```
XREF PROGRAM OF  3DC70 THIS LISTING PRODUCED  19 DEC 1973 TIME 10.28
$ CARD DOCUMENT FINAL DOCONLY SIX SINGLE

CARD FILE IS 0000000/CARD
```

B5500/B5700 SYSTEM OPERATION GUIDE

SECTION 1

INTRODUCTION

1.1 B5500/B5700 DESIGN CONCEPTS

THE CURRENT B5500/B5700 HARDWARE AND SOFTWARE SYSTEMS ARE THE RESULT OF MANY YEARS OF EVOLUTIONARY DEVELOPMENT UNDER THE BURROUGHS APPROACH TO COMPUTER SYSTEM DESIGN. THIS APPROACH IS TO TOTALLY INTEGRATE BOTH HARDWARE AND SOFTWARE THROUGH A POLICY OF CROSS-TRAINING THE DESIGN SPECIALISTS. HARDWARE ENGINEERS LEARN THE INTRICACIES OF SOFTWARE ARCHITECTURE, AND SOFTWARE SPECIALISTS LEARN THE SUBTLETIES OF HARDWARE LOGICAL DESIGN. THESE PEOPLE ARE THEN MERGED INTO A SINGLE SYSTEM DESIGN TEAM.

1.2 DEVELOPMENT AND INSTALLATION HISTORY

IN THE LATE 1950-S, SUCH A TEAM PRODUCED THE HARDWARE AND SOFTWARE DESIGN SPECIFICATIONS FOR THE BURROUGHS B5000 COMPUTER SYSTEM. BY 1962 THE FIRST B5000-S WERE INSTALLED USING THE ORIGINAL SOFTWARE OPERATING SYSTEM CALLED THE MASTER CONTROL PROGRAM (MCP). AT THIS TIME THE B5000 WAS USED ONLY FOR BATCH PROCESSING, THE ACCESS MODE FOR WHICH IT WAS DESIGNED. IN 1965, THE HARDWARE CONFIGURATION WAS EXPANDED. THE MOST SIGNIFICANT CHANGE WAS THE REPLACEMENT OF AUXILARY DRUM STORAGE WITH DISK STORAGE. A REVISED SOFTWARE OPERATING SYSTEM, CALLED THE DISK FILE MASTER CONTROL PROGRAM (DFMCP), WAS SUPPLIED TO UTILIZE THE NEW STORAGE MEDIUM. BATCH PROCESSING WAS STILL THE ONLY MODE OF ACCESS. AT THIS TIME, BURROUGHS CHANGED THE NAME OF THE SYSTEM FROM THE B5000 TO THE B5500.

IN 1966, THE HARDWARE CONFIGURATION WAS AGAIN EXPANDED. THIS TIME THE MOST SIGNIFICANT CHANGE WAS THE ADDITION OF TELEPHONE INTERFACE EQUIPMENT. A REVISED OPERATING SYSTEM, CALLED THE DATA COMMUNICATIONS MASTER CONTROL PROGRAM (DCMCP), WAS SUPPLIED BY BURROUGHS TO SUPPORT THE REMOTE TERMINAL ACCESS MODE IN ADDITION TO THE USUAL BATCH PROCESSING. THESE EVENTS MADE AVAILABLE FOR THE FIRST TIME THE NEW DIMENSION OF REMOTE COMPUTING. THE FEATURES OF THE ORIGINAL MASTER CONTROL PROGRAM SOFTWARE WERE BROAD ENOUGH TO REQUIRE ONLY A MODEST AMOUNT OF ALTERATION TO PRODUCE THE DCMCP. ALTHOUGH THE DCMCP PERMITTED REMOTE USERS TO

INTERACT AND CONVERSE WITH THEIR PROGRAMS, IT TREATED REMOTE JOBS MORE OR LESS THE SAME WAY IT TREATED BATCH JOBS. THIS CAUSED SOME REMOTE USERS TO SIT IDLE FOR EXTENDED PERIODS OF TIME BEFORE THEIR JOBS WERE INITIATED; HOWEVER, ONCE THE JOB WAS INITIATED, THE RESPONCE TIME WAS USUALLY SATISFACTORY. PSYCHOLGICALLY, THIS LEFT SOMETHING TO BE DESIRED. ONE SOLUTION DEVELOPED FOR THIS PROBLEM WAS TIME SLICING, OR TRUE TIME SHARING.

THE B5500 TIME SHARING SYSTEM, RELEASED IN 1969, ELIMINATED THIS INADEQUACY. THE SOFTWARE OPERATING SYSTEM WAS MODIFIED AND CALLED THE TIME SHARING MASTER CONTROL PROGRAM (TSMCP). SINCE BOTH THE DCMCP AND THE TSMCP EVOLVED FROM THE SAME ORIGIN, THEY BOTH CONTAIN THE SAME IMPORTANT FEATURES OF THE ORIGINAL DESIGN, AND THEY DIFFER ONLY IN CERTAIN RESOURCE MANAGEMENT STRATEGIES.

IN 1970, BURROUGHS ANNOUNCED THE AVAILABILITY OF CERTAIN ADDITIONAL HARDWARE FOR THE B5500, SUCH AS EXTENDED CORE MEMORY AND A DATA COMMUNICATIONS PROCESSOR. NEWLY INSTALLED B5500 SYSTEMS WITH THESE HARDWARE FEATURES ARE CALLED B5700 SYSTEMS. SUBSEQUENT TO THE B5700 ANNOUNCEMENT, SOME NEW BURROUGHS PUBLICATIONS AND SEVERAL REVISIONS TO OLDER MANUALS USED THE TERM B5700 IN THEIR TITLES. AT THE PRESENT TIME, ALL SOFTWARE AND PROGRAMMER REFERENCE MANUALS WITH EITHER B5500 OR B5700 IN THEIR TITLES ARE PERTINENT.

1.3 DESIGN FEATURES

THE ORIGINAL B5000 SYSTEM CONTAINED MANY SIGNIFICANT FEATURES, PRIMARILY AS A RESULT OF THE UNIFIED HARDWARE AND SOFTWARE DESIGN APPROACH. ALL THESE ORIGINAL FEATURES ARE STILL PRESENT IN THE SYSTEM AS IT NOW EXISTS, ALTHOUGH MANY CHANGES AND ENHANCEMENTS HAVE BEEN MADE IN THE INTEREST OF USER CONVENIENCE AND OPERATIONAL EFFICIENCY. SOME OF THE MORE SIGNIFICANT FEATURES WILL BE DESCRIBED RATHER BRIEFLY.

1.3.1 THE MASTER CONTROL PROGRAM (MCP)

ONE OF THE B5000 DESIGN OBJECTIVES WAS A SYSTEM CAPABLE OF CONTROLLING ITS OWN RESOURCES AND SCHEDULING WORK ON A DYNAMIC BASIS. THIS WAS ACCOMPLISHED WITH THE OPERATING SYSTEM SOFTWARE, ORIGINALLY CALLED THE MASTER CONTROL PROGRAM, OR MCP. LATER, AS SIGNIFICANT CAPABILITIES WERE ADDED TO THE MCP, IT WAS CALLED THE DFMCP, THE DCMCP, AND THE TSMCP. THE MCP IS THE CONTROLLING TRAFFIC DIRECTOR OF THE ENTIRE SYSTEM. IT ALWAYS IS IN CONTROL OF THE HARDWARE, OR HAS THE MEANS OF REGAINING CONTROL. ITS MAIN FUNCTION IS TO PERMIT THE HARDWARE TO PROCESS ALL USER JOBS PRESENTED TO IT BY THE OPERATOR. FURTHERMORE, IT MUST PERFORM ALL OF THE MANY, MANY TASKS ASSOCIATED WITH JOB PROCESSING AS EFFICIENTLY AS POSSIBLE. TO BE EFFICIENT IT MUST BE COGNIZANT OF THE RESOURCES AVAILABLE AND THE RESOURCES NEEDED BY EACH JOB, AND BE ABLE TO ALLOCATE AND DEALLOCATE RESOURCES TO THE JOB STREAM AS QUICKLY AS POSSIBLE.

1.3.2 HARDWARE STATUS MONITORING

IN MANY OPERATING SYSTEMS THE AVAILABLE RESOURCES (HARDWARE CONFIGURATION) ARE SPECIFIED AS PARAMETERS WHEN THE SOFTWARE IS ASSEMBLED FOR LOADING (SYSTEM GENERATION). IN SUCH SYSTEMS, IF THE HARDWARE CONFIGURATION SHOULD CHANGE EITHER BY ADDING MORE DEVICES SUCH AS CORE MEMORY, TAPE UNITS, CARD READERS, ETC., OR BY REMOVING SOME DEVICE, POSSIBLY FOR TEMPORARY MAINTENANCE, THE OPERATING SYSTEM MAY REQUIRE PARTIAL OR COMPLETE REGENERATION. IN BURROUGHS SYSTEMS, THE MCP CONTINUALLY CHECKS THE STATUS OF THE TOTAL SYSTEM. IT MAINTAINS COMPREHENSIVE BUT COMPACT AVAILABILITY TABLES THAT INDICATE THE NUMBER OF TAPES, DISKS, PRINTERS, I/O CHANNELS, AND OTHER DEVICES WHICH ARE AVAILABLE AND READY. CHANGES IN STATUS ARE EASILY DETECTED BECAUSE OF THE HIGH DEGREE OF HARDWARE-SOFTWARE INTEGRATION.

1.3.3 "FAIL-SOFT" CAPABILITY

THIS DYNAMIC ASSESSMENT OF CURRENT RESOURCES PERMITS A FAIL-SOFT CAPABILITY; I.E., MALFUNCTIONING DEVICES CAN BE DYNAMICALLY REMOVED FOR MAINTENANCE AND THE SYSTEM WILL CONTINUE TO FUNCTION. OF COURSE, THERE ARE FEWER RESOURCES AND THROUGHPUT MAY DECREASE, BUT, NEVERTHELESS, THE MACHINE WILL CONTINUE TO FUNCTION. A MEMORY MODULE, AN I/O CHANNEL, A PERIPHERAL DEVICE, OR EVEN A CENTRAL PROCESSOR IN A TWO PROCESSOR SYSTEM, CAN BE TAKEN OFF-LINE AND THE MCP WILL REROUTE THE WORK AROUND THE MISSING COMPONENTS; CERTAIN DEVICES MAY EVEN BE REMOVED WHILE THE SYSTEM IS OPERATING WITHOUT CAUSING A SYSTEM FAILURE.

1.3.4 DYNAMIC RESOURCE MANAGEMENT

PROVIDED WITH UP-TO-DATE RESOURCE AVAILABILITY TABLES, THE MCP IS ABLE TO DYNAMICALLY ASSIGN CERTAIN RESOURCES TO A PARTICULAR USER JOB. RATHER THAN, AT JOB INITIATION TIME, ASSIGNING ALL THE RESOURCES THAT WILL EVER BE NEEDED, SPECIFIC RESOURCES ARE DYNAMICALLY ASSIGNED AND DEASSIGNED TO EACH JOB AS REQUIRED. THIS DYNAMIC ASSIGNMENT IS ACCOMPILSHED NOT ONLY FOR PERIPHERAL DEVICES SUCH AS TAPES, PRINTERS, ETC., BUT ALSO FOR THE FLOATING I/O CHANNELS AND MAIN CORE MEMORY.

THIS MEANS THAT ONLY THAT PORTION OF MAIN MEMORY (THE MOST EXPENSIVE RESOURCE) WHICH IS ACTUALLY REQUIRED AT THAT INSTANT IS EVER ASSIGNED TO A JOB.

1.3.5 MULTIPROGRAMMING AND MULTIPROCESSING

BECAUSE MOST JOBS REQUIRE ONLY A PORTION OF THE TOTAL SYSTEM AT ANY GIVEN INSTANT (ESPECIALLY CORE MEMORY), THE MCP IS ABLE TO INITIATE SEVERAL JOBS CONCURRENTLY AND ASSIGN THEM DISJOINT SETS OF RESOURCES. WITH ONLY ONE OR TWO CENTRAL PROCESSORS, ONLY ONE OR TWO OF THE JOBS "IN THE MIX" MAY BE IN EXECUTION AT ANY GIVEN INSTANT; HOWEVER, THE PROCESSOR(S) MAY BE ASSIGNED TO DIFFERENT ACTIVE JOBS DURING BRIEF, DISJOINT INTERVALS OF TIME. THIS ABILITY TO HAVE MANY JOBS (OR PROGRAMS) CONCURRENTLY ACTIVE IS CALLED MULTIPROGRAMMING. IF A SYSTEM CONTAINS TWO OR MORE CENTRAL PROCESSORS AND IS ABLE TO EXECUTE SIMULTANEOUSLY A JOB IN EACH PROCESSOR, IT IS SAID TO BE CAPABLE OF BOTH MULTIPROCESSING AND MULTIPROGRAMMING. FURTHERMORE, THE SETS OF RESOURCES ASSIGNED TO THE ACTIVE JOBS NEED NOT BE DISJOINT. SINCE ASSIGNMENTS ARE MADE DYNAMICALLY, THEY NEED ONLY BE DISJOINT WITH RESPECT TO TIME. THIS LEADS DIRECTLY TO THE CONCEPT OF "TIME-SHARING" OF RESOURCES, ALTHOUGH THE CURRENT DISCUSSION PERTAINS TO A FEATURE OF THE BATCH PROCESSING MCP.

1.3.6 AUTOMATIC FILE RECOGINITION

A PRACTICAL NECESSITY OF THE MULTIPROGRAMMING NCP IS THE AUTHORITY TO MAKE SPECIFIC PERIPHERAL DEVICE ASSIGNMENTS. A ~~DESI~~RABLE CONSEQUENCE IS THAT THE PROGRAMMER NEED NOT, IN FACT CAN NOT, MAKE SPECIFIC UNIT ASSIGNMENTS. THE PROGRAMMER NEED ONLY SPECIFY THE TYPE OF DEVICE AND THEN ADDRESS ALL PROGRAM AND DATA FILES BY NAME ONLY, FOR EXAMPLE, IF A PROGRAM NEEDS TO CREATE A TAPE FILE, THE MCP ASSIGNS AN AVAILABLE TAPE UNIT AND THEREAFTER ASSOCIATES THAT UNIT WITH THE PROGRAMMER-S CHOICE OF FILE NAME. SIMILARILY, IF A PROGRAM REQUIRES SOME PARTICULAR TAPE FILE FOR INPUT, IT IS CALLED BY NAME ONLY, THE OPERATOR CAN MOUNT THE TAPE ON ANY AVAILABLE UNIT AND THE MCP AUTOMATICALLY RECOGNIZES THE FILE NAME AND MAKES THE UNIT ASSIGNMENT.

1.3.7 PERIPHERAL DEVICE INDEPENDENCE

AUTOMATIC FILE RECOGNITION ALSO FACILITATES THE REALIZATION OF TRUE DEVICE INDEPENDENCE. THIS PERMITS MAGNETIC TAPE AND DISK TO BE USED AS BACKUP, OR PSEUDO-DEVICES, FOR THE CARD READERS, LINE PRINTERS, AND CARD

PUNCH. INSTEAD OF FORCING A JOB TO WAIT FOR ACCESS TO A BUSY PERIPHERAL DEVICE, THE MCP AUTOMATICALLY ASSIGNS A TEMPORARY BACKUP DEVICE. THIS FEATURE PROVIDES A CONSIDERABLE INCREASE IN EFFICIENCY AND THROUGHPUT.

1.3.8 HOMOGENEOUS HARDWARE ARCHITECTURE

THE HARDWARE ARCHITECTURE WHICH MAKES MULTIPROGRAMMING AND DYNAMIC RESOURCE ALLOCATION PRACTICAL IS CENTERED AROUND TWO EXCHANGES - THE MEMORY EXCHANGE AND THE INPUT/OUTPUT (I/O) EXCHANGE. EACH CENTRAL PROCESSOR CAN ACCESS ANY OF THE EIGHT CORE MEMORY MODULES THROUGH THE MEMORY EXCHANGE. LIKEWISE ANY OF THE FOUR FLOATING I/O CHANNELS CAN ACCESS MEMORY WITHOUT INTERFERING WITH THE PROCESSORS. IN TURN, THE I/O CHANNELS CAN ACCESS ANY OF THE PERIPHERAL DEVICES THROUGH THE I/O EXCHANGE. THUS, BOTH PROCESSORS AND ALL I/O CHANNELS COULD SIMULTANEOUSLY ACCESS DIFFERENT MODULES OF MEMORY. THE EXCHANGES CONTAIN THE LOGIC NECESSARY TO FLOAT THE ACCESS REQUESTS TO A FREE I/O CHANNEL. EACH I/O CHANNEL CONTAINS THE LOGIC AND REGISTERS NECCESSARY TO AUTONOMOUSLY COMPLETE THE INFORMATION TRANSFER ONCE IT HAS BEEN INITIATED.

1.3.9 TYPICAL B5700 CONFIGURATION

A TYPICAL CONFIGURATION IS AS FOLLOWS:

1	CENTRAL PROCESSING UNIT
8	CORE MEMORY MODULES (4,096 WORDS EACH)
3	FLOATING I/O CHANNELS
5	MAGNETIC TAPE UNITS, 7-TRACK, 200/556/800 BPI
5	DISK STORAGE MODULES (1,200,000 WORDS EACH)
2	LINE PRINTERS (1100 LINES PER MINUTE)
1	CARD READER (1400 CARDS PER MINUTE)
1	CARD PUNCH (300 CARDS PER MINUTE)
1	OPERATOR CONSOLE (SPO)
16	TELEPHONE LINE ADAPTERS AND BUFFERS

1.3.10 HARDWARE STACK

THE MACHINE LANGUAGE INSTRUCTIONS OF MOST COMPUTER SYSTEMS CONSIST OF A COMMAND PART AND AN ADDRESS PART - THE COMMAND SPECIFIES THE ACTION AND THE ADDRESS SPECIFIES THE LOCATION OF THE OPERAND. INSTEAD OF THE USUAL FORMAT, THE BURROUGHS HARDWARE EXECUTES INSTRUCTIONS EXPRESSED IN "POLISH NOTATION" AND UTILIZES A PUSH-DOWN STACK. THE STACK CONSISTS OF TWO REGISTERS AND A CONTIGUOUS AREA OF CORE MEMORY. SOME INSTRUCTIONS CAUSE NEW OPERAND VALUES TO BE PLACED IN THE TOP OF THE STACK (THE REGISTERS) CAUSING OLDER VALUES TO BE PUSHED DOWN (INTO CORE MEMORY). OTHER INSTRUCTIONS REMOVE THE VALUE AT THE TOP CAUSING LOWER VALUES TO BE AUTOMATICALLY RAISED, THUS KEEPING THE TWO REGISTERS AT THE TOP EFFECTIVELY ALWAYS FULL. OTHER INSTRUCTIONS SPECIFY AN ARITHMETIC OPERATION TO BE PERFORMED, WHERE THE OPERANDS ARE BY DEFINITION THE VALUES IN THE TOP OF THE STACK.

1.3.11 POLISH NOTATION

FOR EXAMPLE, THE ALGOL ASSIGNMENT STATEMENT (JUST LIKE FORTRAN EXCEPT FOR THE ASSIGNMENT OPERATOR :=):

```
          D := (A+B)/C
```

WHEN EXPRESSED IN POLISH NOTATION BECOMES:

```
          AB+C/D:=
```

POLISH NOTATION ELIMINATES THE NEED FOR CONVENTIONAL RULES OF ARITHMETIC PRECEDENCE AND BRACKET GROUPING OF VALUES WITHIN EXPRESSIONS. THE RULE THAT APPLIES IS: FOLLOW TWO ARITHMETIC VALUES (THE OPERANDS) WITH THE OPERATION (THE COMMAND) THAT IS TO USE THOSE VALUES. THUS, BY DEFINITION, EVERY MATHEMATICAL OPERATION WORKS ON THE MOST RECENTLY OBTAINED PAIR OF OPERANDS WHICH ARE ALWAYS IN THE TWO REGISTERS AT THE TOP OF THE STACK. THE INSTRUCTIONS NEEDED TO EXECUTE THE ASSIGNMENT STATEMENT SHOWN ABOVE ARE AS FOLLOWS:

```
      PUSH     THE VALUE OF "A" INTO THE TOP OF THE STACK.
```

PUSH	THE VALUE OF "B" INTO THE TOP OF THE STACK.
ADD	DESTROYING THE VALUES OF "A" AND "B" AND LEAVING (A+ B) IN THE TOP OF THE STACK.
PUSH	THE VALUE OF "C" INTO THE TOP OF THE STACK.
DIVIDE	DESTROYING THE VALUES OF (A+B) AND "C" AND LEAVING THE QUOTIENT IN THE TOP OF THE STACK.
STORE	THE TOP OF THE STACK AS THE NEW VALUE OF "D", LEAVING THE STACK EMPTY.

1.3.12 REENTRANT CODE

THE B5500/B5700 MACHINE LANGUAGE INSTRUCTIONS MAY BE PLACED ANYWHERE IN CORE MEMORY AND ARE FETCHED SEQUENTIALLY BY A CENTRAL PROCESSOR FOR EXECUTION. THE VALUES OF SIMPLE VARIABLES, POINTERS TO THE LOCATIONS OF MULTI-DIMENSIONAL DATA ARRAYS, POINTERS TO SUBROUTINES, ETC., ARE KEPT SEPARATELY FROM THE INSTRUCTIONS AND THE STACK IN A CONTIGUOUS AREA OF CORE CALLED THE PROGRAM REFERENCE TABLE (PRT). NOTICE THAT THE INSTRUCTIONS NEED ONLY REFERENCE CERTAIN RELATIVE POSITIONS IN THE PRT (AND STACK). THIS MEANS THAT THE SAME SET OF INSTRUCTIONS COULD PRODUCE DIFFERENT RESULTS DEPENDING ON THE CONTENTS OF THE PRT. SUCH INSTRUCTIONS ARE SAID TO BE REENTRANT, MEANING THAT TWO OR MORE JOBS, EACH WITH ITS OWN SEPARATE PRT AND STACK, COULD BE CONCURRENTLY EXECUTING THE SAME SET OF INSTRUCTIONS. B5500 INSTRUCTIONS, OR CODE, ARE ALWAYS REENTRANT, YIELDING GREAT ECONOMY IN THE USE OF CORE MEMORY.

1.3.13 HARDWARE PRESENCE BIT CHECKING

A FURTHER ECONOMY IN THE USE OF CORE MEMORY IS ACCOMPLISHED BY A UNIQUE HARDWARE FEATURE CALLED THE "PRESENCE BIT". A POINTER (DESCRIPTOR) TO EACH DATA ARRAY (AND CODE SEGMENT) IS KEPT IN EACH JOB-S PRT. WHEN AN INSTRUCTION ATTEMPTS TO CALL A VALUE FROM A DATA ARRAY INTO THE STACK (OR BRANCH TO A NEW CODE SEGMENT), THE CENTRAL PROCESSOR EXAMINES THE PRESENCE BIT IN THE DESCRIPTOR. IF THE BIT INDICATES THE DATA ARRAY IS CURRENTLY LOCATED IN CORE, THAT VALUE IS OBTAINED AND

EXECUTION CONTINUES. IF THE PRESENCE BIT INDICATES IT IS NOT PRESENT, THE PROGRAM IS SUSPENDED AND THE MCP IS AUTOMATICALLY CALLED. FROM THE DESCRIPTOR THE MCP DETERMINES WHERE THE ARRAY IS LOCATED ON DISK, COPIES THE ARRAY INTO SOME AVAILABLE CORE AREA, PLACES THE ADDRESS OF THIS AREA IN THE DESCRIPTOR, AND TURNS THE PRESENCE BIT ON. THE SAME INSTRUCTION IS THEN REEXECUTED. BY THIS MEANS, ALL DATA ARRAYS (AND ALL CODE SEGMENTS) FOR A JOB NEED NOT BE PRESENT IN CORE AT THE SAME TIME. THIS PERMITS THE MCP CONSIDERABLE LATITUDE IN ALLOCATING AND DEALLOCATING CORE SPACE AS A FUNCTION OF TIME, THUS MINIMIZING THE TOTAL CORE MEMORY RESOURCE REQUIREMENTS.

1.3.14 VIRTUAL MEMORY

THE HARDWARE PRESENCE BIT AND THE MCP-S ABILITY TO DYNAMICALLY MANAGE CORE SPACE PROVIDE THE PROGRAMMER WITH A LUXURY CALLED "VIRTUAL MEMORY". THIS MEANS THAT THE TOTAL CORE REQUIREMENTS FOR A PROGRAM MAY BE FAR IN EXCESS OF THE ACTUAL CORE INSTALLED AND THE PROGRAM WILL STILL RUN. FURTHERMORE, THIS IS ACCOMPLISHED COMPLETELY AUTOMATICALLY BY THE SYSTEM WITHOUT PROGRAMMER AWARENESS. THE MCP ALLOCATES AND DEALLOCATES CORE SPACE IN EXACT AREA SIZES DEMANDED BY THE PROGRAMS, AND THE HARDWARE GUARANTEES THE DATA OR CODE IS ACTUALLY PRESENT BEFORE IT IS REFERENCED. THE ONLY REMAINING REQUIREMENT IS AUTOMATIC PROGRAM SEGMENTATION WHICH IS ACCOMPLISHED BY THE COMPILERS, OR LANGUAGE PROCESSORS.

1.3.15 ONE-PASS COMPILERS

THE FUNCTION OF A LANGUAGE PROCESSOR IS TO CONVERT A PROGRAM WRITTEN IN SOME HIGH-LEVEL LANGUAGE, SUCH AS ALGOL, BASIC, COBOL, OR FORTRAN INTO MACHINE LANGUAGE INSTRUCTIONS AND AT THE SAME TIME ORGANIZE THE DATA TO BE USED AS OPERANDS FOR THOSE INSTRUCTIONS. DEPENDING ON THE NATURE OF THE MACHINE LANGUAGE INSTRUCTIONS AND HARDWARE ORGANIZATION, THE COMPILATION PROCESS CAN SOMETIMES BY QUITE TEDIOUS AND TIME CONSUMING, REQUIRING MANY PASSES THROUGH THE SOURCE LANGUAGE STATEMENTS TO FILL IN ALL THE CORE ADDRESSES AND ORGANIZE THE DATA. THE RESULTING OBJECT CODE IS OFTEN NOT AS EFFICIENT AS IT MIGHT HAVE BEEN IF THE PROGRAMMER HAD USED SOME LOWER LEVEL LANGUAGE, SUCH AS AN ASSEMBLY LANGUAGE (SIMILAR TO MACHINE LANGUAGE). HOWEVER, THE INTEGRATED HARDWARE-SOFTWARE DESIGN OF THE B5500 PERMITS EXTREMELY FAST ONE-PASS COMPILERS TO USUALLY PRODUCE THE MOST EFFICIENT OBJECT CODE POSSIBLE.

1.3.16 AUTOMATIC PROGRAM SEGMENTATION

THE COMPILER ORIENTED HARDWARE ONLY REQUIRES THE COMPILERS TO PRODUCE MACHINE LANGUAGE INSTRUCTIONS IN POLISH NOTATION, WHICH IS, ITSELF, A HIGH LEVEL LANGUAGE. THE HARDWARE PUSH-DOWN STACK AUTOMATICALLY HOLDING INTERMEDIATE RESULTS, THE HARDWARE INDIRECTLY ADDRESSING THROUGH THE PRT, THE HARDWARE CHECKING PRESENCE-BITS, AND THE SOFTWARE IN THE MCP MANAGING CORE SPACE, ALL CONTRIBUTE TO FURTHER SIMPLIFY THE COMPILERS TASKS. EACH COMPILER CONTRIBUTES TO CORE MANAGEMENT EFFICIENCY BY AUTOMATICALLY SEGMENTING EACH PROGRAM ON THE LOGICAL BOUNDARIES OF THE LANGUAGE; FOR EXAMPLE ALGOL ACCORDING TO BLOCKS, COBOL ACCORDING TO PARAGRAPHS, AND FORTRAN ACCORDING TO SUBROUTINES.

1.3.17 SYSTEM SOFTWARE

WITH FAST, EFFICIENT COMPILERS THERE IS NO NEED FOR PROGRAMMERS TO USE ANYTHING BUT HIGH LEVEL LANGUAGES, SUCH AS ALGOL, FORTRAN, COBOL, OR BASIC. EACH DIFFERENT LANGUAGE HAS ITS OWN UNIQUE FEATURES AND ITS OWN GROUP OF DEVOTEES. HOWEVER, SOME FEATURES OF ALGOL, SUCH AS THE ABILITY OF A PROCEDURE (SUBROUTINE) TO CALL ITSELF RECURSIVELY, COUPLED WITH THE ABILITY OF THE STACK TO HOLD A LARGE NUMBER OF INTERMEDIATE RESULTS, MAKE IT AN EXTREMELY ATTRACTIVE LANGUAGE IN WHICH PROBLEM SOLUTIONS MAY BE STATED VERY CONCISELY. FOR THIS REASON, BURROUGHS CHOSE ALGOL AS THE LANGUAGE IN WHICH TO WRITE ALL THE SYSTEM SOFTWARE WHICH HAS EVOLVED; FOR ALL PRACTICAL PURPOSES, AN ASSEMBLY LANGUAGE WAS NOT USED. NOT ONLY IS THE MCP WRITTEN IS ESPOL (A SPECIAL DIALECT OF ALGOL) BUT ALL OF THE COMPILERS, INCLUDING THE ALGOL COMPILER ITSELF, ARE WRITTEN IN ALGOL THUS, ALL THE SYSTEM SOFTWARE IS QUITE CONCISELY DOCUMENTED AND QUITE EFFICIENT.

1.3.18 COMPREHENSIVE ACCOUNTING

A VERY NECESSARY FEATURE OF THE MULTIPROGRAMMING MCP IS THE ABILITY TO ACCOUNT FOR THE RESOURCES USED BY EACH INDIVIDUAL JOB THAT IS PROCESSED. A VERY COMPREHENSIVE LOGGING SYSTEM PROVIDES DETAILED INFORMATION CONCERNING PROCESSOR TIME, I/O CHANNEL TIME, PERIPHERAL DEVICE USAGE, AND DISK SPACE OCCUPIED.

1.4 TIME-SHARING FEATURES

IN ADDITION TO THE FEATURES ALREADY DESCRIBED, THE TIME SHARING SYSTEM CONTAINS TWO ADDITIONAL ONES. THE FIRST IS THE USE OF THE TIME

SHARING MASTER CONTROL PROGRAM (TSMCP) INSTEAD OF THE DATA COMMUNICATIONS MASTER CONTROL PROGRAM (DCMCP). THE SECOND IS THE COMMAND AND EDIT (CANDE, PRONOUNCED CANDY) LANGUAGE PROCESSOR.

1.4.1 RESPONSE TIME

THE MOST SERIOUS DISADVANTAGE OF THE DCMCP IS THE FACT THAT IT PROCESSES REMOTE JOBS IN THE SAME MANNER AS BATCH JOBS. ITS BATCH PROCESSING CAPABILITIES ARE EXCELLENT, PROVIDING HIGH UTILIZATION OF ALL RESOURCES AND GOOD THROUGHPUT VIA MULTIPROGRAMMING. WHEN THE CORE MEMORY REQUIREMENTS OF THE ACTIVE BATCH JOBS REACH A CERTAIN THRESHOLD, THE DCMCP REFUSES TO INITIATE ADDITIONAL JOBS IN ORDER TO AVOID AN EXCESSIVE OVERLAY SITUATION. A CERTAIN AMOUNT OF CORE MEMORY OVERLAY IS HEALTHY, BUT AN EXCESSIVE AMOUNT IS HIGHLY INEFFICIENT. FOR THIS REASON THE DCMCP IS QUITE EFFICIENT OVER LONG PERIODS OF TIME AND GIVES GOOD RESPONSE TO THE ACTIVE JOBS, BUT MAY FORCE BOTH BATCH AND REMOTE JOBS TO REMAIN UNINITIATED FOR UNCERTAIN PERIODS OF TIME. EVEN A FEW MINUTES OF COMPLETE INACTIVITY CAN BE MOST FRUSTRATING TO A REMOTE USER. THUS, THE UNPREDICTABLE RESPONSE TIME OF THE DCMCP TO REMOTE USERS DEMANDS IS ITS MOST SERIOUS DEFICIENCY. THE TSMCP OVERCOMES THIS DEFICIENCY BY PROVIDING A PREDICTABLE RESPONSE TIME.

1.4.2 TIME SLICING

EACH REMOTE USER DESIRES FAST AND PREDICTABLE RESPONSES. IN FACT, EACH REMOTE USER DESIRES TO HAVE ALL SYSTEM RESOURCES MADE AVAILABLE TO HIM WHENEVER HE NEEDS THEM; I.E., TO HAVE SOLE USE OF THE ENTIRE SYSTEM. HOWEVER IDEAL THIS MIGHT BE FROM THE VIEWPOINT OF THE REMOTE USER, IT IS HIGHLY INEFFICIENT AND IMPRACTICAL. THE TSMCP ATTEMPTS TO CREATE THE SOLE-USER ILLUSION TO MANY REMOTE USERS BY PROVIDING EACH USER WITH REASONABLY FAST, PREDICTABLE, AND EQUITABLE RESPONSES. EACH REMOTE JOB IS EXECUTED FOR A SHORT PERIOD, OR SLICE, OF TIME. IT IS THEN ROLLED OUT (SWAPPED) FROM CORE MEMORY TO DISK AND THE NEXT JOB IS ROLLED IN AND EXECUTED DURING ITS TIME SLICE. THREFORE, EACH REMOTE USER RECEIVES HIS FAIR SHARE OF TIME, AND WITHIN A BRIEF CYCLE OF TIME, ALL USERS RECEIVE A RESPONSE FROM THE SYSTEM. ALL OF THE TIME REQUIRED FOR THE TSMCP TO ROLL JOBS IN AND OUT OF CORE IS, IN MANY CASES, WASTED TIME. HOWEVER, THIS INEFFICIENCY IS THE PRICE THAT MUST BE PAID FOR PREDICTABLE RESPONSE TIME.

1.4.3 THE TIME SLICE ALGORITHM

THE NUMBER OF SUCCESSIVE TIME SLICES REQUIRED TO COMPLETE A GIVEN TASK DEPENDS ON THE NATURE AND MAGNITUDE OF THE TASK. SIMPLE TASKS, SUCH AS LISTING A SOURCE LANGUAGE STATEMENT OR ENTERING A DATA VALUE TO A PROGRAM CAN USUALLY BE COMPLETED DURING ONE TIME SLICE. MORE COMPLEX TASKS, SUCH AS INVERTING A LARGE MATRIX, MAY REQUIRE MANY, MANY TIME SLICES. THE TOTAL TIME REQUIRED TO CYCLE THROUGH ALL REMOTE JOBS AND GIVE EACH USER A RESPONSE DEPENDS ON THE NUMBER OF USERS AND WHETHER EACH JOB UTILIZES ITS FULL TIME SLICE. WHEN A JOB IS FIRST INITIATED (ENTERS THE MIX), IS IS ASSIGNED TO AN AREA IN CORE WHICH MINIMIZES CONFLICT WITH OTHER ACTIVE JOBS. IT IS THEN GIVEN AN IMMEDIATE TIME SLICE. IF NECESSARY, JOBS IN CORE ARE ROLLED OUT TO MAKE ROOM FOR THE NEW JOB, UNLESS THEY ARE ALSO GETTING THEIR FIRST TIME SLICE. IN THAT CASE, THE NEW JOB IS PLACED AT THE HEAD OF THE QUEUE OF JOBS WAITING FOR A TIME SLICE. THERE ARE FIVE MAIN CONDITIONS WHICH CAUSE A REMOTE PROGRAM TO BE ROLLED OUT:

1. THE PROGRAM IS INPUT LIMITED; I.E., IT IS WAITING FOR DATA FROM THE REMOTE TERMINAL.

2. THE PROGRAM IS OUTPUT LIMITED; I.E., IT HAS GENERATED ENOUGH DATA TO FILL THE DISK BUFFER AREA ASSIGNED TO IT.

3. THE PROGRAM HAS USED ITS TIME SLICE WITHOUT BECOMING INPUT OR OUTPUT LIMITED.

4. THE PROGRAM HAS REACHED COMPLETION.

5. THE PROGRAM IS FORCED OUT TO MAKE ROOM FOR AN ENTERING OR RE-ENTERING JOB.

1.4.4 PROCESSING PHILOSOPHIES

THIS SWAPPING FEATURE OF THE TSMCP IS THE MAJOR DIFFERENCE BETWEEN IT AND THE DCMCP. THE JOB PROCESSING PHILOSOPHY OF THE DCMCP MAY BE SUMMARIZED AS FOLLOWS:

1. DON-T INITIATE A NEW JOB UNLESS ITS INITIAL CORE REQUIREMENT CAN BE SATISFIED, BUT INITIATE AS MANY AS POSSIBLE.

2. ONCE INITIATED, DON-T INTERRUPT A JOB UNLESS IT NEEDS ASSISTANCE, SUCH AS AN I/O OPERATION, MORE CORE SPACE, OR JOB COMPLETION.

3. ONCE INTERRUPTED, DON-T REASSIGN ALL OF A JOB-S CORE SPACE BUT ONLY THOSE AREAS THAT CAN BE EASILLY RESTORED.

THE JOB PROCESSING PHILOSOPHY OF THE TSMCP MAY BE SUMMARIZED AS FOLLOWS:

1. ALWAYS INITIATE A NEW REMOTE JOB AS SOON AS IT IS PRESENTED.

2. ALWAYS INTERRUPT A JOB AT THE END OF ITS TIME SLICE.

3. ALWAYS GIVE EACH ACTIVE JOB ITS TIME SLICE IN CYCLIC ORDER.

4. ALWAYS MAKE CORE SPACE AVAILABLE FOR A NEW OR REENTERING JOB, EVEN IF ALL AREAS ASSIGNED TO OLDER JOBS NEED TO BE SWAPPED.

5. IF NONE OF THE REMOTE JOBS CAN UTILIZE THEIR TIME SLICES DURING A CYCLE, THEN DEVOTE A TIME SLICE TO A BATCH JOB.

FROM THESE TWO DESCRIPTIONS, IT SHOULD BE CLEAR THAT THE TSMCP, OR ANY OTHER TIME SHARING SYSTEM, IS INHERENTLY INEFFICIENT. THE TSMCP ATTEMPTS TO OVERCOME SOME OF THIS INEFFICIENCY BY DIVIDING CORE MEMORY INTO TWO PARTS SEPARATED BY A "FENCE". THE AREA BELOW THE FENCE, IN WHICH ONLY CERTAIN KINDS OF TASKS ARE RUN, IS MANAGED ACCORDING TO THE DCMCP PHILOSOPHY, WHILE THE AREA ABOVE THE FENCE IS MANAGED ACCORDING TO THE TSMCP PHILOSOPHY. NEVERTHELESS, INEFFICIENCY DOES EXIST, BUT THE COMPENSATION IS THE PREDICTABLE RESPONSE TIME REQUIRED IN ANY SATISFACTORY TIME SHARING SYSTEM.

SECTION 2 -- SECTION 7

SECTIONS 2 THROUGH 7

AT THE TIME OF THE RELEASE OF THE MARK XV.2 SYSTEM THESE SECTIONS WERE INCOMPLETE AND WERE NOT IN A FORM TO BE INCLUDED. THE MISSING SECTIONS WILL BE SUPPLIED AT A LATER DATE.

SECTION 8

MAINTAINING THE SYSTEM SOFTWARE

8.1 GENERAL

THIS SECTION DESCRIBES THE MATERIALS NEEDED AND THE PROCEDURES USED TO MAINTAIN THE SYSTEM SOFTWARE.

8.2 MATERIALS

THE FOLLOWING MATERIALS ARE REQUIRED FOR MAINTAINING THE SYSTEM SOFTWARE. EACH ITEM IS DISCUSSED IN MORE DETAIL IN A SUBSEQUENT PARAGRAPH.

1. SYSTEM NOTE.

2. SYSTEM RELEASE TAPES.

3. SUPPLEMENTARY PATCHES.

4. "PATCH/MERGE" DECKS.

8.2.1 SYSTEM NOTE

A SYSTEM NOTE DESCRIBING THE CHANGES TO THE SOFTWARE IS SUPPLIED WITH EACH RELEASE OF THE SYSTEM. THE SYSTEM NOTE SHOULD BE STUDIED CAREFULLY BEFORE ATTEMPTING TO COMPILE ANY OF THE SOFTWARE SINCE CHANGES IMPLEMENTED IN THE RELEASE WILL AFFECT THE OPERATION OF THE SYSTEM AND MAY ALTER THE PROCEDURES USED TO MAINTAIN THE SOFTWARE.

FOR CONVENIENCE, A COPY OF THE SYSTEM NOTE IS INCLUDED ON ONE OF THE RELEASE TAPES AS A PRINTER BACK-UP DISK FILE NAMED "PBD/SYSNOTE".

8.2.2 SYSTEM RELEASE TAPES

A SYSTEM RELEASE CONSISTS OF THREE "LIBRARY MAINTENANCE DUMP" TAPES. INCLUDED ON THESE TAPES ARE THE FOLLOWING ITEMS:

1. UPDATED SYMBOLIC FILES.

2. UPDATED OBJECT AND DATA FILES.

3. PATCHES TO THE PREVIOUSLY RELEASED SYMBOLIC FILES WHICH HAVE BEEN INCORPORATED INTO THE SOFTWARE; THEREBY, CREATING THE UPDATED SYMBOLIC FILES. THESE PATCHES ARE INCLUDED FOR DOCUMENTATION PURPOSES ONLY AND MUST NOT BE USED WHEN COMPILING THE SOFTWARE.

4. TEMPORARY PATCHES APPLICABLE TO THE UPDATED SOFTWARE.

5. PRINTER BACK-UP DISK FILE OF THE SYSTEM NOTE.

6. PRINTER BACK-UP DISK FILES OF ADDITIONAL DOCUMENTATION PERTINENT TO THE RELEASE.

THE FILES CONTAINED ON EACH OF THE RELEASE TAPES ARE LISTED BELOW. IT SHOULD BE NOTED THAT THE LOCATION OF A FILE MAY CHANGE, SHOULD THE NUMBER OF FILES OR THE SIZE OF THE FILES CHANGE FROM ONE RELEASE TO THE NEXT, DUE TO THE PHYSICAL LIMITATION OF THE SIZE OF A REEL OF TAPE.

THE FOLLOWING LIST OF SYMBOLIC FILES ARE LOCATED ON THE TAPE LABELED "SYMBOL1/FILE000".

FILE NAME	DESCRIPTION
SYMBOL/INTRINS	ESPOL SYMBOLIC ** SYSTEM INTRINSICS
SYMBOL/MCP	ESPOL SYMBOLIC ** DATACOM MCP
SYMBOL/TSSMCP	ESPOL SYMBOLIC ** TIME-SHARING MCP
SYMBOL/ALGOL	ALGOL SYMBOLIC ** ALGOL AND TSPOL COMPILERS
SYMBOL/BASIC	ALGOL SYMBOLIC ** BASIC COMPILER
SYMBOL/COBOL	ALGOL SYMBOLIC ** COBOL COMPILER
SYMBOL/COBOL68	ALGOL SYMBOLIC ** COBOL68 COMPILER
SYMBOL/ESPOL	ALGOL SYMBOLIC ** ESPOL COMPILER
SYMBOL/FORTRAN	ALGOL SYMBOLIC ** FORTRAN COMPILER
SYMBOL/XALGOL	ALGOL SYMBOLIC ** XALGOL COMPILER

THE FOLLOWING LIST OF SYMBOLIC FILES ARE LOCATED ON THE TAPE

LABELED "SYMBOL2/FILE000".

FILE NAME	DESCRIPTION
SYMBOL/AUXDATA	ALGOL SYMBOLIC ** AUXDATA/MAKER
SYMBOL/AUXTST	ESPOL SYMBOLIC ** AUXILARY MEMORY TEST
SYMBOL/CHECKAL	ALGOL SYMBOLIC ** CHECKAL/TEST
SYMBOL/COOL	ESPOL SYMBOLIC ** COOL AND COLD START ROUTINES
SYMBOL/DC1000	ALGOL SYMBOLIC ** DC1000/CODEGEN
SYMBOL/DCFILL	ALGOL SYMBOLIC ** DCFILL/PRT
SYMBOL/DSKDSK	ESPOL SYMBOLIC ** DISK TO DISK LOADER
SYMBOL/DUMPANL	ALGOL SYMBOLIC ** DUMP/ANALYZE
SYMBOL/KERNEL	ESPOL SYMBOLIC ** HALT LOAD KERNEL ROUTINE
SYMBOL/LOGAN	ALGOL SYMBOLIC ** LOGAN/DISK
SYMBOL/LOGOUT	ALGOL SYMBOLIC ** LOGOUT/DISK
SYMBOL/LOGOUTR	ALGOL SYMBOLIC ** LOGOUTR/DISK
SYMBOL/MAKCAST	ALGOL SYMBOLIC ** MAKCAST/DISK
SYMBOL/MASTEST	COBOL SYMBOLIC ** MASTER/TEST
SYMBOL/MEMDUMP	ESPOL SYMBOLIC ** MEMORY DUMP ROUTINE
SYMBOL/MESSGEN	ALGOL SYMBOLIC ** SYSTEM/MESSGEN
SYMBOL/MLOGAN	ALGOL SYMBOLIC ** LOGANL/MAINT
SYMBOL/OLMAINT	TSPOL SYMBOLIC ** ONLINE/MAINT
SYMBOL/PMERGE	XALGOL SYMBOLIC ** PATCH/MERGE
SYMBOL/ROTO	ALGOL SYMBOLIC ** ROTO/ROOTER
SYMBOL/STATS1	ALGOL SYMBOLIC ** STATS1/ANALYZE
SYMBOL/STATS2	ALGOL SYMBOLIC ** STATS2/ANALYZE
SYMBOL/STATS3	ALGOL SYMBOLIC ** STATS3/ANALYZE
SYMBOL/STATS4	ALGOL SYMBOLIC ** STATS4/ANALYZE
SYMBOL/SYSDISK	ALGOL SYMBOLIC ** SYSDISK/MAKER
SYMBOL/TAPEDSK	ESPOL SYMBOLIC ** TAPE TO DISK LOADER
SYMBOL/TPECNF	OLMAINT SYMBOLIC ** OTPECNF/MAINT
SYMBOL/TSDUMP	ALGOL SYMBOLIC ** TSDUMP/ANALYZE
SYMBOL/TSFILL	ALGOL SYMBOLIC ** TSFILL/PRT
SYMBOL/UPDATE	ALGOL SYMBOLIC ** UPDATE/USERS
SYMBOL/CANDE	TSPOL SYMBOLIC ** CANDE/TSHARER
SYMBOL/APPEND	TSPOL SYMBOLIC ** APPEND/CANDE
SYMBOL/COPY	TSPOL SYMBOLIC ** COPY/CANDE
SYMBOL/DELETE	TSPOL SYMBOLIC ** DELETE/CANDE
SYMBOL/FIND	TSPOL SYMBOLIC ** FIND/DISK
SYMBOL/GUARD	TSPOL SYMBOLIC ** GUARD/DISK
SYMBOL/HARD	TSPOL SYMBOLIC ** HARD/CANDE
SYMBOL/HELP	TSPOL SYMBOLIC ** HELP/DISK
SYMBOL/LFILES	TSPOL SYMBOLIC ** LFILES/CANDE
SYMBOL/LIST	TSPOL SYMBOLIC ** LIST/CANDE
SYMBOL/LOAD	TSPOL SYMBOLIC ** LOAD/CANDE
SYMBOL/MERG	TSPOL SYMBOLIC ** MERGE/CANDE

SYMBOL/PAPER	TSPOL SYMBOLIC ** PAPER/CANDE
SYMBOL/PUNCH	TSPOL SYMBOLIC ** PUNCH/CANDE
SYMBOL/QUIKLST	TSPOL SYMBOLIC ** QUIKLST/CANDE
SYMBOL/REPLACE	TSPOL SYMBOLIC ** REPLACE/CANDE
SYMBOL/RESEQ	TSPOL SYMBOLIC ** RESEQ/CANDE
SYMBOL/RESEQB	TSPOL SYMBOLIC ** RESEQB/CANDE
SYMBOL/SCHEDUL	TSPOL SYMBOLIC ** SCHEDUL/CANDE
SYMBOL/USER	ALGOL SYMBOLIC ** USER/CANDE

THE FOLLOWING IS A LIST OF THE OBJECT CODE AND DATA FILES LOCATED ON THE TAPE LABELED "SYSTEM/FILE000".

MCP RELATED FILES

FILE NAME	DESCRIPTION
MCP/DISK	DATACOM MCP OBJECT CODE FILE
MCP/STUFF	DATACOM MCP "STUFF" FILE
MCP/PRT	DATACOM MCP "PRT" FILE
DC/AUXMCP	DATACOM MCP AUXILARY MEMORY FILE
INT/DISK	DATACOM INTRINSICS OBJECT CODE FILE
INT/STUFF	DATACOM INTRINSICS "STUFF" FILE
DC/AUXINT	DATACOM INTRINSICS AUXILARY MEMORY FILE
TSS/MCP	TIME-SHARING OBJECT CODE FILE
TSSMCP/STUFF	TIME-SHARING "STUFF" FILE
TSS/PRT	TIME-SHARING "PRT" FILE
TSS/AUXMCP	TIME-SHARING AUXILARY MEMORY FILE
TSS/INT	TIME-SHARING INTRINSICS OBJECT CODE FILE
TSSINT/STUFF	TIME-SHARING INTRINSICS "STUFF" FILE
TSS/AUXINT	TIME-SHARING INTRINSICS AUXILARY MEMORY FILE

COMPILER RELATED FILES

FILE NAME	DESCRIPTION
ALGOL/DISK	ALGOL COMPILER
BASIC/DISK	BASIC COMPILER
COBOL/DISK	COBOL COMPILER
COBOL68/DISK	COBOL68 COMPILER

ESPOL/DISK	ESPOL COMPILER
FORTRAN/DISK	FORTRAN COMPILER
MAKCAST/DISK	SYMBOLIC LIBRARY MAINTENANCE PROGRAM
TSPOL/DISK	TSPOL COMPILER
XALGOL/DISK	XALGOL COMPILER

CANDE RELATED FILES

FILE NAME	DESCRIPTION
CANDE/TSHARER	CANDE COMMAND AND EDIT PROGRAM
APPEND/CANDE	CANDE PROGRAM FOR APPEND VERB
COPY/CANDE	CANDE PROGRAM FOR COPY VERB
DELETE/CANDE	CANDE PROGRAM FOR DELETE VERB
FIND/DISK	CANDE PROGRAM FOR FIND VERB
GUARD/DISK	CANDE PROGRAM FOR GUARD VERB
HARD/CANDE	CANDE FILE MAINTENANCE PROGRAM
HELP/DISK	CANDE DISK FILE ERROR RECOVERY PROGRAM
LFILES/CANDE	CANDE PROGRAM FOR LFILES VERB
LIST/CANDE	CANDE PROGRAM FOR LIST VERB
LOAD/CANDE	CANDE PROGRAM FOR LOAD VERB
MERGE/CANDE	CANDE PROGRAM FOR MERGE VERB
MESAGE/CANDE	CANDE ERROR MESSAGE FILE
PAPER/CANDE	CANDE PROGRAM FOR PAPER VERB
PUNCH/CANDE	CANDE PROGRAM FOR PUNCH VERB
QUIKLST/CANDE	CANDE PROGRAM FOR QUIKLST VERB
REPLACE/CANDE	CANDE PROGRAM FOR REPLACE VERB
RESEQ/CANDE	CANDE PROGRAM FOR RESEQ VERB
RESEQB/CANDE	CANDE PROGRAM FOR RESEQB VERB
SCHEDUL/CANDE	CANDE PROGRAM FOR SCHEDUL VERB

SYSTEM UTILITY FILES

FILE NAME	DESCRIPTION
USER/CANDE	UPDATES "USERS/CANDE"
UPDATE/USERS	UPDATES "REMOTE/USERS"
PATCH/MERGE	PATCH MAINTENANCE PROGRAM
SYSDISK/MAKER	CREATES "SYSTEM/DISK"
AUXDATA/MAKER	AUXILARY MEMORY FILE MAINTENANCE PROGRAM

SYSTEM/MESSGEN	CREATES "MESSAGE/OTHEDAY"
DC1000/CODEGEN	GENERATES DC1000 R.J.E. CODE DECK

ANALYSIS RELATED FILES

FILE NAME	DESCRIPTION
ROTO/ROOTER	"SEPTIC" FILE ANALYZER
STATS1/ANALYZE	ANALYZER FOR TIME-SHARING STATISTICS FILE
STATS2/ANALYZE	STATISTICS ANALYZER FOR TIME-SHARING LOG
STATS3/ANALYZE	ANALYZER FOR DATACOM MCP STATISTICS FILE
STATS4/ANALYZE	STATISTICS ANALYZER FOR DATACOM MCP SYSTEM LOG
ONLINE/MAINT	ON-LINE MAINTENANCE PROGRAM
OTPECNF/MAINT	SET OF ON-LINE TAPE CONFIDENCE ROUTINES
LOGAN/DISK	TIME-SHARING LOG ANALYZER
LOGANL/MAINT	MAINTENANCE LOG ANALYZER
LOGOUT/DISK	DATACOM MCP SYSTEM LOG ANALYZER
LOGOUTR/DISK	DATACOM MCP REMOTE LOG ANALYZER
DCFILL/PRT	CREATES "MCP/PRT"
DUMP/ANALYZE	DATACOM MCP MEMORY DUMP ANALYZER
TSFILL/PRT	CREATES "TSS/PRT"
TSDUMP/ANALYZE	TIME-SHARING MCP MEMORY DUMP ANALYZER

PUNCH BACK-UP FILES OF "CARD LOAD SELECT" PROGRAMS

FILE NAME	DESCRIPTION
PUD/COLD	COLD START PROGRAM
PUD/COOL	COOL START PROGRAM
PUD/KERNEL	HALT LOAD KERNEL ROUTINE
PUD/MEMDUMP	MEMORY DUMP ROUTINE
PUD/DSKDSK	DISK TO DISK PROGRAM
PUD/TAPEDSK	TAPE TO DISK PROGRAM
PUD/AUXTST	AUXILARY MEMORY TEST

GENERAL UTILITY FILES

FILE NAME	DESCRIPTION
CHECKAL/TEST	ALGOL MASTER TEST PROGRAM
MASTER/TEST	COBOL MASTER TEST PROGRAM
TAPE/COMPARE	TAPE COMPARISION PROGRAM
TAPCOPY/DISK	TAPE COPY AND COMPARISION PROGRAM
DSKDUMP/UTILITY	LISTS DISK AREAS BY ADDRESS
HDRLST/UTILITY	LISTS DISK DIRECTORY HEADERS BY NAME
LIBLST/UTILITY	LIST SYMBOLIC FILES ON "LIBRARY DUMP" TAPES
XREF/JONES	CROSS REFERENCE AND DOCUMENT EDITING PROGRAM

PATCHES USED TO CREATE THE UPDATED SYMBOLICS

THESE FILES ARE INCLUDED FOR DOCUMENTATION PURPOSES ONLY AND MUST NOT BE USED WHEN COMPILING THE SOFTWARE. THE FILE NAMES WILL BE OF THE FORM:

MARK<MARK LEVEL><RELEASE LEVEL>/<SOFTWARE NAME>

EXAMPLE: MARKXV2/TSSMCP

TEMPORARY PATCHES APPLICABLE TO THE UPDATED SYMBOLICS

THESE PATCHES ARE CORRECTIONS FOR PROBLEMS WHICH WERE NOT FOUND IN TIME TO BE INCLUDED INTO THE SYMBOLICS AND MUST BE USED WHEN COMPILING THE SOFTWARE. THE FILE NAMES WILL BE OF THE FORM:

PATCH/<SOFTWARE NAME>

EXAMPLE: PATCH/TSSMCP

PBD OF THE SYSTEM NOTE

THE FILE NAMED "PBD/SYSNOTE" IS A PRINTER BACK-UP DISK FILE OF THE SYSTEM NOTE. THIS FILE CAN BE PRINTED BY LOADING IT AND THEN CHANGING THE NAME TO A VALID "PBD" FILE NAME.

EXAMPLE: CC LOAD FROM SYSTEM PBD/SYSNOTE;END.

```
          CC CHANGE PBD/SYSNOTE TO PBD/0001001;END.
```

PBD-S OF ANY ADDITIONAL DOCUMENTATION

THE NAMES OF THE ADDITIONAL DOCUMENTATION FILES WILL BE GIVEN IN THE INTRODUCTION OF THE SYSTEM NOTE. THE SAME PROCESS USED TO PRINT THE SYSTEM NOTE MUST BE PREFORMED TO OBTAIN A PRINTER LISTING OF THESE FILES.

8.2.3 SUPPLEMENTARY PATCHES

SUPPLEMENTARY PATCHES ARE THOSE CORRECTIONS FOR SOFTWARE ERRORS THAT ARE GENERATED DURING THE INTERVAL BETWEEN SYSTEM RELEASES. THESE PATCHES ARE INTENDED TO AUGMENT, NOT REPLACE, THE TEMPORARY PATCHES WHICH ARE INCLUDED ON THE SYSTEM RELEASE.

THE SUPPLEMENTARY PATCHES ARE PUBLISHED IN THE "B-5700 SYSTEM FLASH" LETTER. WHENEVER PATCHES OF SIGNIFICANT IMPORTANCE ARE GENERATED, A SYSTEM FLASH IS DISTRIBUTED TO RECIPENTS OF THE SYSTEM NOTE.

8.2.4 "PATCH/MERGE" DECKS

IMPROVEMENTS AND CORRECTIONS TO THE SYSTEM SOFTWARE ARE RELEASED IN THE FORM OF PATCHES. USUALLY A PATCH WILL CHANGE ONLY A SINGLE ASPECT OF THE SYSTEM. SINCE A RELEASE CONTAINS MANY CHANGES, AND THEREFORE MANY PATCHES, A METHOD OF MERGING THESE PATCHES INTO A SINGLE CHANGE DECK IS REQUIRED. THE SYSTEM PROGRAM "PATCH/MERGE" IS USED TO PERFORM THIS MERGING PROCESS. A "PATCH/MERGE" DECK CONSISTS OF THE FOLLOWING ITEMS.

1. CONTROL CARDS FOR EXECUTING "PATCH/MERGE".

2. PATCH/MERGE OPTIONS

3. COMPILER OPTIONS

4. PATCHES TO BE MERGED INTO A CHANGE DECK.

EXAMPLE "PATCH/MERGE" DECK:

```
CONTROL CARDS > > > > > >   ?EXECUTE PATCH/MERGE

                            ?DATA CARD

PATCH/MERGE OPTIONS > > >   $@ MERGE ZIP LIST CONFLICTS

                            $, <N> PATCHES FOR PROGRAM

                            $*COMPILE OBJECT/PROGRAM ALGOL LIBRARY

                            $*DATA CARD

                            $-

COMPILER OPTION > > > > >   $ SET LIST PRT SINGLE

                            $ RESET NEW TAPE

PROGRAM PATCHES > > > > >   $#PATCH 1 FOR PROGRAM CONTAINS 2 CARDS

                            BEGIN

                            END.

CONTROL CARDS > > > > > >   ?END
```

A COMPLETE DESCRIPTION OF THE "PATCH/MERGE" PROGRAM IS INCLUDED AS APPENDIX A TO THIS MANUAL. THE "PATCH/MERGE" DECKS USED IN MAINTAINING THE SOFTWARE ARE DESCRIBED IN SUBSEQUENT PARAGRAPHS.

8.3 COMPILING THE SOFTWARE

THIS SECTION DESCRIBES EACH OF THE "PATCH/MERGE" DECKS NEEDED TO PROPERLY MAINTAIN THE SYSTEM SOFTWARE. IN ADDITION, AS EACH DECK IS DESCRIBED, A GENERAL DESCRIPTION OF THE SOFTWARE INVOLVED IS GIVEN.

REFER TO APPENDCX C FOR A DESCRIPTION OF THE COMPILER CONTROL OPTIONS SHOWN IN THE EXAMPLE DECKS.

8.3.1 DATACOM MCP

THE "DATACOM MCP" IS AN ESPOL PROGRAM WHICH CONTROLS AND COORDINATES THE VARIOUS FUNCTIONS REQUIRED TO PROCESS USER PROGRAMS. SEVERAL COMPILE-TIME OPTIONS ARE AVAILABLE WHICH ALLOW THE MCP TO BE TAILORED TO THE REQUIREMENTS OF EACH INDIVIDUAL SITE. THE FOLLOWING IS A LIST OF THESE OPTIONS WITH A BRIEF DESCRIPTION EXPLAINING THEIR FUNCTIONS.

AUXMEM	IF SET THEN CODE TO SUPPORT "AUXILARY MEMORY" IS INCLUDED.
B6500LOAD	IF SET THEN CODE TO ALLOW LOADING OF B-6700 LIBRARY DUMP TAPES IS INCLUDED.
BREAKOUT	IF SET THEN CODE TO SUPPORT PROGRAM RESTART CAPABILITIES IS INCLUDED.
CHECKLINK	IF SET THEN PROCEDURES WHICH CHECK THE MEMORY LINKS FOR VALIDITY ARE INCLUDED.
DATACOM	IF SET THEN CODE TO HANDLE THE B-249 AND THE B-487 DATA COMMUNICATIONS HARDWARE IS INCLUDED.
DCLOG	IF SET THEN PROCEDURES WHICH LOG REMOTE TERMINAL ACTIVITY ARE INCLUDED.
DCSPO	IF SET THEN REMOTE TERMINALS ARE GIVEN "REMOTE SPO" CAPABILITIES.
DEBUGGING	IF SET THEN CODE WHICH ALLOWS ON-LINE DEBUGGING IS INCLUDED.
DFX	IF SET THEN CODE TO HANDLE A DISK FILE EXCHANGE IS INCLUDED.

DISKLOG	IF SET THEN DISK USAGE INFORMATION IS INCLUDED AS PART OF THE SYSTEM ACTIVITY LOG.
DKBNODFX	IF SET THEN CODE TO HANDLE TWO DISK CONTROLLERS WITHOUT AN EXCHANGE IS INCLUDED.
DUMP	IF SET THEN PROCEDURES WHICH ALLOW THE ON-LINE CREATION OF MEMORY DUMP TAPES IS INCLUDED.
MONITOR	IF SET THEN CODE IS INCLUDED WHICH IMPLEMENTS AN ON-LINE TRACE FUNCTION.
PACKETS	IF SET THEN CODE IS INCLUDED WHICH IMPLEMENTS THE "PACKETS" METHOD OF SYSTEM CONTROL.
SAVERESULTS	IF SET THEN A CYCLIC TABLE OF DATA COMMUNICATIONS ACTIVITY IS MAINTAINED.
SEPTICTANK	IF SET THEN PROCEDURES WHICH ALLOW THE MONITORING OF DATA COMMUNICATIONS ACTIVITY ARE INCLUDED.
SHAREDISK	IF SET THEN CODE IS INCLUDED WHICH ALLOWS TWO OR MORE SYSTEMS TO SHARE COMMON DISK STORAGE.
STATISTICS	IF SET THEN CODE WHICH MONITORS SYSTEM ACTIVITY IS INCLUDED.
RJE	IF SET THEN CODE TO SUPPORT REMOTE JOB ENTRY TERMINALS IS INCLUDED.

EXAMPLE DECK:
------- -----

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@MERGE ZIP LIST CONFLICTS

$,<N> PATCHES FOR MCP

$*EXECUTE ESPOL/DISK

$*FILE  TAPE =  SYMBOL/MCP     SERIAL

$*FILE  LINE =     MCP/LISTING PRINT BACK UP DISK

$*FILE  DISK =     MCP/DISK

$*FILE STUFF =     MCP/STUFF   SERIAL

$*DATA CARD

$-

$ RESET AUXMEM

$ SET   AUXMEM

$ RESET B6500LOAD

$ SET   B6500LOAD

$ RESET BREAKOUT

$ SET   BREAKOUT

$ RESET CHECKLINK

$ SET   CHECKLINK

$ RESET DATACOM

$ SET   DATACOM

$ RESET DCLOG

$ SET   DCLOG
$ RESET DCSPO
$ SET   DCSPO
$ SET   DEBUGGING
$ RESET DEBUGGING
$ RESET DFX
$ SET   DFX
$ RESET DISKLOG
$ SET   DISKLOG
$ SET   DKBNODFX
$ RESET DKBNODFX
$ RESET DUMP
$ SET   DUMP
$ SET   MONITOR
$ RESET MONITOR
$ RESET PACKETS
$ SET   PACKETS
$ RESET SAVERESULTS
$ SET   SAVERESULTS
$ RESET SEPTICTANK
$ SET   SEPTICTANK
$ SET   SHAREDISK
$ RESET SHAREDISK
$ SET   STATISTICS
$ RESET STATISTICS

$ RESET RJE

$ SET   RJE

$ SET   TAPE  STUFF

$ RESET LISTA PRT SINGLE

$ SET   LISTA PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

8.3.2 TIME-SHARING MCP

THE "TIME-SHARING MCP" IS AN ESPOL PROGRAM WHICH IS DESIGNED TO CONTROL AND COORDINATE THE SYSTEM RESOURCES IN A MANNER THAT WILL OPTIMIZE THE RESPONSE TIME OF A REMOTE USER PROGRAM. SEVERAL COMPILE-TIME OPTIONS ARE AVAILABLE WHICH ALLOW A SITE TO TAILOR THE MCP TO THEIR SPECIFIC NEEDS. A LIST OF THESE OPTIONS AND A DESCRIPTION OF THEIR FUNCTIONS FOLLOWS.

AUXMEM	IF SET THEN CODE TO SUPPORT "AUXILARY MEMORY" IS INCLUDED.
B6500LOAD	IF SET THEN CODE TO ALLOW LOADING OF B-6700 LIBRARY DUMP TAPES IS INCLUDED.
CHECKLINK	IF SET THEN PROCEDURES WHICH CHECK THE MEMORY LINKS FOR VALIDITY ARE INCLUDED.
DEBUGGING	IF SET THEN CODE WHICH ALLOWS ON-LINE DEBUGGING IS INCLUDED.
DFX	IF SET THEN CODE TO HANDLE A DISK FILE EXCHANGE IS INCLUDED.
DKBNODFX	IF SET THEN CODE TO HANDLE TWO DISK CONTROLLERS WITHOUT AN EXCHANGE IS INCLUDED.
DUMP	IF SET THEN PROCEDURES WHICH ALLOW THE ON-LINE CREATION OF MEMORY DUMP TAPES IS INCLUDED.
MONITOR	IF SET THEN CODE IS INCLUDED WHICH IMPLEMENTS AN ON-LINE TRACE FUNCTION.
PACKETS	IF SET THEN CODE IS INCLUDED WHICH IMPLEMENTS THE "PACKETS" METHOD OF SYSTEM CONTROL.

SAVERESULTS	IF SET THEN A CYCLIC TABLE OF DATA COMMUNICATIONS ACTIVITY IS MAINTAINED.
SEPTICTANK	IF SET THEN PROCEDURES WHICH ALLOW THE MONITORING OF DATA COMMUNICATIONS ACTIVITY ARE INCLUDED.
SHAREDISK	IF SET THEN CODE IS INCLUDED WHICH ALLOWS TWO OR MORE SYSTEMS TO SHARE COMMON DISK STORAGE.
STATISTICS	IF SET THEN CODE WHICH MONITORS SYSTEM ACTIVITY IS INCLUDED.
TWXONLY	IF SET THEN CODE TO ONLY HANDLE TELETYPES IS INCLUDED.

EXAMPLE DECK:
------- -----

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$, <N> PATCHES FOR TSSMCP

$*EXECUTE ESPOL/DISK

$*FILE  TAPE =  SYMBOL/TSSMCP  SERIAL

$*FILE  LINE =  TSSMCP/LISTING PRINT BACK UP DISK

$*FILE  DISK =     TSS/MCP

$*FILE STUFF =  TSSMCP/STUFF   SERIAL

$*DATA  CARD
```

```
$-

$ RESET AUXMEM

$ SET   AUXMEM

$ RESET B6500LOAD

$ SET   B6500LOAD

$ RESET CHECKLINK

$ SET   CHECKLINK

$ SET   DEBUGGING

$ RESET DEBUGGING

$ RESET DFX

$ SET   DFX

$ SET   DKBNODFX

$ RESET DKBNODFX

$ RESET DUMP

$ SET   DUMP

$ SET   MONITOR

$ RESET MONITOR

$ RESET PACKETS

$ SET   PACKETS

$ RESET SAVERESULTS

$ SET   SAVERESULTS

$ RESET SEPTICTANK

$ SET   SEPTICTANK
$ SET   SHAREDISK
$ RESET SHAREDISK
$ SET   STATISTICS
$ RESET STATISTICS
$ SET   TWXONLY
$ RESET TWXONLY
$ SET   TAPE  STUFF
$ RESET LISTA PRT SINGLE
$ SET   LISTA PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

8.3.3 SYSTEM INTRINSICS

THE "SYSTEM INTRINSICS" ARE A SET OF ESPOL PROCEDURES WHICH PERFORM VARIOUS FUNCTIONS FOR USER PROGRAMS. THESE FUNCTIONS INCLUDE SUCH ITEMS AS I/O HANDLING, MATRIX INVERSION, AND SORTING OR MERGING DATA. THE INTRINSICS CAN BE TAILORED TO A PARTICULAR SITES REQUIREMENTS BY THE USE OF COMPILE-TIME OPTIONS. THE FOLLOWING IS A LIST OF THESE OPTIONS WITH A BRIEF DESCRIPTION FOR EACH.

TIMESHARING	IF SET THEN INTRINSICS COMPATIBLE WITH THE "TIME-SHARING" MCP ARE GENERATED. IF RESET THEN INTRINSICS COMPATIBLE WITH THE "DATACOM" MCP ARE CREATED.
SHAREDISK	IF SET THEN CODE TO HANDLE "RECORD LEVEL LOCKOUT" IS INCLUDED. THIS OPTION SHOULD ONLY BE SET IF SHAREDISK IS SET IN THE APPROPRIATE MCP.

EXAMPLE DECK(DATACOM INTRINSICS):
------- ----------- -----------

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$. <N> PATCHES FOR INTRINSICS

$*EXECUTE ESPOL/DISK

$*FILE  TAPE =  SYMBOL/INTRINS SERIAL

$*FILE  LINE = INTRINS/LISTING PRINT BACK UP DISK

$*FILE  DISK =     INT/DISK

$*FILE STUFF =     INT/STUFF   SERIAL

$*DATA  CARD

$-

$ SET   SHAREDISK

$ RESET SHAREDISK

$ SET   TAPE  STUFF  INTRINSIC  RESET TIMESHARING

$ RESET LISTA PRT SINGLE

$ SET   LISTA PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

EXAMPLE DECK(TIME-SHARING INTRINSICS):

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$, <N> PATCHES FOR INTRINSICS

$*EXECUTE ESPOL/DISK

$*FILE  TAPE =  SYMBOL/INTRINS SERIAL

$*FILE  LINE = INTRINS/LISTING PRINT BACK UP DISK

$*FILE  DISK =     TSS/INT

$*FILE STUFF =  TSSINT/STUFF   SERIAL

$*DATA  CARD

$-

$ SET   SHAREDISK

$ RESET SHAREDISK

$ SET   TAPE  STUFF  INTRINSIC  TIMESHARING

$ RESET LISTA PRT SINGLE

$ SET   LISTA PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

8.3.4 COOL AND COLD START ROUTINES

THE "COOL START" AND "COLD START" ROUTINES ARE "CARD LOAD SELECT" PROGRAMS. THE "COLD START" ROUTINE IS USED TO BUILD A DISK DIRECTORY, CREATE "COLD START" FILES, AND TO INITIALIZE CERTAIN SYSTEM PARAMETERS. THE "COOL START" ROUTINE IS USED TO CHECK AN EXISTING DISK DIRECTORY FOR CORRECTNESS AND TO INITIALIZE CERTAIN SYSTEM PARAMETERS. THE COMPILE-TIME OPTION "COOL" IS USED TO DETERMINE WHICH ROUTINE IS GENERATED SINCE BOTH ROUTINES ARE CONTAINED IN THE SAME SYMBOLIC FILE.

EXAMPLE DECK(COLD START):

```
?EXECUTE PATCH/MERGE
?DATA CARD
$@ MERGE ZIP LIST CONFLICTS
$. <N> PATCHES FOR COOL
$*EXECUTE ESPOL/DISK
$*FILE  TAPE =  SYMBOL/COOL    SERIAL
$*FILE  LINE =    COLD/LISTING PRINT BACK UP DISK
$*FILE  DECK =    COLD/START   PUNCH BACK UP DISK
$*DATA  CARD
$-
$ RESET COOL
$ SET   TAPE  DECK
$ RESET LISTA PRT SINGLE
$ SET   LISTA PRT SINGLE
```

```
<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

EXAMPLE DECK(COOL START):

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$. <N> PATCHES FOR COOL

$*EXECUTE ESPOL/DISK

$*FILE  TAPE =  SYMBOL/COOL     SERIAL

$*FILE  LINE =     COOL/LISTING PRINT BACK UP DISK

$*FILE  DECK =     COOL/START   PUNCH BACK UP DISK

$*DATA  CARD

$-

$ SET    COOL

$ SET    TAPE  DECK

$ RESET LISTA PRT SINGLE

$ SET   LISTA PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

8.3.5 TAPE TO DISK LOADER

THE "TAPE TO DISK" ROUTINE IS A "CARD LOAD SELECT" PROGRAM WHICH LOADS A MCP CODE FILE FROM A "LIBRARY MAINTENANCE DUMP" TAPE TO THE HEAD-PER-TRACK DISK AND ENTERS THE NAME AND STARTING DISK ADDRESS INTO DISK SEGMENT ZERO WITHOUT THE PRESENCE OF A FUNCTIONING OPERATING SYSTEM.

EXAMPLE DECK:

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$. <N> PATCHES FOR TAPEDSK

$*EXECUTE ESPOL/DISK

$*FILE  TAPE =  SYMBOL/TAPEDSK SERIAL

$*FILE  LINE = TAPEDSK/LISTING PRINT BACK UP DISK

$*FILE  DECK = TAPEDSK/LOADER  PUNCH BACK UP DISK

$*DATA  CARD

$-

$ SET   TAPE DECK

$ RESET LIST PRT SINGLE

$ SET   LIST PRT SINGLE

<SUPPLEMEMTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

8.3.6 DISK TO DISK LOADER

THE "DISK TO DISK" LOADER IS A "CARD LOAD SELECT" PROGRAM WHICH IS USED TO ENTER THE NAME AND STARTING DISK ADDRESS OF A MCP CODE FILE INTO DISK SEGMENT ZERO WITHOUT THE PRESENCE OF A FUNCTIONING OPERATING SYSTEM.

EXAMPLE DECK:

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$. <N> PATCHES FOR DSKDSK

$*EXECUTE ESPOL/DISK

$*FILE  TAPE =  SYMBOL/DSKDSK  SERIAL

$*FILE  LINE =  DSKDSK/LISTING PRINT BACK UP DISK

$*FILE  DECK =  DSKDSK/LOADER  PUNCH BACK UP DISK

$*DATA  CARD

$-

$ SET   TAPE DECK

$ RESET LIST PRT SINGLE

$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

8.3.7 MEMORY DUMP ROUTINE

THE "MEMORY DUMP" ROUTINE IS A "CARD LOAD SELECT" PROGRAM WHICH WRITES THE CONTENTS OF MAINFRAME AND AUXILARY MEMORY TO EITHER A MAGNETIC TAPE WHICH WILL BE LABELED "MDUMP" OR AN EXISTING DISK FILE NAMED "MEMORY/DUMP000". THE RESULTING FILE IS THEN USED AS INPUT TO THE APPROPRIATE DUMP ANALYZER PROGRAM.

EXAMPLE DECK:

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$. <N> PATCHES FOR MEMDUMP

$*EXECUTE ESPOL/DISK

$*FILE  TAPE =  SYMBOL/MEMDUMP SERIAL

$*FILE  LINE = MEMDUMP/LISTING PRINT BACK UP DISK

$*FILE  DECK =  MEMORY/DUMP    PUNCH BACK UP DISK

$*DATA  CARD

$-

$ SET   TAPE DECK

$ RESET LIST PRT SINGLE

$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

8.3.8 HALT LOAD KERNEL ROUTINE

THE "HALT LOAD KERNEL" IS A "CARD LOAD SELECT" PROGRAM WHICH PERFORMS THE NECESSARY "HOUSEKEEPING" REQUIRED PRIOR TO BRINGING THE SPECIFIED MCP CODE FILE INTO MEMORY.

EXAMPLE DECK:

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$. <N> PATCHES FOR KERNEL

$*EXECUTE ESPOL/DISK

$*FILE  TAPE =  SYMBOL/KERNEL  SERIAL

$*FILE  LINE =  KERNEL/LISTING PRINT BACK UP DISK

$*FILE  DECK =  KERNEL/LOADER  PUNCH BACK UP DISK

$*DATA  CARD

$-

$ SET   TAPE DECK

$ RESET LIST PRT SINGLE

$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

8.3.9 AUXILARY MEMORY CONFIDENCE TEST

THE "AUXILIARY MEMORY TEST" ROUTINE IS A "CARD LOAD SELECT" PROGRAM WHICH PERFORMS EITHER A READ-ONLY OR A READ-WRITE CONFIDENCE TEST ON A SPECIFIED AUXILARY MEMORY UNIT.

EXAMPLE DECK:

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$. <N> PATCHES FOR AUXTST

$*EXECUTE ESPOL/DISK

$*FILE  TAPE =  SYMBOL/AUXTST  SERIAL

$*FILE  LINE =  AUXTST/LISTING PRINT BACK UP DISK

$*FILE  DECK =  AUXMEM/TEST    PUNCH BACK UP DISK

$*DATA  CARD

$-

$ SET   TAPE DECK

$ RESET LIST PRT SINGLE

$ SET   LIST PRT SINGLE


<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>


?END
```

8.3.10 ALGOL AND TSPOL COMPILERS

THE ALGOL AND TSPOL COMPILERS ARE ALGOL PROGRAMS THAT TRANSLATE ALGOL AND TSPOL SYMBOLIC PROGRAMS INTO EFFICIENT B-5700 MACHINE CODE. EITHER COMPILER CAN BE GENERATED FROM THE SAME SYMBOLIC FILE, "SYMBOL/ALGOL", BY SETTING OR RESETTING THE COMPILE-TIME OPTION "TSPOL".

EXAMPLE DECK(ALGOL COMPILER):

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$. <N> PATCHES FOR ALGOL

$*COMPILE ALGOL/DISC ALGOL LIBRARY

$*ALGOL FILE TAPE =  SYMBOL/ALGOL   SERIAL

$*ALGOL FILE LINE =   ALGOL/LISTING PRINT BACK UP DISK

$*FILE  NEWTAPE   =  "OCRDIMG"      TAPE

$*FILE     TAPE   =  "OCRDIMG"      TAPE

$*FILE     LINE   =   ALGOL/COMPILE PRINT BACK UP DISK

$*FILE     PNCH   =   ERROR/CARDS   PUNCH BACK UP DISK

$*CORE = 10000

$*DATA

$-

$ SET   TAPE  SEQXEQ     ALGOL   RESET TSPOL

$ RESET LISTA PRT SINGLE
$ SET   LISTA PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

EXAMPLE DECK(TSPOL COMPILER):
------- ---------- ---------

```
?EXECUTE PATCH/MERGE
?DATA CARD
$@ MERGE ZIP LIST CONFLICTS
$. <N> PATCHES FOR ALGOL
$*COMPILE TSPOL/DISK ALGOL LIBRARY
$*ALGOL FILE TAPE =  SYMBOL/ALGOL   SERIAL
$*ALGOL FILE LINE =   TSPOL/LISTING PRINT BACK UP DISK
$*FILE  NEWTAPE   =  "OCRDIMG"      TAPE
$*FILE     TAPE   =  "OCRDIMG"      TAPE
$*FILE     LINE   =   TSPOL/COMPILE PRINT BACK UP DISK
$*FILE     PNCH   =   ERROR/CARDS   PUNCH BACK UP DISK
$*CORE = 10000
$*DATA
$-
$ SET   TAPE  SEQXEQ     TSPOL   RESET ALGOL
$ RESET LISTA PRT SINGLE    .
$ SET   LISTA PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

8.3.11 XALGOL COMPILER

THE XALGOL COMPILER IS AN ALGOL PROGRAM WHICH TRANSLATES XALGOL SYMBOLIC PROGRAMS INTO EFFICIENT B-5700 MACHINE CODE.

EXAMPLE DECK:

```
?EXECUTE PATCH/MERGE
?DATA CARD
$@ MERGE ZIP LIST CONFLICTS
$. <N> PATCHES FOR XALGOL
$*COMPILE XALGOL/DISK ALGOL LIBRARY
$*ALGOL FILE TAPE =  SYMBOL/XALGOL  SERIAL
$*ALGOL FILE LINE =  XALGOL/LISTING PRINT BACK UP DISK
$*FILE  NEWTAPE   =  "OCRDIMG"      TAPE
$*FILE     TAPE   =  "OCRDIMG"      TAPE
$*FILE     LINE   =  XALGOL/COMPILE PRINT BACK UP DISK
$*FILE     PNCH   =   ERROR/CARDS   PUNCH BACK UP DISK
$*CORE = 10000
$*DATA
$-
$ SET   TAPE SEQXEQ
$ RESET LIST PRT SINGLE
$ SET   LIST PRT SINGLE
```

```
<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

8.3.12 ESPOL COMPILER

THE ESPOL COMPILER IS AN ALGOL PROGRAM WHICH TRANSLATES ESPOL SYMBOLIC PROGRAMS INTO EFFICIENT B-5700 MACHINE CODE.

EXAMPLE DECK:

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$. <N> PATCHES FOR ESPOL

$*COMPILE ESPOL/DISK ALGOL LIBRARY

$*ALGOL FILE TAPE =  SYMBOL/ESPOL   SERIAL

$*ALGOL FILE LINE =   ESPOL/LISTING PRINT BACK UP DISK

$*FILE  NEWTAPE   =  "OCRDIMG"      TAPE

$*FILE     TAPE   =  "OCRDIMG"      TAPE

$*FILE     LINE   =   ESPOL/COMPILE PRINT BACK UP DISK

$*FILE     PNCH   =   ERROR/CARDS   PUNCH BACK UP DISK

$*CORE = 10000

$*DATA

$-

$ SET   TAPE SEQXEQ

$ RESET LIST PRT SINGLE

$ SET   LIST PRT SINGLE
```

```
<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

8.3.13 FORTRAN COMPILER

THE FORTRAN COMPILER IS AN ALGOL PROGRAM WHICH TRANSLATES FORTRAN SYMBOLIC PROGRAMS INTO EFFICIENT B-5700 MACHINE CODE.

EXAMPLE DECK:

```
?EXECUTE PATCH/MERGE
?DATA CARD
$@ MERGE ZIP LIST CONFLICTS
$. <N> PATCHES FOR FORTRAN
$*COMPILE FORTRAN/DISK ALGOL LIBRARY
$*ALGOL FILE TAPE =  SYMBOL/FORTRAN SERIAL
$*ALGOL FILE LINE = FORTRAN/LISTING PRINT BACK UP DISK
$*ALGOL STACK = 1000
$*FILE  NEWTAPE   =  FORSYM         TAPE
$*FILE     TAPE   =  FORSYM         TAPE
$*FILE     LINE   = FORTRAN/COMPILE PRINT BACK UP DISK
$*CORE = 8000
$*DATA
$-
$ SET   TAPE SEQXEQ
$ RESET LIST PRT SINGLE
$ SET   LIST PRT SINGLE
```

```
<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

8.3.14 BASIC COMPILER

THE BASIC COMPILER IS AN ALGOL PROGRAM WHICH TRANSLATES BASIC SYMBOLIC PROGRAMS INTO EFFICIENT B-5700 MACHINE CODE.

EXAMPLE DECK:

```
?EXECUTE PATCH/MERGE
?DATA CARD
$@ MERGE ZIP LIST CONFLICTS
$. <N> PATCHES FOR BASIC
$*COMPILE BASIC/DISK ALGOL LIBRARY
$*ALGOL FILE TAPE =  SYMBOL/BASIC   SERIAL
$*ALGOL FILE LINE =   BASIC/LISTING PRINT BACK UP DISK
$*FILE  NEWTAPE   =  "OCRDIMG"      TAPE
$*FILE     TAPE   =  "OCRDIMG"      TAPE
$*FILE     LINE   =   BASIC/COMPILE PRINT BACK UP DISK
$*CORE = 6000
$*DATA
$-
$ SET   TAPE SEQXEQ
$ RESET LIST PRT SINGLE
$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>
```

```
?END
```

8.3.15 COBOL COMPILER

THE COBOL COMPILER IS AN ALGOL PROGRAM WHICH TRANSLATES COBOL SYMBOLIC PROGRAMS INTO EFFICIENT B-5700 MACHINE CODE.

EXAMPLE DECK:

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$. <N> PATCHES FOR COBOL

$*COMPILE COBOL/DISK ALGOL LIBRARY

$*ALGOL FILE TAPE =  SYMBOL/COBOL   SERIAL

$*ALGOL FILF LINE =   COBOL/LISTING PRINT BACK UP DISK

$*ALGOL STACK = 1000

$*FILE  NEWTAPE   =    SOLT          TAPE

$*FILE     TAPE   =    SOLT          TAPE

$*FILE     LINE   =   COBOL/COMPILE PRINT BACK UP DISK

$*FILE     PNCH   =   ERROR/CARDS   PUNCH BACK UP DISK

$*CORE = 12000

$*DATA

$-

$ SET   TAPE SEQXEQ

$ RESET LIST PRT SINGLE
$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

8.3.16 COBOL68 COMPILER

THE COBOL68 COMPILER IS AN ALGOL PROGRAM WHICH TRANSLATES COBOL68 SYMBOLIC PROGRAMS INTO EFFICIENT B-5700 MACHINE CODE.

EXAMPLE DECK:

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$. <N> PATCHES FOR COBOL68

$*COMPILE COBOL68/DISK ALGOL LIBRARY

$*ALGOL FILF TAPE =  SYMBOL/COBOL68 SERIAL

$*ALGOL FILE LINE = COBOL68/LISTING PRINT BACK UP DISK

$*ALGOL STACK = 1000

$*FILE  NEWTAPE   =    SOLT          TAPE

$*FILE     TAPE   =    SOLT          TAPE

$*FILE     LINE   = COBOL68/COMPILE PRINT BACK UP DISK

$*FILE     PNCH   =   ERROR/CARDS   PUNCH BACK UP DISK

$*CORE = 12000

$*DATA

$-

$ SET   TAPE SEQXEQ

$ RESET LIST PRT SINGLE
$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

8.3.17 MAKCAST/DISK

THE "MAKCAST/DISK" PROGRAM IS A SYMBOLIC LIBRARY MAINTENANCE PROGRAM WHICH IS USED TO CREATE AND UPDATE THE SYMBOLIC LIBRARY FILES USED AS INPUT TO THE SYSTEM COMPILERS.

EXAMPLE DECK:

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$. <N> PATCHES FOR MAKCAST

$*COMPILE MAKCAST/DISK ALGOL LIBRARY

$*ALGOL FILE TAPE =  SYMBOL/MAKCAST SERIAL

$*ALGOL FILE LINE = MAKCAST/LISTING PRINT BACK UP DISK

$*ALGOL STACK = 1000

$*DATA

$-

$ SET   TAPE SEQXEQ

$ RESET LIST PRT SINLGE

$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

8.3.18 CANDE/TSHARER

"CANDE/TSHARER" IS A TSPOL PROGRAM WHICH PERFORMS THE COMMAND PROCESSING AND TEXT EDITING FOR THE "TIME-SHARING" MCP.

EXAMPLE DECK:

```
?EXECUTE PATCH/MERGE
?DATA CARD
$@ MERGE ZIP LIST CONFLICTS
$. <N> PATCHES FOR CANDE
$*COMPILE CANDE/TSHARER TSPOL LIBRARY
$*TSPOL FILE TAPE =  SYMBOL/CANDE   SERIAL
$*TSPOL FILE LINE =   CANDE/LISTING PRINT BACK UP DISK
$*TSPOL STACK = 1000
$*CORE = 5000
$*DATA
$-
$ SET   TAPE SEQXEQ
$ RESET LIST PRT SINGLE
$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

8.3.19 CANDE VERB PROGRAMS

THE FOLLOWING IS A LIST OF EXAMPLE DECKS FOR THE "CANDE" VERB PROCESSING PROGRAMS AND THE "CANDE" AUXILARY PROGRAMS.

EXAMPLE DECK(APPEND/CANDE):

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$. <N> PATCHES FOR APPEND

$*COMPILE APPEND/CANDE TSPOL LIBRARY

$*TSPOL FILE TAPE =  SYMBOL/APPEND  SERIAL

$*TSPOL FILE LINE =  APPEND/LISTING PRINT BACK UP DISK

$*DATA

$-

$ SET   TAPE SEQXEQ

$ RESET LIST PRT SINGLE

$ SET   LIST PRT SINGLE


<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>


?END
```

EXAMPLE DECK(COPY/CANDE):
------- -----------------

```
?EXECUTE PATCH/MERGE
?DATA CARD
$@ MERGE ZIP LIST CONFLICTS
$, <N> PATCHES FOR COPY
$*COMPILE COPY/CANDE TSPOL LIBRARY
$*TSPOL FILE TAPE =  SYMBOL/COPY    SERIAL
$*TSPOL FILE LINE =    COPY/LISTING PRINT BACK UP DISK
$*DATA
$-
$ SET   TAPE SEQXEQ
$ RESET LIST PRT SINGLE
$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

EXAMPLE DECK(DELETE/CANDE):
------- -------------------

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$. <N> PATCHES FOR DELETE

$*COMPILE DELETE/CANDE TSPOL LIBRARY

$*TSPOL FILE TAPE =  SYMBOL/DELETE  SERIAL

$*TSPOL FILE LINE =  DELETE/LISTING PRINT BACK UP DISK

$*DATA

$-

$ SET   TAPE SEQXEQ

$ RESET LIST PRT SINGLE

$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

EXAMPLE DECK(FIND/DISK):
------- ---------------

```
?EXECUTE PATCH/MERGE
?DATA CARD
$@ MERGE ZIP LIST CONFLICTS
$. <N> PATCHES FOR FIND
$*COMPILE FIND/DISK TSPOL LIBRARY
$*TSPOL FILE TAPE =  SYMBOL/FIND    SERIAL
$*TSPOL FILE LINE =    FIND/LISTING PRINT BACK UP DISK
$*DATA
$-
$ SET   TAPE SEQXEQ
$ RESET LIST PRT SINGLE
$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

FXAMPLE DECK(GUARD/DISK):
------- ----------------

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$. <N> PATCHES FOR GUARD

$*COMPILE GUARD/DISK TSPOL LIBRARY

$*TSPOL FILE TAPE =  SYMBOL/GUARD   SERIAL

$*TSPOL FILF LINE =   GUARD/LISTING PRINT BACK UP DISK

$*DATA

$-

$ SET   TAPE SEQXEQ

$ RESET LIST PRT SINGLE

$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

EXAMPLE DECK(HARD/CANDE):

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$. <N> PATCHES FOR HARD

$*COMPILE HARD/CANDE TSPOL LIBRARY

$*TSPOL FILE TAPE =  SYMBOL/HARD    SERIAL

$*TSPOL FILE LINE =    HARD/LISTING PRINT BACK UP DISK

$*DATA

$-

$ SET   TAPE SEQXEQ

$ RESET LIST PRT SINGLE

$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

EXAMPLE DECK(HELP/DISK):
------- ---------------

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$. <N> PATCHES FOR HELP

$*COMPILE HELP/DISK TSPOL LIBRARY

$*TSPOL FILE TAPE =  SYMBOL/HELP    SERIAL

$*TSPOL FILE LINE =    HELP/LISTING PRINT BACK UP DISK

$*DATA

$-

$ SET   TAPE SEQXEQ

$ RESET LIST PRT SINGLE

$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

EXAMPLE DECK(LFILES/CANDE):
------- -------------------

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$. <N> PATCHES FOR LFILES

$*COMPILE LFILES/CANDE TSPOL LIBRARY

$*TSPOL FILE TAPE =  SYMBOL/LFILES  SERIAL

$*TSPOL FILE LINE =  LFILES/LISTING PRINT BACK UP DISK

$*DATA

$-

$ SET   TAPE SEQXEQ

$ RESET LIST PRT SINGLE

$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

FXAMPLE DECK(LIST/CANDE):
------- ----------------

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$. <N> PATCHES FOR LIST

$*COMPILE LIST/CANDE TSPOL LIBRARY

$*TSPOL FILE TAPE =  SYMBOL/LIST    SERIAL

$*TSPOL FILE LINE =    LIST/LISTING PRINT BACK UP DISK

$*DATA

$-

$ SET   TAPE SEQXEQ

$ RESET LIST PRT SINGLE

$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

EXAMPLE DECK(LOAD/CANDE):
------- ----------------

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$. <N> PATCHES FOR LOAD

$*COMPILE LOAD/CANDE TSPOL LIBRARY

$*TSPOL FILE TAPE =  SYMBOL/LOAD    SERIAL

$*TSPOL FILE LINE =    LOAD/LISTING PRINT BACK UP DISK

$*DATA

$-

$ SET   TAPE SEQXEQ

$ RESET LIST PRT SINGLE

$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

EXAMPLE DECK(MERGE/CANDE):

```
        ?EXECUTE PATCH/MERGE
        ?DATA CARD
        $@ MERGE ZIP LIST CONFLICTS
        $. <N> PATCHES FOR MERG
        $*COMPILE MERGE/CANDE TSPOL LIBRARY
        $*TSPOL FILE TAPE =  SYMBOL/MERG   SERIAL
        $*TSPOL FILE LINE =   MERGE/LISTING PRINT BACK UP DISK
        $*DATA
        $-
        $ SET   TAPE SEQXEQ
        $ RESET LIST PRT SINGLE
        $ SET   LIST PRT SINGLE

        <SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

        ?END
```

EXAMPLE DECK(PAPER/CANDE):
------- ------------------

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$. <N> PATCHES FOR PAPER

$*COMPILE PAPER/CANDE TSPOL LIBRARY

$*TSPOL FILE TAPE =  SYMBOL/PAPER   SERIAL

$*TSPOL FILE LINE =   PAPER/LISTING PRINT BACK UP DISK

$*DATA

$-

$ SET   TAPE SEQXEQ

$ RESET LIST PRT SINGLE

$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

EXAMPLE DECK(PUNCH/CANDE):
------- -----------------

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$. <N> PATCHES FOR PUNCH

$*COMPILE PUNCH/CANDE TSPOL LIBRARY

$*TSPOL FILE TAPE =  SYMBOL/PUNCH   SERIAL

$*TSPOL FILE LINE =   PUNCH/LISTING PRINT BACK UP DISK

$*DATA

$-

$ SET   TAPE SEQXEQ

$ RESET LIST PRT SINGLE

$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

EXAMPLE DECK(QUIKLST/CANDE):
------- --------------------

```
?EXECUTE PATCH/MERGE
?DATA CARD
$@ MERGE ZIP LIST CONFLICTS
$, <N> PATCHES FOR QUIKLST
$*COMPILE QUIKLST/CANDE TSPOL LIBRARY
$*TSPOL FILE TAPE =  SYMBOL/QUIKLST SERIAL
$*TSPOL FILE LINE = QUIKLST/LISTING PRINT BACK UP DISK
$*DATA
$-
$ SET   TAPE SEQXEQ
$ RESET LIST PRT SINGLE
$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

EXAMPLE DECK(REPLACE/CANDE):
------- ------------------

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$. <N> PATCHES FOR REPLACE

$*COMPILE REPLACE/CANDE TSPOL LIBRARY

$*TSPOL FILE TAPE =  SYMBOL/REPLACE SERIAL

$*TSPOL FILE LINE = REPLACE/LISTING PRINT BACK UP DISK

$*DATA

$-

$ SET   TAPE SEQXEQ

$ RESET LIST PRT SINGLE

$ SET   LIST PRT SINGLE


<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>


?END
```

EXAMPLE DECK(RESEQ/CANDE):
------- -----------------

```
?EXECUTE PATCH/MERGE
?DATA CARD
$@ MERGE ZIP LIST CONFLICTS
$. <N> PATCHES FOR RESEQ
$*COMPILE RESEQ/CANDE TSPOL LIBRARY
$*TSPOL FILE TAPE =  SYMBOL/RESEQ   SERIAL
$*TSPOL FILE LINE =   RESEQ/LISTING PRINT BACK UP DISK
$*DATA
$-
$ SET   TAPE SEQXEQ
$ RESET LIST PRT SINGLE
$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

EXAMPLE DECK(RESEQB/CANDE):
------- -------------------

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$, <N> PATCHES FOR RESEQB

$*COMPILE RESEQB/CANDE TSPOL LIBRARY

$*TSPOL FILE TAPE =  SYMBOL/RESEQB  SERIAL

$*TSPOL FILE LINE =  RESEQB/LISTING PRINT BACK UP DISK

$*DATA

$-

$ SET   TAPE SEQXEQ

$ RESET LIST PRT SINGLE

$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

EXAMPLE DECK(SCHEDUL/CANDE):
------- --------------------

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$, <N> PATCHES FOR SCHEDUL

$*COMPILE SCHEDUL/CANDE TSPOL LIBRARY

$*TSPOL FILE TAPE =  SYMBOL/SCHEDUL SERIAL

$*TSPOL FILE LINE = SCHEDUL/LISTING PRINT BACK UP DISK

$*DATA

$-

$ SET   TAPE SEQXEQ

$ RESET LIST PRT SINGLE

$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

8.3.20 SYSTEM UTILITY PROGRAMS

THE FOLLOWING IS A LIST OF EXAMPLE DECKS FOR THE SYSTEM UTILITY PROGRAMS. THESE PROGRAMS PERFORM FUNCTIONS WHICH ARE CLOSELY RELATED TO THE OPERATION OF THE SYSTEM.

EXAMPLE DECK(USER/CANDE):

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$. <N> PATCHES FOR USER

$*COMPILE USER/CANDE ALGOL LIBRARY

$*ALGOL FILE TAPE =  SYMBOL/USER    SERIAL

$*ALGOL FILE LINE =    USER/LISTING PRINT BACK UP DISK

$*DATA

$-

$ SET   TAPE SEQXEQ

$ RESET LIST PRT SINLGE

$ SET   LIST PRT SINGLE


<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>


?END
```

EXAMPLE DECK(UPDATE/USERS):
------- -------------------

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$. <N> PATCHES FOR UPDATE

$*COMPILE UPDATE/USERS ALGOL LIBRARY

$*ALGOL FILE TAPE =  SYMBOL/UPDATE  SERIAL

$*ALGOL FILE LINE =  UPDATE/LISTING PRINT BACK UP DISK

$*DATA

$-

$ SET   TAPE SEQXEQ

$ RESET LIST PRT SINLGE

$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

EXAMPLE DECK(PATCH/MERGE):
------- -----------------

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$, <N> PATCHES FOR PMERGE

$*COMPILE PATCH/MERGE XALGOL LIBRARY

$*XALGOL FILE TAPE =  SYMBOL/PMERGE  SERIAL

$*XALGOL FILE LINE =  PMERGE/LISTING PRINT BACK UP DISK

$*DATA

$-

$ SET   TAPE SEQXEQ

$ RESET LIST PRT SINLGE

$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

EXAMPLE DECK(SYSDISK/MAKER):
------- --------------------

```
?EXECUTE PATCH/MERGE
?DATA CARD
$@ MERGE ZIP LIST CONFLICTS
$. <N> PATCHES FOR SYSDISK
$*COMPILE SYSDISK/MAKER ALGOL LIBRARY
$*ALGOL FILE TAPE =  SYMBOL/SYSDISK SERIAL
$*ALGOL FILE LINE = SYSDISK/LISTING PRINT BACK UP DISK
$*DATA
$-
$ SET   TAPE SEQXEQ
$ RESET LIST PRT SINLGE
$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

EXAMPLE DECK(AUXDATA/MAKER):
------- --------------------

```
?EXECUTE PATCH/MERGE
?DATA CARD
$@ MERGE ZIP LIST CONFLICTS
$. <N> PATCHES FOR AUXDATA
$*COMPILE AUXDATA/MAKER ALGOL LIBRARY
$*ALGOL FILE TAPE =  SYMBOL/AUXDATA SERIAL
$*ALGOL FILE LINE = AUXDATA/LISTING PRINT BACK UP DISK
$*DATA
$-
$ SET   TAPE SEQXEQ
$ RESET LIST PRT SINLGE
$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

EXAMPLE DECK(SYSTEM/MESSGEN):
------- --------------------

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$. <N> PATCHES FOR MESSGEN

$*COMPILE SYSTEM/MESSGEN ALGOL LIBRARY

$*ALGOL FILE TAPE =  SYMBOL/MESSGEN SERIAL

$*ALGOL FILE LINE = MESSGEN/LISTING PRINT BACK UP DISK

$*DATA

$-

$ SET   TAPE SEQXEQ

$ RESET LIST PRT SINLGE

$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

EXAMPLE DECK(DC1000/CODEGEN):
------- --------------------

```
?EXECUTE PATCH/MERGE
?DATA CARD
$@ MERGE ZIP LIST CONFLICTS
$. <N> PATCHES FOR DC1000
$*COMPILE DC1000/CODEGEN ALGOL LIBRARY
$*ALGOL FILE TAPE =  SYMBOL/DC1000  SERIAL
$*ALGOL FILE LINE =  DC1000/LISTING PRINT BACK UP DISK
$*DATA
$-
$ SET   TAPE SEQXEQ
$ RESET LIST PRT SINLGE
$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

8.3.21 ANALYSIS PROGRAMS

THE FOLLOWING IS A LIST OF EXAMPLE DECKS FOR THE SYSTEM ANALYSIS PROGRAMS. THESE PROGRAMS ARE USED TO ANALYZE THE VARIOUS FILES PRODUCED BY THE SYSTEM.

EXAMPLE DECK(ROTO/ROOTER):

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$. <N> PATCHES FOR ROTO

$*COMPILE ROTO/ROOTER ALGOL LIBRARY

$*ALGOL FILE TAPE =  SYMBOL/ROTO    SERIAL

$*ALGOL FILE LINE =    ROTO/LISTING PRINT BACK UP DISK

$*FILE    P =  SEPTIC/LISTING PRINT BACK UP DISK

$*DATA

$-

$ SET   TAPE SEQXEQ

$ RESET LIST PRT SINLGE

$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

EXAMPLE DECK(STATS1/ANALYZE):
------- --------------------

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$, <N> PATCHES FOR STATS1

$*COMPILE STATS1/ANALYZE ALGOL LIBRARY

$*ALGOL FILE TAPE =  SYMBOL/STATS1  SERIAL

$*ALGOL FILE LINE =  STATS1/LISTING PRINT BACK UP DISK

$*DATA

$-

$ SET   TAPE SEQXEQ

$ RESET LIST PRT SINLGE

$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

EXAMPLE DECK(STATS2/ANALYZE):
------- --------------------

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$, <N> PATCHES FOR STATS2

$*COMPILE STATS2/ANALYZE ALGOL LIBRARY

$*ALGOL FILE TAPE =  SYMBOL/STATS2  SERIAL

$*ALGOL FILE LINE =  STATS2/LISTING PRINT BACK UP DISK

$*DATA

$-

$ SET   TAPE SEQXEQ

$ RESET LIST PRT SINLGE

$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

EXAMPLE DECK(STATS3/ANALYZE):
------- --------------------

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$, <N> PATCHES FOR STATS3

$*COMPILE STATS3/ANALYZE ALGOL LIBRARY

$*ALGOL FILE TAPE =  SYMBOL/STATS3  SERIAL

$*ALGOL FILE LINE =  STATS3/LISTING PRINT BACK UP DISK

$*DATA

$-

$ SET   TAPE SEQXEQ

$ RESET LIST PRT SINLGE

$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

EXAMPLE DECK(STATS4/ANALYZE):
------- ---------------------

```
        ?EXECUTE PATCH/MERGE
        ?DATA CARD
        $@ MERGE ZIP LIST CONFLICTS
        $. <N> PATCHES FOR STATS4
        $*COMPILE STATS4/ANALYZE ALGOL LIBRARY
        $*ALGOL FILE TAPE =  SYMBOL/STATS4  SERIAL
        $*ALGOL FILE LINE =  STATS4/LISTING PRINT BACK UP DISK
        $*DATA
        $-
        $ SET   TAPE SEQXEQ
        $ RESET LIST PRT SINLGE
        $ SET   LIST PRT SINGLE

        <SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

        ?END
```

EXAMPLE DECK(ONLINE/MAINT):
------- -------------------

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$. <N> PATCHES FOR OLMAINT

$*COMPILE ONLINE/MAINT TSPOL LIBRARY

$*TSPOL FILE TAPE =  SYMBOL/OLMAINT SERIAL

$*TSPOL FILE LINE = OLMAINT/LISTING PRINT BACK UP DISK

$*DATA

$-

$ SET   TAPE SEQXEQ

$ RESET LIST PRT SINLGE

$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

FXAMPLF DECK(LOGAN/DISK)

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@MERGE ZIP LIST CONFLICTS

$.<N> PATCHES FOR LOGAN

$*COMPILE LOGAN/DISK ALGOL LIBRARY

$*ALGOL FILE TAPE =  SYMBOL/LOGAN   SERIAL

$*ALGOL FILE LINE =   LOGAN/LISTING PRINT BACK UP DISK

$*DATA

$-

$ SET   TAPE SEQXEQ

$ RESET LIST PRT SINGLE

$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

EXAMPLE DECK(LOGANL/MAINT):
------- ------------------

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$. <N> PATCHES FOR MLOGAN

$*COMPILE LOGANL/MAINT ALGOL LIBRARY

$*ALGOL FILE TAPE =  SYMBOL/MLOGAN  SERIAL

$*ALGOL FILF LINE =  MLOGAN/LISTING PRINT BACK UP DISK

$*DATA

$-

$ SET   TAPE SEQXEQ

$ RESET LIST PRT SINLGE

$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

EXAMPLE DECK(LOGOUT/DISK):
------- -----------------

```
?EXECUTE PATCH/MERGE
?DATA CARD
$@ MERGE ZIP LIST CONFLICTS
$. <N> PATCHES FOR LOGOUT
$*COMPILE LOGOUT/DISK ALGOL LIBRARY
$*ALGOL FILF TAPE =  SYMBOL/LOGOUT  SERIAL
$*ALGOL FILE LINE =  LOGOUT/LISTING PRINT BACK UP DISK
$*DATA
$-
$ SET   TAPE SEQXEQ
$ RESET LIST PRT SINLGE
$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

EXAMPLE DECK(LOGOUTR/DISK):
------- ------------------

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$. <N> PATCHES FOR LOGOUTR

$*COMPILE LOGOUTR/DISK ALGOL LIBRARY

$*ALGOL FILE TAPE =  SYMBOL/LOGOUTR SERIAL

$*ALGOL FILE LINE = LOGOUTR/LISTING PRINT BACK UP DISK

$*DATA

$-

$ SET   TAPE SEQXEQ

$ RESET LIST PRT SINLGE

$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

EXAMPLE DECK(DCFILL/PRT):
------- ----------------

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$, <N> PATCHES FOR DCFILL

$*COMPILE DCFILL/PRT ALGOL LIBRARY

$*ALGOL FILE TAPE =  SYMBOL/DCFILL  SERIAL

$*ALGOL FILE LINE =  DCFILL/LISTING PRINT BACK UP DISK

$*DATA

$-

$ SET   TAPE SEQXEQ

$ RESET LIST PRT SINLGE

$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES, AND/OR LOCAL PATCHES>

?END
```

EXAMPLE DECK(DUMP/ANALYZE):
------- ------------------

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ MERGE ZIP LIST CONFLICTS

$. <N> PATCHES FOR DUMPANL

$*COMPILE DUMP/ANALYZE ALGOL LIBRARY

$*ALGOL FILE TAPE =  SYMBOL/DUMPANL SERIAL

$*ALGOL FILE LINE = DUMPANL/LISTING PRINT BACK UP DISK

$*FILE    P =   DCMCP/MEMDUMP PRINT BACK UP DISK

$*COMMON = 2

$*DATA

$-

$ SET   TAPE SEQXEQ

$ RESET LIST PRT SINLGE

$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

FXAMPLE DECK(TSFILL/PRT):
------- ----------------

```
?EXECUTE PATCH/MERGE
?DATA CARD
$@ MERGE ZIP LIST CONFLICTS
$. <N> PATCHES FOR TSFILL
$*COMPILE TSFILL/PRT ALGOL LIBRARY
$*ALGOL FILE TAPE =  SYMBOL/TSFILL  SERIAL
$*ALGOL FILE LINE =  TSFILL/LISTING PRINT BACK UP DISK
$*DATA
$-
$ SET   TAPE SEQXEQ
$ RESET LIST PRT SINLGE
$ SET   LIST PRT SINGLE

<SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

?END
```

EXAMPLE DECK(TSDUMP/ANALYZE):
------- --------------------

```
        ?EXECUTE PATCH/MERGE

        ?DATA CARD

        $@ MERGE ZIP LIST CONFLICTS

        $, <N> PATCHES FOR TSDUMP

        $*COMPILE TSDUMP/ANALYZE ALGOL LIBRARY

        $*ALGOL FILE TAPE =  SYMBOL/TSDUMP  SERIAL

        $*ALGOL FILF LINE =  TSDUMP/LISTING PRINT BACK UP DISK

        $*FILE    P =  TSSMCP/MEMDUMP PRINT BACK UP DISK

        $*COMMON = 3

        $*DATA

        $-

        $ SET   TAPE SEQXEQ

        $ RESET LIST PRT SINLGE

        $ SET   LIST PRT SINGLE

        <SUPPLEMENTARY PATCHES AND/OR LOCAL PATCHES>

        ?END
```

SECTION 9

STRUCTURE AND MAINTENANCE OF THE SYSTEM FILES

9.1 GENERAL

THIS SECTION DESCRIBES THE STRUCTURE OF THE VARIOUS FILES USED BY THE SYSTEM SOFTWARE AND THE PROGRAMS USED TO CREATE AND MAINTAIN THESE FILES.

[TO BE SUPPLIED LATER]

APPENDIX A

"PATCH/MERGE" DESCRIPTION

A.1 INTRODUCTION

THIS PROGRAM IS TO BE USED TO CREATE A MASTER PATCH DECK FROM THE INDIVIDUAL PATCH DECKS RELEASED BY BURROUGHS ON LIBRARY DUMP TAPES. THIS MASTER PATCH DECK IS THEN USED TO COMPILE THE PROGRAM FOR WHICH THE PATCHES WERE RELEASED. OFFICIAL BURROUGHS PATCH RELEASES ASSUME THAT THIS "PATCH/MERGE" PROGRAM IS BEING USED TO COMPILE THE PROGRAM BEING PATCHED.

A.2 INPUT PHASE

DURING THE INPUT PHASE, "PATCH/MERGE" READS ALL CARDS INPUT TO IT FROM CARDS OR,IF SPECIFIED,FROM DISK. THE INPUT IS ANALYZED TO SEE IF ALL THE PATCHES ARE FOR THE SAME PROGRAM, IF THERE ARE AS MANY CARDS IN EACH PATCH AS HAVE BEEN SPECIFIED, AND SO FORTH. IF ANY DISCREPANCIES ARE DETECTED, THEN AN APPROPRIATE, SELF-EXPLANATORY ERROR MESSAGE IS EMITTED, WHETHER LISTI OR LIST IS SET OR NOT.

IF LISTI OR LIST IS SET, ALL INPUT WILL BE LISTED AS IT IS ENCOUNTERED. IF THERE ARE DUPLICATE PATCH NUMBERS IN DIFFERENT FILES, THEN "PATCH/MERGE" WILL LIST THE DUPLICATES AND MARK THEM AS "DISCARDED" ON THE LISTING. IF THE DELETE OPTION HAS BEEN SPECIFIED, THEN "PATCH/MERGE" WILL LIST ALL DELETED PATCHES, MARKING THEM AS "DELETED" ON THE LISTING. IF ANY PATCH IS DESIGNATED TO BE DELETED ON A "$@" CARD, BUT "PATCH/MERGE" DOES NOT ENCOUNTER IT, THEN THE DELETE WILL BE IGNORED AND NO ACTION TAKEN. ANY PATCHES PRESENT IN THE INPUT WHICH ARE DISCARDED OR DELETED SHOULD NOT BE INCLUDED IN THE TOTAL NUMBER OF PATCHES SPECIFIED ON THE "$." CARD.

A.2.1 INPUT FILES

FROM ONE TO THREE FILES CAN BE USED AS INPUT TO "PATCH/MERGE":

	FILE NAME	FILE TYPE	PRECEDENCE	SYMBOL
1.	CARD	CARD	PRIMARY	C
2.	PATCH/<PROGRAM ID>	DISK	SECONDARY	P
3.	PATCHES/<PROGRAM ID>	DISK	TERTIARY	D

WHERE <PROGRAM ID> IS SPECIFIED ON A "$." CARD AND MAY BE ANY COMBINATION OF ALPHANUMERIC CHARACTERS UP TO SEVEN CHARACTERS IN LENGTH.

THE FILES, AS LISTED ABOVE, ARE IN DESCENDING ORDER OF PRECEDENCE. FOR EXAMPLE, IF THERE WERE A PATCH NUMBER 9 IN FILE CARD AND IN FILE PATCH/<PROGRAM ID>, THEN "PATCH/MERGE" WOULD USE THE PATCH FROM CARD AND DISCARD THE ONE IN PATCH/<PROGRAM ID>. SUPPOSING THERE WERE THREE DIFFERENT PATCHES NUMBERED 27, "PATCH/MERGE" WOULD AUTOMATICALLY USE THE ONE FROM THE FILE "CARD" AND DISCARD THE OTHER TWO.

THE NAME OF THE CARD READER INPUT FILE TO "PATCH/MERGE" MUST BE CARD. (SEE THE EXAMPLES BELOW.) THIS IS THE ONLY "PATCH/MERGE" INPUT FILE THAT CAN BE LABEL-EQUATED. FOR EXAMPLE:

```
CC FILE CARD=PMCARD SERIAL
```

THE THREE SYMBOLS INDICATED IN THE ABOVE TABLE ARE THOSE PRINTED ON THE INPUT, CONFLICTS, OR OUTPUT LISTINGS IF LISTI, CONFLICTS, OR LISTG IS SET. THEY INDICATE WHICH FILE A PARTICULAR PATCH IS BEING READ FROM.

A.2.2 "PATCH/MERGE" CONTROL CARDS

THERE ARE SIX TYPES OF "PATCH/MERGE" CONTROL CARDS:

1. $@	4. $-
2. $.	5. $:
3. $*	6. $#

"$@" CARD

THE FIRST CARD (AFTER THE "CC DATA CARD") INPUT TO "PATCH/MERGE" MUST BE EITHER A "$@" CARD OR A "$." CARD. THE "$@" CARD IS OPTIONAL AND, IF IT IS NOT SUPPLIED, THE OPTIONS CARD,LIST,AND ZIP ARE AUTOMATICALLY SET BY DEFAULT.

THE "$@" CARDS MUST HAVE THE CHARACTERS "$@" IN COLUMNS ONE AND TWO. THE REST OF THE CARD MAY CONTAIN ANY OPTIONS THE USER WISHES TO USE DURING THE CURRENT RUN OF "PATCH/MERGE". ANY OPTIONS NOT RECOGNIZED BY "PATCH/MERGE" WILL BE TREATED AS COMMENTS.

IN GENERAL, THE "$@" CARD IS EQUIVALENT TO THE $-CARDS USED WITH COMPILERS WITH ONE CRITICAL EXCEPTION: "$@" OPTIONS CAN NEVER BE RESET IN ANY PARTICULAR RUN, THAT IS, IF MORE THAN ONE "$@" CARD IS USED, THEN THE SECOND, THIRD, ETC. DOES NOT RESET ANY OPTIONS, BUT ONLY SETS THE OPTIONS SPECIFIED ON THAT PARTICULAR CARD.

THE "$@" OPTIONS AVAILABLE FOR "PATCH/MERGE" ARE:

CARD	ALL INPUT TO "PATCH/MERGE" WILL BE FROM THE CARD READER ONLY. "PATCH/MERGE" WILL IGNORE ANY PATCH FILES THAT MAY BE ON DISK.
CONFLICTS	"PATCH/MERGE" WILL LIST ON A LINE PRINTER ANY CONFLICTS AMONG PATCHES AND INDICATE HOW THESE CONFLICTS WERE RESOLVED. SEE THE SECTION "CONFLICTS (MERGE) PHASE."
DELETE	"PATCH/MERGE" WILL PASS OVER ANY PATCHES INPUT WHOSE NUMBERS FOLLOW THE DELETE OPTION, BUT LISTING THE DELETED

PATCHES ON THE LINE PRINTER IF LIST OR LISTI IS SET. FOR EXAMPLE, SUPPOSE THE FOLLOWING CARD WERE INPUT TO "PATCH/ MERGE":

```
$@ DELETE 4,20,103
```

"PATCH/MERGE" WOULD SKIP OVER PATCHES 4, 20, AND 103 IF THEY EXISTED IN ANY OF THE "PATCH/MERGE" INPUT FILES FOR THAT PARTICULAR RUN. IF THE DELETE OPTION IS USED, IT MUST EITHER BE THE LAST OPTION SPECIFIED ON A PARTICULAR "$@" CARD, OR IT MUST BE THE ONLY OPTION SPECIFIED ON A PARTICULAR "$@" CARD. ANY OPTIONS FOLLOWING THE DELETE OPTION ON ANY PARTICULAR "$@" CARD WILL BE IGNORED.

FINAL	THE USE OF THIS OPTION WILL CAUSE THE OPTIONS: MERGE, ZIP, NEW DISK, NONO, AND LIST TO ALL BE SET.
LIST	USE OF THIS OPTION WILL CAUSE THE OPTIONS: LISTI, LISTG, AND CONFLICTS TO BE SET.
LISTG	"PATCH/MERGE" WILL LIST THE SORTED PATCH DECK GENERATED.
LISTI	"PATCH/MERGE" WILL LIST ALL INPUT.
MERGE	IF AT LEAST ONE OF THE DISK FILES EXIST, INPUT TO "PATCH/ MERGE" WILL BE BOTH FROM CARDS AND FROM DISK. "PATCH/ MERGE" WILL AUTOMATICALLY CHECK IF EITHER OR BOTH OF THE FILES "PATCH/<PROGRAM ID>" OR "PATCHES/<PROGRAM ID>" ARE ON DISK. IF ONE OR BOTH IS ON DISK, THEN "PATCH/MERGE" WILL AUTOMATICALLY MERGE THE THREE INPUT FILES. IF NEITHER IS ON DISK, THEN "PATCH/MERGE" WILL USE THE FILE "CARD" FOR INPUT, ONLY.
NEW DISK	"PATCH/MERGE" WILL CREATE A NEW, UNSORTED MASTER FILE ON DISK CONSISTING OF ALL PATCHES INPUT WHICH WERE USED IN GENERATING THE SORTED PATCH DECK. "PATCH/MERGE" WILL NAME THIS FILE "PATCHES/<PROGRAM ID>". IF A PREVIOUS MASTER FILE EXISTED, IT WILL AUTOMATICALLY BE PURGED BEFORE THE NEW FILE IS CLOSED, AS WILL THE FILE PATCH/ <PROGRAM ID>. HOWEVER, IF ANY ERRORS ARE DETECTED BY "PATCH/MERGE" DURING THE INPUT PHASE, THEN NO NEW DISK FILE WILL BE CREATED AND NO DISK FILES WILL BE PURGED, EVEN THOUGH NEW DISK IS SET.
NONO	"PATCH/MERGE" WILL NOT ATTEMPT TO PUT THE PATCH NUMBER ON EACH GENERATED PATCH CARD. IF THIS OPTION IS *NOT* SET,

	THEN ON EACH PATCH CARD GENERATED THAT HAS COLUMNS 68-72 BLANK, "PATCH/MERGE" WILL AUTOMATICALLY INSERT THE PATCH NUMBER OF THE PATCH TO WHICH THE CARD BELONGS.
PUNCHG	"PATCH/MERGE" WILL PUNCH OUT ON AN ON-LINE CARD PUNCH THE GENERATED PATCH DECK, EXCLUSIVE OF MCP AND "PATCH/MERGE" CONTROL CARDS.
PUNCHI	"PATCH/MERGE" WILL PUNCH OUT ON AN ON-LINE CARD PUNCH A DECK CONSISTING OF ALL INPUT PATCHES USED IN GENERATING THE SORTED PATCH DECK.
ZIP	"PATCH/MERGE" WILL AUTOMATICALLY ZIP THE GENERATED PATCH DECK.
NEWFILE	"PATCH/MERGE" WILL CREATE A NEW UNSORTED FILE "PATCH/ <PROGRAM ID>" ON DISK CONSISTING OF ALL THE PATCHES INPUT VIA THE CARD READER. IF A FILE EXISTS, IT WILL BE PURGED AFTER IT HAS BEEN PROCESSED AND A NEW FILE WILL BE CREATED PROVIDED THE FOLLOWING CONDITIONS ARE MET. FIRST, "NEW" OPTION MUST BE SET, SECOND, THERE MUST BE CARD INPUT, AND THIRD, ALL INPUT DISK AND CARD MUST BE ERROR FREE.

"$." CARD

THE "$." CARD MUST HAVE THE CHARACTERS "$." IN COLUMNS ONE AND TWO. THE REST OF THE CARD MUST CONTAIN THE NUMBER OF PATCHES AND THE IDENTIFICATION OF THE PROGRAM WHICH THE PATCHES ARE FOR. THE FORMAT OF THIS CARD IS:

```
$. <NUMBER OF PATCHES> FOR <PROGRAM ID> <COMMENTS>
```

WHERE <NUMBER OF PATCHES> MUST BE AN INTEGER BETWEEN ZERO AND 999, INCLUSIVE, AND <PROGRAM ID> MAY BE ANY IDENTIFIER UP TO SEVEN ALPHANUMERIC CHARACTERS IN LENGTH. IF <NUMBER OF PATCHES> IS ZERO, THEN "PATCH/MERGE" WILL AUTOMATICALLY CHANGE IT TO ONE.

"$*" CARD

"PATCH/MERGE" CREATES AND ZIPS A CONTROL DECK CONTAINING THE MERGED PATCHES. THE MCP CONTROL CARDS USED IN THIS CONTROL DECK MUST BE SPECIFIED BY CARDS WHICH HAVE THE CHARACTERS "$*" IN COLUMNS ONE AND TWO. ("PATCH/MERGE" WILL AUTOMATICALLY SUBSTITUTE A QUESTION MARK AND A BLANK IN COLUMNS ONE AND TWO FOR THE "$*" SUPPLIED). THE REMAINING SEVENTY COLUMNS CAN CONTAIN ANY CONTROL CARD INFORMATION DESIRED. THIS TYPE OF CARD MUST PRECEDE ALL "PATCH/MERGE" CONTROL CARDS EXCEPT THE "$." OR "@" CARD. ANY INPUT TO "PATCH/MERGE" MUST CONTAIN AT LEAST ONE "$*" CARD.

NOTE:THERE MUST BE AT LEAST ONE PSEUDOREADER IN USE OR THE CONTROL DECK WILL NOT BE SCHEDULED BY THE MCP.

"$-" CARD

PATCHES WHICH ARE NOT TO BE MERGED (E.G. DOLLAR CARDS) MAY BE INCLUDED AS THE FIRST PATCH DECK. THE HEADER CARD FOR THIS DECK MUST HAVE THE CHARACTERS "$-" IN COLUMNS ONE AND TWO. THE REMAINDER OF THE CARD MAY CONTAIN COMMENTS. THE CARDS IN THIS DECK WILL BE THE FIRST CARDS OF THE MASTER PATCH DECK.

"$!" CARD

THE "$!" CARD IS NOT A CONTROL CARD, BUT A COMMENT CARD. IT IS USED TO INTERSPERSE COMMENTS WITH "PATCH/MERGE" INPUT. IN GENERAL, ALL "$!" CARDS WILL BE LISTED IF LISTI IS SET, BUT WILL OTHERWISE BE IGNORED BY "PATCH/MERGE". IF THE NEW DISK OPTION HAS BEEN SET, THEN "PATCH/MERGE" WILL INCLUDE THESE COMMENT CARDS IN THE NEW DISK FILE.

"$#" CARD

INDIVIDUAL PATCH DECKS MUST BEGIN WITH A CARD WHICH HAS "$#" IN

COLUMNS ONE AND TWO AND THE PATCH IDENTIFICATION IN COLUMNS THREE THROUGH SEVENTY-TWO. THE STANDARD FORMAT FOR THIS CARD IS:

```
$# PATCH NUMBER <INTEGER> FOR <PROGRAM ID> CONTAINS <INTEGER> CARDS
```

WHERE THE <INTEGER> SPECIFYING PATCH NUMBER MUST BE AN INTEGER BETWEEN ZERO AND 999, INCLUSIVE, AND THE <INTEGER> SPECIFYING THE NUMBER OF CARDS MUST, MINIMALLY, BE ONE.

AN ERROR WILL BE INDICATED IF THE PATCH DECKS ARE NOT IN ASCENDING ORDER, IF THE PROGRAM IDENTIFICATION DOES NOT AGREE WITH THAT OF THE "$." CARD, IF THE NUMBER OF CARDS PRESENT DOES NOT AGREE WITH THE NUMBER OF CARDS GIVEN ON THIS CARD, ETC. ALL PATCHES RELEASED BY BURROUGHS HAVE THE PROPER HEADER CARD AS THE FIRST CARD OF THE DECK.

A.3 CONFLICTS (MERGE) PHASE

BESIDES MERGING PATCH DECKS, "PATCH/MERGE" RESOLVES CONFLICTS AMONG PATCH DECKS. IF TWO OR MORE PATCH DECKS HAVE IDENTICAL SEQUENCE NUMBERS, THEN "PATCH/MERGE" WILL DISCARD ALL BUT THE ONE FROM THE DECK WITH THE HIGHEST PATCH NUMBER.

"$VOID" CARDS

IF A VOID CARD IS TO BE DISCARDED, SPECIAL ACTION IS TAKEN SO THAT THE VOIDING IS ACCOMPLISHED. WHEN A VOID CARD IS ENCOUNTERED, CARDS IN THE VOIDED RANGE ARE DISCARDED FROM THE PATCH DECK IN WHICH THE VOID WAS FOUND, AND IN ALL DECKS OF LOWER PATCH NUMBER. VOID CARDS IN THE DECKS OF LOWER PATCH NUMBER ARE ALSO SO HANDLED, WHEN THEY ARE ENCOUNTERED WHILE DISCARDING. IF THERE ARE CARDS IN DECKS WITH HIGHER PATCH NUMBERS THAN THE VOID CARD WHICH FALL INTO THE VOIDING RANGE, THE NECESSARY VOID CARDS ARE GENERATED SO THAT THE TAPE AREA IS VOIDED BUT THE PATCH CARDS FROM THOSE HIGHER NUMBERED DECKS ARE NOT.

NOTE: FOR THE PURPOSES OF THIS PROGRAM, "VOID" CARDS MUST HAVE THEIR "$" IN COLUMN ONE, AND THE VOIDING SEQUENCE AND RANGE MUST BE EIGHT DIGITS IN LENGTH.

"$VOIDT" CARDS

IN THE MERGE PHASE, "$VOIDT" CARDS ARE TREATED THE SAME AS "$VOID" CARDS WITH RESPECT TO PATCHES LOWER IN NUMBER THAN THE PATCH CONTAINING THE "$VOIDT" CARD. CARDS IN THE "$VOIDT" RANGE WHICH ARE CONTAINED IN PATCHES WITH A NUMBER THE SAME AS OR HIGHER THAN THE PATCH CONTAINING THE "$VOIDT" CARD ARE SIMPLY INSERTED IN THE GENERATED PATCH DECK ALONG WITH THE "$VOIDT" CARD.

WORK FILES

THE WORK FILES USED BY "PATCH/MERGE" HAVE TWENTY ROWS OF 300 RECORDS EACH. THE NUMBER OF ROWS MAY BE CHANGED BY SPECIFYING THE DESIRED VALUE ON A "COMMON" CONTROL CARD WHEN EXECUTING "PATCH/MERGE".

A.4 DECK SET UP AND EXAMPLES

INPUT FROM CARDS ONLY

LET US ASSUME THAT WE WANT TO COMPILE A PROGRAM WHICH HAS A SOURCE FILE, "SOURCE/PRGXII", ON TAPE. WE WANT TO ADD THREE PATCHES TO THIS FILE, AND WE HAVE THE THREE PATCHES PUNCHED ON CARDS SUITABLE FOR INPUT TO "PATCH/MERGE". WE COULD USE THE FOLLOWING DECK SET UP:

```
? EXECUTE PATCH/MERGE

? DATA CARD

$. 3 PATCHES FOR PRGXII

$*COMPILE OBJECT/TST WITH ALGOL FOR LIBRARY

$*ALGOL FILE TAPE=SOURCE/PRGXII TAPE

$*DATA CARD

$-

$TAPE LIST

$#PATCH NUMBER 1 FOR PRGXII CONTAINS 20 CARDS

.      .      .

.      .      .

<PATCH DECK>

.      .      .

.      .      .

$#PATCH NUMBER 2 FOR PRGXII CONTAINS 4 CARDS
```

```
•        •        •

•        •        •

<PATCH DECK>

•        •        •

•        •        •

$#PATCH NUMBER 3 FOR PRGXII CONTAINS 14 CARDS

•        •        •

•        •        •

<PATCH DECK>

•        •        •

•        •        •

? END.
```

SINCE THE "$@" CARD IS MISSING FROM THE ABOVE INPUT DECK, THE OPTIONS CARD,LIST, AND ZIP WILL AUTOMATICALLY BE SET BY DEFAULT. THEREFORE, EVEN IF ONE OR BOTH OF THE FILES "PATCH/PRGXII" OR "PATCHES/PRGXII" ARE ON DISK, "PATCH/MERGE" WILL NOT ATTEMPT TO USE THEM AS INPUT, BUT USE ONLY THOSE PATCHES INCLUDED IN THE CARD DECK.

NOTE THAT, ALTHOUGH "PATCH/MERGE" DOES NOT REQUIRE QUOTES AROUND A <PROGRAM ID> WHICH HAS A SPECIAL CHARACTER(S), ANY MCP CONTROL CARDS USED DO.

INPUT FROM CARDS AND DISK

"PATCH/MERGE" WILL ACCEPT INPUT FROM ONE, TWO, OR THREE FILES: ONE CARD FILE AND TWO DISK FILES. ALL "PATCH/MERGE" CONTROL CARDS INCLUDING: "$@" CARDS, "$." CARDS, "$*" CARDS AND "$-"CARDS, MUST BE ENTERED FROM A CARD READER FILE WHOSE NAME IS CARD. IN OTHER WORDS, ALL "PATCH/MERGE" AND $ CONTROL CARDS UP TO BUT NOT INCLUDING THE FIRST "$#" CARD MUST BE ENTERED FROM A CARD READER FILE LABELED CARD.

FOR EXAMPLE, ASSUMING THAT AT LEAST ONE DISK FILE WITH PATCHES IN IT WERE PRESENT ON DISK, ONE MIGHT ENTER THE FOLLOWING FROM A CARD READER:

```
?EXECUTE PATCH/MERGE

?DATA CARD

$@ FINAL

$. 5 PATCHES FOR ALGOLX

$*COMPILE ALGOL/DISC WITH ALGOL TO LIBRARY

$*ALGOL FILE TAPE=SYMBOL/ALGOL SERIAL

$*FILE LINE=LINE PRINT OR BACK UP

$*DATA CARD

$-

$ SET   TAPE SEQXEQ ALGOL RESET TSPOL

$ RESET LIST PRT SINGLE

$ SET   LIST PRT SINGLE

?END.
```

IF THE ABOVE DECK WERE INPUT FROM A CARD READER, THEN A TOTAL OF FIVE PATCHES (INDICATED BY THE "$." CARD) MUST BE INCLUDED IN ONE OR BOTH FILES ON DISK NAMED "PATCH/ALGOLX" OR "PATCHES/ALGOLX".

SINCE THE OPTION FINAL IS SET:

1. THE OPTIONS MERGE, LISTI, CONFLICTS, LISTG, NONO, NEW DISK, AND ZIP WILL AUTOMATICALLY BE SET BY "PATCH/MERGE". THEREFORE, "PATCH/MERGE" WILL LOOK FOR THE FILES "PATCH/ALGOLX" AND "PATCHES/ALGOLX" ON DISK.

2. THE INPUT WILL BE LISTED SINCE LISTI IS SET.

3. THE CONFLICTS (IF ANY) WILL BE LISTED, SINCE CONFLICTS IS SET.

4. THE GENERATED PATCH DECK WILL BE LISTED, SINCE LISTG IS SET.

5. THE GENERATED PATCH DECK WILL BE ZIPPED, SINCE ZIP IS SET.

6. PATCH NUMBERS WILL NOT BE PUT IN COLUMNS 68-72 OF THE GENERATED PATCH CARDS, SINCE NONO IS SET.

7. FINALLY, A NEW FILE NAMED "PATCHES/ALGOLX" WILL BE CREATED ON DISK CONTAINING ALL THE PATCHES USED IN GENERATING THE SORTED PATCH DECK, AND THE OLD FILES "PATCH/ALGOLX" AND "PATCHES/ ALGOLX" WILL BE PURGED. COMMENT DOCUMENT

APPENDIX B

B5700 REMOTE JOB ENTRY

B.1 INTRODUCTION

THIS APPENDIX DESCRIBES THE REMOTE JOB ENTRY SYSTEM DESIGNED TO RUN WITH THE DATACOMM MCP AND THE B249-487 HARDWARE. IT DESCRIBES THE CONCEPT, OPERATION, AND IMPLEMENTATION OF THE REMOTE JOB ENTRY SYSTEM.

THE REMOTE JOB ENTRY SYSTEM (HEREAFTER CALLED RJE) WAS DESIGNED IN AN EFFORT NOT TO AFFECT THE OPERATION OF THE DATACOMM MCP. IF THE MCP COMPILE TIME OPTION RJE IS SET FALSE THE MCP WILL LOOK PRECISELY AS IT DID BEFORE RJE. IF THE RJE OPTION IS COMPILED TRUE, THEN THE NORMAL MCP OPERATIONS AND NORMAL DATACOMM WILL NOT BE AFFECTED IN ANY WAY.

THE DISCUSSION OF THIS DOCUMENT IS RESTRICTED MAINLY TO THE RJE PART OF THE SYSTEM AND NOT THE NORMAL MCP OPERATIONS. SOME OF THE LATER SECTIONS OF THIS DOCUMENT ARE RELATED TO THE IMPLEMENTATION FROM THE SOFTWARE POINT OF VIEW. THEY SHOULD BE OF INTEREST TO SYSTEMS PROGRAMMERS ATTEMPTING TO MODIFY OR DEBUG THE SYSTEM. THEY NEED NOT BE UNDERSTOOD IN ORDER TO OPERATE THE RJE SYSTEM.

B.2 REMOTE JOB ENTRY CONCEPT

B.2.1 INTRODUCTION

IN A CONVENTIONAL SYSTEM A LARGE AMOUNT OF TURN AROUND DELAY IS ENCOUNTERED SINCE ALL PROGRAMS AND INPUT DATA MUST BE PHYSICALLY TRANSPORTED TO THE CENTRAL SITE AND ALL OUTPUT DATA MUST BE TRANSPORTED BACK. THIS CAN CAUSE ADDITIONAL DELAYS IN THAT A COMPLEX ROUTING CONTROL MUST BE ESTABLISHED TO INSURE THAT THE PROPER OUTPUT DATA IS RETURNED TO THE CORRECT ORIGINATOR.

THE ADVENT OF HIGH SPEED TRANSMISSION LINES AND SMALL TERMINAL PROCESSORS MAKES IT POSSIBLE TO ELIMINATE SOME OF THESE PROBLEMS BY INTRODUCING THE CONCEPT OF REMOTE JOB ENTRY. THE PURPOSE OF A REMOTE JOB ENTRY SYSTEM IS TO INCREASE THE UTILIZATION AND CONVENIENCE OF THE CENTRAL DATA PROCESSING SYSTEM LOCATED SOME DISTANCE AWAY FROM WHERE ITS INPUT DATA IS PRODUCED AND ITS OUTPUT DATA UTILIZED.

B.2.2 CONCEPT

REMOTE JOB ENTRY INVOLVESA REMOTE SATELLITE SYSTEM CONNECTED TO A CENTRAL SYSTEM VIA A COMMUNICATIONS LINE. ALSO CONNECTED TO THE REMOTE SATELLITE ARE PERIPHERAL DEVICES SUCH AS CARD READERS, LINE PRINTERS, ETC. THE OBJECT OF SUCH A SYSTEM IS TO ALLOW:

1. INTRODUCTION OF PROGRAMS FROM A REMOTE READER FOR EXECUTION BY THE CENTRAL SYSTEM.
2. INTRODUCTION OF DATA FROM A REMOTE READER FOR PROCESSING BY THE CENTRAL SYSTEM.
3. OUTPUT DATA PRODUCED BY THE CENTRAL SYSTEM TO BE PRINTED ON THE REMOTE LINE PRINTER.
4. MONITORING AND CONTROLLING OF PROGRAMS ON THE CENTRAL SYSTEM VIA A REMOTE SUPERVISORY CONSOLE.

B.2.3 REMOTE DATA/PROGRAM INTERFACE

THERE EXIST TWO TIME RELATIONSHIP MODES BETWEEN DATA AND PROGRAMS: CONVERSATIONAL AND SPOOLED.

FOR CONVERSATIONAL MODE AN ELEMENT OF REMOTE DATA MUST BE MADE AVAILABLE TO PROCESSING PROGRAM IMMEDIATELY UPON ITS RECEPTION BY THE CENTRAL SYSTEM. LIKEWISE, FOR OUTPUT EACH ELEMENT OF DATA MUST BE RETURNED TO THE REMOTE SITE AS SOON AS IT IS PRODUCED BY THE PROCESSING PROGRAM. THIS IS REQUIRED FOR THOSE CASES, WHERE EACH INPUT IS DEPENDENT UPON THE LAST PREVIOUS OUTPUT. THIS MODE WILL BE IMPLEMENTED ONLY THROUGH THE USE OF THE REMOTE SPO CONSOLE. A PROGRAM CAN USE THE

CONVERSATIONAL MODE BY DECLARING A SPO (TYPE 11) FILE AND ALL I/O TO THIS FILE WILL BE DONE WITH THE RJE TERMINALS SPO.

FOR THE SPOOLED MODE ALL INPUT IS TO BE COLLECTED ON A CENTRAL SYSTEM HIGH SPEED DEVICE BEFORE THE PROGRAM, WHICH IS TO PROCESS THAT DATA, IS ACTIVATED ON THE CENTRAL SYSTEM. LIKEWISE, OUTPUT DATA IS COLLECTED ON A CENTRAL SYSTEM HIGH SPEED DEVICE AND TRANSMITTED TO THE REMOTE DEVICE ONLY AFTER ALL OUTPUT DATA HAS BEEN PRODUCED. THIS MODE IS USED WHEN THE PROCESSING SYSTEM UTILIZES LARGE AMOUNTS OF SYSTEM RESOURCES WHILE PROCESSING. BY COLLECTING DATA ON A CENTRAL SYSTEM HIGH SPEED DEVICE, THE TIME A PROCESSING PROGRAM TIES UP SYSTEM RESOURCES IS MEASUREABLY REDUCED. THIS IS, OF COURSE, ASSUMING THAT THE DATA COLLECTION PROCESS TAKES A MINIMUM OF SYSTEM RESOURCES AND ALSO THAT ADEQUATE SPACE EXISTS ON THE CENTRAL SYSTEM HIGH SPEED DEVICE. THIS METHOD WILL BE USED FOR THE I/O TO ALL THE RJE DEVICES EXCEPT THE SPO. INPUT FROM THE CARD READER WILL BE PLACED IN DECKS AS IF LOAD CONTROL WERE RUNNING FROM ITS CARD READER. OUTPUT TO THE PRINTER OR PUNCH WILL BE ROUTED TO BACK UP DISK AND PRINTED/PUNCHED WHEN THE FILES ARE CLOSED AND THE REMOTE PRINTER/PUNCH BECOMES AVAILABLE.

8.2.4 REMOTE OPERATOR/SYSTEM INTERFACE

AS IN ALL DATA-PROCESSING SITUATIONS, CONDITIONS ARISE WHILE PROCESSING WHICH REQUIRE HUMAN INTERVENTION AND DECISION MAKING AT THE TIME THE CONDITION ARISES. THIS IMPLIES A HUMAN INTERFACE TO THE SYSTEM. THE TYPES OF INTERFACE ARE:

1. PROGRAM INTERFACE

   CERTAIN SITUATIONS ENCODED FOR DETECTION BY PROGRAMS MAY BE CORRECTED BY HAVING THE PROGRAM REQUEST CORRECTION INFORMATION FROM THE REMOTE OPERATOR. THIS IS THE ONLY FORM OF CONVERSATIONAL MODE I/O ALLOWED WITH THE RJE TERMINAL.

2. OPERATING SYSTEM INTERFACE

   CERTAIN SITUATIONS ENCOUNTERED BY THE OPERATING SYSTEM MAY BE CORRECTED BY HAVING THE OPERATING SYSTEM REQUEST CORRECTION INFORMATION FROM THE REMOTE OPERATOR.

3. SYSTEM OPERATOR

SITE CONDITIONS MAY ARISE THAT CAN NOT BE HANDLED BY THE REMOTE OPERATOR. FOR THIS REASON IT MAY BE NECESSARY FOR THE REMOTE OPERATOR TO INFORM THE CENTRAL SYSTEM OPERATOR OF CORRECTIONS NEEDED. THE SYSTEM OPERATOR REMAINS IN TOTAL CONTROL OF THE SYSTEM. HE WILL BE ABLE TO DIRECTLY AFFECT ALL JOBS INITIATED FROM THE RJE TERMINALS AS WELL AS THOSE FROM THE MAIN SITE.

4. OPERATING SYSTEM INFORMATION

THE REMOTE OPERATOR MAY WISH TO REQUEST INFORMATION FROM THE MCP CONCERNING JOBS HE HAS INITIATED OR GENERAL MCP INFORMATION. HE MAY ALSO WANT TO ABNORMALLY AFFECT HIS JOBS, SUCH AS DS-ING THEM. HOWEVER, HE MAY NOT AFFECT ANY OTHER JOBS OR THE SYSTEM AS A WHOLE.

B.3 ELEMENTS OF THE RJE SYSTEM

B.3.1 CONFIGURATION OF THE REMOTE TERMINAL

THE PROGRAM GENERATOR IS CAPABLE OF PRODUCING AN EXECUTIVE TO RUN ANY DC1100 WITH 8192 OR MORE BYTES OF STORAGE. IF AN ATTEMPT IS MADE TO RUN THE EXECUTIVE ON A DC1200, HANGS AND ERRORS MAY RESULT.

THE MINIMUM SYSTEM WHICH THE EXECUTIVE IS CAPABLE OF OPERATING IS THE DC1102. THE DC1102 MUST BE EQUIPPED WITH CARD READER, SPO, SINGLE LINE DATA COMMUNICATIONS CONTROL AND 8192 BYTES OF CORE MEMORY.

THE MAXIMUM SYSTEM IS A DC1103 WITH CARD READER, CARD PUNCH, SPO, LINE PRINTER, SINGLE LINE DATACOM CONTROLLER AND 8192 TO 32768 BYTES OF MEMORY.

THE CARD READER IS A BURROUGHS 200 CARD/MINUTE READER. THIS UNIT IS REQUIRED SO THAT THE EXECUTIVE PROGRAM PRODUCED ON THE B5700 CAN BE LOADED INTO THE MEMORY OF THE DC1000. WHILE THIS UNIT IS REQUIRED FOR INITIALIZATION OF THE SYSTEM, AN EXECUTIVE ROUTINE CAN BE GENERATED WHICH DOES NOT USE THE CARD READER. THE RESULT OF GENERATING A READERLESS EXECUTIVE IS A REDUCTION OF THE CORE REQUIRED FOR PROPER OPERATION .

THE SPO IS A MODIFIED TWX, AND IS REQUIRED FOR OPERATION OF THE RJE

SYSTEM. THE USES OF THE REMOTE SPO ARE AS FOLLOWS:

1. THE BCL CARD LOADER IS READ INTO HIGH CORE FROM THIS DEVICE BY THE BOOTSTRAP LOADER.

2. THE SPO IS REQUIRED FOR LOGIN UPON DIALING IN TO THE SYSTEM OR AFTER A HALT LOAD.

3. CONTROL MESSAGES PERTAINING TO JOBS STARTED BY THE RJE TERMINAL, AND PERTAINING TO PERIPHERALS ATTACHED TO THE TERMINAL ARE PRINTED ON THE REMOTE SPO.

THE EXECUTIVE ALWAYS CONTAINS CODE TO HANDLE THE SPO.

THE THIRD I/O DEVICE WHICH CAN BE CONNECTED TO THE DC1000 IS A LINE PRINTER. A WIDE RANGE OF PRINTERS IS AVAILABLE. PRINTERS CAPABLE OF PRINTING THE 64 CHARACTER SET IN 80-132 CHARACTER LINE LENGTHS AT SPEEDS AS HIGH AS 400 LINES/MINUTE CAN BE INTERFACED TO THE DC1000. THIS DEVICE IS OPTIONAL. EXECUTIVE PROGRAMS WITHOUT PROVISION FOR PRINTER OUTPUT REQUIRE LESS MEMORY.

A CARD PUNCH IS ALSO AVAILABLE ON THE DC1000. WHILE THE EXECUTIVE IS NOT CURRENTLY CAPABLE OF OUTPUTTING TO THE PUNCH, THE PRESENCE OF A PUNCH WILL NOT ADVERSLY AFFECT THE OPERATION OF THE OTHER FUNCTIONS OF THE EXECUTIVE.

THE LAST I/O CONTROLLER AVAILABLE ON THE DC1000 IS THE DATA COMMUNICATIONS CONTROLLER. THE DC-1365-10 IS A HALF OR FULL DUPLEX SYNCHRONOUS CONTROL THAT CAN INTERFACE TO ANY SYNCHRONOUS MODEM CONFORMING TO RS232C INTERFACE SPECIFICATIONS. THE DC-1365-10 CAN TRANSFER DATA AT UP TO 4800 BAUD.

OTHER PERIPHERAL DEVICES ARE AVAILABLE ON THE DC1000. HOWEVER, THE B5700 RJE SYSTEM IS NOT CAPABLE OF UTILIZING THEM. THEY CAN BE CONNECTED TO THE DC1000 AND LEFT OFF LINE WHILE THE DC1000 IS PERFORMING RJE FUNCTIONS, WITHOUT ADVERSELY AFFECTING THE PERFORMANCE OF THE EXECUTIVE. NOTE THAT THE MULTILINE DATACOM CONTROL IS AN EXCEPTION TO THIS RULE.

THE DC1000 MEMORY IS AN ALTERABLE CORE MEMORY WITH A 1.5 MICROSECOND CYCLE TIME. MEMORY IS ORGANIZED INTO NINE BIT WORDS CONSISTING OF EIGHT DATA BITS AND A PARITY BIT. MEMORY IS AVAILABLE IN 4096 WORD INCREMENTS. A MAXIMUM OF EIGHT MODULES OF MEMORY CAN BE ACCOMODATED BY THE DC1000.

THE AMOUNT OF MEMORY NEEDED TO INSURE PROPER OPERATION OF THE EXECUTIVE IS DICTATED BY THE CONFIGURATION AND THE USE TO WHICH THE SYSTEM IS TO BE PUT.

B.3.2 DATACOMM LINK

1. CONTROL

   THE B249 IS THE DATACOMM CONTROL.

2. TERMINAL UNIT

   THE B487 DTTU IS USED AS THE TERMINAL UNIT TO HOLD THE ADAPTERS FOR THE RJE LINES AS WELL AS OTHER DATACOMM.

3. ADAPTERS

   THE DC1000 WILL RUN IN A SYNCHRONOUS MODE. HENCE, THE STANDARD SYNCHRONOUS ADAPTER (B995-10 OR B5665-10) IS USED. IT IS SET UP AS A POINT TO POINT ADAPTER WITH CALL UP/DISCONNECT CAPABILITY.

4. DATA SET

   THE EXPECTED MODE OF OPERATION IS THROUGH THE DIAL UP OF THE SYSTEM VIA THE 201A SYNCHRONOUS DATA SET. THIS ALLOWS AN ADAPTER (RJE LINE) TO BE USED BY MORE THAN ONE USER (THOUGH NOT SIMULTANEOUSLY).

5. LINE SPEED

   THE 201A RUNS AT A LINE SPEED OF 2000 BAUD. FASTER LINE SPEEDS AND DATA SET TURNAROUND TIMES CAN BE GAINED BY GOING TO A LEASED LINE SETUP USING DATA SETS SUCH AS THE 201B, RIXON, ETC.

B.3.3 MCP

THERE WILL BE NO CURTAILMENT OR DEGRADATION OF NORMAL SITE OPERATIONS. THE OPERATOR AT THE CENTRAL SITE CAN PREVENT ANY RJE JOBS FROM RUNNING BY BEING SELECTIVE IN HIS USE OF PSEUDO- READERS.

CERTAIN LINES (TERMINAL UNIT/BUFFER NUMBERS) WILL BE MARKED BY THE MCP AS DEDICATED FOR EXCLUSIVE USE FOR RJE TERMINALS. THERE WILL BE MORE DISCUSSION CONCERNING THIS LATER IN THIS DOCUMENT.

THERE WILL BE NO CURTAILMENT OR EFFECT ON OTHER DATACOMM USE BY THE MCP OR OBJECT PROGRAMS EXCEPT TO EXCLUDE THE USE OF RJE LINES FOR ANY OTHER DATACOMM USE EXCEPT RJE.

B.4 RUNNING THE SYSTEM

B.4.1 COMPILING THE MCP

THE MCP HAS A NEW COMPILE TIME OPTION, RJE. IF RJE IS SET FALSE THE REGULAR MCP (DEPENDENT UPON WHICH OPTIONS ARE COMPILED) WILL REMAIN. THERE WILL BE NO CAPABILITY TO RUN REMOTE JOB ENTRY IN THIS CASE.

IF RJE IS SET TRUE THE FOLLOWING OPTIONS MUST BE SET AS INDICATED BELOW.

1. DATACOM TO TRUE

2. DCSPO TO TRUE

3. INQUIRY TO FALSE.

4. OTHERS MAY BE SELECTED AS DESIRED.

B.4.2 REMOTE USERS FILE

THE "REMOTE/USERS" FILE HAS BEEN EXTENDED TO ALLOW THE SITE TO DECLARE WHICH LINES (TERMINAL UNIT/BUFFER NUMBER) WILL BE USED FOR REMOTE JOB LINES. ONCE DECLARED, THESE LINES MAY NOT BE USED FOR NORMAL DATACOM INTERFACE WITH PROGRAMS. THAT IS, NO PROGRAM MAY LABEL EQUATE A TYPE 14 OR TYPE 19 FILE TO A TERMINAL UNIT/BUFFER NUMBER THAT HAS BEEN DECLARED IN "REMOTE/USERS" AS AN RJE LINE.

THE PROGRAM "UPDATE/USERS" HAS BEEN MODIFIED TO PERFORM THE CHANGES ON THE "REMOTE/USERS" FILE TO ALLOW IT TO SPECIFY RJE LINES. THE DATA CARDS USED TO ADD AND DELETE RJE LINES FROM "REMOTE/USERS" ARE:

```
RJEADD <TERMINAL UNIT>/<BUFFER NUMBER>
RJEDEL <TERMINAL UNIT>/<BUFFER NUMBER>
```

THUS A TYPICAL DECK TO CHANGE THE RJE LINES SPECIFIED IN "REMOTE/USERS" WOULD LOOK LIKE :

```
?EXECUTE UPDATE/USERS
?DATA CODES
RJEADD 2/8
RJEADD 4/12
RJEDEL 1/0
?END
```

THIS DECK WOULD ADD LINES 2/8 AND 4/12 AND DELETE 1/0 FROM THE PREVIOUS "REMOTE/USERS" FILE.

B.4.3 GENERATION OF THE DC1000 EXECUTIVE PROGRAM

THE EXECUTIVE ROUTINES NEEDED TO ENABLE THE DC1000 TO FUNCTION AS AN RJE TERMINAL ARE PRODUCED BY A PROGRAM GENERATOR. THE PROGRAM GENERATOR IS A B5700 EXTENDED ALGOL PROGRAM WHICH PRODUCES A DISK FILE AND A CARD DECK WHICH CONTAINS THE OBJECT CODE FOR THE EXECUTIVE. THE PROGRAM GENERATOR IS CAPABLE OF PRODUCING EXECUTIVE ROUTINES ABLE TO OPERATE SPO, CARD READER, LINE PRINTER AND THE SINGLE LINE SYNCHRONOUS DATACOM CONTROL.

THE CAPABILITIES OF THE EXECUTIVE ARE SPECIFIED BY THE FIRST BLOCK OF DEFINES IN THE PROGRAM GENERATOR. THE SPECIFICATIONS NEEDED TO PRODUCE AN EXECUTIVE ARE:

1. MEMORY - NUMBER OF BYTES OF DC1000 MEMORY.

2. READER - TRUE IF A READER IS TO BE USED, OTHERWISE FALSE.

3. PRINTER - TRUE IF LINE PRINTER OUTPUT IS TO BE HANDLED, OTHERWISE FALSE.

4. PRINTER LINE LENGTH - NUMBER OF CHARACTERS IN A FULL LINE OF PRINTER OUTPUT.

THESE OPTIONS ARE SPECIFIED PRIOR TO COMPILING THE PROGRAM GENERATOR WITH THE ALGOL COMPILER. THE PROGRAM GENERATOR, ONCE COMPILED, WILL ALWAYS PRODUCE AN EXECUTIVE AS SPECIFIED BY THE ABOVE DEFINES. NOTE THAT AN ALGOL STACK OF 800 WORDS IS REQUIRED FOR SUCESSFUL COMPILATION.

THERE ARE TWO OBJECT TIME OPTIONS FOR THE PROGRAM GENERATOR, THEY ARE:

1. PUNCH ONLY - A COMMON VALUE OF 1 SPECIFIES THAT THE PROGRAM GENERATOR IS TO PUNCH THE CONTENTS OF THE DISK FILE ONLY.

2. NO LIST - A COMMON VALUE OF 2 SPECIFIES THAT THE GENERATOR IS TO PRODUCE THE DISK FILE THEN PUNCH IT, BUT IS NOT TO PRODUCE A DEBUGGING LISTING OF THE CODE PRODUCED.

IF NO COMMON VALUE IS PROVIDED BY THE USER, THE DISK FILE CONTAINING THE CODE WILL BE PRODUCED. THE CONTENTS OF THE FILE WILL BE PUNCHED ON A CARD DECK. AND, WHILE THE OBJECT CODE IS BEING CREATED, A LISTING OF THE EXECUTIVE WILL BE PRINTED.

THE CARD DECK PRODUCED IS PUNCHED IN A SUBSET OF BCL, WITH EACH CHARACTER CONTAINING THREE BITS OF INFORMATION. EACH CARD CONTAINS 27 BYTES IF INFORMATION, THE LOAD ADDRESS FOR THE DATA AND INFORMATION FOR DETECTION OF PUNCH OR READER ERRORS.

B.4.4 BRINGING UP AND RUNNING THE SYSTEM

TO BRING UP THE RJE SYSTEM ONE NEEDS ONLY AN MCP COMPILED WITH THE OPTIONS DETAILED ABOVE, THE MODIFIED "REMOTE/USERS" FILE THAT SPECIFIES THE RJE LINES, AND REGULAR BATCH SYSTEM INTRINSICS. THE B-249 AND B487 UNITS SHOULD BE IN REMOTE AND A HALT/LOAD DONE WITH THE RJE MCP.

AT HALT/LOAD TIME THE MCP WILL ATTEMPT TO COMMUNICATE WITH ALL THE LINES SPECIFIED IN "REMOTE/USERS". ANY ACTIVE STATIONS WILL BE CLEARED (I. E. HALT LOADED THEMSELVES) AND ASKED TO LOG ON. LATER IF AN INACTIVE STATION DIALS UP, IT ALSO WILL BE ASKED TO LOG IN. AFTER A

SUCCESSFUL LOG ON SEQUENCE, THE REMOTE TERMINAL WILL BE ABLE TO READ IN DECKS, START JOBS, DO KEY IN MESSAGES, AND PRINT BACK UPS AS IF HE WERE AT THE CENTRAL SITE. MEANWHILE, THE CENTRAL SITE MAY RUN JOBS JUST AS IF THERE WERE NO RJE ACTIVITY GOING ON.

AS THE REMOTE TERMINALS READ IN CARDS, DECKS WILL BE CREATED ON DISK. IT WILL BE UP TO THE OPERATOR AT THE MAIN SITE TO TAKE CARE OF SUCH SYSTEM PARAMETERS AS PSEUDO-READERS,FACTORS,SCHEDULES,ETC. NONE OF THE REMOTE OPERATORS WILL BE PERMITTED TO HANDLE THESE THINGS. IN ALL INSTANCES THE CENTRAL B5700 SITE WILL HAVE FINAL CONTROL OVER THE SYSTEM, JUST AS IS TRUE WITHOUT RJE.

AS PRINTER FILES ARE OPENED BY JOBS INITIATED BY RJE TERMINALS, BACK UP DISK FILES WILL BE CREATED AND MARKED WITH THE RJE LINE FROM WHICH THEY WERE CREATED. WHEN THEY ARE CLOSED, THEY WILL BE PRINTED AT THE RJE PRINTER IF IT IS AVAILABLE. THEY MAY ALSO BE PRINTED AT THE CENTRAL SITE VIA PB OR FORMS. NOTE: THE ABOVE PRINTER ACTION IS INDEPENDENT OF THE CENTRAL SITE SETTING OF PBDONLY AND AUTOPRNT.

B.5 OPERATION OF THE DC1000

B.5.1 BOOTSTRAP FUNCTION

THE BOOTSTRAP FUNCTION IS USED FOR LOADING A SHORT BINARY PAPER TAPE INTO HIGH CORE. THE INFORMATION LOADED IS USUALLY A LOADER OR A DUMP ROUTINE. READING OF THE TAPE IS ACCOMPLISHED BY THE FOLLOWING ACTIONS:

1. TURN THE TWX TO THE ON LINE POSITION.
2. PLACE THE TAPE IN THE TWX READER WITH THE FIRST ROW OF DATA OVER THE READ STATION OF THE READER.
3. PLACE THE CONTROL SWITCH ON THE READER IN THE START POSITION.
4. PRESS THE RESET SWITCH ON THE CONSOLE.
5. PRESS THE BOOTSTRAP BUTTON ON THE CONSOLE.

WHEN THE CONTENTS OF THE TAPE HAS BEEN LOADED, THE PROGRAM CONTAINED ON THE TAPE WILL BEGIN EXECUTION.

B.5.2 LOADING OF CARD DECKS

VIRTUALLY ALL DC1000 PROGRAMS ARE PUNCHED ON CARDS. THESE PROGRAMS CAN BE EXECUTED BY LOADING THEM WITH THE LOADER. TO LOAD A CARD DECK, PERFORM THE FOLLOWING STEPS:

1. PLACE THE PROGRAM DECK FACE DOWN IN THE CARD READER WITH THE TWELVE ROW EDGE OF THE CARD TO THE FRONT OF THE CARD READER.

2. PLACE THE CARD READER ON LINE BY PLACING THE FEED SWITCH IN THE ON POSITION.

3. PERFORM THE BOOTSTRAP FUNCTION USING THE BCL CARD LOADER.

NOTE THAT ONE OR MORE BLANK CARDS ARE REQUIRED ON THE FRONT OF EACH PROGRAM DECK. THE BLANK CARDS ARE USED TO INITIALIZE MEMORY TO INSURE THAT THE BRANCH TO THE PROGRAM AT THE END OF LOAD WILL BE PERFORMED PROPERLY. EACH DATA CARD HAS PUNCHED IN IT THE LOAD ADDRESS FOR THE DATA IT CONTAINS. THEREFORE, THE ORDER OF DATA CARDS WILL NOT AFFECT THE LOADING PROCESS.

EACH DATA CARD ALSO CONTAINS INFORMATION ON IT FOR ERROR DETECTION. SHOULD A CARD BE MISSREAD, OR A PUNCH ERROR BE DETECTED, THE LOADER WILL STOP. THE TOP CARD IN THE OUTPUT STACKER WILL BE THE CARD WITH THE ERROR. TO TRY AGAIN TO READ THE CARD IN ERROR, REPLACE THE CARD IN THE INPUT HOPPER AND DEPRESS THE RUN SWITCH.

AFTER THE BOOTSTRAP LOADER HAS FINISHED LOADING THE LAST CARD IN THE DECK, A NORMAL STATE BRANCH TO THE INITIALIZE CODE FOR THE PROGRAM WILL BE PERFORMED.

B.5.3 SPO OPERATION

THE SPO FOR THE DC1000 IS A MODIFIED MODEL 33 ASR TWX. SINCE THIS DEVICE DOES NOT HAVE CONTROL BUTTONS AS DOES THE SYSTEM SPO, A NUMBER OF

THE STANDARD TWX KEYS HAVE BEEN ASSIGNED THE FUNCTIONS PERFORMED BY THE CONTROL BUTTONS ON THE STANDARD SPO. THE TWX CONTROL KEYS AND THEIR FUNCTIONS ARE:

1. CARRIAGE RETURN SIGNIFIES END OF A KEYBOARD INPUT.
2. RUBOUT IS USED TO DELETE THE KEYIN CURRENTLY IN PROGRESS.
3. BREAK IS USED AS AN INPUT REQUEST WHEN THE SPO IS PRINTING.
4. THE CHARACTER "<" IS THE BACKSPACE.

THE 33 ASR USED ON THE DC1000 IS A FULL DUPLEX DEVICE. THEREFORE, THE EXECUTIVE MUST PRINT EVERY CHARACTER ENTERED ON THE KEYBOARD. AS A RESULT OF THIS FACT, WHEN CHARACTERS ARE ENTERED TOO RAPIDLY IN SUCESSION, INPUT FROM THE KEYBOARD MAY BE IMPROPERLY INTERPRETED. EXPERIENCE WILL SHOW THE USER HOW FAST A DATA ENTRY RATE IS ACCEPTABLE TO THE SPO.

IT IS POSSIBLE TO HAVE INSUFFICIENT MEMORY AVAILABLE TO STORE A KEYBOARD INPUT. WHEN THIS OCCURS, THE KEYIN WILL NOT BE STORED, AND AN ERROR MESSAGE WILL BE PRINTED ON THE SPO.

B.5.4 CARD READER OPERATION

THE CARD READER USED WITH THE DC1000 IS A BURROUGHS MODEL A598 READER WITH A RATED CAPACITY OF 200 CARDS/MINUTE.

THERE IS ONLY ONE CONTROL SWITCH ON THE CARD READER, THE FEED SWITCH. WITH THE FEED SWITCH IN THE ON POSITION, THE FEED CIRCUITS IN THE READER ARE ENABLED, AND THE READER APPEARS TO BE ONLINE TO THE DC1000 EXECUTIVE. WHEN THE FEED SWITCH IS OFF, FEEDING OF CARDS IS NOT POSSIBLE, AND THE SOFTWARE SEES THE READER AS BEING NOT READY.

THE READING OF A DECK IS INITIATED BY PERFORMING THE FOLLOWING FUNCTIONS:

1. PLACE THE FEED SWITCH IN THE OFF POSITION.
2. PUT THE DECK IN THE INPUT HOPPER, FACE DOWN, WITH THE TWELVE

EDGE OF THE DECK TO THE FRONT OF THE READER.

3. SWITCH THE FEED SWITCH ON.

THE DC1000 WILL READ AND TRANSMIT CARDS AS SPACE AND LINE TIME IS AVAILABLE.

A NUMBER OF ERRORS CAN OCCUR WHEN CARDS ARE BEING READ. WHEN ONE OF THESE ERRORS OCCURS, AN ERROR MESSAGE IS PRINTED, AND THE CARD READER IS MADE NOT READY FROM A SOFTWARE VIEW POINT. THE ERROR MESSAGES POSSIBLE AND THEIR CAUSES ARE:

1. NOT READY - ATTEMPTED TO READ A CARD AND NO INPUT CARDS WERE AVAILABLE.

2. READ CHECK - CARD READER ERROR DETECTED DURING READING OF THE LAST CARD.

3. INVALID CHARACTER - AN INVALID CHARACTER WAS DETECTED IN THE LAST CARD READ, IN A COLUMN OTHER THAN THE FIRST COLUMN.

IN THE EVENT THAT A READER NOT READY MESSAGE IS PRINTED AND THE HOPPER IS NOT EMPTY, FOLLOW THE FOLLOWING PROCEDURE:

1. SET THE READER OFF LINE.

2. REMOVE THE DECK AND EXAMINE FOR DAMAGED CARDS OR THE POSSIBILITY THAT THE DECK WAS NOT WELL JOGGLED.

3. CORRECT THE DECK AND REPLACE IT IN THE INPUT HOPPER.

4. SET THE READER ON LINE AND READING OF CARDS WILL BE RESUMED.

HOWEVER, IF THE NOT READY CONDITION IS THE RESULT OF AN EMPTY HOPPER, SET THE READER OFF LINE, ADD MORE CARDS TO THE INPUT HOPPER, THEN PLACE THE READER ON LINE.

ONE WORD OF WARNING: THE PILE OF CARDS IN THE OUTPUT HOPPER SHOULD BE KEPT LESS THAN TWO OR THREE INCHS HIGH. IF THIS DEPTH IS EXCEEDED, SHUFFLING OF CARDS CAN OCCUR AS A RESULT OF NORMAL READER OPERATIONS.

A READ CHECK IS CORRECTED BY THE FOLLOWING STEPS:

1. SET THE READER OFFLINE.

2. REMOVE THE DECK FROM THE INPUT HOPPER.

3. PLACE THE TOP CARD IN THE OUTPUT HOPPER ON THE FRONT OF THE CARD DECK.

4. REPLACE THE DECK IN THE INPUT HOPPER.

5. SET THE READER ONLINE.

REPEATED READ CHECKS WOULD INDICATE A HARDWARE MALFUNCTION OR, MOST PROBABLY, A DEFECTIVE DECK.

THE ERROR RECOVERY ACTION FOR AN INVALID CHARACTER CONSISTS OF REMOVING THE TOP CARD FROM THE OUTPUT HOPPER, RE-PUNCHING IT TO CORRECT THE ERROR, THEN PROCEEDING AS THOUGH A READ CHECK ERROR HAD OCCURED. NOTE THAT THE 8-2 PUNCH IS NOT CONSIDERED TO BE AN INVALID CHARACTER, BUT IS INTERPRETED AS A QUESTIONMARK.

B.5.5 LINE PRINTER OPERATION

THE LINE PRINTER SPECIFIED FOR USE ON THE DC1000 IS THE B-9245 SERIES OF PRINTERS. THIS PRINTER PRINTS THE BCL CHARACTER SET IN LINE LENGTHS OF 80, 120 OR 132 CHARACTERS PER LINE AT 300 OR 400 LINES PER MINUTE.

THE CONTROLS ON THE PRINTER CONSIST OF A SWITCH TO SELECT A SIX OR EIGHT LINE/INCH SPACING, AND A READY/NOT READY SWITCH.

TWO ERROR MESSAGES PERTAINING TO THE LINE PRINTER CAN BE ENCOUNTERED. THEY ARE:

1. NOT READY - THIS ERROR OCCURS WHENEVER THE EXECUTIVE ATTEMPTS TO WRITE TO THE PRINTER AND IT IS NOT READY TO PERFORM AN I/O. THIS COULD BE CAUSED BY LOW PAPER SUPPLY, AN OPEN PRINTER CARRIAGE, A SKIP TO AN INVALID CHANNEL OR SIMPLY THE PRINTER BEING OFFLINE.

2. PRINT CHECK - WHEN A PRINTER ERROR IS DETECTED, THIS MESSAGE IS PRINTED AND THE PRINTER IS MARKED NOT READY.

TO RESUME OUTPUT, CORRECT THE CONDITION WHICH CAUSED THE NOT READY CONDITION, THEN MAKE THE PRINTER READY.

IN THE EVENT OF A PRINT CHECK, OUTPUT WILL CEASE. TO RESUME OUTPUT, MAKE THE PRINTER NOT READY THEN MAKE IT READY.

B.5.6 DATA COMMUNICATIONS LINK TO THE SYSTEM

VERY LITTLE OPERATOR ACTION IS NEEDED TO INSURE PROPER OPERATION OF DATA TRANSFER WITH THE SYSTEM. THE ONLY RESPONSIBILITY THAT THE OPERATOR OF THE DC1000 HAS IS TO ESTABLISH A COMMUNICATIONS LINK WITH THE SYSTEM. ANY PARITY ERRORS, VERTICAL OR HORIZONTAL, TIMEOUTS, ETC. ARE HANDLED BY THE EXECUTIVE OR THE MCP. HOWEVER, WHEN CERTAIN MORE OR LESS FATAL ERRORS OCCUR, AN ERROR MESSAGE WILL BE PRINTED. THE POSSIBLE ERROR MESSAGES AND THEIR CAUSES ARE:

1. NO SYSTEM LINK - DATACOM LINK TO THE B5700 IS NOT PRESENT.

2. CONNECTION MADE - DATA COMMUNICATIONS LINK TO THE SYSTEM WAS JUST ESTABLISHED.

3. NO RESPONSE - FOUR ATTEMPTS WERE MADE TO CONTACT THE SYSTEM AND NO RESPONSE WAS RECEIVED TO ANY OF THEM.

4. TRANSMIT ABORT - DATA SET WENT NOT READY WHILE THE DC1000 WAS TRANSMITTING TO THE SYSTEM.

5. MISSING MESSAGE - TRANSMISSION NUMBER ERROR.

ERROR RECOVERY IS POSSIBLE, BY AND LARGE, ONLY FOR ERROR ONE. CONDITION ONE CAN BE CORRECTED BY CALLING THE SYSTEM AGAIN, OR BY CHECKING TO SEE THAT ALL CONNECTIONS BETWEEN THE DC1000 AND THE DATASET ARE PROPERLY MADE.

MESSAGE TWO INDICATES THAT THE LINE IS AVAILABLE FOR SERVICE AGAIN.

MESSAGE THREE INDICATES THAT THE SYSTEM-S DATACOM HARDWARE IS NOT FUNCTIONING OR THAT THE SYSTEM IS IN THE PROCESS OF PERFORMING A HALT LOAD.

MESSAGE FOUR INDICATES AN EXECUTIVE SOFTWARE, OR A DATACOM HARDWARE FAILURE. THE EXECUTIVE WILL ATTEMPT TO RECOVER FROM THE CONDITION WHICH CAUSED THIS ERROR.

MESSAGE FIVE INDICATES THAT THE EXPECTED TRANSMISSION NUMBER WAS NOT FOUND IN THE HEADER OF A MESSAGE. THIS WOULD INDICATE THE POSSIBILITY OF A MISSING MESSAGE. USUALLY, THIS MESSAGE DOES NOT HAVE ANY DRASTIC SIGNIFICANCE. HOWEVER, THE OPERATOR SHOULD BE AWARE OF THE POSSIBILITY OF MISSING DATA.

WHEN THE DC1000 DATACOM INTERFACE IS THROUGH THE SWITCHED TELEPHONE NETWORK, THE EXECUTIVE IS CAPABLE OF ANSWERING THE TELEPHONE WITHOUT OPERATOR INTERVENTION. AFTER THE TELEPHONE HAS BEEN ANSWERED, THE DC1000 WILL ESTABLISH COMMUNICATIONS WITH THE MCP.

B.6 USE OF REMOTE TERMINAL

B.6.1 USER INFORMATION

WHEN AN RJE TERMINAL LOGS IN A USERCODE WILL BE ENTERED AND WILL BECOME THE DEFAULT USER FOR ALL JOBS RUN FROM THAT TERMINAL. WHEN CONTROLCARDS ARE ENTERED (EITHER FROM THE RJE SPO OR CARD READER) THE MCP WILL SCAN FOR A ?USER=<USERID> CARD AND WILL USE THAT AS THE USER. OTHERWISE, IF IT DOESNT FIND A USER CARD, THE USERID THAT LOGGED IN THAT STATION WILL BE ASSIGNED TO THAT JOB.

B.6.2 USE OF CARD READER

DECKS MAY BE ENTERED THROUGH THE CARD READER ANY TIME AFTER THE LOG IN. ANY DECKS ENTERED BEFORE THE COMPLETION OF THE LOG IN WILL BE DISCARDED BY THE MCP. ANY DECK THAT CAN BE ENTERED IN THE CARD READER AT THE CENTRAL SITE MAY BE READ THROUGH THE RJE CARD READER, WITH A FEW VERY SMALL RESTRICTIONS. THESE RESTRICTIONS EXIST BECAUSE THE CARD IMAGES ARE BEING CONVERTED TO DECKS ON DISK AS IF LOAD CONTROL WERE

RUNNING AT THE RJE TERMINAL. THE RESTRICTIONS TO THE CARD DECKS, THE SAME AS THE RESTRICTIONS FOR LOAD CONTROL IN THE CENTRAL SITE CARD READERS, ARE:

1. A LIMIT OF 12000 CARDS PER DECK.

   IF MORE THAN 12000 CARDS ARE READ IN, THE MCP WILL PRINT OUT A MESSAGE ON THE RJE SPO AND DISCARD THE DECK.

2. A SEPARATE ?END CARD.

3. COMPLETION OF DECK BEFORE ITS USE.

   IF CARDS ARE READ IN WITHOUT A ?END CARD TERMINATING THEM, THE RJE SPO WILL PRINT A CARD READER NOT READY MESSAGE.

B.6.3 USE OF PRINTER

REGARDLESS OF FILE DECLARATION, LABEL EQUATION, OR THE CONDITION OF THE OPTION PBDONLY, ALL PRINTER FILES WILL BE ROUTED TO BACK UP DISK (AS IF PBDONLY WERE SET FOR THE RJE TERMINAL).

THE PRINTER FILES WILL BE PRINTED WHEN THEY ARE CLOSED, IF THE PRINTER IS NOT ALREADY IN USE. THE APPEARANCE TO THE RJE TERMINAL IS THAT THE AUTOPRNT OPTION IS SET FOR IT.

THE REMOTE OPERATOR MAY, THROUGH HIS SPO, "DS", "QT" OR "ST" ANY PRINTER BACK UP PRINTING ON HIS PRINTER.

THE REMOTE OPERATOR MAY PRINT ANY PRINTER BACK UP TAPE OR DISK FILE, THROUGH THE USE OF THE "PB" COMMAND, PROVIDED HIS PRINTER IS NOT IN USE. ERROR MESSAGES WILL RESULT IF THE UNIT OR DISK FILE DOES NOT EXIST OR IS NOT A PRINTER BACK UP.

IF FORMS ARE REQUESTED WITH A PRINTER FILE, A MESSAGE IS PRINTED AT THE RJE SPO, AS WELL AS THE MAIN SPO, INDICATING THAT FORMS ARE DESIRED. THIS CAN BE USED FOR TWO PURPOSES WITH THE RJE TERMINAL:

1. PLACEMENT OF SPECIAL FORMS IN PRINTER.

   WHEN THIS IS DESIRED THE REMOTE OPERATOR WILL PLACE THE SPECIAL

FORMS IN THE PRINTER OR MAKE SURE THAT THEY ARE ALREADY THERE. THEN HE ENTERS <MIX-ID>OK ON HIS SPO AND THE PRINTER BACK UP WILL COMMENCE.

2. PRINTING OF LONG FILES AT MAIN SITE.

THIS MAY PROVE TO BE A USEFUL FEATURE FOR LONG PRINTER BACK UPS AS THE PRINTER COULD BE TIED UP FOR A LONG TIME IF ALLOWED TO PRINT ON THE RJE PRINTER. TO REROUTE IT TO THE CENTRAL SITE, A "FM" COMMAND WILL BE USED. HOWEVER THE RJE OPERATOR MAY NOT DO THIS. HE MAY REQUEST THE MAIN SITE TO DO THE "FM". IF THE MAIN SITE OPERATOR DOES A <MIX-ID>FM<UNIT> KEY IN ON THE MAIN SPO TO AN AN AVAILABLE PRINTER, THE PRINTER BACK UP WILL BE REROUTED TO THAT PRINTER AND THE RJE PRINTER WILL BECOME AVAILABLE. NOTE: WHEN THIS OCCURS THE RJE SITE CAN NO LONGER ACCESS THIS JOB.

B.6.4 USE OF RJE SPO CONSOLE

OUTPUT MESSAGES

1. LOCAL MESSAGES

THESE ARE THE MESSAGES GENERATED BY THE REMOTE SATELLITE HAVING TO DO WITH ITS OWN CONTROL, SUCH AS CARD READER HOPPER EMPTY, INVALID CHARACTER IN A CARD, PRINTER OUT OF PAPER, ETC. THE FORMAT OF THESE MESSAGES WILL VARY DEPENDING UPON THE TYPE OF MESSAGE GENERATED.

2. SYSTEM STATUS MESSAGE

```
THE FORMAT OF THIS MESSAGE IS:
<PRIORITY>:<PROGRAM NAME>:<MIXID>:<STATUS>
WHERE
<PRIORITY>=A NUMBER WITH A RANGE OF 0-1023
<MIXID>   = MIX INDEX REFERENCE NUMBER OF JOB
<STATUS> =
            BOJ        (BEGINNING OF JOB)
            EOJ        (END OF JOB)
            ST-ED      (PROGRAM SUSPENDED)
            DS-ED      (PROGRAM TERMINATED)
            SCHD       (PROGRAM SCHEDULED)
```

3. LOG-ON AND LOG-OFF MESSAGES

IT IS ASSUMED THAT THE INITIALIZATION SOFTWARE IN THE RJE SATELLITE WILL TRANSMIT A "HANDSHAKE" CONTROL MESSAGE TO THE CENTRAL SITE, UPON DIAL UP OR HALT LOAD. IF THE CENTRAL SYSTEM OR DATACOMM LINK IS NOT LIVE THEN THE LACK OF RESPONSE FROM THE CENTRAL SYSTEM INDICATES THERE IS NO CONNECTION TO THE CENTRAL SYSTEM. HOWEVER, IF A CONNECTION DOES EXIST THE MCP WILL RESPOND WITH THE FOLLOWING MESSAGE FOR THE REMOTE SPO:

```
B5700 R. J. E. SYSTEM -- ENTER USERCODE
```

THE REMOTE OPERATOR THEN MUST ENTER THE USER-ID ASSIGNED BY THE CENTRAL SYSTEM. IF THE USERID IS NOT RECOGNIZED BY THE MCP, IT WILL RESPOND WITH:

```
INCORRECT USERCODE -- RE-ENTER USERCODE
```

THREE ATTEMPTS ARE ALLOWED. IF THE USERCODE IS CORRECT THE MCP WILL LOOK IN THE "REMOTE/USERS" FILE TO SEE IF A PASSWORD IS REQUIRED FOR THAT USERCODE. IF SO, IT WILL RESPOND:

```
ENTER PASSWORD PLEASE
```

THE REMOTE OPERATOR THEN MUST ENTER HIS PASSWORD. IF THE PASSWORD IS INCORRECT THE MCP WILL RESPOND WITH:

```
INVALID PASSWORD -- RE-ENTER PASSWORD
```

THREE ATTEMPTS ARE ALLOWED. IF THE PASSWORD IS RECOGNIZED BY THE MCP OR IT WAS NOT REQUIRED THE MCP WILL RESPOND:

```
<STATION-NAME> LOGGED IN AT <TIME>
```

IF ON THE THIRD ATTEMPT EITHER THE USERCODE OR THE PASSWORD IS STILL INCORRECT THE MCP WILL RESPOND:

```
INCORRECT LOG ON SEQUENCE
```

THEN THE MCP WILL IGNORE ALL MESSAGES FROM THAT RJE TERMINAL UNTIL THE NEXT "CONTROL HALT/LOAD" MESSAGE. UNTIL SUCH TIME THAT THE MCP HAS LOGGED THE STATION ON IT WILL THROW AWAY ANY MESSAGES THAT ARE NOT DESIGNATED AS COMING FROM THE REMOTE SPO. WHEN THE SPO DOES LOG ON IT WILL BE PERMITTED TO READ IN DECKS, PRINT BACK UPS, ETC. IF THERE ARE ANY BACK UP FILES ON DISK

PREVIOUSLY CREATED BY THIS RJE TERMINAL, THEY WILL THEN BE PRINTED.

TO TERMINATE THE CONNECTION BETWEEN THE REMOTE SATELLITE AND THE MAIN SYSTEM THE REMOTE OPERATOR LOGS OFF WITH:

```
LO
```

THIS MESSAGE WILL BE IGNORED IF THE CENTRAL SITE IS STILL PROCESSING ANY PROGRAMS OR DATA INTRODUCED FROM THE RJE TERMINAL. IF IT IS ESSENTIALLY IDLE THE MCP WILL RESPOND:

```
<STATION-NAME> LOGGED OFF AT <TIME>
```

IF THE STATION DISCONNECTS OR APPEARS TO DISCONNECT FROM THE MCP, THEN THE JOBS THAT IT INITIATED WILL BE DS-ED AND THE MCP WILL RESPOND WITH:

```
<STATION-NAME> DISCONNECT AT <TIME>.
```

4. ERROR MESSAGES

THE MCP IN ACCEPTING INPUT FROM THE REMOTE SATELLITE MAY ENCOUNTER CERTAIN ERROR SITUATIONS. IT WILL PRINT AN ERROR MESSAGE ON THE RJE SPO FOR ANY OF THE FOLLOWING:

```
CONTROL CARD ERROR
INVALID KEYBOARD
SECURITY MAINTENANCE ERROR
NON EXECUTABLE CODE ERROR
```

5. RESPONSE MESSAGES

THESE MESSAGES TYPES ARE SENT TO THE REMOTE SPO IN RESPONSE TO KEY IN INFORMATION REQUESTS FROM IT.

6. PROGRAM WRITES TO SPO

A PROGRAM MAY DECLARE A SPO FILE (TYPE 11) IN AN OBJECT PROGRAM AND THE OUTPUT WILL BE ROUTED TO THE REMOTE SPO. IF A READ IS DONE ON THIS FILE THE MCP WILL SEND:

```
<MIXID>:<PROGRAM NAME>: ACCEPT
```

TYPE MESSAGE TO REQUEST AN INPUT.

INPUT MESSAGES.

A SUBSET OF KEY IN INPUT MESSAGES HAS BEEN DEFINED AS ALLOWABLE FOR THE RJE TERMINALS. THE PURPOSE IS TO ALLOW THE RJE TERMINALS TO ASK QUESTIONS OF THE SYSTEM, BUT NOT DIRECTLY AFFECT IT. THESE MESSAGES ARE DIVIDED INTO TWO GROUPS - THE INFORMATION MESSAGES AND THE MIX MESSAGES. THE MIX MESSAGES MAY ONLY BE USED WITH MIXES THAT WERE ORIGINATED BY THAT RJE TERMINAL. ANY ATTEMPT TO ACCESS OTHER MIXES WILL RESULT IN AN INVALID KEYBOARD. THE INFORMATION MESSAGES ALLOWED ARE:

MX	PRINT THE MIX (JOBS RUNNING)
PD,EX,LF,LC,LS	PRINT DIRECTORY INFORMAION
PB	PRINT BACK UPS
WD	PRINT THE DATE
WT	PRINT THE TIME
TO	TYPE OUT THE OPTION LIST
CD	PRINT THE DECKS ON DISK
TF	TYPE MULTIPROCESSING FACTOR
TS	TYPE SCHEDULE (JOBS IN SHEET)
SS	STATION TO STATION MESSAGE
WM	PRINT WHICH MCP IS RUNNING
WI	PRINT WHICH INTRINSICS ARE BEING USED
CU	TYPE TOTAL AMOUNT OF CORE IN USE
AU	TYPE TOTAL AMOUNT OF AUXILIARY CORE IN USE
CC	ENTER CONTROLCARDS THROUGH RJE SPO

THE MIX MESSAGES THAT ARE ALLOWED ARE:

DS	TERMINATE A JOB
OK	OK A JOB WAITING ON SOMETHING
AX	ENTER INPUT TO A SPO FILE
TI	REQUEST RUNNING AND I-O TIMES FOR THE JOB
RM	REMOVE A DUPLICATE FILE
ST	SUSPEND A JOB
IN	CHANGE THE PRT OF A JOB
OT	READ THE PRT OF A JOB
QT	QUIT A LIBMAIN OR PRINT BACKUP
CT,XT,TL	PRINT OR CHANGE JOB TIME LIMITS
CU	TYPE AMOUNT OF CORE IN USE BY A JOB
AU	TYPE AMOUNT OF AUXILIARY CORE IN USE BY JOB
ES	REMOVE A JOB FROM THE SCHEDULE

B.7 IMPLEMENTATION OF THE RJE SYSTEM

B.7.1 INTRODUCTION TO IMPLEMENTATION

THIS NEXT SECTION IS A DISCUSSION OF THE IMPLEMENTATION OF THE RJE SYSTEM MAINLY FROM THE POINT OF VIEW OF THE MCP. IT IS NOT NECESSARY TO KNOW ANY OF THE INFORMATION IN THIS SECTION TO RUN THE RJE SYSTEM. IT IS PRIMARILY HERE TO ALLOW SYSTEM PROGRAMMERS TO ADD OR MODIFY THE THE MCP FOR THEIR OWN PURPOSES.

B.7.2 LINE DISCIPLINE

THE LINE DISCIPLINE USED WITH THE REMOTE JOB ENTRY SYSTEM IS THE BURROUGHS CONVERSAIONAL POINT TO POINT STANDARD MODIFIED TO ELIMINATE ERROR RECOVERY. IT IS DRAWN BELOW.

```
SIDE ONE                 :    SIDE TWO
1                        :
:                        :
ENQ                      :
:                        :
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
                         :    :    :
2                        :    ACK  NO RESPONSE
:                        :    :    :
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<    1
:                        :
MESSAGE                  :
:                        :
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
                         :    :    :         :
                         :    ACK  MESSAGE   NAK
                         :    :    :         :
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<    :         2
```

```
:    :                   :         :
:    :<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
:    :              :    :
EOT  NEXT MESSAGE   ACK  :
:    :              :    :
:    >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>ETC.
IDLE STATE
```

B.7.3 NOTES ON THE LINE DISCIPLINE

1. GENERAL - THIS TABLE DEFINES CONVERSATIONAL DATA LINK CONTROL PROCEDURES. CONVERSATIONAL PROCEDURES ALLOW REVERSALS OF DIRECTION OF MESSAGE TRANSFER WITHOUT SEPARATE ESTABLISHMENT AND TERMINATION PROCEDURES; THEY ARE PRIMARILY INTENDED FOR COMMUNICATION BETWEEN AN OPERATOR CONTROLLED TERMINAL AND A COMPUTER. THIS TABLE APPLIES TO POINT-TO-POINT LINES. THE STATION TRANSMITTING IS CONSIDERED THE CONTROL STATION.

2. IN THIS PROCEDURE ALL MESSAGES CONSIST OF A SINGLE TRANSMISSION BLOCK ENDING IN ETX. THE USE OF ETB IS NOT ALLOWED, NOR ARE THE REVERSE INTERRUPT, CANCEL, AND PLAYBACK FUNCTIONS. BLOCKS BEGIN WITH SOH. BCC IS REQUIRED.

3. IN CONVERSATIONAL PROCEDURES, THE NORMAL RESPONSE TO A GOOD MESSAGE IS ANOTHER DATA MESSAGE. THIS RESPONSE CAN ONLY BE MADE IF THE MESSAGE JUST RECEIVED ENDED IN ETX AND WAS RECEIVED WITHOUT ERROR. THE BEGINNING CONTROL CHARACTER OF THE RESPONSE, (SOH), CAN BE CONSIDERED TO IMPLY POSITIVE ACKNOWLEDGEMENT OF THE RECEIVED MESSAGE, AND TO IMPLY THE TRANSITION OF MASTER/ SLAVE RELATIONSHIP.

4. THE ESTABLISHMENT PROCEDURE TO "INITIATE THE CONVERSATION" WHEN THE LINE IS IN THE IDLE STATE INVOLVES THE CONTROL STATION SENDING AN ENQ CHARACTER DOWN THE LINE. THE OTHER STATION WILL RESPOND WITH AN ACK CHARACTER AND THE LINE IS ESTABLISHED. THEN THE CONTROL STATION WILL TRANSMIT HIS FIRST MESSAGE AND THE CONVERSATION START. IF THERE IS NO RESPONSE FROM THE OTHER STATION, THE CONTROL STATION WILL RETRY SENDING THE ENQ N TIMES (WHERE N IS PRESENTLY 4). IF THERE IS STILL NO RESPONSE THE CONTROL STATION WILL EXIT TO ERROR RECOVERY CODE.

5. IF STATION B RECEIVES A RECOGNIZABLE MESSAGE, BUT IT CONTAINS CHARACTER PARITY ERROR(S), BLOCK CHECK ERROR, OR OTHER

DETECTABLE ERROR(S) STATION B RESPONDS WITH A NAK CHARACTER. THE CONTROL STATION RETRANSMITS THE MESSAGE. THIS PROCEDURE MAY BE DONE N TIMES AT WHICH POINT THE MCP END WILL EXIT TO ERROR RECOVERY.

6. IF STATION B RECEIVES A GOOD MESSAGE, BUT HAS NO TRAFFIC TO SEND IT RESPONDS WITH AN ACK CHARACTER. IF THE CONTROL STATION HAS MORE TRAFFIC FOR STATION B IT SENDS ANOTHER MESSAGE. IF THE CONTROL STATION HAS NO MORE TRAFFIC FOR STATION B, THE CONTROL STATION PROCEEDS TO NORMAL TERMINATION. THIS INVOLVES THE CONTROL STATION SENDING AN EOT CHARACTER. AT THIS POINT THE LINE IS CONSIDERED TO BE IN THE IDLE STATE. IF EITHER STATION WISHES TO SEND ANOTHER MESSAGE THEY MUST GO THROUGH THE ESTABLISHMENT PROCEDURE.

7. IF THE CONTROL STATION RECEIVES A RECOGNIZABLE MESSAGE WITH ONE OF THE PARITY ERRORS IT WILL RESPOND WITH A NAK CHARACTER AS IN 5. ABOVE. IT WILL RETRY N TIMES AT WHICH POINT THE MCP WILL EXIT TO ERROR RECOVERY CODE.

8. IF THE CONTROL STATION RECEIVES A GOOD MESSAGE BUT HAS NO MORE TRAFFIC TO SEND TO STATION B, IT WILL RESPOND WITH THE ACK CHARACTER. STATION B CAN THEN SEND ANOTHER MESSAGE OR EXIT TO NORMAL TERMINATION.

9. THE MCP ERROR RECOVERY WILL BE TO TREAT THE LINE AS IF THE LINE WERE DISCONNECTED. IT WILL DS ANY JOBS RUNNING FROM THE RJE TERMINAL, THROW AWAY ANY MESSAGES WAITING TO BE SENT OUT, AND PRINT A MESSAGE INDICATING THAT THE LINE DISCONNECTED ON THE MAIN SITE SPO.

10. THE RJE TERMINAL"S ERROR RECOVERY WILL BE TO SIT IDLE WAITING FOR THE MCP TO INTERRUPT IT. (THIS WILL MOST LIKELY BE A HALT/LOAD MESSAGE FROM THE MCP).

8.7.4 RJE MESSAGES

THE MESSAGE FORMAT FOR THE RJE SYSTEM IS AS FOLLOWS:

```
 S A A   S C :C C:       :C :C C:       : E B
 O D D T T R :C C: <TEXT>:R :C C: <TEXT>: T C
 H 1 2 N X 1 :1 2:       :N :1 2:       : X C
```

1. ALL MESSAGES START WITH THE START OF HEADER CARARCTER SOH. THE HEADER CONSISTS OF THE TWO ADDRESS DESIGNATE CHARACTERS AND THE TRANSMISSION NUMBER. THE BODY OF THE TEXT STARTS WITH THE START OF TEXT CHARACTER STX. EACH SUB TEXT OR BLOCK PART OF THE TEXT STARTS WITH THE CARRIAGE RETURN CHARACTER USED TO SEPARATE THE BLOCKS. EACH BLOCK OF TEXT CONSISTS OF THE OPTIONAL CARRIAGE CONTROL CHARACTERS WHICH WOULD BE USED ONLY FOR THE PRINTER AND THEN THE BODY OF THE TEXT. MORE THAN ONE BLOCK MAY BE USED IN ONE MESSAGE, BUT IS NOT REQUIRED. THE MESSAGE ENDS WITH THE END OF TEXT CHARACTER ETX AND THE BLOCK CHECK CHARACTER. THERE IS A LIMIT OF 400 CHARACTERS SET FOR EACH MESSAGE.

2. THE ADDRESS DESIGNATE CHARACTERS (AD1, AD2) RELATE A MESSAGE TO A DEVICE ON A REMOTE SATELLITE AS FOLLOWS:

AD1 AD2	
0 0	SYSTEM CONTROL
0 1	TO OR FROM REMOTE SPO
0 2	TO CARD PUNCH
0 3	TO LINE PRINTER FROM IS AN ERROR
0 4	TO PAPER TAPE PUNCH FROM PAPER TAPE READER
0 5	TO OR FROM MAGNETIC TAPE
0 6 THROUGH 09	(RESERVED FOR EXPANSION)
1 0 THROUGH 99	AD1 AND AD2 OF DATACOM LINES

3. THE TRANSMISSION NUMBER (TN) IS COUNTED UP ONE MODULO 10 (I. E. HAS A RANGE OF 0-9) WITH EACH TRANSMISSION. IT CAN BE USEFUL FOR DETECTING MISSING OR DUPLICATE TRANSMISSIONS.

4. THE OPTIONAL CARRIAGE CONTROL CHARACTERS (CC1, CC2) DELIVER CARRIAGE CONTROL TO THE REMOTE PRINTER AS FOLLOWS:

CC1	
0	SPACE CC2 LINES NO PRINT
1	SKIP TO CHANNEL CC2 NO PRINT
2	PRINT BEFORE SPACING CC2 LINES
3	PRINT BEFORE SKIPPING TO CHANNEL CC2
4	PRINT AFTER SPACING CC2 LINES
5	PRINT AFTER SKIPPING TO CHANNEL CC2

5. THE ADDRESS DESIGNATE CHARACTERS 00 (SYSTEM CONTROL IN 2 ABOVE) ARE USED FOR TRANSFERRING SPECIAL INFORMATION. THE TEXT PART OF THE MESSAGE CONTAINS THE INFORMATION AS TO WHICH CONTROL MESSAGE IT IS. EACH MESSAGE HAS A TWO DIGIT NUMBER WHICH DEFINES THE MESSAGE AND POSSIBLE INFORMATION AFTER THE TWO DIGITS. THE CONTROL MESSAGES PRESENTLY DEFINED ARE:

00. FROM THE MCP IMPLIES THAT THE MCP JUST HALT LOADED.

THIS MESSAGE ASKS THE RJE TERMINAL TO HALT LOAD ITSELF THAT IS TO RESET ALL OF ITS BUFFERS AND THEN SEND A HALT LOAD MESSAGE ITSELF.

00. FROM THE RJE TERMINAL MEANS THAT IT HALT LOADED.

THE HALT LOAD MESSAGE WILL INCLUDE WITH IT THE DEVICES THAT ARE CONNECTED TO THE RJE TERMINAL. FOR EXAMPLE THE HALT LOAD MESSAGE 0003 WOULD INDICATE THAT THE TERMINAL HAD A PRINTER (03). IF THE TERMINAL IS NOT LOGGED ON WHEN THE HALT LOAD MESSAGE IS SENT, THE MCP WILL RESPOND WITH:

```
B5700 R. J. E. SYSTEM -- ENTER USERCODE PLEASE
```

THEN IT WILL ENTER THE LOG ON SEQUENCE. IF THE MCP HAS THE LINE MARKED AS ALREADY LOGGED ON IT WILL SEND:

```
B5700 R. J. E. SYSTEM CONNECTION REESTABLISHED
```

IT WILL THEN RESUME WHERE IT LEFT OFF WHEN THE RJE TERMINAL HUNG AND REQUIRED THE HALT LOAD.

01. FROM THE RJE TERMINAL MEANS THAT A UNIT WENT NOT READY

THE MESSAGE WILL INCLUDE THE UNIT THAT WENT NOT READY. THUS IF THE SPO WENT NOT READY THE MESSAGE WOULD BE 0101. THE MCP WILL THEN STOP SENDING ANY MESSAGES TO THAT UNIT UNTIL IT GETS A UNIT READY MESSAGE.

02. FROM THE RJE TERMINAL MEANS THAT A UNIT WENT READY

THE MESSAGE WILL INCLUDE THE UNIT THAT WENT READY. THUS IF THE LINE PRINTER WENT READY THE MESSAGE WOULD BE 0203. THE MCP WILL RESUME SENDING MESSAGES TO THAT UNIT THEN.

B.7.5 STATION TABLE FORMAT

THE STATION WORD FOR THE RJE LINES HAS BEEN EXTENSIVELY MODIFIED. THE PREVIOUSLY UNUSED BIT [47:1] OF THE STATION WORD HAS BEEN TAKEN TO MEAN A RJE LINE. IF THIS BIT IS OFF THE LINE IS A REGULAR DATACOM LINE AND MAY BE USED IN ANY PRESENTLY LEGITIMATE WAY WITH THE REST OF THE STATION WORD HAVING THE SAME MEANING. IF THE BIT IS ON THIS IS A RJE LINE AND MAY NOT BE USED FOR ANY NORMAL DATACOM. THE STATION WORD FORMAT FOR THE RJE LINES IS:

FIELD	DESCRIPTION
1:1	OUTPUT IN PROCESS BY STA. MESSAGE WRITER
2:1	NOT USED AND SHOULD NOT BE USED
3:1	NOT USED
4:4	INPUT TRANSMISSION NUMBER (LAST)
8:1	NOT USED AND SHOULD NOT BE USED
9:4	TERMINAL UNIT OF RJE LINE
13:1	SHOULD BE 0
14:4	BUFFER NUMBER OF RJE LINE
18:4	OUTPUT TRANSMISSION NUMBER (LAST)
22:9	I-O INFORMATION SAME AS REGULAR
31:1	LAST MESSAGE NAKKED
32:1	MESSAGE IN OUTPUT QUEUE
33:1	RJEINPUT PROCEDURE RUNNING
34:1	PRINTER IN USE BY JOB
35:1	PRINTER NOT READY
36:1	PRINTER ATTACHED TO TERMINAL
37:1	PUNCH IN USE BY JOB
38:1	PUNCH NOT READY
39:1	PUNCH ATTACHED TO TERMINAL
40:1	SPO NOT READY
41:1	NOT USED
42:2	ERROR COUNT
44:1	LAST MESSAGE RECEIVED IS ACK
45:1	SELECTED
46:1	LOGGED ON
47:1	RJE LINE

B.7.6 MCP CONTROL INFORMATION

THE MCP KEEPS NINE BITS EQUIVALENT TO THE TERMINAL UNIT AND BUFFER NUMBER OF THE RJE LINE. THESE ARE USED TO MARK JOBS AND THEIR INPUT AND

OUTPUT AS ORIGINATING FROM THE RJE TERMINAL INSTEAD OF THE MAIN SITE OR ANOTHER RJE TERMINAL. THE THINGS THAT ARE MARKED ARE:

1. PBD/PUD FILES - HEADER[6].[39:9]

2. DECKS - HEADER[6].[9:9]

3. JOBS - JAR[MIX,6].[9:9]

4. SCHEDULE - SHEET[6].[9:9]

THE KEYIN MESSAGES THAT ARE ALLOWED FOR RJE SPOS TO USE WERE DESIGNED TO ALLOW THE RJE OPERATOR TO ASK INFORMATION OF THE SYSTEM, BUT NOT BE ABLE TO CHANGE ANY THING ABOUT THE SYSTEM EXCEPT FOR JOBS THAT WERE ORIGINATED FROM THAT RJE STATION. FURTHERMORE THE MCP WILL NOT ALLOW ACCESS TO ANY JOBS NOT ORIGINATED FROM THAT RJE TERMINAL. IT DOES THIS BY CHECKING THE JAR[MIX,6].[9:9] TO SEE IF IT IS EQUAL TO THE TERMINAL UNIT AND BUFFER NUMBER OF THE RJE STATION. IF THEY ARE EQUAL THE MCP WILL ALLOW THE KEYIN (IF IT IS ON THE RJE ACCEPTABLE KEYIN LIST). IF THEY ARE NOT EQUAL, THE MCP WILL SEND AN INVALID KEYBOARD MESSAGE.

THE LIST OF RJE ACCEPTABLE KEYIN MESSAGES IS SET UP BY MODIFYING THE LISTS OF KEYINS IN THE PROCEDURES INFOMESSAGES AND MIXMESSAGES. TO MAKE A KEYIN LEGITIMATE FOR RJE USE IN MIXMESSAGES, ONE WOULD ADD FOUR TO THE NUMERICAL PREFIX BEFORE THE KEYIN. FOR EXAMPLE, THE TABLE ENTRY FOR THE KEYIN DS LOOKS NOW LIKE - "2DS". TO MAKE THIS LEGIMATE FOR RJE (WHICH HAS BEEN DONE FOR THE RJE MCP PATCH), THE TABLE ENTRY WOULD NOW LOOK LIKE - "6DS". LIKEWISE "1OK" WOULD BECOME "5OK" AND "3ES" WOULD BECOME "7ES". TO CHANGE THE INFOMESSAGES INVOLVES PLACING A FOUR IN FRONT OF THE KEYIN IN THE STRING. THUS THE TABLE ENTRY FOR FINDING THE MIX IS "MX". TO ALLOW IT FOR RJE THIS WOULD BE CHANGED TO "4MX". "PD" WOULD BECOME "4PD". HOWEVER "CM" WOULD REMAIN THE SAME AND HENCE WOULD BE NOT A LEGITIMATE KEYIN FOR THE RJE LINES. THE USER MAY WANT TO CHANGE THE LIST OF LEGIMATE KEYINS GIVEN EARLIER IN THIS DOCUMENT AND HE MAY DO THIS BY COMPILING THE MCP WITH THE CHANGES AS EXPLAINED IN THIS PARAGRAPH.

B.7.7 NEW RJE PROCEDURES IN THE MCP

1. RJEINPUT - THIS IS THE PROCEDURE THAT PROCESSES ALL MESSAGE INPUT FROM THE RJE LINES. IT RUNS AS A INDEPENDENT RUNNER WITH AT MOST ONE COPY FOR EACH RJE LINE. (BIT [33:1] OF THE STATION WORD IS THE INTERLOCK FOR THE COPY OF RJEINPUT RUNNING FOR EACH

LINE). THIS PROCEDURE BREAKSUP THE MESSAGES, STRIPPING OFF THE INPUT UNIT AND TRANSMISSION NUMBER. IT IS RESPONSIBLE FOR SEPARATING "BLOCKED RECORDS" ON INPUT, CONVERTING THE CONTROL "@" TO A LEFT ARROW, CONVERTING THE "≠≠" TO A SINGLE "≠", AND EXPANDING OUT SUPPRESSED BLANKS. IT WILL IN GENERAL REMAIN RUNNING ONLY AS LONG AS THERE IS INPUT IN THE INPUT QUEUE (DC19Q). HOWEVER, IF IT IS READING IN CARD DECKS IT ACTS AS IF IT WERE LOAD CONTROL (I.E. COM23). IT WILL NOT EXIT (KILL ITSELF) IN THIS CASE UNTIL A ?END CARD IS RECEIVED AND ONLY THEN IF THERE IS NO MORE INPUT WAITING. IT ALSO WILL NOT EXIT IMMEDIATELY UPON NO MORE INPUT IF IT RECEIVES A RJE TERMINAL HALT LOAD MESSAGE. IT WILL WAIT IN CORE WHILE THE LOG ON SEQUENCE IS IN PROGRESS. IT WILL STAY THERE UNTIL THE LINE IS LOGGED ON WITH A VALID USERCODE (AND PASSWORD PERHAPS) OR UNTIL THERE ARE TOO MANY INVALID ATTEMPTS AT WHICH POINT THE LOG ON SEQUENCE IS TERMINATED.

THIS PROCEDURE WILL TAKE DIFFERENT ACTION DEPENDING UPON THE INPUT ADDRESS DESIGNATE CHARACTERS. IF THE INPUT IS FROM THE SPO THE PROCEDURE KEYIN WILL BE FORKED WITH THE INFORMATION PART OF THAT BLOCK OF THE MESSAGE SENT TO KEYIN. IF THE INPUT IS FROM THE CARD READER THE CARD IMAGES ARE PLACED IN DECKS. IF THE INPUT IS CONTROL IT WILL TAKE ACTION DEPENDING UPON THE TYPE OF CONTROL MESSAGE. THE 00 MESSAGE WILL PUT THE PROCEDURE IN THE LOGON SEQUENCE AS THIS IS THE RJE TERMINAL HALT LOAD MESSAGE. THE 01 MESSAGE IS A UNIT NOT READY MESSAGE AND WILL RESULT IN THE PROCEDURE RJEUNITCHANGE BEING CALLED. IF THE MESSAGE 02 IS RECEIVED IT MEANS A UNIT WENT READY AND WILL RESULT IN THE PROCEDURE RJEUNITCHANGE BEING FORKED. ANY OTHER INPUT ADDRESS WILL RESULT IN THE MESSAGE BEING DISCARDED AS THEY ARE PRESENTLY NOT ALLOWED.

2. TWXOUT - THIS IS THE PROCEDURE THAT FORMATS ALL OUTPUT FOR MESSAGES TO THE RJE TERMINALS. IT PUTS ON THE SOH , THE ADDRESS CHARACTERS, THE SLOT FOR THE TRANSMISSION NUMBER, THE STX, THE BLOCKING CHARACTERS - THE CARRIAGE RETURNS, THE INFORMATION PORTION, AND THE ETX. IN THE INFORMATION PORTION IT CONVERTS LEFT ARROWS TO CONTROL "@" AND "≠" CHARACTERS TO "≠≠". IF THE OUTPUT IS GOING TO THE CARD PUNCH OR LINE PRINTER IT WILL DO BLANK SUPPRESSION AND ALSO ATTEMPT TO BLOCK MORE THAN ONE MESSAGE AT A TIME IN AN EFFORT TO CUT DOWN WASTED TIME INVOLVED IN WAITING FOR THE DATA SET TO TURN AROUND.

MESSAGES ARE BLOCKED BY SEPARATING THEM WITH THE CARRIAGE RETURN CHARACTER. IN THE CASE OF THE PRINTER THE PRINTER CONTROL CHARACTERS IMMEDIATELY FOLLOW THE CARRIAGE RETURN.

THEN FOLLOWS THE LINE OR CARD IMAGE. BLANK SUPPRESSION IS DONE UNDER THE ALGORITHM THAT IF THERE IS A STRING OF FIVE BLANKS OR LONGER, THEY ARE REPLACED BY THE SEQUENCE:

ESC CC1 CC2

WHERE ESC IS THE ESCAPE CHARACTER (CONTROL ";") AND CC1 AND CC2 ARE THE TENS AND UNIT PARTS OF THE DECIMAL NUMBER OF CONSECUTIVE BLANKS. THUS A STRING OF 32 BLANKS WOULD LEAD TO THE REPLACEMENT STRING OF : "≠" ";" "≠" "3" "2". BLANK SUPPRESSION IS ALSO HANDLED BY NOT SENDING THE TRAILING BLANKS OF A CARD IMAGE OR A LINE OF PRINT.

AFTER TWXOUT HAS FORMATTED THE MESSAGE IT WILL ATTEMPT TO PLACE THE MESSAGE IN THE OUTPUT QUEUE. FIRST IT MUST SEE IF THE UNIT THAT IT IS SENDING THE MESSAGE TO IS READY. IF IT IS THE MCP CHECKS TO SEE IF THERE ARE ANY SOFTWARE OUTPUT BUFFERS AVAILABLE. (THERE ARE CURRENTLY FOUR BUFFERS ASSIGNED TO THE RJE LINES IN THE DC19Q). IF THERE IS AN AVAILABLE BUFFER IT IS TAKEN AND THE MESSAGE IS PLACED IN THE STATIONMESSAGEHOLDER QUEUE. STATIONMESSAGEWRITER IS FORKED (IF IT IS NOT ALREADY RUNNING) AND IT WILL SEND OUT THE MESSAGE. IF THE UNIT IS NOT READY OR THERE ARE NO AVAILABLE BUFFERS TWXOUT WILL WAIT UNTIL THESE CONDITIONS ARE SATISFIED. THE EXCEPTION TO THIS IS IF THE UNIT IS THE SPO AND IT IS MARKED AS NOT READY THE MESSAGE WILL BE PLACED IN THE RJEWAITQ AND TWXOUT WILL IMMEDIATELY BE EXITED.

3. RJEUNITCHANGE - THIS IS CALLED IF A UNIT WENT NOT READY. THIS PROCEDURE WILL THEN MARK THE UNIT AS NOT READY IN THE STATION WORD AND WILL THEN REMOVE ANY MESSAGES WAITING TO GO OUT IN THE STATIONMESSAGE QUEUE. IT MOVES THEM TO THE RJEWAITQ FOR STORAGE UNTIL THE UNIT BECOMES READY AGAIN.

THIS PROCEDURE IS FORKED IF A UNIT WENT READY. IN THIS CASE THE PROCEDURE WILL MARK THE UNIT AS READY AND WILL REMOVE THE MESSAGES WAITING IN THE RJEWAITQ FOR THAT UNIT TO BECOME READY AGAIN AND WILL PLACE THEM BACK INTO THE STATIONMESSAGE QUEUE. IT MAY HAVE TO WAIT FOR SOFTWARE OUTPUT BUFFERS TO BECOME AVAILABLE WHEN IT IS PLACING ENTRIES BACK IN THE STATIONMESSAGE QUEUE. (THIS IS THE REASON THAT IT IS FORKED INSTEAD OF JUST CALLED). IF WHILE WAITING FOR BUFFERS THE UNIT GOES NOT READY AGAIN THE PROCEDURE WILL KILL ITSELF. OTHERWISE IT WILL WAIT TO KILL ITSELF UNTIL ALL MESSAGES IN THE RJEWAITQ WAITING FOR THE NEWLY READY UNIT ARE REMOVED.

RJEWAITQ IS A NEW PRT CELL USED TO STORE THE MESSAGES WAITING FOR NOT READY UNITS. IT IS INITIALIZED TO

```
P(.RJEWAITQ)&P(.RJEWAITQ)[CTF]&@777[9:39:9];
```

ENTRIES ARE PLACED IN THE QUEUE USING THE LINK-LIST-LOOKUP OPERATOR. THE FORMAT OF THE ENTRY IN THE RJEWAITQ IS:

```
(NEXT LINK)&(LAST LINK)[CTF]&(TU/BUFF)[9:9]&
(NOT READY UNIT)[3:6];
```

ENTRIES ARE PLACED IN THIS QUEUE BY TWXOUT AND RJEUNITCHANGE AND ARE REMOVED BY RJEUNITCHANGE.

B.7.8 OLD MCP PROCEDURES MODIFIED FOR RJE

SEVERAL MCP PROCEDURES WERE ADDED TO OR CHANGED EXTENSIVELY TO ALLOW RJE TO RUN WITH THEM. SOME OF THEM AND THE MANNER IN WHICH THEY WERE CHANGED ARE LISTED BELOW.

1. SPOUT - THIS IS THE PROCEDURE THAT HANDLES ALL OUTPUT TO THE SPO. IT HAS HAD CODE ADDED TO IT TO CHECK TO SEE IF A SPO WRITE IS GOING TO A RJE SPO OR SHOULD BE GOING TO A RJE SPO. IT DETERMINES THE TERMINAL UNIT AND BUFFER TO BE OUTPUT TO FROM THE [9:9] FIELD OF THE PARAMETER PASSED TO IT. IF THIS IS ZERO BUT THERE IS A MIX ASSOCIATED WITH THE JOB IT WILL CHECK THE JAR [MIX,6].[9:9] TO SEE IF THE JOB WAS STARTED BY A RJE TERMINAL. IF IT WAS OR THE TU/BUFF INDICATE THAT THE WRITE SHOULD GO TO A RJE TERMINAL IT WILL CALL THE PROCEDURE TWXOUT. IT WILL BEFORE THE PROCEDURE IS CALLED FORMAT THE MESSAGE TO ELIMINATE ILLEGAL CHARACTERS ("≥","≤",">","<","≠"), PUT CARRIAGE RETURN-LINE FEEDS AT THE END AND POSSIBLY START OF THE MESSAGE, PUT CARRIAGE RETURN-LINE FEEDS AFTER EVERY MULTIPLE OF 72 CHARACTERS (IN ORDER TO PREVENT CHARACTERS FROM WRITING OVER EACH OTHER), AND WILL COUNT THE CHARACTERS. IT WILL PASS TO TWXOUT THE NUMBER OF CHARACTERS TO BE WRITTEN, THE UNIT WHICH IS ONE (FOR THE SPO), THE NUMBER OF CARRIAGE RETURN-LINE FEEDS, THE TERMINAL UNIT AND BUFFER NUMBER OF THE RJE STATION, AND, OF COURSE, THE MESSAGE TO BE SENT OUT.

2. COM19 - THIS IS THE PROCEDURE THAT PRINTS BACK UP DISK AND TAPE FILES. IT ALSO HAS TO CALL TWXOUT INSTEAD OF DOING REGULAR LINE

PRINTER I/O.IF THE BACK UP IS TICKETED FOR A RJE TERMINAL INSTEAD OF A REGULAR LINE PRINTER. IT CHECKS JAR[MIX,6].[9:9] AND IF IT IS NON ZERO IT MUST DO THE TWXOUT. COM19 WILL PASS TO TWXOUT THE TU/BUFF GATHERED FROM JAR[MIX,6].[9:9], THE UNIT - EITHER THREE FOR A PRINTER BACK UP FILE OR TWO FOR A PUNCH BACK UP FILE, AND A BLOCK OF FIVE LINES OF PRINT. TWXOUT WILL ATTEMPT TO SEND ALL FIVE OF THE RECORDS IF IT CAN FIT IN THE SOFTWARE OUTPUT BUFFER (WHICH IS 60 WORDS LONG). IF IT CANNOT IT WILL SEND OUT THE RECORDS IN TWO CHUNKS.

3. PRINTORPUNCHWAIT - THIS IS THE PROCEDURE THAT INITIATES A PRINTER BACK UP JOB BY CREATING THE CONTROLCARD THAT STARTS IT. IT CAN BE CALLED IN THREE WAYS WHEN A UNIT GOES READY, WHEN A BACK UP FILE IS CLOSED, OR WHEN AN OPERATOR DOES A PB KEYIN. THIS PROCEDURE IS ALERTED THAT IT IS LOOKING AT AN ATTEMPT TO DO A RJE BACK UP INSTEAD OF A MAIN SITE ONE BY THE [9:9] FIELD OF THE SECOND PARAMETER PASSED TO IT. IF IT IS PASSED A BACK UP TAPE FILE OR A BACK UP DISK FILE, IT WILL LOOK IN THE STATION WORD TO SEE IF THAT UNIT IS AVAILABLE. IF SO IT WILL GENERATE A CONTROLCARD TO INITIATE THE PRINTERBACKUP INDICATING THAT IT IS SUPPOSED TO GO TO THE RJE LINE SPECIFIED IN THE [9:9] FIELD OF THE SECOND PARAMETER. IF IT IS PASSED A READY UNIT IT WILL ATTEMPT TO FIND ANY BACK UP FILES ON DISK THAT ARE BOTH TICKETED FOR THAT UNIT AND ORIGINATED AT THAT RJE TERMINAL. (THAT IS THE [39:9] FIELD OF HEADER[6] IS EQUAL TO THE [9:9] FIELD OF THE SECOND PARAMETER). IF TRUE THE CONTROLCARD FOR THE PRINTBACKUP WILL BE GENERATED OTHERWISE THE STATION WORD WILL BE UPDATED TO INDICATE THAT THAT UNIT IS AVAILABLE. THOUGH THIS PARAGRAPH MOSTLY MENTIONS PRINTER BACK UPS, IT APPLIES EQUALLY TO PUNCH BACKUPS TOO.

4. CONTROLCARD - THIS IS THE PROCEDURE THAT SCANS THE CONTROL CARDS AND CREATES THE SHEET ENTRIES WHICH ARE THE SKELETONS USED FOR STARTING UP THE JOBS. IT CAN BE CALLED FOR A RJE LINE IN TWO WAYS: BY ENTERING CONTROL CARDS THROUGH THE RJE SPO BY MEANS OF THE CC OR ? KEYINS OR BY READING DECKS THROUGH THE RJE CARD READER. (THE LATTER PRESUMES THAT THE SYSTEM HAS BEEN GIVEN PSEUDO-READERS). CONTROLCARD IS PASSED ONE PARAMETER (CARD) AND THE [9:9] FIELD OF IT IS USED TO INDICATE THE TU/BUFF OF THE DATACOM LINE RESPONSIBLE FOR THIS CONTROL CARD. THE PROCEDURE WILL CHECK TO SEE IF THAT DATACOM LINE IS A RJE LINE (BY CHECKING [47:1] OF THE STATION WORD FOR THAT TU/BUFF) AND IF IT IS THE SHEET ENTRY SHEET[6].[9:9] WILL BE MARKED WITH THE TU/BUFF OF THE RJE LINE. THE PROCEDURE WILL ALSO USE THE USERCODE THAT LOGGED IN THE RJE LINE AS THE DEFAULT USER FOR THE JOB BUT WILL OVERWRITE IT IF THE CONTROL CARDS HAVE A USER CARD WITH

THEM. WHEN THE SHEET ENTRY IS COMPLETED IT WILL BE LINKED AND SELECTRUN CALLED. WHEN THE JOB IS RUN THE SHEET[6] ENTRY WILL BE READ INTO JAR[MIX,6] AND THE JOB WILL BE MARKED AS A RJE-ORIGINATED JOB.

5. FILEOPEN,FILECLOSE,BACKCLOSE,PBIO - THESE PROCEDURES ARE RESPONSIBLE IN THE CASE OF RJE LINES FOR MAKING SURE THAT OUTPUT TO LINE PRINTERS OR CARD PUNCHES ARE CORRECTLY ROUTED TO THE RIGHT PLACE. EACH OF THESE WILL CHECK TO SEE IF THE JOB THAT IS CALLING THEM IS A RJE JOB (BY CHECKING FOR A NON-ZERO JAR[MIX,6]. [9:9] FIELD). IF IT IS A RJE JOB IT WILL THEN CHECK TO SEE IF THE UNIT BEING ACCESSED (PRINTER OR PUNCH) IS ATTACHED TO THE RJE STATION. IF IT IS AND THE FILE IS BEING OPENED IT WILL ROUTE THE OUTPUT TO BACK UP DISK. IF IT IS AND THE FILE IS BEING CLOSED IT WILL MARK THE [39:9] FIELD OF THE HEADER[6] WITH THE TU/BUFF OF THE RJE LINE AND WILL CALL PRINTORPUNCHWAIT FOR THE UNIT AND LINE. FILE OPEN IS ALSO RESPONSIBLE FOR CHECKING THAT NO LABEL EQUATION IS DONE OF A DATACOM FILE TO A RJE LINE. IF A PROGRAM ATTEMPTS TO DO THIS, IT WILL BE DS-ED BY FILEOPEN.

6. STATIONMESSAGEWRITER - THIS IS THE PROCEDURE THAT PERFORMS ALL TYPE 19 OUTPUT, MCP OUTPUT TO DATACOM, AND RJE OUTPUT. IT RUNS AS AN INDEPENDENTRUNNER AND ATTEMPTS TO OUTPUT ALL MESSAGES LINKED IN THE STATIONMESSAGEHOLDER QUEUE. IF IT HAS A MESSAGE TO SEND OUT TO A RJE STATION IT WILL CHECK TO SEE IF THE LINE IS IN USE. IF NOT IT WILL SEE IF THE LINE IS IN THE IDLE STATE OR NOT THAT IS WHETHER IT HAS BEEN SELECTED OR NOT. IF NOT IT WILL SEND OUT THE ENQ CHARACTER WHICH ACTS AS THE SELECTING PROCESS. AFTER IT RECEIVES AN ACK TO THE SELECT (OR IF THE LINE WAS ALREADY SELECTED) IT WILL SEND OUT THE MESSAGE. THIS PROCESS MAY TAKE MORE THAN ONE I/O SINCE THE MESSAGE MAY BE LONGER THAN THE ADAPTER BUFFER LENGTH. WHEN THE MESSAGE HAS BEEN COMPLETELY AND CORRECTLY TRANSMITTED, IT WILL BE DISCARDED AND THE NEXT MESSAGE FOR THAT LINE SENT OUT. IF THERE WAS AN ERROR (THE RJE STATION NAKKED THE MESSAGE) THE MESSAGE WILL BE RETRANSMITTED INSTEAD OF BEING DISCARDED. (NOTE: THE OUTPUT PROCEDURE IS NOT DESTRUCTIVE OF THE MESSAGE FOR RJE MESSAGES. THIS IS DIFFERENT THAN THE REGULAR TYPE 19 OR MCP MESSAGES OUTPUT THROUGH STATIONMESSAGEWRITER AND CONTRARY TO THE COMMENT THAT APPEARS AT THE START OF IT. IT IS NECESSARY THAT IT NOT BE DESTRUCTIVE SO THAT THE MESSAGE MAY BE RESENT IN THE CASE OF A NAK).

7. NINETEENREADER - THIS IS THE PROCEDURE RESPONSIBLE FOR TYPE 19 AND RJE INPUT. IT GATHERS INPUT FROM THE ADAPTER UNTIL IT SENSES THE END OF THE MESSAGE. IN THE CASE OF THE INPUT BEING FROM A RJE LINE THE PROCEDURE WILL TAKE DIFFERENT ACTION

DEPENDING UPON THE TYPE OF MESSAGE THAT IS BEING RECEIVED. IT DOES THIS BY CHECKING THE FIRST CHARACTER OF THE MESSAGE. IF IT IS A "≥" IT IS A SOH AND THIS WAS A REGULAR MESSAGE. IT WILL SEND THE MESSAGE TO RJEINPUT AND THEN WILL RESPOND TO THE MESSAGE BY SENDING ANOTHER MESSAGE IF THERE ARE ANY WAITING IN THE OUTPUT QUEUE (IT DOES THIS BY CALLING STATIONMESSAGEW RITER) OR IT SENDS AN ACK IF THERE ARE NONE. IF THE FIRST CHARACTER IS A "&" THE MESSAGE IS AN ACK. THE PROCEDURE WILL SEND ANOTHER MESSAGE IF THERE ARE ANY TO BE SENT OR WILL SEND AN EOT IF NOT. IF THE FIRST CHARACTER IS A "%" THE MESSAGE IS AN ENQ. THE PROCEDURE WILL RESPOND WITH AN ACK AND MARK THE LINE AS SELECTED. IF THE FIRST CHARACTER IS A "5" THIS IS A NAK. THE STATION RECEIVED THE LAST MESSAGE WITH PARITY ON IT AND IS REQUESTING THAT THE MESSAGE BE RETRANSMITTED. NINETEENREADER WILL TURN ON THE NAKKED BIT AND ASK STATIONMESSAGEWRITER TO RESEND THE LAST MESSAGE. IF THE MESSAGE RECEIVED IS INVALID (I.E. UNRECOGNIZABLE FIRST CHARACTER) ERROR RECOVERY WILL BE ATTEMPTED. NINETEEN READER WILL ALSO BE CHECKING PARITY WHEN IT RECEIVES THE MESSAGE AND WILL SEND A NAK AND DISCARD THE MESSAGE IF IT FINDS PARITY IN IT.

8. DCQUPT,INQUPT - THESE PROCEDURES HANDLE THE INTERRUPTS THAT COME OVER THE B-249. THEY HAVE BEEN CHANGED TO CHECK FOR RJE LINES AND WILL ROUTE THE APPROPRIATE INFORMATION TO THE CORRECT RJE HANDLING SPOT INSTEAD OF THE NORMAL PLACES. (E.G. IF A READ READY INTERRUPT IS RECEIVED FOR A RJE LINE NINETEENREADER WILL BE RUN INSTEAD OF POSSIBLY DCREAD OR INQUPT OR WHATEVER).

B.8 DC1000 EXECUTIVE

B.8.1 NOTHING-TO-DO

THE HEART OF THE EXECUTIVE PROGRAM IS THE NOTHING-TO-DO ROUTINE. ITS BASIC FUNCTION IS TO MONITOR THE STATUS OF THE PERIPHERALS AND INVOKE PROCEDURES AS NEEDED TO HANDLE COMMUNICATIONS WITH THE SYSTEM.

THE FIRST ACTION PERFORMED BY NOTHING-TO-DO IS TO CHECK THE STATE OF DC1000 OPERATIONS. SHOULD THE PROCESSOR BE IN CONTROL STATE (INTERRUPTS DISABLED), THE DC1000 WILL HALT. THE EXECUTIVE MUST BE IN NORMAL STATE UPON ENTRY TO THE NOTHING-TO-DO LOOP. ENTRY WHILE IN CONTROL STATE WOULD INDICATE AN INVALID EXIT FROM INTERRUPT PROCESSING, AND WOULD RESULT IN ALL FUTURE INTERRUPTS BEING IGNORED.

IF THE PROCESSOR IS IN NORMAL STATE, THE NEXT PORTION OF NOTHING-TO-DO CHECKS THE STATUS OF THE COMMUNICATIONS LINK WITH THE SYSTEM. IF THE LINE IS NOT ACTIVE, THE EXECUTIVE WILL CALL A PROCEDURE NAMED MODEM-IDLE. IT IS THE FUNCTION OF MODEM-IDLE TO DETERMINE IF THE CONDITIONS OF THE SYSTEM ARE SUCH THAT LINE ACTIVITY SHOULD BE INITIATED. IF ALL CONDITIONS ARE MET MODEM-IDLE CALLS OPEN-LINE WHICH INITIATES THE MODEM CODE AS AN INDEPENDENT RUNNER.

THE NEXT STEP OF NOTHING-TO-DO IS PERFORMED ONLY IF THE DC1000 HAS BEEN CONFIGURED WITH A PRINTER SPECIFIED AS PRESENT.

STEP THREE CONSISTS OF LOADING THE PRINTER STATUS THEN CALLING A PROCEDURE NAMED PRINTER-CHECK. PRINTER-CHECK KNOWS HOW TO HANDLE ALL OF THE PRINTER CONDITIONS WHICH CAN OCCUR. THE HANDLING OF THE PRINTER IS LEFT ENTIRELY TO PRINTER-CHECK.

THE FOURTH STAGE OF NOTHING-TO-DO IS PERFORMED ONLY IF THE SYSTEM HAS BEEN GENERATED WITH A CARD READER SPECIFIED AS AN INPUT DEVICE.

THIS STEP OF THE LOOP CHECKS THE STATUS OF THE CARD READER. SHOULD THE SOFTWARE STATUS OF THE READER BE NOT READY, THE EXECUTIVE WILL BRANCH TO CARD-READER-NOT-READY. IF ON THE OTHER HAND, NO CARD I/O IS IN PROGRESS, A CALL WILL BE MADE ON CARD-READER- READY.

THE NEXT PHASE OF NOTHING-TO-DO CONSISTS OF A TWO PART TEST OF THE STATUS OF THE SPO. FIRSTLY, IF THE SPO IS WRITE READY, AN INDIRECT BRANCH IS MADE ON SPO-LABEL. AT THIS TIME, SPO-LABEL WILL CONTAIN THE ADDRESS OF THE NEXT PIECE OF SPO OUTPUT CODE THAT IS TO BE EXECUTED. THE SECOND PART OF THE TEST CONSISTS OF CHECKING FOR KEYBOARD INPUT. IF KEYBOARD DATA IS PRESENT, NOTHING-TO-DO WILL BRANCH TO THE SPO INPUT ROUTINE.

THE NEXT TO LAST OPERATION IN NOTHING-TO-DO IS CHECKING FOR DEVICE NOT READY CONDITIONS CAUSED BY LACK OF MEMORY. WHENEVER THE SYSTEM TRIES TO SEND MORE DATA TO A DEVICE THAN MEMORY CAN CONTAIN, A MESSAGE IS SENT TO SYSTEM INDICATING THAT THAT DEVICE IS NOT READY. THIS OPERATION OF NOTHING-TO-DO CHECKS TO SEE IF THE CONDITIONS ARE SUCH THAT A MESSAGE SHOULD BE SENT TO THE SYSTEM INFORMING THE SYSTEM THAT A DEVICE IS NOW READY. IF A DEVICE IS NOT READY, AND ITS QUEUE HAS BEEN EXHAUSTED, A DEVICE READY MESSAGE SILL BE SENT TO THE SYSTEM.

THE FINAL OPERATION CONSISTS OF CHECKING THE ENQ TIMER. THE TIMER IS ACTUALLY CHECKED ONLY IF THE LAST OPERATION ON THE LINE WAS THE TRANSMISSION OF AN ENQ. BY CONVENTION, THE SYSTEM HANDLES ALL OTHER

TIMEOUTS. HOWEVER, THERE ARE PROBLEMS IF BOTH THE MCP AND THE DC1000 TRANSMIT AN ENQ AT THE SAME TIME.

IF THE LAST ACTIVITY ON THE LINE WAS NOT TRANSMITTING AN EOT OR THE CLOCK IS NEGATIVE, INDICATING THAT A TIMEOUT HAS NOT OCCURED YET, A BRANCH TO NOTHING-TO-DO IS PERFORMED. OTHERWISE, IF LESS THAN FOUR ENQS HAVE BEEN SENT, THE LINE STATUS IF CLEANED UP, AND THE TRANSMISSION OF ANOTHER ENQ IS STARTED, FOLLOWED BY A BRANCH TO THE START OF NOTHING-TO-DO. IF, HOWEVER, FOUR ENQS HAVE ALREADY BEEN SENT, AN ERROR MESSAGE IS PRINTED AND A BRANCH TO THE START OF NOTHING-TO-DO IS PERFORMED.

NOTHING-TO-DO IS EXECUTED IN NORMAL STATE, THEREFORE, IT OR ANY OF THE PROCEDURES THAT IT TRANSFERS CONTROL TO CAN BE INTERRUPTED AT ANY TIME. HOWEVER, DUE TO THE ACTUAL DATA LOAD, MOST OF THE DC1000-S TIME WILL BE SPENT EXECUTING THE NOTHING TO DO CODE. THIS INSURES THAT ANY CHANGES OF STATUS WILL BE DETECTED AND ACTED UPON AT THE EARLIEST POSSIBLE MOMENT.

B.8.2 PRINTER-CHECK

PRINTER-CHECK IS A PROCEDURE THAT IS CALLED EACH TIME THAT NOTHING-TO-DO IS EXECUTED. THIS PROCEDURE IS EXECUTED IN NORMAL STATE, EXCEPT FOR 44 - 48 MICRO SECONDS DURING WHICH THE PRINTER QUEUE IS BEING UPDATED. THE INTERRUPTS ARE DISABLED TO PREVENT THE MODEM CODE FROM ATTEMPTING TO UPDATE THE QUEUE AT THE SAME TIME AS PRINTER-CHECK IS PERFORMING ITS UPDATE.

PRINTER-CHECK FIRST TESTS THE STATUS OF THE LINE PRINTER TO DETERMINE IF THE PRINTER HAS BEEN INTERLOCKED. IF IT HAS BEEN INTERLOCKED, THE PRINTER CONTROL IS CLEARED, AND THE PRINTER READY LINE IS SAMPLED. IF THE PRINTER IS NOT READY, THE STATUS BIT WHICH INDICATES THAT THE NEXT TIME THE PRINTER IS READY, I/O CAN BEGIN IS SET. THEN AN EXIT IS PERFORMED. IF, ON THE OTHER HAND, THE PRINTER IS READY, THE SECOND INTERLOCK BIT IS TESTED. IF THE SECOND INTERLOCK IS TRUE, THEN I/O CAN BE STARTED AGAIN. IN THAT CASE, THE INTERLOCKS ARE CLEARED, AND EXECUTION CONTINUES AS IF THE PRINTER WERE NOT INTERLOCKED AT ENTRY TO PRINTERCHECK. IF INTERLOCK2 IS NOT TRUE, CONTROL IS RETURNED TO NOTHING-TO-DO.

IF THE PRINTER IS NOT INTERLOCKED, THE FOLLOWING ACTION TAKES PLACE. IF THE PRINTER IS PERFORMING AN OUTPUT, PRINTER-CHECK WILL EXIT. OTHERWISE, THE RESULTS OF THE PREVIOUS OPERATION ARE EXAMINED TO DETERMINE IF ANY ERRORS OCCURED. IF THERE WERE ERRORS, AN ERROR MESSAGE

IS PRINTED, THE PRINTER IS INTERLOCKED AND THE PROCEDURE IS EXITED.

IF NO ERRORS OCCURED, PRINTER-CHECK WILL ATTEMPT TO START ANOTHER WRITE TO THE PRINTER.

THERE ARE SEVERAL STEPS WHICH MUST BE PERFORMED IN ORDER TO INITIATE AN I/O. FIRSTLY, THE PRINTER CONTROL IS CLEARED. THEN, THE PRINTER-FORMAT-COUNT IS TESTED TO SEE IF THE WRITE JUST COMPLETED WAS NOT THE LAST I/O IN THE WRITE. THE PRINTER-FORMAT- COUNT CONTAINS THE TWOS COMPLEMENT VALUE OF THE NUMBER OF TIMES THAT THE LINE FORMAT IS TO BE DUPLICATED TO PROVIDE THE DESIRED AMOUNT OF PAPER MOTION FOR A GIVEN WRITE. IF THE PRINTER-FORMAT- COUNT IS NOT ZERO THEN PAPER MOTION FOR THE LAST WRITE HAS NOT BEEN COMPLETED. THIS CONDITION IS HANDLED BY BRANCHING TO THE CODE WHICH OUTPUTS THE PRINTER FORMAT BYTE.

IF THE NEXT TEST IS REACHED, THEN THE PRINTER IS READY TO PERFORM ANOTHER WRITE. ACCORDINGLY, PRINTER-CHECK WILL EXAMINE THE PRINTER QUEUE TO DETERMINE IF THERE IS ANY DATA TO WRITE TO THE PRINTER. IF THE QUEUE IS EMPTY, AN EXIT WILL BE MADE TO NOTHING-TO-DO.

HAVING DETERMINED THAT ALL CONDITIONS TO INITIATE A WRITE ARE SATISFIED, THE LINE-FORMAT AND LINE-REPLICATOR OF THE NEXT WRITE WILL BE EXAMINED. IF THE WRITE IS A WRITE AFTER, THE LINE- FORMAT AND LINE-REPLICATOR WILL BE CHANGED TO A WRITE WITHOUT SPACING, AND THE PAPER MOTION WILL BE SET UP AND A BRANCH TO THE CODE WHICH OUTPUTS THE FORMAT CONTROL WORD WILL BE MADE.

IF THE WRITE WAS A WRITE BEFORE OR A WRITE WITHOUT PAPER MOTION, NORMAL DATA TRANSFER TO THE PRINTER WILL TAKE PLACE.

AFTER THE PHYSICAL BUFFER OF THE PRINTER HAS BEEN FILLED, THE SOFTWARE BUFFER IS DELINKED FROM THE QUEUE AND FORGOTTEN. THEN, THE CLOCK IS UPDATED TO RECORD THE TIME NECESSARY FOR THE I/O TO BE INITIATED.

THE LAST PHASE OF PRINTER-CHECK TRANSFERS A FORMAT WORD TO THE PRINTER. THIS FINAL BYTE TRANSFER ORDERS THE PRINTER TO PRINT ITS BUFFER THEN MOVE THE PAPER AS THE FORMAT WORD SPECIFIES. THE PRINTER-FORMAT-COUNT IS INCREMENTED TO ACCOUNT FOR THE PAPER MOTION ORDERED BY THE FORMAT WORD (NEXT-PRINTER-FORMAT).

THE FLOW OF CONTROL IS THEN RETURNED TO NOTHING-TO-DO.

AT A NUMBER OF STRATEGIC POINTS ALONG THIS PROCESS, THE PRINTER READY LEVEL IS TESTED. IF THE PRINTER IS NOT READY AT THE TIME THAT ANY ONE OF THESE TESTS IS MADE, AN ERROR MESSAGE WILL BE PRINTED, THEN THE

PRINTER WILL BE INTERLOCKED.

IN THE INTERESTS OF EFFICIENCY, AN ATTEMPT IS MADE TO TRANSFORM MULTIPLE SINGLE SPACES INTO DOUBLE SPACES. THE RESULT OF THIS OPERATION CAN IN SOME CASES ALMOST DOUBLE THE THROUGHPUT OF THE PRINTER.

B.8.3 CARD-READER-NOT-READY

CARD-READER-NOT-READY IS EXECUTED EACH TIME THAT THE NOTHING TO DO LOOP DETECTS THAT THE CARD READER HAS JUST BEEN MARKED NOT READY.

THIS ROUTINE CHECKS TO SEE IF THE CARD READER HAS BEEN INTERLOCKED. INTERLOCK TRUE IMPLIES THAT AN ERROR MESSAGE HAS ALREADY BEEN PRINTED, AS IT DOES ON ALL OTHER DEVICES. SINCE A PREVIOUS ERROR MESSAGE HAS BEEN PRINTED, IF THE READER IS INTERLOCKED, THE NOT READY MESSAGE WHICH WOULD NORMALLY BE PRINTED IS SUPPPRESSED. IF THE INTERLOCK IS FALSE, THE NOT READY MESSAGE IS PRINTED, THEN THE INTERLOCK IS SET. IN EITHER CASE, THE FLOW OF CONTROL IS RETURNED TO NOTHING-TO-DO.

B.8.4 CARD-READER-READY

CARD-READER-READY IS CALLED WHEN NOTHING-TO-DO CHECKS THE STATUS OF THE READER AND FINDS IT READY AND IDLE.

THE FIRST THING THAT THIS PROCEDURE DOES IS FIND OUT IF A CARD IMAGE HAS BEEN READ INTO MEMORY. IF THERE IS A CARD IMAGE IN CORE, THEN CONTROL IS GIVEN TO A ROUTINE WHICH WILL CONVERT THE CARD TO ASCII AND COMPRESS ALL STRINGS OF SIX OR MORE BLANKS INTO THE PROPER FORM FOR COMPRESSED DATA.

NEXT, THE INTERLOCK BIT FOR THE CARD READER IS TESTED. IF THE READER IS INTERLOCKED, THE SAME ACTION THAT IS APPLIED TO THE PRINTER WHEN IT IS INTERLOCKED IS APPLIED TO THE CARD READER.

IF THE READER WAS NOT INTERLOCKED OR INTERLOCKS WERE JUST CLEARED, THE READING OF A CARD IS INITIATED. THEN CARD READER READY RETURNS TO NOTHING-TO-DO.

THE PROCEDURE WHICH TRANSLATES THE CARD IMAGE IS RATHER STRAIGHT FORWARD. IT GETS A BUFFER THEN PROCEEDS TO MOVE THE CARD IMAGE FROM THE

PHYSICAL I/O BUFFER TO THE SOFTWARE BUFFER. WHILE DOING THIS, IT INTERPRETS THE PUNCHES OF THE CARD AND TRANSLATES THEM INTO THE EQUIVALENT ASCII CHARACTER. AT THE SAME TIME, SEQUENCES OF MORE THAN FIVE BLANKS ARE COMPRESSED INTO THE STANDARD FORM OF COMPRESSED BLANK SEQUENCES AND, A CHECK FOR INVALID CHARACTERS IF PERFORMED. IF AN INVALID CHARACTER IS FOUND, AN ERROR MESSAGE IS PRINTED, THE BUFFER IS FORGOTTEN, THE READER IS INTERLOCKED AND AN EXIT IS PERFORMED.

UPON SUCCESSFUL CONVERSION OF THE CARD IMAGE, THE SOFTWARE BUFFER IS LINKED INTO THE READER QUEUE, THE CLOCK IS UPDATED, THEN, THE NEXT I/O IS INITIATED.

THIS PROCEDURE IS EXECUTED PREDOMINANTLY IN NORMAL STATE. AS WITH THE PRINTER, INTERRUPTS ARE DISABLED ONLY WHILE THE QUEUE IS BEING UPDATED.

B.8.5 SPO-INPUT

SPO-INPUT IS EXECUTED WHEN THE NOTHING-TO-DO LOOP FINDS THAT KEYBOARD INPUT IS WAITING TO BE READ.

SPO-INPUT CHECKS TO SEE IF A KEYIN IS CURRENTLY IN PROGRESS. IF THIS CHARACTER IS THE FIRST IN A MESSAGE, SPACE IS FOUND AND THE STATUS OF THE SPO IS UPDATED TO SHOW THAT INPUT IS IN PROGRESS. THEN THE CHARACTER IS READ AND STORED.

HOWEVER, IF A KEYIN IS IN PROGRESS ALREADY, THE CHARACTER IS READ AND STORED STRAIGHT AWAY.

AFTER THE CHARACTER HAS BEEN STORED, CONTROL IS RETURNED TO NOTHING-TO-DO.

STORAGE OF A CHARACTER ACTUALY INVOLVES QUITE A NUMBER OF STEPS. THE CHARACTER IS STORED IN THE BUFFER AND IN A STORAGE LOCATION NAMED CHAR. THEN THE CHARACTER IS CHECKED TO DETERMINE IF IT IS A RUBOUT OR A CARRIAGE RETURN. IF EITHER ONE IS ENCOUNTERED, THE PROPER ACTION IS TAKEN. AND, FINALLY, A TEST IS MAKE TO FIND OUT IF THE CHARACTER JUST RECEIVED WAS THE 72ND CHARACTER OF THE CURRENT LINE. IF IT IS, THEN THE STATUS OF THE SPO IS CHANGED SO THAT THE ECHO CODE WILL OUTPUT A CARRIAGE RETURN - LINE FEED AFTER THE CHARACTER IS PRINTED. BY THIS METHOD, SPO INPUT MESSAGES OF GREATER THAN 72 CHARACTERS ARE COMPLETELY READABLE.

IF THE CHARACTER READ IS A CARRIAGE RETURN, THE KEYBOARD INPUT WILL BE QUEUED IN THE KEYIN QUEUE FOR TRANSMISSION TO THE SYSTEM. THE SPO IS THEN MARKED INPUT IDLE.

THE OCCURANCE OF A RUBOUT RESULTS IN THE SPACE CURRENTLY BEING HELD FOR THE INPUT BEING FORGOTTEN, AND THE SPO STATUS BEING SET TO IDLE.

FOLLOWING STORAGE OF A CHARACTER AND ANY ACTION WHICH IT (THE CHARACTER) CAUSES, CONTROL IS RETURNED TO NOTHING-TO-DO.

NOTE THAT IN THE EVENT THAT SPO INPUT TRIES TO GET SPACE AND NO SPACE IS AVAILABLE, ERROR ACTION IS TAKEN. ANY SPACE WHICH IS ALREADY IN USE FOR THE KEYIN IS FORGOTTEN, AN ERROR MESSAGE IS QUEUED FOR THE SPO, AND STORING OF CHARACTERS IN THE BUFFER IS SUPPRESSED FOR THE REMAINDER OF THE KEYIN.

B.8.6 SPO-OUTPUT

THIS ROUTINE IS REACHED BY AN INDIRECT BRANCH THROUGH SPOUT- LABEL WHENEVER THE SPO IS WRITE READY. SPOUT-LABEL CONTAINS THE ADDRESS WHERE SPO OUTPUT PROCESSING IS TO RESUME WHEN THE SPO NEXT BECOMES WRITE READY. THE VALUE IN SPOUT-LABEL IS USUALLY PUT THERE BY A CALL ON SPOUT-SLEEP. SPOUT-SLEEP STORES AN ADDRESS IN SPOUT-LABEL AND TRANSFERS CONTROL TO NOTHING-TO-DO.

SPO-OUTPUT IS BASICALLY TWO PATHS OF CONTROL. ONE PATH HAS THE FUNCTION OF PRINTING SPO OUTPUT GENERATED BY THE SYSTEM OR BY THE DC1000. THE OTHER PATH IS RESPONSIBLE FOR ECHOING THE KEYBOARD INPUT BACK TO THE SPO SO THAT THERE WILL BE A VISIBLE RECORD OF THE CHARACTERS ENTERED ON THE SPO BY THE OPERATOR.

THE CODE WHICH ECHOS INPUT FROM THE KEYBOARD TO THE SPO IS VERY SIMPLE. THE VALUE STORED IN CHAR BY SPO INPUT IS TRANSFERED TO THE SPO. THEN A TEST IS MADE TO DETECT WHETHER A CARRIAGE RETURN LINE FEED SEQUENCE SHOULD BE WRITTEN BECAUSE THE LAST CHARACTER WAS THE 72ND CHARACTER OF A LINE. IT THE TEST IS TRUE, THEN A SPOUT- SLEEP WILL BE MADE WITH THE ADDRESS POINTING TO CODE TO PERFORM THIS FUNCTION.

THE ECHO CODE ALSO CHECKS TO SEE IF THE KEYBOARD INPUT HAS BEEN TERMINATED. IF SO, THE SPO WILL BE MARKED OUTPUT IDLE.

THE CODE WHICH PRINTS NORMAL SPO OUTPUT IS EVEN LESS COMPLICATED. THE MAIN DIFFERENCE IS THAT THE CHARACTERS TO BE PRINTED MUST BE FETCHED

FROM A BUFFER. AND, NO CHECKS FOR LONG LINES OR DELETES NEED BE PERFORMED.

WHEN THE I/O HAS BEEN COMPLETED, THE SPO IS MARKED IDLE.

WHEN THE SPO IS IDLE, NOTHING-TO-DO WILL BRANCH TO THE SPO OUTPUT INITIALIZE CODE EACH TIME THAT NOTHING-TO-DO IS EXECUTED. THIS IS DUE TO THE FACT THAT THE SPO IS CONTINUALLY WRITE READY, AND THE FACT THAT WHEN THE SPO IS WRITE IDLE, THE SPOUT LABEL ALWAYS POINTS TO THE INITIALIZE CODE.

THE SPO OUTPUT INITIALIZE CODE CHECKS FOR SYSTEM AND DC1000 GENERATED SPO OUTPUT, IN THAT ORDER. IF NO OUTPUT IS FOUND, CONTROL IS RETURNED TO NOTHING-TO-DO. IF AN OUTPUT MESSAGE IS FOUND, CONTROL IS TRANSFERED TO THE SPO OUTPUT CODE.

NOTE THAT SYSTEM SPO MESSAGES ARE PRINTED BEFORE LOCALLY GENERATED MESSAGES. THE EFFECT OF THIS ACTION IS TO FREE UP MEMORY AS RAPIDLY AS POSSIBLE.

THE NORMAL PRINTING CODE TESTS TO SEE IF THE BREAK KEY HAS BEEN PRESSED WHILE PRINTING WAS BEING DONE. IF A BREAK OCCURS, THE WRITE IN PROGRESS IS TERMINATED AND THE CODE TO ECHO BACK KEYBOARD INPUT IS ENTERED. OUTPUT WILL BE HELD UP UNTIL THE KEYIN JUST STARTED IS TERMINATED.

B.8.7 DATA COMMUNICATIONS ROUTINES

THE CODE WHICH HANDLES THE COMMUNICATIONS WITH THE SYSTEM IS THE LARGEST SINGLE CLASS OF CODE IN THE EXECUTIVE. ONCE ACTIVATED, IT WILL RUN UNTIL NO MESSAGES REMAIN TO BE SENT OR ARE BEING RECEIVED. ALL DETAILS OF HANDLING COMMUNICATIONS WITH THE SYSTEM ARE HANDLED BY THIS BLOCK OF CODE. THE DATACOM CODE RUNS AS NECESSARY, AND SLEEPS AWAITING INTERRUPTS WHEN IT HAS FINISHED HANDLING A CONDITION.

THE INTERRUPTS WHICH THE DATACOM CODE SLEEPS ON ARE:

1. DATA RECEIVED.

2. DATA TRANSMITTED.

3. CLEAR TO SEND STATUS CHANGE.

4. CARRIER STATUS CHANGE.

5. INTERLOCK STATUS CHANGE.

6. RING RECEIVED.

FOR A DETAILED DESCRIPTION OF THE MEANINGS OF THESE INTERRUPTS REFER TO THE DC1000-DC1200 SYSTEMS REFERENCE MANUAL. SUFFICE IT TO SAY, FOR PURPOSES OF THIS DISCUSSION, THAT THESE INTERRUPTS SIGNIFY THE COMPLETION OF AN OPERATION BY THE CONTROLLER, OR AN UNEXPECTED CHANGE IN THE STATUS OF THE COMMUNICATIONS LINK. THESE INTERRUPTS CAUSE CODE TO BE EXECUTED WHICH CAN HANDLE THE CONDITIONS WHICH THE REPRESENT.

THE COMMUNICATIONS CODE MAY BE INITIATED IN TWO WAYS. THE FIRST HAS ALREADY BEEN MENTIONED. THE NOTHING-TO-DO LOOP CAN CALL MODEM-IDLE AND CAUSE THE LINE TO BECOME ACTIVE. THE SECOND WAS THAT THIS BLOCK OF CODE CAN BECOME ACTIVE IS BY A CARRIER INTERRUPT. THEN THE LINE IS IDLE, A CARRIER INTERRUPT IMPLIES THAT THE SYSTEM IS PREPARED TO TRANSMIT A MESSAGE TO THE DC1000.

ONCE STARTED, THE COMMUNICATIONS CODE WILL CONTINUE TO RUN UNTIL THERE IS NO TRAFFIC ON THE LINE. ANY ERRORS WHICH OCCUR WILL BE CORRECTED IF POSSIBLE. ALL ACKING, NAKING AND RETRIES ARE UNDER THE CONTROL OF THE COMMUNICATIONS ROUTINES.

A NUMBER OF THINGS RESULT IN THIS OPERATIONS ABILITY TO PROCESS INDEPENDENTLY OF THE EXECUTIVE. THE PRIME MOVER IS THE ABILITY TO STOP EXECUTION WHILE AWAITING INTERRUPTS. IT IS POSSIBLE TO SLEEP THEN AWAKE AT THE NEXT STATEMENT OR TO AWAKE AT AN ADDRESS QUITE A WAY REMOVED FROM THE SLEEP. A SLEEP RESULTS IN THE ADDRESS TO RESUME EXECUTION AT BEING STORED INTO RECEIVE LABEL, TRANSMIT LABEL, CARRIER LABEL, CLEAR TO SEND LABEL OR RING LABEL. THE LACK OF AN INTERLOCK LABEL IS DUE TO THE FACT THAT AN INTERLOCK INTERRUPT HAS ONLY ONE MEANING, AN IRRECOVERABLE ERROR. THE SECOND FACTOR IN THE INDEPENDENT OPERATION OF THIS GROUP OF ROUTINES IS THE STORAGE LOCATIONS LAST ACTIVITY, MODEM STATUS AND NEXT ACTIVITY. THE VALUES IN THESE LOCATIONS DESCRIBE WHAT HAS HAPPENED, WHAT IS HAPPENING AND WHAT IS TO HAPPEN TO THE LINE.

THE BULK OF THE CODE WHICH HANDLES DATA TRANSFER IS SIMPLE AND STRAIGHT FORWARD. DATA IS RECEIVED OR TRANSMITTED THEN THE NEXT ACTIVITY IS SET IN ACCORDANCE WITH THE ACTIONS PERFORMED. ONLY A SMALL NUMBER OF ROUTINES HAVE ANY AMOUNT OF DECISION MAKING ABILITY. THE ROUTINES WHICH DO HAVE DECISION MAKING CAPACITY ARE:

1. ACKNOWLEDGE
2. DECYPHER RESPONSE
3. MESSAGE TRANSMISSION CODE
4. MESSAGE STORAGE CODE

B.8.7.1 ACKNOWLEDGE

A MESSAGE CAN BE ACKNOWLEDGED WITH EITHER AN ACK OR A MESSAGE. THE REASON FOR ALLOWING A MESSAGE AS AN ACKNOWLEDGEMENT IS TO DECREASE TO A MINIMUM THE AMOUNT OF TIME SPENT TURNING THE LINE AROUND BY REDUCING THE NUMBER OF TIMES THAT THE LINE MUST BE CHANGED FROM RECEIVE TO TRANSMIT OR TRANSMIT TO RECEIVE MODE. ACCORDINGLY, THE CODE EXECUTED TO ACKNOWLEDGE A MESSAGE MUST DECIDE IF A MESSAGE IS TO BE SENT RATHER THAN AN ACK. THIS DECISION IS MADE BY CALLING A SUBROUTINE NAMED FIND A MESSAGE. IF FIND A MESSAGE CANNOT FIND ANY THING TO TRANSMIT, A VALUE OF ZERO WILL BE RETURNED.

IN THE EVENT THAT A ZERO RESULTS FROM THE CALL, AN ACK WILL BE SENT. THEN NEXT ACTIVITY WILL BE SET AS DECYPHER RESPONSE. AND, FINALLY, THE LINE WILL BE CLOSED BY BRANCHING TO FINISH TRANSMIT.

IF, ON THE OTHER HAND, A MESSAGE IS FOUND TO BE READY FOR TRANSMISSION TO THE SYSTEM, THE STATUS OF THE LINE IS SET TO TRANSMIT MESSAGE. THEN A BRANCH IS MADE TO THE CODE WHICH TRANSMITS MESSAGES.

B.8.7.2 DECYPHER-RESPONSE

THE SAME REASONING MUST BE APPLIED TO THE ROUTINE WHICH LOOKS FOR THE ACKNOWLEDGEMENT TO A MESSAGE JUST SENT TO THE SYSTEM. ANY ONE OF A NUMBER OF RESPONSES MAY OCCUR TO THE ENDING OF A MESSAGE. THE CHARACTERS WHICH MAY BE RECEIVED IN RESPONSE TO A MESSAGE ARE ACK, NAK, ENQ, EOT OR SOH. EACH ONE OF THE ABOVE CHARACTERS HAS A DISTINCT AND DIFFERENT MEANING. THE CHARACTERS AND THEIR MEANINGS ARE:

ACK	LAST MESSAGE OK

NAK	RETRANSMIT LAST MESSAGE
ENQ	HALT LOAD HAS OCCURED
EOT	HALT LOAD HAS OCCURED
SOH	LAST MESSAGE WAS RECEIVED OK AND MESSAGE TO FOLLOW

THE ROUTINE WHICH LOOKS FOR ACKS IS, THEREFORE, CAPABLE OF RECOGNIZING EACH OF THESE CHARACTERS AND PERFORMING THE FUNCTIONS IN ACCORDANCE WITH THEIR IMPLIED MEANINGS. THE FOLLOWING IS THE FLOW OF CONTROL FOR DECYPHER RESPONSE.

IF AN ACK IS FOUND, THE SPACE OCCUPIED BY THE LAST MESSAGE TRANSMITTED TO THE SYSTEM IS FORGOTTEN. THEN, THE TRANSMIT TRANSMISSION NUMBER IS INCREMENTED. NEXT, FIND A MESSAGE IS CALLED. IF NO MESSAGE IS FOUND, NEXT ACTIVITY IS SET TO TRANSMIT AN EOT. THEN A SLEEP ON IGNORE INTERRUPTS IS DONE. IF, ON THE OTHER HAND, A MESSAGE IS READY TO SEND, NEXT ACTIVITY IS SET TO TRANSMIT A MESSAGE. THEN AN IGNORE INTERRUPTS SLEEP IS DONE.

THE OCCURANCE OF A NAK INDICATES THAT THE LAST MESSAGE WAS NOT RECEIVED PROPERLY BY THE SYSTEM. SO, THE DATA SENT IN THE LAST MESSAGE IS LINKED BACK INTO THE PROPER DEVICE QUEUE. THEN, NEXT ACTIVITY IS SET TO CAUSE MESSAGE TRANSMISSION. FINALLY, THE IGNORE INTERRUPTS SLEEP IS PERFORMED.

IF AN EOT OR AN ENQ IS RECEIVED, A HALT LOAD HAS OCCURED. THEREFORE, CLEANUP ACTION IS DONE SO THAT THE HALT LOAD MESSAGE CAN BE PROPERLY HANDLED. THE LINE WILL BE SET TO IDLE, THEN AN IGNORE INTERRUPTS SLEEP WILL BE PERFORMED.

SHOULD THE CHARACTER SOH BE RECEIVED, ACTION SIMILAR TO THAT ELICITED BY AN ACK WILL OCCUR. THE LAST MESSAGE"S SPACE WILL BE FORGOTTEN, THEN THE TRANSMISSION NUMBER WILL BE INCREMENTED. STATUS WILL THEN BE SET TO RECEIVE MESSAGE. AND, INSTEAD OF AN IGNORE INTERRUPTS SLEEP, A SLEEP TO AWAKE AT THE STORE MESSAGE CODE WILL BE PERFORMED.

THE OCCURANCE OF ANY OTHER CHARACTER WILL IDLE THE LINE.

B.8.7.3 TRANSMIT-MESSAGE

THE ROUTINE WHICH TRANSMITS MESSAGES DOES QUITE A BIT OF CHECKING AND DECISION MAKING. IT STARTS OFF BY TRANSMITTING THE HEADER THEN CALLING FIND A MESSAGE TO GET THE FIRST MESSAGE TO BE SENT. THEN THE DATA IS SENT. WITH THE TRANSMISSION OF A CHARACTER OF DATA, A TEST IS MADE TO FIND OUT IF ALL OF THE DATA IN THE PARTICULAR BUFFER BEING HANDLED HAS BEEN SENT TO THE SYSTEM. IF END OF DATA HAS NOT OCCURRED, A SLEEP IS PERFORMED. IF, HOWEVER, AN END OF MESSAGE OCCURS, THE FOLLOWING ACTION WILL TAKE PLACE. FIND A MESSAGE WILL BE CALLED TO FIND OUT IF THERE IS ANY MORE TRAFFIC TO BE SENT TO THE SYSTEM. IF NO MORE DATA IS READY FOR TRANSMISSION, A SLEEP IS DONE TO CAUSE AN ETX THEN THE BCC TO BE TRANSMITTED TO THE SYSTEM. FOLLOWING THE TRANSMISSION OF THE BCC, STATUS UPDATING WILL BE PERFORMED AND A BRANCH TO FINISH TRANSMIT WILL BE DONE.

IF, HOWEVER, A MESSAGE WAS FOUND BY FIND A MESSAGE, THE FOLLOWING ACTION TAKES PLACE. THE NEW BUFFER IS ADDED TO THE PART OF THE MESSAGE THAT HAS ALREADY BEEN SENT. THEN A SLEEP IS DONE SO THAT A CARRIAGE RETURN WILL BE SENT BEFORE THE TRANSMISSION OF THE NEW DATA BUFFER BEGINS.

B.8.7.4 RECEIVE-MESSAGE

THE PIECE OF CODE WHICH RECEIVES MESSAGES IS THE LONGEST SINGLE ROUTINE IN THE COMMUNICATIONS CODE. THE NEED FOR THIS IS BROUGHT ABOUT BY THE LARGE NUMBER OF DECISIONS WHICH MUST BE MADE. DECISIONS CONCERNING THE LINE AND THE PECULARITIES OF THE PERIPHERALS MUST BE MADE. EVEN THOUGH THE DECISIONS ARE TABLE DRIVEN TO THE GREADEST EXTENT POSSIBLE, A LARGE VOLUME OF CODE IS NEEDED TO HANDLE ALL OF THE CONDITIONS WHICH CAN OCCUR. THE RECEPTION OF A MESSAGE CONSISTS OF THE FOLLOWING STEPS.

FIRST, THE HEADER IS RECEIVED. ALL CONSTANT CHARACTERS IN THE HEADER ARE CHECKED FOR CONTENT. FAILURE TO MATCH THE EXPECTED CHARACTER IS A FORMAT ERROR.

AFTER THE TWO CHARACTERS SPECIFING WHICH UNIT THE MESSAGE IS FOR HAVE BEEN INTERPRETED, THE STORAGE LOCATIONS WHICH CONTAIN THE DEVICE DEPENDENT DATA ARE INITIALIZED.

NEXT, THE TRANSMISSION NUMBER IS RECEIVED AND COMPARED WITH THE NUMBER THAT IS EXPECTED. IF THE TWO DON"T MATCH, AN ERROR MESSAGE WILL BE PRINTED.

THEN, THE FINAL CHARACTER OF THE HEADER IS RECEIVED AND CHECKED.

FOLLOWING THE STX, THE RECEIVING AND STORING OF TEXT BEGINS. WHEN THE CODE WAKES UP AFTER THE FIRST CHARACTER ARRIVES, THE FIRST CHARACTER WILL BE FOUND TO BE A CARRIAGE RETURN. THE CARRIAGE RETURN WILL CAUSE THE NORMAL END OF BLOCK BRANCH TO BE TAKEN.

END OF BLOCK

THE END OF BLOCK ROUTINE PERFORMS ALL OF THE HOUSE KEEPING NECESSARY UPON ENCOUNTERING THE END OF A BLOCK. THE OPERATIONS ARE PERFORMED IF A BUFFER WAS BEING FILLED PRIOR TO THE CARRIAGE RETURN WHICH CAUSED THE END OF BLOCK BRANCH. THEREFORE, THE FIRST END OF BLOCK BRANCH WILL NOT HAVE ANY POINTER UPDATING TO PERFORM.

THE HOUSE KEEPING NEEDED TO WRAPUP A BUFFER CONSISTS OF THE FOLLOWING ACTIONS. ANY TRAILING BLANKS THAT ARE NEEDED ARE ADDED TO THE END OF THE BUFFER. THE CHARACTER COUNT FOR THE BUFFER IS UPDATED. THEN, THE BUFFER IS LINKED INTO THE PREVIOUS BLOCKS OF THE MESSAGE.

THE REST OF THE END OF BLOCK CODE IS ALWAYS PERFORMED.

NEXT, A GET SPACE IS DONE TO OBTAIN A BUFFER FOR THE NEXT PORTION OF THE MESSAGE. SHOULD NO SPACE BE AVAILABLE, ANY SPACE ALREADY USED BY THE MESSAGE WILL BE RETURNED. THEN, A UNIT NOT READY MESSAGE WILL BE QUEUED, AND THE NEXT ACTIVITY WILL BE SET TO ACKNOWLEDGE. AFTER THESE OPERATIONS HAVE BEEN PERFORMED, INTERRUPTS WILL BE IGNORED.

IF THE GET SPACE IS SUCESSFUL, MESSAGE POINTERS AND COUNTERS ARE INITIALIZED. THEN, IF APPLICABLE, ANY CARRIAGE CONTROL CHARACTERS ARE HANDLED BEFORE CONTROL IS GIVEN TO THE MESSAGE STORAGE CODE.

STORAGE OF DATA

THE CODE WHICH ACTUALLY STORES A MESSAGE IS RATHER SHORT. IT STORES THE CHARACTER THEN MAKES A NUMBER OF SIMPLE TESTS. IF THE CHARACTER IS A CARRIAGE RETURN, END OF BLOCK ACTION TAKES PLACE. IF IT IS AN ETX, END OF MESSAGE CODE WILL BE EXECUTED. IF BOTH OF THE ABOVE TESTS FAIL, POINTERS ARE UPDATED. THEN THE NUMBER OF CHARACTERS STORED SO FAR IN THIS BUFFER IS TESTED AGAINST THE MAXIMUM NUMBER OF CHARACTERS ALLOWABLE FOR THE DEVICE TO WHICH THIS MESSAGE BELONGS. IF THE NUMBER OF CHARACTERS RECEIVED SO FAR IS LESS THAN THE MAXIMUM, A SLEEP IS PERFORMED WITH THE ADDRESS OF THE CHARACTER STORAGE CODE AS THE AWAKE ADDRESS.

IF THE NUMBER OF CHARACTERS STORED IS GREATER THAN THE DEVICE MAXIMUN, THE INCOMING CHARACTERS WILL BE CHECKED FOR END OF BLOCK AND END OF MESSAGE, BUT, THE STORAGE OF DATA WILL BE SUPPPRESSED.

END OF MESSAGE

END OF MESSAGE ACTION IS IN SOME WAYS SIMILAR TO THE END OF BLOCK ACTION. ANY TRAILING BLANKS NEEDED TO FILL A BUFFER ARE PROVIDED, THEN THE CHARACTER COUNT OF THE BLOCK IS SET. THE BUFFER IS THEN LINKED TO ANY PRECEEDING PARTS OF THE MESSAGE. THEN THE DATACOM PROCEDURES SLEEP AWAITING THE BCC.

UPON AWAKING AND RECEIVING THE BCC, A CHECK IS MADE TO SEE IF THE RECEIVED BCC IS EQUAL TO THE CALCULATED BCC. FAILURE OF THIS TEST IS HANDLED AS A FORMAT ERROR.

WHEN THE BCC TEST HAS BEEN PASSED, ONE HAS DETERMINED THAT THE MESSAGE WAS RECEIVED WITHOUT ANY DETECTABLE ERRORS. SO, FINAL WRAPUP OPERATIONS ARE STARTED. THE RECEIVE TRANSMISSION NUMBER IS INCREMENTED. THEN A CHECK OF THE DEVICE QUEUE IS MADE TO DETERMINE IF THE CONTENTS OF THE QUEUE PLUS THE MESSAGE AMOUNTS TO MORE MESSAGES THAN ARE ALLOWED FOR THE DEVICE.

IF THERE WILL NOT BE TOO MANY MESSAGES, THE MESSAGE IS QUEUED INTO THE DEVICE QUEUE. THE NEXT ACTIVITY IS SET TO ACKNOWLEDGE. AND, AN IGNORE INTERRUPTS SLEEP IS PERFORMED.

BUT, IF THE QUEUE WOULD BE TOO LARGE, THE MESSAGE IS FORGOTTEN INSTEAD OF BEING QUEUED. THEN, A DEVICE NOT READY MESSAGE IS QUEUED PRIOR TO SETING NEXT ACTIVITY TO ACKNOWLEDGE.

NOTES ON DATACOM OPERATIONS

EACH ARRIVING CHARACTER IS CHECKED TO SEE THAT IT HAS ODD PARITY. EVEN PARITY IS TREATED AS A FORMAT ERROR.

A FORMAT ERROR IS HANDLED IN THE FOLLOWING MANNER. NEXT ACTIVITY IS SET SO THAT A NAK WILL BE SENT IN RESPONSE TO THE MESSAGE. ANY SPACE OCCUPIED BY THE MESSAGE WILL BE RETURNED. AND, FINALLY, AN IGNORE INTERRUPTS SLEEP WILL BE PERFORMED.

THE ABOVE CODE IS RESPONSIBLE FOR CONTROLLING THE ACTION OF THE LINE. BUT TO FULLY UNDERSTAND HOW CONTROL IS PASSED FROM ROUTINE TO

ROUTINE ONE NEEDS TO UNDERSTAND NOT ONLY THE SLEEP, BUT THE OPERATION OF THE DATACOM INTERRUPT CODE. CONTROL IS PASSED FROM ONE PIECE OF RECEIVE CODE TO ANOTHER, OR FROM ONE PIECE OF TRANSMIT CODE TO ANOTHER BY MEANS OF A SLEEP. BUT, THIS MECHANISM WILL NOT WORK FOR PASSING CONTROL FROM TRANSMIT TO RECEIVE OR THE REVERSE DIRECTION. THIS FORM OF CONTROL SWITCHING IS ACCOMPLISHED BY THE DATACOM INTERRUPT CODE. WHEN A CHANGE OF STATUS INTERRUPT OCCURS, THE INTERRUPT CODE WILL USUALLY LOOK AT LAST ACTIVITY, MODEM STATUS AND NEXT ACTIVITY. WHEN, EACH OF THESE VARIABLES WILL BE UPDATED IN ACCORDANCE WITH THE STATUS CHANGE. IF NEEDED, AN ERROR MESSAGE MAY BE PRINTED. AND, SHOULD THE STATUS CHANGE BE CORRECT, A NEW ROUTINE MAY BE INITIATED. UNDER NORMAL CONDITIONS, THE CODE FOR HANDLING THE CARRIER INTERRUPT DOES MOST OF THE STATUS CHANGING.

B.8.8 MISCELLANEOUS PROCEDURES

THE ABOVE DISCUSSION COVERS ALL BUT A HANDFULL OF GENERALL ROUTINES. THESE ROUTINES ARE COVERED BELOW.

GET SPACE

GET SPACE IS CALLED TO GET A BUFFER. IT RETURNS IN THE C REGISTER THE ADDRESS OF A BUFFER. IF NO SPACE IS AVAILABLE, A VALUE OF ZERO IS RETURNED. PRESISION AT ENTRY IS UNIMPORTANT. PRECISION ON EXIT IS TWO. IF THE OVERFLOW FLIP-FLOP IS TRUE ON ENTRY, A PRINTER BUFFER WILL BE RETURNED, OTHER WISE, A BUFFER SUITABLE FOR THE SHORTER DEVICES WILL BE RETURNED.

CONTROL GET SPACE

THE SAME AS GET SPACE EXCEPT THAT IT IS CALLED FROM CONTROL STATE.

FORGET SPACE

FORGETS THE BUFFER WHOSE ADDRESS IS IN THE C REGISTER. AS WITH GET SPACE, THE OVERFLOW FLIP-FLOP SPECIFIES LONG OR SHORT BUFFER.

CONTROL FORGET SPACE

THIS ROUTINE FORGETS A BUFFER OR A LIST OF BUFFERS WHOSE ADDRESS IS

IN THE C REGISTER. THE OVERFLOW FLIP-FLOP HAS THE SAME MEANING AS ABOVE. THIS ROUTINE SHOULD BE CALLED ONLY FROM CONTROL STATE.

FIND A MESSAGE

IT IS THE JOB OF FIND A MESSAGE TO FIND AND DELINK BUFFERS FROM DEVICE QUEUES. THE ADDRESS OF A BUFFER IS RETURNED IN THE C REGISTER. IF NO BUFFERS ARE AVAILABLE, A ZERO IS RETURNED.

QUITE A BIT OF LOGIC IS INVOLVED IN THE ABOVE OPERATION. IF THERE IS NO CURRENT UNIT, AN ATTEMPT WILL BE MADE TO FIND A UNIT. IF NO UNIT IS FOUND TO HAVE TRAFFIC FOR THE SYSTEM A ZERO IS RETURNED. OTHERWISE, THE CURRENT UNIT IS SET TO THE UNIT FOUND TO HAVE DATA FOR THE SYSTEM.

ONCE THE UNIT HAS BEEN FOUND, OR IF THERE IS A CURRENT UNIT, THE FOLLOWING ACTION TAKES PLACE. THE QUEUE FOR THE CURRENT UNIT IS EXAMINED. IF IT IS EMPTY A ZERO IS RETURNED.

WHEN A BUFFER IS FOUND, THE NUMBER OF CHARACTERS IN THE MESSAGE PLUS THE NUMBER OF CHARACTERS IN THE BUFFER IS CHECKED TO SEE IF THEIR SUM IS GREATER THAN 400. FOUR HUNDRED CHARACTERS IS THE MAXIMUM MESSAGE SIZE ALLOWABLE. IF 400 CHARACTERS WOULD BE EXCEEDED BY THE NEW BLOCK A ZERO IS RETURNED, OTHERWISE, THE ADDRESS OF THE BUFFER IS RETURNED.

LINK BACK

LINK BACK LINKS A MESSAGE INTO THE HEAD OF THE QUEUE OF THE CURRENT UNIT.

MAKE NOT READY

MAKE NOT READY STORES A DEVICE NOT READY MESSAGE AND SETS THE NRMASK BIT FOR A DEVICE. THE C REGISTER CONTAINS THE BINARY VALUE OF THE UNIT DESIGNATE FOR THE UNIT IN QUESTION. PRECISION MUST BE TWO ON ENTRY.

MAKE READY

THIS PROCEDURE QUEUES A DEVICE READY MESSAGE AND RESETS THE DEVICE"S BIT IN THE NRMASK. THE CALLING PROCEDURE IS THE SAME AS FOR MAKE NOT READY.

B.8.9 MISCELLANEOUS DC1000 SOFTWARE

LINE PRINTER DUMP

THE DUMP ROUTINE IS A PROGRAM CONTAINED ON A SMALL CARD DECK. WHEN LOADED, IT WILL DUMP THE CONTENTS OF THE MEMORY OF THE DC1000 TO THE LINE PRINTER. IT WILL OVERLAY PART OF THE LITERAL TABLE OF THE EXECUTIVE AND PART OF THE CODE OF THE EXECUTIVE, BUT IT WILL LEAVE ALL DATA AREAS AND CRITICAL SUBROUTINES INTACT. THIS IS INCLUDED AS AN AID IN THE DEBUGGING OF THE RJE SYSTEM.

TWX DUMP

THIS DUMP ROUTINE IS A RATHER BASIC ONE. IT IS A BINARY PAPER TAPE IN BOOTSTRAP FORM. WHEN IT HAS BEEN BOOTSTRAPPED INTO THE HIGHEST MODULE OF MEMORY IT WILL STOP. ENTER INTO THE X REGISTER THE ADDRESS FROM WHICH THE DUMP IS TO BE STARTED. THEN PRESS THE RUN SWITCH. A CORE DUMP WILL BE PRODUCED ON THE SPO.

TABLE OF CONTENTS