Burroughs

B 6700

MASTER CONTROL PROGRAM

INFORMATION MANUAL

inted in U.S. America

11-70

5000086

LIST OF EFFECTIVE PAGES

NOTE: Insert latest changed page; dispose of superseded pages.

TOTAL NUMBER OF PAGES IN THIS MANUAL IS 212 CONSISTING OF THE FOLLOWING:

Page No.	Issue	Page No.	Issue
TITLE	ORIGINAL		
LIST OF EFFECTIVE PAGES	ORIGINAL		
CONTENTS	ORIGINAL		
ILLUSTRATIONS	ORIGINAL		
1 thru 204	ORIGINAL		

Burroughs Corporation believes the program described 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 the use of this material. The information contained herein is subject to change. Revisions may be issued to advise of such changes and/or additions.

B6700 MASTER CONTROL PROGRAM

TABLE OF CONTENTS

ILLUSTRATIONS

PREFACE

THIS DOCUMENT DESCRIBES THE FUNCTION AND STRUCTURE OF THE B6700 MASTER CONTROL PROGRAM (MCP). SECTIONS 2 AND 3 DESCRIBE CONTROL FUNCTIONS WHILE SECTIONS 4 AND 5 ARE MORE CONCERNED WITH INPUT AND OUTPUT. SECTION 6 DESCRIBES A NUMBER OF UTILITY FUNCTIONS PROVIDED AS PART OF THE B6700 SYSTEM SOFTWARE FOR THE CONVENIENCE OF THE USER.

THE READER SHOULD HAVE SOME ACQUAINTANCE WITH THE B6700 MACHINE HARDWARE AND THE B6700 ESPOL LANGUAGE.

SECTION 1

INTRODUCTION

1. INTRODUCTION

1.1. FUNCTION OF THE MCP

THE PRIMARY FUNCTION OF THE B6700 MASTER CONTROL PROGRAM IS TO INCREASE THE EFFICIENCY OF THE B6700 INFORMATION PROCESSING SYSTEM. BY INTEGRATING USERS OBJECT PROGRAMS WITH THE SOFTWARE-COMPATIBLE B6700 HARDWARE SYSTEM AND HIGH SPEED DISK, THE MCP OPTIMIZES THE PRODUCTIVITY OF THE B6700 INFORMATION PROCESSING SYSTEM. THE MCP IS, THEREFORE, AN ESSENTIAL PART OF THE PROCESSING ENVIRONMENT OF THE B6700 SYSTEM.

1.1.1. INTEGRATIVE ACTION OF THE MCP

THE INTEGRATIVE ACTION OF THE B6700 MCP IS ACHIEVED IN THREE WAYS:

1. BY ITS CAPABILITY OF COORDINATING THE EXECUTION OF MANY PROGRAMS OR JOBS IN THE PROCESSOR OR PROCESSORS.

2. BY ITS CAPABILITY OF CONTROLLING BOTH INPUT AND OUTPUT SO AS TO MAKE OPTIMAL USE OF THE RELATIVELY SLOW PERIPHERAL DEVICES.

3. BY ITS CAPABILITY OF TAKING EXECUTIVE ACTION TO MINIMIZE THE ADVERSE AFFECTS OF SYSTEM DEGRADATION.

1.1.2. RATE OF PROCESSING AND THE MCP

THE OVERALL RATE AT WHICH JOBS CAN BE PROCESSED IS INCREASED IN THE FOLLOWING THREE WAYS:

1. BY INCREASING THE SPEED OF EXECUTION OF INDIVIDUAL USER PROGRAMS. THIS CAN BE ACHIEVED BY THE USE OF A COMBINATION OF SEVERAL FACILITIES, AS FOLLOWS:

   A. PARALLEL PROCESSING WITH THE INTRODUCTION OF A SECOND

PROCESSOR.

B. MULTIPROGRAMMING - THE RUNNING OF SEVERAL JOBS CONCURRENTLY. THE MCP MAINTAINS A LIST OF JOBS READY TO RUN ORDERED BY PRIORITY. WHEN A RUNNING JOB MUST FOR EXAMPLE WAIT FOR AN INPUT OR OUTPUT OPERATION TO BE COMPLETED THE MCP WILL START UP THE NEXT JOB IN THE READY LIST, AND RESTART THE ORIGINAL JOB WHEN THE I/O OPERATION HAS BEEN COMPLETED AND NO JOBS OF HIGHER PRIORITY ARE IN THE READY LIST.

C. RE-ENTRANT CODE - WHEREBY A SINGLE COPY OF A ROUTINE IN MEMORY CAN BE SHARED BY SEVERAL PROGRAMS.

D. TASKING - WHEREBY FAMILIES OF TASKS ARE PROCESSED IN A COORDINATED MANNER.

2. BY INCREASING THE SPEED OF DATA HANDLING. FOR THIS PURPOSE TWO FACILITIES ARE PROVIDED IN THE UTILITY SECTION OF THE MCP.

A. LOADCONTROL - THIS ENABLES CARD INPUT TO BE TRANSFERRED TO DISK OR TAPE. THE CARD IMAGE FILES SO FORMED ARE ASSIGNED TO DIFFERENT PSEUDO CARD READERS WHICH ARE THEN TREATED BY THE MCP AS IF THEY WERE REAL PHYSICAL CARD READERS.

B. PRINTER AND PUNCH BACKUP - THIS ENABLES OUTPUT TO BE PLACED ON TAPE OR DISK AND THEN PRINTED OR PUNCHED OUT AT A LATER, MORE CONVENIENT TIME.

3. BY INCREASING THE EASE OF OPERATING THE MACHINE. SIMPLE ENGLISH-LIKE OPERATOR ATTENTION AND ERROR MESSAGES, AUTOMATIC ASSIGNMENT OF LABELED FILES TO JOBS WITHOUT OPERATOR INTERVENTION, A SIMPLIFIED CONTROL CARD LANGUAGE AND OTHER FEATURES HELP INCREASE THROUGHPUT ON THE B6700 SYSTEM.

1.2. THE MCP AND ESPOL

THE B6700 MASTER CONTROL PROGRAM IS WRITTEN IN THE B6700 ESPOL LANGUAGE. B6700 ESPOL IS AN EXTENSION OF B6700 ALGOL AND IS DESCRIBED IN THE B6700 ESPOL INFORMATION MANUAL.

SECTION 2

MACHINE - MCP INTERACTION

2. MACHINE-MCP INTERACTION

2.1. SYSTEM INITIALIZATION AND INITIALIZATION FUNCTION

THE LOADER FUNCTION IS USED TO LOAD THE MCP CODE FILE FROM TAPE OR DISK TO DISK ADDRESS ZERO. FROM THIS ADDRESS THE HARDWARE "DISK LOAD SELECT" FUNCTION CAN PLACE THE MCP IN CONTROL OF THE SYSTEM.

SYNTAX:

```
<SYSTEM INITIALIZATION FUNCTION> ::= <DATE FUNCTION> /
     <SCREENS FUNCTION> / <OPTION FUNCTION> /
     <OVERLAY FUNCTION> / <DIRECTORY FUNCTION> /
     <LOAD FUNCTION> / <TERMINAL FUNCTION>
```

SEMANTICS:

1. THE SYSTEM INITIALIZATION FUNCTIONS ARE PUNCHED IN EBCDIC ON CARDS IN FREE FORMAT.

2. SELECTED CARDS ARE PLACED AT THE END OF THE SYSTEM/UTILITY FILE DECK

3. THE LOAD FUNCTION IS CONTAINED IN SYSTEM/UTILITY.

2.1.1. DATE.FUNCTION

SYNTAX:

```
<DATE FUNCTION> ::= DATE <MONTH> <SLASH> <DAY> <SLASH> <YEAR>
<MONTH> ::= <DIGIT> <DIGIT> / < DIGIT>
<DAY> ::= <DIGIT> <DIGIT> / <DIGIT>
<YEAR> ::= <DIGIT> <DIGIT> / <DIGIT>
```

SEMANTICS

THE YEAR PART OF THE DATE IS THE YEAR MODULO 100.

EXAMPLE:

```
     DATE  8/14/70
```

2.1.2. SCREENS FUNCTION

SYNTAX:

```
<SCREENS FUNCTIONS> ::= SCREENS : <CHANNEL DESIGNATE LIST>
<CHANNEL DESIGNATE LIST> ::= <CHANNEL DESIGNATE> /
     <CHANNEL DESIGNATE LIST> : <CHANNEL DESIGNATE>
<CHANNEL DESIGNATE> ::= CHANNEL <CHANNEL NO.> ON <CONN LIST> :
     <DISPLAY LIST>
<CONN LIST> ::= <CONN NO.> / <CONN LIST>, <CONN NO.>
<DISPLAY LIST> ::= <DISPLAY ID> / <DISPLAY LIST>, <DISPLAY ID>
<DISPLAY ID> ::= MIX / LP / CR / CP / MT / DK / SC
<CONN NO> ::= ≤DECIMAL INTEGER BETWEEN 1 AND MAXIMUM NUMBER OF
     GIVEN DISPLAY UNIT≥
```

SEMANTICS:

1. CHANNEL NUMBER REFERS TO A PSEUDO CHANNEL.

2. THE CONN NUMBER IS A UNIT DESIGNATE FOR A SUPERVISORY CONSOLE.

3. THE DISPLAY LIST IS A LIST OF IDENTIFIERS.

EXAMPLE:

```
SCREENS : CHANNEL 1 ON 1 : MIX
```

2.1.3. OVERLAY FUNCTION

SYNTAX:

```
OVERLAY FUNCTION> ::= OLAYROW <ROW SIZE>
<ROW SIZE> ::= <INTEGER>
```

SEMANTICS:

1. ROW SIZE SPECIFIES THE NUMBER OF SEGMENTS TO BE PLACED IN TH INITIAL ROW OF THE MCP OVERLAY FILE.

2. THE INCLUSION OF AN OVERLAY CARD WILL FORCE A COLD START. A COL START REMOVES ALL DISK DIRECTORY INFORMATION. NOTE: THE RESIDEN MCP STILL HAS ACCESS TO ITSELF.

EXAMPLE:

```
OLAYROW 400
```

2.1.4. DIRECTORY FUNCTION

SYNTAX:

```
<DIRECTORY FUNCTION> ::= <DIRECTORY LOCATION> / <DIRECTORY ROW>
<DIRECTORY LOCATION> ::= DIRECTORYLOC <ELECTRONIC UNIT NO.>:
     <SEGMENT NO.>
<DIRECTORY ROW> ::= DIRECTORYROW <ROW SIZE>
```

SEMANTICS:

1. THE DIRECTORYLOC CARD WILL FORCE A COLD START.

2.1.5. LOAD FUNCTION

SYNTAX:

```
<LOAD FUNCTION> ::= <LOAD TAPE> / <LOAD DISK>
<LOAD TAPE> ::= LOAD <FILE NAME> FROM <TAPE LABEL>
<LOAD DISK> ::= LOAD <FILE NAME> DISK
```

EXAMPLES:

```
LOAD SYSTEM/MCP FROM SYSTEM
LOAD SYSTEM/MCP DISK
```

2.1.6. TERMINAL FUNCTION

SYNTAX:

```
<TERMINAL FUNCTION> ::= STOP /*
```

SEMANTICS:

1. THE STOP CARD CAUSES ANY CARDS FOLLOWING IT TO BE FLUSHED AND IGNORED UNTIL AN END CARD IS ENCOUNTERED. CARDS WITH THE FUNCTIONS REQUIRED ARE PLACED BEFORE THE STOP CARD AND THE REMAINDER AFTER IT. IN THIS WAY THE INTEGRITY OF THE DECK OF CARDS CAN BE MAINTAINED.

2. THE ASTERISK FUNCTION * TRANSFERS CONTROL TO THE OTHER PERIPHERAL DEVICE (CARD OR SPO). SCANNING WILL BE RESUMED WHERE IT LEFT OFF, I. E., JUST BEYOND THE * WHEN CONTROL IS RETURNED.

2.1.7. OPTION FUNCTION

SYNTAX:

```
<OPTION FUNCTION> ::= <SETTING> <OPTION WORD>
<SETTING> ::= <SET> / <RESET>
<OPTION WORD> ::= <OPTIONING> / <OPTION PAIR>
<OPTIONING> ::= <OPEN> / <RET> / <TERMINATE> / <SEGMENT> / <ALL>
<OPTION PAIR> ::= <OPTION> <DIGIT> <DIGIT> / <OPTION> <DIGIT>
```

SEMANTICS:

1. <OPEN> WHEN SET RETURNS A FILE OPEN MESSAGE ON THE SPO.

2. <RET> WHEN SET RETURNS A MESSAGE ON THE SPO IF TAPES WITH WRITE RINGS ARE NOT TO BE PURGED. E.G., THE EXPIRATION DATE HAS NOT YET OCCURRED.

3. <TERMINATE> WHEN SET GIVES SINGLE PROGRAM DUMPS. WHEN TERMINATE IS RESET A FULL CORE DUMP IS OBTAINED ON PROGRAM ABORT. NOTE: AFTER A COLD START TERMINATE IS SET AUTOMATICALLY.

4. IF <SEGMENT> IS SET THE MCP WILL SEGMENT LARGE ARRAYS. IF IT IS RESET NO SEGMENTATION WILL OCCUR.

5. <ALL> ACTIVATES ALL THE OPTIONS WHEN SET.

6. THE <OPTION> FUNCTION ENABLES AN OPTION TO BE SPECIFIED BY THE NUMBER FOLLOWING IT. THE NUMBER SHOULD BE BETWEEN 0 AND 47.

EXAMPLE:

```
SET TERMINATE
```

ESPOL CODE FILE ON DISK, BEFORE TRANSFER TO MEMORY:

	MSCW	SEGMENT DESCRIPTOR FOR SAVE1 CODE	LENGTH OF MCP CODE FILE	PCW TO INTERRUPT PROCEDURE	MEMORY ARRAY DESCRIPTOR	SAVE PROCEDURE CODE	SAVE ARRAY DATA		SAVE1 CODE	REST OF MCP
WORD #:	0	1	2	3	4	5···				

RESIDENT MCP IMMEDIATELY AFTER TRANSFER TO MEMORY:

	MSCW	SEGMENT DESCRIPTOR FOR SAVE1 CODE	DISK ADDR. OF MCP INFO TABLE *	PCW TO INTERRUPT PROCEDURE	MEMORY ARRAY DESCRIPTOR	SAVE PROCEDURE CODE	SAVE ARRAY DATA		MSCW	IRW TO WORD 3	P1	P2
WORD #:	0	1	2	3	4	5···			8191			

INFORMATION ON DISK AFTER TRANSFER OF CODE FILE TO MEMORY:

MCP	MCP INFO TABLE	FIRST ROW OVERLAY DISK

* (LATER IT WILL BE THE STACK VECTOR DESCRIPTOR)

FIGURE F2-1. ESPOL CODE FILE DURING INITIALIZATION OF MCP

2.2. HARDWARE INTERRUPTS

2.2.1. INTRODUCTION

THE INTERRUPT HANDLING MECHANISM OF THE MCP DEALS WITH TWO CLASSES OF INTERRUPTS: HARDWARE INTERRUPTS AND SOFTWARE INTERRUPTS. THE HARDWARE INTERRUPTS ARE GENERATED AUTOMATICALLY BY THE B6700 SYSTEM AND ARE HANDLED BY THE MCP INTERRUPT PROCEDURE. SOFTWARE INTERRUPTS ARE PROGRAMMATICALLY DEFINED FOR USE BY THE MCP AND OBJECT PROGRAM PROCESSES. SOFTWARE INTERRUPTS ALLOW PROCESSES TO COMMUNICATE WITH EACH OTHER AND WITH THE MCP.

THE B6700 PROCESSOR HARDWARE INTERRUPT SYSTEM IS THE PRIMARY INTERFACE BETWEEN THE MCP AND THE HARDWARE. BECAUSE OF THE IMPORTANCE OF THIS INTERFACE, THE RELEVANT FEATURES OF THE B6700 PROCESSOR WILL BE DESCRIBED ALONG WITH THE DISCUSSION OF INTERRUPT HANDLING.

AN INTERRUPT IS A MEANS OF DISCONTINUING A PROCESS SUBJECT TO THE OCCURRENCE OF CERTAIN CONDITIONS. IN ORDER TO FULLY UNDERSTAND THE OPERATION OF B6700 INTERRUPTS, AN UNDERSTANDING OF THE CONDITION "CONTROL STATE" IS REQUIRED.

2.2.2. CONTROL AND NORMAL STATE.

THE B6700 PROCESSOR MAY OPERATE IN ONE OF TWO DISTINCT STATES: NORMAL STATE OR CONTROL STATE. THE PRIMARY DIFFERENCE BETWEEN NORMAL STATE AND CONTROL STATE IS THAT EXTERNAL INTERRUPTS ARE DISABLED WHILE A PROCESSOR IS IN CONTROL STATE. ALSO, THERE ARE CERTAIN OPERATORS, SUCH AS SOME FORMS OF "SCNO" (SCAN OUT) WHICH CAN ONLY BE EXECUTED BY A PROCESSOR IN CONTROL STATE.

A PROCESSOR IN NORMAL STATE MAY ENTER CONTROL STATE BY EXECUTING A "DEXI" (DISABLE EXTERNAL INTERRUPTS) INSTRUCTION OR BY ENTERING OR EXITING TO A CONTROL STATE PROCEDURE. RECIPROCALLY, A PROCESSOR IN

CONTROL STATE MAY ENTER NORMAL STATE BY EITHER EXECUTING AN "EEXI" (ENABLE EXTERNAL INTERRUPTS) INSTRUCTION OR BY ENTERING OR EXITING TO A NORMAL STATE PROCEDURE.

IT SHOULD BE NOTED THAT WHILE IN CONTROL STATE, A PROCESSOR CAN SELECTIVELY MASK OUT ANY OR ALL MPX (I/O MULTIPLEXOR) INTERRUPTS BEFORE EXECUTING AN EEXI. THE RESULT IS A PROCESSOR IN NORMAL STATE WHICH DOES NOT RECEIVE THE MASKED MPX INTERRUPTS.

2.2.3. HARDWARE INTERRUPTS AND STACK STRUCTURE.

WHEN AN INTERRUPT CONDITION OCCURS, THE INTERRUPTED PROCESSOR MARKS THE STACK, AND INSERTS THREE WORDS IN THE TOP OF THE STACK. THESE THREE WORDS ARE FIRST THE IRW (INDIRECT REFERENCE WORD) POINTING TO D[0]+3, FOLLOWED BY TWO INTERRUPT PARAMETERS, P1 AND P2, WHICH CONTAIN INFORMATION INDICATING THE NATURE OF THE INTERRUPT CONDITION. IT IS EXPECTED THAT D[0]+3 WILL CONTAIN A PCW (PROGRAM CONTROL WORD) POINTING TO THE MCP HARDWARE INTERRUPT PROCEDURE: HOWEVER, AN IRW OR IRW CHAIN POINTING TO A PCW IS A LEGITIMATE CONDITION. THE PROCEDURE POINTED TO BY THE PCW IS NOW ENTERED AND P1 AND P2 ARE PASSED TO IT AS PARAMETERS. WHEN THE PROCESSOR ENTERS THE MCP HARDWARE INTERRUPT PROCEDURE, IT ENTERS CONTROL STATE SO THAT EXTERNAL INTERRUPTS ARE DISABLED. THIS IS ACCOMPLISHED BY GENERATING THE INTERRUPT PCW WITH THE CONTROL BIT ON. UPON ENTRY TO THE HARDWARE INTERRUPT PROCEDURE, THE PARAMETER P1 IS ANALYZED TO DETERMINE THE TYPE OF INTERRUPT WHICH OCCURRED. FOR SOME INTERRUPTS, SUCH AS PRESENCE-BIT INTERRUPTS, P2 CONTAINS ADDITIONAL INFORMATION TO BE USED BY THE INTERRUPT PROCEDURE.

THE ACTION TO BE TAKEN FOR EACH KIND OF INTERRUPT IS DESCRIBED IN THE FOLLOWING SECTIONS. THE DESCRIPTION COVERS THE THREE CLASSES OF HARDWARE INTERRUPTS:

1. SYLLABLE DEPENDENT INTERRUPTS.
2. ALARM INTERRUPTS.
3. EXTERNAL INTERRUPTS.

THE STACK STRUCTURE PRIOR TO CALLING THE INTERRUPT PROCEDURE IS SHOWN IN FIGURE F2-2.

IF THE PROCESSING OF THE INTERRUPT IS EXPECTED TO BE TIME CONSUMING, E.G. AN I/O ERROR, AN INDEPENDENT PROCESS IS ACTIVATED TO DO IT, THEREBY QUICKLY FREEING THE "MAIN" PROCESS FOR FURTHER EXECUTION.

IN ALL CASES, TOWARDS THE END OF HARDWARE INTERRUPT PROCESSING THE MCP PROCEDURE GEORGE IS CALLED.

BEFORE EXITING BACK TO THE INTERRUPTED PROCESS, GEORGE PERFORMS THE PROCEDURES ASSOCIATED WITH ANY SOFTWARE INTERRUPTS FOR THE CURRENT STACK THAT ARE NAMED IN THE SOFTWARE INTERRUPT QUEUE, PROVIDED THOSE INTERRUPTS ARE STILL ENABLED. IF THEY ARE NOT ENABLED, THE REQUESTS FOR THEIR PROCEDURES ARE MERELY DELETED.

AFTER ENTERING THE INTERRUPT PROCEDURE, THE PROGRAM BASE REGISTER IS POINTING AT THE INTERRUPT PROCEDURE, PIR AND PSR ARE POINTING AT THE INTERRUPT PROCEDURE ENTRY POINT AND THE RETURN CONTROL WORD FOR THE INTERRUPT PROCEDURES EXIT IS POINTING BACK TO THE OBJECT PROGRAMS CODE, AS SHOWN IN FIGURE F2-3.

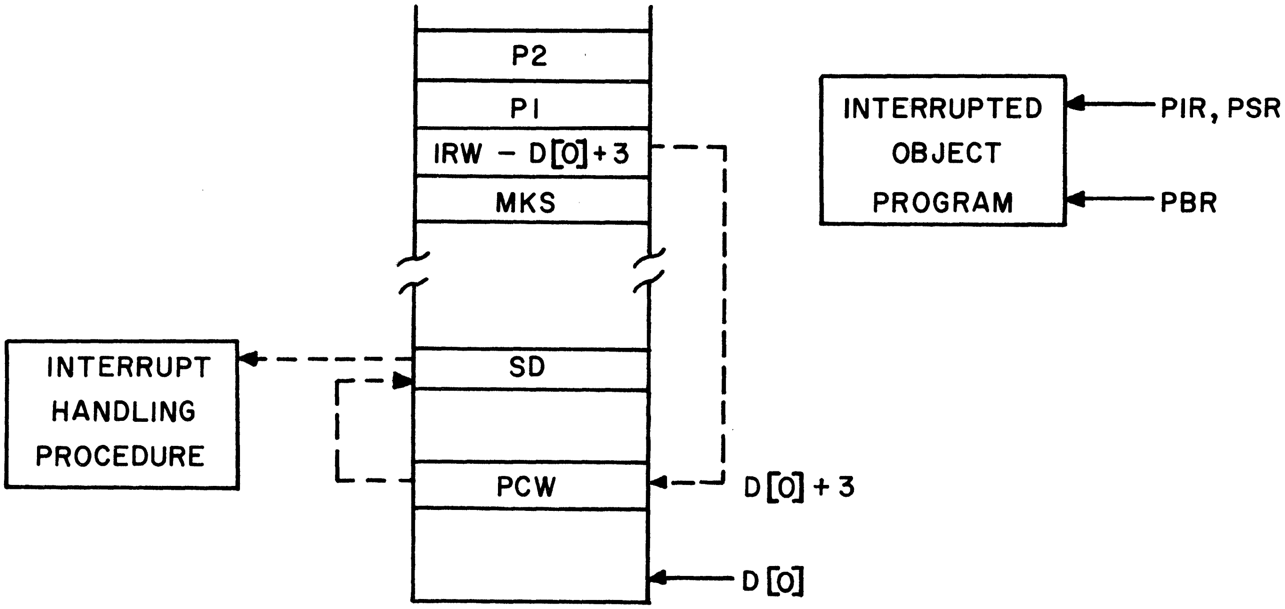


FIGURE F2-2. STACK PRIOR TO INTERRUPT PROCEDURE ENTRY

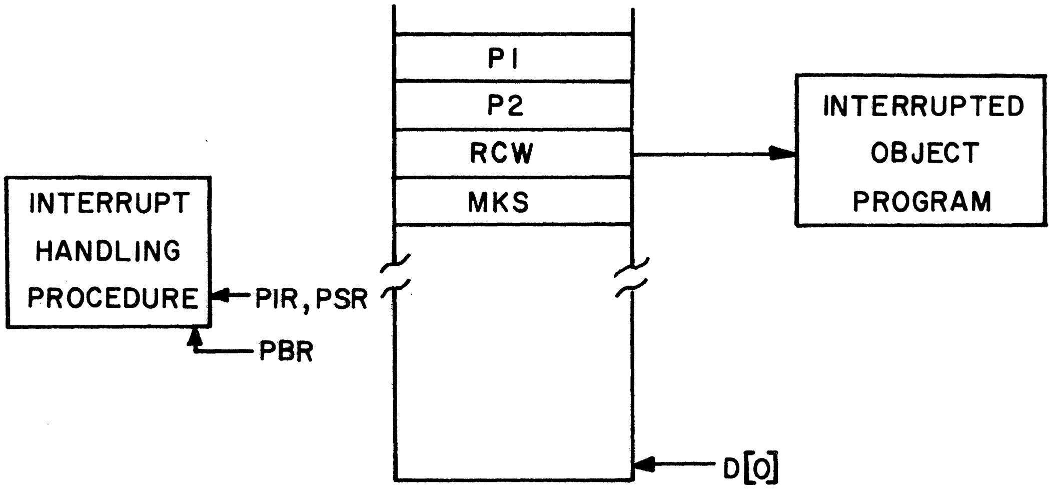


FIGURE F2-3. STACK FOLLOWING INTERRUPT PROCEDURE ENTRY

2.2.3.1. SYLLABLE DEPENDENT INTERRUPTS

SYLLABLE DEPENDENT INTERRUPTS ARE DETECTED BY THE PROCESSOR OPERATOR LOGIC. THERE ARE EIGHT TYPES OF SYLLABLE DEPENDENT INTERRUPTS:

(1) ARITHMETIC ERROR:

THIS GROUP OF INTERRUPTS INCLUDES THE DIVIDE-BY-ZERO, EXPONENT OVERFLOW, EXPONENT UNDERFLOW, INVALID INDEX, AND INTEGER OVERFLOW INTERRUPTS. NORMAL MCP ACTION IN RESPONSE TO THIS GROUP OF INTERRUPTS IS TO TERMINATE THE PROGRAM WHICH INCURRED THE INTERRUPT. MEANS ARE AVAILABLE TO ALLOW A USER PROGRAM TO OVERRIDE THIS ACTION AND RETAIN CONTROL.

(2) PRESENCE BIT:

THIS INTERRUPT OCCURS WHEN THE PROCESSOR ACCESSES A DATA DESCRIPTOR OR SEGMENT DESCRIPTOR WITH THE "PRESENCE BIT" OFF, INDICATING THAT WHATEVER THE DESCRIPTOR REFERENCES IS NOT PRESENT IN MEMORY. ON DETECTING A PRESENCE BIT INTERRUPT THE PROCEDURE "PRESENCEBIT" IS CALLED BY THE INTERRUPT PROCEDURE (SEE SECTION 2-3).

(3) MEMORY PROTECT:

THIS INTERRUPT OCCURS WHEN THE PROCESSOR ATTEMPTS TO WRITE IN A MEMORY LOCATION THAT CURRENTLY HAS THE MEMORY PROTECT BIT OF THE TAG-FIELD ON. IN RESPONSE TO THIS INTERRUPT, THE MCP WILL TERMINATE THE PROGRAM WHICH GENERATED THE INTERRUPT.

(4) BOTTOM OF STACK:

THIS INTERRUPT INDICATES THAT THE MARK STACK CONTROL WORD AT EXIT TIME TRIED TO POINT BELOW THE BOTTOM OF THE STACK. THIS INTERRUPT INDICATES A HARDWARE OR SOFTWARE ERROR. IN RESPONSE TO THIS INTERRUPT, THE MCP WILL TERMINATE THE PROGRAM WHICH GENERATED THE INTERRUPT.

(5) SEQUENCE ERROR:

THIS INTERRUPT INDICATES THAT AN INDIRECT REFERENCE HAS ENCOUNTERED AN INVALID CONDITION OR REFERENCE SEQUENCE (E.G. THE "F" REGISTER NOT POINTING TO A MSCW). THIS INTERRUPT INDICATES A HARDWARE OR SYSTEMS SOFTWARE ERROR. IN RESPONSE TO THIS INTERRUPT, THE MCP WILL TERMINATE THE PROGRAM WHICH GENERATED THE INTERRUPT.

(6) SEGMENTED ARRAY:

THE OCCURRENCE OF THIS INTERRUPT INDICATES THAT THE MCP HAS SEGMENTED AN ARRAY ROW WHEN ALLOCATING STORAGE FOR IT AND HAS JUST ATTEMPTED TO INDEX BEYOND THE END OF THE CURRENT SEGMENT. THE MCP INTERRUPT PROCEDURE MAKES THE NEXT SEGMENT PRESENT AND CONTINUES EXECUTING THE PROCESS.

(7) PROGRAMMED OPERATOR:

THIS INTERRUPT INDICATES THAT THE CURRENT OR ACTIVE STACK HAS ATTEMPTED TO EXECUTE AN OPERATOR CODE WHICH IS NOT CURRENTLY ASSIGNED. IT ALLOWS THE MCP TO SIMULATE THE OPERATOR PROGRAMMATICALLY, IF DESIRED. CURRENTLY, THE PROCESS IS TERMINATED.

(8) INVALID OPERAND:

THIS INTERRUPT OCCURS WHEN THE PROCESSOR ATTEMPTS TO EXECUTE A VALID OPERATOR ON DATA WHICH IS INVALID FOR THAT OPERATOR. IT RESULTS IN TERMINATION OF THE PROGRAM.

2.2.3.2. ALARM INTERRUPTS

THESE INTERRUPT CONDITIONS ARE NOT NORMALLY ANTICIPATED BY THE PROCESSOR OPERATOR LOGIC. THEY SERVE TO INFORM THE PROCESSOR OF SOME DETRIMENTAL CHANGE IN ENVIRONMENT AND CAN RESULT FROM HARDWARE FAILURE AS WELL AS PROGRAMMING ERRORS. THEY ALL RESULT IN TERMINATION OF THE PROCESS INVOLVED. THERE ARE SEVEN AS FOLLOWS:

(1) LOOP:

THIS INTERRUPT OCCURS WHEN THE PROCESSOR HAS EXPENDED AT LEAST TWO SECONDS IN THE EXECUTION OF ONE OPERATOR.

(2) MEMORY PARITY:

THIS INTERRUPT INDICATES A FAULTY READ FROM MEMORY.

(3) SCAN BUS PARITY:

THIS INTERRUPT INDICATES FAULTY RECEPTION OF DATA FROM THE SCAN BUS.

(4) STACK UNDERFLOW:

THIS INTERRUPT OCCURS WHEN THE S REGISTER CONTAINS A VALUE EQUAL TO OR LESS THAN THAT CONTAINED IN THE CURRENT F REGISTER. THIS WOULD HAVE THE EFFECT OF THE PROCESSOR ATTEMPTING TO HAVE ACCESS BELOW THE CURRENT STACK.

(5) INVALID ADDRESS:

THIS INTERRUPT INDICATES THAT THE PROCESSOR ATTEMPTED TO ADDRESS A MEMORY ADDRESS WHICH IS NOT AVAILABLE TO THE SYSTEM. THE MEMORY MODULE MAY NOT EXIST OR IT MAY BE INOPERATIVE.

(6) INVALID PROGRAM WORD:

THIS INTERRUPT INDICATED THAT THE PROCESSOR HAS ENCOUNTERED A WORD WHICH IS SUPPOSED TO BE A PROGRAM INSTRUCTION WORD BUT IS IN FACT NOT.

(7) STACK OVERFLOW:

THIS INTERRUPT OCCURS WHEN THE PROCESS STACK HAS EXCEEDED ITS ORIGINAL MEMORY SPACE ALLOCATION. PRESENTLY, THIS RESULTS IN A TERMINATION OF THE PROGRAM. IN THE FUTURE, HOWEVER, IT IS EXPECTED THAT THE HARDWARE

INTERRUPT PROCEDURE WILL FIND MORE SPACE FOR THE STACK, MOVE THE STACK TO THE NEW SPACE AND RESUME EXECUTION OF THE PROCESS.

2.2.3.3. EXTERNAL INTERRUPTS

EXTERNAL INTERRUPTS ARE LIKE THE ALARM INTERRUPTS IN THAT THEY ARE NOT ANTICIPATED BY THE OPERATOR LOGIC. HOWEVER, THEY DO NOT NORMALLY REQUIRE IMMEDIATE ACTION AND DO NOT NECESSARILY RESULT IN TERMINATION OF THE PROGRAM. AS MENTIONED ABOVE, NONE OF THE EXTERNAL INTERRUPTS CAN INTERRUPT A PROCESSOR IN CONTROL STATE EXCEPT FOR THE STACK OVERFLOW INTERRUPT. THERE ARE THREE EXTERNAL INTERRUPTS TYPES:

(1) INTERVAL TIMER:

THIS INTERRUPT OCCURS AFTER THE PERIOD OF TIME "SET" ON THE INTERVAL TIMER FOR A PROCESSOR. IF THE TIMER IS "RESET", AN INTERRUPT WILL NOT OCCUR. THIS INTERRUPT IS USED BY THE MCP TO DISTRIBUTE PROCESSOR EXECUTION TIME AMONG THE PROCESSES ACCORDING TO THEIR CURRENT PRIORITIES.

(2) PROCESSOR TO PROCESSOR:

THIS INTERRUPT OCCURS WHEN ONE PROCESSOR EXECUTES THE "HEYU" OPERATOR, WHICH ENABLES ONE PROCESSOR TO INTERRUPT A SECOND PROCESSOR EXCEPT IF IT IS RUNNING IN CONTROL STATE. IF A PROCESSOR IS IN CONTROL STATE, THE INTERRUPT IS HELD IN ABEYANCE UNTIL IT ATTEMPTS TO RESUME NORMAL STATE PROCESSING.

(3) MPX:

THIS INTERRUPT GROUP INCLUDES I/O FINISH, MULTILINE CONTROL (MLC), GCA, EXTERNAL MCP AND CHANGE OF PERIPHERAL UNIT STATUS INTERRUPTS. THESE INTERRUPTS OCCUR WHEN A MULTIPLEXOR WISHES TO COMMUNICATE WITH A PROCESSOR. THEY ARE HANDLED IN VARIOUS WAYS DEPENDING ON THE SPECIFIC TYPE.

A. WHEN AN I/O FINISH INTERRUPT OCCURS THE MCP INTERRUPT HANDLING

PROCEDURE CALLS THE I/O FINISH PROCEDURE, WHICH CHECKS FOR ERRORS WHICH MAY HAVE OCCURRED. IF NO ERROR IS FOUND, I/O FINISH INITIATES A NEW I/O (IF THE I/O REQUEST QUEUES ARE NOT EMPTY). THERE ARE TWO QUEUE STRUCTURES RELATED TO THE I/O OPERATIONS: THE WAITCHANNELQUES, ONE FOR EACH MULTIPLEXOR AND THE UNITQUES, ONE FOR EACH UNIT. WHEN THE I/O FINISH PROCEDURE INITIATES ANOTHER I/O, IT FIRST CHECKS THE WAITCHANNELQUE OF THE MULTIPLEXOR IT HAS JUST FINISHED WITH AND INITIATES THE FIRST I/O REQUEST IN THAT QUEUE. IT THEN CHECKS THE UNITQUE FOR THE UNIT IT HAS JUST USED, REMOVES THE TOP ENTRY FROM THAT QUEUE AND INSERTS IT IN THE WAITCHANNELQUE.

IN ORDER TO PREVENT CONFUSION, THE WAITCHANNELQUES ARE NOT ALLOWED TO CONTAIN MORE THAN ONE I/O REQUEST FOR ANY GIVEN UNIT. IF AN I/O REQUEST OCCURS FOR A UNIT THAT IS ALREADY IN A WAITCHANNELQUE (FOR ANY MULTIPLEXOR) THEN THE REQUEST IS ENTERED IN THE APPROPRIATE UNITQUE.

B. THE MLC (MULTILINE CONTROL) INTERRUPTS INDICATE THAT SOMETHING LIKE A DATA COMMUNICATIONS SYSTEM WISHES TO COMMUNICATE TO THE PROCESSOR THROUGH A WORD INTERFACE OF THE MULTIPLEXOR. THE WAY THIS INTERRUPT IS HANDLED DEPENDS ON THE NATURE OF THE DEVICE WHICH IS ATTEMPTING TO COMMUNICATE TO THE PROCESSOR. AT PRESENT THE PROCESSOR IS CAPABLE OF DISTINGUISHING FOUR DIFFERENT MLC INTERRUPTS, SINCE THERE CAN BE FOUR MULTILINE CONTROLS. HOWEVER, THE SIGNIFICANCE OF THE VARIOUS MLC INTERRUPTS HAS NOT BEEN DEFINED AND WILL PROBABLY VARY DEPENDING ON THE PARTICULAR TYPE OF INSTALLATION.

C. GCA (GENERAL CONTROL ADAPTER) INTERRUPTS INDICATE THAT SOME SORT OF SPECIAL CONTROL DEVICE (AN ANALOG DEVICE, A PLOTTER, OR SOME MACHINE THAT THE COMPUTER IS CONTROLLING) WISHES TO COMMUNICATE TO THE PROCESSOR. SINCE THERE IS ONLY ONE GCA INTERRUPT, IT IS CLEAR THAT ONLY ONE SUCH DEVICE CAN BE HANDLED AT A TIME. IT IS ALSO EVIDENT THAT THE HANDLING OF THIS INTERRUPT IS DEPENDENT ON THE NATURE OF THE DEVICE IN QUESTION.

D. WHEN A MULTIPLEXOR IS ATTACHED TO THE WORD INTERFACE OF ONE OF THE

SYSTEM MULTIPLEXORS, IT BECOMES NECESSARY TO HANDLE INTERRUPTS FROM THE "EXTERNAL" MULTIPLEXOR. THIS IS THE FUNCTION OF THE EXTERNAL MPX INTERRUPT, WHICH INDICATES THAT THE PROCESSOR MUST FIRST INTERROGATE THE EXTERNAL MULTIPLEXOR TO DETERMINE THE NATURE OF THE MPX INTERRUPT.

E. A CHANGE OF PERIPHERAL STATUS INTERRUPT INDICATES THAT A DEVICE HAS JUST CHANGED STATE. THE SYSTEM DETERMINES WHAT THIS DEVICE IS AND THEN TAKES THE APPROPRIATE ACTION E.G., CHANGE OF STATUS ON THE SPO CAUSES THE MCP TO INITIATE A READ REQUEST.

2.3. STORAGE CONTROL.

2.3.1. DYNAMIC STORAGE ALLOCATION.

THE B6700 MCP PERFORMS DYNAMIC STORAGE ALLOCATION FOR ALL SYSTEM STORAGE MEDIA: MAIN MEMORY, MAGNETIC DISK AND SYSTEM LIBRARY MAGNETIC TAPE. THE MCP CONTROLS ALLOCATION AND DEALLOCATION OF ALL SYSTEM MEMORY, CONSIDERING THE DIFFERENT SYSTEM STORAGE MEDIA AS A HIERARCHY OF MEMORY.

THE MCP DYNAMICALLY ALLOCATES THE USE OF MAIN MEMORY AS A RESOURCE AMONG THE CURRENT PROCESSES. IF A PROCESS NEEDS MORE MEMORY THAN IS CURRENTLY AVAILABLE, THE MCP WILL SELECT A SUITABLE IN-USE AREA, OVERLAY THE CONTENTS TO DISK, AND THEN ASSIGN THAT AREA TO THE PROCESS.

IN ADDITION TO ALLOCATING MAIN MEMORY, THE MCP ALLOCATES DISK AREAS. IF A PROCESS OR THE MCP REQUIRES MORE DISK AREA THAN IS CURRENTLY AVAILABLE, THE MCP WILL SELECT THE OLDEST DISK FILES WHICH ARE CONTAINED IN A SUITABLE AREA AND PROCEED TO AUTOMATICALLY CREATE A SYSTEM LIBRARY TAPE CONTAINING THE FILES WHICH ARE TO BE OVERLAYED. WHEN THE AREA HAS BEEN CLEARED, THE MCP WILL ADJUST THE DISK DIRECTORY INFORMATION TO SHOW THAT THE OVERLAYED FILES RESIDE ON SYSTEM LIBRARY TAPE, AND THEN REALLOCATE THE AREA TO THE PROCESS REQUIRING IT.

IN ORDER TO BE ABLE TO RECALL THE FILES WHICH ARE LOCATED ON SYSTEM LIBRARY TAPES, THE MCP REQUIRES VOLUME SERIAL NUMBERS FOR THE TAPES. THE VOLUME SERIAL NUMBER IS USED FOR MCP-OPERATOR COMMUNICATION IN DENOTING WHICH LIBRARY TAPE TO LOAD FOR FUTURE RECALLS OF THE FILES.

2.3.2. ADDRESS SPACE CONTROL

THESE ROUTINES BREAK NATURALLY INTO FIVE GROUPS SHOWN ON THE DEPENDENCY CHART (FIGURE F2-4).

I. PRIMARY CORE CONTROL ROUTINES WHICH OPERATE THROUGH MEMORY LINKS.

II. OVERLAY ROUTINES WHICH MOVE INFORMATION FROM CORE TO DISK, AND CORE TO CORE FREEING THE CORE FOR RE-USE.

III. OVERLAY DISK CONTROL ROUTINES WHICH STORE INFORMATION CONCERNING WHICH DISK AREAS HAVE BEEN OVERLAYED.

IV. FAILURE ACTION ROUTINES WHICH HANDLE CONFLICTING CLAIMS FOR THE CONTROL OF MEMORY LINKS AND DEMANDS FOR SPACE THAT CANNOT CURRENTLY BE MET.

V. PROGRAM INTRINSICS WHICH ARE PROVIDED FOR THE MAIN CORE CONTROL ROUTINES TO SATISFY THE THE REQUIREMENT OF BOTH USERS AND MCP FOR ARRAYS IN AN ALGOL ENVIRONMENT OF NESTED BLOCKS.

THE FOLLOWING SECTIONS DESCRIBE THE FUNCTION IN GREATER DETAIL.

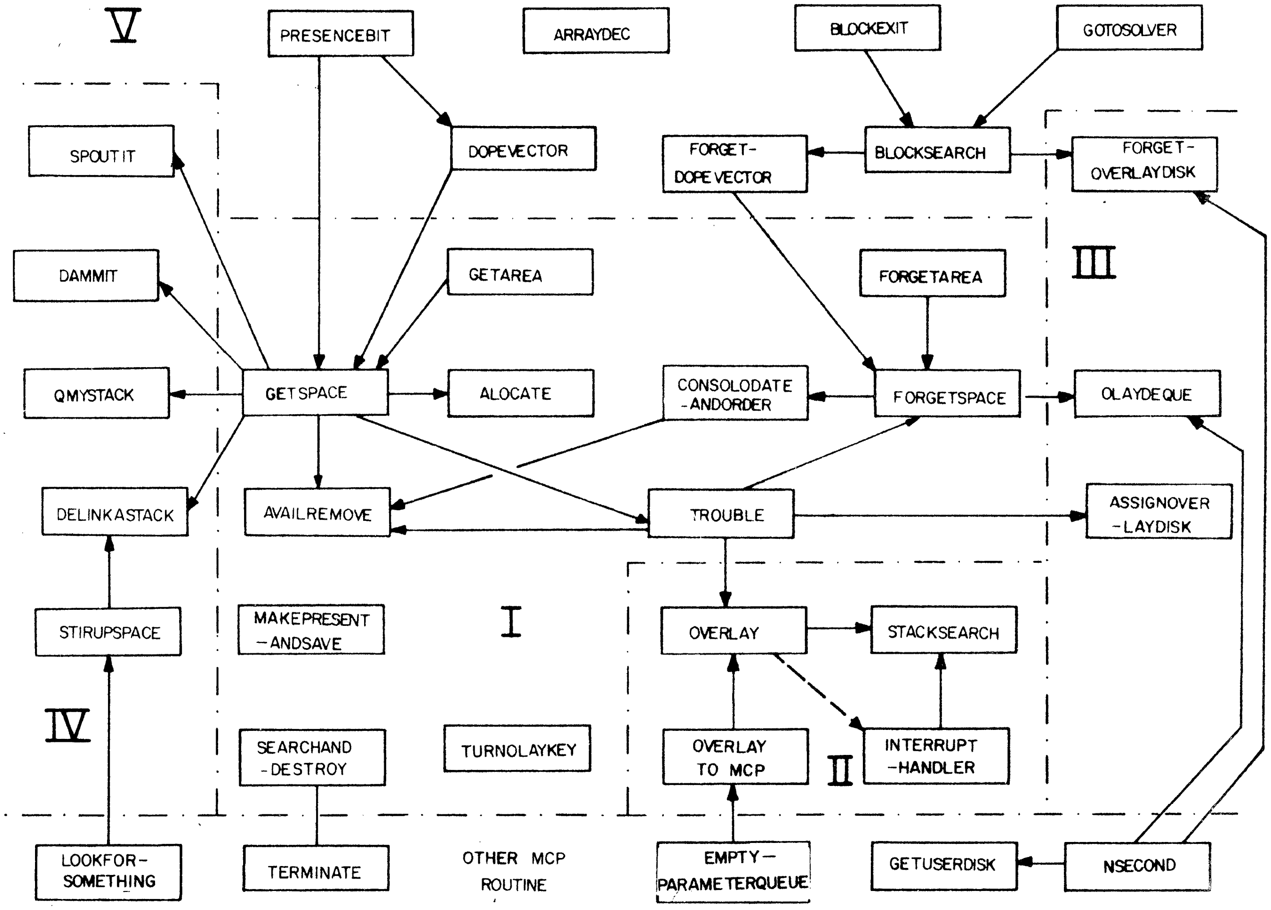


FIGURE F2-4. DEPENDENCY CHART FOR STORAGE CONTROL

2.3.2.1. PRIMARY CORE CONTROL ROUTINES

MEMORY LINKS ARE CONTROL WORDS EMBRACING DISCRETE ASSIGNED AREAS OF CORE AND LINKING THEM INTO LISTS ACCORDING TO THEIR CURRENT USE. THE LEFT-OFF LIST LINKS IN-USE AREAS (WHICH MAY BE "SAVE" OR "OVERLAYABLE") IN CHRONOLOGICAL ORDER OF ALLOCATION. THE AVAILABLE LIST LINKS AVAILABLE AREAS BY SIZE STARTING AT CONTIGUOUS LOW ADDRESSES IN CORE.

"GETSPACE" IS CALLED BY OTHER MCP ROUTINES STATING SIZE NEEDED, OVERLAYABILITY, REQUESTING STACK NUMBER. GETSPACE COMPARES THE REQUIRED SIZE WITH THE LARGEST AREA IN EACH HALF OF THE AVAILABLE LIST (CORRECT TYPE FIRST), AND IF EITHER IS LARGE ENOUGH DOES A LINKED LIST LOOKUP (LLLU) ON THAT HALF TO FIND THE SMALLEST AREA LARGE ENOUGH. THIS IS REMOVED FROM THE LIST USING "AVAILREMOVE". IF NEITHER IS LARGE ENOUGH, GETSPACE CALLS "TROUBLE" WHICH TAKES EACH ENTRY IN THE LEFT-OFF (IN-USE) LIST, OLDEST FIRST, AND COUNTS ALL SPACE PHYSICALLY BELOW IT UNTIL A "SAVE" AREA IS ENCOUNTERED. ANYTIME ENOUGH CONTIGUOUS MEMORY IS FOUND, TROUBLE CLAIMS IT BY CALLING AVAILREMOVE FOR THE AVAILABLE COMPONENTS, AND OVERLAY TO COPY TO DISK OR CORE (SEE SECTION 2.3.2.2.) THEN FORGETSPACE (WITH A NEGATIVE ARGUMENT TO MERELY REMOVE FROM LEFT-OFF LIST) FOR THE IN-USE ONES. IF TROUBLE FAILS TO FIND ADEQUATE SPACE, IT RETURNS A NEGATIVE ANSWER.

ONCE SUFFICIENT SPACE HAS BEEN OBTAINED, GETSPACE CALLS ALLOCATE (ANOTHER LOCAL PROCEDURE) TO MAKE UP IN-USE LINKS AND TO ENTER THE AREA INTO THE MOST RECENTLY ALLOCATED END OF THE LEFT-OFF LIST, RETURNING ANY EXCESS SPACE TO THE AVAILABLE LIST.

FORGETSPACE REMOVES A NAMED AREA FROM THE LEFT-OFF LIST. IF THE ARGUMENT IS NEGATIVE, THE CALLER IS THE PROCEDURE TROUBLE AND NO FURTHER ACTION IS NEEDED, OTHERWISE IT CALLS CONSOLIDATEANDORDER TO COMBINE AFFECTED AREA WITH ANY ADJACENT AVAILABLE AREAS (OBTAINED USING AVAILREMOVE), TO ADD THE RESULT TO THE AVAILABLE LIST (USING AVAILREMOVE) AND THEN TO ADD THE RESULT TO THE AVAILABLE LIST (USING ORDER).

MAKEPRESENTANDSAVE, GIVEN A DESCRIPTOR, ENSURES THAT THE AREA IS PRESENT AND MARKED "SAVE". THE SOLUTION TO THE ORIGINAL OVERLAYABILITY BEING REMEMBERED IS TO CALL TURNOVERLAYKEY WHICH CHANGES THE AREA TO OVERLAYABLE PROVIDED THAT FIRSTLY IT ONCE EXISTED, AND SECONDLY THERE IS STILL A MOTHER DESCRIPTOR FOR IT. GETSPACE ALWAYS ASSIGNS "SAVE" AREAS, WITH THE ACTUAL OVERLAYABILITY REMEMBERED AS IF MAKEPRESENTANDSAVE HAD BEEN CALLED. THIS ALLOWS THE CALLER TO SET INITIAL VALUES FOR ITS NEW

AREA WITHOUT INTERRUPTION, AFTER WHICH IT CAN USE TURNOVERLAYKEY IF NECESSARY.

SEARCHANDDESTROY (USED BY TERMINATE) SEARCHES A STACK AND CALLS FORGETSPACE FOR THE AREA DESCRIBED BY ANY PRESENT MOTHER DESCRIPTOR THAT IT FINDS.

"AREAMANAGER" HAS ABSOLUTE TOP SYSTEM PRIORITY. IT ENSURES THAT THE SYSTEM ALWAYS HAS SUFFICIENT AREA TO RUN ON. "AREAMANAGER" ALLOCATES AND DEALLOCATES AREAS USING "GETAREA" AND "FORGETAREA" RESPECTIVELY.

"GETAREA" PERFORMS TWO FUNCTIONS: THE FIRST IS TO HANDLE SMALL AREAS WHERE THE OVERHEAD PENALTY FOR USING MEMORY LINKS WOULD BE TOO HIGH. SECONDLY "GETAREA" GUARANTEES TO THE SYSTEM THAT AREA WILL BE AVAILABLE WHEN OVERLAY IS FORBIDDEN.

"FORGETAREA" RETURNS THE AREA TO THE POOL WHICH IS MAINTAINED BY "AREAMANAGER".

2.3.2.2. OVERLAY ROUTINES

WHEN OVERLAY IS CALLED BY TROUBLE TO WRITE THE CONTENTS OF SOME CORE AREA TO DISK OR CORE, IT MUST ALSO LOOK FOR AND MODIFY ANY COPIES OF THE DESCRIPTOR FOR THE AREA. SINCE IT MUST INTERRUPT ALL OTHER PROCESSORS, IN CASE THEY ARE USING THE AREA IT IS GOING TO OVERLAY, IT ARRANGES FOR THEM TO HELP WITH THE SEARCHING.

IT DOES THIS BY SETTING A GLOBAL PROCESSOR-ID-RELATED FLAG AND CAUSING A "HEYU" INTERRUPT. THE OTHER PROCESSORS (WHEN THEY GET INTO THE INTERRUPT-HANDLER) RESPOND TO THE SET GLOBAL PROCESSOR-ID-RELATED FLAG BY THEIR OWN FLAGS AND THEN LOOPING UNTIL THE FLAGS ARE CLEARED AGAIN. THE GLOBAL PROCESSOR , MEANWHILE, LOOPS UNTIL ALL THE FLAGS ARE SET, THEN CLEARS THEM.

AT THIS POINT (UNLESS A CODE AREA IS BEING DEALT WITH) ALL CALL STACKSEARCH; THE EXCEPTION IS NECESSARY BECAUSE SEGMENT DESCRIPTORS DO

NOT HAVE COPIES. STACKS ARE SEQUENTIALLY SELECTED AND SEARCHED FOR COPIES OF THE SIMULTANEOUS MULTIPLE ACCESS AND UPDATE OF THE STACK NUMBER COUNTER, SO EACH STACK IS SEARCHED ONLY ONCE. AS EACH PROCESSOR DISCOVERS THAT THERE ARE NO MOVE STACKS, IT SETS ITS FLAG, AND WAITS UNTIL ALL THE FLAGS ARE SET. THUS THEY ALL EXIT TOGETHER. (THE OTHER PROCESSORS NOW RETURN TO THEIR INTERRUPTED ACTIVITIES).

THE PROCESSOR (CONTINUING IN OVERLAY) NOW DOES THE DISK I/O AND ON A SINGLE-PROCESSOR SYSTEM WAITS (ALLOWING POSSIBLE SWITCH TO ANOTHER STACK) BUT ON A MULTIPLE-PROCESSOR SYSTEM LOOPS WITH INTERRUPTS ENABLED. WHEN THE I/O IS SUCCESSFULLY COMPLETED, IT EXITS.

2.3.2.3. OVERLAY DISK CONTROL ROUTINES.

EACH STACK CONTAINS A DESCRIPTOR FOR A STANDARD DISK HEADER WHICH CONTAINS ALL THE OVERLAY DISK ALLOCATION FOR THAT STACK. TO FACILITATE THE DEALLOCATION OF OVERLAY DISK WHEN DESCRIPTORS ARE ABANDONED DURING BLOCK EXIT, (SEE SECTION 2.3.2.5.) EACH ROW OF DISK CONTAINS OVERLAY AREAS FOR THE DESCRIPTORS OF JUST ONE BLOCK. BELOW EACH GENUINE MSCW (SEPARATING THE PARAMETERS AND LOCAL VARIABLES FOR THE PREVIOUS BLOCK FROM THE INTERMEDIATE RESULTS ACCUMULATED BEFORE THIS ENTRY OCCURRED) IS A WORD WITH TAG = 6 AND [47:1] = 1. THE WORD BELOW THE MSCW FOR A GIVEN BLOCK CONTAINS THE HEAD OF A CHAIN CONNECTING THE HEADER WORDS OF THE ROWS USED BY THAT BLOCK, AND A NOTE OF HOW FULL THE LATEST ROW IS. THERE IS ALSO A CHAIN OF DEALLOCATED ROWS, WHOSE HEAD (NEXTAVAIL) IS IN WORD 9 OF THE HEADER.

FORGETOVERLAYDISK, CALLED BY BLOCKEXIT, LINKS THE ROW WORD(S) ALLOCATED TO THE SUBJECT BLOCK INTO THE DEALLOCATED CHAIN.

WHEN "TROUBLE" NOTICES AN "OVERLAYABLE" CORE AREA, WHICH DOES NOT YET HAVE AN OVERLAY DISK ADDRESS, IT CALLS ASSIGNOLAYDISK, WHICH CHAINS FROM THE MEMORY LINK TO THE DESCRIPTOR, MASKSEARCHES DOWN TO THE OLAYDISKINFO (TAG = 6) WORD, AND ASSIGNS DISK FROM THE LATEST ROW IF THERE IS ENOUGH LEFT. IF THERE IS NOT, A NEW ROW IS OBTAINED FROM THE DEALLOCATED CHAIN, MARKING NEXTAVAIL: = -1 IF THE CHAIN IS NOW EMPTY. WHEN ASSIGNOLAYDISK SUBSEQUENTLY ENCOUNTERS THIS, TROUBLE IS TOLD TO MARK THE AREA "SAVE" TEMPORARILY, AND THE AREA IS ADDED TO THE SYSTEM OLAYDISKQ.

OLLAYSCOUT IS RUN AS AN INDEPENDENT RUNNER TO ALLOCATE NEW DISK SPACE TO OVERLAY FILES.

2.3.2.4. GETSPACE FAILURE MECHANISM

IF A STACK MUST WAIT FOR SOMETHING, AND NOT BE ACTIVATED UNTIL NOTHING ELSE (OF HIGHER PRIORITY) WANTS IT, IT CALLS QUEUEMYSTACK, NAMING A QUEUE HEAD INTO WHICH IT WILL BE LINKED, A STACK LOCATION AVAILABLE FOR USE AS A LINK (ACTUALLY THE THREE SUBSEQUENT LOCATIONS MUST BE FREE TOO, BECAUSE THEY ARE GOING TO BE USED). A SORT KEY (PRIORITY) FOR ENTERING IT INTO THE QUEUE, AND AN EVENT FOR LATER USE BY DELINKASTACK IS ALSO CALLED. IT DELINKS THE TOP ENTRY FROM THE NAMED QUEUE, AND CAUSES ITS EVENT.

THESE TWO ROUTINES ARE ACTUALLY QUITE GENERALIZED, BUT ARE CURRENTLY ONLY USED FOR SPACEQ, A QUEUE OF STACKS REQUIRING CONTROL OF SPACELOCK WHICH IS A WORD USED TO PREVENT SIMULTANEOUS MANIPULATION OF THE MEMORY LINKS.

2.3.2.5. PROGRAM INTRINSICS.

TO FACILITATE RELOCATION OF CODE AND DATA, DESCRIPTORS CONTAINING A BASE (ADDRESS) AND LENGTH ARE USED TO DEFINE CORE AREAS. A "MOM" DESCRIPTOR DEFINES THE ENTIRE AREA: ELEMENTS (WORDS/CHARACTERS) ARE ACCESSED BY COPYING THE DESCRIPTOR TO THE STACK INDEXING IT WITH A SUBSCRIPTS VALUE, AND FINALLY USING THE INDEXED COPY WITH A FETCH OR STORE OPERATION.

WHEN AN ATTEMPT IS MADE TO READ INTO DATA WHOSE PRESENCE BIT IS ZERO, A "PRESENCE BIT" INTERRUPT IS GENERATED BY THE HARDWARE. THE SOFTWARE MAKES THE DATA PRESENT, AND OPERATION IS RESUMED.

WHEN AN AREA IS MADE PRESENT, ALL THE COPIES ARE NOT ADJUSTED TO REFLECT THIS BUT IT IS NECESSARY TO KNOW WHEN ANOTHER COPY IS OBTAINED SO AS NOT TO DO THE OVERLAY AGAIN, THUS WHEN THE HARDWARE COPIES AN ABSENT ORIGINAL ("MOTHER") DESCRIPTOR, IT CHANGES THE ADDRESS TO POINT TO THE MOTHER, AND TURNS ON A "COPY BIT". IF THE MOTHER IS PRESENT, THE ADDRESS IS NOT CHANGED. THE COPY BIT IS TURNED ON. NOTE THAT WHEN AN

AREA IS MOVED TO DISK, COPIES MADE WHILE THE MOTHER WAS PRESENT MUST BE SEARCHED FOR: THIS TEDIOUS TASK IS PERFORMED USING THE MASKED SEARCH FOR EQUAL (SRCH) OPERATOR TO LOOK FOR THE CORRECT ADDRESS FIELD.

(1) PRESENCEBIT ROUTINE AND DESCRIPTOR WITH SPECIAL MEANINGS

THE ACTION REQUIRED OF PRESENCEBIT DEPENDS UPON THE KIND OF ABSENT DESCRIPTOR ENCOUNTERED, ALTHOUGH A NEW PRESENT COPY MUST ALWAYS BE CONSTRUCTED AND RETURNED.

(A) FOR AN ABSENT COPY OF A MOTHER ALREADY MADE PRESENT, NOTHING ELSE IS REQUIRED.

(B) FOR A CODE SEGMENT DESCRIPTOR, GETSPACE MUST BE ASKED FOR AN "OVERLAYABLE" TYPE CORE AREA OF SUFFICIENT LENGTH, AND A DISK READ PERFORMED FROM A CODE FILE.

(2) ARRAY INFORMATION TABLE

ALL COMPILERS USE THE SAME MECHANISM FOR HANDLING MULTIDIMENSIONAL AND/ OR SEGMENTED AND/OR DYNAMIC ARRAYS. WHEN AN ARRAY IS DECLARED, INFORMATION IS PASSED TO AN MCP PROCEDURE CONCERNING THE NUMBER OF DIMENSIONS, SIZE OF EACH DIMENSION, LOWER BOUNDS IF OWN, TYPE OF ARRAY, AND LOCATION OF MOM DESCRIPTOR IN THE STACK. THE PROCEDURE RECORDS THIS INFORMATION IN THE ARRAY INFORMATION TABLE (AIT) OR OWN ARRAY TABLE (OAT) BOTH BEING LINKED INTO THE D(2) STACK FOR THAT JOB. THE MOM DESCRIPTOR NOW CONTAINS AN INDEX INTO THE AIT OR OAT.

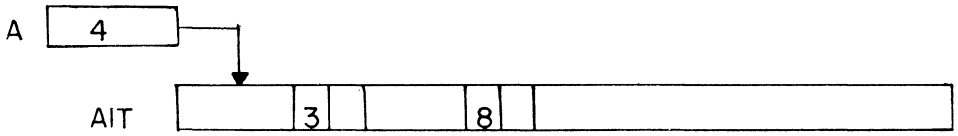
AT PRESENCE BIT TIME IT IS NOTED THAT THE MOM DESCRIPTOR CONTAINS AN INDEX INTO THE AIT AND THE FOLLOWING ACTION TAKES PLACE: FIRST AN AREA OF SAVE MEMORY IS OBTAINED WHOSE SIZE EQUALS THAT CONTAINED IN THE LENGTH FIELD OF THE NON-PRESENT MOM, SECOND THIS AREA IS FILLED WITH THE WORD FROM THE AIT POINTED TO BY THE MOM. THIS MAY BE A DESCRIPTOR CONTAINING AN INDEX INTO THE AIT; SUBSEQUENT PRESENCE BIT ACTION WILL CAUSE THE ABOVE PROCEDURE TO BE REPEATED.

EXAMPLE:

DECLARATIONS:

A[4,3,8]

AT BLOCK ENTRY TIME THIS WILL BE SET UP AS FOLLOWS:



USING A[0,0,0]:=1 AS AN EXAMPLE TO CAUSE PRESENCE BIT ACTION, THE FOLLOWING THREE STEPS WILL OCCUR:

LET P= PRESENT

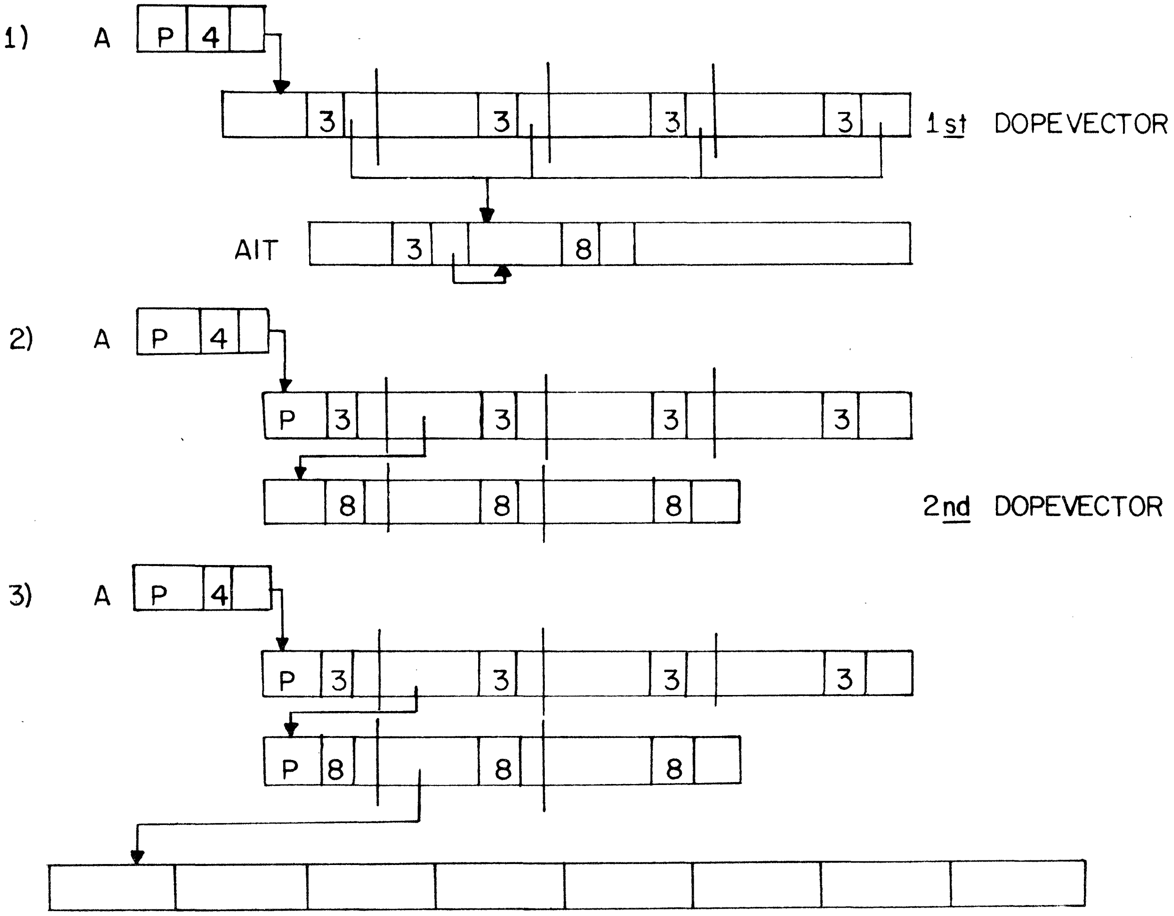
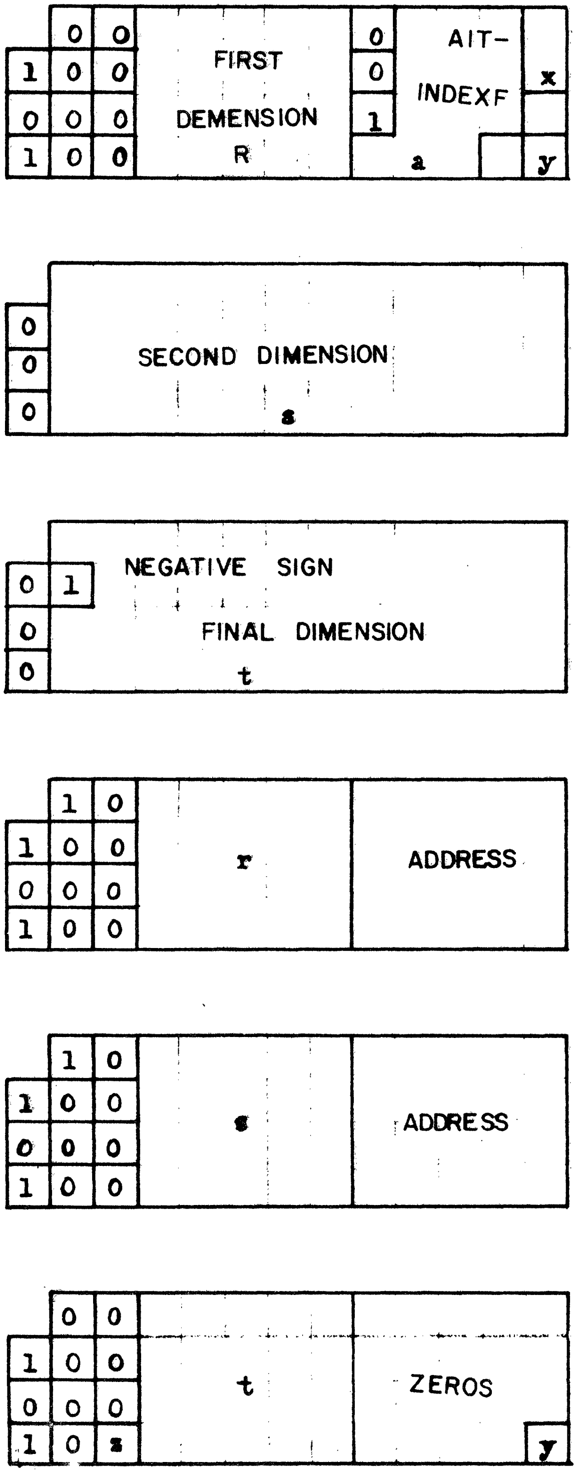


FIGURE F2-5. DYNAMIC ARRAYS



"ABSENT" MOM FOR A [*, *, *]

AIT [A]

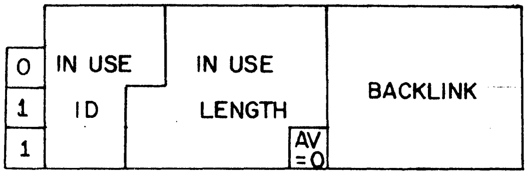
AIT [A+1]

"PRESENT" MOM FOR A [*,*,*]

DOPE VECTOR FOR A [1,*,*]

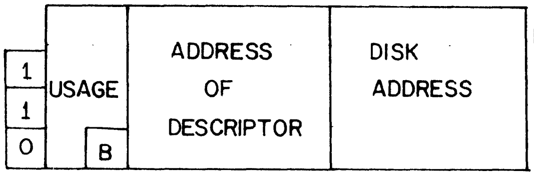
DOPE VECTOR FOR A [1,1,*]

IGURE F2-6. EXPANSION OF AIT ENTRIES INTO DOPE VECTORS AND AFTER TWO CALLS ON PRESENCEBIT



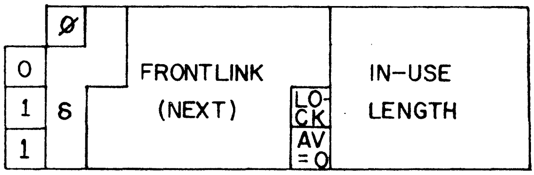
LINKA

AV- AVAILABLE



LINKB

B = 1 IF IN OVERLAY DISK QUEUE. DISK ADDRESS- AFTER BLIND MOVE FROM INITIALIZED DESCRIPTOR (SEE PRESENCE-BIT WRITE-BIT IN STORAGE CONTROL FOR FUNNY SETTINGS)

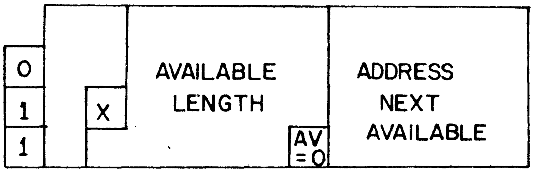


δ -SPACE NOT CONSOLIDATED

LINKZ

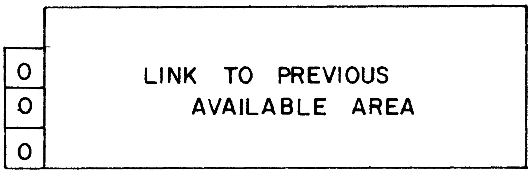
0 - SET BY GETSPACE IF OVERLAY POSSIBLE (QOUTED BY CALLER). LOCK = 1 IF CALLER ALLOWS OVERLAY- THEN HE SETS FROM 0 BY CALLING TURNOVERLAYKEY. IN-USE LENGTH - CAN BE USED FOR LINKING DISK OVERLAY Q BECAUSE ENTRIES ALL. SAVE - SO TROUBLE NEVER TRIES TO LINK BACK PAST THEM.

AVAILABLE LINKS

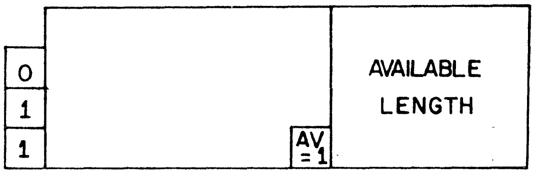


AVAILA

X - OVERLAY BIT OF PREVIOUS AREA.



AVAILB



AVAILZ

FIGURE F2-7. MEMORY LINKS

(3) BLOCKEXIT, BLOCKSEARCH, AND FORGETDOPEVECTORS

CALL ON BLOCKEXIT IS COMPILED PRECEDING EXIT FROM A BLOCK OR PROCEDURE N WHICH ARRAYS, FILES, OR OTHER LANGUAGE ELEMENTS REQUIRING MEMORY REAS HAVE BEEN DECLARED:

LOCKSEARCH, USING THE MASKED SEARCH FOR EQUAL OPERATOR, SCANS BETWEEN HE GIVEN TOP AND BOTTOM ADDRESSES FOR ALL OCCURRENCES OF [47:2] = 2. HESE MAY BE:

(A) TAG 5: THESE COULD BE PRESENT MOM DATA DESCRIPTORS, FOR WHICH FORGETDOPEVECTORS IS CALLED, OR FIB DESCRIPTORS, FOR WHICH CLOSE IS EMPLOYED, OR

(B) TAG7: SOFTWARE INTERRUPTS WHICH ARE SUITABLE DELINKED FROM RESPECTIVE QUEUES.

(C) TAG6: BLOCKCONTROLWORDS, FOR WHICH OVERLAYDISK IS RETURNED TO THE FILE AND THE CRITICAL BLOCK COUNT IS CHECKED.

ORGETDOPEVECTORS RECURSIVELY RETURNS DOPE VECTOR SPACE.

(4) GOTOSOLVER

"GO TO" WHICH LEAVES THE CURRENT LEXICOGRAPHICAL LEVEL IS COMPILED AS CALL ON THIS ROUTINE, PASSING AS A PARAMETER AN SIRW WITH OR WITHOUT AG) ZERO) TO A PCW (DESCRIBING THE POINT IN THE CODE TO WHICH WE WISH O GO) PREVIOUSLY GENERATED IN THE STACK AT THE LEVEL TO WHICH WE ARE ETURNING. SHOWN BELOW IS THE STACK (A) JUST AFTER ENTERING, AND (B) UST BEFORE LEAVING GOTOSOLVER.

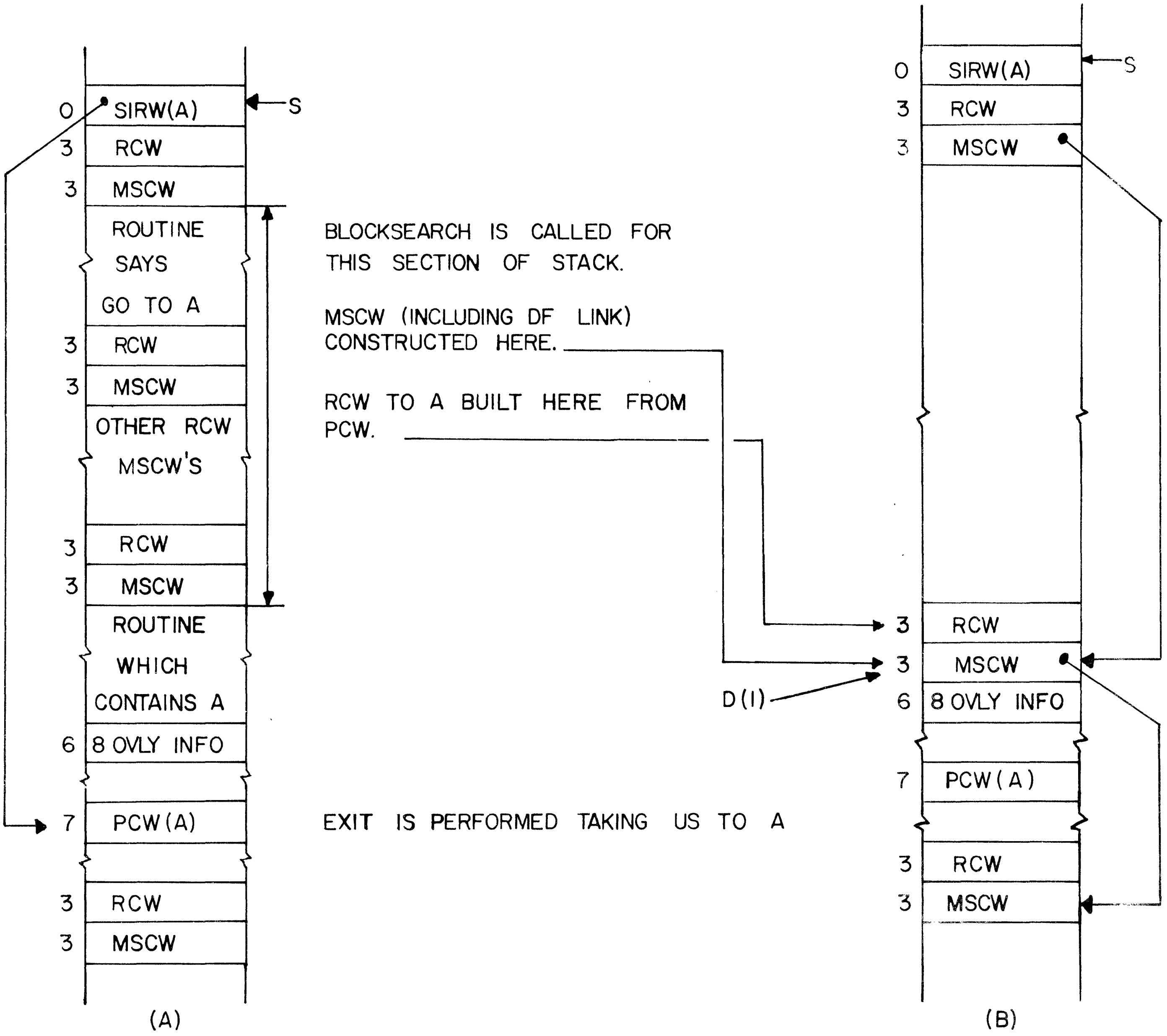


FIGURE F2-8. GOTOSOLVER ACTION

2.4. SYSTEM RECONFIGURATION

IN THE EVENT THAT A HARDWARE MODULE FAILS OR MUST BE SHUT DOWN FOR MAINTENANCE, THE SYSTEM MUST BE RECONFIGURED TO ELIMINATE THE MODULE WHICH IS NO LONGER AVAILABLE TO THE SYSTEM. THE ABILITY TO RECONFIGURE THE SYSTEM EASILY AND EFFICIENTLY IS DESIGNED INTO THE B6700, SO THAT THE LOSS OF A PARTICULAR KEY MODULE WILL NOT BE CATASTROPHIC TO THE SYSTEM UNLESS THAT MODULE IS UNIQUE. FOR EXAMPLE, THE FAILURE OF ONE PROCESSOR IN A TWO PROCESSOR SYSTEM WOULD CAUSE A DEGRADATION OF SYSTEM PERFORMANCE, BUT THE SYSTEM WOULD STILL BE OPERABLE. IN THE B6700 SYSTEM, THERE ARE VERY FEW SYSTEM MODIFICATIONS WHICH ARE NOT HANDLED AUTOMATICALLY BY THE MCP, AND THERE ARE EVEN FEWER WHICH CANNOT BE HANDLED WITH THE AID OF A "HALT-LOAD".

THE BASIC CRITERION FOR BEING ABLE TO SHUT DOWN OR DISCONNECT A UNIT IS WHETHER IT IS CURRENTLY IN USE BY SOME PROCESS. IF, FOR EXAMPLE, A MEMORY MODULE IS SHUT DOWN, THE INFORMATION CURRENTLY STORED IN THAT MODULE IS INACCESSIBLE. SUCH A SITUATION WOULD ALMOST CERTAINLY LEAD TO AN INVALID ADDRESS. HOWEVER, IF A UNIT NOT CURRENTLY IN USE (SUCH AS A MAGNETIC TAPE DEVICE) IS SHUT DOWN, THE SYSTEM WILL CONTINUE TO FUNCTION AS IF NOTHING HAS HAPPENED. IT WILL BE POSSIBLE TO ISSUE A COMMAND TO THE MCP INDICATING THAT A PARTICULAR UNIT IS TO BE SHUT DOWN, AND THAT THE MCP WILL RESPOND BY REARRANGING THE SYSTEM TO AVOID THE USE OF THE UNIT, AT WHICH TIME IT WILL INDICATE THAT THE UNIT HAS BEEN DETACHED FROM THE SYSTEM.

THERE ARE, HOWEVER, CERTAIN MAJOR HARDWARE MODIFICATIONS TO THE SYSTEM WHICH WILL REQUIRE MODIFICATION OF THE MCP. THIS IS DUE TO THE FACT THAT HANDLING OF DCP (DATACOM PROCESSOR) INTERRUPTS AND THE GCA (GENERAL CONTROL ADAPTER) INTERRUPTS IS CONTINGENT ON THE NATURE OF THE DEVICE INVOLVED. IF, FOR INSTANCE, A DATACOM SYSTEM OR A SUBSTANTIALLY DIFFERENT DATACOM SYSTEM IS TO BE ADDED, IT WILL BE NECESSARY TO ALTER THE MCP. IT ALSO MEANS THAT BEFORE CONNECTING A DEVICE SUCH AS A PLOTTER OR ANALOG INTERPRETER, IT WILL BE NECESSARY TO SPECIFY THE NATURE OF THE GCA INTERRUPT. IN OTHER WORDS, IT WILL BE NECESSARY TO

SPECIFY HOW THE MCP IS TO HANDLE GCA INTERRUPTS BY CHANGING THE MCP. EXCEPT FOR THESE TWO CONTINGENCIES, RECONFIGURATION OF THE SYSTEM WILL NOT REQUIRE MODIFICATION OF THE MCP.

SECTION 3

MULTIPROCESSING

3. MULTIPROCESSING

THE NORMAL MODE OF OPERATION OF THE B6700 MASTER CONTROL PROGRAM ASSUMES THE EXISTENCE OF MULTIPLE JOBS OR "PROCESSES" RUNNING CONCURRENTLY. THE OBJECT OF RUNNING PROCESSES CONCURRENTLY (MULTIPROCESSING OPERATION) IS TO MAXIMIZE THE UTILIZATION OF THE B6700 SYSTEM RESOURCES, THEREBY INCREASING THE THROUGHPUT OF JOBS.

IN ORDER TO OBTAIN THE GREATEST THROUGHPUT OF JOBS IN A MULTIPROCESSING ENVIRONMENT, IT IS ESSENTIAL TO MINIMIZE THE AMOUNT OF MCP OVERHEAD REQUIRED TO EXECUTE THE JOBS CURRENTLY IN PROGRESS. IN ORDER TO MINIMIZE OVERHEAD, THE B6700 MCP CONTROLS STORAGE ALLOCATION FOR EACH PROCESS ACCORDING TO ITS CURRENT REQUIREMENTS. BY BRINGING PROGRAM SEGMENTS INTO MEMORY ONLY WHEN THEY ARE NEEDED, MEMORY IS ASSIGNED IN AN EFFICIENT MANNER. IN THE EVENT THAT SEVERAL PROCESSES REQUIRED MORE MEMORY THAN IS CURRENTLY AVAILABLE, THE MCP RE-ALLOCATES MEMORY FOR EACH JOB AS REQUIRED AND THE LEAST-USED SEGMENTS WHICH ARE PRESENT IN MEMORY ARE OVERLAYED. DATA SEGMENTS WHICH ARE OVERLAYED MUST BE WRITTEN ON THE DISK, SINCE THE DATA MAY HAVE BEEN MODIFIED WHILE IT WAS IN MEMORY. PROGRAM SEGMENTS AND READ-ONLY DATA SEGMENTS WHICH ARE OVERLAYED NEED NOT BE SAVED, SINCE THE ORIGINAL COPY OF THE SEGMENT IS STILL PRESENT ON THE DISK.

3.1. RE-ENTRANT OBJECT PROGRAMS

3.1.1. INTRODUCTION

OBJECT PROGRAMS RESIDE ON THE DISK WHERE THEY ARE REFERENCED AS CODE FILE BY THE MCP THROUGH THE DISK DIRECTORY. THE REFERENCE WILL BE THE CONSEQUENCE OF EITHER AN "EXECUTE" REQUEST OR THE "GO" PART OF A "COMPILE AND GO". IN EITHER CASE, THE CODE FILE WILL HAVE BEEN CONSTRUCTED BY A COMPILER.

THE MAIN FUNCTION OF A COMPILER IS TO CONVERT SYMBOLIC SOURCE STATEMENTS INTO OBJECT MACHINE LANGUAGE CODE. EFFICIENT UTILIZATION OF MEMORY IN A

MULTIPROGRAMMING ENVIRONMENT REQUIRES THAT OBJECT CODE FILES BE SEGMENTED, SO THAT DURING EXECUTION OF AN OBJECT PROGRAM THE ONLY PORTION OF THE CODE FILE REQUIRED TO BE RESIDENT IN CORE IS THAT SEGMENT CURRENTLY BEING PROCESSED. SEGMENTATION OF THE OBJECT CODE FILE IS AN ADDITIONAL FUNCTION OF THE COMPILER.

3.1.2. SEGMENTATION

THE DESIGN OF THE B6700 ENABLES THE LENGTH OF A PROGRAM SEGMENT TO BE VARIABLE, DEPENDING ON THE PROGRAM LOGIC AND LANGUAGE USED. ALGOL PROGRAM SEGMENTATION IS BASED ON THE BLOCK STRUCTURE OF THE SOURCE PROGRAM, WHERE EACH BLOCK IS COMPILED INTO A CODE SEGMENT. COBOL PROGRAMS ARE SEGMENTED BY SECTION, UNLESS SPECIFIED OTHERWISE BY THE PROGRAMMER. FORTRAN PROGRAMS ARE SEGMENTED BY PROGRAM UNIT (SUB-ROUTINE OR MAIN PROGRAM), AND IF NECESSARY, THESE UNITS ARE FURTHER SEGMENTED TO OPTIMIZE SEGMENT SIZE. SEGMENTATION OF FORTRAN PROGRAMS MAY ALSO BE EFFECTED BY PROGRAMMER OPTION.

THE CODE FILE CONSISTS OF A NUMBER OF VARIABLE-LENGTH RECORDS, THE FIRST RECORD IN THE FILE IS ONE DISK SEGMENT 30 WORDS IN LENGTH. IT CONTAINS LINKAGE AND OBJECT PROGRAM INFORMATION FOR USE BY THE MCP AT JOB INITIATION. FOLLOWING THIS RECORD ARE THE PROGRAM SEGMENTS, FOR ALL BUT THE "OUTER-MOST" SEGMENT OF THE PROGRAM. ADDITIONAL RECORDS CONTAIN CODE RELATED TO FORMATS, LISTS, AND OTHER COMPILER-GENERATED DATA. ALSO INCLUDED IS COMPILE TIME INFORMATION FOR PROGRAM INPUT-OUTPUT FILES.

THE FINAL RECORDS IN THE CODE FILE IS A DIRECTORY REFERENCING ALL PREVIOUSLY WRITTEN LOGICAL RECORDS. EACH ENTRY IN THIS RECORD IS ONE WORD IN LENGTH AND CONTAINS THE RELATIVE RECORD ADDRESS AND THE SIZE (WORDS) OF THE RECORD IT REFERENCES. THESE WORDS ARE CALLED AS SEGMENT DESCRIPTORS, AND THE RECORD IS CALLED THE SEGMENT DICTIONARY.

THE SEGMENT DICTIONARY IS READ INTO CORE BY THE MCP AT JOB INITIATION. IN CONJUNCTION WITH DISPLAY REGISTER D(1), THE SEGMENT DICTIONARY IS USED TO REFERENCE THE OBJECT CODE SEGMENTS WITHIN THE CODE FILE. THE CODE SEGMENTS ARE READ INTO CORE BY THE MCP AS A CONSEQUENCE OF

PRESENCEBIT INTERRUPTS INCURRED BY THE SEGMENT DESCRIPTOR FOR THE CODE SEGMENT. THE FREQUENCY AND ORDER IN WHICH THE CODE SEGMENTS ARE PROCESSED IS DETERMINED BY THE DYNAMIC FLOW OF THE OBJECT PROGRAM.

3.1.3. RE-ENTRANT CODE

RE-ENTRANT CODE IS COMMON OBJECT CODE WHICH CAN BE EXECUTED BY MORE THAN ONE PROCESS AT A SAME TIME.

FOR THIS TO BE POSSIBLE NO MODIFICATION OF OBJECT CODE CAN BE ALLOWED DURING EXECUTION. SINCE EACH PROCESS ON THE B6700 SYSTEM IS ASSIGNED A SEPARATE MEMORY AREA AS A PUSH DOWN STACK AND HIGHER LEVEL LANGUAGES DO NOT GENERATE SELF-MODIFYING CODE, BOTH OBJECT PROGRAMS AND MCP ROUTINES ARE RE-ENTRANT. OBJECT CODE WORDS ARE ALSO MEMORY PROTECTED BY THE "TAG" BITS ASSOCIATED WITH EACH WORD IN MEMORY SO THAT ANY ATTEMPT TO MODIFY CODE WORDS IN MEMORY CAUSES A "MEMORY PROTECTION" INTERRUPT.

THE ASSIGNED STACKS CONTAIN TEMPORARY RESULTS, SIMPLE VARIABLE VALUES, DATA ARRAY DESCRIPTORS, AND PROGRAM CONTROL INFORMATION. FOR THE RELATIVE LOCATION IN THE SEGMENT DICTIONARY OF THE OBJECT CODE. (SEE FIGURE 3-1).

LEGEND

D[I] DISPLAY REGISTERS (I = 0 THRU 31, I INDICATES LEX LEVEL)
D.A. - DISK ADDRESS
D.D. - DATA DESCRIPTOR
DF - DELTA FIELD, STACK HISTORY LINKAGE
DISP - DISPLACEMENT ADDRESSING ENVIROMENT LINKAGE
MSCW - MARK STACK CONTROL WORD
P1 - PROCESSOR NO. 1
P2 - PROCESSOR NO. 2
PCW - PROGRAM CONTROL WORD
RCW - RETURN CONTROL WORD
S.D. - SEGMENT DESCRIPTOR
BOSR - BASE OF STACK REGISTER
----- PCW REFERENCE TO A SEGMENT DESCRIPTOR

NOTE - THE FOLLOWING ALGOL PROGRAM WAS ASSUMED:

BLOCK STRUCTURE OF PROGRAMS

```
LEX. LEVEL  2  3  4  5
               PROCEDURE A
                  PROCEDURE B
                     PROCEDURE C
OUTER
BLOCK
                  PROCEDURE D
                     PROCEDURE E
                     PROCEDURE F
```

1st COPY:

PROCEDURE A CALLS B, B CALLS C, C CALLS D,
D CALLS E, AND E CALLS F

2nd COPY:

PROCEDURE A CALLS B, B CALLS C (C HAS NOT YET CALLED D)

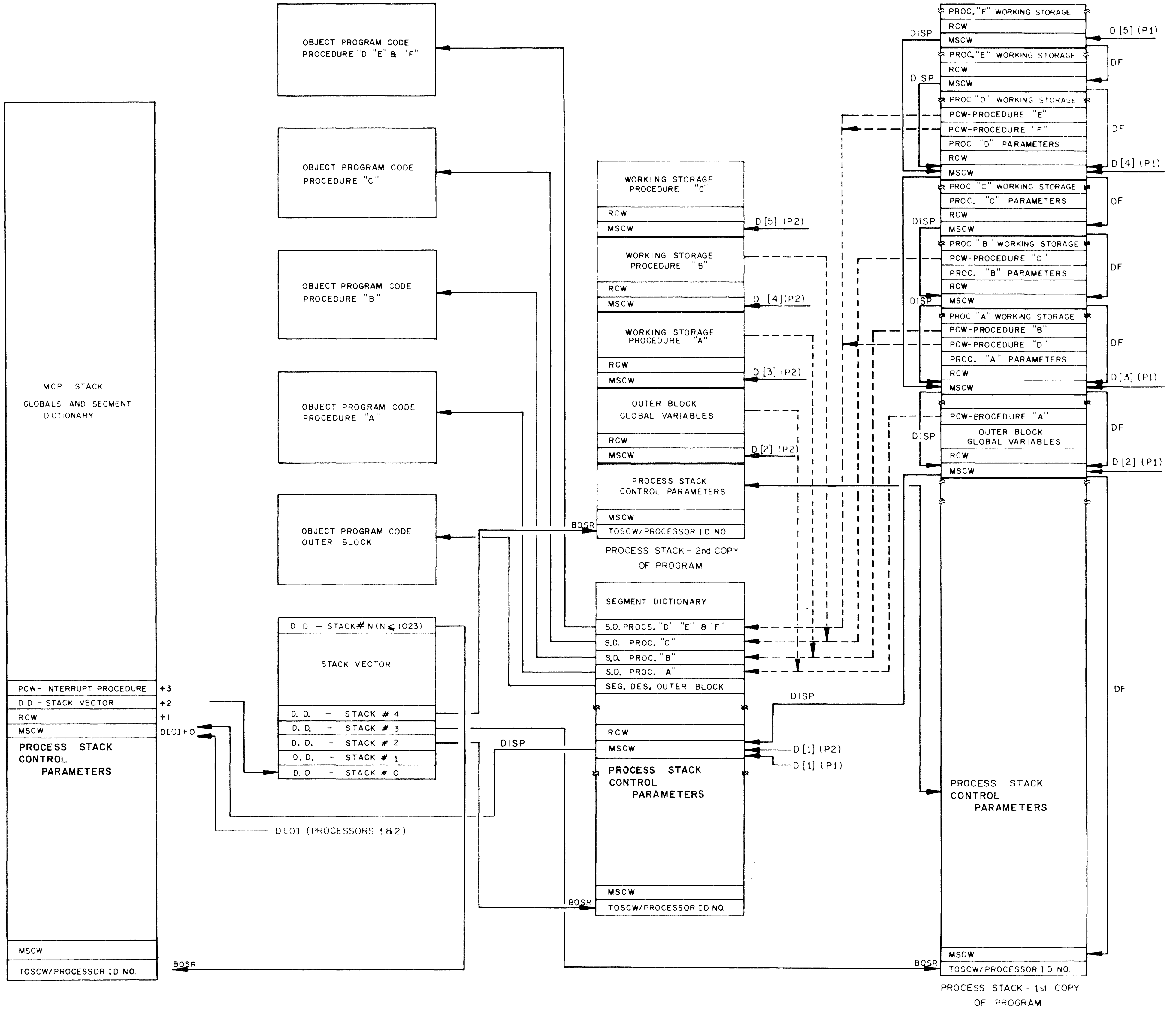


FIGURE F3-1. B6700 REENTRANT PROGRAM STACK STRUCTURE

CODE SEGMENTS FOR ALL PROGRAMS ACCESSING A COMMON CODE FILE ARE MADE REENTRANT BY USING THE SAME SEGMENT DICTIONARY FOR ALL SUCH PROGRAMS. THIS ALSO ENSURES THAT ONLY ONE COPY OF A SEGMENT WILL BE PRESENT IN MEMORY.

A PROGRAM CONTROL WORD IS BUILT WHEN A PROCEDURE IS DECLARED AND IS USED DURING PROCEDURE ENTRY. IT CONTAINS THE FOLLOWING INFORMATION:

1. THE LOCATION IN THE SEGMENT DICTIONARY OF THE OBJECT CODE SEGMENT DESCRIPTOR.

2. THE PROCEDURE ENTRY POINT REFERENCES BOTH THE WORD INDEX RELATIVE TO THE BEGINNING OF THE CODE SEGMENT, AND THE SYLLABLE INDEX RELATIVE TO THE BEGINNING OF THE ENTRY POINT WORD.

THE RUNNING COUNT AND LINK WORD IN THE SEGMENT DICTIONARY INDICATES THE NUMBER OF PROCESSES ACCESSING THE SEGMENT DICTIONARY. WHENEVER THIS COUNT REACHES ZERO THE MEMORY SPACE FOR THE SEGMENT DICTIONARY AND ALL REMAINING CODE SEGMENTS ARE DE-ALLOCATED.

THIS REENTRANT CAPABILITY IS ALSO EXTENDED TO INCLUDE DATA WHICH DOES NOT CHANGE IN VALUE SUCH AS LITERAL STRINGS AND OTHER TYPES OF READ-ONLY DATA. THIS IS ACCOMPLISHED BY PLACING THEIR ASSOCIATED DESCRIPTORS IN THE SEGMENT DICTIONARY.

3.2. COMPILATION

3.2.1. INTRODUCTION

A COMPILER IS A SPECIAL PURPOSE COMPUTER PROGRAM WHICH ACCEPTS SOURCE STATEMENTS IN THE LANGUAGE FOR WHICH THE COMPILER IS WRITTEN. THE OUTPUT OF A COMPILER IS A DISK FILE WHICH CONSISTS OF OBJECT CODE. A COMPILATION REQUIRES CERTAIN FUNCTIONS TO BE PERFORMED BY THE COMPILER AND THE MCP, WHICH INVOLVES:

1. COMMUNICATION BETWEEN A COMPILER AND THE MCP.

2. CONSTRUCTION OF A STANDARD OBJECT CODE FILE BY A COMPILER.

3.2.2. RECOGNITION OF A COMPILER BY THE MCP

A COMPILE CARD IS A REQUEST TO THE MCP TO SCHEDULE A PARTICULAR COMPILER FOR EXECUTION AND PROVIDE SPECIAL HANDLING FOR THIS PROGRAM. THE SCHEDULE ENTRY FOR THIS EXECUTION WILL ALSO REFLECT THE OPTION ASSOCIATED WITH THE COMPILATION (COMPILE AND GO, COMPILE FOR SYNTAX, COMPILE TO LIBRARY).

3.2.3. COMMUNICATION BETWEEN A COMPILER AND THE MCP

EACH COMPILER IS A REAL PROCEDURE WITH ONE FORMAL PARAMETER, A ONE DIMENSIONAL ARRAY. THIS ARRAY IS CREATED AND INITIALIZED BY THE MCP. IT CONTAINS INFORMATION DERIVED FROM THE CONTROL CARDS USED TO REQUEST THE COMPILATION. THE FIRST PART OF THE ARRAY WILL BE USED BY THE COMPILER IN CONSTRUCTING SEGMENT ZERO OF THE CODE FILE. THIS INFORMATION IS FOLLOWED BY ANY FILE PARAMETER INFORMATION SUPPLIED.

3.2.4. CONSTRUCTION OF COMPILER OBJECT CODE FILES

IF THE SOURCE FILE CONTAINS NO SYNTAX ERRORS, AND THE COMPILATION IS NOT

FOR SYNTAX ONLY, THE COMPILER WRITES THE OBJECT CODE TO DISK AND CLOSES THIS FILE WITH LOCK, THROUGH THE MCP. THE MCP RECOGNIZES THE CALLING PROGRAM AS A COMPILER, AND IF COMPILATION REQUEST SPECIFIED "GO" THE OBJECT CODE FILE IS IMMEDIATELY SCHEDULED FOR EXECUTION. IF THE COMPILATION REQUEST SPECIFIED "LIBRARY" THEN THE FILE HEADER IS ENTERED IN THE DISK DIRECTORY, MAKING THE CODE FILE PERMANENT.

3.2.5. SCHEDULING INFORMATION

IN ADDITION TO GENERATING THE OBJECT CODE FILE, THE COMPILERS ARE RESPONSIBLE FOR SUPPLYING SCHEDULING INFORMATION TO THE MCP. THIS INFORMATION, IN THE FIRST RECORD OF THE CODE FILE, INCLUDES THE CORE ESTIMATE, STACK SIZE, AND POINTERS INTO THE CODE FILE FOR LOCATING THE SEGMENT DICTIONARY, THE FILE PARAMETER BLOCK, AND THE FIRST EXECUTABLE CODE SEGMENT. IF PROGRAM PARAMETER CARDS HAD BEEN INCLUDED WITH THE COMPILE REQUEST. THESE CHANGES WILL BE IN EFFECT FOR ALL SUBSEQUENT EXECUTIONS OF THE OBJECT CODE FILE, UNLESS OVERRIDDEN BY PROGRAM CARDS IN THE EXECUTE REQUEST.

3.3. PROCESS HANDLING

3.3.1. "CONTROL CARD" PROCEDURE

THE "STATUS" PROCEDURE OF THE MCP PERIODICALLY USES THE "SCAN-IN PERIPHERAL STATUS" COMMAND TO DETERMINE THE STATUS OF THE PERIPHERAL UNITS. "STATUS", UPON RECOGNIZING THAT THE DEVICE IS "READY", READS THE FIRST RECORD AND CREATES AN INDEPENDENT PROCESS CALLED "CONTROLCARD."

"CONTROLCARD" CREATES AN I/O CONTROL BLOCK CONTAINING A REFERENCE TO A PARTICULAR DEVICE THAT IS "READY". THE RECORDS ARE THEN READ UNTIL A "DATA", "END", "BCL" OR ANOTHER "EXECUTE" OR "COMPILE" IS OBTAINED.

"CONTROLCARD" INTERPRETS THE INFORMATION CONTAINED IN THE RECORD AND IF THE JOB IN THE DEVICE IS A PROGRAM EXECUTION, IT READS SEGMENT ZERO, THE FIRST SEGMENT OF THE CODE FILE BUILT BY THE COMPILER, WHICH CONTAINS:

1. ESTIMATED AMOUNT OF MAIN MEMORY REQUIRED BY THE PROCESS.

2. PRIORITY.

3. MAXIMUM PROCESSTIME.

4. MAXIMUM I/O TIME.

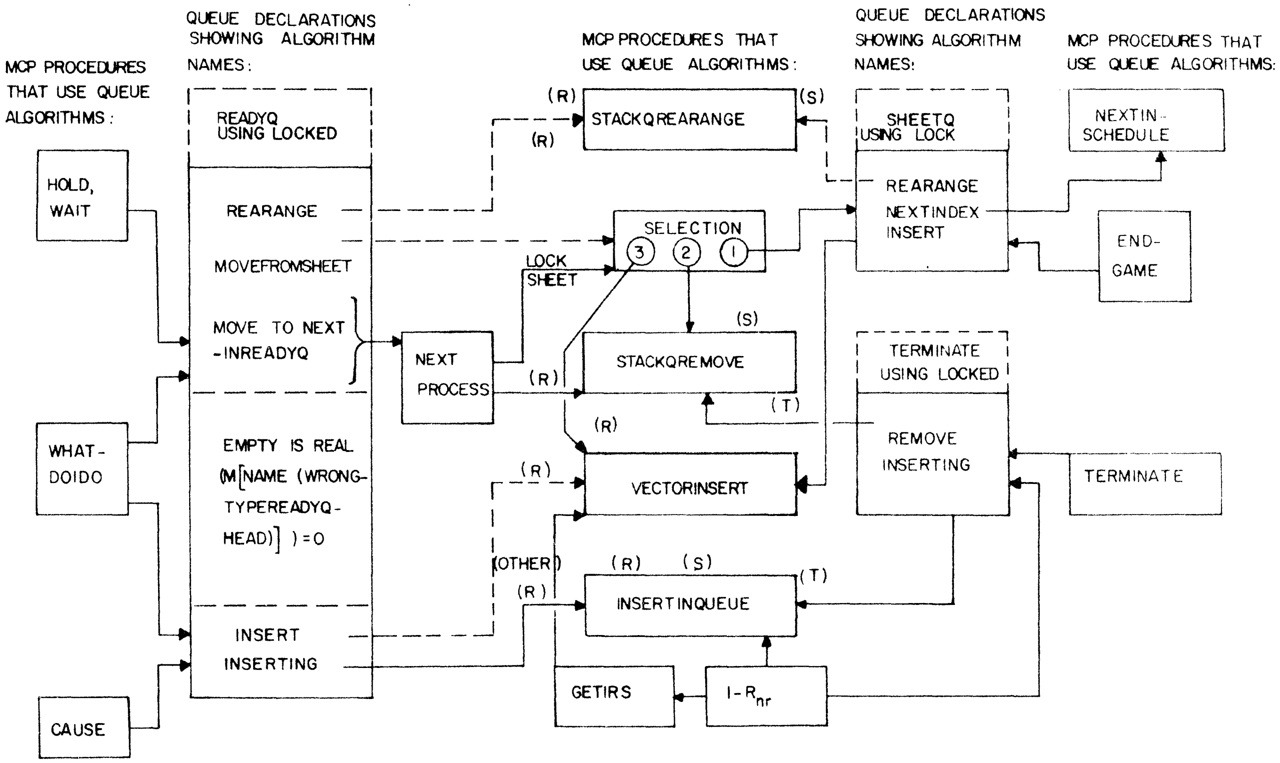
5. FILE PARAMETER BLOCK SIZE AND LOCATION.

6. THE WORKING STORAGE STACK SIZE.

7. SIZE AND LOCATION OF THE SEGMENT DICTIONARY.

WHEN A TERMINAL CONTROL CARD IS READ, "CONTROLCARD" MERGES THE INFORMATION CONTAINED IN SEGMENT ZERO WITH THE INFORMATION IN THE PARAMETER CARDS AND FILE CARDS TO MAKE A SHEET QUEUE; FINALLY IT TERMINATES ITSELF.

THE SHEET QUEUE IS A LINKED LIST OF PROCESSES WHICH ARE SCHEDULED TO BE EXECUTED AS SOON AS SUFFICIENT SYSTEM RESOURCES, SUCH AS MEMORY, ARE AVAILABLE. EACH ENTRY IN THE SHEET QUEUE IS IN THE FORM OF A PARTIALLY BUILT PROCESS STACK.



KEY:
(R), (S), (T) INDICATE MANIPULATION OF READYQ, SHEETQ, TERMQ, RESPECTIVELY.
——► INDICATES CALLS ON ALGORITHMS - - - -► INDICATES POSSIBLE CALLS, NOT INVOKED

FIGURE F3-2. MCP USE OF QUEUE ALGORITMS FOR READYQ, SHEETQ, TERMQ

3.3.2. PROCESS INITIATION

WHEN A JOB IS FIRST INTRODUCED INTO THE SYSTEM AN ENTRY FOR THAT JOB IS MADE IN A QUEUE CALLED THE SHEET QUEUE BY THE MCP CONTROL CARD ROUTINE. AFTER THE CONTROL CARD ROUTINE COMPLETES ITS TASKS, THE ENTRY WILL BE EXAMINED BY A PROCEDURE CALLED SELECTION. SELECTION IS CALLED WHENEVER SOMETHING NEW APPEARS IN THE SHEET QUEUE OR WHENEVER A JOB IS TERMINATED.

SELECTION CHECKS CORE ESTIMATES OF JOBS RUNNING (ADJUSTED FOR REENTRANCY) AGAINST THE RESOURCES AVAILABLE. IF SUFFICIENT RESOURCES ARE CURRENTLY FREE, A PROCESS CALLED INITIATE WILL BE STARTED TO CREATE A PROCESS STACK (AND, IF NECESSARY A SEGMENT DICTIONARY) OUT OF THE SHEET ENTRY AND CODE FILE AND LINKS THIS NEW STACK INTO THE READY QUEUE.

THE ROUTINE RUN HAS A SPECIAL SIGNIFICANCE, BEING THE ULTIMATE "OUTERBLOCK" FOR EVERY STACK. WHEN A STACK IS CONSTRUCTED, SOME CONTROL WORDS ARE "FORGED" TO GIVE THE APPEARANCE THAT RUN WAS INTERRUPTED JUST BEFORE EXECUTING ITS FIRST SYLLABLE. THESE CONTROL WORDS PROVIDE AN SIRW TO THE PCW FOR (TYPICALLY) A JOB OUTER BLOCK: RUN INHERITS THIS AS A "PARAMETER", WHICH IT THEN PROCEEDS TO CALL, CAUSING ENTRY TO THE JOB. THIS PCW MUST BE IN THE USERS SEGMENT DICTIONARY, CAUSING D[1] TO POINT THERE ON ENTRY, THUS MAKING THE SEGMENT DICTIONARY VISIBLE FOR FURTHER USE. IF THE JOB IS AN MCP PROCEDURE OPERATING AS AN INDEPENDENTRUNNER, THE PCW WILL BE IN THE D[0] STACK.

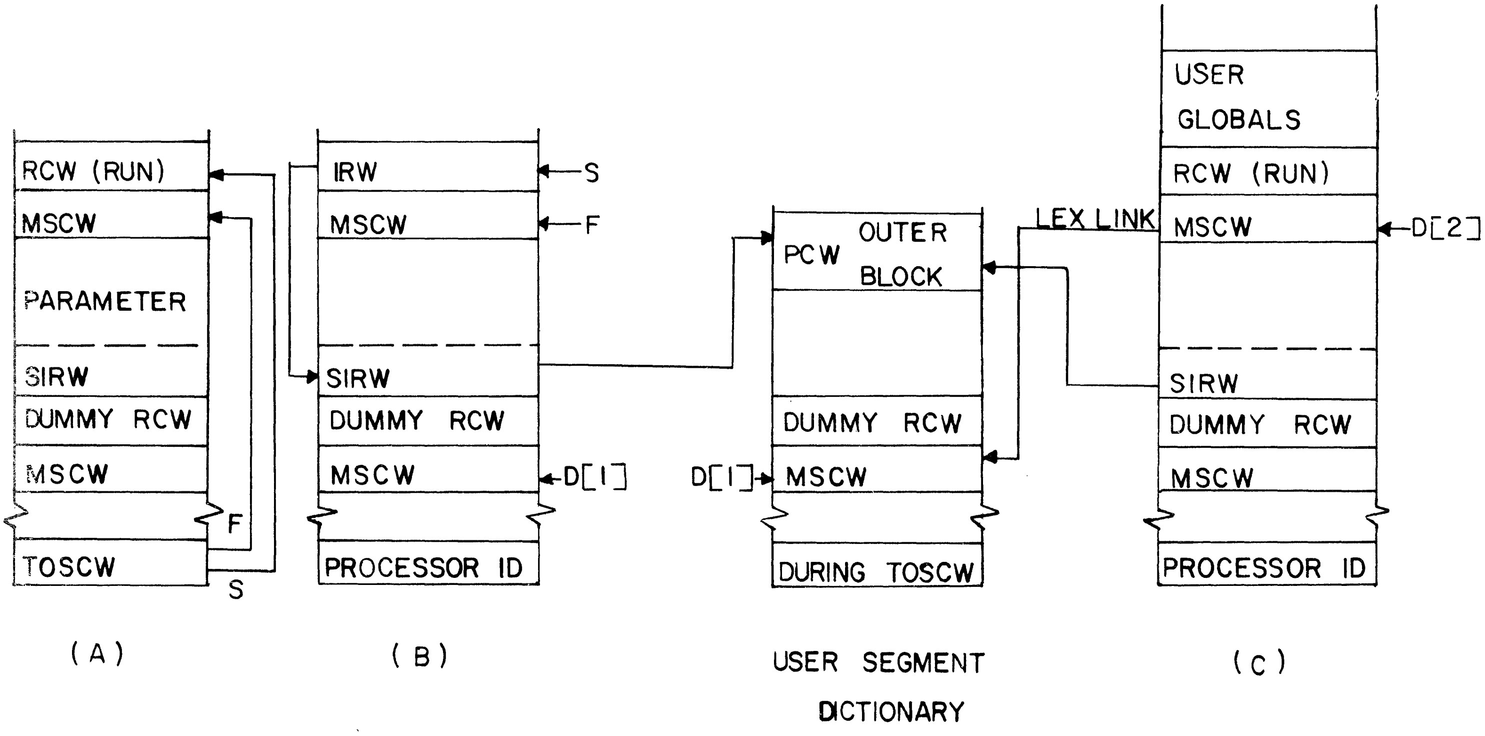


FIGURE F3-3. STACK WAITING IN READYQ, ABOUT TO CALL USER, USER'S OUTER BLOCK

3.3.3. PROCESS EXECUTION

AS SOON AS CONTROL IS TRANSFERRED TO THE NEW PROCESS, A "PRESENCEBIT" INTERRUPT MAY OCCUR BECAUSE THE OUTER BLOCK CODE SEGMENT IS NOT PRESENT IN MAIN MEMORY. THE PRESENCEBIT PROCEDURE OF THE MCP IS ENTERED AND THE FOLLOWING ACTIONS OCCUR IN ORDER TO MAKE THE SEGMENT PRESENT:

1. THE PRESENCEBIT PROCEDURE CALLS THE GETSPACE PROCEDURE TO ALLOCATE AN AREA IN MAIN MEMORY FOR THE CODE SEGMENT.

2. WHEN AN AREA HAS BEEN ALLOCATED FOR THE SEGMENT, PRESENCEBIT CALLS "DISKIO", THE DISK INPUT/OUTPUT PROCEDURE, AND WAITS ON AN EVENT WHICH INDICATES THAT THE SEGMENT HAS BEEN READ IN.

3. "DISKIO" LINKS THE REQUEST INTO I/O QUEUE. WHEN THE I/O REQUEST COMES TO THE HEAD OF THE QUEUE, THE DISK I/O IS PERFORMED. AT THE COMPLETION OF THE DISK I/O, THE EVENT IS CAUSED, THEREBY NOTIFYING PRESENCEBIT THAT THE SEGMENT IS NOW AVAILABLE.

4. PRESENCEBIT MARKS THE SEGMENT DESCRIPTOR PRESENT AND EXITS BACK TO THE PROCESS AT THE POINT OF INTERRUPTION.

AS THE PROCESS RUNS, ADDITIONAL SEGMENTS OF PROGRAM CODE AND DATA WILL BE REQUIRED. THE PROCESS STACK CONTAINS THE STORAGE LOCATIONS FOR SIMPLE VARIABLES AND ARRAY DATA DESCRIPTORS, BUT PROGRAM CODE SEGMENTS AND ARRAY ROWS ARE ASSIGNED THEIR OWN AREAS OF MEMORY. THE ASSIGNMENT OF SEPARATE MEMORY AREAS FOR CODE SEGMENTS AND ARRAY ROWS ALLOWS SEGMENTS AND DATA TO BE ABSENT FROM MAIN MEMORY UNTIL THEY ARE ACTUALLY NEEDED. IN THE B6700 SYSTEM, A REFERENCE TO DATA OR CODE (THROUGH A DATA DESCRIPTOR OR A SEGMENT DESCRIPTOR) CAUSES THE PROCESSOR TO CHECK THE "PRESENCE" BIT, BIT NUMBER 47, IN THE DESCRIPTOR.

IF THE PRESENCE BIT IS OFF, AN INTERRUPT OCCURS WHICH TRANSFERS CONTROL TO THE PRESENCEBIT PROCEDURE IN THE MCP, PASSING THE NON-PRESENT DESCRIPTOR AS A PARAMETER. THE PRESENCEBIT PROCEDURE READS THE ADDRESS

FIELD OF THE DESCRIPTOR (THE ADDRESS FIELD CONTAINS THE DISK ADDRESS OF THE DATA OR SEGMENT FOR NON-PRESENT DESCRIPTORS). THEN PRESENCEBIT CALLS "GETSPACE" TO ALLOCATE A MEMORY AREA OF THE SIZE SPECIFIED IN THE DESCRIPTOR. GETSPACE RETURNS THE MEMORY ADDRESS OF THE AREA IT HAS ALLOCATED AND PRESENCEBIT CAUSES THE INFORMATION TO BE READ FROM DISK INTO MEMORY.

WHEN THE DISK READ IS FINISHED, PRESENCEBIT STORES THE MEMORY ADDRESS OF THE INFORMATION INTO THE ADDRESS FIELD OF THE DESCRIPTOR, TURNS THE PRESENCE BIT ON AND UPDATES THE DESCRIPTOR IN THE PROCESS STACK. PRESENCEBIT THEN RETURNS CONTROL BACK TO THE PROCESS WHICH WAS INTERRUPTED AND THE PROCESS HAS ACCESS AGAIN TO THE INFORMATION. NOW THE INFORMATION IS PRESENT IN MEMORY, WHICH IS INDICATED BY THE PRESENCE BIT BEING "ON". THE INFORMATION IS OBTAINED AND THE PROCESS EXECUTION CONTINUES IN THE NORMAL MANNER.

FOR PURPOSES OF DISCUSSION, ASSUME THAT THE PROCESS EXPECTS TO READ A DATA FILE WITH THE SYMBOLIC NAME "INCARD" AND THAT A CARD FILE TITLED "INCARD" HAS BEEN RECOGNIZED BY THE CONTROL CARD ROUTINE AND THE TITLE ENTERED IN THE MCP "UNIT" TABLE. WHEN THE PROCESS FIRST PERFORMS A READ OPERATION ON THE SYMBOLIC FILE "INCARD", THE PROCESS FILE INFORMATION BLOCK (FIB) IS ACCESSED. A BIT IN THE FIB INDICATES THAT THE FILE HAS NOT BEEN OPENED (I.E. THE LABEL EQUATION BLOCK (LEB) HAS NOT BEEN INITIALIZED). THE "FILE OPEN" ACTION CONSISTS OF FINDING THE PROPER FILE ON SOME PHYSICAL UNIT, PROVIDING A MEMORY I/O AREA FOR THE RECORDS OF THE FILE AND ASSIGNING THE UNIT TO THE PROCESS WHICH IS OPENING THE FILE. IN ORDER TO FIND THE PROPER FILE, THE PROCESS PARAMETER BLOCK (PPB) MUST BE USED TO DETERMINE IF THE NAME OF THE FILE HAS BEEN EQUATED TO SOME NAME OTHER THAN THE SYMBOLIC NAME DEFINED IN THE SOURCE PROGRAM. SEE SECTION 4.4 FOR MORE DETAILS.

SINCE THE SYMBOLIC FILE NAMED "INCARD" IS NOT LABEL-EQUATED TO ANOTHER NAME, THE SEARCH OF THE PPB IS UNSUCCESSFUL, AND THE SYMBOLIC NAME, INCARD, IS EXPECTED TO BE THE "TITLE" NAME OF THE FILE TO BE ASSIGNED. IN ORDER TO FIND THE PHYSICAL UNIT CONTAINING THE FILE TITLED "INCARD", THE "UNITINFO" TABLE IS SEARCHED AND THE PHYSICAL UNIT ASSOCIATED WITH

THE TITLE "INCARD" (A CARD READER IN THIS INSTANCE) IS ASSIGNED TO THE PROCESS. THE CARD READER IS MARKED AS BEING IN USE IN THE UNIT TABLE AND MEMORY IS ALLOCATED FOR THE REQUIRED NUMBER OF BUFFERS.

WHEN THE JOB OR INDEPENDENT RUNNER IS FINISHED, IT EXITS BACK TO RUN, WHICH CANNOT ITSELF EXIT (THERE IS NOWHERE TO GO). INSTEAD IT CALLS GEORGE, THAT WILL ABANDON THE STACK. RUN ALSO FIXES BOJ AND EOJ MESSAGES AND RECOGNIZES THE SPECIAL REQUIREMENTS OF COMPILERS, HANDLES SPECIAL TERMINATION TASKS AND DELINKS TASKS FROM THE PARENT STACK.

3.3.4. PRIORITY CONSIDERATIONS

THE PROCESSOR ALLOCATION ALGORITHM FOR THE B6700 IS DESIGNED TO ATTAIN MAXIMUM THROUGH-PUT FOR A JOB MIX. THIS IS ATTAINED BY THE MINIMIZATION OF INTER-JOB INTERFERENCE AND MCP OVERHEAD INVOLVED IN PROCESS SWITCHING.

THE BASIC STRATEGY USED IS TO ALWAYS RUN THE JOB FROM THE READYQ WHICH HAS THE HIGHEST PRIORITY. THIS SCHEME HAS ONE DISADVANTAGE IN THAT GIVEN TWO JOBS OF EQUAL PRIORITY, A SWAPPING SITUATION COULD CAUSE AN OSCILLATION TO DEVELOP SO THAT EACH JOB WAS REQUIRED TO OVERLAY THE OTHERS MAIN STORAGE IN ORDER TO OBTAIN ROOM TO RUN. IT COULD ALSO HAPPEN THAT A JOB WHICH WAS PROCESSOR BOUND COULD DOMINATE ONE WHICH WAS PRIMARILY WAITING FOR I/O COMPLETIONS, THUS GETTING LESS THAN OPTIMUM USE FROM THE I/O SUBSYSTEM. THE ALGORITHM USED HAS THEREFORE BEEN DESIGNED SO THAT NO TWO JOBS CAN HAVE EXACTLY THE SAME PRIORITY, AND THAT PROCESSOR BOUND JOBS WILL TEND TO THE BACKGROUND WHEN THERE ARE I/O BOUND JOBS AT THE SAME PRIORITY LEVEL. IN NO WAY CAN THE SYSTEM ADJUSTMENTS OF PRIORITY CAUSE A JOB TO ADVANCE OR RECEDE TO A PRIORITY LEVEL DIFFERENT FROM THE USER SPECIFIED PRIORITY NUMBER.

WHEREAS THIS SCHEME WILL CAUSE THE B6700 TO RUN AT MAXIMUM THROUGH-PUT, CONCERNS EXTERNAL TO THE OPERATING SYSTEM MAY MAKE OTHER PRIORITY MANIPULATIONS DESIRABLE. THERE ARE TWO KINDS OF PRIORITY MANIPULATIONS WHICH ARE APPROPRIATE TO BE CONSIDERED. PRIORITY IS OFTEN A FUNCTION OF OTHER ASPECTS OF A JOB SUCH AS ITS PROCESSOR AND I/O TIME ESTIMATES, ITS CORE ESTIMATE, ETC. WHEN THIS IS THE CASE, THE APPROPRIATE PRIORITY

SHOULD BE DETERMINED PRIOR TO SCHEDULING THE JOB. SINCE SCHEDULE ENTRIES ARE ALL ESTABLISHED BY THE ROUTINE SCHEDULE, AND ALL PROGRAM PARAMETER CARD INFORMATION IS AVAILABLE AT THAT TIME, AN APPROPRIATE USER DEFINED ALGORITHM CAN EASILY BE INSERTED TO COMPUTE THE PRIORITY FOR ALL JOBS INTRODUCED TO THE SYSTEM. THIS CHANGE IS RECOMMENDED WHEREVER SUCH AN ALGORITHM CAN BE DEFINED SINCE IT CAN IMPROVE SYSTEM PERFORMANCE AT INSIGNIFICANT COST. IT IS SOMETIMES DESIRABLE TO ALTER THE PRIORITIES OF JOBS WHICH ARE RUNNING BASED ON SOME PERFORMANCE CRITERION. THE DRAW BACK TO THIS IS THAT THE DECISION PROCESS REQUIRES SOME DEGREE OF SYSTEM OVERHEAD, AND UNLESS THE ORIGINAL PRIORITIES WERE IN IMPROPER RELATIONS, REARRANGEMENT OF RUNNING PRIORITIES WILL USUALLY DECREASE SYSTEM THROUGHPUT. THE MOST CONVENIENT MEANS OF DYNAMICALLY ADJUSTING PRIORITIES IS TO PERIODICALLY EXAMINE THE MIX AND COMPUTE NEW PRIORITIES. THE CALENDAR MECHANISM IS BEST SUITED FOR PERIODIC FUNCTIONS. THE ROUTINES GEORGE AND POST MANIPULATE THE CALENDAR, AND NSECOND IS AN EXAMPLE OF A PERIODICALLY RUN INDEPENDENT-RUNNER.

BURROUGHS CAN SUPPLY MORE DETAILED INFORMATION FOR THOSE DESIRING TO IMPLEMENT PRIORITY SCHEMES AS DESCRIBED; BUT BECAUSE OF THE INSTALLATION DEPENDENT CONSIDERATIONS REQUIRED, BURROUGHS HAS NO PLANS FOR INCLUDING SUCH SCHEMES IN THE STANDARD OPERATING SYSTEM.

3.3.5. PROCESS TERMINATION

TERMINATE IS A PROCEDURE FOR WINDING UP STACKS AND RECOVERING THE SYSTEM RESOURCES ALLOCATED TO THEM. IT MUST BE AN INDEPENDENT-RUNNER BECAUSE WHEN IT ENDS, THE SUBJECT STACK NO LONGER EXISTS, BUT TERMINATE IS REQUIRED TO BE CONTINUOUSLY PRESENT. IT TERMINATES ALL STACKS AS THEY ARE LINKED INTO A QUEUE CALLED "MORGUE". WHEN MORGUE IS EMPTY, TERMINATE WAITS ON AN EVENT CALLED "DEATH", GEORGE LINKS STACKS INTO THE MORGUE AND THEN CALLS DEATH.

WHEN A PROCESS EXECUTION IS TERMINATED, THE FOLLOWING ACTIONS OCCUR:

1. ANY OUTSTANDING I/O REQUESTS ARE COMPLETED (IF POSSIBLE). ANY "OPEN" FILES ARE CLOSED, THE UNITS RELEASED AND THE BUFFER AREAS

ARE RETURNED TO THE AVAILABLE MEMORY TABLE.

2. ALL OVERLAYABLE DISK AREAS ALLOCATED TO THE PROCESS ARE RETURNED TO THE AVAILABLE DISK TABLE.

3. ALL PROCESS OBJECT CODE AND DATA ARRAY AREAS OF MAIN MEMORY ARE RETURNED TO THE AVAILABLE MEMORY TABLE.

4. AN EOJ ENTRY IS MADE IN THE SYSTEM LOG FOR THE PROCESS.

5. THE PROCESS STACKS ARE LINKED INTO THE "TERMINATE" QUEUE.

3.4. TASKING

3.4.1. TASKING IMPLEMENTATION

THE B6700 MCP PROVIDES FACILITIES FOR CREATING AND CONTROLLING FAMILIES OF TASKS. THESE TASKS MAY COMMUNICATE AMONG THEMSELVES THROUGH PARAMETERS AND BY HAVING AN ACCESS TO COMMON DATA AREAS. SUCH A FAMILY WILL BE CALLED A TASK FAMILY, AND INDIVIDUAL MEMBERS WILL BE CALLED TASKS.

THE PRINCIPLE MEANS OF CONTROL AND COORDINATION WITHIN A FAMILY IS BY USE OF A UNIQUE DATA TYPE CALLED A TASK VARIABLE.

EACH TASK CONSISTS OF ONE ACTIVE STACK, AND EACH STACK HAS ASSOCIATED WITH IT ONE TASK VARIABLE.IN THE CASE OF TASKS STARTED BY CONTROL CARD AND INDEPENDENT TASKS THE TASK VARIABLE IS NOT A DECLARED ITEM, IN ALL OTHER TASKS STARTED BY ANOTHER TASK IT IS AN EXPLICITLY DECLARED ITEM. IT SHOULD BE NOTED HOWEVER, THAT INDEPENDENT TASKS ARE NOT MEMBERS OF THE FAMILY THAT STARTS THEM, AND THEIR TASK VARIABLES ARE COPIES OF WHAT IS USED TO START THEM.

ALL TASKS HAVING DECLARED TASK VARIABLES WILL BE REFERRED TO AS SUBTASKS ALL SUBTASKS ARE CREATED BY THE INTRINSIC "DELIVERY", BUT SINCE DELIVERY MUST BE PASSED A VARIABLE NUMBER OF PARAMETERS (BECAUSE THE SUBTASK MAY HAVE ANY NUMBER OF PARAMETERS) IT CALLS THE PROCEDURE DOCTOR TO PROCESS THE INTRINSIC DELIVERY.

DELIVERY IS CAPABLE OF CREATING DIFFERENT TYPES OF TASKS. THESE DIFFERENT TYPES ARE DESCRIBED AS:

	DEPENDENT	SYNCHRONOUS	COMPILER
IA	NO	NO	NO
IAC	NO	NO	YES
DA	YES	NO	NO

DAC	YES	NO	YES
DS	YES	YES	NO
DSC	YES	YES	YES

DEPENDENT MEANS THEY WILL BE A MEMBER OF THE CALLERS TASK FAMILY, SYNCHRONOUS MEANS THEY WILL RUN AS A COROUTINE, AND COMPILER MEANS THAT THE SYSTEM WILL RECOGNIZE CODE FILES GENERATED BY THEM.

EACH SUB-TASK HAS ASSOCIATED WITH IT A CRITICAL BLOCK. THIS IS THE HIGHEST ADDRESSING LEVEL THAT SOME ANCESTOR MAY NOT EXIT FROM WITHOUT RELEASING SOMETHING THAT THIS TASK HAS AN ACCESS INTO. IT MAY BE THE TASK VARIABLE, A CALL BY NAME PARAMETER, A PCW (POSSIBLY ITS OWN), OR VARIABLES GLOBAL TO ITSELF. TASKS ARE LINKED INTO FAMILIES BY MEANS OF THE FAMILY LINKED WORD WITHIN THE STACK OF EACH TASK

3.4.2. TASK ATTRIBUTES.

NUMBER	NAME	TYPE	CLASS	CC EQV
0	NAME	POINTER	0	
1	STACKNO	REAL	1	
2	COREESTIMATE	REAL	0	CORE
3	DECLAREDPRIORITY	REAL	0	PRIORITY
4	MAXPROCTIME	REAL	0	PROCESS
5	MAXIOTIME	REAL	0	IO
6	TARGETTIME	REAL	0	TARGETTIME
7	STACKSIZE	REAL	0	STACK
8	PREFIX	POINTER	0	
9	TASKVALUE	REAL	2	
10	HISTORY	REAL	1	
11	TYPE	TYPE	1	
12	STATUS	REAL	2	
13	PROCESSTIME	REAL	1	
14	PROCESSIOTIME	REAL	1	
15	STARTTIME	REAL	1	
16	EXCEPTIONTASK	TASK	2	

17	LOCKED	BOOLEAN	2
18	STOPPOINT	REAL	1
19	PARTNER	TASK	2
20	STATION	REAL	0
21	EXCEPTIONEVENT	EVENT	1

CLASSES 0: READ OR WRITE, BUT THE VALUE AT PROCESS INITIATION IS SAVED BY THE SYSTEM FOR USE THROUGHOUT PROCESS EXECUTION

1: READ ONLY

2: READ OR WRITE

NAME: THE NAME ATTRIBUTE IS REFERENCED IN THE SAME MANNER AS THE TITLE ATTRIBUTE OF A FILE. IT IS USED ONLY WHEN INITIATING AN EXTERNAL PROCEDURE, IN WHICH CASE IT IS THE FILE NAME OF THE CODE FILE TO BE USED AS THAT PROCEDURE.

STACKNO: STACKNO IS ZERO FOR A TASK VARIABLE THAT HAS NEVER BEEN THE TASK VARIABLE OF AN ACTIVE PROCESS , THE STACK NUMBER FOR AN ACTIVE PROCESS AND THE NEGATIVE OF THE STACKNO FOR TERMINATED PROCESSES.

COREESTIMATE: CORE REQUIREMENT USED FOR SCHEDULING BY THE SYSTEM.

DECLAREDPRIORITY: PRIORITY USED FOR SCHEDULING BY THE SYSTEM.

MAXPROCTIME: PROCESS TIME LIMIT FOR THE PROCESS.

MAXIOTIME: IO TIME LIMIT FOR THE PROCESS.

TARGETTIME: THE TARGET TIME FOR PROGRAM COMPLETION OF THE PROCESS, THIS IS USED IN SCHEDULING.

STACKSIZE: REQUIRED STACK SIZE FOR THE PROCESS.

PREFIX: FILE PREFIX FOR THE PROCESS. (THIS IS NOT CURRENTLY BEING USED BY THE FILE HANDLING SYSTEM).

TASKVALUE: PROVIDE SOLELY FOR USE BY THE PROGRAMMER AS A POSSIBLE MEANS OF COMMUNICATION AMONG PROCESSES.

HISTORY: THIS WILL PROVIDE THE REASON FOR TERMINATION OF TERMINATED PROCESSES. THE VALUES HAVE NOT BEEN SPECIFIED.

TYPE: TYPE OF PROCESS.

AD	0	ASYNCHRONOUS DEPENDENT NON-COMPILER
SD	1	SYNCHRONOUS DEPENDENT NON-COMPILER
I	2	INDEPENDENT NON-COMPILER
ADC	4	ASYNCHRONOUS DEPENDENT COMPILER
SDC	5	SYNCHRONOUS DEPENDENT COMPILER
IC	6	INDEPENDENT COMPILER

STATUS: CURRENT STATUS OF THE PROCESS,

0: NOT BEEN USED.
1: SCHEDULED.
2: ACTIVE.
3: SUSPENDED.
-1: EOJ TERMINATED

PROCESSTIME: ACCUMULATED PROCESSOR TIME FOR THE PROCESS.

PROCESSIOTIME: ACCUMULATED I/O TIME FOR THE PROCESS.

STARTTIME: THE TIME AT WHICH THE PROCESS WAS STARTED.

EXCEPTIONTASK: THIS ATTRIBUTE SPECIFIES WHAT PROCESS IS TO BE NOTIFIED OF CHANGES OF STATUS OF A PROCESS.

LOCKED: PROVIDED FOR USE BY THE PROGRAMMER. SETTING THIS TO TRUE, LOCKS THE ATTRIBUTE. IF IT IS ALREADY LOCKED, THE LOCKING PROCESS IS SUSPENDED UNTIL SOME OTHER PROCESS UNLOCKS IT.

STOPPOINT: STOPPOINT IS SET TO CONTAIN THE SEGMENT AND RELATIVE ADDRESS OF THE LAST ARITHMETIC FAULT OCCURRED IN THE PROCESS. IT IS ALSO SET UPON TERMINATION OR SUSPENSION

PARTNER: THE PARTNER ATTRIBUTE SPECIFIES WHAT THE CONTROL WILL PASS TO WHEN A SPECIFIED TASK VARIABLE IS EXECUTED.

STATION: RESERVED FOR DATACOM USAGE.

EXCEPTIONEVENT: THE EXCEPTIONEVENT FOR A TASK WILL BE CAUSED WHENEVER ANY PROCESS HAVING THIS TASK AS ITS EXCEPTIONTASK UNDERGOES A CHANGE IN STATUS.

NOTES

ANYTIME A PROCESS STARTS ANOTHER PROCESS, THE "STARTER" PROCESS WILL BECOME THE EXCEPTIONTASK OF THE "STARTED" PROCESS UNLESS AN EXCEPTIONTASK HAS BEEN EXPLICITLY STORED. THE SAME THING OCCURS FOR PARTNER IF THE "STARTED" PROCESS IS SYNCHRONOUS.

A REFERENCE TO EXCEPTIONEVENT IS VALID ONLY IF THE TASK IS ITS OWN OR THAT OF ONE OF ITS DIRECT ANCESTORS.

3.4.3. TASK VARIABLES

THESE TASK VARIABLES ARE ASSIGNED TO TASKS BY PROGRAMMER AND THEY CAN INTERRUPT OTHER TASKS.

CRITICAL BLOCK- BLOCK IN STACK OF PARENT WHICH IS HIGHEST BLOCK NUMBER WHICH ONLY OFFSPRING HAS REFERENCE TO. EXITING FROM THE CRITICAL BLOCK IS A TERMINAL ERROR. EXCEPTION EVENT: THE EXCEPTION EVENT IN THE STACK OF THE EXCEPTION TASK IS CAUSED EVERY TIME THE STATUS OF A PROCESS CHANGES.

3.5. MULTIPROGRAMMING

MULTIPROGRAMMING OPERATION ON A COMPUTER SYSTEM REQUIRES THE ABILITY TO INTERLEAVE EXECUTION OF PROCESSES. THIS MEANS THAT A PROCESSOR IS NOT EXCLUSIVELY ALLOCATED TO A PROCESS FOR THE ENTIRE EXECUTION OF THAT PROCESS. MULTIPROGRAMMING ON THE B6700 SYSTEM IS IMPLEMENTED BY QUEUEING PROCESSES IN THE "READYQ" QUEUE AND ALLOCATING (OR REALLOCATING) PROCESSORS TO THE PROCESSES WITH THE HIGHEST PRIORITY.

SINCE A PROCESSOR IS NOT ALLOCATED EXCLUSIVELY TO A PROCESS, THE PROCESS MUST CONTAIN ALL OF THE INFORMATION NECESSARY TO DESCRIBE ITS STATUS WHEN IT IS NOT BEING EXECUTED BY A PROCESSOR. THIS IS ACCOMPLISHED BY CREATING A SEPARATE "STACK" FOR EACH PROCESS THAT IS TO BE EXECUTED AND PERFORMING A MOVE STACK (MVST) INSTRUCTION.

THE HARDWARE MVST INSTRUCTION COPIES S, F, LL ETC. REGISTER SETTING INTO WORD ZERO FOR THE CURRENT STACK, CHANGES THE SNR (STACK NO.) REGISTER AS INDICATED BY THE B REGISTER, AND RESETS THE S, F, LL ETC. FROM WORD ZERO OF THE NEW STACK, REPLACING THAT WORD BY THE PROCESSOR ID. NO. (WHEN WORD ZERO CONTAINS THESE REGISTER SETTINGS, IT IS KNOWN AS A "TOP OF STACK CONTROL WORD" SINCE IT DESCRIBES THE TOP OF THE STACK.)

THE MVST INSTRUCTION IS CONTAINED IN "GEORGE" INSIDE ANOTHER PROCEDURE CALLED KEEPITMOVEING.

A STACK CAN HAVE TWELVE STATES:

PRENATEL	STACK NOT IN USE =AN AVAILABLE STACK NUMBER)
QUIESCENT	IN USE (BUT NOT AS A PROCESSSTACK)
FOETAL	STACK IN SHEET QUEUE
DELIVERY	OUT OF SHEET BUT NOT YET INITIATED
ALIVE	CAPABLE OF RUNNING
TIRED	WAITING ON EVENT
ASLEEP	HOLD LOOP (WAITING FOR SOFTWARE INTERRUPT)
LOOSE	STACK IS A COROUTINE THAT EXECUTED A VISIT STATEMENT (MOVED PROCEDURE OVER TO ANOTHER STACK)

ABANDONED	TERMINATES COROUTINES
DISEASED	PROCESSING IS TERMINATED
DEAD	BEING LINKED TO MORGUE
UNEMPLOYED	GEORGE FIRES ITSELF UP

A PROCESS IS "ACTIVE" WHEN IT IS BEING EXECUTED BY A PROCESSOR. AN ACTIVE PROCESS MAY BE MADE "INACTIVE" BY THE MCP IF A HIGHER PRIORITY PROCESS NEEDS A PROCESSOR. AN ACTIVE PROCESS MAY CAUSE ITSELF TO BE "SUSPENDED" BY EXECUTING A WAIT OR HOLD OR BY REQUESTING AN I/O OPERATION WHICH RESULTS IN A WAIT ON THE "I/O COMPLETE" EVENT.

WHEN AN ACTIVE PROCESS IS MADE INACTIVE BY THE MCP, IT IS PLACED IN THE READYQ. THE READYQ CONTAINS ONLY THOSE PROCESSES THAT ARE "READY TO RUN" AND ARE WAITING FOR A PROCESSOR.

IF AN ACTIVE PROCESS SUSPENDS ITSELF, THE SUBSEQUENT ACTION OF THE MCP DEPENDS UPON THE TYPES OF QUEUES WITH WHICH THE PROCESS IS ASSOCIATED. IF THE SUSPENDED PROCESS IS LINKED TO AN EVENT WAIT QUEUE, THE PROCESS WILL GET PUT INTO THE READYQ WHEN THAT EVENT IS CAUSED.

IF A PROCESS IS LINKED INTO A SOFTWARE INTERRUPT QUEUE (EVENT INTERRUPT QUEUE), THE PROCESS WILL BE MOVED TO THE READYQ UPON THE OCCURRENCE OF THE EVENT IF IT IS SUSPENDED OR IT WILL BE INTERRUPTED IF IT IS ACTIVE.

IF A SUSPENDED PROCESS IS NOT LINKED INTO AN EVENT QUEUE OR THE READYQ, IT CAN NOT BE REACTIVATED. FOR EXAMPLE, WHEN A PROCESS IS TERMINATED, THE PROCESS IS SUSPENDED, ANY QUEUE ENTRIES ARE DELINKED AND THE PROCESS IS LINKED INTO THE TERMINATE QUEUE.

3.6. PARALLEL PROCESSING

PARALLEL PROCESSING CAN OCCUR ON B6700 SYSTEMS WHICH HAVE DUAL PROCESSORS. THE FACT THAT MULTIPLE PROCESSORS ARE AVAILABLE DOES NOT PRECLUDE THE MULTIPROGRAMMING OF PROCESSES. IT MERELY MEANS THAT PROCESSES MAY BE EXECUTED SIMULTANEOUSLY TO INCREASE THE THROUGHPUT OF THE SYSTEM.

THE STRUCTURE OF THE MCP IS SUCH THAT PROCESSORS ARE CONSIDERED A RESOURCE TO BE ALLOCATED LIKE OTHER SYSTEM RESOURCES. THEREFORE, THE ONLY ADDITIONAL REQUIREMENTS FOR PARALLEL PROCESSING ARE THE INCLUSION OF SOME ADDITIONAL MCP "LOCK" VARIABLES TO PREVENT SIMULTANEOUS EXECUTION OF EXCLUSIVE MCP FUNCTIONS. FOR INSTANCE, IT WOULD NOT BE DESIRABLE TO HAVE TWO PROCESSORS SIMULTANEOUSLY TRYING TO MAKE AN ABSENT PROGRAM SEGMENT PRESENT IN MEMORY. THIS CIRCUMSTANCE IS PREVENTED BY AN MCP "LOCK" WHICH IS SET AND TESTED BY THE PRESENCE BIT PROCEDURE. THE FIRST PROCESSOR ENTERING PRESENCE BIT LOCKS ALL OTHERS OUT UNTIL IT IS SAFE FOR THEM TO ENTER THE PROCEDURE.

3.7. PROCESS TO PROCESS COMMUNICATION

SOFTWARE INTERRUPTS AND EVENTS

SOFTWARE INTERRUPTS ARE PROGRAMMATICALLY DEFINED FOR USE BY THE MCP AND OBJECT PROGRAM PROCESSES TO ALLOW PROCESSES TO COMMUNICATE WITH EACH OTHER AND WITH THE MCP.

SOFTWARE INTERRUPTS ALLOW A PROCESS EITHER TO STOP RUNNING (THEREBY RELEASING THE PROCESSOR) UNTIL A SPECIFIED EVENT OCCURS OR TO RUN AND BE INTERRUPTED IF AN EVENT OCCURS. A SOFTWARE INTERRUPT OCCURS WHEN A PROCESS IS INTERRUPTED BY THE DIRECT ACTION OF SOME OTHER PROCESS. IN THE FOLLOWING DISCUSSION THE IMPLEMENTATION OF THIS CONCEPT WILL BE DEVELOPED AS IT RELATES TO THE QUEUES, THE STACK STRUCTURE AND THE MCP ROUTINES THAT CONCERN THEMSELVES WITH SOFTWARE INTERRUPTS.

A PROCESS CAN BE INTERRUPTED IF IT HAS AN INTERRUPT DECLARATION WITHIN ITS SCOPE.

IF A BLOCK HAVING AN INTERRUPT DECLARATION IS EXECUTED, A STUFFED INDIRECT REFERENCE WORD AND A PROGRAM CONTROL WORD ARE PLACED IN THE STACK. THIS INTERRUPT DECLARATION MUST OCCUR WITHIN THE SCOPE OF ITS ASSOCIATED EVENT DECLARATION, AS SHOWN IN FIGURE F3-4.

```
          EXAMPLE: EVENT EVNT;
```

EXAMPLE

```
    EVENT  EVNT;

 [  PROCEDURE  A;
 [  INTERRUPT  I1: ON EVNT,  <STATEMENT>;
 [  ENABLE  (I1);
 [*

 [  PROCEDURE  B;
 [  INTERRUPT  I2; ON EVNT,  <STATEMENT>;
 [  ENABLE  (I2);
 [ *HOLD;

 [  PROCEDURE  C;
 [ *WAIT (EVNT);

 [  PROCEDURE  D;
 [ *WAIT  (EVNT);

 [  PROCEDURE  E;
 [  CAUSE  (EVNT);
```

FIGURE F3-4. EXAMPLE

AN EVENT DECLARATION RESERVES TWO WORDS IN THE STACK AND DEFINES THE IDENTIFIER OF A QUANTITY WHICH MAY BE USED TO RECORD AN OCCURRENCE. THE STACK CONTAINING THE INTERRUPT DECLARATION IS LINKED INTO THE EVENT INTERRUPT QUEUE BY THE EVENT DECLARATION AS SHOWN IN FIGURE F3-5. IF A BLOCK IS EXECUTED IN A SECOND PROCESS AND IF THE BLOCK CONTAINS AN INTERRUPT DECLARATION FOR THE SAME EVENT, THEN ITS STACK IS LINKED INTO THE EVENT INTERRUPT QUEUE AS SHOWN IN FIGURE F3-6. PROCESSES RUNNING IN STACK FOUR AND STACK TWO WILL CONTINUE TO RUN UNTIL THE EVENT OCCURS, WHEN THE EVENT IS CAUSED, ALL OF THE PROCESSES IN THE INTERRUPT QUEUE FOR THAT EVENT ARE INTERRUPTED. IF A PROCESS CAUSES THE OCCURRENCE OF AN EVENT, THE MCP SCANS THE EVENT INTERRUPT QUEUE. AS THE MCP SCANS THE EVENT INTERRUPT QUEUE, IT WILL CHECK TO SEE IF THE INTERRUPT HAS BEEN ENABLED.

```
EXAMPLE: ENABLE (12);
```

THE ENABLING OF AN INTERRUPT TURNS ON THE SOFTWARE INTERRUPT ENABLE BIT (BIT 46) OF THE PROGRAM CONTROL WORD OF THE TWO WORD INTERRUPT DECLARATION MENTIONED PREVIOUSLY. IF AN INTERRUPT IS NOT ENABLED AND THE EVENT IS CAUSED, NO ACTION IS TAKEN BY THE MCP ON THAT INTERRUPT AND IT LOOKS AT THE NEXT INTERRUPT IN THE QUEUE.

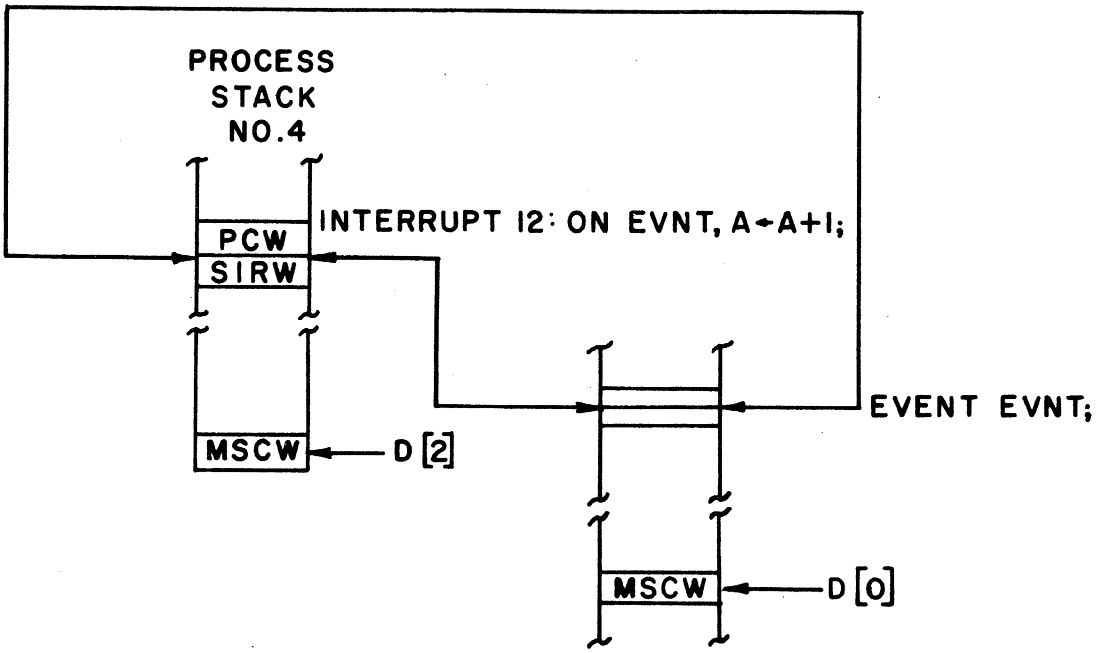


FIGURE F3-5. EVENT INTERRUPT QUEUE, SINGLE PROCESS

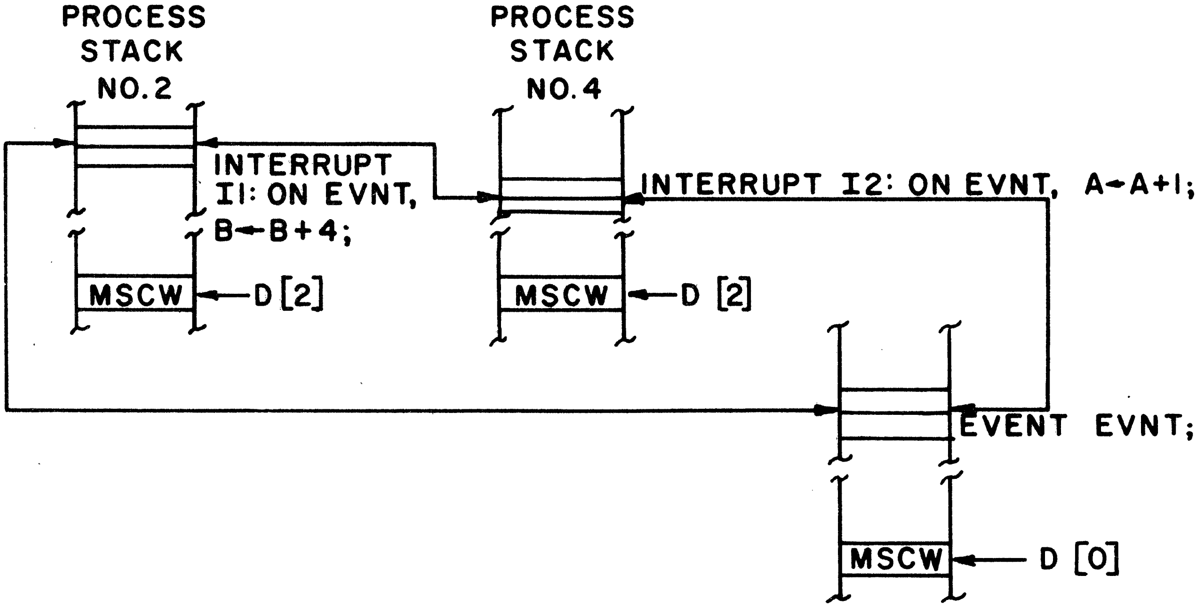


FIGURE F3-6. EVENT INTERRUPT QUEUE, MULTIPLE PROCESS

IF INTERRUPTS ARE ENABLED FOR THAT STACK, THE MCP MAKES AN ENTRY IN THE SOFTWARE INTERRUPT QUEUE. IF THE STACK IS ACTIVE, I.E. ANOTHER PROCESSOR IS WORKING IN THE STACK, THE MCP WILL INTERRUPT THAT PROCESSOR WITH A PROCESSOR TO PROCESSOR INTERRUPT.

THE MCP NEXT FORCES A TRANSFER OF CONTROL TO THE STATEMENT RELATED TO THE INTERRUPT DECLARATION. UPON COMPLETION OF THIS STATEMENT, THE PROCESS WILL RETURN TO ITS PREVIOUS POINT OF CONTROL UNLESS A TRANSFER OF CONTROL IS SPECIFIED IN THE INTERRUPT STATEMENT. IN THIS CASE, THE PROCESS WILL NOT RETURN THE POINT OF CONTROL BEFORE THE INTERRUPT, BUT WILL TRANSFER CONTROL AS SPECIFIED IN THE INTERRUPT STATEMENT.

AS THE MCP SCANS THE EVENT INTERRUPT QUEUE FINDING ENABLED INTERRUPTS IN INACTIVE STACKS IT MAKES AN ENTRY IN THE SOFTWARE INTERRUPT QUEUE DOING NOTHING WITH THAT STACK UNTIL IT BECOMES ACTIVE. IMMEDIATELY AFTER MAKING THE STACK ACTIVE, THE MCP CHECKS THE SOFTWARE INTERRUPT QUEUE TO SEE IF THERE IS AN INTERRUPT POINTING TO THAT STACK. IF AN INTERRUPT IS FOUND, THE MCP FORCES A TRANSFER OF CONTROL TO THE STATEMENT REFERRED TO BY THE INTERRUPT DECLARATION. UPON COMPLETION OF THE STATEMENT, CONTROL IS TRANSFERRED AS DESCRIBED ABOVE.

IT IS POSSIBLE FOR A PROCEDURE TO BE ENTERED, GET LINKED INTO THE EVENT INTERRUPTQUEUE AND THEN EITHER EXIT FROM THE PROCEDURE WITHOUT ENABLING THE INTERRUPT OR EXIT FROM THE PROCEDURE BEFORE THE EVENT IS CAUSED. IN EITHER CASE THIS INTERRUPT IS UNLINKED FROM THE EVENT INTERRUPT QUEUE

IF A PROCESS ENABLED A SOFTWARE INTERRUPT, IT IS SOMETIMES DESIRABLE TO SUSPEND FURTHER PROCESSING OF THE CODE UNTIL AN ENABLED SOFTWARE INTERRUPT OCCURS. THIS SUSPENSION CAN BE BROUGHT ABOUT BY USING THE HOLD STATEMENT.

```
        EXAMPLE: ENABLE (12);
                        .
                        .
                        .
```

```
                ENABLE (13);
                      .
                      .
                      .
                HOLD;
```

WHERE HOLD IS A PROCEDURE CALL ON THE PROCEDURE GEORGE.

WHEN AN EVENT IS CAUSED AND THE RELATED INTERRUPT STATEMENT EXECUTED CONTROL WILL PASS TO THE STATEMENT FOLLOWING THE HOLD.

A PROCESS CAN ALSO BE SUSPENDED BY THE EXECUTION OF A WAIT STATEMENT.

EXAMPLE: WAIT (EVNT);

THE PARAMETER OF A WAIT STATEMENT IS AN EVENT WHOSE SCOPE INCLUDES THE BLOCK IN WHICH THE WAIT RESIDES. UPON EXECUTION OF WAIT, THE STACK OF THAT PROCESS IS LINKED TO THE EVENT DECLARATION FOLLOWING AN EVENT WAIT QUEUE AS SHOWN IN FIGURE F3-11.

STACKS ARE REMOVED FROM THE WAIT QUEUE WHEN ANOTHER PROCESS EXECUTES A CAUSE STATEMENT.

EXAMPLE: CAUSE (EVNT):

STACKS REMOVED FROM THE WAIT QUEUE ARE LINKED INTO THE READY QUEUE. THE STACKS SHOWN IN FIGURE F3-11 REPRESENT THE EVENT INTERRUPT QUEUE AND THE WAIT QUEUE AT A POINT IN TIME WHEN THE PROCEDURES ARE AT A PLACE IN THEIR CODE STRING INDICATED BY THE ASTERISK (*) AS SHOWN IN FIGURE F3-7. PROCEDURE A IS RUNNING IN PROCESS STACK FOUR, PROCEDURE B IN STACK TWO, PROCEDURE C IN STACK TWENTY, AND PROCEDURE D IN STACK SEVEN. BOTH QUEUES ARE LINKED TO THE EVENT DECLARATION IN THE D[0] STACK.

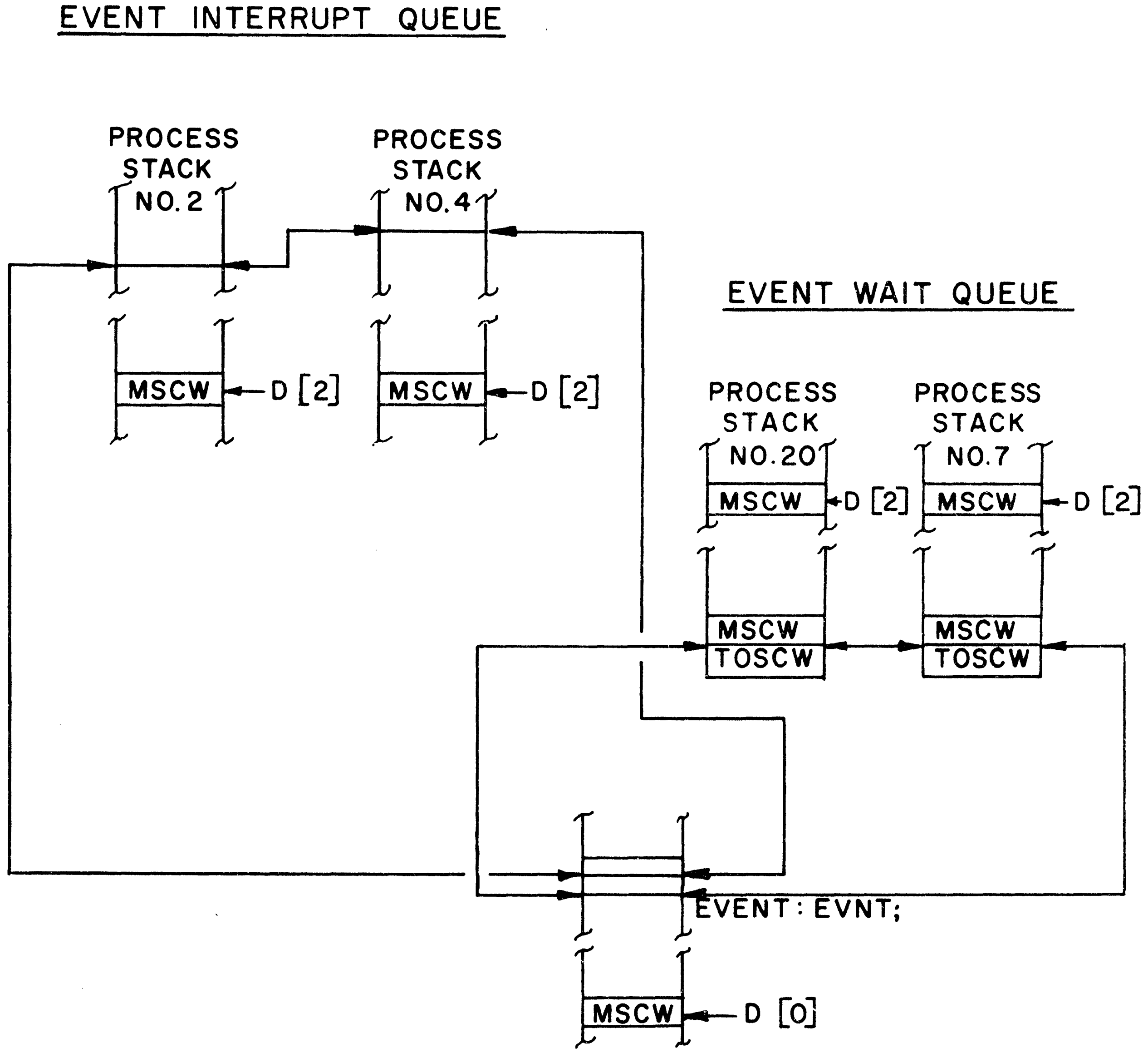


FIGURE F3-7. EVENT QUEUES

SECTION 4

FILE HANDLING

4. FILE HANDLING

4.1. INTRODUCTION

SINCE THE B6700 COMPILERS ALLOW THE USE OF SYMBOLIC FILES, THE MCP MUST BE ABLE TO RECOGNIZE THE PHYSICAL FILES PRESENT ON THE PERIPHERAL UNITS AND ASSIGN THE UNITS TO A SYMBOLIC PROCESS FILE. THE FILE CONTROL FUNCTIONS OF THE MCP CONSIST OF RECOGNIZING THE EXISTENCE OF A FILE ON A PERIPHERAL UNIT AND ASSIGNING THE PERIPHERAL UNIT TO THE APPROPRIATE PROCESS.

TO RECOGNIZE AND ASSIGN A FILE, INFORMATION ABOUT THE PHYSICAL FILE IS RECORDED IN THE UINFO TABLE AND THE UNIT TABLE.

THE UINFO TABLE, OR LABEL TABLE, CONTAINS DESCRIPTORS THAT POINT TO LABEL INFORMATION, AND IS INDEXED BY UNIT NUMBER (OR LOGICAL UNIT NUMBER IF LOGICAL UNIT NUMBERS ARE UNIQUE). IT IS MAINTAINED BY THE PROCEDURES READALABEL AND CONTROLCARD.

THERE ARE TWO TYPES OF PHYSICAL FILES RECOGNIZED BY THE B6700 SYSTEM, LABELED FILES AND UNLABELED FILES.

LABELED FILES ARE THOSE FILES WHICH CONTAIN A LABEL RECORD (OR RECORDS) AS THE FIRST RECORD OF A FILE. SINCE THE LABEL RECORD CONTAINS A FILE LABEL NAME, THE MCP CAN RECOGNIZE THE EXISTENCE OF A LABELED FILE AND ASSOCIATE THE APPROPRIATE PERIPHERAL UNIT WITH A SYMBOLIC PROCESS FILE. NO OPERATOR ASSOCIATION OF JOB AND PERIPHERAL UNITS IS REQUIRED.

UNLABELED FILES, HOWEVER, MUST BE ASSIGNED BY THE OPERATOR AT THE TIME THAT A PROCESS REQUIRES ACCESS TO THE FILE.

4.2. FILE LABEL FORMAT FOR PERIPHERAL UNITS

THE FORMAT OF FILE LABELS FOR VARIOUS TYPES OF PERIPHERAL UNITS ARE DESCRIBED IN THE FOLLOWING SECTIONS. THE PHYSICAL FILE NAMING SYSTEM USED ALLOWS FILE NAMES TO BE FORMED FROM A SEQUENCE OF FILE IDENTIFIERS SEPARATED BY SLASHES. A FILE IDENTIFIER IS DELIMITED BY A BLANK OR A SLASH AND MAY BE OF ANY LENGTH, BUT ONLY THE FIRST 17 CHARACTERS ARE USED IF THE IDENTIFIER EXCEEDS 17 CHARACTERS IN LENGTH.

THE FOLLOWING ARE EXAMPLES OF FILE NAMES:

```
          A
          B/C
          D/E/F
          G/H/I/J
WHERE  A, C, F AND J ARE FILE IDENTIFIERS,
       B, E AND I ARE VOLUME IDENTIFIERS AND
       D, H AND G ARE FILE DIRECTORY IDENTIFIERS.
```

NOTE: TWO FILE NAMES SUCH AS A/B AND A/B/C CANNOT BE USED ON THE SYSTEM SINCE A/B WOULD INDICATE BOTH A FILE AND A DIRECTORY.

THE ORGANIZATION OF FILES IS DEPENDENT ON THE I/O DEVICES HOLDING THE FILE, EACH OF WHICH IS DISCUSSED INDIVIDUALLY.

4.2.1. CARD FILES

THE FORMAT OF CARD FILES IS AS FOLLOWS:

```
          LABEL CARD
          (DATA DECK)
          END CARD
```

THE FORMAT OF THIS LABEL CARD IS:

```
<I> DATA <FILE NAME> . <ANY COMMENT>
OR
<I> BCL <FILE NAME>.  <ANY COMMENTS>
OR
<I> BINARY <FILE NAME>.
<ANY COMMENT>
```

AN EXAMPLE OF A LABEL CARD IS:

```
<I> DATA CARD
```

THE FORMAT OF AN END CARD IS:

```
<I> END <ANY COMMENT>
```

THE <I> REPRESENTS AN INVALID CHARACTER AND MUST BE IN COLUMN 1. DATA INDICATES THE <DATA DECK> IS PUNCHED USING THE EBCDIC (8 BIT) CHARACTER SET. "BCL" INDICATES THE DATA DECK IS PUNCHED USING THE BCL (6 BIT) CHARACTER SET. EXCEPT FOR THE INVALID CHARACTER IN COLUMN 1, THE CARD IS FREE FIELD.

4.2.2. PRINTER FILES

UPON OPENING A LABELED PRINTER FILE, THE OPERATING SYSTEM WILL:

SKIP TO TOP OF PAGE,
WRITE THE HEADER LABEL RECORD(S) AND
SKIP TO TOP OF PAGE.

UPON CLOSING A LABELED PRINTER FILE, THE OPERATING SYSTEM WILL:

SKIP TO TOP OF PAGE,
WRITE A TRAILER LABEL RECORD AND
SKIP TO TOP OF PAGE.

HEADER AND TRAILER LABELS ARE IN STANDARD USASI FORMAT.

4.2.3. CARD-PUNCH

THE FORMAT OF A CARD DECK PRODUCED AT THE CARD PUNCH IS:

```
          LABEL RECORD
          (DATA DECK)
          LABEL RECORD
```

THE FORMAT OF THE LABEL RECORD IS:

```
          <I> <MODE> <FILE NAME>

       <MODE> ::= BCL/BINARY/DATA
```

AN EXAMPLE OF A LABEL RECORD IS:

```
          BINARY DECKA
```

THE ENDING RECORD OF A CARD PUNCH FILE HAS THE FORM

```
          <I> END <FILE-NAME>
```

FOR BCL AND EBCDIC FILE OR A BURROUGHS BINARY END ("BEND") CARD FOR BINARY FILES.

4.2.4. PAPER TAPE

PAPER TAPE FILES ARE ALWAYS CONSIDERED UNLABELED. FOR HANDLING OF UNLABELED FILES, SEE UNLABELED TAPE.

4.2.5. UNLABELED TAPE FILES

UNLABELED TAPE FILES ARE THOSE WHICH DO NOT HAVE ANY WAY OF BEING SELF-IDENTIFIED. THE SYSTEM ASSUMES FOR INPUT OR GENERATES FOR OUTPUT THE

FOLLOWING DATA FORMATS:

SINGLE FILE VOLUMES	<DATA> **
MULTI-FILE VOLUMES	<DATA> * <DATA>*---------*<DATA>**

WHERE * DENOTES A TAPE MARK.

THE SOURCE LANGUAGES CAN SPECIFY THAT INPUT AND OUTPUT FILES ARE TO BE UNLABELED. TO PRODUCE MULTI-FILE VOLUMES THE SOURCE PROGRAM MUST CLOSE WITH NO REWIND, THEN OPEN OUTPUT WITH NO REWIND FOR EACH DATA SET ON THE VOLUME (CLOSE WITH NO REWIND PRODUCES A TAPE MARK). WHEN A SINGLE FILE VOLUME OR MULTI-FILE VOLUME IS CLOSED COMPLETELY, THE SYSTEM PRODUCES THE DOUBLE TAPE MARK AT THE END. WHEN, IN THE PROCESS OF CREATING THE FILE, AND WHEN PHYSICAL END OF TAPE IS ENCOUNTERED, THE OPERATING SYSTEM WRITES THE DOUBLE TAPE MARK AND ASSIGNS ANOTHER TAPE.

WHEN AN UNLABELED FJLE IS REQUESTED FOR INPUT AND NO "UNIT" CONTROL STATEMENT HAS BEEN SEEN, THE OPERATOR IS NOTIFIED BY A " <MIX> NO FILE <FILE NAME> " MESSAGE. THE OPERATOR MUST MOUNT THE FILE AND ENTER THE <MIX> UL <UNIT DESIGNATE> MESSAGE. IF A "UNIT" CONTROL STATEMENT WAS SPECIFIED, THE SPECIFIED UNIT WILL BE ASSIGNED TO THE FILE. IF A TAPE MARK IS ENCOUNTERED, THE OBJECT PROGRAM IS NOTIFIED VIA AN END-OF-FILE CONDITION. TO READ THE DATA SET FOLLOWING A SINGLE TAPE MARK, THE OBJECT CODE MUST CLOSE NO REWIND, THEN OPEN INPUT.

4.2.6. LABELED TAPE-FILES

THE OPERATING SYSTEM WILL RECOGNIZE TWO LABELING CONVENTIONS FOR TAPE INPUT FILES: THE B5500 LABEL RECORD AND THE PROPOSED USASI STANDARD TAPE LABEL.

THE SYSTEM WILL PRODUCE ONLY THE USASI LABEL FORMAT FOR LABELED OUTPUT TAPES. THE FORMAT OF THE VARIOUS RECORDS OF THE USASI LABEL IS SHOWN IN THE USASI LABEL FORMAT.

THE USER CAN SPECIFY THE CREATION OF SINGLE FILE VOLUMES OR MULTIFILE

VOLUMES. IN ADDITION, THE OPERATING SYSTEM WILL, FOR EITHER OF THE ABOVE CASES, DO VOLUME SWITCHING WHEN THE DATA BEING WRITTEN EXCEEDS THE CAPACITY OF A VOLUME. IT WILL ALSO DO AUTOMATIC VOLUME SWITCHING ON INPUT WHEN REQUIRED. THE TAPE FORMAT IS SHOWN AS FOLLOWS (NOTE - "*" DENOTES TAPE MARK):

```
        SINGLE FILE - SINGLE VOLUME
            VOL1   HDR1   HDR2   *   DATA * EOF1   EOF2 **

        MULTI - VOLUME FILE
            VOL1   HDR1   HDR2   *   FIRST VOLUME DATA * EOV1 **
            VOL1   HDR1   HDR2   *   LAST VOLUME DATA * EOF1   EOF2 **

        MULTI - FILE VOLUME
            VOL1   HDR1   HDR2   *   FILE 1 * EOF1   EOF2 *
            HDR1   HDR2   *   FILE 2 * EOF1   EOF2 **

        MULTI - FILE MULTI VOLUME
            VOL1   HDR1   HDR2   *   FILE 1 * EOF1   EOF2 *
            HDR1   HDR2   *   FIRST PART FILE 2 * EOV1 **
            VOL1   HDR1   HDR2   *   PART OF FILE 2 * EOV1 **
            VOL1   HDR1   HDR2   *   REMAINDER FILE 2 * EOF1   EOF2 *
            HDR1   HDR2   *   FILE 3 * EOF1   EOF2 **
```

USER HEADER LABELS MAY APPEAR IMMEDIATELY AFTER [HDR2] AND USERS TRAILER LABELS MAY APPEAR AFTER EITHER [EOF2] OR [EOV1].

TO CREATE OR READ MULTI-FILE VOLUMES, THE USER MUST SPECIFY THE SAME VOLUME NAME FOR ALL THE FILES IN THE SET. ONLY ONE FILE IN THE SET CAN BE OPENED AT A TIME. TO CREATE A MULTI-FILE VOLUME, THE USER MUST CLOSE NO-REWIND, THE CURRENT FILE IN THE SET, AND USE OPEN OUTPUT NO-REWIND FOR THE NEXT FILE IN THE SET.

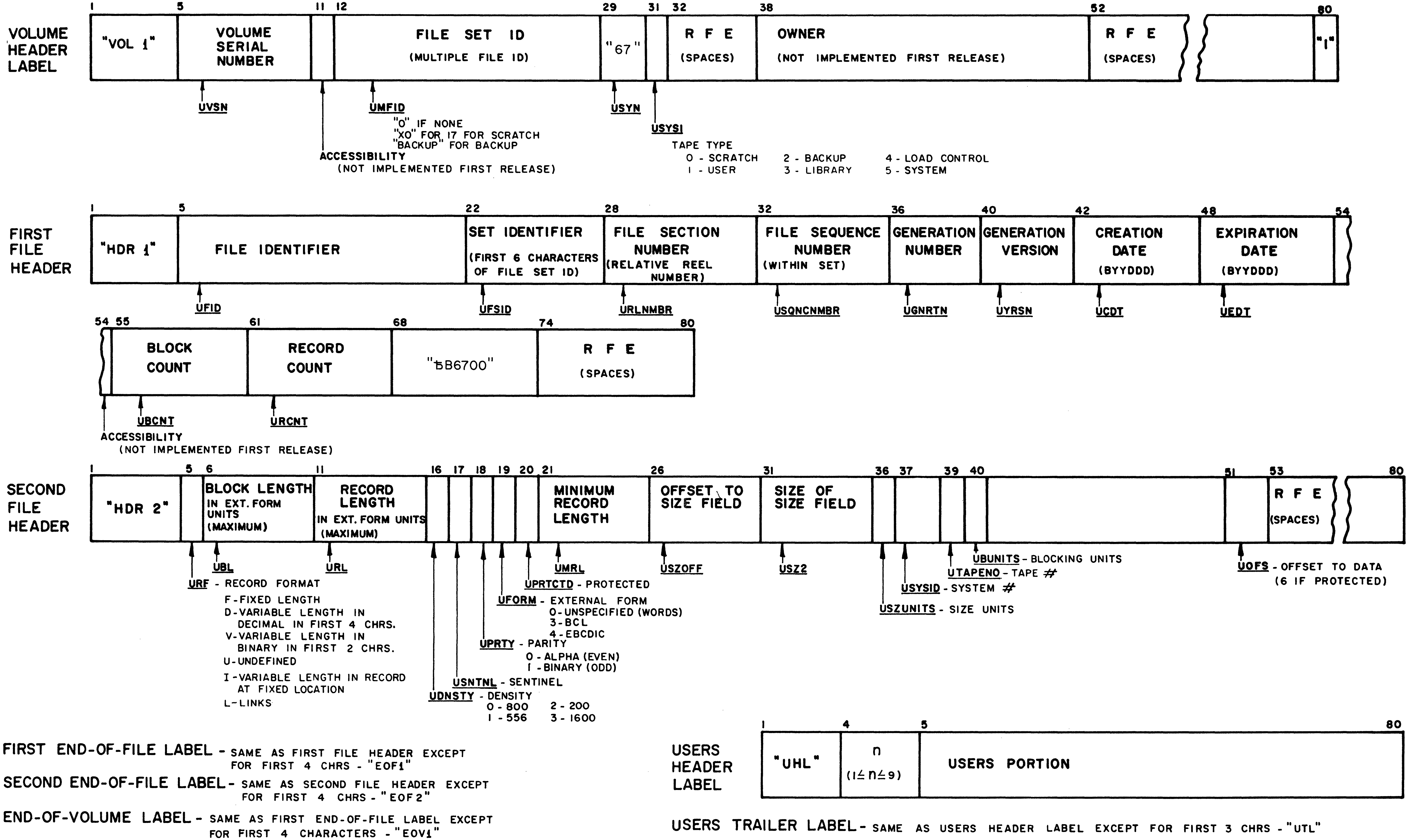
TO HANDLE INPUT, THE OPERATING SYSTEM WILL GIVE BACK TO THE OBJECT CODE AN [END-OF-FILE] CONDITION WHEN AN [EOF] LABEL IS ENCOUNTERED. THE USER THEN MUST CLOSE NO-REWIND ON THE CURRENT FILE AND OPEN INPUT NO-REWIND

ON THE NEXT (OR SOME OTHER) FILE IN THE SET.

THE [EOV] LABEL, WHEN ENCOUNTERED ON INPUT, IS THE SENTINEL BY WHICH THE OPERATING SYSTEM CAN DETECT WHEN VOLUME SWITCHING IS REQUIRED. THIS IS DONE BY LOCATING THE NEXT VOLUME OR REQUESTING THE OPERATOR TO LOAD A VOLUME WHICH HAS THE SAME VOLUME NAME AS THE CURRENT VOLUME AND HAS A FILE SECTION NUMBER (IN HDR1) ONE GREATER THAN THE CURRENT VOLUME.

IT IS INTENDED THAT THE VOLUME SERIAL NUMBER IN THE VOL1 LABEL IS USED AS A PHYSICAL LOCATION NUMBER. WHEN AN EMPTY REEL OF TAPE IS PRESENTED TO THE SYSTEM, THE OPERATOR MUST INDICATE THE TAPE IS AVAILABLE FOR OUTPUT AND WHAT VOLUME SERIAL NUMBER IS TO BE ASSOCIATED WITH THE TAPE BY ENTERING: PG <UNIT DESIGNATE> <VOLUME SERIAL NUMBER> THIS WILL CAUSE A SCRATCH LABEL CONTAINING THE VOLUME SERIAL NUMBER, TO BE WRITTEN ON THIS TAPE. LATER, WHEN FILE CONTROL ASSIGNS AN OUTPUT FILE TO THE UNIT CONTAINING A SCRATCH LABEL, THE VOLUME SERIAL NUMBER IS READ AND PLACED IN VOL1 LABEL OF THE VOLUME BEING CREATED. IF THE USER HAS ALSO SPECIFIED A FILE NAME CONTAINING ONE OR MORE DIRECTORY IDENTIFIERS, THE HIERARCHICAL STRUCTURE FOR THE VOLUME AND THE VOLUME SERIAL NUMBER IS ENTERED INTO THE DIRECTORY. LATER, IF THE FILE IS REQUESTED FOR INPUT, THE OPERATOR CAN BE NOTIFIED AS TO THE PHYSICAL LOCATION OF THE VOLUME CONTAINING THE FILE, IF IT IS NOT ALREADY MOUNTED ON A TAPE DRIVE.

VOLUME SERIAL NUMBER ZERO (0) CAN BE USED FOR TAPES GENERATED ON THE SYSTEM, BUT WHICH ARE TO BE USED ELSEWHERE. FOR THIS REASON, VOLUME SERIAL NUMBER (0) CANNOT BE USED FOR TAPES WHICH ARE TO BE CONTROLLED THROUGH THE DIRECTORY.



FIRST END-OF-FILE LABEL - SAME AS FIRST FILE HEADER EXCEPT FOR FIRST 4 CHRS - "EOF1"

SECOND END-OF-FILE LABEL - SAME AS SECOND FILE HEADER EXCEPT FOR FIRST 4 CHRS - "EOF2"

END-OF-VOLUME LABEL - SAME AS FIRST END-OF-FILE LABEL EXCEPT FOR FIRST 4 CHARACTERS - "EOV1"

USERS TRAILER LABEL - SAME AS USERS HEADER LABEL EXCEPT FOR FIRST 3 CHRS - "UTL"

FIGURE F4-1. B6700 USASI FILE HEADERS

4.2.7. DISK FILE STRUCTURE

4.2.7.1. DISK FILE AREA

EACH DISK ADDRESS REFERENCES A DISK SEGMENT, WHICH IS AN AREA OF DISK, CONTAINING ROOM FOR 30 WORDS OF INFORMATION. A DISK FILE CONSISTS OF A FILE HEADER AND A NUMBER OF AREAS, WHICH ARE NOT NECESSARILY CONTIGUOUS WITH EACH OTHER. EACH DISK AREA IS AN UNINTERRUPTED SEQUENCE OF SEGMENTS AND ALL OF THE AREAS FOR A GIVEN FILE HAVE THE SAME SIZE. A FILE HEADER IS AN UNINTERRUPTED SEQUENCE OF DISK SEGMENTS OF VARIABLE LENGTH DEPENDING UPON THE NUMBER OF AREAS USED BY THE FILE. (SEE FIGURE F4-2).

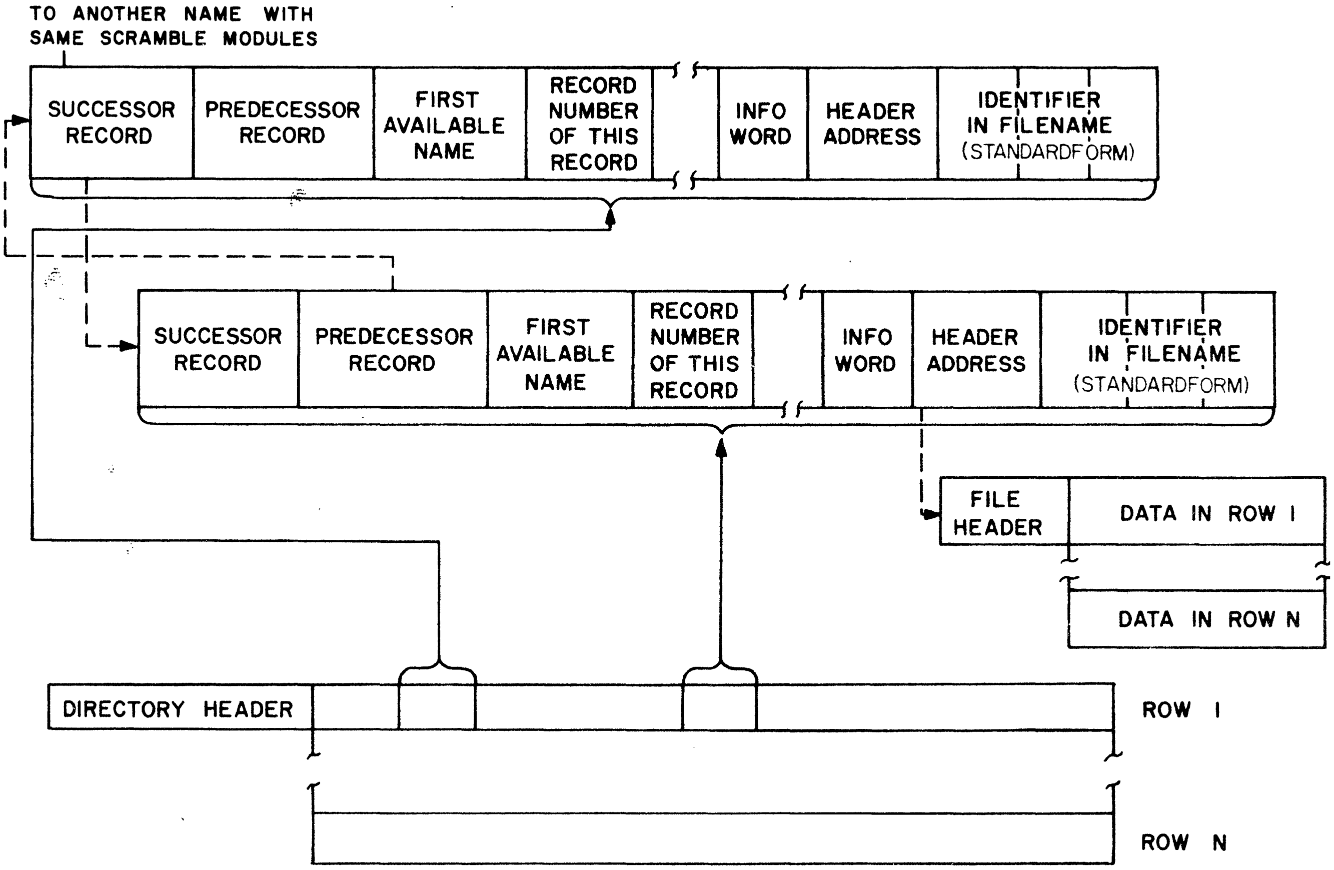


FIGURE F4-2. B6700 FILE DIRECTORY

4.2.7.1.1. FILE HEADER

THE FILE HEADER CONSISTS OF A VARIABLE NUMBER OR WORDS, DEPENDING ON THE NUMBER OF AREAS ASSIGNED TO THE FILE. THE FIRST TEN WORDS CONTAIN THE FOLLOWING INFORMATION:

WORD 0 CONTAINS THE CORE INDEX OR CORE INDEX. THE DISK ADDRESS IS -1 IF THERE ARE NO CURRENT USERS OF THE FILE. IF THERE ARE USERS, THE INDEX FIELD POINTS TO THE COPY OF THE HEADER IN MAIN MEMORY.

WORD 1 CONTAINS AN UPDATE BIT THAT IS TRUE IF THE FILE IS UPDATED, AND FIELDS FOR THE NUMBER OF PROCESSORS LOOKING AT THE HEADER, FILE TYPE (PROGRAM, DIRECTORY...), SIZE (IN WORDS) OF THE HEADER, CLASS OF SECURITY, SIZE (IN WORDS) OF THE SECURITY INFORMATION IN THE HEADER, DISK FILE SPEED AND LOGICAL MODE (SUCH AS EBCDIC, DOUBLE, ETC.).

WORD 2 CONTAINS INFORMATION ABOUT FILE ORGANIZATION AND RECORD TYPE, AND INDICATES WHETHER OR NOT THE FILE IS PACKED, TEMPORARY, PROTECTED, CRUNCHED OR EXCLUSIVE USE IS REQUESTED.

WORD 3 GIVES THE BLOCK SIZE, AND MAXIMUM AND MINIMUM RECORD SIZE OF THE FILE.

WORD 4 CONTAINS THE END OF FILE COUNT - THE RELATIVE NUMBER OF THE LAST LOGICAL RECORD IN THE FILE. THE COUNT IS -1 WHEN THE FILE IS EMPTY.

WORD 5 CONTAINS ROW INFORMATION - THE NUMBER OF ROWS FOR WHICH ROW ADDRESS WORDS ARE ASSIGNED, THE SIZE (IN SEGMENTS) OF EACH ROW, AND THE HEADER FORMAT. ROW ADDRESS WORDS CONTAIN VOLUME TYPE, VOLUME OR EV UNIT AND BLOCK ADDRESS AT BLOCK NUMBER.

WORD 6 CONTAINS NAME QUALIFICATION INFORMATION, INCLUDING THE SAVE FACTOR, CREATION DATE, USASI GENERATION NUMBER, AND MAXIMUM NUMBER OF GENERATIONS.

WORD 7 CONTAINS THE DATE THE FILE WAS LAST USED.

WORD 8 IS USED FOR VARIOUS PURPOSES DEPENDING ON THE FILE KIND. FOR DIRECTORIES, IT IS A SCRAMBLE MODULUS. FOR CODE FILES, IT IS THE STACK NUMBER FOR THE D[1] STACK WHEN A COPY OF THE PROGRAM IS RUNNING. FOR PSEUDO READER DECKS, IT IS THE DECK NUMBER.

WORD 9 IS THE NEXT AVAILABLE RECORD NUMBER IN A DIRECTORY.

WORDS 10 - 14 ARE CURRENTLY UNUSED.

STARTING WITH WORD 15 THERE IS A WORD FOR EACH DISK AREA ASSIGNED TO THE FILE. THE WORD CONTAINS THE ABSOLUTE DISK ADDRESS OF THE FIRST SEGMENT OF THE AREA. IF THE AREA HAS NOT YET BEEN ACCESSED (ALLOCATED BY THE MCP) THE DISK ADDRESS IS ZERO. NOTE: FILE HEADER FORMAT IS SUBJECT TO CHANGE.

4.2.7.1.2. DISK FILE RECORDS

THE RECORDS IN THE FILE ARE ADDRESSED RELATIVE TO THE BEGINNING OF THE FILE, WHERE 0 IS THE FIRST RECORD AND N IS THE LAST RECORD IN THE FILE. ASSUMING THAT THERE ARE 1000 RECORDS PER AREA AND RECORD 2345 IS REQUESTED, THE DISK AREA REQUIRED IS AREA NUMBER 2 (2345 DIV 1000). KNOWING THE DISK AREA NUMBER, THE INITIAL DISK ADDRESS OF THIS AREA IS OBTAINED FROM THE APPROPRIATE ADDRESS CARDS. THE DISK ADDRESS OF THE SEGMENT CONTAINING THE BEGINNING OF THE RECORD IS THEN COMPUTED BY ADDING THE AREA INITIAL ADDRESS TO (2345 MOD 1000)xK, WHERE K=((THE NUMBER OF WORDS PER LOGICAL RECORD +29) DIV 30), THE NUMBER OF SEGMENTS REQUIRED FOR A SINGLE RECORD.

4.2.7.2. DISK DIRECTORY

ALL FILES ON THE B6700 SYSTEM ARE REFERRED TO BY AN "ACTUAL FILE NAME" OR LABEL. THE ACTUAL FILE NAME IS A SEQUENCE OF IDENTIFIERS SEPARATED BY THE SYMBOL "/". EACH IDENTIFIER MAY BE OF ARBITRARY LENGTH, BUT IF THE IDENTIFIER IS LONGER THAN 17 CHARACTERS ONLY THE FIRST 17 WILL BE USED. ANY NUMBER OF IDENTIFIERS UP TO 14 MAY BE USED TO CONSTRUCT A FILE NAME.

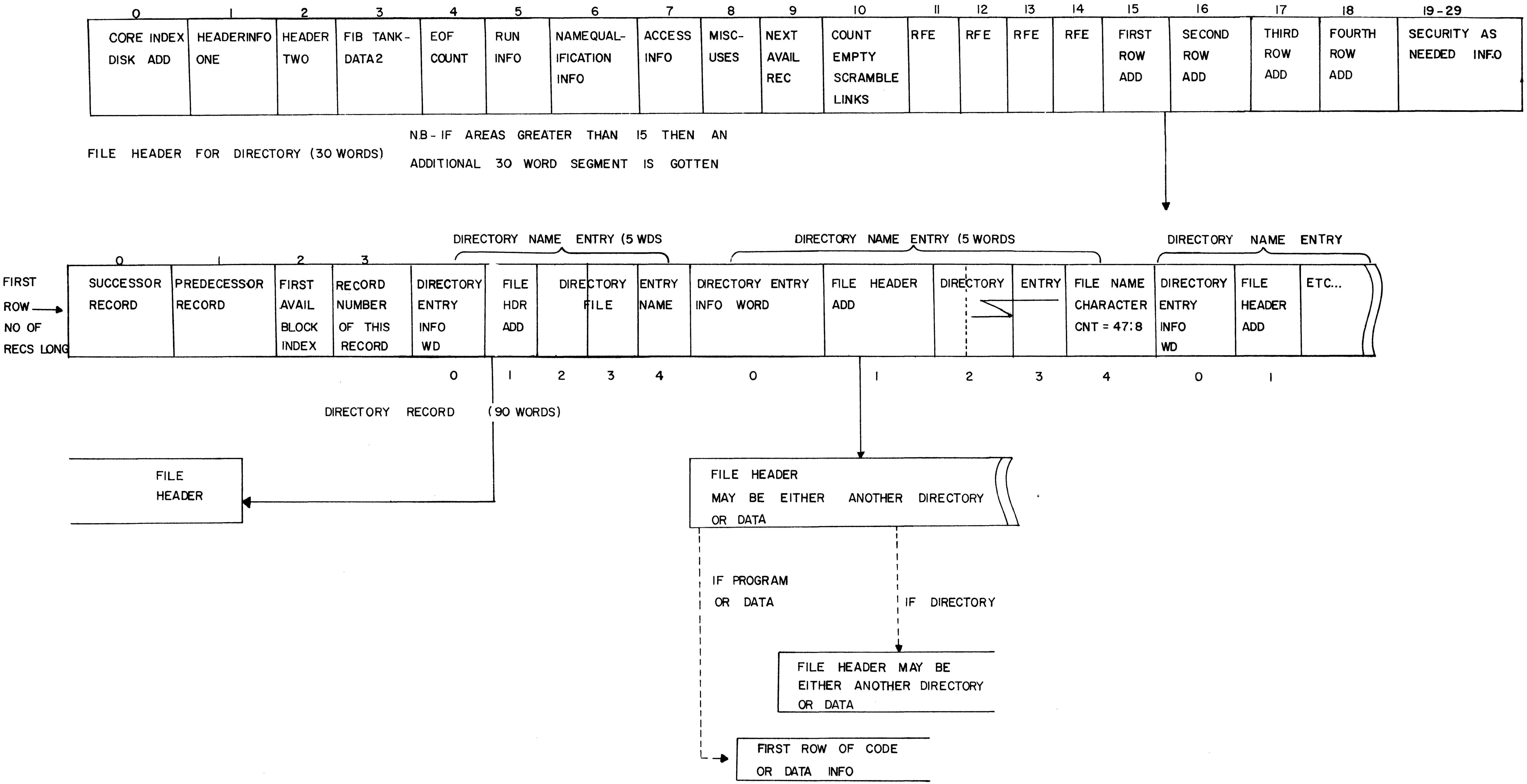
CORRESPONDINGLY, THE DISK DIRECTORY IS REALLY A COLLECTION OF FILES ORGANIZED AS A TREE STRUCTURE. SUCH A FILE WILL BE REFERRED TO AS A DIRECTORY. THE DIRECTORY AT THE ORIGIN OF THE TREE STRUCTURE WILL BE REFERRED TO AS THE MASTER DIRECTORY. THE DIRECTORY BODY IS COMPOSED OF RECORDS WHICH ARE 90 WORDS (3 SEGMENTS) LONG. EACH RECORD CONTAINS A LIST OF ENTRIES AND A LINK TO ANOTHER RECORD IN THE SAME DIRECTORY. EACH ENTRY IN A GIVEN RECORD HAS SEVERAL PARTS WHICH ARE DISCUSSED BELOW.

> THE IDENTIFIER PART CONTAINS AN IDENTIFIER WHICH IS SEVENTEEN CHARACTERS IN LENGTH.
>
> THE ADDRESS PART CONTAINS THE ADDRESS OF THE HEADER OF A FILE WHICH MAY OR MAY NOT BE ANOTHER DIRECTORY.
>
> THE FILE TYPE PART INDICATES THE NATURE OF THE FILE TO WHICH THE ADDRESS PART POINTS.
>
> THE VOLUME NUMBER IS USED TO SPECIFY WHICH TAPE IS NEEDED IF THE FILE HAS BEEN DUMPED TO TAPE.

IN ORDER TO REFERENCE A DISK FILE BY MEANS OF THE ACTUAL FILE NAME, EACH IDENTIFIER IS USED TO FIND A DIRECTORY WHICH IS REACHED THROUGH THE PRECEDING IDENTIFIERS. A DIRECTORY HEADER CONTAINS THE INFORMATION AS TO HOW THE DIRECTORY IS TO BE INDEXED. THE MAIN DIRECTORY WILL BE INDEXED BY SCRAMBLING THE IDENTIFIER. OTHER DIRECTORIES WILL BE INDEXED OR SEARCHED, DEPENDING ON THE SIZE OF THE DIRECTORY AT THAT LEVEL.

IF A DIRECTORY IS SCRAMBLED, THE DIRECTORY RECORD WILL BE INDEXED AND THEN SEARCHED FOR A MATCHING IDENTIFIER.

IF THE END OF THE RECORD IS FOUND, THE LINK INDICATES THE NEXT RECORD TO BE SEARCHED. IF A MATCH IS FOUND, THE ADDRESS PART OF THE ENTRY WILL SPECIFY THE LOCATION OF THE FILE WHICH IS TO BE SEARCHED NEXT. IF THIS FILE IS A DIRECTORY, THEN ANOTHER LABEL MAY EXPECTED. IF, HOWEVER, THE FILE IS NOT A DIRECTORY FILE, THE LOCATION IS THE DISK ADDRESS OF THE HEADER OF THE FILE BEING REFERENCED.



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19-29
CORE INDEX DISK ADD	HEADERINFO ONE	HEADER TWO	FIB TANK-DATA2	EOF COUNT	RUN INFO	NAMEQUALIFICATION INFO	ACCESS INFO	MISC-USES	NEXT AVAIL REC	COUNT EMPTY SCRAMBLE LINKS	RFE	RFE	RFE	RFE	FIRST ROW ADD	SECOND ROW ADD	THIRD ROW ADD	FOURTH ROW ADD	SECURITY AS NEEDED INF.O

FILE HEADER FOR DIRECTORY (30 WORDS)

N.B - IF AREAS GREATER THAN 15 THEN AN ADDITIONAL 30 WORD SEGMENT IS GOTTEN

FIRST ROW → NO OF RECS LONG

0	1	2	3	DIRECTORY NAME ENTRY (5 WDS					DIRECTORY NAME ENTRY (5 WORDS					DIRECTORY NAME ENTRY		
				0	1	2	3	4	0	1	2	3	4	0	1	
SUCCESSOR RECORD	PREDECESSOR RECORD	FIRST AVAIL BLOCK INDEX	RECORD NUMBER OF THIS RECORD	DIRECTORY ENTRY INFO WD	FILE HDR ADD	DIRE	CTORY FILE	ENTRY NAME	DIRECTORY ENTRY INFO WORD	FILE HEADER ADD	DIRECTORY	ENTRY	FILE NAME CHARACTER CNT = 47:8	DIRECTORY ENTRY INFO WD	FILE HEADER ADD	ETC...

DIRECTORY RECORD (90 WORDS)

FILE HEADER

FILE HEADER MAY BE EITHER ANOTHER DIRECTORY OR DATA

IF PROGRAM OR DATA

IF DIRECTORY

FILE HEADER MAY BE EITHER ANOTHER DIRECTORY OR DATA

FIRST ROW OF CODE OR DATA INFO

4.3. PERIPHERAL UNIT AND SYMBOLIC FILE ASSIGNMENT

THE MCP AUTOMATICALLY ASSIGNS PERIPHERAL I/O UNITS TO SYMBOLIC FILES WHENEVER POSSIBLE, IN ORDER TO MINIMIZE THE AMOUNT OF OPERATOR ATTENTION WHICH IS REQUIRED BY EACH PROCESS.

INPUT FILES REQUESTED BY A PROCESS CAUSE THE MCP TO SEARCH ITS TABLES FOR THE APPROPRIATE PERIPHERAL UNIT WHICH CONTAINS THE FILE REQUESTED. IF THE FILE NAMES SPECIFIED BY THE PROCESS IS FOUND ON A PARTICULAR UNIT, THAT UNIT IS MARKED "IN USE" AND ASSIGNED TO THE PROCESS. IN THE CASE OF A DISK FILE. THE FILE HEADER IS MARKED "IN USE" INCREASING THE "USER" COUNT BY ONE, AND THE INDEX OF THE HEADER IS PASSED TO THE PROCESS.

OUTPUT FILES REQUESTED BY A PROCESS ARE AUTOMATICALLY ASSIGNED BY THE MCP IF A SUITABLE UNIT EXISTS FOR THE FILE. IN THE CASE OF DISK FILES, IF THE FILE IS NOT PRE-EXISTENT ON THE DISK, AND THE REQUIRED DISK SPACE IS ALLOCATED FOR THE FILE, A ROW AT A TIME

4.3.1. MCP PROCEDURE "STATUS"

"STATUS" IS CONCERNED WITH THE CURRENT STATUS OF PERIPHERALS. "STATUS" USES THE "SCAN-IN" OPERATOR TO FIND OUT WHETHER A UNIT IS READY OR NOT, AND TO COMPARE THE CURRENT STATE WITH THE FORMER STATE OF THE UNIT:

1. IF A UNIT GOES NOT READY, THE CHANGE IS NOTED IN THE UNIT TABLE. IF THE UNIT IS ASSIGNED, THERE IS A "NOT READY" MESSAGE.

2. IF A UNIT GOES READY, THE UNIT TABLE ENTRY IS MARKED.

   A. IF A UNIT GOES READY AND IS NOT ASSIGNED, A CHECK IS MADE TO SEE WHETHER IT SHOULD BE SAVED, LOCKED, AND/OR PURGED. DEPENDING ON UNIT TYPE:

(1) IF A CARD READER, IT IS MARKED AS SCRATCH, AND CONTROLCARD IS CALLED. TO PROCESS THE RECORD AS DESCRIBED BY THE AREA DESCRIPTOR OF THE IOCB AND READS ADDITIONAL RECORDS UNTIL A "BCL", "DATA", OR "END" IS ENCOUNTERED.

(2) IF A MAGNETIC TAPE, "READALABEL" IS CALLED, "READALABEL" READS VARIOUS MAGNETIC TAPES AND STUFFS THE INFORMATION THEY CONTAIN INTO THE LABEL TABLE. ALL ENTRY ACTION IS ACCOMPLISHED THROUGH THE "IO ERROR" PROCEDURE. UNEXPECTED IO ERROR IS ONLY CALLED FOR MEMORY ERRORS. A PARITY CONDITION CAUSES THE UNIT TO BE MARKED AS NOT-READY AND SAVED.

B. IF THE UNIT IS ASSIGNED, "STARTIO" IS CALLED. "STARTIO" FIRST CHECKS FOR CHANNEL IF THE UNIT IS IN A USABLE STATE. IF THE CHANNEL IS NOT AVAILABLE, IT INSERTS THE ENTRY INTO THE WAITCHANNELQUEUE. OTHERWISE IT CALLS "INITIATEIO". "INITIATEIO" INITIATES IO AND INITIALIZES IO FOR THE USER. IT ALSO CHECKS FOR THE UNIT TYPE AND UPDATES THE TRANSACTION COUNTER APPROPRIATELY.

FOR SERIAL TAPE THE "NSECOND" PROCEDURE MAINTAINS THE STATUS FOR THE UNIT. THUS, FOR EACH FILE MARKED AS LABELED AND INPUT, "STATUS" OBTAINS THE FILE LABEL(S) FROM THE FILE LABEL RECORDS AND SAVES IT IN A LABEL TABLE. THIS LABEL TABLE IS USED AT FILE OPEN TIME TO ASSOCIATE INPUT FILE NAMES WITH THE ACTUAL HARDWARE DEVICE UPON WHICH THE FILE IS MOUNTED.

4.4. FILE AND CONTROL BLOCKS

THE RELATIONSHIP BETWEEN A FILE NAME AND A FILE LABEL CAN BE ESTABLISHED BY SOURCE LANGUAGE STATEMENTS AT COMPILE TIME OR LABEL EQUATION CARDS AT RUN TIME. IN ADDITION, CERTAIN MCP MESSAGES CAN ASSOCIATE A FILE WITH A PROCESS USING A LABEL EQUATION BLOCK (LEB) AND A FILE INFORMATION BLOCK (FIB) GENERATED BY THE COMPILERS. THE LOGICAL ASSOCIATION OF THE FILE NAME AND FILE LABEL IS MADE UTILIZING THE PROCESS PARAMETER BLOCK (PPB).

(SEE FIGURE F4-4).

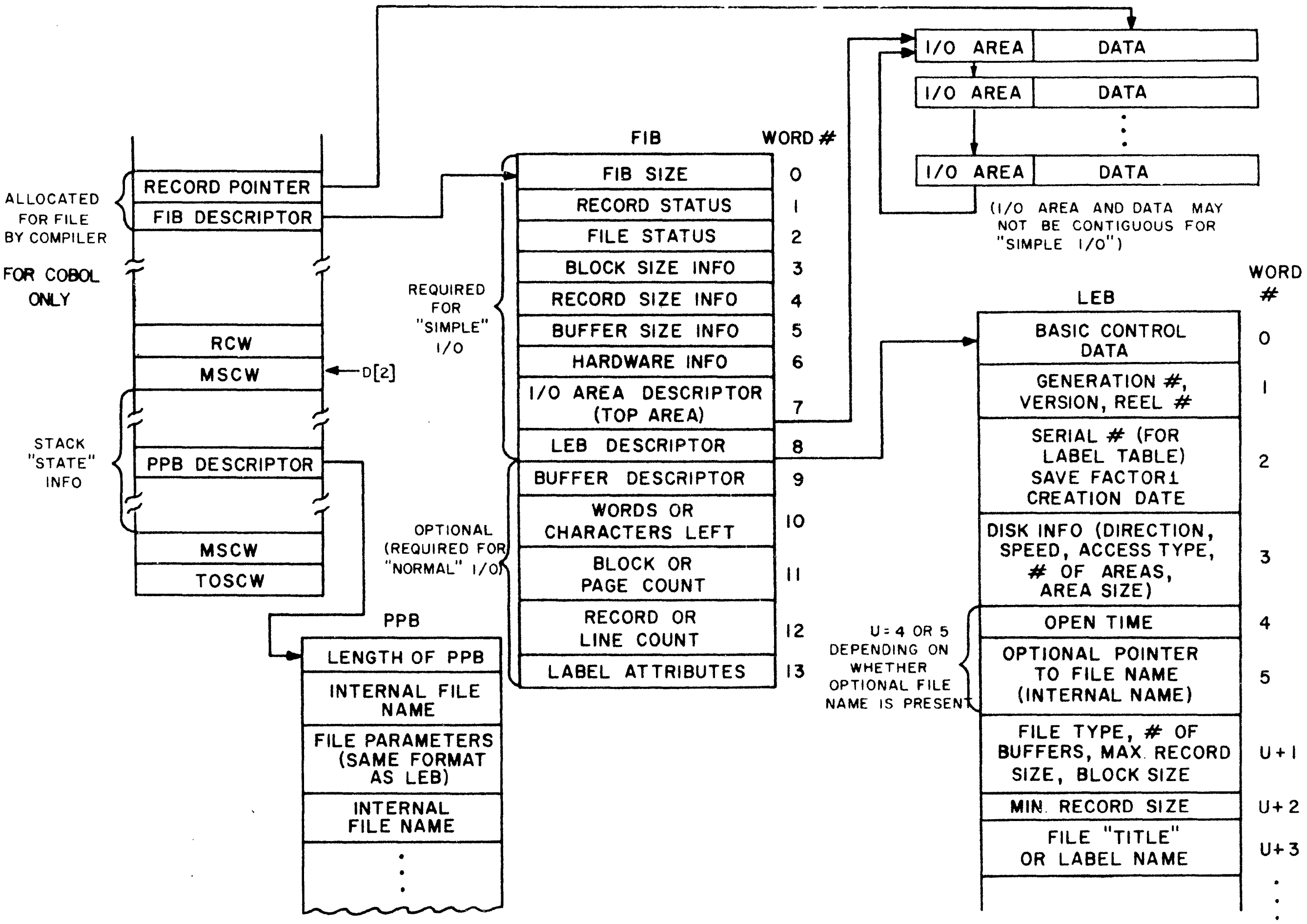

ALLOCATED FOR FILE BY COMPILER
RECORD POINTER
FIB DESCRIPTOR
FOR COBOL ONLY
RCW
MSCW
D[2]
STACK "STATE" INFO
PPB DESCRIPTOR
MSCW
TOSCW
PPB
LENGTH OF PPB
INTERNAL FILE NAME
FILE PARAMETERS (SAME FORMAT AS LEB)
INTERNAL FILE NAME
REQUIRED FOR "SIMPLE" I/O
OPTIONAL (REQUIRED FOR "NORMAL" I/O)
FIB
WORD #
FIB SIZE 0
RECORD STATUS 1
FILE STATUS 2
BLOCK SIZE INFO 3
RECORD SIZE INFO 4
BUFFER SIZE INFO 5
HARDWARE INFO 6
I/O AREA DESCRIPTOR (TOP AREA) 7
LEB DESCRIPTOR 8
BUFFER DESCRIPTOR 9
WORDS OR CHARACTERS LEFT 10
BLOCK OR PAGE COUNT 11
RECORD OR LINE COUNT 12
LABEL ATTRIBUTES 13
I/O AREA
DATA
I/O AREA
DATA
I/O AREA
DATA
(I/O AREA AND DATA MAY NOT BE CONTIGUOUS FOR "SIMPLE I/O")
LEB
WORD #
BASIC CONTROL DATA 0
GENERATION #, VERSION, REEL # 1
SERIAL # (FOR LABEL TABLE) SAVE FACTOR1 CREATION DATE 2
DISK INFO (DIRECTION, SPEED, ACCESS TYPE, # OF AREAS, AREA SIZE) 3
OPEN TIME 4
OPTIONAL POINTER TO FILE NAME (INTERNAL NAME) 5
U=4 OR 5 DEPENDING ON WHETHER OPTIONAL FILE NAME IS PRESENT
FILE TYPE, # OF BUFFERS, MAX. RECORD SIZE, BLOCK SIZE U+1
MIN. RECORD SIZE U+2
FILE "TITLE" OR LABEL NAME U+3

FIGURE F4-4. PPB, LEB, FIB, FORMAT

4.4.1. PROCESS PARAMETER BLOCK (PPB)

THE PPB IS A ONE DIMENSIONAL ARRAY CREATED AND MAINTAINED BY THE MCP CONTROL CARD ROUTINE FOR ALL FILES WHICH HAVE BEEN LABEL-EQUATED IN A PROCESS. THE PPB CONTAINS ALL OF THE LABEL EQUATION AND FILE ATTRIBUTE INFORMATION FOR THE FILES BELONGING TO A PROCESS.

A LABEL EQUATION CARD IN THE FORM:

```
        <I> FILE <SYMBOLIC FILE NAME> = <FILE LABEL> <FILE ATTRIBUTES>
```

IS NORMALLY USED TO ASSOCIATE A FILE LABEL (OR ACTUAL FILE NAME) WITH A SYMBOLIC FILE NAME. THIS CARD FOLLOWS THE COMPILE CARD OR THE EXECUTE CARD. WHEN THE MCP CONTROL CARD ROUTINE SEES LABEL EQUATION CARDS. IT SAVES THE INFORMATION IN THE PPB. THIS INFORMATION IS USED LATER TO MODIFY FILE LEBS AND FIBS WHEN THE FILES ARE FIRST OPENED.

4.4.2. LABEL EQUATION BLOCK (LEB)

A "LABEL EQUATION BLOCK" IS CREATED BY THE COMPILER AND MAINTAINED BY THE I/O INTRINSIC FUNCTIONS FOR EACH FILE IN A PROCESS.

THE LABEL EQUATION BLOCK CONTAINS THE CURRENT LABEL EQUATION AND FILE ATTRIBUTE INFORMATION FOR EACH FILE IN A PROCESS. BOTH THE LEB AND FIB ARE REFERRED TO BY A DESCRIPTOR IN THE WORKING STACK, WHICH ALLOWS THE DYNAMIC SPECIFICATION OF FILE ATTRIBUTES TO BE IMPLEMENTED IN AN EFFICIENT MANNER.

4.4.3. FILE INFORMATION BLOCK (FIB)

A FILE INFORMATION BLOCK IS ALSO CREATED BY THE COMPILERS FOR EACH PROGRAM AND MAINTAINED BY THE I/O INTRINSIC FUNCTIONS FOR EACH FILE USED BY THE PROGRAM.

THE FIB CONTAINS THE CURRENT STATUS OF A FILE, THE PROGRAMMERS DESCRIPTION OF DATA IN THE FILE, FILE ATTRIBUTES, DISK ATTRIBUTES AND PRINTER ATTRIBUTES.

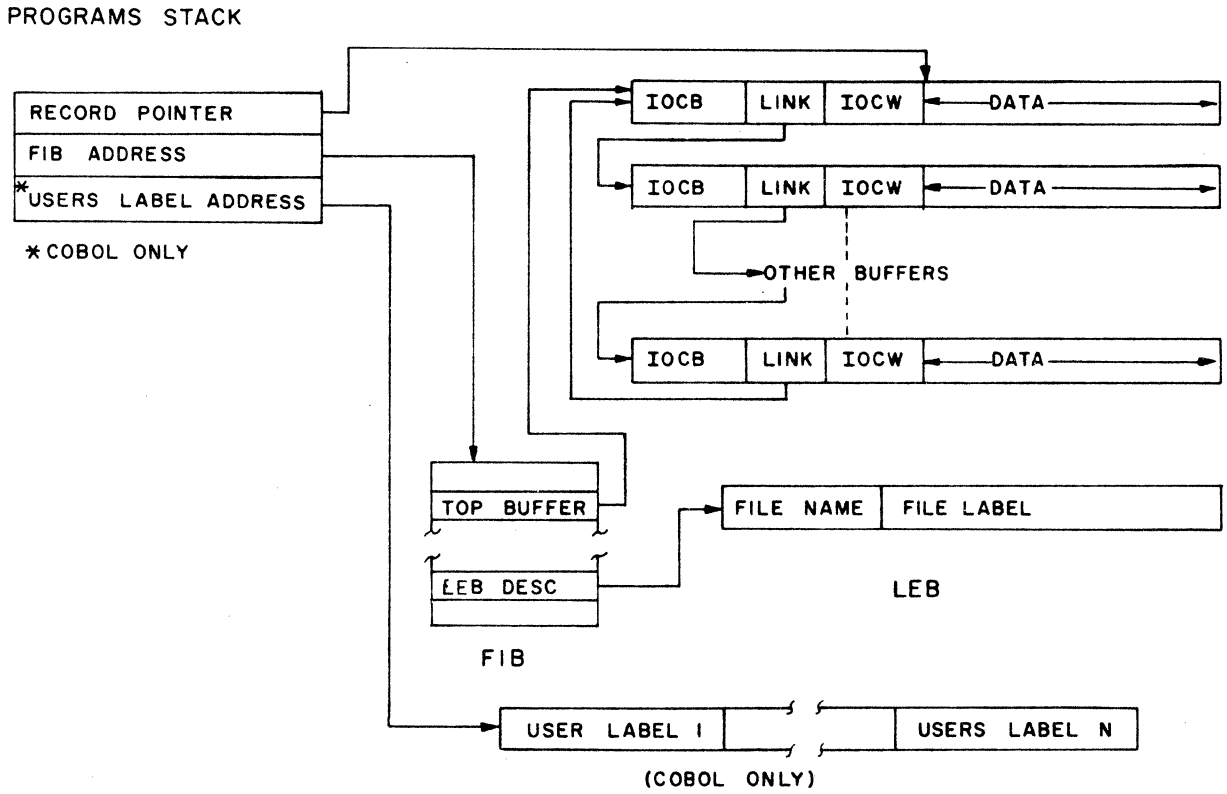
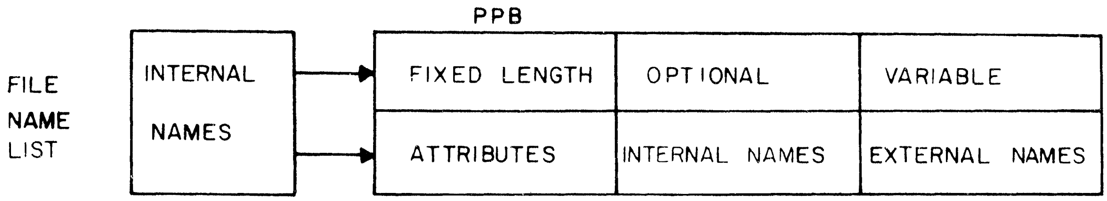
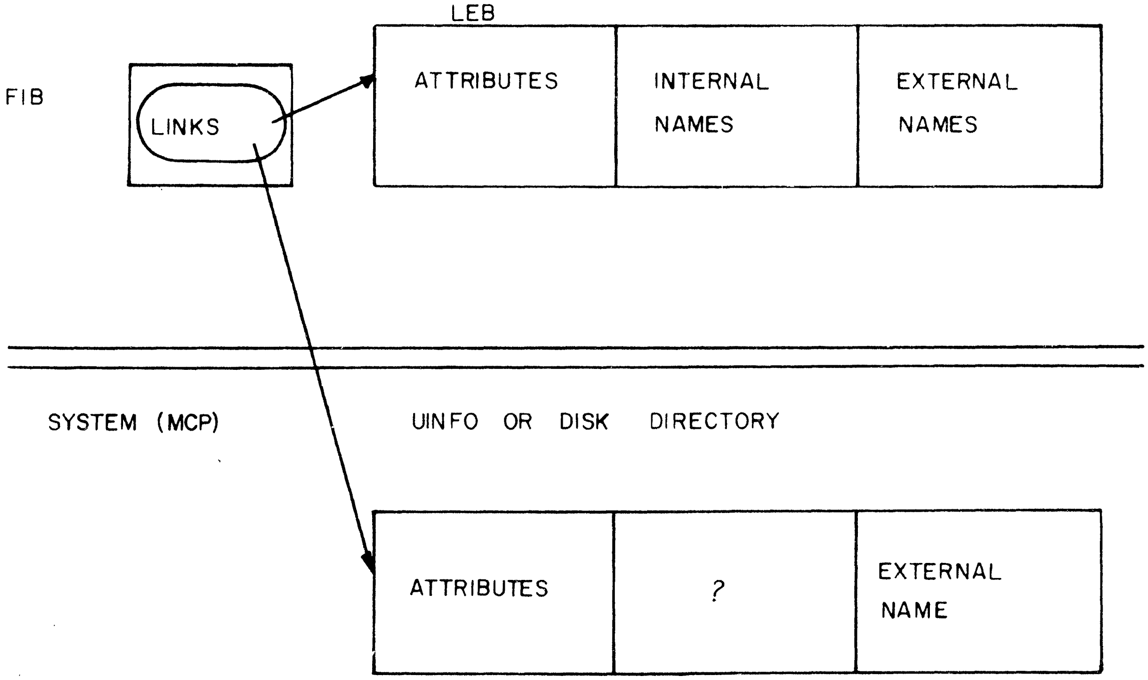


FIGURE F4-5. FILE/STACK RELATION

JOB ASSOCIATION (PRODUCED BY COMPILER EXECUTE CARDS)



FILE ASSOCIATION (FOR EACH INCARNATION)



OPEN: A) FILLS $\text{EXTERNAL}_{\text{LEB}}$ FROM $\text{INTERNAL}_{\text{LEB}}$ VIA FILE NAME LIST AND PPB.

B) SCANS UINFO OR DISK DIRECTORY FOR MATCH WITH LEB

FIGURE F4-6. JOB AND FILE ASSOCIATION

4.5. FILE OPEN

4.5.1. STEPS IN OPENING A FILE

1. THE FIRST FUNCTION PERFORMED BY THE OPERATING SYSTEM WHEN REQUESTED TO OPEN A FILE IS TO MAP THE FILE NAMES AND FILE ATTRIBUTES FROM LABEL EQUATION CARDS INTO THE FIB AND PPB. THIS ALLOWS ASSOCIATION OF A FILE NAME WITH A FILE LABEL. THE <FILE ATTRIBUTES> PART OF THE LABEL EQUATION CARD, ALLOWS ALTERING THE SOURCE LANGUAGE DESCRIPTION OF A FILE ATTRIBUTES IN A WAY THAT SUCH THINGS AS FILE BLOCKING, OUTPUT FILE DEVICE, ETC. CAN BE MODIFIED AT EXECUTE TIME WITHOUT RE-COMPILING THE SOURCE LANGUAGE. FOR EXAMPLE, A PROGRAM WRITTEN TO PRODUCE ITS OUTPUT ON CARD-PUNCH CAN BE LABEL-EQUATED TO PRODUCE ITS OUTPUT ON A LINE-PRINTER OR A BLOCKED MAGNETIC TAPE WITHOUT RE-COMPILING.

2. THE SECOND STEP IN OPENING A FILE IS TO ASSIGN A DEVICE TO THE FILE. THE ACTION TAKEN DEPENDS UPON THE TYPE OF FILE:

   A. IF THE FILE IS "OUTPUT" AND IS NOT A DISK FILE, LOCATE A DEVICE OF THE CORRECT TYPE MARKED "AVAILABLE" AND "OUTPUT", DETERMINE ITS UNIT DESIGNATION, WRITE LABEL RECORDS AND USERS LABEL RECORDS AND, FOR COBOL PROGRAMS, EXECUTE USER "USE" ROUTINES, IF ANY.

   B. IF THE FILE IS "OUTPUT" AND A NEW DISK FILE, GENERATE A FILE HEADER IN MEMORY AND ASSIGN ACTUAL DISK SPACE FOR THE FIRST DISK AREA.

   C. IF THE FILE IS A "PRE-EXISTENT FILE ON DISK" (INPUT OR OUTPUT), LOCATE FILE IN FILE DIRECTORY, READ ITS DISK HEADER INTO MAIN MEMORY AND CHECK THE ATTRIBUTES SPECIFIED IN FIB AGAINST ATTRIBUTES SPECIFIED IN THE FILE HEADER. IF THE ATTRIBUTES ARE

INCOMPATIBLE, TERMINATE THE PROCESS.

D. IF THE FILE IS "INPUT" AND NOT A DISK FILE, LOCATE THE FILE IN THE LABEL TABLE WHICH ALSO INDICATES THE UNIT DESIGNATION AND READ THE FILE LABELS AND USERS LABELS (EXECUTING APPROPRIATE USERS "USE" ROUTINES). COMPARE FILE ATTRIBUTES IN FIB WITH FILE ATTRIBUTES IN THE FILE LABEL. IF THE ATTRIBUTES ARE NOT COMPATIBLE, TERMINATE THE PROGRAM.

3. THE THIRD STEP IS TO ALLOCATE MEMORY SPACE FOR THE BUFFERS.

4. THE FOURTH STEP IS TO CONSTRUCT I/O CONTROL WORDS AND "IOAREA" CONTROL AREAS (SEE IOAREA - I/O BUFFER LAYOUT BELOW).

5. LASTLY, FOR ALL INPUT FILES AND BLOCKED OUTPUT FILES ON PRE-EXISTENT DISK FILES, THE BUFFERS ARE PRELOADED WITH DATA.

4.5.2. THE RECORD POINTER

ASSOCIATED WITH EACH FILE IS A RECORD POINTER. ALL DATA IS ACCESSED BY A PROGRAM THROUGH THE RECORD POINTER. THIS POINTER CONTAINS A BASE AND A MAXIMUM RECORD SIZE. THE VARIOUS LANGUAGES CAN SPECIFY RECORDS OF VARIABLE SIZE AND A MAXIMUM RECORD SIZE. ALSO SOME LANGUAGES DEPEND UPON THE PROGRAM TO ESTABLISH THE RECORD SIZE. CERTAIN HARDWARE CHECKS WILL CAUSE PROGRAM TERMINATION IF THE PROGRAM ESTABLISHES A RECORD SIZE EXCEEDING ITS SPECIFIED MAXIMUM RECORD SIZE. THIS ESTABLISHES ONE LEVEL OF SYSTEM INTEGRITY, I.E., A PROGRAM CANNOT ALTER OR DESTROY DATA OUTSIDE OF THE PROGRAM DATA AREA LIMIT.

EACH I/O STATEMENT MAKES A RECORD AVAILABLE TO A PROGRAM ALTERING THE BASE FIELD OF THE RECORD POINTER. IN THE CASE OF BLOCKED RECORDS AND ALSO ANOTHER RECORD IN THE BLOCK EXISTS, THEN THE NEXT RECORD CAN BE OBTAINED BY INCREMENTING THE POINTER BASE BY THE SIZE OF THE PREVIOUS RECORD. IN THE CASE WHERE ALL LOGICAL RECORDS IN A BLOCK HAVE BEEN PROCESSED THE BASE CAN BE SET TO EITHER THE NEXT BUFFER, IF MULTIPLE BUFFERS ARE SPECIFIED, OR THE FRONT OF THE BUFFER, IF ONLY ONE BUFFER IS

SPECIFIED.

ANYTIME THE RECORD POINTER IS SET (RATHER THAN INDEXED), THE ADDRESS OF THE I/O CONTROL AREA "IOAREA" IS PASSED TO THE MCP WHICH ACTIVATES AN ACTUAL I/O OPERATION ON THAT BUFFER. AT THE SAME TIME, THE EVENT ASSOCIATED WITH THE BUFFER TO WHICH THE RECORD POINTER HAS BEEN SET, IS CHECKED.

THE BUFFER EVENT SERVES TO INTERLOCK THE BUFFER WITH THE PROGRAM SUCH THAT THE PROGRAM CANNOT REFERENCE A BUFFER WHICH HAS AN I/O IN PROGRESS ON IT. WHEN A BUFFER IS PASSED TO THE MCP, ITS EVENT IS SET TO THE STATE "NOT HAPPENED". UPON COMPLETION OF THE I/O FOR THAT BUFFER, THE MCP ROUTINE, IOFINISH, "CAUSES" THE EVENT TO HAPPEN. PRIOR TO RETURNING TO THE PROGRAM AFTER SETTING THE RECORD POINTER TO A BUFFER, ITS EVENT IS CHECKED FOR THE STATE "HAPPENED". IF THE EVENT HAS NOT "HAPPENED", THEN A WAIT (EVENT) IS EXECUTED. THIS WILL CAUSE THE PROGRAM TO BE MOVED FROM THE READY QUEUE TO THE WAIT QUEUE, I.E., THE PROGRAM IS SUSPENDED AND ANOTHER PROGRAM IN THE READY QUEUE IS STARTED. LATER, WHEN IOFINISH "CAUSES" THE EVENT TO HAPPEN, ALL OF THE PROCESSES WAITING ON THAT EVENT ARE MOVED FROM THE WAIT QUEUE TO THE READY QUEUE. THE PROCESSES WILL BE RE-ACTIVATED ACCORDING TO THEIR PRIORITIES IN THE READY QUEUE.

4.6. FILE CLOSE

4.6.1. TYPES OF FILE CLOSE

THERE ARE 10 TYPES OF FILE CLOSE:

1. CLOSE BUT DO NOT RELEASE UNIT

2. RELEASE UNIT

3. PURGE UNIT

4. DO NOT REWIND UNIT

5. ENTER TEMPORARY DISK FILE INTO DISK DIRECTORY

6. COMPRESS (CRUNCH) DISK FILE

7. CLOSE HERE - USED WHEN CHANGING FROM INPUT TO OUTPUT. THE TAPE HEAD IS POSITIONED IN FRONT OF THE CURRENT BLOCKS.

8. CLOSE * (ASTERISK) - USED FOR POSITIONING IN MULTI - FILE AND FILE - SET CASES.

9. SUSPEND (USED WHEN PROGRAM IS SUSPENDED)

10. BLOCK EXIT

4.6.2. FILE CLOSE ERRORS

FILE CLOSE ERRORS INCLUDE:

1. FILE NOT OPEN

2. IRRECOVERABLE I/O ERROR DURING CLOSE PROCESSING

3. RECORD COUNT ERROR

4. BLOCK COUNT ERROR

4.7. DISK FILE SECURITY

ALL DISK FILES ARE DIVIDED INTO THREE MAIN SECURITY CLASSES. EACH OF THESE CLASSES WILL BE DISCUSSED BRIEFLY IN THE FOLLOWING PARAGRAPHS.

4.7.1. CLASS A FILES

ACCESS TO CLASS A FILES IS CONTROLLED BY THE MCP AT FILE-OPEN TIME. THE FACTORS GOVERNING ACCESS TO A FILE ARE THE USER CLASS AND SECURITY CLASS OF THE USER DESIRING ACCESS, THE TYPE OF ACCESS DESIRED BY THE USER, AND THE FILE TYPE OF THE FILE WHICH THE USER DESIRES TO ACCESS. EACH SYSTEM USER WILL HAVE A USER ID AND HIS USER CLASS AND SECURITY CLASS WILL BE ASSOCIATED TO THIS ID BY MEANS OF A SPECIAL FILE. FROM THE POINT OF VIEW OF FILE SECURITY THERE ARE FOUR TYPES OF FILE ACCESS WHICH A USER MAY REQUEST: READ ONLY, READ/WRITE, LIBRARY MAINTENANCE AND SECURITY MAINTENANCE. THE FILE TYPE INDICATES WHETHER ACCESS TO THE FILE IS RESTRICTED AND IF SO HOW. IF ACCESS IS RESTRICTED THEN THERE WILL BE A CLASS OF PRIVILEGED USERS WHICH MAY BE DETERMINED ON THE BASIS OF USER CLASS, SECURITY CLASS, OR BOTH OF THESE. A NON-PRIVILEGED USER IS NOT NECESSARILY DENIED ALL ACCESS TO THE FILE; HE MAY, FOR EXAMPLE, BE ALLOWED READONLY ACCESS TO A FILE WHICH ONLY PRIVILEGED USERS MAY ALTER. ON THE OTHER HAND, A NON-PRIVILEGED USER MIGHT BE DENIED ALL ACCESS TO A FILE WHICH PRIVILEGED USERS HAVE RANDOM ACCESS TO.

4.7.2. CLASS B FILES

ACCESS TO CLASS B FILES IS CONTROLLED BY A SPECIFIED PROCEDURE AT FILE-OPEN TIME. THIS PROCEDURE WILL HAVE BEEN SPECIFIED BY THE CREATOR OF THE FILE AND ITS IDENTIFICATION WILL BE CONTAINED IN THE FILE HEADER. WHEN A USER ATTEMPTS TO ACCESS THE FILE THE MCP WILL CALL THIS PROCEDURE PASSING HIS USER CLASS, SECURITY CLASS, AND USER ID AS WELL AS THE TYPE OF ACCESS DESIRED. THE PROCEDURE WILL THEN DETERMINE THE VALIDITY OF THE REQUEST. SUCH A PROCEDURE COULD BE USED, FOR EXAMPLE, TO ALLOW ACCESS TO A FILE ONLY AT SPECIFIC TIME OF THE DAY OR TO ALLOW SPECIFIC TYPES OF ACCESS AT SPECIFIC TIMES OF DAY BY SPECIFIC PEOPLE.

4.7.3. CLASS C FILES

ACCESS TO CLASS C FILES IS CONTROLLED BY A SPECIFIC PROCEDURE NOT AT FILE-OPEN TIME, BUT RATHER AT THE RECORD LEVEL. THIS PROCEDURE WILL HAVE BEEN SPECIFIED BY THE CREATOR OF THE FILE AND ITS IDENTIFICATION WILL BE CONTAINED IN THE FILE HEADER. WHEN A USER ATTEMPTS TO ACCESS A RECORD IN THE FILE THE MCP WILL CALL THIS PROCEDURE PASSING HIS USER CLASS, SECURITY CLASS, THE RECORD, AND USER ID AS WELL AS THE TYPE OF ACCESS DESIRED. THE PROCEDURE WILL THEN DETERMINE THE VALIDITY OF THE REQUEST. SUCH A PROCEDURE COULD BE USED, FOR EXAMPLE, TO RESTRICT ACCESS TO CERTAIN RECORDS IN THE FILE OR TO RESTRICT THE MANNER IN WHICH THESE RECORDS ARE ACCESSED OR MASK CERTAIN PORTIONS OF THE RECORD.

4.7.4. RELATIONSHIP BETWEEN THE SECURITY CLASSES

IT SHOULD BE CLEAR THAT CLASS C SECURITY IS STRONGER THAT CLASS B IN THE SENSE THAT THE EFFECT OF CLASS B SECURITY CAN BE ACHIEVED BY AN APPROPRIATE CLASS C SECURITY PROCEDURE. IT IS ALSO CLEAR THAT A CLASS B FILE CAN BE MADE TO LOOK LIKE A CLASS A FILE. HOWEVER, IN EACH OF THESE INSTANCES, ACCESS TO THE FILE WOULD BE SLOWED BY THE IMPOSITION OF UNNECESSARY SECURITY. FINALLY, IT SHOULD BE NOTED THAT A FILE MAY BE AT ONCE TYPE A, TYPE B, AND TYPE C OR ANY COMBINATION OF THESE THREE, SIMPLY BY IMPOSING VARIOUS CLASSES OF SECURITY AT THE DIFFERENT LEVELS OF THE DISK DIRECTORY STRUCTURE.

SECTION 5

INPUT/OUTPUT

5. INPUT/OUTPUT

5.1. MCP I/O PROCEDURES

THE MCP COORDINATES INPUT/OUTPUT OPERATIONS FOR ALL JOBS AND PERIPHERAL DEVICES ON THE B6700 SYSTEM. WHAT FOLLOWS IS A DETAILED DISCUSSION OF MCP PROCEDURES CONCERNED WITH INPUT/OUTPUT (PROCEDURE NAMES ARE SET OFF BY QUOTATION MARKS). THE NAMES AND RELATIONSHIPS OF THESE PROCEDURES ARE SUBJECT TO CHANGE.

5.1.1. PERIPHERALINITIALIZE

"PERIPHERALINITIALIZE" SETS UP THE UNIT TABLE AND THE TABLE WHICH CONTAINS THE UNIT NUMBER FOR EACH UNIT TYPE. AFTER "PERIPHERALINITIALIZE" HAS SET UP THE TABLES, IT INITIATES A REWIND ON TAPES AND STARTS "STATUS".

5.1.2. STATUS

"STATUS" MATCHES THE CURRENT UNIT STATUS WITH THE OLD STATUS BY CHECKING THE 8 STATUS VECTORS, USING BOTH THE SCNI (SCAN IN) OPERATOR AND THE INTERROGATE PERIPHERAL STATUS WORD. THREE RECORDS OF USASI LABELS ARE READ EACH TIME THE UNIT GOES READY; AT WHICH TIME, "READALABEL" IS CALLED AS AN INDEPENDENT RUNNER.

5.1.3. READALABEL

"READALABEL" READS THE VARIOUS TAPE LABELS AND STUFFS THE INFORMATION THEY CONTAIN INTO THE LABEL TABLE. ALL RETRY ACTION IS ACCOMPLISHED THROUGH "IOERROR". "UNEXPECTEDIOERROR" IO ERROR IS ONLY CALLED FOR MEMORY ERROR. A PARITY CONDITION CAUSES THE UNIT TO BE MASKED AS NOT-READY AND SAVED. THEN EITHER "WAITIO" OR "DISKWAIT" BUILDS AND IOCB AND CALLS "IOREQUEST", WHICH LINKS THE IOCB INTO IOQUEUE.

5.1.4. WAITIO

"WAITIO" PASSES "IOREQUEST" A LOCAL EVENT ON WHICH IT WAITS. IF THERE ARE NO ERRORS OTHER THAN THOSE ACCOUNTED FOR BY THE IOERROR MASK THEN "WAITIO" RETURNS THE RESULT DESCRIPTOR, OTHERWISE IT CALLS "UNEXPECTEDIOERROR". THERE ARE THREE PARAMETERS PASSED TO "WAITIO": IOERRORMASK, USER, AND AREA. IOERRORMASK IS AN ERROR MASK PROVIDED BY THE CALLER TO DO ITS OWN ERROR-HANDLING. THE BUFFERLENGTH FIELD OF IOERRORMASK HAS A MASK WH!CH DETERMINES UNEXPECTED I/O ERROR.

5.1.5. DISKWAIT

"DISKWAIT" CALLS "DISKIO" WHICH PASSES PARAMETERS CORE, INDEX, SIZE, DISK, MASK AND AN EVENT TO BE CAUSED ON I/O COMPLETE AND WAITS ON THE EVENT. THE PARAMETER CORE IS THE ARRAY DESCRIPTOR FOR THE CORE AREA. THIS AREA MUST BE NON-OVERLAYABLE ("DISKWAIT" GUARANTEES THIS). THE PARAMETER INDEX IS THE STARTING INDEX FOR THE CORE AREA DESCRIPTOR, WHILE THE PARAMETER SIZE IS THE NUMBER OF WORDS TO TRANSFER. DISK IS THE SOFTWARE DISK ADDRESS. MASK IS THE MASK OF THE STANDARD I/O CONTROL WORD.

5.1.6. DISKIO

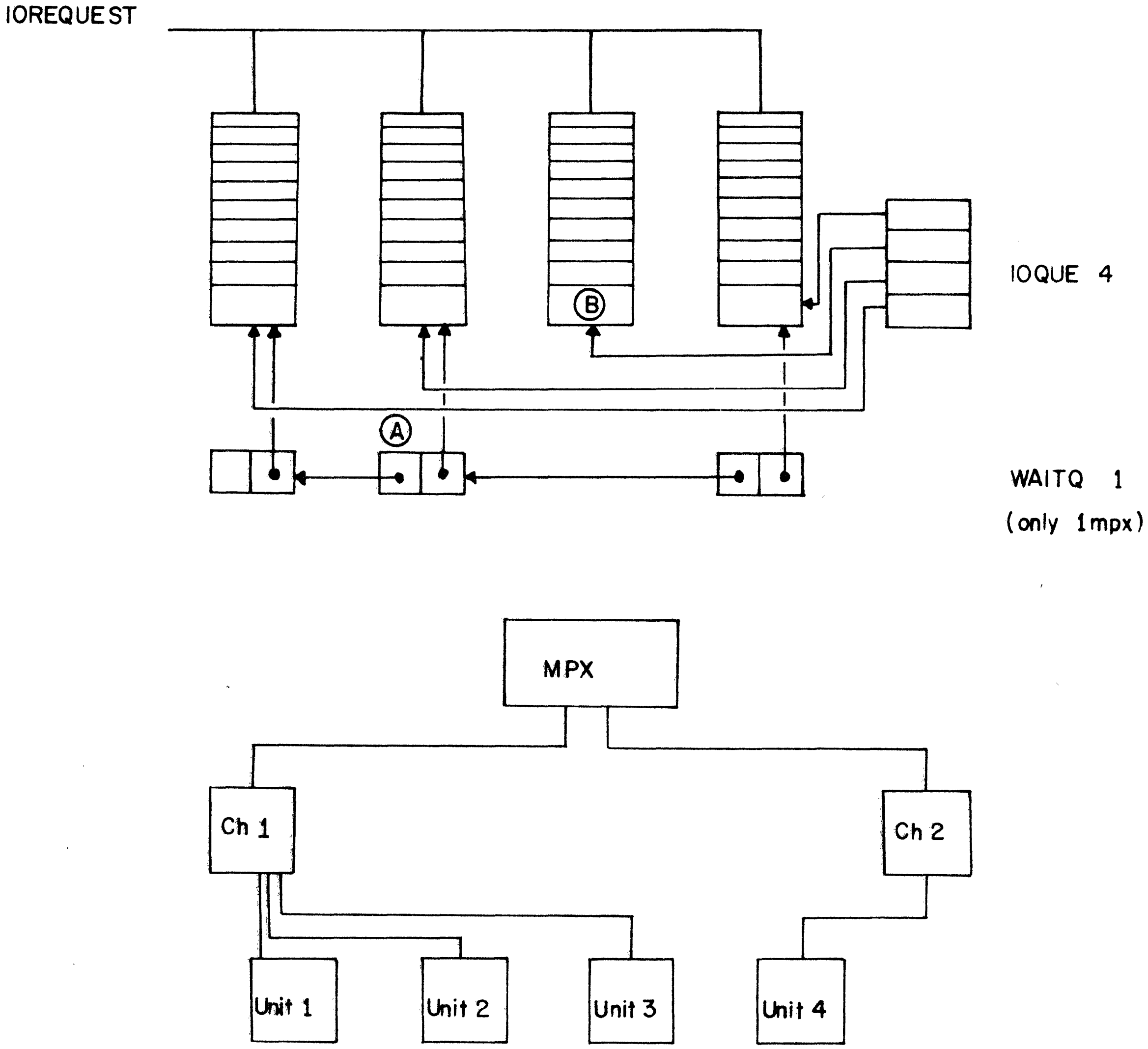
"DISKIO" CONVERTS THE SOFTWARE DISK ADDRESS INTO HARDWARE DISK ADDRESS AND MAKES UP AN IOCW. IT ALSO MAKES UP THE AREA DESCRIPTOR AND INSERTS IT INTO THE IOCB. IT THEN CALLS "IOREQUEST".

5.1.7. IOREQUEST

"IOREQUEST" QUEUES UP THE I/O IN THE I/O QUEUE. IF THERE IS MORE THAN ONE ENTRY IN THE QUEUE, "IOREQUEST" RETURNS TO THE REQUESTING PROCESS. OTHERWISE, "IOREQUEST" CALLS "STARTIO". "IOREQUEST" IS PASSED AN IOCB, A DESCRIPTOR POINTING TO THE ENTRY BLOCK FOR THE I/O QUEUE.

WAITCHANNELQUE (ONE PER MPX)

THIS MAY CONTAIN UP TO ONE ENTRY FOR EACH UNIT, PROVIDED THAT UNIT IS NOT "IN PROCESS OF DOING I/O". THE REQUEST REMAIN LINKED INTO THE IOQUE # UNITS.



AT I/O COMPLETE "A" WILL BE INITIATED.
"B" LINKED TO WAITCHANNELQUE.

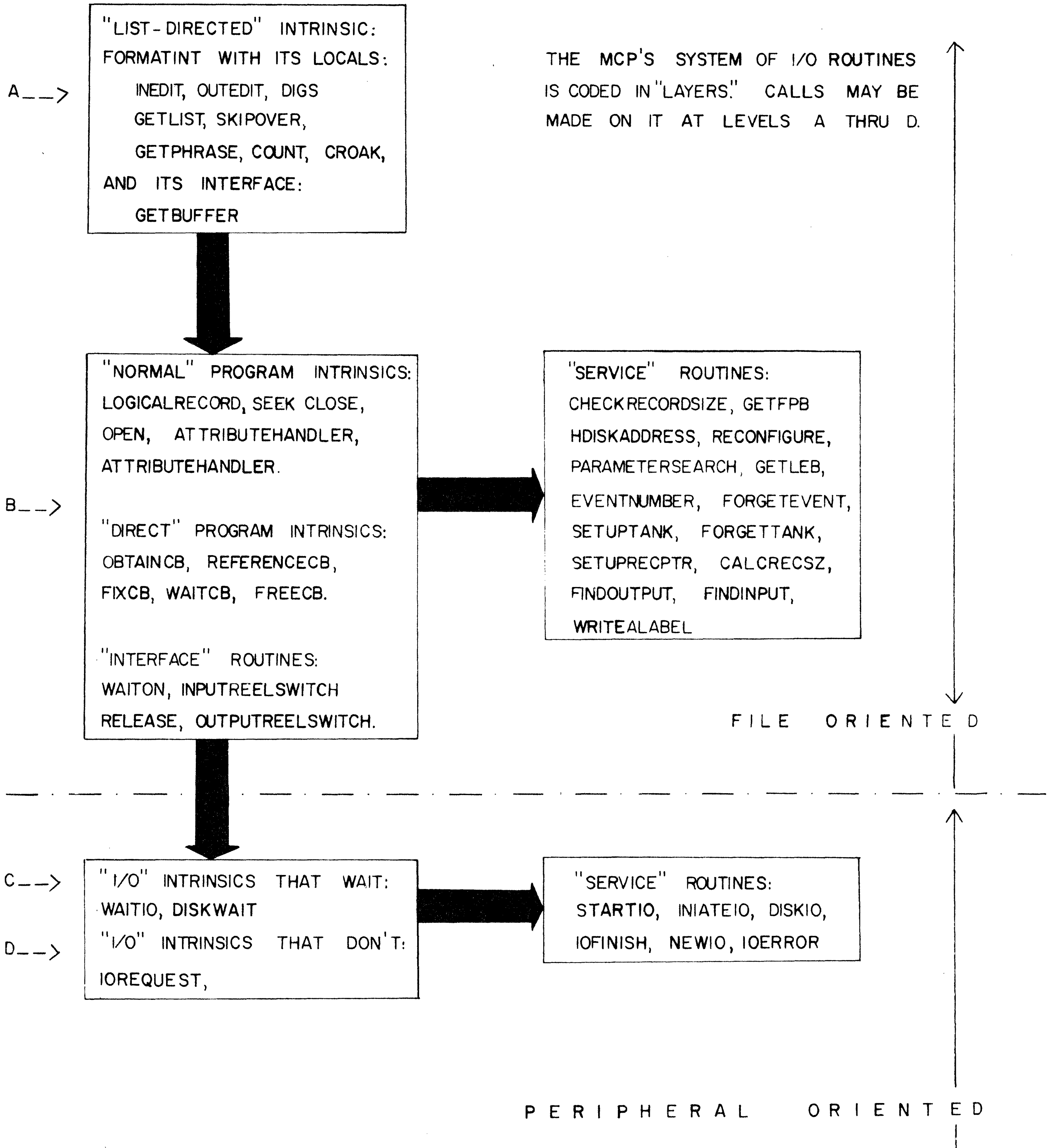
FIGURE F5-1. WAITCHANNELQUE EXPLANATION

AS PREVIOUSLY STATED, EITHER "WAITIO" OR "DISKWAIT" INITIATES I/O BY PASSING THE ADDRESS OF AN I/O AREA TO THE MCP PROCEDURE "IOREQUEST". THE PRIMARY PURPOSE OF "IOREQUEST" IS TO QUICKLY SET UP AN I/O REQUEST AND RETURN TO THE CALLING PROGRAM. TO SET UP AN I/O REQUEST, SEVERAL THINGS MUST BE CONSIDERED:

1. FIRST, SINCE "IOREQUEST" IS HANDLING I/O OPERATIONS ON ALL BUFFERS OF ALL PROGRAMS IN THE MIX, EACH I/O MUST BE ASSOCIATED WITH A PARTICULAR BUFFER OF A PARTICULAR PROGRAM.

2. A SECOND CONSIDERATION IS THAT "IOREQUEST" MUST SET UP AN I/O OPERATION AND RETURN TO THE CALLER, EVEN IF THE I/O REQUEST IS ON A DEVICE THAT CANNOT BE INITIATED. THE DEVICE MAY ALREADY BE IN USE BY A PRIOR REQUEST OR ALL MULTIPLEXOR CHANNELS MAY BE BUSY PERFORMING I/O OPERATIONS ON OTHER DEVICES. THIS ALSO IMPLIES THE SET-UP MUST INCLUDE THE CAPABILITY OF LATER SENDING THE REQUEST TO THE MULTIPLEXOR WHEN THE DEVICE DOES BECOME AVAILABLE.

3. FINALLY, THE SET-UP MUST ALSO INCLUDE THE ABILITY TO INTERLOCK THE I/O BUFFER AND LATER, WHEN THE I/O OPERATION IS COMPLETE, UNLOCK THE BUFFER. THIS INTERLOCKING MUST BE TRANSPARENT TO THE PROGRAMMER; IN ADDITION IT MUST ALLOW THE PROGRAM TO RUN UNTIL THE PROGRAM ATTEMPTS TO PROCESS DATA IN A BUFFER FOR WHICH AN I/O REQUEST HAS BEEN MADE, BUT IS NOT YET COMPLETED. THE MCF UTILIZES A QUEUE FOR EACH UNIT IN THE HANDLING OF AN I/C REQUEST, AS SHOWN IN FIGURE F5-2, THE MCP I/O QUEUE.



A--> "LIST-DIRECTED" INTRINSIC:
FORMATINT WITH ITS LOCALS:
INEDIT, OUTEDIT, DIGS
GETLIST, SKIPOVER,
GETPHRASE, COUNT, CROAK,
AND ITS INTERFACE:
GETBUFFER

THE MCP'S SYSTEM OF I/O ROUTINES IS CODED IN "LAYERS." CALLS MAY BE MADE ON IT AT LEVELS A THRU D.

B--> "NORMAL" PROGRAM INTRINSICS:
LOGICALRECORD, SEEK CLOSE,
OPEN, ATTRIBUTEHANDLER,
ATTRIBUTEHANDLER.

"DIRECT" PROGRAM INTRINSICS:
OBTAINCB, REFERENCECB,
FIXCB, WAITCB, FREECB.

"INTERFACE" ROUTINES:
WAITON, INPUTREELSWITCH
RELEASE, OUTPUTREELSWITCH.

"SERVICE" ROUTINES:
CHECKRECORDSIZE, GETFPB
HDISKADDRESS, RECONFIGURE,
PARAMETERSEARCH, GETLEB,
EVENTNUMBER, FORGETEVENT,
SETUPTANK, FORGETTANK,
SETUPRECPTR, CALCRECSZ,
FINDOUTPUT, FINDINPUT,
WRITEALABEL

FILE ORIENTED

C--> "I/O" INTRINSICS THAT WAIT:
WAITIO, DISKWAIT
D--> "I/O" INTRINSICS THAT DON'T:
IOREQUEST,

"SERVICE" ROUTINES:
STARTIO, INIATEIO, DISKIO,
IOFINISH, NEWIO, IOERROR

PERIPHERAL ORIENTED

FIGURE F5-2. I/O CONTROL SYSTEM FUNCTIONAL BLOCK DIAGRAM

EACH DEVICE IN THE SYSTEM (EACH READER, TAPE, DISK ELECTRONICS UNIT, ETC.) HAS A UNIQUE UNIT NUMBER AND A UNIQUE I/O QUEUE.

THE "IOREQUEST" FUNCTIONS ARE AS FOLLOWS:

1. THE I/O AREA IS LINKED INTO THE I/O QUEUE BY EXECUTING THE "INSERT" ALGORITHM OF THE I/O QUEUE. IF THERE IS MORE THAN ONE ENTRY IN THE QUEUE, "IOREQUEST" RETURNS TO THE REQUESTING PROCESS. OTHERWISE "IOREQUEST" CALLS STARTIO". "STARTIO" MAKES UP A UNITWORD, WHICH SPECIFIES THE UNIT AND MULTIPLEXOR, AND THE HARDWARE INSTRUCTION WHICH INTERROGATES FOR I/O PATH TO BE EXECUTED.

2. IF A PATH (I/O CHANNEL) IS AVAILABLE, THEN "STARTIO" CALLS "INITIATEIO" WHICH CAUSES THE MULTIPLEXOR TO START TRANSFERRING INFORMATION. "INITIATEIO" ALSO RECORDS THE INITIATE TIME WHICH THE "IOFINISH" ROUTINE WILL USE TO CALCULATE I/O TIME FOR THE PROCESS. CONTROL IS RETURNED TO THE PROCESS REQUESTING I/O ACTION.

3. IF A PATH IS NOT AVAILABLE, THEN THE UNITWORD IS ENTERED INTO THE WAITCHANNEL QUEUE AS SHOWN IN FIGURE F4-7. CONTROL IS RETURNED TO THE PROCESS REQUESTING I/O ACTION. THE MULTIPLEXOR, AFTER OBTAINING AN I/O REQUEST VIA A PROCESSOR "INITIATE I/O" INSTRUCTION, PROCEEDS TO HANDLE THE REQUEST INDEPENDENTLY OF THE PROCESSOR. IN THE PROCESS OF DOING THE I/O, THE MULTIPLEXOR BUILDS A RESULT DESCRIPTOR. UPON COMPLETION OF THE I/O OPERATION, IT GENERATES AN I/O FINISH INTERRUPT TO THE PROCESSOR THE MCP ROUTINE "IOFINISH" IS ACTIVATED BY THIS INTERRUPT.

5.1.8. STARTIO

"STARTIO" FIRST CHECKS THE CHANNEL FOR AVAILABILITY IF THE UNIT IS IN A USABLE STATE. IF THE CHANNEL IS AVAILABLE, IT CALLS "INITIATEIO", OTHERWISE IT INSERTS THE ENTRY INTO THE WAITCHANNEL QUEUE TO WAIT FOR AN

AVAILABLE CHANNEL.

5.1.9. INITIATEIO

"INITIATEIO" INITIATES I/O, INITIALIZES I/O TIME FOR THE USER, CHECKS FOR THE UNIT TYPE AND UPDATES THE TRANSACTION COUNTER APPROPRIATELY. PARAMETERS PASSED TO "INITIATEIO" ARE AREADESC, A DESCRIPTOR POINTING TO THE IOCW WHICH PRECEDES (NONOVERLAYABLE) I/O AREA, UNITWORD, A CONTROL WORD FOR THE UNIT ON WHICH I/O IS TO BE INITIATED AND USERIDNO, A USER IDENTIFICATION NUMBER FOR BOOKKEEPING.

5.1.10. IOFINISH

"IOFINISH" READS A RESULT DESCRIPTOR FOR A SPECIFIED MULTIPLEXOR AND DOES ALL THE ERROR CHECKING. IF NO ERROR IS FOUND, IT FIRES UP NEW I/O AND INSERTS THE I/O REQUEST FOR THE UNIT ON WHICH THE I/O WAS FINISHED INTO A WAITCHANNELQUE. IF NO UNIT IS WAITING FOR A CHANNEL, IT REMOVES THE FINISHED I/O ENTRY FROM THE UNIT QUEUE AND, IF THE QUEUE IS NOT EMPTY, THEN CALLS "STARTIO". IF AN ERROR IS FOUND, IT SETS APPROPRIATE BITS IN THE UNIT TABLE. IF THERE IS A DISK ERROR, IT WILL TRY 10 TIMES TO RECOVER FROM THE ERROR. FOR OTHER ERRORS, IT CALLS PROCESS "IOERROR". IT KEEPS TRACK OF I/O TIME AND FOR ERROR FREE OPERATION, IT GIVES THE WORD COUNT FOR A READ OPERATION.

THE FIRST OPERATION OF "IOFINISH" IS TO EXECUTE THE INSTRUCTION "READ RESULT DESCRIPTOR" FOR THE INTERRUPTING MULTIPLEXOR. THIS INSTRUCTION TRANSFERS THE RESULT DESCRIPTOR FROM THE MULTIPLEXOR TO THE TOP OF THE STACK IN THE PROCESSOR AT THE SAME TIME, IT CLEARS THE INTERRUPT MECHANISM IN THE MULTIPLEXOR SO THAT IT BECOMES CAPABLE OF GENERATING ANOTHER I/O COMPLETE FOR SOME OTHER DEVICE. THE RESULT DESCRIPTOR HAS THREE FIELDS OF CONCERN: THE UNIT NUMBER, ERROR BIT AND ERROR FIELD. THE ERROR BIT IS OFF IF NO ERRORS WERE DETECTED. IF THE BIT IS ON, THEN THE RESULT DESCRIPTOR IS PASSED TO "IOERROR". "IOERROR" ANALYZES THE ERROR FIELD. THE ERROR FIELD DENOTES SUCH ERRORS AS END OF PAGE, END OF FILE, PARITY, NOT READY, ETC. DEPENDING ON THE TYPE OF ERROR, "IOERROR"

WILL TAKE APPROPRIATE ACTION.

ASSUMING "IOERROR" CORRECTED THE ERROR, OR THERE WAS NO ERROR, "IOFINISH" CONTINUES AS FOLLOWS:

1. THE I/O JUST COMPLETED IS REMOVED FROM THE I/O QUEUE.

2. IF THE WAITCHANNELQUEUE IS NOT EMPTY "NEWIO" IS CALLED TO INITIATE AN I/O OPERATION ON THE FIRST UNIT WAITING IN THE QUEUE THIS I/O QUEUE IS THEN CHECKED TO SEE IF IT IS EMPTY. IF IT IS NOT EMPTY, THEN THE NEXT I/O OPERATION REQUESTED IS PLACED IN THE WAITCHANNEL QUEUE.

3. IF THE WAITCHANNEL QUEUE IS EMPTY, "STARTIO" IS CALLED TO INITIATE THE NEXT I/O OPERATION IN THE I/O QUEUE FOR THIS UNIT.

4. THE USER I/O TIME IS RECORDED IN THE SYSTEM LOG.

5. THE I/O FINISH EVENT IS "CAUSED" WHICH MOVES THE PROCESS IN THE EVENTS "WAIT" QUEUE INTO THE READY QUEUE.

SINCE "IOFINISH" WAS ACTIVATED BY AN INTERRUPT RATHER THAN BEING CALLED, THE EXIT FROM "IOFINISH" IS DONE BY BRANCHING TO A ROUTINE WHICH WILL ACTIVATE THE PROCESS OR PROGRAM WHICH IS IN THE TOP OF THE READY QUEUE.

5.1.11. FINDINPUT AND FINDOUTPUT

THE PROCEDURES "FINDINPUT" AND "FINDOUTPUT" FIND THE UNIT INPUT OR OUTPUT RESPECTIVELY FOR THE FILE UNDER CONSIDERATION, BY SEARCHING THE UNIT TABLE, DISK DIRECTORY AND LINE AND TERMINAL FILE. NOTE: TRANSLATION FOR THE MULTIPLEXOR DOES NOT WORK ON A BINARY TAPE FILE, SO IT IS ADVISABLE TO HAVE EBCDIC TAPE FILES.

5.2. DIRECT I/O

DIRECT I/O FACILITATES THE MAXIMUM OVERLAPPING OF USER "I/O TIME" AND "PROCESSOR TIME", PERMITS WORKING DIRECTLY OUT OF BUFFERS, AND ALLOWS THE SPECIFICATION OF CERTAIN UNUSUAL FILE ATTRIBUTES.

THE CHARACTERISTICS OF DIRECT I/O ARE:

1. THE USER EXPLICITLY PROVIDES HIS OWN BUFFERS IN THE FORM OF AN ARRAY "CROSS-SECTION" (SOME PORTION OF AN ARRAY ROW).

2. ALL RECORDS ARE HANDLED AS UNBLOCKED.

3. CONTROL IS RETURNED TO THE USER BEFORE THE "I/O COMPLETE" EVENT OCCURS.

4. THE USER MAY PROVIDE HIS OWN BUFFER EVENT(S).

5. A POOL OF I/O CONTROL BLOCKS (IOCB) IS MAINTAINED BY THE SYSTEM FOR THE USER.

6. THE USER MUST "FREE" AN IOCB BEFORE IT CAN BE USED IN ANY FURTHER I/O ACTION.

7. CERTAIN "PRIVILEGED" ATTRIBUTES, SUCH AS "FILE DENSITY" CAN BE CHANGED WHILE THE FILE IS OPEN.

5.3. BUFFERED I/O

USER I/O USES MCP PROVIDED INTRINSICS FOR FORMATTING, DECOMPOSITION INTO LISTS, AND BLOCKING.

AT FILE OPEN TIME, THE LABEL EQUATION INFORMATION IS CHECKED FOR CONSISTENCY. IF THE BIT IN THE FIB THAT MARKS THE FIRST OPENING OF A FILE IN A BLOCK IS NOT ON, IT IS NECESSARY TO PROCESS LABEL EQUATION INFORMATION. TO DO THIS, THE PPB LOOKS IN THE PROCESS STACK FOR INFORMATION AND MAPS THAT INFORMATION INTO THE FIB. AFTER THE FILE IS OPENED, "SRTUPTANK" SETS UP THE BUFFERS AND LINKS THEM TOGETHER AND BUILDS THE IOAREA (IOCB + LINKS + BUFFER).

THE AUTOMATIC TRANSFER OF "LOGICAL" RECORDS BETWEEN A FILE AND PROCESS NEEDS BOTH INFORMATION ABOUT RECORD SIZE, BLOCK SIZE AND BLOCKING, AS WELL AS THE FUNCTIONING OF SERIAL I/O AND RANDOM I/O.

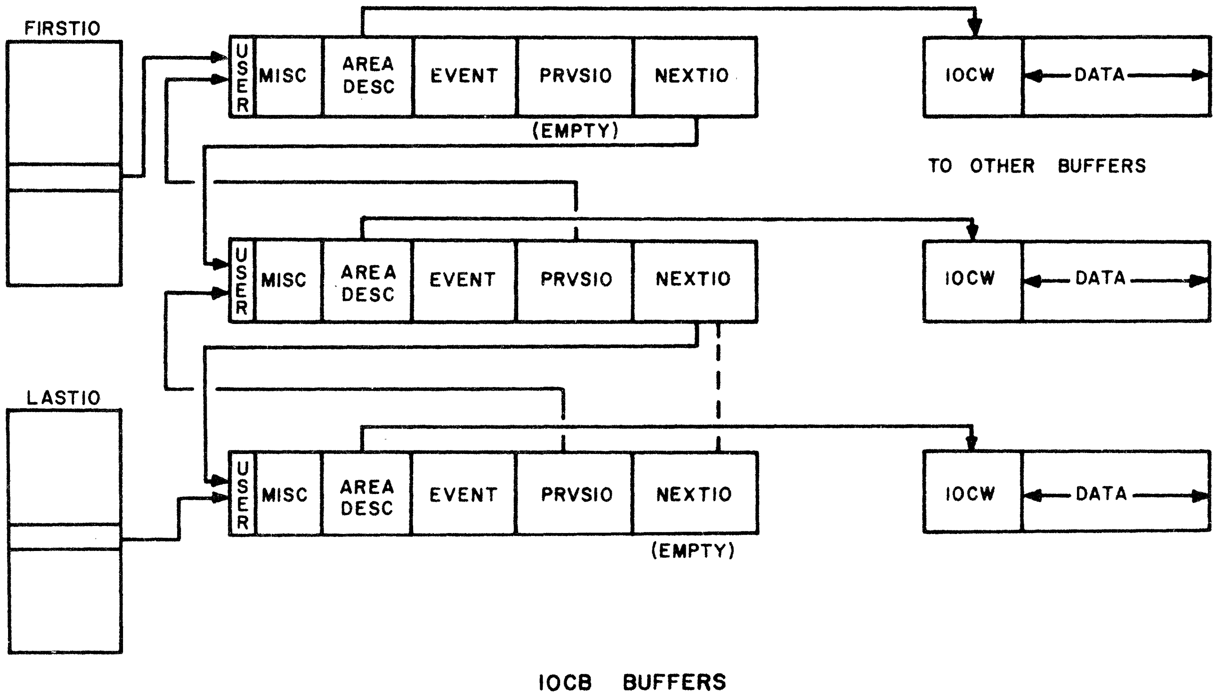


FIGURE F5-3. MCP I/O QUEUE

5.3.1. LOGICAL RECORD AND PHYSICAL RECORD

A LOGICAL RECORD CONSISTS OF THE INFORMATION WHICH THE PROCESS REFERENCES WITH ONE READ OR WRITE STATEMENT. THE SIZE OF A LOGICAL RECORD DOES NOT IN GENERAL COINCIDE WITH THE SIZE OF THE PHYSICAL RECORD OR "BLOCK" ACCESSED BY THE HARDWARE I/O OPERATIONS.

THE BLOCK SIZE IS THE SIZE OF A SET OF DATA THAT CAN BE PROCESSED BY THE HARDWARE ON EACH ACTUAL HARDWARE I/O OPERATION. THE LIMITING FACTOR IN SIZE OF A BLOCK IS DEPENDENT ON EACH HARDWARE DEVICE. FOR EXAMPLE: CARD READERS ARE FIXED AT 80 CHARACTERS PER BLOCK, TAPE IS VARIABLE IN INCREMENTS OF 1 TO 16,767 WORDS AND DISK BLOCK SIZE IS VARIABLE IN INCREMENTS OF 30 WORD SEGMENTS. WHEN A PHYSICAL RECORD CONTAINS MORE THAN ONE LOGICAL RECORD; THE FILE IS REFERRED TO AS A "BLOCKED" FILE.

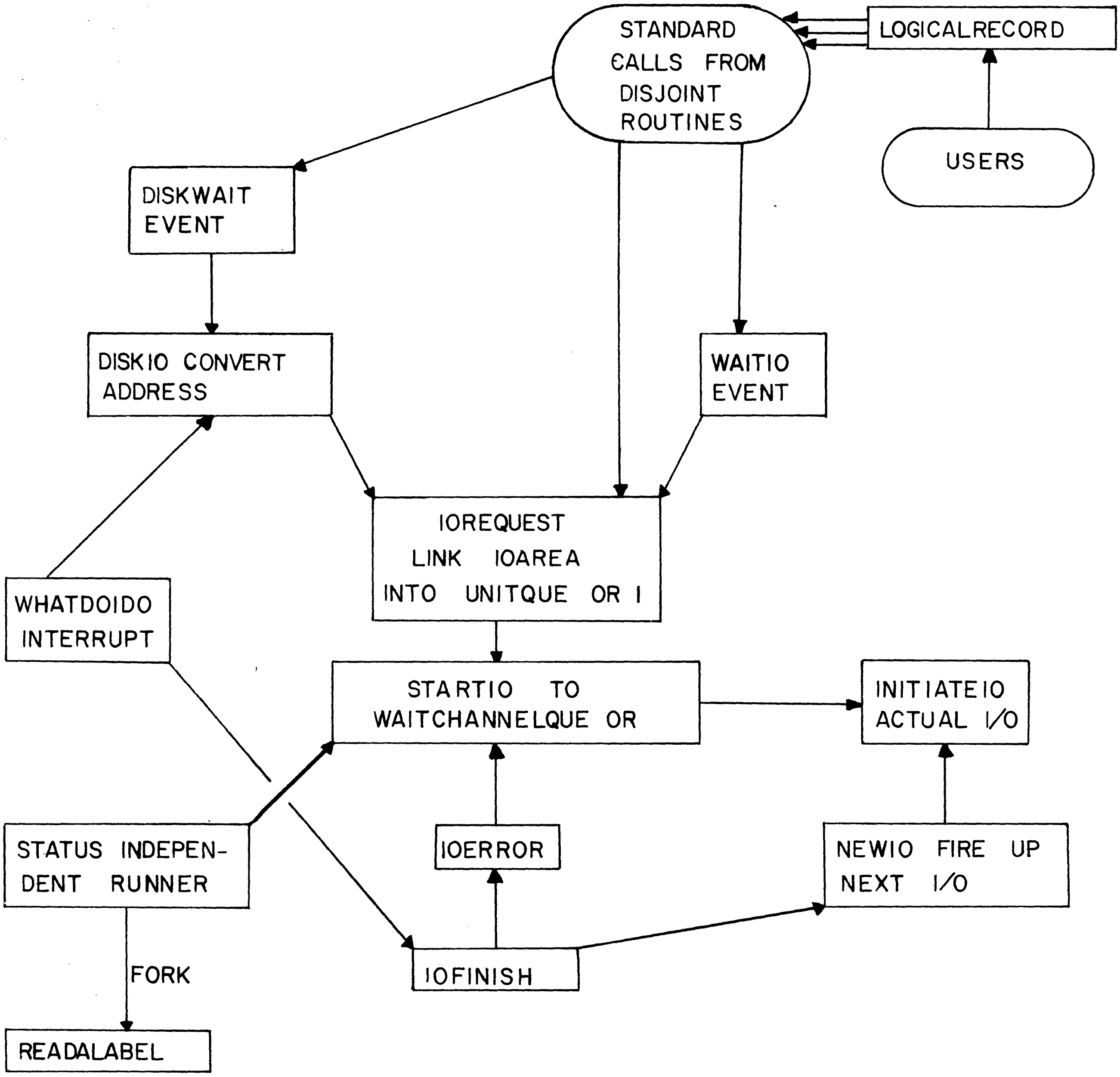


FIGURE F5-4. INTRINSIC CALLS

5.3.2. BLOCKING

FILES MAY BE BLOCKED IN ORDER TO CONSERVE STORAGE SPACE IN THE FILE MEDIA OR TO INCREASE THE RATE OF PROCESSING THE DATA BY REDUCING THE NUMBER OF FILE ACCESSES REQUIRED.

WHEN A PROCESS HAS ACCESS TO A FILE A PHYSICAL RECORD IS WRITTEN EITHER FROM OR READ TO A MEMORY AREA. THIS MEMORY AREA IS CALLED A "BUFFER" AREA FOR THE FILE. THE BUFFER AREA PROVIDES THE INTERFACE BETWEEN THE HARDWARE DEVICE AND THE SOURCE LANGUAGE I/O STATEMENTS. THERE MUST ALWAYS BE ONE MEMORY AREA USED AS A BUFFER FOR EACH FILE. IF THE FILE IS BLOCKED, THE HARDWARE MUST PROCESS ONE BLOCK OF DATA ON EACH I/O OPERATION; THEREFORE THE MEMORY AREA SHOULD BE AT LEAST AS LARGE AS ONE BLOCK. THE MCP MAINTAINS A RECORD POINTER INTO A BUFFER USED FOR A BLOCKED FILE. THIS POINTER IS USED BY THE PROCESS TO ACCESS THE CURRENT LOGICAL RECORD, SUPPLYING ONE RECORD AT A TIME FROM THE BLOCK TO THE PROGRAM. IF THE NEXT RECORD REQUIRED IS NOT ALREADY PRESENT IN A BUFFER, THEN THE MCP AUTOMATICALLY PERFORMS THE REQUIRED I/O OPERATION.

THE MCP ATTEMPTS TO KEEP ALL INPUT BUFFERS FULL AND ALL OUTPUT BUFFERS EMPTY FOR EACH PROCESS, REGARDLESS OF STATUS, THEREBY MINIMIZING THE TIME THAT A PROCESS IS SUSPENDED WAITING FOR AN I/O OPERATION TO BE COMPLETED.

5.3.3. MULTIPLE BUFFERS

THE USE OF MORE THAN ONE BUFFER ALSO CAN BE USED TO INCREASE PROCESSING SPEED OF DATA, SINCE MULTIPLE BUFFERS ALLOW I/O TO BE PERFORMED ON ONE BUFFER AT THE SAME TIME A LOGICAL RECORD IS BEING ACCESSED IN ANOTHER BUFFER.

THE ORIGINAL PURPOSE OF MULTIPLE BUFFERS WAS TO INCREASE SYSTEM EFFICIENCY BY OVERLAPPING I/O OPERATIONS WITH PROCESSOR COMPUTATIONS. SINCE MULTIPROCESSING ALLOWS OVERLAP OF I/O OPERATIONS AND PROCESSOR

COMPUTATIONS BETWEEN DIFFERENT PROCESSES, MUCH OF THE ORIGINAL NEED FOR MULTIPLE BUFFERS WOULD SEEM TO BE OBVIATED.

IN THE B6700 SYSTEM, HOWEVER, THE EXISTENCE OF PARALLEL I/O MULTIPLEXOR CHANNELS ALLOWS MULTIPLE BUFFERS TO STILL BE EFFECTIVE IN INCREASING THROUGHPUT FOR PROCESSES WHICH REQUIRE GROUPS OF PHYSICAL RECORDS AT A TIME. SINCE THE MCP PERFORMS ALL OBJECT PROGRAM I/O ACTION, A PROCESS WITH MULTIPLE BUFFERS ALLOCATED FOR A FILE ALLOWS THE MCP TO PERFORM I/O OPERATIONS INDEPENDENT OF THE STATUS OF THE PROCESS.

THE DETERMINATION OF THE NUMBER OF BUFFERS REQUIRED FOR EFFICIENT EXECUTION OF A PROCESS DEPENDS UPON MANY FACTORS. THESE FACTORS INCLUDE: THE TYPE OF FILES BEING USED, THE PARTICULAR HARDWARE CONFIGURATION BEING USED, THE PROCESSING CHARACTERISTICS OF THE PROCESS, THE MEMORY REQUIREMENT OF THE PROCESS AND THE MIX OF PROCESSES WHICH ARE TYPICALLY MULTIPROCESSED.

PARTICULAR NOTE SHOULD BE MADE OF THE FACT THAT THE USE OF EXCESSIVELY LARGE BUFFERS OR AN EXCESSIVE NUMBER OF BUFFERS FOR PROCESSES CAN CAUSE UNNECESSARY OVERLAYS OF PROGRAM SEGMENTS AND DATA. THIS, IN TURN, WILL RESULT IN REDUCED SYSTEM THROUGHPUT AND POOR SYSTEM PERFORMANCE.

5.3.4. RANDOM RECORD ACCESS

SINCE ACTUAL I/O OPERATIONS MAY INVOLVE BLOCKS OF RECORDS, WHEN A READ OR UPDATE IS REQUIRED ON A RECORD, THE ENTIRE BLOCK CONTAINING THE RECORD MUST BE READ. IF THE I/O ACTION IS A "RANDOM" ACTION, WHICH MAY BE SPECIFIED FOR FILES SUCH AS DISK FILES, THE RECORD REQUIRED MAY HAVE BEEN INCLUDED IN A BLOCK OF RECORDS WHICH WAS PREVIOUSLY ACCESSED. THEREFORE, IN ORDER TO ELIMINATE UNNECESSARY I/O ACTIONS,THE MCP REMEMBERS WHICH LOGICAL RECORDS ARE CURRENTLY HELD IN EACH BUFFER. WHEN A REQUEST IS MADE FOR A PARTICULAR RECORD, THE BUFFERS ARE FIRST CHECKED TO DETERMINE WHETHER THE RECORD ALREADY EXISTS IN THE BUFFER. IF IT IS, THEN THE RECORD POINTER IS SET TO POINT AT IT AND CONTROL RETURNS TO THE OBJECT PROGRAM IMMEDIATELY. IF THE RECORD IS NOT ALREADY IN THE BUFFER, THEN THE MCP MUST BE CALLED TO LOAD THE BLOCK CONTAINING THE RECORD, THE

PROGRAM MUST BE SUSPENDED UNTIL THE RECORD IS LOADED, THEN THE RECORD POINTER IS SET TO POINT AT IT.

5.3.5. SEEK

THE SEEK FUNCTION CAN BE ACTIVATED BY THE SOURCE LANGUAGE. ASSOCIATED WITH THE SEEK IS A RECORD ADDRESS. SEEK HAS TWO PURPOSES DEPENDING ON WHETHER OR NOT THE FILE IS SERIAL OR RANDOM. SEEK ON SEQUENTIAL FILES SERVES TO RESET THE FILE TO START SEQUENTIAL PROCESSING AT THE RECORD INDICATED IN THE SEEK STATEMENT. FOR EXAMPLE: THE FIRST I/O STATEMENT AFTER A "SEEK (RECORD N)" WOULD MAKE RECORD N AVAILABLE TO THE PROGRAM. THE RECORD BEING PROCESSED PRIOR TO THE SEEK IS NOT DISTURBED BY THE SEEK THAT IS. ITS STILL AVAILABLE.

THE SEEK STATEMENT ON RANDOM ACCESS IS INTENDED TO CAUSE THE SYSTEM TO PRELOCATE THE NEXT RECORD WHILE THE PROGRAM IS PROCESSING THE CURRENT RECORD. THE SEEK FIRST EXAMINES THE BUFFERS TO SEE IF THE RECORD ALREADY EXISTS IN A BUFFER. IF THE RECORD IS IN A BUFFER, CONTROL IS RETURNED TO THE PROGRAM.

IF THE RECORD IS NOT IN A BUFFER AND THERE IS ONLY ONE BUFFER, THEN CONTROL IS RETURNED TO THE PROGRAM. IT SHOULD BE NOTED THAT THE USE OF "SEEK" ON FILES WITH ONLY ONE BUFFER CAUSES UNNECESSARY MCP OVERHEAD AND SHOULD BE AVOIDED. IF THERE ARE MULTIPLE BUFFERS, THEN THE MCP MAY BE CALLED TO LOAD THE BLOCK CONTAINING THAT RECORD, DEPENDING UPON THE NUMBER OF BUFFERS THAT HAVE PREVIOUSLY BEEN SOUGHT. ASSUMING THE RECORD POINTER IS POINTING AT BUFFER NUMBER 1, THEN CONSECUTIVE READ SEEKS ARE ALTERNATED THROUGH BUFFERS 2 THROUGH N.

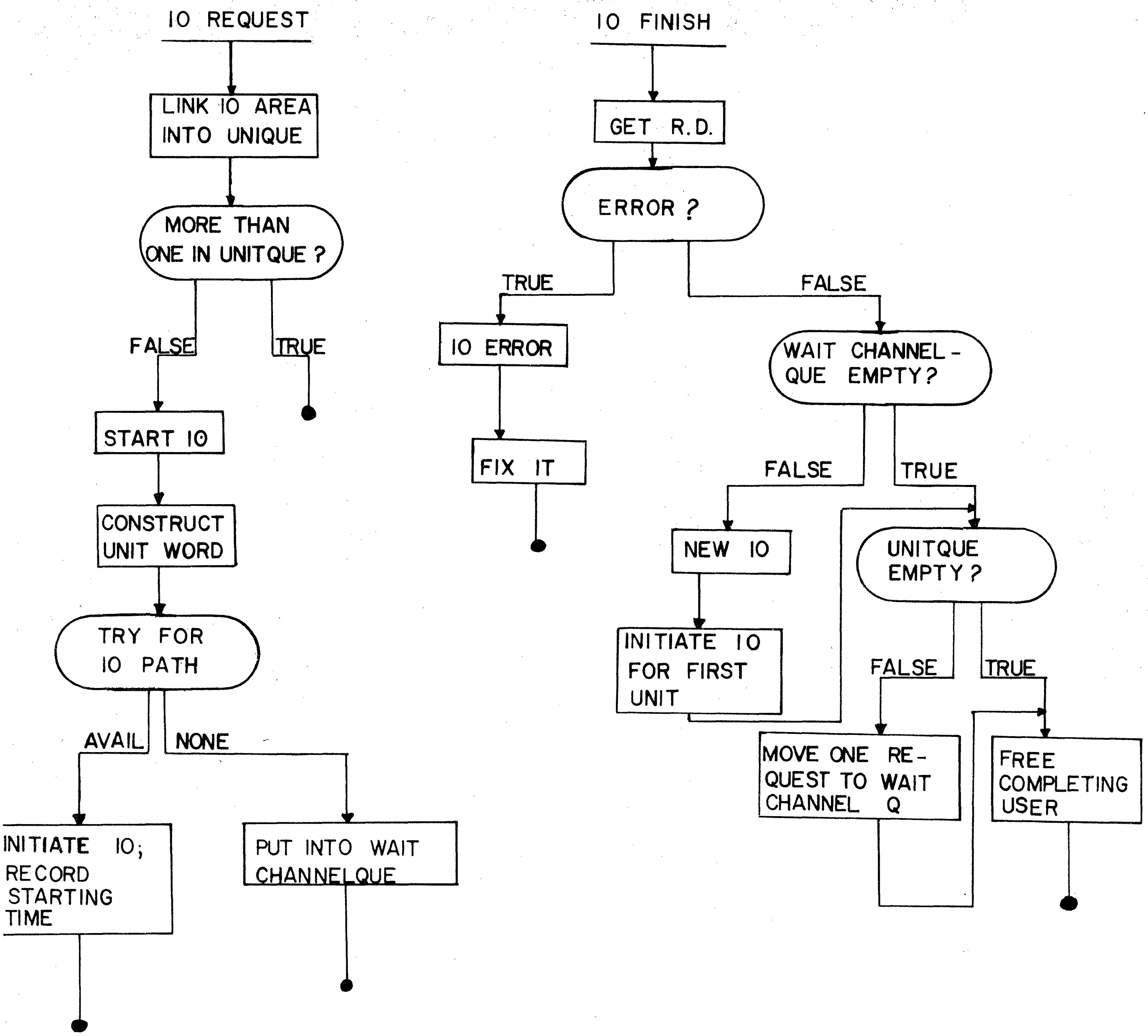


FIGURE F5-5. DETAIL OF I/O INTRINSIC CALLS

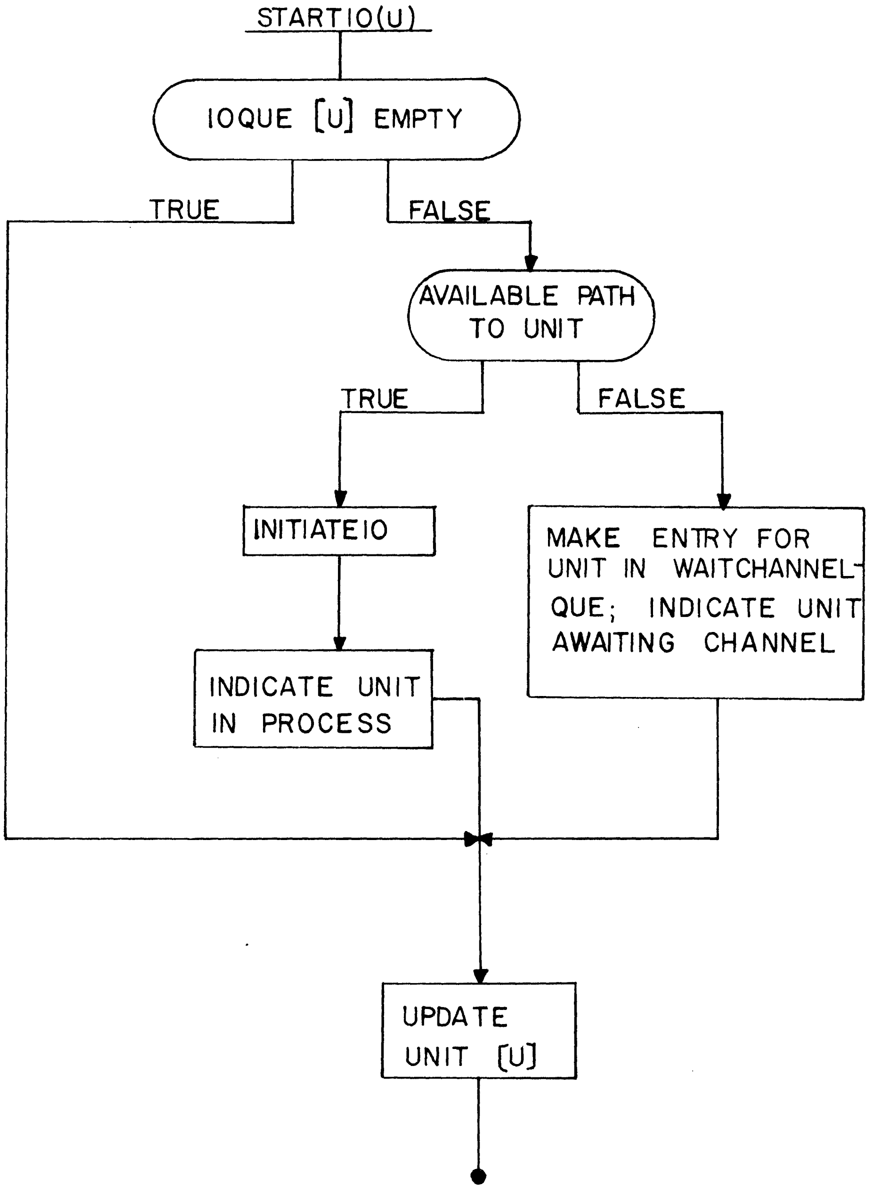


FIGURE **F5-6.** STARTIO(U) FLOW

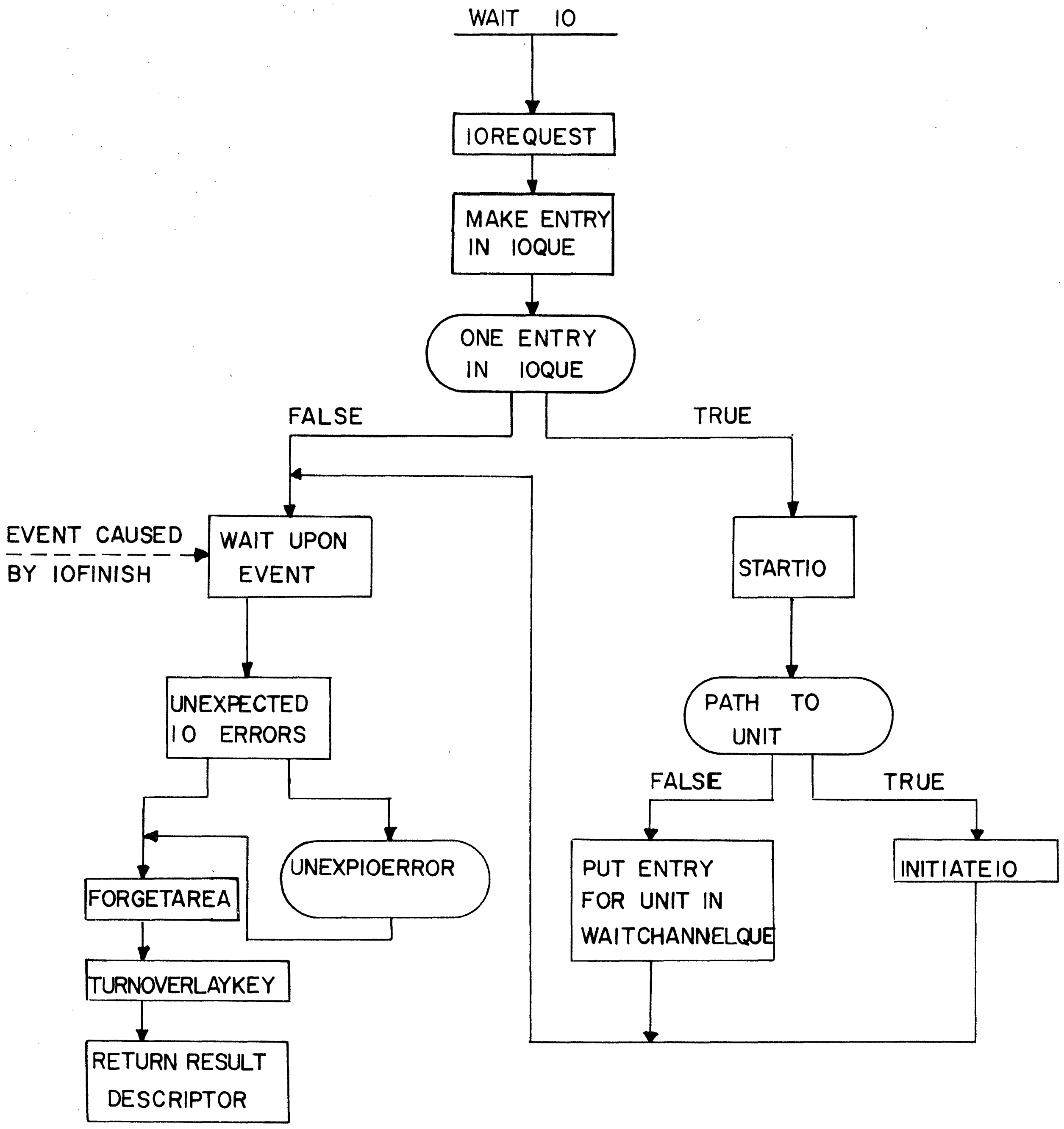


FIGURE F5-7. WAITIO GENERAL FLOW

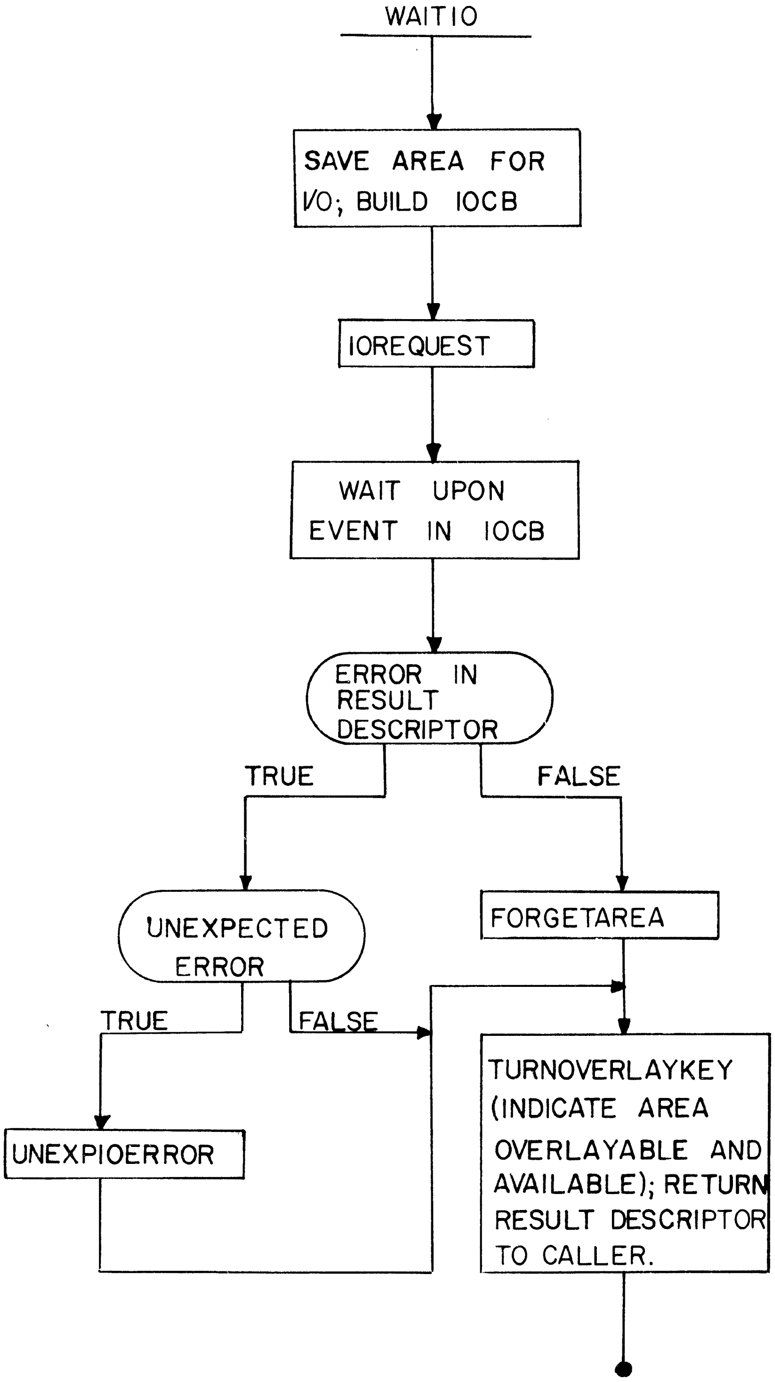

WAITIO
SAVE AREA FOR I/O; BUILD IOCB
IOREQUEST
WAIT UPON EVENT IN IOCB
ERROR IN RESULT DESCRIPTOR
TRUE
FALSE
UNEXPECTED ERROR
FORGETAREA
TRUE
FALSE
UNEXPIOERROR
TURNOVERLAYKEY (INDICATE AREA OVERLAYABLE AND AVAILABLE); RETURN RESULT DESCRIPTOR TO CALLER.

FIGURE F5-8. WAITIO FLOW

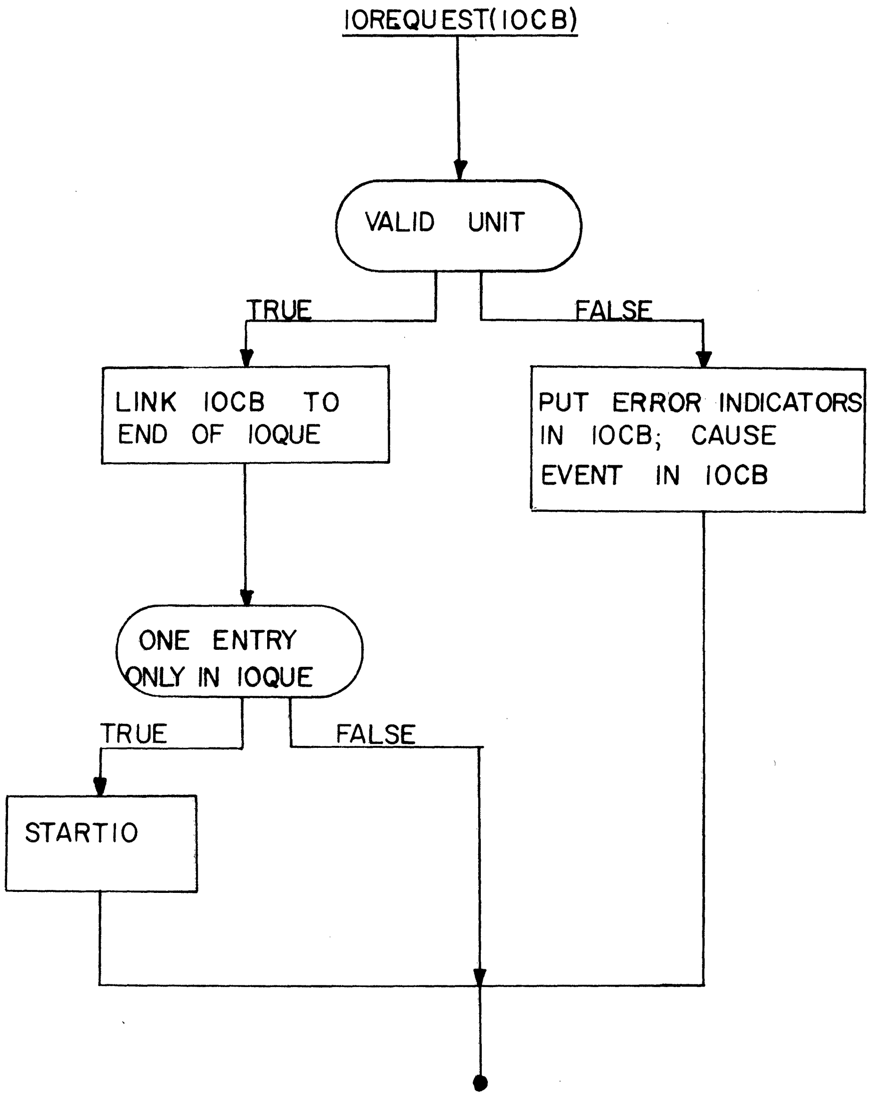

IOREQUEST(IOCB)
VALID UNIT
TRUE
FALSE
LINK IOCB TO END OF IOQUE
PUT ERROR INDICATORS IN IOCB; CAUSE EVENT IN IOCB
ONE ENTRY ONLY IN IOQUE
TRUE
FALSE
STARTIO

FIGURE F5-9. IOREQUEST(IOCB) FLOW

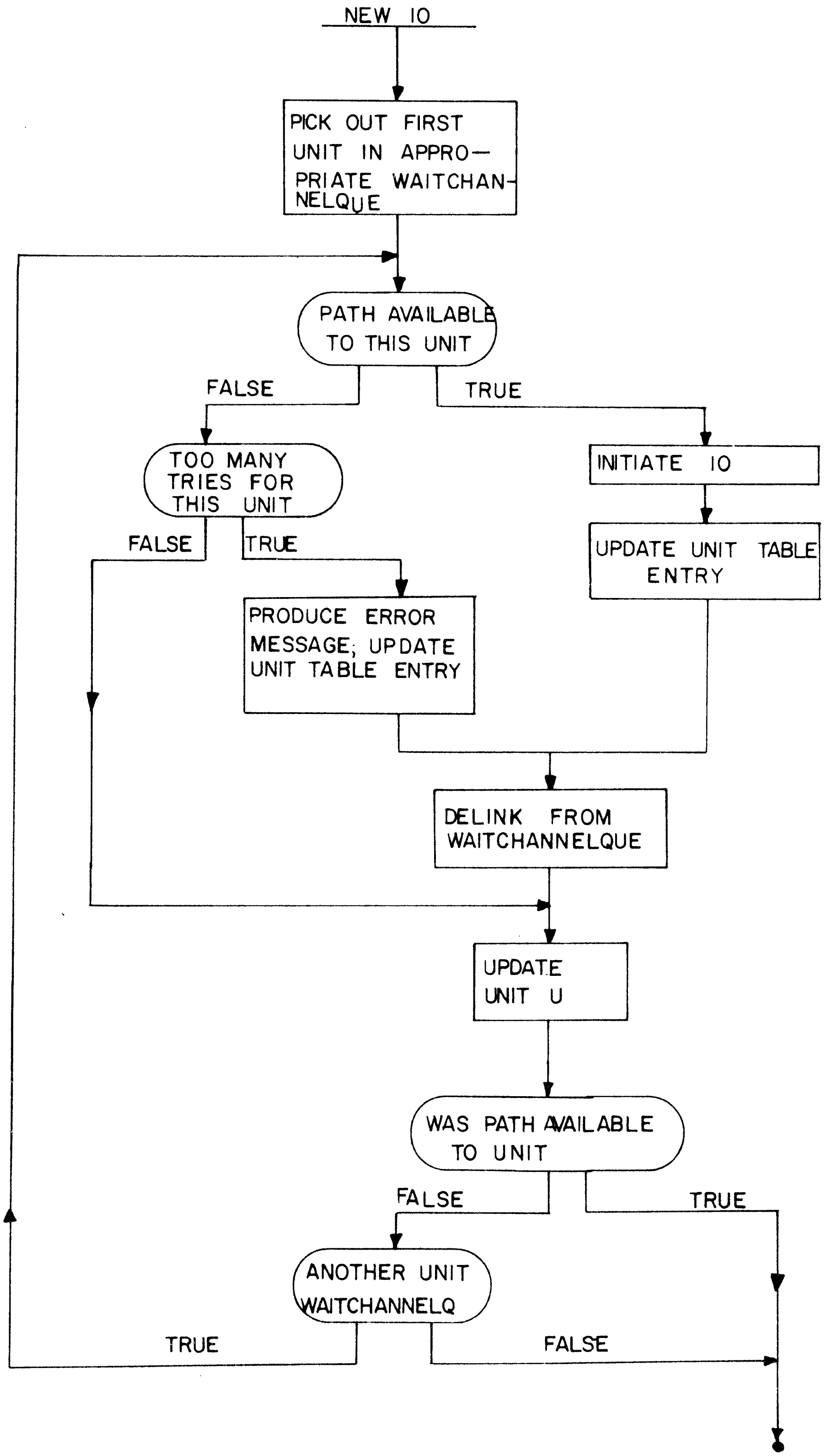

NEW IO
PICK OUT FIRST UNIT IN APPRO- PRIATE WAITCHAN- NELQUE
PATH AVAILABLE TO THIS UNIT
FALSE
TRUE
TOO MANY TRIES FOR THIS UNIT
INITIATE IO
FALSE
TRUE
UPDATE UNIT TABLE ENTRY
PRODUCE ERROR MESSAGE; UPDATE UNIT TABLE ENTRY
DELINK FROM WAITCHANNELQUE
UPDATE UNIT U
WAS PATH AVAILABLE TO UNIT
FALSE
TRUE
ANOTHER UNIT WAITCHANNELQ
TRUE
FALSE

FIGURE F5-10. NEWIO

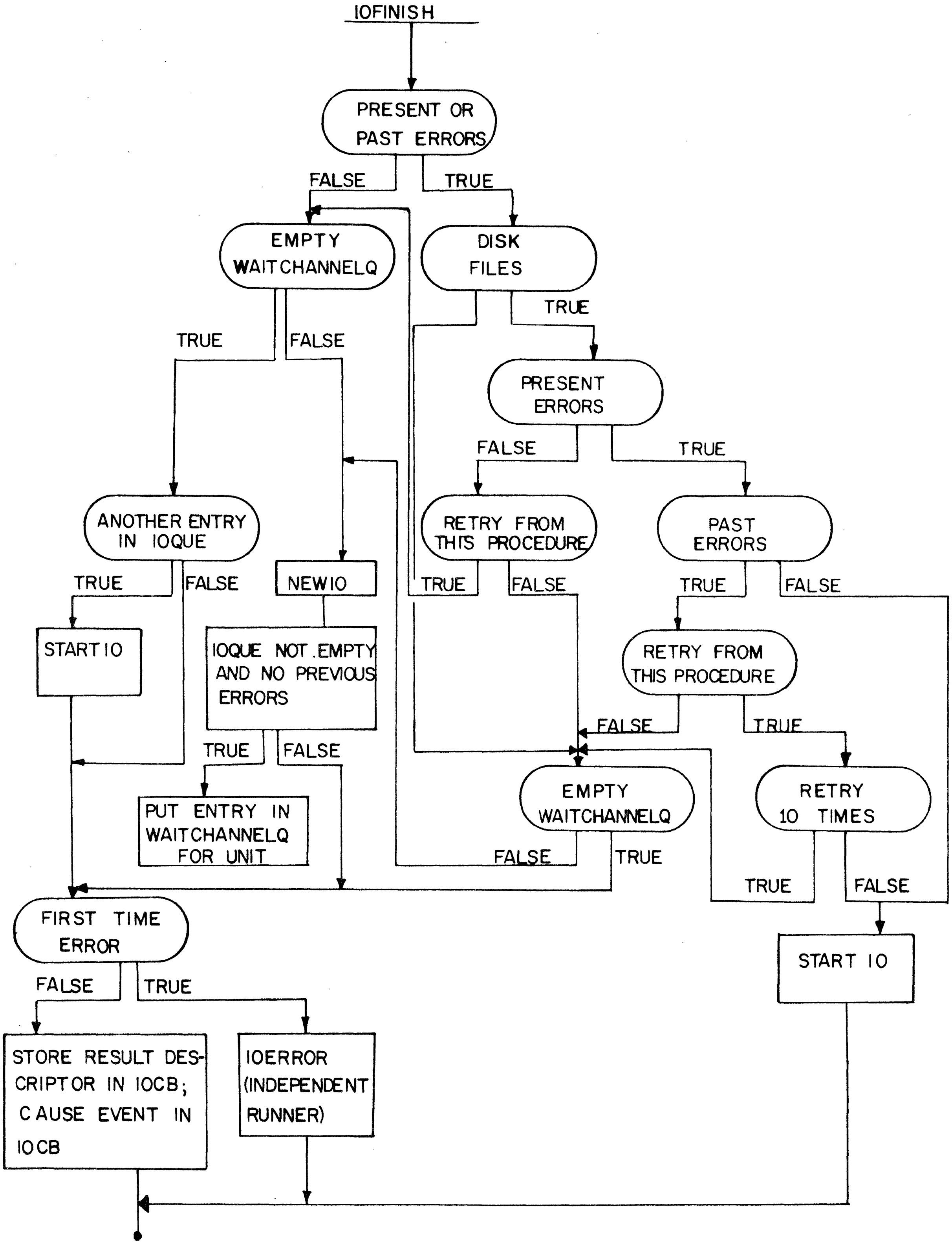


FIGURE F5-11. IOFINISH GENERAL FLOW

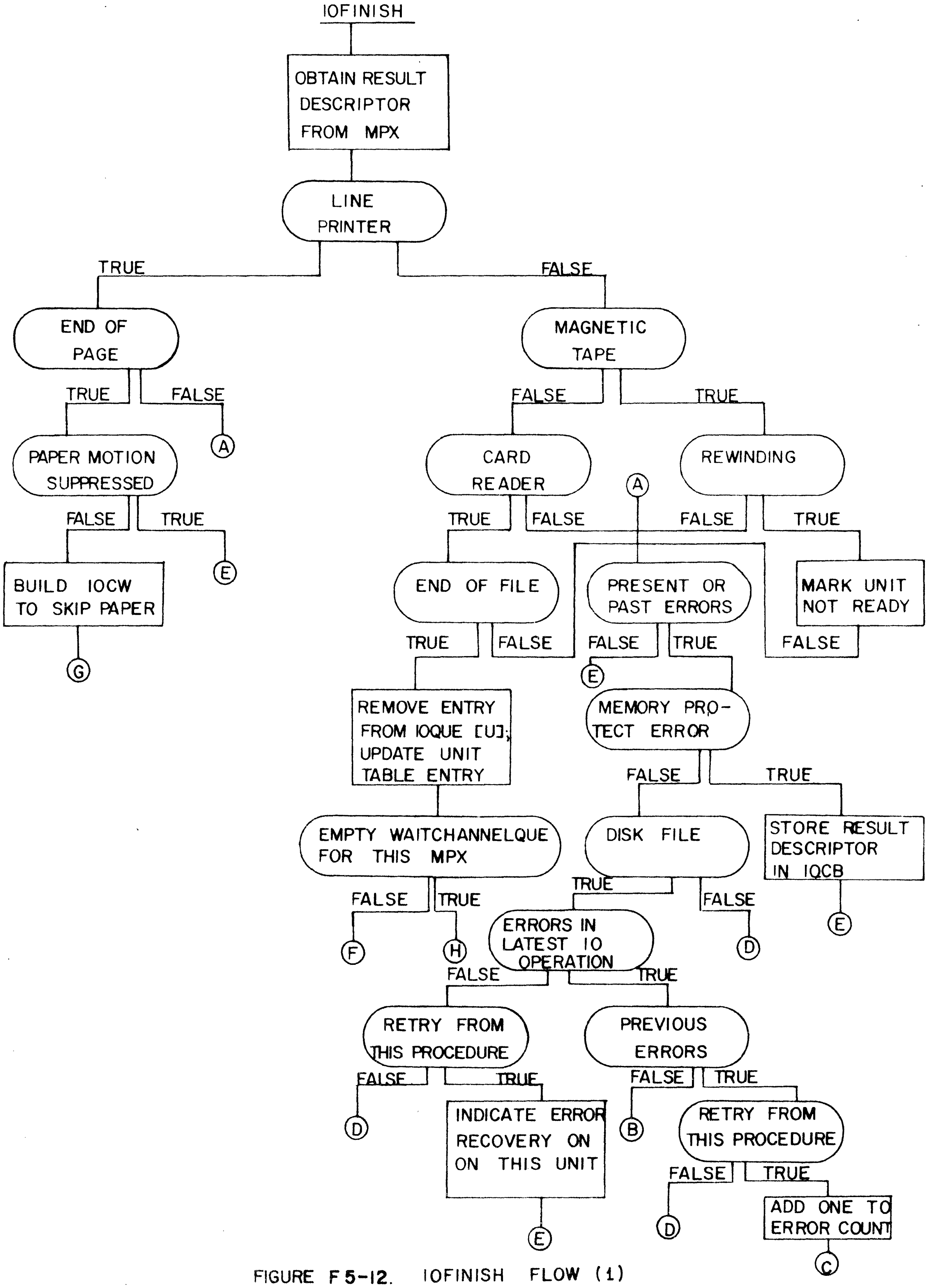


FIGURE F5-12. IOFINISH FLOW (1)

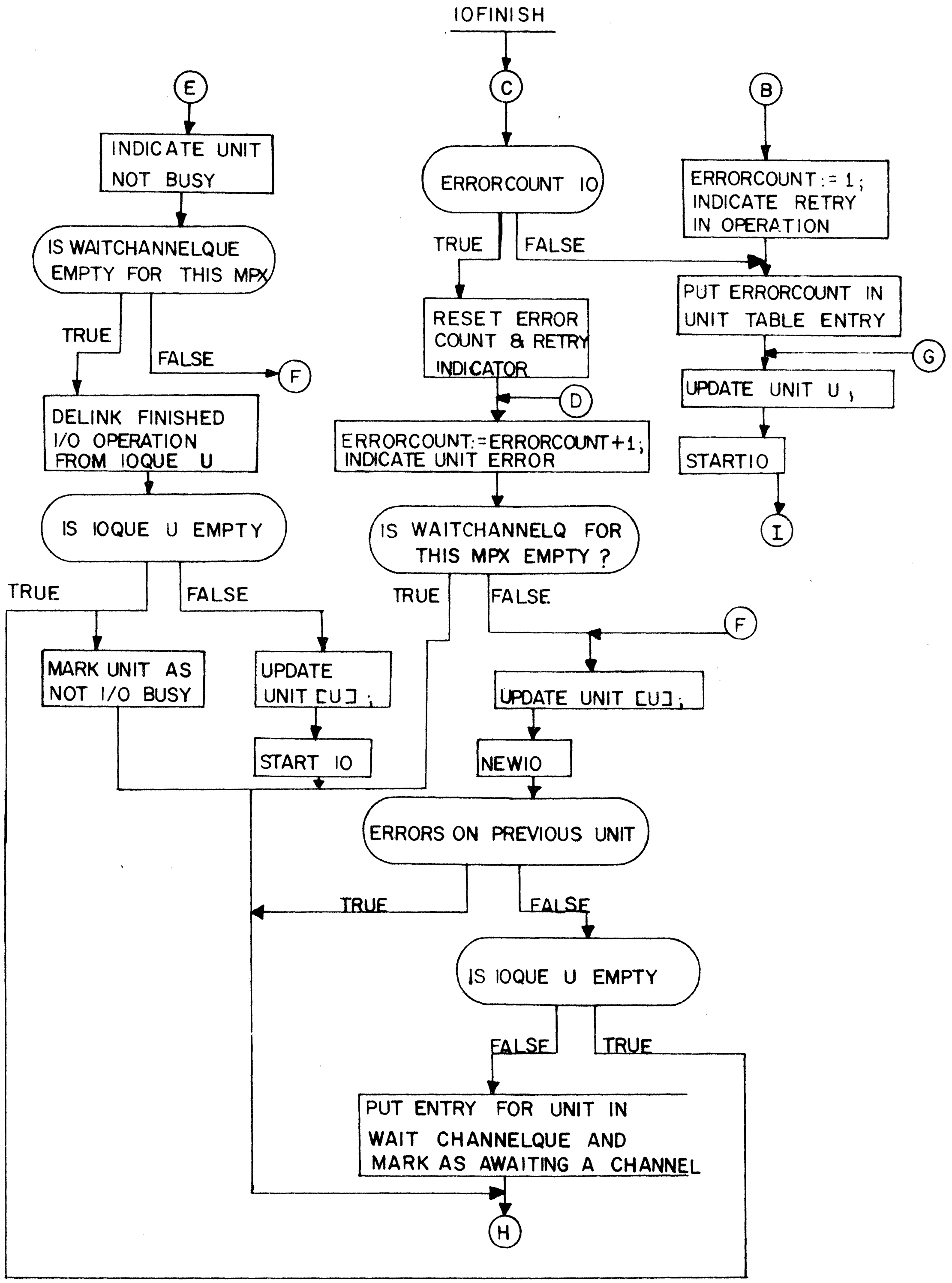

IOFINISH
C
ERRORCOUNT IO
TRUE
FALSE
RESET ERROR COUNT & RETRY INDICATOR
D
ERRORCOUNT:=ERRORCOUNT+1; INDICATE UNIT ERROR
IS WAITCHANNELQ FOR THIS MPX EMPTY ?
TRUE
FALSE
F
UPDATE UNIT [U];
NEWIO
ERRORS ON PREVIOUS UNIT
TRUE
FALSE
IS IOQUE U EMPTY
FALSE
TRUE
PUT ENTRY FOR UNIT IN WAIT CHANNELQUE AND MARK AS AWAITING A CHANNEL
H
E
INDICATE UNIT NOT BUSY
IS WAITCHANNELQUE EMPTY FOR THIS MPX
TRUE
FALSE
F
DELINK FINISHED I/O OPERATION FROM IOQUE U
IS IOQUE U EMPTY
TRUE
FALSE
MARK UNIT AS NOT I/O BUSY
UPDATE UNIT [U];
START IO
B
ERRORCOUNT:= 1; INDICATE RETRY IN OPERATION
PUT ERRORCOUNT IN UNIT TABLE ENTRY
G
UPDATE UNIT U ;
STARTIO
I

FIGURE F5-13. IOFINISH FLOW (2)

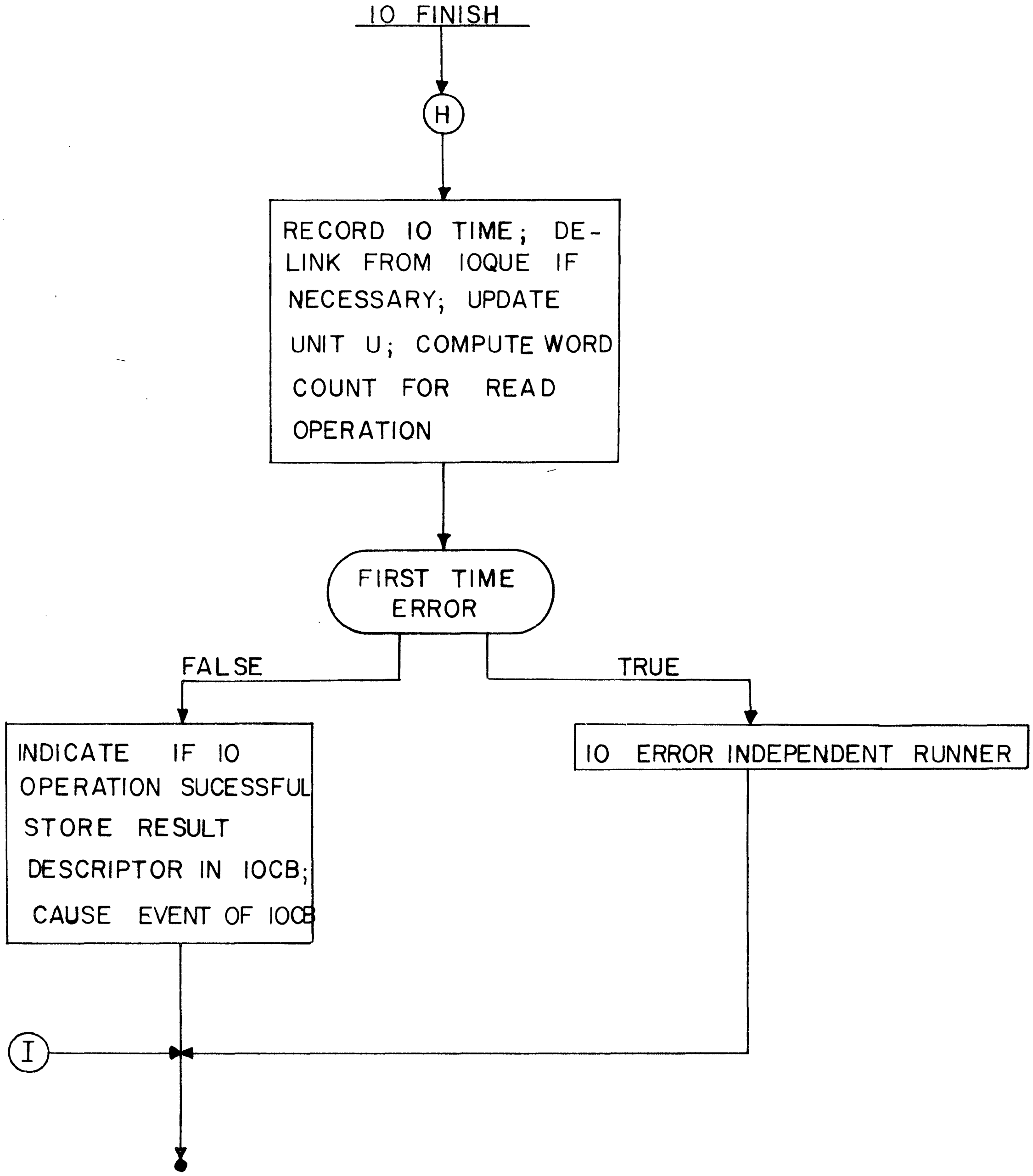

IO FINISH
H
RECORD IO TIME; DE-LINK FROM IOQUE IF NECESSARY; UPDATE UNIT U; COMPUTE WORD COUNT FOR READ OPERATION
FIRST TIME ERROR
FALSE
TRUE
INDICATE IF IO OPERATION SUCESSFUL STORE RESULT DESCRIPTOR IN IOCB; CAUSE EVENT OF IOCB
IO ERROR INDEPENDENT RUNNER
I

FIGURE F5-14. IOFINISH FLOW (3)

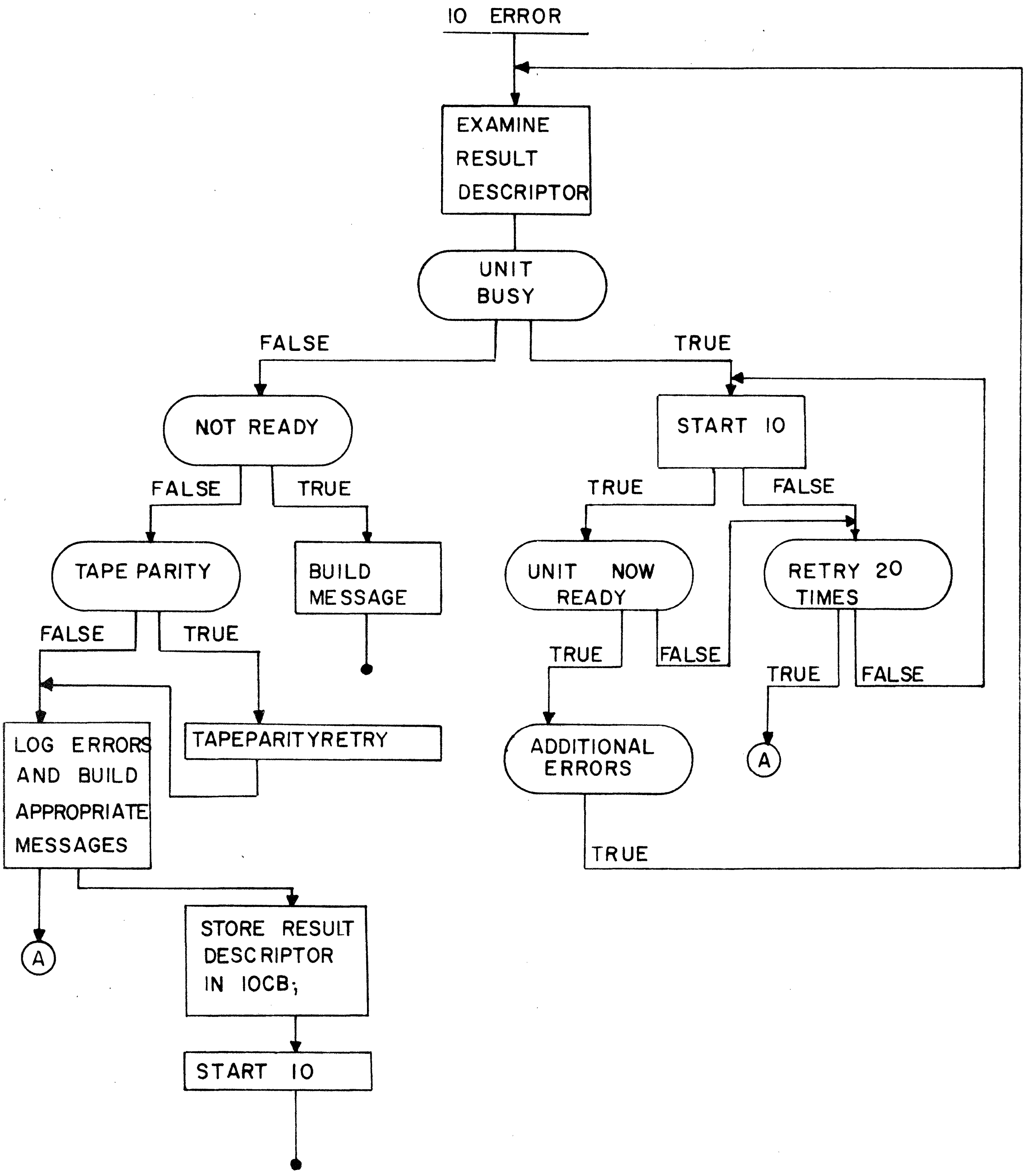


FIGURE F5-15. IOERROR GENERAL FLOW

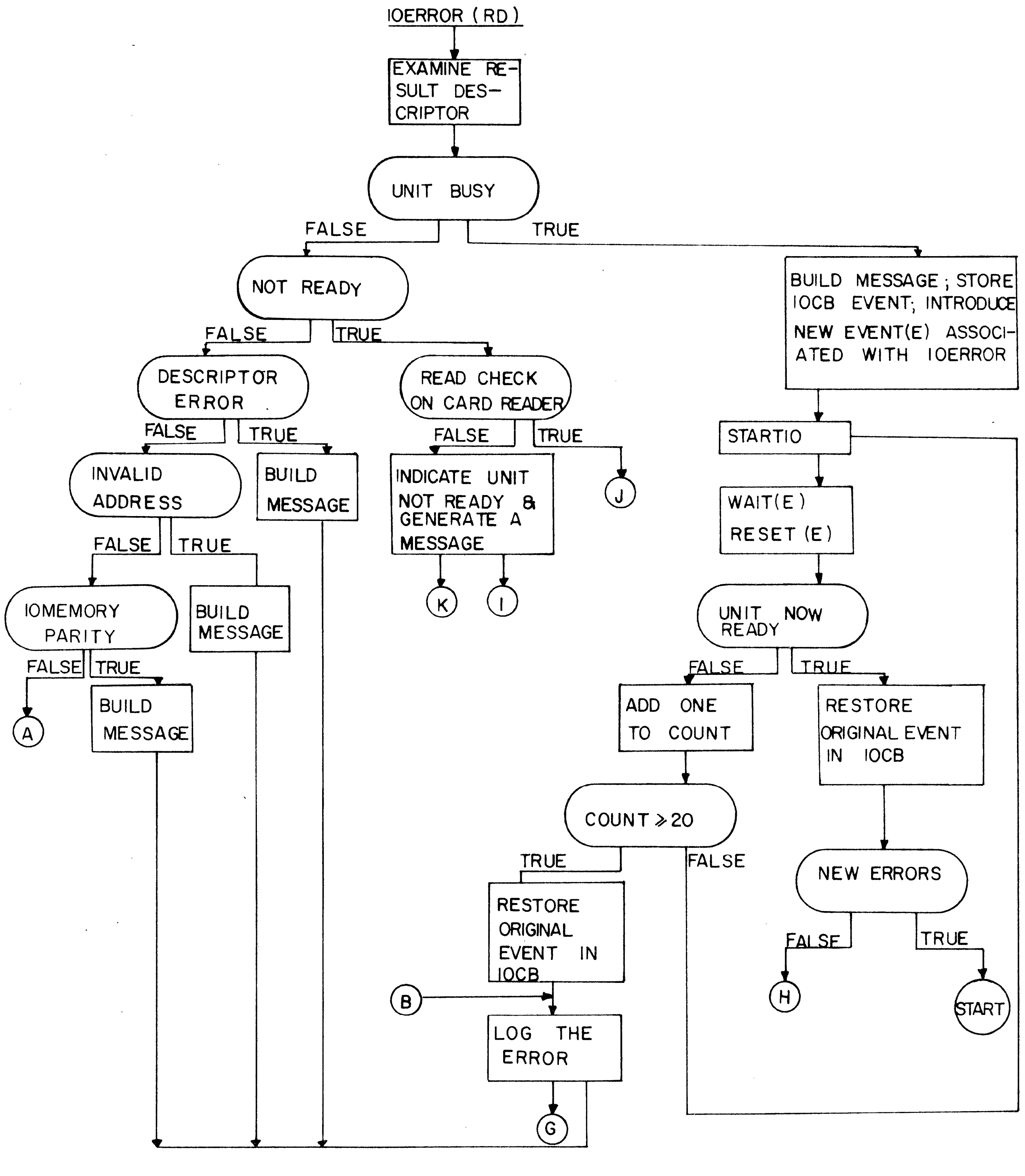


FIGURE F5-16. IOERROR(RD) FLOW (I)

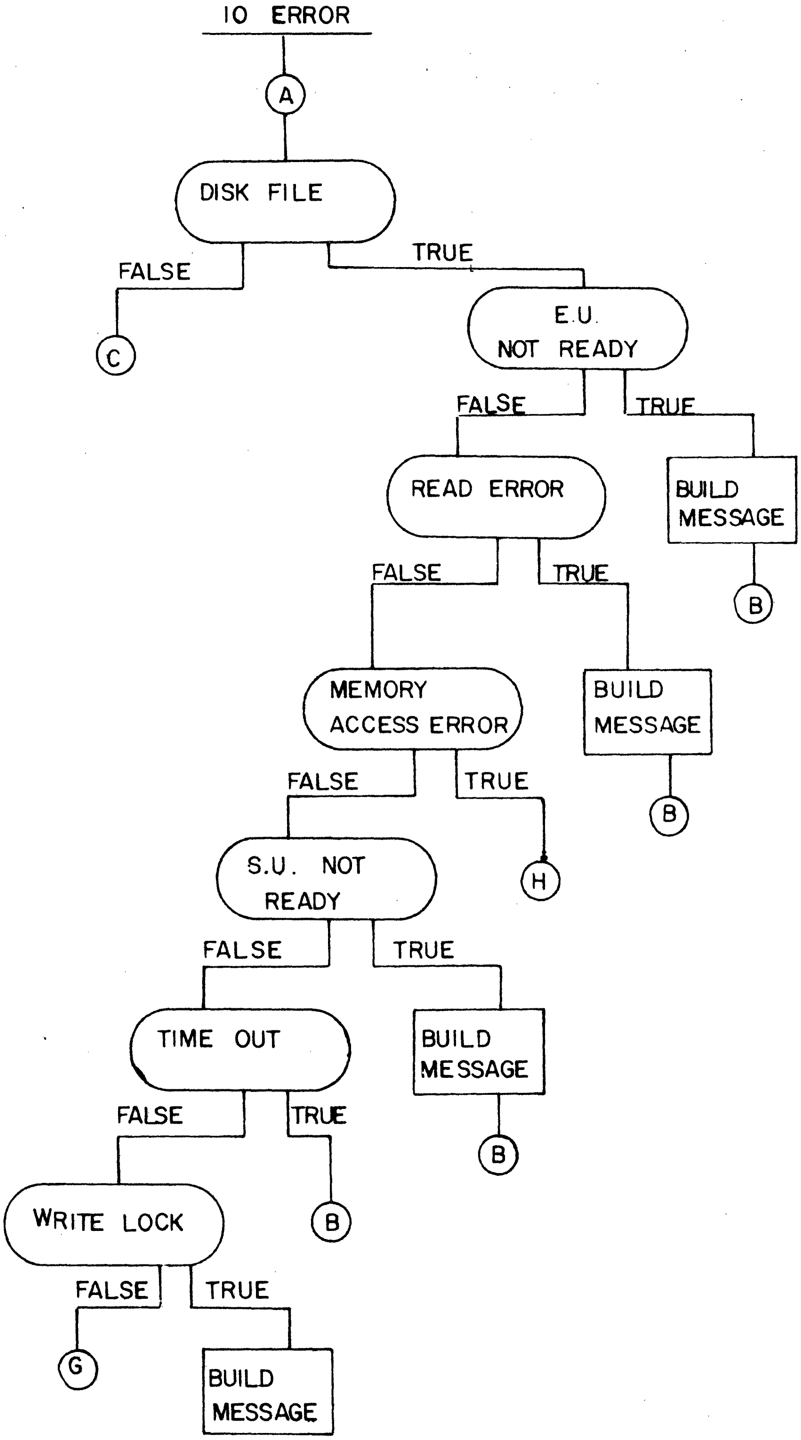


FIGURE F5-17. IOERROR(RD) FLOW (2)

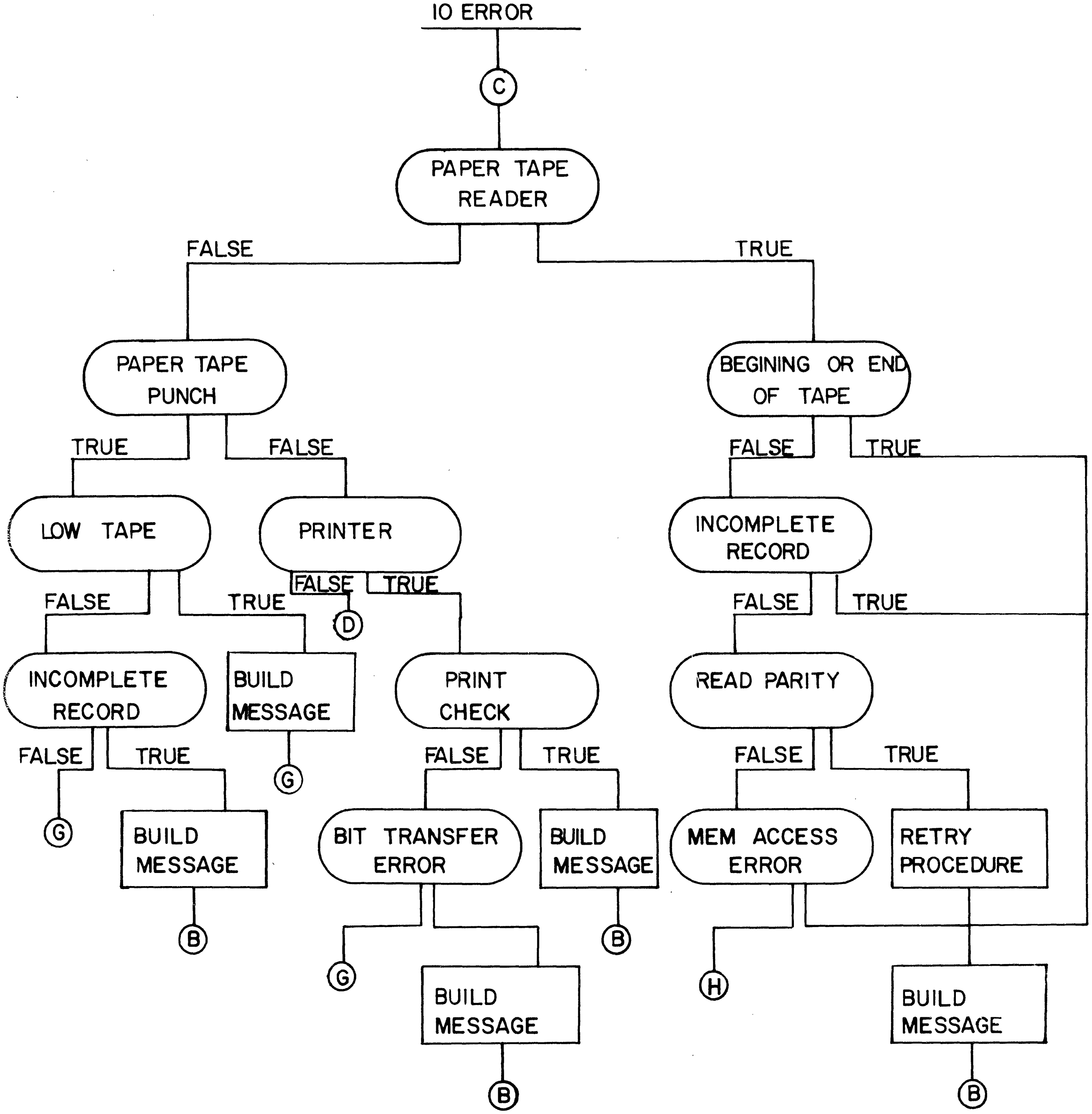


FIGURE F5-18. IOERROR(RD) FLOW(3)

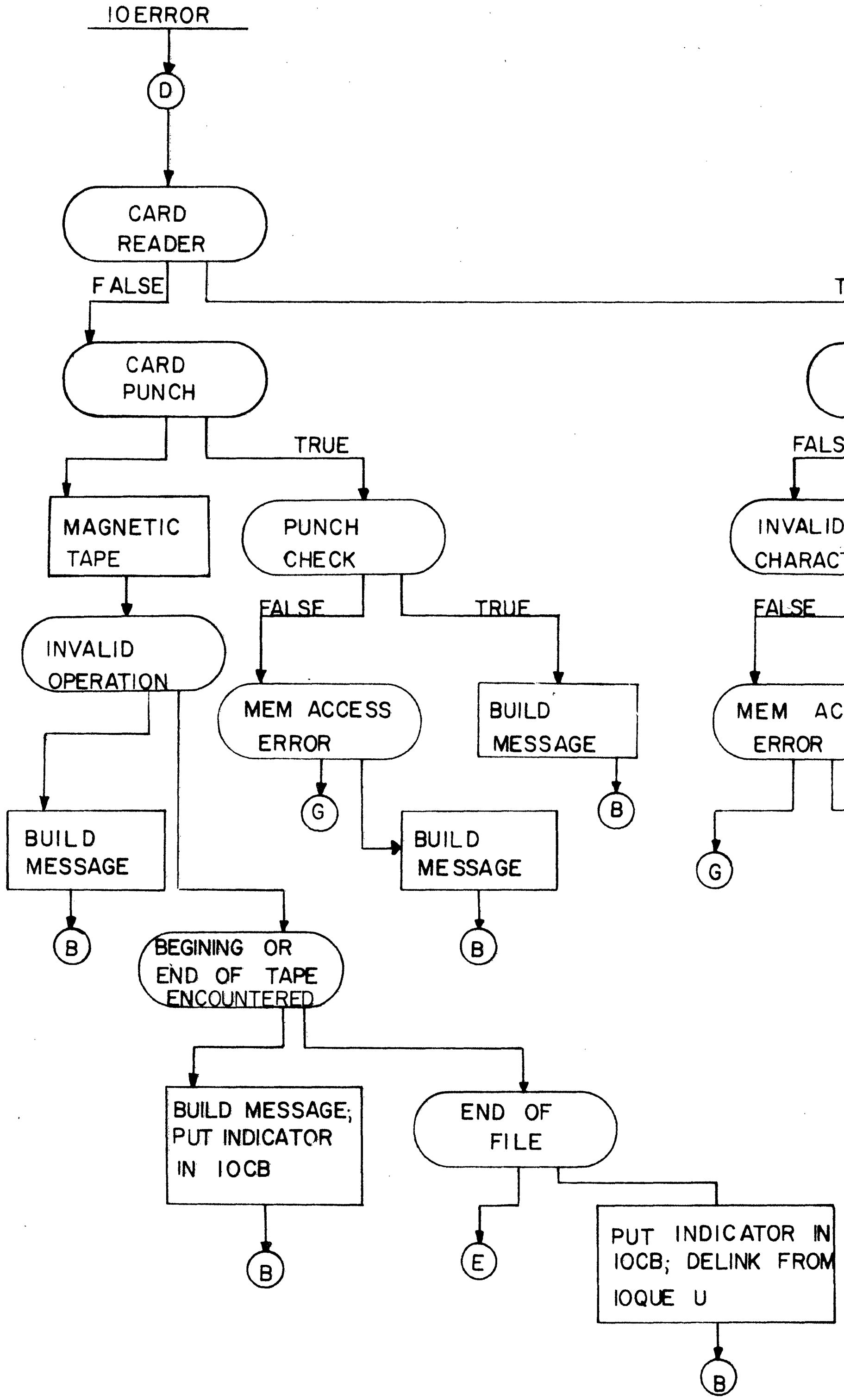


FIGURE F5-19. IOERROR(RD) FLOW (4)

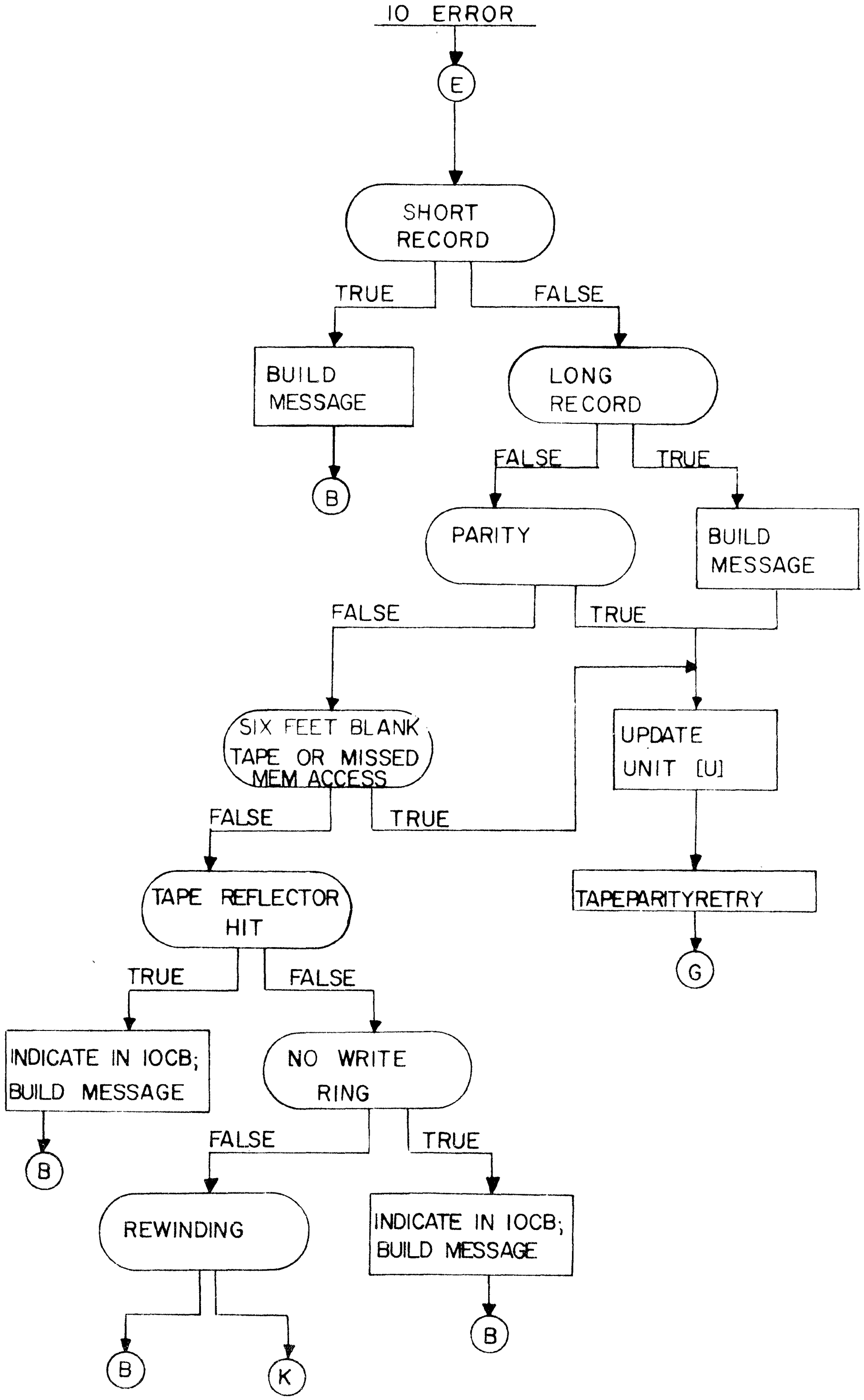


FIGURE F5-20. IOERROR(RD) FLOW (5)

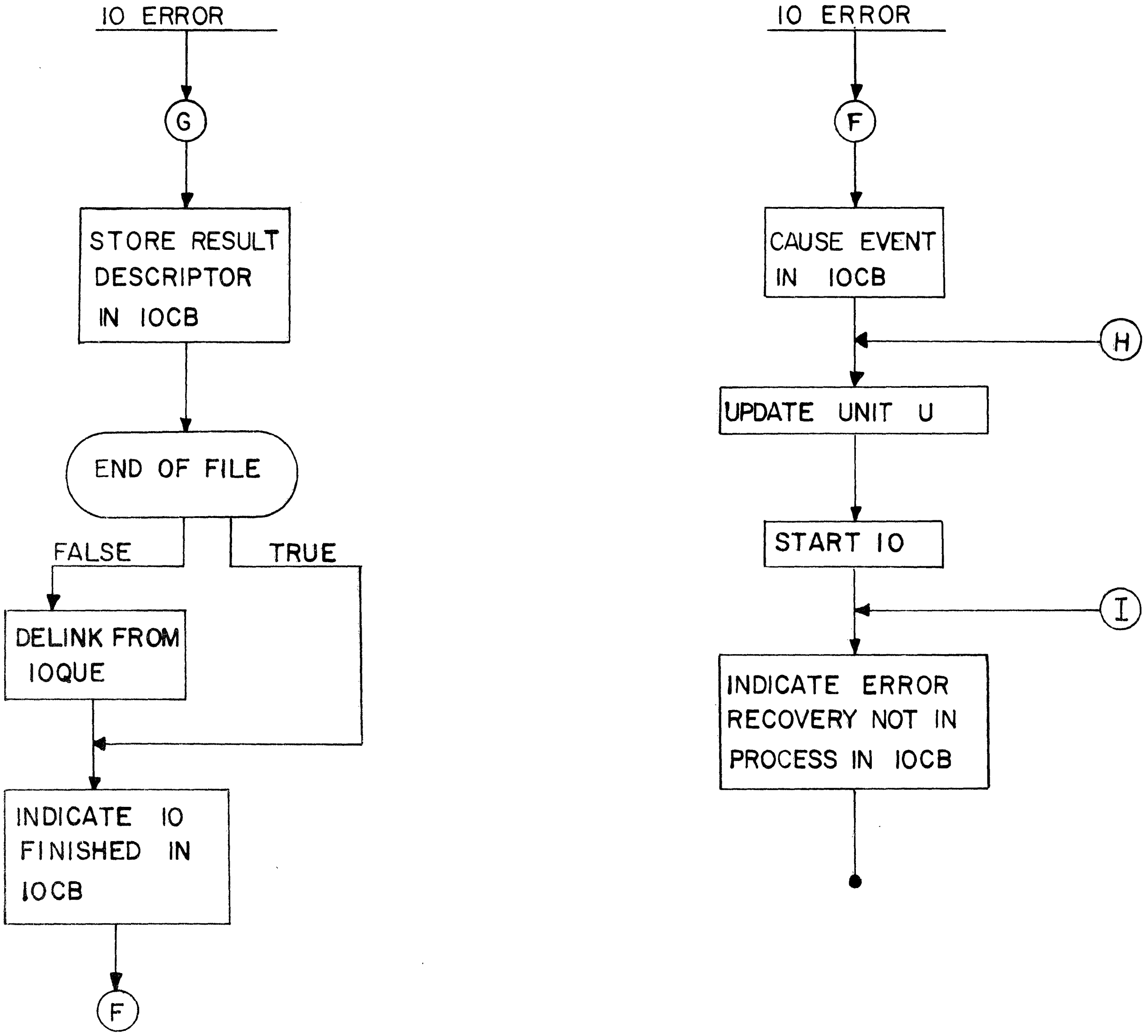

IO ERROR
G
STORE RESULT DESCRIPTOR IN IOCB
END OF FILE
FALSE
TRUE
DELINK FROM IOQUE
INDICATE IO FINISHED IN IOCB
F
IO ERROR
F
CAUSE EVENT IN IOCB
H
UPDATE UNIT U
START IO
I
INDICATE ERROR RECOVERY NOT IN PROCESS IN IOCB

FIGURE F5-20 CONTINUED

SECTION 6

UTILITY FUNCTIONS

6. UTILITY FUNCTIONS

A NUMBER OF UTILITY FUNCTIONS ARE PROVIDED AS PART OF THE B6700 SYSTEM SOFTWARE FOR THE CONVENIENCE OF THE USER. THESE FUNCTIONS ARE INCORPORATED IN THE MCP IN DIFFERING DEGREES. THEY ALLOW THE USER TO USE RELATIVELY SLOW PERIPHERAL DEVICES AT THEIR MAXIMUM RATED SPEEDS (LOAD CONTROL AND PRINTER AND PUNCH BACKUP), TRANSFER FILES TO AND FROM LIBRARY TAPES AND DISK (LIBRARY MAINTENANCE), DISPLAY AND MAINTAIN A HISTORY OF JOBS RUN ON THE SYSTEM (SYSTEM LOGS), INTERROGATE THE CONTENTS OF THE DISK DIRECTORY TO FIND WHICH FILES ARE CURRENTLY STORED ON DISK (LIST DIRECTORY), USE COMMON INTRINSIC FUNCTIONS IN SOURCE LANGUAGE PROGRAMS (INTRINSICS), AND LIST OR DUPLICATE CARD DECKS (CARD/ LINE). EACH OF THE FUNCTIONS IS DISCUSSED BELOW.

6.1. LOAD CONTROL

THE MCP LOAD CONTROL FACILITY PROVIDES A MEANS WHEREBY CARD DECK INFORMATION, INCLUDING SYSTEM CONTROL CARD INFORMATION, CAN BE PLACED ON THE DISK IN THE FORM OF A PSEUDO CARD FILE AND THEN USED AS THOUGH IT WERE BEING READ DIRECTLY FROM A CARD READER.

6.1.1. INTRODUCTION

WHEN OPERATING A SYSTEM IN A MULTIPROGRAMMING ENVIRONMENT, THERE ARE FREQUENTLY A LARGE NUMBER OF I/O OPERATIONS OCCURRING AT THE SAME TIME. THUS THE TOTAL THROUGHPUT OF THE SYSTEM IS OFTEN LIMITED BY ITS ABILITY TO PASS INFORMATION THROUGH ITS SLOWER PERIPHERAL DEVICES SUCH AS CARD READERS, CARD PUNCHES, AND LINE PRINTERS. THE LOAD CONTROL FACILITY IS PROVIDED IN ORDER TO MINIMIZE THIS LIMITATION WITH RESPECT TO CARD READERS BY KEEPING ALL CARD READERS OPERATING AT THEIR MAXIMUM SPEED, AND BY SIMULATING THE EXISTENCE OF ADDITIONAL CARD READERS.

THIS IS ACCOMPLISHED THROUGH THE USE OF PSEUDO CARD READERS, PSEUDO CARD DECKS, AND THE MCP PROCESS SYSTEM/LOADCONTROL.

6.1.2. SYSTEM/LOADCONTROL

THE PRIMARY FUNCTION OF SYSTEM/LOADCONTROL IS TO READ A CARD DECK FILE, CALLED A CONTROL DECK, AND WRITE IT ONTO DISK AS ONE OR MORE "PSEUDO CARD DECKS" SO THAT THESE "PSEUDO DECKS" MAY SUBSEQUENTLY BE AVAILABLE TO THE SYSTEM AS INPUT FROM "PSEUDO CARD READERS". THE SECONDARY FUNCTION OF SYSTEM/LOADCONTROL IS TO COPY THE CONTROLDECK FILE ONTO A CONTROLDECK TAPE.

THE INPUT CONTROLDECK FILE IS EITHER A CARD DECK OR A CARD IMAGE TAPE FILE. IF THE INPUT FILE IS A TAPE FILE IT MUST BE FORMATTED IN 15 WORD RECORDS, BLOCKED 14 RECORDS PER BLOCK. THE LAST RECORD ON THE CONTROLDECK FILE MUST BE A CONTROL CARD CONTAINING

"INVALID CHARACTER END CONTROL".

THE SYSTEM/LOADCONTROL PROCESS MAY BE CALLED OUT EITHER BY A KEYBOARD INPUT MESSAGE OR CONTROL CARDS.

IF SYSTEM/LOADCONTROL PROCESS MAY BE CALLED OUT TO PLACE A CONTROL DECK ON THE DISK. EITHER THE KEYBOARD INPUT MESSAGE "LDDK" OR A CONTROL CARD CONTAINING "INVALID CHARACTER EXECUTE SYSTEM/LOADCONTROL" MAY BE USED.

SYSTEM/LOADCONTROL TERMINATES AFTER IT HAS FINISHED PROCESSING THE "? END CONTROL" CARD.

6.1.3. PSEUDO CARD DECKS ON DISK

WHEN THE SYSTEM/LOADCONTROL PROGRAM READS A CONTROLDECK FILE, IT NORMALLY PLACES IT ON DISK AS ONE OR MORE PSEUDO CARD DECKS. THE NUMBER OF PSEUDO DECKS CREATED DEPENDS UPON THE NUMBER OF END CARDS LOCATED WITHIN THE CONTROLDECK. THAT IS, EACH TIME AN END CARD IS ENCOUNTERED, IT IS TAKEN TO DENOTE THE END OF A DECK; CREATION OF ANOTHER PSEUDO DECK IS THEN INITIATED AND AS EACH NEW PSEUDO DECK IS CREATED, IT IS GIVEN AN IDENTIFICATION OF THE FORM #<INTEGER>.

IT SHOULD BE NOTED THAT WHAT IS REFERRED TO AS A PSEUDO DECK IS ANALOGOUS TO A SINGLE CONTINUOUS DECK THAT WOULD BE PLACED IN A CARD READER. THEREFORE, IF A PSEUDO DECK CONTAINS MORE THAN ONE FILE, EACH FILE FOLLOWING THE FIRST WILL BE RECOGNIZED ONLY WHEN THE FILE PRECEDING IT HAS BEEN PASSED. ALSO NOTE THAT THERE IS NO SET LIMIT TO THE NUMBER OF CARDS THAT MAY BE CONTAINED IN A CONTROL DECK FILE.

6.1.4. ERROR CHECKING IN SYSTEM/LOADCONTROL

IF A PARITY ERROR IS ENCOUNTERED IN A CONTROL DECK FILE BEING READ FROM MAGNETIC TAPE, THE REMAINDER OF THAT PSEUDO DECK IS SKIPPED AND THE PSEUDO DECK CONTAINING THE PARITY IS COMPLETELY IGNORED.

6.1.5. PSEUDO CARD READERS

THE MCP WILL ACCEPT CONTROL CARDS AND DATA CARDS FROM A DISK FILE WHICH HAS BEEN ASSIGNED TO A PSEUDO CARD READER JUST AS IT WOULD FROM A CARD READER. AN MCP PROCESS, SYSTEM/LOADCONTROL, IS USED TO LOAD THE SYSTEM CONTROL DECKS TO DISK. THE MCP THEN ASSIGNS THESE CONTROLDECKS TO PSEUDO CARD READERS AS THE PSEUDO READERS BECOME AVAILABLE.

TO MAKE USE OF PSEUDO CARD DECKS, THE MCP CONTAINS LOGIC WHICH CAN, IN EFFECT, SUPPLY THE SYSTEM WITH A NUMBER OF PSEUDO CARD READERS. THESE PSEUDO CARD READERS, IN MANY WAYS, APPEAR TO BE MUCH LIKE PHYSICAL PERIPHERAL UNITS. THAT IS, SYSTEM MESSAGES ARE TYPED FOR THE PSEUDO CARD READERS AS THOUGH THEY WERE CARD READERS, AND KEYBOARD INPUT MESSAGES CAN REFERENCE THE PSEUDO CARD READERS. THE PSEUDO CARD READERS ARE IDENTIFIED BY SPECIFYING "CD UNIT NUMBER" AS SHOWN BELOW:

```
CD1
CD2
CD3
...
...
...
```

CD NN

AT HALT-LOAD TIME, ALL PSEUDO CARD READERS ARE TURNED OFF. THE SYSTEM OPERATOR MAY CAUSE THESE READERS TO BE TURNED ON THROUGH THE USE OF AN "RN" KEYBOARD INPUT MESSAGE. WHEN AN "RN <DIGIT>" MESSAGE IS INITIALLY ENTERED AND THE <DIGIT> IS NOT EQUAL TO ZERO, THE MCP AUTOMATICALLY SEARCHES FOR PSEUDO CARD DECKS TO SATISFY THE NEED OF THE SPECIFIED NUMBER OF PSEUDO CARD READERS. THEREAFTER, AS LONG AS PSEUDO CARD READERS ARE ON, AND PSEUDO CARD DECKS ARE AVAILABLE, THE MCP WILL KEEP THE PSEUDO READERS LOADED.

IF THE SYSTEM OPERATOR WISHES TO TURN OFF PSEUDO CARD READERS, HE NEED ONLY TYPE IN AN "RN" MESSAGE THAT SPECIFIES THE NUMBER OF PSEUDO CARD READERS HE WANTS LEFT ON. THE MCP WILL THEN TURN OFF A SUFFICIENT NUMBER OF READERS TO MEET THESE REQUIREMENTS AS SOON AS THE READERS COMPLETE PROCESSING THEIR CURRENT DECK.

IF, FOR ANY REASON, IT IS DESIRED TO REMOVE A DECK FROM A PSEUDO CARD READER (E.G., A CARD FILE NEVER OPENED BY A PROGRAM THAT WAS DISCONTINUED) THE REMOVAL CAN BE ACCOMPLISHED BY ENTERING AN "ED" (ELIMINATE DECK) KEYBOARD INPUT MESSAGE. THE SYNTAX OF THE "ED" MESSAGE IS TO BE SPECIFIED.

6.1.6. ERROR HANDLING IN THE PSEUDO CARD DECK

WHILE A PSEUDO CARD DECK IS BEING READ, AN ERROR IS DETECTED IN A CONTROL CARD OR PROGRAM-PARAMETER CARD, THE MCP WILL REMOVE THE DECK IN WHICH THE ERRONEOUS CARD APPEARS AND WILL CONTINUE TO THE NEXT AVAILABLE PSEUDO DECK.

6.2. PRINTER AND PUNCH BACKUP

6.2.1. INTRODUCTION

TO KEEP SYSTEM THROUGHPUT AT A MAXIMUM, IT IS DESIRABLE THAT SLOWER PERIPHERALS SUCH AS LINE PRINTERS AND CARD PUNCHES BE KEPT WORKING AT THEIR MAXIMUM RATED PERFORMANCE WITH A MINIMUM OF OVERHEAD TO THE SYSTEM. THIS IS ACCOMPLISHED ON THE B6700 BY SIMULATING PRINTERS AND PUNCHES WITH DISK FILES OR MAGNETIC TAPE UNITS. THUS, INFORMATION WHICH IS INTENDED TO BE OUTPUT TO A PRINTER OR PUNCH MAY BE ROUTED TO A DISK OR TAPE BACKUP FILE. WHEN THE BACKUP FILE IS CLOSED AND THE APPROPRIATE OUTPUT DEVICE IS AVAILABLE THE BACKUP FILE CAN THEN BE PRINTED OR PUNCHED.

A BACKUP-FILE ON TAPE IS LABELED "BACKUPT"/NNNN. BACKUP TAPES MAY BE WRITTEN AS MULTI-REEL FILES AND MULTI-FILE REELS. BACKUP FILES ON DISK ARE NAMED "BD"/NNNNNN, WHERE NNNNNN IS THE SYSTEM ASSIGNED FILE NUMBER.

BACKUP FILES ARE VARIABLE LENGTH RECORD, FIXED BLOCK FILES. EACH RECORD ON A BACKUP FILE IS COMPOSED OF TWO CONTROL WORDS FOLLOWED BY N WORDS OF DATA, N GREATER THAN OR EQUAL TO ZERO.

THE FIRST WORD OF CONTROL INFORMATION IS A COPY OF THE IOCW (I/O CONTROL WORD) WHICH WOULD HAVE BEEN USED HAD THE RECORD BEEN WRITTEN DIRECTLY TO PRINTER/PUNCH RATHER THAN TO BACKUP. THIS CONTROL WORD IS FORMATTED AS A STANDARD IOCW AND IS USED IN DETERMINING THINGS SUCH AS THE INTERNAL MODE OF THE DATA TO BE OUTPUT.

THE SECOND CONTROL WORD IS A UNIT FEATURE WORD USED PRIMARILY FOR CARRIAGE CONTROL. THE LENGTH FIELD (BITS 39:20) OF THE UNIT FEATURE WORDS CONTAINS THE LENGTH OF THE RECORD IN WORDS (NUMBER OF WORDS OF DATA PLUS 2).

THE MCP WILL BUILD A BACKUP FILE WHEN SPECIFIED BY MCP SYSTEM OPTIONS, PROGRAM FILE ATTRIBUTES, OR THE SYSTEM OPERATOR. BACKUP FILES ARE

PRINTED/PUNCHED BY THE SYSTEM/BACKUP PROGRAM EITHER UPON COMMAND BY THE SYSTEM OPERATOR OR, IF THE AUTOPRINT OPTION IS SET, AUTOMATICALLY WHEN THE APPROPRIATE DEVICE BECOMES AVAILABLE.

THE SELECTION OF A PHYSICAL UNIT FOR A BACKUP FILE IS DETERMINED AS FOLLOWS. IF THE FILE MAY GO TO TAPE, AN EXISTING PRINTER BACKUP TAPE (PBT) FILE IS USED, IF ONE IS AVAILABLE. OTHERWISE, IF A SCRATCH TAPE IS AVAILABLE, A NEW PBT IS CREATED AND USED.

IF A UNIT IS NOT FOUND FOR THE FILE, A MESSAGE IS DISPLAYED TO INFORM THE OPERATOR. IF A UNIT OF THE SPECIFIED TYPE IS MADE AVAILABLE, IT IS USED. OTHERWISE THE OPERATOR MAY REPLY WITH AN "OU" MESSAGE TO ASSIGN A DIFFERENT TYPE OF OUTPUT UNIT, SUCH AS A PB DISK FILE, A PRINTER, OR A CARD PUNCH.

6.2.2. SPECIAL FORMS

IF THE "SPECIAL FORMS" FEATURE IS DESIRED ON A PRINT FILE OPENED AS A PRINTER BACKUP FILE, ANY SPECIAL FORMS REQUIREMENT IS DEFERRED UNTIL THE BACKUP FILE IS PRINTED. IF THE PRINT FILE IS OPENED ON A PRINTER, THE FOLLOWING OPERATIONS ARE PERFORMED:

1. THE OPERATOR IS INFORMED THAT SPECIAL FORMS ARE REQUIRED BY THE MESSAGE

   "# <UNIT> FM RQD---"

   OR BY A SPECIAL PROGRAM-GENERATED MESSAGE.

2. THE OPERATOR MAY THEN

   A. LOAD THE FORMS ONTO THAT UNIT AND KEY IN THE MESSAGE

      "<MIX> FM <UNIT>"

   OR

B. KEY IN THE MESSAGE

"<MIX> OUMT"

OR THE MESSAGE

"<MIX> OUDK"

TO FORCE THE CHOSEN PRINTER TO BE RELEASED TO OPEN A BACKUP FILE.

WHEN A BACKUP FILE IS PRINTED WHICH REQUIRED SPECIAL FORMS THE FOLLOWING MESSAGE WILL BE TYPED:

"#FM RQD <UNIT> FOR <MFID>/<FID> OF <PROGRAM NAME>"

THE OPERATOR MAY RESPOND TO THIS MESSAGE WITH AN "OK", "WY", OR "DS" MESSAGE.

6.2.3. CLOSING A BACKUP FILE ON DISK

IF THE SYSTEM OPTION AUTO-PRINT IS SET WHEN A BACKUP FILE ON DISK IS CLOSED, IT IS SCHEDULED TO BE PRINTED. IF AUTO-PRINT IS NOT SET, A MESSAGE IS TYPED TO INFORM THE OPERATOR THAT A BACKUP DISK <BD> FILE EXISTS AND MAY BE OUTPUT BY THE MESSAGE "PBD XXXX". WHEN AN OUTPUT FILE IS PRINTED FROM A BD FILE AN ENTRY IS MADE INTO THE LOG CONTAINING THE HEADER CARD INFORMATION OF THE PROGRAM.

6.2.4. PRINTER BACKUP SPO INPUT MESSAGES

SYNTAX:

```
<PB MESSAGE> ::= ?PB <INPUT DESIGNATOR> <OPTIONAL OUTPUT DESIGNATOR>
     <KEY DECLARATION> <PB OPTION PART>
     <INPUT DESIGNATOR>::= MT <INTEGER>/D<INTEGER>
     <OPTIONAL OUTPUT DESIGNATOR> ::= <EMPTY>/LP<INTEGER>/CP<INTEGER>
     <PB OPTION PART::=<GENERAL OPTIONS>/<PB OPTION PART>,<GENERAL
```

```
          OPTIONS>/
          <PB OPTION PART> <ASSOCIATIVE OPTIONS>
     <GENERAL OPTIONS> ::= <EMPTY>/SAVE/COPIES=<INTEGER>
     <ASSOCIATIVE OPTIONS>::=<EMPTY>/RANGE<LITERAL><SPACER><LITERAL>/
          EQUAL <LITERAL>/<RECORD NUMBER OPTION>
     <RECORD NUMBER OPTION> ::= RECORD>INTEGER>/RECORD>INTEGER><SPACER>
          <INTEGER>
     <SPACER> ::= <BLANK>/<SPACE> <BLANK>

<QT MESSAGE> ::= <MIX INDEX> QT <ASSOCIATIVE OPTIONS>

<CO MESSAGE> ::= <MIX INDEX> CO <ASSOCIATIVE OPTIONS>

<OU MESSAGE> ::= <MIX INDEX> OU <FILE OUTPUT DESIGNATOR> /
          <KEY DECLARATION>
     <FILE OUTPUT DESIGNATOR> ::= ≤TO BE SPECIFIED ELSEWHERE≥
     <KEY DECLARATION> ::= <EMPTY>/KEY
     <KEY SPECIFIER> ::= ALGOL/COBOL/FORTRAN/
          <KEY START COLUMN> <SPACER> <KEY LENGTH>
          <KEY START COLUMN> ::= <INTEGER>
          <KEY LENGTH> ::= <INTEGER>
```

EXAMPLES:

```
     ? PB MT6
     ? PB D2 LP 2
     ? PB MT1 SAVE COPIES = 10 RECORD 1000 2000
     ? PB D25 CP1 SAVE RANGE "1Q765AB" "GG654"
     ? PB MT5 EQUAL "ERROR"

     1 QT
     1 QT RANGE 500 999999

     3CO EQUAL 10342700
     2CO RECORD 1000

     1 OU MT1
     1 OU DK KEY ALGOL
```

```
1 OU MT5 KEY 2 6
```

SEMANTICS:

1. THE <PB MESSAGE> IS USED TO INITIATE THE PROCESS OF OUTPUTTING A BACKUP FILE TO THE LINE PRINTER OR CARD PUNCH.

   THE <INPUT DESIGNATOR> SPECIFIES THE TAPE UNIT OR DISK FILE ON WHICH THE BACKUP FILE IS STORED.

   THE <OPTIONAL OUTPUT DESIGNATOR> MAY BE USED TO SPECIFY THE PRINTING OR PUNCHING OF A BACKUP FILE ON A SPECIFIC PRINTER OR PUNCH UNIT.

   THE "SAVE" OPTION ALLOWS THE USER TO SPECIFY THAT HIS BACKUP TAPE OR DISK FILE IS TO BE SAVED INSTEAD OF PURGED ONCE IT HAS BEEN OUTPUT.

   THE "COPIES" OPTION ALLOWS THE USER TO SPECIFY HOW MANY COPIES OF THE BACKUP FILE HE WANTS TO HAVE PRINTED OR PUNCHED.

   THE <ASSOCIATIVE OPTIONS> ENABLES THE USER TO SELECTIVELY OUTPUT A BACKUP FILE BY RECORD NUMBER OR BY A KEY. THE KEY MAY BE SPECIFIED AT THE TIME THE BACKUP FILE IS CREATED OR WHEN THE PB MESSAGE IS INPUT, AND IS EITHER A STANDARD COMPILER SEQUENCE NUMBER FIELD OR ANY SINGLE SET OF CONTIGUOUS CHARACTERS WITHIN THE BACKUP RECORD.

   THE "RANGE <LITERAL> <LITERAL>" OPTION WILL CAUSE THE KEY FIELD OF EACH RECORD ON THE BACKUP FILE TO BE COMPARED TO THE TWO LITERALS SPECIFIED IN THE RANGE INPUT MESSAGE. IF THE KEY LIES WITHIN THE RANGE SPECIFIED, I.E., FIRST LITERAL ≤ KEY ≤ SECOND LITERAL, THE RECORD WILL BE OUTPUT. OTHERWISE, THAT RECORD WILL BE SKIPPED AND THE NEXT RECORD ON THE OUTPUT FILE WILL BE TESTED.

   THE "EQUAL <LITERAL>" OPTION WILL CAUSE THE KEY FIELD OF EACH RECORD ON THE BACKUP FILE TO COMPARE WITH THE LITERAL SPECIFIED IN

THE EQUAL OPTION. IF THE COMPARISON IS EQUAL THE RECORD IS OUTPUT; OTHERWISE, IT IS SKIPPED.

THE <RECORD NUMBER OPTION> ENABLES THE USER TO SELECTIVELY OUTPUT THE BACKUP FILE BY DESIGNATING A SPECIFIC RECORD NUMBER OR RANGE OF RECORD NUMBERS. THE INTEGERS SPECIFIED IN THE <RECORD NUMBER OPTION> CORRESPOND TO THE LOCATIONS OF SPECIFIC RECORDS WITHIN THE FILE. FOR EXAMPLE, RECORD I WOULD CAUSE THE ITH RECORD ON THE BACKUP FILE TO BE OUTPUT; RECORD I - J WOULD CAUSE ALL RECORDS FROM THE ITH RECORD THROUGH THE JTH RECORD INCLUSIVE TO BE OUTPUT.

2. THE <QT MESSAGE> IS USED TO TERMINATE A CURRENT PROCESS OF OUTPUTTING A BACKUP FILE. THE QT MESSAGE MAY BE USED WITHOUT SPECIFYING ASSOCIATIVE OPTIONS.

   IF THE QT MESSAGE IS USED WITHOUT SPECIFYING ASSOCIATIVE OPTIONS, THEN THE REMAINDER OF THE BACKUP FILE CURRENTLY BEING OUTPUT IS SKIPPED. IF THE FILE WHICH WAS AFFECTED BY THE QT MESSAGE WAS CONTAINED ON A MULTI-FILE BACKUP TAPE, THEN THE NEXT BACKUP FILE ON TAPE IS OUTPUT UNDER CONTROL OF THE SAME GENERAL OPTIONS AS WERE SPECIFIED FOR THE PREVIOUS FILE. THE ASSOCIATE OPTIONS ARE RESET.

   IF THE FILE WHICH WAS AFFECTED BY THE QT MESSAGE WAS A DISK BACKUP FILE OR THE LAST OR ONLY FILE ON A BACKUP TAPE, THEN THE OUTPUT PROCESS IS TERMINATED.

   IF THE QT MESSAGE IS USED WITH ASSOCIATIVE OPTIONS, THEN THE FILE CURRENTLY BEING OUTPUT IS SKIPPED AND THE NEXT FILE ON A MULTI-FILE BACKUP TYPE IS OUTPUT UNDER CONTROL OF THE ASSOCIATIVE OPTIONS SPECIFIED IN THE QT MESSAGE.

3. THE <CO MESSAGE> IS USED TO CHANGE THE ASSOCIATIVE OPTIONS WHICH CONTROL THE OUTPUT OF A BACKUP FILE CURRENTLY IN PROCESS. THE PROCESS OF OUTPUTTING THE BACKUP FILE IS ALTERED SO THAT THE REMAINDER OF THE BACKUP FILE IS OUTPUT UNDER CONTROL OF THE ASSOCIATIVE OPTIONS SPECIFIED IN THE CO MESSAGE. IF NO ASSOCIATIVE

OPTIONS ARE SPECIFIED ALL OPTIONS ARE RESET AND EVERY RECORD IS OUTPUT.

4. THE <OU BACKUP> FORM OF THE OU MESSAGE IS USED TO SPECIFY THAT A LINE PRINTER OR CARD PUNCH OUTPUT FILE IS TO BE WRITTEN TO A BACKUP FILE.

THE BACKUP <FILE OUTPUT DESIGNATOR> IS USED TO SPECIFY THAT THE BACKUP FILE BE WRITTEN TO A SPECIFIC TAPE UNIT OR TO DISK.

THE <KEY DECLARATION> SPECIFIES A SET OF CONTIGUOUS CHARACTERS IN EACH OUTPUT RECORD WHICH MAY BE USED AS THE KEY FILED FOR SUBSEQUENTLY OUTPUTTING THE FILE UNDER CONTROL OF ASSOCIATIVE OPTIONS.

THE <KEY SPECIFIER> IS EITHER A STANDARD COMPILER NAME OR THE STARTING LOCATION AND LENGTH OF THE KEY FIELD IN THE OUTPUT RECORD. IF A STANDARD COMPILER NAME IS SPECIFIED, THEN THE SEQUENCE NUMBER FIELD FOR THAT PARTICULAR COMPILER LANGUAGE IS USED AS THE KEY FIELD.

6.3. LIBRARY MAINTENANCE

THE MCP PROVIDES A LIBRARY MAINTENANCE PROCESS TO PERFORM LIBRARY UTILITY OPERATIONS. THIS PROCESS IS INITIATED EITHER FROM THE SUPERVISORY DISPLAY UNIT OR FROM A SYSTEM CONTROL CARD. OPTIONS PROVIDED INCLUDE COPY STATEMENTS AND MOVE STATEMENTS.

6.3.1. COPY AND MOVE STATEMENTS

THE COPY AND MOVE STATEMENTS ALLOW THE USER TO TRANSFER FILES TO AND FROM LIBRARY TAPES AND DISK STORAGE. THE COPY STATEMENT MAKES A COPY OF THE SPECIFIED FILE; THE MOVE STATEMENT MAKES A COPY OF THE SPECIFIED FILE AND (IF THE ORIGINAL IS A DISK FILE) DESTROYS THE ORIGINAL FILE. IF THE SOURCE FILE IS A TAPE FILE COPY AND MOVE OPERATIONS ARE IDENTICAL, THAT IS, THE SOURCE FILE IS COPIED BUT NOT DESTROYED.

SYNTAX:

```
<COPY AND MOVE STATEMENT> ::= <INVALID CHARACTER> <STATEMENT VERB>
     <STATEMENT LIST>; END
<STATEMENT VERB> ::= COPY / MOVE
<STATEMENT LIST> ::= <STATEMENT PART> / <SOURCE LIST> /
     <STATEMENT PART>, <STATEMENT LIST>
<STATEMENT PART> ::= <SOURCE LIST> <DESTINATION LIST> /
     <SOURCE LIST>, <FILE LIST> <DESTINATION LIST>
<SOURCE LIST> ::= <FILE LIST> FROM <SOURCE> /
     <SOURCE LIST>, <FILE LIST> FROM <SOURCE>
<FILE LIST> ::= <FILE NAME> / <FILE LIST>, <FILE NAME>
<DESTINATION LIST> ::= TO <DESTINATION> /
     <DESTINATION LIST>, <DESTINATION>
<FILE NAME> ::= <IDENTIFIER> / <FILE NAME> <SLASH> <IDENTIFIER>
<SOURCE> ::= <DESTINATION> ::= DISK/<IDENTIFIER>
```

SEMANTICS:

1. <SLASH> STANDS FOR THE SYMBOL "/".

2. <IDENTIFIER> IS UNDERSTOOD TO CONSIST OF AT MOST 17 CHARACTERS.

3. <FILENAME> CONSISTS OF AT MOST 14 <IDENTIFIER>S.

4. <DESTINATION LIST> MAY CONTAIN AT MOST 46 <DESTINATION>S.

5. AT MOST 1000 FILES WILL BE COPIED TO ANY LIBRARY TAPE.

6. <INVALID CHARACTER> IS A "?" FOR SPO INPUT; FOR CONTROL CARD INPUT IT IS AN INVALID PUNCHED CHARACTER.

THE SYNTAX IS DESIGNED TO ALLOW A DEFAULT SPECIFICATION OF DISK AS THE <SOURCE> OR <DESTINATION> WHEREVER THIS IS POSSIBLE.

EXAMPLES:

```
  1.              ? COPY A,B,C TO SYS;END
```

IS THE SAME AS

```
                  ? COPY A,B,C FROM DISK TO SYS;END
```

```
  2.              ? COPY A,B,C FROM SYS;END
```

IS THE SAME AS

```
                  ? COPY A,B,C FROM DISK TO SYS;END
```

```
  3.              ? COPY A,B,C FROM SYS TO SYS, TO SYS, TO SYS2;END
```

WILL CREATE THREE LIBRARY TAPES CALLED SYS,SYS,AND SYS2 EACH HAVING THE FILES A,B,AND C ON IT. THIS IS DONE SIMULTANEOUSLY, I.E., EACH RECORD IS COPIED TO EACH DESTINATION BEFORE GOING TO THE NEXT RECORD.

4.

```
? COPY A,B, FROM SYS,D/F,G FROM TP TO NEW;END
```

WILL CREATE A LIBRARY TAPE CALLED NEW HAVING THE FILE A,B,D/F,AND G ON IT. THIS IS NOT DONE VIA DISK.

5.

```
? COPY A,B,C FROM SYS TO DISK,E/F,G TO SYS2;END
```

THE FILES E/F AND G IN THE SECOND PART OF THIS STATEMENT WILL BE FOUND ON DISK. THESE EXAMPLES DO NOT EXHAUST ALL POSSIBLITIES.

6.4. SYSTEM LOGS

THE MCP MAINTAINS A HISTORY OF USER PROGRAM AND MCP ACTIVITY; THIS INFORMATION IS CONTAINED IN THREE SEPARATE LOGS: THE SYSTEM LOG, THE MAINTENANCE LOG, AND THE DATA COMM LOG. THE SYSTEM LOG CONTAINS INFORMATION REGARDING THE HISTORY OF INDIVIDUAL JOBS THAT ARE RUN ON THE SYSTEM, AS WELL AS INFORMATION REGARDING THE OPERATION OF THE SYSTEM ITSELF. THE MAINTENANCE LOG CONTAINS INFORMATION REGARDING ERRORS DETECTED DURING THE OPERATION OF THE SYSTEM, SUCH AS PARITY ERRORS, DESCRIPTOR ERRORS, ETC. THE CONTENTS OF THE DATA COMM LOG ARE TO BE DEFINED.

6.4.1. THE SYSTEM LOG

THE SYSTEM LOG CONTAINS INFORMATION REGARDING NORMAL OPERATION OF THE SYSTEM. INFORMATION REGARDING ERRORS (OTHER THAN SECURITY ERRORS) IS STORED IN THE MAINTENANCE LOG. THE INFORMATION CONTAINED IN THE SYSTEM LOG CAN BE DIVIDED INTO THREE MAIN CATEGORIES: JOB-ORIENTED ENTRIES, PERIPHERAL-ORIENTED ENTRIES AND MISCELLANEOUS ENTRIES.

JOB-ORIENTED ENTRIES CONTAIN INFORMATION REGARDING THE HISTORY OF INDIVIDUAL JOBS THAT HAVE BEEN RUN ON THE SYSTEM. THIS INFORMATION INCLUDES SUCH THINGS AS CONTROL CARD INFORMATION, SCHEDULING INFORMATION, BOJ INFORMATION, EOJ INFORMATION, OPERATOR RESPONSES, PRIORITY CHANGES, AND ABORT INFORMATION.

PERIPHERAL-ORIENTED ENTRIES INCLUDE INFORMATION ON FILE OPENINGS AND CLOSINGS, AS WELL AS POINTERS TO I/O ERROR INFORMATION WHICH IS CONTAINED IN THE MAINTENANCE LOG.

MISCELLANEOUS ENTRIES CONTAIN INFORMATION REGARDING SYSTEM OPERATION, SUCH AS HALT/LOAD INFORMATION, TIME/DATE CHANGES, SYSTEM OVERHEAD INFORMATION, AND OPERATOR INPUT MESSAGES.

6.4.1.1. SYSTEM LOG ENTRIES

DETAILED INFORMATION REGARDING SYSTEM LOG ENTRIES IS PROVIDED IN APPENDIX B.

6.4.1.2. SYSTEM LOG RELEASE.

THE SYSTEM OPERATOR IS INFORMED OF THE SIZE OF THE SYSTEM LOG BY THE FOLLOWING MESSAGE:

"LOG <INTEGER>% FULL"

THIS MESSAGE IS FIRST GIVEN WHEN THE LOG BECOMES 5 PERCENT FULL, THEN IS GIVEN WITH EACH 5 PERCENT INCREASE IN LOG SIZE UNTIL THE LOG BECOMES 95 PERCENT FULL. WHEN THE SYSTEM LOG BECOMES 95 PERCENT FULL A NEW SYSTEM LOG IS AUTOMATICALLY INITIATED, AND THE OPERATOR IS INFORMED WITH THE FOLLOWING MESSAGE:

"LOG 95% FULL-AUTOMATIC LR"

THE OPERATOR IS ALSO ABLE TO CREATE AN EMPTY SYSTEM LOG BY MEANS OF THE "LR" MESSAGE. IN EITHER CASE, THE "LR" MESSAGE CAUSES THE CURRENT LOG TO BE NAMED

LOG / <LOG SERIAL NUMBER>

(WHERE LOG SERIAL NUMBER IS AN INTEGER BETWEEN 0 AND 999999), AND CAUSES ANOTHER SYSTEM LOG FILE TO BE CREATED. ALL ENSUING LOG ENTRIES WILL BE STORED IN THE NEW SYSTEM LOG. THE RENAMED LOG CAN BE REMOVED FROM THE SYSTEM WHEN DESIRED.

THE SYSTEM OPERATOR IS KEPT INFORMED OF THE SIZE OF THE SYSTEM LOG BY THE "LOG <INTEGER> % FULL" MESSAGE WHICH IS GIVEN AT EVERY 5 PERCENT. THE OPERATOR IS ABLE TO CREATE AN EMPTY SYSTEM LOG BY MEANS OF THE "LR" MESSAGE. WHEN "LR" IS RECEIVED BY THE SYSTEM, THE CURRENT LOG IS NAMED LOG/<TODAYSDATE>/<LOG SERIAL NO.> (WHERE <LOG SERIAL NO.> IS AN INTEGER (BETWEEN 0 AND 999999) AND ANOTHER SYSTEM LOG FILE IS CREATED.

ALL ENSUING LOG ENTRIES WILL BE STORED IN THE NEW SYSTEM LOG AND THE RENAMED LOG CAN BE REMOVED FROM THE SYSTEM IF DESIRED.

IF THE SYSTEM LOB BECOMES 95% FULL, AND "LR" IS AUTOMATICALLY INITIATED WITH THE FOLLOWING MESSAGE: "LOG 95% FULL-AUTOMATIC LR"

6.4.1.3. SYSTEM LOG RETRIEVAL

THREE BASIC OPTIONS ARE AVAILABLE FOR RETRIEVING INFORMATION FROM THE SYSTEM LOG, THEY ARE RETRIEVAL OF SPECIFIED ENTRY TYPES, RETRIEVAL OF JOBS, AND RETRIEVAL OF FILES. THE INFORMATION IS RETRIEVED BY USING THE SPO MESSAGE:

```
                         LOG <RANGE> <REQUEST LIST>
```

SYNTAX FOR SPO INPUT MESSAGES:

```
<RANGE>::= <TIME>/<DATE>/<DATE>TO<DATE>/<TIME><DATE><TO<TIME><DATE>
     "/" <LOGID> / <NULL>

<LOG ID> ::= <INTEGER>
<TIME>::= <MILITARY TIME NOTATION>
<DATE>::= <MONTH> "/" <DAY> "/" <YEAR>
<REQUEST LIST> ::= <REQUEST>/<REQUEST LIST>, <REQUEST>
<REQUEST> ::= <JOB TYPE LIST> / <MISC TYPE LIST> /
     <JOB TYPE LIST> JOB <ID> / FILE <ID> / <NULL>
<JOB TYPE LIST> ::= <JOB TYPE>/<JOB TYPE LIST> <JOB TYPE>
<JOB TYPE> ::= CC/SCHD/BOJ/PRIORITY/INPUT/RSVP/EOJ/ABORT/IO/<NULL>
<MISC TYPE LIST> ::= <MISC TYPE> / <MISC TYPE LIST> <MISC TYPE>
<MISC TYPE> ::= HL/TD/OVHD/SECURITY
<ID> ::= <JOB ID> / <FILE ID> / <MIX INDEX>
```

EXAMPLES:

```
LOG                 (RETRIEVES ALL ENTRIES.)
LOG JOB             (RETRIEVES ALL JOBS.)
LOG FILE          (RETRIEVES ALL FILES -- SAME AS LOG IO.)
LOG/5             (RETRIEVES ALL ENTRIES IN FILE LOG/000005.)
LOG/5 JOB A/B     (RETRIEVES ALL ENTRIES OF JOB A/B IN FILE LOG/000005.)
LOG 1100 JOB A/B (RETRIEVES ALL ENTRIES OF JOB A/B FROM 1100 HOURS TO
                  PRESENT.)
LOG 1200 5/11/70 TO 1700 5/11/70 JOB A/B
LOG FILE C/D
LOG IO JOB A/B
LOG OVHD SECURITY, CC JOB A/B, I/O JOB X/Y
LOG OVHD SECURITY, CCJOB A/B, OPEN ERROR CLOSE JOBX/Y
```

A NULL RANGE WILL INCLUDE ENTRIES OF THE MOST RECENT FOUR HOURS.

6.4.1.4. OPERATOR INPUT

THE KEYED INPUT MNEMONIC "LC" WILL CAUSE ANY FOLLOWING CHARACTERS (UP TO A MAXIMUM OF 80) TO BE ENTERED INTO THE SYSTEM LOG AS A COMMENT.

6.4.2. MAINTENANCE LOG.

THE MAINTENANCE LOG CONTAINS INFORMATION REGARDING ERRORS DETECTED DURING SYSTEM INFORMATION. THE ERROR TYPES REPORTED ARE DESCRIPTOR ERRORS, INVALID MEMORY ADDRESS ERRORS, I/O MEMORY PARITY ADDRESS ERRORS, MEMORY PROTECT ERRORS, PARITY ERRORS, AND WRITE LOCKOUT ERRORS. IN ADDITION TO THE ERROR TYPE, INFORMATION CONTAINED IN THE MAINTENANCE LOG INCLUDES SUCH ITEMS AS THE DATE, TIME, UNIT NUMBER AND TYPE, LOCATION OF RECORD, I/O CONTROL WORD AND SO FORTH. THE FORMAT OF THE MAINTENANCE LOG IS DESCRIBED IN DETAIL IN APPENDIX B.

6.5. LIST DIRECTORY

THE PROGRAM LIST/DIRECTORY CAN BE CALLED OUT EITHER BY USE OF THE "DIR" MESSAGE OR BY THE FOLLOWING RUN CARD:

" ?RUN LIST/DIRECTORY; END."

THIS PROGRAM WILL PRODUCE A LISTING OF ALL FILES WHICH ARE STORED ON DISK. THE PROGRAM HAS AN OPTION KNOWN AS THE "MAP" OPTION. IF LIST/DIRECTORY IS COMPILED WITH THIS OPTION SET, THEN IN ADDITION TO THE LIST OF FILES THE PROGRAM WILL ALSO PROVIDE THE DISK ADDRESS AND SIZES OF DISK AREAS IN USE BY EACH FILE, A LIST OF AREAS (SORTED BY ADDRESS) WHICH CAN BE MADE AVAILABLE BY THE REMOVAL OF ONE FILE, AND A MAPPING OF DISK CHECKERBOARDING.

LIST/DIRECTORY IS RELEASED WITH THE "MAP" OPTION SET, BUT CAN BE RECOMPILED WITH THE OPTION RESET IF DESIRED. NOTE: THE PROGRAM WILL RUN MORE RAPIDLY IF THE "MAP" OPTION IS RESET.

6.5.1. PARTIAL DIRECTORY INFORMATION.

THE "PD" (PRINT DIRECTORY) SPO MESSAGE ALLOWS THE USER TO SELECTIVELY DISPLAY ON THE SPO THE NAMES OF FILES CONTAINED IN THE DISK DIRECTORY. THIS MESSAGE HAS THE FOLLOWING FORM:

"PD <FILE SET SPECIFIER>"

WHERE <FILE SET SPECIFIER> ::= <FILE LABEL> / <FILE LABEL> <SLASH> = IF A FULL FILE NAME IS SPECIFIED WITHOUT AN EQUAL SIGN THEN THE DISK DIRECTORY WILL BE SEARCHED AND A MESSAGE DISPLAYED ON THE SPO STATING WHETHER OR NOT THE FILE IS PRESENT. IF A PARTIAL FILE NAME IS SPECIFIED WITH AN EQUAL SIGN, THEN A LIST OF ALL DISK FILES BEGINNING WITH THE PARTIAL FILE NAME WILL BE DISPLAYED. IF NO FILE IS FOUND THE MESSAGE "NULL" WILL BE DISPLAYED.

EXAMPLES:

```
PD  CARD/LINE
PD  SYSTEM/ =
```

6.6. INTRINSIC FUNCTIONS

IN A MULTIPROCESSING SYSTEM, THE INCLUSION OF COMMON INTRINSIC FUNCTIONS IN EACH PROGRAM CAUSES MULTIPLE COPIES OF THE FUNCTIONS TO BE PRESENT IN MAIN MEMORY. IN ORDER TO MAKE MORE EFFECTIVE USE OF MAIN MEMORY, THE LOCATION OF THE CODE FILE FOR THESE ROUTINES IS KNOWN TO THE OPERATING SYSTEM AND IS ACCESSIBLE TO ALL USER PROGRAMS AS INTRINSIC FUNCTIONS.

SINCE ALL PROGRAMS IN THE B6700 SYSTEM ARE WRITTEN IN HIGH-LEVEL LANGUAGES, THE USE OF INTRINSIC FUNCTIONS IS IMPLEMENTED BY THE COMPILERS. EACH COMPILER RECOGNIZES THE NAMES OF THOSE INTRINSICS THAT ARE ALLOWABLE IN EACH LANGUAGE. AN INTRINSIC NAME WHICH OCCURS IN A SOURCE LANGUAGE STATEMENT IS PROCESSED BY A COMPILER AS A PRE-COMPILED PROCEDURE. EACH COMPILER IS RESPONSIBLE FOR VERIFYING THAT ACTUAL PARAMETERS AGREE WITH THE FORMAL PARAMETERS SPECIFIED FOR EACH INTRINSIC.

FOR EACH INTRINSIC REQUIRED BY THE OBJECT PROGRAM THE COMPILER EMITS A "STUFFED" INDIRECT REFERENCE WORD (SIRW) WHICH POINTS TO THE APPROPRIATE PROGRAM CONTROL WORD (PCW) IN THE D[0] (MCP) STACK.

THE PCW FOR AN INTRINSIC CONTAINS A 14 BIT SEGMENT DESCRIPTOR INDEX (SDI) FIELD. WHICH REFERS TO THE D[0] (MCP) STACK. THE LOW-ORDER 13 BITS OF THE INDEX FIELD LOCATES THE SEGMENT DESCRIPTOR WITHIN THE D[D] STACK. THIS DESCRIPTOR CONTAINS THE MEMORY/DISK ADDRESS FOR THE REQUIRED INTRINSIC CODE. SINCE THE D[0] STACK IS GLOBAL TO THE TOTAL ADDRESSING ENVIRONMENT, ANY SEGMENT DESCRIPTOR IN THIS STACK IS ACCESSIBLE FROM ANY PROGRAM WHICH REFERENCES A PCW CONTAINING A SDI REFERENCING D[0]. BECAUSE THERE IS A SINGLE SEGMENT DESCRIPTOR IN THE D [0] STACK FOR EACH INTRINSIC, ONLY ONE COPY OF THE OBJECT CODE IS PRESENT IN MEMORY. THUS THE INTRINSICS ARE RE-ENTRANT.

6.7. CARD/LINE

CARD/LINE LISTS BCL AND EBCDIC DATA AND PUNCHES BCL, BCL, EBCDIC AND BINARY DATA, BY LABEL EQUATING THE OUTPUT FILE "LINE" TO THE CARD PUNCH. INPUT SHOULD BE OF THE FORM:

FIRST CARD:

```
                  ? RUN CARD/LINE
OR
                  ? EXECUTE CARD/LINE
```

SECOND CARD:

```
                  ? BCL CARD             (CARDS PUNCHED IN BCL)
OR
                  ? DATA CARD            (CARDS PUNCHED IN EBCDIC)
OR
                  ? BINARY CARD          (CARDS PUNCHED IN BINARY)

                  <CARDS TO BE LISTED>
```

LAST CARD:

```
                  ? END                  (BCL AND EBCDIC DATA)
OR
                  "BEND CARD"           (BINARY DATA)
```

THE LABEL EQUATION CARD TO PUNCH THE OUTPUT SHOULD BE THE SECOND CARD IN THE DECK AND HAVE THE FOLLOWING FORM:

```
   ? FILE LINE = LINE PUNCH
```

A "BEND CARD" (BINARY END CARD) HAS THE LETTERS "BEND" WRITTEN IN PUNCHSCRIPT ACROSS THE CARD.

APPENDIX A

OPERATOR-MCP COMMUNICATIONS

APPENDIX A - OPERATOR-MCP COMMUNICATIONS

COMMUNICATION WITH THE MCP IS ACCOMPLISHED WITH A COMBINATION OF DISPLAY UNITS (CRT DEVICES), CONTROL UNITS (DISPLAY UNITS WITH ASSOCIATED KEYBOARDS) AND CONTROL CARDS (SPECIAL RECORDS RECOGNIZED BY THE MCP). THE FOLLOWING DISCUSSION IS BASED ON A SYSTEM WITH ONE CONTROL UNIT AND ONE DISPLAY UNIT, ALTHOUGH A SYSTEM MAY HAVE ANY COMBINATION FROM A MINIMUM OF ONE DISPLAY UNIT TO A MAXIMUM OF THIRTY DISPLAY AND CONTROL UNITS COMBINED.

A-1. DISPLAY OF STATUS

THE STATUS OF THE SYSTEM AND OF THE PROCESSES IN PROGRESS IS PRESENTED ON THE DISPLAY UNITS. VARIOUS TABLES MAY BE CALLED FOR DISPLAY BY ENTERING THE APPROPRIATE KEYBOARD INPUT MESSAGES. IN ADDITION, SPECIFIC QUESTIONS REQUIRING SHORT ANSWERS MAY BE ENTERED FROM THE KEYBOARD. THESE QUESTIONS AND ANSWERS ARE DISPLAYED AS THEY OCCUR. THE DISPLAY TABLES ARE DESCRIBED BELOW.

MIX TABLE

THE MIX TABLE IS DISPLAYED CONTINUOUSLY EXCEPT FOR BRIEF PERIODS WHEN IT IS REPLACED BY ANOTHER TABLE. EACH JOB BEING EXECUTED HAS AN ENTRY, THE CONTENTS OF WHICH DEPEND ON WHETHER THE JOB IS ACTIVE OR SUSPENDED. IT IS CALLED OUT BY THE "MIX" INPUT MESSAGE.

ACTIVE ENTRY

IF A JOB IS BEING EXECUTED NORMALLY OR WAS TERMINATED BETWEEN THE TWO MOST RECENT UPDATES, ITS ENTRY CONTAINS THE FOLLOWING INFORMATION:

MI = <MIX INDEX>
C= COMPILER NAME IF COMPILE JOB
JOB = <JOB NAME>
P = <PRIORITY>

S = STATUS

BOJ = BEGINNING OF JOB
EOJ = END OF JOB
DS-ED = DISCONTINUED
<EMPTY> = RUNNING

A TYPICAL ACTIVE ENTRY IS AS FOLLOWS:

```
-------------------------------------------
:                                         :
:  __MI___ C_____ JOB_________ P S        :
:  ------- ------ ------------ - -        :
:   0013   ALGOL  CHECK/WRITER@5* BOJ     :
:                                         :
-------------------------------------------
```

SUSPENDED ENTRY

IF A JOB IS SUSPENDED FOR ANY REASON ITS MIX ENTRY CHANGES FROM ACTIVE TO SUSPENDED AND CONTAINS THE FOLLOWING INFORMATION:

MI = <MIX INDEX>
P = <PRIORITY>
REASON = AN OUTPUT MESSAGE GIVING THE REASON FOR SUSPENSION
ACTION = ABBREVIATION FOR ONE OR MORE INPUT MESSAGES
         REQUIRED TO REACTIVATE PROCESSING.

A TYPICAL SUSPENDED ENTRY IS AS FOLLOWS:

```
--------------------------------------------
:  MI    P         REASON           ACTION  :
 ------  - ----------------- ----------- 
: 0013@5:NO FILE = MASTER/FILE:OF,UL,IL,DS :
--------------------------------------------
```

SCHEDULE TABLE

FOLLOWING ENTRY OF THE INPUT MESSAGE "SCH" AT THE CONTROL UNIT, THE SCHEDULE TABLE WILL REPLACE THE MIX TABLE FOR A PERIOD OF TIME, THE LENGTH OF WHICH DEPENDS UPON THE NUMBER OF ENTRIES. THE ENTRY FOR A JOB IN THE SCHEDULE CONTAINS THE FOLLOWING INFORMATION:

```
    SI = SCHEDULE INDEX -- THIS WILL BECOME THE <MIX INDEX> UPON
                           ENTRY INTO THE MIX.
    JOB = <JOB NAME>
    P = <PRIORITY>
    C = COMPILER NAME IF COMPILE JOB

    CR = CORE REQUIRED (TENTHS OF PERCENT OF USABLE CORE)
    ST = TIME IN MINUTES SINCE ENTRY INTO SCHEDULE
```

A TYPICAL SCHEDULE TABLE ENTRY IS AS FOLLOWS:

```
--------------------------------------------
:   SI    C    JOB            P  CR     ST   :
    --    -    ---            -  --     --
:  0013        CORP/PAYCH@5:62.5,5.7         :
--------------------------------------------
```

PERIPHERAL UNIT TABLE

THIS TABLE IS CALLED WITH THE INPUT MESSAGE "PER <UNIT MNEMONIC>" AND HAS AN ENTRY FOR EACH PERIPHERAL UNIT IN THE SYSTEM. AN ENTRY CONTAINS THE MINIMUM INFORMATION NECESSARY FOR DETERMINATION OF THE STATUS AND CONTENT OF A GIVEN UNIT.

A TYPICAL PERIPHERAL UNIT TABLE ENTRY IN RESPONSE TO THE "PERMT" MESSAGE IS AS FOLLOWS:

```
-------------------------------------------------
:  UNIT       STATUS                            :
   ----       ------
:                                               :
:  MT001      RW/L UNLABELLED                   :
:  MT002      SCRATCH                           :
:  MT003      RW/L UNLABELLED                   :
:  MT004      SYSTEM/FILE                       :
:  MT005      IN USE BY READALABEL              :
-------------------------------------------------
```

LABEL TABLE

THIS TABLE IS CALLED WITH THE INPUT MESSAGE "OL <UNIT MNEMONIC>" AND CONTAINS AN ENTRY FOR EACH I/O UNIT OF THE DESIGNATED TYPE WHICH IS ON LINE. IF NO UNITS OF THE DESIGNATED TYPE ARE ON LINE THE OUTPUT MESSAGE "NULL <UNIT MNEMONIC> TABLE" WILL APPEAR.

A TYPICAL LABEL TABLE ENTRY IS AS FOLLOWS:

```
------------------------------------------------------------------------
:                                                                      :
:  UNIT   FILEID         STATUS      JOB ID                            :
   ----   ------         ------      ------
:                                                                      :
:  CR001  INVEN/RECVD    IN USE BY   INVEN/UPDATE                      :
:                                                                      :
------------------------------------------------------------------------
```

DISK DIRECTORY TABLE

THIS TABLE IS CALLED WITH THE "PD <FILE SET SPECIFIER>" INPUT MESSAGE. IT CONTAINS ALL FILE NAMES IN THE DISK DIRECTORY WHICH ARE IN THE SET SPECIFIED BY THE INPUT MESSAGE. IF THE SPECIFIED

SET IS EMPTY THE OUTPUT MESSAGE "NULL" WILL APPEAR.

A TYPICAL DIRECTORY TABLE IN RESPONSE TO THE INPUT MESSAGE "PD SYSTEM/=" IS:

```
SYSTEM
ALGOL (PROG), COBOL (PROG),
FORTRAN (PROG)
```

FOR A COMPLETE LISTING ON THE PRINTER OF ALL FILES STORED ON DISK USE THE "DIR" INPUT MESSAGE.

JOB TABLE

THIS TABLE IS CALLED WITH THE "JOB" INPUT MESSAGE SPECIFYING ANY JOB IN THE MIX. IT CONTAINS DETAILED INFORMATION ABOUT THE JOB AS FOLLOWS:

1. MIX TABLE ENTRY
2. LISTING OF CONTROL CARDS
3. CORRELATION OF PHYSICAL UNITS WITH <FILE NAME>S
4. ASSOCIATED PROCESSES (<MIX INDEX> SUFFIXES)

A TYPICAL JOB TABLE IS A FOLLOWS:

```
13.097=CORPO/PAYCH:5: :R,184,3.2
EXECUTE CORPORATIONX/PAYCHECKWRITER.FOR
WEEK ENDING 1-3-69
FILE CARD=HOURLY
FILE DISK=PAYROLLINFO/HOURLY
FILE NEWDISK=PAYROLLINFO/HOURLY/UPDATED
FILE LINE=LINE PRINT OR BACKUP
CD010=CARD
LP002=LINE
.1,.2,.2.1,.3
```

A-2. MESSAGES

THE OPERATOR COMMUNICATES DIRECTLY WITH THE MCP THROUGH THE USE OF INPUT/OUTPUT MESSAGES. ALL INPUT MESSAGES AND CERTAIN OUTPUT MESSAGES ARE DISPLAYED AS THEY OCCUR IF A DISPLAY UNIT IS AVAILABLE. OTHERWISE THEY WILL APPEAR ONLY IN THE SYSTEM LOG.

INPUT MESSAGES

INFORMATION MAY BE SUPPLIED TO THE MCP THROUGH THE USE OF INPUT

MESSAGES ENTERED IN FREE FIELD FORMAT AT THE CONTROL UNIT KEYBOARD. THESE MESSAGES ARE NOT INTENDED TO PROVIDE DETAILED INFORMATION ABOUT INDIVIDUAL PROGRAMS, E.G., THE SETTINGS FOR REGISTERS OR THE CONTENTS FOR MEMORY LOCATIONS.

TO ENTER A MESSAGE THE OPERATOR MUST FIRST DEPRESS THE LOC KEY. AFTER KEYING IN THE MESSAGE, HE MUST DEPRESS THE ETX KEY THEN THE XMIT KEY. IF THE MESSAGE IS NOT RECOGNIZABLE THE MCP WILL NOT ACT UPON IT EXCEPT TO GIVE AN "INV KBD" (INVALID KEYBOARD) OUTPUT MESSAGE.

THE INPUT MESSAGES APPEAR BELOW WITH THEIR REQUIRED SPELLING. FOLLOWING EACH MESSAGE IS A BRIEF DESCRIPTION OF ITS PURPOSE AND EFFECT. MESSAGES WHICH MAY RESULT IN THE DISPLAY OF A TABLE HAVE THREE LETTER MNEMONICS.

<MIX INDEX> AX

THIS MESSAGE IS ENTERED IN RESPONSE TO A "<MIX INDEX> ACCEPT" COBOL OUTPUT MESSAGE.

? <CONTROL STATEMENT LIST>;END

WHERE

```
   <CONTROL STATEMENT LIST> ::= <CONTROL STATEMENT> /
           <CONTROL STATEMENT LIST>; <CONTROL STATEMENT>
```

ANY CONTROL STATEMENT ALLOWED ON A CONTROL CARD MAY BE ENTERED. MULTIPLE CONTROL STATEMENTS MAY BE ENTERED ON A LINE BY SEPARATING THEM WITH SEMICOLONS. THE LAST CONTROL STATEMENT MUST BE AN END STATEMENT.

CL <UNIT MNEMONIC>

ALL EXCEPTION FLAGS MAINTAINED BY. THE MCP FOR THE SPECIFIED UNIT WILL BE RESET (CLEARED). IF THE SPECIFIED UNIT IS A PSEUDO CARD READER, THE DECK IT CONTAINS WILL BE ELIMINATED. (NOTE: CLEARING OF A UNIT ASSIGNED TO A JOB WILL RESULT IN

IMMEDIATE DISCONTINUATION OF THE JOB.)

CM <FILE NAME>

THE RUNNING MCP WILL BE CHANGED TO THE MCP SPECIFIED BY <FILE NAME>. FILE NAME SHOULD BE THE NAME OF THE NEW MCP CODE FILE. THE MESSAGE "MCP CHANGE PENDING" WILL BE DISPLAYED. THE CHANGE WILL THEN OCCUR AUTOMATICALLY WHEN MIX COUNTS EQUALS ZERO.

<MIX INDEX> CO <ASSOCIATIVE OPTIONS>

THE "CO" (CHANGE OPTIONS) MESSAGE IS USED TO CHANGE THE ASSOCIATIVE OPTIONS WHICH CONTROL THE OUTPUTTING OF A BACKUP FILE CURRENTLY IN PROCESS. FOR A MORE COMPLETE DISCUSSION SEE SECTION 6.2 (PRINTER AND PUNCH BACKUP) OF THIS DOCUMENT.

DIR

THE PROGRAM "LIST/DIRECTORY" WILL BE ENTERED INTO THE MIX AND A LISTING OF FILES STORED ON DISK PRODUCED AT THE PRINTER.

<MIX INDEX> DS

THE SPECIFIED PROGRAM WILL BE DISCONTINUED.

DR <INTEGER>/ <INTEGER> / <INTEGER>

THE DATE USED BY THE MCP WILL BE RESET TO THAT SPECIFIED. THE THREE <INTEGER>S ARE MONTH (1 TO 12), DAY (1 TO 31), AND YEAR (0 TO 99), RESPECTIVELY.

<MIX INDEX> FM <UNIT MNEMONIC>

THIS MESSAGE MUST BE ENTERED IN RESPONSE TO A "FM RQD" MESSAGE. THE <UNIT MNEMONIC> SPECIFIES THE UNIT TO BE USED FOR THE SUBJECT FILE.

<MIX INDEX> FR

THIS MESSAGE SPECIFIES THAT THE INPUT REEL, THE READING OF

WHICH WAS JUST COMPLETED, WAS THE FINAL REEL OF AN UNLABELED FILE.

<MIX INDEX> IL <UNIT MNEMONIC>

THIS MESSAGE IS ENTERED IN RESPONSE TO A "NO FILE" MESSAGE AND SPECIFIES THE UNIT ON WHICH THE REQUIRED INPUT FILE IS LOCATED. THE FILE MAY BE EITHER LABELED OR UNLABELED.

<MIX INDEX> JOB

THE JOB TABLE FOR THE SPECIFIED JOB WILL BE DISPLAYED ON THE UNIT WHERE THIS MESSAGE IS ENTERED.

LC <COMMENT>

THE "LC" (LOG COMMENT) MESSAGE CAUSES ANY OPERATOR-ENTERED <COMMENT> TO BE ENTERED INTO THE SYSTEM/LOG.

LD DK   OR
LD MT

THE "SYSTEM/LOADCONTROL" PROGRAM WILL SEARCH FOR A TAPE OR CARD FILE WITH A <FILE LABEL> OF "CONTROLDECK". IF FOUND, THE FILE WILL BE PLACED ON DISK AS A PSEUDO CARD DECK FOR DK, OR ON MAGNETIC TAPE FOR MT.

LOG <RANGE> <REQUEST LIST>

THIS MESSAGE WILL SELECTIVELY RETRIEVE INFORMATION FROM THE SYSTEM/LOG AND LIST IT ON THE PRINTER. FOR A MORE COMPLETE DISCUSSION SEE SECTION 6.4 (SYSTEM LOGS) OF THIS DOCUMENT.

LR

THE "LR" (LOG RELEASE) MESSAGE CAUSES THE CURRENT SYSTEM/LOG TO BE RENAMED AND SAVED AND ANOTHER SYSTEM/LOG FILE TO BE CREATED. SEE SECTION 6.4 (SYSTEM LOGS) FOR A MORE COMPLETE DISCUSSION.

MIX OR
MIX SC <INTEGER>

THE MIX TABLE WILL BE DISPLAYED ON THE SPECIFIED DISPLAY UNIT. IF NO DISPLAY UNIT IS SPECIFIED, THE ONE AT WHICH THE MESSAGE IS ENTERED WILL BE ASSUMED.

NEXT

THIS MESSAGE CAUSES THE NEXT "PAGE" OF SPO OUTPUT TO BE DISPLAYED. SPO DISPLAY INFORMATION REQUIRING MORE THAN ONE FULL SCREEN IS "PAGED". NORMALLY ONLY THE FIRST PAGE IS DISPLAYED.

<MIX INDEX> OF

THIS MESSAGE MAY BE ENTERED IN RESPONSE TO A "NO FILE" MESSAGE IF THE FILE IS AN OPTIONAL FILE. THE SPECIFIED PROGRAM WILL THEN PROCEED WITHOUT IT BY TAKING "END OF FILE" ACTION ON THE SPECIFIED FILE.

<MIX INDEX> OK

THE MCP WILL REACTIVATE A JOB WHICH WAS SUSPENDED BECAUSE OF AN OPERATOR "ST" (SUSPEND TEMPORARILY) MESSAGE.

OL <UNIT MNEMONIC>

THE LABEL TABLE WILL BE DISPLAYED ON THE UNIT WHERE THIS MESSAGE IS ENTERED.

<MIX INDEX> OT <DELTA>

THE CONTENTS OF THE STACK CELL GIVEN BY <DELTA> OF THE JOB INDICATED BY <MIX INDEX> WILL BE DISPLAYED. <DELTA> IS SOME INTEGER FROM TWO TO THE NUMBER OF DECLARED VARIABLES IN THE PROGRAMS OUTER BLOCK.

<MIX INDEX> OU <OUTPUT CODE>

THIS MESSAGE MAY BE ENTERED IN RESPONSE TO AN OUTPUT MESSAGE REQUESTING A LINE PRINTER OR PRINTER BACKUP TAPE. THE <OUTPUT CODE> MAY BE <EMPTY> OR ONE OF THE FOLLOWING TWO LETTER CODES: LP = LINE PRINTER, MT = MAGNETIC TAPE (PRINTER BACKUP TAPE), DK = DISK (PRINTER BACKUP DISK). THE SUBJECT LINE PRINTER FILE MUST BE PRODUCED ON THE SPECIFIED UNIT. IF THE <OUTPUT CODE> IS <EMPTY> EITHER LP OR MT MAY BE USED.

PD <FILE SET SPECIFIER>

WHERE

```
<FILE SET SPECIFIER> ::= <FILE LABEL> / <FILE LABEL> <SLASH>=
```

THE PD (PRINT DIRECTORY) SPO MESSAGE ALLOWS THE USER TO DISPLAY SELECTIVELY ON THE SPO THE NAMES OF FILES IN THE DISK DIRECTORY

<FILE SET SPECIFIER> IS A PARTIAL OR FULL FILE NAME. IF A FULL FILE NAME IS SPECIFIED WITHOUT AN EQUAL SIGN THEN THE DISK DIRECTORY WILL BE SEARCHED AND A MESSAGE DISPLAYED ON THE SPO STATING WHETHER THE FILE IS PRESENT. IF A PARTIAL FILE NAME IS GIVEN WITH AN EQUAL SIGN THEN A LIST OF ALL DISK FILES BEGINNING WITH THE PARTIAL FILE NAME WILL BE DISPLAYED. IF NO FILE IS FOUND THE MESSAGE "NULL" WILL BE DISPLAYED.

EXAMPLES:

```
PD CARD/LINE
PD SYSTEM/=
```

FOR A COMPLETE LISTING OF THE DIRECTORY ON THE PRINTER USE THE "DIR" INPUT MESSAGE.

PER <UNIT TYPE MNEMONIC>

WHERE:

```
<UNIT TYPE MNEMONIC> ::= <UNIT MNEMONIC> / CD / CP / CR /
                         LP / MT / MTX / PP / PR / SP
```

THE SPECIFIED PERIPHERAL TABLE WILL BE DISPLAYED ON THE UNIT WHERE THIS MESSAGE IS ENTERED.

<MIX INDEX> PR = <PRIORITY>

THE <PRIORITY> OF THE SPECIFIED JOB IN THE MIX OR SCHEDULE WILL BE SET TO <PRIORITY>.

<MIX INDEX> QT <ASSOCIATIVE OPTIONS>

THE "QT" (QUIT) MESSAGE IS USED TO TERMINATE THE PRINTING OF A BACKUP FILE. FOR A DISCUSSION OF <ASSOCIATIVE OPTIONS> SEE SECTION 6.2 (PRINTER AND PUNCH BACKUP) OF THIS DOCUMENT.

RD =
RD <DECK LIST>

WHERE:
<DECK LIST> ::= <DECK NUMBER> / <DECK LIST>,<DECK NUMBER>
<DECK NUMBER> ::= #<INTEGER>

THE SPECIFIED PSEUDO CARD DECKS WILL BE REMOVED FROM DISK. IF THE FORM "RD =" IS USED, ALL PSEUDO CARD DECKS WILL BE REMOVED.

<MIX INDEX> RM

THIS MESSAGE MAY BE USED IN RESPONSE TO A "DUP LIBRARY" OUTPUT MESSAGE. THE DISK FILE WITH THE LABEL SPECIFIED IN THE "DUP LIBRARY" MESSAGE WILL BE REMOVED.

RN OR
RN <INTEGER>

THE <INTEGER> SPECIFIES THE NUMBER OF PSEUDO CARD READERS TO BE USED. AT "HALT-LOAD" TIME THE NUMBER SPECIFIED IS ZERO. IF THIS MESSAGE REQUIRES THAT PSEUDO READERS BE TURNED OFF, THE MCP WILL COMPLETE THE HANDLING OF PSEUDO CARD DECKS IN PROCESS, IF ANY, BEFORE BEING TURNED OFF. IF NO <INTEGER> IS INCLUDED, THE CURRENT NUMBER OF PSEUDO CARD READERS WILL BE DISPLAYED.

RO -- (SEE SO)

RW <UNIT MNEMONIC>

A REWIND AND LOCK ACTION WILL BE PERFORMED ON THE FILE ON THE SPECIFIED MAGNETIC TAPE UNIT. IF THE UNIT IS IN USE THE ACTION WILL BE PERFORMED UPON THE RELEASE OF THE FILE.

RY <UNIT MNEMONIC>

THE SPECIFIED UNIT WILL BE MADE READY FOR USE IF IT IS IN "REMOTE" STATUS AND IS NOT IN USE.

SCH OR
SCH SC <INTEGER>

THE SCHEDULE TABLE WILL BE DISPLAYED ON THE SPECIFIED DISPLAY UNIT. IF NO DISPLAY UNIT IS SPECIFIED, THE ONE AT WHICH THE MESSAGE IS ENTERED WILL BE ASSUMED.

SO <OPTION SPECIFIER> OR
RO <OPTION SPECIFIER> OR
TO <OPTION SPECIFIER>

WHERE
<OPTION SPECIFIER> ::= OPEN / RET / TERMINATE / SEGMENT / <EMPTY>

THE SPECIFIED OPTION WILL BE SET, RESET, OR TYPED (DISPLAYED) RESPECTIVELY. THE OPTIONS AND MNEMONICS ARE TO BE SPECIFIED. THE OPTION SPECIFIERS MAY BE <EMPTY> WHICH CAUSES ALL OPTIONS TO BE SET, RESET OR TYPED.

AT PRESENT FOUR OPTIONS ARE AVAILABLE. WITH "OPEN" SET, FILE OPEN MESSAGES ARE DISPLAYED ON THE SPO. WITH "RETAIN" SET, TAPES WITH EXPIRED "SAVE" FACTORS AND WRITE RINGS ARE AUTOMATICALLY PURGED; OTHERWISE A "RETAIN" MESSAGE FOR THE TAPE IS DISPLAYED. WITH "TERMINATE" SET, ABNORMAL JOB TERMINATIONS WILL RESULT IN AN ATTEMPTED PROGRAM DUMP RATHER THAN A FULL MEMORY DUMP. WITH "SEGMENT" SET, ONE-DIMENSIONAL ARRAYS WILL BE DIVIDED INTO 256-WORD SEGMENTS.

SN MT <UNIT NUMBER> <SERIAL NUMBER>

THE TAPE ON THE SPECIFIED MAGNETIC TAPE UNIT WILL BE PURGED IF THE TAPE UNIT IS IN READY, NOT IN USE AND IF THE TAPE HAS A WRITE RING.

<MIX INDEX> ST

THE SPECIFIED JOB WILL BE SUSPENDED TEMPORARILY. IT MAY BE REACTIVATED WITH AN "OK" MESSAGE.

SV <UNIT MNEMONIC>

THE SPECIFIED UNIT WILL BE MADE INACCESSIBLE AS SOON AS IT IS NOT IN USE. IT MAY BE MADE ACCESSIBLE WITH AN "RY" MESSAGE OR A "HALT-LOAD" OPERATION. THE MESSAGE "<UNIT MNEMONIC> TO BE SAVED" OR "<UNIT MNEMONIC> SAVED" WILL BE DISPLAYED AS APPROPRIATE.

<MIX INDEX> TI

THE FOLLOWING OUTPUT MESSAGE WILL BE DISPLAYED:

<MIX INDEX> : <PROCESSOR TIME> IN FOR <ELAPSED TIME>

WHERE <PROCESSOR TIME> IS THE PROCESSOR TIME USED AND <ELAPSED TIME> IS THE ELAPSED TIME SINCE THE JOB ENTERED THE MIX. BOTH ARE GIVEN IN MINUTES AND TENTHS OF MINUTES.

TO -- (SEE SO)

TR <INTEGER>

THE TIME WILL BE RESET TO THAT SPECIFIED BY THE <INTEGER> WHICH MUST BE FOUR <DIGIT>S. THE FIRST TWO <DIGIT>S SPECIFY THE HOUR (0 TO 23) AND THE LAST TWO SPECIFY THE MINUTE (0 TO 59).

<MIX INDEX> UL <UNIT MNEMONIC>

THIS MESSAGE MAY BE USED IN RESPONSE TO A "NO FILE" MESSAGE IN

ORDER TO DESIGNATE THE UNIT ON WHICH AN UNLABELED FILE IS LOCATED. THE SUBJECT FILE MAY BE EITHER LABELED OR UNLABELED. ALL RECORDS INCLUDING THE LABEL IF ANY WILL BE READ AS DATA. (THIS MESSAGE DIFFERS FROM THE "IL" MESSAGE IN THAT WITH THE "IL" MESSAGE THE LABEL IS NOT READ AS DATA.)

WD

THE MCP WILL DISPLAY THE DATE CURRENTLY BEING USED BY THE SYSTEM. THE DATE IS GIVEN IN THE FORMAT MM/DD/YY.

WM

THE MCP WILL DISPLAY THE MODIFICATION LEVEL AND PATCH REVISION NUMBER IN THE FORM:

```
B6700 MCP   LEVEL XX.PPP
```

WT

THE MCP WILL DISPLAY THE TIME OF DAY AT THE TIME THE MESSAGE WAS ENTERED. THE TIME IS GIVEN IN HOURS AND MINUTES BASED ON A 24 HOUR CLOCK.

OUTPUT MESSAGES

OUTPUT MESSAGES WHICH APPEAR ONLY AS ANSWERS TO DIRECT QUESTIONS WILL BE DESCRIBED WITH THE CORRESPONDING INPUT MESSAGE. THE REMAINDER OF THE OUTPUT MESSAGES APPEAR BELOW AS THEY ARE DISPLAYED. FOLLOWING EACH MESSAGE IS A BRIEF DESCRIPTION OF ITS MEANING AND ANY REQUIRED OPERATOR RESPONSE.

<MIX INDEX> ACCEPT

AN OBJECT PROGRAM EXECUTED AN "ACCEPT" STATEMENT. AN "AX" INPUT MESSAGE IS REQUIRED.

<MIX INDEX> <FUNCTION NAME> <INVALID ARG> <PARAMETER VALUE>

<TERMINAL REFERENCE>

AN INVALID ARGUMENT TO A MATHEMATICAL INTRINSIC FUNCTION WAS ENCOUNTERED. THE PROGRAM IS TERMINATED AND THE MESSAGE IS DISPLAYED TO THE OPERATOR. IN ADDITION, THE MESSAGE IS RECORDED IN THE SYSTEM LOG. IF A PRINTER FILE WAS OPEN AT THE TIME THE ERROR OCCURRED, THE MESSAGE IS ALSO WRITTEN ON THE PRINTER FILE. THE FOLLOWING INTRINSIC FUNCTIONS MAY GENERATE AN INVALID ARGUMENT MESSAGE:

LNGAMMA
LOG
LN
ARCOS
ARSIN
ARCTAN
ARCTAN2
CABS
CCOS
CDIV
CEXP
CLN
CMUL
COS
COSH

COTAN
CSIN
CSQRT
CTOD
CTOP
DARCTAN
DARCTAN2
DCOS
DELTA
DEXP
DLOG
DLN
DSIN
DSQRT
DTOD

ERF
EXP
GAMMA
RTOD
RTOP
RANDOM
SIN
SINH
SQRT
TAN
TANH
TIME

(NOTE: "X"TOP MEANS X RAISED TO AN INTEGRAL OR REAL POWER).

<FILE LABEL> CHANGED TO <FILE LABEL>

THE MCP HAS PERFORMED AN OPERATION SPECIFIED IN A "CHANGE" CONTROL STATEMENT.

DECK <INTEGER> REMOVED

THE SPECIFIED CONTROL DECK WAS REMOVED FROM DISK BECAUSE OF COMPLETION OF THE JOB OR AN INPUT MESSAGE.

<MIX INDEX> DIV BY ZERO <TERMINAL REFERENCE>

AN OBJECT PROGRAM ATTEMPTED A DIVIDE OPERATION USING A ZERO DIVISOR.

<FILE LABEL> COPIED

THE MCP HAS PERFORMED THE OPERATION SPECIFIED IN A "COPY" CONTROL STATEMENT.

<MIX INDEX> DUP FIL <FILE LABEL>

THE OBJECT PROGRAM ATTEMPTED TO ACCESS AN INPUT FILE BUT THE MCP FOUND MORE THAN ONE FILE WITH THE SPECIFIED <FILE LABEL>. THE CONDITION CAN BE CORRECTED BY MAKING ONLY ONE OF THE FILES AVAILABLE, THEN ENTERING A "<MIX INDEX> OK" MESSAGE OR BY ENTERING A <MIX INDEX> "IL" OR <MIX INDEX> "UL".

<MIX INDEX> DUP LIBRARY <FILE LABEL>

AN ATTEMPT WAS MADE TO ENTER A FILE IN THE DISK LIBRARY WHEN ITS <FILE LABEL> WAS IDENTICAL TO A <FILE LABEL> ALREADY IN THE DISK DIRECTORY. THE CONDITION MAY BE CORRECTED BY USING A "CHANGE" OR "REMOVE" CONTROL STATEMENT FOLLOWED BY A <MIX INDEX> OK MESSAGE OR BY ENTERING A <MIX INDEX> RM MESSAGE.

<MIX INDEX> EXPON OVERFLOW <TERMINAL REFERENCE>

AN OBJECT PROGRAM PERFORMED AN OPERATION WHICH CAUSED AN EXPONENT OVERFLOW TO OCCUR.

<MIX INDEX> INTGR OVERFLOW <TERMINAL REFERENCE>

AN OBJECT PROGRAM PERFORMED AN OPERATION WHICH CAUSED AN INTEGER OVERFLOW TO OCCUR.

<MIX INDEX> INV ADDRESS <TERMINAL REFERENCE>

AN OBJECT PROGRAM PERFORMED AN OPERATION WHICH ADDRESSED A NON-EXISTENT MEMORY LOCATION.

<MIX INDEX> INVALID INDEX <TERMINAL REFERENCES>

AN OBJECT PROGRAM ATTEMPTED TO INDEX OUT OF THE RANGE OF AN ARRAY BEING REFERENCED.

<MIX INDEX> <UNIT MNEMONIC> INVALID CHR. IN COL. <INTEGER>

AN <INVALID CHARACTER> HAS APPEARED IN A POSITION OTHER THAN CHARACTER POSITION 1 OF A RECORD. THE <INTEGER> IS THE COLUMN NUMBER.

INV KBD <TYPED-IN INFORMATION>

THE MCP WAS NOT ABLE TO RECOGNIZE A MESSAGE ENTERED FROM THE KEYBOARD.

<UNIT MNEMONIC> I/O INV ADDRESS

AN INVALID ADDRESS OCCURRED WHEN DATA WAS TO BE TRANSFERRED BETWEEN AN I/O CHANNEL AND PRIMARY MEMORY. THE MCP RECOGNIZES THE ERROR CONDITION AND, IF POSSIBLE, RECTIFIES THE ERRORS. THE PRIMARY PURPOSE OF THIS MESSAGE IS TO DRAW ATTENTION TO A CONDITION WHICH COULD DENOTE A HARDWARE FAILURE.

<UNIT MNEMONIC> I/O MEM PAR

A PARITY ERROR OCCURRED WHEN DATA WAS TO BE TRANSFERRED BETWEEN AN I/O CHANNEL AND PRIMARY MEMORY. THE MCP RECOGNIZES THE ERROR CONDITION AND, IF POSSIBLE, RECTIFIES THE ERRORS. THE PRIMARY PURPOSE OF THIS MESSAGE IS TO DRAW ATTENTION TO A CONDITION WHICH COULD DENOTE A HARDWARE FAILURE.

<FILE LABEL> COPIED

THE MCP HAS PERFORMED THE OPERATION SPECIFIED IN A "LOAD" CONTROL STATEMENT.

<UNIT MNEMONIC> LP BACKUP

A PRINTER BACKUP TAPE IS ON LINE. IF THE TAPE IS TO BE

PRINTED, A PB MESSAGE MUST BE ENTERED.

<UNIT MNEMONIC> NEW PBT

A NEW PRINTER BACKUP TAPE WAS OPENED.

<MIX INDEX> NO FILE <FILE LABEL>

A PROGRAM NEEDS AN INPUT FILE WHICH IS APPARENTLY UNAVAILABLE. IF THE FILE IS LABELED IT MUST BE MADE AVAILABLE. IF THE FILE IS NOT LABELED, AN "UL" MESSAGE IS REQUIRED. IF IT IS AN OPTIONAL FILE, AN "OF" MESSAGE IS REQUIRED. IF A PROGRAM HAS READ THE FINAL VOLUME OF A MULTI-VOLUME UNLABELED FILE, A "FR" MESSAGE IS REQUIRED.

<MIX INDEX> NO MEM

THE MCP WAS UNABLE TO OBTAIN REQUIRED PRIMARY MEMORY. <MIX INDEX> OK OR DS IS REQUIRED.

<FILE LABEL> NOT COPIED -- NOT ON <TAPE OR DISK>

LIBRARY MAINTENANCE COULD NOT LOCATE A FILE IT WAS TOLD TO COPY.

<PROGRAM ID> NOT IN DIRECTORY

AN "EXECUTE", "RUN" OR "COMPILE" STATEMENT REFERENCED A PROGRAM WHICH WAS NOT IN THE DISK DIRECTORY.

<MIX INDEX> <UNIT MNEMONIC> NOT READY

AN I/O OPERATION WAS ATTEMPTED ON A UNIT THAT WAS "NOT READY".

<MIX INDEX> OPERATOR STOPPED

THE SPECIFIED JOB WAS SUSPENDED IN RESPONSE TO AN ST INPUT MESSAGE. AN "OK" MESSAGE IS REQUIRED TO CONTINUE PROCESSING.

<MIX INDEX> <PROGRAM ID> DS-ED <TERMINAL REFERENCE>

THE SPECIFIED JOB WAS DISCONTINUED IN RESPONSE TO A "DS" INPUT MESSAGE.

<UNIT MNEMONIC> SCRATCH

A TAPE WAS PURGED BY AN INPUT MESSAGE OR A PROGRAM.

<MIX INDEX> <UNIT MNEMONIC> PRINT CHECK

A PRINT CHECK ERROR OCCURRED DURING PRINTING OF A LINE ON A LINE PRINTER. PROCESSING CONTINUES NORMALLY.

<MIX INDEX> <UNIT MNEMONIC> PUNCH CHECK

AN IRRECOVERABLE PUNCH CHECK ERROR OCCURRED DURING THE PUNCHING OF A CARD WHICH REQUIRES OPERATOR ATTENTION TO THE PUNCH UNIT. PROCESSING CONTINUES NORMALLY.

<MIX INDEX> <UNIT MNEMONIC> READ CHECK

A READ CHECK ERROR OCCURRED ON A CARD READER. THE LAST CARD READ MUST BE RE-READ, IF THE SECOND CARD ALSO FAILS, THE CARD HOLE PUNCHES MAY BE OFF OR THE CARD READER MAY NEED SERVICING.

<FILE LABEL> REMOVED

AN OPERATION SPECIFIED IN A "REMOVE" CONTROL STATEMENT HAS BEEN COMPLETED.

<MIX INDEX> <FILE LABEL> REQUIRES <UNIT MNEMONIC>

A JOB REQUIRES A PERIPHERAL DEVICE AND NONE WAS AVAILABLE.

<UNIT MNEMONIC> RW/L

A TAPE HAS BEEN REWOUND AND LOCKED.

<MIX INDEX> STACK OVERFLOW <TERMINAL REFERENCE>

THE OPERATIONS PERFORMED BY AN OBJECT PROGRAM HAVE CAUSED ITS

STACK TO OVERFLOW ITS LIMIT, AND THE MCP WAS UNABLE TO EXTEND IT.

<MIX INDEX> <UNIT MNEMONIC> WRITE LOCK-OUT

A PROGRAM ATTEMPTED TO WRITE ON A MAGNETIC TAPE WITH NO WRITE RING, OR A DISK WHICH HAS BEEN LOCKED OUT WITH HARDWARE LOCKOUT SWITCHES.

A-3. CONTROL CARDS

INFORMATION MAY BE PASSED TO THE MCP THROUGH THE USE OF PUNCHED CARDS CALLED CONTROL CARDS. CONTROL CARDS ARE DISTINGUISHED FROM OTHER CARDS BY AN <INVALID CHARACTER> IN COLUMN 1. CONTROL INFORMATION (WITH OR WITHOUT <COMMENT>S) IS PUNCHED IN COLUMNS 2 - 80. THE FORMAT FOR THIS INFORMATION IS FREE FIELD. ALL IDENTIFIERS AND CONSTANTS ARE TERMINATED BY A SPECIAL CHARACTER (<SPACE>, ",", ETC.). IF A PERIOD APPEARS IN A CONTROL CARD, ALL OF THE INFORMATION FOLLOWING IT ON THE SAME CARD IS CONSIDERED TO BE COMMENTARY AND IS IGNORED BY THE MCP.

NORMALLY, BUT NOT NECESSARILY, ONE CONTROL CARD CONTAINS ONE CONTROL STATEMENT. IF TWO OR MORE CONTROL STATEMENTS ARE PUNCHED ON A SINGLE CONTROL CARD, THEY MUST BE SEPARATED BY SEMICOLONS. THE <INVALID CHARACTER> IS NOT REQUIRED OR ALLOWED FOLLOWING A SEMICOLON.

IF A CONTROL STATEMENT CANNOT BE CONTAINED ON ONE CONTROL CARD, THE STATEMENT MAY BE CONTINUED BY THE INSERTION OF A HYPHEN ON ALL BUT THE LAST CARD (AN <IDENTIFIER> MAY NOT BE DIVIDED BY A HYPHEN). ONLY THE FIRST CARD OF SUCH A GROUP MAY CONTAIN AN <INVALID CHARACTER>.

CONTROL STATEMENTS MAY ALSO BE ENTERED AT THE SUPERVISORY CONSOLE. (SEE INPUT MESSAGES)

THE FOLLOWING PARAGRAPHS DESCRIBE THE FORMAT AND FUNCTION OF EACH CONTROL STATEMENT ACCEPTED BY THE MCP.

COMPILE STATEMENT

<INVALID CHARACTER> COMPILE <PROGRAM NAME> <COMMENT> <COMPILER NAME>
<COMMENT> <DISPOSAL> <COMMENT>

WHERE:
<DISPOSAL>::= <EMPTY> / LIBRARY / SYNTAX

THE COMPILE STATEMENT DESIGNATES THE COMPILER TO BE USED AND THE TYPE OF COMPILE RUN TO BE MADE. THIS MUST BE THE FIRST CONTROL STATEMENT IN A COMPILATION JOB.

1. COMPILE AND EXECUTE (<DISPOSAL> = <EMPTY>) AFTER AN ERROR FREE COMPILATION THE COMPILED PROGRAM IS SCHEDULED FOR EXECUTION BUT THE <PROGRAM NAME> IS NOT ENTERED IN THE DISK DIRECTORY. THE DISK SPACE USED BY THE PROGRAM IS RELEASED AFTER THE EXECUTION IS TERMINATED.

2. COMPILE FOR LIBRARY (<DISPOSAL> = LIBRARY) THE OBJECT CODE FROM AN ERROR FREE COMPILATION IS LEFT ON DISK AND THE <PROGRAM NAME> IS ENTERED IN THE DISK DIRECTORY. THE COMPILED PROGRAM IS NOT EXECUTED.

3. COMPILE FOR SYNTAX CHECK (<DISPOSAL> = SYNTAX) THE COMPILED PROGRAM IS NOT EXECUTED AND THE <PROGRAM NAME> IS NOT ENTERED IN THE DISK DIRECTORY. THE DISK SPACE USED BY THE PROGRAM IS RELEASED UPON COMPLETION OF COMPILATION.

EXAMPLES:

```
? COMPILE A/B WITH ALGOL
? COMPILE C/D COBOL LIBRARY
? COMPILE E/F WITH FORTRAN FOR SYNTAX
```

EXECUTE STATEMENT OR RUN STATEMENT

<INVALID CHARACTER> EXECUTE <PROGRAM NAME> <COMMENT>

<INVALID CHARACTER> RUN <PROGRAM NAME> <COMMENT>

THE DESIGNATED PROGRAM IS CALLED FROM THE DISK AND EXECUTED. THIS MUST BE THE FIRST CONTROL STATEMENT IN A JOB NOT REQUIRING COMPILATION.

EXAMPLES:

```
? EXECUTE SYSTEM/LOADCONTROL
? RUN INVENTORY/UPDATE
```

COPY AND MOVE STATEMENTS

```
    <INVALID CHARACTER> <STATEMENT VERB> <STATEMENT LIST>
```

WHERE:

```
<STATEMENT VERB> ::= COPY/MOVE
<STATEMENT LIST> ::= <STATEMENT PART> / <SOURCE LIST> /
     <STATEMENT PART>, <STATEMENT LIST>
<STATEMENT PART ::= <SOURCE LIST> <DESTINATION LIST> /
     <SOURCE LIST>, <FILE LIST> <DESTINATION LIST>
<SOURCE LIST> ::= <FILE LIST> FROM <SOURCE> /
     <SOURCE LIST>, <FILE LIST> FROM <SOURCE>
<FILE LIST> ::= <FILE NAME> / <FILE LIST>, <FILE NAME>
<DESTINATION LIST> ::= TO <DESTINATION> /
<FILE NAME> ::= <IDENTIFIER> / <FILE NAME> <SLASH> <IDENTIFIER>
<SOURCE> ::= <DESTINATION> ::= DISK / <IDENTIFIER>
```

THE COPY AND MOVE STATEMENTS ALLOW THE USER TO TRANSFER FILES TO AND FROM LIBRARY TAPES AND DISK STORAGE. THE COPY STATEMENT MAKES A COPY OF THE SPECIFIED FILE; THE MOVE STATEMENT MAKES A COPY OF THE SPECIFIED FILE AND (IF THE ORIGINAL IS A DISK FILE) DESTROYS THE ORIGINAL FILE. IF THE SOURCE FILE IS A TAPE FILE COPY AND MOVE OPERATIONS ARE IDENTICAL, THAT IS, THE SOURCE FILE IS COPIED BUT NOT DESTROYED.

EXAMPLES:

```
          ? COPY SYSTEM/ALGOL, SYSTEM/COBOL FROM SYSTEM; END
          ? MOVE DISK/FILE TO FIRSTTAPE, TO SECONDTAPE; END
```

CHANGE STATEMENT

```
<INVALID CHARACTER> CHANGE <CHANGE LIST>
```

```
    WHERE:
    <CHANGE LIST> ::= <CHANGE ELEMENT> /
                      <CHANGE LIST>,<CHANGE ELEMENT>
    <CHANGE ELEMENT> ::= <FILE LABEL>TO<FILE LABEL>
```

THE FILE SPECIFIED BY THE FIRST <FILE LABEL> IN THE <CHANGE ELEMENT> IS RELABELED USING THE SECOND <FILE LABEL>.

EXAMPLE:

```
? CHANGE OLD/NAME TO NEW/NAME;END
```

REMOVE STATEMENT

<INVALID CHARACTER> REMOVE <FILE LABEL>

THE FILE SPECIFIED BY <FILE LABEL> WILL BE REMOVED FROM THE DISK DIRECTORY AND THE SPACE IT OCCUPIES MADE AVAILABLE AS SOON AS THE FILE IS NOT IN USE.

EXAMPLE:

```
? REMOVE OLD/FILE; END
```

DATA STATEMENT

<INVALID CHARACTER> DATA <FILE LABEL>

THE INFORMATION ON ALL CARDS AFTER THIS CONTROL STATEMENT UNTIL ANOTHER CONTROL CARD WILL BE DESIGNATED AS DATA AND WILL BE PLACED IN A FILE CALLED <FILE LABEL>. THIS <FILE LABEL> MUST BE THE SAME AS THE <FILE NAME> USED IN THE PROGRAM OR MUST BE LABEL EQUATED TO IT. THE DATA STATEMENT MUST BE THE LAST CONTROL STATEMENT BEFORE THE ACTUAL DATA.

BCL STATEMENT

<INVALID CHARACTER> BCL <FILE LABEL>

SAME AS ABOVE EXCEPT THE FOLLOWING CARDS CONTAIN BCL DATA.

BINARY STATEMENT

<INVALID CHARACTER> BINARY <FILE LABEL>

SAME AS ABOVE EXCEPT THE FOLLOWING CARDS CONTAIN BINARY DATA. NOTE: BINARY DATA DECKS ARE TERMINATED WITH A "BEND" (BINARY END) CARD RATHER THAN A NORMAL "END" CARD. A "BEND" CARD HAS THE LETTERS "BEND" WRITTEN IN PUNCHSCRIPT ACROSS THE CARD.

END STATEMENT

<INVALID CHARACTER> END

THIS STATEMENT DESIGNATES END-OF-FILE INFORMATION FOR A PARTICULAR PROGRAM AND IS REQUIRED WHENEVER A PROGRAM IS TERMINATED FOR ANY REASON WHILE IT HAS CARD INFORMATION YET TO BE READ. CONSEQUENTLY, IF AN END STATEMENT APPEARS IT MUST BE THE LAST CARD IN A DECK PERTAINING TO A PROGRAM. HOWEVER, AN END STATEMENT IS NOT NECESSARY TO DENOTE THE END OF A DATA FILE. AN ATTEMPT TO READ ANY CONTROL CARD AS DATA WILL CAUSE AN END-OF-FILE NOTIFICATION, HENCE, IF A PROGRAM REQUIRES MORE THAN ONE CARD FILE, THE END OF ONE FILE WILL BE DENOTED BY THE DATA STATEMENT FOR THE NEXT.

PROCESS TIME STATEMENT

<INVALID CHARACTER> <OPTIONAL COMPILER NAME> PROCESS
      <COMMENT> <INTEGER>

THIS STATEMENT SPECIFIES THE MAXIMUM PROCESS TIME IN SECONDS FOR THE OBJECT PROGRAM OR THE COMPILER. IF THE PROCESS TIME EXCEEDS THAT SPECIFIED, THE JOB WILL BE TERMINATED.

IO TIME STATEMENT

<INVALID CHARACTER> <OPTIONAL COMPILER NAME> IO <COMMENT> <INTEGER>

THIS STATEMENT SPECIFIES THE MAXIMUM IO TIME IN MINUTES FOR THE OBJECT PROGRAM OR THE COMPILER. IF THE IO TIME EXCEEDS THAT SPECIFIED THE JOB WILL BE TERMINATED.

STACK SIZE STATEMENT

<INVALID CHARACTER> <OPTIONAL COMPILER NAME> STACK <COMMENT> <INTEGER>

THIS STATEMENT SPECIFIES THE NUMBER OF WORDS TO BE ASSIGNED IN PRIMARY MEMORY FOR THE WORKING STACK OF THE COMPILER OR OBJECT PROGRAM. IF NO STACK SIZE STATEMENT APPEARS, THE WORKING STACK SIZE WILL BE 512 WORDS.

PRIORITY STATEMENT

<INVALID CHARACTER> <OPTIONAL COMPILER NAME> PRIORITY
    <COMMENT> <INTEGER>

THIS STATEMENT SPECIFIES THE PRIORITY TO BE ASSIGNED TO A COMPILATION OR AN OBJECT PROGRAM EXECUTION. PRIORITIES MAY RANGE FROM 0 TO MM WHERE 0 IS THE LOWEST PRIORITY AND MM (MIX MAX) IS THE HIGHEST PRIORITY. UNLESS OTHERWISE SPECIFIED A PRIORITY OF MM/2 WILL BE ASSUMED. FOR A "COMPILE AND EXECUTE" JOB, A PRIORITY ASSIGNED TO THE COMPILATION WILL ALSO APPLY TO THE EXECUTION UNLESS A SPECIFIC <PRIORITY> IS ASSIGNED WITH A CONTROL STATEMENT FOR THE EXECUTION OF THE PROCESS.

FILE (LABEL EQUATION) STATEMENT

```
<INVALID CHARACTER> <OPTIONAL COMPILER NAME> FILE
         <FILE NAME> = <FILE LABEL>
```

THE FILE STATEMENT IS USED TO ASSOCIATE THE <FILE NAME> USED IN THE PROGRAM WITH A PARTICULAR DATA FILE FOR EXECUTION.

```
DEVICE OPTION ::= <BACKUP OPTION>/
                  <INPUT DEVICE OPTION> /
                  <OUTPUT DEVICE MEMORY OPTION>/
                  <BACKUP OPTION> <OUTPUT DEVICE MEMORY OPTION>/
                  <EMPTY>

BACKUP OPTION ::= <BACK>/<BACKUP>/<BACK UP>/<EMPTY>

INPUT DEVICE OPTION ::= <PAGE READER>/<READER>
                  <EMPTY>

OUTPUT DEVICE MEMORY OPTION ::= <DISK>/<DISPLAY>/<PAPER PUNCH>/
                  <PRINTER>/<PUNCH>
                  <OUTPUT DEVICE MEMORY OPTION> OR
                  <EMPTY>
```

EXAMPLES:

```
? FILE A = B
? FILE PRINT = PRINT BACKUP DISK
? ALGOL FILE TAPE = SYMBOL/X/ALGOL
```

COMMON STATEMENT

TO BE SPECIFIED

IO UNIT STATEMENT

<INVALID CHARACTER> UNIT <UNIT MNEMONIC> <COMMENT> <FILE LABEL> <RDC>

THIS STATEMENT ASSOCIATES A <FILE LABEL> WITH A PARTICULAR IO UNIT. IT MAY BE USED WHEN AN INPUT FILE DOES NOT HAVE A LABEL AND OPERATOR INTERVENTION IS NOT DESIRED.

CORE REQUIRED STATEMENT

<INVALID CHARACTER> <OPTIONAL COMPILER NAME> CORE <COMMENT> <INTEGER>

THE STATEMENT INFORMS THE MCP OF THE CORE REQUIREMENT, IN WORDS, OF THE PROGRAM. IT WILL OVERRIDE THE ESTIMATE MADE BY THE COMPILER.

SAVE STATEMENT

<INVALID CHARACTER> SAVE <COMMENT> <INTEGER> <COMMENT>

THIS STATEMENT SPECIFIES THE NUMBER OF DAYS FROM LAST ACCESS FOR WHICH A PROGRAM IS TO BE SAVED IN THE DISK LIBRARY.

PRINTER BACKUP STATEMENT

<INVALID CHARACTER> PB <INPUT DESIGNATOR>
<OPTIONAL OUTPUT DESIGNATOR> <KEY DECLARATION> <PB OPTION PART>

WHERE:

<INPUT DESIGNATOR> ::= MT <INTEGER> / D <INTEGER>
<OPTIONAL OUTPUT DESIGNATOR> ::= <EMPTY> / LP <INTEGER> / CP <INTEGER>
<PB OPTION PART> ::= <GENERAL OPTIONS> / <PB OPTION PART>,
<GENERAL OPTIONS> / <PB OPTION PART> <ASSOCIATIVE OPTIONS>
<GENERAL OPTIONS> ::= <EMPTY> / SAVE / COPIES = <INTEGER>
<ASSOCIATIVE OPTIONS> ::= <EMPTY> / RANGE <LITERAL> <SPACER>
<LITERAL> / EQUAL <LITERAL> / <RECORD NUMBER OPTION>

```
<RECORD NUMBER OPTION> ::= RECORD <INTEGER> / RECORD <INTEGER>
     <SPACER> <INTEGER>
<SPACER> ::= <BLANK> / <SPACE> <BLANK>
```

THIS STATEMENT CAUSES PRINTER BACKUP FILES ON TAPE OR DISK TO BE PRINTED. FOR A DISCUSSION OF THE OPTIONS AVAILABLE SEE THE PRINTER BACKUP PART OF SECTION 6 OF THIS DOCUMENT.

EXAMPLES:

```
    ? PB MT6; END
    ? PB MT1 SAVE COPIES = 10 RECORD 1000 2000; END
    ? PB D25 CP1 SAVE RANGE "1Q765AB" "GG654"; END
```

APPENDIX B

SYSTEM LOG FORMATS

APPENDIX B - SYSTEM LOG FORMATS

THE FORMAT OF THE SYSTEM LOGS IS PROVIDED FOR THE CONVENIENCE OF USERS WISHING TO WRITE LOG-ANALYSIS AND BILLING PROGRAMS.

THERE ARE THREE LOGS: THE SYSTEM LOG, MAINTENANCE LOG AND DATA COMMUNICATIONS LOG. THE FORMAT OF THE DATACOM LOG IS TO BE SPECIFIED.

NOTE: THE FORMAT OF THE LOGS IS SUBJECT TO CHANGE.

B-1. SYSTEM LOG

EACH PHYSICAL BLOCK OF THIS LOG CONTAINS 30 WORDS, DIVIDED INTO FIVE 6-WORD RECORDS. EACH ENTRY TYPE HAS AT LEAST ONE FIXED 6-WORD RECORD; IN ADDITION, IT MAY HAVE A VARIABLE NUMBER OF 6-WORD RECORDS WITH THE NUMBER OF RECORDS BEING CONSTANT FOR MOST ENTRY TYPES (SEE TABLE B-1).

RECORDS--EACH 6 WORDS IN LENGTH. EACH ENTRY TYPE HAS AT LEAST ONE FIXED 6-WORD RECORD; IN ADDITION, IT MAY HAVE A VARIABLE NUMBER OF 6-WORD RECORDS WITH THE NUMBER OF RECORDS BEING CONSTANT FOR MOST ENTRY TYPES (SEE TABLE 1).

SYSTEM LOG ENTRY TYPES

	ENTRY	TYPE CODE	# OF RECORDS-ENTRY
MISCELLANEOUS ENTRIES		0-255	
	HALT/LOAD	0	2
	TIME/DATE CHANGE	1	1
	SYSTEM OVERHEAD	2	2
	OPERATOR INPUT MESSAGE	3	2-4 DEPENDING ON USE SIZE (14 WORDS OR LESS)
	SECURITY ERROR	16	2

	ENTRY	TYPE CODE	# OF RECORDS-ENTRY
JOB-ORIENTED ENTRIES		256-767	
	CONTROL CARD	256	2-4 DEPENDING ON SIZE OF CC. (14 WORDS OR LESS)
	SCHEDULED	257	2 OR MORE DEPENDING ON LENGTH OF JOB IDENTIFIER (43 WORDS OR LESS)
	BOJ	258	2 (IF NO SCHEDULE ENTRY FOR THIS JOB THEN 2 OR MORE DE-PENDING ON LENGTH OF JOB IDENTIFIER)
	PRIORITY CHANGE	272	1
	OPERATOR RSVP	273	3
	ABORTS	289	3
	EOJ	288	3
PERIPHERAL-ORIENTED ENTRIES		512-1023	
	FILE OPEN	512	2 OR MORE DEPENDING ON LENGTH OF JOB IDENTIFIER (43 WORDS OR LESS)
	I/O ERROR	513	2
	FILE CLOSE	544	2

WORD 0 AND WORD 1 ARE THE DATE AND TIME RESPECTIVELY. LOG ENTRIES ARE BACKWARDS-LINKED IN SEVERAL LISTS TO FACILITATE RETRIEVAL. (FOR EXAMPLE, ALL EOJ ENTRIES ARE LINKED, ALL ENTRIES PERTAINING TO A PARTICULAR JOB ARE LINKED). THUS, ALL LINKS ARE CONTAINED IN THE LAST RECORD OF EACH ENTRY AND THE ENTRY TYPE CODE IS THE LAST WORD OF THE LAST RECORD OF EACH ENTRY REGARDLESS OF THE LENGTH OF THE ENTRY. EVERY

LINK CONTAINS TWO PARTS: THE LOG SERIAL NUMBER (SEE LOG RELEASE), AND THE ZERO-RELATIVE ADDRESS OF THE LAST RECORD OF THE ENTRY TO WHICH THE LINK POINTS. A ZERO-LINK INDICATES THE TERMINUS OF THE LIST. THE LINKS FOR THE "SAME-ENTRY-TYPE" LIST ARE ALWAYS THE NEXT TO LAST WORD OF THE LAST RECORD OF EACH ENTRY.

THE FOLLOWING TABLE LISTS THE VARIABLE PARTS OF ENTRY FORMATS.

	WORD	FIELD	CONTENTS
HALT/LOAD	2		SYSTEM NUMBER (INITIAL ENTRY ONLY)
	3		MCP LEVEL
	4		MCP OPTIONS INCLUDED
	5	47:4	CPU CONFIGURATION
	5	43:4	MPXR CONFIGURATION
	5	39:8	DCP CONFIGURATION
	5	31:32	MEMORY MODULE CONFIGURATION
	6-9		RESERVED
	10		LINK TO PREVIOUS HALT/LOAD
	11	23:24	ENTRY TYPE
		47:24	ENTRY LENGTHS IN RECORD
TIME/DATE CHANGE	2		NEW DATE
	3		NEW TIME
	4		LINK TO PREVIOUS TIME/DATE CHANGE
	5	23:24	ENTRY TYPE
		47:24	ENTRY LENGTH IN RECORDS
SYSTEM OVERHEAD	2		AVAILABLE CORE
	3		SAVE CORE USAGE
	4		OVERLAY CORE USAGE
	5		OVERLAY DISK SPACE USAGE
	6		OVERLAY TIME
	7		PROCESSOR TIME

	WORD	FIELD	CONTENTS
	8		I/O TIME
	9		RESERVED
	10		LINK TO PREVIOUS SYSTEM OVERHEAD ENTRY
	11	23:24	ENTRY TYPE
		47:24	ENTRY LENGTH IN RECORDS
OPERATOR INPUT	2		INPUT (14 OR LESS WORDS)
	LAST WORD-2		LINK TO PREVIOUS ENTRY THIS JOB (IF APPLICABLE)
	LAST WORD-1		LINK TO PREVIOUS OPERATOR INPUT
	LAST WORD	47:24	NUMBER OF RECORDS THIS ENTRY
		23:24	ENTRY TYPE
SECURITY ERROR	2		INFILTRATOR (JOB ID, IF APPLICABLE, OR USER ID.)
	3		PASSWORD USED
	4		METHOD OF ATTEMPTED ACCESS
	5		TERMINAL ADDRESS OF ACCESSOR
	6-8		RESERVED
	9		LINK TO PREVIOUS ENTRY THIS JOB (IF APPLICABLE)
	10		LINK TO PREVIOUS ENTRY THIS TYPE
	11	23:24	ENTRY TYPE
		47:24	ENTRY LENGTH IN RECORDS
CONTROL CARD	2		CONTROL CARD INFO (14 WORDS OR LESS)
	LAST WORD-3	47:24	PRIORITY
		23:24	MIXID
	LAST WORD-2		LINK TO PREVIOUS ENTRY THIS JOB
	LAST WORD-1		LINK TO PREVIOUS CC ENTRY
	LAST WORD	47:24	NUMBER OF RECORDS THIS ENTRY
	LAST WORD	23:24	ENTRY TYPE
SCHEDULE	2		JOB ID. (43 WORDS OR LESS)
	LAST WORD-3	47:24	PRIORITY

	WORD	FIELD	CONTENTS
		23:24	MIX ID
	LAST WORD-2		LINK TO PREVIOUS ENTRY THIS JOB
	LAST WORD-1		LINK TO PREVIOUS ENTRY THIS TYPE
	LAST WORD	47:24	NUMBER OF RECORDS THIS ENTRY
		23:24	ENTRY TYPE
BOJ	2		JOB ID. (ONLY IF NO SCHEDULE ENTRY EXISTS FOR THIS JOB)
	LAST WORD-3	47:24	PRIORITY
		23:24	MIX ID
	LAST WORD-2		LINK TO PREVIOUS ENTRY THIS JOB
	LAST WORD-1		LINK TO PREVIOUS ENTRY THIS TYPE
	LAST WORD	47:24	NUMBER OF RECORDS THIS ENTRY
		23:24	ENTRY TYPE
PRIORITY CHANGE	2	47:24	NEW PRIORITY
	2	23:24	MIX ID
	3		LINK TO PREVIOUS ENTRY THIS JOB
	4		LINK TO PREVIOUS ENTRY THIS TYPE
	5	23:24	ENTRY TYPE
		47:24	ENTRY LENGTH IN RECORDS
OPERATOR RSVP	2		AVAILABLE CORE
	3		OVERHEAD SAVE CORE USAGE
	4		OVERHEAD OVERLAY CORE USAGE
	5		OVERHEAD OVERLAY DISK SPACE
	6		OVERHEAD OVERLAY TIME
	7		OVERHEAD PROCESSOR TIME
	8		OVERHEAD I/O TIME
	9		ATTENTION TYPE
	10		RESPONSE TYPE
	11-13		RESERVED
	14	23:24	MIXID
	15		LINK TO PREVIOUS ENTRY THIS JOB

	WORD	FIELD	CONTENTS
	16		LINK TO PREVIOUS ENTRY THIS TYPE
	17	23:24	ENTRY TYPE
		47:24	ENTRY LENGTH IN RECORDS
EOJ	2		AVAILABLE CORE
	3		SAVE CORE USAGE
	4		OVERLAY CORE USAGE
	5		OVERLAY DISK SPACE USAGE
	6		OVERLAY TIME
	7		PROCESSOR TIME
	8		I/O TIME
	9-12		RESERVED
	13		LINK TO ENTRY CONTAINING JOB ID. (SCHEDULE OR BOJ ENTRY)
	14	47:24	PRIORITY
		23:24	MIXID
	15		LINK TO PREVIOUS ENTRY THIS JOB
	16		LINK TO PREVIOUS ENTRY THIS TYPE
	17	23:24	ENTRY TYPE
		74:24	ENTRY LENGTH IN RECORDS
ABORT	2		AVAILABLE CORE
	3		SAVE CORE
	4		OVERLAY CORE
	5		OVERLAY DISK
	6		OVERLAYTIME
	7		PROCEDURETIME
	8		I/O TIME
	9	13:14	SEGMENT ABORTION OCCURRED
		32:13	WORD ABORTION OCCURRED
		35:3	SYLLABLE ABORTION OCCURRED
	10		ABORT REASON. (3 WORDS OR LESS DEPENDING ON CAUSE OF ABORTION)
	13		LINK TO ENTRY CONTAINING JOB ID. (SCHEDULE OR BOJ ENTRY)

	WORD	FIELD	CONTENTS
	14	47:24	PRIORITY
		23:24	MIX ID
	15		LINK TO PREVIOUS ENTRY THIS JOB
	16		LINK TO PREVIOUS ENTRY THIS TYPE
	17	23:24	ENTRY TYPE
		47:24	ENTRY LENGTHS IN RECORDS
FILE			
OPEN		47:1	1 IF STACK TYPE
	2		UNIT MNEMONIC OR DISK FILE TYPE
	3		PHYSICAL SERIAL OR DISK PACK NO.
	4	47:6	NUMBER OF BUFFERS
		37:19	MAXIMUM BLOCK SIZE IN WORDS
		18:19	MAXIMUM BLOCK SIZE IN WORDS
	5		FILE ID. (43 WORDS OR LESS)
	6-8		RESERVED
	LAST WORD-3		LINK TO PREVIOUS ENTRY THIS UNIT
	LAST WORD-2		LINK TO PREVIOUS ENTRY THIS JOB
	LAST WORD-1		LINK TO PREVIOUS ENTRY THIS TYPE
	LAST WORD	47:24	NUMBER OF RECORDS THIS ENTRY
		23:24	ENTRY TYPE
I/O ERROR	2		POINTER TO CORRESPONDING ENTRY IN MAINTENANCE LOG
	3	47:8	UNIT NUMBER
		39:8	UNIT TYPE
	4-7		RESERVED
	8		LINK TO PREVIOUS ENTRY THIS FILE
	9		LINK TO PREVIOUS ENTRY THIS UNIT
	10		LINK TO PREVIOUS ENTRY THIS TYPE
	11		ENTRY TYPE
FILE CLOSE	2		TRANSACTION COUNT
	3		I/O TIME (INCLUDES DISK SPACE IF APPLICABLE)
	4-6		RESERVED

WORD	FIELD	CONTENTS
7		LINK TO FILE OPEN THIS FILE
8		LINK TO PREVIOUS ENTRY THIS FILE
9		LINK TO PREVIOUS ENTRY THIS JOB
10		LINK TO PREVIOUS ENTRY THIS TYPE
11		ENTRY TYPE

B-2. MAINTENANCE LOG

EACH PHYSICAL BLOCK OF THIS LOG CONTAINS 30 WORDS, DIVIDED INTO TWO 15-WORD RECORDS. EACH ENTRY CONSISTS OF AT LEAST ONE RECORD, AND IN ADDITION, DEPENDING ON THE TYPE OF ERROR, ENOUGH RECORDS TO ADEQUATELY CONTAIN DATA WHICH HAS BEEN TRANSFERRED OR IS TO BE TRANSFERRED. THE FORMAT OF THE MAINTENANCE LOG IS SHOWN IN TABLE R2-1. THIS LOG HAS A 30-WORD PHYSICAL BLOCK DIVIDED INTO TWO 15-WORD RECORDS. EACH ENTRY CONSISTS OF AT LEAST 1 RECORD AND IN ADDITION, DEPENDING ON ERROR TYPE, ENOUGH RECORDS TO ADEQUATELY CONTAIN DATA TRANSFERRED OR TO BE TRANSFERRED.

WORD	FIELD	CONTENTS
0		DATE
1		TIME
2		ERROR TYPE
3		UNIT NUMBER
4		UNIT TYPE
5	7:8	FRAME SIZE
	17:3	DENSITY
6		TRANSACTION COUNT
7		RETRIES (IF APPLICABLE)
8		REEL NO. OR ROW (IF DISK)
9	39:20	BLOCK NUMBER IN REEL OR ROW
	19:20	RECORD NUMBER WITH IN BLOCK
10		RESULT DESCRIPTOR
11		IOCW
12	39:20	NUMBER OF SUCCEEDING RECORDS

WORD	FIELD	CONTENTS
		CONTAINING DATA TO BE TRANSFERRED
	19:20	NUMBER OF SUCCEEDING RECORDS CONTAINING DATA TRANSFERRED.
13		POINTER TO CORRESPONDING ENTRY SYSTEM LOG
14		RESERVED

THE ENTRIES ARE BACKWARDS LINKED BY UNIT

ERROR TYPES ARE:

0 - DESCRIPTOR ERROR
1 - INVALID MEMORY ADDRESS
2 - I/O MEMORY PARITYDRESS
3 - MEMORY PROTECT ERRORS
4 - PARITY
5 - WRITE LOCKOUT