 2 INTERRUPTS. CONSEQUENTLY, IF BOTH A PROCESSOR 1 INTERRUPT AND A PROCESSOR 2 INTERRUPT WERE SET AT THE SAME TIME, THE PROCESSOR 1 INTERRUPT WOULD BE HANDLED BY THE TIME THE PROCESSOR 2 INTERRUPT WAS INTERROGATED.

WHEN A PROGRAM ON PROCESSOR 1 GENERATES AN INTERRUPT CONDITION, IT MAY BE SUCH THAT THE CONDITION CANNOT BE IMMEDIATELY RECTIFIED. FOR EXAMPLE, WHEN A PRESENCE BIT INTERRUPT OCCURS, AN I/O OPERATION TO BRING IN PROGRAM SEGMENT MAY HAVE TO TAKE PLACE. CONSEQUENTLY, THAT PROGRAM IDENTIFIED BY P1MIX IS TEMPORARILY SUSPENDED BE CALLING SLEEP. WHEN THE PROGRAM IDENTIFIED BY P1MIX IS SUSPENDED, NOTHINGTODO IS CALLED AND P1MIX IS SET TO ZERO. SUBSEQUENTLY, NOTHINGTODO RE-INITIATES A PROCEDURE FROM THE BED, AND P1MIX IS ASSIGNED THE VALUE OF THE PERTINENT PROGRAM'S MIX INDEX. IT SHOULD BE RECALLED THAT PROCESSING ON PROCESSOR 1 CAN ALSO BE INTERRUPTED BY THE OCCURRENCE OF AN INTERRUPT RELATED TO PROCESSOR 2. THEREFORE, IT IS OFTEN THE CASE THAT THE PROGRAM ON PROCESSOR 1 IS ONLY INTERRUPTED SO THAT THE MCP CAN HANDLE A PROCESSOR 2 INTERRUPT. WHEN THIS IS THE CASE, HOWEVER, P1MIX WILL NOT HAVE BEEN SET TO ZERO; IT WILL STILL CONTAIN THE VALUE OF THE MIX INDEX OF THE PROGRAM FROM PROCESSOR 1. IT SHOULD ALSO BE NOTED THAT WHEN THIS IS THE CASE, THE PROGRAM IDENTIFIED BY P1MIX IS READY TO BE INITIATED; IT REQUIRES NO MCP FIX-UP. BECAUSE OF THE NOTED CONDITIONS, PROCESSOR 2 INTERRUPTS ARE ALWAYS HANDLED AS FOLLOWS:

1. THE VALUES OF P1MIX AND P2MIX ARE EXCHANGED ONE FOR THE OTHER.

2. THEN, IF P2MIX IS NOT ZERO, THE PROGRAM WHOSE MIX INDEX IS SPECIFIED BY P2MIX IS INITIATED ON PROCESSOR 2 AND OPERATION WOULD CONTINUE AS NOTED BY THE STEP BELOW. IF P2MIX IS ZERO, OPERATION WOULD IMMEDIATELY CONTINUE AS NOTED BY THE STEP BELOW.

3. CONTROL IS TRANSFERRED TO THE PROCESSOR 1 INTERRUPT LOCATION THAT CORRESPONDS WITH THE PERTINENT PROESSOR 2 INTERRUPT, AND THE INTERRUPT IS HANDLED.

- INTERRUPT HANDLING -

IT SHOULD BE NOTED THAT INTERRUPT CONDITIONS ARE NOT RELATED TO PROCESSORS, BUT RATHER TO PROGRAMS. WHEN A PROGRAM IS INTERRUPTED, ALL INFORMATION REQUIRED FOR ITS RE-INITIATION IS CONTAINED IN CONTROL WORDS STORED IN CORE. CONSEQUENTLY, IT MAKES NO DIFFERENCE IF VALUES OF P1MIX AND P2MIX GET EXCHANGED OR THAT ALL INTERRUPTS BECOME ASSOCIATED TO P1MIX.

- INTERRUPT HANDLING -

INTERRUPT HANDLING IN THE MCP

WHEN AN INTERRUPT CONDITION ARISES, CERTAIN FLIPFLOPS IN THE PROCESSOR AND/OR IN CENTRAL CONTROL ARE SET; INDICATING ITS PRESENCE. IF THE CONDITION IS PROCESSOR-DEPENDENT, THEN, THAT PROCESSOR WILL BE INTERRUPTED. IF P2 IS INTERRUPTED, IT IDLES UNTIL P1 CAN HANDLE THE INTERRUPT FOR IT, FOR P1 ALSO RECEIVES INTERRUPTS FOR P2. IN ANY CASE, IF THE PROCESSOR IS IN NORMAL STATE, IT IS INTERRUPTED UPON COMPLETION OF THE CURRENT INSTRUCTION (SECL TRUE). IF P1 IS IN CONTROL STATE, THE INTERRUPT IS SAVED UNTIL THE EXECUTION OF THE INI INSTRUCTION. IF SEVERAL INTERRUPTS OCCUR, THEY ARE NOTED AND PROCESSED IN PRIORITY ORDER. THE MCP WILL NOT RETURN TO NORMAL STATE UNTIL ALL INTERRUPTS HAVE BEEN PROCESSED.

WHEN AN INTERRUPT IS ANSWERED, P1 ENTERS CONTROL STATE (IF NOT ALREADY IN IT) AND BRANCHES TO AN ABSOLUTE LOCATION IN VERY LOW CORE DEPENDENT UPON THE TYPE OF INTERRUPT. THE INTERRUPT INDICATION IS CLEARED.

THE FOLLOWING TABLE PRESENTS RELEVANT INFORMATION CONCERNING INTERRUPT HANDLING

TYPE	INDICATION	PRIORITY	MEMORY LOCATION
INITIAL LOAD	LODF	0	@20
P1 MEMORY PARITY ERROR	P1=101F	1	@60
P1 INVALID ADDRESS	P1=102F	2	@61
TIME INTERVAL	CC=103F	3	@22
I=O BUSY	CC=104F	4	@23
KEYBOARD REQUEST	CC=105F	5	@24
I=O #1 FINISHED	CC=108F	6	@27
I=O #2 FINISHED	CC=109F	7	@30
I=O #3 FINISHED	CC=110F	8	@31
I=O #4 FINISHED	CC=111F	9	@32
PRINTER 1 FINISHED	CC=106F	10	@25
PRINTER 2 FINISHED	CC=107F	11	@26
P2 BUSY	CC=112F	12	@33
INQUIRY REQUEST	CC=113F	13	@34
SPECIAL INTERRUPT 1	CC=114F	14	@35
DF READ CHECK FINISHED 1	CC=115F	15	@36
DF READ CHECK FINISHED 2	CC=116F	16	@37
P1 STACK OVERFLOW	P1=103F	17	@62
P1 COMMUNICATE	P1=BCD4	18	@64
P1 PROGRAM RELEASE	P1=BCD5	19	@65
P1 CONTINUITY BIT	P1=BCD6	20	@66
P1 PRESENCE BIT	P1=BCD7	21	@67

P1 FLAG BIT	P1-BCD8	22	@70
P1 INVALID INDEX	P1-BCD9	23	@71
P1 EXPONENT UNDERFLOW	P1-BCD10	24	@72
P1 EXPONENT OVERFLOW	P1-BCD11	25	@73
P1 INTEGER OVERFLOW	P1-BCD12	26	@74
P1 DIVIDE BY ZERO	P1-BCD13	27	@75
P2 MEMORY PARITY ERROR	P2-101F	28	@40
P2 INVALID ADDRESS	P2-102F	29	@41
P2 STACK OVERFLOW	P2-103F	30	@42
P2 COMMUNICATE	P2-BCD4	31	@44
P2 PROGRAM RELEASE	P2-BCD5	32	@45
P2 CONTINUITY BIT	P2-BCD6	33	@46
P2 PRESENCE BIT	P2-BCD7	34	@47
P2 FLAG BIT	P2-BCD8	35	@50
P2 INVALID INDEX	P2-BCD9	36	@51
P2 EXPONENT UNDERFLOW	P2-BCD10	37	@52
P2 EXPONENT OVERFLOW	P2-BCD11	38	@53
P2 INTEGER OVERFLOW	P2-BCD12	39	@54
P2 DIVIDE BY ZERO	P2-BCD13	40	@55

A PSEUDO STACK OVERFLOW IS CREATED WHEN THE MCP ATTEMPTS TO INITIATE A PROGRAM WHOSE (R+0) DOES NOT CONTAIN @2525252525252525.

DETAILED INTERRUPT EXAMPLE
PRESENSE 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 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+010) (OCTAL).

I. FORCE AN INI (INTERROGATE INTERRUPT) OPERATOR INTO THE T REGISTER.

J. TRANSFER TO EITHER CELL @55 OR CELL @67, DEPENDING ON WHETHER THIS WAS A PRESENCE BIT ON P1 OR P2. SET THE R REGISTER TO ZERO. SET THE S REGISTER TO @100.

K. PLACE AN 18 IN THE TOS (TOP OF STACK) AT CELL @101.

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 PLACE IN THE TOP OF STACK AT @101 IN K, ABOVE.

P. IN THIS CASE, THE CODE WILL CALL MAKEPRESENT (ANALYSIS). NOTE THAT ANALYSIS IS A TYPED PROCEDURE. BEFORE ENTERING MAKEPRESENT, ANALYSIS IS ENTERED, RETURNING WITH A VALUE TO BE PASSED AS A PARAMETER TO MAKEPRESENT. MAKEPRESENT IS THE ROUTINE, THEN, WHICH EVENTUALLY HANDLES PRESENCE BIT INTERRUPTS.

INTERVAL

CONTAINS INTERVAL OF TIME REMAINING UNTIL STATISTICS FILE IS TO BE UPDATED.

INTRNSC

THE INTRNSC ARRAY CONTAINS INFORMATION PERTAINING TO THE CURRENT SET OF INTRINSICS. THE ARRAY CONSISTS OF A MAIN AND SUB-TABLE. THE MAIN TABLE HAS ONE ENTRY FOR EACH INTRINSIC AND IS INDEXED BY INTRINSIC NUMBER. THE MAIN TABLE HAS THE FOLLOWING FORMAT:

[1:1] INTERLOCK BIT FOR MAKING INTRNSIC PRESENT.

[2:1] DENOTES PRESENSE OF TYPE 13 COPY.

[6:27] ABSOLUTE DISK ADDRESS OF INTRINSIC.

[33:15] SIZE (# OF WORDS) OF INTRINSIC IF NOT PRESENT.
ABSOLUTE DISK ADDRESS, IF PRESENT FOR TYPE 7 CALLS.

THE SUB-TABLE HOLDS THE ABSOLUTE CORE ADDRESS OF THE TYPE 13 COPY OF THE INTRINSIC PRESENT IN CORE. IF [2:1] OF THE TYPE 7 ENTRY IS 1 THEN THE SUB-TABLE CONTAINS A VALID ADDRESS. THE SIZE OF THE SUB-TABLE IS (INTSIZE DIV 3)+1. EACH WORD CONTANS ADDRESSES FOR 3 INTRINSICS SO THAT ACCESS TO THE SUB-TABLE IS MDE AS (INTRINSC DIV 2). THE FORMAT OF EACH WORD IS AS FOLLOWS: [3:15] = ABSOLUTE CORE ADDRESS FOR THE TYPE 13 COPY OF INT. #N

[18:15] ABSOLUTE CORE ADDRESS FOR TYPE 13 COPY OF INT. #N+1

[33:15] ABSOLUTE CORE ADDRESS FOR TYPE 13 COPY OF INT. #N+2

INTSIZE

USED TO DETERMINE ROW SIZE FOR EACH MIX INDEX IN INTABLE.

I/O ERROR MESSAGES

THE FOLLOWING IS A LIST OF I/O ERROR MESSAGES OF THE FORMAT:

I/O ERROR <INTEGER> <FILE DESIGNATOR> : <JOB SPECIFIER>

<INTEGER VALUE>	MEANING
1	A COBOL PROGRAM ATTEMPTED TO OPEN AN INPUT FILE THAT WAS NOT CLOSED. CONSEQUENTLY, PROCESSING OF THE PROGRAM WAS DISCONTINUED.
3	A COBOL PROGRAM ATTEMPTED TO OPEN REVERSE A FILE THAT WAS NOT CLOSED. CONSEQUENTLY, PROCESSING OF THE PROGRAM WAS DISCONTINUED.
5	A COBOL PROGRAM ATTEMPTED TO OPEN REVERSE A FILE THAT WAS NOT BLOCKED PROPERLY. CONSEQUENTLY, PROCESSING OF THE PROGRAM WAS DISCONTINUED.
6	A COBOL PROGRAM ATTEMPTED TO OPEN AN OUTPUT FILE THAT WAS NOT CLOSED. CONSEQUENTLY, PROCESSING OF THE PROGRAM WAS DISCONTINUED.
11	AN ATTEMPT WAS MADE TO CLOSE AN INPUT FILE WHICH WAS CLOSED OR NEVER OPENED.
12	AN ATTEMPT WAS MADE TO CLOSE AN OUTPUT FILE WHICH WAS CLOSE OR NEVER OPENED.
15	AN ATTEMPT WAS MADE TO READ A FILE FOR WHICH AT END HAS ALREADY BEEN PROCESSED.
16	THE RECORD COUNT ON AN INPUT TAPE DOES NOT AGREE WITH THE INTERNALLY ACCUMULATED RECORD COUNT. THE EXTERNAL RECORD OR BLOCK COUNT IS PRINTED OUT FIRST IN THE ERROR MESSAGE. THEN THE INTERNAL RECORD OR BLOCK COUNT IS PRINTED.
17	THE BLOCK COUNT ON AN INPUT TAPE DOES NOT AGREE WITH THE INTERNALLY ACCUMULATED BLOCK COUNT. THE EXTERNAL RECORD OR BLOCK COUNT IS PRINTED OUT FIRST IN THE ERROR MESSAGE. THEN THE INTERNAL RECORD OR BLOCK COUNT IS PRINTED.
18	THE HASH TOTAL ON A COBOL INPUT TAPE DOES NOT AGREE

WITH THE INTERNALLY ACCUMULATED HASH TOTAL.

19 AN IRRECOVERABLE PARITY ERROR HAS OCCURRED DURING READING OF A FILE ASSIGNED TO DISK OR TAPE. THE MESSAGE IS TYPED ONCE FOR EACH BLOCK WHICH IS IN ERROR UNLESS A USE PROCEDURE HAS BEEN SPECIFIED. THE USE PROCEDURE (IF ANY) WILL BE EXECUTED AND CONTROL WILL BE TRANSFERRED TO THE STATEMENT FOLLOWING THE READ STATEMENT.

20 AN IRRECOVERABLE PARITY ERROR OCCURRED ON AN OUTPUT TAPE OR DISK FILE. THE USE PROCEDURE HAS BEEN EXECUTED, ALLOWING PROGRAMMATIC CLOSING OF FILES WHICH MUST BE SAVED, AND THE PROGRAM IS NOW BEING DS-ED.

21 AN ATTEMPT WAS MADE TO READ FROM A FILE OPENED AS OUTPUT. THE PROGRAM IS DS-ED.

22 AN ATTEMPT WAS MADE TO READ FROM A ROW OF A DISK FILE WHICH WAS NEVER CREATED. TO GET THIS ERROR, THE RECORD NUMBER MUST BE LESS THAN THE HIGHEST RECORD NUMBER WRITTEN AND GREATER THAN 1. WHEN A RANDOM FILE IS WRITTEN, BUT RECORDS FALL ONLY IN ROWS 1 AND 3 OF A 3-ROW FILE, ATTEMPTS TO ACCESS RECORDS IN ROW 2 WILL CAUSE IO ERROR 22 INSTEAD OF EXECUTING INVALID KEY STATEMENTS.

A ROW OF DISK SPACE WILL BE ASSIGNED, AND THE APPROPRIATE RECORD WILL BE MADE AVAILABLE. THE CONTENTS OF THE RECORD WILL BE UNPREDICTABLE.

23 AN ATTEMPT WAS MADE TO WRITE ON A FILE WHICH WAS OPENED AS INPUT. THE PROGRAM IS DS-ED.

24 AN ATTEMPT WAS MADE TO WRITE ON A FILE WHICH WAS OPENED REVERSED. THE PROGRAM IS DS-ED.

25 IMPROPER CODE WAS PASSED TO THE COBOLIO INTRINSICS. THE PROGRAM WAS DS-ED.

26 A BLOCK OF LESS THAN EIGHT CHARACTERS HAS BEEN READ, OR A ZERO RECORD SIZE HAS BEEN ENCOUNTERED DURING THE READING OF A TECHNIQUE-B OR TECHNIQUE-C FILE WHICH UTILIZES THE SIZE DEPENDING OPTION. THE PROGRAM IS DS-ED.

27 NOT USED.

28 WHILE OPERATING UNDER TIME SHARING, A SEEK HAS BEEN ISSUED FOR A DATA COMMUNICATIONS DEVICE. THE PROGRAM IS DS-ED.

29 AN IRRECOVEABLE PARITY ERROR HAS OCCURRED WHILE READING A TAPE FILE WHICH WAS OPENED REVERSED. THE MESSAGE WILL BE TYPED ONCE FOR EACH BLOCK WHICH IS IN ERROR UNLESS A USE PROCEDURE HAS BEEN SPECIFIED. THE USE PROCEDURE (IF ANY) WILL BE EXECUTED, AND CONTROL WILL BE TRANSFERED TO THE STATEMENT FOLLOWING THE READ STATEMENT.

30 NOT USED.

31 AN ATTEMPT WAS MADE TO READ FROM A FILE WHICH IS CLOSED OR WAS NEVER OPENED. THE PROGRAM IS DS-ED.

32 AN ATTEMPT AS MADE TO WRITE A FILE WHICH IS CLOSED OR WAS NEVER OPENED. THE PROGRAM IS DS-ED.

33 AN ATTEMPT WAS MADE TO SEEK ON A FILE WHICH IS CLOSED OR WAS NEVER OPENED. THE PROGRAM IS DS-ED.

34 AN ATTEMPT WAS MADE TO WRITE BLOCK ON AN INPUT FILE. THE PROGRAM IS DS-ED.

35 AN ATTEMPT WAS MADE TO WRITE BLOCK ON A FILE OPENED REVERSED. THE PROGRAM IS DS-ED.

36 AN ATTEMPT HAS BEEN MADE TO WRITE A RECORD WHOSE SIZE IS LESS THAN ONE WORD.

37 AN ATTEMPT WAS MADE TO WRITE BLOCK ON A FILE WHICH IS CLOSED OR WAS NEVER OPENED. THE PROGRAM IS DS-ED.

40 FILE WAS OPEN AT TIME OF EXECUTION OF "SET" STATEMENT.

NOTE: NO ATTRIBUTE OF A FILE MAY BE SET WHILE THE FILE IS OPEN.

41 ATTEMPT HAS BEEN MADE TO SET A READ-ONLY ATTRIBUTE. READ-ONLY ATTRIBUTES ARE:

EOF
MAXRECSIZE
FIB
FPB
LABEL

42 ATTEMPT HAS BEEN MADE TO SET AN ATTRIBUTE TO AN ILLEGAL VALUE.

43 ATTEMPT MADE TO CHANGE NUMBER OF BUFFERS TO BE USED FOR RANDOM DISK FILE.

44 ATTEMPT MADE TO INCREASE NUMBER OF BUFFERS FOR A FILE.

45 BLOCK SIZE WAS CHANGED AND IS NO LONGER A MULTIPLE OF RECORD SIZE.

46 A CHANGE WAS MADE TO BLOCK SIZE OF A CARD, PRINTER, RANDOM DISK, OR DATACOM FILE.

47 ATTEMPT WAS MADE TO ACCESS "LABEL" WHEN THE FILE WAS NOT IN THE OPEN STATE OR THE FILE IS UNLABELED.

48 ATTEMPT WAS MADE TO CHANGE "TYPE" OF A DISK OR DATACOM FILE.

49 AN ILLEGAL VALUE HAS BEEN PASSED FOR ATTIBUTE NUMBER.

69 AN ATTEMPT WAS MADE TO WRITE ON DISK AT AN ADDRESS LESS THAN 100. THE PROGRAM WILL HANG IN A DO UNTIL FALSE LOOP. THE PROGRAM MAY BE DS-ED BY THE OPERATOR.

71 THE NUMBER OF RECORDS WITHIN A STRING ON A TAPE, USED BY A COBOL SORT PROGRAM, WAS WRONG. THIS WAS DUE TO AN INCORRECT READ OR WRITE ON THAT TAPE. CONSEQUENTLY, PROCESSING OF THE PROGRAM WAS DISCONTINUED.

71 PARITY OR BLANK TAPE IN THE SORT.

74 PARITY OR BLANK TAPE IN THE MERGE.

76 AN ERROR OCCURRED WITHIN A STRING BEING WRITTEN BY A COBOL SORT PROGRAM. THE NUMBER OF RECORDS THAT SHOULD HAVE BEEN WRITTEN DID NOT EQUAL THE NUMBER WRITTEN ON THE DESIGNATED UNIT. CONSEQUENTLY, THE PROCESSING OF THE PROGRAM WAS DISCONTINUED.

79 THE NUMBER OF RECORDS THAT SHOULD HAVE BEEN READ FROM OTHER TAPE UNITS IN THE FINAL MERGE PASS OF A SORT, BEING PERFORMED BY A COBOL SORT PROGRAM, DID NOT EQUAL THE NUMBER OF RECORDS WRITTEN ONTO THE FINAL OUTPUT TAPE. HOWEVER, AFTER ACTION WAS TAKEN TO TYPE THIS MESSAGE, THE SORT CLOSED THE FINAL OUTPUT REEL OR EXECUTED THE USER-S OUTPUT ROUTINE, SIGNALING END-OF-FILE. CONSEQUENTLY, THE OUTPUT TAPE MAY BE USED IN SPITE OF THIS ERROR MESSAGE. THE TAPE UNIT INDICATED IN THIS MESSAGE IS MEANINGLESS.

80 THE TOTAL NUMBER OF RECORDS ENTERED AS INPUT TO THE SORT, BEING PERFORMED BY A COBOL SORT PROGRAM, WAS NOT EQUAL TO THE NUMBER OF RECORDS PRODUCED AS OUTPUT FROM

THE SORT IN THE FINAL MERGE PASS. HOWEVER, AFTER ACTION WAS TAKEN TO WRITE THIS MESSAGE, THE SORT CLOSED THE FINAL OUTPUT FILE OR EXECUTED THE USER-S OUTPUT ROUTINE, SIGNALING END-OF-FILE. CONSEQUENTLY, THE OUTPUT TAPE MAY BE USED IN SPITE OF THIS MESSAGE. THE TAPE UNIT INDICATED IN THIS MESSAGE IS MEANINGLESS.

81 THE AMOUNT OF AVAILABLE DISK IS INSUFFICIENT FOR A DISK-ONLY OR ITD MODE SORT.

82 THE NUMBER OF RECORDS READ FROM THE INPUT DOES NOT MATCH THE NUMBER WRITTEN TO THE FINAL OUTPUT.

83 A DISK FILE WAS PASSED AS AN OUTPUT FILE WHICH WAS NOT LARGE ENOUGH TO HOLD ALL OF THE SORTED OUTPUT DATA.

84 IN DISK-ONLY SORT MODE, THE AMOUNT OF DISK SPECIFIED IS INSUFFICIENT TO DO A DISK-ONLY SORT.

85 IN ITD SORT MODE, THE NUMBER OF RECORDS READ FROM A STRING ON TAPE IS NOT THE SAME NUMBER WRITTEN.

86 NO RECORDS HAVE BEEN PASSED TO A SORT.

87 A SORT RECORD DESCRIPTION IS GREATER IN LENGTH THAN THE RECORD DESCRIPTION OF A FILE PASSED AS AN OUTPUT FILE IN A COBOL SORT.

IOFINISH

IOFINISH(R,U) IS A PROCEDURE WHICH HANDLES RESULT DESCRIPTOR "R" ON LOGICAL UNIT "U" AND DEALS WITH ERRORS IF THE INITIATING CALL FOR THE I/O REQUESTS IT CALLED FROM THE I/O COMPLETE INTERRUPT CELLS.

IOMASK

VARIABLE USED AS A MASK TO SLEEP UNTIL THE COMPLETE I/O ACTION IS FINISHED. ITS VALUE IS @2000000000.

I/O QUEUE

"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 TO BE RETURNED TO THE CALLER. "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".

IOQUEAVAIL

POINTER TO THE FIRST AVAILABLE SPACE IN IOQUE.

IOREQUEST

IOREQUEST(FINAL,IODESC,LOCATION) IS A PROCEDURE WHICH MAKES A REQUEST FOR I/O USING THE DESCRIPTOR AT "IODESC", TO RETURN AT LOCATION "FINAL", AND BE REBUILT AT "LOCATION". IOREQUEST TERMINATES AND DOES NOT WAIT FOR THE I/O TO BE COMPLETED.

IOTIME

DESCRIPTOR POINTING TO THE IOTIME ARRAY. IOTIME[I] CONTAINS I/O TIME FOR JOB WITH MIX INDEX = I.

ISTACK

DESCRIPTOR POINTING TO THE INDEPENDENT STACK. A TWENTY (20) WORD TABLE HAS BEEN ESTABLISHED IN THE HIGH ADDRESS END OF ISTACK, TO RECORD INFORMATION FOR RECOVERABLE DISK AND DRUM ERRORS. THE TWENTY WORDS ARE BROKEN DOWN INTO A FOUR WORD AREA (WITH ONE WORD RESERVED FOR EACH OF FOUR UNITS FOR WHICH INFORMATION IS BEING KEPT, DKA, DKB, DRA, DRB) AND A 16 WORD AREA USED AS A CIRCULAR BUFFER OF FOUR WORDS EACH TO STORE INFORMATION NECESSARY TO MAKE AN ENTRY IN THE MAINTENANCE LOG.

JAR

THE "JAR" CONTAINS INFORMATION ABOUT JOBS ACTUALLY RUNNING. THE "SELECTION" ROUTINE WILL FILL THE "JAR" FROM THE "SHEET" WHEN ENOUGH SPACE IS AVAILABLE TO RUN A JOB. ENTRIES IN THE "JAR" ARE AS FOLLOWS:

WORD	FIELD	CONTENTS
0	[1:1]	1 = JOB IS A COMPILER
	[6:42]	OBJECT PROGRAM-S <MFID>
1	[1:1]	1 = JOB IS IN PROCESS OF BEING DS-ED
	[6:42]	OBJECT PROGRAM-S <FID>
2	[1:1]	1 = PROGRAM WAS COMPILED USING COBOL (AFTER SELECTION).
	[1:2]	DURING SELECTION AS FOLLOWS: 0 = NORMAL 2 = IF JOB HAS BEEN XS-ED 3 = IF JOB HAS BEEN ES-ED
	[3:1]	1 = JOB HAS ENTRY IN MAINTANANCE LOG DUE TO PERIPHERAL UNIT ERROR.
	[5:1]	1 = JOB HAS DECLARED SOFTWARE INTERRUPTS (IPC).
	[6:1]	1 = THIS IN AN INVOKING OR INVOKED IPC.
	[7:1]	1 = THIS IS AN INVOKED IPC JOB.
	[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 ON COMPILE-AND-GO JOB
	[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
5	[1:23]	STARTING DATE FOR THE LOG (BINARY).
	[24:24]	STARTING TIME FOR THE LOG
6	[9:9]	TU-BUF IF STARTED FROM RJE
	[18:15]	SIZE OF LOG INFORMATION IN ESP DISK
	[33:15]	LOCATION OF THE FIRST RECORD OF THE LOG INFORMATION IN ESP DISK. IF JAR[P1MIX,2].[8:10] = 0, THEN THIS IS THE COMPILE PART OF THE LOG INFORMATION.

7		IDLE TIME
8		LENGTH OF EACH ROW OF THE CODE FILE.
9	[18:15]	IF NEQ 0 THEN ESP DISK ADDRESS FOR CHAIN.
	[33:15]	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 DISK. THE FOLLOWING FORMULA MAY BE USED TO LOCATE A GIVEN SEGMENT OF CODE ON DISK FOR A GIVEN PROGRAM:

```
ASSUME RD          =THE RELATIVE DISK ADDRESS FROM SEGMENT DICTIONARY
                    ENTRY [33:15] FIELD.

DISK SEGMENT ADDRESS    =(JAR[P1MIX,(RD DIV JAR[P1MIX,8]) +10]) +
                         (RD MOD JAR[P1MIX,8])
```

MIX INDEXES WHICH ARE INACTIVE ARE INDICATED BY A ZERO ENTRY IN JARROW[MIX].

JOB INITIATION

CONTROL CARD PROCEDURES

PROGRAM SCHEDULING INFORMATION, SUCH AS INSTRUCTIONS TO COMPILE A PROGRAM OR TO EXECUTE A LIBRARY PROGRAM, AND PROGRAM PARAMETER INFORMATION, SUCH AS PRIORITY SPECIFICATIONS, IS PROVIDED TO THE MCP THROUGH USE OF CONTROL CARDS AND PROGRAM PARAMETER CARDS. THE CARDS ARE MARKED WITH AN INVALID CHARACTER IN COLUMN 1 AND SPECIFY THEIR FUNCTION IN WORD DESCRIPTIONS IN A FREE FIELD FORMAT. WHEN CONTROL INFORMATION IS READ FROM MEDIA OTHER THAN CARDS, THE MEANS OF IDENTIFICATION IS DIFFERENT, BUT HANDLING PROCEDURES ARE SIMILAR. WHEN THE CONTROLCARD PROCEDURES ARE CALLED, THE CONTROLCARD PROCEDURE ANALYZE THE CONTROL INFORMATION AND MAKES APPROPRIATE ENTRIES IN A "SCHEDULE SHEET". THE CONTROLCARD PROCEDURES THEN REQUEST THAT THE SELECTION PROCEDURE BE CALLED.

SELECTION PROCEDURE

THE SELECTION PROCEDURE SELECTS A PROGRAM, IF ONE IS AVAILABLE, FROM THE SCHEDULE SHEET ON A PRIORITY BASIS, ASSIGNS IT A MIX INDEX, AND SETS UP CONDITIONS NECESSARY FOR THE PROGRAM TO BE INITIATED.

ALL PROGRAMS TO BE EXECUTED MUST BE ON THE DISK AS LIBRARY PROGRAMS AND, THEREFORE, HAVE ENTRIES IN THE DISK DIRECTORY. IF A PROGRAM FROM A LIBRARY TAPE IS TO BE RUN, IT MAY BE LOADED TO THE DISK THROUGH USE OF A CONTROL CARD. THE COMPILERS AUTOMATICALLY PLACE PROGRAMS ON DISK AS LIBRARY PROGRAMS; HOWEVER, FOR COMPILE-AND-GO RUNS, THE PROGRAMS ARE AUTOMATICALLY REMOVED WHEN THEY ARE SET UP TO BE INITIATED. SELECTION READS THE FILE HEADER FOR THE FILE OF THE PROGRAM TO BE INITIATED. CONTAINED IN THE FILE HEADER FOR A PROGRAM FILE IS THE DISK ADDRESS OF THE ZERO SEGMENT OF THE FILE.

THE ZERO SEGMENT OF A PROGRAM FILE IS A SPECIAL SEGMENT CONTAINING SUCH INFORMATION AS THE LOCATION WITHIN THE PROGRAM FILE OF THE PROGRAM-S PRT AND SEGMENT DICTIONARY AND THE SIZE OF EACH, AND THE PROGRAM SEGMENT TO BE EXECUTED. THE SEGMENT DICTIONARY IS A TABLE WHICH CONTAINS THE RELATIVE DISK ADDRESS AND THE SIZE OF EACH PROGRAM SEGMENT IN THE PROGRAM. SEGMENTS ARE ASSIGNED NUMBERS BY THE COMPILERS FOR REFERENCE PURPOSES.

THE SELECTION PROCEDURE READS THE ZERO SEGMENT INTO CORE, EXAMINES THE INFORMATION, AND THEN RESERVES AREAS IN CORE FOR THE PRT AND STACK AND THE SEGMENT DICTIONARY ACCORDING TO THEIR SPECIFIED SIZES. THESE AREAS ARE MARKED NON-OVERLAYABLE. THEN THE PRT AND SEGMENT DICTIONARY ARE READ INTO THEIR CORE AREAS, AND THE ADDRESS OF THE SEGMENT DICTIONARY IS PLACED IN A RESERVED PRT CELL. FROM THE

SEGMENT DICTIONARY, SELECTION DETERMINES THE DISK ADDRESS AND SIZE OF THE FIRST PROGRAM SEGMENT AND IT IS READ INTO CORE. THE PROGRAM CAN BE INITIATED AFTER THE PERFORMANCE OF THESE OPERATIONS AND OTHER NECESSARY FIX-UP OPERATIONS THAT MAY BE SPECIFIED IN CONTROL INFORMATION.

TO CAUSE A PROGRAM TO BE INITIATED, THE SELECTION PROCEDURE CONSTRUCTS AN INTERRUPT CONTROL WORD AND AN INTERRUPT RETURN CONTROL WORD AND PLACES THEM IN THE PROGRAM-S CORE STACK AREA. THE REGISTER SETTINGS IN THESE WORDS ARE GIVEN SUCH VALUES THAT IN APPEARANCE THEY INDICATE THAT THE PROGRAM WAS INTERRUPTED JUST BEFORE EXECUTING ITS FIRST SYLLABLE. SELECTION ALSO PLACES AN APPROPRIATE INITIATE CONTROL WORD IN THE PROGRAM-S PRT AND SETS UP MCP TABLES SO THAT THEY CONTAIN ALL NEEDED INFORMATION, INCLUDING THE ADDRESS OF THE PROGRAM-S PRT. SELECTION THEN REQUESTS THAT THE RUN PROCEDURE BE CALLED AND PROVIDES THE PROGRAM-S MIX INDEX AS A PARAMETER TO RUN.

RUN PROCEDURE

THE RUN PROCEDURE SETS UP CERTAIN VARIABLES AS NEEDED TO INITIATE A GIVEN PROGRAM. SPECIFICALLY, ONE VARIABLE ASSIGNED THE VALUE OF THE MIX INDEX WHICH WAS PASSED TO RUN AS A PARAMETER. RUN THEN TRANSFERS CONTROL TO THE INITIATE ROUTINE.

MIX INDEX

PROGRAMS THAT ARE SELECTED FROM THE SCHEDULE SHEET AND PUT IN PROCESS ARE CONSIDERED TO BE IN THE MIX. EVERY PROGRAM IN THE MIX HAS BEEN ASSIGNED A MIX INDEX. A PROGRAM-S MIX INDEX IS ACTUALLY AN INDEX INTO A MCP TABLE CALLED THE PRT ARRAY. THE PRT ARRAY CONTAINS A DESCRIPTOR FOR EVERY PROGRAM IN THE MIX. THE DESCRIPTOR FOR A GIVEN PROGRAM ADDRESSES THE BASE OF THE PROGRAM-S PRT. THE DESCRIPTOR FOR A PARTICULAR PROGRAM IS IN THE PRT ARRAY LOCATION CORRESPONDING TO THAT PROGRAM-S MIX INDEX. THROUGH USE OF THE PRT ARRAY AND THE MIX INDEXES, THE MCP CAN LOCATE THE PRT OF ANY PROGRAM IN THE MIX.

INITIATE ROUTINE

CONTROL IS TRANSFERED TO THE INITIATE PORTION OF THE SELECTION FOR THE PURPOSE OF INITIATING THE PROGRAM WHOSE MIX INDEX IS SPECIFIED BY P1MIX. TO INITIATE THE PROGRAM, INITIATE OBTAINS THE INITIATE CONTROL WORD FROM THE PROGRAM-S PRT WHICH IS LOCATED THROUGH USE OF THE PRT ARRAY AND P1MIX. BEFORE INITIATING THE PROGRAM ON PROCESSOR 1, HOWEVER, A CHECK IS MADE TO SEE IF IT COULD BE INITIATED ON PROCESSOR 2. IF PROCESSOR 2 IS AVAILABLE AND NOT BUSY, A VARIABLE CALLED P2MIX IS GIVEN THE VALUE OF P1MIX AND THE PROGRAM IS INITIATED ON PROCESSOR 2; OTHERWISE, THE PROGRAM IS INITIATED ON PROCESSOR 1.

P1MIX AND P2MIX

AFTER THE MCP GIVES CONTROL TO A NORMAL STATE PROGRAM, CONTROL WILL NOT RETURN TO THE MCP UNTIL THE PROGRAM IS INTERRUPTED. WHEN A NORMAL STATE PROGRAM IS INTERRUPTED, ALL INTERRUPT INFORMATION IS STORED IN THE PROGRAM'S STACK AND PRT, AND RR AND RS ARE SET TO CONTROL STATE AREAS. THE MCP MUST, HOWEVER, HAVE SOME LINK BACK TO THE PROGRAM THAT WAS PROCESSING. P1MIX PROVIDES THIS LINK FOR PROCESSOR 1 AND P2MIX FOR PROCESSOR 2. BEFORE A PROGRAM IS INITIATED ON PROCESSOR 1, P1MIX IS GIVEN THE VALUE OF THE MIX INDEX OF THE PROGRAM; LIKEWISE, FOR PROCESSOR 2 AND P2MIX. CONSEQUENTLY, WHEN AN INTERRUPT OCCURS, THE MCP KNOWS THE MIX INDEX OF THE PROGRAM INTERRUPTED ON PROCESSOR 2 AND/OR PROCESSOR 1.

JOBNUM

POINTER TO THE LAST ENTRY IN THE BED. THE NUMBER OF ENTRIES IN THE BED IS EQUAL TO (JOBNUM DIV 2) +1).

JUNK

TEMPORARY STORAGE

KEYBOARD INPUT MESSAGES

THE AU MESSAGE

THE AU MESSAGE ALLOWS THE OPERATOR TO DETERMINE THE AMOUNT OF AUXILIARY MEMORY IN USE BY A SNGLE PROGRAM OR FOR ALL PROGRAMS IN THE MIX. THE AU MESSAGE HAS THE FOLLOWING FORMATS:

```
<MIX INDEX>AU
AU
```

THE AX MESSAGE

THE AX MESSAGE ALLOWS THE CONSOLE OPERATOR TO COMMUNICATE WITH THE OBJECT PROGRAM THROUGH THE SPO IN RESPONSE TO AN ACCEPT MESSAGE.

ALL CHARACTERS FOLLOWING THE AX INCLUDING BLANKS WILL BE AVAILABLE TO THE OBJECT PROGRAM. A GROUP MARK IS INSERTED AFTER THE LAST CHARACTER ENTERED.

THE AX MESSAGE HAS THE FOLLOWING FORMAT:

```
<MIX INDEX> AX<MESSAGE DATA>
```

THE BK MESSAGE

PERFORMS THE EQUIVALENT OF THE BREAK KEY FUNCTION FOR A SPO CONSOLE OR THE SPO AND HAS THE FOLLOWING FORMATS:

```
<MIX INDEX>BK
BK
```

THE BS MESSAGE

SETS THE STATION INDICATED BY <TU>/<BUFF> AS A SPO CONSOLE. (SEE DESCRIPTION OF SPO CONSOLES.) THE BS MESSAGE HAS THE FOLLOWING FORMAT:

```
BS <TU>/<BUFF>
```

THE BS SPO MESSAGE

CAUSES SPO OUTPUT TO BE PRINTED ON THE SPO. THE BS SPO MESSAGE HAS

THE FOLLOWING FORMAT:

```
BS SPO
```

THE CC MESSAGE -- ?MESSAGE

THE CC MESSAGE ALLOWS THE SYSTEM OPERATOR TO SUPPLY CONTROL INFORMATION TO THE MCP VIA THE CONSOLE TYPEWRITER. THE INFORMATION FOLLOWING THE LETTERS CC IN THE CC MESSAGE ARE RECOGNIZED IN THE SAME FASHION AS THE INFORMATION FOLLOWING THE CHARACTER "?" ON CONTROL CARDS AND PROGRAM-PARAMETER CARDS.

THE CHARACTER ? CAN BE USED IN LIEU OF THE CHARACTERS CC IN THE CC MESSAGE, IF DESIRED.

WHEN A CC MESSAGE IS ENTERED AND THE END OF INPUT SWITCH IS PRESSED, THE TYPEWRITER WILL BECOME READY AGAIN UNLESS THE CC MESSAGE CONTAINED END CARD INFORMATION. CONSEQUENTLY, THE LAST CC MESSAGE MUST ALWAYS BE AN END CARD MESSAGE.

THE TERM <CONTROL INFORMATION> USED BELOW IS DEFINED AS ANY INFORMATION DEFINED VALID FOR USE ON CONTROL CARDS OR PROGRAM-PARAMETER CARDS.

THE CC MESSAGE MAY HAVE EITHER OF THE TWO FOLLOWING FORMATS:

```
CC <CONTROL INFORMATION>
?<CONTROL INFORMATION>
```

THE CD MESSAGE

THE CD MESSAGE CAUSES THE MCP TO TYPE THE NAME AND FIRST CARD IMAGE OF EACH PSEUDO CARD DECK THAT WAS PLACED ON THE DISK BY THE LDCNTRL/DISK PROGRAM. IF THERE ARE NO PSEUDO CARD DECKS ON THE DISK, THE FOLLOWING WILL BE TYPED:

```
NO DECKS ON DISK
```

THE CD MESSAGE HAS THE FOLLOWING FORMAT:

```
CD
```

THE CE MESSAGE

THE CE MESSAGE STARTS CANDE ON THE TIME SHARING SYSTEM AND HAS THE FOLLOWING FORMAT:

```
CE
```

- KEYBOARD INPUT MESSAGES -

THE CI MESSAGE

THE CI MESSAGE ALLOWS THE OPERATOR TO CHANGE INTRINSICS FILES IF THE SPECIFIED FILE IS PRESENT ON DISK. IF THE FILE IS PRESENT, THE MCP ENTERS THE NAME OF THE NEW FILE IN DISK SEGMENT ZERO. THE MCP WILL WAIT UNTIL THE ONLY JOBS BEING PROCESSED ARE LDCNTRL/DISK AND PRNPBT/DISK (OR THE MIX IS NULL), AND THEN PERFORM THE CHANGE.

THE CI MESSAGE HAS THE FOLLOWING FORMAT:

CI <MULTI FILE ID>/<FILE ID>

THE CL MESSAGE

THE CL MESSAGE IS USED TO CLEAR A SPECIFIC UNIT OF ANY JOB THAT IS USING IT, AND ALSO DS-ES THAT JOB. THE CL MESSAGE HAS THE FOLLOWING FORMAT:

CL<UNIT MNEMONIC>

THE CM MESSAGE

THE CM MESSAGE IS USED TO DESIGNATE A NEW MCP FILE TO BE USED FOLLOWING A HALT/LOAD. IF THE FILE IS PRESENT, AN ENTRY IS MADE IN DISK SEGMENT ZERO OF THE FILE NAME AND STARTING DISK ADDRESS OF THE FILE. THE MESSAGE HAS THE FORMAT:

CM<MULTI FILE ID>/<FILE ID>

THE CT MESSAGE

THE CT MESSAGE IS USED TO CHANGE THE TIME LIMITS FOR A JOB. FOR EITHER I/O OR PROCESSOR TIME LIMITS, IF THE INPUT IS A NON-ZERO INTEGER, THEN THE TIME LIMIT WILL BE CHANGED TO THESE NEW VALUES. THE CT MESSAGE HAS THE FOLLOWING FORMAT:

<MIX INDEX> CT <PROCESSOR PART> <I/O PART>

THE CU MESSAGE

THE CU MESSAGE ALLOWS THE CONSOLE OPERATOR TO DETERMINE THE CORE USAGE FOR A SINGLE PROGRAM OR ALL PROGRAMS IN THE MIX. THE CU MESSAGE HAS THE FOLLOWING FORMATS:

<MIX INDEX> CU

CU

THE DS MESSAGE

THE DS MESSAGE ALLOWS THE SYSTEM OPERATOR TO CAUSE A PROGRAM TO BE TERMINATED.

THERE ARE TWO FORMS OF THE DS MESSAGE. ONE FORM OF THE MESSAGE REQUIRES THAT THE PROGRAM TO BE TERMINATED BE IDENTIFIED THROUGH USE OF A "<MIX INDEX> DS"; THE OTHER MESSAGE REQUIRES THAT THE PROGRAM BE IDENTIFIED THROUGH USE OF A "DS <PROGRAM SPECIFIER>".

IF MORE THAN ONE PROGRAM IN A MIX HAVE THE SAME <PROGRAM NAME> AND A MESSAGE USING A <PROGRAM SPECIFIER> IS ENTERED, THE MCP WILL ARBITRARILY TERMINATE THE PROGRAM -- WITH THE NAME SPECIFIED -- THAT HAS THE LOWEST <MIX INDEX>. CONSEQUENTLY, IF A SITUATION SUCH AS NOTED SHOULD OCCUR, THE DS MESSAGE WHICH IDENTIFIES THE PROGRAM THROUGH USE OF THE "<MIX INDEX> DS" SHOULD BE USED.

THE DS MESSAGE MAY HAVE EITHER OF THE TWO FOLLOWING FORMATS:

```
<MIX INDEX> DS
DS <PROGRAM SPECIFIER>
```

THE DT MESSAGE

THE DT MESSAGE ALLOWS THE SYSTEM OPERATOR TO CHANGE THE VALUE OF THE CURRENT DATE WORD USED BY THE MCP. THE DT MESSAGE REQUIRES THE USE OF THREE <INTEGER>S, THE FIRST TWO OF WHICH MUST BE FOLLOWED BY THE CHARACTER /. THE FIRST <INTEGER> IS RECOGNIZED AS THE NUMBER OF THE MONTH OF THE YEAR, THE SECOND <INTEGER> IS RECOGNIZED AS THE DAY OF THE MONTH, AND THE THIRD <INTEGER> IS RECOGNIZED AS THE LAST TWO <DIGIT>S OF THE YEAR.

THE DT MESSAGE HAS THE FOLLOWING FORMAT:

```
DT <INTEGER> / <INTEGER> / <INTEGER>
```

THE ED MESSAGE

THE ED MESSAGE CAN BE USED TO ELIMINATE A PSEUDO CARD DECK WHICH IS CONTAINED IN A PSEUDO CARD READER IF THE READER IS NOT IN USE. THE ED MESSAGE HAS THE FOLLOWING FORMAT:

```
ED <DECK MNEMONIC>
```

THE ES MESSAGE

THIS MESSAGE TERMINATES A PROGRAM WHICH IS STILL IN THE SCHEDULE. THE ES MESSAGE WILL CAUSE THE PROGRAM TO BE LOADED INTO THE MIX AND DS TO BE PERFORMED BEFORE ANY OF ITS STATEMENTS ARE EXECUTED. THE MESSAGE HAS THE FOLLOWING FORMATS:

```
<SCHEDULE INDEX> ES
ES <JOB SPECIFIER>
```

THE EX MESSAGE

THE EX MESSAGE TYPES OUT ALL EXPIRED DISK FILE NAMES. THE EX MESSAGE FORMATS ARE AS FOLLOWS:

```
EX <MFID>/=
EX =/<FID>
EX =/=
EX <MFID>/<FID>
```

THE FM MESSAGE

THE FM MESSAGE MUST BE ENTERED IN RESPONSE TO A # FM RQD MESSAGE. THE <MIX INDEX> IN THE MESSAGE MUST AGREE WITH THE <MIX INDEX> IN THE # FM RQD MESSAGE, AND THE <UNIT MNEMONIC> MUST DESIGNATE THE UNIT TO BE USED FOR THE SUBJECT FILE. THE FM MESSAGE HAS THE FOLLOWING FORMAT:

```
<MIX INDEX> FM <UNIT MNEMONIC>
```

THE FR MESSAGE

THE FR MESSAGE ALLOWS THE SYSTEM OPERATOR TO SPECIFY THAT THE INPUT REEL, THE READING OF WHICH WAS JUST COMPLETED, WAS THE FINAL REEL OF AN UNLABELED FILE. THE FR MESSAGE HAS THE FOLLOWING FORMAT:

```
<MIX INDEX> FR
```

THE HD MESSAGE

THE HD MESSAGE ALLOWS THE OPERATOR TO INTERROGATE THE STATUS OF DISK. IN RESPONSE TO THE HD MESSAGE, THE MCP WILL PRINT THE FOLLOWING:

```
DKA/B EU <INTEGER> SU <INTEGER,INTEGER> ARE READY
```

THE FORMAT OF THE HD MESSAGE IS AS FOLLOWS:

```
HD
```

THE IL MESSAGE

THE IL MESSAGE IS USED IN RESPONSE TO A NO-FILE MESSAGE AND ALLOWS THE SYSTEM OPERATOR TO DESIGNATE THE UNIT ON WHICH A PARTICULAR INPUT FILE IS LOCATED. THE UNIT DESIGNATED IN THE IL MESSAGE MAY DENOTE THE LOCATION OF A NON-STANDARD FILE (A FILE WITH NO STANDARD B5500 LABEL) OR A STANDARD FILE (A LABELED FILE). IN EITHER CASE, THE FILE ON THE UNIT DESIGNATED IN THE IL MESSAGE WILL BE ASSUMED TO BE THE FILE REQUIRED IN THE RELATED NO-FILE MESSAGE. THE IL MESSAGE MUST HAVE THE FOLLOWING FORMAT:

```
<MIX INDEX> IL <UNIT MNEMONIC>
```

THE IN MESSAGE

THE IN MESSAGE ALLOWS THE SYSTEM OPERATOR TO INSERT AN <UNSIGNED INTEGER> INTO THE PROGRAM REFERENCE TAPE (PRT) OF THE PROGRAM SPECIFIED BY THE <MIX INDEX> AT THE RELATIVE LOCATION SPECIFIED BY THE OCTAL <INDEX> UNLESS THE SPECIFIED PRT CELL CONTAINS A DESCRIPTOR, OR THE <INDEX> IS LESS THAN 25 <OCTAL> OR OUT OF THE PRT BOUND. THE IN MESSAGE HAS THE FOLLOWING FORMAT:

```
<MIX INDEX> IN <PRT INDEX> = <UNSIGNED INTEGER>
```

THE LD MESSAGE

THE LD MESSAGE CAUSES THE LDCNTRL/DISK PROGRAM TO BE CALLED OUT FOR EXECUTION. THE LDCNTRL/DISK PROGRAM THEN SEARCHES FOR A TAPE OR CARD FILE WITH THE <MULTIPLE FILE IDENTIFICATION> CONTROL AND THE <FILE IDENTIFICATION> DECK

THEN, IF THE MESSAGE ENTERED WAS LD DK, THE FILE CONTROL/DECK IS PLACED ON DISK IN SUCH A FASHION THAT THE MCP CAN READ THE FILE AS A PSEUDO CARD DECK. IF THE MESSAGE ENTERED WAS LD MT, THE FILE CONTROL/DECK IS PLACED ON A MAGNETIC TAPE.

THE LD MESSAGE MAY HAVE EITHER OF THE FOLLOWING FORMATS:

```
LD DK
LD MT
```

THE LN MESSAGE

THE LN MESSAGE CAUSES THE CONTENTS OF THE CURRENT SYSTEM/LOG TO BE COPIED TO A NEW FILE ON DISK AND REINITIALIZES THE SYSTEM/LOG, OR CAUSES ALL DISK FILES TO BE LOGGED OUT IF THE DISKLOG MCP COMPILE-

TIME OPTION IS SET. THE LN MESSAGE HAS THE FOLLOWING FORMATS:

```
LN
LN DK
```

THE LR MESSAGE

THE LR MESSAGE WILL CAUSE AN EXISTING "REMOTE/LOG" FILE TO BE GIVEN THE NEW NAME "<MONTH><DAY><SERIAL>/REMLOG". FOLLOWING CREATION OF THE NEW "REMOTE/LOG" FILE, THE REMOTE/LOG IS INITIALIZED TO AN EMPTY CONDITION. THE LR MESSAGE HAS THE FOLLOWING FORMAT:

```
LR
```

THE MX MESSAGE

THE MX MESSAGE ALLOWS THE SYSTEM OPERATOR TO REQUEST THAT THE MCP TYPE A LIST OF <PROGRAM SPECIFIER>S DENOTING THE PROGRAMS IN THE MIX; THE PRIORITY AND <MIX INDEX> FOR EACH PROGRAM IS ALSO LISTED. SPECIFICALLY, EACH ITEM IN THE LIST TYPED BY THE MCP IN RESPONSE TO THE MX MESSAGE HAS THE FOLLOWING FORMAT:

```
<PRIORITY> : <PROGRAM SPECIFIER> = <MIX INDEX>
```

IF THERE IS NOTHING IN THE MIX, THE FOLLOWING MESSAGE WILL BE TYPED:

```
NULL MIX
```

THE MX MESSAGE MUST HAVE THE FOLLOWING FORMAT:

```
MX
```

THE MC MESSAGE

THE MC MESSAGE ALLOWS TO OPERATOR TO MARK A DISK FILE AS A SPECIAL COMPILER. THE MESSAGE WILL CAUSE THE FILE NAMES TO BE CHANGED TO "<MFID>/DISK" AND [8:1] OF WORD 4 OF THE PROGRAM-S DISK FILE HEADER WILL BE SET INDICATING A COMPILER. THE MC MESSAGE HAS THE FOLLOWING FORMAT:

```
MC <MFID>/<FID>
```

THE MR MESSAGE

THE MR MESSAGE ALLOWS THE OPERATOR TO CREATE A FILE LABELED "RESERVE/ DISK" WHICH CONTAINS 2000 SEGMENTS. THE FILE WILL BE AUTOMATICALLY REMOVED WHEN A "NO USER DISK" CONDITION OCCURS AND

ACTS AS A BUFFER TO BOTH NOTIFY THE OPERATOR THAT FILES NEED TO BE UNLOADED AND ALLOW THE OPERATOR MORE TIME TO ACCOMPLISH THE UNLOAD. THE FORMAT OF THE MR MESSAGE IS AS FOLLOWS:

MR

THE OC MESSAGE

THE OC MESSAGE ALLOWS THE OPERATOR TO PLACE A COMMENT INTO THE SYSTEM LOG. THE MESSAGE IS RESTRICTED TO A MAXIMUM SIZE OF 550 CHARACTERS AND HAS THE FOLLOWING FORMAT:

OC <MESSAGE>

THE OF MESSAGE

THE OF MESSAGE ALLOWS THE SYSTEM OPERATOR TO SPECIFY THAT A FILE REQUESTED FOR A COBOL PROGRAM WAS OPTIONAL, SO THAT THE SPECIFIED PROGRAM CAN PROCEED WITHOUT IT. THE OF MESSAGE HAS THE FOLLOWING FORMAT:

<MIX INDEX> OF

THE OK MESSAGE

THE OK MESSAGE CAUSES THE MCP TO RESUME PROCESSING OF A PROGRAM WHICH HAS BEEN TEMPORARILY SUSPENDED BECAUSE OF THE CONDITION DESIGNATED BY THE # DUP LIBRARY MESSAGE, THE NO USER DISK MESSAGE, THE NO FILE ON DISK, THE # OPRTR ST=ED MESSAGE, OR THE # FM RQD MESSAGE.

THE OK MESSAGE HAS THE FOLLOWING FORMAT:

<MIX INDEX> OK

THE OL MESSAGE

THE OL MESSAGE ALLOWS THE SYSTEM OPERATOR TO REQUEST THAT THE MCP TYPE INFORMATION PERTAINING TO LABELS OF FILES ON I/O UNITS.

THE OL MESSAGE HAS MANY FORMATS. ONE FORMAT SPECIFIES THAT A SPECIFIC <UNIT MNEMONIC> MAY BE ENTERED. THE OTHER FORMATS REQUIRE TWO-LETTER CODES WHICH SPECIFY A TYPE OF I/O UNIT. THE CODES AND THE I/O UNITS THEY REPRESENT ARE AS FOLLOWS:

CODE	I/O UNIT
CD	PSEUDO CARD READER

CP	CARD PUNCH
CR	CARD READER
LP	LINE PRINTER
MT	MAGNETIC TAPE
PP	PAPER TAPE PUNCH
PR	PAPER TAPE READER

IF AN OL MESSAGE SPECIFYING A SPECIFIC <UNIT MNEMONIC> IS ENTERED, THE RESPONSE MESSAGE WILL HAVE ONE OF THE FOLLOWING FORMATS, WHICHEVER IS RELEVANT.

<UNIT MNEMONIC> IN USE BY <PROGRAM SPECIFIER> : <MULTIPLE FILE IDENTIFICATION> <FILE IDENTIFICATION> <RDC>

<UNIT MNEMONIC> LABELED <MULTIPLE FILE IDENTIFICATION> <FILE IDENTIFICATION> <RDC>

<UNIT MNEMONIC> NOT READY
<UNIT MNEMONIC> SCRATCH
<UNIT MNEMONIC> UNLABELED

IF AN OL MESSAGE SPECIFYING A TYPE OF I/O UNIT IS ENTERED, AND IF A UNIT OF THE SPECIFIED TYPE IS IN USE AND/OR LABELED, THE RESPONSE MESSAGE WILL HAVE ONE OF THE FOLLOWING FORMATS, WHICHEVER IS RELEVANT.

<UNIT MNEMONIC> IN USE BY <PROGRAM SPECIFIER> : <MULTIPLE FILE IDENTIFICATION> <FILE IDENTIFICATION> <RDC>

<UNIT MNEMONIC> LABELED <MULTIPLE FILE IDENTIFICATION> <FILE IDENTIFICATION> <RDC>

<UNIT MNEMONIC> UNLABELED

IF AN OL MESSAGE SPECIFYING A TYPE OF I/O UNIT IS ENTERED, AND NO UNIT OF THAT TYPE IS IN USE AND/OR LABELED, THE FOLLOWING MESSAGE WILL BE TYPED:

NULL <UNIT MNEMONIC> TABLE

THE OL MESSAGE MAY HAVE ONE OF THE FOLLOWING FORMATS:

OL <UNIT MNEMONIC>
OL CD
OL CP
OL CR
OL LP
OL MT
OL PP
OL PR

THE OT MESSAGE

THE OT MESSAGE ALLOWS THE SYSTEM OPERATOR TO REQUEST THE MCP TO TYPE OUT THE VALUE OF A CELL IN A PROGRAM'S PROGRAM REFERENCE TABLE (PRT). THE PROGRAM IS SPECIFIED BY THE <MIX INDEX> AND THE CELL BY THE OCTAL <INDEX>. THE MCP MESSAGE TYPED WILL HAVE THE FOLLOWING FORMAT:

<JOB SPECIFIER> : R# <INDEX> = <PRT DATA>

THE VALUE OF <PRT DATA> WILL BE EXPRESSED AS AN OCTAL NUMBER FOR A DESCRIPTOR, OR AN INTEGER OF UP TO EIGHT DIGITS FOR AN OPERAND.

THE OT MESSAGE HAS THE FOLLOWING FORMAT:

<MIX INDEX> OT <PRT INDEX>

THE OU MESSAGE

THE OU MESSAGE ALLOWS THE SYSTEM OPERATOR TO DESIGNATE THE OUTPUT MEDIA OPTION FOR A LINE PRINTER FILE IF A # LP RQD, A # LP PBT MT RQD, OR A # PBT MT RQD MESSAGE HAS BEEN TYPED WHICH REFERENCES THE JOB THAT USES THE FILE.

THE OU LP FORM OF THIS MESSAGE SPECIFIES THAT THE SUBJECT LINE PRINTER FILE MUST BE PRODUCED AS OUTPUT ON A LINE PRINTER.

THE OU MT FORM OF THIS MESSAGE SPECIFIES THAT THE SUBJECT LINE PRINTER FILE MUST BE PRODUCED AS OUTPUT ON A PRINTER BACKUP TAPE.

THE OU DK FORM OF THIS MESSAGE SPECIFIES THAT THE SUBJECT LINE PRINTER FILE MUST BE PRODUCED AS OUTPUT ON PRINTER BACKUP DISK.

THE OU FORM OF THIS MESSAGE SPECIFIES THAT THE SUBJECT LINE PRINTER FILE MAY BE PRODUCED AS OUTPUT EITHER ON A LINE PRINTER OR A PRINTER BACKUP TAPE. THE OU MESSAGE MAY HAVE ANY OF THE FOLLOWING FORMATS:

<MIX INDEX> OU LP
<MIX INDEX> OU MT
<MIX INDEX> OU
<MIX INDEX> OU DK

THE PB MESSAGE

THE PB MESSAGE ALLOWS THE SYSTEM OPERATOR TO SPECIFY THAT A PRINTER OR PUNCH BACKUP FILE ON A PARTICULAR UNIT IS TO BE PRINTED OR PUNCHED. IF A SPECIFIED TAPE IS NOT A PRINTER OR PUNCH BACKUP TAPE,

THE FOLLOWING MESSAGE WILL BE TYPED:

NOT PRINT/PUNCH BACKUP TAPE

THE TERM <PBD NUMBER> MAY BE UP TO FOUR DIGITS AND IS THE NNNN PART OF THE PBD FILE NAME. THE PB MESSAGE HAS THE FOLLOWING FORMATS:

PB <UNIT MNEMONIC>
PB <PBD NUMBER>

THE PD MESSAGE

THE PD MESSAGE ALLOWS THE SYSTEM OPERATOR TO REQUEST THAT THE MCP TYPE INFORMATION PERTAINING TO WHAT FILES ARE LISTED IN THE DISK DIRECTORY. THE FORMATS OF THE PD MESSAGE ARE SHOWN BELOW. THE ACTION CAUSED BY THE PD MESSAGE DEPENDS UPON THE FORMAT OF THE MESSAGE. SPECIFICALLY, THE ACTIONS CAUSED BY THE PD MESSAGE ARE AS FOLLOWS.

IF A MESSAGE OF THE FORM

PD =/=

IS ENTERED, A LIST CONTAINING A <FILE SPECIFIER> FOR EACH FILE IN THE DISK DIRECTORY IS TYPED.

IF A MESSAGE OF THE FORM

PD <FILE SPECIFIER>

IS ENTERED AND THE FILE DESIGNATED IN THE MESSAGE IS IN THE DISK DIRECTORY, THE <FILE SPECIFIER> FOR THE FILE WILL BE TYPED. IF THE FILE DESIGNATED IN THE MESSAGE IS NOT IN THE DISK DIRECTORY, THE MESSAGE

NULL PD <FILE SPECIFIER>

WILL BE TYPED.

IF A MESSAGE OF THE FORM

= / <FILE IDENTIFICATION>
= / <PROGRAM IDENTIFICATION SUFFIX>

IS ENTERED, A LIST OF ALL FILES IN THE DISK DIRECTORY WHICH HAVE THE DESIGNATED <FILE IDENTIFICATION> OR <PROGRAM IDENTIFICATION SUFFIX>, IF ANY, WILL BE TYPED. IF NO SUCH FILES ARE IN THE DISK DIRECTORY, A MESSAGE OF THE FORM

NULL PD <FILE IDENTIFICATION PREFIX>/=

```
      NULL PD <FILE IDENTIFICATION PREFIX>
      NULL PD <PROGRAM IDENTIFICATION>/=
      NULL PD <PROGRAM IDENTIFICATION
```

WILL BE TYPED.

IF A MESSAGE OF THE FORM

```
      PD<MFID>/<FID> DATE
```

IS ENTERED, AND THE FILE IS PRESENT, THE CREATION DATE OF THAT FILE WILL BE TYPED, I.E. <MFID>/<FID> CREATED: MM/DD/YY.

IF A MESSAGE OF THE FORM

```
      PD<MFID>/<FID> LAST
```

IS ENTERED, AND THE FILE IS PRESENT, THE LAST DATE THAT FILE WAS ACCESSED WILL BE TYPED, I.E. <MFID>/<FID> ACCESSED: MM/DD/YY.

IF A MESSAGE OF THE FORM

```
      PD<MFID>/<FID> RECS
```

IS ENTERED, AND THE FILE IS PRESENT, THE NUMBER OF RECORDS IT CONTAINS WILL BE TYPED, I.E. <MFID>/<FID> RECORDS: NN.

IF A MESSAGE OF THE FORM

```
      PD<MFID>/<FID> SIZE
```

IS ENTERED, AND THE FILE IS PRESENT, THE NUMBER OF SEGMENTS THE FILE CONTAINS WILL BE TYPED, I.E. <MFID>/<FID> SEGMENTS: NN. IN TOTAL, THE PD MESSAGE MAY HAVE ANY ONE OF THE FOLLOWING FORMATS:

```
      PD =/=
      PD <FILE SPECIFIER>
      PD =/<FILE IDENTIFICATION>
      PD =/<PROGRAM IDENTIFICATION SUFFIX>
      PD <FILE IDENTIFICATION PREFIX>/=
      PD <FILE IDENTIFICATION PREFIX>
      PD <PROGRAM IDENTIFICATION>/=
      PD <PROGRAM IDENTIFICATION>
      PD <MFID>/<FID> DATE
      PD <MFID>/<FID> LAST
      PD <MFID>/<FID> RECS
      PD <MFID>/<FID> SIZE
```

THE PG MESSAGE
--- -- -------

THE PG MESSAGE ALLOWS THE SYSTEM OPERATOR TO PURGE A MAGNETIC TAPE ON A UNIT THAT IS READY, IN WRITE STATUS, AND NOT IN USE.

ANOTHER FORM OF THE PG MESSAGE ALLOWS THE OPERATOR TO WRITE A NUMERIC PHYSICAL REEL NUMBER IN THE TAPE LABEL. THE REEL NUMBER APPEARS IN THE SYSTEM LOG AND WILL PRINT ON THE SPO.

THE PG MESSAGE HAS THE FOLLOWING FORMATS:

```
PG <UNIT MNEMONIC>
PG <UNIT MNEMONIC>=<PHYSICAL REEL NUMBER>
```

THE PO MESSAGE

THE PO MESSAGE ALLOWS THE SYSTEM OPERATOR TO INTERROGATE THE STATUS OF A SPECIFIC OPTION.

THE MCP PROVIDES A NUMBER OF FEATURES WHICH ARE OPTIONAL AND CAN BE EVOKED OR INHIBITED THROUGH KEYBOARD INPUT MESSAGES. ALL OPTIONS HAVE A CORRESPONDING BIT POSITION IN THE MCP-S OPTION WORD WHICH IS MAINTANED BOTH IN MAIN MEMORY AND ON DISK.

EACH NON-COMPILE-TIME OPTION CAN BE INTERROGATED OR MODIFIED THROUGH USE OF THE <OPTION MNEMONIC> OR <OPTION NUMBER> WHICH CORRESPONDS TO THE BIT POSITION IN THE OPTION WORD. THE FORMATS OF THE PO MESSAGE ARE AS FOLLOWS:

```
PO OPTN <OPTION NUMBER>
PO <OPTION MNEMONIC>
```

THE PR MESSAGE

THE PR MESSAGE PROVIDES A MEANS WHEREBY THE SYSTEM OPERATOR CAN SPECIFY THE PRIORITY TO BE ASSIGNED A PROGRAM CURRENTLY IN THE MIX. THE PRIORITY TO BE ASSIGNED IS SPECIFIED BY THE TERM <PRIORITY>; THE PROGRAM TO WHICH THE PRIORITY IS TO BE ASSIGNED IS SPECIFIED BY THE <MIX INDEX>. THE TERM <PRIORITY> MUST BE AN <INTEGER>. THE PR MESSAGE HAS THE FOLLOWING FORMAT:

```
<MIX INDEX>PR<PRIORITY>
```

THE PS MESSAGE

THE PS MESSAGE CAN BE USED TO ALTER THE PRIORITY OF A PROGRAM IN THE SCHEDULE. THE PRIORITY TO BE ASSIGNED IS SPECIFIED BY THE TERM <PRIORITY>; THE PROGRAM IN THE SCHEDULE TO WHICH THE PRIORITY IS TO BE ASSIGNED IS SPECIFIED BY THE <SCHEDULE INDEX>. BOTH TERMS ARE INTEGERS. THE PS MESSAGE HAS THE FOLLOWING FORMAT:

<SCHEDULE INDEX>PS<PRIORITY>

THE QT MESSAGE

THE QT MESSAGE IS USED TO CAUSE PRNPBT/DISK TO SKIP A PRINTER BACKUP FILE AND NOT REMOVE THE FILE. IN RESPONSE TO THIS MESSAGE, THE FOLLOWI ACTIONS WILL BE TAKEN:

1. PRINTING OF THE CURRENT FILE IS DISCONTINUED.
2. PRINTING RESUMES WITH THE FIRST RECORD OF THE SUCCEEDING FILE.

THE QT MESSAGE HAS THE FOLLOWING FORMAT:

<MIX INDEX> QT

THE QV MESSAGE

THE Q-SECOND TIMEOUT FACILITY PROVIDES A MEANS FOR CAUSING THE MCP TO CLEAR READ-READY CONDITIONS FROM THE TERMINAL UNIT BUFFERS OF REMOTE STATIONS WHICH HAVE SPO CAPABILITIES IN CASES WHERE AN OBJECT PROGRAM (TO WHICH THE STATION IS ASSIGNED) DOES NOT READ THE INPUT WITHIN A GIVEN NUMBER OF SECONDS; I.E., WITHIN "Q" SECONDS. THE VALUE OF Q MAY BE SPECIFIED THROUGH USE OF A QV KEYBOARD INPUT MESSAGE OF THE FORM:

QV = <INTEGER>

IN RESPONSE TO THIS MESSAGE, THE MCP WILL SET Q TO THE VALUE SPECIFIED AND OUTPUT AN SPO MESSAGE OF THE FORM:

REMOTE SPO Q VALUE = <INTEGER> SECS

ALSO, IF A KEYBOARD MESSAGE OF THE FORM

QV

IS ENTERED, THE MCP WILL RESPOND WITH A MESSAGE, AS SHOWN ABOVE, WHICH SPECIFIES THE CURRENT VALUE OF Q.

THE RD MESSAGE

THE RD MESSAGE MAY BE USED TO REMOVE PSEUDO CARD DECKS FROM DISK WHICH WERE PLACED ON DISK BY THE LDCNTRL/DISK PROGRAM. PSEUDO CARD DECKS ARE IDENTIFIED BY NAMES HAVING THE FOLLOWING FORMAT:

```
      #<INTEGER>;
```

THE FORMAT OF THE RD MESSAGE IS AS FOLLOWS:

```
      RD #<4 DIGITS>
      RD <INTEGER>
      RD <PSEUDO DECK MNEMONIC>
```

THE RM MESSAGE

THE RM MESSAGE CAN BE USED IN RESPONSE TO A # DUP LIBRARY MESSAGE. THE RM MESSAGE CAUSES THE FILE ON DISK (WITH A NAME IDENTICAL TO THE FILE CREATED BY THE PROGRAM SPECIFIED IN THE # DUP LIBRARY MESSAGE) TO BE REMOVED, AND THEN CAUSES THE SUBJECT PROGRAM TO RESUME PROCESSING.

THE RM MESSAGE HAS THE FOLLOWING FORMAT:

```
      <MIX INDEX> RM
```

THE RN MESSAGE

THE RN MESSAGE IS USED TO SPECIFY THE NUMBER OF PSEUDO CARD READERS TO BE USED. IN TOTAL, THERE ARE 32 PSEUDO CARD READERS. AT HALT-LOAD TIME, THE NUMBER OF PSEUDO CARD READERS SPECIFIED TO BE USED IS ZERO.

AN RN MESSAGE MAY BE ENTERED AT ANY TIME. IF AN RN MESSAGE SPECIFIES THAT MORE PSEUDO CARD READERS ARE TO BE USED THAN ARE CURRENTLY BEING USED, THE MCP WILL SEARCH FOR PSEUDO CARD DECKS ON DISK AND MAKE USE OF AS MANY OF THE SPECIFIED PSEUDO CARD READERS AS POSSIBLE. IF AN RN MESSAGE SPECIFIES THAT FEWER PSEUDO CARD READERS ARE TO BE USED THAN ARE CURRENTLY BEING USED, A SUFFICIENT NUMBER OF THE PSEUDO READERS WILL BE "TURNED OFF" AS SOON AS THE READERS COMPLETE HANDLING OF THE PSEUDO CARD DECK IN PROCESS, IF ANY.

IF NO <DIGIT> IS ENTERED, THE DIGIT 1 IS ASSUMED.

THE RN MESSAGE ALSO ALLOWS THE OPERATOR TO SPECIFY A SPECIFIED PSEUDO DECK NUMBER WHICH WILL OPEN AN ADDITIONAL PSEUDO READER TO THAT DECK ONLY.

THE RN MESSAGE HAS ONE OF THE FOLLOWING FORMATS:

```
      RN
      RN <DIGIT>
      RN#<4 DIGITS>
```

THE RO OPTION

THE RO MESSAGE ALLOWS THE SYSTEM OPERATOR TO RESET OPTIONS.

THE MCP PROVIDES A NUMBER OF FEATURES WHICH ARE OPTIONAL AND CAN BE EVOKED OR INHIBITED THROUGH KEYBOARD INPUT MESSAGES. ALL OPTIONS HAVE A CORRESPONDING BIT POSITION IN THE OPTION WORD WHICH IS MAINTAINED BOTH IN MAIN MEMORY AND ON DISK

EACH NON COMPILE-TIME OPTION CAN BE MODIFIED OR INTERROGATED THROUGH USE OF THE <OPTION MNEMONIC> OR <OPTION NUMBER> WHICH CORRESPONDS TO THE BIT POSITION IN THE OPTION WORD. THE FORMATS OF THE RO MESSAGE ARE AS FOLLOWS:

RO USE OPTN <OPTION NUMBER>
RO <NOISE CHARACTER(S)> <OPTION MNEMONIC>

THE RR MESSAGE

THE DCMCP HAS A MEANS FOR "REMEMBERING" THE LOCATIONS OF REMOTE STATIONS WHICH HAVE SPO CAPABILITIES. IT IS SOMETIMES DESIRED, HOWEVER, TO DESIGNATE THAT CERTAIN STATIONS SHOULD NO LONGER BE IDENTIFIED AS "REMOTE SPO"S"; E.G., WHEN AN ADAPTER WHICH DOES NOT FACILITATE SPO FACILITIES REPLACES A TWX ADAPTER. IN ORDER THAT STATIONS MAY BE MARKED AS "NON-SPO STATIONS" THE RR (REMOVE REMOTE) KEYBOARD INPUT MESSAGE IS BEING PROVIDED.

RR <INTEGER> / <INTEGER>

WHEN THE MESSAGE IS ENTERED (WHERE THE FIRST <INTEGER> SPECIFIES A TERMINAL UNIT NUMBER AND THE SECOND A BUFFER NUMBER) THE MCP WILL REMOVE THAT STATION-S IDENTITY AS AN "SPO STATION". ALSO, IF THE STATION IS LOGGED-IN, IT WILL LOG IT OUT.

THE RR MESSAGE HAS THE FOLLOWING FORMAT:

RR <INTEGER>/<INTEGER>

THE RS MESSAGE

THE RS MESSAGE SPECIFIES THAT A BREAK FILE IS TO BE LOADED FROM THE FRONT OF AN OUTPUT TAPE WITH A WRITE RING TO DISK. THE MESSAGE DOES NOT INITIATE THE RESTART PROCESS. THE RS MESSAGE HAS THE FOLLOWING FORMATS:

RS <UNIT MNEMONIC>

THE RW MESSAGE

THE RW MESSAGE ALLOWS THE SYSTEM OPERATOR TO CAUSE A REWIND-AND-LOCK ACTION TO BE PERFORMED ON A MAGNETIC TAPE FILE THAT IS NOT IN USE. THE RW MESSAGE HAS THE FOLLOWING FORMAT:

RW <UNIT MNEMONIC>

THE RY MESSAGE

THE RY MESSAGE ALLOWS THE SYSTEM OPERATOR TO CAUSE, BY ENTERING A KEYBOARD MESSAGE, AN EFFECT ANALOGOUS TO THE EFFECT CAUSED BY PLACING A MAGNETIC TAPE UNIT IN LOCAN AND THEN REMOTE. THAT IS, IF A DESIGNATED UNIT IS NOT IN USE AND IN REMOTE, THE MCP WILL ATTEMPT TO READ A FILE LABEL.

THE RY MESSAGE CAUSES LOCKED FILES TO BE MADE ACCESSIBLE AND CAUSES LABEL CARDS (OR DATA CARDS), WHICH HAVE BEEN READ BUT NOT REFERENCED, TO BE IGNORED.

THE RY MESSAGE HAS THE FOLLOWING FORMAT:

RY <UNIT MNEMONIC>

THE SC MESSAGE

THE SC MESSAGE CAUSES A MESSAGE TO BE TYPED WHICH INDICATES THE SPO CONSOLES. THE SC MESSAGE HAS THE FOLLOWING FORMAT:

SC

THE SD MESSAGE

THE SD MESSAGE IS A VARIANT OF THE DS MESSAGE WHICH TERMINATES A PROGRAM AND DOES NOT REMOVE THE CONTROL DECK FROM DISK. THE FORMATS OF THE MESSAGE ARE:

<MIX INDEX>SD
SD <MFID>/<FID>

THE SF MESSAGE

THE MULTIPROCESSING FACTOR CAN BE CHANGED BY TYPING IN SF <DECIMAL NUMBER> (MEANING TO SET-THE-FACTOR). THE <DECIMAL NUMBER> IS DEFINED AS IN ALGOL, WITH THE RESTRICTION THAT <UNSIGNED INTEGER>S ARE AT THE MOST TWO DIGITS LONG:

<DECIMAL NUMBER>::=<UNSIGNED INTEGER> OR <DECIMAL FRACTION>
<UNSIGNED INTEGER> <DECIMAL FRACTION>

<DECIMAL FRACTION>::=.<UNSIGNED INTEGER>
<UNSIGNED INTEGER>::=<DIGIT> OR <DIGIT> <DIGIT>

THE SI MESSAGE

THE SI MESSAGE IS ONLY APPLICABLE IF THE MCP COMPILE-TIME OPTION "STATISTICS" HAS BEEN SET. THE SI MESSAGE IS USED TO SET THE STATISTICS INTERVAL TIMER. EACH TIME THE SI IS ENTERED OR THE TIMER COUNTS DOWN, THE STATISTICS FILE IS WRITTEN FROM THE FILE "SYSTEM <SM>/STATS" STORAGE TO THE FILE "SYSTAT<SM>/DISK". THE SI MESSAGE HAS THE FOLLOWING FORMAT:

SI <TIME IN SECONDS>

THE SL MESSAGE

THE SL MESSAGE IS ONLY APPLICABLE IF THE MCP COMPILE-TIME OPTION "STATISTICS" HAS BEEN SET. THE SL MESSAGE IS USED TO TRANSFER INFORMATION FROM THE FILE "STLOG<SM>/STATS" TO A FILE LABELED "<SERIAL>ON<JULIAN DATE>/STLOG<SM>".

THE SO MESSAGE

THE SO MESSAGE ALLOWS THE SYSTEM OPERATOR TO SET OPTIONS.

THE MCP PROVIDES A NUMBER OF FEATURES WHICH ARE OPTIONAL AND CAN BE EVOKED OR INHIBITED THROUGH KEYBOARD INPUT MESSAGES. ALL OPTIONS HAVE A CORRESPONDING BIT POSITION IN THE OPTION WORD WHICH IS MAINTAINED BOTH IN MAIN MEMORY AND ON DISK.

EACH NON COMPILE-TIME OPTION CAN BE MODIFIED OR INTERROGATED THROUGH USE OF THE <OPTION MNEMONIC> OR <OPTION NUMBER> WHICH CORRESPONDS TO THE BIT POSITION IN THE OPTION WORD. THE FORMATS OF THE SO MESSAGE ARE AS FOLLOWS:

SO USE OPTN <OPTION NUMBER>
SO <NOISE CHARACTER(S)> <OPTION MNEMONIC>

THE SS ALL MESSAGE

THE SS ALL MESSAGE PROVIDES A MEANS WHEREBY A MESSAGE CAN BE SENT TO ALL "REMOTE SPO" USERS ON THE SYSTEM. THE SS ALL MESSAGE HAS THE FOLLOWING FORMAT:

SS ALL: <ANY CHARACTERS EXCLUDING THOSE HAVING CONTROL>
SIGNIFICANCE

THE MIX SS ALL MESSAGE

THE MIX SS ALL MESSAGE PROVIDES A MEANS WHEREBY A MESSAGE CAN BE SENT TO "REMOTE SPO" USERS OF A PARTICULAR PROGRAM REGARDLESS OF WHETHER THEY HAVE REQUESTED MESSAGES VIA THE SM MESSAGE OR NOT.

IF THE GIVEN MIX HAS NO USERS, THE MESSAGE

NO STATIONS ON MIX = <MIX INDEX>

WILL BE RETURNED.

THE MIX SS ALL MESSAGE HAS THE FOLLOWING FORMAT:

<MIX INDEX> SS ALL: <ANY CHARACTERS ASIDE FROM THOSE HAVING CONTROL SIGNIFICANCE>

THE SS MESSAGE.

THE SS MESSAGE MAY BE USED AT THE CENTRAL SPO OR, IF PRECEDED BY A QUESTION MARK, ON A REMOTE STATION WITH SPO CAPABILITIES. IT DIRECTS A MESSAGE TO A REMOTE STATION WHICH HAS SPO CAPABILITIES, OR TO THE SPO. IF THE STATION ADDRESSED IS NOT RECOGNIZED TO HAVE SPO CAPABILITIES OR IS NOT READY, AN INV STN MESSAGE IS RETURNED. THE MESSAGE, AS PROVIDED AT THE ADDRESSED STATION, HAS A PREFIX WHICH INCLUDES THE ADDRESS OF THE ORIGINATOR. THE SS MESSAGE HAS THE FOLLOWING FORMAT:

SS <REMOTE STATION ADDRESS> : <REMOTE STATION MESSAGE>
SS SPO : <REMOTE STATION MESSAGE>

THE MIX SS MESSAGE

THE MIX SS MESSAGE PROVIDES A MEANS WHEREBY A MESSAGE CAN BE SENT TO ALL "REMOTE SPO" USERS OF A PARTICULAR MIX, WHO HAVE REQUESTED MIX MESSAGES VIA THE SM MESSAGE. IF NO USERS HAVE REQUESTED MESSAGES, THE MESSAGE

NO SM STATIONS ON MIX = <MIX INDEX>

WILL BE RETURNED. THE MIX SS MESSAGE HAS THE FOLLOWING FORMAT:

<MIX INDEX> SS: ANY CHARACTERS ASIDE FROM THOSE HAVING CONTROL SIGNIFICANCE

THE ST MESSAGE

THE ST MESSAGE ALLOWS THE SYSTEM OPERATOR TO SUSPEND THE PROGRAM REFERENCED BY THE <MIX INDEX> AS SOON AS THAT PROGRAM BECOMES READY TO BE RETURNED TO NORMAL STATE BY THE MCP. TO RESUME PROCESSING OF THE PROGRAM, THE OPERATOR MUST USE THE OK MESSAGE. THE ST MESSAGE HAS THE FOLLOWING FORMAT:

<MIX INDEX> ST

THE SV MESSAGE

THE SV MESSAGE MAY BE USED TO CAUSE A PERIPHERAL UNIT TO BE MADE INACESSIBLE UNTIL A HALT-LOAD OPERATION OCCURS OR UNTIL AN RY MESSAGE REFERENCING THE INACCESSIBLE UNIT IS ENTERED. IF, WHEN THE SV MESSAGE IS ENTERED AND THE SPECIFIED UNIT IS NOT IN USE, THE MESSAGE

<UNIT MNEMONIC> SAVED

WILL BE TYPED. IF A UNIT IS IN USE WHEN AN SV MESSAGE REFERENCING IT IS ENTERED, THE MESSAGE

<UNIT MNEMONIC> TO BE SAVED

WILL BE TYPED, AND THE UNIT WILL BECOME INACCESSIBLE AS SOON AS IT IS NO LONGER IN USE. UNTIL AN RY MESSAGE REFERENCING THE UNIT IS ENTERED OR A HALT-LOAD OCCURS, THE SAVED UNIT WILL NOT APPEAR NOT READY.

THE SV MESSAGE HAS THE FOLLOWING FORMAT:

SV <UNIT MNEMONIC>

THE SY MESSAGE

THE SY MESSAGE IS ONLY APPLICABLE IF THE MCP COMPILE-TIME OPTION "STATISTICS" HAS BEEN SET. THE MESSAGE CAUSES THE FILE "SYSTAT<SM>/DISK" TO BE COPIED TO THE FILE "<SERIAL>ON<JULIAN DATE> / SYSTAT<SN>.

THE TF MESSAGE

THIS MESSAGE ALLOWS THE OPERATOR TO INTERROGATE THE FACTOR AND HAS THE FOLLOWING FORMAT:

TF

THE TI MESSAGE

THE TI MESSAGE CAUSES THE MCP TO TYPE OUT THE AMOUNT OF PROCESSOR TIME AND I/O TIME THAT THE SUBJECT PROGRAM HAS USED UP AT THE TIME THE TI MESSAGE WAS ENTERED. THE TI MESSAGE HAS THE FOLLOWING FORMAT:

<MIX INDEX> TI

THE TL MESSAGE

THE TL MESSAGE PROVIDES A MEANS WHEREBY THE MCP TYPES THE PROCESSOR AND I/O TIME LIMITS FOR A DESIGNATED JOB. THE TL MESSAGE FORMAT IS:

<MIX INDEX> TL

THE TO MESSAGE

THE TO MESSAGE CAUSES THE SYSTEM TO TYPE A LIST OF OPTIONS AND THEIR SETTINGS.

THE MCP PROVIDES A NUMBER OF FEATURES WHICH ARE OPTIONAL AND CAN BE EVOKED OR INHIBITED THROUGH KEYBOARD INPUT MESSAGES. ALL OPTIONS HAVE A CORRESPONDING BIT POSITION IN THE OPTION WORD WHICH IS MAINTAINED BOTH IN MAIN MEMORY AND ON DISK.

EACH NON COMPILE-TIME OPTION CAN BE MODIFIED OR INTERROGATED THROUGH USE OF THE <OPTION MNEMONIC> OR <OPTION NUMBER> WHICH CORRESPONDS TO THE BIT POSITION OF THE OPTION WORD. THE FORMATS OF THE TO MESSAGE ARE AS FOLLOWS:

TO

THE TR MESSAGE

THE TR MESSAGE ALLOWS THE SYSTEM OPERATOR TO CHANGE THE VALUE OF THE TIME WORD USED BY THE MCP. THE TIME SPECIFIED BY THE <INTEGER> IN THE TR MESSAGE, IS DESIGNATED ACCORDING TO A 24-HOUR CLOCK I.E., MILITARY TIME. THE TR MESSAGE HAS THE FOLLOWING FORMAT:

TR <INTEGER>

THE TS MESSAGE

THE TS MESSAGE MAKES IT POSSIBLE TO DETERMINE THE PROGRAMS IN THE SCHEDULE. THE MCP WILL TYPE OUT THE NAMES OF EACH JOB IN THE SCHEDULE, TOGETHER WITH THE AMOUNT OF CORE SPACE NEEDED BY THE PROGRAM AND THE AMOUNT OF TIME THE PROGRAM HAS BEEN IN THE SCHEDULE. THE FORMATS OF THE TS MESSAGE ARE:

TS
<SCHEDULE INDEX>TS

THE UL MESSAGE

THE UL MESSAGE IS USED IN RESPONSE TO A NO FILE MESSAGE, AND ALLOWS THE SYSTEM OPERATOR TO DESIGNATE THE UNIT ON WHICH A PARTICULAR UNLABELED FILE IS LOCATED. THE UNIT DESIGNATED IN THE UL MESSAGE MAY DENOTE THE LOCATION OF A STANDARD FILE (A FILE ON WHICH THE FIRST RECORD IS A STANDARD B5500 LABEL) OR A NON-STANDARD FILE (A FILE WITH NO STANDARD LABEL). HOWEVER, IN EITHER CASE ALL RECORDS ON THE FILE INCLUDING THE STANDARD LABEL, IF ANY, WILL BE RECOGNIZED AS DATA RECORDS. THIS MESSAGE DIFFERS FROM THE IL MESSAGE IN THAT, WHEN THE IL MESSAGE IS USED IN REFERENCE TO A STANDARD FILE, A STANDARD LABEL WILL NOT BE RECOGNIZED AS A DATA RECORD. THE UL MESSAGE HAS THE FOLLOWING FORMAT:

<MIX INDEX> UL <UNIT MNEMONIC>

THE MIX WA MESSAGE

THE MIX WA MESSAGE PROVIDES A MEANS FOR DETERMINING WHAT STATIONS ARE ASSIGNED TO A PARTICULAR PROGRAM. IF ANY STATIONS ARE ASSIGNED TO THE GIVEN MIX, THE MCP WILL RETURN A MESSAGE OF THE FORM:

<INTEGER>/<INTEGER> ASSIGNED TO <PROGRAM SPECIFIER>

IF NO STATIONS ARE ASSIGNED TO THE GIVEN MIX, THE MCP WILL RETURN THE MIX WA MESSAGE PRECEDED BY THE WORD NULL. THE MIX WA MESSAGE HAS THE FOLLOWING FORMAT:

<MIX INDEX> WA

THE WD MESSAGE

THE WD MESSAGE CAUSES THE MCP TO TYPE THE DATE CURRENTLY BEING USED BY THE SYSTEM. THE DATE IS GIVEN IN THE MM/DD/YY FORMAT. THE WD MESSAGE HAS THE FOLLOWING FORMAT:

WD

THE WM MESSAGE

THE WM MESSAGE PROVIDES A MEANS OF DETERMINING THE MCP LEVEL, AND THE COMPILE-TIME OPTIONS THAT WERE SET. THE FORMAT OF THE MESSAGE IS:

WM

THE WP MESSAGE

THE WP MESSAGE PROVIDES A MEANS FOR DETERMINING WHAT PROGRAMS ARE ASSIGNED TO WHAT REMOTE STATIONS. IF THE WP MESSAGE IS FOLLOWED BY TU/BUF (WHERE TU AND BUF ARE EACH ONE OR TWO DIGIT NUMBERS), THE MCP WILL RETURN A MESSAGE SPECIFYING WHAT PROGRAMS ARE ASSIGNED TO THE SPECIFIED STATIONS, IF ANY. IF WP ALONE IS ENTERED, THE MCP WILL RETURN A COMPLETE LIST SPECIFYING WHAT PROGRAMS ARE ATTACHED TO WHAT STATIONS.

THE MESSAGE USED TO SPECIFY WHAT PROGRAMS ARE ATTACHED TO WHAT STATIONS IS AS FOLLOWS:

<INTEGER> / <INTEGER> ASSIGNED TO <PROGRAM SPECIFIER>

IF NO POSITIVE RESPONSE CAN BE PROVIDED FOR A WP MESSAGE, THE MESSAGE WILL BE RETURNED PRECEDED BY THE WORD NULL.

THE WP MESSAGE MAY HAVE EITHER OF THE FOLLOWING FORMATS:

WP <INTEGER> / <INTEGER>
WP

THE WT MESSAGE

THE WT MESSAGE CAUSES THE MCP TO TYPE OUT THE TIME OF DAY CURRENTLY RECOGNIZED BY THE SYSTEM. THE TIME IS GIVEN ACCORDING TO A 24-HOUR CLOCK. THE WT MESSAGE HAS THE FOLLOWING FORMAT:

WT

THE WU MESSAGE

THE WU MESSAGE PROVIDES A MEANS FOR DETERMINING THE USER IDENTIFICATIONS OF REMOTE SPO USERS. IF THE WU MESSAGE IS PRECEDED BY A MIX INDEX, THE MCP WILL IDENTIFY THE USERS OF THAT MIX, IF ANY. IF THE WU IS FOLLOWED BY TU/BUF (WHERE TU AND BUF ARE EACH ONE OR TWO DIGIT NUMBERS), THE MCP WILL IDENTIFY THE USER OF THE GIVEN SECTION, IF ANY. IF THE WU IS USED ALONE, THE MCP WILL IDENTIFY ALL USERS OF REMOTE SPO STATIONS, IF ANY. THE MESSAGE USED TO IDENTIFY THE USER OF A REMOTE STATION IS AS FOLLOWS:

<INTEGER> / <INTEGER> USED BY <USER CODE>

IF NO USERS ARE REFERENCED BY A WU MESSAGE, THE MESSAGE WILL BE

RETURNED PRECEDED BY THE WORD NULL.

THE WU MESSAGE MAY HAVE ONE OF THE FOLLOWING FORMATS:

```
<MIX INDEX> WU
WU <INTEGER> / <INTEGER>
WU
```

THE WY MESSAGE

THE WY MESSAGE ALLOWS THE SYSTEM OPERATOR TO REQUEST THAT THE MCP PROVIDE INFORMATION AS TO WHY A PROGRAM HAS BEEN TEMPORARILY SUSPENDED, PROVIDING THAT THE PROGRAM HAS BEEN TEMPORARILY SUSPENDED BECAUSE OF A REASON PREVIOUSLY DESIGNATED IN A SYSTEM MESSAGE WHICH: (1) WAS PRECEDED BY THE CHARACTER # AND (2) CONTAINED A <JOB SPECIFIER>; E.G., A PROGRAM WHICH WAS SUSPENDED BECAUSE OF THE CONDITION DENOTED BY A PREVIOUS # NO FILE MESSAGE.

IN RESPONSE TO THE WY MESSAGE, THE MCP DOES THE FOLLOWING: (1) LISTS THE TWO-LETTER CODES FOR ALL KEYBOARD INPUT MESSAGES WHICH COULD BE ENTERED TO ELIMINATE THE CONDITION THAT CAUSED THE PROGRAM TO BE TEMPORARILY SUSPENDED, AND (2) RETYPES THE # MESSAGE THAT WAS PREVIOUSLY TYPED TO INFORM THE SYSTEM OPERATOR OF THE CONDITION THAT CAUSED THE PROGRAM TO BE SUSPENDED.

THE WY MESSAGE HAS THE FOLLOWING FORMAT:

```
<MIX INDEX> WY
```

THE XS MESSAGE

THE XS MESSAGE CAUSES A PROGRAM WHICH IS IN THE SCHEDULE TO BE LOADED IN SPITE OF THE FACT THAT THE MCP DOES NOT THINK THE PROGRAM WILL RUN EFFICIENTLY WITH THE JOBS ALREADY IN THE MIX. THIS IS DONE BY TYPING IN THE NEW MESSAGE XS <JOB SPECIFIER>, MEANING TO EXECUTE FROM SCHEDULE. THE FORMAT OF THE MESSAGE IS:

```
XS <SCHEDULE INDEX>
```

THE XT MESSAGE

THE XT MESSAGE IS USED TO EXTEND THE TIME LIMITS FOR A JOB. IF <PROCESSOR PART> IS <INTEGER>, THEN THE PROCESSOR TIME LIMIT WILL BE EXTENDED BY <INTEGER> MINUTES. IF EITHER THE PROCESSOR OF I/O TIME LIMITS WERE EXTENDED, A MESSAGE WILL BE PRINTED TO NOTIFY THE OPERATOR OF THIS. IN ANY EVENT, THE PROCESSOR AND I/O LIMITS WILL BE TYPED OUT. THE XT MESSAGE FORMAT IS:

* KEYBOARD INPUT MESSAGES *

<MIX INDEX> XT <PROCESSOR PART> <I/O PART>
<PROCESSOR PART> ::= <EMPTY> OR <INTEGER> OR *
<I/O PART> ::= <EMPTY> OR , <EMPTY> OR , <INTEGER> OR , *

KEYBOARDCOUNTER

COUNTER FOR THE NUMBER OF UNPROCESSED KEYBOARD REQUEST INTERRUPTS

KEYIN

KEYIN(B) IS A PROCEDURE WHICH READS THE SPO AND EVALUATES THE RESULTING INPUT. "B" IS THE BUFFER.

KILL

KILL(A) IS A PROCEDURE WHICH SETS "S" TO 100, DOES A FORGETSPACE ON "A", AND GOES TO NOTHINGTODO.

KLUDGE

KLUDGE IS A PROGEDURE WHICH RESERVES SPACE FOR A DISK ADDRESS.

KLUMP

CELL @174.

LABELTABLE

THE LABELTABLE IS THE PRIMARY TABLE IN THE I/O QUEUE. THE ENTRY OF A UNIT INTO THE TABLE SPECIFIES ONE OF THE FOLLOWING:

@114	THE UNIT IS NOT READY.
-@14	THE UNIT IS IN USE BY "CONTROL CARD".
@214	THE UNIT IS REWOUND AND LOCKED.
@314	THE UNIT IS READY AND CONTAINS AN UNLABELED TAPE.
+	THE UNIT IS READY AND AVAILABLE.
-	THE UNIT IS IN USE.
0	THE UNIT IS READY AND CONTAINS A SCRATCH TAPE.

PLEASE REFER TO THE I/O QUEUE DOCUMENTATION FOR ADDITIONAL INFORMATION.

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

ENTRIES FOR LABEL EQUATION ARE AS FOLLOWS:

WRD	FIELD	CONTENTS
0		MULTI-FILE ID. (7 CHARACTERS)
1		FILE ID. (7 CHARACTERS)
2	[0:18]	REEL NO.
	[18:30]	CREATION DATE
	[42:1]	= 1 IF CREATION DATE IS ENTERED ON A LABEL EQUATION CARD SO A CHECK WILL BE MADE AT FILEOPEN TIME (SEE MCP 20435500 & 39087000) DS:=5 OCT TAKES SIGN OF ZONE BIT OF RIGHT MOST CHARACTER
3	[0:12]	CYCLE
	[42:1]	OPERATOR NOTIFICATION BIT
	[43:5]	FILE TYPES 0 = CP 1 = LP 2 = MT 3 = FOR SPECIFIC UNIT 4 = LP/PBT 5 = SPECIFIC UNIT (UNLABELED) 6 = PBT 7 = PT 8 = PT UNLABELED 9 = MT UNLABELED 10 = DISK RANDOM 11 = SPO 12 = DISK SERIAL 13 = DISK UPDATE 14 = DATA COMMUNICATION 15 = PBD 16 = PBD/PBD 17 = LP/PBD 18 = LP/PBT/PBD 19 = REMOTE 20 = PUT (PUNCH BACK-UP TAPE) 21 = CP/PUT 22 = PUD (PUNCH BACK-UP DISK) 23 = CP/PUD 24 = PUT/PUD

- LABEL EQUATION TABLE -

		25 = CP/PUT/PUD
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.

LABEL RECORD

WORD	CHARACTER (WORD)	CHARACTER (RECORD)	FIELD DESCRIPTION
1	1-8	1-8	MUST CONTAIN " LABEL "
2	1	9	MUST BE ZERO.
2	2-8	10-16	MULTI-FILE ID.
3	1	17	MUST BE ZERO.
3	2-8	18-24	FILE ID.
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 (3 = BLOCKED, 0 = NOT BLOCKED).
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.

- LABEL RECORD -

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 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.

THE FILE CONTROL ROUTINES WILL ALSO AUTOMATICALLY ACCEPT, BUT WILL NOT AUTOMATICALLY PRODUCE, USASI-FORMATTED LABELLED TAPES.

LABEL WORD

THE OPERAND CONSTRUCTED BY GOTOSOLVER.

FIELD	CONTENTS
[8:10]	BLOCK COUNTER SETTING
[18:15]	PROPER "F" REGISTER SETTING. IF 0 THEN OUTER BLOCK.
[33:15]	"R" RELATIVE ADDRESS OF THE LABEL DESCRIPTOR.

THE OPERAND IS USUALLY CONTAINED IN THE STACK OR IN THE PRT (FOR RTE). IT IS USED TO REPRESENT ACTION LABELS OR FORMAL LABELS.

LABELASCRATCH

LABELASCRATCH(LBL) IS A PROCEDURE WHICH OBTAINS A TAPE FOR MCP USE (DUMP REQUESTS) AND LABELS IT WITH THE LABEL AT ADDRESS "LBL".

LASTCDNUM

CONTAINS LAST CONTROL DECK NUMBER CURRENTLY IN USE.

LASTDECK

POINTS TO THE HEADER OF THE LAST PSEUDO DECK IN THE QUEUE.

LASTEU

CONTAINS THE IDENTIFICATION OF THE LAST EU ACCESSED BY THE SYSTEM FOR DISK ALLOCATION PURPOSES IN DISK FILE EXCHANGE CONFIGURATION. IT IS USED FOR THE PURPOSE OF ALLOCATING DISK FROM DIFFERENT EU-S IF POSSIBLE.

LBMESS

LBMESS(FN,SN,A,B,C,D,E) IS A PROCEDURE WHICH OUTPUTS MOST LIBRARY MAINTENANCE MESSAGES.

LEFTOFF

CONTAINS A POINTER TO THE LAST ADDRESS ACCESSED BY OLAY IN OVERLAYING CORE. IT IS USED TO ATTEMPT TO MAKE CORE ALLOCATION CYCLIC.

LIBRARY MAINTENANCE

LIBRARY MAINTENANCE IS A FUNCTION PROVIDED THROUGH THE OPERATING SYSTEM WHICH ALLOWS USERS TO EASILY "DUMP" FILES FROM DISK TO TAPE AND "LOAD" FILES FROM TAPE TO DISK. A DISCUSSION OF THE NECESSARY CONTROL CARD SYNTAX APPEARS IN THE "B5500/B5700 ELECTRONIC INFORMATION PROCESSING SYSTEM INFORMATION MANUAL", PUBLICATION NO. 1024916.

A USER IS ABLE TO DUMP A MAXIMUM OF 511 DISK FILES TO A LOGICAL TAPE FILE. THE NUMBER OF PHYSICAL REELS OF TAPE REQUIRED IS DEPENDENT ON THE NUMBER OF FILES AND SIZE OF EACH FILE. IF MORE THAN 511 FILES ARE SPECIFIED, THE OPERATING SYSTEM WILL AUTOMATICALLY INITIATE ANOTHER COPY OF LIBRARY MAINTENANCE WHICH HANDLES THE NEXT 511 FILES.

THE FORMAT OF EACH FILE DUMPED DETERMINES THE PHYSICAL CONTENTS OF THE APPROPRIATE PORTION OF THE DUMP TAPE. THE OPERATING SYSTEM EXAMINES THE DISK FILE HEADER OF A FILE WHICH IS TO BE DUMPED AND WILL CAUSE EACH OPEN ROW TO BE DUMPED IN ENTIRETY REGARDLESS OF THE END OF FILE POINTER. IT IS THUS EXTREMELY WASTEFUL TO DECLARE A ONE ROW DISK FILE WHICH COULD CONTAIN 100,000 ONE-SEGMENT RECORDS IF THE FILE ACTUALLY CONTAINS 100 RECORDS.

A MCP COMPILE TIME OPTION, B6500LOAD, PERMITS USERS TO LOAD FILES FROM B6500 TAPES ON A B5500 WITH THE FOLLOWING RESRTRICTIONS:

1. THE FILE MUST HAVE 20 OR LESS ROWS (PAGES) DECLARED.
2. THE BLOCK SIZE MUST BE 1023 WORDS OR LESS.
3. THE MAXIMUM RECORD SIZE MUST BE 1023 WORDS OR LESS.
4. FILES MUST BE WORD OR CHARACTER (DECIMAL, EBDIC, OR BCL) ORIENTED.
5. RECORDS/BLOCK IS COMPUTED BY DIVIDING BLOCKSIZE UNITS BY MAXRECSIZE UNITS.

THE FORMAT OF A LIBRARY MAINTENANCE TAPE IS AS FOLLOWS:

PHYSICAL RECORD NUMBER	CONTENTS
1	TAPE LABEL
2	TAPE MARK
3	NAME BLOCK, 1023 WORDS MAXIMUM SIZE EACH FILE REQUIRES TWO WORDS FOR <MFID> AND <FID> LAST ENTRY DENOTED BY AN @14
4	TAPE MARK
5	COPY OF RECORD NO. 1 TAPE LABEL
6	LABEL FOR FILE NO. 1
7	TAPE MARK
8	FILE HEADER FROM DIRECTORY (30 WORDS)
9	ENTIRE CONTENTS OF FILE (ROW BY ROW) IF ROW SIZE GTR 900 WORDS THEN 900 WORD BLOCKS ELSE ROW SIZE BLOCKS
10	TAPE MARK
11	LABEL FOR FILE NO. 1 (COPY OF RECORD NO. 6)
X	LABEL FOR FILE NO. 2
X+1	TAPE MARK
.	
.	
.	
N	TAPE MARK (LAST RECORD ON TAPE)

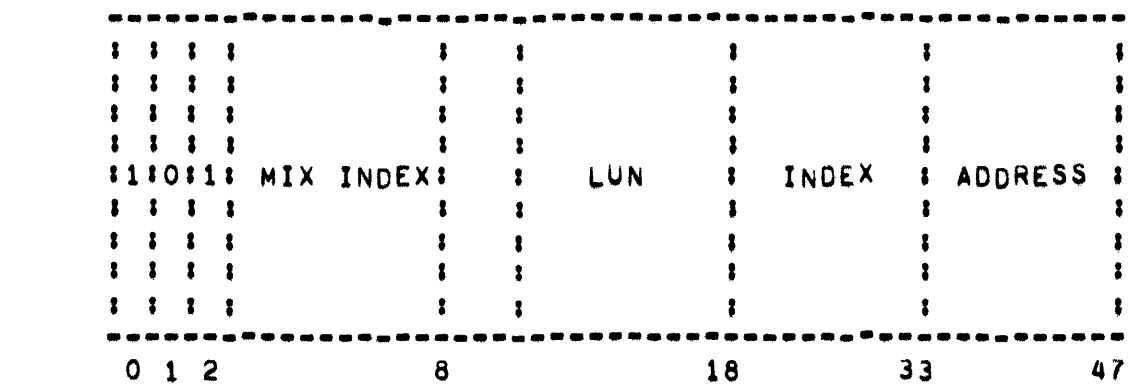
PHYSICAL RECORD NUMBER 6 THROUGH 11 IS REPEATED FOR EACH FILE.

AN ESPDISK ENTRY IS MADE FOR EACH LIBRARY MAINTENANCE OPERATION. THE FORMAT OF THE ENTRY IS AS FOLLOWS:

WORD	FIELD	CONTENTS
0	[2:6]	LOGICAL UNIT NUMBER 23, 24, 25, OR GEQ 30
	[8:1]	1=DUMP EXPIRED FILES
	[9:1]	1=DUMP ACCESSED FILES
	[18:15]	CODE NUMBER INDICATING "LOAD", "DUMP", ETC (SEE THE MCP PROCEDURE "RESWDS" FOR THE CODE NUMBER)
1		TAPE LABEL
2		FILE NO. 1 <MFID>
3		FILE NO. 1 <FID>
4		FILE NO. 2 <MFID>
5		FILE NO. 2 <FID>
.		
.		
.		
29		ESPDISK ADDRESS LINK FOR ADDITIONAL FILE NAME ENTRIES 0 = NO ADDITIONAL ENTRIES

LOCATQUE

ALL ENTRIES IN "LOCATQUE" HAVE THE FOLLOWING FORMAT:



FIELD	CONTENTS	DESCRIPTION
[0:3]	5	DESCRIPTOR IDENTIFICATION BITS.
[3:5]	MIX INDEX	MIX INDEX OF PROGRAM WHICH REQUESTED THE I/O OPERATION.
[8:1]		SHARED DISK
[9:1]		SUPPRESS ERROR MESSAGE
[10:1]		TIME SHARING (CANDE I=0)
[11:1]		ERROR RECOVERY IN PROGRESS
[12:6]	LUN	LOGICAL UNIT NUMBER OF UNIT ON WHICH THE I/O IS TO BE EXECUTED.
[18:15]	INDEX	INDEX INTO I/O-QUEUE OF NEXT I/O REQUEST ON THIS UNIT. @77777 IF NO ADDITIONAL REQUESTS.
[33:15]	ADDRESS	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.

SEE ALSO "I/O QUEUE" FOR ADDITIONAL INFORMATION.

- LOGARRAY, LOGENTRY, LOGFREE, LOGHOLDER -

LOGARRAY

DESCRIPTOR POINTING TO LOGARRAY ARRAY WHICH CONTAINS PSEUDO-LOG ENTRY INFORMATION FOR STATISTICS INFORMATION.

LOGENTRY

MAINTENANCE LOG LOGICAL ENTRY NUMBER, INCREASED BY 1 FOR EACH ENTRY.

LOGFREE

VARIABLE USED TO INTERLOCK THE LOG AND TO SPECIFY THE CURRENT LOCATION IN THE LOG.

LOGHOLDER

QUEUE WORD TO KEEP TRACK OF REQUESTS TO MAKE ENTRIES IN THE MAINTENANCE LOG.

LOGHOLDER.[18:15] = ADDRESS OF LAST ENTRY.
LOGHOLDER.[33:15] = ADDRESS OF 1ST ENTRY.

LOGGING

SYSTEM LOG

THE MCP MAINTAINS AN ACCOUNTING LOG RECORDING THE SYSTEM TIME AND OTHER INFORMATION CONCERNING THE SYSTEM. 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 LABELED "SYSTEM/LOG" WHICH MUST BE DECLARED IN THE COLD START DECK.

THE "SYSTEM/LOG" FILE IS RESTRICTED TO ONE ROW. EACH LOGICAL RECORD CONSISTS OF FIVE (5) WORDS AND EACH PHYSICAL BLOCK CONTAINS SIX (6) LOGICAL RECORDS (1 SEGMENT OR 30 WORDS).

THE FIRST RECORD IN THE SYSTEM LOG IS USED BY THE MASTER CONTROL TO INDICATE THE POSITION OF THE LAST VALID ENTRY IN THE LOG. THE FORMAT OF THE RECORD IS:

WORD	DESCRIPTION
0	NUMBER OF VALID ENTRIES IN THE LOG
1	MAXIMUM SIZE OF THE LOG (RECORDS)
2	NUMBER OF RECORDS IN THE LAST ENTRY
3	1 = LOG WRAPAROUND HAS OCCURRED 2 = 1 SEGMENT LEFT IN LOG 3 = LOG IS HALF FULL
4	LITERAL "DISKLOG"

A PROGRAM TO PRINT THE SYSTEM LOG IS PROVIDED BY THE BURROUGHS CORPORATION. HOWEVER, ANY USER PROGRAM IS ABLE TO READ THE LOG FILE. CONSEQUENTLY, EACH INSTALLATION IS ABLE TO PRODUCE A LOG ANALYSIS PROGRAM EXCLUSIVELY SUITED TO ITS NEEDS.

IF THE FILE "SYSTEM/LOG" IS DECLARED IN THE "COLD/START" DECK, THE FILE WILL NOT BE DESTROYED BY A "COLD/START", A "COOL/START" OR A HALT/LOAD OPERATION. USERS SHOULD "FREE" THE SYSTEM LOG FILE FOLLOWING A "COLD/START" TO INSURE PROPER OPERATION.

WHEN THE LOG BECOMES HALF FULL, A MESSAGE IS TYPED ON THE SPO TO NOTIFY OPERATOR. WHEN THE LOG IS ALMOST FULL OR THE OPERATOR ENTERS THE "LN" MESSAGE, AN MCP ROUTINE IS STARTED WHICH COPIES THE EXISTING SYSTEM LOG TO A NEW FILE AND INITIALIZES THE EXISTING LOG FILE. A MESSAGE OF THE FORM:

**** NEW LOG FILE IS <IDENTIFIER>/SYSLOG

IS TYPED ON THE SPO TO NOTIFY THE OPERATOR OF THE NEW NAME. THE OPERATOR SHOULD IMMEDIATELY "DUMP" THIS FILE TO A TAPE TO PRESERVE THE INFORMATION AS THE "COPIED" LOG FILE IS UNPROTECTED (CONSIDERED TO BE THE SAME AS A USER FILE).

THE <IDENTIFIER> OF THE FILE NAME IS INTERPRETED AS <MMDDSSS> WHERE:

<MM> ::= A TWO DIGIT NUMBER REPRESENTING THE MONTH.
<DD> ::= A TWO DIGIT NUMBER REPRESENTING THE DAY.
<SSS> ::= A THREE DIGIT NUMBER WHICH IS INCRIMENTED EACH TIME AN "LN" IS ENTERED THROUGH THE SPO ON A SPECIFIC DAY.

INFORMATION IS PLACED IN THE LOG BY THE MCP PROCEDURES SIGNOFF AND LOGSPACE. SIGNOFF OBTAINS INFORMATION FROM SEVERAL ARRAYS DURING THE COMPLETION OF A PROGRAM AND CREATES AN ARRAY WHICH CONTAINS THE LOG ENTRY. LOGSPACE CHECKS THE FIRST RECORD OF THE SYSTEM LOG FILE TO DETERMINE WHERE TO PLACE THE ENTRY AND MAKES THE ACTUAL WRITE TO THE FILE. THE PROCEDURE STARTIMING (ALSO DEFINED TO BE STOPTIMING) CREATES ADDITONAL ENTRIES IN THE FILE PARAMETER BLOCK WHICH REFLECT THE NUMBER OF TIMES THE FILE HAS BEEN OPENED AND CLOSED AND CAUSES ADDITIONAL INFORMATION OF A STATISTICAL NATURE TO BE ENTERED. THE FPB ENTRIES BECOME "FILE" ENTRIES IN THE LOG.

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
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 THE LOG INFORMATION FOR THE COMPILER IS LISTED IN A COMPILE-ONLY ENTRY.

C. IF AN EXECUTE RUN (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 (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 AND OVERWRITES A FILE WITH THE SAME NAME ON DISK.

4. WHEN THE OPERATOR ENTERS A LOG-OUT INSTRUCTION, "LNDK". THIS CAUSES ALL DISK FILES TO BE LOGGED-OUT AND RESETS THEIR CREATION DATE AND TIME.

DISK LOG FORMAT

THE FORMAT OF DISK LOG ENTRIES IN THE SYSTEM LOG IS:

WORD	CONTENTS	DESCRIPTION
0	TYPE CODE	= 8
1	<MFID>	FIRST NAME OF FILE (7 CHRS)
2	<FID>	SECOND NAME OF FILE (7 CHRS)
3	USER CODE	(7 CHRS)
4	CREATION DATE	YYDDD FORMAT (BCD)
5	CREATION TIME	60TH OF A SEC. (OCT)
6	DATE LOGGED	YYDDD FORMAT (BCD)
7	TIME LOGGED	60TH OF A SEC. (OCT)
8	NUMBER OF SEGMENTS	(OCT)
9		RESERVED FOR EXPANTION

GENERAL FORMAT OF THE THREE OTHER TYPES OF LOG ENTRIES:

COMPILE-AND-GO ENTRY

	CONTROL CARD INFORMATION	10R 2	COMPILER INFORMATION		0	OBJECT PROGRAM INFORMATION	
C O D E	1ST 72 COLUMNS OF COMPILE CARD	C O D E	GENERAL PROGRAM INFOR- MATION	FILE INFOR- MATION	C O D E	GENERAL PROGRAM INFOR- MATION	FILE INFOR- MATION
3	2 RECORDS	2	2 RECORDS	N REC.-	0	2 RECORDS	M REC.-

COMPILE-ONLY ENTRY

	CONTROL CARD INFORMATION	10R 2	COMPILER INFORMATION	
C O D E	1ST 72 COLUMNS OF COMPILE CARD	C O D E	GENERAL PROGRAM INFOR- MATION	FILE INFOR- MATION
3	2 RECORDS	2	2 RECORDS	N REC.-

EXECUTE ENTRY

	CONTROL CARD INFORMATION	0	OBJECT PROGRAM INFORMATION	
C O D E	1ST 72 COLUMNS OF COMPILE CARD	C O D E	GENERAL PROGRAM INFOR- MATION	FILE INFOR- MATION
3	2 RECORDS	0	2 RECORDS	M REC.-

GENERAL PROGRAM INFORMATION ENTRY FORMAT

WORD	FIELD	CONTENTS	DESCRIPTION
0	[1:2]	SYSTEM	INTEGER, 0, 1, 2, 3
	[3:45]	TYPE CODE	0 = EXECUTE 1 = ALGOL COMPILER 2 = COBOL COMPILER 3 CONTROL CARD INFORMATION 4 = END OF LOG 5 = PRINTER BACK UP 6 = FORTRAN COMPILER 7 = BASIC COMPILER 8 = DISK FILES 9 = XALGOL COMPILER 10= TSPOL COMPILER
1		FILES	INTEGER = NO. FILES DECLARED
2		PROCESS TIME	INTEGER = 60THS OF A SECOND
3		I-O TIME	INTEGER = 60THS OF A SECOND
4		PRORATED TIME	INTEGER = 60THS OF A SECOND
5		START DATE	BCL = YYDDD FORMAT
6		START TIME	INTEGER = 60THS OF A SECOND SINCE LAST HALT LOAD
7		STOP TIME	INTEGER = 60THS OF A SECOND SINCE LAST HALT LOAD
8	[1:30]	STOP DATE	BINARY = YYDDD FORMAT
	[42:6]	FINISH CODE	BINARY 0 = EOJ 1 = SYNTAX ERROR 2 = DS=ED OR ES=ED 3 = ABORT
9		USERCODE	BCL

FILE INFORMATION ENTRY

THE FILE INFORMATION ENTRY IS AN EXACT COPY OF THE PROGRAM=S FILE PARAMETER BLOCK (FPB) IN MAIN MEMORY. EACH LOGICAL ENTRY IN THE FPB CONTAINS 5 WORDS. A DETAILED EXPLANATION OF THE FPB APPEARS IN ANOTHER PORTION OF THIS DOCUMENT.

REMOTE LOGGING

THE MASTER CONTROL PROGRAM AUTOMATICALLY MAINTAINS A LOG FILE ON DISK FOR REMOTE DEVICES IF THE OPTIONS "DATACOM" AND "DCLOG" WERE SET TRUE WHEN THE MCP WAS COMPILED. THE LOG INFORMATION IS WRITTEN TO A FILE ON USER DISK LABELED "REMOTE/LOG". EACH LOGICAL RECORD IN THE FILE CONSISTS OF FIVE WORDS AND EACH PHYSICAL RECORD IS ONE SEGMENT (30 WORDS) LONG. THE FILE MAY ONLY HAVE ONE ROW AND SHOULD BE DECLARED IN THE COLD/START DECK.

LOGGING FOR DATA COMMUNICATIONS IS BYPASSED IF A FILE LABELED "REMOTE/LOG" IS NOT ON DISK. THE MESSAGE:

```
#NULL REMOTE/LOG
```

IS TYPED ON THE SPO WHEN THE MCP FIRST ATTEMPTS TO ACCESS THE FILE "REMOTE/LOG" AND THE FILE DOES NOT EXIST. RECOVERY IS PROVIDED BY ENTERING "WR" THROUGH THE SPO WHICH WILL CAUSE THE FILE "REMOTE/LOG" TO BE CREATED AND WILL CAUSE THE FOLLOWING MESSAGE TO BE OUTPUT ON THE SPO:

```
#REMOTE LOG ON DISK
```

THE FILE CREATED WILL BE 1000 SEGMENTS LONG.

THE FIRST RECORD OF THE FILE "REMOTE/LOG" (I.E., THE RECORD WITH RELATIVE ADDRESS 0) DESCRIBES THE REMAINDER OF THE FILE. THE CONTENTS OF RECORD 0 ARE:

WORD	DESCRIPTION
0	VALUE OF WORD EQUALS THE NUMBER OF LOGICAL RECORDS WRITTEN IN THE FILE "REMOTE/LOG".
1	VALUE OF WORD EQUALS THE RECORD CAPACITY (IN LOGICAL RECORDS) OF THE FILE "REMOTE/LOG".
2-4	RESERVED FOR EXPANTION.

THE REMOTE LOG MAINTENANCE PROCEDURS HAVE BEEN MODIFIED TO MORE CLOSELY RESEMBLE THOSE ASSOCIATED WITH OTHER LOG FILES AND TO PROVIDE A MORE READILY USABLE LOGGING FACILITY. A PROCEDURE HAS BEEN INPLEMENTED WHEREBY THE FILE REMOTE/LOG CAN BE RETAINED UNDER AOTHER NAME AND SUBSEQUENTLY RE-INITIALIZED TO GATHER ADDITIONAL LOGGING INFORMATION.

LR MESSAGE

THE LR KEYBOARD MESSAGE HAS BEEN ALTERED TO ALLOW THE SAVING OF REMOTE LOG FILES, AND ALTHOUGH THE FORMAT OF THE MESSAGE HAS NOT BEEN CHANGED, ITS ORIGINAL FUNCTION OF CALLING OUT THE LOG ANALYSIS PROGRAM, LOGOUTR/DISK, HAS BEEN ELIMINATED. AN LR MESSAGE WILL NOW CAUSE AN EXISTING REMOTE/LOG FILE TO BE GIVEN THE NEW NAME <M><D><C>/REMLOG; WHERE <M> IS A 2-DIGIT NUMBER REPRESENTING THE CURRENT MONTH, <D> IS A 2-DIGIT NUMBER REPRESENTING THE DAY OF THE MONTH, AND <C> IS A 3-DIGIT FILE NUMBER INCREMENTED AFTER EVERY LR MESSAGE. AFTER CREATION OF THE NEW LOG FILE, REMOTE/LOG IS INITIALIZED TO AN EMPTY CONDITION. EACH NEWLY CREATED REMLOG FILE WILL OCCUPY ONLY THE DISK SPACE NEEDED TO CONTAIN THE LOG ENTRIES PRESENT IN ITS PARENT LOG FILE, I.E., THE ABORT LOG SEGMENTS AND ANY UNUSED SEGMENTS WILL NOT BE TRANSFERRED TO THE NEW FILE.

UPON COMPLETION OF THE FILE MAINTENANCE OPERATIONS, A KEYBOARD MESSAGE WILL BE PRINTED GIVING THE NAME OF THE NEW FILE. THE FORMAT OF THE MESSAGE IS AS FOLLOWS:

****NEW REMOTE LOG FILE IS <M><D><C>/REMLOG.

WHEN THE REMOTE LOG IS ONE-HALF FULL, THE OPERATOR WILL BE NOTIFIED BY THE FOLLOWING MESSAGE:

*REMOTE/LOG HALF FULL.

A SIMILAR MESSAGE IS PRINTED WHEN THE LOG IS FULL.

AN LR KEYBOARD MESSAGE SHOULD BE ENTERED WHEN THE HALF FULL CONDITION OCCURS; LOG WRAP AROUND WILL BEGIN AFTER RECEIPT OF THE LOG FULL MESSAGE.

SHAREDISK OPERATION

WHEN OPERATING IN A SHAREDISK ENVIRONMENT, EACH SYSTEM WILL, IF GENERATED WITH THW DCLOG OPTION SET, HAVE ITS OWN REMOTE/LOG. EACH LOG WILL BE DEFFERENTIATED BY HAVING A SYSTEM DESIGNATION, I.E., A, B, C, OR D, AS THE LAST CHARACTER OF ITS <FID>, THUS FOR EXAMPLE, FOF SYSTEM A, REMOTE/LOGA WILL YIELD <M><D><C>/REMLOGA AFTER AN LR MESSAGE HAS BEEN ENTERED ON THAT SYSTEM.

FILE CREATION

THE FILE REMOTE/LOG CAN BE CREATED BY EITHER INCLUSION OF A PERTINENT FILE CARD GROUP IN THE COLD/START DECK OR BY ENTERING THE WR KEYBOARD MESSAGE. IN EITHER CASE, ONCE THE FILE HAS BEEN CREATED, IT WILL ALWAYS OCCUPY THE SAME AREA OF DISK; REMLOG FILES

MAY EXIST AYWHERE ON DISK.

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.

FIELDS OF THE WORD 0 ENTRY WILL BE DIFFERENT FOR DCP DATACOM SYSTEMS.

TYPE 1 LOG-OUT ENTRY

THE FOLLOWING INFORMATION IS ENTERED INTO THE FILE "REMOTE/LOG" WHEN A DATA COMMUNICATIONS STATION LOGS OUT.

WORD	FIELD	DESCRIPTION
0	.[9:4]	TERMINAL UNIT
	.[14:4]	BUFFER
	.[42:6]	CODE = 1
1		USERCODE
2		CURRENT DATE BCL - YYDDD FORMAT
3-4		UNUSED

TYPE 2 LOG-IN ENTRY

THE FOLLOWING INFORMATION IS ENTERED INTO THE FILE "REMOTE/LOG" WHEN A DATA COMMUNICATIONS STATION LOGS IN:

WORD	FIELD	DESCRIPTION
0	.[9:4]	TERMINAL UNIT
	.[14:4]	BUFFER
	.[42:6]	CODE = 2
1		USERCODE
2		CURRENT DATE BCL - YYDDD FORMAT
3		"XCLOCK" VALUE - TIME OF DAY AT LOG IN
4		UNUSED

TYPE 3 CONTROL CARD ENTRY

THE FOLLOWING INFORMATION IS ENTERED INTO THE FILE "REMOTE/LOG" WHEN A JOB IS SELECTED TO "RUN". EVERY "RUN" OR "EXECUTE" FROM A REMOTE STATION IS LOGGED.

WORD	FIELD	DESCRIPTION
0	.[9:4]	TERMINAL UNIT
	.[14:4]	BUFFER
	.[18:24]	RUN NUMBER *
	.[42:6]	CODE = 3
1-4		CONTENTS OF CONTROL CARD (MAXIMUM 31 CHARACTERS)

* ENTRIES IN THE FILE "REMOTE/LOG" CORRESPONDING TO ENTRIES IN THE FILE "SYSTEM/LOG" HAVE THE SAME "RUN NUMBER" WHERE A JOBS "RUN NUMBER" IS DEFINED TO BE ITS START TIME (IN 60THS OF A SECOND) AS SPECIFIED IN THE "SYSTEM/LOG".

TYPE 4 CONTROL CARD ENTRY

THE FOLLOWING INFORMATION IS ENTERED INTO THE FILE "REMOTE/LOG" WHEN A JOB IS SELECTED TO "RUN". EVERY "RUN" OR "EXECUTE" FROM A REMOTE STATION IS LOGGED. THIS ENTRY IS IDENTICAL TO THE TYPE 3 ENTRY EXCEPT THAT PROVISION IS MADE FOR CONTROL CARD INFORMATION 32-72 CHARACTERS IN LENGTH.

WORD	FIELD	DESCRIPTION
0	.[9:4]	TERMINAL UNIT
	.[14:4]	BUFFER
	.[18:24]	RUN NUMBER *
	.[42:6]	CODE = 4
1-9		CONTENTS OF CONTROL CARD (32-72 CHARACTERS)

* 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 FOLLOWING INFORMATION IS ENTERED IN THE FILE "REMOTE/LOG" WHEN A STATION DETACHES FROM A JOB:

WORD	FIELD	DESCRIPTION
0	.[2:1]	0 = ATTACHED BY "READ", "SEEK",OR "WRITE" 1 = ATTACHED BY ENTERING "RUN" OR "EXECUTE" CARD
	.[9:4]	TERMINAL UNIT
	.[14:4]	BUFFER
	.[18:24]	RUN NUMBER (SEE TYPE 3 & 4 ENTRIES)
	.[42:6]	CODE = 5
1		USERCODE
2		<MFID> OF OBJECT PROGRAM (7 CARACTERS)
3		<FID> OF OBJECT PROGRAM (7 CHARACTERS)
4		PROCESSOR TIME IN 60THS OF A SECOND I.E.,PROCESSOR TIME USED FOR THIS STATION, OUT OF THE TOTAL USED BY THE JOB
5		PRO-RATED TIME IN 60THS OF A SECOND; I.E.,PRO-RATED TIME USED BY THIS STATION OUT OF THE TOTAL USED BY THE JOB
6		I-O TIME IN 60THS OF A SECOND I.E., I-O TIME USED BY THIS STATION, OUT OF THE TOTAL USED BY THE JOB
7	.[3:21]	START DATE; .E., DATE WHEN JOB ATTACHED TO THIS STATION (BINARY - YYDDD FORMAT)
	.[27:21]	STOP DATE; I.E., DATE WHEN JOB DETACHED FROM STATION (BINARY - YYDDD FORMAT)
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 FOLLOWING INFORMATION IS ENTERED INTO THE FILE "REMOTE/LOG" SUBSEQUENT TO A HALT/LOAD:

WORD	FIELD	DESCRIPTION
0	.[2:1]	0 = ATTACHED BY "READ", "SEEK",OR "WRITE" 1 = ATTACHED BY ENTERING "RUN" OR "EXECUTE" CARD
	.[9:4]	TERMINAL UNIT
	.[14:4]	BUFFER
	.[18:24]	RUN NUMBER (SEE TYPE 3 & 4 ENTRIES)
	.[42:6]	CODE = 6
1		USERCODE
2		<MFID> OF OBJECT PROGRAM (7 CARACTERS)
3		<FID> OF OBJECT PROGRAM (7 CHARACTERS)
4		PROCESSOR TIME IN 60THS OF A SECOND I.E.,PROCESSOR TIME USED FOR THIS STATION, OUT OF THE TOTAL USED BY THE JOB
5		PRO-RATED TIME IN 60THS OF A SECOND; I.E.,PRO-RATED TIME USED BY THIS STATION OUT OF THE TOTAL USED BY THE JOB
6		I-O TIME IN 60THS OF A SECOND I.E., I-O TIME USED BY THIS STATION, OUT OF THE TOTAL USED BY THE JOB
7	.[3:21]	START DATE; .E., DATE WHEN JOB ATTACHED TO THIS STATION (BINARY - YYDDD FORMAT)
	.[27:21]	STOP DATE; I.E., DATE WHEN JOB DETACHED FROM STATION (BINARY - YYDDD FORMAT)
8		ATTACH TIME; I.E., TIME WHEN JOB ATTACHED TO STATION
9		DETACH TIME; I.E., TIME AT LAST HALT LOAD

CREATION OF REMOTE LOG ENTRIES

AS PREVIOUSLY INDICATED, 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 JOBS 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 TE TIME USED FOR 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 TIMES. 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 BEGING 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(TUBUFF,0) OR STATUS(TUBUFF,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 STATUSWORD. EACH TIME SUCH AN INTERROGATE IS PERFORMED, THE MCP CHECKS TO SEE IF THE TERMINAL BUFFER ADDRESS CURRENTLY IN THE PROGRAMS USERSTA LOCATION IS BIFFERENT FROM THE ONE SPECIFIED IN THE INTERROGATE STATEMENT. IF IT IS, THE OLD STATION IS CHARGED WITH ALL TIMES SINCE THE PREVICUS 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×(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×ABRTLNGTH-1. IN THE EVENT THAT A HALT/LOAD IS NECESSARY, THE ABORT AREA OF THE FILE "REMOTE/LOG" IS CHECKED TO DETERMINE IF ANY REMOTE TERMINALS WERE ATTACHED PRIOR TO THE HALT/LOAD SEQUENCE. ABORT INFORMATION IS FORMATTED AS A TYPE 6 ENTRY, AND PLACED IN THE FIRST (N-ABRTLNGTH) SEGMENTS.

LOGICAL UNIT NUMBERS

THE "MCP" ASSOCIATES ONE UNIQUE LOGICAL UNIT NUMBER WITH EACH I/O UNIT (THIS IS DIFFERENT FROM THE HARDWARE UNIT NUMBERS). 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" OPERATOR IS STORED IN THE FIELDS [17:31]. NUMBERING FROM RIGHT TO LEFT, BIT [47:1] IS NUMBERED 0, AND BIT [22:1] IS NUMBERED 25.

UNIT	LOGICAL UNIT NUMBER	UNIT	LOGICAL UNIT NUMBER	UNIT	LOGICAL UNIT NUMBER
MTA	0	LPB	21	CDL	42
MTB	1	CPA	22	CDM	43
MTC	2	CRA	23	CDN	44
MTD	3	CRB	24	CDP	45
MTE	4	SPO	25	CDQ	46
MTF	5	PPA	26	CDR	47
MTH	6	PRA	27	CDS	48
MTJ	7	PPB	28	CDT	49
MTK	8	PPB	29	CDU	50
MTL	9	DCA	30	CDV	51
MTM	10	XXX	31	CDW	52
MTN	11	CDA	32	CDX	53
MTP	12	CDB	33	CDY	54
MTR	13	CDC	34	CDZ	55
MTS	14	CDD	35	CD2	56
MTT	15	CDE	36	CD3	57
DRA	16	CDF	37	CD4	58
DRB	17	CDG	38	CD5	59
DKA	18	CDH	39	CD6	60
DKB	19	CDJ	40	CD7	61
LPA	20	CDK	41	CD8	62
				CD9	63
				MTX	64

LOOKQ

"LOOKQ" IS A VARIABLE IN THE MCP-S PRT WHICH IS THE HEAD OF A LINKED-LIST OF LOGGED-IN USERS OF B487 REMOTE DEVICES. 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 TE MCP VARIABLE "LOOKQ" 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

THE FORMAT OF A "LOOKQ" ENTRY IS:

WORD	FIELD	CONTENTS
	(@4 AND @6)	
0		PRIMARY MEMORY LINK WORD (SAVE)
1		SECONDARY MEMORY LINK WORD
	[0:9]	0
	[9:9]	TERMINAL UNIT AND BUFFER NUMBER
	[18:15]	ADDRESS OF SECONDARY LINK WORD OF NEXT ENTRY
	[33:15]	ADDRESS OF SECONDARY LINK WORD OF LAST ENTRY
2		USER CODE
3		CCMASK1
4		CCMASK2
5		INFOMASK1
6		INFOMASK2
7		MIXMASK
8		TIME WHEN USER LOGGED-IN IN 60THS OF A SECOND
9		RESERVED FOR EXPANSION

- LSLATE, LQAVAIL, LQUE -

LSLATE

POINTER TO LAST ENTRY PLACED IN THE SLATE.

LQAVAIL

CONTAINS AN INDEX SPECIFYING FIRST EMPTY WORD IN THE LQUE.

LQUE

DESCRIPTOR POINTING TO THE LQUE ARRAY WHICH CONTANS THE ADDRESS AND INDEX INTO THE IOQUE FOR EVERY DISK I/O THAT RESULTED IN A LOCKED ADDRESS THAT HASN'T BEEN UNLOCKED.

MAKEPRESENT

MAKEPRESENT(C) IS A PROCEDURE WHICH FORCES CORE PRESENCE FOR CODE, DATA, AND I/O BUFFER INFORMATION WHEN A NON-PRESENT DESCRIPTOR ARISES INNORMAL STATE (ESPB IT HANDLES CONTROL STATE CODE). "C" IS THE PROCEDURE DESCRIPTOR.

MAINTLOGARRAY

AN ARRAY DESCRIPTOR FOR A 30 WORD ARRAY USED IN WRITING TO THE MAINTENANCE LOG ("MAINT/LOG").

MCP

VARIABLE CONTAINING THE USER IDENTIFICATION OF THE PRIVILEGED USER OBTAINED FROM THE REMOTE/USER FILE. IF THE FILE IS NOT PRESENT, THE DEFAULT VALUE OF (NOT 0) IS USED.

MCPBASE

CONTAINS THE DISK ADDRESS (OCTAL) OF THE BEGINNING OF THE MCP THAT IS CURRENTLY IN USE. THIS ADDRESS IS PASSED TO THE MCP BY THE LOADER ROUTINE AT EACH HALT/LOAD IN M[0].[18:30]. WHEN THE ESPBIT ROUTINE IS CALCULATING THE DISK ADDRESS OF AN MCP SEGMENT, IT ADDS MCPBASE TO THE ADDRESS THAT IS CONTAINED IN THE PRT CELL FOR THAT SEGMENT.

MCPTOP

CONTAINS DISK ADDRESS OF END OF FILE CONTAINING THE CURRENT MCP, FOR USE OF STATISTICS CODE.

MDELTA

A POINTER USED IN STORING 5 WORD RECORDS INTO A 30 WORD ARRAY FOR THE MAINTENANCE LOG.

MLOG

A POINTER USED TO KEEP TRACK OF THE NUMBER OF DISK SEGMENTS IN THE FILE "MAINT/LOG".

MEMASK

DESCRIPTOR POINTING TO MEMASK ARRAY USED TO INDICATE WHICH MEMORY MODS ARE CONSIDERED "UP" AND WHICH ARE CONSIDERED "DOWN".

MEMORY

@ 50000000 00000000.

MEMORY CONTENTS - CELLS 0-200 (OCTAL)

CELL	CONTENTS
0	SAME FORMAT AS MEMORY LINK (USED AT HALT LOAD TO PASS THE BASE ADDRESS OF THE MCP TO THE MCP AND SYSNO IN [16:12])
1	MCP
2	MSCW (PLACED BY THE KERNEL)
3	PATCH LEVEL IN DECIMAL (ALSO USED FOR DCP INITIALIZATION)
4	USED FOR DCP INITIALIZATION
5	JUNK - USED BY THE MCP
6	USED BY TAPEPARITYRETRY RETRY SOMETIMES CONTAINS THE WORD COUNT
7	MSCW - COPY OF THE LAST MSCW FOR WHICH MSFF WAS TRUE
10	INCW FOR P2 OR I-O DESCRIPTOR ADDRESS
11	MDELTA
12	MLOG
13	MROW
14	I-O 1 RESULT DESCRIPTOR
15	I-O 2 RESULT DESCRIPTOR
16	I-O 3 RESULT DESCRIPTOR
17	I-O 4 RESULT DESCRIPTOR
20	FIRST CELL READ FROM ON HALT LOAD
21	UNDEFINED
22	TIME INTERVAL INTERRUPT (CCI03F)
23	I-O BUSY INTERRUPT (CCI04F)
24	KEYBOARD REQUEST INTERRUPT (CCI05F)
25	PRINTER 1 FINISHED INTERRUPT (CCI06F)
26	PRINTER 2 FINISHED INTERRUPT (CCI07F)
27	I-O 1 FINISHED INTERRUPT (CCI08F)
30	I-O 2 FINISHED INTERRUPT (CCI09F)
31	I-O 3 FINISHED INTERRUPT (CCI10F)
32	I-O 4 FINSIHED INTERRUPT (CCI11F)
33	P2 BUSY INTERRUPT (CCI12F)
34	INQUIRY REQUEST INTERRUPT (CCI13F)
35	SPECIAL INTERRUPT 1 (CCI14F)
36	READ AFTER WRITE CHECK 1 INTERRUPT (CCI15F) REPORT FREE ADDRESS INTERRUPT ON SHAREDISK SYSTEMS
37	READ AFTER WRITE CHECK 2 INTERRUPT (CCI16F)
40	P2 MEMORY PARITY ERROR INTERRUPT (PK-I01F)
41	P2 INVALID ADDRESS INTERRUPT (PK-I02F)
42	NOT USED

- MEMORY CONTENTS - CELLS 0-200 (OCTAL) -

```
43          P2 STACK OVERFLOW INTERRUPT (PK=I03F)
44          P2 COMMUNICATE INTERRUPT (PK=BCD 4)
45          P2 PROGRAM RELEASE INTERRUPT (PK=BCD 5)
46          P2 CONTINUITY BIT INTERRUPT (PK=BCD 6)
47          P2 PRESENSE BIT INTERRUPT (PK=BCD 7)
50          P2 FLAG BIT INTERRUPT (PK=BCD 8)
51          P2 INVALID INDEX INTERRUPT (PK=BCD 9)
52          P2 EXPONENT UNDERFLOW INTERRUPT (PK=BCD 10)
53          P2 EXPONENT OVERFLOW INTERRUPT (PK=BCD 11)
54          P2 INTEGER OVERFLOW INTERRUPT (PK=BCD 12)
55          P2 DIVIDE BY ZERO INTERRUPT (PK=BCD 13)
56          LOGHOLDER
57          NUMAINTMESS
60          P1 MEMORY PARITY ERROR INTERRUPT (PK=I01F)
61          P1 INVALID ADDRESS INTERRUPT (PK=I02F)
62          P1 STACK OVERFLOW INTERRUPT (PK=I03F)
63          LOGENTRY
64          P1 COMMUNICATE INTERRUPT (PK=BCD 4)
65          P1 PROGRAM RELEASE INTERRUPT (PK=BCD 5)
66          P1 CONTINUITY BIT INTERRUPT (PK=BCD 6)
67          P1 PRESENSE BIT INTERRUPT (PK=BCD 7)
70          P1 FLAG BIT INTERRUPT (PK=BCD 8)
71          P1 INVALID INDEX INTERRUPT (PK=BCD 9)
72          P1 EXPONENT UNDERFLOW INTERRUPT (PK=BCD 10)
73          P1 EXPONENT OVERFLOW INTERRUPT (PK=BCD 11)
74          P1 INTEGER OVERFLOW INTERRUPT (PK=BCD 12)
75          P1 DEVIDE BY ZERO INTERRUPT (PK=BCD 13)
76          NXDISK
77          MAINTLOGREADY
100         BASE OF MCP "STACK"
101-157     "MCP" STACK
101         ONEOHONE
102         ONEOHTWO
133         TEMPORARY DESCRIPTOR TO EUIO
            USED ONLY DURING A HALT LOAD
160         NT1
161         NT2
162         NT3
163         NT4
164         NT5
165         NT6
166         NT7
167         DATE
170         CLOCK
171         XCLOCK
172         READY
173         RESERVED DISK ADDRESS
174         LASTCDNUM
175         FIRSTDECK
176         LASTDECK
177         DIRDSK
```

- MEMORY CONTENTS - CELLS 0-200 (OCTAL) -

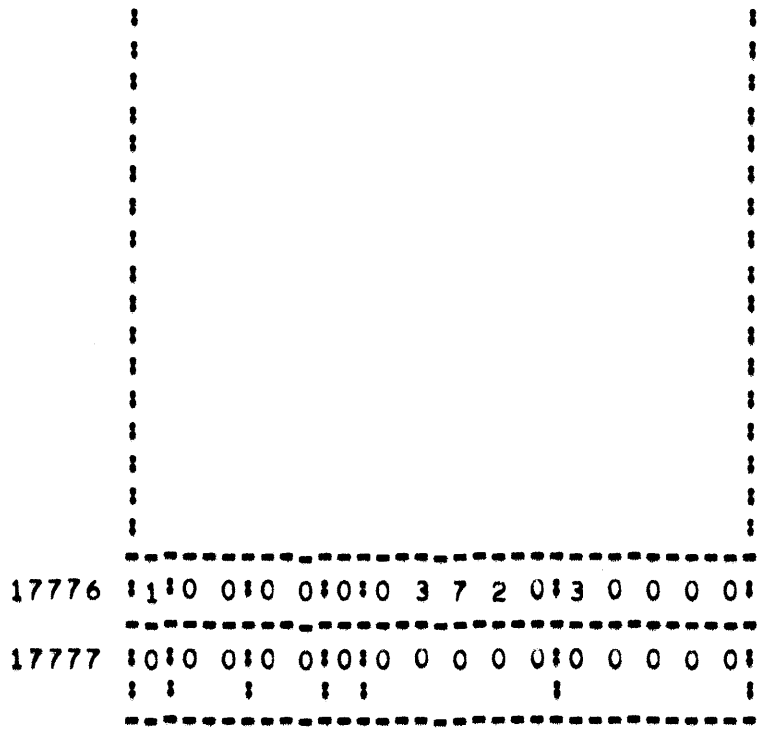
200 PRT FOR "MCP"

- MEMORY LAYOUT -

MEMORY LAYOUT
------ ------
MODULE 0

```
        ---------------------------------
00000   |1|0 1|0 0|0|4 7 7 7 5|0 3 7 2 0|
        ---------------------------------
00001   |0|0 0|0 0|0|0 0 0 0 0|0 0 0 0 0|
        ---------------------------------
        |                               |
        |                               |
        |                               |
        |                               |
        ---------------------------------
03720   |2|0 0|0 0|0|0 0 0 0 0|1 7 7 7 6|
        ---------------------------------
03721   |0|0 0|0 0|0|1 4 0 5 4|3 0 0 0 1|
        ---------------------------------
03722   |0|0 0|0 0|0|0 0 0 0 0|4 7 7 7 6|
        ---------------------------------
        |                               |
        |                               |
        |                               |
        |                               |
        |                               |
        |                               |
        |                               |
        |                               |
        |                               |
        |                               |
        |                               |
        |                               |
        |                               |
        |                               |
        |                               |
        |                               |
        |                               |
        ---------------------------------
                  MODULE 1
        ---------------------------------
        |                               |
        |                               |
        |                               |
        |                               |
        |                               |
        |                               |
        |                               |
        |                               |
        |                               |
        |                               |
```

- MEMORY LAYOUT -



CORE MEMORY AT HALT-LOAD TIME

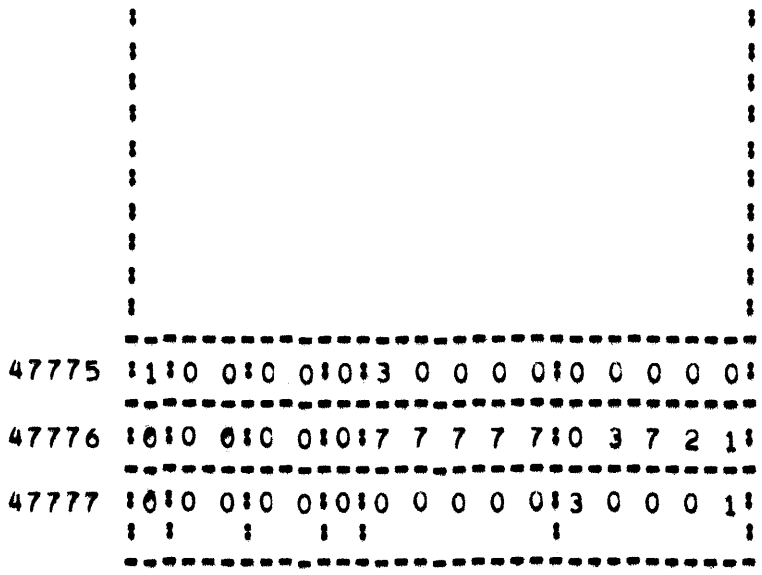
- MEMORY LAYOUT -

MODULE 3

30000	2	0	0	0	0	0	1	7	7	7	6	4	7	7	7	5
30001	0	0	0	0	0	0	1	7	7	7	3	4	7	7	7	6
30002	0	0	0	0	0	0	0	0	0	0	0	0	3	7	2	1

MODULE 4

- MEMORY LAYOUT -



MODULES 0, 1, 3, AND 4 ON LINE

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 UNASIGNED AREA. A MEMORY LINK FOR AVAILABLE STORAGE OCCUPIES THREE WORDS. THESE WORDS PROVIDE THE FOLLOWING INFORMATION:

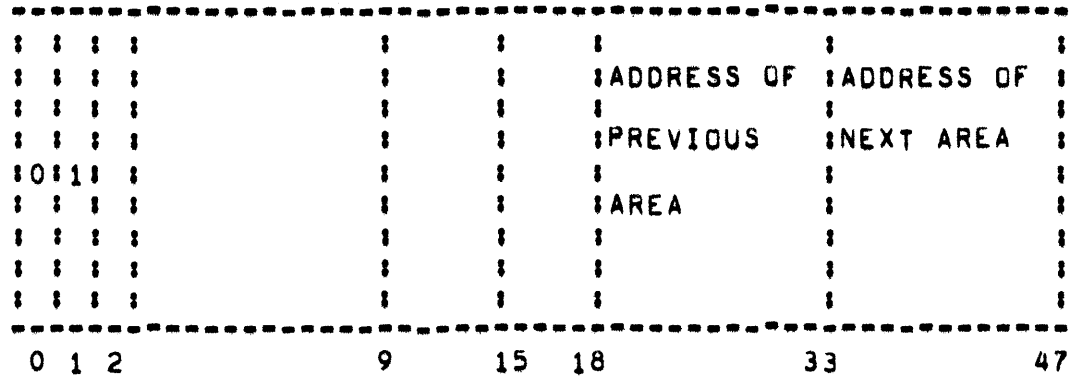
A. THEY SPECIFY THAT THE AREA IS AVAILABLE.
B. THEY SPECIFY THE SIZE OF THE AREA.
C. THEY PROVIDE THE ADDRESS OF THE FOLLOWING AVAILABLE AREA.

WHEN CORE STORAGE IS CLASSIFIED AND ORGANIZED FOR THE FIRST TIME AFTER A HALT/LOAD, 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.

LINKS FOR AVAILABLE AREA (3 WORDS)

FIELD VALUES AND FUNCTIONS

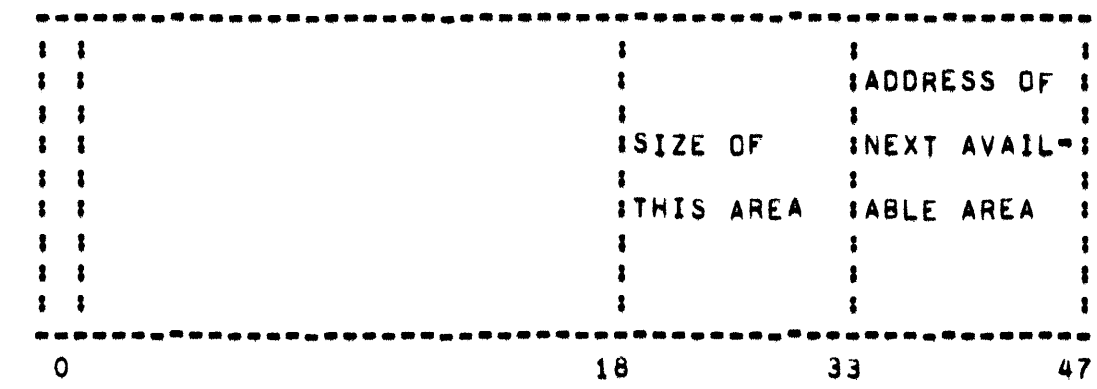
WORD 1



FIELD	FIELD VALUE	FUNCTION
[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 1ST WORD OF LINK FOR PREVIOUS AREA
[33:15]	ADDRESS	ADDRESS OF 1ST WORD OF LINK FOR NEXT AREA
	NA*	= NOT APPLICABLE

WORD 2



FIELD	FIELD VALUE	FUNCTION
[0:18]	0	ZEROS REQUIRED BY "LLL" OPERATOR
[18:15]	VARIES	SIZE OF AVAILABLE AREA
[33:15]	ADDRESS	ADDRESS OF 2ND WORD IN LINK FOR NEXT AVAILABLE AREA

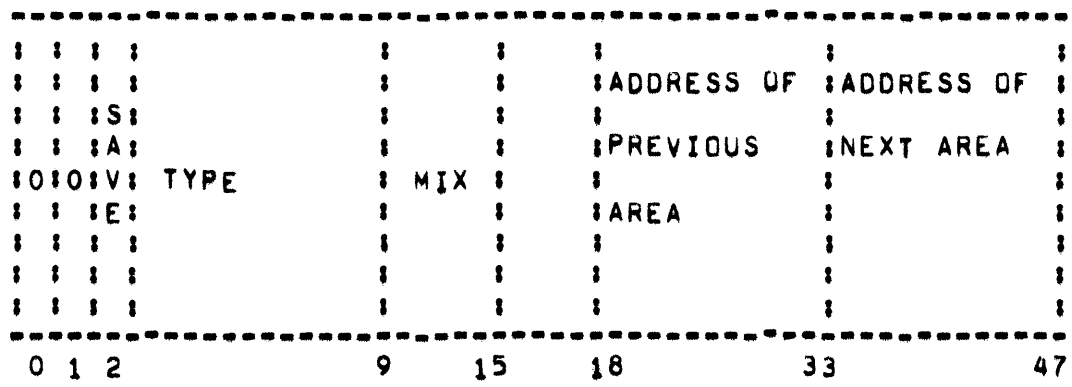
WORD 3

FIELD	FIELD VALUE	FUNCTION
[0:33]	0	NONE
[33:15]	ADDRESS	ADDRESS OF 2ND WORD IN LINK FOR PREVIOUS AVAILABLE AREA

LINK FOR IN-USE AREA (2 WORDS)

FIELD VALUES AND FUNCTIONS

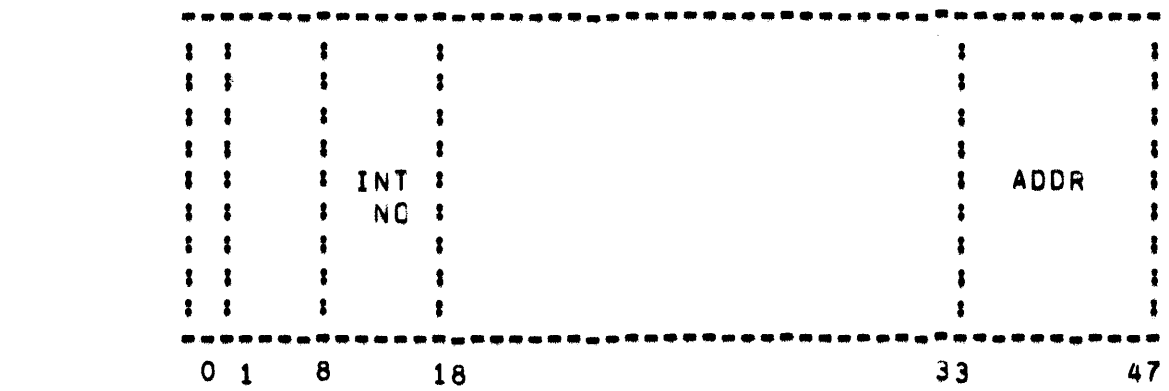
WORD 1



FIELD	FIELD VALUE	FUNCTION
[0:1]	0	FLAG BIT
[1:1]	0	AVAILABILITY
[2:1]	0/1	SAVE (1 = "SAVE" AREA)
[3:6]	1-63	TYPE OF AREA 0 = DEFALT 1 = PROGRAM, SEGMENT DICTIONARY 2 = DATA 3 = I/O BUFFERS 4 = ALGOL FIB AND TANK 5 = FILL WITH INQUIRY BUFFER OR ATTACHED ROWS OR STATION TABLE ROWS 6 = COBOL OR SORT FILE FIB 7 = INTRINSIC (IF SAVE) 8 = DISK FILE HEADER 9 = INFO USED BY DISKERROR 10 = EGGSELECT STACK 12 = INDEPENDENT PROCESS STACKS 13 = INTRINSIC CALLED BY ANOTHER INTRINSIC 20 = CIDROW OR ATTACHED LABEL 21 = SNAPSHOT (BREAKOUT/RESTART)
[9:6]	0 - MIXMAX	MIX INDEX OF PROGRAM USING AREA, INTRINSICS & MCP MIX=0 RE-ENTRANT JOB HAS MIX OF FIRST JOB

[15:3]	0=DELTA	EXCESS AREA SPACE
[18:15]	ADDRESS	ADDRESS OF 1ST WORD OF LINK FOR PREVIOUS AREA
[33:15]	ADDRESS	ADDRESS OF 1ST WORD FOR NEXT AREA

WORD 2



FIELD	FIELD VALUE	FUNCTION
[8:10]	INTRINSIC	INTRINSIC NUMBER (INTRINSICS ONLY)
[18:15]	VARIES	PROGRAM SEGMENT - SIZE IF CIDROW OR LABEL (TYPE 20), CIDTABLE INDEX.
[33:15]	ADDRESS	IF DATA, ADDRESS OF ARRAY DESCRIPTOR. IF OBJECT PROGRAM, SEGMENT NUMBER. IF BUFFER AREA, TOP I/O DESCRIPTOR. IF "MCP" PROGRAM SEGMENT, "PRT" ADDRESS.

MEMORY ORGANIZATION

THE CONTENTS OF MEMORY IS ORGANIZED BY A MECHANISM KNOWN AS A MEMORY LINK. A MEMORY LINK IDENTIFIES THE CONTENTS OF THE MEMORY AREA IMMEDIATELY FOLLOWING IT AND THE SIZE OF THE AREA. ALL AREAS IN MEMORY, WHETHER AVAILABLE OR IN-USE, ARE IDENTIFIED THROUGH THIS STRUCTURE. EACH MEMORY LINK ALSO CONTAINS THE ADDRESS OF THE PRECEEDING AND FOLLOWING LINK SO THAT THE ENTIRE CONTENTS OF MEMORY CAN BE QUICKLY KNOWN BY SCANNING THE 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 WOTH 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 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 THEN 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 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 NOT CURRENTLY IN USE. SUCH STORAGE CAN BE ASSIGNED AS NEEDED.

THE FORMAT OF THE MEMORY LINKS APPEARS UNDER "MEMORY LINKS".

MEND

POINTER TO LAST STORAGE LINK IN MEMORY.

MESS

MESS(T) IS DEFINED AS PUNT(I).

MESSAGEHOLDER

MESSAGEHOLDER IS THE HEAD OF A QUEUE OF SPO MESSAGES AND HAS THE FOLLOWING FORMAT:

FIELD	CONTENTS
[18:15]	POINTS TO THE LAST BUFFER ADDED TO THE SPO QUEUE. THE FIRST WORD OF EACH MESSAGE STARTING AT MESSAGEHOLDER.[33:15] IS A SECONDARY 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 ZEROS IN THE FIRST WORD.
[33:15]	POINTS TO THE SPO MESSAGE THAT IS CURRENTLY BEING PRINTED OR TO THE NEXT SPO MESSAGE TO BE PRINTED, IF NONE IS ACTUALLY BEING PRINTED. IT WILL CONTAIN A "1" IF IN THE PROCESS OF LINKING.

A MAXIMUM OF 100 MESSAGES OF VARIABLE LENGTH, DEPENDING ON THE ROUTINE THAT CALLS SPOUT, MAY BE PLACED IN THE SPO QUEUE. (NUMESS + 100) IS THE NUMBER OF MESSAGES LEFT IN THE SPO QUEUE, THUS, IF NUMESS EQUALS -100, THE SPO QUEUE IS EMPTY. THE MESSAGES MUST CONTAIN A GROUP MARK (LEFTARROW). IF THE GROUP MARK IS OMITTED, THE SPO WRITE WILL CONTINUE UNTIL A GROUP MARK IS ENCOUNTERED SOMEWHERE IN CORE.

SPOUT IS THE ROUTINE WHICH PLACES INFORMATION INTO THE MESSAGEHOLDER QUEUE. IT DOES THIS WHEN IT DESIRES TO SEND A MESSAGE TO BE TYPED ON THE SPO. MESSAGEWRITER IS THE ROUTINE WHICH NORMALLY REMOVES INFORMATION FROM THE MESSAGEHOLDER QUEUE AFTER A MESSAGE IS TYPED. KEYIN WILL ALSO FLUSH THE QUEUE WHEN A "BK" IS ENTERED THROUGH THE SPO.

MESSAGEWRITER

MESSAGEWRITER IS A PROCEDURE WHICH IS INITIATED AS AN INDEPENDENTRUNNER BY SPOUT WHEN IT ADDS AN ENTRY TO MESSAGEHOLDER AND DISCOVERS THAT MESSAGEWRITER IS NOT ALREADY RUNNING. MESSAGEWRITER TERMINATES ONLY WHEN ITS QUEUE IS EMPTY.

MIXMASK

MASK FOR DEFAULT LEGAL INPUT MIX MESSAGES FROM REMOTES.

MIXNUM

CONTAINS MIX NUMBER OF PROGRAM FOR WHICH STATISTICS CODE IS GATHERING INFORMATION

MSTART

CONTAINS THE ADDRESS OF THE FIRST AREA OF STORAGE AFTER THE END OF THE MCP SAVE PROCEDURES AND THE OUTER BLOCK CODE.

MTXIN

MTXIN(I,U,BUFF) IS A PROCEDURE WHICH CREATES A LOGICAL UNIT NUMBER, "U", AND MASK, "I", FOR THE APPROPRIATE UNIT MNEMONIC SPECIFIED IN "BUFF". THE UNIT IS ALSO CHECKED TO DETERMINE IF IT IS READY.

MULTITABLE

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, MULTITABLE, AND 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 THE TABLES.

- NAMEID, NEUP, NEXTSLOT, NEXTWAIT -

NAMEID

NAMEID(A,KTR) IS A PROCEDURE WHICH PLACES AN IDENTIFIER FROM "KTR" INTO "A". A ZERO IS PLACED IN THE MOST SIGNIFICANT CHARACTER POSITION AND THE IDENTIFIER IS LEFT JUSTIFIED WITH SPACE (BLANK) FILL.

NEUP

CONTAINS INFORMATION OF THE NUMBER OF EU-S ON THE SYSTEM:
.[1:1] = UNUSED.
.[18:15] = TOTAL NO. OF EU-S ON DKA AND DKB.
.[33:15] = NUMBER OF EU-S ON DKA.

NEXTSLOT

VARIABLE USED TO INDICATE THE NEXT AVAILABLE POSITION FOR DISK FILE HEADER ENTRY INTO THE DISK DIRECTORY.

NEXTWAIT

POINTER INTO WAITQUE AT NEXT AVAILABLE SLOT.

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)
                           .[1:17] = CLOCK TIME AT BOJ.
                           .[18:15] = CORE ESTIMATE DIV 64.
                           .[33:15] = LOCATION OF STACKBOTTOM
```

NOPROCESSTOG

NEGATIVE IF NORMAL STATE PROCESSING IS ALLOWED.

NSECOND

NSECOND IS A PROCEDURE WHICH IS EVOKED WHEN A CERTAIN NUMBER OF TIMER INTERRUPTS OCCUR. THE NUMBER IS DEPENDENT ON THE NUMBER OF PROCESSORS. IT TERMINATES JOBS SO INDICATED AND EXAMINES THE BED TO "WAKE" JOBS UP. THE PROCEDURE ALSO UPDATES THE TIME FOR JOBS THAT ARE RUNNING.

NSLATE

POINTER TO LAST ENTRY WHICH WAS STARTED FROM SLATE.

NT1 - NT7

TEMPORARY STORAGE FOR MCP

NUMAINTMESS

COUNTER OF NUMBER OF REQUESTS IN QUEUE FOR THE MAINTENANCE LOG.

NUMESS

COUNTER FOR NUMBER OF UNPROCESSED MESSAGES WAITING TO BE TYPED ON THE SPO AND/OR ALTERNATE SPO'S. IT IS BIASED BY MINUS 100.

NXDISK

POINTER INTO CIRCULAR BUFFER IN ISTACK FOR RECOVERABLE DISK AND DRUM ERROR ENTRIES FOR THE MAINTENANCE LOG.

OBJECT PROGRAM I/O FACILITIES

INTRODUCTION

THE HANDLING OF OBJECT PROGRAM I/O FACILITIES ON THE B5700 IS A FUNCTION OF THE MCP AS WELL AS THE OBJECT PROGRAM. IT IS THE RESPONSIBILITY OF THE OBJECT PROGRAM TO SPECIFY, FOR EACH FILE USED, THE FILE HANDLING TECHNIQUES SUCH AS:

1. NUMBER OF BUFFER AREAS TO USE.
2. SIZE OF BUFFER AREAS.
3. BLOCKING TECHNIQUES.
4. IN THE CASE OF DISK, RECORD ACCESSING TECHNIQUE (I.E., SERIAL, RANDOM, AND UPDATE).

ALSO, FOR EACH I/O STATEMENT, THE OBJECT PROGRAM MUST SPECIFY THE FILE AND I/O ACTION TO BE USED, AND PROVIDE THE DATA FOR THE FILE'S BUFFER AREAS. HOWEVER, IT IS THE RESPONSIBILITY OF THE MCP TO LOCATE FILES, TO PROVIDE BUFFER AREAS AND HANDLER THEIR USE, TO PERFORM BLOCKING AD RECORD ACCESSING, AND TO EXECUTE I/O OPERATIONS TO READ OR WRITE THE FILES.

I/O INTRINSICS

IN THE AREA OF I/O, THE SYSTEM PROVIDES A NUMBER OF PROCEDURES THAT EXECUTE IN NORMAL STATE RATHER THAN CONTROL STATE. SUCH PROCEDURES ARE CALLED "INTRINSICS". WHEN A COMPILER DETERMINES THAT A PROGRAM REQUIRES THE USE OF AN INTRINSIC, A PROCEDURE DESCRIPTOR FOR THE INTRINSIC IS PLACED IN THE PROGRAM'S PRT. THE PROGRAM CAN THEN MAKE CALLS ON THE INTRINSIC IN THE SAME FASHION AS ANY OTHER PROCEDURE.

SPECIFICATION OF FILE HANDLING TECHNIQUES

FILE HANDLING TECHNIQUES FOR OBJECT PROGRAM FILES ARE SPECIFIED IN THE SOURCE LANGUAGE REPRESENTATION OF THE PROGRAM. IN A SOURCE PROGRAM BEFORE THE FILE IDENTIFIERS ARE USED IN I/O STATEMENTS, EACH FILE IDENTIFIER IS ASSOCIATED WITH THE FILE HANDLING TECHNIQUES TO BE USED WITH THE FILE. IN THIS SECTION OF THE PROGRAM, THE FILE IDENTIFIER IS ALSO ASSOCIATED WITH THE FILE NAME OF THE FILE CONCERNED. HOWEVER, AT RUN TIME, IT IS POSSIBLE TO ASSOCIATE A FILE IDENTIFIER WITH A DIFFERENT FILE NAME.

FILE AND FILE NAME.

IN RESPECT TO FILE NAMES, THERE ARE TWO TYPES ON THE B5700; STANDARD FILES AND NON-STANDARD FILES. A STANDARD FILE IS A FILE WHICH HAS A FILE NAME PHYSICALLY ASSOCIATED WITH IT. A NON-STANDARD FILE IS A FILE THAT REQUIRES OUTSIDE INTERVENTION TO HAVE A NAME ASSOCIATED WITH IT. IN THE CASE OF FILES ON DISK, ALL FILES ARE STANDARD. NAMES ARE ASSOCIATED WITH THESE FILES THROUGH THE DISK DIRECTORY. NAMES ARE ASSOCIATED WITH OTHER STANDARD FILES BY STANDARD LABELS. A STANDARD LABEL IS A RECORD WITH A GIVEN FORMAT THAT APPEARS AS THE FIRST RECORD IN A FILE. ONE OF THE ENTRIES IN A STANDARD LABEL IS THE FILE-S NAME. FILES THAT DO NOT HAVE STANDARD LABELS ARE NON-STANDARD FILES. AN EXAMPLE OF A NON-STANDARD WOULD BE A MAGNETIC TAPE WITHOUT A STANDARD LABEL. TO ASSOCIATE A FILE NAME WITH A NON-STANDARD FILE REQUIRES THAT SPECIAL INFORMATION BE PROVIDED TO ASSOCIATE THE FILE NAME WITH THE I/O UNIT WHERE THE FILE IS LOCATED. THE INFORMATION MAY BE SUPPLIED THROUGH USE OF A LABEL EQUATION CARD OR AN OPERATOR INPUT MESSAGE.

FILE PARAMETER BLOCK.

EACH PROGRAM FOR THE B5700 HAS A FILE PARAMETER BLOCK (FPB). THE FPB IS CREATED WHEN A PROGRAM IS COMPILED, AND 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 INDENTIFIER, IS THEN USED IN ALL REFERENCES MADE TO THE CORRESPONDING FILE BY THE OBJECT PROGRAM. FOR EACH FILE NUMBER, 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 FILE NUMBER. THE LOCATION AND SIZE OF THE COMPILER 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 LAGUAGE REPRESENTATION OF A PROGRAM. EACH LABEL EQUATION CARD CONTAINS THE FILE IDENTIFIER CONCERNED, AND THE EQUATION INFORMATION WHICH INCLUDES THE MULTIPLE FILE IDENTIFICATION AND FILE IDENTIFICATION TO BE ASSOCIATED WITH THE FILE IDENTIFIER. WHEN SELECTION OBTAINS A PROGRAM-S COMPILER FPB, IT ALSO OBTAINS ALL LABEL EQUATION CARDS FOR THE PROGRAM, IF ANY. THEN THE FILE IDENTIFIERS IN THE COMPILER FPB ENTRIES ARE COMPARED WITH THE FILE IDENTIFIERS ON LABEL EQUATION CARDS. IF A MATCH IS FOUND, INFORMATION IN THE COMPILER 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 COMPILER 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 DESCRIPTOR AND A FILE NUMBER, THE OBJECT PROGRAM IS ABLE TO MAKE ALL NECESSARY REFERENCES TO FPB ENTRIES.

FILE INFORMATION BLOCKS.

AT RUN TIME, THERE IS ONE FILE INFORMATION BLOCK (FIB) GENERATED FOR EACH FILE TO BE USED BY A PROGRAM. AN FIB IS GENERATED BY AN OBJECT PROGRAM WHENEVER A FILE DECLARATION IS ENCOUNTERED IN THE CORRESPONDING 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. THIS AREA IS COMMONLY REFERRED TO AS THE "TANK".

LOGICAL UNIT NUMBERS.

AS IS NOTED IN THE SECTION ABOUT THE STATUS PROCEDURE, THE BURROUGHS B5700 HAS A SPECIAL SYLLABLE WHICH, WHEN EXECUTED, PROVIDES A RESULT GIVING THE STATUS OF EVERY PERIPHERAL UNIT. THE BITS IN THIS RESULT WORD ARE ASSUMED TO BE NUMBERED FROM RIGHT TO LEFT, STARTING WITH 0. THE NUMBER OF THE BIT REPRESENTING A PARTICULAR I/O UNIT IS TAKEN TO BE THAT UNIT-S LOGICAL UNIT NUMBER. REFERENCES TO I/O UNITS MADE BY THE I/O PROCEDURES AND ROUTINES OF THE DF MCP ARE MADE THROUGH USE OF THE UNIT-S LOGICAL NUMBER.

FILE NAMES VS I/O UNITS.

THE ASSOCIATION OF FILE NAMES WITH FILES ON DISK THROUGH USE OF THE DISK DIRECTORY IS STRAIGHTFORWARD SINCE DISK FILE LOCATIONS ARE RELATIVELY STATIC. THE ASSOCIATION OF FILE NAMES WITH I/O UNITS OTHER THAN DISK, HOWEVER, REQUIRES CHECKING EACH TIME STATUS NOTES THAT A UNIT NOT IN-USE IS MADE READY. SINCE FILES ON UNITS SUCH AS MAGNETIC TAPE UNITS ARE PROVIDED BY A SYSTEM OPERATOR, THE MCP MUST HAVE A DYNAMIC DIRECTORY FOR KEEPING A RECORD OF WHAT FILES, IF ANY, ARE ON WHAT UNITS. THIS DIRECTORY IS, IN FACT, MADE UP OF THREE TABLES: THE LABELTABLE, THE MULTITABLE, AND THE RDCTABLE. THERE IS AN ENTRY IN EACH OF THESE TABLE FOR EACH UNIT THAT MAY BE ON THE SYSTEM. ENTRIES WITHIN EACH TABLE ARE KEPT IN LOGICAL UNIT NUMBER ORDER.

THE LABELTABLE IS THE PRIMARY TABLE IN THE GROUP. A UNIT-S ENTRY IN THIS TABLE SPECIFIES ONE OF THE FOLLOWING: (1) THE UNIT IS NOT

READY, (2) 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), (3) 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 (4) 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 RDCTABLE CONTAINS THE REEL NUMBER, PURGE DATE, AND CYCLE NUMBER OF THE FILE, IF ANY, ON THE UNIT REPRESENTED BY THE TABLE ENTRY. THE PRNTABLE CONTAINS AN INDICATION OF WHETHER OR NOT A TAPE HAS A WRITE-RING, THE PHYSICAL REEL NUMBER OF THE TAPE, AND OTHER INFORMATION. INFORMATION IN THE LABELTABLE, THE MULTITABLE, AND RDCTABLE IS OBTAINED FROM THE STANDARD LABELS ON THE FILES, IF THE FILES ARE SO LABELED; OTHERWISE, THE INFORMATION CAN BE SUPPLIED THROUGH USE OF LABEL EQUATION CARDS OR OPERATOR MESSAGES. THE STATUS PROCEDURE HAS THE PRIMARY RESPONSIBILITY OF MAINTAINING THESE TABLES.

OPENING A FILE.

WHEN A PROGRAM FIRST REQUIRES A FILE, THE FILE MUST BE "OPENED". THE PROCESS OF OPENING A FILE INCLUDES LOCATING THE FILE AND PROVIDING BUFFER AREAS. WHEN A FILE IS TO BE OPENED, A MCP FILEOPEN PROCEDURE IS CALLED. IF A PROGRAM REQUIRES AN INPUT FILE, THE FILEOPEN PROCEDURE LOCATES THE PROGRAM-S FPB TO OBTAIN THE NAME OF THE FILE. IF THE FILE IS A DISK FILE, THE DISK DIRECTORY IS SEARCHED AND THE FILE HEADER FOR THE DESIRED FILE IS READ INTO CORE. OTHERWISE, THE LABELTABLE, THE MULTITABLE, AND THE RDCTABLE ARE SEARCHED TO LOCATE THE UNIT CONTAINING THE FILE AND MARK IT IN-USE. IF A FILE CANNOT BE LOCATED, A MESSAGE IS TYPED TO NOTIFY THE OPERATOR. THEN BUFFER AREAS, AS SPECIFIED IN THE FILE-S FIB, ARE OBTAINED BY CALLING GETSPACE. SUBSEQUENTLY, THE I/O DESCRIPTORS IN THE FILE-S BUFFER DESCRIPTOR AREA ARE PROVIDED WITH THE ADDRESSES OF THE BUFFER AREAS, AND THE BUFFER AREAS ARE FILLED WITH RECORDS FROM THE INPUT FILE.

IF A PROGRAM REQUIRED AN OUTPUT FILE, THE TYPE OF FILE MUST BE SPECIFIED (I.E., MAGNETIC TAPE, DISK, ETC.). BUFFER AREAS AND DESCRIPTORS FOR OUTPUT FILES ARE OBTAINED AS NOTED ABOVE; THEN, IF THE FILE IS FOR DISK, A DISK AREA OF THE SIZE SPECIFIED WILL BE OBTAINED FROM THE AREAS NOTED IN THE AVAILABLE-DISK TABLE. IF ANOTHER TYPE FILE IS DESIRED, THE LABELTABLE IS SEARCHED AND, IF A FILE IS AVAILABLE, IT IS ASSIGNED; OTHERWISE, A MESSAGE IS TYPED TO NOTIFY THE OPERATOR THAT THE FILE IS NEEDED. IF A LINE PRINTER OR CARD PUNCH IS REQUESTED AND NOT AVAILABLE, A CHECK IS MADE TO SEE IF THE PROGRAM SPECIFIED A BACK-UP OPTION. IF NOT, A MESSAGE IS TYPED TO NOTIFY THE OPERATOR THAT A PRINTER IS REQUIRED, AND SLEEP IS CALLED TO AWAIT A PRINTER. IF THE BACK-UP OPTION IS SPECIFIED, THE FILEOPEN PROCEDURE SETS THE CONTINUITY BIT IN THE FILE-S I/O

DESCRIPTORS. THE SIGNIFICANCE OF SETTING THIS BIT WILL BE NOTED LATER.

BUFFER AREA ACCESSED BY OBJECT PROGRAMS.

AS WAS NOTED, THERE IS A BUFFER DESCRIPTOR AREA FOR EACH FILE TO BE USED BY A PROGRAM. THE BUFFER DESCRIPTOR AREA MAY BE SET UP TO CONTAIN ONE OR MORE I/O DESCRIPTORS, AS SPECIFIED BY THE PROGRAM. AND, BECAUSE OF THE TECHNIQUE USED TO HANDLE THESE DESCRIPTORS, A PROGRAM IS NOT PROGRAMMATICALLY DEPENDENT UPON THE NUMBER OF BUFFER AREAS ASSIGNED. IN GENERAL, I/O DESCRIPTORS IN BUFFER DESCRIPTOR AREAS ARE HANDLED IN THE FOLLOWING MANNER. THE PROGRAM ALWAYS ACCESSES THE TOP I/O DESCRIPTOR IN THE AREA. WHEN THE PROGRAM HAS COMPLETED ITS USE WITH THE AREA ADDRESSED BY THE TOP DESCRIPTOR, THE AREA IS RELEASED. THEN THE MCP PERFORMS AN I/O TO REFILL OR WRITE-OUT THE BUFFER, WHICHEVER IS THE CASE. ALSO, THE MCP ROTATES THE I/O DESCRIPTOR; THAT IS, IF THE BUFFER AREA IS SET UP FOR MORE THAN ONE I/O DESCRIPTOR, THE DESCRIPTORS BELOW THE TOP DESCRIPTOR ARE MOVED UP AND THE PREVIOUS TOP DESCRIPTOR IS MOVED TO THE BOTTOM OF THE AREA. THEN CONTROL IS RETURNED TO THE OBJECT PROGRAM WHICH CAN CONTINUE ACCESSING THE AREA ADDRESSED BY THE TOP DESCRIPTOR.

IT SHOULD BE NOTED THAT EACH TIME BEFORE ACCESSING THE TOP I/O DESCRIPTOR, THE INTRINSICS THAT HANDLE I/O WILL CHECK THE I/O FINISH BIT IN THE DESCRIPTOR. THIS IS DONE SINCE IT WOULD BE POSSIBLE FOR AN I/O DESCRIPTOR TO COME TO THE TOP BEFORE THE I/O ON ITS BUFFER AREA WAS COMPLETE. IF THE PROGRAM FINDS THE I/O BUFFER HAS NOT BEEN COMPLETED, A SLEEP COMMUNICATE IS PERFORMED, PASSING THE ADDRESS OF THE I/O DESCRIPTOR AND A MASK FOR THE I/O FINISH BIT.

THE SLEEP COMMUNICATE IS PERFORMED BY PLACING SLEEP PROCEDURE PARAMETERS IN THE PROGRAM STACK TOGETHER WITH A CODE SPECIFYING THE TYPE OF COMMUNICATE, AND THEN EXECUTING A COMMUNICATE OPERATOR. THE MCP SUBSEQUENTLY MAKES A CALL ON THE SLEEP PROCEDURE. THEN, WHEN THE NECESSARY CONDITION OCCURS, CONTROL IS TRANSFERRED BACK TO THE PROGRAM AND IT CAN CONTINUE PROCESSING.

COMMUNICATE OPERATIONS.

THE GENERAL WAY IN WHICH OBJECT PROGRAMS INTERACT WITH THE MCP IS IMPLICITLY THROUGH INTERRUPT ACTION SUCH AS WITH PRESENCE BIT INTERRUPTS. ALSO, AS NOTED PREVIOUSLY, THE INTRINSIC TECHNIQUE IS USED SO THAT AN OBJECT PROGRAM AND MCP CAN INTERACT WITHOUT A NEED FOR SPECIAL ENTRIES TO CONTROL STATE. IN OTHER CASES, HOWEVER, A SPECIAL OPERATOR (THE COMMUNICATE OPERATOR) IS USED SO THAT CONTROL CAN BE EXPLICITLY TRANSFERRED FROM NORMAL STATE OPERATION TO CONTROL STATE OPERATION.

TO USE THE COMMUNICATE OPERATOR, A NORMAL STATE PROGRAM FIRST PLACES PARAMETERS IN ITS STACK. THEN THE COMMUNICATE OPERATOR IS EXECUTED. THE COMMUNICATE OPERATOR CAUSES A COMMUNICATE INTERRUPT. THE MCP

ROUTINE THAT HANDLES THIS INTERRUPT FIRST LOCATES THE STACK OF THE PROGRAM THAT CAUSED THE INTERRUPT. THEN ACCORDING TO A CODE VALUE IN THE PARAMETERS IN THE PROGRAM-S STACK, THE MCP TRANSFERS CONTROL TO THE SECTION OF THE MCP DESIGNED TO HANDLE A COMMUNICATE INTERRUPT WITH THAT CODE.

PERFORMANCE OF I/O BY MCP.

TO RELEASE A BUFFER AREA THAT IS TO BE REFILLED OR WRITTEN-OUT, A NORMAL STATE PROGRAM PERFORMS A PROGRAM RELEASE OPERATION. THIS OPERATION IS PERFORMED BY FIRST PLACING, IN THE A REGISTER, THE ADDRESS OF THE I/O DESCRIPTOR THAT ADDRESSES THE AREA TO BE RELEASED; THEN A PROGRAM RELEASE SYLLABLE CAUSES THE ADDRESS OF THE I/O DESCRIPTOR TO BE PLACED IN THE PROGRAM-S PRT AT (R+9) AND CHECKS THE CONTINUITY BIT IN THE I/O DESCRIPTOR. IF THE CONTINUITY BIT IS SET, THE CONTINUITY BIT INTERRUPT IS SET; IF NOT, THE PROGRAM RELEASE INTERRUPT IS SET. THE SETTING OF THE INTERRUPT CAUSES THE PROGRAM TO BE INTERRUPTED AND, SUBSEQUENTLY, CONTROL IS TRANSFERRED TO THE INTERRUPT LOCATION FOR THE INTERRUPT. FROM THIS POINT, THE PROGRAM RELEASE PROCEDURE OR CONTINUITY BIT ROUTINE IS CALLED.

PROGRAM RELEASE PROCEDURE.

THE PROGRAM RELEASE PROCEDURE OBTAINS THE ADDRESS OF THE I/O DESCRIPTOR FROM THE PROGRAM-S PRT AND SETS THE PRESENCE BIT OF THE DESCRIPTOR TO ZERO. IT ALSO CHECKS TO ENSURE THAT THE I/O DESCRIPTOR IS NOT REFERENCING A DISK UNIT. IF SO, AND THE RELTOG OPTION IS SET, THE PROGRAM IS DS-ED WITH THE MESSAGE "INVALID PRL...". (THE PRESENCE BIT OF AN "IN-PROCESS" I/O DESCRIPTOR IS SET TO ZERO SO THAT A PRESENCE BIT INTERRUPT WILL OCCUR IF THE DESCRIPTOR IS ACCESSED BEFORE THE I/O IS COMPLETE. SINCE I/O INTRINSICS CHECK AN I/O DESCRIPTOR BEFORE ACCESSING IT, THIS IS NOT LIKELY TO HAPPEN. HOWEVER, WHEN USING STREAM PROCEDURES AND RELEASE STATEMENTS IN EXTENDED ALGOL, IT IS POSSIBLE FOR A PROGRAM TO MAKE SUCH AN ACCESS. THE PRESENCE BIT ROUTINE REMEDIES THE SITUATION, SHOULD IT OCCUR.) THEN THE IOREQUEST PROCEDURE IS CALLED. WHEN IOREQUEST RETURNS CONTROL, THE I/O DESCRIPTORS IN THE BUFFER DESCRIPTOR AREA ARE ROTATED, AS DISCUSSED ABOVE, AND FINALLY, CONTROL IS TRANSFERRED TO THE INITIATE ROUTINE.

CONTINUITY BIT ROUTINE AND PRINTER BACKUP PROCEDURE.

THE CONTINUITY BIT ROUTINE CHECKS TO SEE IF THE I/O DESCRIPTOR THAT CAUSED THE INTERRUPT WAS A LINE PRINTER DESCRIPTOR, AS WOULD BE THE CASE FOR PRINTER BACKUP FILES. IF THIS IS THE CASE, THE PROCEDURE THAT HANDLES PRINTER BACKUP (AND PUNCH) IS CALLED. THE PRINTER BACKUP PROCEDURE THEN WRITES THE PRINT OR PUNCH LINE, TOGETHER WITH A COPY OF THE I/O DESCRIPTOR WHICH WOULD HAVE BEEN USED TO WRITE ON A BACKUP FILE. THE REMAINDER OF THE HANDLING OF THE I/O IS AS DESCRIBED FOR THE PROGRAM RELEASE PROCEDURE. AT A LATER POINT IN TIME, WHEN A LINE PRINTER IS AVAILABLE, THE PRINTER BACKUP FILE CAN

BE PRINTED. THE I/O DESCRIPTOR THAT ACCOMPANIES A LINE OF PRINT ON THE BACKUP FILE IS USED WHEN THE LINE IS PRINTED. CONSEQUENTLY, THE FORMAT OF THE PRINTOUT WILL BE DESIGNATED BY THE PROGRAM THAT CREATED IT.

IOREQUEST PROCEDURE.

IT IS THE FUNCTION OF THE IOREQUEST PROCEDURE TO EITHER INITIATE AN I/O, OR QUEUE UP AN I/O DESCRIPTOR IN THE I/O QUEUE. THE I/O QUEUE IS MADE UP OF FOUR TABLES; THE IOQUE, THE FINALQUE, THE LOCATQUE, AND THE UNIT TABLE. WHEN A REQUEST IS MADE TO PERFORM AN I/O, THE REQUEST IS QUEUED WITH RESPECT TO LOGICAL UNIT NUMBER. EACH REQUEST FOR AN I/O REQUIRES AN ENTRY IN IOQUE, FINALQUE, AND LOCATQUE. IOQUE CONTAINS A COPY OF THE I/O DESCRIPTOR FOR THE REQUEST. FINALQUE CONTAINS INFORMATION ABOUT A DESCRIPTOR TO BE RETURNED TO THE NORMAL STATE PROGRAM AFTER THE I/O IS COMPLETED. LOCATQUE CONTAINS THE ADDRESS OF THE TOP I/O DESCRIPTOR IN THE PERTINENT BUFFER DESCRIPTOR AREA, THE MIX INDEX OF THE PROGRAM THAT MADE THE REQUEST, THE LOGICAL UNIT NUMBER OF THE I/O UNIT, AND AN INDEX TO THE NEXT REQUEST QUEUED FOR THE UNIT. THE UNIT TABLE CONTAINS INFORMATION ABOUT EACH I/O UNIT. ENTRIES IN THE UNIT TABLE ARE IN LOGICAL UNIT NUMBER ORDER. AN ENTRY FOR A PARTICULAR UNIT INCLUDES INFORMATION SPECIFYING THE TYPE OF UNIT (E.G., MAGNETIC TAPE, CARD READER, ETC.), IF THE UNIT IS READY, IF THE UNIT IS CURRENTLY PROCESSING AN I/O, AND IF THE UNIT IS AWAITING ERROR RECOVERY.

WHEN IOREQUEST IS CALLED, A CHECK IS MADE TO SEE IF THE I/O QUEUE IS FULL. IF IT IS, THE SLEEP PROCEDURE IS CALLED AND IOREQUEST WILL NOT CONTINUE UNTIL SPACE IN THE QUEUE IS AVAILABLE. WHEN QUEUE SPACE IS AVAILABLE, A CHECK IS MADE TO SEE IF ANY OTHER I/O-S FOR THE UNIT ARE QUEUED OR IN PROCESS. IF SO, THE CURRENT REQUEST IS LINKED INTO THE LIST OF REQUESTS FOR THE UNIT. IF NOT, A TEST IS MADE TO SEE IF AN I/O CHANNEL IS AVAILABLE. IF A CHANNEL IS AVAILABLE THE INITIATEIO PROCEDURE IS CALLED TO INITIATE THE I/O. IF A CHANNEL IS NOT AVAILABLE, THE QUEUEUP PROCEDURE IS CALLED TO ENTER THE LOGICAL UNIT NUMBER OF THE UNIT IN WAITQUE, A TABLE USED BY THE IOFINISH PROCEDURE.

INITIATEIO PROCEDURE.

THE INITIATEIO PROCEDURE IS CALLED WHEN A UNIT IS READY FOR AN I/O AND AN I/O CHANNEL IS AVAILABLE. INITIATEIO MAKES USE OF A TABLE CALLED "CHANNEL" WHICH HAS AN ENTRY FOR EACH I/O CHANNEL. IN THE ENTRY FOR THE CHANNEL TO BE USED FOR THE I/O, INITIATE STORES THHE LOGICAL UNIT NUMBER OF THE UNIT WHICH WILL PERFORM THE I/O. THEN THE I/O IS INITIATED AND NOTE IS MADE OF I/O TIMING INFORMATION FOR LOGGING AND STATISTICS.

IOFINISH PROCEDURE.

WHEN AN IOFINISH INTERRUPT OCCURS, THE IOFINISH PROCEDURE IS CALLED.

AT THIS TIME, AT LEAST ONE I/O CHANNEL IS AVAILABLE. IOFINISH FIRST CHECKS TO SEE IF THE RESULT DESCRIPTOR, FOR THE COMPLETED I/O, NOTED ANY ERRORS. IF SO, ACTION IS TAKEN TO PROHIBIT FURTHER I/O ON THE UNIT UNTIL THE ERROR CONDITION IS RECTIFIED; HOWEVER, ERROR CONDITIONS, IF ANY, ARE NOT HANDLED AT THIS TIME. INSTEAD, A CHECK IS MADE TO SEE IF ANY UNITS IN WAITQUE ARE WAITING FOR AN I/O CHANNEL. IF SO, AN I/O IS INITIATED ON THE FIRST UNIT NOTED IN WAITQUE. IF THERE ARE NO ENTRIES IN WAITQUE, A CHECK IS MADE TO SEE IF ANOTHER I/O CAN BE MADE ON THE UNIT THAT JUST COMPLETED AN I/O. IF SO, AN I/O IS INITIATED FOR THAT UNIT. IF AN I/O WAS INITIATED ON A UNIT FROM WAITQUE, AND A SECOND UNIT WHICH JUST COMPLETED AN I/O WAS READY FOR ANOTHER I/O, THE SECOND UNIT WOULD BE QUEUED IN WAITQUE.

AFTER ALL ACTIONS TO INITIATE I/O-S FOR QUEUED REQUESTS HAVE BEEN COMPLETED, IOFINISH PROCEEDS TO HAVE ERROR CONDITIONS RECTIFIED, IF ANY, OR PERFORMS FINAL HANDLING OF THE I/O REQUEST. IF ERRORS WERE NOTED, IOFINISH WOULD CALL INDEPENDENTRUNNER TO REQUEST THAT THE IOERR PROCEDURE BE CALLED. IOERR WOULD THEN TAKE ACTION TO RECTIFY THE ERROR. IF NO ERRORS OCCURRED, IOFINISH WOULD PERFORM FINAL HANDLING OF THE REQUEST, INCLUDING SETTING THE I/O FINISH AND PRESENCE BITS IN THE I/O DESCRIPTOR TO 1. AFTER COMPLETING ALL ADJUSTMENTS, THE BUFFER AREA CONCERNED IS READY FOR FURTHER PROGRAM USE.

INTERROGATING PERIPHERAL UNITS.

AFTER OTHER ACTIVITIES HAVE BEEN COMPLETED, CONTROL IS TRANSFERRED TO THE CONTROL SECTION OF THE MCP. ONE OF THE ACTIVITIES OF THE CONTROL SECTION IS TO CHECK FOR CHANGES IN STATUS OF I/O UNITS (CHECK TO SEE IF ANY I/O UNITS CHANGED FROM READY TO NOT READY OR NOT READY TO READY). THIS CHECK IS MADE THROUGH USE OF THE INTERROGATE PERIPHERAL STATUS OR READ READY REGISTER OPERATOR WHICH PLACES IN THE TOP OF THE STACK A WORD REPRESENTING THE CURRENT STATUS OF PERIPHERAL UNITS. ONE BIT IN THE WORD IS ASSOCIATED WITH EACH I/O UNIT; A UNIT-S BIT IS 0 IF THE UNIT IS NOT READY AND 1 IF IT IS READY. THE WORD PROVIDED BY THE OPERATOR IS COMPARED WITH A WORD (RRRMECH) THAT REFLECTS THE PREVIOUSLY NOTED STATUS OF THE UNITS DURING INITIALIZATION. THE WORD USED FOR COMPARISON IS SET TO INDICATE ALL UNITS ARE NOT READY. WHEN A CHANGE IN STATUS OCCURS, A REQUEST IS MADE TO HAVE THE STATUS PROCEDURE CALLED.

STATUS PROCEDURE.

THE STATUS PROCEDURE IS RESPONSIBLE, IN PART, FOR MAINTAINING THE MCP TABLES WHICH CONTAIN INFORMATION, FOR SPECIFYING WHAT UNITS ARE READY OR NOT READY. WHEN AN INPUT OR INPUT-OUTPUT UNIT BECOMES READY FOR THE FIRST TIME, STATUS CAUSES THE FIRST RECORD ON THE UNIT TO BE READ. STATUS THEN EXAMINES PERTINENT INFORMATION AND ENTERS LABEL INFORMATION IN APPROPRIATE TABLES OR SPECIFIES THAT A FILE IS A SCRATCH FILE, ETC. IF, ON AN APPROPRIATE UNIT, THE FIRST RECORD

ON A FILE IS "CONTROLCARD" INFORMATION, STATUS REQUESTS THAT THE CONTROLCARD PROCEDURE BE CALLED.

OLAY

OLAY(LOC) IS A PROCEDURE WHICH OVERLAYS THE INFORMATION AT LOCATION "LOC", IF POSSIBLE.

OLAYMASK

CONTAINS INFORMATION USED TO LOCK OUT GETMOREOLAYDISK ROUTINE BY MIX INDEX.

OLDIDLETIME

CONTAINS PREVIOUS VALUE OF IDLETIME.

ONEOHONE

CONTENTS OF CELL @101.

ONEOHTWO

CONTENTS OF CELL @102.

OPTION WORD

A WORD STORED IN THE "MCP PRT" AND IN "DIRECTORYTOP" TO SET AND RESET OPTIONS. THE OPTION WORD CAN BE BE SET VIA THE "COLD START" ROUTINE OR THROUGH KEYBOARD INPUT.

FIELD	OPTION
[47:1]	DRA
[46:1]	DRB
[45:1]	BOJ
[44:1]	EOJ
[43:1]	OPEN
[42:1]	TERMNATE
[41:1]	DATE
[40:1]	TIME
[39:1]	ONEBREAK (BATCH ONLY)
[38:1]	AUTOPRNT
[37:1]	CLEARWRS (BATCH ONLY)
[36:1]	DISCONDC (BATCH ONLY)
[35:1]	CMPLFILE
[34:1]	CLOSE
[33:1]	ERRORMSG
[32:1]	RET
[31:1]	LIBMSG
[30:1]	SCHEDMSG
[29:1]	SECMSG
[28:1]	DSKTOG
[27:1]	RELTOG
[26:1]	PBDREL
[25:1]	CHECK
[24:1]	DISKMSG
[23:1]	DISKLOG (TSS ONLY)
[22:1]	LIBERR (TSS ONLY)
[21:1]	PBDONLY
[20:1]	SAVEPBT
[19:1]	RSTOG
[18:1]	AUTOUNLD
[17:1]	RNALL (SHAREDISK ONLY)
[16:1]	CODEOLAY (BATCH ONLY)
[15:1]	COREST
[14:1]	DATAOLAY (BATCH ONLY)
[13:1]	HALT (TSS ONLY)
[12:1]	REMOTE (TSS ONLY)
[11:1]	CEMESS (TSS ONLY)
[10:1]	BATCHZIP (TSS ONLY)

- OPTION WORD -

[9:1]	NOBATCH (TSS ONLY)
[8:1]	UNUSED
[7:1]	UNUSED
[6:1]	UNUSED
[5:1]	UNUSED
[4:1]	UNUSED
[3:1]	UNUSED
[2:1]	MOD III I-O (NOT ACCESSIBLE THROUGH THE KEYBOARD)

ORR WORD

THE "ORR" WORD IS USED IN CONJUNCTION WITH "INQUIRY". PLEASE REFER TO THE DOCUMENTATION FOR "INQUIRY" FOR ADDITIONAL INFORMATION.

PBCOUNT

COUNTER FOR NUMBER OF COMPLETED BUT AS YET UNPRINTED PBD-S AND PUD"S.

PBIO

PBIO(ALPHA,POINTER) IS A PROCEDURE WHICH BLOCKS OR DE-BLOCKS BACKUP FILES IN MAIN MEMORY. "ALPHA" IS THE ADDRESS OF THE TOP I/O DESCRIPTOR. IF "ALPHA" IS LESS THAN ZERO, THE FILE IS AN INPUT FILE. "POINTER" IS FIB[14].

PBLOG

PBLOG(ST) IS A PROCEDURE WHICH MAKES THE "DUMMY" LOG ENTRIES FOR JOBS THAT CREATE PRINTER BACK UP FILES WHEN THE FILE IS PRINTED (USING PRNPBT/DISK). THE INFORMATION IN THE LOG ENTRY IS CONSTRUCTED FROM THE LABEL INFORMATION IN THE PRINT FILE.

PEUIO

CONTAINS THE SAME TYPE OF INFORMATION AS EUIO.

PINGO

USED TO LINK TANKED MCP INPUT MESSAGES TOGETHER.

PICKTHELOCK

PICKTHELOCK IS A PROCEDURE WHICH HANDLES THE LOCKING AND UNLOCKING OF LOCK-ITEMS FOR TASKING. IT ALSO HANDLES THE MAINTENANCE OF A WAIT QUEUE AND PASSES CONTROL OF THE LOCK TO THE FIRST PROCESS IN THE WAIT QUEUE, AFTER IT HAS BEEN RELEASED BY ANOTHER PROCESS. THE HEAD OF THE QUEUE IS THE LOCK-ITEM OF THE PROCESS CURRENTLY IN CONTROL AND ENTRIES ARE MADE AT THE END OF THE QUEUE. LOCK-ITEMS ARE IN THE PRT. "LOCKPTR" IS THE PARAMETER PASSED AND HAS THE FOLLOWING FORMAT:

[1:1] 0 = LOCK, 1 =UNLOCK

[2:1] 1 = TEST LOCK BIT, LOCK IF UNLOCKED AND RETURN A 0 ELSE RETURN A 1.

[33:15] RELATIVE PRT ADDRESS OF LOCK-ITEM.

POINTERS

THE FORMAT OF AN INDVIDUAL POINTER IS AS FOLLOWS:

FIELD	DESCRIPTION
.[18:10]	= WORD
.[28:3]	= CHARACTER
.[31:2]	= STRING TYPE
	0 = 6 BIT
	1 = 8 BIT
	2 = 3 BIT
.[33:15]	= ADDRESS OF ARRAY

PRNTABLE

FOR UNITS 0 THROUGH 15 PRNTABLE[I] CONTAINS THE PHYSICAL REEL NUMBER IN [30:18], 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.

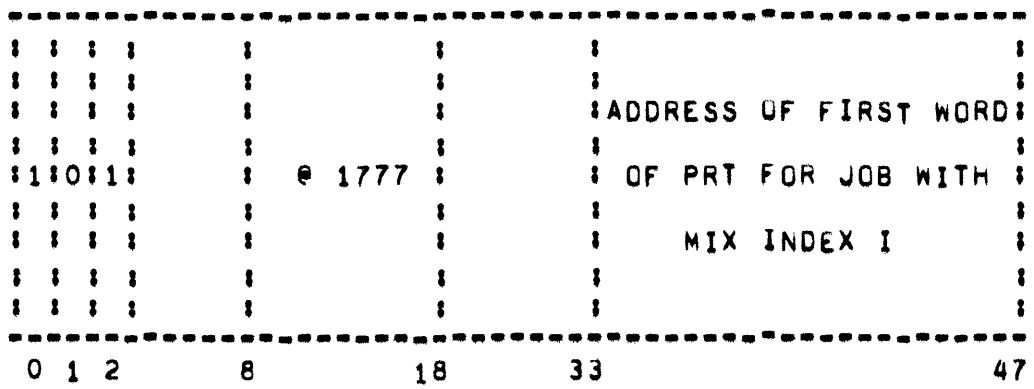
PROCTIME

DESCRIPTOR POINTING TO THE PROCTIME ARRAY.

PRT[*,*]

THE PROGRAM REFERENCE TABLE (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. THAT IS, ACCESS CAN BE MADE TO THE "PRT" OF A GIVEN PROGRAM BY ACCESSING THE "PRT" ARRAY USING 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.



FIELD	CONTENTS	DESCRIPTION
[0:3]	5	IDENTIFICATION
[3:4]	PSF ON TSS	= 0 IN SWAPPER IF PUT AT FORCEND = 1 AFTER CALL TERMINATE = 2 AFTER "ST",RESET @ BY STOPM = 2 ALSO IN PAUSE (SEE C33) = 3 IN SWAPPER IF FORCESWAP
[7:1]	BOOLEAN	USED TO SIGINAL "FILECLOSE" ON A PBD FILE BEING DS-ED WHILE NO-USER-DISK AND CURRENT RECORD REQUIRES MORE DISK (SEE MCP 411034400) OR THAT A JOB WHICH WAS BEING RESTARTED IS NOW BEING DS-ED.
[8:10]	@1777	
[18:15]	NA	
[33:15]	ADDRESS	ADDRESS OF THE FIRST WORD IN "PRT" OF OBJECT PROGRAM. (R+0)

FORMAT OF "OBJECT PROGRAMS PRT" (EXCEPT FOR FORTRAN)

R+0	"EEEEEEEE"	USED BY "MCP" TO DENOTE BEGINNING OF "PRT". ALSO USED TO CHECK FOR STACK OVERFLOW
1		USED BY "ANALYSIS" FOR BRANCH TO NON-PRESENT LABEL.
2	5 000.....0	"MEMORY" FOR NORMAL STATE. (BUILT BY SELECTION)
3	FPB	DESCRIPTOR POINTING TO "FILE PARAMETER BLOCK" (FPB) (BUILT BY SELECTION)
4	SD	DESCRIPTOR POINTING TO "SEGMENT DECTIONARY" (SD). (BUILT BY SELECTON).
5	BC	DESCRIPTOR POINTING TO "BLOCK CONTROL" INTRINSICS.
6	AIT	DESCRIPTOR POINTING TO "ARRAY INFORMATION TABLE" (AIT). (BUILT BY MCP)
7	MSCW	MARK STACK CONTROL WORD.
10	INCW	INITIATE CONTROL WORD.
11	COM,PRL	LOCATION TO STORE THE LAST PARAMETER FOR THE "COMMUNICATE" AND "PROGRAM RELEASE" OPERATORS.
12	BASE OF STACK & PRT	DATA DESCRIPTOR POINTING TO R+0. F FIELD POINTS TO LOCATION OF STACK BOTTOM. (BUILT BY SELECTION).
13	SIZEERROR OWN ARRAY TABLE	DESCRIPTOR POINTING TO "OAT" IN "ALGOL" AND ON SIZE ERROR PRT IN [FF] FOR COBOL
14	ALGOL WRITE COBOL FCR	PROGRAM DESCRIPTOR POINTING TO WRITE INTRNSICS FOR "ALGOL", "FCR" FOR "COBOL".
15	ALGOL READ	PROGRAM DESCRIPTOR POINTING TO READ INTRINSICS FOR "ALGOL".
16	ALGOL SELECT COBOL I-O	READ WRITE DESCRIPTOR POINTING TO SELECT ROUTINE FOR "ALGOL" AND COBOLREAD ROUTINE FOR "COBOL".
17	0	ZERO.
20	BLOCKCTR	BLOCK LEVEL COUNTER (STARTS AT 1 WITH OUTER MOST BLOCK AT SYMBOLIC PROGRAMS).
21	JUNK	TEMPORARY STORAGE LOCATION FOR USE BY SOFTWARE PRIMARILY FOR INTEGERIZATION
22	EXITR	CHARACTER MODE DESCRIPTOR -- REFERENCES THE FIRST SYLLABLE OF THE PROGRAM, I.E., THE OUTERMOST BLOCK WHICH IS GENERATED BY THE

		COMPILER. THE FIRST SYLLABLE IS @00 WHICH IS SIMULTANIOUSLY A "LITC 0" AND A "CMX" WHICH HAS THE EFFECT OF CUTTING BACK THE STACK.
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 ERROR COUNT. FIRST "PRT LOCATION ASSIGNED BY COMPILER". ALSO THE LOCATION OF THE "SAME" STORAGE FOR COBOL POGRAMS.
26	SAVE	TEMPORARILY USED FOR SAVE FACTOR FOR CODE FILE. GIVEN BY MCP. = 0 IF COMPILE TO GO = (-1) IF SYNTAX.

PRT (FOR COBOL)

WORD	CONTENTS
1-13	SAME AS WORDS 1-13 DISCUSSED UNDER OBJECT PROGRAM-S PRT
14	FCR
16	READ-WRITE
24	10*8
25	COMMON
26	INDEX LOCATION FOR OBJECT TIME CONTROL
27	OBJECT TIME CONTROL PROCEDURE
30	PROGRAM FIB
31	SUBSCRIPT WORKING STORAGE
32	SUBSCRIPT COMMUNICATION WORKING STORAGE
33	
34	
35	GET DATANAME SIGN(O,MKS,WORD,CHAR)
36	CLEARWORDS TO BLANK
37	
40	WORD OCCURS INDEX
41	CHARACTER OCCURS INDEX
42	TEMPORARY OBJECT TIME ARRAY
43	POWERS OF 10
44	TRASH LOCATION
45	STORE DATANAME SIGN(SIGN,MKS,WORDLOC,CHAR)
46	WORD TRANSFER
47	CLEAR CHARACTER TO ZERO
50	CHARACTER TRANSFER
51	CLEAR CHARACTER TO BLANKS
52	CLEAR WORDS TO BLANKS
53	NORMAL JUSTIFIED SOURCE TO RIGHT JUSTIFIED DESTINATION
54	RIGHT JUSTIFIED SOURCE TO NORMAL JUSTIFIED DESTINATION
55	RIGHT JUSTIFIED SOURCE TO RIGHT JUSTIFIED DESTINATION
56	ALPHA COMPARE
57	NUMERIC COMPARE
60	INPUT CONVERT PROCEDURE
61	CHARACTER MODE INPUT CONVERT PROCEDURE 0 LSS X LSS 9
62	CHARACTER MODE INPUT CONVERT PROCEDURE 8 LSS X LSS 12
63	OUTPUT CONVERT (OCIL)
64	OUTPUT CONVERT (OC8L)
65	INPUT CONVERT DOUBLE LENGTH
66	OUTPUT CONVERT DOUBLE LENGHT
67	WORD MODE OUTPUT CONVERT LENGTH,SGN,CHAR,WRD,MKS,VA
70	10*11

71 OUTER BLOCK PRT DESCRIPTOR

PRT (FOR COBOL68)

WORDS	CONTENTS
1-12	SAME AS WORDS 1-12 OF OBJECT PROGRAM-S PRT
13	SIZE ERROR
14	FCR
15	PERFORMGEN ROUTINE WHICH HANDLES GENERATION OF FT CALLS
16	COBOLIO NEW DISK (SERIAL I-O INT.)
17	COBOLRANDOM
20	BLOCKCOUNTER
21	JUNK
22	INPUT CONVERT INTRINSIC
23	JUNK
24	ATTRIBUTE ROUTINE
25	RESERVED FOR BREAKOUT-RESTART
26	TASK ARRAY DESCRIPTOR
27	DURIO ARRAY
30	USE ROUTINE ARRAY
31	DMOD (DOUBLE PRECISION MATH INTRINSIC)
32	OUTPUT CONVERT INTRINSIC
33	INTERRUPTER INTRINSIC (IF ANY DECLARED)

PRT (FOR FORTRAN)

WORD	CONTENTS	DESCRIPTION
@1-@21		SAME AS WORDS 0-17 OF OBJECT PROGRAM-S PRT
@22	BASENSIZE	
@23	LISTRTN	
@24	CLASN	
@25	HOLTOG	COMMON VALUE
@26		POWERS OF TEN ARRAY
@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	
@43	DISN	
@44	ATAN2	
@45	CABS	
@46	DMOD	
@47	DEXP	
@50	DSQRT	
@51		FORTRAN RUN-TIME ERROR RECOVERY CONTROL WORD

PRT (FOR THE MCP)

THE MCP-S PRT CONTAINS VARIABLES, DATA DESCRIPTORS, AND PROGRAM DESCRIPTORS IN A FASHION SIMILAR TO THAT OF AN ALGOL PROGRAM-S PRT ABOVE R+@25.

THIS SECTION OF THE MANUAL CONTAINS DETAILED DESCRIPTIONS OF MOST OF THE ARRAYS AND VARIABLES. THE ENTRIES IN THIS SECTION ARE IN ALPHABETIC ORDER TO FACILITATE ITS USE AS A REFERENCE DOCUMENT. IN THE MCP, THE ACTUAL PRT LOCATIONS TAKEN BY EACH OF THE ENTRIES CHANGES WITH THE COMPILE-TIME MODULES INCLUDED OR EXCLUDED, AND MAY CHANGE WITH SUBSEQUENT MCP LEVELS.

PROCTIME

DESCRIPTOR POINTING TO THE PROCTIME ARRAY. PROCTIME[I] CONTAINS PROCESSOR TIME FOR JOB WITH MIX INDEX = I.

PRT

DESCRIPTOR POINTING TO THE PRT ARRAY. A DETAILED EXPLANATION OF THE PRT APPEARS IN THE "MCP TABLES" SECTION OF THIS DOCUMENT.

PRYOR

DESCRIPTOR POINTING TO PRIORITY ARRAY.

PSEUDOCOPY

COUNTER FOR THE NUMBER OF COPIES OF CONTROLCARD CURRENTLY RUNNING.

PUNT

PUNT(I) IS DEFINED AS MESS(I) AND PRINTS A SYSTEM HALT MESSAGE ON THE SPO, THEN GOES INTO A DYNAMIC HALT. THE MESSAGE COMES FROM ADDRESS "I" AND IS ONE OF THE FOLLOWING:

```
ARRAY PUNTER[38]

    [1] PARITY ON RESTART TAPE
    [4] READ ERROR RESTARTING
    [7] HOLDLIST OVERFLOW
    [8] HOLDLIST OVERFLOW
    [11] UNEXP I-O ERROR
    [13] INVALID ADDRESS
    [15] SLATE OVERFLOW
    [17] BED OVERFLOW
    [19] NO USER DISK
    [21] DIRECTORY FULL
    [23] NO MCP DISK
    [25] INVALID LINK
    [27] LOCK QUEUE FULL
    (SHAREDISK ONLY)
    [30] DISK ERROR
    [31] INV DISK ADDRESSES
```

PUNTER

DESCRIPTOR POINTING TO A TABLE CONTAINS VARIOUS MESSAGES THE MCP TYPES DURING PUNT OPERATIONS.

PURGEIT

PURGEIT(U) IS A PROCEDURE WHICH IS USED BY THE MCP TO PURGE A TAPE ON LOGICAL UNIT "U".

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. IF PROCESSOR 2 IS IDLE, P2MIX IS ZERO. IF PROCESSOR 2 IS ABSENT, P2MIX IS (-1).

QTIMES

DESCRIPTOR POINTING TO QTOME ARRAY, WHICH CONTAINS INFORMATION CONCERNING B487 TERMINAL TIME OUT BY MIX INDEX.

QVLEFT

CONTAINS TIME UNTIL NEXT TIME THAT SOME B487 TERMINAL WILL TIME OUT.

QV
--

CONTAINS CURRENT "Q" VALUE FOR REMOTE TERMINAL TIMEOUT BY STATION ON B487.

RDCTABLE

RDCTABLE[I] CONTAINS THE FOLLOWING FOR LOGICAL UNIT "I":

FIELD	CONTENTS
[1:2]	I-O CHANNEL NUMBER OF LAST I-O ON A PARTICULAR UNIT ON WHICH AN ERROR OCCURRED.
[3:5]	HARDWARE UNIT NUMBER OF UNIT FROM RESULT DESCRIPTOR ON WHICH ERROR OCCURRED.
[8:6]	MIX INDEX.
[14:10]	LOGICAL REEL NUMBER.
[24:17]	RETENTION DATE.
[41:7]	CYCLE.

- READERA, READERB -

READERA

CONTAINS A POINTER TO AN AREA OF CORE CONTAINING THE NEXT CONTROL CARD FOR CRA, ASSUMING IT HAS ALREADY BEEN READ, OTHERWISE IT IS ZERO.

READERB

SAME AS READERA EXCEPT FOR CRB.

READFROMDISK

THE ROUTINE READFROMDISK IS USED FOR ALL ACCESSES TO PSEUDOREADER ONCE THEY HAVE BEEN OPENED. ITS PROCEDURE DECLARATION IS AS FOLLOWS:

```
BOOLEAN PROCEDURE READFROMDISK(H,IB);

    VALUE H,IB;
    ARRAY H,IB[*];
```

WHERE "H" IS A POINTER TO CIDROW FOR THIS PSEUDOREADER-S LOGICAL UNIT NUMBER AND "IB" IS A POINTER TO THE AREA IN WHICH TO PLACE THE NEXT CARD IMAGE.

READFROMDISK RETURNS A VALUE OF TRUE IF "IB" CONTANS A CONTROL CARD IMAGE AND FALSE IF "IB" CONTAINS A NON-CONTROL CARD IMAGE.

(SEE ALSO CIDROW).

READQ

THE HEAD OF THE QUEUE OF ALL SOUGHT B487 TERMINAL/BUFFERS.

READY

CONTAINS THE CONTENTS OF THE READY REGISTER ON THE LAST READ READY REGISTER (RRR) INSTRUCTION.

REMOTEFILE

CONTAINS CURRENT RECORD NUMBER IN USE IN REMOTE/LOG.

REMOTELOGGER

THE REMOTELOGGER ROUTINE IS RESPONSBLE FOR MAKING ALL NON-ABORT ENTRIES INTO THE FILE "REMOTE/LOG". ITS PROCEDURE DECLARATION IS AS FOLLOWS:

```
PROCEDURE REMOTELOGGER(TYPE,RMCARD,STA,MIX);

    VALUE TYPE,RMCARD,STA,MIX;
    REAL TYPE,RMCARD,STA,MIX;
```

THE INTERPRETATION OF THE PARAMETERS IS A FOLLOWS:

PARAMETER	VALUE	MEANING
TYPE	= 1	LOG-OUT ENTRY
TYPE	= 2	LOG-IN ENTRY
TYPE	= 3	EXECUTE OR RUN ENTRY
TYPE	= 4	OBJECT PROGRAM DID A WRITE OR READ SEEK ON STATION
TYPE	= 5	DETACH STATION FROM MIX
TYPE	= 6	ALGOL OR COBOL INTERROGATE
TYPE	= 7	XEQ OR RUN CARD ENTERED AT REMOTE STATION CAUSED ATTACH
TYPE	= 8	STATION 0-0 DETACHING
RMCARD		ADDRESS OF CONTROL CARD OR OTHER INFORMATION PASSED TO ROUTINE
STA,[9:9]		REMOTE TU & BUFF
MIX		MIX INDEX OF JOB

REMOTELOGGER IS ALSO RESPONSIBLE FOR MUCH OF THE MAINTENANCE OF THE USERSTA ARRAY.

REMOVETHEDECK

REMOVETHEDECK(N,TUSTA) IS A PROCEDURE WHICH REMOVES PSEUDO-DECK "N" FROM DISK.

RE-ENTRANT CODE HANDLING

THE MCP DETERMINES AT THE TIME OF JOB INITIALIZATION (SELECTRUN PROCEDURE) WHETHER CONDITIONS EXIST WHEREBY A NEW JOB IS TO USE THE SAME CODE THEN BEING USED BY ONE OR MORE OTHER JOBS IN THE MIX. IF SUCH CONDITIONS DO EXIST, THE JOB IS ENTERED INTO A LINKED LIST OF JOBS USING THE CODE. AT JOB TERMINATION (COM5 PROCEDURE) THIS JOB WILL BE DE-LINKED FROM THIS RE-ENTRY LINKED LIST.

ONCE A JAR ENTRY FOR A NEW JOB HAS BEEN CREATED AND ITS PRT READ IN FROM DISK, THE SELECTRUN PROCEDURE DETERMINES IF RE-ENTRANCY EXISTS. THIS IS ACCOMPLISHED BY A CALL ON THE BOOLEAN SUBOUTINE "REENTRY" (THE CALL IS AT SEQUENCE 20108100). IF REENTRY RETURNS A VALUE OF TRUE, THE JOB WILL USE THE SEGMENT DICTIONARY ALREADY IN CORE FOR THE RE-ENTRANT CODE, OTHERWISE, THE PROGRAM-S SEGMENT DICTIONARY AS CONSTRUCTED DURING COMPILATION, WILL BE READ INTO CORE. IF CONDITIONS FOR RE-ENTRANCY DO EXIST, THE REENTRY SUBROUTINE WILL EITHER ENTER THE JOB INTO THE LINKED LIST OF JOBS USING THE CODE INVOLVED OR CREATE A NEW LIST IF THE JOB IS THE SECOND ONE TO MAKE USE OF THE CODE. THE CRITERIA BY WHICH THE MCP DETERMINES WHETHER A PARTICULAR JOB ENTERING THE SYSTEM IS TO BE TREATED AS RE-ENTRANT ARE:

A. THE JOB MUST NOT BE THE "GO" PORTION OF A COMPILE-AND-GO.

B. THERE MUST EXIST AT LEAST ONE OTHER JOB IN THE MIX WITH THE SAME <MFID> (WHICH MUST ALSO SATISFY THE FIRST CONDITION) AND

C. THE NEW JOB MUST EITHER BE A COMPILE OR HAVE THE SAME <FID> AS THAT JOB WITH WHICH IT SHARES THE SAME <MFID>.

THE COM5 PROCEDURE PERFORMS TWO MAJOR FUNCTIONS WITH RESPECT TO RE-ENTRANT CODE AT JOB TERMINATION:

A. IT DETERMINES IF IN-CORE CODE AREAS FOR A JOB ARE BEING USED BY OTHER JOBS AND CANNOT BE RELEASED TO THE SYSTEM AND

B. IT DE-LINKS THE JOB IN THE RE-ENTRANT LIST OR REMOVES THE LIST IF ONLY ONE OTHER JOB IS USING THE CODE.

THE POINTERS OF THE LINKED LIST STRUCTURE DISCUSSED ABOVE ARE AACTUALLY TWO (FORWARD AND BACKWARD POINTERS) SIX-BIT FIELDS IN THE SEGMENT DICTIONARY NAME DESCRIPTOR KEPT IN PRT[P1MIX,4] FOR A PARTICULAR JOB. THIS DESCRIPTOR IS ALSO KEPT IN THE NFO ARRAY AS NFO[(MIX-1)×NDX+1] WHERE NDX IS THE NUMBER OF ELEMENTS OF THE ARRAY ALOTTED TO EACH MIX NUMBER.

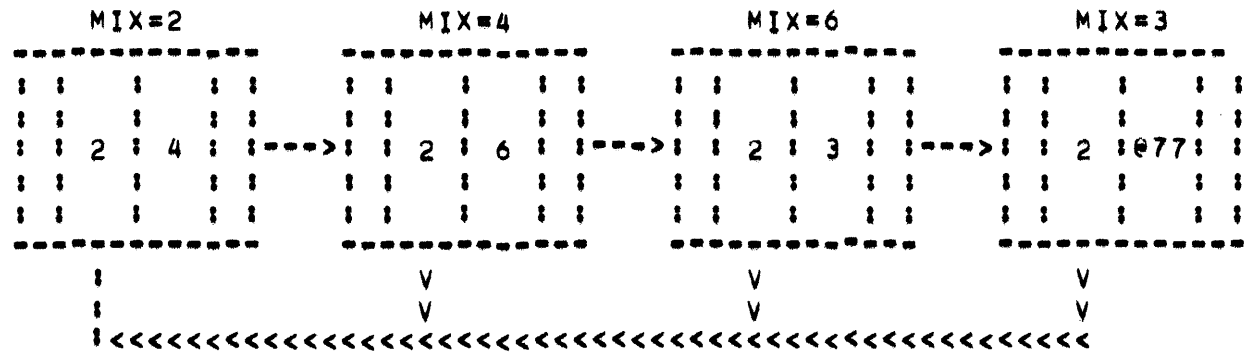
RE-ENTRANT CODE HANDLING

THE CONTENTS OF THE POINTER ARE AS FOLLOWS:

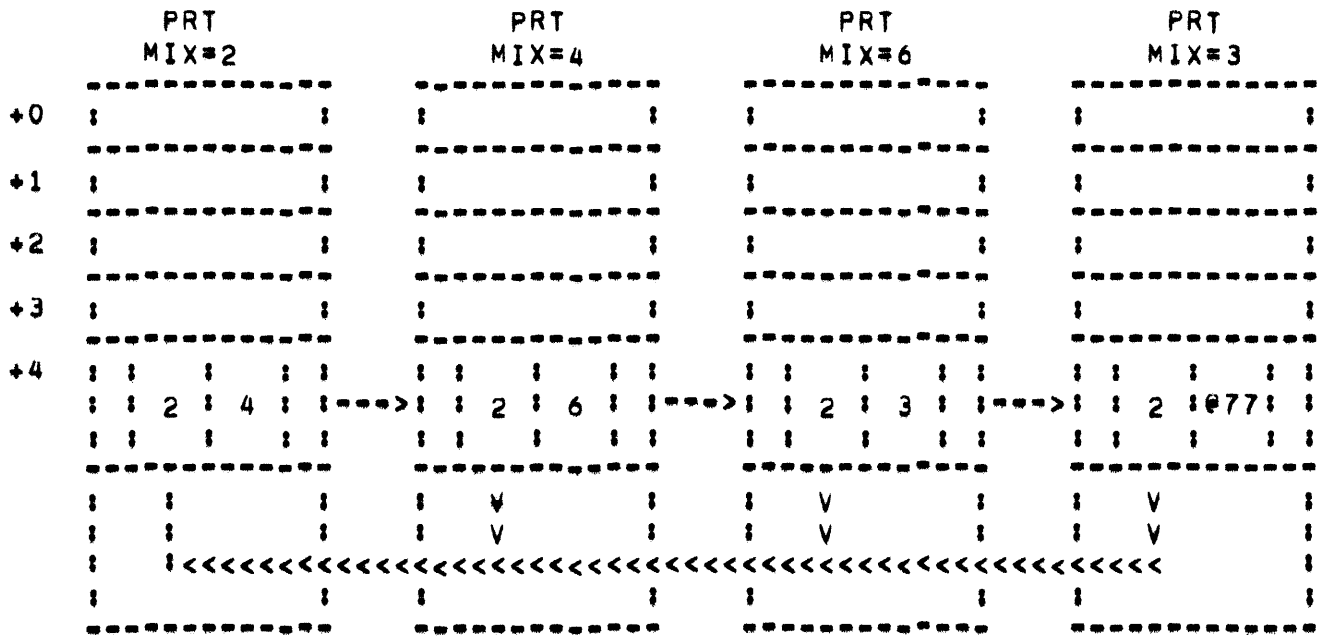
PRT[MIX,4].[18:12]	0 FOR NON-RE-ENTRANT CODE.
PRT[MIX,4].[18:6]	MIX INDEX FOR THE FIRST PROGRAM TO USE THIS RE-ENTRANT CODE (ALL JOBS USING THIS CODE WILL HAVE THE SAME MIX INDEX AS THE BACKWARD POINTER).
PRT[MIX,4].[24:6]	MIX INDEX OF THE NEXT JOB USING THIS CODE (=@77 IF THE LAST ENTRY IN THE LIST).

THUS, A TYPICAL LIST MIGHT BE AS FOLLOWS:

A). NFO[(MIX-1)×NDX+1]:

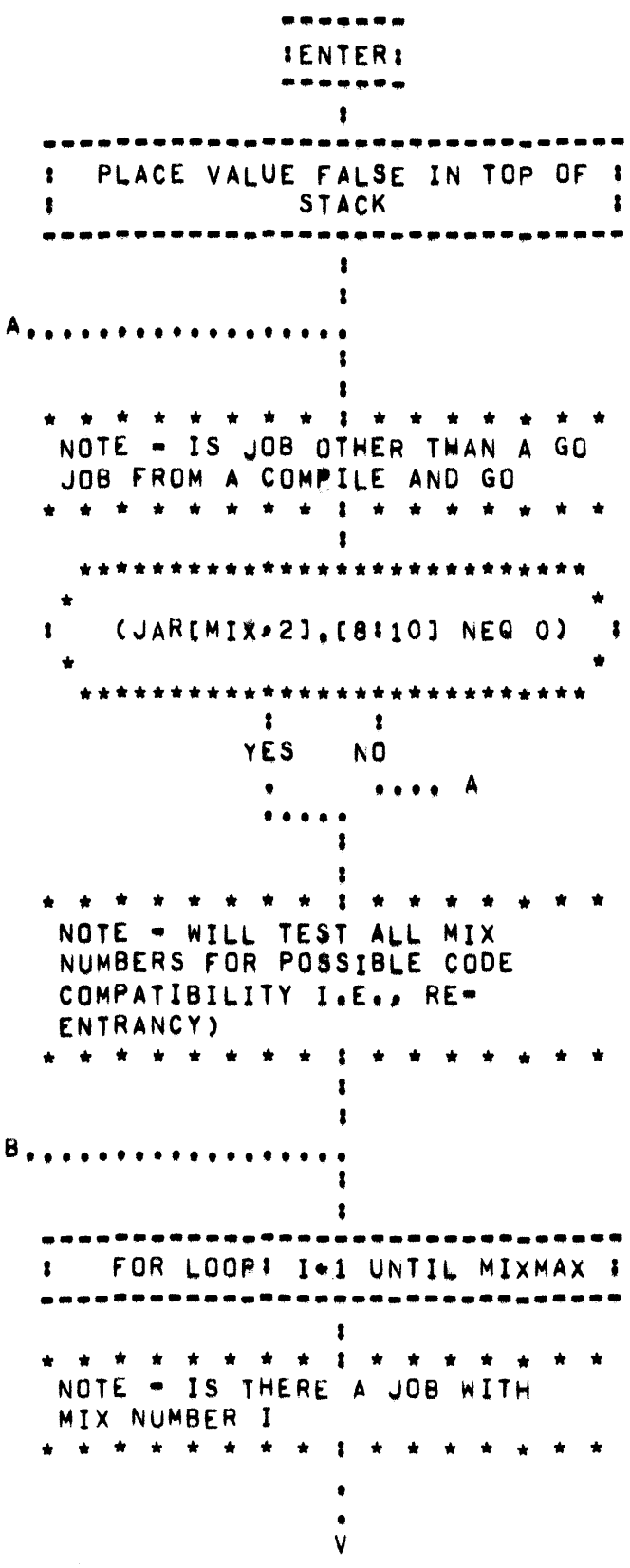


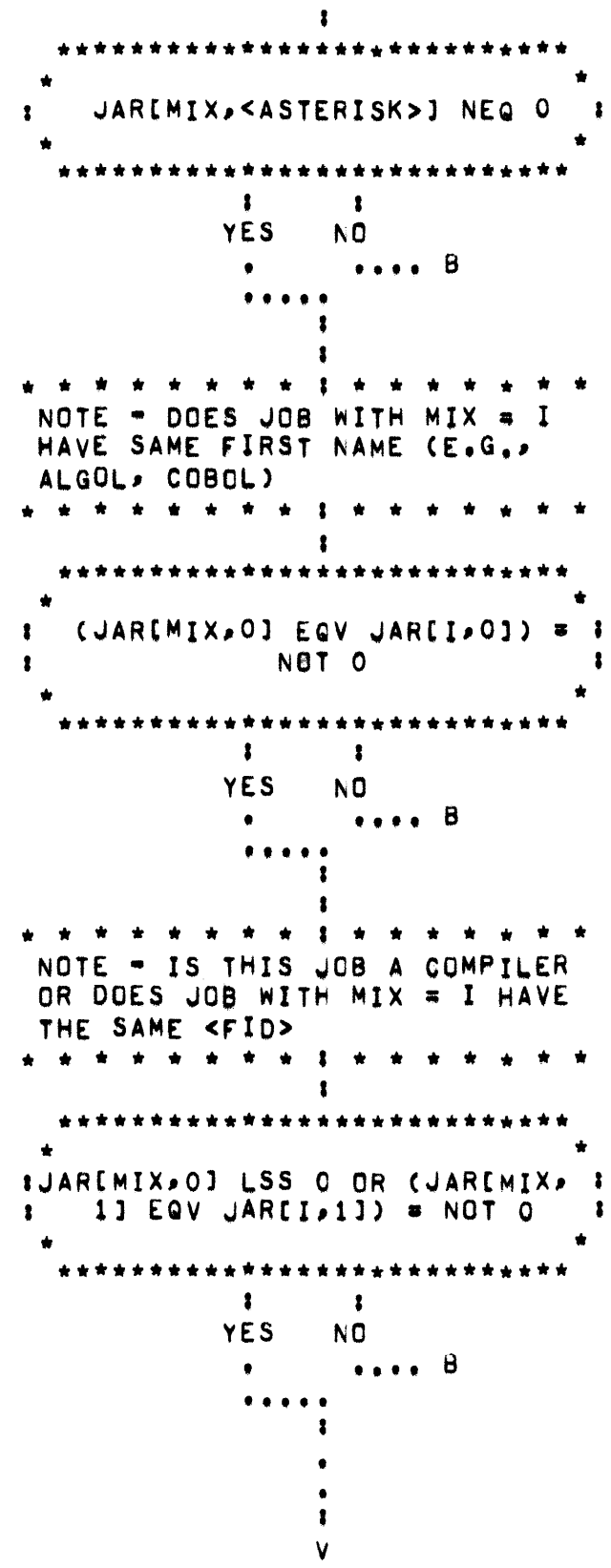
B). PRT[MIX,4]:

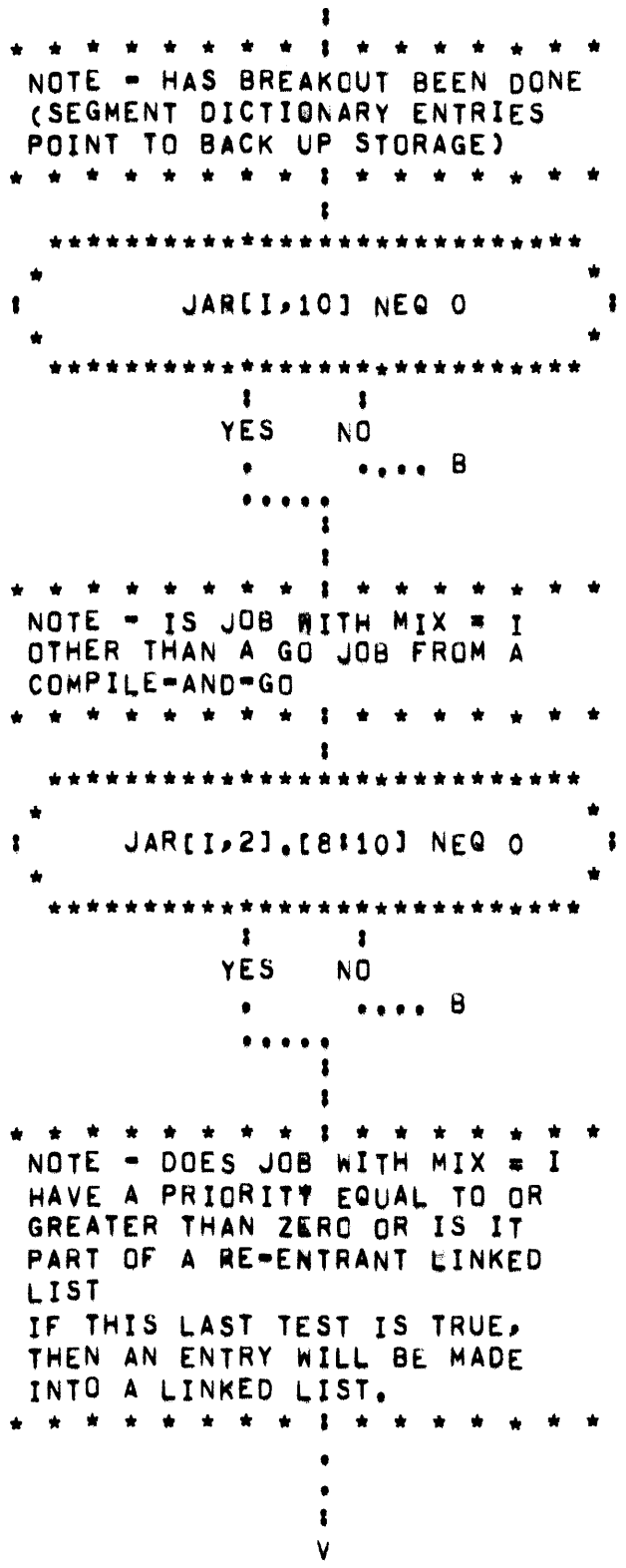


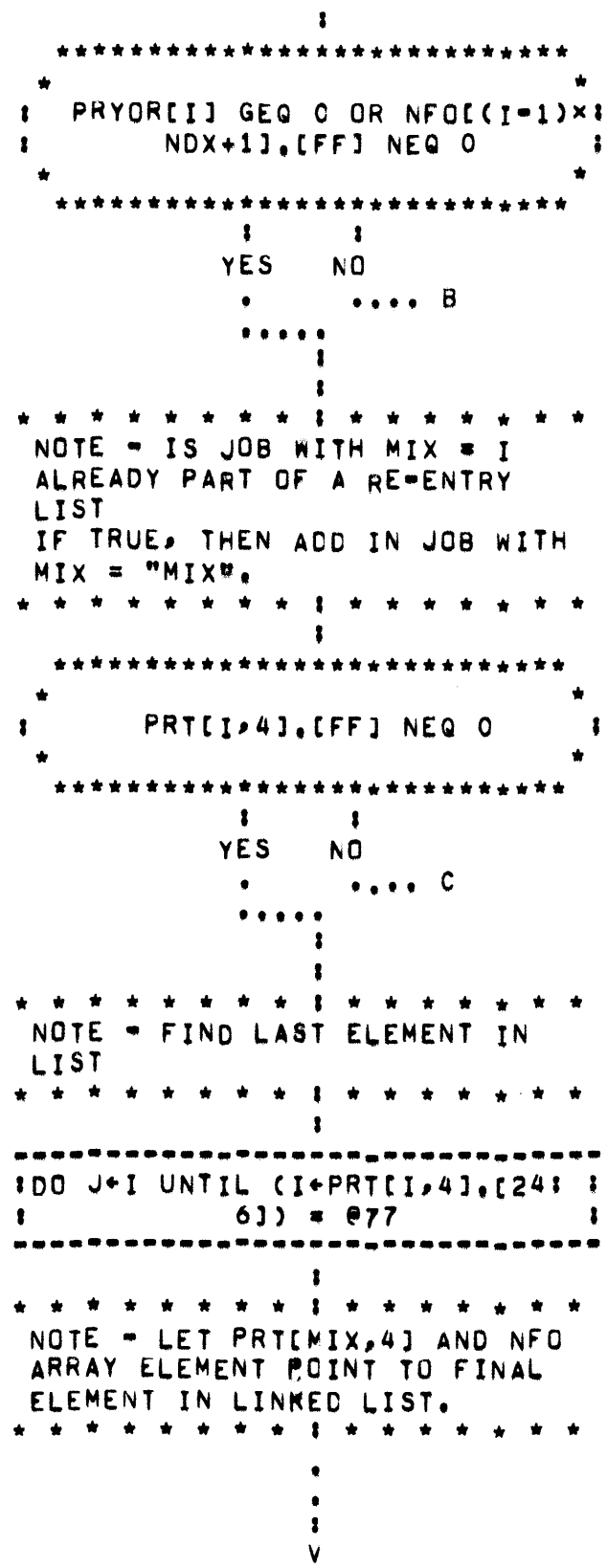
IN THIS INSTANCE, THE JOB WITH MIX=2 WAS THE FIRST TO USE THE CODE AND IT WILL BE THIS PROGRAM'S SEGMENT DICTIONARY WHICH WILL BE USED WITH OTHER JOBS. THE JOB WITH MIX=3 WILL ALWAYS BE THE LAST ENTRY IN THE LIST. NEW ENTRIES WILL BE INSERTED DIRECTLY BEFORE IT.

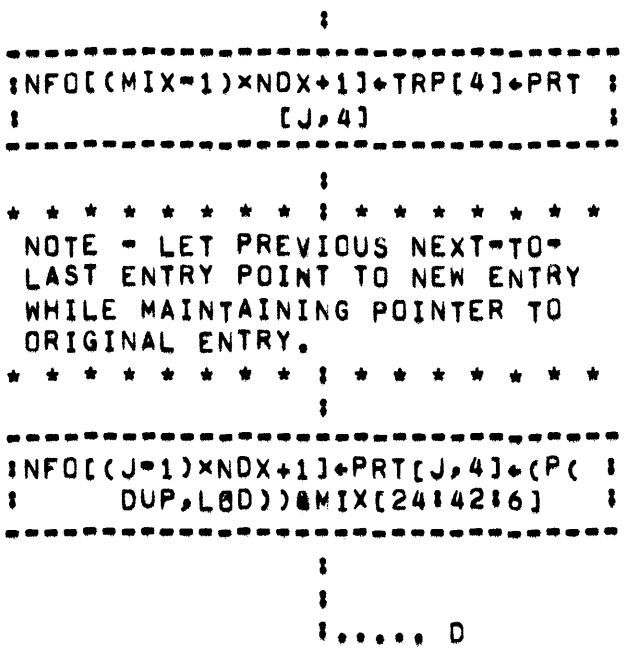
THE FOLLOWING FLOWCHARTS DESCRIBE THE RE-ENTRANT CODE HANDLING PORTIONS OF THE SELECTRUN PROCEDURE (I.E., REEENTRY SUBROUTINE)

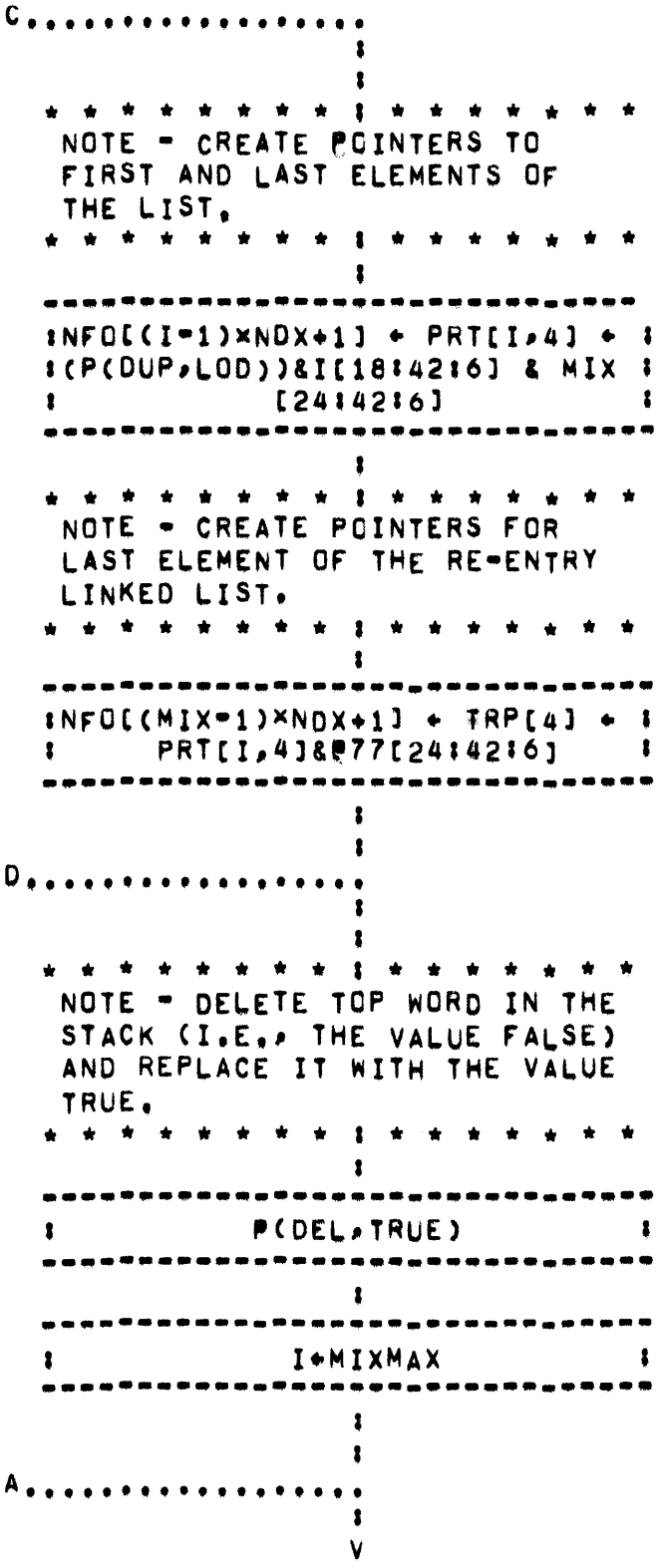

ENTER
PLACE VALUE FALSE IN TOP OF STACK
A
NOTE - IS JOB OTHER THAN A GO JOB FROM A COMPILE AND GO
(JAR[MIX,2],[8:10] NEQ 0)
YES
NO
A
NOTE - WILL TEST ALL MIX NUMBERS FOR POSSIBLE CODE COMPATIBILITY I.E., RE-ENTRANCY)
B
FOR LOOP: I←1 UNTIL MIXMAX
NOTE - IS THERE A JOB WITH MIX NUMBER I

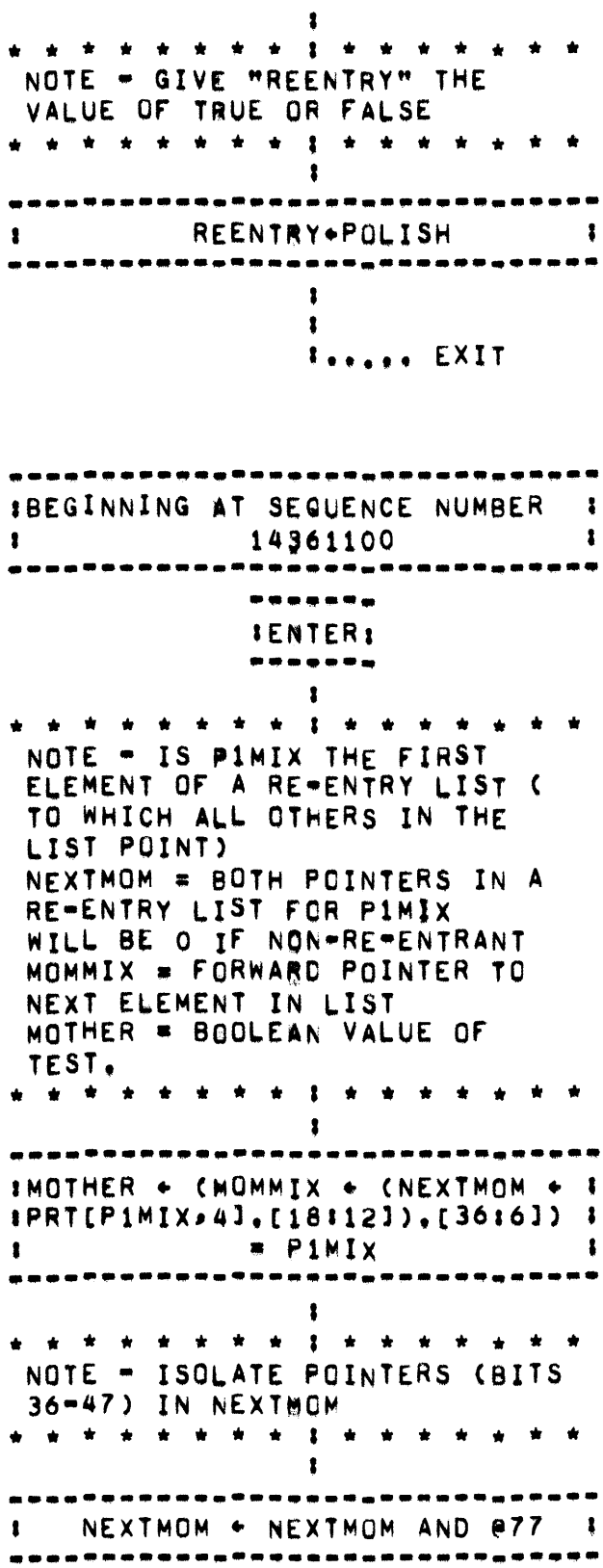

JAR[MIX,<ASTERISK>] NEQ 0
YES
NO
B
NOTE - DOES JOB WITH MIX = I HAVE SAME FIRST NAME (E.G., ALGOL, COBOL)
(JAR[MIX,0] EQV JAR[I,0]) = NOT 0
YES
NO
B
NOTE - IS THIS JOB A COMPILER OR DOES JOB WITH MIX = I HAVE THE SAME <FID>
JAR[MIX,0] LSS 0 OR (JAR[MIX, 1] EQV JAR[I,1]) = NOT 0
YES
NO
B

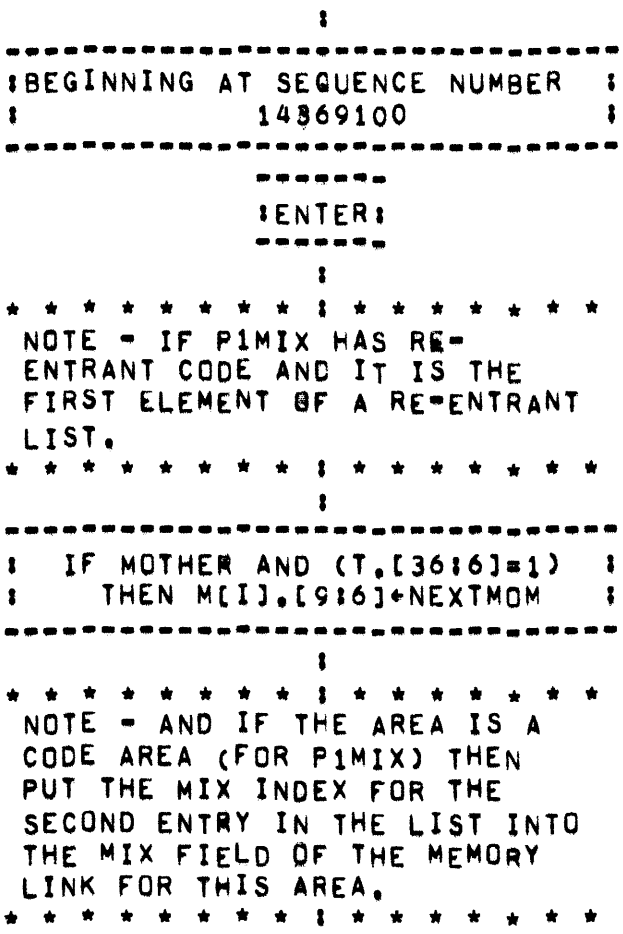

NOTE - HAS BREAKOUT BEEN DONE (SEGMENT DICTIONARY ENTRIES POINT TO BACK UP STORAGE)
JAR[I,10] NEQ 0
YES
NO
B
NOTE - IS JOB WITH MIX = I OTHER THAN A GO JOB FROM A COMPILE-AND-GO
JAR[I,2].[8:10] NEQ 0
YES
NO
B
NOTE - DOES JOB WITH MIX = I HAVE A PRIORITY EQUAL TO OR GREATER THAN ZERO OR IS IT PART OF A RE-ENTRANT LINKED LIST
IF THIS LAST TEST IS TRUE, THEN AN ENTRY WILL BE MADE INTO A LINKED LIST.

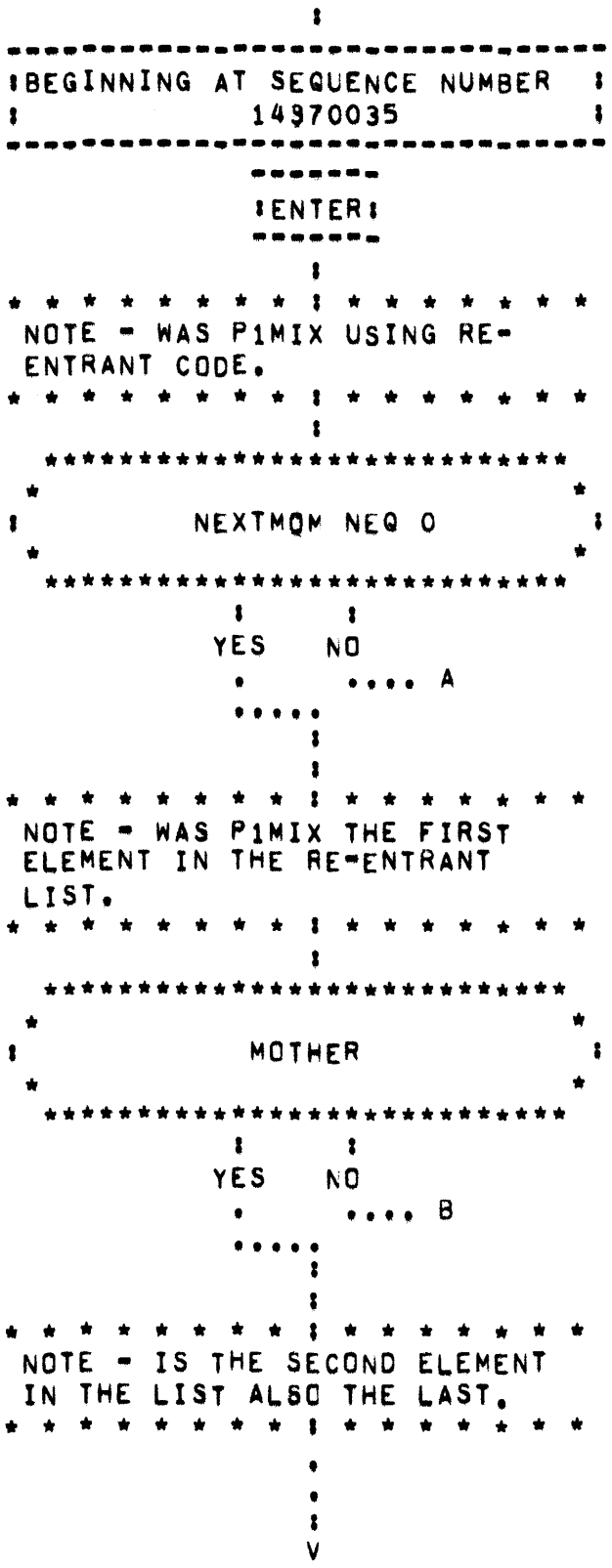

PRYOR[I] GEQ 0 OR NFO[(I-1)×NDX+1],[FF] NEQ 0
YES
NO
B
NOTE = IS JOB WITH MIX = I ALREADY PART OF A RE-ENTRY LIST
IF TRUE, THEN ADD IN JOB WITH MIX = "MIX".
PRT[I,4],[FF] NEQ 0
YES
NO
C
NOTE = FIND LAST ELEMENT IN LIST
DO J←I UNTIL (I←PRT[I,4],[24:6]) = @77
NOTE = LET PRT[MIX,4] AND NFO ARRAY ELEMENT POINT TO FINAL ELEMENT IN LINKED LIST.

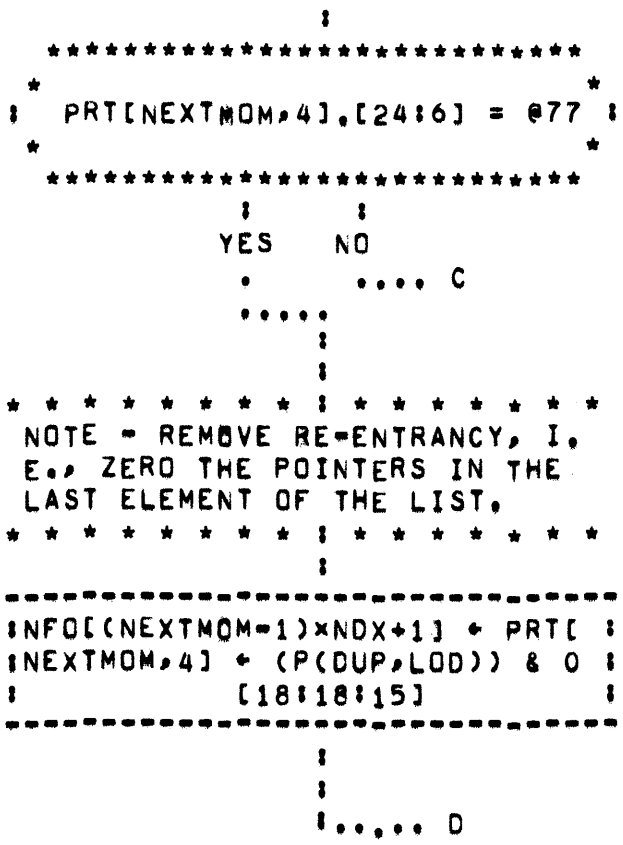

NFO[(MIX-1)×NDX+1]←TRP[4]←PRT
[J,4]
NOTE - LET PREVIOUS NEXT-TO-
LAST ENTRY POINT TO NEW ENTRY
WHILE MAINTAINING POINTER TO
ORIGINAL ENTRY.
NFO[(J-1)×NDX+1]←PRT[J,4]←(P(
DUP,L⌀D))⌀MIX[24142:6]
D

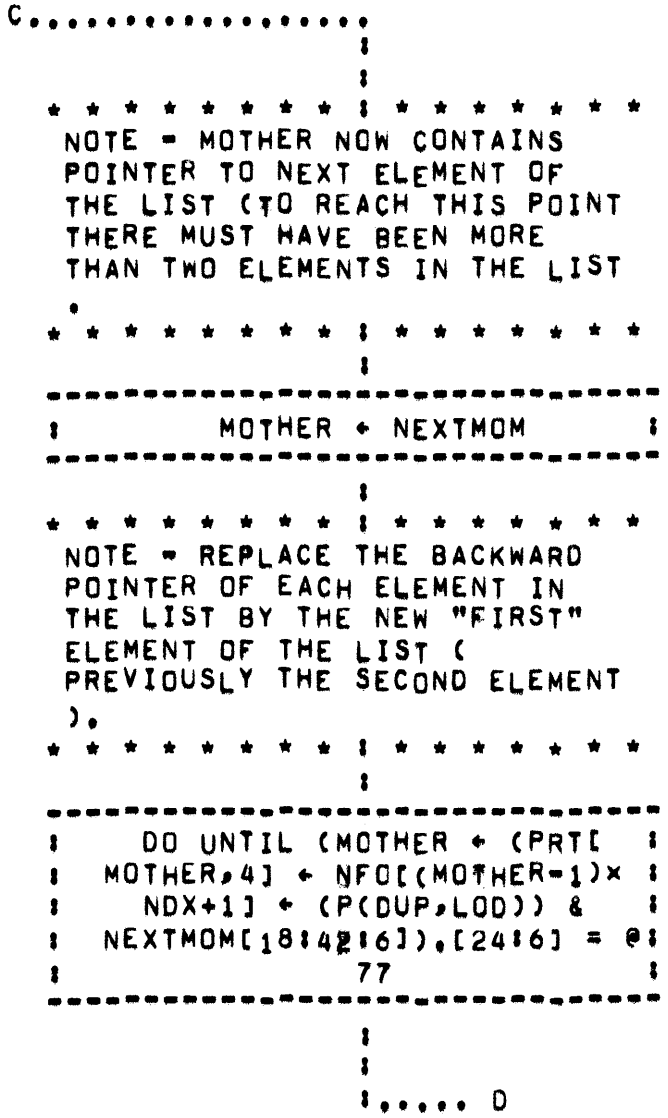

C
* * * * * * * * * * * * * * * *
NOTE - CREATE POINTERS TO FIRST AND LAST ELEMENTS OF THE LIST.
* * * * * * * * * * * * * * * *
NFO[(I-1)×NDX+1] ← PRT[I,4] ←
(P(DUP,LOD))&I[18:42:6] & MIX [24:42:6]
* * * * * * * * * * * * * * * *
NOTE - CREATE POINTERS FOR LAST ELEMENT OF THE RE-ENTRY LINKED LIST.
* * * * * * * * * * * * * * * *
NFO[(MIX-1)×NDX+1] ← TRP[4] ←
PRT[I,4]&@77[24:42:6]
D
* * * * * * * * * * * * * * * *
NOTE - DELETE TOP WORD IN THE STACK (I.E., THE VALUE FALSE) AND REPLACE IT WITH THE VALUE TRUE.
* * * * * * * * * * * * * * * *
P(DEL,TRUE)
I←MIXMAX
A

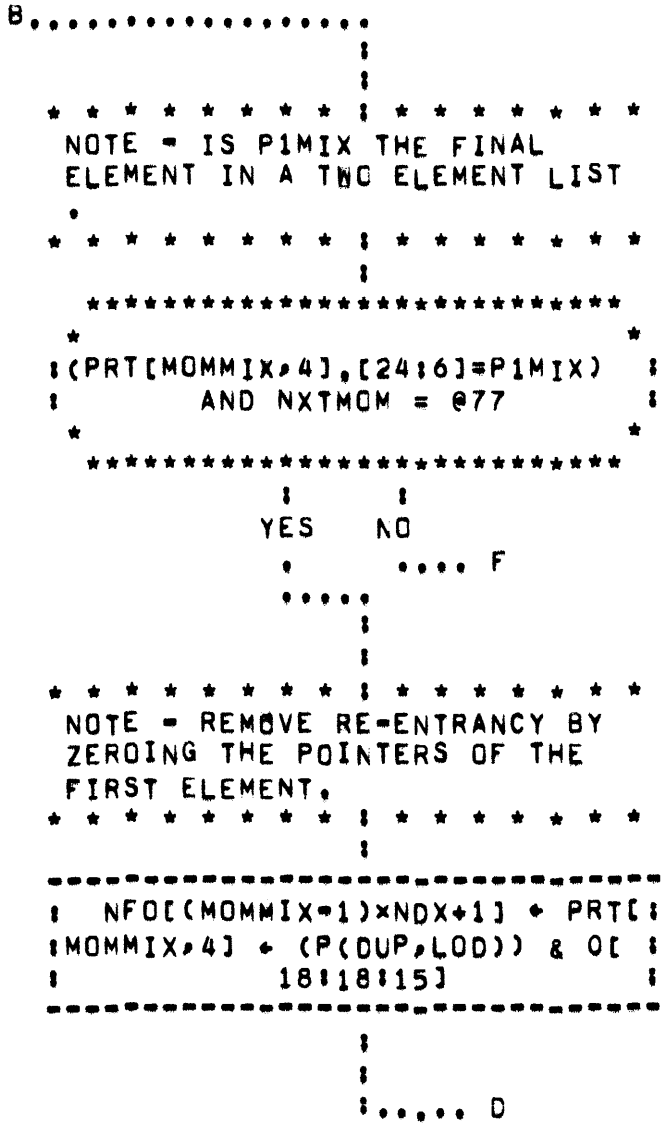

NOTE - GIVE "REENTRY" THE VALUE OF TRUE OR FALSE
REENTRY←POLISH
EXIT
BEGINNING AT SEQUENCE NUMBER 14361100
ENTER
NOTE - IS P1MIX THE FIRST ELEMENT OF A RE-ENTRY LIST (TO WHICH ALL OTHERS IN THE LIST POINT)
NEXTMOM = BOTH POINTERS IN A RE-ENTRY LIST FOR P1MIX WILL BE 0 IF NON-RE-ENTRANT
MOMMIX = FORWARD POINTER TO NEXT ELEMENT IN LIST
MOTHER = BOOLEAN VALUE OF TEST.
MOTHER ← (MOMMIX ← (NEXTMOM ← PRT[P1MIX,4].[18:12]).[36:6]) = P1MIX
NOTE - ISOLATE POINTERS (BITS 36-47) IN NEXTMOM
NEXTMOM ← NEXTMOM AND @77

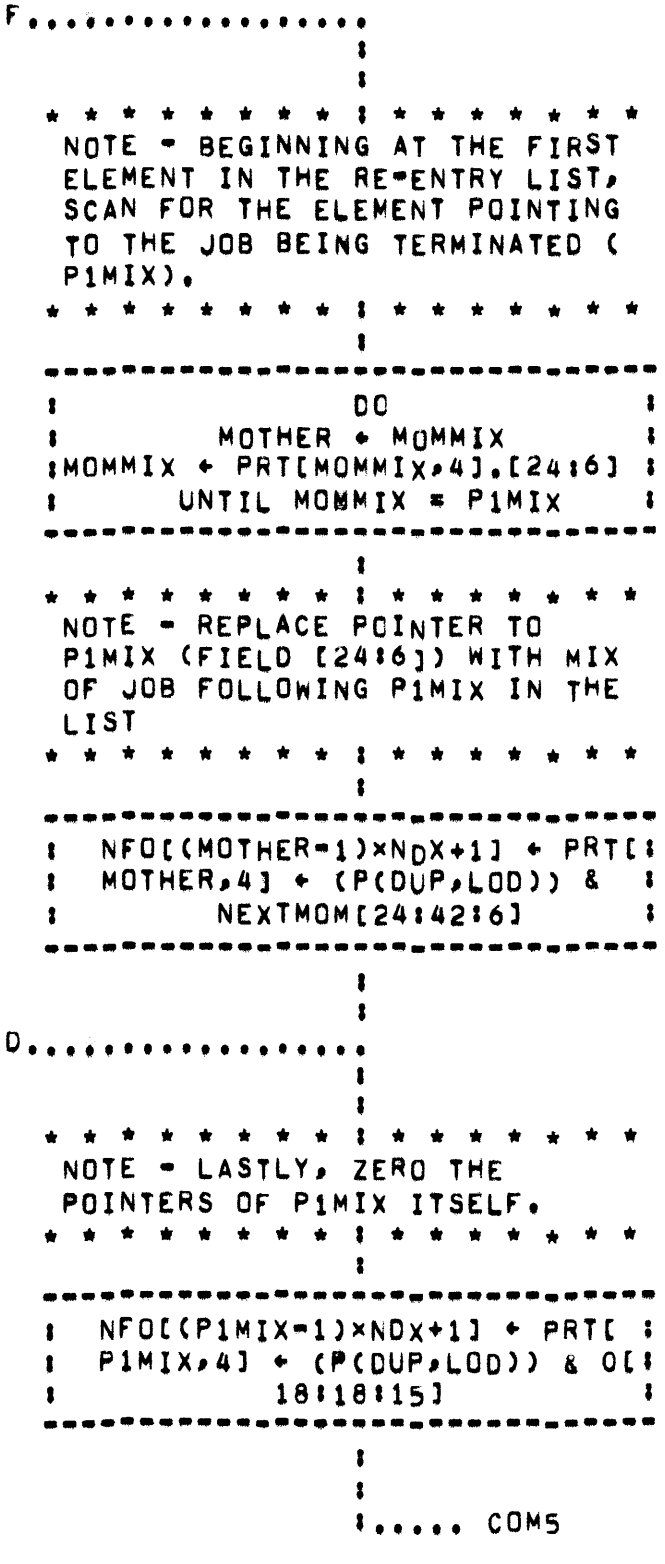

BEGINNING AT SEQUENCE NUMBER
14869100
ENTER
NOTE - IF P1MIX HAS RE-ENTRANT CODE AND IT IS THE FIRST ELEMENT OF A RE-ENTRANT LIST.
IF MOTHER AND (T.[36:6]=1)
THEN M[I].[9:6]←NEXTMOM
NOTE - AND IF THE AREA IS A CODE AREA (FOR P1MIX) THEN PUT THE MIX INDEX FOR THE SECOND ENTRY IN THE LIST INTO THE MIX FIELD OF THE MEMORY LINK FOR THIS AREA.


BEGINNING AT SEQUENCE NUMBER
14970035
ENTER
NOTE - WAS P1MIX USING RE-ENTRANT CODE.
NEXTMOM NEQ 0
YES
NO
A
NOTE - WAS P1MIX THE FIRST ELEMENT IN THE RE-ENTRANT LIST.
MOTHER
YES
NO
B
NOTE - IS THE SECOND ELEMENT IN THE LIST ALSO THE LAST.


PRT[NEXTMOM,4].[24:6] = @77
YES
NO
C
NOTE - REMOVE RE-ENTRANCY, I.
E., ZERO THE POINTERS IN THE
LAST ELEMENT OF THE LIST.
NFO[(NEXTMOM-1)*NDX+1] ← PRT[
NEXTMOM,4] ← (P(DUP,LOD)) & 0
[18:18:15]
D


C
NOTE = MOTHER NOW CONTAINS POINTER TO NEXT ELEMENT OF THE LIST (TO REACH THIS POINT THERE MUST HAVE BEEN MORE THAN TWO ELEMENTS IN THE LIST
MOTHER ← NEXTMOM
NOTE = REPLACE THE BACKWARD POINTER OF EACH ELEMENT IN THE LIST BY THE NEW "FIRST" ELEMENT OF THE LIST (PREVIOUSLY THE SECOND ELEMENT).
DO UNTIL (MOTHER ← (PRT[MOTHER,4] ← NFO[(MOTHER-1)× NDX+1] ← (P(DUP,LOD)) & NEXTMOM[18:42:6]),[24:6] = @ 77
D


B
NOTE - IS P1MIX THE FINAL ELEMENT IN A TWO ELEMENT LIST
(PRT[MOMMIX,4],[24:6]=P1MIX) AND NXTMOM = @77
YES
NO
F
NOTE - REMOVE RE-ENTRANCY BY ZEROING THE POINTERS OF THE FIRST ELEMENT.
NFO[(MOMMIX-1)×NDX+1] ← PRT[MOMMIX,4] ← (P(DUP,LOD)) & 0[18:18:15]
D


F
NOTE - BEGINNING AT THE FIRST ELEMENT IN THE RE-ENTRY LIST, SCAN FOR THE ELEMENT POINTING TO THE JOB BEING TERMINATED (P1MIX).
DO
MOTHER ← MOMMIX
MOMMIX ← PRT[MOMMIX,4].[24:6]
UNTIL MOMMIX = P1MIX
NOTE - REPLACE POINTER TO P1MIX (FIELD [24:6]) WITH MIX OF JOB FOLLOWING P1MIX IN THE LIST
NFO[(MOTHER-1)×NDX+1] ← PRT[MOTHER,4] ← (P(DUP,LOD)) & NEXTMOM[24:42:6]
D
NOTE - LASTLY, ZERO THE POINTERS OF P1MIX ITSELF.
NFO[(P1MIX-1)×NDX+1] ← PRT[P1MIX,4] ← (P(DUP,LOD)) & 0[18:18:15]
..... COM5

REPLY

REPLY RETURNS THE UNIT NUMBER AND A CODE INDICATING THE MESSAGE USED. IT IS SET NEGATIVE WHEN THE JOB IS SLEEPING AND BECOMES POSITIVE WHEN THE JOB WAKES UP.

RESTARTING

CONTAINS NEGATIVE OF MIX INDEX OF JOB BEING RESTART"ED OR ZERO IF NO JOB IS BEING RESTARTED. IT IS USED FOR CONTROL PURPOSES, PRIMARILY TO PREVENT TWO SIMULATANEOUS RESTARTS, AND TO PREVENT INCORRECT TERMINATION OF RESTARTING JOB.

RESULT1

RESULT DESCRIPTOR TO I/O CHANNEL 1.

RESULT2

RESULT DESCRIPTOR FOR I/O CHANNEL 2.

RESULT3

RESULT DESCRIPTOR FOR I/O CHANNEL 3.

RESULT4

RESULT DESCRIPTOR FOR I/O CHANNEL 4.

REWINDANDLOCK

REWINDANDLOCK(WHAT) IS A PROCEDURE WHICH DOES THE FOLLOWING IF THE SPECIFIED UNIT "WHAT" IS NOT IN USE:

1. "RRRMECH" BIT FOR THIS UNIT IS TURNED ON.
2. UNIT IS MARKED IN USE BY "CONTROLCARD".
3. UNIT IS REWOUND.
4. "RRRMECH" BIT FOR THIS UNIT IS TURNED OFF.
5. "READY" BIT FOR THIS UNIT IS TURNED OFF.
6. "LABELTABLE[U]" IS MARKED AS RW/L (@214).

RQONE

USED AS TEMPORARY STORAGE BY REMOTELOGGER ROUTINE.

RQTWO

USED AS TEMPORARY STORAGE BY REMOTELOGGER ROUTINE.

RRNCOUNT

COUNT OF READ-READY-NORMAL DATA COMMUNICATIONS INTERRUPTS.

RRRMECH

MASK WORD USED BY STATUS TO CHECK I/O DEVICES FOR READY OR NOT READY STATUS.

RUN

RUN(MIX) IS A PROCEDURE WHICH SETS P1MIX TO "MIX", INITIALIZES PROCESSOR TIME, SETS STACKUSE TRUE, AND BRANCHES TO ENTERNAL.

RUNUMBER

COUNTER FOR THE NUMBER OF PSEUDO READERS CURRENTLY OPEN.

SAVERESULT

DESCRIPTOR POINTING TO THE SAVERESULT ARRAY WHICH MAY CONTAIN DEBUGGING INFORMATION STORED CYCLICALLY BY REFERENCING THE PARAMETER DEFINE "STORAWAY". THE LAST ENTRY IS POINTED TO BY SVRESULT. THE SIZE OF THE ARRAY, SPECIFIED BY A DEFINE AT LINE 00006500 IN THE MCP, IS CURRENTLY 128 WORDS. IN ORDER TO INCLUDE THIS DEBUGGING FACILITY, THE COMPILE-TIME OPTION SAVERESULTS MUST BE SET TRUE.

SAVEWORD

USED TO INDICATE WHICH IN-USE DEVICES ARE TO BE LOGICALLY SAVED.

SHEDULEIDS

CONTAINS A BIT MASK MAINTAINED BY THE SELECTION ROUTINE TO ASSIGN SCHEDULE IDENTIFICATIONS TO JOBS BEING SCHEDULED.

SCN

SCN(UNITNO,CARDLOC,SOURCE,ACCUM,COUNT,LASTSCAN) IS THE MAIN ROUTINE FOR CONTROL CARDS (USUALLY CALLED BY A SUBROUTINE SCAN) FROM LOGICAL UNIT "UNITNO". "CARDLOC" AND "SOURCE" ARE LOCATIONS OF THE INFORMATION. "ACCUM" IS AN ARRAY WHICH HOLDS THE RESULT OF THE SCAN. "COUNT" IS OF CHARACTERS. "LASTSCAN" IS A REMEMBERING FLAG. SCN RETURNS ARE CODED AS FOLLOWS:

0	PERIODS.
1	SLASH.
2	QUESTION MARK.
3	SPECIAL CHARACTERS.
4	IDENTIFIERS WHICH ARE NOT RESERVED.
5	IDENTIFIER IN DIRECT.
13	PRIORITY.

SECONDCTR

COUNTER USED BY NSECOND ROUTINE AND OUTER BLOCK CODE FOR COORDINATION AND INTERLOCK PURPOSES

SECURITYCHECK

SECURITYCHECK(MID,FID,USERID,HEADER) IS A PROCEDURE WHICH LOCATES AND GUARENTEES SECURTY RESTRICTIONS FOR FILES ON DISK. THE PARAMETERS ARE DEFINED AS FOLLOWS:

PARAM	VALUE	DESCRIPTION
MID		MULTI FILE ID OF FILE TO BE CHECKED
FID		FILE ID OF FILE TO BE CHECKED
USERID		USER IDENTIFICATION (USERCODE)
HEADER	=0	DO DIRECTORYSEARCH AND PASS BACK HEADER
	>0	MEMORY LOCATION OF FILE HEADER (DISK ADDRESS OF FILE HEADER FOR FILE "MID"
	<0	SCRAMBLE BLOCK DISK ADDRESS OF FILE HEADER. CALL IN HEADER AND DO PRIMARY SECURITY CHECK (HEADER IS NOT RETURNED)

THE RESULT FROM SECURITYCHECK IS AS FOLLOWS:

0 = NO LEGITIMATE USER FOUND
2 = TERTIARY USER (INPUT ONLY)
3 = SECONDARY USER (INPUT/OUTPUT)
7 = PRIMARY USER (INPUT/OUTPUT/LIBRARY MAINTENANCE)

SECURITYMAINT

SECURITYMAINT(TYPE,SMID,SFID,CMM,SFH,CARD) IS A PROCEDURE WHICH HANDLES THE DISK FILE SECURING/RELEASING CONTROL CARDS FOR FILES WITH ASSOCIATED SECURITY FILE "SMID"/"SFID". "TYPE" IS A FUNCTION CODE, AND "CMM", "SFH", AND "CARD" THE INFORMATION PASSED BY CONTROLCARD.

SEEKNAM

SEEKNAM(A,B,C,D,E) IS A PROCEDURE WHICH SEARCHES THE DISK DIRECTORY FOR "A"/"B", STARTING AT DISK ADDRESS "C" AND RETURNING THE NAMES FOUND IN "D" AND "E".

SEGMENT DICTIONARY

EACH PROGRAM HAS A SEGMENT DICTIONARY CONTAINING ONE ENTRY FOR EVERY PROGRAM SEGMENT IN THE PROGRAM, AND ONE 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 SEGMENTS SEGMENT 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.

1. 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.)

2. THE SIZE OF THE SEGMENT.

3. AN INDEX, INTO THE "PRT", OF THE FIRST PROGRAM DESCRIPTOR THAT REFERENCES THE SEGMENT.

4. A FLAG SPECIFYING IF THE SEGMENT IS A TYPE 2 SEGMENT OR NOT.

ENTRIES FOR INTRINSICS PROVIDE NO SEGMENT SIZE AND HAVE THE INTRINSICS NUMBER IN LIEU OF THE RELATIVE DISK ADDRESS, BUT ARE OTHERWISE THE SAME. THE INTRINSIC CORRESPONDING TO A GIVEN NUMBER OR THE NUMBER CORRSPONDING TO AN INTRINSIC MAY BE DERRIVED FROM A LISTING OF THE INTRINSICS.

PRT ENTRIES

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:

1. THE RELATIVE ADDRESS WITHIN THE SEGMENT, PERTINENT TO THE PROGRAM DESCRIPTOR.

2. THE INDEX, INTO THE SEGMENT DICTIONARY, OF THE ENTRY FOR THE SEGMENT TO WHICH THE PROGRAM DESCRIPTOR PERTAIN (THIS INDEX IS EQUAL TO THE SEGMENT-S SEGMENT NUMBER).

3. A LINK (INDEX) TO THE NEXT PROGRAM DESCRIPTOR WHICH ADDRESSES THE SAME SEGMENT.

4. A STOPPER BIT IF THE PROGRAM DESCRIPTOR ENTRY IS THE LAST ONE PERTAINING TO THE SEGMENT.

FIELDS AND THEIR VALUES
FOR SEGMENT DICTIONARY AND RELATED PRT CELLS

PRT

FIELD	CONTENTS
[0:4]	NON-PRESENT PROGRAM DESCRIPTOR BITS.
[4:2]	MODE AND ARGUMENT BITS.
[6:1]	STOPPER BIT SHOWING END OF LINK LIST.
[7:11]	IF STOPPER BIT IS ON, INDICATING THE ENTRY TO BE THE LAST ENTRY, THIS FIELD CONTAINS THE INDEX INTO THE SEGMENT DICTIONARY; OTHERWISE, ITS A LINK (INDEX) TO THE NEXT PROGRAM DESCRIPTOR THAT REFERENCES THE SEGMENT.
[18:15]	"F" REGISTER FIELD USED AT RUN-TIME IN LABEL AND ACCIDENTAL ENTRY DESCRIPTORS
[33:15]	CORE ADDRESS FOR PRESENT SEGMENTS OR RELATIVE ADDRESS FOR ABSENT SEGMENTS (RELATIVE TO BEGINNING OF SEGMENT)

SEGMENT DICTIONARY

FIELD	CONTENTS
[0:1]	----
[1:1]	1 FOR TYPE 2 SEGMENTS, 0 OTHERWISE
[2:1]	1 FOR INTRINSICS, 0 OTHERWISE
[3:1]	= 1 IF BEING MADE PRESENT, = 0 OTHERWISE (INTERLOCK FOR RE-ENTRANT CODE) ([3:3]=6 WHEN LOCKING)
[4:2]	= 0 FOR NORMAL SEGMENTS = 3 FOR SEGMENTS OVERLAID TO AUX. MEM. = 2 FOR SEGMENTS TO BE OVERLAID TO AUXILIARY MEMORY WHICH HAVEN"T BEEN
[6:2]	UNUSED
[8:10]	LINK TO PROGRAM DESCRIPTOR IN "PRT" (LINKS TO 1ST ENTRY IN LINK-LIST)

[18:15] SIZE OF SEGMENT, IF PROGRAM SEGMENT;
1 = NEVER-PRESENT INTRINSICS
[33:15] DISK ADDRESS OF SEGMENT OR INTRINSIC NUMBER

- SEGMENT DICTIONARY -

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	.[18:15]	LOCATION OF LINE DICTIONARY
	.[33:15]	LOCATION OF SEGMENT DICTIONARY
1		SIZE OF SEGMENT DICTIONARY
1	.[1:1]	= 0 MAKE A "OAT" ENTRY IN PRT[11]
		= 1 JOB COMPILED BY COBOL
	.[9:39]	RELATIVE LOCATION OF THE PRT
3		SIZE OF PRT
4		LOCATION OF "FPB"
5		SIZE OF "FPB"
6	.[1:1]	1 = NEW FORMAT SEGMENT ZERO
	.[2:1]	1 = FORTRAN ERROR RECOVERY
	.[9:39]	STARTING SEGMENT NUMBER
7	.[18:15]	CORE REQUIREMENT DIV 64
	.[33:15]	NUMBER OF FILES
8-14		UNDEFINED
15		DISK ADDRESS OF LABEL EQUATION ENTRIES PRESENTED WHEN PROGRAM WAS COMPILED
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		SAVE TIME (FROM COMPILATION)
23-29		UNDEFINED

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.

FORMAT OF FIRST 30 WORDS (1 DISK SEGMENT)
OF ALL RESTART FILES

WORD	FIELD	CONTENTS
0	[8:10]	ROW OF LAST ML ML SEGMENT 10,11,12...
	[18:15]	RELATIVE ADDRESS IN THAT ROW OF THAT SEGMENT
	[33:15]	TOTAL SIZE OF ML
1		GRSD[X] = POINTER TO RSD
	[1:1]	1 = EI BREAK CODE
	[8:5]	OLD UNIT + 1 (AFTER REELER ONLY)
	[18:7]	BREAK COUNT 00,01,02...
	[25:8]	SELECTION FILE COUNT
	[33:15]	OVERLAY DISK ADDRESS OF RSD
2	[1:2]	0 = NORMAL 2 = IF JOB HAS BEEN XS-ED 3 = IF JOB HAS BEEN ES-ED (DURING SELECTION)
	[18:15]	ROW OF FIRST CHUNK 10,11,...
	[33:15]	RELATIVE ADDRESS IN THAT ROW OF THAT CHUNK
3	[1:1]	1 = MARKS THIS AS A RESTART FILE
	[18:15]	RSD SIZE - 1
	[33:15]	LIMIT OF PLACEMENT NEEDS
4		PRTROW[X] = PRT POINTER
5	[18:15]	NFO ADDRESS, LOWER BOUND OF MCP TABLES + 1 (*1)
	[33:15]	UPPER BOUND OF MCP TABLES (*1)
6	[1:1]	1 = NEW FORMAT OF SEGMENT ZERO
	[8:10]	BIT MAP INDICATING MEMORY MOD SITUATION (*1)
	[33:15]	MSTART
7	[18:15]	CORE ESTIMATE
	[33:15]	NUMBER OF FILES
8-29		SAME AS NORMAL PROGRAM SEGMENT (*2)

*1 SELECTION MUST MATCH THESE BEFORE ALLOWING RESTART
*2 MAY BE MODIFIED BY CONTROL CARDS.

M[NFO-1] CONTAINS THE CONTENTS OF THE WORD BEFORE NFO USED AS A CHECK FOR SOME MCP-S.

- SEGMENT DICTIONARY -

LINE DICTIONARY

THE LINE DICTIONARY IS USED TO CREATE THE LINE NUMBER OUTPUT ON ABNORMAL PROGRAM TERMINATION RATHER THAN THE SEGMENT NUMBER AND ADDRESS. THIS FEATURE IS ENABLED WHEN THE PROGRAM IS COMPILED WITH THE DOLLAR SIGN OPTION "SEQXEQ".

ENTRY FOR EACH SEGMENT (=1 MEANS NO ENTRY)
[FF] LENGTH OF LINE SEGMENT
[CF] LOCATION OF LINE SEGMENT IN CODE FILE

ENTRY FOR EACH CARD
[10:28] SEQUENCE NUMBER IN BINARY
[38:10] RELATIVE ADDRESS IN CODE SEGMENT.

SEGMENT ZERO (DISK)

DISK SEGMENT ZERO CONTAINS THE NAMES OF THE MCP AND INTRINSIC FILES FOR ALL SYSTEMS. THE "KERNEL" READS THIS INFORMATION DURING A HALT/LOAD OPERATION AND LOADS THE SPECIFIED MCP FILE. THE MCP READS SEGMENT ZERO TO DETERMINE THE APPROPRIATE INTRINSICS FILE.

INFORMATION IS ENTERED IN THE SEGMENT THROUGH THE "CM" AND "CI" KEYBOARD MESSAGES AND BY THE "COOL START" AND "COLD START" PROGRAMS. THE FORMAT OF THE SEGMENT IS AS FOLLOWS:

```
0  NUMBER OF SYSTEMS ATTACHED 1,2,3,4
1  DISK ADDRESS OF DIRECTORYTOP
2
3
4  DIRECT
5  SYSTEM #1          .[18:15] = SERIAL NO. OF NEXT
                                 STLOG/STATS FILE
                      .[33:15] = SERIAL NO. OF NEXT
                                 SYSTEM/STATS FILE
6  SYSTEM #2          .[18:15] = SERIAL NO. OF NEXT
                                 STLOG/STATS FILE
                      .[33:15] = SERIAL NO. OF NEXT
                                 SYSTEM/STATS FILE
7  SYSTEM #3          .[18:15] = SERIAL NO. OF NEXT
                                 STLOG/STATS FILE
                      .[33:15] = SERIAL NO. OF NEXT
                                 SYSTEM/STATS FILE
8  SYSTEM #4          .[18:15] = SERIAL NO. OF NEXT
                                 STLOG/STATS FILE
                      .[33:15] = SERIAL NO. OF NEXT
                                 SYSTEM/STATS FILE
9
10 SYSTEM #1          MFID OF MCP
11                    FID OF MCP
12                    BASE ADDRESS OF MCP
13                    MFID OF INTRINSICS
14                    FID OF INTRINSICS
15 SYSTEM #2          MFID OF MCP
16                    FID OF MCP
17                    BASE ADDRESS OF MCP
18                    MFID OF INTRINSICS
19                    FID OF INTRINSICS
20 SYSTEM #3          MFID OF MCP
21                    FID OF MCP
22                    BASE ADDRESS OF MCP
```

- SEGMENT ZERO (DISK) -

23		MFID OF INTRINSICS
24		FID OF INTRINSICS
25	SYSTEM #4	MFID OF MCP
26		FID OF MCP
27		BASE ADDRESS OF MCP
28		MFID OF INTRINSICS
29		FID OF INTRINSICS

SEGMENT ZERO (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	.[18:15]	LOCATION OF LINE DICTIONARY
	.[33:15]	LOCATION OF SEGMENT DICTIONARY
1		SIZE OF SEGMENT DICTIONARY
1	.[1:1]	= 0 MAKE A "OAT" ENTRY IN PRT[11]
		= 1 JOB COMPILED BY COBOL
	.[9:39]	RELATIVE LOCATION OF THE PRT
2	.[2:1]	=1 JOB HAS DECLARED INTERRUPTS (IPC)
	.[3:1]	=1 IPC PROGRAM FILE
		WHETHER INVOKING OR INVOKED
	.[4:1]	=1 INVOKED IPC PROGRAM FILE
3		SIZE OF PRT
4		LOCATION OF "FPB"
5		SIZE OF "FPB"
6	.[1:1]	1 = NEW FORMAT SEGMENT ZERO
	.[9:39]	STARTING SEGMENT NUMBER
7	.[2:1]	1 = FORTRAN ERROR RECOVERY
	.[18:15]	CORE REQUIREMENT DIV 64
	.[33:15]	NUMBER OF FILES
8		FOR IPC PROGRAM FILE
		NUMBER OF TASK PARAMETERS TO BE RECEIVED
9		FOR IPC PROGRAM FILE
		DISK ADDRESS OF PARAMETER DESCRIPTION SEGMENT
10-14		UNDEFINED
15		DISK ADDRESS OF LABEL EQUATION ENTRIES
		PRESENTED WHEN PROGRAM WAS COMPILED
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		SAVE TIME (FROM COMPILATION)
23-29		UNDEFINED

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.

SEGMENT ZERO (RESTART FILES)

THE FORMAT OF THE FIRST 30 WORDS (1 DISK SEGMENT) OF ALL RESTART FILES IS AS FOLLOWS:

WORD	FIELD	CONTENTS
0	[8:10]	ROW OF LAST ML ML SEGMENT 10,11,12,...
	[18:15]	RELATIVE ADDRESS IN THAT ROW OF THAT SEGMENT
	[33:15]	TOTAL SIZE OF ML
1		GRSD[X] = POINTER TO RSD
	[1:1]	1 = EI BREAK CODE
	[8:5]	OLD UNIT + 1 (AFTER REELER ONLY)
	[18:7]	BREAK COUNT 00,01,02,...
	[25:8]	SELECTION FILE COUNT
	[33:15]	OVERLAY DISK ADDRESS OF RSD
2	[1:2]	0 = NORMAL 2 = IF JOB HAS BEEN XS-ED 3 = IF JOB HAS BEEN ES-ED (DURING SELECTION)
	[18:15]	ROW OF FIRST CHUNK 10,11,...
	[33:15]	RELATIVE ADDRESS IN THAT ROW OF THAT CHUNK
3	[1:1]	1 = MARKS THIS AS A RESTART FILE
	[18:15]	RSD SIZE - 1
	[33:15]	LIMIT OF PLACEMENT NEEDS
4		PRTROW[X] = PRT POINTER
5	[18:15]	NFO ADDRESS, LOWER BOUND OF MCP TABLES + 1 (*1)
	[33:15]	UPPER BOUND OF MCP TABLES (*1)
6	[1:1]	1 = NEW FORMAT OF SEGMENT ZERO
	[8:10]	BIT MAP INDICATING MEMORY MOD SITUATION (*1)
	[33:15]	MSTART
7	[18:15]	CORE ESTIMATE
	[33:15]	NUMBER OF FILES
8-29		SAME AS NORMAL PROGRAM SEGMENT (*2)

*1 SELECTION MUST MATCH THESE BEFORE ALLOWING RESTART
*2 MAY BE MODIFIED BY CONTROL CARDS.

M[NFO-1] CONTAINS THE CONTENTS OF THE WORD BEFORE NFO USED AS A CHECK FOR SOME MCP-S.

SELECTION

SELECTION IS DEFINED AS:

```
INDEPENDENTRUNNER(P(,SELECTRUN1),0);
```

SELECTRUN

SELECTRUN IS A PROCEDURE WHICH STARTS JOBS ON THE FOLLOWING BASIS:

1. AN "XS" OR "ES" MESSAGE HAS BEEN ENTERED FOR THIS JOB (IN WHICH CASE, SHEETDIDDLER TURNED ON S[2].[1:1] AND CALLED SELECTION).

2. THE SUM OF THE JOB'S CORE REQUIREMENTS (S[20]) PLUS THE SUM OF THE CORE REQUIREMENTS OF ALL OTHER JOBS ACTUALLY RUNNING (CORE.[FF]) IS LESS THAN THE TOTAL AMOUNT OF CORE AVAILABLE FOR USER PROGRAMS (THE INITIAL SPACE AVAILABLE ,CORE.[CF], TIMES THE MULTIPROCESSING FACTOR, CORE.[4:14]).

3. THE SUM OF THE CORE REQUIREMENTS OF ALL OTHER JOBS ACTUALLY RUNNING EQUALS ZERO (PRESUMABLY ONLY "LDCNTRL/DISK" OR "PRNPBT/DISK" IS RUNNING, SO THE JOB SHOULD BE ABLE TO RUN).

SELECTRUN1

SELECTRUN1 IS A PROCEDURE WHICH CREATES A STACK, SETS STACKUSE TRUE, DOES A CALL ON SELECTRUN, AND CALLS "KILL" TO CLEAN UP THE RESULTS AFTER SELECTRUN.

SETNOTINUSE

SETNOTINUSE(U,RWL) IS A PROCEDURE WHICH MARKS LOGICAL UNIT "U" AS NO LONGER IN USE BY A PROGRAM OR THE MCP, AND MAKES IT NOT READY IF "RWL" IS TRUE.

SHAREDISK INFORMATION

INTRODUCTION

ONE CHARACTERISTIC OF MULTI-SYSTEM INFORMATION PROCESSING INSTALLATIONS IS THAT A GIVEN OBJECT PROGRAM TENDS TO BE TIED TO ONE PARTICULAR SYSTEM. FOR THE MOST PART THIS IS DUE TO THE FACT THAT AN OBJECT PROGRAM ALONG WITH ITS ASSOCIATED DATA FILES ARE FREQUENTLY KEPT ON DISK IN ONLY ONE SYSTEM. IT IS OF COURSE POSSIBLE TO MAINTAIN DUPLICATE OBJECT PROGRAMS AND DATA FILES ON DISK IN SEVERAL SYSTEMS; HOWEVER, THIS ALTERNATIVE HAS DEFINITE DISADVANTAGES IN CASES WHERE THE DATA FILES MUST BE UPDATED FREQUENTLY OR ARE UNUSUALLY LARGE. IN THE FIRST INSTANCE, ONE FACES THE OBVIOUS PROBLEM OF UPDATING SEVERAL COPIES OF THE PARTICULAR FILE EACH TIME CHANGES ARE REQUIRED. SECONDLY, KEEPING MULTIPLE COPIES OF A LARGE FILE ON DISK REPRESENTS RATHER INEFFICIENT USE OF DISK STORAGE RESOURCES. ASSUMING THAT THE USER WISHES TO AVOID THESE TWO PROBLEMS AND CONFINE HIMSELF TO RUNNING A PARTICULAR JOB ON A SINGLE SYSTEM SEVERAL PROBLEMS MAY ARISE.

SUPPOSE AN INSTALLATION HAS TWO B5500 SYSTEMS USING THE STANDARD BATCH MCP, AND THAT A PROGRAM WHICH MAINTAINS A LARGE STOCK INVENTORY FILE ON SYSTEM 1 DISK IS TO BE RUN. IF IT IS FOUND THAT SYSTEM 1 IS PRESENTLY BUSY WITH ADDITIONAL JOBS IN ITS SCHEDULE SEVERAL RATHER UNATTRACTIVE COURSES OF ACTION ARE AVAILABLE. FIRST, THE JOB MAY BE GIVEN A HIGHER PRIORITY THAN THE PREVIOUSLY SCHEDULED JOBS THUS ALLOWING IT TO RUN AS SOON AS THERE IS SUFFICIENT ROOM IN THE MIX. IF THE JOBS IN THE MIX REQUIRE CONSIDERABLE TIME TO COMPLETE AND IT IS NECESSARY TO RUN THE INVENTORY PROGRAM IMMEDIATELY, THE ONLY ALTERNATIVE IS TO DS THOSE JOBS CURRENTLY RUNNING AND INITIATE THE INVENTORY PROGRAM. IN EITHER CASE, THE RUNNING OF THIS PROGRAM IS DONE AT THE EXPENSE OF OTHER JOBS HAVING TO WAIT AND/OR HAVING TO BE RESTARTED. IF ON THE OTHER HAND, IF IT IS DESIRED TO ALLOW THE SCHEDULED JOBS TO BE PROCESSED WITHOUT WAITING FOR AN ADDITIONAL PROGRAM TO BE INSERTED AHEAD OF THEM THE INVENTORY PROGRAM FINDS ITSELF WAITING IN A LOAD CONTROL DECK ON DISK TO BE EXECUTED AFTER WHAT MAY BE A CONSIDERABLE WAIT.

WHILE MEASURES ARE BEING TAKEN TO ALLOW THE PROGRAM TO BE RUN ON SYSTEM 1 IT MAY WELL BE THE CASE THAT SYSTEM 2 IS AVAILABLE. ANOTHER ALTERNATIVE WOULD THEREFORE BE TO DUMP THE INVENTORY FILE AND THE OBJECT PROGRAM TO TAPE AND RELOAD THEM ON SYSTEM 2. BESIDES BEING TIME-CONSUMING THIS APPROACH HAS THE DISADVANTAGE THAT THE SYSTEM 1 VERSION OF THE FILE STILL WILL NOT BE UPDATED.

ANOTHER PROBLEM WHICH MAY PRESENT ITSELF IS THAT OF A HARDWARE

FAILURE ON SYSTEM 1. IN THIS CASE IT WILL MOST LIKELY BE IMPOSSIBLE TO DUMP THE INVENTORY FILE TO TAPE AND HENCE THE PROGRAM CANNOT BE EXECUTED UNTIL SYSTEM 1 IS AGAIN RUNNING.

THE B-5500 SHAREDISK SYSTEM IS INTENDED TO ELIMINATE THE TYPE OF PROBLEMS JUST DISCUSSED. IT WILL ALLOW UP TO FOUR B5500S TO SHARE THE SAME DISK FILES SIMULTANEOUSLY, WITH THE RESULT THAT OBJECT PROGRAMS REQUIRING THE USE OF DISK DATA FILES MAY BE EXECUTED ON ANY AVAILABLE SYSTEM WITHOUT THE NEED FOR MANUALLY TRANSFERRING THE FILES TO THAT SYSTEM. IF THE INSTALLATION IN THE ABOVE EXAMPLE WAS PROVIDED WITH SHAREDISK AND IT WAS AGAIN FOUND THAT SYSTEM 1 WAS FOR ANY REASON UNAVAILABLE, IT WOULD BE POSSIBLE TO IMMEDIATELY TRY SYSTEM 2 AND IF SYSTEM 2 WAS AVAILABLE THE JOB COULD BE STARTED THERE. IF THE PROGRAM WAS PLACED IN A CONTROL DECK ON SYSTEM 1 AND SYSTEM 2 BECAME AVAILABLE BEFORE THE PROGRAM WAS EXECUTED , IT WOULD BE POSSIBLE TO EXECUTE THE JOB ON SYSTEM 2 BY MERELY ENTERING AN APPROPRIATE MESSAGE AT THE SYSTEM 2 SUPERVISORY PRINTER. AN OPTION HAS ALSO BEEN PROVIDED WHICH AUTOMATICALLY PROVIDES FOR EXECUTION OF SOME TYPES OF JOBS WHEN AT LEAST ONE SYSTEM IS ON TIME SHARING AND AT LEAST ONE SYSTEM IS ON "BATCH". ALSO, SINCE ANY PRINTER BACK UP DISK MAY BE PRINTED ON ANY (OR ALL) SYSTEMS, LINE PRINTER OUTPUT FROM THE PROGRAM COULD BE PRINTED ON A SYSTEM 1 LINE PRINTER, OR IF THE SYSTEM 1 PRINTERS WERE BUSY THE OUTPUT COULD BE TRANSFERRED TO SYSTEM 2. THESE FEATURES WILL BE MORE FULLY EXPLAINED IN SECTION III.

THE SHAREDISK SYSTEM ALSO PROVIDES THE ADVANTAGE THAT THE SCHEDULING OF MACHINE TIME FOR VARIOUS ACTIVITIES IS MADE SOMEWHAT EASIER IN THAT THE ACTUAL B5500 SYSTEM TO BE USED FOR A PARTICULAR JOB IS NOT INFLUENCED BY THE LOCATION OF ASSOCIATED DATA FILES ON DISK.

A DEVICE KNOWN AS A FILE PROTECT MEMORY (FPM) HAS MADE THE DEVELOPMENT OF SHAREDISK POSSIBLE BY PROVIDING A MEANS BY WHICH SYSTEMS COULD RESTRICT ACCESS TO VARIOUS AREAS ON DISK. RESTRICTION IS ACHIEVED THROUGH USE OF THE "READ/LOCK" DISK OPERATOR WHICH WILL CAUSE AN ENTRY TO BE MADE IN THE FPM AND WILL EITHER RETURN INFORMATION OR AN INDICATION THAT THE AREA REQUESTED WAS IN USE. WHEN THE AREA IS RELEASED, USUALLY THROUGH USE OF THE "WRITE/UNLOCK" DISK OPERATOR, ALL SYSTEMS THAT HAD REQUESTED THAT ADDRESS WHILE IT WAS LOCKED ARE NOTIFIED BY THE HARDWARE THAT A REQUESTED ADDRESS HAS BEEN FREED. NOTIFICATION BY THE HARDWARE REDUCES THE AMOUNT OF MCP "OVERHEAD" REQUIRED TO RETRY "LOCKED" ADDRESSES. THE FPM ALSO ELIMINATES THE PROBLEM CAUSED BY TWO DISK FILE CONTROLS ATTEMPTING TO ACCESS THE SAME ELECTRONICS UNIT.

ONLY THE MCP IS ABLE TO INITIATE DISK OPERATIONS WHICH UTILIZE THE FPM. USER PROGRAMS ARE ABLE TO ACHIEVE SOME EXPLICIT CONTROL OF FILES THROUGH USE OF THE "FILE ATTRIBUTE" FACILITIES PROVIDED IN THE ALGOL, TSPOL, AND XALGOL COMPILERS. "FILE ATTRIBUTES" ALLOW USER PROGRAMS TO RESTRICT ACCESS TO DISK FILES OPENED BY THE PROGRAM AS LONG AS THE FILES REMAIN OPEN. THE FACILITIES ARE AVAILABLE ON BOTH

- SHAREDISK INFORMATION -

SHAREDISK AND NON-SHAREDISK SYSTEMS.

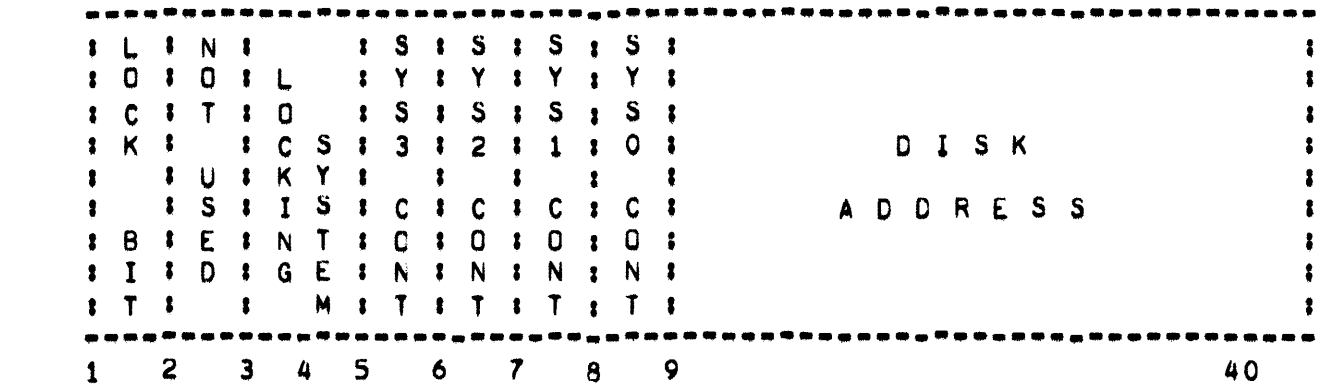
ON SHAREDISK SYSTEMS, WHEN A PROGRAM ATTEMPTS TO OPEN A FILE THE MCP EXAMINES THE OPEN COUNT OF THE DISK FILE HEADER WHICH NOW CONTAINS INFORMATION ON ACCESS RESTRICTION AND USE OF THE FILE. THE MCP WILL READ/LOCK THE HEADER TO PRECLUDE ANOTHER SYSTEM FROM ALTERING IT DURING EXAMINATION. THE OPEN COUNT DETERMINES IF THE REQUESTED ACCESS IS PERMITTED. IF PERMITTED THE OPEN COUNT WILL BE CHANGED, THE HEADER WILL BE RELEASED, AND THE PROGRAM WILL CONTINUE PROCESSING. IF ACCESS IS INHIBITED THE HEADER IS RELEASED AND THE PROGRAM EITHER WAITS ON FILE AVAILIBILITY OR TAKES AN ACTION LABEL BRANCH.

- SHAREDISK INFORMATION -

FPM - CONTENTS OF MEMORY WORD

THE FPM MEMORY CONSISTS OF 40 BIT WORDS, EACH OF WHICH IS INDIVIDUALLY ADDRESSABLE AND FIELD CHANGEABLE. THE CONTENTS OF EACH WORD IS AS FOLLOWS:

FPM MEMORY WORD



BIT 1 — LOCK BIT (1 IF LOCKED)

BIT 2 — FPM CODE (NOT DEFINED)

BITS 3-4 — LOCKING SYSTEM

11 = SYSTEM 3
10 = SYSTEM 2
01 = SYSTEM 1
00 = SYSTEM 0

BIT 5 — SYSTEM 3 CONTENTION BIT

BIT 6 — SYSTEM 2 CONTENTION BIT

BIT 7 — SYSTEM 1 CONTENTION BIT

BIT 8 — SYSTEM 0 CONTENTION BIT

BITS 9-40 — DISK ADDRESS

[9:4] = ALL BITS ON (DFC FUNCTION)
[13:4] = EU NUMBER
[17:8] = DISK NUMBER
[25:8] = TRACK NUMBER
[33:8] = SEGMENT NUMBER

DISK I/O OPERATORS

THE HARDWARE MODIFICATIONS NECESSARY TO IMPLEMENT SHAREDISK INVOLVED EXTENSIVE MODIFICATION OF THE DISK FILE CONTROL UNIT, MODIFICATION OF THE I/O CHANNELS, THE ADDITION OF THE FILE PROTECT MEMORY UNIT, AND THE CREATION OF A NEW SERIES OF DISK OPERATORS. ADDITIONAL BITS IN THE DISK DESCRIPTOR AND THE ZONE BITS IN THE DISK ADDRESS ARE USED TO DENOTE THE REQUIRED OPERATION.

THREE TYPES OF DISK OPERATIONS ARE POSSIBLE. THE NORMAL "READ" AND "WRITE" DO NOT ACCESS THE FPM UNIT. THE "READ/LOCK" AND "WRITE/UNLOCK" OPERATIONS CAUSE ACCESS TO BE MADE BOTH TO DISK AND THE FPM, AND THE FINAL TYPES OF OPERATIONS ACCESS THE FPM UNIT ONLY.

THE DISK FILE CONTROL UNIT (DFC) HAS BEEN MODIFIED SO THAT IT PROVIDES "CHANNELS" OF COMMUNICATION BETWEEN A SYSTEM, DISK, AND THE FPM. IT IS NOT POSSIBLE TO ENTER A FPM COMMAND DIRECTLY IN THE DFC, THE DESIRED OPERATION MUST BE ENTERED THROUGH THE I/O CHANNEL OF A B-5700. THE FOLLOWING EXPLANATIONS, THEREFORE, ASSUME A LOGICAL FLOW FROM THE I/O CHANNEL TO THE DFC THEN TO DISK AND/OR THE FPM.

THE I/O OPERATION IS INITIATED NORMALLY. ON RECEIPT OF INFORMATION FROM THE I/O CHANNEL, THE DFC INTERROGATES THE I/O DESCRIPTOR AND THE ZONE BITS OF THE DISK ADDRESS TO DETERMINE IF A FPM OPERATION IS REQUIRED. IF NO FPM OPERATION IS SPECIFIED, THE ACCESS TO DISK IS MADE, AND THE OPERATION TERMINATES NORMALLY.

EIGHT BITS ARE TRANSFERRED IN PARALLEL BETWEEN THE FPM AND DFC. THE TIME NECESSARY FOR THIS OPERATION IS ONE CLOCK. IN A NORMAL DFC TO FPM OPERATION A CONTROL CHARACTER IS TRANSMITTED TO THE FPM ON THE FIRST CLOCK AND, DEPENDENT ON THE OPERATION BEING PERFORMED, FOUR ADDRESS CHARACTERS MAY BE TRANSMITTED IN THE NEXT CONSECUTIVE FOUR CLOCKS. THE CONTROL CHARACTER IS PLACED IN THE VARIANT AND N REGISTERS, AND THE ADDRESS IS PLACED IN THE ADDRESS REGISTERS. THE FPM GENERATES THE LOCKING AND CONTENTION CODES BASED ON THE CONTENTS OF THE V AND N REGISTERS AND COMPLETES THE SPECIFIED OPERATION.

NO ADDITIONAL I/O TIME IS REQUIRED WHEN A FPM OPERATION IS PERFORMED IN CONJUNCTION WITH THE NORMAL DISK I/O OPERATION WITH THE EXCEPTION OF WRITE/UNLOCK AND FPM ONLY OPERATIONS. IN THE CASE OF THE WRITE/UNLOCK INSTRUCTION, THE AVERAGE ADDITIONAL TIME REQUIRED IS ANTICIPATED TO BE 70 TO 80 MICROSECONDS AND IS DEPENDENT ON SYSTEM CONTENTION FOR THE FPM. THE AVERAGE ACCESS TIME REQUIRED FOR A FPM ONLY TYPE OF OPERATION, IN WHICH NO ACCESS IS MADE TO DISK, IS ANTICIPATED TO BE 60 MICROSECONDS, WHICH IS AGAIN DEPENDENT ON SYSTEM CONTENTION FOR THE FPM.

THERE ARE CERTAIN FPM OPERATIONS WHICH REQUIRE ONLY THE CONTROL CHARACTER TO BE TRANSFERRED TO THE FPM, AND THE AMOUNT OF TIME REQUIRED FOR THE OPERATION IS CORRESPONDINGLY SHORTER. AN ENTRY IN

THE FPM IS MARKED AS VALID BY EITHER THE LOCK BIT (BIT 1) OR ONE OR MORE OF THE CONTENTION BITS (BITS 5-8) BEING ON. IF NONE OF THE AFOREMENTIONED BITS ARE ON, THE FPM ADDRESS IS CONSIDERED AVAILABLE AND THE PRESENT CONTENTS OF THE WORD ARE UNACCESSIBLE. ALL OPERATIONS INVOLVING ACCESS TO THE FPM MEMORY ACTIVATE A SCANNER WHICH SCANS ALL OF THE FPM MEMORY UNTIL A MATCH, AVAILABLE AREA, OR MEMORY FULL CONDITION OCCURS DEPENDENT ON THE FPM OPERATION BEING PERFORMED. THE SCAN IS ALWAYS FROM THE FIRST TO LAST ADDRESS OF THE FPM MEMORY WHICH PRECLUDES THE POSSIBILITY OF DUPLICATE DISK ADDRESS ENTRIES.

THE DISK OPERATORS POSSIBLE UNDER SHAREDISK ARE:

READ

THIS IS THE NORMAL READ WITHOUT LOCK. THE FPM IS NOT ACCESSED, AND THE DFC PERFORMS A NORMAL DISK READ.

NOTE: IT IS POSSIBLE TO CIRCUMVENT DISK ADDRESS LOCKOUT ESTABLISHED IN THE FPM WITH THIS OPERATION.

READ/LOCK

THIS INSTRUCTION UTILIZES BOTH FPM AND NORMAL DISK ACCESS FUNCTIONS. WHEN THE DFC RECEIVES THE INSTRUCTION, THE REQUEST FOR DISK ACCESS IS MADE. IMMEDIATELY AFTER THIS REQUEST THE DFC WILL INITIATE THE FPM OPERATION. IT IS ANTICIPATED THAT THE TIME REQUIRED TO EXECUTE THE FPM OPERATION WILL NEVER EXCEED THE TIME REQUIRED TO COMPLETE THE ACCESS TO DISK, EVEN UNDER THE CONDITION WHERE FOUR SYSTEMS ARE CONTENDING FOR THE FPM.

IF THE FPM DISCOVERS THAT THE REQUESTED DISK ADDRESS IS LOCKED, IT SENDS A SIGNAL TO THE DFC WHICH IN TURN INHIBITS THE DISK I/O OPERATION AND SENDS A SIGNAL BACK TO THE REQUESTING SYSTEM INDICATING THAT THE DISK ADDRESS WAS LOCKED. THE INHIBITED I/O OPERATION SHOULD RETURN THE RESULT DESCRIPTOR TO THE REQUESTING SYSTEM FASTER THAN THE NORMAL I/O OPERATION WOULD. THE CONTENTION BIT FOR THE REQUESTING SYSTEM IS SET.

IF THE DISK ADDRESS IN THE FPM IS UNLOCKED AND THE CONTENTION BIT FOR THE REQUESTING SYSTEM IS NOT SET AND SPACE IS AVAILABLE IN THE FPM, THE LOCK BIT WILL BE SET, AND THE "LOCKER" CODE WILL BE ENTERED. THE FPM SHOULD RESPOND TO THE DFC BEFORE THE DISK I/O OPERATION GOES TO A NORMAL COMPLETION. IF SPACE IS NOT AVAILABLE IN THE FPM, THE REQUESTING SYSTEM WILL RECEIVE A "FPM FULL" RESULT DESCRIPTOR AND THE OPERATION WILL BE TERMINATED.

IF THE REQUESTED DISK ADDRESS IS NOT LOCKED, AND THE CONTENTION BIT FOR THE REQUESTING SYSTEM IS SET, THE DFC AND FPM OPERATIONS WILL

TERMINATE IN THE SAME MANNER AS IF A LOCKED ADDRESS HAD BEEN ENCOUNTERED IN THE FPM.

THE COMPLETION OF THE DISK I/O OPERATION BEFORE THE TERMINATION OF THE FPM OPERATION WOULD INDICATE A FPM FAILURE AND THE RESULTS ARE UNPREDICTABLE.

WRITE

THIS IS A NORMAL WRITE WITHOUT LOCK. THE FPM IS NOT ACCESSED, AND THE DFC PERFORMS A NORMAL DISK WRITE.

NOTE: IT IS POSSIBLE TO CIRCUMVENT DISK ADDRESS LOCKOUT ESTABLISHED IN THE FPM WITH THIS OPERATION.

WRITE/UNLOCK

THIS INSTRUCTION REQUIRES A TWO PART FPM OPERATION. THE FIRST PART CONSISTS OF A SEARCH BY THE FPM THROUGH ITS MEMORY TO FIND A SPECIFIED DISK ADDRESS. IF THE ADDRESS IS FOUND, THE FPM REMEMBERS THE LOCATION OF IT AND SENDS A SIGNAL TO THE DFC INDICATING COMPLETION. IF THE DISK ADDRESS WAS NOT FOUND, THE FPM SETS AN INDICATOR TO THAT EFFECT AND SENDS A SIGNAL TO THE DFC INDICATING COMPLETION. THIS TERMINATES THE PART ONE OPERATION.

IF THE ADDRESS WAS NOT FOUND, THE "PART TWO" OF THE OPERATION WILL NOT BE PERFORMED. IT IS THUS POSSBLE FOR A WRITE/UNLOCK OPERATION TO BE PERFORMED SUCCESSFULLY WHEN THE SECIFIED DISK ADDRESS IS NOT LOCKED, AND THE REQUESTING SYSTEM WILL HAVE NO INDICATION OF THE POSSIBLE ERROR CONDITION.

PART TWO OF THE FPM OPERATION CONSISTS OF RESETTING THE LOCK AND CONTENTION BITS OF THE REMEMBERED ADDRESS FOR THE REQUESTING SYSTEM.

ON RECEIPT OF THE I/O OPERATION, THE DFC WILL INITIATE PART ONE OF THE FPM OPERATION. THE DFC WILL DELAY INITIATION OF THE DISK ACCESS UNTIL IT HAS RECEIVED A SIGNAL FROM THE FPM INDICATING ITS OPERATION IS COMPLETE. ONCE THE SIGNAL IS RECEIVED, THE DISK ACCESS IS STARTED.

IF THE DFC RECEIVES A RESULT FROM THAT OPERATION INDICATING THE DISK I/O WAS UNSUCCESSFUL, PART TWO OF THE FPM OPERATION IS NOT PERFORMED. THE ADDRESS REMAINS IN ITS ORIGINAL STATE. A SIGNAL IS SENT TO THE REQUESTING SYSTEM INDICATING THAT A WRITE ERROR OCCURRED.

IF THE WRITE WAS SUCCESSFUL, AN APPROPRIATE SIGNAL IS SENT TO THE REQUESTING SYSTEM. IMMEDIATELY AFTER THIS, THE DFC WILL INITIATE PART TWO OF THE FPM OPERATION. THE DFC REMAINS IN A BUSY STATE UNTIL A SIGNAL IS RECEIVED FROM THE FPM INDICATING COMPLETION OF ITS

OPERATION.

NOTE: THE FPM DOES NOT CHECK THE CODE OF THE LOCKER IN THE PART ONE OPERATION. IT IS POSSIBLE FOR ANY SYSTEM TO UNLOCK ANY ADDRESS BY USING THE WRITE/UNLOCK INSTRUCTION.

WHEN A PREVIOUSLY LOCKED DISK ADDRESS IS FREED, THE FPM CHECKS TO SEE IF ANY CONTENTION BITS ARE ON. IF NO CONTENTION BITS ARE ON THAT FPM ADDRESS IS CONSIDERED TO BE AVAILABLE AND THE OPERATION IS TERMINATED. IF ANY OF THE CONTENTION BITS ARE ON, INTERRUPTS ARE SENT TO THE APPROPRIATE SYSTEMS BY THE FPM. AT THE TERMINATION OF THE NEXT FPM OPERATION, IF AN ADDRESS FOR THE SYSTEM PERFORMING THAT OPERATION HAD BEEN FREED AT A PRIOR OPERATION, THE FPM WILL SEND THE ADDRESS TO THE DFC. THE DFC CHECKS TO SEE IF A REPORT FREE ADDRESS INSTRUCTION IS BEING EXECUTED. IF THE INSTRUCTION IS BEING EXECUTED, THE DISK ADDRESS AND RESULT DESCRIPTOR ARE SENT BACK TO THE REQUESTING SYSTEM. IF THE REPORT FREE ADDRESS INSTRUCTION IS NOT BEING EXECUTED, THE ADDRESS SENT BY THE FPM IS IGNORED BY THE DFC.

CLEAR ALL CONTENTION BITS OF SYSTEM "N"

THIS IS A FPM ONLY TYPE OF OPERATION IN WHICH NO ACCESS IS MADE TO DISK. THE DFC TRANSMITS A CONTROL CHARACTER TO THE FPM. IN ONE OPERATION THE FPM WILL SCAN ALL OF ITS MEMORY AND RESET THE CONTENTION BIT FOR THE SPECIFIED SYSTEM IN EACH FPM MEMORY WORD WHERE IT IS ENCOUNTERED.

NOTE: THE REQUESTING SYSTEM CAN SPECIFY THAT THE CONTENTION BITS OF ANOTHER SYSTEM ARE TO BE CLEARED.

UNLOCK ALL ADDRESSES OF SYSTEM "N"

THIS IS A FPM ONLY TYPE OF OPERATION IN WHICH NO ACCESS IS MADE TO DISK. THE DFC INITIATES THE FPM OPERATION AND TRANSMITS A SINGLE CONTROL CHARACTER TO THE FPM. IN ONE OPERATION THE FPM WILL SCAN ALL OF ITS MEMORY AND WILL RESET THE LOCK BIT OF EACH MEMORY WORD WHOSE LOCK BITS ARE EQUAL TO THOSE SPECIFIED IN THE CONTROL CHARACTER SENT BY THE DFC. THE DFC OBTAINED THE SYSTEM NUMBER FROM THE DISK ADDRESS FIELD OF THE REQUESTING SYSTEM. THE REQUESTING SYSTEM CAN SPECIFY THAT THE LOCK BITS OF ANY SYSTEM ARE TO BE RESET. AT THE COMPLETION OF THE OPERATION A SINGLE INTERRUPT IS SENT TO EACH SYSTEM WHICH HAD A CONTENTION BIT SET WHEN THE ADDRESS WAS FREED.

NOTE: ONLY ONE INTERRUPT WILL BE SENT TO EACH SYSTEM REGARDLESS OF THE NUMBER OF CONTENTION BITS THE SYSTEM HAD SET. THE FPM WILL, IN TURN, TRANSMIT THE READ CHECK INTERRUPT APPLICABLE SYSTEMS.

CLEAR CONTENTION BIT OF SYSTEM "N" IN ADDRESS "X"

THIS IS A FPM ONLY TYPE OF OPERATION IN WHICH NO ACCESS IS MADE TO DISK. THE DFC INITIATES THE FPM OPERATION AND TRANSMITS A CONTROL CHARACTER AND FOUR ADDRESS CHARACTERS TO THE FPM. THE FPM WILL SCAN ITS MEMORY UNTIL IT EITHER GETS A MATCH ON DISK ADDRESS OR COMPLETES THE SCAN CYCLE. IF A MATCH IS MADE, THE CONTENTION BIT FOR THE SPECIFIED SYSTEM IS RESET. THE REQUESTING SYSTEM CAN SPECIFY THAT THE CONTENTION BIT OF ANY SYSTEM WITH ANY DISK ADDRESS IS TO BE RESET. THE DFC WILL SEND THE NORMAL COMPLETION SIGNAL TO THE REQUESTING SYSTEM.

UNLOCK ADDRESS "X"

THIS IS A FPM ONLY TYPE OF OPERATION IN WHICH NO ACCESS IS MADE TO DISK. THE DFC INITIATES THE FPM OPERATION AND TRANSMITS A SINGLE CONTROL CHARACTER TO THE FPM. THE FPM SCANS ITS MEMORY UNTIL EITHER A MATCH ON DISK ADDRESS OCCURRS OR THE SCAN CYCLE GOES TO COMPLETION. IF A MATCH IS MADE, THE LOCK BIT IS RESET REGARDLESS OF THE SYSTEM WHICH ORIGINALLY LOCKED THE ADDRESS.

IF ONE OR MORE CONTENTION BITS ARE SET FOR A SYSTEM, THAT SYSTEM WILL RECEIVE ONE READ CHECK INTERRUPT (CCI15F OR CCI16F) REGARDLESS OF THE NUMBER OF INDIVIDUAL CONTENTION BITS SET.

REPORT FREE ADDRESS (1)

THIS IS A FPM ONLY TYPE OF OPERATION IN WHICH NO ACCESS IS MADE TO DISK. THE REPORT FREE ADDRESS INSTRUCTION HAS BEEN INITIATED BECAUSE THE REQUESTING SYSTEM HAS RECEIVED A READ CHECK INTERRUPT WHICH INDICATES THAT A PREVIOUSLY CONTENDED ADDRESS IS NOW AVAILABLE. IF THE REQUESTING SYSTEM INITIATES A REPORT FREE ADDRESS (1), THE DFC WILL SEND A COMMAND TO THE FPM INITIATING THE FPM OPERATION. THE FPM WILL SCAN ITS MEMORY FOR AN UNLOCKED ADDRESS WHICH HAS A CONTENTION BIT SET FOR THE REQUESTING SYSTEM. EITHER THE FIRST FREED ADDRESS ENCOUNTERED OR A SIGNAL INDICATING THAT NO FREE ADDRESS IS AVAILABLE FOR THE REQUESTING SYSTEM IS SENT TO THE DFC. THE DFC WILL THEN TRANSMIT THE RESULT TO THE REQUESTING SYSTEM. IF AN ADDRESS IS RETURNED, THE CONTENTION BIT FOR THE REQUESTING SYSTEM WILL NOT BE RESET.

THE DISK ADDRESS WILL BE RETURNED AT THE CORE ADDRESS POINTED TO BY THE ORIGINAL DISK I/O DESCRIPTOR + 1. BIT NO. 42 OF THE RESULTING DISK ADDRESS MUST BE TESTED TO DETERMINE IF A FREE ADDRESS WAS AVAILABLE. THIS IS NECESSARY BECAUSE THE FPM HANDLES REQUESTS ON A FIRST COME FIRST SERVED BASIS AND THE ADDRESS COULD HAVE BEEN LOCKED BY A SYSTEM WHICH WAS ABLE TO RESPOND FASTER. IF THE BIT IS ON, THE

ADDRESS IS VALID. BITS 4 AND 5 OF THE RETURNED DISK ADDRESS WILL CONTAIN THE SYSTEM NUMBER.

REPORT FREE ADDRESS (2)

THIS IS A FPM ONLY TYPE OF OPERATION IN WHICH NO ACCESS IS MADE TO DISK. THE REPORT FREE ADDRESS INSTRUCTION HAS BEEN INITIATED BECAUSE THE REQUESTING SYSTEM HAS RECEIVED A READ CHECK INTERRUPT WHICH INDICATES THAT A PREVIOUSLY CONTENDED ADDRESS IS NOW AVAILABLE. IF THE REQUESTING SYSTEM INITIATES A REPORT FREE ADDRESS (2), THE DFC WILL SEND A COMMAND TO THE FPM INITIATING THE FPM OPERATION. THE FPM WILL SCAN ITS MEMORY FOR AN UNLOCKED ADDRESS WHICH HAS A CONTENTION BIT SET FOR THE REQUESTING SYSTEM. EITHER THE FIRST FREED ADDRESS ENCOUNTERED OR A SIGNAL INDICATING THAT NO FREE ADDRESS IS AVAILABLE FOR THE REQUESTING SYSTEM IS SENT TO THE DFC. THE DFC WILL, IN TURN, TRANSMIT THE RESULT TO THE REQUESTING SYSTEM, AND THE CONTENTION BIT FOR THAT SYSTEM IS RESET.

THE DISK ADDRESS WILL BE RETURNED AT THE CORE ADDRESS POINTED TO BY THE ORIGINAL DISK I/O DESCRIPTOR + 1. BIT NO. 42 OF THE RESULTING DISK ADDRESS MUST BE TESTED TO DETERMINE IF A FREE ADDRESS WAS AVAILABLE. THIS IS NECESSARY BECAUSE THE FPM HANDLES REQUESTS ON A FIRST COME FIRST SERVED BASIS AND THE ADDRESS COULD HAVE BEEN LOCKED BY A SYSTEM WHICH WAS ABLE TO RESPOND FASTER. IF THE BIT IS ON, THE ADDRESS IS VALID. THE SYSTEM NUMBER (BITS 4 & 5) IS NOT RETURNED IN THE DISK ADDRESS FIELD.

OPERATION OF THE SHAREDISK SYSTEM

OPERATIONALLY THE SHAREDISK SYSTEM IS VERY SIMILAR TO THE PRESENT BATCH SYSTEM. IF A SYSTEM WITHIN THE NETWORK IS HALT/LOADED, IT FIRST DETERMINES IF THERE ARE OTHER SYSTEMS ALSO IN OPERATION. IF SO, THE HALT/LOADED SYSTEM CLOSES ALL OF THE DISK FILES THAT IT HAD OPEN AND RETURNS ALL OF THE USER DISK THAT IT HAD IN USE (WITHOUT DISTURBING THE OTHER SYSTEMS). IF ON THE OTHER HAND THE HALT/LOADED SYSTEM WAS THE ONLY ONE PRESENTLY RUNNING IT WILL TAKE THE SAME ACTION THAT A NON-SHARED DISK SYSTEM DOES AT HALT/LOAD (REBUILDS THE DISK DIRECTORY, ETC.). IF A SYSTEM CEASES TO FUNCTION AND CANNOT BE RESTARTED DO TO A HARDWARE FAILURE, THE MESSAGE "CL(SYSTEM MNEMONIC)" SHOULD BE TYPED INTO A FUNCTIONING SYSTEM. THIS WILL CAUSE ALL ADDRESSES IN THE FILE PROTECT MEMORY THAT WERE LOCKED BY THE DISABLED SYSTEM TO BE UNLOCKED, WILL CLOSE ALL OF THE FILES THAT IT HAD OPENED AND WILL RETURN ALL OF THE USER DISK THAT IT HAD IN USE. AFTER THIS PROCESS IS COMPLETED THE MESSAGE "#SYSTEM N CLEARED" WILL BE TYPED.

NOTE: IT IS RECOMMENDED THAT THE "CL" MESSAGE BE USED WHEN A SYSTEM CEASES TO FUNCTION FOR ANY REASON. IT HAS BEEN FOUND THAT THE PERFORMANCE OF THE OTHER SYSTEMS WITHIN THE NETWORK MAY BE SERIOUSLY REDUCED WHILE A HALT/LOAD IS BEING PERFORMED, UNLESS THE DISABLED SYSTEM IS FIRST CLEARED VIA THE "CL" MESSAGE.

OTHER CHANGES IN SYSTEM OPERATION ARE:

MCP

THE SYSTEMS MAY ALL USE THE SAME MCP FILE, OR THEY MAY ALL USE SEPARATE MCP FILES, OR ANY COMBINATION THEREOF. SHAREDISK CAPIBILITY IS PROVIDED FOR BOTH STANDARD AND TIME SHARING MCP"S AND THE SYSTEMS MAY ALL USE STANDARD, OR THEY MAY ALL USE TIME SHARING, OR ANY COMBINATION THEREOF.

INTRINSICS

THE SYSTEMS MAY ALL USE THE SAME INTRINSIC FILE, OR THEY MAY ALL USE SEPARATE INTRINSIC FILES, OR ANY COMBINATION THEREOF.

PRINTER BACKUP DISK

ANY PRINTER BACKUP DISK FILE MAY BE PRINTED ON ANY (OR ALL) SYSTEMS. IF THE FILE IS BEING PRINTED SIMULTANEOUSLY ON TWO OR MORE SYSTEMS (NOT CONSIDERED A NORMAL ACTIVITY), THE LAST SYSTEM TO FINISH PRINTING IT WILL REMOVE IT FROM THE DISK DIRECTORY. THE PB COMMAND

REMAINS UNCHANGED.

LOAD CONTROL

ALL CONTROL DECKS ARE IDENTIFIED WITH THE SYSTEM WHICH LOADED THEM AND IF NO SPECIAL ACTION IS TAKEN, THEY WILL RUN ON THAT SYSTEM. A NEW VARIATION OF THE "RN" KEYBOARD MESSAGE HAS BEEN IMPLEMENTED: "RN #NNNN" WHERE NNNN IS THE NUMBER OF THE CONTROL DECK ON DISK. PROVIDING THAT THE SPECIFIED DECK IS NOT ALREADY IN USE, THE TYPING IN OF THIS MESSAGE WILL PLACE THE DECK IN A PSEUDOREADER REGARDLESS OF WHICH SYSTEM LOADED IT. IF THE DECK IS IN USE, THE MESSAGE "DECK #NNNN IN USE BY SYSTEM N" WILL BE TYPED.

IF THE KEYBOARD MESSAGE "RD*" IS ENTERED, ONLY THOSE DECKS THAT WERE LOADED ON THE SYSTEM INTO WHICH THE MESSAGE IS ENTERED WILL BE REMOVED. IF THE MESSAGE "RD#NNNN" IS ENTERED, THE SPECIFIED DECK WILL BE REMOVED REGARDLESS OF WHICH SYSTEM LOADED IT, PROVIDING THAT THE DECK IS NOT IN USE.

THE "CD" MESSAGE HAS BEEN MODIFIED SO THAT IT IS POSSIBLE TO SPECIFY THE SYSTEM WHOSE DECKS ARE TO BE PRINTED. THE NEW MESSAGES ARE AS FOLLOWS: CDSYA, CDSYB, CDSYC, CDSYD, CDALL. IF "CD" ALONE IS TYPED, THE SYSTEM INTO WHICH THE MESSAGE IS ENTERED IS IMPLIED.

THE OPTION "RNALL" ALLOWS AUTOMATIC CROSS-SCHEDULING OF CONTROL DECKS BETWEEN TWO OR MORE SHAREDISK SYSTEMS. TWO-WAY OR MUTUAL CROSS-SCHEDULING MAY BE DONE WHEN RUNNING SYSTEMS IN EITHER BATCH OR TIMESHARING MODE.

IF ONE OF THE SYSTEMS IS RUNNING TIMESHARING, THE CONTROL DECKS ENTERED FROM IT WILL BE RUN IN BATCH MODE, ON THE OTHER SYSTEM(S) CURRENTLY SHARING THE SAME DISK IF THIS NEW OPTION IS SET ON THE BATCH SYSTEM AND THE TIMESHARING SYSTEM IS PREVENTED FROM RUNNING CONTROL DECKS BY RESETTING "BATCHTOG", AND/OR TURNING OFF ALL PSEUDO-READERS, OR VICE VERSA.

NOTE: ANY TSPOL PROGRAM THAT DOES A NEGATIVE COMMUNICATE AND IS RUN ON THE BATCH SYSTEM, WILL BE TERMINATED WITH AN "INVALID COM". THEREFORE, SUCH PROGRAMS SHOULD NOT BE ENTERED AS A CONTROL DECK FOR EXECUTION IF THE TIMESHARING SYSTEM IS SHARING DISK WITH A BATCH SYSTEM WHICH HAS "RNALL" CURRENTLY SET.

SYSTEM LOG

THERE IS ONE SYSTEM LOG FOR ALL SYSTEMS. THE SYSTEM ID, (0,1,2,3) IS PLACED IN THE [1:2] FIELD OF THE CODE WORD IN EACH LOG ENTRY.

COLD START

THE "COLD START" DECK IS GENERATED WITH "$ COOL" SET FALSE IN THE ESPOL "COMPILE" DECK. THE "COLD START" PROGRAM IN ADDITION TO BUILDING THE DISK DIRECTORY, BUILDS THE SKELETON OF DISK SEGMENT ZERO AND INSERTS THE NUMBER OF SYSTEMS AND ADDRESS OF DIRECTORYTOP WHILE ZEROING OUT THE REST OF THE SEGMENT. TO SPECIFY THE NUMBER OF SYSTEMS, A NEW PARAMETER CARD "SYSTEMS = (PHYSICAL NUMBER OF SYSTEMS)" IS REQUIRED FOR THE "COLD START" DECK. THERE IS ALSO A REQUIREMENT FOR A DUMMY FILE CARD FOR THE MCP WHICH WILL BE LOADED FROM TAPE. (E.G. "FILE MCP/DISK,1×1170,999).

TAPE TO DISK LOADER

THE "TAPE TO DISK" DECK LOADS THE "KERNEL" AND ALSO ALLOWS FOR CONTROL CARDS TO CHANGE THE DEFAULT CASES. "FILE =" WILL CHANGE THE MCP FILE DEFAULT CASE FROM "MCP/DISK" TO ANY DESIRED FILE ID. "TAPE =" WILL CHANGE THE TAPE DEFAULT CASE FROM "SYSTEM"

I.E.	FILE = MCP/TEST	THE TAPE LOADER WILL LOAD A FILE CALLED MCP/TEST FROM THE TAPE TO A DISK FILE CALLED MCP/TEST.
	TAPE = TESTER	THE LOADER WOULD LOOK FOR A LIBRARY TAPE CALLED "TESTER".

IT IS IMPORTANT TO STRESS THAT THE DUMMY MCP FILE SET UP AT "COLD START" TIME MUST HAVE THE SAME NAME AS THE FILE TO BE LOADED WITH THE "TAPE TO DISK" LOADER.

AFTER LOADING THE MCP, THE DISK HEADER IN THE DIRECTORY IS UPDATED TO THE ACTUAL MCP SIZE AND SEGMENT ZERO IS UPDATED TO SHOW THE NAME AND ADDRESS OF THE MCP THAT WAS LOADED. A HALT/LOAD MAY BE DONE AT THIS TIME. THE OTHER SYSTEM(S) MAY BE BROUGHT UP BY USE OF THE "DISK TO DISK" LOADER. THE "COLD START", "TAPE TO DISK" AND "COOL START" DECKS MAY NOT BE USED IF AN MCP IS RUNNING ON ANY OF THE SYSTEMS ATTACHED TO THE FPM. "DISK TO DISK" AND "KERNEL" DECKS ON THE OTHER HAND MAY BE RUN WHILE AN MCP IS OPERATING ON ONE OR MORE OF THE SYSTEMS.

DISK TO DISK

THE "DISK TO DISK" LOADER TAKES THE MCP NAME FROM THE CARD AFTER THE "KERNEL" AND UPDATES SEGMENT ZERO WITH ITS NAME AND ADDRESS. THE CARD IS FREE FORM (I.E. MCP/DISK OR MCP DISK). NOTE THAT THE "DISK TO DISK" MUST BE DONE ON EACH SYSTEM AND THAT THE LOADER DOES NOT HAVE TO HAVE THE "KERNEL" PRESENT IF A GOOD ONE IS ON THE DISK.

THE HALT LOAD KERNEL

THE "KERNEL" IS THE PROGRAM THAT IS USED TO BRING THE MCP IN AND CLEARS THE FPM FOR THE SYSTEM. THE "KERNEL" MAY BE LOADED BY ANY OF THE OTHER LOADERS BY PLACING IT AFTER THE ESPOL DECK AND BEFORE THE PARAMETER CARDS, IF ANY. IT MAY ALSO BE RUN FROM THE CARD READER.

THE COOL START DECK

THE "COOL START" DECK IS NOW GENERATED BY SETTING "$ COOL" TO "TRUE" IN THE ESPOL "COMPILE" DECK. THE "COOL START" PROGRAM CHECKS TO SEE IF IT MIGHT REMOVE THE MCP. IF SO IT PRINTS A MESSAGE TO SAY THAT A "TAPE TO DISK" SHOULD BE DONE TO RELOAD THE MCP. IT ALSO REMOVES THE INTRINSICS FOR THE SYSTEM THAT THE "COOL START" IS RUN ON. WHEN IT FINISHES CHECKING THE DIRECTORY, THE OPERATOR MAY RUN THE "TAPE TO DISK" TO RELOAD THE MCP.

FPM - MAINTENANCE KIT INTERPRETATION

THE LABELS PROVIDED WITH THE FPM UNIT MUST BE ATTACHED TO THE APPROPRIATE POSITIONS ON THE MAINTENANCE BLOCKS.

THERE ARE SEVEN MAINTENANCE BLOCKS ON THE FPM. THEY CONSIST OF A SWITCH BLOCK AND SIX INDICATOR BLOCKS. EACH INDICATOR ALSO SERVES AS A SWITCH. THE INDICATORS DO NOT AUTOMATICALLY RESET, SO THE USER MUST CLEAR THE AREA HE WISHES TO UTILIZE BEFORE ATTEMPTING TO EXECUTE AN OPERATION.

THE FOLLOWING IS A BRIEF DESCRIPTION OF THE SWITCHES AND INDICATORS AND THEIR USAGE:

LOCAL - REMOTE

THE NORMAL OPERATING POSITION FOR THIS SWITCH IS "REMOTE". WHEN PLACED IN "LOCAL" IT ENABLES THE OPERATOR TO ENTER INFORMATION THROUGH THE SWITCHES. THE SYSTEMS ATTACHED TO THE FPM COMMUNICATE WITH IT REGARDLESS OF THE POSITION OF THIS SWITCH. THE USER MUST RESTRICT HIS ENTRIES INTO THE FPM SO THAT THEY WILL NOT CAUSE DISRUPTION OF SYSTEM OPERATION.

WHEN IN REMOTE ALL OF THE FOLLOWING SWITCHES ARE DISABLED.

SINGLE PULSE - RUN

THE NORMAL POSITION OF THIS SWITCH IS "RUN". WHEN PLACED IN THE "SINGLE PULSE" POSITION IT WILL INHIBIT NORMAL OPERATION BETWEEN THE DFC AND FPM WILL ENABLE THE OPERATOR TO ISSUE SINGLE PULSES.

START

DEPRESSION OF THE START SWITCH CAUSES ONE CLOCK PULSE TO BE ISSUED.

CLEAR

DEPRESSION OF THIS SWITCH WILL CAUSE THE INDICATOR FLIP FLOPS TO BE CLEARED. THE FPM MEMORY IS NOT AFFECTED.

SETWRDS - SCANSW - NORMAL

THE NORMAL OPERATING POSITION OF THIS SWITCH IS "NORMAL".

IF THIS SWITCH IS IN THE SETWRDS POSITION IT WILL CAUSE THE CONTENTS OF THE ADDRESS REGISTERS (ADR1, ADR2, ADR3, ADR4, AND ADR5) TO BE PLACED IN THE FPM MEMORY LOCATION SPECIFIED BY THE WDC REGISTER IF THE V AND N REGISTERS ARE ZEROED AND THE LOCAL-REMOTE SWITCH IS IN THE LOCAL POSITION.

IF THIS SWITCH IS IN THE SCANSW POSITION IT WILL ALLOW REPETITIVE CYCLING OF AN OPERATION IN THE V AND N REGISTERS.

C00LMRS - NORMAL

NORMAL OPERATING POSITION FOR THIS SWITCH IS THE "NORMAL" POSITION. IF THE SWITCH IS PLACED IN THE "C00LMRS" POSITION IT WILL NOT INHIBIT TRANSFER BETWEEN THE DFC PLUGGED IN AT POSITION 0 AND THE FPM UNIT, AND IS A MEANS OF LOCALLY ENABLING THE REQUEST FOR SERVICE LINE FOR THE DFC. THE POSITION NUMBER REFERS TO THE LOGICAL CABLE POSITION ASSUMING THAT IT IS POSSIBLE TO ATTACH FROM ONE TO FOUR DFC-S TO THE FPM.

C01LMRS - NORMAL

NORMAL OPERATING POSITION FOR THIS SWITCH IS THE "NORMAL" POSITION. IF THE SWITCH IS PLACED IN THE "C01LMRS" POSITION IT WILL NOT INHIBIT TRANSFER BETWEEN THE DFC PLUGGED IN AT POSITION 1 AND THE FPM UNIT, AND IS A MEANS OF LOCALLY ENABLING THE REQUEST FOR SERVICE LINE FROM THE DFC. THE POSITION NUMBER REFERS TO THE LOGICAL CABLE POSITION ASSUMING THAT IT IS POSSIBLE TO ATTACH FROM ONE TO FOUR DFC-S TO THE FPM.

C02LMRS - NORMAL

NORMAL OPERATING POSITION FOR THIS SWITCH IS THE "NORMAL" POSITION. IF THE SWITCH IS PLACED IN THE "C02LMRS" POSITION IT WILL NOT INHIBIT TRANSFER BETWEEN THE DFC PLUGGED IN AT POSITION 2 AND THE FPM UNIT, AND IS A MEANS OF LOCALLY ENABLING THE REQUEST FOR SERVICE LINE FROM THE DFC. THE POSITION NUMBER REFERS TO THE LOGICAL CABLE POSITION ASSUMING THAT IT IS POSSIBLE TO ATTACH FROM ONE TO FOUR DFC-S TO THE FPM.

C03LMRS - NORMAL

NORMAL OPERATING POSITION FOR THIS SWITCH IS THE "NORMAL" POSITION. IF THE SWITCH IS PLACED IN THE "C03LMRS" POSITION IT WILL NOT INHIBIT TRANSFER BETWEEN THE DFC PLUGGED IN AT POSITION 3 AND THE FPM UNIT, AND IS A MEANS OF LOCALLY ENABLING THE REQUEST FOR SERVICE LINE FROM THE DFC. THE POSITION NUMBER REFERS TO THE LOGICAL CABLE

POSITION ASSUMING THAT IT IS POSSIBLE TO ATTACH FROM ONE TO FOUR DFC-S TO THE FPM.

V (VARIANT REGISTER)

THE INSTRUCTION TRANSMITTED FROM THE DFC ON THE FIRST DATA TRANSFER CLOCK PULSE IS PLACED IN THE V AND N REGISTERS. THE V REGISTER CONTAINS THE COMMAND TO BE EXECUTED. A DESCRIPTION OF THE POSSIBLE COMMANDS APPEARS UNDER THE DISCUSSION OF THE N REGISTER.

N REGISTER

THE INSTRUCTION TRANSMITTED FROM THE DFC IS PLACED IN THE V AND N REGISTERS. THE N REGISTER CONTAINS THE SYSTEM NUMBER WHICH IS DECODED AS FOLLOWS:

N(8) = SYSTEM 3
N(4) = SYSTEM 2
N(2) = SYSTEM 1
N(1) = SYSTEM 0

THE INSTRUCTIONS FOR BOTH THE V AND N REGISTERS ARE AS FOLLOWS:

V 8421	N 8421	
0010	0000	SCAN FOR THE FOLLOWING ADDRESS (USED IN REPORT FREE ADDRESS (1)).
0100	NNNN	SCAN FOR AN UNLOCKED AND CONTENDED FOR ADDRESS IN THIS SYSTEM. IF FOUND, REMOVE CONTENTION BIT AND SEND ADDRESS TO CONTROL. (USED FOR REPORT FREE ADDRESS (2)).
0011	NNNN	CLEAR THE CONTENTION BIT FOR THE GIVEN SYSTEM(S) IN THE FOLLOWING ADDRESSES
0101	0000	LOCK THE FOLLOWING ADDRESS
0110	0000	UNLOCK THE FOLLOWING ADDRESS (PART 1) (WRITE/UNLOCK)
0111	0000	UNLOCK THE FOLLOWING ADDRESS
1011	NNNN	CLEAR ALL CONTENTION BITS FOR THE GIVEN SYSTEM(S) IN ALL ADDRESSES
1110	0000	UNLOCK THE ADDRESS GIVEN IN PART 1, IF AND ONLY

- SHAREDISK INFORMATION -

IF LOCK WAS RECEIVED IN PART 1.
(WRITE/UNLOCK PART 2)

1111 NNNN UNLOCK ALL ADDRESSES FOR THE GIVEN SYSTEM(S)

1010 0000 READ CONTENTS OF SPECIFIED FPM ADDRESS
(IN WDC) INTO ADDRESS REGISTERS
(MAINTENANCE FUNCTION ONLY - USED FOR TESTING)

WHERE NNNN IS

NOOO = SYSTEM 3
ONOO = SYSTEM 2
OONO = SYSTEM 1
OOON = SYSTEM 0

NOTE: B-5500 CAN ONLY OPERATE ON ONE SYSTEM NUMBER AT ONE TIME. EACH SYSTEM OPERATED ON REQUIRES A SEPERATE OPERATION FROM THE DFC.

WDC

WDC IS THE REGISTER WHICH INDICATES THE FPM MEMORY WORD BEING ACCESSED.

LOCKED

IF ON INDICATES THE DISK ADDRESS AT WDC IS LOCKED.

ADR1

THIS AREA CONTAINS THE "LOCKER" CODE AND SYSTEM CONTENTION BITS. SYID2 AND SYID1 ARE THE "LOCKER" BITS AND ARE DECODED AS FOLLOWS:

SYID2	SYID1	
1	1	= SYSTEM 3
1	0	= SYSTEM 2
0	1	= SYSTEM 1
0	0	= SYSTEM 0

THE SYSTEM CONTENTION BITS ARE APPROPRIATELY MARKED. EX: SY3 TRUE INDICATES SYSTEM 3 IS CONTENDING FOR THE ADDRESS

- SHAREDISK INFORMATION -

LOCK

WHEN TRUE INDICATES THE ADDRESS IS LOCKED.

ENDOP

WHEN TRUE INDICATES THE END OF A REQUESTED OPERATION.

LKSET

WHEN TRUE INDICATES MEMORY IS AVAILABLE TO ENTER AND LOCK A DISK ADDRESS.

ENDOVER

WHEN TRUE INDICATES THAT THE OPERATION IS COMPLETE.

LMWDA

WHEN TRUE INDICATES THAT THE ADDRESS IS ABOUT TO BE LOCKED.

LMAL

WHEN TRUE INDICATES THAT LOCK MEMORY IS AVAILABLE.

SC

INDICATES WHICH DFC MAY BE GRANTED ACCESS TO THE FPM IF THERE ARE NO PREVIOUS REQUESTORS WAITING.

LMCP (COUNTER)

THIS SERIES OF INDICATORS DISPLAYS THE LOCK MEMORY CLOCK PULSES FOR THE TRANSFER CLOCK.

LMCP (FLIP FLOP)

TRANSITION FROM FALSE TO TRUE INDICATES A CHANGE OF WRITE/READY LINES.

1MC

WHEN TRUE INDICATES THAT A ONE MEGACYCLE CLOCK IS IN OPERATION. THE RATE WHEN FALSE WOULD BE 500 KC.

ADRFILL

WHEN TRUE INDICATES A TRANSFER FROM THE ADDRESS REGISTERS IS TAKING PLACE.

ADREND

WHEN TRUE INDICATES THAT THE WORD BEING TRANSFERRED IS THE ONLY WORD TO BE TRANSFERRED.

ENDFLO

WHEN TRUE INDICATES THE END OF THE TRANSFER.

FLOSET

WHEN TRUE INDICATES THAT AN UNLOCKED AND CONTENDED FOR ADDRESS FOR THE REQUESTING SYSTEM HAS BEEN FOUND.

LMOP

WHEN TRUE INDICATES THE LOCK MEMORY OPERATION IS COMPLETE. THIS WILL REMAIN TRUE UNTIL THE DFC TURNS OFF THE REQUEST.

STOP + 1

WHEN TRUE INDICATES OPERATION IS ONE CLOCK BEYOND STOP. IT INDICATES A WRITE TO MEMORY WILL OCCUR AT CLOCK TIME.

STOP

WHEN TRUE INDICATES THAT THE WORD COUNT IS STOPPED.

MC ≠ ADR

WHEN TRUE INDICATES THAT THE CONTENTS OF THE MEMORY CELLS AS SPECIFIED BY WDC ARE NOT EQUAL TO THE CONTENTS OF THE ADDRESS REGISTER.

* SHAREDISK INFORMATION *

SRP1, SRP2, AND SRP3

PRIORITY STACK FOR REQUESTORS.

ULOK

INDICATES INTERRUPTS TO BE SENT TO REQUESTORS AS A RESULT OF A CONTENDED ADDRESS BEING FREED. IT IS DECODED AS INDICATED ON THE LABEL. EX: IF ULOK3 IS TRUE, AN INTERRUPT IS SENT TO SYSTEM 3 AT LMOP TIME.

ADR2, ADR3, ADR4, AND ADR5

THESE REGISTERS CONTAIN THE DISK ADDRESS. IN THE FPM BITS 8-5 OF ADR2 ARE ALL SET TRUE, AND ARE NOT USED . THE DISK ADDRESS IS IN BCD IN THE SAME CONFIGURATION AS IN THE DISK ADDRESS FIELD IN CORE.

THE DISPLAY ON THE DFC WILL REFLECT THE INSTRUCTION SENT TO THE FPM UNTIL THE COMPLETION OF THAT OPERATION.

- SHAREDISK INFORMATION -

DFC - INTERPRETATION OF INDICATORS

HARDWARE MODIFICATIONS MADE TO IMPLEMENT SHARED DISK MAKE USE OF BITS IN THE DISK ADDRESS AND DESCRIPTOR THAT WERE NOT PREVIOUSLY USED. THE ZONE BITS IN THE DISK ADDRESS ARE USED TO INDICATE THE FPM OPERATION TO BE PERFORMED. UNFORTUNATELY, THE DISK FILE CONTROL HAS NO PROVISION FOR DISPLAYING THESE BITS, SO THE USER MUST DECODE THE FPM INSTRUCTION FROM THE DISPLAY REGISTERS ON THE FPM.

- SHAREDISK INFORMATION -

I/O DESCRIPTOR AND ADDRESS WORD FORMATS

NOTE: THE ENTRY "X" IN THE FOLLOWING TABLE SHOULD BE INTERPRETED AS ANY OF THE POSSIBLE ENTRIES FOR THE FIELD INDICATED ARE PERMISSIBLE.

INSTRUCTION	ADDRESS WORD BIT NUMBER						I/O WD CNT	DESCRIPTOR --BITS-- 18		23	24	SEG CNT
	30	31	36	37	42	43						
READ	0	0	0	0	0	0	X	0	X	1	X	
READ/LOCK	0	0	0	0	1	0	X	0	X	1	X	
WRITE	0	0	0	0	0	0	X	0	X	0	X	
WRITE/UNLOCK	0	0	0	0	1	0	X	0	X	0	X	
CLEAR ALL CONTENTION BITS OF SYSTEM "N"	N	N	1	0	1	1	0	0	1	0	0	
UNLOCK ALL ADDRESSES OF SYSTEM "N"	N	N	1	1	1	1	0	0	1	0	0	
CLEAR CONTENTION BIT SYSTEM "N" ADDRESS "X"	N	N	0	0	1	1	0	0	1	0	0	
UNLOCK ADDRESS "X"	0	0	0	1	1	1	0	0	1	0	0	
REPORT FREE ADDRESS(1)	0	0	0	0	1	0	0	1	0	0	0	
REPORT FREE ADDRESS(2)	0	0	0	1	0	0	0	1	0	0	0	

NOTE: THE CODE FOR SYSTEM NUMBERS IS AS FOLLOWS:

00 = SYSTEM 0
01 = SYSTEM 1
10 = SYSTEM 2
11 = SYSTEM 3

DISK FILE RESULT DESCRIPTORS WITH THE FILE PROTECT OPTION

WHEN USING THE FILE PROTECT OPTION, THE FOLLOWING BITS IN THE RESULT DESCRIPTOR CONTAIN THIS ADDITIONAL MEANING:

BITS 25 AND 27

THE FILE PROTECT MEMORY IS NOT AVAILABLE AND A FPM OPERATION WAS REQUESTED.

BITS 25 AND 28

A READ/LOCK OPERATION WAS REQUESTED AND THE ADDRESS WAS LOCKED.

BITS 25 AND 29

A READ/LOCK OPERATION WAS REQUESTED AND THE FPM WAS FULL.

- SHAREDISK INFORMATION -

HARDWARE REQUIREMENTS FOR A SHAREDISK SYSTEM

THE MAXMUM B-5700 SHAREDISK SYSTEM CONFIGURATION IS ONE WHERE FOUR B-5700"S SHARE THE SAME DISK STORAGE. THE PRINCIPLE LIMITING FACTOR IN ANY SHAREDISK NETWORK IS THAT A MAXIMUM OF FOUR DISK FILE CONTRUL UNITS ARE ALLOWED; HENCE ONE DFC PER B-5700 YIELDS THE MAXIMUM CONFIGURATION. OTHER POSSIBLE COMBINATIONS INCLUDE: 1) TWO B5500 SYSTEMS WITH ONE OR TWO DFCS PER SYSTEM, AND 2) THREE B5500S WITH ONE OF THE SYSTEMS POSSIBLY UTILIZING TWO DFCS.

THE MAXIMUM NUMBER OF ELECTRONICS UNITS(EU"S) PERMISSIBLE IS TEN; HENCE WITH 5 DISK STORAGE MODULES PER EU THE MAXIMUM STORAGE AVAILABLE UNDER SHAREDISK IS 50 DISK MODULES OR 480 MILLION CHARACTERS. IF ONLY TWO B5500 SYSTEMS ARE TO SHARE DISK STORAGE OF 25 MODULES (5 EUS) OR LESS IT IS POSSIBLE TO USE ONLY ONE B451 EXPANDED DISK FILE CONTROL (DISK FILE EXCHANGE).

IT SHOULD BE NOTED THAT A B451 EXPANDED CONTROL UNIT IS REQUIRED ON ALL B-5500S UTILIZING A FILE PROTECT MEMORY; THE DISK FILE CONTRUL CANNOT FUNCTION PROPERLY WITH THE FILE PROTECT MEMORY UNLESS THE B451 IS PRESENT.

- SHAREDISK INFORMATION -

THE HARDWARE REQUIRED TO ADD SHAREDISK TO AN EXISTING B-5500 OR B-5700 IS AS FOLLOWS:

UNIT	QUANTITY	M&E NUMBER
INDEPENDENT AUXILIARY CABINET	1	1906 0270
BASIC FILE PROTECT MEMORY UNIT	1	1635 5539
ADAPTER FILE PROTECT	NOTE 1	1635 5547
B-5500 DISK FILE CONTROL IB-FPM	4 MAXIMUM	1904 8388
B-5500 I/O CONTROL MOD III FPM	NOTE 2	1904 8370
B-5500 CABLE KIT	NOTE 3	1904 3264
B451 EXPANDED DISK FILE CONTROL	NOTE 4	1106 2270
MAINTENANCE CARDS	6	1143 7795
INDICATOR BLOCKS	6	1129 1812
SWITCH BLOCK	1	1145 3511

NOTE 1: UP TO 8 ADAPTERS (128 WORDS) CAN BE USED. ONE ADAPTER (16 WORDS) IS REQUIRED. ALL INITIAL B-5500 SHAREDISK INSTALLATIONS SHOULD HAVE 8 ADAPTERS UNTIL A METHOD FOR DETERMINING THE "MINIMUM NECESSARY" FOR PARTICULAR APPLICATIONS IS DETERMINED.

NOTE 2: MODIFICATION IS REQUIRED TO THE EXISTING B-5500 I/O CONTROL.

NOTE 3: ONE CABLE KIT IS REQUIRED FOR EACH DISK FILE CONTROL CONECTED TO THE FILE PROTECT MEMORY.

NOTE 4: ONE REQUIRED FOR 1 TO 5 ELECTRONICS UNITS; TWO REQUIRED FOR 6 TO 10 ELECTRONICS UNITS.

SHEET

THE "SHEET" PROVIDES INFORMATION TO THE "SELECTION" ROUTINE TO INTRODUCE JOBS INTO THE MIX. ENTRIES IN THIS TABLE ARE MADE BY THE CONTROLCARD ROUTINES INTO ESP DISK. THE ENTRIES ARE AS FOLLOWS:

WORD	FIELD	CONTENTS
0	[1:1]	= 1 THEN JOB WAS A COMPILE
	[6:42]	<MFID> OF OBJECT PROGRAM
1	[1:1]	= 1 THEN COMPILED BY COBOL
	[6:42]	<FID> OF OBJECT PROGRAM
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 (COMPILE AND GO) SET TO 2 LATER 2 = EXECUTE JOB 3 = COMPILER (SYNTAX CHECK) 4 = COMPILER (COMPILE TO LIBRARY) 5 = RUN JOB
	[18:15]	SKELETON DISK ADDRESS (IF SHEET[2],[8:10] = 1, 2, OR 4)
	[33:15]	PRIORITY.
3	[2:1]	JOB IS A RESTART
	[8:10]	SCHEDULE ID
	[33:15]	ESTIMATED PROCESSOR TIME
4		ESTIMATED I-O TIME
5	[1:23]	STARTING DATE FOR LOG (BINARY)
	[24:24]	STARTING TIME FOR LOG
6	[1:1]	1 = NEW FORMAT FOR LABEL EQUATION
	[9:9]	TU-BUF FOR JOBS STARTED FROM RJE
	[18:15]	FPB INFORMATION
	[33:15]	CONTAINS THE DISK ADDRESS OF THE FIRST CHARACTERS OF A CONTROL CARD, INFORMATION USED BY SIGNOFF WHICH IS CALLED BY COMS
7		COMMON VALUE OR 0
12		STACK SIZE (512 DEFAULT VALUE) (THIS RUN ONLY)
15		DISK ADDRESS OF LABEL EQUATION ENTRIES (LABEL EQUATES DONE AT COMPILATION TIME)
16		ESTIMATED PROCESSOR TIME

- SHEET -

17		ESTIMATED I-O TIME
18		PRIORITY
19		COMMON VALUE
20		ESTIMATED CORE REQUIREMENT
21		STACK SIZE
22		SAVE FACTOR FOR OBJECT FILE (ON COMPILE TO LIBRARY)
23	[9:9]	REMOTE STATION ADDRESS (IF ANY)
	[31:17]	TIME JOB WAS ENTERED IN SHEET (FOR TS MESSAGE)
24		USER CODE
29		DISK ADDRESS OF NEXT "SHEET" ENTRY (=0 IF LAST ENTRY AT THE SAME PRIORITY)

SHEETDIDDLER

SHEETDIDDLER(BUFF,TYPE,STD) IS A PROCEDURE WHICH MANIPULATES THE SHEET FROM INFORMATION IN "BUFF", DEPENDING UPON "TYPE". "SID" IS A SCHEDULE INDEX FOR THE JOB. THE PROCEDURE IS CALLED TO COMPLETE THE ELIMINATION OF A PROGRAM FROM CORE. TYPE IS DEFINED AS FOLLOWS:

VALUE	FUNCTION
18	PS - CHANGE PRIORITY OF JOB IN SCHEDULE
19	XS - EXECUTE JOB IN SCHEDULE (FORCE SELECTION)
20	ES - ELIMINATE JOB FROM SCHEDULE (FORCE SELECTION THEN "DS")

SIGNOFF

SIGNOFF(VECTOR,FILEBLOCK) IS A PROCEDURE WHICH CREATES THE LOG ENTRY FOR A JOB AND CALLS LOGFREE TO WRITE THE ENTRY TO DISK. "VECTOR" IS THE PRT ROW FOR THE JOB AND "FILEBLOCK" IS A DESCRIPTOR POINTING AT THE JOB'S FILE PARAMETER BLOCK.

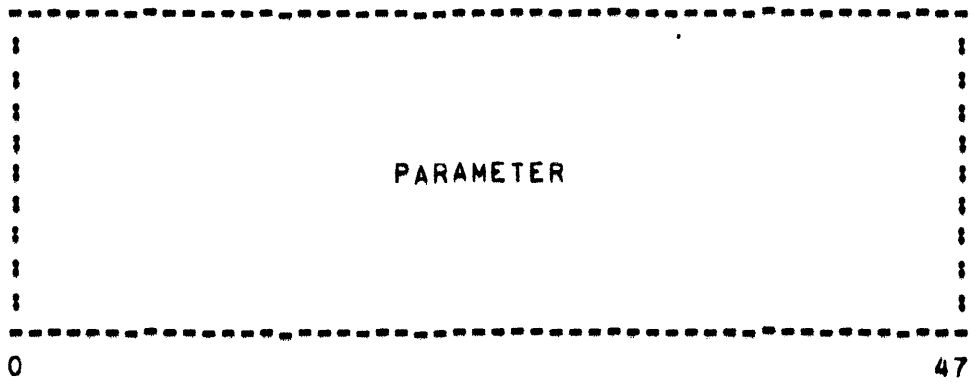
SLATE[*]

THE "SLATE" IS A QUEUE OF REQUESTS TO RUN INDEPENDENT "MCP" ROUTINES (I.E., ROUTINES WHOSE FUNCTIONS ARE NOT DIRECTLY RELATED TO OBJECT PROGRAMS SUCH AS STATUS, CONTROLCARD, SELECTION, RUN, AND DCWRITE.

"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 NOTED BELOW.

WORD 1



THE PARAMETER FOR THE INDEPENDENT ROUTINE.

WORD 2

	FIELD	CONTENTS
[0:33]	0	
[33:15]	ADDRESS	ADDRESS POINTS TO PROGRAM DESCRIPTOR OF INDEPENDENT ROUTINE.

FOR ADDITIONAL INFORMATION, PLEASE REFER TO THE DISCUSSION OF "INDEPENDENTRUNNER".

SLEEP

THE SLEEP ROUTINE IS DEFINED TO BE A CALL ON THE SNOOZE ROUTINE PASSING ITS TWO PARAMETERS AS THE SECOND TWO PARAMETERS OF SNOOZE AND PRYOR[P1MIX] AS THE FIRST PARAMETER.

SNOOZE

THE PROCEDURE DECLARATION FOR SNOOZE IS AS FOLLOWS:

```
SAVE PROCEDURE SNOOZE(PRYR,ADDRESS,MASK);

     VALUE PRYR,ADDRESS,MASK;
     REAL  PRYR,ADDRESS,MASK;
```

THE INTERPRETATION OF THE PARAMETERS IS AS FOLLOWS:

PRYR	PRIORITY WITH WHICH PROCESS IS TO BE PLACED IN THE BED.
ADDRESS	MEMORY ADDRESS OF WORD TO BE TESTED AGAINST MASK.
MASK	MASK WORD.

THE SLEEP ROUTINE IS DEFINED TO BE A CALL ON THE SNOOZE ROUTINE PASSING ITS TWO PARAMETERS AS THE SECOND TWO PARAMETERS OF SNOOZE AND PRYOR[P1MIX] AS THE FIRST PARAMETER.

SOFTI

CONTAINS THE NUMBER OF JOBS IN THE MIX WHICH HAVE SOFTWARE INTERRUPTS DECLARED.

SPACECTR

COUNTS THE (NUMBER OF REQUESTS) MINUS THE (NUMBER OF ALLOCATIONS) OF USER DISK FOR USE IN CASE OF NO USER DISK.

SPOUT

SPOUT IS THE ROUTINE WHICH PLACES INFORMATION INTO THE MESSGEHOLDER QUEUE AND IS EVOKED WHEN SOMETHING DESIRES TO TYPE A MESSAGE ON THE SPO. MESSAGEWRITER IS THE ROUTINE WHICH NORMALLY REMOVES INFORMATION FROM THE MESSAGEHOLDER QUEUE AFTER A MESSAGE IS TYPED. KEYIN WILL ALSO FLUSH THE QUEUE WHEN A "BK" MESSAGE IS ENTERED FROM THE SPO.

SPOUT IS CALLED BY AMOST ANY MCP ROUTINE WICH DESIRES TO WRITE A MESSAGE ON THE SPO, OR ON A REMOTE TERMINAL. SPOUT IS TYPELESS AND HAS ONE EXPLICIT PARAMETER WHICH IS INTERPRETED AS FOLLOWS:

FIELD	CONTENTS
[1:1]	1 = SEND MESSAGE TO SPO
[2:1]	1 = CARRIAGE RETURN & LINE FEED AT START OF MSG
[9:9]	TU AND BUF OF REMOTE TERMINAL
[33:15]	POINTER TO BEGINNING OF MESSGE

THE MSCW IS ALSO USED AS AN IMPLICIT PARAMETER. IF THE [33:15] FIELD OF THE MSCW (ACTUALLY THE WORD AT (F=2)) IS NON-ZERO, IT IS USED AS THE FLAG THAT THE DISPLAY ROUTINE IS CALLING, PASSING THE EXACT NUMBER OF CHARACTERS IN THE MESSAGE. IF DISPLAY IS NOT CALLING, SPOUT REMOVES EXCESS SPACES AND PRODUCES ITS OWN COUNT OF CHARACTERS TO BE TYPED.

SPOUT DIALS OUT THE [9:6] FIELD OF THE FRST WORD OF THE MEMORY LINK JUST BEFORE THE MESSAGE TO DETERMINE THE MIX INDEX OF THE JOB FOR WHICH THE MESSAGE IS BEING TYPED. IT THEN CHECKS TUSTABABYMIX [P1MIX] AND ATTACHED[P1MIX] TO DETERMINE THE REM@TE TERMINALS, IF ANY, THAT ARE ATTACHED TO THE JOB. IF MESSAGES ARE TO BE SENT TU REMOTE TERMINALS, SPOUT COPIES THEM, AND LINKS THE COPIES WITH PROPER DESTINATION INFORMATION INTO THE STATIONMESSAGEHOLDER QUEUE.

IF THE [9:9] FIELD OF ITS PARAMETER IS ZERO (SPO) OR ONE (ALTERNATE SPO), OR THE [1:1] FIELD IS ONE, SPOUT MUST SEND INFORMATION TO THE SPO AND/OR ALTERNATE SPO"S. TO DETERMINE WHERE INFORMATION MUST BE TYPED, SPOUT EXAMINES THE SPOWORD. IF THE SPOWORD IS NON-NEGATIVE, THE MESSAGE MUST AT LEAST BE TYPED ON THE SPO BY BEING PLACED IN THE MESSAGEHOLDER QUEUE. FURTHERMORE, IF ANY OF THE FIELDS [12:19], [21:9], [30:9], OR [39:9] ARE NON-ZERO, THOSE FIELDS CONTAIN THE TU/BUFF ADDRESS AND SPOUT MUST COPY THE MESSAGE AND LINK IT INTO THE STATIONMESSAGEHOLDER QUEUE WITH THE PROPER DESTINATION INFORMATION SUCH THAT THE MESSAGE IS SENT TO THE PROPER ALTERNATE SPO"S, IF ANY.

SPOWORD

THE SPOWORD SPECIFIES WHAT SPO-S EXIST, THE CENTRAL SITE KEYBOARD IS ALWAYS A SPO, THE MESSAGE PRINTER MAY OR MAY NOT BE A SPO, UP TO FOUR ADDITIONAL B487 REMOTE TERMINALS MAY BE DESIGNATED AS REMOTE SPO"S.

```
SPOWORD.[1:1]      1 = MESSAGE PRINTER NOT A SPO,
       .[3:9]      IF NON-ZERO, TU/BUFF IS ALTERNATE SPO.
       .[12:9]     IF NON-ZERO, TU/BUFF IS ALTERNATE SPO.
       .[21:9]     IF NON-ZERO, TU/BUFF IS ALTERNATE SPO.
       .[30:9]     IF NON-ZERO, TU/BUFF IS ALTERNATE SPO.
       .[39:9]     IF NON-ZERO, TU/BUFF IS ALTERNATE SPO.
```

THE SPOWORD IS INITIALIZED FROM DIRECTORYTOP AND IS CHANGED AND INTERROGATED BY THE "SPOSIN" ROUTINE. DIRECTORYTOP IS THUS RE-WRITTEN EACH TIME THE SPOWORD IS CHANGED.

STARTADECK

STARTADECK(N) IS A PROCEDURE WHICH ATTACHES A PSEUDO-DECK TO A PSEUDO-READER AND CALLS CONTROLACRD, IF POSSIBLE.

STATIONMESSAGEHOLDER

STATIONMESSAGEHOLDER IS A CIRCULAR QUEUE OF MESSGES TO BE SENT TO A B487. THESE MESSAGES MUST BE RANKED BY THEIR [9:9] FIELDS. STATIONMESSAGEWRITER TAKES THEM IN ORDER BY THESE SUB-GROUPS. THE SECOND STORAGE LINK IS USED AS THE QUEUE WORD AND IS DESCRIBED AS FOLLOWS:

FIELD	CONTENTS
[0:1]	0 = FLAG BIT
[1:1]	1 = PART OF THE MESSAGE HAS BEEN OUTPUT
[2:1]	1 = PROGRAM OUTPUT
	0 = OTHERWISE
[3:1]	MESSAGE COMPLETED (TIME TO CHECK BREAK)
[4:5]	MIX IF MIX MESSAGE, OTHERWISE 0
[3:6]	INDEX OF MESSAGES ALREADY OUTPUT FOR RJE LINES
[9:4]	TU ADDRESS
[13:1]	0 TO SELECT DCTU TRANSLATOR, NEVER 0
[14:4]	BUFFER ADDRESS
[18:15]	BACK LINK TO PREVIOUS MESSAGE OR STATIONMESSAGEHOLDER
[33:15]	STATION MESSAGE LINK TO NEXT MESSAGE OR TO STATIONMESSAGEHOLDER

STATIONMESSAGEHOLDER IS BUILT BY VARIOUS ROUTINES IN THE MCP, PRIMARILY SPOUT AND COM36, AND IS ACTED UPON BY STATIONMESSAGEWRITER AND KEYIN (IN RESPONSE TO THE "BK" MESSAGE).

THE STATIONMESSAGEWRITER PROCEDURE HANDLES ALL B487 OUTPUT. BUFFER SIZE NEED NOT BE KNOWN AS THE PROCEDURE LEARNS FROM THE RESULT DESCRIPTOR HOW MUCH WAS WRITTEN. A MESSAGE IS OUTPUT ON ONE AND ONLY ONE STATION AS THE OUTPUT METHOD IS DESTRUCTIVE. THE MAXIMUM MESSAGE SIZE HANDLED IS 4095 WORDS. THE OUTPUT IS TRANSPARENT IN THAT NO LINE CONTROL IS ADDED.

STATIONMESSAGEWRITER IS CALLED INDEPENDENTLY BY ANY ROUTINE WHICH ADDS AN ENTRY TO STATIONMESSAGEHOLDER AND DISCOVERS THAT STATIONMESSAGEWRITER IS NOT RUNNING. STATIONMESSAGEWRITER TERMINATES ONLY WHEN ITS QUEUE IS EMPTY.

STATION TABLE

THE STATION TABLE IS PROVIDED AS A DUMP DEBUGGING AID AND EACH WORD HAS THE FOLLOWING FORMAT:

FIELD		CONTENTS
[0:1]		FLAG BIT (OFF)
[1:1]		OUTPUT IN PROCESS BY STATIONMESSAGEWRITER
[2:1]		SPO CONSOLE INPUT REQUEST FLAG (BIT 32 SHOULD ALSO BE ON)
[3:1]		CANNOT BE USED
[4:4]		TU INDEX INTO STATION FOR NEXT CONTROL STATION. IF NOT A CONTROL STATION, ITS OWN INDEX.
[8:1]		CANNOT BE USED
[9:4]	*	TU ADDRESS FOR THIS WORD
[13:1]	*	DTCU TRANSLATOR BYPASSED 1 = TRANSLATE, 0 = OMIT TRANSLATE 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 0 = GROUP MARK, 1 = IFAL
[26:1]		BREAK
[27:1]	*	WRITE READY 0 = GROUP MARK FINISH WRITE 1 = WRITE WITHOUT GROUP MARK ENDING (ADDITIONAL WRITE REQUIRED TO CLEAR BUFFER)
[28:1]	*	INPUT ERROR
[29:1]		WRITE IN PROCESS
[30:1]	*	STATION NOT READY
[31:1]		MIX MESSAGES NOT DESIRED FLAG 0 = OUTPUT MESSAGES, 1 = INHIBIT MESSAGES
[32:1]		SPO CONSOLE FLAG. (WHEN THIS FLAG IS ON, ALL INPUT IS TREATED AS IF IT HAD ORIGINATED AT THE SPO).
[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] STATION IS RJE TERMINAL

IN ADDITION, STATION[0,0] CONTAINS THE HEAD OF THE LINKED LISTS OF CONTROL STATIONS IN THE .[4:4] AND .[18:4] FIELDS OF THE OTHER CONTROL WORDS IN THE STATION ARRAY.

THE STATION ARRAY IS MANIPULATED BY MANY DIFFERENT ROUTINES IN THE MCP. IT IS USUALLY INTERROGATED BY "GET" AND "WHATSUP".

- STATUS, STATWORD, STOPJOB, SVRESULT -

STATUS

STATUS IS A PROCEDURE WHICH IS INDEPENDENTLY RUN WHEN THE STATUS OF A PERIPHERAL UNIT CHANGES. STATUS READS LABELS AND SETS A SERIES OF INTERNAL TABLES TO IDENTIFY UNITS.

STATWORD

CONTAINS INFORMATION USED BY STATISTICS CODE IN MCP.

STOPJOB

CONTAINS THE MIX INDEX OF A JOB BEING ST"ED OR @1777 IF NO JOB IS BEING ST"ED.

SVRESULT

CONTAINS TOTAL NUMBER OF ITEMS WHICH HAVE BEEN STORED IN SAVERESULT.

SYSTEM STATISTICS FACILITY

STATISTICS LOG

THE FOLLOWING SECTION DESCRIBES THE OPERATION AND USE OF THE SYSTEM MEASUREMENT FACILITIES AS PRESENTLY IMPLEMENTED ON BOTH THE STANDARD AND TIME SHARING VERSIONS OF THE MASTER CONTROL PROGRAM. THE FACILITIES ARE OFFERED AS A COMPILE TIME OPTION AND CAN BE INCLUDED IN A SYSTEM BY GIVING THE "$ SET" CARD, WITH THE STATISTICS PARAMETER, THE VALUE OF TRUE. THESE FACILITIES FORM PORTIONS OF A BASIC SOFTWARE MONITOR OF SYSTEM PERFORMANCE. GENERALLY, THEY CONCENTRATE ON DISK AGTIVITY AS IT RELATES TO THE TOTAL SYSTEM, AND DETAILED INFORMATION ON RESOURCE ALLOCATION ON A JOB-TO-JOB BASIS. IT CAN BE EXPECTED THAT BOTH THE CONFIGURATION AND SCOPE OF THE MEASUREMENT FACILITIES NOW PROVIDED WILL CHANGE AS STATISTICAL REQUISITES VARY IN THE FUTURE.

GENERAL CHARACTERISTICS.

THE OVERALL APPROACH HAS BEEN TO PROVIDE A STATISTICAL DATA BASE OF A DUAL NATURE, THE TWO PORTIONS OF WHICH PROVIDE A GLIMPSE INTO THE MACRO- AND MICRO-LEVELS OF SYSTEM UTILIZATION. THESE PORTIONS ARE, AT PRESENT, COMPLEMENTARY, BUT, TAKEN TOGETHER, THEY PROVIDE A REPRESENTATIVE PICTURE OF SYSTEM UTILIZATION.

ONE PART OF THIS DATA BASE IS AN EXTENDED LOG WHICH CONTAINS A DETAILED LOOK AT THE ACTUAL RESOURCES USED AND SYSTEM OVERHEAD ENCOUNTERED DURING JOB EXECUTION. DATA IS COLLECTED AT VARIOUS POINTS IN THE MCP TO REFLECT THE VARIOUS STAGES OF JOB EXECUTION. THESE ARE GATHERED TOGETHER AND ENTERED INTO THE LOG AT JOB TERMINATION. SUCH A DATA BASE PROVIDES A PICTURE OF THE OPERATING ENVIRONMENT IN WHICH A PARTICULAR JOB RAN OR THAT ENVIRONMENT WHICH EXISTED OVER A SPECIFIED PERIOD OF TIME.

THE OTHER PORTION OF THE BASIC DATA BASE EXISTS IN THE FORM OF A SYSTEM STATISTICS FILE WHICH IS CONTINUOUSLY UPDATED AND PERIODICALLY TRANSFERRED TO PERMANENT DISK STORAGE. THIS MASTER STATISTICS FILE IS UPDATED BY INFORMATION WHICH IS EITHER TIME OR TASK RELATED. TIME RELATED INFORMATION IS INITIATED VIA THE SYSTEM TIMER. THE NATURE OF THE INFORMATION CONTAINED IN THIS FILE HAS BEEN, UP TO NOW, INCREMENTAL AND CUMULATIVE IN NATURE. THIS INFORMATION IS UPDATED IN CORE AND PERIODICALLY (I.E., WHENEVER NSECOND IS CALLED) WRITTEN ON DISK FOR TEMPORARY STORAGE. THIS TEMPORARILY STORED INFORMATION IS, IN TURN, PERIODICALLY

(APPROXIMATELY EVERY 30 MINUTES) TRANSFERRED AS ONE RECORD TO A PERMANENT SYSTEM FILE ON DISK (SYSTAT <SYSTEM MNEMONIC>/DISK) FROM A TEMPORARY SYSTEM FILE ON DISK (SYSTEM <SYSTEM MNEMONIC>/STATS), AFTER WHICH THE TEMPORARY STORAGE AREA IN CORE IS RE-INITIALIZED TO ZERO AND THE ACQUISITION OF STATISTICS BEGINS ANEW. THE SI SYSTEM MESSAGE ALLOWS THE ADJUSTMENT OF THIS VALUE. THE FORMAT OF THIS MESSAGE IS SI <INTEGER>. FOR BOTH TIME SHARING AND STANDARD MCP'S AN EMPTY SYSTEM STATISTICS FILE CAN BE CREATED WITH THE "SY" MESSAGE ENTERED THROUGH A SPO. THE NAME OF THE CURRENT STATISTICS FILE (I.E., SYSTAT <SYSTEM MNEMONIC>/ DISK) IS CHANGED TO <NUMBER1> ON <NUMBER2>/SYSTAT <SYSTEM MNEMONIC> AND A NEW FILE, SYSTAT <SYSTEM MNEMONIC>/<DISK>, IS CREATED, WHERE:

<NUMBER1> IS A 2-DIGIT NUMBER (00 THROUGH 99) OF ONE OF THE STATISTICS FILES FOR A PARTICULAR DATE, AND

<NUMBER2> IS THE DAY CREATED (1 THROUGH 365).

IF THE MCP INVOLVED IS COMPILED WITH THE SHAREDISK $SET OPTION TRUE, THE SYSTEM MNEMONIC REFERS TO THE SYSTEM (I.E., SYSTEM A, B, C, OR D) FOR WHICH THE FILE IS CREATED. OTHERWISE, THE SYSTEM MNEMONIC IS BLANK.

THE SYSTEM STATISTICS FILE IS AUTOMATICALLY FILLED WITH ONE RECORD AFTER A 30 MINUTE INTERVAL, BEGINNING WITH THE TIME OF THE INITIAL HALT-LOAD OF THE SYSTEM. AS THE SY MESSAGE CAUSES SUBSEQUENT TIME INTERVALS TO BEGIN AT THE TIME THE MESSAGE IS ENTERED, IT CAN BE USED TO CAUSE THE ACCUMULATION OF STATISTICS FOR PERIODS BEGINNING ON THE HOUR AND HALF HOUR.

TIME SHARING.

STATISTICS RELATING TO THE TIME SHARING SYSTEM ARE KEPT ON A SYSTEM STATISTICS FILE AND IN THE TS LOG. THE LOG STATISTICS ARE KEPT IN BOTH THE TYPE 8, EOJ STATISTICS MESSAGE, ALREADY PRESENT IN THE TS LOG, AND A NEW, TYPE 19, MESSAGE USED FOR STATISTICS ONLY.

STANDARD SYSTEM.

STATISTICS RELATING TO THE BATCH SYSTEM ARE KEPT IN THE SYSTEM STATISTICS FILE PREVIOUSLY MENTIONED WITH RESPECT TO TOTAL SYSTEM USAGE AND IN PSEUDO LOG FILE FOR INDIVIDUAL JOB STATISTICS. THIS LATTER FILE CONTAINS ONE RECORD PER JOB RUN ON THE SYSTEM, EACH OF WHICH CONTAINS SOME OF THE INFORMATION FOUND IN THE REGULAR SYSTEM LOG (PROCESSOR TIME, I/O TIME) IN ADDITION TO THE DATA OBTAINED THROUGH INCREASED JOB MONITORING. THIS APPROACH IS TAKEN TO EASE OFF-LINE ANALYSIS OF JOB STATISTICS WHILE NOT HAMPERING NORMAL LOG ANALYSIS. THE PSEUDO LOG HAS BEEN IMPLEMENTED IN SUCH A WAY THAT AN SL MESSAGE ENTERED VIA THE SUPERVISORY PRINTER CAUSES THE CURRENT

PSEUDO LOG FILE TO BE SAVED IN MUCH THE SAME WAY THE SY MESSAGE CAUSES THE SYSTEM STATISTICS FILE TO BE SAVED. THAT IS, THE CURRENT STATISTICS LOG FILE, STLOG <SYSTEM MNEMONIC>/STATS, IS CHANGED TO <NUMBER1> ON <NUMBER2>/STLOG <SYSTEM MNEMONIC>, AND A NEW STLOG <SYSTEM MNEMONIC>/STATS IS CREATED.

OPERATION.

ONCE INITIALIZED THE SYSTEM BEGINS ACCUMULATING STATISTICS WITHOUT OPERATOR INTERVENTION. AS NOTED PREVIOUSLY, THE SYSTEM STATISTICS FILE IS UPDATED AT HALF-HOUR INTERVALS. THE STATISTICS LOG FILE, IF PRESENT, CONTAINS A NEW ENTRY FOR EACH JOB INITIATED ON THE SYSTEM. AS THE STATISTICAL INFORMATION IN CORE IS STORED IN THE CURRENT SYSTEM <SYSTEM MNEMONIC>/STATS FILE AND SUBSEQUENTLY RE-INITIALIZED TO ZERO AFTER A SY MESSAGE, THIS MESSAGE CAN BE USED TO GATHER DATA FOR PERIODS OF LESS THAN 30 MINUTES. THAT IS, AN SY MESSAGE ENTERED BEFORE AND AFTER A SPECIFIED PERIOD OF TIME CREATES A SINGLE RECORD FILE CONTAINING MEASUREMENT DATA FOR THAT PERIOD.

IF EITHER STATISTICS FILE BECOMES FULL, THE SYSTEM AUTOMATICALLY SAVES THE FILLED FILE IN THE SAME MANNER AS IF EITHER AN SY OR SL MESSAGE HAS BEEN ENTERED.

FILE DESCRIPTIONS.

SYSTEM STATISTICS FILE.

THIS FILE IS COMPOSED OF 60-WORD LOGICAL RECORDS. THE FIRST WORD OF THE RECORD FOLLOWING THE LAST STATISTICAL RECORD CONTAINS A FILE TERMINATE WORD IN THE FORM OF THE NUMBER @377777777777777. THE CONTENTS OF BOTH THE TIME SHARING AND BATCH FILES ARE WITH THE EXCEPTION THAT INFORMATION NOT APPLICABLE TO THE STANDARD SYSTEM (E. G., DATA RELATED TO SWAPPING HAVE ZERO ENTRIES IN THE PERTINENT ELEMENTS OF THE RECORD. THE FORMAT OF A TYPICAL RECORD IS AS FOLLOWS (ENTRIES MARKED BY * PERTAIN ONLY TO THE TIME SHARING FILE. WORD NUMBERS INCLUDED WITHIN PARENTHESES CONTAIN THE NUMBER OF DISK SEGMENTS INVOLVED FOR THE NUMBER OF I/O'S CONTAINED BY THE WORD WHOSE NUMBER IS TO THE LEFT):

WORD	DESCRIPTION
0	TOTAL NUMBER OF DISK I-O OPERATIONS FOR TIME PERIOD INVOLVED.
1	TIME SINCE LAST HALT-LOAD.
2	TOTAL NUMBER OF DISK I-O OPERATIONS HANDLED BY DISK FILE CONTROLLER A.
3	TOTAL NUMBER OF TIMER INTERRUPTS.
4	EU 0 DISK ACTIVITY (NUMBER OF DISK I-O OPERATIONS).
5	EU 1 DISK ACTIVITY (NUMBER OF DISK I-O OPERATIONS).
6	EU 2 DISK ACTIVITY (NUMBER OF DISK I-O OPERATIONS).
7	NUMBER OF TIME INTERRUPTS OCCURRING WHILE DISK FILE CONTROLLER (DFC) A IS IN USE.
8	NUMBER OF TIMER INTERRUPTS OCCURRING WHILE DFC B IS IN USE.
9	NUMBER OF NORMAL STATE DISK I-O OPERATIONS.
10(40)	NUMBER OF DISK I-O OPERATIONS INVOLVING MCP CODE.
11(41)	NUMBER OF DISK I-O OPERATIONS INVOLVING ESP CODE.
12(42)	NUMBER OF DISK I-O OPERATIONS INVOLVING BYPASS DIRECTORY.
13*	NUMBER OF DISK I-O OPERATIONS ORIGINATING BELOW THE FENCE.
14	NUMBER OF TIMER INTERRUPTS FOR WHICH THE MIX IS NOT ZERO.
15*(45)	NUMBER OF DISK I-O OPERATIONS RESULTING FROM SWAPPING.
16*(46)	NUMBER OF DISK ACCESSES TO DATACOM INPUT-OUTPUT

	TANKS.
17	NUMBER OF TIME INTERRUPTS OCCURRING WHILE BOTH DFC A AND DFC B ARE IN USE.
18 (48)	NUMBER OF DISK I=O OPERATIONS RESULTING FROM LIBRARY MAINTENANCE.
19*(49)	NUMBER OF DISK I=O OPERATIONS RESULTING FROM CODE BELOW FENCE.
20*(50)	NUMBER OF DISK I=O OPERATIONS RESULTING FROM CODE ABOVE FENCE.
21*(51)	NUMBER OF DISK I=O OPERATIONS RESULTING FROM DATA BELOW FENCE.
22*(52)	NUMBER OF DISK I=O OPERATIONS RESULTING FROM DATA ABOVE FENCE.
23*(53)	NUMBER OF DISK I=O OPERATIONS INVOLVING SYSTEM DISK.
24 (54)	NUMBER OF DISK I=O OPERATIONS INVOLVING LOG.
25 (55)	NUMBER OF DISK I=O OPERATIONS INVOLVING NAME PORTION OF A DIRECTORY SECTION.
26 (56)	NUMBER OF DISK I=O OPERATIONS INVOLVING PORTIONS OF A DIRECTORY SECTION.
27 (57)	NUMBER OF DISK ACCESSES TO PROGRAM FILES.
28	DATE • TIME SHARING SYSTEM MMDDYY • STANDARD SYSTEM YYDDD
29	TIME OF DAY RECORD IS ENTERED INTO FILE.
30	TOTAL NUMBER OF DISK SEGMENTS INVOLVED FOR ALL DISK I=O OPERATIONS RECORDED.
31	NUMBER OF TIMER INTERRUPTS WHILE MIX IS IN ZERO.
32	CUMULATIVE DISK DELAY, I.E., TIME FROM I=O REQUEST TO I=O INITIATION.
33*	CUMULATIVE SWAP DELAY.
34*	NUMBER OF NON=ZERO SWAP DELAYS.
35	NUMBER OF DISK I=O OPERATIONS VIA I=O CHANNEL 1.
36	PROCESSOR IDLE=BUSY: NUMBER OF TIMER INTERRUPTS OCCURRING WHILE MCP IS IN NOTHINGTODO STATE.
37	NUMBER OF DISK I=O OPERATIONS VIA I=O CHANNEL 2.
38*	NUMBER OF TIME INTERRUPTS FOR WHICH P1MIX IS EQUAL TO THE MIX NUMBER OF CANDE.
39	NORMAL STATE=DISK I=O OVERLAY (NUMBER OF TIME INTERRUPTS FOR WHICH MIX IS NOT ZERO AND DFC A AND DFC B IS IN USE.
43	CONTROL STATE=DISK I=O OVERLAP.
44	ELECTRONICS UNIT 3 DISK ACTIVITY.
47	TIME OF DAY WHEN DATA BEGAN BEING GATHERED FOR THE PARTICULAR RECORD INVOLVED.
58	I=O CHANNEL 3 DISK ACTIVITY.
59	I=O CHANNEL 4 DISK ACTIVITY.

STATISTICS LOG FILE.

THE STATISTICS LOG FILE CONTAINS ONE 15-WORD LOGICAL RECORD FOR EACH JOB BEGUN ON THE DCMCP. THE FIRST WORD OF THE RECORD FOLLOWING THE LAST LOG ENTRY CONTAINS THE END-OF-FILE MARKER AS THE SYSTEM STATISTICS FILE. THE FORMAT FOR ONE RECORD IS AS FOLLOWS:

WORD	FIELD	DESCRIPTION
0		PREFIX.
1		SUFFIX.
2		STARTING TIME.
3		TIME OF JOB TERMINATION.
4		PROCESSOR TIME (60THS OF A SECOND).
5		I-O TIME (60THS OF A SECOND).
6	[18:15]	AMOUNT OF CORE USED BY JOB.
7	[33:15]	AMOUNT OF CORE IN USE BY ALL JOBS IN MIX.
8		JOB TYPE:
	[42:6]	0 = NOT A COMPILATION. 1 = ALGOL. 2 = COBOL. 6 = FORTRAN. 7 = BASIC. 9 = XALGOL. 10 = TSPOL.
	[36:6]	0 = UNKNOWN OBJECT PROGRAM TYPE. 1 = BASIC OBJECT PROGRAM. 2 = ALGOL OBJECT PROGRAM. 3 = COBOL OBJECT PROGRAM. 4 = FORTRAN OBJECT PROGRAM. 5 = TSPOL OBJECT PROGRAM. 6 = XALGOL OBJECT PROGRAM.
9		UNUSED.
10	[1:23]	NUMBER OF DATA PRESENCE BIT INTERRUPTS.
	[24:24]	NUMBER OF CODE PRESENCE BIT INTERRUPTS.
11	[1:23]	NUMBER OF DATA OVERLAYS.
	[24:24]	NUMBER OF CODE OVERLAYS.
12		NUMBER OF SECONDARY CODE PRESENCE BIT INTERRUPTS.
13		NUMBER OF JOBS REMAINING IN MIX AFTER JOB TERMINATION.
14		UNUSED.

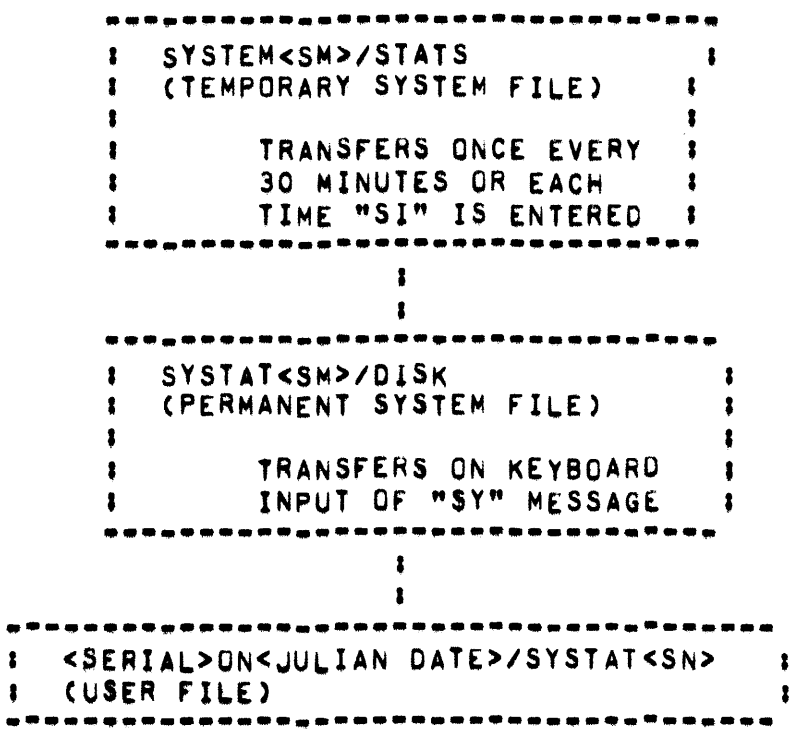
TIME-SHARING LOG ADDITIONS.

THE TYPE 19 STATISTICS MESSAGE CONTAINS ADDITIONAL JOB INFORMATION NOT FOUND IN THE TYPE 8, EOJ STATISTICS, MESSAGE. THE CONTENTS OF THE TYPE 19 MESSAGE IS AS FOLLOWS:

WORD	FIELD	DESCRIPTION
1		ACTUAL TIME IN CORE.
2	[3:15]	DALOC[P1MIX, 0],[33:14] = NUMBER OF 500-SEGMENT SECTIONS OBTAINED FOR OVERLAY FOR P1MIX.
	[13:30]	DALOC[P1MIX, (DALOC[P1MIX, 0],[33:15])] = NUMBER OF 100-SEGMENTS SUBSECTIONS OF LAST 500-SEGMENT SECTION IN USE BY JOB.
3		TIME SPENT IN READYQUE.
4		[6:6]
	[12:6]	NUMBER OF LAST CHUNK ASSIGNED TO JOB.
	[13:6]	NUMBER OF FIRST CHUNK ASSIGNED TO JOB.
	[24:6]	TYPE OF OBJECT PROGRAM: 1 = BASIC. 2 = ALGOL. 3 = COBOL. 4 = FORTRAN. 5 = TSPOL. 6 = XALGOL.
	[30:18]	CREATION DATE OF OBJECT FILE.
5	[1:23]	NUMBER OF FORCED SWAPS.
	[27:27]	NUMBER OF TIME SWAPS.
6	[1:23]	NUMBER OF DATA PRESENCE BIT INTERRUPTS.
	[24:24]	NUMBER OF CODE PRESENCE BIT INTERRUPTS.
7	[1:23]	NUMBER OF DATA OVERLAYS.
	[27:27]	NUMBER OF CODE OVERLAYS.
8		NUMBER OF JOBS REMAINING IN MIX AT JOB TERMINATION.
9		NUMBER OF SECONDARY CODE PRESENCE BIT INTERRUPTS.

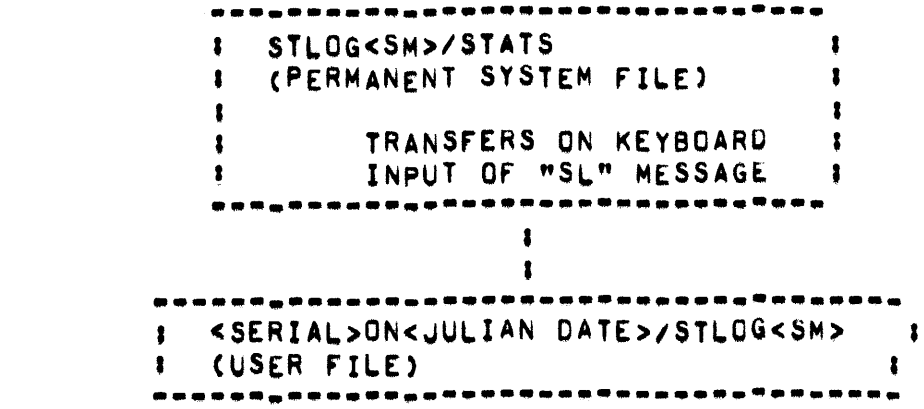
DETAILED MCP OPERATION.

BOTH PERMANENT AND TEMPORARY STATISTICS FILES ARE UPDATED AT SPECIFIED PERIODS BY THE MCP. THE SYSTEM STATISTICS FILE IS CONSTRUCTED FROM CORE STORAGE TABLES THAT ARE TRANSFERRED EVERY N-SECOND BY THE FOLLOWING PROCESS:



NSECOND CALLS FILLSYSTAT WHICH UPDATES THE DISK FILE AND ZEROS THE APPROPRIATE TABLES IN CORE AFTER THE UPDATE. FILLSYSTAT IS A TYPELESS, PARAMETERLESS, PROCEDURE.

THE PROGRAM STATISTICS FILE IS CONSTRUCTED FROM CORE STORAGE TABLES THAT ARE TRANSFERRED AT EACH "EOJ" BY THE FOLLOWING PROCESS:



THE ROUTINE PSEUDOLOGENTRY PERFORMS THE ACTION OF WRITING THE JOB INFORMATION OUT OF THE UV TABLE INTO A PSEUDO-LOG FILE.

WHEN SL IS ENTERED, SAVESTATISTICS IS CALLED TO PRODUCE THE USER FILE CONTAINING THE PROGRAM STATISTICS FILE.

TO FACILITATE THE CODING OF THE MCP STATISTICS CODE, CERTAIN DEFINITIONS WERE MADE. THE MAJOR ONES FOLLOW:

DEFINITION	MEANING
CODEPBITS[CODEPBITS1]	UV[CODEPBITS1,1]
DATAPBITS[DATAPBITS1]	UV[DATAPBITS1,2]
DATAOLAYS[DATAOLAYS1]	UV[DATAOLAYS1,3]
CODEOLAYS[CODEOLAYS1]	UV[CODEOLAYS1,4]
MORECPBITS[MORECPBITS1]	UV[MORECPBITS1,5]
PTSTAT[PTSTAT1]	UV[PTSTAT1,6]
IOTSTATE[IOTSTAT1]	UV[IOTSTAT1,7]
CORESTAT[CORESTAT1]	UV[CORESTAT1,8]
EOJTSTAT[EOJTSTAT1]	UVEOJTSTAT1,9]
BOJTSTAT[BOJTSTAT1]	UVBOJTSTAT1,10]
FIDSTAT[FIDSTAT1]	UV[FIDSTAT1,11]
MFIDSTAT[MFIDSTAT1]	UV[MFIDSTAT1,12]
JOBTYPE[JOBTYPE1]	UVJOBTYPE1,13]
MOREDPBITS[MOREDPBITS1]	UV[MOREDPBITS1,14]
OLAYUSED[OLAYUSED1]	UV[OLAYUSED1,15]
STLOGDB	STATWORD.[33:15]
SYSTATBASE	STATWORD.[13:20]
STATPOINTER	STATWORD.[1:1]
STATLOCK	STATWORD.[2:1]

```
PBUSY
COUNTUP(COUNTUP1,COUNTUP2)


COUNTUPBY(COUNTUPBY1,COUNTUPBY2)
```

```
STATWORD,[3:1]
BEGIN COUNTUPBY(COUNTUP1,1)
COUNTUPBY((COUNTUP1)+30,COUNTUP2)
END
COUNTARRAY[COUNTUPBY1]:=
*P(DUP)+(COUNTUPBY2)
```

SYLLABLE

CONTAINS THE CODE SYLLABLE CAUSING AN INTERRUPT.

SYSMAX

CONTAINS THE MAXIMUM NUMBER OF SYSTEMS THAT CAN BE CONNECTED TOGETHER IN A SPECIFIC SHAREDISK CONFIGURATION. SYSNO AND SYSMAX ARE PASSED TO THE MCP BY THE KERNEL DURING A HALT/LOAD.

```
SYSNO = M[0].[16:2]
SYSMAX = M[0].[14:2]
```

SYSNO

CONTAINS THE HARDWARE SYSTEM NUMBER. (SEE ALSO SYSMAX).

TABLEOFCONTENTS

TABLEOFCONTENTS(B) IS A PROCEDURE WHICH IS CALLED BY KEYIN TO LIST PSEUDO-DECKS ON THE SPO OR A TERMINAL.

TACH

THE TACH ROUTINE HANDLES THE ATTACHMENT AND DETACHMENT OF REMOTE B487 STATIONS TO AND FROM PROGRAMS. ITS PROCEDURE DECLARATION IS AS FOLLOWS:

```
PROCEDURE TACH(STA,MIX,FUNC);

    VALUE MIX,STA;
    REAL  STA,MIX,FUNC;
```

THE INTERPRETATION OF THE PARAMETERS IS AS FOLLOWS:

PARAMETER	CONTENTS	DESCRIPTION
FUNC	0	IF STA NEQ 0 AND MIX NEQ 0 THEN DETACH GIVEN STATION FROM GIVEN MIX IF STA NEQ 0 AND MIX = 0 THEN DETACH GIVEN STATION FROM ALL MIXES AND DETACH THE STATIONS SPO MESSAGES, IF ANY IF STA = 0 AND MIX NEQ 0 THEN DETACH ALL STATIONS FROM GIVEN MIX
	1	TEST TO SEE IF STA IS ATTACHED
	2	ATTACH STA
	3	TEST TO SEE IF STA AND IF NOT, THEN ATTACH
	4	TEST TO SEE IF ANY ATTACHED STA IS READ READY
	5	MIX SET TO P1MIX FROM COM5 IS STA = 0 THEN JOB GOES TO EOJ
STA	TU&BUF	,[9:9] FIELD

TACH IS RESPONSIBLE FOR MAINTAINING THE ATTACHED AND TUSTABABYMIX ARRAYS.

TAPEPARITYRETRY

TAEPARITYRETRY(R,U) IS A PROCEDURE WHICH HANDLES READ AND WRITE ERRORS ON TAPE. "R" IS THE RESULT DESCRIPTOR AND "U" IS THE LOGICAL UNIT NUMBER.

ON A WRITE PARITY THE MCP:

1. BACKSPACES OVER TWO RECORDS.
2. SPACES FORWARD OVER ONE RECORD.
3. ERASES IN A FORWARD DIRECTION (THE AMOUNT OF TAPE ERASED IS A FUNCTION OF THE PHYSICAL RECORD SIZE).
4. REWRITES THE RECORD.

THESE STEPS ARE REPEATED UNTIL A GOOD RECORD IS WRITTEN WITH NO PARITY OR UNTIL A PROGRAMMATIC COUNTER HAS BEEN EXCEEDED. THE FOLLOWING IS A CHART WHICH REPRESENTS THE AMOUNT OF TAPE (IN WORDS) ERASED ON CONSECUTIVE ATTEMPTS TO WRITE A PROPER RECORD. THE LAST NUMBER IN EACH COLUMN IS THE TOTAL AMOUNT OF TAPE ERASED (IN WORDS) WHEN YOU GET A "WR PARITY":

RETRY	10 WD BUF	100 WD BUF	1000 WD BUFF
1	10	100	1000
2	44	134	1034
3	134	402	3102
4	308	938	7238
5	660	2010	15510
6	1364	4154	32054
7	2772	8422	
8	5588	17018	
9	11220		
10	22484		

FLOW CHARTS OF TAPEPARITYRETRY FOR BOTH READ AND WRITE OPERATIONS APPEAR IN THE "ONLINE/MAINT" MANUAL.

TAPEPURGE

TAPEPURGE(BUFF) IS A PROCEDURE WHICH PURGES TAPE (WRITES A SCRATCH LABEL ON A TAPE). "BUFF" IS A DESCRIPTION WHICH POINTS TO THE KEYBOARD INPUT MESSAGE I.E. "PG MTA" OR "PG MTA=12345".

TERMINALMESSGE

TERMINALMESSAGE(N) IS A PROCEDURE WHICH SETS NT1 TO "N" AND CALLS TERMINALMESSAGEA(NT1).

TERMNALMESSAGEA

TERMINALMESSAGEA(N) IS A PROCEDURE WHICH CREATES A MESSAGE FOR RUN-TIME ERRORS EITHER FROM THE LIST "TERMINALMESSAGES" IF "N" IS POSITIVE, OR FROM THE ADDRESS OF "N" IF "N2 IS NEGATIVE.

TERMINATE

TERMINATE(MIX) IS A PROCEDURE WHICH MARKS A MIX INDEX "MIX" SO THAT IT WILL BE TERMINATED BY NSECOND FOR A RUN-TIME ERROR.

TISKTASK

TISKTASK IS A PROCEDURE WHICH HANDLES THE SCHEDULING OF A JOB WHOSE CODE FILE IS GIVEN BY <MFID>/<FID> PASSING TO IT A NUMBER OF TASK PARAMETERS GIVEN BY N/2. THE PARAMETERS OF TISKTASK ARE AS FOLLOWS:

MFID — MULTI FILE ID OF CODE FILE

FID — FILE ID OF CODE FILE

N — NUMBER OF "F=" PARAMETERS BETWEEN F=7 AND THE MSCW.

THERE WILL BE A PAIR OF F= CELLS FOR EACH TASK PARAMETER. F(=I) CONTAINS THE NAME OR VALUE OF THE PARAMETER.

F(=(I+1)) CONTAINS THE TYPE

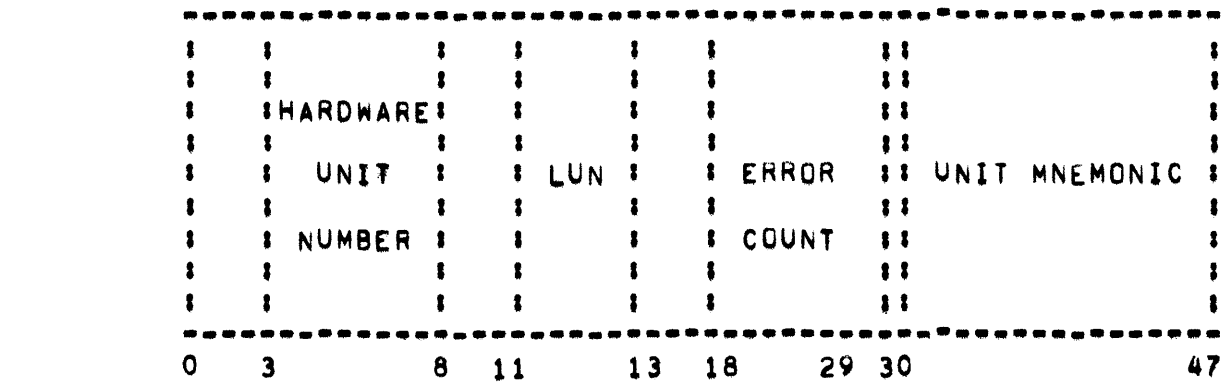
0	TASK ARRAY	NAME
1	EVENT,LOCK	NAME
2	PRT CELL	NAME
3	PRT CELL	VALUE
4	(SAVE)ARRAY	NAME
5	ARRAY	VALUE

TISKTASK MAKES A TEST FOR AGREEMENT BETWEEN THE TASK PARAMETERS DESCRIBED IN THE PARAMETER DESCRIPTION SEGMENT OF THE CODE FILE AND THOSE DESCRIBED BY THE F= CELLS. LACK OF AGREEMENT CAUSES THE PARENT TO TAKE THE ON EXCEPTION BRANCH (IF ANY HAS BEEN SPECIFIED IN THE COBOL PROGRAM).

TISKTASK COPIES THE CODE FILE, FILLING NAME AND VALUE PARAMETERS INTO THE NEW PRT AND WRITING OUT VALUE ARRAYS AS TYPE=2 SEGMENTS. THE JOB IS ENTERED IN THE SCHEDULE AND THE SCHEDULE-ID IS ENTERED IN THE TASK ARRAY. THE NEW SHEET ENTRY IS FLAGGED AS A GO JOB (AS FROM A COMPILE=AND=GO).

TINU

"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. THAT IS, INFORMATION ABOUT THE UNIT WITH LOGICAL UNIT NUMBER, "LUN", WOULD BE IN "TINU[LUN]". ALL ENTRIES IN "TINU" HAVE THE FOLLOWING FORMAT:



FIELD	CONTENTS	DESCRIPTION
[0:3]	0	NOT USED.
[3:5]		UNIT NUMBER RECOGNIZED BY HARDWARE.
[8:5]	0	NOT USED.
[11:7]		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 ZEROES EXCEPT IN THE BIT LOCATION CORRESPONDING TO THE "RRR" RESULT BIT LOCATION DESIGNATED FOR THE UNIT REPRESENTED BY "TINU[LUN]".
[18:12]		ERROR COUNT THIS UNIT
[30:18]	UNIT MNEMONIC	THREE CHARACTER ABBREVIATION FOR THE UNIT REPRESENTED BY "TINU[LUN]". (E.G., CRA.)

TOGLE

TOGLE IS A MCP VARIABLE WHICH CONTAINS THE FOLLOWING:

```
.[47:1]   HP2TOG                  HP2MASK          =       @1
.[46:1]   STATUSBIT               STATUSMASK       =       @2
.[45:1]   SHEETFREE               SHEETMASK        =       @4
.[44:1]   STACKUSE                STACKMASK        =      @10
.[43:1]   STOREDY                 STOREMASK        =      @20
.[42:1]   USERSPACEREADY          USERSPACEMASK    =      @40
.[41:1]   HOLDFREE                HOLDMASK         =     @100
.[40:1]   NSECONDREADY            NSECONDMASK      =     @200
.[39:1]   ABORTABLE               ABORTMASK        =     @400
.[38:1]   BUMPTUTIME              BUMPTUMASK       =    @1000
.[37:1]   KEYBOARDREADY           KEYBOARDMASK     =    @2000
.[36:1]   NOBACKTALK              NOBACKTALKMASK=      @4000
.[35:1]   QTRDY                   QTRDYMASK        =   @10000
.[34:1]   INTFREE                 FREEMASK         =   @20000
.[33:1]   SPOEDNULLOG
.[32:1]   REMOTELOGFREE           REMOTELOGMASK    =  @100000
.[31:1]   EGGSELECTSTOPPED
.[30:1]   STARTOG
.[29:1]   NINETEENNOTREADING      NINETEENMASK   =@1000000
.[28:1]   SMWSTOPPED              SMWSTOPPEDMASK=@2000000
.[27:1]   DCWAITING
.[26:1]   DCOPTSTOPPED
.[25:1]   INQUPTSTOPPED
.[24:1]   MCPFREE                 MCPMASK=@40000000
.[23:1]   SCRATCHDIRECTORYREADY   SCRATCHDIRECTORYMASK=@100000000
.[22:1]   FINDINGADDRESS
.[21:1]   CDFREE                  CDMASK=@400000000
.[15:6]   NOMEM
.[14:1]   BREAKTOG                BREAKMASK=@10000000000
.[13:1]   BCPTOG
.[12:1]   DCPLOAD                 DCPLOADMASK=@4000000000000
.[11:1]   DCPRUN
.[10:1]   WORKING
.[9:1]    SYSDISKTOG              SYSDISKMASK=@40000000000000
```

TRACETABLE1

DESCRIPTOR POINTING TO TRACETABLE1 ARRAY, AS DESCRIBED IN THE SECTION ON THE DEBUGGING FACILITY.

TRACETABLE2

DESCRIPTOR POINTING TO TRACETABLE2 ARRAY, AS DESCRIBED IN THE SECTION ON THE DEBUGGING FACILITY.

TRANSACTION

DESCRIPTOR POINTING TO THE TRANSACTION ARRAY.

TUMAX

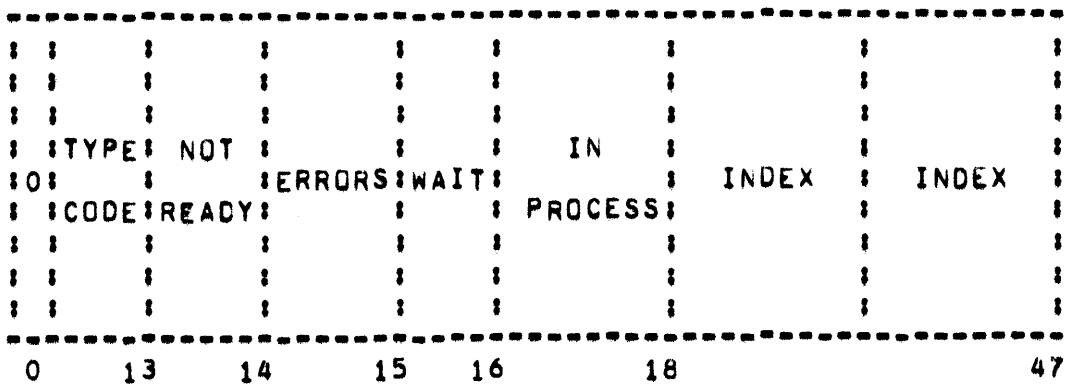
CONTAINS MAXIMUM TU NUMBER OF B487-S FOUND ATTACHED TO SYSTEM.

TUSTABABYMIX

DESCRIPTOR POINTING TO THE TUSTABABYMIX ARRAY, WHICH CONTAINS BY MIX INDEX THE NUMBER OF ATTACHED STATIONS, ON B487.

UNIT

ALL ENTRIES IN "UNIT" HAVE THE FOLLOWING FORMAT:



FIELD	CONTENTS	DESCRIPTION
[0:1]	0	FLAG BIT.
[1:4]	TYPE CODE	0 = CARD READER 1 = LINE PRINTER 2 = MAGNETIC TAPE 3 = DRUM 4 = DISK 5 = SPO 6 = CARD PUNCH
[5:8]	ERROR FIELD	ERROR FIELD OF LAST I/O DONE ON THIS UNIT. (USED BY I/O ERROR ROUTINE AS A RETRY COUNTER. ORIG SET TO @200 & INCRIMENTED BY 1 UNTIL GTR @210)
[13:1]	0/1	NOT READY BIT 0 = UNIT READY 1 = UNIT NOT READY
[14:1]	0/1	ERROR FLAG BIT 0 = NO ERRORS 1 = ERRORS
[15:1]	0/1	WAITING FOR I/O CHANNEL 1 = I/O AWAITING AN I/O CHANNEL
[16:2]	VARIES	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.
[18:15]	INDEX	INDEX OF FIRST I/O REQUEST FOR WHICH SERVICE IS NOT COMPLETE. @77777 IF NONE.
[33:15]	INDEX	INDEX OF LAST I/O REQUEST FOR WHICH SERVICE IS NOT COMPLETE.

UNITCODE

DESCRIPTOR POINTING TO THE UNITCODE ARRAY, CONTAINING USER CODES BY LOGICAL UNIT NUMBER.

UNITIN

UNITIN(TINU,WHAT) IS A PROCEDURE WHICH CALCULATES A LOGICAL UNIT NUMBER FROM THE ARRAY "TINU" AND THE THREE CHARACTER MNEMONIC "WHAT".

USERCODE

DESCRIPTOR POINTING TO THE USERCODE ARRAY, WHICH CONTAINS USER CODES BY MIX INDEX.

USERDISK

DESCRIPTOR POINTING TO THE USERDISK ARRAY.

USERDISKBOTTOM

POINTS TO FIRST SEGMENT OF USER DISK AVAILABLE TABLE.

USERSTA

THE MCP MAINTAINS AN ARRAY CALLED USERSTA WHICH CONTAINS ONE WORD 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 FOR 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; 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 BEGING BEING CHARGED FOR THE TIMES ASSIGNED TO THE PROGRAM, ETC.

THE MCP MAINTAINS REMOTE TERMINAL PROCESSOR TIME IN ASSIGNEDPROCESTIME [*], WHICH HAS ONE WORD FOR EACH PROGRAM IN THE MIX, SIMILARLY, I/O TIME IS MAINTAINED IN ASSIGNEDIOTIME[*] AND PRORATED TIME IS MAINTANED IN ASSIGNEDIDLETIME[*].

THE WAY IN WHICH A PROGRAM DESIGNATES THE ADDRESS TO BE PLACED IN USERSTA IS TO PERFORM EITHER A PASSIVE OR ACTIVE INTERROGATE STATEMENT REFERENCING THE STATION. IN ALGOL, THIS INVOLVES A STATEMENT OF THE FORM STATUS(TUBUFF,0) OR STATUS(TUBUFF,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 STATUSWORD. EACH TIME SUCH AN INTERROGATE IS PERFORMED, THE MCP CHECKS TO SEE IF THE TERMINAL BUFFER ADDRESS CURRENTLY IN THE PROGRAMS 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.

UV
--

DESCRIPTOR POINTING TO THE UV ARRAY WHICH CONTAINS UVSIZE COUNTERS PER MIX INDEX FOR STATISTICS PURPOSES.

WAITQUE

THE WATIQUE IS A QUEUE OF UNITS FOR WHICH THERE ARE REQUESTS BUT NO I/O CHANNEL IS AVAILABLE. "NEXTWAIT" AND "FIRSTWAIT" ARE POINTERS INTO THE WAITQUE. "NEXTWAIT" IS THE FIRST AVAILABLE SLOT IN THE WAITQUE AND "FIRSTWAIT" POINTS AT THE NEXT UNIT TO BE USED WHEN AN I/O BECOMES AVAILABLE.

WAITIO

WAITIO(IOD,MASK,U) IS A PROCEDURE WHICH INITIATES AND WAITS FOR COMPLETION OF I/O DESCRIPTOR "IOD" ON LOGICAL UNIT "U". "MASK" DETERMINES WHICH ERROR CONDITIONS ARE TO BE HANDLED, AND WHICH RETURNED TO THE CALLING ROUTINE. THOSE BITS TURNED ON IN "MASK" SIGNIFY THE ERROR CONDITIONS TO BE HANDLED BY OTHER MCP PROCEDURES. THOSE BITS TURNED OFF IN "MASK" WILL BE HANDLED BY THE ROUTINE CALLING "WAITIO". FOR MOD III I/O CHANNELS WITH MAG TAPE, A DOUBLE LENGTH MASK OF TEN OCTAL DIGITS IS QUITE OFTEN USED.

WATER

CONTAINS THE HEAD OF THE LIST USED BY EGGTIMER FACILITY;

WEEKDAY

CONTAINS THE NAME OF THE CURRENT DAY OF THE WEEK, RIGHT-JUSTIFIED, MINUS THE SYLLABLE "DAY".

WHATSUP

WHATSUP IS A PROCEDURE WHICH IS CONCERNED WITH THE INTERROGATION OF THE CONTENTS OF THE STATION ARRAY. THE PROCEDURE DECLARATION IS AS FOLLOWS:

```
REAL PROCEDURE WHATSUP(STA,MIX,FUNC,ARAE);

     VALUE STA,MIX,FUNC;
     REAL  STA,MIX,FUNC;
     ARRAY ARAE;
```

WHATSUP, WHEN CALLED FROM KEYIN, HANDLES THE FOLLOWING CASES:

WHATSUP CALL	INPUT FROM KEYBOARD	TYPED OUTPUT FROM WHATSUP
WS(X,0,1,X)	WU TU-BU	USERID FOR TU-BU
WS(0,X,1,X)	<MIX> WU	ALL STATIONS AND USERIDS FOR <MIX>
WS(0,0,1,X)	WU	ALL STATIONS AND USERIDS
WS(X,0,2,X)	WP TU-BU	PROGRAM SPEC. ASSIGNED TO TU-BU
WS(0,0,2,X)	WP	ALL PROGRAM SPEC, ASSIGNED TO ALL TU-BU
WS(0,X,2,X)	<MIX> WA	ALL STATIONS ASSIGNED TO <MIX>
WS(0,X,3,X)	<MIX> SS:MESSAGE	SENDS MESSAGE TO ALL STATIONS THAT ASKED FOR <MIX> MESSAGES
WS(0,0,3,X)	SS ALL:MESSAGE	SENDS MESSAGE TO ALL READY SPO"S
WS(0,X,4,X)	<MIX> SS ALL:MESSAGE	SENDS MESSAGE TO ALL READY SPO-S ON THAT MIX

WHATTODO

VARIABLE USED BY ESPBIT TO DETERMINE INFORMATION CONCERNING ABSENT MCP ROUTINES.

WHYSLEEP

WHYSLEEP(MASK) IS A PROCEDURE WHICH CHECKS IF A JOB IS ASLEEP WAITING FOR SPO INPUT, AND THE LEGALITY OF THAT INPUT.

XCLOCK

CONTAINS TIME OF DAY IN 60THS OF A SECOND.

ZIPPER

ZIPPER(W1,W2,USERSTA) IS A PROCEDURE WHICH EXECUTES THE PROGRAM "W1"/"W2".

- ZIPPER -

TABLE OF CONTENTS

PREFACE . . . . . PAGE 1
INTRODUCTION. . . . . PAGE 2
ABORT TABLE . . . . . PAGE 3
ABORTFILE . . . . . PAGE 5
ABORTSPEC . . . . . PAGE 5
ACTDATE . . . . . PAGE 5
ACTUALIOERR . . . . . PAGE 5
ARRAY INFORMATION TABLE . . . . . PAGE 6
ARRAY SPACE DECLARATION . . . . . PAGE 8
ATTACHED. . . . . PAGE 9
AUXCODE . . . . . PAGE 9
AUXDATA . . . . . PAGE 9
AUXILIARY MEMORY AND RESOURCE ALLOCATION. . . . . PAGE 10
    ABSTRACT . . . . . PAGE 10
    INTRODUCTION . . . . . PAGE 10
    GENERAL DIFFERENCES. . . . . PAGE 13
    COBOL DIFFERENCES. . . . . PAGE 14
    ALGOL DIFFERENCES. . . . . PAGE 16
    MCP REFERENCE DOCUMENT . . . . . PAGE 18
        NARRATIVE DESCRIPTION . . . . . PAGE 18
        TECHNICAL INFORMATION . . . . . PAGE 20
    CONCLUSION . . . . . PAGE 26
AVAIL . . . . . PAGE 27
AVAILABLE DISK TABLE. . . . . PAGE 28
BACKWATER . . . . . PAGE 35
BED[*]. . . . . PAGE 36
BREAKOUT/RESTART. . . . . PAGE 40
    INTRODUCTION . . . . . PAGE 40
    BREAKOUT . . . . . PAGE 41
    RESTART. . . . . PAGE 45
BUFFER (NORMAL) . . . . . PAGE 55
BUFFER (PRINTER BACK UP). . . . . PAGE 56
BUILDLABEL. . . . . PAGE 57
BYPASS. . . . . PAGE 57
CCMASK1 . . . . . PAGE 58
CCMASK2 . . . . . PAGE 59
CCTOG . . . . . PAGE 60
CHANNEL . . . . . PAGE 60
CHECK . . . . . PAGE 60
CIDROW FORMAT . . . . . PAGE 61
CLOCK . . . . . PAGE 62
COMMUNICATES. . . . . PAGE 63
COMPLEXSNOOZE . . . . . PAGE 76
COM11 . . . . . PAGE 77
COM13 . . . . . PAGE 78
COM19 . . . . . PAGE 78
COM23 . . . . . PAGE 78
COM5. . . . . PAGE 78
CORE. . . . . PAGE 79

COUNTARRY . . . . . PAGE 79
CTABLE. . . . . PAGE 79
DALOC . . . . . PAGE 80
DATE. . . . . PAGE 83
DBADR . . . . . PAGE 83
DBARRAY . . . . . PAGE 83
DCQARA. . . . . PAGE 83
DCQPTSTACK. . . . . PAGE 83
DC19Q . . . . . PAGE 84
DEBUGGING FACILITIES. . . . . PAGE 86
GENERAL. . . . . PAGE 86
MEMORY DUMP. . . . . PAGE 86
SPO ACCESS TO MEMORY.. . . . . PAGE 86
DEBUGGING STATEMENTS . . . . . PAGE 87
DEBUGGING STATEMENT.. . . . . PAGE 88
MODE PART.. . . . . PAGE 88
CALL STATEMENT. . . . . PAGE 88
DELETE CALL STATEMENT.. . . . . PAGE 88
DISPLAY STATEMENT.. . . . . PAGE 88
REPLACE STATEMENT.. . . . . PAGE 88
DEFINE STATEMENT. . . . . PAGE 88
PRINT SYMBOLS STATEMENT.. . . . . PAGE 89
EXIT. . . . . PAGE 89
EXPRESSION. . . . . PAGE 89
PRIMARIES.. . . . . PAGE 89
CONSOLE ACCESS TO DISK.. . . . . PAGE 90
TRACE. . . . . PAGE 90
STORING.. . . . . PAGE 90
PRINTING THE TRACED DATA. . . . . PAGE 91
DECKREMOVER . . . . . PAGE 92
DIRECT. . . . . PAGE 92
DIRECTORY - BYPASS. . . . . PAGE 93
DIRECTORY - FILE HEADER . . . . . PAGE 95
DIRECTORY - MAIN. . . . . PAGE 97
DIRECTORY - SCRATCH (SHAREDISK) . . . . . PAGE 98
DIRECTORYFREE . . . . . PAGE 102
DIRECTORYTOP. . . . . PAGE 103
DIRECTORYTOP - AUXILIARY. . . . . PAGE 105
DIRECTORYTOP + 1 (SHAREDISK). . . . . PAGE 106
DIRECTORYTOP + 2 (SHAREDISK). . . . . PAGE 106
DIRECTORYTOP + 3 (SHAREDISK). . . . . PAGE 106
DISK ORGANIZATION . . . . . PAGE 107
DISKADDRESS . . . . . PAGE 110
DISKBOTTOM. . . . . PAGE 110
DISKERROR . . . . . PAGE 110
DISKIO. . . . . PAGE 110
DISKOUNT. . . . . PAGE 110
DISKWAITIME . . . . . PAGE 110
EGGCLK. . . . . PAGE 111
EGGENTER. . . . . PAGE 112
EGGREMOVE . . . . . PAGE 112

EGGSELECT . . . . . PAGE 112
ENDOFDECK . . . . . PAGE 114
ENTERCONTROLDECK. . . . . PAGE 114
ENTERUSERFILE . . . . . PAGE 114
ESPBIT. . . . . PAGE 115
ESPCOUNT. . . . . PAGE 116
ESPDISKBOTTOM . . . . . PAGE 116
ESPDISKTOP. . . . . PAGE 116
ESPTAB. . . . . PAGE 116
EUIO. . . . . PAGE 116
EUQ . . . . . PAGE 116
EUW . . . . . PAGE 116
FETCH . . . . . PAGE 117
FILE INFORMATION BLOCK. . . . . PAGE 118
FILE PARAMETER BLOCK. . . . . PAGE 124
FILE TANK . . . . . PAGE 130
FILE SECURITY . . . . . PAGE 131
LEVELS OF SECURITY.. . . . . PAGE 131
IDENTIFIED USER. . . . . PAGE 131
USER AND FILE CLASSIFICATION.. . . . . PAGE 132
MCP ACTIONS. . . . . PAGE 133
USER CONTROL CARD. . . . . PAGE 134
LIBRARY AND SECURITY FILE MAINTENANCE. . . . . PAGE 135
FORMAT OF SECURITY FILE ENTRIES. . . . . PAGE 139
DISK FILE HEADER FORMAT - FILE SECURITY. . . . . PAGE 140
REMOTE USER FILE SECURITY. . . . . PAGE 141
USER CODE AND AUTHENTICATION CODE.. . . . . PAGE 141
LOG-IN PROCEDURE FOR DATA COMMUNICATIONS. . . . . PAGE 142
REMOTE/USERS DISK FILE. . . . . PAGE 143
CCMASK1, CCMASK2, MIXMASK, AND INFOMASK . . . . . PAGE 146
FILECLOSE . . . . . PAGE 150
FILEOPEN. . . . . PAGE 150
FINALQUE. . . . . PAGE 150
FINDINPUT . . . . . PAGE 150
FINDOUTPUT. . . . . PAGE 150
FIRSTDECK . . . . . PAGE 151
FIRSTWAIT . . . . . PAGE 151
FORGETESPDISK . . . . . PAGE 152
FORGETSPACE . . . . . PAGE 152
FORGETUSERDISK. . . . . PAGE 152
FORMESS . . . . . PAGE 152
FS. . . . . PAGE 153
GET . . . . . PAGE 154
GETESPDISK. . . . . PAGE 154
GETSPACE. . . . . PAGE 154
GETUSERDISK . . . . . PAGE 155
GRSD. . . . . PAGE 155
HOLDER. . . . . PAGE 156
HOLDLIST. . . . . PAGE 156
ILL . . . . . PAGE 157
INDENPENDENTRUNNER. . . . . PAGE 158

INFOMASK1 . . . . . PAGE 160
INFOMASK2 . . . . . PAGE 162
INITIALIZE. . . . . PAGE 163
INTERRUPT . . . . . PAGE 163
INQCT . . . . . PAGE 163
INQUIRY . . . . . PAGE 164
INTABLE . . . . . PAGE 166
INTER-PROGRAM COMMUNICATION . . . . . PAGE 167
INTERRUPT HANDLING. . . . . PAGE 170
    INTRODUCTION . . . . . PAGE 170
    METHODS OF MAKING INFORMATION PRESENT. . . . . PAGE 171
        MAKING THE DATA PRESENT FOR THE FIRST TIME. . . . PAGE 171
        MAKING OVERLAID DATA PRESENT. . . . . PAGE 171
        MAKING PROGRAM SEGMENTS PRESENT.. . . . . PAGE 172
        CONTROL SECTION OF MCP. . . . . PAGE 172
        NOTHINGTODO ROUTINE . . . . . PAGE 173
    A NOTE ON PARALLEL PROCESSING. . . . . PAGE 176
    INTERRUPT HANDLING IN THE MCP. . . . . PAGE 178
    DETAILED INTERRUPT EXAMPLE . . . . . PAGE 180
INTERVAL. . . . . PAGE 182
INTRNSC . . . . . PAGE 183
INTSIZE . . . . . PAGE 184
I/O ERROR MESSAGES. . . . . PAGE 185
IOFINISH. . . . . PAGE 190
IOMASK. . . . . PAGE 190
I/O QUEUE . . . . . PAGE 191
IOQUEAVAIL. . . . . PAGE 192
IOREQUEST . . . . . PAGE 192
IOTIME. . . . . PAGE 192
ISTACK. . . . . PAGE 192
JAR . . . . . PAGE 193
JOB INITIATION. . . . . PAGE 195
    CONTROL CARD PROCEDURES. . . . . PAGE 195
    SELECTION PROCEDURE. . . . . PAGE 195
    RUN PROCEDURE. . . . . PAGE 196
    MIX INDEX. . . . . PAGE 196
    INITIATE ROUTINE . . . . . PAGE 196
    P1MIX AND P2MIX. . . . . PAGE 196
JOBNUM. . . . . PAGE 198
JUNK. . . . . PAGE 198
KEYBOARD INPUT MESSAGES . . . . . PAGE 199
KEYBOARDCOUNTER . . . . . PAGE 224
KEYIN . . . . . PAGE 224
KILL. . . . . PAGE 224
KLUDGE. . . . . PAGE 224
KLUMP . . . . . PAGE 224
LABELTABLE. . . . . PAGE 225
LABEL EQUATION TABLE. . . . . PAGE 226
LABEL RECORD. . . . . PAGE 228
LABEL WORD. . . . . PAGE 230
LABELASCRATCH . . . . . PAGE 231

LASTCDNUM . . . . . PAGE 231
LASTDECK. . . . . PAGE 231
LASTEU. . . . . PAGE 231
LBMESS. . . . . PAGE 231
LEFTOFF . . . . . PAGE 231
LIBRARY MAINTENANCE . . . . . PAGE 232
LOCATQUE. . . . . PAGE 235
LOGARRAY. . . . . PAGE 236
LOGENTRY. . . . . PAGE 236
LOGFREE . . . . . PAGE 236
LOGHOLDER . . . . . PAGE 236
LOGGING . . . . . PAGE 237
SYSTEM LOG . . . . . PAGE 237
REMOTE LOGGING . . . . . PAGE 243
LOGICAL UNIT NUMBERS. . . . . PAGE 252
LOOKQ . . . . . PAGE 253
LSLATE. . . . . PAGE 254
LQAVAIL . . . . . PAGE 254
LQUE. . . . . PAGE 254
MAKEPRESENT . . . . . PAGE 255
MAINTLOGARRAY . . . . . PAGE 255
MCP . . . . . PAGE 255
MCPBASE . . . . . PAGE 255
MCPTOP. . . . . PAGE 255
MDELTA. . . . . PAGE 256
MLOG. . . . . PAGE 256
MEMASK. . . . . PAGE 256
MEMORY. . . . . PAGE 256
MEMORY CONTENTS - CELLS 0-200 (OCTAL) . . . . . PAGE 257
MEMORY LAYOUT . . . . . PAGE 260
MEMORY LINKS. . . . . PAGE 264
MEMORY ORGANIZATION . . . . . PAGE 269
MEND. . . . . PAGE 270
MESS. . . . . PAGE 270
MESSAGEHOLDER . . . . . PAGE 271
MESSAGEWRITER . . . . . PAGE 272
MIXMASK . . . . . PAGE 272
MIXNUM. . . . . PAGE 272
MSTART. . . . . PAGE 272
MTXIN . . . . . PAGE 272
MULTITABLE. . . . . PAGE 272
NAMEID. . . . . PAGE 273
NEUP. . . . . PAGE 273
NEXTSLOT. . . . . PAGE 273
NEXTWAIT. . . . . PAGE 273
NFO . . . . . PAGE 274
NOPROCESSTOG. . . . . PAGE 275
NSECOND . . . . . PAGE 275
NSLATE. . . . . PAGE 275
NT1 - NT7 . . . . . PAGE 275
NUMAINTMESS . . . . . PAGE 276

NUMESS. . . . . PAGE 276
NXDISK. . . . . PAGE 276
OBJECT PROGRAM I/O FACILITIES . . . . . PAGE 277
I/O INTRINSICS . . . . . PAGE 277
SPECIFICATION OF FILE HANDLING TECHNIQUES. . . . . PAGE 277
FILE AND FILE NAME.. . . . . PAGE 277
FILE PARAMETER BLOCK.. . . . . PAGE 278
FILE INFORMATION BLOCKS. . . . . PAGE 279
LOGICAL UNIT NUMBERS.. . . . . PAGE 279
FILE NAMES VS I/O UNITS. . . . . PAGE 279
OPENING A FILE.. . . . . PAGE 280
BUFFER AREA ACCESSED BY OBJECT PROGRAMS. . . . . PAGE 281
COMMUNICATE OPERATIONS.. . . . . PAGE 281
PERFORMANCE OF I/O BY MCP. . . . . PAGE 282
PROGRAM RELEASE PROCEDURE. . . . . PAGE 282
CONTINUITY BIT ROUTINE AND PRINTER BACKUP PROCEDURE. . . PAGE 282
IOREQUEST PROCEDURE. . . . . PAGE 283
INITIATEIO PROCEDURE.. . . . . PAGE 283
IOFINISH PROCEDURE.. . . . . PAGE 283
INTERROGATING PERIPHERAL UNITS.. . . . . PAGE 284
STATUS PROCEDURE.. . . . . PAGE 284
OLAY. . . . . PAGE 286
OLAYMASK. . . . . PAGE 286
OLDIDLETIME . . . . . PAGE 286
ONEOHONE. . . . . PAGE 286
ONEOHTWO. . . . . PAGE 286
OPTION WORD . . . . . PAGE 287
ORR WORD. . . . . PAGE 289
PBCOUNT . . . . . PAGE 290
PBIO. . . . . PAGE 290
PBLOG . . . . . PAGE 290
PEUIO . . . . . PAGE 290
PINGO . . . . . PAGE 290
PICKTHELOCK . . . . . PAGE 291
POINTERS. . . . . PAGE 292
PRNTABLE. . . . . PAGE 293
PROCTIME. . . . . PAGE 293
PRT[*,*]. . . . . PAGE 294
PRT (FOR COBOL) . . . . . PAGE 297
PRT (FOR COBOL68) . . . . . PAGE 299
PRT (FOR FORTRAN) . . . . . PAGE 300
PRT (FOR THE MCP) . . . . . PAGE 301
PROCTIME. . . . . PAGE 302
PRT . . . . . PAGE 302
PRYOR . . . . . PAGE 302
PSEUDOCOPY. . . . . PAGE 302
PUNT. . . . . PAGE 303
PUNTER. . . . . PAGE 304
PURGEIT . . . . . PAGE 304
P1MIX . . . . . PAGE 304
P2MIX . . . . . PAGE 304

QTIMES. . . . . PAGE 305
QVLEFT. . . . . PAGE 305
QV. . . . . PAGE 305
RDCTABLE. . . . . PAGE 306
READERA . . . . . PAGE 307
READERB . . . . . PAGE 307
READFROMDISK. . . . . PAGE 308
READQ . . . . . PAGE 309
READY . . . . . PAGE 309
REMOTEFILE. . . . . PAGE 309
REMOTELOGGER. . . . . PAGE 310
REMOVETHEDECK . . . . . PAGE 311
RE-ENTRANT CODE HANDLING. . . . . PAGE 312
REPLY . . . . . PAGE 328
RESTARTING. . . . . PAGE 328
RESULT1 . . . . . PAGE 328
RESULT2 . . . . . PAGE 328
RESULT3 . . . . . PAGE 328
RESULT4 . . . . . PAGE 328
REWINDANDLOCK . . . . . PAGE 329
RQONE . . . . . PAGE 329
RQTWO . . . . . PAGE 329
RRNCOUNT. . . . . PAGE 329
RRRMECH . . . . . PAGE 329
RUN . . . . . PAGE 329
RUNUMBER. . . . . PAGE 330
SAVERESULT. . . . . PAGE 331
SAVEWORD. . . . . PAGE 331
SHEDULEIDS. . . . . PAGE 331
SCN . . . . . PAGE 331
SECONDCTR . . . . . PAGE 331
SECURITYCHECK . . . . . PAGE 332
SECURITYMAINT . . . . . PAGE 333
SEEKNAM . . . . . PAGE 333
SEGMENT DICTIONARY. . . . . PAGE 334
SEGMENT ZERO (DISK) . . . . . PAGE 340
SEGMENT ZERO (PROGRAM FILES). . . . . PAGE 342
SEGMENT ZERO (RESTART FILES). . . . . PAGE 343
SELECTION . . . . . PAGE 344
SELECTRUN . . . . . PAGE 344
SELECTRUN1. . . . . PAGE 344
SETNOTINUSE . . . . . PAGE 344
SHAREDISK INFORMATION . . . . . PAGE 345
INTRODUCTION . . . . . PAGE 345
FPM - CONTENTS OF MEMORY WORD. . . . . PAGE 348
DISK I/O OPERATORS . . . . . PAGE 349
READ. . . . . PAGE 350
READ/LOCK . . . . . PAGE 350
WRITE . . . . . PAGE 351
WRITE/UNLOCK. . . . . PAGE 351
CLEAR ALL CONTENTION BITS OF SYSTEM "N" . . . . . PAGE 352

UNLOCK ALL ADDRESSES OF SYSTEM "N". . . . . PAGE 352
CLEAR CONTENTION BIT OF SYSTEM "N" IN ADDRESS "X" . PAGE 353
UNLOCK ADDRESS "X". . . . . PAGE 353
REPORT FREE ADDRESS (1) . . . . . PAGE 353
REPORT FREE ADDRESS (2) . . . . . PAGE 354
OPERATION OF THE SHAREDISK SYSTEM. . . . . PAGE 355
MCP . . . . . PAGE 355
INTRINSICS. . . . . PAGE 355
PRINTER BACKUP DISK . . . . . PAGE 355
LOAD CONTROL. . . . . PAGE 356
SYSTEM LOG. . . . . PAGE 356
COLD START. . . . . PAGE 356
TAPE TO DISK LOADER . . . . . PAGE 357
DISK TO DISK. . . . . PAGE 357
THE HALT LOAD KERNEL. . . . . PAGE 357
THE COOL START DECK . . . . . PAGE 358
FPM - MAINTENANCE KIT INTERPRETATION . . . . . PAGE 359
LOCAL - REMOTE. . . . . PAGE 359
SINGLE PULSE - RUN. . . . . PAGE 359
START . . . . . PAGE 359
CLEAR . . . . . PAGE 359
SETWRDS - SCANSW - NORMAL . . . . . PAGE 359
COOLMRS - NORMAL. . . . . PAGE 360
CO1LMRS - NORMAL. . . . . PAGE 360
CO2LMRS - NORMAL. . . . . PAGE 360
CO3LMRS - NORMAL. . . . . PAGE 360
V (VARIANT REGISTER). . . . . PAGE 361
N REGISTER. . . . . PAGE 361
WDC . . . . . PAGE 362
LOCKED. . . . . PAGE 362
ADR1. . . . . PAGE 362
LOCK. . . . . PAGE 362
ENDOP . . . . . PAGE 363
LKSET . . . . . PAGE 363
ENDOVER . . . . . PAGE 363
LMWDA . . . . . PAGE 363
LMAL. . . . . PAGE 363
SC. . . . . PAGE 363
LMCP (COUNTER). . . . . PAGE 363
LMCP (FLIP FLOP). . . . . PAGE 363
1MC . . . . . PAGE 363
ADRFILL . . . . . PAGE 364
ADREND. . . . . PAGE 364
ENDFLO. . . . . PAGE 364
FLOSET. . . . . PAGE 364
LMOP. . . . . PAGE 364
STOP + 1. . . . . PAGE 364
STOP. . . . . PAGE 364
MC ≠ ADR. . . . . PAGE 364
SRP1, SRP2, AND SRP3. . . . . PAGE 364
ULOK. . . . . PAGE 365

ADR2, ADR3, ADR4, AND ADR5. . . . . PAGE 365
DFC - INTERPRETATION OF INDICATORS . . . . . PAGE 366
I/O DESCRIPTOR AND ADDRESS WORD FORMATS. . . . . PAGE 367
HARDWARE REQUIREMENTS FOR A SHAREDISK SYSTEM . . . . . PAGE 368
SHEET . . . . . PAGE 370
SHEETDIDDLER. . . . . PAGE 372
SIGNOFF . . . . . PAGE 372
SLATE[*]. . . . . PAGE 373
SLEEP . . . . . PAGE 375
SNOOZE. . . . . PAGE 376
SOFTI . . . . . PAGE 377
SPACECTR. . . . . PAGE 377
SPOUT . . . . . PAGE 378
SPOWORD . . . . . PAGE 379
STARTADECK. . . . . PAGE 380
STATIONMESSAGEHOLDER. . . . . PAGE 381
STATION TABLE . . . . . PAGE 382
STATUS. . . . . PAGE 384
STATWORD. . . . . PAGE 384
STOPJOB . . . . . PAGE 384
SVRESULT. . . . . PAGE 384
SYSTEM STATISTICS FACILITY. . . . . PAGE 385
STATISTICS LOG . . . . . PAGE 385
GENERAL CHARACTERISTICS. . . . . PAGE 385
OPERATION. . . . . PAGE 387
FILE DESCRIPTIONS. . . . . PAGE 388
SYSTEM STATISTICS FILE. . . . . PAGE 388
STATISTICS LOG FILE.. . . . . PAGE 390
DETAILED MCP OPERATION.. . . . . PAGE 392
SYLLABLE. . . . . PAGE 395
SYSMAX. . . . . PAGE 395
SYSNO . . . . . PAGE 395
TABLEOFCONTENTS . . . . . PAGE 396
TACH. . . . . PAGE 397
TAPEPARITYRETRY . . . . . PAGE 398
TAPEPURGE . . . . . PAGE 399
TERMINALMESSGE. . . . . PAGE 399
TERMNALMESSAGEA . . . . . PAGE 399
TERMINATE . . . . . PAGE 399
TISKTASK. . . . . PAGE 400
TINU. . . . . PAGE 401
TOGLE . . . . . PAGE 402
TRACETABLE1 . . . . . PAGE 403
TRACETABLE2 . . . . . PAGE 403
TRANSACTION . . . . . PAGE 403
TUMAX . . . . . PAGE 403
TUSTABABYMIX. . . . . PAGE 403
UNIT. . . . . PAGE 404
UNITCODE. . . . . PAGE 406
UNITIN. . . . . PAGE 406
USERCODE. . . . . PAGE 406

USERDISK. . . . . PAGE 406
USERDISKBOTTOM. . . . . PAGE 406
USERSTA . . . . . PAGE 407
UV. . . . . PAGE 408
WAITQUE . . . . . PAGE 409
WAITIO. . . . . PAGE 409
WATER . . . . . PAGE 409
WEEKDAY . . . . . PAGE 409
WHATSUP . . . . . PAGE 410
WHATTODO. . . . . PAGE 412
WHYSLEEP. . . . . PAGE 412
XCLOCK. . . . . PAGE 413
ZIPPER. . . . . PAGE 414