B6700 SOFTWARE IMPROVEMENTS
MARK II.3
2 OCTOBER 1972

SYSTEMS PROGRAMMING ACTIVITY
LARGE SYSTEMS PLANT
LARGE COMPUTER SYSTEMS DIVISION
BURROUGHS CORPORATION
INDUSTRY, CALIFORNIA

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

NEW FEATURES AND DOCUMENTATION CHANGES

TABLE OF CONTENTS

NEW FEATURES AND DOCUMENTATION CHANGES

NEW FEATURES AND DOCUMENTATION CHANGES

NEW FEATURES AND DOCUMENTATION CHANGES

NEW FEATURES AND DOCUMENTATION CHANGES

D0001 SOURCE LINE IDENTIFICATION - 06-20-72

THE COMPILERS HAVE BEEN MODIFIED TO PROVIDE LINE OR SEQUENCE NUMBER INFORMATION FOR DIAGNOSTIC PURPOSES. WHEN A PROGRAM IS DISCONTINUED FOR ANY REASON OR WHEN A PROGRAM DUMP IS OBTAINED, IT IS NOW POSSIBLE TO RECEIVE SOURCE LINE IDENTIFICATION OF THE POINT OF THE PROGRAM WHERE THE ERROR OCCURRED. THIS FACILITY IS ALSO PROVIDED WITH THE "MONITOR" AND "DUMP" STATEMENTS.

TO OBTAIN SOURCE LINE IDENTIFICATION INFORMATION, THE DOLLAR CARD OPTION "LINEINFO" HAS BEEN ADDED TO ALL COMPILERS. THIS SAVES SEQUENCE OR LINE NUMBER INFORMATION AT COMPILE TIME AND ITS RELATION TO THE CODE EMITTED AT COMPILE TIME. IF A PROGRAM SHOULD THEN TERMINATE ABNORMALLY OR MONITORING IS OCCURRING, THE LINE INFORMATION WILL BE DISPLAYED OR PRINTED, AS APPROPRIATE.

THE "LINEINFO" INFORMATION MAY REQUIRE A SIGNIFICANT ADDITIONAL AMOUNT OF DISK STORAGE IN THE CODE FILE OF A COMPILED PROGRAM. FOR THIS REASON, IT MAY NOT BE DESIRABLE TO SET THE OPTION FOR DEBUGGED PROGRAMS.

"LINEINFO" IS AUTOMATICALLY SET FOR ALL COMPILATIONS ORDERED THROUGH CANDE. WHEN BINDING, THE DOLLAR OPTION "LINEINFO" IS SET BY DEFAULT SO THAT LINE INFORMATION IS PRESERVED. PROGRAM UNITS WITH AND WITHOUT LINE INFO MAY BE COMBINED. RESETTING "LINEINFO" WILL CAUSE THE BINDER TO DISCARD ALL LINE INFORMATION. THIS MAY BE DONE WHETHER OR NOT ANY BINDING IS DONE, EVEN IF THE HOST IS A COMPLETE PROGRAM.

D0002 FREE FORM INPUT - 06-16-72

FREE FORM INPUT TO THE FORTRAN AND COBOL COMPILERS HAS BEEN IMPLEMENTED. THE PURPOSE OF THIS IMPLEMENTATION IS TO ALLOW SIMPLE INPUT FROM REMOTE DEVICES. THE USE OF THIS OPTION IS GOVERNED BY

THE DOLLAR CARD OPTIONS "FREE" AND "FREETAPE".

FREE OPTION

THIS OPTION WHEN SET CAUSES THE COMPILER TO ACCEPT FREEFORM INPUT FROM THE SOURCE FILE CALLED "CARD". THIS OPTION IS INITIALLY SET FOR ALL JOBS INITIATED THROUGH CANDE, AND RESET OTHERWISE.

FREETAPE OPTION (FORTRAN ONLY)

THIS OPTION WHEN SET CAUSES THE COMPILER TO ACCEPT FREEFORM INPUT FROM THE "TAPE" FILE. THIS OPTION IS INITIALLY RESET.

IF "NEW" IS SET AND EITHER "FREE" OR "FREETAPE" IS SET, THE COMPILER WILL PERFORM A MINOR AMOUNT OF EDITING ON THE OUTPUT "NEWTAPE" FILE, TO ENSURE THAT IT IS ACCEPTABLE AS EITHER FREE OR FIXED FORM INPUT.

FORTRAN FREE FORM EDITING

THE EDITING CONSISTS OF ENSURING THAT ALL CONTINUATION CARDS USE EITHER THE ASTERISK (*) OR DASH (-) IN COLUMN SIX TO DENOTE CONTINUATION AND THAT THE SECOND COLUMN OF ALL COMMENT CARDS IS EITHER AN ASTERISK OR A DASH.

FORTRAN FREE FORM FORMAT

THE FOLLOWING RULES ARE APPLIED TO INPUT CARD IMAGES FOR EDITING:

COMMENTS:

COMMENTS ARE RECOGNIZED BY A "C" IN COLUMN ONE, AND A DASH (-) OR ASTERISK (*) IN COLUMN TWO.

CONTINUATION CARDS:

A CONTINUATION CARD IS DENOTED BY THE OCCURRENCE OF A DASH OR ASTERISK AS THE FIRST NON-BLANK CHARACTER IN THE FIRST SIX CARD COLUMNS. THE CHARACTERS IMMEDIATELY FOLLOWING THE ASTERISK OR DASH ARE INSERTED BEGINNING IN COLUMN SEVEN. NOTE THAT IT IS POSSIBLE TO HAVE MORE THAN 66 NON-BLANK CHARACTERS BETWEEN THE

CONTINUATION CHARACTER AND COLUMN 73. NO DEBLANKING IS PERFORMED WHEN THE CARD IMAGE IS SHIFTED. ANY CHARACTERS BEYOND THE 66TH CHARACTER BUT BEFORE COLUMN 73 WILL BE DELETED. IF THESE CHARACTERS ARE NOT BLANK, A WARNING MESSAGE WILL BE ISSUED.

FORTRAN STATEMENTS WITH STATEMENT NUMBERS

IMPLEMENTED IN ALGOL AND ESPOL AS A RELATIONAL OPERATOR.

STATEMENTS BEGINNING WITH STATEMENT NUMBERS ARE PROCESSED BY ISOLATING THE STATEMENT NUMBER AND INSERTING IT IN COLUMNS ONE THROUGH FIVE, INSERTING A BLANK IN COLUMN SIX, AND INSERTING THE FIRST CHARACTER FOLLOWING THE STATEMENT NUMBER INTO COLUMN APPEARS BEFORE THE PATCH NUMBER IS ON THE RIGHT SEVEN.

AS WITH CONTINUATION CARDS, TRUNCATION MAY OCCUR. IF IT DOES, IT IS FLAGGED.

THE ISOLATION OF THE STATEMENT NUMBER IS DONE BY FINDING THE FIRST DIGIT OF THE STATEMENT NUMBER (WHICH MUST LIE WITHIN COLUMNS ONE THROUGH SIX OF THE FREE FORM CARD IMAGE). THE NEXT CHARACTER IS EXAMINED. IF IT IS BLANK OR A DIGIT, IT IS INCLUDED IN THE STATEMENT NUMBER. THIS PROCESS IS REPEATED SUCH THAT NO MORE THAN FIVE CHARACTERS ARE EXAMINED. THE EXAMINATION TERMINATES WHEN A NON-DIGIT, NON-BLANK CHARACTER IS ENCOUNTERED. FOR EXAMPLE:

```
COLUMNS:  123456789
CARD      ------1....     NOT RECOGNIZED AS STATEMENT NO.
CARD      --1-1A.....     RECOGNIZED AS: 11----A
CARD      1-1A.......     SAME AS ABOVE
CARD      1-1-11A....     RECOGNIZED AS: 111--1A
```

NOTE: THE DASHES REPRESENT BLANKS.

FORTRAN DOLLAR CARDS

A DOLLAR SIGN ($) IN COLUMN ONE OR A DOLLAR SIGN IN COLUMN TWO WITH A BLANK IN COLUMN ONE CAUSES THE COMPILER TO INTERPRET THIS CARD IMAGE AS A COMPILER OPTION CARD.

FILE AND DEBUG STATEMENT

THESE STATEMENTS ARE RECOGNIZED BY THE STRING "FILE " OR "DEBUG " IN COLUMNS ONE THROUGH SIX.

ALL OTHER CARDS ARE ADJUSTED SO THAT COLUMNS ONE THROUGH SIX ARE BLANK, AND INTERPRETED AS STATEMENTS.

COBOL FREE FORM

THE "FREE" OPTION IN COBOL ALLOWS ALL INPUT TO BEGIN IN COLUMN SEVEN OF THE CARD OR TAPE IMAGE. A HYPHEN IN COLUMN SEVEN INDICATES CONTINUATION, WHILE A $ IN COLUMN SEVEN INDICATES A DOLLAR CARD.

TO PUT A COMMENT IN THE SYMBOLIC FILE, AN ASTERISK (*) OR A SLASH (/) INDICATES THAT THE REST OF THE CARD IS A COMMENT, PROVIDED THAT IT APPEARS IN COLUMN SEVEN.

ALL PARAGRAPH, SECTION, AND DIVISION HEADINGS MUST BEGIN IN COLUMNS 7-11 INCLUSIVE. ALL OTHER SOURCE IMAGES MAY BEGIN AT ANY POSITION IN THE RECORD. UNDER THIS OPTION, COLUMNS 73 - 80 ARE STILL TREATED AS COMMENTS.

USE WITH CANDE

THE USER SHOULD BE AWARE OF THE CONVENTIONS FOLLOWED BY CANDE IN MAPPING THE INPUT LINE INTO A RECORD. FOR COBOL SYMBOLIC FILES, THE SEQUENCE NUMBER IS RIGHT-JUSTIFIED IN COLUMNS ONE THROUGH SIX; THE TEXT FIELD (BEGINNING WITH THE FIRST NON-NUMERIC CHARACTER OR THE SEVENTH CHARACTER, WHICH EVER OCCURS FIRST), IS PLACED IN THE RECORD BEGINNING AT COLUMN SEVEN. FOR FORTRAN, THE SEQUENCE NUMBER IS RIGHT-JUSTIFIED IN COLUMNS 73 THROUGH 80; THE TEXT FIELD (BEGINNING WITH THE FIRST NON-NUMERIC CHARACTER, OR THE EIGHTH CHARACTER, WHICHEVER OCCURS FIRST), IS PLACED IN THE RECORD BEGINNING AT COLUMN ONE. THUS, THE FIRST TEXT CHARACTER TYPED BY THE CANDE USER IS THE FIRST CHARACTER EXAMINED BY EACH COMPILER TO ANALYZE FREEFORM RECORDS.

D0003 ASCII CAPABILITY - 08-15-72

THE CAPABILITY OF HANDLING DATA AND SOURCE PROGRAMS IN ASCII HAS BEEN ADDED TO THE SOFTWARE. IN GENERAL, ALL COMPILERS (EXCEPTING ESPOL AND XALGOL) WILL ACCEPT SOURCE PROGRAMS IN ASCII. BY USE OF THE MCP SOFT TRANSLATION FEATURE, DATA FILES CAN BE ACCEPTED BY SETTING FILETYPE TO EIGHT ON LABEL EQUATION CARDS. SOME SPECIFIC CHANGES TO INDIVIDUAL COMPILERS FOLLOW.

ALGOL

FOR ALGOL, A NEW TYPE OF ASCII HAS BEEN IMPLEMENTED. CHARACTER ARRAYS MAY BE DECLARED TO BE OF ASCII TYPE. POINTERS BECOME ASCII POINTERS BY GIVING THEM A LENGTH ATTRIBUTE OF SEVEN (ALTHOUGH EACH ASCII CHARACTER STILL TAKES UP EIGHT BITS).

ASCII MAY BE USED IN THE TRUTHSET AND TRANSLATETABLE CONSTRUCTS TO PERMIT SOFTWARE COMPARISONS AND TRANSLATIONS WITHIN THE ASCII CHARACTER SET. BECAUSE OF HARDWARE LIMITATION, HOWEVER, IT IS NOT PERMITTED TO REPLACE AN ASCII POINTER FOR A SPECIFIED NUMBER OF DIGITS. IN ADDITION, THE INTEGER AND DOUBLE TYPE TRANSFER FUNCTIONS FOR POINTERS ARE NOT AVAILABLE FOR ASCII POINTERS. THESE MAY RESULT IN ERRORS AT EXECUTION TIME.

COBOL

THE IMPLEMENTATION OF ASCII IN COBOL PERMITS INTERNAL USAGE OF ASCII DATA. THIS MAY BE DONE BY ADDING THE DECLARATIVE "USAGE IS ASCII". SUCH USAGES FOR FIELDS DECLARED AT OTHER THAN THE 01 LEVEL MUST BE CONTAINED WITHIN 01 LEVEL FIELDS DECLARED AS ASCII USAGE.

ALL LEGITIMATE NON-MIXED MODE COMPARISONS WILL WORK CORRECTLY. IN ADDITION, CERTAIN MIXED MODE COMPARISONS HAVE BEEN IMPLEMENTED. ASCII-BCL COMPARISONS WILL USE THE ASCII COLLATING SEQUENCE FOR COMPARISONS, WHILE ASCII-EBCDIC COMPARISONS WILL USE THE EBCDIC COLLATING SEQUENCE. PURE ASCII OPERATIONS WILL OPERATE ACCORDING TO THE ASCII COLLATING SEQUENCE.

NUMERIC AND NUMERIC EDITED ITEMS ARE NOT PERMITTED IN ASCII. A

"USAGE IS ASCII" FIELD MAY NOT HAVE SUBORDINATE COMP, COMP-2, OR DISPLAY-1 ITEMS.

FILES FOR WHICH THE FIRST RECORDAREA IS DECLARED AS "USAGE IS ASCII" WILL BE ASSIGNED AN INTERNAL MODE OF ASCII. THESE FILES WILL, THEREFORE, RETURN ASCII DATA.

FORTRAN

THE STRING DELIMITING CHARACTER IN ASCII FORTRAN SOURCE DECKS REMAINS THE APOSTROPHE. TWO APOSTROPHES MUST APPEAR ADJACENT TO EACH OTHER INSIDE A STRING TO INDICATE ONE APOSTROPHE IN THE RESULTING DATUM. ALL COMPARISONS ARE BASED UPON THE EBCDIC COLLATING SEQUENCE.

BASIC

BASIC WILL CONTINUE TO RELY UPON EBCDIC COLLATING SEQUENCE. ALL STRING COMPARISONS WILL BE BASED ON THIS SEQUENCE. FOR INPUT DATA IN ASCII, LABEL EQUATION AS DESCRIBED AT THE BEGINNING OF THIS NOTE MAY BE REQUIRED.

D0004 COMPILE TIME BINDING - 08-14-72

AUTOBINDING IS A COMPILER OPTION WHICH COMBINES THE PROCESSES OF COMPILING AND PROGRAM BINDING INTO ONE JOB. DURING COMPILATION, THE COMPILER PRODUCES A SET OF INSTRUCTIONS TO BE PASSED TO THE BINDER. IN MANY CASES, THESE BINDER INSTRUCTIONS ARE SELF-SUFFICIENT FOR BINDING PURPOSES AND THE USER NEED NOT BE CONCERNED WITH BINDER CONTROL CARDS. IN THOSE CASES WHERE BINDER INSTRUCTIONS ARE REQUIRED, THE USER MAY INSERT HIS OWN BINDER CONTROL CARDS.

AUTOBIND IS CURRENTLY IMPLEMENTED IN ALGOL AND FORTRAN. THE AUTOBINDING FEATURE OF THE COMPILER IS INVOKED BY THE DOLLAR OPTION "AUTOBIND". THE AUTOBIND OPTION MAY BE SET OR RESET AT ANY POINT THROUGHOUT COMPILATION. HOWEVER, IT IT RECOMMENDED THAT IT BE SET OR RESET ONLY ONCE AT THE BEGINNING OF COMPILATION FOR THE

FOLLOWING TWO REASONS:

1. ONLY THE STATUS OF AUTOBIND AT THE END OF COMPILATION IS SIGNIFICANT. SPECIFICALLY, IF FOUR PROCEDURES ARE BEING COMPILED, THE FIRST THREE WITH AUTOBIND RESET AND THE LAST ONE WITH AUTOBIND SET, THE BINDER WILL STILL ATTEMPT TO BIND ALL FOUR PROCEDURES TO THE SPECIFIED HOST.

2. COMPILE AND GO ON A SEPARATE PROCEDURE WITH AUTOBIND RESET, WILL RESULT IN THE MCP ATTEMPTING TO EXECUTE THE PROCEDURE AS SOON AS IT IS COMPILED. THIS MAY RESULT IN ERRORS AS THE PROCEDURE HAS NOT YET BEEN BOUND INTO A HOST. IF AUTOBIND IS SET THROUGHOUT COMPILATION, EXECUTION OF THE RESULTANT PROGRAM WILL TAKE PLACE AFTER BINDING.

SPECIFYING A HOST FILE:

IN ALGOL A SEPARATE PROCEDURE COMPILED AT LEVEL TWO OR AN OUTER BLOCK MAY SERVE AS A HOST FOR BINDING. IN FORTRAN, A MAIN PROGRAM MAY SERVE AS A HOST. SEPARATE ALGOL PROCEDURES COMPILED AT LEVEL THREEE (DEFAULT LEVEL) OR FORTRAN SUBPROGRAMS MAY BE BOUND INTO A HOST. AT MOST, ONE HOST MAY BE COMPILED IN A JOB ALONG WITH ANY NUMBER OF SEPARATE PROCEDURES OR SUBROUTINES. IN ALGOL, THE HOST MUST BE THE LAST PROGRAM UNIT COMPILED. IF AN APPROPRIATE HOST FILE IS COMPILED WITH AUTOBIND SET, IT IS ASSUMED TO BE THE HOST FOR BINDING. THIS ASSUMPTION CANNOT BE OVERRIDDEN BY EITHER OF THE METHODS GIVEN NEXT FOR SPECIFYING A HOST.

IF NO ELIGIBLE HOST IS BEING COMPILED, A HOST MUST BE SPECIFIED. TWO METHODS ARE AVAILABLE:

<I> COMPILER FILE HOST (TITLE = FILEID1/----/FILEIDN)

OR A BINDER HOST CARD, SUCH AS

$ HOST IS FILEID1/----/FILEIDN;

DISPOSITION OF PROCEDURE FILES:

THE CODE FILE OF ANY LEVEL THREE PROCEDURE COMPILED WITH AUTOBIND SET IS MARKED AS (NON-EXECUTABLE). IF NOT EXECUTED VIA INTER-PROGRAM COMMUNICATION, THE PROCEDURE MUST BE BOUND INTO A HOST FILE BY THE BINDER BEFORE BEING EXECUTED.

CODE FILES OF ANY LEVEL THREE PROCEDURES (OR HIGHER) COMPILED ARE PURGED AFTER BEING BOUND INTO A HOST BY AUTOBIND. TO RETAIN SUCH AS CODE FILE, IT IS NECESSARY TO REFER TO IT SPECIFICALLY IN A BINDER CONTROL STATEMENT. EITHER OF THE FOLLOWING STATEMENTS WILL ALLOW THE PROCEDURES CODE FILE TO REMAIN:

```
$ BIND PROCEDURENAME

        OR

$ EXTERNAL PROCEDURENAME
```

THE FIRST STATEMENT PERFORMS THE SAME FUNCTION AS THE DEFAULT COMPILER-GENERATED BIND STATEMENT, EXCEPT THE CODE FILE WILL NOT BE PURGED. THE SECOND STATEMENT INSTRUCTS THE BINDER NOT TO BIND THE PROCEDURE INTO THE HOST EVEN THOUGH IT HAS BEEN COMPILED WITH AUTOBIND SET AND THERE IS AN EXTERNAL REFERENCE TO IT IN THE HOST.

COMPILER GENERATED BINDER STATEMENTS:

GIVEN THE FOLLOWING COMPILE CARD:

```
<I>COMPILE FILEID1/FILEID2-----/FILEIDN ALGOL LIBRARY
```

THE COMPILER WILL PLACE ANY SEPARATELY COMPILED LEVEL THREE PROCEDURES INTO A FILE WITH THE NAME FILEID1/FILEID2-----/FILEIDN-1/ PROCEDURENAME. THE COMPILER REPLACES THE FILEID FOLLOWING THE LAST SLASH BY THE COMPILED PROCEDURE IDENTIFIER TO CREATE A FILE NAME FOR THE COMPILED PROCEDURE CODE. THE COMPILER THEN ISSUES THE FOLLOWING UNIVERSAL BIND STATEMENT TO THE BINDER IF AUTOBIND IS SET:

```
CBIND = FROM FILEID1/FILEID2----/FILEIDN-1/ = ;
```

PLUS ONE INSTRUCTION OF THE FORM:

```
CBIND PROCEDURENAME;
```

FOR EACH SEPARATE LEVEL THREE PROCEDURE COMPILED.

THE CBIND STATEMENT IS EQUIVALENT TO THE BIND STATEMENT WHICH THE USER INPUTS TO THE BINDER. ANY CBIND STATEMENT CONFLICTING WITH A USERS BINDER CONTROL STATEMENT IS OVERRIDDEN BY THE USERS STATEMENT.

THE FIRST INSTRUCTION INDICATES TO THE BINDER THAT WHEN A PROCEDURE IS TO BE BOUND INTO THE HOST, IT SHOULD LOOK FOR THAT PROCEDURE IN A FILE NAMED FILEID1/FILEID2.../FILEIDN-1/PROCEDURENAME. THE SECOND INSTRUCTION, ONE FOR EACH SEPARATELY COMPILED PROCEDURE, INSTRUCTS THE BINDER TO BIND THE PROCEDURE INTO THE HOST IN THE FOLLOWING MANNER:

1. IF AN EXTERNAL REFERENCE EXISTS FOR THIS PROCEDURE, THE EXTERNAL REFERENCE WILL BE RESOLVED BY BINDING IN THE PROCEDURE.

2. IF A REPLACEABLE PROCEDURE (ANY LEVEL 3 PROCEDURE WHICH CONTAINS LOCAL DECLARATIONS IS REPLACEABLE) ALREADY EXISTS IN THE HOST, THE NEWLY COMPILED PROCEDURE WILL REPLACE IT.

3. IF NEITHER OF THE ABOVE TWO CONDITIONS HOLD, THE PROCEDURE WILL SIMPLY BE ADDED. PRESUMABLY, THE USER MIGHT, IN THE FUTURE, REPLACE A REPLACEABLE PROCEDURE WITH A PROCEDURE WHICH WILL REFERENCE THE ADDED PROCEDURE.

IF THE FILENAME ON THE COMPILE CARD CONTAINS NO SLASHES, THEN THE COMPILER WILL NAME ITS CODE FILE "PROCEDURENAME" AND WILL NOT ISSUE A UNIVERSAL BIND STATEMENT, AS IT IS NOT REQUIRED.

PASSING CONTROL INFORMATION TO THE BINDER:

TWO TYPES OF CONTROL INFORMATION MAY BE PASSED TO THE BINDER WHEN USING THE AUTOBINDING FEATURE OF A COMPILER. BINDER OPTION CARDS (I.E., DOLLAR CARDS) ARE INPUT BY PUNCHING "$ BINDER" FOLLOWED BY THE LIST OF OPTIONS TO BE RESET OR SET. OTHER THAN THE WORD

"BINDER" FOLLOWING THE DOLLAR SIGN, THE CARD WOULD BE WRITTEN EXACTLY AS IF IT WERE A DOLLAR CARD INPUT DIRECTLY TO THE BINDER.

BINDER INPUT STATEMENTS ARE PASSED TO THE BINDER BY PLACING A DOLLAR SIGN BEFORE THEM. OTHER THAN THE DOLLAR SIGN, THE INPUT STATEMENT WOULD BE WRITTEN EXACTLY AS IF IT WERE INPUT DIRECTLY TO THE BINDER, INCLUDING A SEMICOLON AT THE TERMINATION OF THE STATEMENT.

SEE THE PROGRAM BINDER MANUAL FOR A DESCRIPTION OF BINDER OPTIONS AND INPUT STATEMENTS.

NOTE THAT IN ALGOL, IF THE WORDS "BINDER", "USE", "BIND", "HOST", "EXTERNAL", OR "PURGE" ARE MISSPELLED WHEN INPUTTING BINDER STATEMENTS, THE CARD WILL BE INTERPRETED AS USER OPTIONS TO BE SET. IN FORTRAN, MISSPELLINGS RESULT IN A SYNTAX ERROR.

EXAMPLE:

THE FOLLOWING EXAMPLE INDICATES USAGE OF BINDER INSTRUCTIONS, BUT ALSO INDICATES HOW OVERRIDING THE COMPILERS UNIVERSAL BIND STATEMENT WITH ANOTHER CAN LEAD TO PROBLEMS.

THE FOLLOWING DECK IS USED TO COMPILE SEVERAL SEPARATE PROCEDURES AND BIND THEM INTO A HOST FILE NAMED HOST/FILE. THE HOST FILE CONTAINS SEVERAL OTHER EXTERNAL PROCEDURES WHICH ARE NOT BEING COMPILED IN THE PRESENT JOB. IT IS DESIRED TO HAVE THE BINDER RESOLVE THESE EXTERNAL REFERENCES BY LOOKING IN THE PREVIOUSLY COMPILED CODE FILES A/=, B/=, AND C/= WHERE THE PROCEDURENAME SOUGHT FOR BINDING PURPOSES IS SUBSTITUTED FOR THE "=" IN EACH CASE TO INDENTIFY THE CORRECT FILE.

```
<I>COMPILE BAD/EXAMPLE ALGOL LIBRARY GO
<I>ALGOL FILE HOST (TITLE = HOST/FILE)
<I>DATA
$ SET AUTOBIND
$ BINDER SET LIST STACK
$ BIND = FROM A/=, B/=, C/=
.
```

```
.
PROCEDURE P1
BEGIN
.
.
END;
PROCEDURE P2
BEGIN
.
.
END.
<I>END.
```

THE COMPILER PRODUCED TWO CODE FILES DURING COMPILATION; BAD/P1 AND BAD/P2 CONTAINING THE CODE FOR THE TWO SEPARATE PROCEDURES. THE COMPILER ALSO GENERATED THE FOLLOWING BINDER STATEMENTS:

```
CBIND = FROM BAD/=;
CBIND P1;
CBIND P2;
```

THUS, THE BINDER COULD FIND THE TWO PROCEDURES IN THE APPROPRIATE TWO FILES JUST COMPILED. HOWEVER, THE USERS BIND CARD HAS OVERRIDDEN THE COMPILERS UNIVERSAL BIND STATEMENT. THEREFORE, THE BINDER WILL NOT LOOK FOR BAD/P1 OR BAD/P2; RATHER IT LOOKS FOR A/P1, B/P1, C/P1, AND SO ON, BUT WILL NEVER FIND THE JUST COMPILED PROCEDURES. THE USER, IF HE WISHED THE JUST COMPILED PROCEDURES TO BE BOUND, SHOULD HAVE USED

```
$ BIND = FROM BAD/=, A/=, B/=, C/=;
```

SO THAT THE BINDER COULD FIND THE JUST COMPILED PROCEDURES.

EXAMPLE 2:

THE FOLLOWING INPUT DECK IS USED TO COMPILE A MAIN PROGRAM NAMED "MAIN" AND A SUBROUTINE NAMED "SUBR", AND REPLACE AN EXISTING SUBROUTINE NAMED "OLDSUBR" WITH THE NEWLY COMPILED SUBR. THE PROGRAM IS THEN EXECUTED. THE PARAMETERS OF SUBR AND OLDSUBR MUST MATCH IN NUMBER AND TYPE. IF THE SUBROUTINE HAD THE SAME NAME, THE

"$ USE" CARD WOULD NOT BE NECESSARY:

```
<I>COMPILE MAIN/PROG FORTRAN LIBRARY GO
<I>DATA
$ SET SEPARATE
$ SET AUTOBIND
$ USE OLDSUBR FOR SUBR;
    MAIN PROGRAM
        .
        .
        .
    SUBROUTINE SUBR
        .
        .
        .
<I>END.
```

D0005 MONITOR STATEMENT - 06-19-72

THIS STATEMENT HAS BEEN GENERALIZED THROUGHOUT THE COMPILERS. THE MONITOR STATEMENT HAS BEEN IMPLEMENTED FOR THE FIRST TIME IN FORTRAN AND HAS BEEN EXTENDED IN ALGOL. "MONITOR ALL" AND "DUMP ALL" HAVE BEEN ADDED TO COBOL.

MONITOR IN FORTRAN

THE SYNTAX OF THE MONITOR STATEMENT IN FORTRAN IS AS FOLLOWS:

```
<MONITOR STATEMENT>::=DEBUG MONITOR
   (<CONSTANT FILE SPECIFIER>)<MONITOR LIST>

<MONITOR LIST>::=<MONITOR LIST ELEMENT>/
   <MONITOR LIST>,<MONITOR LIST ELEMENT>

<MONITOR LIST ELEMENT>::=<SIMPLE VARIABLE>/
   <ARRAY IDENTIFIER>/<ARRAY IDENTIFIER>
   <SUBSCRIPT LIST>

<SUBSCRIPT LIST>::=<SUBSCRIPT LIST ELEMENT>/
```

<SUBSCRIPT LIST>,<SUBSCRIPT LIST ELEMENT>

<SUBSCRIPT LIST ELEMENT>::= (A LEGAL SUBSCRIPT OF THE ARRAY WHERE EACH INDIVIDUAL SUBSCRIPT IS EITHER A CONSTANT OR A SIMPLE VARIABLE. AN INDIVIDUAL SUBSCRIPT MAY NOT BE AN ARITHMETIC EXPRESSION.)

<CONSTANT FILE SPECIFIER>::= (AN INTEGER LITERAL THAT IS A LEGAL FILE SPECIFIER, I.E., 1-99.)

DEBUG MUST APPEAR IN COLUMNS ONE THROUGH FIVE. MONITOR CAN START IN COLUMNS SEVEN THROUGH 72. THERE CAN BE MORE THAN ONE MONITOR STATEMENT PER PROGRAM UNIT, AND MONITOR STATEMENTS CAN BE CONTINUED THROUGH CONTINUATION CARDS. ARRAYS MUST BE DIMENSIONED PRIOR TO APPEARING IN A MONITOR STATEMENT.

AT EXECUTION TIME, WHENEVER THE VALUE OF A MONITORED VARIABLE IS CHANGED, A RECORD IS WRITTEN ON THE SPECIFIED FILE CONSISTING OF THE PROGRAMS MIX NUMBER, STACK HISTORY, THE MONITORED VARIABLES IDENTIFIER (AND SUBSCRIPT, IF ANY), THE OLD VALUE OF THE VARIABLE, THE NEW VALUE, AND THE NEW VALUE IN HEX. MONITORING ACTION TAKES PLACE FOR A SIMPLE VARIABLE WHOSE VALUE IS CHANGED THROUGH AN ASSIGNMENT STATEMENT, A READ STATEMENT, OR THROUGH BEING PASSED AS AN ACTUAL PARAMETER TO A SUBPROGRAM.

MONITORING ACTION TAKES PLACE FOR ARRAY ELEMENTS ONLY WHEN THEIR VALUE IS CHANGED THROUGH AN ASSIGNMENT STATEMENT. IF A MONITOR LIST ELEMENT IS AN ARRAY IDENTIFIER, THEN EVERY ELEMENT IN THAT ARRAY IS MONITORED. FOR AN ARRAY FOLLOWED BY A SUBSCRIPT LIST, ONLY THE SPECIFIED ELEMENTS ARE MONITORED. ONLY THE FIRST APPEARANCE OF A SIMPLE VARIABLE OR AN ARRAY IDENTIFIER IN A MONITOR LIST IS PROCESSED. ALL SUCCESSIVE APPEARANCES ARE IGNORED, AND A WARNING IS ISSUED.

FORTRAN EXAMPLES:

```
      DIMENSION A(5), B(5), C(5)
DEBUG MONITOR (6) X, A(1), (I), (5), B, Y, C, Z, A
      I=3
      A(3)=1
```

```
B(4)=2
X=3
A(4)=4
READ (5,/) Y, A(1)
R= SUB (Z, C)
STOP
END
SUBROUTINE SUB (T, V)
DIMENSION V (5)
V(1)=4
T=3
RETURN
END
```

IN THIS EXAMPLE, MONITORING ACTION WILL TAKE PLACE FOR A(3), B(4), X, Y, AND Z.

ALGOL AND XALGOL MONITOR EXTENSIONS

A SUBSCRIPTED VARIABLE OR LABEL MAY NOW BE MONITORED TO A FILE IN ALGOL. THE SYNTAX FOR THE MONITOR DECLARATION IS CHANGED TO INCLUDE <LABELID> OR <ARRAYID>[<SUBSCRIPT EXPRESSION>]. WHERE "LABELID" IS A LABEL AND "ARRAYID" IS AN ARRAY IDENTIFIER.

FOR EXAMPLE, WHERE "A" IS DECLARED TO BE A TWO-DIMENSIONAL ARRAY AND "L" IS DECLARED TO BE A LABEL,

```
MONITOR LINE (A[I+J , K DIV M] , L) ;
```

WILL PERMIT MONITORING THE SUBSCRIPTED VARIABLE AND LABEL SHOWN. WHEN MONITORING A LABEL, THE LABEL IDENTIFIER WILL BE LISTED ON THE FILE WHEN THE LABEL IS ENCOUNTERED. WHEN MONITORING A SUBSCRIPTED VARIABLE CONTAINING A SUBSCRIPT EXPRESSION, THE EXPRESSION IS EVALUATED EACH TIME ANY ELEMENT OF THE ARRAY IS STORED INTO.

COBOL EXTENSIONS

IN THE COBOL MONITOR STATEMENT, THE WORD "ALL" MAY NOW APPEAR IN THE LIST OF ITEMS TO BE SUBJECT TO THE MONITOR, CAUSING ALL

PROCEDURE NAMES TO BE MONITORED. WHEN "ALL" APPEARS IN THE LIST, THE PRESENCE OF PROCEDURE-NAMES OR ADDITIONAL OCCURRENCES OF THE WORD "ALL" WILL BE IGNORED.

D0006 DUMP STATEMENT - 08-24-72

THIS STATEMENT HAS BEEN GENERALIZED THROUGHOUT THE COMPILERS. THE DUMP STATEMENT HAS BEEN IMPLEMENTED FOR THE FIRST TIME IN ALGOL AND FORTRAN WHILE "DUMP ALL" HAS BEEN ADDED TO COBOL. THE DUMP STATEMENT ALLOWS SURVEILLANCE OF DESIGNATED VARIABLES DURING EXECUTION OF THE USERS PROGRAM.

DUMP IN ALGOL:

ALGOL SYNTAX FOR THE DUMP STATEMENT IS AS FOLLOWS:

<DUMP DECLARATION>::=DUMP<DUMP PART>

<DUMP PART>::=<FILE IDENTIFIER>(<DUMP LIST>)
<CONTROL PART>/<DUMP PART>,<FILE IDENTIFIER>
(<DUMP LIST>)<CONTROL PART>

<DUMP LIST>::=<DUMP LIST ELEMENT>/<DUMP LIST>,
<DUMP LIST ELEMENT>

<DUMP LIST ELEMENT>::=<SIMPLE VARIABLE>/<ARRAY IDENTIFIER>/
<LABEL IDENTIFIER>

<CONTROL PART>::=<MOD EXPRESSION>(<DUMP INDICATOR>
,<LOWER BOUND>,<UPPER BOUND>)/
<LABEL><MOD EXPRESSION>

<MOD EXPRESSION>::=<UNSIGNED INTEGER>/<EMPTY>

<DUMP INDICATOR>::=<SIMPLE ARITHMETIC VARIABLE>/
<EMPTY>

<LOWER BOUND>::=<ARITHMETIC EXPRESSION>/<EMPTY>

<UPPER BOUND>::=<ARITHMETIC EXPRESSION>/<EMPTY>

THE <DUMP DECLARATION> DECLARES WHICH IDENTIFIERS ARE TO BE PLACED

UNDER SURVEILLANCE. DIAGNOSTIC INFORMATION REQUESTED BY <DUMP DECLARATION> IS WRITTEN ON THE FILE DESIGNATED BY <FILE IDENTIFIER>. DIAGNOSTIC INFORMATION IS GIVEN ONLY WHEN <CONTROL PART> PARAMETERS ARE SATISFIED, I.E.,

1. EXECUTION CONTROL HAS PASSED TO THE SOURCE STATEMENT INDICATED BY <LABEL> IN <CONTROL PART>;

2, IF <DUMP INDICATOR> IS EMPTY, THEN, A) THE NUMBER OF TIMES CONTROL HAS PASSED TO <LABEL> IS GREATER THAN OR EQUAL TO <LOWER BOUND> AND LESS THAN OR EQUAL TO <UPPER BOUND> AND, B) THE NUMBER OF TIMES CONTROL HAS PASSED TO <LABEL> MUST BE DIVISIBLE BY <:MOD EXPRESSION>;

3. IF <DUMP INDICATOR> IS NOT EMPTY THEN THE VALUE OF THE DESIGNATED VARIABLE IS USED AS THE CONTROL VALUE.

4. IF <LOWER BOUND> IS EMPTY THEN ZERO IS ASSUMED;

5. IF <UPPER BOUND> IS EMPTY THEN INFINITY IS ASSUMED;

6. IF <MOD EXPRESSION> IS EMPTY THEN ONE IS ASSUMED.

RESTRICTIONS: ONLY THE FIRST SIX CHARACTERS OF ANY IDENTIFIER ARE WRITTEN. ARRAY IDENTIFIERS MUST BE SINGLE DIMENSIONED ARRAYS.

ALGOL EXAMPLE

```
BEGIN ARRAY A [0:1]; BOOLEAN B; REAL R; INTEGER I; LABEL L1, L2;
FILE LP (KIND = PRINTER, MAXRECSIZE = 15);
DUMP LP (A, B, I, L1, R) L2 : 2 (I, 2, 5), LP (L2) L1:L;
DO L1 : L2 : UNTIL R : R + 1 > 10;
END.
```

FOR THE ABOVE PROGRAM THE FOLLOWING IS PRINTED ON FILE LP:

```
L1 FROM 002:001C:1.
L2 = 0,
L1 FROM 002:001C:L.
L2 = 1,
L2 FROM 002:001D:3.
```

```
A = 0.0, 0.0, B = .FALSE., I =02, L1 = 2, R = 1.0000000000,
L1 FROM 002:001C:1.
L2 = 2,
L1 FROM 002:001C:1.
L2 = 3,
L2 FROM 002:001D:3.
A = 0.0, 0.0, B = .FALSE., I = 4, L1 = 4, 4 = 3.0000000000,
L1 FROM 002:001C:1.
L2 = 4,
L1 FROM 002:001C:1.
L2 = 5,
L1 FROM 002:001C:1.
L2 = 6,
L1 FROM 002:001C:1.
L2 = 7,
L1 FROM 002:001C:1.
L2 = 8,
L1 FROM 002:001C:1.
L2 = 9,
L1 FROM 002:001C:1.
L2 = 10,
```

IF THE OPTION LINEINFO IS SET WHEN COMPILING, SEQUENCE NUMBER INFORMATION WILL BE PRINTED.

DUMP IN FORTRAN

FORTRAN SYNTAX FOR THE DUMP STATEMENT IS AS FOLLOWS:

> DEBUG DUMP (FI) L, F (N=I1,I2)/D1, D2, ... WHERE FI IS A CONSTANT NUMBER FILE IDENTIFIER, L IS A LABEL IDENTIFIER, F, I1, I2 ARE UNSIGNED INTEGERS, N IS ANY SIMPLE VARIABLE, AND D1, D2, ... IS A LIST OF VARIABLES. "DEBUG" MUST APPEAR IN COLUMNS ONE THROUGH FIVE. THE REMAINDER OF THE STATEMENT CAN APPEAR IN COLUMNS SEVEN THROUGH 72 AND ONTO CONTINUATION CARDS.

DIAGNOSTIC INFORMATION CONTAINING THE VALUES OF THE DESIGNATED VARIABLESWILL BE WRITTEN ON FILE FI WHENEVER EXECUTION CONTROL

PASSES TO LABEL L AND THE CONTROL VARIABLE N, SATISFIES THE CONTROL PARAMETERS F , I1, I2. THE LIST OF VARIABLES UNDER SURVEILLANCE MUST ADHERE TO THE SYNTAX FOR NAMELIST ELEMENTS WITH THE ADDITION THAT LABEL IDENTIFIERS ARE ALLOWED.

THE RANGE IN WHICH DIAGNOSTIC OPERATIONS ARE ACTIVE IS DETERMINED BY THE LOWER BOUND, I1, AND THE UPPER BOUND, I2. WHEN THE CONTROL VARIABLE IS WITH THE RANGE, [I1, I2], THEN DUMP ACTION WILL BE PERFORMED WITH A FREQUENCY OF F.

IF F IS NOT PRESENT THEN 1 WILL BE ASSUMED AND THE VALUES OF THE SPECIFIED VARIABLES WILL BE WRITTEN EVERY TIME THE SOURCE STATEMENT DESIGNATED BY L IS TO BE EXECUTED AND THE CONTROL VARIABLE IS WITHIN THE RANGE [I1, I2]. IF I1 IS NOT PRESENT THEN 1 WILL BE ASSUMED; IF I2 IS NOT PRESENT THEN THERE WILL BE NO UPPER BOUND ON THE CONTROL RANGE, I.E., DIAGNOSTIC ACTION WILL DEPEND ONLY ON F AND I1.

IF THE CONTROL VARIABLE, N=, IS NOT PRESENT, THEN DIAGNOSTIC CONTROL WILL BE MAINTAINED VIA THE NUMBER OF TIMES EXECUTION CONTROL PASSES TO THE LABEL, L.

THERE MAY BE MORE THAN ONE DEBUG STATEMENT PER PROGRAM UNIT. EACH DEBUG STATEMENT MUST PRECEDE THE OCCURRENCE OF ALL LABELS REFERENCED BY IT.

FORTRAN EXAMPLE:

THE FOLLOWING PROGRAM -

```
DEBUG DUMP (6) 10, 2 (K=1, 15)/M
      DO 10 M = 1, 6
      K = M + 10
10    CONTINUE
      STOP
      END
```

WILL PRINT ON FILE6

```
10 FROM 0002:000B:0
M = 2
```

```
10 FROM 0002:000B:0
M = 4.
```

IF THE OPTION, "LINEINFO", IS SET WHEN COMPILING, SEQUENCE NUMBER INFORMATION WILL BE PRINTED.

DUMP IN COBOL

THE COBOL SYNTAX HAS BEEN EXTENDED TO THE FOLLOWING:

```
DUMP (<FILE-NAME>/PRINTER)
   ( (<DATANAME-D/(<PROCEDURE-NAME-D/ALL) )
      [ (<DATA-NAME-2>/(PROCEDURE-NAME-2>)/ALL) )])...)
         <PARAGRAPH-NAME-N> : (<LITERAL>/<DATA-NAME-N>)
```

THUS "ALL" MAY BE SUBSTITUTED FOR ANY PROCEDURE-NAME, IN WHICH CASE ALL PROCEDURES ARE SUBJECT TO THE "DUMP" OPERATION. DUE TO THE REQUIREMENT THAT THE OUTPUT FILE BE OPEN PRIOR TO THE FIRST USE OF DUMP OPERATION, <PARAGRAPH-NAME-N> CANNOT BE THE FIRST PROCEDURE-NAME OF THE PROGRAM.

COBOL EXAMPLE:

```
PROCEDURE DIVISION
   DUMP PRINTER (ALL) DUMP-PARAGRAPH : 2.
   FIRST-PARAGRAPH
   OPEN-OUTPUT PRINTER
   P1.          GO TO DUMP-PARAGRAPH.
                .
                .
                .
   P2.          GO TO P1
   P3.          .
                .
                .
   DUMP-PARAGRAPH.  GO TO P2.
```

A DIAGNOSTIC OPERATION WILL BE PERFORMED EVERY OTHER TIME CONTROL PASSES TO DUMP-PARAGRAPH. THE OUTPUT WILL LIST

```
FIRST-PARAGRAPH = +1
P1 =              +2
P2 =              +1
P3 =              +0
DUMP-PARAGRAPH =  +2
FIRST-PARAGRAPH = +1
P1 =              +4
P2 =              +3
P3 =              +0
DUMP-PARAGRAPH =  +4
```

D0007 ALGOL - USER CONTROLLED SEGMENTATION - 07-17-72

TWO NEW DOLLAR OPTIONS, "BEGINSEGMENT" AND "ENDSEGMENT" HAVE BEEN IMPLEMENTED IN ALGOL AND ESPOL TO ALLOW FURTHER USER CONTROL OF PROCEDURE SEGMENTATION. PROCEDURES ENCOUNTERED BETWEEN THE BEGINSEGMENT AND ENDSEGMENT WILL ALL BE PLACED IN THE SAME SEGMENT.

THE BEGINSEGMENT OPTION MUST APPEAR BEFORE THE DECLARATION OF THE FIRST PROCEDURE TO BE INCLUDED IN THE USER SEGMENT. THE FIRST PROCEDURE IN THE USER SEGMENT MUST BE THE ONE WHICH THE COMPILER WOULD NORMALLY SEGMENT.

THE ENDSEGMENT MUST APPEAR AFTER THE LAST SOURCE IMAGE OF THE LAST PROCEDURE IN THE USER SEGMENT.

ONLY PROCEDURES AND BLOCKS COMPLETELY CONTAINED INSIDE A PROCEDURE IN THE SEGMENT MAY BE INCLUDED IN A USER SEGMENT.

USER SEGMENTS MAY BE NESTED, I.E., A BEGINSEGMENT MAY APPEAR IN A USER SEGMENT. IN THIS CASE AN ENDSEGMENT APPLIES TO THE USER SEGMENT CURRENTLY BEING COMPILED.

IF A $ BEGINSEGMENT APPEARS BEFORE THE BEGINNING OF A SEPARATELY COMPILED PROCEDURE, A $ ENDSEGMENT IS ASSUMED AT THE END OF THE PROCEDURE EVEN IF NONE APPEARS. THE DRIVER PROCEDURE CREATED FOR PROCEDURES COMPILED AT LEVEL THREE WILL ALWAYS BE IN A DIFFERENT SEGMENT.

EXPLANATION OF ERROR MESSAGES:

1. EXTERNAL PROCEDURES ILLEGAL IN A USER SEGMENT:

AN EXTERNAL PROCEDURE DECLARATION HAS APPEARED IN A USER SEGMENT.

2. USER SEGMENTS MAY CONTAIN ONLY PROCEDURES:

AN ATTEMPT HAS BEEN MADE TO INCLUDE A NON-PROCEDURAL BLOCK IN A USER SEGMENT.

3) ILLEGAL BEGINSEGMENT OR ENDSEGMENT:

A BEGINSEGMENT OR ENDSEGMENT CARD HAS BEEN SEEN OUTSIDE DECLARATIONS.

4) A PROCEDURE MAY NOT BE SPLIT ACROSS SEGMENTS:

ENDSEGMENT MISSING AFTER NESTED USERSEGMENTED PROCEDURE.

5) ENDSEGMENT REQUIRED:

A BEGINSEGMENT APPEARED AND THERE WAS NO ENDSEGMENT.

6) USER SEGMENT MAY NOT HAVE SAVE PROCEDURES (ESPOL ONLY):

AN ATTEMPT WAS MADE TO INCLUDE A SAVE PROCEDURE IN A USER SEGMENT.

USE OF THE OPTIONS:

BEGINSEGMENT AND ENDSEGMENT MAY NOW BE SET, RESET, OR POPPED.

THE PRINTOUT FOR SEGMENT INFORMATION IS MODIFIED FOR USER SEGMENTS. USER SEGMENTS WILL BE NUMBERED CONSECUTIVELY IN A PROGRAM BEGINNING WITH ONE. (I.E., THE FIRST BEGINSEGMENT WILL CREATE USERSEGMENT1, THE SECOND BEGINSEGMENT WILL CREATE USERSEGMENT2.) AT THE BEGINNING OF A USER SEGMENT, ITS SEGMENT NUMBER WILL BE PRINTED OUT. THE LENGTH OF EACH USER SEGMENT WILL BE PRINTED AT ITS END. PROCEDURES THAT WOULD NORMALLY BE SEGMENTED, BUT ARE NOT BECAUSE OF USER SEGMENTATION, WILL PRINT OUT AS BEING "IN" A PARTICULAR SEGMENT.

EXTERNAL PROCEDURES MAY NOT BE DECLARED IN A USER SEGMENT.

FORWARD PROCEDURE DECLARATIONS ARE NOT AFFECTED BY USER SEGMENTATION.

A PROCEDURE MAY NOT BE SPLIT ACROSS USER SEGMENTS.

IS ESPOL ONLY, USER SEGMENTATION SHOULD NOT BE USED IF STACK BUILDING CODE IN A USERSEGMENT INVOKES A PROCEDURE IN THAT USERSEGMENT. FOR EXAMPLE:

```
    $ BEGINSEGMENT
  REAL PROCEDURE P;
    BEGIN
      P:=1;
    END;
  REAL R:=P;
  $ ENDSEGMENT
```

WILL RESULT IN INCORRECT CODE.

WARNING MESSAGES:

IF MORE THAN ONE $ BEGINSEGMENT APPEARS BEFORE A PROCEDURE, THE WARNING MESSAGE "EXTRA $BEGINSEGMENT IGNORED" WILL BE PRINTED.

IF A $ ENDSEGMENT APPEARS WHEN THE USER IS NOT CONTROLLING SEGMENTATION, THE WARNING MESSAGE "EXTRA ENDSEGMENT IGNORED" WILL BE PRINTED.

D0008 GLOBALS IN BINDING - 08-24-72

SEVERAL CHANGES HAVE BEEN MADE TO THE BINDER AND THE COMPILERS TO FACILITATE SEPARATE COMPILATIONS AND BINDING. THESE INCLUDE LIBERALIZATION OF THE RULES FOR LISTING GLOBAL DECLARATIONS, ADDING GLOBALS AT BIND TIME, AND ACCESSING FORTRAN COMMON BLOCKS IN ALGOL.

GLOBAL DECLARATIONS

IN ALGOL AND DCALGOL SEPARATE COMPILATIONS, THE GLOBAL DECLARATIONS (WITHIN BRACKETS) MAY NOW HAVE THE SAME FORM AS GLOBAL DECLARATIONS

FOR NORMAL COMPILATION, EXCEPT THAT PROCEDURES WITH BODIES ARE NOT ALLOWED.

ARRAYS THAT ARE DECLARED WITHIN THE GLOBAL DECLARATIONS MAY EITHER LIST JUST LOWER BOUNDS OR BOTH LOWER AND UPPER BOUNDS. IF ONE UPPER BOUND IS LISTED FOR A SPECIFIC DECLARATION, ALL UPPER BOUNDS MUST BE LISTED.

EXTRANEOUS INFORMATION WILL BE IGNORED. UPPER BOUNDS FOR ARRAYS, FOR EXAMPLE, ARE INFORMATION THAT WILL FREQUENTLY BE IGNORED. UPPER BOUNDS ARE REQUIRED, HOWEVER, FOR GLOBAL ARRAYS THAT ARE BEING ADDED AT BIND TIME (DESCRIBED NEXT).

ADDING GLOBALS AT BIND TIME

ANY VARIABLE, ARRAY, OR PROCEDURE WHICH IS DECLARED GLOBAL BUT IS NOT FOUND IN THE HOST WILL BE ADDED AS A NEW GLOBAL AT BIND TIME. THIS FACILITY IS DESIGNED TO REDUCE THE NEED FOR RECOMPILING HOST PROGRAMS.

TO ADD A NEW GLOBAL ARRAY, IT IS NECESSARY TO DECLARE ITS SIZE. THIS IS A CASE, THEREFORE, WHERE THE UPPER BOUNDS MUST BE DECLARED WHEN THE ARRAY IS DECLARED AS GLOBAL. FAILURE TO DO SO WILL RESULT IN A BIND TIME ERROR. IF A GLOBAL ARRAY IS BEING ADDED THAT IS REFERENCED BY MORE THAN ONE PROCEDURE, IT WILL BE ASSIGNED THE MAXIMUM LENGTH OF ALL DECLARATIONS. THE LENGTH OF AN ARRAY IN THE HOST PROGRAM, HOWEVER, WILL NOT BE CHANGED (EXCEPT FOR FORTRAN COMMON BLOCKS).

A NEW GLOBAL PROCEDURE NOT FOUND IN THE HOST WILL BE TREATED AS IF IT WERE DECLARED EXTERNAL IN THE HOST (ASSUMING THE HOST WAS COMPILED AT LEVEL TWO) AND WILL BE BOUND AUTOMATICALLY.

FORTRAN FILES MAY BE ADDED AT BIND TIME. HOWEVER, ALGOL AND DCALGOL FILES MAY NOT. False (II.6): files may be added but declared attributes are ignored.

ACCESSING FORTRAN COMMON IN ALGOL

A FORTRAN COMMON BLOCK MAY NOW BE ACCESSED BY ALGOL (AND DCALGOL) AS EITHER A SINGLE PRECISION ARRAY, A DOUBLE PRECISION ARRAY, OR

BOTH. IN ADDITION, AN ALGOL (OR DCALGOL) SINGLE OR DOUBLE PRECISION ARRAY MAY BE ACCESSED BY FORTRAN THROUGH COMMON.

ALL ARRAYS MUST BE GLOBAL, AND COMMON BLOCK NAMES INCLUDE SLASHES. THE NAME FOR THE BLANK COMMON BLOCK IS /.BLNK./.

TO BIND ALGOL TO A FORTRAN HOST ACCESSING COMMON, THE FOLLOWING PROCEDURES MAY BE USED:

1) TO ACCESS A COMMON BLOCK AS SINGLE PRECISION, DECLARE AN ALGOL ARRAY AND EQUATE IT TO THE FORTRAN COMMON BLOCK WITH A BINDER "USE" STATEMENT. FOR EXAMPLE, WITH A SINGLE PRECISION ALGOL ARRAY A AND A FORTRAN COMMON BLOCK BLK, THE BINDER INSTRUCTION WOULD BE:

```
USE /BLK/ FOR A;
```

2) TO ACCESS A COMMON BLOCK AS DOUBLE PRECISION, USE THE SAME STATEMENT AS ABOVE, EXCEPT DECLARE THE ALGOL ARRAY AS DOUBLE.

3) TO ACCESS A COMMON BLOCK AS BOTH SINGLE AND DOUBLE, DECLARE BOTH A SINGLE AND DOUBLE PRECISION ALGOL ARRAY AND EQUATE BOTH OF THEM TO THE FORTRAN COMMON BLOCK WITH A BINDER "USE" STATEMENT. FOR EXAMPLE, WITH A SINGLE PRECISION ALGOL ARRAY A, A DOUBLE PRECISION ALGOL ARRAY D, AND A FORTRAN COMMON BLOCK BLK, THE BINDER INSTRUCTION WOULD BE:

```
USE /BLK/ FOR A,D;
```

TO BIND FORTRAN TO AN ALGOL HOST, IN WHICH IT IS DESIRED TO HAVE A FORTRAN COMMON BLOCK ACCESS ALGOL GLOBAL ARRAYS, THE FOLLOWING TECHNIQUES MAY BE USED:

1) TO ACCESS A SINGLE PRECISION ARRAY THROUGH COMMON, DECLARE AN ALGOL ARRAY AND EQUATE IT TO THE COMMON BLOCK WITH A BINDER "USE" STATEMENT. FOR EXAMPLE, WITH AN ALGOL SINGLE PRECISION ARRAY A AND A FORTRAN COMMON BLOCK BLK, THE BINDER INSTRUCTION WOULD BE:

```
USE A FOR /BLK/;
```

2) TO ACCESS A DOUBLE PRECISION ARRAY THROUGH COMMON, USE THE SAME STATEMENT AS ABOVE, EXCEPT DECLARE THE ALGOL ARRAY AS DOUBLE.

3) TO ACCESS AN ALGOL ARRAY AS BOTH SINGLE AND DOUBLE THROUGH COMMON, DECLARE A SINGLE PRECISION ALGOL ARRAY, AN ADJACENT DOUBLE PRECISION ALGOL COPY ARRAY, AND EQUATE THE SINGLE PRECISION ARRAY TO THE COMMON BLOCK WITH A BINDER "USE" STATEMENT.

FOR EXAMPLE, WITH AN ALGOL SINGLE PRECISION ARRAY A, AN ALGOL DOUBLE PRECISION ARRAY D, AND A FORTRAN COMMON BLOCK BLK, THE ALGOL ARRAYS MUST BE DECLARED IN THE ALGOL PROGRAM SIMILARLY TO:

```
REAL ARRAY A[0:99];
DOUBLE ARRAY D[ 0 ]= A;
```

THEN THE BINDER INSTRUCTION WOULD BE:

```
USE A FOR /BLK/;
```

FORTRAN REFERENCES TO SINGLE PRECISION ITEMS IN /BLK/ WILL BE CHANGED TO REFER TO A, AND FORTRAN REFERENCES TO DOUBLE OR COMPLEX ITEMS IN /BLK/ WILL BE CHANGED TO REFER TO D. NOTE THAT IT IS NOT SUFFICIENT FOR D TO MERELY BE A COPY OF A, IT MUST ALSO BE DECLARED ADJACENT TO A;

SIMULATING COMMON IN ALGOL

A FORTRAN COMMON BLOCK IS A ONE DIMENSIONAL SINGLE PRECISION ARRAY WITH AN ADJACENT DOUBLE PRECISION COPY DESCRIPTOR. THE CONTENTS OF THE ARRAY MAY BE DETERMINED BY MAPPING THE ELEMENTS OF THE COMMON STATEMENT INTO A CONTIGUOUS ARRAY. THIS CAN BE SIMULATED IN ALGOL AS ILLUSTRATED BY THE FOLLOWING EXAMPLE:

FORTRAN STATEMENTS:

```
DOUBLE PRECISION DA(10)
COMMON /C/ RA(7), X, DA
```

ALGOL STATEMENTS:

```
ARRAY A[0:27];
DOUBLE ARRAY D[0] = A;
DEFINE DA(N) = D[(N)+3]#,
       RA(N) = A[(N)+1]#,
       X    = A[7]#;
```

THE APPROPRIATE BINDER INSTRUCTION SHOULD BE

```
USE A FOR /C/;
```

IN THIS EXAMPLE, SUBSCRIPTS HAVE BEEN ADJUSTED SO THAT D[1] AND A[1] IN ALGOL ARE THE SAME AS D(1) AND A(1) IN FORTRAN.

D0009 ALGOL CHARACTER ARRAYS - 05-12-72

ALGOL SYNTAX FOR <ARRAY DECLARATION>S AND <SIMPLE POINTER EXPRESSION>S HAS BEEN EXTENDED TO ALLOW THE DECLARATION AND USE OF STRING OR CHARACTER ARRAYS.

THE FOLLOWING DEFINITIONS SHOULD EITHER REPLACE OR BE ADDED TO THE SYNTAX FOR ARRAY DECLARATIONS ON PAGE 10-5 OF THE B6700 ALGOL LANGUAGE DOCUMENT.

```
<ARRAY DECLARATION>::=<ARRAY KIND> ARRAY <ARRAY LIST>/
   <STRING ARRAY KIND> ARRAY <ARRAY LIST>

<STRING ARRAY KIND>::=<DIRECT SPECIFIER>
   <LOCAL OR OWN STRING TYPE>/ LONG
   <LOCAL OR OWN STRING TYPE>

<LOCAL OR OWN STRING TYPE>::=<STRING TYPE>/OWN<STRING TYPE>

<STRING TYPE>::=HEX/BCL/ASCII/EBCDIC

<ARRAY SEGMENT>::=<IDENTIFIER LIST>[<BOUND PAIR LIST>]/
   <EQUIVALENCE PART>

<EQUIVALENCE PART>::=<IDENTIFIER>[<LOWER-BOUND>]
   = <ARRAY ROW>/<IDENTIFIER>[<LOWER-BOUND>]
```

= <STRING ARRAY ROW>

A HEX ARRAY REFERENCES DATA BY MEANS OF A FOUR-BIT STRING DESCRIPTOR, A BCL ARRAY WITH A SIX-BIT STRING DESCRIPTOR, AND ASCII AND EBCDIC ARRAYS WITH EIGHT-BIT STRING DESCRIPTORS.

THE <EQUIVALENCE PART> ALLOWS ONE TO REFERENCE AN <ARRAY ROW> WITH A (COPY) DESCRIPTOR OF A DIFFERENT WORD OR CHARACTER SIZE.

STRING ARRAYS MAY BE PASSED AS PARAMETERS, USED AS <ARRAY ROW>S, OR USED AS SIMPLE POINTER EXPRESSIONS.

THE FOLLOWING CHART SHOWS THE CORRESPONDENCE BETWEEN COBOL USAGE AND ALGOL STRING TYPE THAT IS REQUIRED FOR PASSING STRING ARRAYS AS PARAMETERS BETWEEN COBOL AND ALGOL.

COBOL 01 LEVEL USAGE	ALGOL STRING ARRAY TYPE
COMP-2	HEX
DISPLAY-1	BCL
ASCII	ASCII
DISPLAY	EBCDIC

THE DEFINITION OF <SIMPLE POINTER EXPRESSION> ON PAGE 7-15 OF THE B6700 ALGOL LANGUAGE DOCUMENT SHOULD BE REPLACED WITH:

<SIMPLE POINTER EXPRESSION>::=<POINTER PRIMARY>
<SKIP>/<POINTER ASSIGNMENT>/<STRING ARRAY PART>

<STRING ARRAY PART>::=<STRING ARRAY NAME>/
<STRING ARRAY ROW>/<SUBSCRIPTED STRING ARRAY VARIABLE>

<STRING ARRAY NAME>::=<STRING ARRAY IDENTIFIER>/
<DIRECT STRING ARRAY IDENTIFIER>

<STRING ARRAY ROW>::=<STRING ARRAY NAME>
[<ROW DESIGNATOR>]

<SUBSCRIPTED STRING ARRAY VARIABLE>::=
<STRING ARRAY NAME>[<SUBSCRIPT LIST>]

<STRING ARRAY IDENTIFIER>::=<IDENTIFIER>

<DIRECT STRING ARRAY IDENTIFIER>::=<IDENTIFIER>

STRING ARRAYS OF TWO OR MORE DIMENSIONS MAY BE SWAPPED IF THEY ARE OF THE SAME CHARACTER TYPE. STRING ARRAYS MAY ALSO BE PASSED AS PARAMETERS TO INSTALLATION INTRINSICS.

STRING ARRAYS ARE ALSO PERMITTED TO BE DECLARED AS THE FORMAL NAME PARAMETERS FOR THE INPUT, OUTPUT, OR COMPARE PROCEDURES REFERENCED BY A SORT OR MERGE STATEMENT. IN THIS CASE, THE RECORD-LENGTH PART OF THE SORT STATEMENT WILL BE USED AS A CHARACTER LENGTH RATHER THAN A WORD LENGTH. ALL ARRAYS DECLARED AS FORMAL NAME PARAMETERS FOR THESE PURPOSES MUST, OF COURSE, BE CONSISTENT WITH THE ACTUAL PARAMETERS.

IN ESPOL, STRING ARRAY DECLARATIONS ARE ESSENTIALLY THE SAME AS IN ALGOL, EXCEPT FOR THE FOLLOWING:

1. THE <EQUIVALENCE PART> IS NOT ALLOWED;
2. RULES APPLICABLE TO ARRAY DECLARATIONS, SUCH AS ASTERISK BOUNDS, ADDRESS EQUATION, INITIALIZATION, SAVE DECLARATIONS, ETC., ALSO APPLY TO STRING ARRAY DECLARATIONS.

D0010 TRUTHSETS AND TRANSLATE TABLES - 06-20-72

TWO NEW DECLARATIONS, TRUTHSETS AND TRANSLATE TABLES, HAVE BEEN ADDED TO ALGOL AND ESPOL. <TRUTHSET DECLARATION> DEFINES ONE OR MORE TRUTHSETS WHICH MAY BE USED WITH THE SCAN AND REPLACE STATEMENTS AND WITH THE TABLE MEMBERSHIP BOOLEAN PRIMARY.

TRUTHSET DECLARATION SYNTAX:

<TRUTHSET DECLARATION>::=TRUTHSET <TRUTHSET LIST>

<TRUTHSET LIST>::=<TRUTHSET ELEMENT>/
<TRUTHSET LIST>,<TRUTHSET ELEMENT>

<TRUTHSET ELEMENT>::=<TRUTHSET IDENTIFIER>
(<MEMBERSHIP EXPRESSION>)

<MEMBERSHIP EXPRESSION>::=<MEMBERSHIP SECONDARY>/
<MEMBERSHIP EXPRESSION><BOOLEAN OPERATOR>
<MEMBERSHIP SECONDARY>

<MEMBERSHIP SECONDARY>::=<MEMBERSHIP PRIMARY>/
NOT <MEMBERSHIP PRIMARY>

<MEMBERSHIP PRIMARY>::=<STRING>/<TRUTHSET IDENTIFIER>/
ALPHA/ALPHA6/ALPHA8/(<MEMBERSHIP EXPRESSION>)

SEMANTICS:

ALL MEMBERSHIP PRIMARIES OF A MEMBERSHIP EXPRESSION MUST BE OF THE SAME CHARACTER TYPE (4, 6, 7, OR 8), THEREBY DETERMINING THE TYPE OF THE TRUTHSET. THE CHARACTER SIZE OF STRINGS IS OBTAINED FROM THE MAXIMUM INTERNAL CHARACTER SIZE OF THE STRING.

THE MEMBERSHIP EXPRESSION IS EVALUATED ACCORDING TO THE NORMAL RULES OF PRECEDENCE FOR BOOLEAN OPERATORS.

THE MEANING OF ALPHA, ALPHA6, AND ALPHA8 IS DEFINED ON PAGE 7-13 OF THE ALGOL LANGUAGE DOCUMENT.

THE SYNTACTICAL DEFINITION OF <TABLE POINTER> ON PAGE 7-10 OF THE ALGOL LANGUAGE DOCUMENT SHOULD BE CHANGED TO:

<TABLE POINTER>::=<TRUTHSET IDENTIFIER>/ALPHA/ALPHA6/
ALPHA8/<SUBSCRIPTED VARIABLE>

ALL TRUTHSETS DECLARED BY A SINGLE DECLARATION ARE MADE COMMON TO A SINGLE READ-ONLY ARRAY. SEPARATE DECLARATIONS PRODUCE SEPARATE READ-ONLY ARRAYS.

TRANSLATETABLE CONSTRUCT

A <TRANSLATETABLE DECLARATION> DEFINES ONE OR MORE TRANSLATE TABLES WHICH MAY BE USED WITH THE REPLACE STATEMENT.

TRANSLATE TABLE DECLARATION SYNTAX:

<TRANSLATE TABLE DECLARATION>::=TRANSLATETABLE

<TRANSLATE TABLE LIST>

<TRANSLATE TABLE LIST>::=<TRANSLATE TABLE ELEMENT>/
<TRANSLATE TABLE LIST>,<TRANSLATE TABLE ELEMENT>

<TRANSLATE TABLE ELEMENT>::=<TRANSLATE TABLE IDENTIFIER>
(<TRANSLATION LIST>)

<TRANSLATION LIST>::=<TRANSLATION SPECIFIER>/
<TRANSLATION LIST>,<TRANSLATION SPECIFIER>

<TRANSLATION SPECIFIER>::=<SOURCE PART> TO
<DESTINATION PART>/<TRANSLATE TABLE IDENTIFIER>

<SOURCE PART>::=<STRING>/<CHARACTER SET>

<DESTINATION PART>::=<STRING>/<CHARACTER SET>/
<SPECIAL DESTINATION CHARACTER>

<CHARACTER SET>::=BCL/EBCDIC/ASCII/HEX

<SPECIAL DESTINATION CHARACTER>::=<STRING>

SEMANTICS:

THE <CHARACTER SET> IS EQUIVALENT TO A STRING CONTAINING ALL CHARACTERS IN THE SPECIFIED SET, IN ASCENDING BINARY SEQUENCE, WHOSE LENGTH IS EQUAL TO THE NUMBER OF CHARACTERS IN THE SET, I. E., THE SCOPE OF A <CHARACTER SET> IS THE FULL CHARACTER SET.

THE SCOPE OF A STRING IS THE CHARACTERS IN THE STRING. THE LENGTH OF A STRING IS ITS LENGTH IN TERMS OF ITS MAXIMUM INTERNAL CHARACTER SIZE.

EACH SUCCEEDING <TRANSLATION SPECIFIER> OVERRIDES, WITHIN ITS SCOPE, PREVIOUS <TRANSLATION SPECIFIER>S.

WITHIN A <TRANSLATION LIST>, ALL SOURCE CHARACTER SIZES MUST BE THE SAME AND ALL DESTINATION CHARACTER SIZES MUST BE THE SAME, ALTHOUGH THE CHARACTER SIZES OF THE SOURCE AND DESTINATION PARTS NEED NOT BE THE SAME.

THE LENGTH OF THE <DESTINATION PART> MUST EQUAL THE LENGTH OF

THE <SOURCE PART>, UNLESS THE <CHARACTER SET> IS USED FOR BOTH THE <SOURCE PART> AND <DESTINATION PART> OR THE <SPECIAL DESTINATION CHARACTER> IS USED. IF THE <SPECIAL DESTINATION CHARACTER> IS USED, ALL CHARACTERS WITHIN THE SCOPE OF THE <SOURCE PART> ARE TRANSLATED TO THE <SPECIAL DESTINATION CHARACTER> WHICH MUST BE A STRING WHOSE LENGTH IS ONE IN TERMS OF ITS MAXIMUM INTERNAL CHARACTER SIZE.

EVERY TRANSLATE TABLE HAS A DEFAULT BASE IN WHICH ALL SOURCE CHARACTERS ARE TRANSLATED TO ZERO CHARACTERS (ALL BITS OFF). THE USE OF A <CHARACTER SET> FOR BOTH THE SOURCE AND DESTINATION PARTS INVOKES THE STANDARD TABLE FROM THE MCP AND PROVIDES A HANDY WAY OF OBTAINING A LEGITIMATE BASE UPON WHICH ADDITIONAL <TRANSLATION SPECIFIERS> MAY BE USED, IF DESIRED, TO OVERRIDE CERTAIN PARTS OF THE STANDARD TABLE. THE USE OF A <TRANSLATE TABLE IDENTIFIER> AS A <TRANSLATION SPECIFIER> MAY ALSO BE USED TO PROVIDE A BASE.

WHEN STRINGS OF EQUAL LENGTH ARE USED FOR THE SOURCE AND DESTINATION PARTS, TRANSLATION IS BASED UPON THE CORRESPONDING POSITIONS OF THE SOURCE AND DESTINATION CHARACTERS, STARTING FROM THE LEFT AND PROCEEDING TO THE RIGHT.

THE SYNTACTICAL DEFINITION OF <TRANSLATE TABLE> ON PAGE 9-9 OF THE ALGOL LANGUAGE DOCUMENT SHOULD BE CHANGED TO:

```
<TRANSLATE TABLE>::=<TRANSLATE TABLE IDENTIFIER>/
    <SUBSCRIPTED VARIABLE>/BCLTOEBCDIC/
    EBCDICTOBCL/HEXTOBCL/HEXTOEBCDIC/
    BCLTOHEX/EBCDICTOHEX/
```

ALL TRANSLATE TABLES DECLARED BY A SINGLE DECLARATION ARE MADE COMMON TO A SINGLE READ-ONLY ARRAY. SEPARATE DECLARATIONS PRODUCE SEPARATE READ-ONLY ARRAYS.

IN ESPOL, TRUTHSET AND TRANSLATE TABLE DECLARATIONS ARE ESSENTIALLY THE SAME AS ALGOL, EXCEPT FOR THE FOLLOWING:

1. THE "ALPHA" CONSTRUCT IS NOT RECOGNIZED FOR TRUTHSET DECLARATIONS.

2. THE "SAVE" DECLARATOR MAY BE USED AT LEX LEVEL ZERO, AS WITH INITIALIZED ARRAYS.

3. TRUTHSETS AND TRANSLATE TABLES MAY BE GIVEN EXPLICIT ADDRESSES.

D0011 STRING CODES IN ALGOL - 06-19-72

THE <STRING CODE> SYNTAX HAS BEEN EXTENDED TO ALLOW EXPLICIT SPECIFICATION OF AN INTERNAL STRING CHARACTER SIZE THAT IS A MULTIPLE OF THE SOURCE STRING CHARACTER SIZE. UNDER THE SYNTAX FOR STRINGS IN SECTION 5.3 OF THE ALGOL LANGUAGE DOCUMENT, REPLACE THE DEFINITIONS OF <BINARY CODE>, <OCTAL CODE>, AND <HEXADECIMAL CODE> WITH THE FOLLOWING:

<BINARY CODE>::=1/10/12/120/13/130/14/140/16/160/
17/170/18/180

<QUATERNARY CODE>::=2/20/24/240/26/260/27/270/28/280

<OCTAL CODE>::=3/30/36/360

<HEXADECIMAL CODE>::=4/40/48/480/47/470

SECTION 5.3.2, STRING CODE SEMANTICS, SHOULD BE REPLACED BY:

THE STRING CODE DETERMINES THE INTERPRETATION OF THE CHARACTERS BETWEEN THE QUOTES. IT SPECIFIES THE CHARACTER SET AND, FOR STRINGS OF LESS THAN 48 BITS, THE JUSTIFICATION. THE FIRST DIGIT OF THE STRING CODE SPECIFIES THE CHARACTER SET IN WHICH THE SOURCE STRING IS WRITTEN. THE NEXT NON-ZERO DIGIT SPECIFIES THE INTERNAL CHARACTER SIZE OF THE STRING CREATED BY THE COMPILER. IF NO SECOND SIZE IS SPECIFIED, THE INTERNAL SIZE IS THE SAME AS THE SOURCE SIZE. IF THE INTERNAL SIZE IS DIFFERENT FROM THE SOURCE SIZE, THE LENGTH OF THE STRING MUST BE AN INTEGRAL NUMBER OF INTERNAL CHARACTERS. A TRAILING ZERO IN THE STRING CODE SPECIFIES THAT THE STRING IS TO BE LEFT JUSTIFIED WITHIN A WORD, WITH TRAILING ZEROS, IF THE STRING CONTAINS FEWER THAN 48 BITS. IF THE STRING IS MORE THAN 48

BITS LONG, IT IS ALWAYS LEFT JUSTIFIED. IF THE STRING CODE IS <EMPTY>, ITS DEFAULT VALUE IS EIGHT. SEE SECTION 3.3.

D0012 INTRINSICS IN ESPOL - 04-18-72

THE INTRINSIC "INTEGERT", EMITTING AN NTIA, HAS BEEN ADDED TO ESPOL TO DO INTEGER TRUNCTION. IT TAKES ONE PARAMETER AND INTEGERIZES IT.

THE INTRINSIC "SCALERIGHTS" HAS BEEN ADDED TO ESPOL. IT TAKES THREE OR FOUR PARAMETERS. THE VALUE IT RETURNS AND THE FIRST THREE PARAMETERS ARE THE SAME AS FOR THE OTHER SCALE INTRINSICS. THE FOURTH PARAMETER, IF PRESENT, MUST BE A VARIABLE NAME OR ARRAY ELEMENT. INTO THIS PARAMETER IS STORED THE QUOTIENT AFTER SCALING THE NUMBER.

THE INTRINSICS UNPACK (UNTYPED) AND UNPACKU (POINTER TYPE) HAVE BEEN ADDED TO ESPOL. THEY CONVERT A SCALED INTEGER TO CHARACTER TYPE.

WRAPAROUND IS NOW PERMITTED IN A CONCATENATION EXPRESSION. FOR EXAMPLE:

```
X := 0 & Y [35:7:36]
```

WILL CONCATENATE BITS 7-0 AND 47-20 (IN THAT ORDER) INTO THE FIELD 35:36 OF X.

ALSO, THE FOLLOWING MNEMONICS ARE NOW IMPLEMENTED IN ESPOL:

```
LSS FOR <
GTR FOR >
LEQ FOR ≤
GEQ FOR ≥
EQL FOR =
NEQ FOR ≠
```

D0013 COMPILATION OF COMPILERS - 06-19-72

THE TABLES GENERATED BY THE ALGOL TABLE GENERATOR PROGRAM WILL NOW BE WRITTEN TO A DISK FILE INSTEAD OF A CARD PUNCH FILE. THE FILE

WILL BE "INCLUDED" DURING A COMPILATION. THE FILE NAME IS "ALGOLTABLES" AND MAY BE LABEL EQUATED. IF THE FILE IS NOT PRESENT DURING A COMPILE, THE MESSAGE "NO FILE INCLO" WILL BE DISPLAYED ON THE SPO.

IN ADDITION, TABLES FOR THE ESPOL AND XALGOL COMPILERS ARE GENERATED BY A TABLE GENERATION PROGRAM AND MERGED WITH THE SYMBOL FILE OF THE COMPILER TO CREATE A NEW SYMBOLIC.

D0014 ANSI AND B5500 COMPATIBILITY - 06-19-72

SEVERAL NEW DOLLAR OPTION HAVE BEEN ADDED TO THE COBOL COMPILER WHICH WILL RESTRICT RECOGNITION OF RESERVED WORDS. THESE "SYSTEM OPTIONS" ARE "B6700", "B5700", "B2500", "USASI" AND "S360". THE SYSTEM OPTION "B5700" IS TO BE USED FOR B5500 AND B5700; "B2500" IS TO BE USED FOR ALL MODELS OF B2500, B3500, B3700, AND B4700; "USASI" IS USED FOR PROGRAMS WHICH CONFORM TO THE COBOL STANDARD AS SPECIFIED IN THE PUBLICATION "USASI COBOL, X3-23, 1968"; AND "S360" IS TO BE USED FOR IBM-360 AND IBM-370.

SETTING OF A SYSTEM OPTION EXCLUDES FROM RECOGNITION AS A RESERVED WORD ANY WORD WHICH IS RESERVED ON THE B6700 COMPILER, BUT NOT RESERVED ON THE SYSTEM INDICATED BY THE SYSTEM OPTION WHICH IS THEN SET. THUS, THE WORD "MODIFY" IS RESERVED ONLY WHEN THE SYSTEM OPTION "B6700" IS SET, BUT WHEN B6700 IS NOT SET, THE WORD "MODIFY" MAY BE DECLARED TO BE (AND USED AS) A DATA-NAME, PROCEDURE-NAME, ETC.

A SYSTEM OPTION MAY BE SET BY A "$" CARD OR BY A "$ SET" CARD. ANY ATTEMPT TO "POP" OR "RESET" A SYSTEM OPTION WILL RESULT IN A WARNING MESSAGE AND THE SYSTEM OPTION WILL BE IGNORED. SETTING OF A SYSTEM OPTION CAUSES ALL OTHER SYSTEM OPTIONS TO BE RESET. A "$" CARD WITHOUT "SET", "POP", OR "RESET" WILL RESET ALL OPTIONS EXCEPT THOSE SPECIFIED ON THE "$" CARD AND WILL LEAVE THE SYSTEM OPTION SET TO "B6700" UNLESS SOME OTHER SYSTEM OPTION IS SPECIFIED BY THAT DOLLAR CARD. IF NO DOLLAR CARD IS USED, THEN "B6700" IS ASSUMED.

UNDER CONTROL OF THE SYSTEM OPTIONS, SEVERAL NEW WORDS HAVE BEEN

ADDED TO THE RESERVED WORD LIST. THE USES OF THESE WORDS ARE DESCRIBED IN THE COBOL MANUALS FOR THE APPLICABLE SYSTEMS. THE FOLLOWING IS A LIST OF THESE NEW WORDS WITH INDICATION OF THE SYSTEM OPTION SETTING(S) WHICH CAUSE THEM TO BE RECOGNIZED AS RESERVED WORDS:

B2	B5700, B2500
CMP	B5700, B2500
CMP-1	B5700, B2500
PC	B5700, B2500
MD	B5700
SY	B5700, B2500
REMARKS	B5700, B2500, USASI, S360
NOTE	B5700, B2500, USASI, S360
UNIT	B6700, B2500, USASI, S360
PROCESSING	B2500, USASI, S360
OC	B6700, B5700, B2500
VA	B6700, B5700, B2500
VALUES	B6700, B5700, USASI, S360
OTHERWISE	B6700, B2500, USASI
PT-READER	B2500
PT-PUNCH	B2500
TAPE-7	B2500
TAPE-9	B2500
TAPE-PE	B2500
DISK	B2500

IN ADDITION TO RECOGNIZING THE ABOVE WORDS SYNTACTICALLY, THEY WILL PRODUCE THE RESULTS EXPECTED BY THE SYSTEM INDICATED BY THE SYSTEM OPTION SINCE IN MOST CASES THEY REPRESENT ONLY AN ALTERNATE WORD TO ONE WHICH IS ALREADY IMPLEMENTED ON THE B6700. "NOTE", "REMARKS", AND "PROCESSING" (AS IN "PROCESSING MODE IS SEQUENTIAL") ARE NEW IMPLEMENTATIONS WHICH INVOLVE ONLY IGNORING THE CONSTRUCT.

OF THE WORDS WHICH ARE NEW AND VALID ON B6700, "OC" IS AN ABBREVIATION FOR "OCCURS", "VA" IS AN ABBREVIATION FOR "VALUE", "VALUES" IS AN ALTERNATIVE FOR "VALUES", "OTHERWISE" IS AN ALTERNATIVE FOR "ELSE", AND "UNIT" IS TREATED AS AN ALTERNATIVE FOR

"REEL". THE CONFIGURATION SECTION OF THE ENVIRONMENT DIVISION IS NOW OPTIONAL. ITS PARAGRAPHS ARE OPTIONAL, BUT MUST BE OMITTED IF THE SECTION HEADER IS OMITTED.

THE SOURCE COMPUTER STATEMENT TAKES THE FORM:

SOURCE-COMPUTER. <COMMENT ENTRY>.

THE COMPUTER NAME MAY BE ANY SINGLE WORD WHERE REQUIRED IN OBJECT-COMPUTER.

MEMORY SIZE MAY BE DESCRIBED AS AN INTEGER FOLLOWED BY "CHARACTERS". THE INTEGER PROVIDED IS DIVIDED BY SIX TO DETERMINE THE MEMORY SIZE TO BE USED BY THE SORT.

IF THE FILE-CONTROL HEADER IN THE INPUT/OUTPUT SECTION DOES NOT BEGIN IN COLUMN EIGHT, A WARNING IS GIVEN.

FILE-CONTROL ACCEPTS THE CONSTRUCTS "PROCESSING MODE IS SEQUENTIAL", "FOR MULTIPLE REEL", AND "FOR MULTIPLE UNIT" (WITH ONLY THE FIRST WORD BEING REQUIRED). AS THESE CONSTRUCTS ARE AUTOMATIC ON THE B6700, THEY ARE TREATED AS COMMENTS BY THE COMPILER.

THE IDENTIFICATION DIVISION HAS BEEN MODIFIED TO GIVE A WARNING WHEN THE PROGRAM-ID IS MISSING OR INCORRECTLY SPELLED. THE PARAGRAPHS OF THE IDENTIFICATION DIVISION OTHER THAN "PROGRAM-ID" MAY NOW APPEAR IN ANY ORDER.

D0015 DYNAMIC PROCEDURES IN ESPOL - 05-26-72

A NEW CONSTRUCT HAS BEEN IMPLEMENTED IN ESPOL TO PERMIT DECLARING PROCEDURES WITHOUT ALLOCATING MEMORY CELLS FOR THE PROCEDURE. THIS CONSTRUCT IS CALLED A DYNAMIC PROCEDURE. IT IS INDICATED BY PLACING PARENTHESES AROUND THE PROCEDURE NAME IN THE PROCEDURE DECLARATION AND FORWARD DECLARATION, IF ANY.

FOR EXAMPLE:

```
PROCEDURE (P);
```

```
BEGIN

    COMMENT HERE IS THE PROCEDURE BODY;

END;
```

CODE FOR THE PROCEDURE WILL BE COMPILED AND A SEGMENT DESCRIPTOR, IF NECESSARY, WILL BE ASSIGNED. THERE WILL NOT BE ANY STACK LOCATION ASSIGNED FOR P, SO IT CANNOT BE INVOKED DIRECTLY. ATTEMPTING TO DO SO WILL CAUSE A SYNTAX ERROR. ALSO, SINCE THERE IS NO LOCATION FOR P IT MAY NOT BE ADDRESS EQUATED TO ANY VARIABLE OR HAVE ANY VARIABLE ADDRESS EQUATED TO IT. DYNAMIC PROCEDURES MAY NOT BE EXTERNAL, GLOBAL, OR REPLACEABLE. TO INVOKE A DYNAMIC PROCEDURE, THE CONSTRUCT "MAKEPCW" HAS BEEN CREATED. THIS WILL MAKE AN APPROPRIATE PCW FOR EXECUTING THE PROCEDURE. THIS MAY BE USED IN ONE OF TWO WAYS:

```
WORD W;

W:=MAKEPCW(P);

GO TO W;
```

WILL CAUSE A BRANCH TO THE PROCEDURE CODE.

```
PROCEDURE Q; NULL;

WORD W=Q;

W:=MAKEPCW(P);
```

THIS CAUSES THE PCW FOR P TO BE PLACED IN THE STACK CELL RESERVED FOR PROCEDURE Q. THEN, INVOKING PROCEDURE Q ACTUALLY INVOKES PROCEDURE P.

THE PURPOSE OF DYNAMIC PROCEDURE IS TO AVOID HAVING TO MAKE PCWS FOR PROGRAMS CONTAINING A LARGE NUMBER OF PROCEDURES WHERE ONLY A FEW OF THOSE PROCEDURES ARE ACTUALLY USED DURING ANY GIVEN EXECUTION. HOWEVER, THE MAKEPCW CONSTRUCT IS PERMITTED FOR ANY PROCEDURE NAME, AND WILL MERELY PRODUCE A COPY OF THE EXISTING PCW FOR NON-DYNAMIC PROCEDURES.

D0017 ESPOL VECTOR MODE CHANGES - 06-19-72

TO REFLECT A HARDWARE CHANGE, ESPOL VECTOR MODE SYNTAX HAS BEEN GENERALIZED TO PERMIT REFERRING AND STORING INTO DECLARED VARIABLES. THESE VARIABLES MAY NOT BE DECLARED IN THE VECTOR MODE BLOCK AND MUST BE OPERANDS--NOT REFERENCES (CALL-BY-NAME PARAMETERS), SUBSCRIPTED VARIABLES, DESCRIPTORS, OR ACCIDENTAL ENTRIES--TO AVOID BEING INTERRUPTED. IN ADDITION, THE TYPE OF THE VARIABLE BEING STORED SHOULD MATCH THE TYPE OF THE VARIABLE BEING STORED INTO.

THE SYNTAX FOR THE VECTOR MODE CALL HAS ALSO BEEN CHANGED SLIGHTLY TO PRODUCE BETTER OBJECT CODE. THE CHANGE REGARDS THE ORDER OF SPECIFYING INCREMENTS. ALL INCREMENTS MUST BE LISTED FIRST, ENCLOSED IN BRACKETS, THEN THE VECTORS FOLLOW. THE NEW FORM OF THE STATEMENT IS AS FOLLOWS:

```
DO VECTORMODE ([INCREMENT, INCREMENT, INCREMENT],
   VECTORID, VECTORID, VECTORID, FOR COUNT)....
```

THE INCREMENTS, WHICH MAY BE EXPRESSIONS, APPEAR FIRST. IF FEWER INCREMENTS ARE SPECIFIED THAN VECTORS (OR THE INCREMENT PHRASE AND ITS ENCLOSING BRACKETS ARE OMITTED, WHICH IS PERMITTED), THE DEFAULT INCREMENT OF ONE IS ESTABLISHED FOR THOSE MISSING. THE NUMBER OF INCREMENTS SPECIFIED MAY BE THREE OR LESS AND SHOULD BE NO MORE THAN THE NUMBER OF VECTORS SPECIFIED. AS BEFORE, THE NUMBER OF VECTORS SPECIFIED MAY VARY FROM ONE TO THREE.

D0018 SYNTAX FOR ALGOL GLOBALS - 06-19-72

IN ALGOL AND DCALGOL, DECLARATIONS FOR GLOBAL SWITCH DECLARATIONS AND GLOBAL SWITCH FILE DECLARATIONS MAY NOW USE STANDARD SYNTAX WHEN BEING PROCESSED AS GLOBALS FOR A SEPARATE COMPILATION.

D0019 EVENT RELATED SYNTAX CHANGES - 05-18-72

IN ESPOL, "DSWAITANDRESET" MAY NOW BE USED IN PLACE OF "DSWAITNRESET".

"CAUSEANDRESET" MAY NOW BE USED IN PLACE OF "CAUSENRESET".

D0020 FORWARD INTERRUPT DECLARATIONS - 06-16-72

FORWARD INTERRUPT DECLARATIONS HAVE BEEN IMPLEMENTED IN ALGOL AND DCALGOL. THE SYNTAX IS:

```
<FORWARD INTERRUPT DECLARATION>::=INTERRUPT
   <INTERRUPT IDENTIFIER> FORWARD
```

ONCE DECLARED, THESE DECLARATIONS WILL HAVE THE SAME AFFECT AS A NON-FORWARD INTERRUPT DECLARED AT THAT POINT.

D0021 NEW LABEL FEATURES IN ESPOL - 06-16-72

TWO NEW RELATED FEATURES ARE IMPLEMENTED IN ESPOL.

1) IF A LABEL OCCURS WITH TWO TRAILING COLONS INSTEAD OF THE USUAL ONE COLON, THE SYLLABLE POINTER WILL BE ADJUSTED TO THE NEXT HALF WORD BOUNDARY BY EMITTING NO OP-S INTO THE CODE FILE. SUCH A LABEL WILL BE CALLED A HALF WORD ADJUSTED (HWA) LABEL.

2) THE "REAL" TRANSFER FUNCTION WILL ACCEPT AN HWA LABEL AS ITS ARGUMENT. THE VALUE LEFT IN THE STACK IS AN OPERAND SUITABLE FOR DOING A DYNAMIC BRANCH TO THE HWA LABEL USING THE GO TO <WORD VARIABLE> CONSTRUCT. A SYNTAX ERROR WILL OCCUR IF A LABEL ARGUMENT TO "REAL" IS NOT AN HWA LABEL.

EXAMPLE 1:

```
   LABEL HARDL, EASYL;

   LV:= IF HARD THEN REAL (HARDL) ELSE
      REAL (EASYL);

   GO TO WORD(LV);

HARDL::

EASYL::

   GO TO WORD (LV);
```

EXAMPLE 2:

```
FL: LV:=REAL (FL); IS ILLEGAL;
```

D0022 ISNT OPERATOR - 06-16-72

THE ISNT OPERATOR (LOGICAL EQUIVALENT TO NOT IS) HAS BEEN IMPLEMENTED IN ESPOL AS A RELATIONAL OPERATOR. FOR EXAMPLE,

```
A ISNT B
   WILL HAVE THE SAME EFFECT AS NOT (A IS B).
```

IS AND ISNT ARE EQUIVALENT TO EQL AND NEQ, RESPECTIVELY, WHEN USED IN POINTER RELATIONS.

D0023 UNTYPED QUEUES - 06-15-72

THIS ESPOL CHANGE PERMITS QUEUE ITEMS TO BE DECLARED WITHOUT TYPES. IF A SUBSEQUENT USE REQUIRES A TYPE, AN ERROR IS GIVEN. IF NO TYPE IS GIVEN, "NULL" MUST TERMINATE THE QUEUE DECLARATION.

D0024 PATCH IDENTIFICATION - 06-14-72

ALGOL HAS BEEN MODIFIED TO PERMIT PATCH LEVEL IDENTIFICATION SIMILAR TO ESPOL. IF THE ALGOL CARD FILE RECORD SIZE IS 16 WORDS (RECORDS CREATED BY SYSTEM/PATCH) OR THE TAPE FILE RECORD IS 15 WORDS, THEN CHARACTER POSITIONS 81-88 ARE ASSUMED TO CONTAIN A CHARACTER STRING. THE STRING WILL BE PRINTED ON THE FAR RIGHT HAND SIDE OF THE LISTING ON THE LINE FILE AND WILL BE WRITTEN ON THE ALGOL NEWTAPE FILE, IF ONE IS CREATED. THIS NEWTAPE FILE WILL HAVE A RECORD SIZE OF 15 WORDS.

IF THE INFORMATION IS NOT DESIRED, THE ALGOL TAPE AND NEWTAPE FILES SHOULD HAVE THEIR MAXRECSIZE ATTRIBUTE LABEL-EQUATED TO 14 WORDS.

SYSTEM/PATCH WILL PRODUCE THIS INFORMATION IF THE OPTION "MARK" IS SET WHEN RUNNING.

IN ADDITION, ESPOL WILL NOW PRINT OUT AN ASTERISK WHEN LISTING PATCHES OF THE CURRENT MARK NUMBER. THIS ASTERISK APPEARS BEFORE THE PATCH NUMBER IS PRINTED OUT ON THE RIGHT HAND SIDE OF THE

LISTING.

ON EBCDIC INPUT FILES, 88 CHARACTERS WILL NOW BE TRANSLATED SO THAT MARK LEVEL AND PATCH NUMBERS WILL BE TRANSLATED.

D0025 FILE AND TASK ATTRIBUTES - 06-19-72

THE FOLLOWING CHANGES HAVE BEEN MADE IN FILE AND TASK ATTRIBUTES:

1) SEVERAL MNEMONICS HAVE BEEN CHANGED FOR THE ALGOL, DCALGOL, AND ESPOL FILE AND DISKHEADER ATTRIBUTE "FILEKIND". "COMPILER" AND "DATAFILE" HAVE BEEN DELETED. THE VALUE OF "DATA" IS NOW 129, OF "XFORTRANSYMBOL" IS NOW 74, AND OF "XFORTRANCODE" IS NOW 42. THE MNEMONICS XDISKFILE (15), COMPILERCODEFILE (20), BINDERSYMBOL (94), SEQDATA (193), AND GUARDFILE (194) HAVE BEEN ADDED.

2) THE POINTER VALUED TASK ATTRIBUTE "STACKHISTORY" (30) HAS BEEN ADDED. "STACKHISTORY" PROVIDES INFORMATION ABOUT SEQUENCE NUMBERS OF PROCEDURES ENTERED WHEN A PROGRAM TERMINATES ABNORMALLY.

3) THE POINTER VALUE FILE ATTRIBUTE "PACKNAME" (39) HAS BEEN ADDED TO ALGOL, ESPOL, AND DCALGOL. "PACKNAME" IS USED TO FIND THE NAME OF A DISK PACK.

4) FILE AND TASK ATTRIBUTES ARE NOW IN SEPARATE VALUE ARRAYS WITHIN THE COMPILERS. THIS ARRAY IS SEARCHED ONLY WHEN A FILE OR TASK ARRAY IS EXPECTED. THIS MEANS THAT VARIABLES MAY HAVE THE SAME NAME AS FILE AND TASK ATTRIBUTES WITHOUT CONFLICT. FILE MNEMONICS ARE ALSO NOW IN A SIMILAR VALUE ARRAY.

DEFINES WILL BE EXPANDED BEFORE ANY OF THESE ARRAYS ARE SEARCHED.

D0026 ESPOL I-O - 06-19-72

THIS SYSTEM NOTE HAS BEEN CHANGED TO P0615.

D0027 PRINTER AND PUNCH-BACKUP - 09-09-72

INTRODUCTION

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

CREATING BACKUP TAPE FILES

THE USUAL METHOD OF CREATING BACKUP TAPE FILES IS THROUGH USE OF LABEL EQUATION CARDS:

```
<I> FILE LINE (KIND = PRINTER BACKUP TAPE)
<I> FILE PUNCH (KIND = PUNCH BACKUP TAPE)
```

THE USE OF THE ABOVE TYPE OF LABEL EQUATION CARDS WILL OVERRIDE THE SETTING OF "LPBDONLY" AND "CPBDONLY" DESCRIBED BELOW.

ANOTHER METHOD OF CREATING BACKUP TAPE FILES IS THROUGH USE OF THE SPO INPUT MESSAGE "OUMT". IF A PROGRAM REQUIRES A LINE PRINTER OR CARD PUNCH, BUT NONE IS AVAILABLE, THE OPERATOR CAN KEY IN "<MIX INDEX>OU MT", AND THE SYSTEM WILL AUTOMATICALLY LOOK FOR AN UPTAPE BACKUP TAPE THAT IS AVAILABLE. IF ONE IS AVAILABLE, THE FILE WILL BE WRITTEN ON THAT TAPE. IF AN UPTAPE BACKUP TAPE IS NOT AVAILABLE, THE SYSTEM WILL LOOK FOR AN AVAILABLE SCRATCH TAPE. IF ONE IS AVAILABLE, THE SYSTEM WILL DESIGNATE IT AS A "BACKUP" TAPE AND WRITE THE PRINTER OR PUNCH FILE ON IT.

NOTE: AT THE TIME OF THIS WRITING, KEYING IN "<MIX INDEX>OU MTNNN", WHERE NNN IS A MAGNETIC TAPE UNIT NUMBER, CANNOT BE USED IF AN UPTAPE BACKUP TAPE IS MOUNTED ON UNIT NNN.

CREATING BACKUP DISK FILES

PRINTER AND PUNCH FILES CAN BE DIRECTED TO BACKUP DISK BY USE OF CONTROL CARDS. FOR EXAMPLE:

```
<I> FILE PRNT (KIND = PRINTER BACKUP DISK)
<I> FILE PNCH (KIND = PUNCH BACKUP DISK)
```

ALL PRINT AND PUNCH FILES CAN BE DIRECTED TO BACKUP DISK BY INPUTTING "SO PBDONLY" AT THE SPO. DOING THIS WILL AUTOMATICALLY SET BOTH OPTIONS 4 (LPBDONLY) AND 14 (CPBDONLY). IF IT IS DESIRED TO SEND ONLY ONE OR THE OTHER TO BACKUP DISK, THEN EACH INDIVIDUAL OPTION CAN BE SET OR RESET.

ANOTHER WAY OF CREATING BACKUP DISK FILES IS THROUGH USE OF THE SPO INPUT MESSAGE "OUDK". IF A PROGRAM REQUIRES A LINE PRINTER OR CARD PUNCH, BUT NONE IS AVAILABLE, THE OPERATOR CAN KEY IN "<MIX INDEX> OU DK", AND THE SYSTEM WILL AUTOMATICALLY CREATE A BACKUP DISK FILE.

PRINTING BACKUP DISK FILES - THE "AP" MESSAGE

THE GENERAL METHOD OF PRINTING BACKUP DISK FILES IS THROUGH USE OF THE "AP" SPO INPUT MESSAGE:

```
AP <INTEGER>
```

FOR EXAMPLE, KEYING-IN:

```
AP 2
```

WILL CAUSE A MAXIMUM OF TWO COPIES OF AUTOPRINT TO BE ENTERED INTO THE MIX, IF THERE ARE TWO SCRATCH PRINTERS AND AT LEAST TWO PRINTER BACKUP DISK FILES TO BE PRINTED.

"AP-ING" A UNIT

LINE PRINTERS AND CARD PUNCHES CAN BE DESIGNATED AS "BACKUP" UNITS BY INPUTTING:

```
AP LP MMM
AP CP NNN
```

WHERE MMM AND NNN ARE THE UNIT NUMBERS OF THE LINE PRINTER AND CARD PUNCH, RESPECTIVELY.

AN "AP-ED" PRINTER WILL BE GIVEN PREFERENCE AND USED FOR PRINTING BACKUP FILES IF THE "AP" SETTING IS LESS THAN OR EQUAL TO THE NUMBER OF "AP-ED" PRINTERS. "NON-AP-ED" PRINTERS, REGARDLESS OF THE NUMBER OF PRINTER BACKUP DISK FILES, WILL NOT BE USED FOR PRINTING BACKUP FILES UNLESS "AP" IS GREATER THAN ZERO AND AN "AP-ED" PRINTER IS UNAVAILABLE (SAVED, IN LOCAL, IN USE BY A PROGRAM, ETC.).

IF THE "AP" SETTING IS LARGER THAN THE NUMBER OF "AP-ED" PRINTERS, THEN THE "AP-ED" UNITS WILL ALWAYS BE GIVEN PREFERENCE WHENEVER AUTOPRINT IS LOOKING FOR AN OUTPUT UNIT.

THE MAXIMUM NUMBER OF COPIES OF AUTOPRINT ALLOWED TO RUN IN THE MIX AT ANY ONE TIME IS DETERMINED BY WHICHEVER IS LARGER: THE NUMBER OF "AP-ED" PRINTERS OR THE SETTING OF "AP".

THE MAXIMUM COPIES OF AUTOPUNCH ALLOWED TO RUN IN THE MIX AT ANY ONE TIME IS DETERMINED STRICTLY BY THE NUMBER OF "AP-ED" CARD PUNCHES.

IF AUTOBACKUP IS USING A "NON-AP-ED" UNIT AND IT HAS FINISHED OUTPUTTING A "TREE" (ALL THE FILES IN A DIRECTORY FOR ONE MIX INDEX SUCH AS BD/0003798), IT WILL LOOK TO SEE IF ANOTHER PROGRAM IN THE MIX NEEDS AN OUTPUT UNIT AT THAT TIME. IF SO, IT WILL RELEASE THE OUTPUT UNIT IT IS USING SO THAT THE PROGRAM REQUIRING IT CAN USE IT. IF NO OTHER PROGRAM REQUIRES THE OUTPUT UNIT AUTOBACKUP IS USING, THEN AUTOBACKUP WILL LOOK FOR ANOTHER "TREE" OF FILES TO OUTPUT WITHOUT FIRST RELEASING THE UNIT TO THE SYSTEM.

IF AUTOBACKUP IS USING AN "AP-ED" UNIT, IT WILL NOT RELEASE THAT UNIT TO THE SYSTEM UNTIL ALL BACKUP FILES ON DISK HAVE BEEN PRINTED OR PUNCHED.

THE "PB" MESSAGE.

BACKUP DISK AND TAPE FILES CAN ALSO BE PRINTED/PUNCHED THROUGH USE OF THE "PB" SPO INPUT MESSAGE. FOR EXAMPLE:

```
PB MT20
```

WOULD CAUSE AUTOBACKUP TO BE FIRED UP AND MT20 TO BE REWOUND, THEN THE BACKUP FILES ON THE TAPE WOULD BE PRINTED AND/OR PUNCHED.

PB <MIX INDEX>

WOULD CAUSE THE TREE(S) OF FILES ASSOCIATED WITH <MIX INDEX> TO BE PRINTED AND/OR PUNCHED.

PB <MIX INDEX> LP

WILL CAUSE ANY PRINTER BACKUP DISK FILES ASSOCIATED WITH <MIX INDEX> TO BE PRINTED, BUT LEAVING ANY PUNCH BACKUP DISK FILES ON DISK WITHOUT PUNCHING THEM.

PB <MIX INDEX> CP

WILL CAUSE ANY PUNCH BACKUP DISK FILES ASSOCIATED WITH <MIX INDEX> TO BE PUNCHED, BUT LEAVING ANY PRINTER BACKUP DISK FILES ON DISK WITHOUT PRINTING THEM.

"PB-ING" FORCES AT LEAST ONE COPY OF AUTOBACKUP INTO THE MIX, REGARDLESS OF THE SETTING OF "AP" OR THE NUMBER OF "AP-ED" UNITS. THEREFORE, IF ONE INPUTS:

PB 3594 LP

THEN AUTOPRINT WILL BE FIRED UP, AND IT WILL FIRST LOOK FOR AN "AP-ED" PRINTER. IF NONE IS AVAILABLE, IT WILL LOOK FOR A SCRATCH PRINTER. IF NONE IS AVAILABLE, IT WILL HANG IN THE MIX WAITING FOR A PRINTER TO BECOME AVAILABLE.

BACKUP TAPE FILES

A BACKUP FILE ON TAPE IS LABELED "BACKUP"/<FILE NAME>, WHERE <FILE NAME>IS THE NAME OF THE FILE AS SPECIFIED BY THE PROGRAM CREATING IT. BACKUP TAPES MAY BE WRITTEN AS MULTIREEL FILES AND AS MULTIFILE REELS. THE SYSTEM WILL INTERMIX PRINTER AND PUNCH BACKUP FILES ON A BACKUP TAPE. ONE COPY OF AUTOBACKUP WILL THEN PRINT OR PUNCH EACH INDIVIDUAL FILE AS IT IS ENCOUNTERED.

BACKUP DISK FILES

THE GENERAL FORMAT OF PRINTER BACKUP DISK FILE NAMES IS:

BD/<MIX INDEX>/<FILE NAME>

THE GENERAL FORMAT OF PUNCH BACKUP DISK FILE NAMES IS:

BP/<MIX INDEX>/<FILE NAME>

IF MORE THAN ONE BACKUP DISK FILE IS CREATED BY A PROGRAM, THE SYSTEM CREATES A "TREE" OF FILES IN THE DISK DIRECTORY "BD/<MIX INDEX>". FOR EXAMPLE:

BD/0000365/000DIAGNOSTICS
BD/0000365/001LINE
BD/0000365/002PRNT

BACKUP FILE STRUCTURE

SYSTEM BACKUP FILES ARE VARIABLE-LENGTH RECORD, FIXED-LENGTH BLOCK FILES. EACH BLOCK IS 300 WORDS LONG. WITHIN A BLOCK, EACH LOGICAL RECORD IS COMPOSED OF ONE CONTROL WORD FOLLOWED BY ZERO OR MORE WORDS OF DATA. A TERMINAL CONTROL WORD OF ALL ZEROES INDICATES THAT THERE ARE NO MORE RECORDS IN THE PRESENT BLOCK.

NOTE: AUTOBACKUP WILL NO LONGER RECOGNIZE THE OLD FORMAT BACKUP FILES, I.E., 150 WORDS PER BLOCK. IF IT BECOMES NECESSARY TO PRINT SUCH A FILE, THE PROGRAM SYSTEM/BACKUP SHOULD BE USED INSTEAD OF AUTOBACKUP.

EACH CONTROL WORD IS DIVIDED INTO SPECIFIC FIELDS:

FIELD	CONTENTS
47:28	IDENTICAL WITH THE CORRESPONDING PORTION OF AN I-O CONTROL WORD.
19:3	CHARACTER COUNT RESIDUE FOR THE DATA RECORD (IF THE RECORD TO BE PRINTED CONSISTS OF COMPLETE WORDS, THE VALUE OF THE FIELD WILL BE ZERO.)
16:17	WORD COUNT FOR THE FOLLOWING DATA IN THE RECORD IN FULL WORDS, NOT COUNTING THE

FIELD	CONTENTS
	CONTROL WORD.

THE VERY FIRST RECORD OF THE FILE IS A CONTROL RECORD CONTAINING INFORMATION WHICH IS NOT PRINTED OR PUNCHED. THIS FIRST RECORD IS, MINIMALLY, 12 WORDS LONG, EXCLUDING THE CONTROL WORD. THE FIRST WORD OF THE FIRST RECORD IS THE CONTROL WORD.THE THIRD WORD CONTAINS THE FOLLOWING:

FIELD	CONTENTS
47:1	1 IF FORMS ARE REQUIRED, 0 IF FORMS ARE NOT REQUIRED.
4:5	THE UNIT TYPE OF THE BACKUP FILE. E.G., 11 (DECIMAL) IF CARD PUNCH, 22 IF LINE PRINTER.

IN ADDITION, IF A FORMS MESSAGE WAS SPECIFIED WHEN THE FILE WAS CREATED, THIS INFORMATION IS CONTAINED IN AS MANY WORDS AS ARE PERMITTED AND REQUIRED, STARTING AT WORD 13 OF THE FIRST RECORD.

D0028 B6700 DISK PACK SOFTWARE - 08-15-72

THIS DOCUMENT DESCRIBES THE OPERATING CHARACTERISTICS AND LANGUAGE EXTENSIONS BROUGHT ABOUT BY THE ADDITION OF MOVABLE HEAD DISK PACK HARDWARE TO THE B6700 SYSTEMS CONFIGURATION. IT ALSO DESCRIBES BOTH THE BURROUGHS COMMON INTERCHANGE CAPABILITY AND THE B6700 HEAD-PER -TRACK DISK COMPATIBLE OR "NATIVE" MODE OF OPERATION. THIS SUPPLANTS ALL PREVIOUS DISK PACK DOCUMENTATION; AND WHERE DISPARITIES BETWEEN IT AND PREVIOUS VERSIONS OCCUR, THIS SHOULD BE TAKEN AS THE AUTHORITY. MOST FUNCTIONS FOR DISK PACKS ARE IDENTICAL BETWEEN "INTERCHANGE" AND "NATIVE" MODE OF OPERATION. WHERE DIFFERENCES OCCUR OR A FUNCTION IS AVAILABLE FOR ONLY ONE MODE OF OPERATION, THIS FACT WILL BE NOTED PARENTHETICALLY. SEE THE LAST PAGE FOR MNEMONIC DEFINITIONS.

INITIALIZATION AND RECONFIGURATION

DUE TO THE DESIGN OF DISK PACKS, AN INITIAL SET OF ADDRESSES

DETERMINING SECTOR LAYOUT MUST BE CREATED PRIOR TO USE. THIS FUNCTION IS PERFORMED BY THE USER ON SITE. INITIALIZATION CAN PROCEED INDEPENDENTLY OF OTHER MCP FUNCTIONS APART FROM NORMAL SYSTEM INTERFERENCES BUT WILL TAKE APPROXIMATELY 10 MINUTES TO COMPLETE. IF THE DISK PACK TO BE INITIALIZED IS CONNECTED TO A 2 X N CONFIGURATION, THEN INITIALIZATION CAN BE PERFORMED INDEPENDENTLY OF OTHER PACK USAGE; OTHERWISE, THE CONTROLLER IS BUSY DURING THE PROCESS. THE INITIALIZATION PROCESS IS INITIATED BY THE "IV" OPERATOR-INPUT MESSAGE. (SEE APPENDIX B FOR SYNTAX OF SYSTEM INPUT MESSAGES.) INITIALIZATION IS ACCOMPLISHED IN THREE PHASES: INITIALIZE, VERIFY, AND CONFIGURE.

INITIALIZE

THE SYSTEM WRITES ADDRESSES AND SECTOR BOUNDARIES SEQUENTIALLY BY SECTOR, SURFACE, AND CYLINDER THROUGHOUT THE PACK. AFTER THE BASIC PACK STRUCTURE HAS BEEN DETERMINED, THE FUNCTIONAL AREAS ARE DELINEATED.

VERIFY

THE SYSTEM SCANS AN INITIALIZED PACK, CYLINDER, OR TRACK SEEKING PARITY AND ADDRESS ERRORS. A MAXIMUM OF FIVE ERRONEOUS SECTORS PER CYLINDER MAY BE RELOCATED TO SPARE SECTORS. RELOCATION IS PERFORMED WITHOUT USER INTERVENTION. IF MORE THAN FIVE ERRONEOUS SECTORS ARE FOUND IN A GIVEN CYLINDER, THEN THE SYSTEM WILL AUTOMATICALLY CREATE AN "XD-AREA" FOR THE REMAINING ERROR SECTORS. THE MASTER AVAILABLE TABLE IS CONSTRUCTED CONTAINING ALL PACK SPACE EXCEPT NON-RELOCATABLE ERROR SECTORS.

CONFIGURE

A DIRECTORY IS BUILT AND LABEL DATA IS WRITTEN ON THE PACK. (REFER TO APPENDIX C FOR LABEL FORMATS AND APPENDIX D FOR DIRECTORY FORMATS.) THE PACK IS THEN RECOGNIZED AS AVAILABLE TO THE SYSTEM.

A PACK MAY BE NAMED BY AN IDENTIFIER OF UP TO 17 CHARACTERS VIA THE "RC" MESSAGE OR MAY BE LEFT UNNAMED AS A SYSTEM RESOURCE PACK ("NATIVE" MODE ONLY). (SEE APPENDIX B FOR THE SYNTAX OF DISK PACK MESSAGES.) AN "RC" MESSAGE FOR A PACK HAS THE EFFECT OF ELIMINATING ALL FILES FROM THE DIRECTORY ON THAT PACK AND, THUS, IS NOT A MEANS OF CHANGING THE PACK NAME AND LEAVING THE FILES INTACT. THIS MEANS WILL BE PROVIDED IN A FUTURE SOFTWARE RELEASE.

PROGRAMS SELECT THE DISK PACK THEY WISH TO USE VIA THE PACKNAME FILE ATTRIBUTE. THIS ATTRIBUTE IS ASSIGNED THE (UP TO 17 CHARACTERS) NAME OF THE DISK PACK ON WHICH THE REFERENCED FILE IS TO BE FOUND. WHEN THIS ATTRIBUTE IS UNSPECIFIED, THEN THE "NATIVE" MODE "SYSTEM RESOURCE PACK" IS SELECTED. (SEE APPENDIX H FOR NAME CONVENTIONS.)

FILE NAMES FOR "INTERCHANGE" MODE ARE RESTRICTED TO ONE IDENTIFIER OF A MAXIMUM OF EIGHT CHARACTERS IN LENGTH (IDENTIFIERS NOT MEETING THIS CRITERION ARE TRUNCATED). "NATIVE" MODE IMPLEMENTATION ALLOWS THE FULL FOURTEEN NAME, SEVENTEEN CHARACTERS PER NAME IDENTIFIER STRUCTURE AVAILABLE FOR THE HEAD-PER-TRACK DIRECTORY STRUCTURE.

PACK RECOGNITION AND FILE OPENING

AFTER A PACK HAS BEEN MOUNTED AND THE SYSTEM DETECTS A CHANGE IN STATUS OF THE DISK PACK DRIVE FROM NOT READY TO READY, THE LABEL IS READ AND ANALYZED TO UPDATE THE PERTINENT TABLES IN CORE MEMORY.

IF THE DRIVE DOES NOT HAVE "WRITE ENABLE" ON, THE PACK IS MARKED "W/L" AND AVAILABLE FOR INPUT ONLY. IF THE PACK LABEL HAS THE SAME <PACK NAME> AND PHYSICAL SERIAL NUMBER AS ANOTHER PACK CURRENTLY ON-LINE, THE NEW PACK WILL BE MARKED "DUP SN" AND NOT ACCEPTED BY THE SYSTEM IF ANY OF THE FOLLOWING ARE TRUE:

1. INTERCHANGE PACKS WITH THE SAME SERIAL NUMBER.
2. CONTINUATION PACKS (NATIVE MODE) WHICH HAVE THE SAME BASE PACK.

3. BASE PACKS WITH THE SAME SERIAL NUMBER.

A FILE INTERNAL TO A PROGRAM CAN BE EQUATED TO A DESIRED EXTERNALLY-NAMED PACK THROUGH PROGRAMMING LANGUAGE FILE DESCRIPTION OR BY CONTROL CARD FILE EQUATION. IN ADDITION TO THE FILE ATTRIBUTES SUCH AS KIND, FILETYPE AND AREAS, WHICH HAVE THEIR USUAL MEANING, (KIND IS SET TO DISK PACK) ADDITIONAL ATTRIBUTES HAVE BEEN IMPLEMENTED. A DESCRIPTION IS PROVIDED FOR THESE ATTRIBUTES.

SINGLEPACK (BOOLEAN, ATTRIBUTE #40)

THIS ATTRIBUTE SPECIFIES THAT THE AREAS OF AN OUTPUT FILE ARE TO BE ASSIGNED TO A SINGLE PACK. THE DEFAULT VALUE IS FALSE WHICH PROVIDES FOR DISTRIBUTION OVER MULTIPLE PACKS, THE NUMBER OF PACKS BEING THE NUMBER PRESENT WHEN THE FILE IS OPEN. IF AN ADDITIONAL MEMBER OF THE PACK SET IS MOUNTED SUBSEQUENT TO FILE OPEN, THEN IT WILL BE INSERTED INTO THE ROTATION SCHEME OF ALLOCATION.

CYLINDERMODE (BOOLEAN, ATTRIBUTE #41)

WHEN TRUE, THIS ATTRIBUTE PROVIDES FOR ASSIGNMENT OF SPACE BY CYLINDERS; I.E., SPACE ASSIGNED TO EACH AREA LIES WITHIN THE BOUNDS OF A SINGLE CYLINDER. DEFAULT VALUE IS FALSE WHICH PROVIDES SPACE ASSIGNMENT WITHOUT REGARD FOR CYLINDER BOUNDS BUT RATHER ON A SPACE AVAILABLE BASIS. AREASIZE MUST NOT BE GREATER THAN CYLINDER SIZE WHEN CYLINDER MODE IS SET; AN ATTEMPT TO INTRODUCE THIS CONDITION WILL RESULT IN AN "OPEN ERROR #20" SYSTEM MESSAGE.

INTERCHANGE (BOOLEAN, ATTRIBUTE #66)

WHEN TRUE, THIS ATTRIBUTE INDICATES THAT THE FILE IS STORED OR IS TO BE STORED ON A BURROUGHS INTERCHANGE PACK.

TO ACCOMMODATE THE ADDITIONAL ATTRIBUTES, PROGRAMMING LANGUAGES HAVE BEEN EXTENDED. FOR PARTICULAR SYNTAX, REFER TO THE APPROPRIATE PROGRAMMING LANGUAGE DOCUMENTATION. THE FOLLOWING ARE EXAMPLES OF THESE EXTENSIONS:

A. ALGOL & DCALGOL:
```
FILE F (KIND = DISKPACK, SINGLEPACK, CYLINDERMODE,...);
FILE F2 (KIND = 17, SINGLEPACK = TRUE,
CYLINDERMODE = TRUE,...);
```

B. COBOL:
```
SELECT F ASSIGN TO ... DISKPACK SINGLE BY CYLINDER.
```

C. FORTRAN:
```
FILE 2 = F, UNIT = DISKPACK, SINGLEPACK, CYLINDERMODE, ...
```

FILE MAINTENANCE

THE ANALOGOUS ATTRIBUTE SYNTAX AND SEMANTICS APPLY TO THE CONTROL CARD MECHANISM. ADDITIONAL SYNTAX HAS BEEN IMPLEMENTED TO FACILITATE LIBRARY MAINTENANCE ON DISK PACKS, NAMELY:

<I> REMOVE <REMOVE STRING>

<I> CHANGE <CHANGE STRING>

<REMOVE STRING> ::= <FILE LIST> <KEY> <UNIT DESIGNATION>
<REMOVE STRING>, <FILE LIST> <KEY> <UNIT DESIGNATION>

<CHANGE STRING> ::= <CHANGE LIST> <KEY> <UNIT DESIGNATION>
<CHANGE STRING>, <CHANGE LIST> <KEY> <UNIT DESIGNATION>

<FILE LIST> ::= <FILE IDENTIFICATION> <FILE LIST>,
<FILE IDENTIFICATION>

<CHANGE LIST> ::= <FILE IDENTIFICATION> TO
<FILE IDENTIFICATION><CHANGE LIST>,<FILE IDENTIFICATION> TO
<FILE IDENTIFICATION>

<UNIT DESIGNATION> ::= <PACK IDIENTIFIER> PACK
<PACK IDENTIFIER>(<ATTRIBUTE LIST>) PACK
(<ATTRIBUTE LIST>) DISK

<KEY> ::= ON/FROM

<ATTRIBUTE LIST> ::= <ATTRIBUTE> <ATTRIBUTE LIST>, <ATTRIBUTE>

<ATTRIBUTE> ::= IC/UNITNO = <NUMBER>/KIND = PACK

IC SPECIFIES THAT THE PACK MUST BE INTERCHANGE FORMAT. OMISSION OF THIS ATTRIBUTE IMPLIES THAT THE PACK IS NATIVE FORMAT.

THE UNITNO ATTRIBUTE, WHEN USED, OVERRIDES ALL OTHER SPECIFICATIONS SUCH THAT THE COMMAND IS EFFECTIVE ONLY ON THE PACK MOUNTED ON THE PARTICULAR DRIVE DESIGNATED BY THE UNIT NUMBER GIVEN.

EXAMPLE:

<I> REMOVE MASTER, BACKUP ON COMPATIBLE (IC); END.

<I> REMOVE A/B FROM X(IC), A/B FROM DISK, A/B ON PACK; END.

<I> CHANGE PAYROLL/MASTESR TO PAYROLL/BACKUP ON ACCOUNTINGSPACK; END.

<I> CHANGE A/B TO X/Y ON MYPACK (KIND = PACK), A/B TO X/Y ON PACK; END.

NORMALLY, FILES ARE REMOVED FROM AND CHANGED ON ALL PACKS OF A GIVEN SET; BUT THERE MAY BE CONDITIONS WHICH REQUIRE MAINTENANCE ON ONLY A SUBSET OF A PACK SET; FOR EXAMPLE, IF THE BASE PACK BECOMES DAMAGED.

LIBRARY FACILITIES (NATIVE MODE ONLY)

LIBRARY MAINTENANCE FACILITIES ARE PROVIDED SIMILAR TO THOSE EXISTING FOR HPT FILES. SPECIFYING DISK PACKS HAS BEEN ADDED TO THE LIBRARY MAINTENANCE SYNTAX BY EXTENDING THE ALLOWABLE ATTRIBUTES FOR A SOURCE OR DESTINATION TO INCLUDE KIND = PACK.

EXAMPLE:

<I> COPY PAYROLL/BACKUP TO ARCHIVEPACK (KIND = PACK); END.

<I> COPY PAYROLL/BACKUP AS PAYROLL/MASTER FROM ARCHIVEPACK (KIND = PACK); END.

DIRECTORY SEARCH AND UPDATE

DIRECTORIES RESIDE ON THE PACK IN ALL CASES. (SEE APPENDIX D FOR DIRECTORY FORMATS.) EACH PACK DIRECTORY WITH ITS ASSOCIATED FILE HEADERS IS COMPLETE FOR FILES IN THAT PACK; AND, FOR MULTI-PACK FILES, IT CONTAINS SERIAL NUMBER REFERENCES TO OTHER PACKS CONTAINING AREAS OF THE SAME FILE THAT WERE CREATED OR UPDATED AT A TIME WHEN THE SUBJECT PACK WAS ON-LINE. THE COMPLETE PACK DIRECTORY AVAILABLE TO THE SYSTEM AT ANY TIME IS COMPRISED OF THE DIRECTORIES OF THE PACKS ON-LINE AT THAT TIME. EACH DIRECTORY SEARCH IS DONE SERIALLY.

AT OPEN TIME, A SUFFICIENT SEARCH IS MADE TO DETECT POSSIBLE PRESENCE OF DUPLICATE FILES.

THE OPERATOR IS NOTIFIED BY THE "DUP FILE" SYSTEM MESSAGE; HE IS EXPECTED TO ISOLATE THE DESIRED FILE EITHER BY LIBRARY MAINTENANCE IN SOME CASES, BY THE OPERATOR-INPUT MESSAGE "IL" IN MOST CASES, BY PHYSICAL OR LOGICAL REMOVAL OF ALL UNDESIRABLE FILES FROM THE SYSTEM, OR BY DISCONTINUING THE APPROPRIATE JOB.

OTHER CONFLICTS CONCERNING DUPLICATION MAY ARISE IF A NEW OUTPUT FILE IS BEING CREATED:

A. IF LOCKING A NEWLY CREATED OUTFILE IS ATTEMPTED WHILE OTHER FILES OF THE IDENTICAL NAME ARE RESIDENT ON SOME DISK PACK, A "DUP LIB" SYSTEM MESSAGE WILL RESULT WHICH CAN BE RESOLVED IN THE USUAL MANNER BY PREVIOUSLY SETTING AUTORM OPTION, USING THE OPERATOR-INPUT MESSAGE "RM" OR OTHER MEANS OF PHYSICALLY OR LOGICALLY REMOVING THE FILE FROM THE SYSTEM. THIS PRECLUDES DUPLICATE FILE IDENTIFIERS ON THE SAME PACK.

IF A PACK FILE IS OPENED INPUT, INPUT/OUTPUT, OR OUTPUT TO A PREVIOUSLY CREATED FILE, BUT THE NECESSARY DISK PACK IS NOT PRESENT, A "NO PACK" SYSTEM MESSAGE IS DISPLAYED. THE OPERATOR CAN EITHER MOUNT THE REQUIRED PACK OR DISCONTINUE THE APPROPRIATE JOB.

IF A FILE IS BEING OPENED OUTPUT, BUT NOT TO A PREVIOUSLY CREATED FILE, AND THE PACKNAME IDENTIFIER DOES NOT MATCH A PACK NAME FOR ANY DISK PACK KNOWN TO THE SYSTEM, A "REQUIRES" SYSTEM MESSAGE WILL BE GENERATED. IT CAN BE RESOLVED BY DISCONTINUING THE APPROPRIATE

JOB OR BY MAKING THE NECESSARY DISK PACK AVAILABLE.

IF A FILE, PREVIOUSLY OPEN, ATTEMPTS TO ACCESS AN AREA WHICH NECESSITATES CONTINUATION TO ANOTHER MEMBER OF THE PACK SET WHICH IS NOT PRESENT, THEN A "REQUIRES PK WITH SN#" MESSAGE WILL BE GIVEN. (SEE APPENDIX A FOR SYNTAX.) THE JOB MAY DISCONTINUE OR ELSE THE REQUIRED PACK WITH CORRECT SERIAL NUMBER MUST BE MOUNTED.

SPACE ALLOCATION

SPACE ALLOCATION IS DEPENDENT ON THE TWO ATTRIBUTES SINGLEPACK AND CYLINDERMODE. AS INDICATED PREVIOUSLY, CYLINDERMODE DEFINES INTRA-PACK ORGANIZATION WHEREAS SINGLEPACK RESTRICTS THE FILE FROM BEING DISTRIBUTED ACROSS MULTIPLE PACKS. IF THE SINGLEPACK ATTRIBUTE OF A FILE IS TRUE AND AN ATTEMPT IS MADE TO ALLOCATE AN ADDITIONAL AREA WHEN THERE IS NOT ENOUGH SPACE REMAINING TO ACCOMMODATE THE REQUEST, THEN A "PK SECTORS REQD" SYSTEM MESSAGE WILL BE DISPLAYED. THE OPERATOR MAY RESOLVE THE CONDITION BY DISCONTINUING THE APPROPRIATE JOB, BY REMOVING ONE OR MORE FILES FROM THE PACK, OR BY USING THE OPERATOR-INPUT MESSAGE "OU PK <NNN>." IF THE "OU" MESSAGE HAS BEEN GIVEN, THE SYSTEM WILL ATTEMPT TO ALLOCATE SPACE ON THE PACK SPECIFIED (WHEN DEALING WITH NAMED PACKS, THIS PACK MUST BE IDENTICALLY NAMED). ALL ADDITIONAL SPACE FOR THE FILE WILL NOW BE ALLOCATED FROM THIS PACK UNTIL ANOTHER "PK SECTORS REQD" CONDITION ARISES. IF THERE IS INSUFFICIENT SPACE AVAILABLE OF THE PACK NAME OR MODE OF THE NEW PACK ARE NOT IDENTICAL TO THE PREVIOUS PACK, THE "PK SECTORS REQD" MESSAGE WILL BE REDISPLAYED.

IF A FILE IS SPECIFIED AS A MULTI-PACK FILE, THEN, AS LONG AS POSSIBLE, SPACE IS ALLOCATED IN SUCH A MANNER THAT THE NUMBER OF AREAS ASSIGNED TO ANY PACK OF THE SET DOES NOT EXCEED THE NUMBER ASSIGNED TO ANY OTHER PACK BY MORE THAN ONE. SHOULD AN ATTEMPT BE MADE TO ALLOCATE ANOTHER AREA BUT THE NEXT PACK TO BE USED HAS NOT SUFFICIENT SPACE, THEN ASSIGNMENT IS MADE ON A SPACE-AVAILABLE BASIS TO SOME OTHER MEMBER OF THE SET. THIS IS DONE WITHOUT OPERATOR NOTIFICATION. SHOULD NO PACK IN THE SET CONTAIN SUFFICIENT SPACE AVAILABLE, THEN A "PK SECTORS REQD" SYSTEM MESSAGE IS GIVEN AND RESOLUTION IS PERFORMED BY MAKING ADDITIONAL PACKS

AVAILABLE, REMOVING FILES, OR DISCONTINUING THE APPROPRIATE JOB. (OF COURSE, UNIFORMITY OF AREA DISTRIBUTION AS DEFINED ABOVE IS NO LONGER GUARANTEED.)

SEEK, READ AND WRITE FUNCTIONS

SEEK, READ AND WRITE FUNCTIONS WILL OPERATE IN THE SAME MANNER AS THEY CURRENTLY DO FOR HPT DISK EXCEPT IN THOSE INSTANCES REQUIRING PHYSICAL ARM MOVEMENT. IN ALL CASES, THE DIFFERENCES WILL BE TRANSPARENT TO THE PROGRAMMING LANGUAGES.

SEEK

A SEEK FUNCTION OPERATES SIMILARLY TO HPT BY FILLING THE PROGRAMS BUFFER OR PREPARING THE BUFFER FOR A WRITE, WHICHEVER IS APPROPRIATE.

READ

A DISK PACK READ, AS FOR HPT, CAN TAKE EITHER OF THE TWO FOLLOWING FORMS:

A. A PHYSICAL INPUT OPERATION WHICH TRANSFERS DATA FROM THE DEVICE TO THE PROGRAMS BUFFER, FOLLOWED BY A MOVEMENT OF THE DATA FROM THE BUFFER TO THE WORK AREA, IF APPLICABLE.

B. A MOVEMENT OF PREVIOUSLY OBTAINED DATA FROM THE BUFFER TO THE WORK AREA.

IF THE RECORD BEING READ IS NOT IN CORE AND AVAILABLE TO THE PROGRAM, AN IMPLICIT SEEK IS PERFORMED, AND CONTROL IS NOT RETURNED TO THE PROGRAM UNTIL DATA IS MADE AVAILABLE TO THE PROGRAM.

WRITE

A WRITE FUNCTION OPERATES SIMILARLY TO HPT.

PACK REMOVAL

DUE TO THE POSSIBILITY OF MULTIPLE USERS OF A DISK PACK, POWERING DOWN TO PERMIT PACK REMOVAL IS CONTROLLED BY THE MCP. OPERATOR REQUEST FOR PERMISSION TO REMOVE A PACK IS ACCOMPLISHED VIA THE "PO" INPUT MESSAGE. (SEE APPENDIX B FOR SYNTAX.) ON RECEIPT OF A REMOVAL REQUEST, IF THE PACK IS NO LONGER IN USE, IT IS POWERED DOWN, OTHERWISE AN "INV KBD" RESPONSE RESULTS. MANUAL POWERING DOWN AND PACK REMOVAL WITHOUT SYSTEM OPERATOR DIALOGUE MAY RESULT IN A PACK INTEGRITY VIOLATION AND REQUIRE A RECOVERY ATTEMPT AS DISCUSSED BELOW.

SHOULD A DISK PACK BE MANUALLY POWERED DOWN WHILE THERE EXISTS OUTSTANDING I/O OPERATION REQUESTS, THEN THE FOLLOWING OCCURS:

A. A DISK PACK I/O ERROR MESSAGE WILL BE GIVEN.

B. ALL PROCESSES WHICH HAVE INITIATED USER I/O REQUESTS TO THE ABNORMAL DISK DRIVE WILL BE DISCONTINUED.

C. ALL PROCESSES WHICH HAVE OPEN FILES RESIDING ON THE ABNORMAL PACK WILL BE DISCONTINUED.

D. PROCESSES WHICH ARE INITIALIZING OR RECONFIGURING A PACK ON THE ABNORMAL DISK DRIVE WILL TERMINATE.

RECOVERY

AT THE TIME OF PACK RECOGNITION, THE INTEGRITY FLAG IS CHECKED, AND THE SYSTEM DETERMINES WHETHER THE PACK WAS PREVIOUSLY ACTIVE AND A FILE RECOVERY IS NECESSARY. FILES ARE CONSIDERED ACTIVE IF THE FILE HEADER CORE ADDRESS IS GREATER THAN ZERO.

FOR FILES WHICH WERE ACTIVE, RECOVERY ENTAILS RESETTING IN THE DISK PACK FILE HEADER THE NUMBER OF USERS, NUMBER OF OPEN-OUTPUT USERS AND HEADER CORE ADDRESS. (SEE APPENDIX E FOR FILE HEADERS.) FOR PACKS WITH SUSPECT FILES, A "PK CONTAINS ABNORMALLY CLOSED FILES" MESSAGE WILL BE DISPLAYED.

WHEN ALL FILE ABNORMALITIES HAVE BEEN RESOLVED, THE LABEL INTEGRITY

FLAG IS RESET; AND THE PACK IS RELEASED TO THE SYSTEM.

ON INTERCHANGE PACKS WHICH HAVE BEEN ABNORMALLY REMOVED FROM A B6700 SYSTEM, THE SPACE-AVAILABLE TABLE MAY BE INCOMPLETE AND INVALID. FOR THIS REASON, THE USER IS ADVISED TO BE CAUTIOUS ABOUT REMOVING A DISK PACK FROM THE SYSTEM.

A HALT/LOAD WITH DISK PACK FILES OPEN WILL NECESSITATE THE SAME FILE RECOVERY ACTION WHEN THE SYSTEM NEXT RECOGNIZES THE PACK.

DIRECTORY INFORMATION RETRIEVAL

THE CONTENTS OF DISK PACK DIRECTORIES CAN BE OBTAINED VIA THE OPERATOR INPUT MESSAGE "DIR <PACK INDICATOR>", WHICH WILL CAUSE SYSTEM/PACKLIST (INTERCHANGE) OR SYSTEM/PACKDIR (NATIVE) TO BE INITIATED. THE FORMER IS AN OPERATING SYSTEM ROUTINE, AS THE DIRECTORY FOR AN INTERCHANGE FORMAT PACK IS NOT ACCESSIBLE TO A USER PROGRAM. THE LATTER IS A USER PROGRAM WHICH IS SIMILAR IN STRUCTURE TO SYSTEM/LISTDIRECTORY AND PROCESSES THE PACK DIRECTORY BY OPENING THE MASTER DIRECTORY FOR THE PACK. THE NAME OF THIS FILE IS EITHER THE PACK NAME FOR NAMED PACKS OR "SYSTEMPACK" FOR SYSTEM RESOURCE PACKS.

FILE SECURITY (NATIVE MODE ONLY)

BOTH CLASS A AND CLASS B SECURITY AS DEFINED BY HPT IS AVAILABLE FOR DISK PACK FILES ON BOTH NAMED PACKS OR SYSTEM RESOURCE PACKS. THE ONLY IMPORTANT DIFFERENCE IS THAT WHEREAS FILES CREATED UNDER A USERCODE OR SYSTEM RESOURCE PACKS. THE ONLY IMPLEMENTATION FILES IS AVAILABLE FOR DISK PACK FILES ON BOTH NAMED PACKS IN THE HPT ARE LOCATED UNDER THE "USERCODE" DIRECTORY, ON DISK PACK, ONLY THE USERCODE IDENTIFIER IS APPENDED TO THE FILE. THE FILE "TEST/DATA", WHEN CREATED BY USER "STATRUN" ON HPT, WOULD ACTUALLY BE CALLED "USERCODE/STATRUN/TEST/DATA." THE SAME FILE PLACED ON DISK PACK WOULD BE TITLED "STATRUN/TEST/DATA." IT IS, THEREFORE, POSSIBLE FOR FILES CREATED WITHOUT USERCODES TO CONFLICT IN NAME WITH THOSE CREATED WITH USERCODES WHEN THE PRIME NAME HAPPENS TO CORRESPOND TO A USER CODE IDENTIFIER.

SYSTEM FUNCTIONS

THE EXISTING DISK PACK SOFTWARE DOES NOT PROVIDE FOR THE USE OF DISK PACKS FOR SUCH FUNCTIONS AS AUTOPRINT, CONTROL DECKS OR PROGRAM EXECUTIONS. THESE TYPES OF FILES CAN BE PLACED ON DISK PACKS ONLY VIA LIBRARY MAINTENANCE FUNCTIONS.

SORT

THE SORT INTRINSIC CAN BE CAUSED TO PLACE ITS WORK FILE ON DISK PACK BY APPROPRIATELY LABEL EQUATING IT. THE SORT USES TWO FILES: DISKC IS THE CONTROL FILE AND IS A RELATIVELY SMALL FILE, DISKF IS THE STRINGING FILE. BOTH SHOULD BE LABEL EQUATED IF DISK PACK RECOVERY IS DESIRED; OTHERWISE, DISKF, THE LARGER OF THE TWO FILES, IS SUFFICIENT.

XD

SPACE ON THE DISK PACK MAY BE XD-ED VIA THE XD KEYBOARD MESSAGE. ON AN INTERCHANGE PACK, THE FILE CREATED IS NAMED ADDDDDDD, WHERE DDDDDD IS THE BASE SEGMENT ADDRESS. NATIVE MODE PACKS CREATE FILES NAMED AS ON HPT DISK.

SCR DISK PACK FACILITIES

FILE DECLARATIONS FOR BADDISK PACK FILES AND A VERIFY VARIANT FOR PACKS ARE AVAILABLE IN SCR.

THE FILE DECLARATION IS OF THE FORM:

```
DISKPACK FILE <IDENTIFIER> = "<PACKID>/AD<NUMBER>";
```

PACKS AND BADDISK FOR NATIVE MODE PACKS. THE <NUMBER> WHERE <PACKID> IS THE PACK NAME FOR INTERCHANGE IS THE SIX DECIMAL DIGIT STARTING ADDRESS FOR THE FILE, SPECIFIED IN THE XD SPO INPUT MESSAGE.

THE VERIFY STATEMENT FOR PACKS IS OF THE FORM:

```
VERIFY DISKPACK FILE <IDENTIFIER> (<DISK TEST PARAMETERS>) OR
```

VERIFY <UNIT SPECIFIER> (<DISK TEST PARAMETERS>)

WHERE <UNIT SPECIFIER> CORRESPONDS TO A UNIT MNEMONIC OF "PK."

THE DESCRIPTIONS OF THE INDIVIDUAL TESTS WILL BE ADDED TO THE SCR MANUAL.

APPENDIX A

SYSTEM OUTPUT MESSAGES

<PK UNIT DESIGNATE> RECONFIGURE ABORTED

THIS MESSAGE IS A RESPONSE TO THE "RC" SYSTEM INPUT MESSAGE. IT OCCURS WHENEVER AN I/O ERROR IS DETECTED WHILE CONFIGURING A PACK.

<PK UNIT DESIGNATE> INITIALIZATION ABORTED

THIS MESSAGE IS A RESPONSE TO THE "IV" SYSTEM INPUT MESSAGE. IT OCCURS WHENEVER AN I/O ERROR IS DETECTED WHILE INITIALIZING A DISK PACK. RETRY IS NOT PROBABLE. THIS MAY OCCUR AT ANY TIME DURING THE INITIALIZATION PROCESS.

EXAMPLE: PK123 INITIALIZATION ABORTED.

<MIX-ID> FILE OPEN ERROR # 20: <FILE-ID>

THIS MESSAGE IS DISPLAYED WHENEVER AN ATTEMPT IS MADE TO OPEN A FILE SPECIFIED CYLINDERMODE AND AREASIZE IS GREATER THAN THE SIZE OF A CYLINDER.

EXAMPLE: 0132 FILE OPEN ERROR # 20: NUMAST/A

<MIX-ID> <FILE-ID> REQUIRED PK WITH SN# = <PHYSICAL SERIAL NUMBER>

THIS MESSAGE OCCURS WHEN A MULTI-PACK FILE, OPEN INPUT OR INPUT/OUTPUT, ATTEMPTS CONTINUATION TO ANOTHER MEMBER OF THE SET WHICH IS NOT PRESENT.

EXAMPLE: 0314 NUMAST REQUIRES PK WITH SN# = 123456

VALID KEYBOARD MESSAGES ARE OK AND DS.

<PK UNIT DESIGNATE> NAME REQUIRED

THIS IS A RESPONSE TO THE "RC" OR "IV" SYSTEM INPUT MESSAGE. (SEE APPENDIX B FOR SYNTAX.)

<PK UNIT DESIGNATE> SERIAL NUMBER REQUIRED

THIS IS A RESPONSE TO THE "RC" OR "IV" SYSTEM INPUT MESSAGE. (SEE APPENDIX B FOR SYNTAX.)

<PK UNIT DESIGNATE> MISSING ROW

THIS MESSAGE OCCURS WHEN A FILE ACCESSES A MEMBER OF A PACK SET WHICH DOES NOT CONTAIN ANY ROW ASSIGNED TO THE FILE AS PER THE BASE PACK FILE HEADER. THE FILE MOST PROBABLY WAS REMOVED FROM THE PACK VIA FILE MAINTENANCE CONSTRUCTS.

APPENDIX B

SYSTEM INPUT MESSAGES FOR DISK PACKS

1. IV <PK UNIT DESIGNATE><SUFFIX LIST>

   <SUFFIX LIST> ::= <SUFFIX>/<SUFFIX LIST>,<SUFFIX>

   <SUFFIX> ::= <MODE>/NAME = <PACKNAME>/SERIAL = <PHYSICAL SERIAL NUMBER>

   <MODE> ::= INTERCHANGE/IC/<EMPTY>

   <PACK NAME> ::= LEQ STRING WITH MAXIMUM LENGTH OF 17 GEQ

   <PHYSICAL SERIAL NUMBER> ::= LEQ INTEGER FROM 1 TO 999999 GEQ

THIS INPUT MESSAGE IS USED TO INITIALIZE A DISK PACK.

EXAMPLE: IV PK 213, NAME = NUDP, SERIAL = 123456

THE FOLLOWING RESPONSES ALSO APPLY:

   A) INVALID SYNTAX

   B) SERIAL NUMBER REQUIRED

   C) NAME REQUIRED

   D) PACK REJECTED, TOO MANY BAD SECTORS

   E) INITIALIZATION ABORTED

   F) PACK IS IN USE

   G) UNIT NOT AVAILABLE

2. RC <PK UNIT DESIGNATE><SUFFIX LIST>

   <SUFFIX LIST> ::= <SUFFIX>/<SUFFIX LIST>,<SUFFIX>

   <SUFFIX> ::= <MODE>/NAME = <PACK NAME>/SERIAL =

<DESIGNATE>/<TYPE> = <DESIGNATE>

<DESIGNATE> ::= <PHYSICAL SERIAL NUMBER>

<TYPE> ::= BP/BASEPACK

THIS INPUT MESSAGE IS USED TO RECONFIGURE A PACK. NAME AND SERIAL ARE OPTIONAL AND, WHEN OMITTED, THE EXISTING LABEL INFORMATION IS USED.

EXAMPLE: RC PK132 NAME = OLDP, SERIAL = 654321
RC PK81 NAME = B, SERIAL = 123
RC PK82 SERIAL = 55, BP = 123

NOTE:

BASEPACK = 123 INDICATES THIS PACK IS A CONTINUATION PACK WITH THE BASE PACK OF SERIAL NUMBER 123.

IF THE BASEPACK SYNTAX IS ATTEMPTED FOR THE CONFIGURATION OF AN INTERCHANGE PACK, THE CONFIGURATION WILL BE ABORTED WITH THE MESSAGE, "INVALID SYNTAX".

THE BASE PACK MUST BE PRESENT DURING THE CONFIGURATION OF CONTINUATION PACKS; IF NOT, THE CONFIGURATION WILL BE ABORTED WITH THE MESSAGE, "BASE PACK REQUIRED".

IF THE PACK NAME IS OMITTED FOR CONFIGURATION OF A CONTINUATION PACK, THE PACK WILL TAKE THE NAME OF THE BASE PACK. IF THE NAME IS SPECIFIED BUT NOT THE SAME AS THE BASE PACK, THE CONFIGURATION WILL BE ABORTED WITH THE MESSAGE, "INCORRECT PACK NAME".

THE RECONFIGURING OF A CONTINUATION PACK WILL MAP THE CONTINUATION PACK-S MASTER AVAILABLE TABLE ONTO THE BASE PACK-S MASTER AVAILABLE TABLE; THEREFORE, THE BASE PACK MUST BE WRITE ENABLE; IF NOT, THE RECONFIGURE WILL BE ABORTED WITH THE MESSAGE, "BASE PACK REQUIRED".

THE RECONFIGURING OF A CONTINUATION PACK WITH A SERIAL NUMBER THAT CURRENTLY EXISTS ON THE BASE PACK-S "PACK SET LINKAGE TABLE" WILL RESULT IN THE RECONFIGURE BEING ABORTED WITH THE MESSAGE, "EXISTING SERIAL NUMBER". THE EXCEPTION TO THE ABOVE IS IF THE CONTINUATION

PACK IS CURRENTLY A MEMBER OF THE SET IN GOOD STANDING. IN THIS CASE THE CONTINUATION PACKS TABLE WILL BE REWRITTEN IDENTICALLY AS BEFORE AND THE STATUS QUO MAINTAINED. THIS IS INTENDED FOR TWO PURPOSES: (1) TO ELIMINATE ACCIDENTALLY REMOVING THE CONTINUATION PACK FROM THE PACK SET, AND (2) TO ATTEMPT TO RECOVER THE PACK IF THE SYSTEM REFUSES TO ACCEPT THE PACK DUE TO SOME DATA DESTRUCTION IN THE PACK-S LABEL, OF AN OTHERWISE GOOD PACK.

ALL ABNORMAL TERMINATIONS OF "RC", EXCEPT THE "INVALID SYNTAX" ABORT, WILL LEAVE THE PACK UNLABELED.

THE FOLLOWING RESPONSES ALSO APPLY:

A) INVALID SYNTAX

B) SERIAL NUMBER REQUIRED

C) NAME REQUIRED

D) RECONFIGURE ABORTED

E) PACK IS IN USE

F) BASE PACK REQUIRED

G) INCORRECT PACK NAME

H) UNIT NOT AVAILABLE

I) EXISTING SERIAL NUMBER

3. PO <PK UNIT DESIGNATE>

   THIS MESSAGE IS USED TO REQUEST PERMISSION FROM THE SYSTEM TO POWER OFF THE INDICATED DISK DRIVE TO FACILITATE DISK PACK REMOVAL.

   EXAMPLE: PO PK 321

   THE FOLLOWING RESPONSE ALSO APPLIES:

   A) PACK IS IN USE

4. DIR <PACK INDICATOR>

<PACK INDICATOR> ::= <EMPTY>/<PACK NAME>/<PK UNIT DESIGNATE>

THIS INPUT MESSAGE INITIATES DIRECTORY INFORMATION RETRIEVAL. IF <PACK INDICATOR> IS EMPTY, THEN HEAD-PER-TRACK DISK DIRECTORY INFORMATION IS LISTED. IF <PACK NAME> IS USED, THEN ALL FILES RESIDING ON ALL PRESENT MEMBERS OF THE SET ARE LISTED. (INTERCHANGE ONLY.) IF <PK UNIT DESIGNATE> IS USED, A LISTING IS MADE OF ALL FILES FOR WHICH THE INDICATED PACK IS A BASE-PACK.

APPENDIX C

PACK LABEL FORMAT

POSITION	LENGTH (BYTES)	CONTENTS
000-003	004	"VOL1"
004-009	006	PACK SERIAL NUMBER
010	001	ACCESS CODE *
011-027	017	PACK IDENTIFICATION
028-029	002	SYSTEM-INTERCHANGE CODE NATIVE MODE =67 INTERCHANGE = 00
030	001	PACK CODE *
031-036	006	RESERVED
037-050	014	OWNERS IDENTIFICATION *
051-078	028	RESERVED
079	001	BLANK
080-083	004	"VOL2"
084-088	005	INITIALIZATION DATE
089-094	006	INITIALIZING SYSTEM NATIVE - 67MC <MARK DIGIT><LEVEL NO> INTERCHANGE=<SYSTEM SERIES> <OPERATING SYSTEM><VERSION>
095-102	008	DIRECTORY LINK NATIVE MODE LINKS TO PACK MASTER HEADER, INTERCHANGE LINKS TO FIRST DIRECTORY BLOCK.
103-110	008	MASTER AVAILABLE TABLE LINK
111-118	008	AVAILABLE TABLE LINK NATIVE MODE UNUSED
119	001	INTEGRITY FLAG
120-125	006	ACTUAL ERROR COUNT *
126-131	006	MISSING SECTOR COUNT *

POSITION	LENGTH (BYTES)	CONTENTS
132-179	048	RESERVED
180-359	180	RESERVED FOR SECURITY INFORMATION

* CURRENTLY UNUSED.

APPENDIX D

INTERCHANGE DIRECTORY RECORD FORMAT

POSITION	LENGTH (DIGITS)	CONTENTS
000-007	008	FORWARD LINK
008-015	008	BACKWARD LINK
016-023	008	ADDRESS OF THIS SECTOR
024	001	MARKER
025-027	003	RESERVED
028	001	HEXADECIMAL "F" (1111)
029	001	RESERVED
----DIRECTORY ENTRIES----		
030-037	008	ADDRESS OF HEADER
038-041	004	LENGTH OF HEADER
042-057	016	NAME
058	001	VALIDITY FLAG
059	001	RESERVED
060-359	300	10 OR MORE DIGIT ENTRIES

APPENDIX E

INTERCHANGE FILE HEADER FORMAT

POSITION	LENGTH	CONTENTS
000-005	006	CORE ADDRESS
006-013	008	SELF POINTER
004-017	004	HEADER SIZE
018-019	002	FILE TYPE
020-031	012	RESERVED
032-037	006	RECORD SIZE
038-040	003	RECORDS-BLOCK
041-049	009	BLOCK SIZE
050-055	006	BLOCKS-AREA
056-061	006	SECTORS-AREA
062-065	004	AREAS REQUESTED
066-069	004	AREA COUNTER
070-079	010	END-OF-FILE POINTER
080-081	002	RECORD FORMAT
082-089	008	LINK TO USER HEADER
090-093	004	RESERVED
094-098	005	CREATION DATE
099-103	005	LAST ACCESS DATE
104-104	005	SAVE FACTOR (PURGE DATE)
109-128	020	RESERVED
129-161	033	RESERVED FOR SECURITY INFORMATION
162-163	002	OPEN TYPE AND PERMANENT FLAG
164-165	002	NUMBER OF USERS
166-167	002	NUMBER OPEN OUTPUT
168-199	032	RESERVED
200-207	008	FIRST AREA LINK
208-359	152	19 MORE AREA LINKS

APPENDIX F

GLOSSARY

BASE PACK (INTERCHANGE) -- THAT PACK WHICH CONTAINS THE FIRST AREA ASSIGNED TO THE FILE AND THROUGH WHICH ALL MULTI-PACK REFERENCING IS ACCOMPLISHED.

BASE PACK (NATIVE MODE) -- THAT PACK WHICH CONTAINS THE DIRECTORY AND AVAILABLE TABLES FOR ALL MEMBERS OF A PACK SET.

BURROUGHS INTERCHANGE PACK -- A DISK PACK WHICH HAS BEEN INITIALIZED TO A MULTISECTOR FORMAT AS DEFINED BY BURROUGHS CORPORATION. THIS FORMAT ALLOWS DISK PACK COMPATIBILITY AMONG ALL BURROUGHS SYSTEMS FROM B1700 TO B7700, INCLUSIVE.

CYLINDER -- A SET OF TRACKS (ONE PER SURFACE) WHICH ARE ACCESSIBLE FROM A SINGLE POSITION OF THE RECORDING ARM. CYLINDERS ARE NUMBERED 0-405, FROM THE OUTSIDE EDGE OF THE PACK TOWARDS THE CENTER.

INTEGRITY FLAG -- INFORMATION WITHIN A DISK PACK LABEL WHICH INDICATES THAT THE FLAG IS NOW IN USE OR WAS IN USE WHEN IT WAS ABNORMALLY REMOVED FROM THE SYSTEM.

PACK -- EITHER A B9974-1 OR A B9974-4 REMOVABLE DISK PACK CONTAINING 11 PLATTERS.

PACK SET -- A COLLECTION OF PACKS WHICH ARE RELATED BY THE RESIDENCY OF SOME MULTI-PACK FILE WHICH HAS AT LEAST ONE AREA ALLOCATED TO EACH MEMBER.

PLATTER - ONE OF 11 PHYSICAL RECORDING MEDIA IN A PACK. EACH PLATTER CONTAINS TWO SURFACES EXCEPT FOR THE TOP AND BOTTOM

PLATTERS OF A PACK WHICH MAKES USE OF ONLY THE INSIDE SURFACES. PLATTERS ARE NUMBERS FROM ZERO TO 10 STARTING AT THE TOP.

SECTOR -- A CONTIGUOUS AREA OF 180 EBCDIC BYTES OR 30 B6700 WORDS. SECTOR IS THE SMALLEST DATA AREA ADDRESSABLE BY THE DISK PACK DRIVE CONTROLLER. THERE ARE 33 OR 60 SECTORS PER TRACK.

SPARE SECTOR -- A SECTOR NOT AVAILABLE TO USER PROGRAMS DIRECTLY BUT ALLOCATED BY THE SYSTEM TO REPLACE BAD DATA SECTORS DISCOVERED DURING THE INITIALIZATION PROCESS. THERE ARE FIVE SUCH SECTORS PER CYLINDER, OCCUPYING THE LAST FIVE SECTORS OF TRACK ZERO OF EACH CYLINDER.

SURFACE -- ONE SIDE OF A PLATTER USED AS A RECORDING MEDIUM. THERE ARE TWO SURFACES PER PLATTER, EXCEPT THE TOP PLATTER WHICH DOES NOT USE THE UPPER SURFACE AND THE BOTTOM PLATTER WHICH DOES NOT USE THE LOWER SURFACE. EACH SURFACE CONTAINS 406 TRACKS AND ARE NUMBER 0-19 FROM THE TOP DOWNWARD.

TRACK -- ALL SECTORS OF A SINGLE SURFACE ACCESSIBLE FROM A SINGLE POSITION OF THE RECORDING ARM. THERE ARE 33 OR 60 SECTORS PER TRACK AND 406 TRACKS PER SURFACE. TRACKS HAVE THE SAME NUMBER AS THEIR ASSOCIATED CYLINDER. TRACKS ARE NUMBERED FROM 0-19, FROM THE TOP DOWNWARD.

APPENDIX G

FILE ATTRIBUTES

THE FOLLOWING FILE ATTRIBUTES FUNCTION SLIGHTLY DIFFERENT WHEN APPLIED TO DISK PACKS THAN TO HEAD-PER-TRACK DISK.

UNITNO

THIS ATTRIBUTE IS AN ERROR FOR HPT FILES. WHEN SPECIFIED FOR DISK PACK FILES, IT CAUSES THE SYSTEM TO USE ONLY THE PACK ON THE SPECIFIED UNIT WHEN LOOKING FOR A FILE.

ROWADDRESS

THIS ATTRIBUTE RETURNS THE DISK ADDRESS OF A FILE ROW FOR HPT FILES. FOR DISK PACK FILES, IT RETURNS A 48-BIT OPERAND WHICH IS FURTHER DIVIDED AS FOLLOWS:

BITS	NAME	DESCRIPTION
47:1	BASEPACKF	VALUE = 1 IF THIS IS THE FIRST ROW ASSIGNED TO THE FILE.
44:1	INITIALROWF	VALUE = 1 IF THIS IS THE FIRST ROW ALLOCATED TO THIS PARTICULAR PACK ASSIGNED TO THE FILE.
43:22	SERIALNUMBERF	THIS IS THE SERIAL NUMBER OF THE PACK ON WHICH THIS ROW IS ALLOCATED.
21:22	BASEADDRESSF	THIS IS THE BASE ADDRESS OF THE ROW.

FLEXIBLE

NOW ALLOWED FOR DISK PACK.

DUPLICATED

NOT ALLOWED FOR DISK PACK.

APPENDIX H

NAME CONVENTIONS FOR DISK PACKS (KIND = DISKPACK)

<PACKNAME>	<FILENAME>	PACK NAME	RESULTING FILE NAME ON PACK
PN	A	IC	A ON PN
		NP	A ON PN
		SR	(NOT OBTAINABLE)
PN	A-B	IC	B ON PN
		NP	A-B ON PN
		SR	(NOT OBTAINABLE)
PN	A-B-C...-N	IC	N ON PN
PN	A-B	IC	B ON PN
	MAX 14 NAMES)	NP	A-B-C...- ON PN
		SR	(NOT OBTAINABLE)
<NONE>	A	IC	A ON A
		NP	(NOT AVAILABLE)
		SR	A ON "SR"
<NONE>	A-B	IC	B ON A
		NP	(NOT OBTAINABLE)
		SR	A-B ON "SR"
<NONE>	A-B-C...-N	IC	N ON A
		NP	(NOT OBTAINABLE)
		SR	A-B-C...-N ON "SR"

DEFINITIONS OF SOME MNEMONICS USED IN DOCUMENT

IC = INTERCHANGE PACK

BP = BACKPACK

SR = SYSTEM RESOURCE PACK

NP = NAMED PACK

<I> = ANY INVALID CHARACTER

HPT = HEAD-PER-TRACK DISK

D0029 JOB SWAPPING - 08-29-72

THE PURPOSE OF JOB SWAPPING IS TO PROVIDE A MEANS OF MORE EFFICIENTLY USING MEMORY OF THE B6700 SYSTEM IN A DATACOM ENVIRONMENT. THIS IS ACCOMPLISHED BY ORGANIZING THE MEMORY SPACE FOR EACH SWAP-JOB IN SUCH A WAY THAT ALL OF THE PARTS OF THE JOB THAT CAN BE OVERLAYED (SWAPPED) ARE IN ONE CONTIGUOUS AREA OF MEMORY. THIS ALLOWS A SINGLE DISK OPERATION TO WRITE THE ENTIRE AREA TO DISK (THEREBY FREEING THE MEMORY AREA) OR A SINGLE DISK READ TO RETURN THE ENTIRE AREA TO MEMORY FOR RESUMPTION OF PROCESSING.

THE AREA OF MEMORY WHICH IS TO CONTAIN SWAP-JOBS IS REFERRED TO AS THE "SWAP-AREA". THE SIZE OF THE SWAP-AREA IS INSTALLATION SPECIFIED AND CAN BE LOCATED ANYWHERE IN MAIN MEMORY. SOME OF THE CHARACTERISTICS OF JOBS IN THE SWAP AREA ARE:

1. THE AMOUNT OF CORE ALLOCATED FOR A SWAP-JOB IS IN MULTIPLES OF 990 WORDS, AND THIS IS CALCULATED FROM THE CORE ESTIMATE OF THE JOB WHEN IT IS INITIATED. NOTE THAT THE AMOUNT OF CORE IS ROUNDED UP TO THE NEXT MULTIPLE OF 990 WORDS.

2. THE VISIBLE NAME OF THE SWAP-JOB HAS A HYPHEN, "-", BETWEEN THE MIX NUMBER AND ITS NAME AS DISPLAYED ON THE SCREEN.

3. A SWAP-JOB IS SWAPPED OUT WHEN:

   A. THE OPERATOR ST-S THE JOB;

   B. THE JOB ATTEMPTS TO DO A DATACOM READ, AND THE INPUT QUEUE IS EMPTY;

   C. THE JOB PRODUCES MORE THAN ITS LIMIT OF DATACOM OUTPUT.

4. A SWAP-JOB IS SWAPPED BACK IN WHEN:

   A. THE OPERATOR OK-S THE JOB;

   B. A DATACOM INPUT MESSAGE IS INSERTED INTO THE INPUT QUEUE OF THE SWAP-JOB;

   C. THE DCP HAS, IN FACT, TRANSMITTED MORE THAN HALF OF THE LIMIT OF DATACOM OUTPUT FOR THE SWAP-JOB.

5. ON A SWAP-IN, THE SWAP-JOB MAY BE RETURNED TO ANYWHERE WITHIN THE SWAP-AREA; I.E., IT DOES NOT HAVE TO BE PUT IN THE SAME SPOT IT WAS WHEN IT WAS SWAPPED-OUT. THOSE WORDS CONTAINING ABSOLUTE ADDRESSES ARE ALTERED BEFORE PROCESSING OF THE SWAP-JOB IS ACTUALLY RESUMED. (IT SHOULD BE NOTED THAT THIS FIXUP AMOUNTS TO APPROXIMATELY 2.7 MICROSECONDS FOR EACH WORD THAT HAS TO BE ALTERED.) THE SWAP-IN ALGORITHM ATTEMPTS TO PACK THE JOB TOWARDS ONE END OF THE SWAP AREA TO AVOID CHECKERBOARDING.

6. THE DISTACK OF A JOB (USUALLY REFERRED TO AS ITS "SEGMENT DICTIONARY") IS LOCATED OUTSIDE OF THE SWAP-AREA. FURTHERMORE, ALL CODE AND VALUE ARRAYS REFERENCED BY THE DISTACK ARE LOCATED

OUTSIDE OF THE SWAP-AREA. NOTE THAT THIS FACILITATES RE-ENTRANCE BY A MULTIPLICITY OF JOBS WHETHER THEY ARE LOCATED INSIDE OR OUTSIDE OF THE SWAP-AREA.

7. THE "TASK VARIABLE" OF EACH SWAP-JOB IS KEPT AS A NON-OVERLAYABLE ITEM OUTSIDE OF THE SWAP-AREA SUCH THAT THE VARIOUS TASK ATTRIBUTES MAY BE INVESTIGATED WHETHER THE SWAP-JOB IS IN OR OUT OF CORE.

8. SEVERAL KEYBOARD MESSAGES ARE TREATED AS INVALID OUTPUT WHEN THEY REFERENCE A SWAP-JOB: CU, HI, OG, OT, AND PR.

9, A JOB WILL BE TERMINATED IF, AS A SWAP-JOB, IT EXECUTES A CALL, PROCESS, OR RUN STATEMENT. HOWEVER, WHILE A PARENT TASK CANNOT BE LOCATED WITHIN THE SWAP-AREA, THE OFFSPRING TASKS MAY. THERE IS A TASK ATTRIBUTE, "SUBSPACES", WHICH INDICATES THE DESIRE FOR THE TASK TO BE INITIATED INTO THE SWAP-AREA. THERE ARE SEVERAL CONDITIONS WHICH MAY DENY THE REQUEST:

A. THE "INDEPENDENT RUNNER" (CALLED "SWAPPER") MAY NOT BE IN THE MIX, OR IT MAY BE IN A DS-ED CONDITION (I.E., TOLD TO GO AWAY, BUT IT CANNOT SINCE THERE ARE ONE OR MORE SWAP-JOBS STILL IN EXECUTION);

B. THE CORE ESTIMATE OF THE TASK IS GREATER THAN THE ENTIRE SWAP-AREA (IN WHICH CASE THE TASK IS DS-ED DUE TO RESOURCE-CAUSE).

IF THE REQUEST IS DENIED, THE TASK ATTRIBUTE IS SET TO ZERO, AND THE INITIATOR CAN IMMEDIATELY KNOW WHETHER THE INITIATOR MADE IT INTO THE SWAP-AREA.

10. THE MAXIMUM SIZE OF THE ENTIRE SWAP-AREA IS LIMITED BY THE TOTAL AMOUNT OF MEMORY ON THE SYSTEM, BUT THE MAXIMUM SIZE OF A SWAP-JOB IS 131K WORDS (2 17-1). THIS LIMIT IS ESTABLISHED BY THE MAXIMUM SIZE OF DATA THAT CAN BE TRANSFERRED IN ONE DISK OPERATION.

11. ONCE INITIATED, A TASK CANNOT BE ALLOCATED MORE SPACE IN THE SWAP-AREA; I.E., REQUESTS FOR MORE SPACE WILL SIMPLY

RESULT IN OVER-LAYING WITHIN THE AREA THAT BELONGS TO THE TASK. STATED IN YET ANOTHER WAY, A TASK WITHIN THE SWAP-AREA CONTENDS ONLY WITH ITSELF FOR MEMORY; IT DOES NOT "EXPAND".

12. ONCE INITIATED, A TASK IS EITHER IN THE SWAP-AREA OR IT IS NOT; FURTHERMORE, IT WILL NOT BE CHANGED FROM SWAPPABLE TO NON-SWAPPABLE OR VICE-VERSA.

113. SWAPPER DOES NOT REMEMBER THE PRIOR POSITION OF A SWAP-JOB ON DISK. ON A SWAP-OUT, SWAPPER ALLOCATES A PLACE WITHIN SYSTEM/SWAPDISK TO WHICH THE SWAP-JOB WILL BE WRITTEN. NOTE THAT THIS ALLOCATION IS DONE ON A ROTATIONAL BASIS AMOUNG THE ROWS OF SYSTEM/SWAPDISK. THE PURPOSE IS TO DISTRIBUTE THE DISK TRAFFIC AMONGST THE EU-S.

SWAPPER

THE ACTUAL SWAPPING PROCESS OF THE MCP IS ACCOMPLISHED BY A VISIBLE INDEPENDENT RUNNER NAMED SWAPPER. SWAPPER WAITS ON AN EVENT CALLED SWAPREQ WHICH IS CAUSED WHEN A NEW REQUEST IS GIVEN TO SWAPPER OR WHEN A CONDITION HAS CHANGED WHICH IS OF INTEREST TO SWAPPER.

SWAPPER IS INITIATED BY THE KEYBOARD REQUEST "SW". (NOTE THAT THIS IS THE SOLE INPUT REQUIRED TO PRODUCE THE SWAPPING ENVIRONMENT.) UPON INITIATION, SWAPPER CHECKS FOR THE AVAILABILITY OF A FILE ON DISK CALLED "SYSTEM/SWAPDISK". THIS FILE NOT ONLY CONTAINS THE PARAMETER NEEDED BY SWAPPER AS FAR AS MAXIMUM SIZSE OF THE SWAP-AREA, BUT IT IS THE DISK AREA TO/FROM WHICH JOBS ARE SWAPPED.

SYSTEM/SWAPDISK

IN ORDER TO PERFORM SWAPPING, THIS FILE MUST EXIST AND PASS SEVERAL CONSISTENCY CHECKS; A SIMPLE PROGRAM WHICH CREATES AN ACCEPTABLE FILE IS AS FOLLOWS:

```
<I> COMPILE SYSTEM/SWAPDISKMAKER ALGOL LIBRARY
<I> EBCDIC
$ SET LEVEL 2
PROCEDURE MAKER (MAXSWAPAREASLOTS, MAXSLOTSPERDISKROW,
```

```
MAXDISKROWS);
VALUE MAXSWAPAREASLOTS, MAXLOTSPERDISKROW, MAXDISKROWS;
REAL MAXSWAPAREASLOTS, MAXSLOTSPERDISKROW, MAXDISKROWS;

BEGIN

  FILE SWAP (TITLE = "SYSTEM/SWAPDISK.", MAXRECSIZE=1320,
    BUFFERS=1, KIND=DISK);
  ARRAY A[0:29];
  REAL I;
  SWAP.AREAS := MAXDISKROWS;
  SWAP.AREASIZE := MAXSLOTSPERDISKROW;
  REPLACE POINTER(A) BY "SYSTEM/SWAPDISK.  ", 990 *
    MAXSWAPAREASLOTS FOR 6;
  FOR I := 0 STEP 1 UNTIL MAXDISKROWS-1 DO
    WRITE (SWAP [I * MAXSLOTSPERDISKROW], 30, 1);
    LOCK (SWAP);
  END.
  <I> END
```

FROM THE ABOVE PROGRAM, THE FOLLOWING OBSERVATIONS MAY BE MADE:

1. THE TERM "SLOT" REPRESENTS 990 WORDS OF MEMORY. (NOTE THAT THE MAXRECSIZE OF 1320 IS, IN FACT, 44 30-WORD SEGMENTS. THIS DIFFERENCE IS DUE TO THE DISK READS/WRITES INCLUDING "TAG TRANSFERS".);

2. WORDS 0, 1, AND 2 CONTAIN THE EBCDIC STRING "SYSTEM/SWAPDISK. ";

3. WORD 3 CONTAINS THE SIZE OF THE SWAP-AREA TO BE ESTABLISHED IN MEMORY;

4. ALL ROWS OF THE FILE ARE WRITTEN SUCH THAT DISK SPACE FOR EACH ROW IS ESTABLISHED;

5. THE FILE IS ESTABLISHED ON HEAD-PER-TRACK DISK ONLY (NOTE THAT THE FILE CAN BE ON EITHER SYSTEM-ALLOCATED OR INSTALLATION-ALLOCATED DISK);

6. THE ABOVE PROGRAM CAN BE INITIATED VIA KEYBOARD INPUT;

RUN SYSTEM/SWAPDISKMAKER (20, 20, 4)

A. THE FIRST 20 INDICATES THAT 20 TIMES 990, OR 19,800 WORDS, IS THE SWAP-AREA SIZE.

B. THE SECOND 20 INDICATES THAT EACH ROW OF THE FILE IS TO CONTAIN 20 SLOTS.

C. THE 4 INDICATES THAT THERE ARE TO BE FOUR ROWS IN THE FILE.

ERROR MESSAGES

SYSTEM/SWAPDISK NOT THERE	(MEANS THAT SWAPPER HAS CHECKED THE DIRECTORY, AND THERE IS NO FILE ON DISK WITH THE NAME SYSTEM/SWAPDISK.)
VALIDITY CHECK FAILURE	(MEANS THAT THE FIRST THREE WORDS OF RECORD ZERO OF SYSTEM/SWAPDISK DOES NOT CONTAIN THE EBCDIC STRING "SYSTEM/ SWAPDISK. ")
SWAPCORE NOT VALID	(MEANS THAT THE AMOUNT OF CORE SPECIFIED IN WORD 3 IS NOT AN EXACT MULTIPLE OF 990 WORDS.)
SWAPDISK NOT VALID	(MEANS THAT THE ROWSIZE OF THE SYSTEM/SWAPDISK FILE IS NOT AN EXACT MULTIPLE OF 44 SEGMENTS.)
MISSING ROW	(MEANS THAT THERE IS AT LEAST ONE OF THE ROWS OF SYSTEM/SWAPDISK WHICH, ALTHOUGH DECLARED, HAS NOT ACTUALLY BEEN ALLOCATED ON DISK.)
CORE NOT AVAILABLE	(MEANS THAT SWAPPER HAS REQUESTED A CONTIGUOUS SPACE IN MEMORY - THE AMOUNT =WORD 3 - BUT THE REQUEST COULD NOT BE GRANTED. SWAPPER "GOES AWAY", AND THE OPERATOR MUST INPUT "SW" AGAIN TO INITIATE

ANOTHER ATTEMPT TO GAIN THE NEEDED SPACE.)

GOING AWAY (MEANS THAT SWAPPER IS DS-ED AND WILL GO AWAY WHEN THE LAST SWAP-JOB GOES AWAY.)

D0031 MCP - COPY-COMPARE - 05-24-72

THE COMPARE OPTION IN THE MCP HAS BEEN IMPLEMENTED.

SYNTAX:

<COPY CS>::= COPY <COMPARE OPTION><COPY LIST>

<ADD CS>::= ADD <COMPARE OPTION><COPY LIST>

<COMPARE OPTION>::=<EMPTY>x/COMPARE

THE COMPARE OPTION MAY BE INVOKED ON ANY "COPY" OR "ADD" STATEMENT.
EXAMPLE: COPY/COMPARE = FROM A TO DISK, TO T.

ALGORITHM:

WHEN THE COMPARE OPTION IS REQUESTED, EACH FILE IS COMPARED AFTER IT HAS BEEN COPIED AND BEFORE THE NEXT FILE IS STARTED.

ALL TAPE SOURCES AND DESTINATIONS ARE BACKSPACED TO THE BEGINNING OF THE FILE. ALL DESTINATIONS ARE THEN READ AND COMPARED AGAINST THE SOURCE. IF A BAD DESTINATION IS DETECTED, A MESSAGE IS DISPLAYED:

```
(MIX LINE)  1234 COMPARE ERROR - MT 17
            ////////////////////////////
 (RSVP)       1234 RECOPY REQD - ABCD/WXYZ
```

THE OPERATOR CAN RESPOND TO THIS MESSAGE WITH:

A. DS - DS THE ENTIRE JOB;

B. FR - DS THE GIVEN DESTINATION BUT CONTINUE TO COPY TO ALL OTHERS;

C. OF - CONTINUE TO COPY, BUT DELETE THIS FILE FROM THIS DESTINATION. IF IT IS A TAPE, THEN BACKSPACE THE TAPE AND REWRITE THE FIRST RECORD (THE DISK HEADER) TURNING ON BIT 47 OF THE TRANSACTION COUNT TO PREVENT THE FILE FROM BEING COPIED BACK IN. IF IT IS A DISK DESTINATION, THEN DO NOT ENTER THE FILE IN THE DIRECTORY. THE FILE WILL BE COPIED NORMALLY TO ALL OTHER DESTINATIONS.

D. OK - FINISH COPYING THE FILE TO ALL OTHER DESTINATIONS THEN ATTEMPT TO COPY TO THIS ONE.

WHEN THE COMPARE IS FINISHED, ALL NECESSARY RECOPIES AND RECOMPARES ARE DONE AND THE NEXT FILE IS STARTED.

IF A REEL SWITCH OCCURS DURING THE COPY AND THE COMPARE OPTION HAS BEEN SPECIFIED, THE FIRST REEL CONTAINING THE FILE IS LEFT UP TAPE. ALL OTHER REELS ARE RELEASED. THE FIRST REEL IS RELEASED AFTER IT HAS BEEN READ BY THE COMPARE PHASE.

THIS ALGORITHM, AS OPPOSED TO THE METHOD OF COMPARING AFTER ALL FILES HAVE BEEN COPIED, WAS CHOSEN FOR THE FOLLOWING REASONS:

A. NO FILE IS ENTERED INTO THE DIRECTORY AND MADE AVAILABLE TO USER PROGRAMS UNTIL IS HAS BEEN VERIFIED.

B. IF A COPY ERROR IS DETECTED, THE OPERATOR CAN RECOPY THAT PARTICULAR FILE WITHOUT RESTARTING THE ENTIRE COPY.

C. ALL REELS OF A MULTI-REEL LIBRARY TAPE WILL NOT HAVE TO BE PRESENT SIMULTANEOUSLY.

D0032 SCR - DISK PACK CONFIDENCE TESTS - 08-15-72

A NEW IMPLEMENTATION IN SCR IS A "VERIFY" TYPE CONFIDENCE TEST FOR DISK PACKS. THE SYNTAX FOR USING THIS FEATURE IS MUCH LIKE THAT FOR THE HEAD-PER-TRACK DISK VERIFY TESTS WITH A FEW MAJOR EXCEPTIONS AS FOLLOWS:

1. THE FILE DECLARATION SYNTAX FOR DISK PACK FILES IS:

DISK PACK FILE <FILE ID> = "<DISK PACK FILE NAME>"

WHERE <DISK PACK FILE NAME>::= <PACK NAME><SLASH>AD<6 DIGIT NO>, E. G., FOR DISK PACK A, A FILE DECLARED AT ADDRESS 10000 WOULD BE:

A/AD010000

2. THE VERIFY STATEMENT SYNTAX IS:

VERIFY DISK PACK <UNIT OR FILE SPECIFIER>
(<DISK PACK TEST PARAMETERS>)

E.G., VERIFY DISK PACK FILE BD (SELECT ALL)

WHERE BD WAS PREVIOUSLY DEFINED AS A DISK PACK FILE.

AT PRESENT, THE <UNIT OR FILE SPECIFIER> IN THE DISK PACK VERIFY IS LIMITED TO <FILE SPECIFIER> AS MAT WILL NOT ALLOW WRITES WHICH ARE SPECIFIED BY <UNIT SPECIFIER> AND SEGMENT, AND ALL THE DISK PACK TESTS CONTAIN WRITE PORTIONS. THE FILETYPE FOR DISK PACK FILES MUST BE XDISK.

THE FORM OF THE <DISK PACK TEST PARAMETERS> IS THE SAME AS THAT FOR HEAD-PER-TRACK EXCEPT THAT THE NUMBER OF CONFIDENCE TESTS AND THE TESTS THEMSELVES ARE DIFFERENT.

TEST ONE:

TEST ONE USES A 30-WORD DATA PATTERN WHICH CONSISTS OF ONE-WORD CONTAINING THE DISK PACK ADDRESS, IN DECIMAL, FOLLOWED BY 29 WORDS OF A "WORST CASE" DATA PATTERN. THIS RECORD IS WRITTEN TO EACH SEGMENT OF THE FILE, WITH APPROPRIATE CHANGES IN THE ADDRESS KEY, AND THEN IS READ BACK AND CHECKED. THE READ AND WRITE PORTIONS OF THE TEST WILL DETECT AND PRINT OUT RESULT DESCRIPTOR ERRORS, AND THE READ PORTION WILL PRINT THOSE PORTIONS OF THE DATA WHICH ARE NOT EQUAL TO THE WRITTEN PATTERN.

TEST TWO:

TEST TWO WRITES A KEYED 30-WORD WORST-CASE DATA PATTERN TO EACH SEGMENT OF THE FILE; THIS OPERATION IS PERFORMED BY WRITING ENTIRE

33-SEGMENT TRACKS USING A SINGLE I/O OPERATION, WITH SPECIAL HANDLING FOR THE BEGINNING AND END OF THE FILE, WHICH MAY NOT BE ALIGNED ON A TRACK BOUNDARY.

AT THE COMPLETION OF THIS INITIALIZATION, THE ADDRESS KEY IN THE FIRST WORD OF EACH SEGMENT IS READ BACK AND CHECKED. THE ADDRESS PATTERN USED FOR THE READ BACK CONSISTS OF SEGMENTS 1, N, 2, N-1, 3, N-2, ... N DIV 2, N DIV 2 + 1, ...N, 1. THE ENTIRE FILE IS "COVERED" TWICE. RESULT-DESCRIPTOR AND DATA ERRORS ARE NOTED ON THE OUTPUT DEVICE.

TEST THREE:

TEST THREE CHECKS SEGMENT AND TRACK CROSSOVER BY:

1. WRITING TWO SEGMENTS PER I/O AND THEN READING BACK THE TWO SEGMENTS INDIVIDUALLY, USING TWO I/O OPERATIONS, AND
2. WRITING TWO SEGMENTS USING TWO I/O OPERATIONS AND THEN READING THESE TWO SEGMENTS BACK USING ONE I/O OPERATION.

THIS OPERATION IS PERFORMED ON EACH PAIR OF SEGMENTS IN THE FILE, WITH APPROPRIATE DIAGNOSTIC OUTPUT FOR RESULT-DESCRIPTOR AND DATA-COMPARISON ERRORS. THE MINIMUM FILE SIZE FOR THIS TEST IS FOUR SEGMENTS.

TEST FOUR:

TEST FOUR IS DESIGNED TO TEST MULTIPLEXOR TAG-TRANSFER AND DISK PACK DATA RECORDING AND ADDRESSING. THE METHOD CONSISTS OF LOCATING A WRITE-OPERATION IOCW IN FRONT OF THE MCP NON-OVERLAYABLE CODE SEGMENT (SEGMENT FIVE) IN ORDER TO TRANSFER THIS DATA ONTO THE DISK PACK AND THEN READING THE DATA BACK INTO A LOCAL ARRAY AND COMPARING THE LOCAL ARRAY CONTENTS TO THE SEGMENT FIVE CODE.

THERE ARE FIVE IOCW LOCATIONS ALLOCATED FOR THIS WRITE-IOCW, AND TEST FOUR USES ALL FIVE IN ORDER TO "MOVE" THE DATA PATTERN ON THE DISK PACK SURFACE.

THE ENTIRE DISK PACK FILE IS CHECKED USING EACH OF THESE IOCW LOCATIONS SO THAT ANY GIVEN AREA ON THE DISK PACK SURFACE WILL BE OCCUPIED BY A DIFFERENT WORD OF THE SEGMENT FIVE DATA WHEN THE IOCW IS PLACED IN A DIFFERENT ONE OF THE FIVE IOCW SLOTS.

THE 7100-WORD SIZE OF SEGMENT FIVE PRECLUDES SHOWING ALL OF THE DATA WHEN A DATA COMPARISON ERROR IS ENCOUNTERED; THE COMPROMISE ADOPTED HERE IS THAT 20 LINES OF "DATA READ" AND "DATA EXPECTED" ARE DISPLAYED ON THE OUTPUT DEVICE, AND A COUNT OF THE NUMBER OF UNEQUAL WORDS IN THE REMAINDER OF THE BUFFERS IS SUPPLIED BUT THE DATA ITSELF IS NOT SHOWN.

THE SIZE OF THE DATA ON THE DISK PACK (ROUGHLY 64x7100 DIV 48 DIV 30 = 315 SEGMENTS) IS NOT NECESSARILY AN INTEGRAL NUMBER OF SEGMENTS, SO THE BEGINNING DISK PACK ADDRESS FOR SUCCESSIVE WRITES IS TRUNCATED TO ONE LARGER THAN THE LAST FULL 30-WORD SEGMENT ACCESSED; THIS TEST WILL "COVER" ALL DATA AREAS IN THE FILE.

THIS TEST WILL NOT BE RUN IF THE SIZE OF THE FILE IS LESS THAN ONE ENTIRE TRACK (33 SEGMENTS).

TEST FIVE:

TEST FIVE USES TAG-TRANSFER IN CONJUNCTION WITH A SEQUENCE OF I/O LENGTHS WHICH WERE COMPUTED TO PRODUCE A DATA SIZE ON THE DISK PACK SURFACE WHICH IS NOT AN INTEGRAL NUMBER OF 48-BIT WORDS. THESE I/O LENGTHS BEGIN WITH THREE AND 23; SUCCESSIVE VALUES ARE THE SUM OF THE TWO PRECEDING VALUES.

TEST SIX:

DISK PACK TEST SIX USES THE SAME ADDRESS AND DATA PATTERNS AS HEAD-PER-TRACK DISK TEST 12. THE I/O LENGTHS VARY BETWEEN THREE AND 60 SEGMENTS; THE BEGINNING ADDRESS FOR EACH I/O IS SELECTED IN SUCH A MANNER THAT ALL SEGMENTS ARE WRITTEN WITH BOTH AN ALL-ONES AND AN ALL-ZEROES DATA PATTERN. THESE I/O LENGTHS ARE AN INTEGRAL NUMBER OF SEGMENTS, AND THE DATA IS TRANSFERRED WITH IOCW BITS 37 AND 38 EQUAL TO ZERO (NO TAG TRANSFER).

THIS TEST WILL NOT BE RUN IF THE FILE IS SMALLER THAN FOUR SEGMENTS.

D0033 PROGRAMDUMP - 07-17-72

ONE OF THE NEW FEATURES OF THE II.3 SOFTWARE RELEASE IS THE NEW PROGRAMDUMP. PROGRAMDUMP IS AN MCP PROCEDURE WHICH CAN BE INTENTIONALLY INVOKED BY THE USER PROGRAM, THE OPERATOR CAN CAUSE ONE TO HAPPEN ON A JOB OPERATING IN THE MIX, OR ON A "FAULT" OR DS BY OPERATOR ONE CAN BE INVOKED. THE PRINTOUT IS ALMOST IDENTICAL TO THAT PRODUCED BY SYSTEM/DUMPANALYZER.

SEVERAL OPTIONS ARE AVAILABLE AS TO WHICH ITEMS OTHER THAN THE "USER PORTION" OF THE STACK ARE TO BE DUMPED AND ANALYZED. THE ITEMS ARE SPECIFIED BY NAMING THEM EITHER IN THE CONTROL CARD; OR IN THE CASE OF ALGOL OR DCALGOL, THEY CAN BE PASSED AS PARAMETERS IN THE INVOCATION OF PROGRAMDUMP.

THE (DEFAULT) PRINTOUT IS ONLY THE USER PORTION OF THE PROCESS STACK (ALSO REFERRED TO AS THE D2 STACK).

IF THE CONTENTS OF THE PROGRAMS ARRAYS ARE TO BE PRINTED, THE OPTION "ARRAYS" MUST BE SPECIFIED.

THE BOTTOM (OR "BASE") OF THE USERS STACK WILL BE PRINTED IF THE "BASE" OPTION IS SPECIFIED. THE MCP USES THE BASE PORTION OF EACH STACK TO CONTAIN VARIOUS WORDS NEEDED TO CONTROL, IDENTIFY, LOG, ETC., THAT STACK.

THE SEGMENT DICTIONARY OF THE PROGRAM IS PRINTED OUT AS A SEPARATE STACK IF THE "CODE" OPTION IS SPECIFIED. FURTHERMORE, THE ACTUAL CODE WILL BE PRINTED FOR ONLY THOSE SEGMENTS WHICH WERE REFERENCED BY THE RCW-S IN THE USER-S STACK. NOTE THAT VALUE ARRAYS OF THE SEGMENT DICTIONARY WILL BE PRINTED WHEN BOTH CODE AND ARRAYS ARE SPECIFIED.

IF A PROGRAM WANTS ITS FILES TO BE PRINTED AND ANALYZED, THE "FILES" OPTION MUST BE SPECIFIED. AS EACH FILE IS ENCOUNTERED, EACH WORD OF THE FIB IS SEPARATELY NAMED AND, IN SOME CASES, FURTHER ANALYZED.

PROGRAM INITIATION:

WHEN A PROGRAM WANTS TO CAUSE A PROGRAMDUMP, IT SIMPLY EXECUTES A NEW STATEMENT:

LANGUAGE	CONSTRUCT
ALGOL	<PROGRAMDUMP STATEMENT>::=PROGRAMDUMP <PARAMETER PART>
OR	
	<PARAMETER>::=<EMPTY>/(<PARAMETER LIST>)
DCALGOL	<PARAMETER LIST>::=<PARAMETER ITEM>/ <PARAMETER LIST>,<PARAMETER ITEM>/ <ARITHMETIC EXPRESSION> SKIP 1 INDENT 14 <PARAMETER ITEM>::=ARRAY/ARRAYS/BASE/CODE/ FILE/FILES/ALL
	(NOTE - THE BITS OF THE <ARITHMETIC EXPRESSION> INDICATE THE <PARAMETER ITEM>; EXPLAINED LATER)
COBOL	CALL PROGRAM DUMP

OPERATOR INTIATION:

IF A "SNAPSHOT" OF AN OPERATING PROGRAM IS DESIRED, THE OPERATOR CAN INITIATE SUCH ACTION BY KEYING IN:

<MIX NUMBER> DP

NOTE THAT THE PROGRAM WILL BE TEMPORARILY SUSPENDED WHILE THE DUMP OCCURS; THEN IT WILL RESUME. FURTHERMORE, THE PROGRAM WILL HAVE NO KNOWLEDGE THAT A DUMP OCCURRED.

DUMP ON FAULT OR DS:

THE OPTIONWORD OF A STACK CAN BE SET SO THAT A PROGRAMDUMP WILL BE INVOKED UPON EITHER AN INTERNAL PROGRAM FAULT OR AN OPERATOR DS-ING THE JOB. OPTIONWORD OF THE STACK CAN BE SET VIA CONTROLCARD AS FOLLOWS:

<I> OPTION = FAULT

OR

<I> OPTION = DSED

OR

<I> OPTION = FAULT DSED (DUMP IF EITHER ONE OCCURS)

SPECIFYING PROGRAMDUMP OPTIONS VIA CONTROLCARD:

THE OPTIONWORD ALSO SPECIFIES WHICH ITEMS (IF ANY) ARE TO BE DUMPED IF A FAULT OR DS OCCURS. IN ADDITION, THE OPTIONWORD ALSO SPECIFIES WHICH ITEMS ARE TO BE DUMPED IN COBOL OR IN ALGOL IF THE <PARAMETER> IS <EMPTY>.

THE OPTIONWORD CAN BE LOADED BY HAVING A <PROGRAMDUMP OPTION LIST> ON THE OPTION CONTROL CARD (WHEN THE JOB WAS INITIATED):

<PROGRAMDUMP OPTION LIST>::=<PROGRAMDUMP OPTION>/
<PROGRAMDUMP OPTION LIST>,<PROGRAMDUMP OPTION>

<PROGRAMDUMP OPTION>::=ARRAY/ARRAYS/BASE/CODE/FILE/FILES

PROGRAMMATICALLY SPECIFYING PROGRAMDUMP OPTIONS:

THE OPTIONWORD CAN BE LOADED FROM WITHIN THE PROGRAM ITSELF VIA THE OPTION TASK ATTRIBUTE AND SETTING THE FOLLOWING BITS:

```
 7:1 = 1   IF THE BASE OF THE USER STACK IS TO BE PRINTED;
 8:1 = 1   IF ALL ENCOUNTERED ARRAYS ARE TO BE PRINTED;
 9:1 = 1   IF CODE (I.E., THE SEGMENT DICTIONARY STACK)
           SEGMENTS ARE TO BE PRINTED;

10:1 = 1   IF FILES ARE TO BE PRINTED AND ANALYZED;
10:4 = 15  IF ALL PORTIONS OF THE PROGRAM ARE TO BE
           PRINTED OR ANALYZED.
```

LANGUAGE	CONSTRUCT

```
ALGOL     MYSELF.OPTION := MYSELF.OPTION & 1[7:1]; % BASE
  OR      MYSELF.OPTION := MYSELF.OPTION & 1[8:1]; % ARRAY(S)
DCALGOL   MYSELF.OPTION := MYSELF.OPTION & 1[9:1]; % CODE
          MYSELF.OPTION := MYSELF.OPTION & 1[10:1]; % FILE(S)
          MYSELF.OPTION := MYSELF.OPTION & 15[10:4]; % EVERYTHING

COBOL     SET MYSELF (OPTION) TO VARB.
```

ALGOL/DCALGOL PASSING <ARITHMETIC EXPRESSION> AS <PARAMETER ITEM>:

BY NOTING THE BITS SET ABOVE AND THE DUMPING OF THE CORRESPONDING ITEM, THE PROGRAMMER CAN SET BITS AS NEEDED IN THE <ARITHMETIC EXPRESSION>.

D0034 MCP - LIBRARY MAINT FOR DISK PACKS - 07-26-72

THE MARK II.3. RELEASE WILL IMPLEMENT ALL LIBRARY MAINTENANCE FUNCTIONS TO AND FROM NATIVE MODE DISK PACKS.

THE FOLLOWING FILE ATTRIBUTES MAY BE SET ON A DISK PACK SOURCE OR DESTINATION.

1. AREACLASS
2. UNITNO = UNITNUMBER OF A MEMBER OF A PACK FAMILY. BASE PACK WILL BE REQUESTED IF NOT PRESENT.
3. KIND

PACK IS A RESERVED NAME DESIGNATING SYSTEM RESOURCE PACK FAMILY. AS BEFORE, DISK IS A RESERVED NAME DESIGNATING HEAD-PER-TRACK DISK. ANY OTHER NAME WILL BE, BY DEFAULT, A TAPE NAME UNLESS THE KIND ATTRIBUTE IS SET TO PACK (OR DISK PACK). EXAMPLES:

```
COPY X TO PACK;

ADD X TO MYPACK (KIND = PACK, UNITNO = 83);

ADD/COMPARE = FROM MYPACK (KIND = PACK) TO DISK, TO T;
```

D0035 SORT IMPROVEMENTS - 08-25-72

THE B6700 SORT HAS BEEN EXPANDED TO PROVIDE CERTAIN ADDITIONAL FEATURES AND TO CORRECT PREVIOUSLY REPORTED PROBLEMS. THIS SYSTEM NOTE IS INTENDED TO DESCRIBE FEATURES NOT DESCRIBED ELSEWHERE AND TO CONVEY THE CURRENT MARK II.3 STATUS OF THE SORT. FEATURES OF THE MARK II.3 SORT WILL BE EXPLAINED IN MORE DETAIL AND CONSIST OF THE FOLLOWING:

1. THE ABILITY TO RESTART THE SORT AFTER A SYSTEM HALT OR DS OF THE SORT.

2. EXTENDED ERROR RECOVERY WHEN I/O ERRORS OCCUR WHILE THE SORT IS ACCESSING A SORT WORK FILE OR USER OUTPUT FILE.

3. THE ABILITY TO DO A CORE SORT.

4. MORE USER CONTROL OVER SORT MEMORY ALLOCATION.

5. ADDITIONAL SORT ERROR MESSAGES AND SOME NON-FATAL MESSAGES.

RESTART

THE ABILITY FOR THE SORT TO RESUME PROCESSING AT THE MOST RECENT CHECKPOINT FOLLOWING THE DISCONTINUANCE OF THE PROGRAM HAS BEEN IMPLEMENTED. OPERATION OF THE SORT IN THIS MODE PROVIDES THE NECESSARY RESTARTING INFORMATION FOR THE SORT AND REQUIRES CERTAIN PROGRAM INPUTS WHICH WILL BE DEFINED LATER. IT IS NECESSARY FOR THE PROGRAM TO PROVIDE LOGIC TO RESTORE AND MAINTAIN STACK VARIABLES, ARRAYS, FILES, POINTERS, ETC. THAT ARE DEFINED FOR AND BY THE PROGRAM. IN OTHER WORDS, THE PROGRAM MUST PROVIDE THE MEANS TO RESTORE EVERYTHING THAT IS NECESSARY FOR THE PROGRAM TO CONTINUE FROM THE POINT OF INTERRUPTION. THIS MAY BE A SIMPLE OR DIFFICULT TASK AND IS ENTIRELY PROGRAM DEPENDENT.

PLEASE NOTE THAT RESTART CAPABILITY IS IMPLEMENTED FOR DISK SORT ONLY AND NOT TAPE SORT, BUT IT IS POSSIBLE TO HAVE A PARTIALLY RESTARTABLE ITD SORT. WHEN TAPE FILES ARE NEVER USED IN AN ITD SORT IT FUNCTIONS AS A DISK SORT. AFTER THE DATA IS ONCE WRITTEN FROM DISK TO TAPE (DURING AN ITD SORT) THE SORT CAN NO LONGER BE RESTARTED. ONCE WRITTEN MEANS THE FIRST TIME THE SORT HAS WRITTEN

ANY DATA TO TAPE AND HAS SUBSEQUENTLY RETURNED TO PROCESS MORE INPUT DATA. IT IS POSSIBLE TO BEGIN A SORT AS A DISK ONLY RESTARTABLE SORT SUCH THAT INSUFFICIENT DISK IS PROVIDED TO ACCOMPLISH THE SORT. WHEN THIS HAPPENS THE SORT WILL TERMINATE WITH SORT ERROR #4 OR 5. AFTER ERROR TERMINATION, YOU CAN RESTART THE SORT AS AN ITD SORT BY INDICATING A RESTART AND SPECIFYING THE NUMBER OF TAPES YOU DESIRE. ANY OTHER KIND OF RESTART IS NOT POSSIBLE UNDER THIS CONDITION. IF A RESTARTABLE SORT TERMINATES DURING THE FIRST OUTPUT OF DATA FROM DISK TO TAPE (POSSIBLY AS A RESULT OF AN IRRECOVERABLE TAPE I/O ERROR), THE SORT CAN BE RESTARTED AND THE DATA WILL BE WRITTEN TO TAPE AS IF NO PROBLEM HAD EVER OCCURRED.

WHEN USING THE SORT IN RESTARTABLE MODE, IT IS DESIRABLE TO GIVE UNIQUE FILE TITLES TO THE TWO SORT DISK FILES. THIS IS ACCOMPLISHED BY USE OF THE TITLE ATTRIBUTE OF A FILE CONTROL CARD AND WILL BE COVERED IN MORE DETAIL LATER. IT IS IMPORTANT TO RECOGNIZE THE POSSIBILITY OF CONFLICTS WHEN TWO OR MORE SORTS ARE USING IDENTICAL FILE TITLES FOR THEIR SORT DISK FILES.

WHEN THE SORT IS ATTEMPTING TO RESTART A PREVIOUSLY INCOMPLETE SORT, A MINIMAL AMOUNT OF INFORMATION IS VERIFIED TO INSURE THAT CONTINUATION WILL BE COMPATIBLE WITH THE PREVIOUS SORT. THE TWO ITEMS CURRENTLY VERIFIED ARE SORT RECORD SIZE AND CHARACTER SIZE OF THE SORT RECORD CHARACTERS. FOR ALGOL PROGRAMS, RECORD SIZE IS EXPLICITELY SPECIFIED BY THE PROGRAM WHILE CHARACTER SIZE IS ZERO (DEFAULT SIZE OF EIGHT). COBOL PROGRAMS USE THE SD TO DETERMINE SORT RECORD AND CHARACTER SIZE. WHEN RECORD SIZE OR CHARACTER SIZE DO NOT MATCH THE PREVIOUS SORT, ERROR TERMINATION WILL OCCUR. MODIFICATION OF OTHER SORT PARAMETERS (EXCEPT FOR NUMBER OF TAPES AS PREVIOUSLY STATED) WILL BE ALLOWED. DIFFERENT VALUES FOR MEMORY SIZE OR DISK SIZE WILL BE IGNORED AND THE ORIGINAL VALUES WILL BE USED, HOWEVER, BOTH MUST BE VALID NON-ZERO VALUES. DIFFERENT PROCEDURES CAN BE SPECIFIED, IF DESIRED, FOR INPUT, OUTPUT, OR COMPARING AND FILES MAY BE INTERCHANGED WITH INPUT OR OUTPUT PROCEDURES. FROM THE PREVIOUS DISCUSSION IT SHOULD BE APPARENT THAT THE PROGRAM REQUESTING THE RESTART NEED NOT BE THE ORIGINATING

PROGRAM. IT WILL BE THE USERS RESPONSIBILITY TO INSURE THAT HE GETS THE DESIRED RESULT SINCE THE SORT CAN ONLY ATTEMPT TO MEET THE USER REQUEST AND CANNOT DETERMINE APPROPRIATENESS OF REQUESTS.

LANGUAGE SYNTAX

IMPLEMENTATION OF RESTART AND I/O ERROR RECOVERY HAVE REQUIRED EXTENSIONS TO THE SORT STATEMENTS IN THE COBOL AND ALGOL COMPILERS.

THE SYNTAX FOR COBOL IS:

```
SORT <FILE-NAME-1> ON (ASCENDING/DESCENDING)
   KEY <DATA-NAME-1> [,<DATA-NAME-2>]...
   [, ON (ASCENDING/DESCENDING) KEY
   <DATA-NAME-3> [, <DATA-NAME-4>]...]...

(USING <FILE-NAME-2>/INPUT PROCEDURE IS
   <SECTION-NAME-1> [THRU <SECTION-NAME-2>])

(GIVING <FILE-NAME-3>/OUTPUT PROCEDURE IS
   <SECTION-NAME-3> [THRU <SECTION-NAME4>])

[RESTART IS <FORMULA/DATA-NAME-5/LITERAL-1>]
```

THE VALUE OF THE LEAST SIGNIFICANT (RIGHT MOST) FIVE BITS OF THE FORMULA, DATA-NAME-5, OR LITERAL-1 IS PASSED TO THE SORT TO INDICATE THE DESIRED SORT ACTION. THIS PARAMETER IS BASICALLY BIT ORIENTED AND CAN TAKE ON A NUMBER OF VALUES THAT WILL BE DEFINED LATER. PLEASE REFER TO THE COBOL LANGUAGE MANUAL FOR SORT STATEMENT INFORMATION PERTAINING TO EVERYTHING EXCEPT THE RESTART OPTION.

THE SYNTAX FOR ALGOL IS:

```
<SORT STATEMENT>::=SORT (<OUTPUT OPTION>,
   <INPUT OPTION>, <NUMBER OF TAPES>,
   <COMPARE PROCEDURE>, <RECORD LENGTH>
   <SIZE SPECIFICATIONS>)<RESTART SPECIFICATIONS>

<RESTART SPECIFICATIONS>::=<EMPTY>/
   [RESTART=<ARITHMETIC EXPRESSION>]
```

THE VALUE OF THE LEAST SIGNIFICANT (RIGHT MOST) FIVE BITS OF THE RESTART EXPRESSION WILL BE PASSED TO THE SORT TO INDICATE DESIRED ACTION. THE VALUES THAT THIS PARAMETER CAN TAKE ON WILL BE DESCRIBED LATER. PLEASE REFER TO THE ALGOL LANGUAGE MANUAL FOR INFORMATION PERTAINING TO THE SORT STATEMENT EXCEPT FOR THE RESTART OPTION.

RESTART PARAMETER VALUES

THE SORT WILL INSPECT VARIOUS BITS OF THIS PARAMETER TO DETERMINE THE COURSE OF ACTION IT WILL TAKE. INDIVIDUAL BITS AND COMBINATIONS OF BITS MAY BE SET BY THE PROGRAM TO CONTROL THE SORT. THE VARIOUS BITS AND THEIR MEANING ARE:

BIT 0: ON - MEANS THAT THE PROGRAM IS RESTARTING A PREVIOUS SORT. THE SORT WILL TRY TO OPEN ITS TWO DISK FILES AND OBTAIN RESTART INFORMATION. AFTER SUCCESSFULLY OBTAINING THIS INFORMATION, THE SORT WILL CONTINUE FROM THE LAST KNOWN RESTART POINT. OFF - MEANS THE SORT IS STARTING FROM THE BEGINNING. IF THE SORT IS A RESTARTABLE SORT AND PREVIOUS SORT FILES WITH IDENTICAL TITLES EXIST, THEY WILL BE REMOVED AND REPLACED BY NEW SORT FILES.

BIT 1: ON - MEANS THAT THE PROGRAM IS REQUESTING A RESTARTABLE SORT. THE SORT WILL SAVE ITS TWO INTERNAL FILES AND CAN BE RESTARTED UPON PROGRAM REQUEST. IF BIT 2 IS ON BIT 1 IS SET BY DEFAULT. OFF - MEANS THAT A NORMAL SORT IS REQUESTED AND NO SORT FILES ARE SAVED (UNLESS BIT 2 IS ON WHICH SETS BIT 1 BY DEFAULT).

BIT 2: ON - MEANS THAT THE PROGRAM IS REQUESTING A RESTARTABLE SORT AND DESIRES EXTENSIVE ERROR RECOVERY (FROM I/O ERRORS). WITH THIS OPTION SET THE SORT WILL ATTEMPT TO BACK TRACK AND REMERGE STRINGS, AS NECESSARY, IF I/O ERRORS OCCUR WHILE ACCESSING EITHER OF THE TWO SORT FILES. TO USE THIS OPTION THE PROGRAM MUST PROVIDE AT LEAST THREE TIMES AS MUCH DISK SPACE AS REQUIRED TO CONTAIN THE INPUT DATA. WHEN LESS THAN THIS MUCH SPACE IS PROVIDED THE SORT WILL EMIT AN ERROR MESSAGE, CHANGE

TO RESTARTABLE ONLY MODE, AND CONTINUE THE SORT WITHOUT FURTHER CAPABILITY TO BACK-TRACK. OFF - MEANS THAT RECOVERY FROM INTERNAL ERRORS IS NOT REQUESTED.

BIT 3: THIS BIT HAS MEANING ONLY IF A RESTARTABLE SORT WAS REQUESTED. THE USE OF THIS OPTION CONTROLS THE SORT DURING THE STRINGING PHASE AS THE USER INPUT IS BEING READ BY THE SORT. USE OF THIS BIT WILL DETERMINE HOW THE SORT WILL RESTART (WHEN A RESTART IS REQUESTED) ONLY IF THE RESTART OCCURS WHILE THE SORT IS IN THE STRINGING PHASE.

ON - MEANS THAT THE PROGRAM DESIRES THAT THE SORT RESTART AT THE BEGINNING OF THE USERS INPUT. IT IS THE EQUIVALENT OF STARTING AN ENTIRELY NEW SORT. IN CASE THE RESTARTED SORT HAD PASSED FROM THE STRINGING PHASE INTO THE MERGE PHASE. IT WILL CONTINUE FROM THE MERGE PHASE. THIS BIT MAY BE SET DURING A RESTART EVEN IF IT WAS NOT INITIALLY SET. ONCE SET, IT CANNOT BE RESET BY SUBSEQUENT RESTARTS.

OFF - MEANS THAT THE PROGRAM DESIRES THE ABILITY TO RESTART AT THE LAST RESTART POINT THAT OCCURRED DURING THE STRINGING PHASE. IF THE SORT IS STILL IN THE STRINGING PHASE, IT WILL SKIP OVER THE RECORDS ALREADY PROCESSED AND CONTINUE FROM THE LAST RESTART POINT. THIS WILL BE DESCRIBED IN MORE DETAIL LATER. IF THE SORT IS IN THE MERGE PHASE IT WILL CONTINUE FROM THE LAST MERGE PHASE RESTART POINT. USE OF THIS OPTION (BY NOT SETTING THE BIT) IS NORMALLY LESS EFFICIENT BECAUSE MORE STRINGS ARE CREATED DURING THE STRINGING PHASE.

BIT 4: THIS BIT IS RESERVED FOR EXPANSION AND IS NOT CURRENTLY USED BY THE SORT.

WHEN A PROGRAM IS INITIALLY STARTING A SORT AND DESIRES RESTART ABILITY THE RESTART VALUE SHOULD BE:

1. DECIMAL 2 - (BIT 1 ON) IF YOU DESIRE A RESTARTABLE SORT THAT WILL BE CAPABLE OF RESTARTING AT ANY POINT DURING THE STRINGING OR MERGE PHASE.

2. DECIMAL 10-(BITS 1 AND 3 ON) IF YOU DESIRE A RESTART-

ABLE SORT THAT CAN RESTART AT ANY POINT DURING THE MERGE PHASE BUT ONLY AT THE BEGINNING OF THE STRINGING PHASE.

3. DECIMAL 4 OR 6 (BIT 2 ON OR BITS 1 AND 2 ON) IF YOU DESIRE A RESTARTABLE SORT THAT CAN ATTEMPT EXTENSIVE RECOVERY FROM INTERNAL SORT I/O ERRORS AND CAN RESTART AT ANY POINT DURING STRINGING OR MERGE PHASE.

4. DECIMAL 12 OR 14 - (BITS 2 AND 3 ON OR BITS 1, 2, AND 3 ON) IF YOU DESIRE A RESTARTABLE SORT THAT CAN ATTEMPT EXTENSIVE RECOVERY FROM INTERNAL SORT I/O ERRORS AND CAN RESTART AT ANY POINT DURING THE MERGE PHASE BUT ONLY AT THE BEGINNING OF THE STRINGING PHASE.

5. DECIMAL 1, 3, 5, OR 7-(THE SIGNIFICANT BITS ARE BIT 0 ON AND BIT 3 OFF) IF YOU DESIRE A RESTART OF A PREVIOUSLY INCOMPLETE SORT. THE PRIOR INCOMPLETED SORT MUST HAVE BEEN CAPABLE OF RESTART AND THE TWO SORT DISK FILES MUST BE PRESENT. A RESTART WILL BE ATTEMPTED USING THE VALUES OBTAINED FROM THE SORT FILES. THE PREVIOUS SETTING OF BIT 3 WILL CONTROL THE SORT IF IT IS RESTARTED DURING THE STRINGING PHASE. THE PREVIOUS VALUES OF BITS 1 AND 2 ARE USED.

6. DECIMAL 9, 11, 13, OR 15 (THE SIGNIFICANT BITS ARE BIT 0 AND BIT 3) IF YOU DESIRE A RESTART OF A PREVIOUSLY INCOMPLETE SORT AND ALSO DESIRE A RESTART FROM THE BEGINNING OF INPUT IF RESTARTED DURING THE STRINGING PHASE. THE PRIOR INCOMPLETED SORT MUST HAVE BEEN CAPABLE OF RESTART AND THE TWO SORT DISK FILES MUST BE PRESENT. A RESTART WILL BE ATTEMPTED USING THE VALUES OBTAINED FROM THE SORT FILES. BIT 3 WILL BE SET AND REMAIN SET THROUGH ALL SUBSEQUENT RESTARTS. BITS 1 AND 2 WILL TAKE ON THEIR PREVIOUS VALUES.

7. DECIMAL 0 OR 8 (NO BITS ON OR BIT 3 ON) WILL CAUSE THE SORT TO DO A NORMAL SORT WITH NO RESTART CAPABILITY

THE FOLLOWING DISCUSSION ON RESTARTING IS APPLICABLE WHEN A PROGRAM HAS REQUESTEED A RESTART AT ANY POINT DURING THE STRINGING PHASE.

RESTARTING DURING THE STRINGING PHASE (WHILE THE SORT IS STILL READING INPUT RECORDS) IS CAUSE FOR SPECIAL CONSIDERATION. IF THE SORT IS PASSED A FILE, A SEEK WILL BE DONE OR RECORDS WILL BE READ UNTIL THE DESIRED RESTART POINT IS REACHED. AN INPUT PROCEDURE PRESENTS A DIFFERENT KIND OF PROBLEM, HOWEVER, SINCE THE PROGRAM MUST FIND THE PROPER RESTART RECORD. TO ACCOMPLISH THIS DESIRED RESULT, THE SORT WILL PLACE VALUES IN THE FIRST WORD OF THE ARRAY PASSED TO THE INPUT PROCEDURE. THE VALUES WILL BE NEGATIVE OR POSITIVE INTEGERS IN BINARY FORM OR ZERO TO INDICATE THAT NOTHING SPECIAL IS HAPPENING. A POSITIVE INTEGER IS PLACED IN THE FIRST WORD (WORD 0) TO TELL THE INPUT PROCEDURE THE RELATIVE NUMBER OF THE NEXT RECORD DESIRED BY THE SORT (IF THE SORT HAS PREVIOUSLY PROCESSED AND SAVED 99 RECORDS IT WILL REQUEST RECORD NUMBER 100). A POSITIVE NON-ZERO INTEGER WILL OCCUR IN THE FIRST WORD ONLY ONCE AND THEN IT WILL BE ON THE FIRST CALL TO THE INPUT PROCEDURE. THE SORT WILL PLACE A NEGATIVE NON-ZERO INTEGER IN THE FIRST WORD TO INFORM THE INPUT PROCEDURE THAT THE SORT HAS JUST ESTABLISHED A RESTART POINT. THE NUMBER RETURNED REPRESENTS (IN ABSOLUTE VALUE) THE NUMBER OF RECORDS SAVED (FOR RESTART PURPOSES) BY THE SORT. THIS INFORMATION COULD BE USED BY THE PROGRAM TO ESTABLISH ITS OWN SEPARATE RESTART POINTS.

ERROR RECOVERY

THE MARK II.3 SORT CONTAINS EXTENSIVE ERROR RECOVERY ABILITY FOR IRRECOVERABLE I/O ERRORS THAT OCCUR AS A RESULT OF ACCESSING A SORT DISK FILE. THE SORT DISK FILES UNDER DISCUSSION ARE THE WORK FILE THAT CONTAINS THE DATA BEING SORTED AND THE CONTROL FILE THAT CONTAINS CONTROL INFORMATION FOR THE SORT. THESE TWO FILES WILL BE REFERENCED AS THE WORK FILE AND CONTROL FILE RESPECTIVELY AND THEIR INTERNAL NAMES AND EXTERNAL TITLES WILL BE DESCRIBED LATER.

I/O ERROR RECOVERY LOGICALLY SEGMENTS INTO SEVERAL AREAS OF INTEREST RELATED TO THE FILE IN QUESTION AND KIND OF ERROR ENCOUNTERED. THE DEGREE OF RECOVERY POSSIBLE IS ALWAYS DEPENDENT UPON THE REQUEST FOR ERROR RECOVERY BY THE PROGRAM. WHEN ERROR RECOVERY IS REQUESTED, THE SORT WILL MAINTAIN TWO COPIES OF EACH RECORD IN THE CONTROL FILE AND MAKE A SECOND COPY OF THE ORIGINAL STRINGS OF INPUT DATA IN THE WORK FILE. WITH ERROR RECOVERY THE CONTROL FILE IS LOGICALLY SEGMENTED INTO TWO FILES WITH A DUPLICATE RECORD MAINTAINED IN BOTH HALVES OF THE FILE WHILE THE WORK FILE IS LOGICALLY SEGMENTED INTO THIRDS. DUPLICATE RECORDS ARE CREATED IN THE WORK FILE DURING THE STRINGING PHASE ONLY AND ARE USED FOR ERROR RECOVERY WHEN THE PRIMARY COPY OF THE ORIGINAL STRING IS UNREADABLE. DATA IS NOT DUPLICATED DURING THE MERGE PHASE AND ERROR RECOVERY IS ACCOMPLISHED BY BACK-TRACKING TO REMERGE PREVIOUSLY MERGED DATA. IN NO CASE IS ERROR RECOVERY ATTEMPTED BEYOND ONE LEVEL OF RECOVERY (I.E., IF RECOVERY IS ATTEMPTED WHILE RECOVERYING FROM A PRIOR ERROR THE SORT WILL TERMINATE).

THE FOLLOWING IS A DISCUSSION OF THE PRIMARY AREAS OF ERROR RECOVERY.

1. ERROR RECOVERY OF CONTROL FILE INPUT ERRORS

   AN ATTEMPT IS MADE TO OBTAIN THE RECORD BY REREADING THE "ERROR" RECORD SEVERAL TIMES. IF THE "ERROR" RECORD IS UNREADABLE AND ERROR RECOVERY IS NOT REQUESTED THE SORT WILL TERMINATE. IF ERROR RECOVERY IS REQUESTED THE SORT WILL ATTEMPT TO READ ITS DUPLICATE COPY OF THE "ERROR" RECORD.

2. ERROR RECOVERY OF CONTROL FILE OUTPUT ERRORS

   AN ATTEMPT IS MADE TO SUCCESSFULLY WRITE THE "ERROR" RECORD. IF WRITING IS NOT SUCCESSFUL AFTER SEVERAL RETRIES AND ERROR RECOVERY IS NOT REQUESTED THE SORT WILL TERMINATE. IF ERROR RECOVERY IS REQUESTED THE SORT WILL XD THE ROW OF DISK CONTAINING THE "ERROR" RECORD. THE SORT WILL RETAIN THE OTHER COPY OF THE XD-ED ROW FOR SUBSEQUENT USE. IF POSSIBLE THE SORT

WILL CONTINUE IN FULL ERROR RECOVERY MODE, OTHERWISE, THE SORT WILL DISPLAY "SORT ERROR 31" AND CONTINUE IN ERROR RECOVERY MODE FOR THE WORK FILE WHILE FURTHER ERROR RECOVERY FOR THE CONTROL FILE IS NO LONGER POSSIBLE. IN EITHER CASE IF THE SORT IS UNABLE TO WRITE THE "ERROR" RECORD AFTER XD-ING THE BAD ROW THE SORT WILL TERMINATE. ONLY ONE DISK ROW WILL BE XD-ED UPON AN ERROR SO THAT IT IS NOT POSSIBLE TO GET INTO A LOOP AND XD LARGE QUANTITIES OF DISK.

3. ERROR RECOVERY OF WORK FILE INPUT ERRORS

AN ATTEMPT IS MADE TO OBTAIN THE RECORD BY REREADING THE "ERROR" RECORD SEVERAL TIMES. IF THE "ERROR" RECORD IS UNREADABLE AND ERROR RECOVERY IS NOT REQUESTED THE SORT WILL TERMINATE. IF ERROR RECOVERY IS REQUESTED AND THE DATA HAS BEEN DUPLICATED AN ATTEMPT IS MADE TO READ THE DUPLICATE COPY. IF THE "ERROR" RECORD WAS WRITTEN BY THE MERGE PHASE NO DUPLICATE COPY EXISTS AND THE SORT WILL ATTEMPT TO RECREATE THE STRING OF INFORMATION THAT CONTAINED THE "ERROR" RECORD. BEFORE BACK-TRACKING TO THE PREVIOUS MERGE THE SORT WILL WRITE AND READ A TEST RECORD IN THE "ERROR" RECORD LOCATION. IF THE TEST IS UNSUCCESSFUL THE SORT WILL XD THE ROW OF DISK CONTAINING THE "ERROR" RECORD. AFTER TESTING AND POSSIBLE XD-ING OF DISK THE SORT WILL BACK-TRACK TO THE DESIRED POINT FOR RESTARTING THE MERGE PHASE. ONE ROW OF DISK, AT MOST, WILL BE XD-ED FOR EACH OCCURRENCE OF AN INPUT ERROR FOR THE WORK FILE.

4. ERROR RECOVERY OF WORK FILE OUTPUT ERRORS

AN ATTEMPT IS MADE TO SUCCESSFULLY WRITE THE "ERROR" RECORD. IF WRITING IS NOT SUCCESSFUL AFTER SEVERAL RETRIES AND ERROR RECOVERY IS NOT REQUESTED THE SORT WILL TERMINATE. IF ERROR RECOVERY IS REQUESTED THE SORT WILL XD THE ROW OF DISK CONTAINING THE "ERROR" RECORD. IF POSSIBLE THE SORT WILL CONTINUE IN FULL ERROR

RECOVERY MODE, OTHERWISE THE SORT WILL DISPLAY "SORT ERROR 32" AND CONTINUE WITH ERROR RECOVERY RESET. IF THE SORT IS IN THE STRINGING PHASE, AN ATTEMPT WILL BE MADE TO WRITE THE "ERROR" RECORD AND IF THE ATTEMPT IS UNSUCCESSFUL THE SORT WILL TERMINATE. IF THE SORT IS IN THE MERGE PHASE, IT WILL BACK-TRACK TO THE DESIRED POINT OF RESTARTING THE MERGE PHASE. ONE ROW OF DISK WILL BE XD-ED AT MOST FOR EACH OCCURRENCE OF AN OUTPUT ERROR FOR THE WORK FILE.

5. ERROR RECOVERY OF USER OUTPUT FILE ERRORS

WHEN THE PROGRAM HAS GIVEN THE SORT AN OUTPUT FILE (RATHER THAN AN OUTPUT PROCEDURE), THE SORT WILL CLOSE AND PURGE THE OUTPUT FILE AND RESTART THE OUTPUT FROM THE FIRST OUTPUT RECORD. IF THE OUTPUT FILE IS A DISK FILE AND INSUFFICIENT SPACE WAS ALLOCATED TO CONTAIN THE DATA, THE SORT WILL EITHER SET THE FLEXIBLE ATTRIBUTE BEFORE RESTARTING THE OUTPUT OR TERMINATE WITH "SORT ERROR 8" IF SETTING THE FLEXIBLE ATTRIBUTE IS NOT POSSIBLE. OUTPUT ERROR RECOVERY IS NOT DEPENDENT UPON PROGRAM REQUEST FOR ERROR RECOVERY.

6. ERROR RECOVERY OF WORK FILE INPUT ERRORS DURING USER OUTPUT.

THIS TYPE OF RECOVERY IS A MIXTURE OF PARAGRAPHS 3 AND 5 ABOVE. WHEN THE USER OUTPUT IS A FILE, IT WILL BE CLOSED AND PURGED AND THE SORT WILL ATTEMPT TO REMERGE THE DESIRED STRING AS SPECIFIED IN PARAGRAPH 3. IF THE OUTPUT IS A PROCEDURE AND ERROR RECOVERY WAS SPECIFIED, THE SORT WILL REPOSITION ITSELF TO REMERGE THE DESIRED STRING AND SUBSEQUENTLY TERMINATE WITH "SORT ERROR 19". WHEN THE SORT IS RESTARTED, IT WILL REMERGE THE DESIRED STRING AND START USER OUTPUT WITH THE FIRST OUTPUT RECORD.

CORE SORTING

PRIOR TO THE MARK II.3 RELEASE THE SORT WOULD TERMINATE WITH "SORT ERROR 3" WHEN A SORT WAS REQUESTED WITH NO DISK AND NO TAPES SPECIFIED. THE PROGRAM MUST EXPLICITLY REQUEST THIS CONDITION OTHERWISE DEFAULT VALUES ARE SPECIFIED. WHEN NO DISK AND NO TAPES ARE SPECIFIED TO THE MARK II.3 SORT IT WILL BE INTERPRETED TO MEAN THAT A CORE SORT IS DESIRED. THE SORT WILL NOT OPEN ANY SORT FILES AND WILL ATTEMPT TO READ THE USER INPUT INTO MEMORY. IF SORT MEMORY IS FILLED BEFORE THE LAST USER INPUT RECORD IS READ, THE SORT WILL TERMINATE WITH "SORT ERROR 3". IF THE SORT IS ABLE TO CONTAIN ALL OF THE INPUT RECORDS, THE SORT WILL PROCEED NORMALLY TO PRODUCE USER OUTPUT. WHEN THIS TYPE OF SORT IS DESIRED, THE CORRECT SPECIFICATION FOR SORT MEMORY SIZE IS THE NUMBER OF RECORDS TO BE SORTED TIMES THE SIZE OF A RECORD (IN WORDS). FOR COBOL SORTS THE USER MUST DETERMINE THE SIZE FROM THE SD AND MUST ROUND THE SIZE UP TO THE NUMBER OF WORDS REQUIRED TO CONTAIN THE RECORD. THIS TYPE OF SORT IS OF PARTICULAR VALUE WHEN THE NUMBER OF RECORDS TO BE SORTED IS SMALL.

THE SORT DISK FILES

THE MARK II.3 SORT USES TWO DISK FILES (WHEN DISK OR ITD SORTS ARE REQUESTED) FOR CONTAINING THE DATA AND CONTROL RECORDS FOR THE SORT. PREVIOUS VERSIONS OF THE SORT UTILIZED ONLY ONE DISK FILE WHICH CONTAINED BOTH DATA AND CONTROL RECORDS. IMPLEMENTATION OF RESTART AND THE DESIRE FOR OTHER IMPROVEMENT LEAD TO THE CHOICE OF TWO SEPARATE FILES.

THE CONTROL FILE IS NORMALLY A VERY SMALL DISK FILE WHOSE SIZE IS BASED UPON THE MAXIMUM NUMBER OF STRINGS THAT COULD BE PRODUCED FOR THE SORT CURRENTLY BEING EXECUTED.

CONTROL FILE RECORDS ARE THREE WORDS AND BLOCKS ARE NINETY WORDS. THE MAXIMUM NUMBER OF ROWS OF THE CONTROL FILE IS SIXTY FOUR AND THE NUMBER OF PHYSICAL BLOCKS PER ROW IS FOUR (UNLESS THE AMOUNT OF DISK PROVIDED IS EXTREMELY LARGE). THE LAST TWO ROWS OF THE CONTROL FILE CONTAINS RESTART INFORMATION WHEN RESTARTABLE SORTS ARE REQUESTED. THE INTERNAL NAME FOR THE CONTROL FILE IS "DISKC" AND THE TITLE IS "SORT/DISKC". WHEN A RESTARTABLE SORT IS DESIRED,

YOU SHOULD USE A FILE CONTROL CARD TO GIVE A UNIQUE TITLE TO THE CONTROL FILE. AN EXAMPLE IS:

```
<I> FILE DISKC (TITLE=JOBNAME/SORTCONTROL)
```

OTHER ATTRIBUTES THAT MAY BE "SET" BY USE OF A FILE CONTROL CARD ARE LIMITED TO ASSIGNMENT OF THE CONTROL FILE TO DISK PACK. USE OF OTHER ATTRIBUTES IS NOT PROHIBITED BUT EXTREME CARE MUST BE EXERCISED ELSE THE SORT WILL NOT BE ABLE TO FUNCTION. IN FACT THERE IS A HIGH PROBABILITY OF FAILURE IF ATTRIBUTES SUCH AS AREAS, AREASIZE, MAXRECSIZE, BLOCKSIZE, ETC. ARE MODIFIED BY A FILE CONTROL CARD.

THE WORK FILE IS USED BY THE SORT TO CONTAIN THE DATA OR RECORDS BEING SORTED. WORK FILE SIZE IS PROVIDED EXPLICITELY OR IMPLICITELY BY THE USER PROGRAM. THE SORT WILL FIRST DETERMINE A DESIRED BLOCKSIZE AND THEN COMPUTE THE NUMBER OF DISK ROWS PROVIDED BY THE USER. THE MAXIMUM NUMBER OF ROWS THE SORT WILL ALLOCATE FOR THE WORK FILE IS 183 AND PARTIAL ROWS WILL BE ROUNDED UP TO A FULL ROW. ROW SIZES WILL NOT EXCEED 1320 SEGMENTS UNLESS AN EXTREMELY LARGE AMOUNT OF DISK IS PROVIDED. THE INTERNAL NAME FOR THE WORK FILE IS "DISKF" AND TITLE IS "SORT/DISKF". WHEN A RESTARTABLE SORT IS REQUESTED, YOU SHULD USE A FILE CONTROL CARD TO GIVE A UNIQUE TITLE TO THE WORK FILE. AN EXAMPLE IS:

```
<I> FILE DISKF(TITLE=JOBNAME/SORTWORK)
```

OTHER ATTRIBUTES THAT MAY BE "SET" BY USE OF A FILE CONTROL CARD ARE LIMITED TO ASSIGNMENT OF THE WORK FILE TO DISK PACK AND THE ABILITY TO PROVIDE A NEW BLOCKSIZE AND MAXRECSIZE. USE OF OTHER ATTRIBUTES IS NOT PROHIBITED BUT EXTREME CARE MUST BE EXERCISED ELSE THE SORT WILL NOT FUNCTION. THERE IS A HIGH PROBABILITY OF FAILURE WHEN ATTRIBUTES SUCH AS AREAS, AREASIZE, ETC. ARE MODIFIED BY A FILE CONTROL CARD. THE SORT WILL RECOGNIZE CHANGES IN BLOCKSIZE, HOWEVER, BLOCKSIZE AND MAXRECSIZE MUST AGREE IN ORDER TO OPEN THE FILE. THE ABILITY TO MODIFY THIS VALUE IS PROVIDED AS MEANS OF OVER RIDING THE NORMAL MEMORY ALLOCATION ALGORITHMS OF THE SORT. SOME CARE SHOULD BE EXERCISED IN THE USE OF THIS ABILITY TO MODIFY THE BUFFER SIZE OF THE SORT. WHEN THE BUFFER SIZE IS

INCREASED THE NUMBER OF DISK SEGMENTS PER DISK ROW IS PROPORTIONATELY INCREASED AND THE SORT WILL MERELY PROCEED USING THE LARGER DISK ROWS. IF THE BUFFER SIZE IS DECREASED THE NUMBER OF DISK SEGMENTS PER DISK ROW IS PROPORTIONATELY DECREASED WHICH MAY RESULT IN A WORK FILE NOT LARGE ENOUGH TO ACCOMPLISH THE SORT. ONE METHOD OF ALLEVIATING THIS CONDITION IS TO SPECIFY A LARGER QUANTITY OF DISK. MODIFICATION OF THE BUFFER SIZE IS USUALLY LESS EFFECTIVE THAN PROVIDING A DIFFERENT MEMORY SIZE FOR USE BY THE SORT. HOWEVER, SORTS ARE ALWAYS DATA DEPENDENT AND IT IS POSSIBLE WITH SOME ORDERING OF DATA A BETTER SORT COULD BE OBTAINED BY JUDICIOUS SELECTION OF BUFFER SIZE.

SORT MEMORY ALLOCATION

THE MARK II.3 SORT WILL ATTEMPT TO STAY WITHIN THE MEMORY ESTIMATE PROVIDED BY THE USER EVEN TO THE EXTENT OF SORTING WITH ONLY SIX RECORDS IN MEMORY. MEMORY ALLOCATION PROCEEDS (IN GENERAL) THROUGH THE FOLLOWING STEPS:

1. MEMORY SIZE PROVIDED BY THE PROGRAM IS REDUCED BY 1500 WORDS. THE REDUCED SIZE IS USED FOR ALL SUBSEQUENT CALCULATIONS. THE REDUCTION IS A GENEROUS ESTIMATE OF THE AMOUNT OF SPACE REQUIRED FOR CERTAIN WORKING STORAGE AND THE SPACE REQUIRED FOR VARIOUS SORT PROCEDURES.

2. A BUFFER SIZE IS SELECTED FOR THE SORT-S INTERNAL DISK AND/OR TAPE FILES. THE SORT WILL TRY TO SELECT BUFFER SIZES SUCH THAT THE SORT WILL NOT BE I/O BOUND. FOR DISK SORTING, THE SORT WILL NORMALLY ALLOCATE TWO BUFFERS PER STRING. FOR TAPE SORTING WITH N TAPES, THE SORT WILL ALLOCATE 1/N OF MEMORY AS BUFFERS FOR EACH TAPE.

3. DURING EXECUTION OF THE STRINGING PHASE, TWO OUTPUT BUFFERS ARE NORMALLY ALLOCATED THUS LEAVING THE REST OF MEMORY FOR THE SORT VECTOR. DURING EXECUTION OF THE MERGE PHASE, VIRTUALLY ALL OF AVAIL-

ABLE MEMORY IS USED TO CONTAIN BUFFERS.

THE MARK II.2 SORT (AND PREVIOUS SORT RELEASES) COMPUTED BUFFER SIZES FIRST AND THEN ATTEMPTED TO FIT BUFFERS INTO AVAILABLE MEMORY. FOR INSTANCES WHERE SORT RECORDS WERE VERY LARGE AND MEMORY ALLOCATED WAS RELATIVELY SMALL IT WAS POSSIBLE THAT THE MEMORY USED COULD EXCEED THE PROGRAM MEMORY ESTIMATE. BECAUSE OF THE CHANGES TO THE MEMORY ALLOCATION ALGORITHMS FOR THE MARK II.3 SORT, THIS CONDITION SHOULD NO LONGER OCCUR. AS A CONSEQUENCE OF THE NEW MEMORY ALLOCATION THERE MAY EXIST SOME SMALL NUMBER OF SORTS THAT WILL REQUIRE MORE ELAPSED TIME TO COMPLETE THE SORT. THIS SITUATION MAY BE EASILY SOLVED BY CHANGING THE PROGRAM TO PROVIDE A LARGER MEMORY SIZE FOR THE SORT TO UTILIZE.

SUGGESTIONS FOR MORE EFFICIENT SORTING

SELECTION OF SORT PARAMETERS IS NOT ALWAYS A SIMPLE MATTER WHEN THE AMOUNT OF DATA VARIES WIDELY FROM RUN TO RUN AND THE USER IS CONCERNED WITH TOTAL SYSTEM EFFICIENCY. SORTING CAN BE A SIGNIFICANT PART OF THE WORK LOAD OF A COMPUTER INSTALLATION AND SHOULD BE USED JUDICIOUSLY. IN MANY INSTANCES DEFAULT PARAMETERS ARE USED WHEN THEY DO NOT PROVIDE EFFICIENT UTILIZATION OF THE SYSTEM RESOURCES. THE RELATIVE PRIORITY OF THE JOB SHOULD BE A DEFINITE CONSIDERATION IN THE SELECTION OF SORT PARAMETERS. MEMORY SIZE IS CERTAINLY THE MOST VOLATILE PARAMETER GIVEN THE SORT. SINCE THE SORT IS CAPABLE OF HAVING A 65535 RECORD SORT VECTOR AND CAPABLE OF DOING A 65535 ORDER MERGE WITH AS MANY AS 256 BUFFERS PER STRING, IT IS EASY TO SEE THAT THE SORT CONTAINS VIRTUALLY NO UPPER LIMIT ON THE AMOUNT OF MEMORY THAT COULD BE UTILIZED. MOST COMPUTER SYSTEMS HAVE LESS THAN 1.099 TRILLION WORDS OF MEMORY SO THAT SOMETHING LESS THAN MAXIMUM SORTS WILL HAVE TO BE CONSIDERED. PROVIDING MORE MEMORY (UNTIL THE SORT IS DONE ENTIRELY IN MEMORY) SHOULD YIELD FASTER AND FASTER SORTS. IT HAS BEEN DEMONSTRATED THAT SOME SETS OF DATA WILL REACH A POINT WHERE SLOWER SORTS WILL OCCUR AS MEMORY SIZE IS INCREASED. A POINT OF DIMINISHING RETURN WILL PROBABLY OCCUR BEFORE AN ENTIRELY CORE SORT IS REACHED. SOME VERY GENERAL GUIDELINES ARE SUGGESTED FOR DISK SORTING -

FAST SORTS - MEMORY SIZE SHOULD PROVIDE ENOUGH SPACE TO CONTAIN AT LEAST 2,000 RECORDS.

REASONABLY FAST SORTS - MEMORY SIZE SHOULD PROVIDE ENOUGH SPACE TO CONTAIN AT LEAST 1,200 RECORDS.

ADEQUATE SORTS - CAN USUALLY BE OBTAINED WHEN THE MEMORY SIZE PROVIDES ENOUGH SPACE FOR 600 RECORDS.

TO USE THE ABOVE GUIDELINES YOU MUST CONVERT THE SORT RECORD SIZE TO THE NUMBER OF WORDS REQUIRED TO CONTAIN A SINGLE RECORD. FOR EXAMPLE A ONE CHARACTER RECORD REQUIRES ONE WORD AND 15 6-BIT CHARACTERS REQUIRE TWO WORDS WHILE 15 8-BIT CHARACTERS REQUIRE THREE WORDS. WHEN RECORD SIZE IS CONVERTED TO WORDS, YOU MUST ADD THREE ADDITIONAL WORDS TO RECORD SIZE (USED BY SORT) AND THEN MULTIPLY THIS NEW RECORD SIZE BY THE DESIRED NUMBER OF RECORDS. AFTER MEMORY HAS BEEN COMPUTED FOR NUMBER OF RECORDS TIMES RECORD SIZE, YOU MUST ADD 1500 WORDS FOR SORT WORKING SPACE.

SOME VERY GENERAL GUIDELINES FOR TAPE SORTING ARE:

FAST SORTS - MEMORY SIZE SHOULD PROVIDE ENOUGH SPACE FOR 300 RECORDS PER WORK TAPE.

REASONABLY FAST SORT - MEMORY SIZE SHOULD PROVIDE ENOUGH SPACE FOR 200 RECORDS PER WORK TAPE.

ADEQUATE SORT - MEMORY SIZE SHOULD PROVIDE ENOUGH SPACE FOR 100 RECORDS PER WORK TAPE.

RECORD SIZE MUST BE CONVERTED TO WORDS WITH THREE ADDITION WORDS ADDED (AS STATED ABOVE) AND THEN MULTIPLIED BY THE NUMBER OF TAPES SPECIFIED IN THE SORT STATEMENT. AGAIN 1500 WORDS MUST BE ADDED TO PROVIDE FOR SORT WORKING SPACE.

TAPE SORTS ARE SIMILAR TO DISK SORTS IN THAT PROVIDING MORE MEMORY SHOULD YIELD FASTER SORTS. HOWEVER, THE POINT OF DIMINISHING RETURN IS MORE DATA DEPENDENT FOR TAPE SORTING. IT IS GENERALLY

BETTER TO USE MORE SORT WORK TAPES THAN TO PROVIDE LARGE ADDITIONAL INCREMENTS OF MEMORY. PROVIDING MORE MEMORY AND MORE TAPES IS THE IDEAL WHEN SPEED IS THE MOST IMPORTANT FACTOR. TAPE SORTS SHOULD RUN VERY SATISFACTORILY IN THE BACKGROUND WHILE OTHER JOBS ARE IN THE MIX. USERS ARE STRONGLY ADVISED TO USE GOOD TAPES AND TAPE DRIVES IN GOOD WORKING CONDITION WHEN DOING TAPE SORTS.

ITD SORTS ARE CAPABLE OF IMPROVING TAPE SORTS BY A SUBSTANTIALLY LARGE FACTOR (50% OR MORE). THE REASON FOR THIS DEGREE OF IMPROVEMENT IS THAT FEWER STRINGS ARE CREATED ON TAPE WHICH CAUSES TAPE MERGING TO BE COMPLETED MUCH SOONER. THE AMOUNT OF IMPROVEMENT DEPENDS UPON THE INHERENT SEQUENCE OF THE DATA AND THE AMOUNT OF DISK PROVIDED. IN MOST CASES, 100,000 WORDS OF DISK (OR LESS) WILL BE SUFFICIENT TO OBTAIN THE INCREASED SPEED FROM AN ITD SORT.

DISK SORTING SPEEDS CAN BE AFFECTED BY THE AMOUNT OF DISK SPECIFIED BY THE PROGRAM. THIS UNHAPPY CIRCUMSTANCE RESULTS FROM THE SORT BEING UNABLE TO MERGE AS MANY DISK STRINGS (AS IT WOULD WANT TO DO) BECAUSE INSUFFICIENT DISK IS AVAILABLE TO CONTAIN THE OUTPUT OF THE MERGE PHASE. THE SORT, THEREFORE, PROCEEDS TO MERGE FEWER STRINGS (WHEN POSSIBLE) AND THE PROGRAM RUNS A HIGH DEGREE OF RISK OF BEING TERMINATED WITH A "SORT ERROR 5". ESTIMATING A PROPER AMOUNT OF DISK IS DIFFICULT BECAUSE OF THE GAPS CREATED BY UNFILLED BUFFERS AT THE END OF STRINGS. HOWEVER, A METHOD FOR ESTIMATING DISK SPACE IS SUGGESTED:

1. CONVERT THE RECORD SIZE TO WORDS AS STATED ABOVE. (DO NOT ADD THREE ADDITIONAL WORDS.)

2. MULTIPLE THE RECORD SIZE (IN WORDS) BY THE NUMBER OF RECORDS TO BE SORTED.

3. THE NUMBER OBTAINED BY STEP 2 SHOULD THEN BE MULTIPLIED BY:

   A. 2.25 TO OBTAIN A SAFE ESTIMATE
   B. 1.5 TO OBTAIN A NEAR MINIMUM ESTIMATE
   C. 3.5 OR MORE IF YOU ARE DOING A RESTARTABLE

MUCH OF THE MEMORY USED BY THE SORT IS NON-OVERLAYABLE OR SAVE SPACE. FOR THIS REASON IT IS EASY TO SEE THAT THE SORT CAN HAVE A DEFINITE IMPACT UPON THE THROUGHPUT OF OTHER JOBS THAT ARE EXECUTING CONCURRENTLY WITH THE SORT. SORTING PROGRAMS THAT CONTAIN LENGTHY INPUT OR OUTPUT PROCEDURES CAN CONTRIBUTE IN LARGE MEASURE TO THIS CONDITION. IT IS NOT SUGGESTED THAT THE USE OF INPUT OR OUTPUT PROCEDURES BE DISCONTINUED, BUT THAT THEY BE CONSIDERED IN PROPER PERSPECTIVE FOR THE JOB THAT IS TO BE ACCOMPLISHED. OVERALL SYSTEM PERFORMANCE CAN BE IMPROVED IN SOME CASES BY HAVING THE INPUT PROCEDURE PRODUCE A FILE THAT IS READ BY THE SORT AND HAVING THE SORT PRODUCE A FILE THAT IS PROCESSED BY THE OUTPUT PROCEDURES. THE PROCESS OF CALLING INPUT OR OUTPUT PROCEDURES DOES NOT PRESENT ANY UNDUE BURDEN TO THE SORT OR THE SYSTEM, BUT IT DOES REQUIRE AT LEAST ONE ADDITIONAL PROCEDURE ENTRY.

BLOCKING FACTORS OF INPUT AND OUTPUT FILES CAN HAVE A DEFINITE EFFECT UPON SORT TIMINGS EVEN IF INPUT OR OUTPUT PROCEDURES ARE USED TO READ OR WRITE THE FILES. TO PREVENT THIS I/O BOUND SITUATION THE OUTPUT FILE SHOULD CONTAIN APPROXIMATELY 100 (OR MORE) RECORDS PER BLOCK WHILE 50 (OR MORE) RECORDS PER BLOCK IS USUALLY SUFFICIENT FOR THE INPUT FILE. SORTS HAVE BEEN OBSERVED TO RUN AS MUCH AS FOUR TIMES FASTER WHEN INPUT AND/OR OUTPUT FILE BLOCKING WAS IMPROVED. OTHER CASES COULD BE DERIVED THAT PROVIDE BOTH LARGER AND SMALLER IMPROVEMENTS.

HISTORICALLY, SORTS HAVE BEEN CHARACTERIZED AS BEING PROCESSOR BOUND DURING THE STRINGING PHASE AND I/O BOUND DURING THE MERGE PHASE. THE MARK II.3 SORT CAN AND WILL OPERATE IN THIS FASHION, HOWEVER, IT WILL ALWAYS ATTEMPT TO BE PROCESSOR BOUND IN BOTH STRINGING AND MERGE PHASE. UNLESS THE MEMORY SIZE SPECIFIED IS RELATIVELY SMALL (FOR THE PARTICULAR SORT), IT WILL ACHIEVE THE GOAL OF BEING PROCESSOR BOUND. WHEN THIS CONDITION IS OBTAINED, SPEED IMPROVEMENTS CAN BE REALIZED ONLY BY METHODS THAT REDUCE PROCESSOR TIME. WHEN INPUT OR OUTPUT PROCEDURES ARE USED, THEY ARE CANDIDATES FOR THIS KIND OF IMPROVEMENT. THE LARGEST POTENTIAL GAIN LIES IN IMPROVEMENT OF THE COMPARE PROCEDURE SINCE IT IS

CALLED MANY TIMES FOR EACH RECORD WHILE INPUT OR OUTPUT PROCEDURES ARE CALLED ONCE FOR EACH RECORD. PROGRAMS CAN OBTAIN IMPROVEMENT BY SIMPLIFYING THE INDIVIDUAL KEYS AND CONSOLIDATING THEM INTO A SINGLE KEY. PARTIAL WORD COMPARES ARE NORMALLY FASTER THAN STRING COMPARES WHEN CHARACTERS WITHIN A WORD ARE TESTED. COBOL SORTS SHOULD USE 8-BIT CHARACTERS FOR STRING COMPARISON (TRANSLATE THEM TO 8-BIT ON INPUT AND BACK TO 6-BIT ON OUTPUT) WHENEVER POSSIBLE. XALGOL PROGRAMS SHOULD USE THE SAME TECHNIQUE FOR STRING COMPARISONS AND SHOULD CONSIDER BINDING TO COMPARE PROCEDURES WRITTEN IN ALGOL. DECREASING THE AMOUNT OF PROCESSOR TIME HELPS SYSTEM THROUGHPUT AS WELL AS REDUCING SORT TIMINGS. WITH CERTAIN KINDS OF SORTS, THAT MAY BE CHARACTERIZED BY SMALL KEYS AND LARGE RECORD SIZES, A TYPE OF SORTING KNOWN AS TAG SORTING MAY BE ADVANTAGEOUS. THE BASIC TECHNIQUE IS FOR THE PROGRAM TO PASS ONLY THE SORT KEY (FROM THE RECORD) AND A RELATIVE RECORD NUMBER OF WHERE THE RECORD RESIDES (IN DISK STORAGE) TO THE SORT. THE SORT WILL RUN FASTER SINCE IT IS SORTING MUCH SMALLER RECORDS. THE TOTAL AMOUNT OF DISK REQUIRED TO COMPLETE THE JOB WILL PROBABLY BE SMALLER ALSO SINCE THE SORT WILL REQUIRE LESS. RETRIEVING THE OUTPUT RECORDS WILL LIKELY BE THE MOST TIME CONSUMING FACTOR AND WILL BE HIGHLY DATA DEPENDENT. ONE METHOD THAT THE PROGRAM CAN EMPLOY IS:

1. DECLARE SEVERAL BUFFERS FOR USE BY THE PROGRAM TO READ ITS ORIGINAL COPY OF THE DATA. FOR PURPOSES OF THIS DISCUSSION WE WILL ASSUME WE HAVE DECLARED 5 BUFFERS.

2. USING THE FIRST 5 RECORDS OF SORTED OUTPUT WE WILL ISSUE SEEKS TO FILL THE 5 BUFFERS.

3. USING THE 6TH AND ALL SUBSEQUENT RECORDS WE WILL:

   A. READ THE RECORD THAT WE DID A SEEK TO OBTAIN (I.E., IF WE DID A SEEK FOR RECORD 1 WE WILL NOW READ RECORD 1).
   B. WRITE THE RECORD WE READ TO OUR OUTPUT FILE.
   C. ISSUE A SEEK FOR THE LATEST RECORD OBTAINED

FROM THE SORT OUTPUT.

KNOWLEDGE OF THE PARTICULAR SORT AND EXPERIEMENTATION CAN PROVIDE BETTER RETRIEVAL METHODS. UNFORTUNATELY LITTLE EVIDENCE CURRENTLY EXISTS AS AN AID FOR SUGGESTION OF LIKELY CANDIDATES FOR TAG SORTING.

A COMMENT ON INPUT, OUTPUT, AND COMPARE PROCEDURES

THE SORT FUNCTIONS BY CALLING INPUT, OUTPUT, OR COMPARE PROCEDURES (AS APPROPRIATE) THAT ARE CONTAINED IN THE USER PROGRAM. THE SORT PASSES ARRAY DESCRIPTORS OR POINTERS TO THE DESIRED PROCEDURE OF THE USER PROGRAM. THE DESCRIPTOR OR POINTER WILL NORMALLY POINT TO A RECORD AREA THAT IS A PORTION OF AN ARRAY LARGE ENOUGH TO CONTAIN MANY RECORDS. IF THE USER PROGRAM IS NEGLIGENT IN ACCESSING THE RECORD AREA PASSED TO HIS PROGRAM, IT IS POSSIBLE THAT HE CAN ACCESS INTO ADJACENT RECORD AREAS AND EITHER COMPARE INCORRECTLY OR MODIFY RECORD CONTENT INADVERTENTLY. A SEGMENTED ARRAY ERROR, INVALID INDEX, OR INVALID OPERATOR ARE OTHER LIKELY CONSEQUENCES OF SUCH ACTION. IT IS UNLIKELY THAT THIS CONDITION COULD OCCUR IN A COBOL PROGRAM, BUT ALGOL OR XALGOL PROGRAMS COULD PERPETRATE THIS OFFENCE.

SORT ERROR MESSAGES

SORT ERROR MESSAGES ARE CHANGED SOMEWHAT FOR THE MARK II.3 SORT. SOME NEW ERROR NUMBERS ARE ADDED FOR CLARITY AND TO REFLECT NEW ERROR CONDITIONS. A NEW TYPE OF ERROR IS IDENTIFIED THAT IS NON-TERMINAL ALTHOUGH AN ERROR IS DISPLAYED. ALL ERROR MESSAGES ARE DISPLAYED ON THE SYSTEM DISPLAY IN THE FORM:

<JOB NUMBER> SORT ERROR #NN

THE ERROR NUMBERS (NN) ARE:

1. THE RECORD SIZE SPECIFIED WAS IN THE RANGE OF -1/2 <RECORD SIZE<1/2

2. COUNT OF SORT INPUT RECORDS AND OUTPUT RECORDS DOES NOT AGREE. (INTERNAL SORT PROBLEM)

3. INSUFFICIENT MEMORY SPECIFIED FOR A CORE SORT. DISK SIZE AND NUMBER OF TAPES ARE BOTH ZERO.

4. SORT DISK WAS EXHAUSTED DURING THE STRINGING PHASE AND NO TAPES WERE SPECIFIED.

5. SORT DISK WAS EXHAUSTED DURING THE MERGE PHASE AND NO TAPES WERE SPECIFIED.

6. INPUT FILE PASSED TO SORT WAS ALREADY OPEN. THE INPUT FILE MUST BE CLOSED WHEN PASSED TO SORT.

7. DURING A TAPE OR ITD SORT THE BLOCK NUMBER OF THE LAST RECORD READ FROM A SORT WORK TAPE DID NOT MATCH THE EXPECTED BLOCK NUMBER.

8. THE OUTPUT FILE PASSED TO THE SORT WAS NOT LARGE ENOUGH TO CONTAIN THE OUTPUT AND THE SORT WAS UNABLE TO EXPAND THE OUTPUT FILE. INCREASE THE SIZE OF THE OUTPUT FILE OR DECREASE THE NUMBER OF RECORDS TO BE SORTED.

9. THE OUTPUT FILE PASSED TO SORT WAS ALREADY OPEN. THE OUTPUT FILE MUST BE CLOSED WHEN PASSED TO SORT.

10. AN IRRECOVERABLE I/O ERROR OCCURRED WHILE READING A SORT WORK TAPE OR WORK DISK FILE.

11. AN IRRECOVERABLE I/O ERROR OCCURRED WHILE WRITING A SORT WORK TAPE OR WORK DISK FILE.

12. AN IRRECOVERABLE I/O ERROR OCCURRED WHILE READING OR WRITING CONTROL RECORDS IN THE SORT CONTROL FILE.

13. AN IRRECOVERABLE I/O ERROR OCCURRED WHILE WRITING THE USER OUTPUT FILE.

14. AN IRRECOVERABLE I/O ERROR OCCURRED WHILE READING THE USER INPUT FILE.

15. A RESTART WAS ATTEMPTED BUT THE RECORD SIZE (OR CHARACTER SIZE OF THE RECORD) DID NOT MATCH THE ORIGINATING SORT. WHEN A RESTARTABLE SORT IS NOT ABLE TO CONTINUE IT WILL SAVE RESTART

INFORMATION SUCH THAT THIS ERROR WILL OCCUR UPON SUBSEQUENT RESTART ATTEMPTS.

16. A RESTART WAS ATTEMPTED AND THE USER INPUT FILE REACHED END-OF-FILE BEFORE THE RESTART RECORD WAS READ. THE USER INPUT FILE IS SHORTER AT RESTART TIME THAN THE ORIGINAL FILE.

17. A RESTART WAS ATTEMPTED BUT THE SORT WAS UNABLE TO OBTAIN THE NECESSARY RESTART INFORMATION FROM THE CONTROL FILE. MAY BE DUE TO PARITY ERRORS, ETC.

19. THE SORT IS TERMINATING BECAUSE AN IRRECOVERABLE ERROR OCCURRED WHILE READING THE SORT WORK FILE AND SOME OUTPUT RECORDS HAVE ALREADY BEEN PASSED TO THE USER OUTPUT PROCEDURE. THIS ERROR TERMINATION WILL ONLY OCCUR FOR RESTARTABLE SORTS WITH OUTPUT PROCEDURES. SINCE THE SORT IS RESTARTABLE, YOU MEARLY RESTART AND THE SORT WILL DO THE NEC- ESSARY RECOVERY.

84. THIS ERROR OCCURS BECAUSE THE CONTROL FILE IS NOT LARGE ENOUGH TO CONTAIN ALL OF THE CONTROL RECORDS. IT IS AN INTERNAL SORT ERROR AND CAN BE CIRCUMVENTED BY SPECIFYING MORE DISK AND/OR A DIFFERENT MEMORY SIZE.

THE FOLLOWING WILL APPEAR ON THE DISPLAY JUST LIKE ANY OTHER SORT ERROR MESSAGE. THE SORT WILL NOT TERMINATE AS A DIRECT RESULT OF THESE ERRORS.

30. THE CONTROL FILE IS NOT LARGE ENOUGH TO CONTAIN TWO COPIES OF THE CONTROL RECORDS. TWO COPIES ARE MAINTAINED FOR SORTS WITH ERROR RECOVERY. THIS IS AN INTERNAL SORT PROBLEM AND CAN BE CIRCUMVENTED BY SPECIFYING MORE DISK AND/OR A DIFFERENT MEMORY SIZE. THE SORT WILL CONTINUE WHEN THIS ERROR OCCURS, WITH THIS FUNCTION OF ERROR RECOVERY DISABLED.

31. AN IRRECOVERABLE I/O ERROR OCCURRED WHILE WRITING THE CONTROL FILE FOR AN ERROR RECOVERY SORT. ONE COPY OF THE CONTROL RECORDS ARE DISCARDED AND THE SORT WILL CONTINUE USING ONE COPY OF THE CONTROL RECORDS.

32. AN IRRECOVERABLE I/O ERROR OCCURRED WHILE WRITING THE SORT

WORK FILE. ERROR RECOVERY MODE IS ABANDONED AND THE SORT WILL CONTINUE AS A NORMAL RESTARTABLE SORT.

33. INSUFFICIENT DISK SPACE WAS PROVIDED FOR THE WORK FILE TO CONTAIN THREE COPIES OF THE DATA. THIS ONLY HAPPENS WHEN ERROR RECOVERY MODE IS REQUESTED. THE SORT WILL CONTINUE AS A NORMAL RESTARTABLE SORT WITH ERROR RECOVERY RESET.

MISCELLANEOUS INFORMATION

A NUMBER OF MODIFICATIONS WERE PLACED IN THE SORT WHEN RESTART AND ERROR RECOVERY WAS IMPLEMENTED. SOME MAY BE DOCUMENTED ELSEWHERE AND THE FOLLOWING IS AN ATTEMPT TO MENTION THE MOST SALIENT AT THE RISK OF SOME DUPLICATION.

A - THE SORT VECTOR SIZE LIMIT HAS BEEN INCREASED FROM 2048 TO 65535 AND IS NO LONGER REQUIRED TO BE A POWER OF TWO.

B - THE ORDER OF MERGE LIMIT HAS BEEN INCREASED FROM 128 TO 65535.

C - A PREVIOUS SORT LIMITATION OF 524,287 RECORDS (AS THE MAXIMUM STRING SIZE) HAS BEEN ELIMINATED AND THE ASSOCIATED ERROR NUMBERS (40, 41, AND 42) HAVE BEEN REMOVED. THE NEW LIMIT FOR A DISK STRING IS 549,775,813,887 RECORDS AND THE MAXIMUM NUMBER OF BLOCKS THAT CAN BE CONTAINED IN THE SORT WORK FILE IS ALSO 549,775,813,887. THE NEW TAPE LIMIT ALLOWS AN UNLIMITED STRING LENGTH BUT LIMITS THE NUMBER OF BLOCKS THAT CAN BE OBTAINED ON ANY SINGLE WORK TAPE (REEL SWITCHES MAY OR MAY NOT OCCUR BUT FOR THIS DISCUSSION CONSIDER A VERY LARGE TAPE REEL) TO THE RANGE OF 2,097,151 TO 4,294,967,295 DEPENDING ON THE NUMBER OF RECORDS PER BLOCK. PLEASE NOTE THAT THE ULTIMATE LIMIT FOR TAPE SORTING IS THE STATED TAPE LIMIT TIMES THE NUMBER OF SORT WORK TAPES SPECIFIED.

D - THE FILE TITLE(S) FOR SORT WORK TAPES HAVE BEEN CHANGED FROM "SORT/TAPEN" TO "SORTATAPEN" (WHERE N IS A NUMBER BETWEEN ZERO AND SEVEN). THIS CHANGE WAS IMPLEMENTED TO ELIMINATE THE NECESSITY FOR OPERATOR INTERVENTION TO RESOLVE "DUP FILE"

MESSAGES.

E - WHEN THE SORT HAS BEEN GIVEN SOME WORK DISK TO USE, IT WILL ATTEMPT TO RECOGNIZE THE CONDITION WHERE THE INPUT DATA IS LESS THAN 40 PER CENT IN SEQUENCE AND WILL SWITCH COMPARISON FROM ASCENDING TO DESCENDING OR VICE VERSA. THE SORT WILL REMEMBER THE CHANGE OF MODE AND PROCESS THE DATA ACCORDINGLY. THE SWITCH WILL BE DONE AS OFTEN AS NECESSARY IN ORDER TO PRODUCE LONGER STRINGS. GIVEN A SET OF INPUT DATA IN EXACT REVERSE SEQUENCE, THE SORT WILL PRODUCE TWO STRINGS (RATHER THAN THE MAXIMUM USER OF STRINGS) AND COMPLETE THE SORT MUCH FASTER THAT BEFORE.

F - RESTART AND ERROR RECOVERY CAPABILITY CAN BE EXCLUDED FROM THE SORT BY RECOMPILING AND BINDING THE SORT. THE METHOD HAS BEEN PREVIOUSLY DOCUMENTED. OMITTING THE CODE ASSOCIATED WITH RESTART/ERROR RECOVERY SHOULD RESULT IN SORTS THAT RUN BETWEEN TWO AND 15 PER CENT FASTER WITH AN AVERAGE IMPROVEMENT OF SIX TO 10 PER CENT. DISK AND ITD SORTS ARE THE ONLY SORTS MEASUREABLY IMPROVED BY THIS OMISSION.

G - DISK BUFFER SIZE AND DISK RECORD ADDRESSING HAS BEEN SPECIFICALLY CHOSEN TO REDUCE DISK LATENCY FOR THE SORT WORK DISK. WHEN THE PROCESSOR IS RUNNING AT THE HIGHEST CLOCK RATES AND HIGHEST SPEED MEMORY IS USED, THE SORT SHOULD BE SLIGHTLY I/O BOUND WHEN USING 40 MIL DISK. WHETHER THE SORT IS I/O BOUND OR COMPUTE BOUND IS ALSO DEPENDENT UPON MANY OTHER ASSOCIATED FACTORS, BUT THIS SMALL INSIGHT MAY PROVE HELPFUL.

H - BECAUSE OF THE NATURE OF THE CHANGES TO THE MARK II.3 SORT, VIRTUALLY ALL SORTS WILL EXECUTE IN A DIFFERENT AMOUNT OF TIME THAN PREVIOUS SORT RELEASES. EXPERIENCE TO DATE HAS INDICATED THAT MOST WILL RUN AT LEAST 10 PER CENT FASTER AND IMPROVEMENTS OF 50 PER CENT ARE NOT UNCOMMON. HOWEVER, 15 TO 20 PER CENT IS APPROXIMATELY THE AVERAGE IMPROVEMENT (ADDITIONAL IMPROVEMENT IS POSSIBLE IF RESTART/ERROR RECOVERY IS OMITTED). TAPE SORTS SHOULD YIELD A GREATER IMPROVEMENT BUT INSUFFICIENT TIMING COMPARISONS EXIST TO PREDICT AN AMOUNT THAT COULD BE

ANTICIPATED. SOME SORTS MAY EVEN RUN SLOWER (ALTHOUGH NONE THAT HAVE BEEN TESTED HAVE RUN SLOWER) SINCE THE SORT ATTEMPTS TO STAY WITHIN THE USER MEMORY SIZE SPECIFICATION. ENLARGING THE SORT VECTOR HAS THE EFFECT OF SHIFTING MORE OF THE BURDEN OF SORTING TO THE STRINGING PHASE SO DO NOT BE UNDULY ALARMED IF THE SORT DOES NOT APPEAR TO BE READING YOUR INPUT TAPE AS FAST AS IT USED TO READ IT. ANY ADDITIONAL TIME SPENT IN BUILDING LONGER STRINGS IS WELL COMPENSATED FOR BY REQUIRING LESS TIME DURING THE MERGE PHASE.

D0036 RJE - RJE AUTODIALOUT - 08-17-72

TO IMPLEMENT THE CALL BACK FUNCTION, A NEW RJE MESSAGE, PH <PHONE NUMBER>, AND A NEW STATION OPTION, CALLBACK, HAVE BEEN IMPLEMENTED. IN ADDITION, THE FUNCTION OF THE STATION LOGOFF MESSAGE, BYE, HAS BEEN MODIFIED.

THE PH INPUT MESSAGE WILL ACCEPT A TELEPHONE NUMBER CONSISTING OF A MAXIMUM OF 11 DECIMAL DIGITS. THIS CAUSES RJE TO ASSOCIATE THAT TELEPHONE NUMBER WITH THE STATION, AND RJE WILL USE THAT NUMBER TO DIALOUT SHOULD AN AUTOMATIC DIALOUT BE REQUIRED. TO INTERROGATE THE PRESENTLY ACTIVE PHONE NUMBER, A USER MAY TYPE IN PH WITH NO NUMBER FOLLOWING.

THE CALLBACK OPTION CAN BE SET, RESET, AND INTERROGATED VIA THE SO, RO, AND TO MESSAGES IN THE SAME MANNER AS ANY OTHER STATION OPTION. THE DEFAULT SETTING OF CALLBACK IS RESET.

THE BYE MESSAGE WILL LOG THE STATION OFF, DS ANY RSVP JOBS, AND ALLOW ALL OTHER JOBS TO CONTINUE. (IF RUNNING IN THE LOGON MODE, NO FURTHER INPUT WILL BE ACCEPTED EXCEPT THE PROPER LOGON SEQUENCE.) IF CALLBACK IS RESET, PRINTER/BACKUP IS NOT RUNNING, AND THE LINE IS DECLARED TO BE SWITCHED IN NDL, RJE WILL PERFORM A DISCONNECT. IF PRINTER/BACKUP IS RUNNING, THE DISCONNECT WILL BE DELAYED UNTIL IT COMPLETES.

WHEN CALLBACK IS SET, THE FUNCTION OF THE BYE MESSASE DEPENDS ON THE LINE TYPE FOR THAT STATION, EITHER NON-SWITCHED, DIALIN ONLY,

OR DIALOUT.

1) NON-SWITCHED - PRINTER OUTPUT WILL BE PRINTED AS IT BECOMES AVAILABLE.

2) DIALIN ONLY - A DISCONNECT WILL OCCUR ONLY WHEN THERE ARE NO ACTIVE JOBS AND NO PRINTER OUTPOT TO BE PRINTED. IF, AFTER A DISCONNECT OCCURS, PRINTER OUTPUT BECOMES AVAILABLE, E.G., DIRECTED BACKUP, THIS OUTPUT WILL BE PRINTED ONLY WHEN A MANUAL DIALIN IS EFFECTED.

3) DIALOUT - A DISCONNECT WILL OCCUR IF NO PRINTER/BACKUP IS RUNNING. IF PRINTER/BACKUP IS RUNNING, THE DISCONNECT WILL BE DELAYED UNTIL IT GOES TO EOJ. WHEN PRINTER OUTPUT BECOMES AVAILABLE, RJE WILL AUTOMATICALLY DIALOUT TO THE STATION, PRINT THE OUTPUT, AND AUTOMATICALLY DISCONNECT.

FOR RJE TO GO TO EOJ, THE FOLLOWING CONDITIONS MUST NOW BE TRUE:

1) ALL STATIONS MUST BE SIGNED-OFF;

2) RJE MUST NOT HAVE ANY ACTIVE TASKS;

3) NO STATIONS MUST BE IN THE PROCESS OF SIGNING-ON;

4) RJE MUST NOT BE IN A WAITING STATE FOR EITHER A CONNECT OR DISCONNECT (I.E., RJE HAS EITHER GIVEN A DIALOUT REQUEST OR IS PREPARING TO DISCONNECT).

D0037 MCS LOGGING - 06-13-72

IN ORDER TO PERMIT LOGGING OF INFORMATION PECULIAR TO MULTIPLE-USER MCS-S (E.G., CANDE, RJE), THE FOLLOWING STEPS HAVE BEEN TAKEN:

1) A NEW LOG ENTRY HAS BEEN DEFINED (MCSMSG, TYPE = 263);

2) A NEW INSTALLATION INTRINSIC, SYSTEMLOG, HAS BEEN IMPLEMENTED TO INTERFACE BETWEEN MCS AND MCP;

3) APPROPRIATE CHANGES HAVE BEEN MADE TO THE MCP AND SYSTEM/LOGOUT.

MECHANISM:

THE MCS MUST BE COMPILED WITH THE $ OPTION "INSTALLATION" SET. THE MCS CALLS THE NEW INTRINSIC AS FOLLOWS:

LINK := SYSTEMLOG (CODE, RECORD[*]) WHERE CODE IS AN INTEGER VALUE (263) AND RECORD[*] IS AN ARRAY ROW CONTAINING MOST OF THE LOG INFORMATION.

SYSTEMLOG CHECKS THAT THE CALLER IS AN ACTIVE MCS AND THAT THE CODE IS VALID (263). IT THEN PASSES BOTH PARAMETERS TO LOGGER IN THE MCP, WHERE THE INFORMATION IS COMBINED WITH DATE, TIME, AND LINKS AND PLACED IN THE LOG FILE.

A LINK TO THE RECORD IN THE LOG FILE FOR THE ENTRY IS RETURNED AS THE VALUE OF SYSTEMLOG; IT IS ALSO RETURNED IN THE SECOND WORD OF THE ARRAY PARAMETER (E.G., RECORD[1]). THIS ALLOWS THE MCS TO MAINTAIN ANY LINKS IT MAY NEED. THE LINK VALUE IS ZERO IF THE RECORD WAS NOT ENTERED IN THE LOG (INCORRECT CODE, NOT AN MCS, ETC.).

FORMATS:

SEVERAL ENTRY SUBTYPES ARE CURRENTLY DEFINED WITHIN THE NEW ENTRY TYPE. IF ADDITIONAL SUBTYPES BECOME NECESSARY, NO MCP CHANGES ARE NEEDED; ONLY LOGOUT MUST BE MODIFIED TO RECOGNIZE NEW FORMATS.

EACH ENTRY MUST BE A MULTIPLE OF SIX WORDS IN LENGTH (THE ARRAY ROW PARAMETER MUST BE AT LEAST THIS LONG -- EXTRA WORDS ARE IGNORED). THE FOLLOWING VALUES MUST BE SUPPLIED BY THE MCS VIA THE ARRAY PARAMETER (ASSUME ENTRY IS 6N WORDS LONG):

WORD 0: N (2 LEQ N LEQ 5)
WORD 6N-6: LOGICAL STATION NUMBER (LSN)
WORD 6N-5: LINK TO PREVIOUS ENTRY FOR THIS LSN (0 IF NONE)
WORD 6N-4: SUBTYPE CODE

IN ADDITION, THE FOLLOWING MUST BE PROVIDED BY THE MCS FOR THE SPECIFIC SUBTYPES:

1. LOG ON (SUBTYPE = 1)

   WORD 2: USERCODE (STRING), FOLLOWED BY STATION NAME (STRING)

2. NORMAL LOG OFF (SUBTYPE = 2)

   WORD 2: PROCESSOR TIME SPENT IN MCS STACK FOR THIS USER

   WORD 3: I/O TIME SPENT IN MCS STACK FOR THIS USER

3. ABNORMAL LOG OFF (SUBTYPE = 3)

   WORDS 2, 3: SAME AS NORMAL LOG OFF

   WORD 4: REASON (STRING)

4. CHARGE CODE (SUBTYPE = 4)

   WORD 2: CHARGE CODE (STRING)

5. DATACOM ERROR (SUBTYPE = 5)

   WORDS 2, 3, 4: SAME AS FIRST THREE WORDS OF DATACOM ERROR RESULT (MESSAGE 99)

6. RJE CONTROL CARD (SUBTYPE = 6)

   WORD 2: CONTROL CARD TEXT (STRING)

7. PROCESS INTITIATION (SUBTYPE = 7)

   WORD 2: MIX NUMBER OF INITIATED PROCESS

   WORD 3: ARBITRARY TEXT (STRING)

8. GENERAL MESSAGE (SUBTYPE = 8)

   WORD 2: ANY TEXT (STRING)

NOTE: IN ALL CASES, A "STRING" IS TERMINATED BY A NULL CHARACTER (4"00").

D0038 CANDE - 09-08-72

THE MARK II.3 SYSTEM RELEASE CONTAINS A COMPLETELY NEW VERSION OF THE COMMAND AND EDIT PROGRAM, SYSTEM/CANDE. THE GENERAL FUNCTIONS AND CAPABILITIES OF THE NEW VERSION ARE CONSISTENT WITH BOTH THE PREVIOUS B6700 VERSION, AND ALSO THE B5700 CANDE PROGRAM.

A NEW INFORMATION MANUAL, #5000318, DATED SEPTEMBER 1972, PRESENTS COMPLETE DOCUMENTATION OF CANDE OPERATIONAL FEATURES AND FUNCTIONS, AND THE SPECIFIC LANGUAGE SYNTAX.

D0039 COBOL DOCUMENTATION CHANGES - 06-16-72

THE FOLLOWING CHANGES HAVE BEEN MADE IN DOCUMENTATION FOR COBOL:

1) THE NUMBER OF ITEMS IN A DATA MANAGEMENT SET IS NOW PERMITTED TO BE UP TO 499.

2) MOVING A STRING TO A COMP OR COMP-1 DATA ITEM IS NOW PROHIBITED. THERE IS NO CLEAR INTERPRETATION OF THIS TYPE OF MOVE, AND THE DOCUMENTATION SHOULD NOT INCLUDE THIS CASE.

3) PAGE ADVANCING AND SKIP TO CHANNEL ON THE PRINTER HAVE BEEN IMPLEMENTED. THE SYNTAX OF THE CHANNEL STATEMENT INDICATED BELOW HAS BEEN ADDED:

CHANNEL<INTEGER>IS<MNEMONIC-NAME>

THE INTEGER PERMITTED FOR THE PRINTER CHANNEL MUST BE POSITIVE AND LESS THAN 12.

TO MATCH THIS USAGE, THE SYNTAX FOR THE ADVANCING OPTION OF WRITE HAS BEEN AUGMENTED TO PERMIT ONE OF THE FOLLOWING OPTIONS:

BEFORE/AFTER ADVANCING TO CHANNEL <INTEGER>

BEFORE/AFTER ADVANCING TO <MNEMONIC-NAME>

BEFORE/AFTER PAGE

THE INTEGER MUST BE POSITIVE AND NO MORE THAN 11, AS ABOVE. THE MNEMONIC-NAME MUST HAVE BEEN ASSIGNED IN THE CHANNEL STATEMENT. PAGE IS INTERPRETED AS A SKIP TO CHANNEL ONE.

D0040 DIRECT PROCEDURE CALL IN ESPOL - 06-14-72

THIS ALLOWS A PROCEDURE CALL TO TAKE THE FORM:

<PROCEDURE NAME>[<EXP>] <POSSIBLE PARAMETER>

WHERE THE BRACKETED EXPRESSION MUST EVALUATE TO AN IRW USED TO LOCATE THE PROCEDURE TO BE ENTERED, INSTEAD OF THE NORMALLY STATIC PROCEDURE LOCATION (I.E., IRW-TO-PCW) ESTABLISHED AT COMPILE TIME. NORMAL PARAMETER CHECKING TAKES PLACE AS FOR THE (NAMED) PROCEDURE.

D0041 ESPOL - ESPOL FORK STATEMENT - 07-03-72

A NEW STATEMENT, "FORK", HAS BEEN ADDED TO THE ESPOL LANGUAGE TO FACILITATE THE INITIATION OF INDEPENDENT RUNNERS IN THE MCP.

SYNTAX:

<FORK STATEMENT>::=FORK<PROCEDURE STATEMENT>
[<FORK PARAMETERS>]

<FORK PARAMETERS>::=<STACK SIZE>,<PRIORITY>
<VISIBLE NAME INDEX>

<STACK SIZE>::=<ARITHMETIC EXPRESSION>

<PRIORITY>::=<ARITHMETIC EXPRESSION>

<VISIBLE NAME INDEX>::=<EMPTY>/,<POINTER EXPRESSION>

SEMANTICS:

THE PROCEDURE REFERENCED MAY BE TYPED OR UNTYPED. IT MAY NOT BE FORMAL OR DYNAMIC. IF TYPED, THE VALUE IS LOST.

IF THERE ARE ACTUAL PARAMETERS, THEY MUST AGREE IN NUMBER AND TYPE WITH THE FORMAL PARAMETERS SPECIFIED FOR THE PROCEDURE.

THERE MAY BE TWO OR THREE <FORK PARAMETERS>. THE FIRST TWO SPECIFY THE STACK SIZE AND PRIORITY, RESPECTIVELY, OF THE INITIATED PROCESS.

THE THIRD <FORK PARAMETER>, IF PRESENT, IS A POINTER INTO AN ARRAY OF INDEPENDENT RUNNER NAMES (INDEPRUNNERNAMES). ITS PRESENCE INDICATES THE SPECIFIED NAME WILL BE DISPLAYED ON THE SPO WITH BOJ AND EOJ MESSAGES. IF ABSENT, SPO MESSAGES ARE NOT TO BE DISPLAYED.

THE FIRST CHARACTER OF WHAT THE POINTER REFERENCES MUST BE THE NUMBER OF CHARACTERS IN THE ID.

A NEW MCP PROCEDURE, LOCATED AT (0, 175), IS CALLED TO INITIATE THE PROCESS.

D0042 ESPOL LABEL ADDRESS-EQUATION - 06-19-72

THIS ALLOWS THE PROGRAMMER TO SPECIFY THAT A PCW IS REQUIRED FOR THE LABEL, BY USING A LOCAL ADDRESS-EQUATION IN THE DECLARATION. WHEN THIS IS DONE, EXECUTION OF A BRANCH TO THE LABEL WILL BE DONE DYNAMICALLY, REQUIRING THE CONSTRUCT OF A PCW IN THE ADDRESS INDICATED IN THE ADDRESS-EQUATION.

D0043 XALGOL - REFERENCING USER OPTIONS - 06-27-72

IF A USER OPTION APPEARS FOR THE FIRST TIME WHILE PROCESSING A PROCEDURE PARAMETER LIST, FUTURE CALLS ON THE PROCEDURE WILL GENERATE SPURIOUS SYNTAX ERRORS. TO AVOID THIS PROBLEM, A SYNTAX ERROR WILL NOW BE GIVEN, "USER OPTION SHOULD BE REFERENCED PREVIOUSLY", IF THE FIRST REFERENCE TO A USER OPTION APPEARS WHILE COMPILING A PARAMETER LIST. THE ERROR CAN BE AVOIDED BY REFERENCING THE OPTION PRIOR TO THE PROCEDURE DECLARATION.

EXAMPLE:

```
BEGIN
   PROCEDURE P(X,
$ SET OMIT = MYOPTION
   Y
$ POP OMIT
   );
   WILL CAUSE A SYNTAX ERROR ON MYOPTION.

BEGIN
$ RESET MYOPTION
PROCEDURE P(X,
$ SET OMIT = MYOPTION
```

```
   Y
$ POP OMIT
   );
```

WILL NOT CAUSE A SYNTAX ERROR BECAUSE MYOPTION WAS PREVIOUSLY REFERENCED.

D0044 MCP - DISK PACK FILE SECURITY - 06-07-72

THE "?SECURITY" CONTROL CARD HAS BEEN EXTENDED TO INCLUDE NATIVE MODE DISK PACK FILES. THE NEW SYNTAX IS:

<SECURITY PART>:=<FILE ID><PACK DESIGNATE>
<SECURITYTYPE PART>

<PACK DESIGNATE>:=<EMPTY>/ON PACK/ON <PACKNAME>/

SEMANTICS:

<EMPTY> IMPLIES HEAD-PER-TRACK DISK

ON PACK IMPLIES SYSTEM RESOURCE DISK PACK

ON <PACKNAME> IMPLIES THE FILE IS ON A DISK PACK WITH THE NAME "<PACKNAME>".

FILE SECURITY FOR NATIVE MODE DISKPACKS HAS ALSO BEEN IMPLEMENTED. EXTERNAL CHARACTERISTICS OF THIS IMPLEMENTATION ARE EXACTLY THE SAME AS HEAD-PER-TRACK DISK WITH THE FOLLOWING EXCEPTIONS:

1. GUARDFILES WHEN USED WITH DISK PACK FILES MUST RESIDE ON HPT DISK;
2. GUARDFILES NAMES ON DISK PACK FILES MUST NOT EXCEED 117 CHARACTERS (INCLUDING SLASHES) IN LENGTH.

D0045 MCP - FILE ATTRIBUTES - 06-07-72

THE FILETYPE ATTRIBUTE DESCRIBES THE FORMAT OF THE RECORDS IN THE FILE. WHEN FILETYPE IS SET TO SEVEN (7), THE FORMAT OF THE RECORDS IN THE LOGICAL FILE IS DETERMINED BY THE FORMAT OF THE RECORDS IN

THE PHYSICAL OR PERMANENT FILE. THAT IS TO SAY, FILETYPE, MINRECSIZE, MAXRECSIZE, BLOCKSIZE, SIZEMODE, SIZEOFFSET, SIZE2, UNITS, AND INTMODE WILL BE CHANGED TO AGREE WITH THE PHYSICAL FILE WHEN THE FILE IS OPENED. WHEN THERE IS NO PERMANENT FILE, AS IN THE CASE OF OUTPUT TO THE PRINTER, FILETYPE IS SET TO ZERO (0) (FIXED LENGTH RECORDS) AND THE OTHER ATTRIBUTES ARE SET TO DEFAULT VALUES DEPENDING ON THE PHYSICAL DEVICE ASSOCIATED WITH THE FILE.

WITH THE ADVENT OF SOFTWARE TRANSLATION, IT SEEMED DESIRABLE TO ALLOW THE PROGRAMMER TO SPECIFY THE LOGICAL MODE (INTMODE) OF HIS PROGRAM, BUT STILL ALLOW THE PERMANENT FILE TO DEFINE ITS OWN STRUCTURE. A NEW FILETYPE WAS DEFINED TO ALLOW JUST THAT. SETTING FILETYPE TO EIGHT (8) SIGNIFIES TO THE I/O SUBSYSTEM THAT ALL THE ATTRIBUTES OTHER THAN INTMODE ARE TO BE DETERMINED BY THE PERMANENT OR PHYSICAL FILE.

DEFAULT VALUES FOR ATTRIBUTES WHEN PERMANENT FILE DOES NOT EXIST AND FILETYPE = 7 OR 8.

DISK

EXTMODE = INTMODE
MAXRECSIZE = BLOCKSIZE = 30 WORDS

CONSOLE

INTMODE = EXTMODE = EBCDIC
MAXRECSIZE = BLOCKSIZE = 10 WORDS

REMOTE

INTMODE = EXTMODE = EBCDIC
MAXRECSIZE = BLOCKSIZE = 12 WORDS

PAPER READER

INTMODE = EXTMODE = BCL
MAXRECSIZE = BLOCKSIZE = 10 WORDS

PAPER PUNCH

INTMODE = EXTMODE = BCL
MAXRECSIZE = BLOCKSIZE = 10 WORDS

LINE PRINTER

INTMODE = EXTMODE
MAXRECSIZE = BLOCKSIZE = IF EXTMODE = BCL THEN 17 ELSE 22

CARD READER

INTMODE = EXTMODE (UNLESS BINARY THEN EBCDIC)
MAXRECSIZE = BLOCKSIZE = IF EXTMODE IS BINARY THEN 20
ELSE IF EXTMODE IS EBCDIC THEN 14
ELSE 10; - WORDS

PSEUDO READER

INTMODE = EXTMODE
MAXRECSIZE = BLOCKSIZE - SAME AS CARD READER

CARD PUNCH

INTMODE = SAME AS CARD READER
MAXRECSIZE = BLOCKSIZE - SAME AS CARD READER

MAG TAPE

EXTMODE = INTMODE
MAXRECSIZE = BLOCKSIZE = 10 WORDS

A BUG HAS BEEN FIXED WHEREBY INTMODE WAS NOT SET TO EXTMODE OF THE PERMANENT FILE IN ALL CASES WHEN FILETYPE WAS SET TO SEVEN, MAKING IT IMPOSSIBLE TO SUCCESSFULLY LABEL EQUATE FILES FROM DISK TO READER TO TAPE.

NOTE: FILETYPE = 7 DOES NOT CHANGE BLOCKSIZE AND MAXRECSIZE WHEN THERE IS NOT A PERMANENT FILE, IF THEIR VALUES ARE NON-ZERO.

FILETYPE = 8 DOES ENFORCE THE USE OF DEFAULT VALUES IN ALL CASES WHEN THERE IS NO PERMANENT FILE.

D0046 LOADER - LOADING DCP PROGRAMS - 05-17-72

IN ORDER TO LOAD AND EXECUTE OFF-LINE DCP PROGRAMS, THE FOLLOWING FUNCTIONS ARE MADE AVAILABLE WITH THE LOADER:

LOADCP

DCPCARD

SIZE

SCANOUT

LOADCP

THIS FUNCTION WILL LOAD DCP CODE FROM TAPE (B5500 GENERATED, NOT LIBRARY FILE) OR FROM DISK (LIBRARY FILE - B5500 OR B6700) TO A SPECIFIED MEMORY ADDRESS. THE SYNTAX IS:

TAPE LOAD (NOT LIBRARY FILE)
LOADCP <FILEID>FROM<TAPE><MEMORY ADDRESS-DECIMAL>

LIBRARY FILE (RESIDING ON DISK)
LOADCP <FILEID>DISK<MEMORY ADDRESS-DECIMAL>

DCPCARD <MEMORY ADDRESS-DECIMAL>

THIS FUNCTION WILL LOAD CARDS WHICH HAVE BEEN PUNCHED IN THE FORMAT FOR USE WITH THE PACK UNIT AND THE A594 CARD READER. (SEE THE CARD READER SPECIFICATIONS A-1917 3590.)

THE CARD DECK FOR THIS FUNCTION MUST BE IN THE FOLLOWING ORDER:

DCPCARD <MEMORY ADDRESS-DECIMAL>
.
.
.
DATACARDS

.
.
.

ENDLOAD

SIZE

THE DCP LOADER FUNCTIONS ASSUME A MAXIMUM CODE SIZE UP TO 4,000 WORDS. IF A LARGER CODE FILE IS TO BE LOADED, THIS FUNCTION CARD MUST BE USED BEFORE THE LOAD CARD.

SYNTAX:

SIZE <NUMBER OF WORDS, DECIMAL>

THE NUMBER SHOULD BE EQUAL TO OR GREATER THAN THE NUMBER OF WORDS TO BE LOADED.

SCANOUT

THIS FUNCTION WILL AFFECT A SCANOUT-INITIALIZE OF THE DESIGNATED DCP, TO THE MEMORY ADDRESS SPECIFIED.

SCANOUT <K> <MEMORY ADDRESS, DECIMAL>

K - DCP NUMBER (0-3)

D0047 TYPE TRANSFER FUNCTIONS - 07-19-72

TYPE TRANSFER FUNCTIONS HAVE BEEN EXTENDED IN BOTH ALGOL AND ESPOL.

THE SYNTAX AND SEMANTICS OF THE INTEGER (PE, AE) AND DOUBLE (PE, AE) TYPE TRANSFER FUNCTIONS ON PAGES 6-17 AND 6-19 OF THE EXTENDED ALGOL LANGUAGE INFORMATION MANUAL (JUNE 1972) SHOULD BE REPLACED BY THE FOLLOWING:

INTEGER (<UPDATE POINTER><POINTER EXPRESSION>,
<ARITHMETIC EXPRESSION>)

DOUBLE (<UPDATE POINTER><POINTER EXPRESSION>,
<ARITHMETIC EXPRESSION>)

THESE FUNCTIONS RETURN, AS A SINGLE PRECISION INTEGER OR EXTENDED PRECISION VALUE, RESPECTIVELY, THE DECIMAL VALUE REPRESENTED BY THE STRING OF CHARACTERS STARTING WITH THE CHARACTER INDICATED BY THE POINTER EXPRESSION. THE LENGTH OF THE STRING AS DETERMINED FROM THE ARITHMETIC EXPRESSION MUST BE LESS THAN 24. A ZONE BIT CONFIGURATION OF "1101" FOR 8-BIT CHARACTERS OR "10" FOR 6-BIT CHARACTERS IN THE LEAST SIGNIFICANT CHARACTER POSITION WILL CAUSE THE RESULT TO BE NEGATIVE.

THE STATE OF THE POINTER EXPRESSION AT THE EXHAUSTION OF THE COUNT MAY BE PRESERVED BY AN UPDATE POINTER. PERMITTING THIS UPDATE POINTER IS THE EXTENSION IMPLEMENTED.

THE EXTENSIONS TO ESPOL ARE SIMILAR TO THOSE IN ALGOL. ONE ADDITION IS THAT THE UPDATE POINTER MAY BE EITHER A WORD VARIABLE OR PROCEDURE NAME OR A POINTER VARIABLE OR PROCEDURE NAME.

D0048 DUPSUPERVISOR & DUPINTRINSICS - 07-03-72

AS OF MARK II.2, TWO NEW RUN TIME OPTIONS ARE AVAILABLE IN THE MCP. BOTH ARE BASICALLY INTENDED FOR FAILSOFT SYSTEMS AND THOSE WHO RUN WITH RECONSTRUCTION. THE TWO OPTIONS ARE "DUPSUPERVISOR" AND "DUPINTRINSICS".

WHEN DUPSUPERVISOR IS SET AND THE SUPERVISOR NAME IS (FOR EXAMPLE) A/B, THEN THE MCP WILL FIRE UP A/B/EU#032 AFTER HALT/LOADING FROM EU32. IF WE HAD HALT/LOADED FROM EU33 TO GET RECONSTRUCTION ACTION, A/B/EU#033 WOULD HAVE BEEN THE NAME USED. IF THE OPTION DUPSUPERVISOR IS RESET, A/B WOULD HAVE BEEN USED.

DUPINTRINSICS IS ENTIRELY ANALAGOUS. WHEN THE OPTION IS SET THE NAME OF THE HALT/LOAD EU IS APPENDED TO THE INTRINSIC-FILE NAME, DEPENDING ON WHICH DISK WAS USED.

THE NATURAL STRATEGY HERE IS TO MAKE SURE THAT A/B/EU#032 IS ACTUALLY ON PHYSICAL UNIT 32 AND A/B/EU#33 IS ON ITS CORRESPONDING UNIT, SO THAT IN THE CASE OF COMPLETE EU DISASTER, THE SUPERVISOR PROGRAM AND THE SYSTEM INTRINSICS WILL BE AVAILABLE ON THE BACKUP EU.

TO CREATE THE KIND OF FILE DESCRIBED, THE FOLLOWING CONTROL CARD WILL SUFFICE:

```
<I>COPY SYSTEM/INTRINSICS AS S/I/"EU#032"
   TO DISK (AREACLASS =32)
```

IF AN INSTALLATION IS RUNNING UNDER RECONSTRUCTION AND DOES NOT TAKE THESE PRECAUTIONS WITH SYSTEM/INTRINSICS, A SUCCESSFUL HALT/LOAD FROM THE BACKUP EU CANNOT BE GUARANTEED. IN PARTICULAR, IF THE PRECEDING EU IS MADE NOT READY AND "SYSTEM INTRINSICS" RESIDES ON EU32, AND WE ATTEMPT TO RECONSTRUCT FROM EU#33, WE WILL HAVE PROBLEMS TRYING TO GET SYSTEM/INTRINSICS FROM THE NOT-READY DISK.

D0049 ESPOL - ESPOL "VERSION" DOLLAR OPTION - 06-26-72

THIS ESPOL CHANGE CAUSES THE VALUE OF VERSION TO BE PRINTED IN THE TRAILER PRINTOUT IF "VERSION" HAS BEEN SET.

D0051 BASIC - "FOR" STATEMENT IN BASIC - 06-23-72

THE HANDLING OF THE "FOR" STATEMENT IN BASIC HAS BEEN CORRECTED SO THAT THE LOOP CONTROL VARIABLE IS NOW COMPARED WITH THE LOOP LIMIT BEFORE THE LOOP IS EXECUTED. THUS, IF THE "FOR" STATEMENT SPECIFIES AN IMPOSSIBLE COMBINATION (E.G., "FOR I=1 TO -2" OR "FOR I = -5 TO -7 STEP 1" OR "FOR 1 = 5 TO 1 STEP 3"), THEN THE LOOP WILL NOT BE EXECUTED AT ALL AND CONTROL WILL PROCEED TO THE STATEMENT FOLLOWING THE "NEXT" STATEMENT ASSOCIATED WITH THE FOR STATEMENT.

WARNING: EXISTING BASIC PROGRAMS WHICH ARE DEPENDENT ON ALL FOR LOOPS BEING EXECUTED AT LEAST ONCE MAY HAVE TO BE ALTERED.

D0052 ALGOL LOGICAL NOT - 05-12-72

THE LOGICAL NOT EBCDIC SPECIAL CHARACTER MAY NOW BE USED IN PLACE OF THE MNEMONIC NOT.

D0053 ZIP WITH <ARRAY ROW> - 06-20-72

THE FIRST PARAGRAPH CONCERNING THE SEMANTICS OF "ZIP WITH <ARRAY ROW>" ON PAGE 9-39 OF THE ALGOL LANGUAGE DOCUMENT SHOULD BE DELETED AND REPLACED WITH:

> THE INFORMATION IN THE <ARRAY-ROW> MUST APPEAR AS IT NORMALLY WOULD ON PUNCHED CARDS, I.E., AS BCL OR EBCDIC CHARACTERS. THE <ARRAY ROW> MAY BE A BCL OR EBCDIC STRING ARRAY ROW OR A NON-STRING ARRAY ROW. IF THE <ARRAY ROW> IS NOT A STRING ARRAY, THE CHARACTER SET EXPECTED BY THE ZIP INTRINSIC IS DETERMINED BY THE SETTING OF THE BCL $ OPTION. THE FIRST CHARACTER OF THE <ARRAY ROW> MUST BE A QUESTION MARK (EBCDIC 4" 6F" OR BCL 3"14"). THE LAST "CARD" IN THE <ARRAY ROW> MUST CONTAIN THE WORD "END" FOLLOWED BY A PERIOD. THE "ARRAY ROW" IS PROCESSED AS ONE PUNCHED CARD, BUT MAY INCLUDE MORE THAN 72 CHARACTERS. A SEMICOLON IS USED TO SEPARATE CONTROL "CARDS" WITHIN THE <ARRAY ROW>. ONLY ONE QUESTION MARK CHARACTER MAY APPEAR IN THE <ARRAY ROW>.

D0054 MCP - AUTOPRINT - 05-24-72

THIS IMPLEMENTATION TO THE MCP ALLOWS THE FOLLOWING SYNTAX FOR AP LINE PRINTER AND AP CARD PUNCH, AND LD CARD READER. THESE FUNCTIONS ASSIGN THE PARTICULAR UNIT TO AUTOMATIC BACKUP.

LD-ED CARD READERS WILL BE ASSIGNED TO AUTO LOAD CONTROL EVEN WHEN OPTION 7 (CDONLY) IS RESET. AP-ED PRINTERS AND PUNCHES WILL BE ASSIGNED (BUT NOT RESERVED) FOR AUTOBACKUP TO PRINT OR PUNCH.

TO ENABLE AUTOBACKUP, THE KEYIN MESSAGES

```
AP LP <NNN>
AP CP <NNN>

  OR

LD CR <NNN>
```

ARE USED.

TO DISABLE AUTOBACKUP, THE KEYIN MESSAGES

AP - LP <NNN>
AP - CP <NNN>

OR

LD - CR <NNN>

ARE USED.

PREVIOUS FUNCTIONS FOR AP AND LD HAVE NOT BEEN CHANGED. HOWEVER, THE NUMBER OF COPIES OF AUTOBACKUP PRINTING IS CONTROLLED BY THE MAXIMUM OF AP <N> (AP MAX) AND THE NUMBER OF AP-ED PRINTERS. THUS,

AP 0
AP LP 11

WILL ALLOW ONE AUTOBACKP PRINTING ON LP 11.

D0055 XALGOL - XALGOL FILE DECLARATION - 07-10-72

IN XALGOL, THE SYNTAX FOR THE IN-OUT PART OF A FILE DECLARATION IS EXPANDED TO INCLUDE THE ATTRIBUTE "IO".

SYNTAX:

<IN-OUT PART>::=IN/OUT/IO/EMPTY

SEMANTICS:

IF IO IS SPECIFIED, THE ATTRIBUTE MYUSE WILL BE SET TO THREE. THIS IS INTENDED TO FACILITATE XALGOL OBJECT IO VIA CANDE.

D0056 ESPOL - ESPOL TYPE TRANSFER FUNCTIONS - 07-05-72

THE INTEGER AND DOUBLE TYPE TRANSFER FUNCTIONS DEALING WITH A <POINTER EXPRESSION> AND <ARITHMETIC EXPRESSION> HAVE BEEN EXTENDED TO ALLOW AN OPTIONAL UPDATE POINTER TO BE USED (VIA THE ICVU OPERATOR); I.E.,

INTEGER (<UPDATE POINTER><PE>,<AE>)

DOUBLE (<UPDATE POINTER><PE>,<AE>.

THE NEW SYNTAX IS:

```
<UPDATE POINTER>::=<EMPTY>/
   <POINTER IDENTIFIER:/
   <WORD IDENTIFIER>:/
   <POINTER PROCEDURE IDENTIFIER>:/
   <WORD PROCEDURE IDENTIFIER>:
```

D0057 POINTER SKIP - 07-12-72

IN THE STATEMENT

P:=P + (EXPRESSION)

WHERE P IS A POINTER, IF THE EXPRESSION IS NOT POSITIVE, NO SKIP WILL BE PERFORMED. SIMILARLY, IN THE STATEMENT

P:=P - (EXPRESSION)

NO SKIP WILL BE PERFORMED - FORWARDS OR BACKWARDS - IF THE VALUE OF THE EXPRESSION IS NEGATIVE. THIS IS DUE TO HARDWARE DESIGN.

D0058 MCP - FILE ATTRIBUTES - 06-07-72

RESIDENT ATTRIBUTE

RESIDENT IS A BOOLEAN FILE ATTRIBUTE WHICH, WHEN INTERROGATED, RETURNS TRUE IF A PHYSICAL OR PERMANENT FILE EXISTS WITH THE SAME TITLE AS THE LOGICAL FILE AND IS AVAILABLE FOR ASSIGNMENT TO BE LOGICAL FILE. INTERROGATING THIS ATTRIBUTE DOES NOT ASSIGN A PHYSICAL FILE TO THE LOGICAL FILE AND DOES NOT CAUSE THE FILE TO BE OPENED. RESIDENT ALWAYS RETURNS TRUE WHEN ACCESSED FOR A LOGICAL OUTPUT FILE, SINCE NO PERMANENT FILE IS REQUIRED TO EXIST TO OPEN THE FILE.

TESTING TO SEE WHETHER OR NOT A FILE EXISTED USING THE ATTRIBUTE RESIDENT, IN SOME CASES, HAD THE SIDE AFFECT OF CHANGING THE VALUES OF THE ATTRIBUTES BLOCKSIZE, MAXRECSIZE, AND INTMODE WHEN THE FILE

WAS NOT DECLARED FILETYPE = 7. WITH THIS PATCH, TESTING RESIDENT CHANGES THE VALUES OF BLOCKSIZE, MAXRECSIZE, AND INTMODE ONLY WHEN THE FILE IS DECLARED FILETYPE = 7. THE ACTION IS THE SAME WHEN FILETYPE = 8, EXCEPT THAT INTMODE WILL NOT BE CHANGED.

D0059 USERS GUIDE TO MEMORY CONTROL - 08-30-72

INTRODUCTION:

THE PURPOSE OF THIS DOCUMENT IS TO PROVIDE INFORMATION REGARDING THE IMPLEMENTATION RATIONALE AND USE OF THE VIRTUAL MEMORY MANAGEMENT TECHNIQUE UTILIZED IN THE B6700.

PRELIMINARY:

BEFORE PROCEEDING IT IS NECESSARY TO DEFINE A FEW TERMS AND CERTAIN ASPECTS OF THE B6700 MEMORY MANAGEMENT TECHNIQUES.

1. VIRTUAL MEMORY:

THE B6700 MEMORY MANAGEMENT SCHEME IS BASED ON THE VIRTUAL MEMORY CONCEPT. ESSENTIALLY, THE LANGUAGE COMPILERS, WHEN OPERATING ON A PROGRAM, DIVIDE THE PROGRAM-S CODE INTO A NUMBER OF SEGMENTS. ASSOCIATED WITH THESE SEGMENTS IS A SEGMENT DESCRIPTOR. THESE SEGMENTS AND A DICTIONARY OF SEGMENT DESCRIPTORS ARE KEPT ON SOME MEMORY EXTENSION DEVICE SUCH AS HEAD-PER-TRACK DISK. ADDITIONALLY, DATA ARRAYS ARE ALSO SEGMENTED AND A DATA DESCRIPTOR IS ASSOCIATED WITH EACH. A TWO PART ADDRESS IS USED TO REFERENCE CODE AND DATA. THE FIRST PART IS USED TO REFERENCE A PARTICULAR DESCRIPTOR, THE SECOND PART IS USED AS AN OFFSET INTO THE SEGMENT ADDRESSED BY THE DESCRIPTOR. THE DESCRIPTOR CONTAINS FIELDS WHICH INDICATE THE SEGMENT LENGTH, THE BASE ADDRESS OF SEGMENT AND A FLAG (OR PRESENCE BIT) WHICH DENOTES WHETHER THE BASE ADDRESS IS A MAIN MEMORY ADDRESS OR A DISK ADDRESS.

THE OPERATING SYSTEM MAINTAINS A LINKED LIST OF AVAILABLE SPACE SEGMENTS IN MAIN MEMORY. WHEN A PROGRAM, DURING EXECUTION,

ATTEMPTS TO ADDRESS DATA OR CODE WHICH CURRENTLY RESIDES ONLY ON DISK (PRESENCE BIT OFF), A HARDWARE INTERRUPT IS GENERATED. THE HARDWARE, ON SENSING THIS INTERRUPT, WILL AUTOMATICALLY CALL A "GETSPACE" PROCEDURE PASSING THE DESCRIPTOR AS A PARAMETER. THE GETSPACE ROUTINE EXTRACTS, FROM THE DESCRIPTOR, THE SEGMENT SIZE AND DISK ADDRESS OF THE DESCRIPTOR. IT WILL THEN SEARCH THE AVAILABLE SPACE TABLE FOR AN AVAILABLE SPACE SEGMENT OF ADEQUATE SIZE, FIX THE DESCRIPTOR TO POINT AT THIS SPACE, DELINK THE SPACE FROM THE AVAILABLE SPACE LIST AND LOAD THE SEGMENT FROM DISK TO THIS SPACE. FOR THE CASE OF A REFERENCE TO A DATA SPACE NOT PREVIOUSLY REFERENCED, THE DISK ADDRESS OF THE DESCRIPTOR WOULD HAVE BEEN ZERO. THE GETSPACE ROUTINE DETECTS THIS AND ONLY ALLOCATES MAIN MEMORY AND BYPASSES THE LOAD OF DISK TO MAIN MEMORY.

2. OVERLAY:

IT CAN HAPPEN THAT THE GETSPACE ROUTINE, IN ATTEMPTING TO LOAD A SEGMENT, CANNOT FIND AN AREA IN THE AVAILABLE SPACE LIST LARGE ENOUGH TO HOLD THE SEGMENT. WHEN THIS OCCURS, SOME OF THE IN-USE SEGMENTS MUST BE "OVERLAYED". EACH IN-USE SEGMENT IS PRECEDED BY A MEMORY LINK WHICH CONTAINS THE ADDRESS OF THE DESCRIPTOR POINTING AT THIS SEGMENT AND THE HOME DISK ADDRESS OF THE SEGMENT. THE SYSTEM ALSO CONTAINS A LEFT-OFF POINTER, I. E., THE ADDRESS OF SOME SEGMENT IN MAIN MEMORY WHERE THE LAST INVOCATION OF THE OVERLAY PROCESS LEFT OFF.

THE OVERLAY PROCESS CONSISTS OF EXAMINING ONE OR MORE CONSECUTIVE SEGMENTS, STARTING WITH THE LEFT-OFF POINTER, FIXING THE DESCRIPTOR POINTING AT THIS SEGMENT BY TURNING ITS PRESENCE BIT OFF AND CHANGING ITS ADDRESS FIELD FROM A CORE TO DISK ADDRESS. FOR CODE SEGMENTS, THIS COMPLETES THE PROCESS OF OVERLAYING A SEGMENT. SINCE CODE IS NOT MODIFIABLE, THE CODE SEGMENT ON DISK IS ALWAYS IDENTICAL TO THE CODE SEGMENT IN MAIN MEMORY AND, THEREFORE, NEED NOT BE RETURNED TO DISK. FOR DATA SEGMENTS, HOWEVER, IT IS NECESSARY TO WRITE THE SEGMENT BACK TO DISK. THE SYSTEM MAINTAINS AN OVERLAY DISK FILE FOR EACH

PROGRAM. THE FIRST TIME A DATA SEGMENT MUST BE OVERLAYED (DETECTED BY THE DISK ADDRESS IN THE MEMORY LINK BEING ZERO), SPACE IS ALLOCATED FOR THIS SEGMENT IN THE OVERLAY FILE. THE DATA SEGMENT IS THEN WRITTEN TO DISK WITH THE DISK ADDRESS BEING RECORDED IN THE DATA DESCRIPTOR.

THIS OVERLAY MECHANISM PROCEEDS UNTIL ENOUGH CONSECUTIVE SPACE HAS BEEN OVERLAYED TO SATISFY THE ORIGINAL DEMAND. ON COMPLETION OF AN OVERLAY, THE LEFT-OFF POINTER IS SET TO POINT AT THE SEGMENT BEYOND THE LAST ONE OVERLAYED.

ONE OF THE UNDESIRABLE EFFECTS OF A VIRTUAL MEMORY SYSTEM BASED ON THE VARIABLE LENGTH SEGMENT APPROACH IS THE CHECKERBOARDING EFFECT. AFTER THE SYSTEM HAS BEEN RUNNING FOR A WHILE, MEMORY TENDS TO CONTAIN A LARGE NUMBER OF SMALL AVAILABLE AREAS, TOO SMALL TO BE USABLE. THIS OVERLAY MECHANISM UTILIZING THE LEFT-OFF POINTER TENDS TO REDUCE THIS PROBLEM AS THESE SMALL AREAS ARE ELIMINATED DURING THE OVERLAY PROCESS.

3. THRASHING:

A PHENOMENON KNOWN AS "THRASHING" CAN OCCUR IN ANY TYPE OF VIRTUAL MEMORY SYSTEM. THE SITUATION CAN OCCUR WHERE A HIGH PERCENTAGE OF DEMANDS FOR A NEW SEGMENT INVOKES THE OVERLAY MECHANISM. THE SYSTEM SPENDS SO MUCH TIME OVERLAYING THAT PERFORMANCE IS DEGRADED BEYOND TOLERABLE LIMITS. THE SITUATION OCCURS BECAUSE THE SYSTEM HAS ACCEPTED FOR EXECUTION MORE PROGRAMS THAN CAN CONVENIENTLY FIT IN MAIN MEMORY.

TWO TECHNIQUES CAN BE USED TO RESOLVE THE THRASHING PROBLEM:

ONE TECHNIQUE IS TO HAVE THE COMPILERS ANALYZE THE PROGRAM IT IS COMPILING AND GENERATE AN ESTIMATE OF THE PROGRAM-S MEMORY REQUIREMENT. THE SYSTEM WILL ACCEPT A PROGRAM FOR EXECUTION ONLY WHEN AVAILABLE MAIN MEMORY EXCEEDS THIS MEMORY ESTIMATE. DUE TO DYNAMIC NATURE OF SOME LANGUAGES, THE COMPILER-GENERATED ESTIMATE CAN BE EXTREMELY INACCURATE. MORE ELABORATE TECHNIQUES TO GENERATE ACCURATE ESTIMATES HAVE BEEN CONSIDERED BUT NONE HAVE BEEN FOUND THAT ARE ECONOMICALLY FEASIBLE.

A SECOND TECHNIQUE IS TO HAVE THE SYSTEM DETECT WHEN THRASHING OCCURS AND SUSPEND EXECUTION, ON A PRIORITY BASIS, OF ONE OF MORE PROGRAMS. THE OVERLAY MECHANISM WILL ROLL OUT THE SUSPENDED PROGRAM-S DATA AND CODE SEGMENTS, THUS REDUCING THRASHING. MOST ATTEMPTS TO DO THIS INVOLVE MONITORING THE DISK TO MAIN MEMORY OVERLAY RATE AND SUSPENDING PROGRAMS WHEN THIS RATE EXCEEDS A CERTAIN LIMIT. THE PROBLEM WITH THIS TECHNIQUE IS THAT AN OVERLAY RATE WHICH REPRESENTS THRASHING FOR ONE APPLICATION OR CONFIGURATION DOES NOT NECESSARILY REPRESENT THRASHING ON SOME OTHER SYSTEM.

OVERLAY CONTROL:

SO FAR, DISCUSSED IS CERTAIN TERMINOLOGY AND IN GENERAL TERMS, THE MEMORY MANAGEMENT SCHEME USED ON THE B5500 AND B6700.

STUDIES OF THIS VIRTUAL MEMORY MANAGEMENT SCHEME HAVE INCLUDED PLOTTING A PROGRAMS CORE UTILIZATION AS A FUNCTION OF ITS OVERLAY RATES. THE RESULTS FOR MOST PROGRAMS ARE INDICATED IN FIGURE ONE. SEVERAL OBSERVATIONS CAN BE MADE REGARDING THESE RESULTS.

1. THE AMOUNT OF CORE A PROGRAM IS USING IS DEPENDENT ON ITS OVERLAY RATE. MOST INSTALLATIONS HAVE AN ACCOUNTING MECHANISM TO CHARGE FOR RESOURCES UTILIZED BY A PROGRAM. ONE OF THOSE RESOURCES CHARGED FOR IS CORE UTILIZATION. THE OVERLAY MECHANISM PREVIOUSLY DESCRIBED OPERATES ON A DEMAND BASIS. THUS, THE AMOUNT OF OVERLAY AND THEREFORE, THE MEMORY BEING UTILIZED BY A PROGRAM VARIES WIDELY, DEPENDENT ON WHAT OTHER PROGRAMS ARE BEING MULTI-PROCESSED WITH IT.

2. THE OVERLAY RATE HAS A LARGE INFLUENCE ON THE BEHAVIOR OF A PROGRAM, THUS, ALLOWING A USER CONTROL OF THE OVERLAY RATE WOULD ALLOW A DEGREE OF CONTROL OF THE PROGRAM.

3. A CONSTANT BUT CONTINUOUS OVERLAY RATE TENDS TO KEEP CORE FREE OF UNUSED SEGMENTS. THE VALUE THE SYSTEM KEEPS FOR THE AMOUNT OF MAIN MEMORY AVAILABLE BECOMES MORE MEANINGFUL. AS A

RESULT, THE SYSTEM CAN PROVIDE BETTER AUTOMATIC SCHEDULING CAPABILITIES. THIS ALSO IN TURN TENDS TO DECREASE THE THRASHING PROBLEM.

4. THE CURVE OF FIGURE ONE SHOWS A POINT WHERE THERE IS AN ABRUPT CHANGE IN SLOPE. MR. DENNING [1][2] DEFINES THE TERM "WORKING SET" AS, "THAT MINIMUM COLLECTION OF A PROGRAMS SEGMENTS WHICH MUST RESIDE IN MAIN MEMORY FOR A PROCESS TO RUN EFFICIENTLY." THAT POINT ON THE CURVE IN FIGURE ONE WHERE THE SLOPE OF THE CURVE CHANGES ABRUPTLY REPRESENTS A PROGRAMS "WORKING SET". THIS POINT ALSO INDICATES THAT THERE IS AN OPTIMUM OVERLAY RATE FOR A PROGRAM. AN OVERLAY RATE HIGHER THAN THIS POINT REPRESENTS DEGRADED PROGRAM PERFORMANCE WHICH OVERLAY RATES BELOW THIS OPTIMUM REPRESENT INEFFICIENT MEMORY UTILIZATION.

AS A RESULT OF THE ABOVE OBSERVATIONS, AN OVERLAY CONTROL CAPABILITY WAS ADDED TO THE SYSTEM. THIS ESSENTIALLY CONSISTS OF ALLOWING THE USER TO SPECIFY AN OVERLAY GOAL AND THEN, ON THE BASIS OF THIS OVERLAY GOAL AND PROGRAM PRIORITY, GENERATE AND MAINTAIN A CONTINUOUS AND CONSTANT OVERLAY RATE AGAINST EVERY PROGRAM BEING MULTI-PROCESSED.

IMPLEMENTATION

1. THE OVERLAY GOAL:

TWO OPERATOR MESSAGES, THE SF AND OG, CAN BE UTILIZED (SEE APPENDIX A FOR DETAILS) TO SET THE OVERLAY GOAL. THE SF ALLOWS SETTING A SYSTEM DEFAULT OVERLAY GOAL. THE UNITS OF THIS OVERLAY GOAL ARE IN TERMS OF PERCENTAGE OF OVERLAYABLE CODE PER MINUTE. THIS DEFAULT RATE IS USED IN CONJUNCTION WITH A PROGRAM-S PRIORITY TO GENERATE A PROGRAM OVERLAY RATE AS FOLLOWS:

```
PROGRAMOVERLAYRATE = INTEGER (DEFAULTPRIORITY X
   ((100 - PROGRAMPRIORITY)/50)))
```

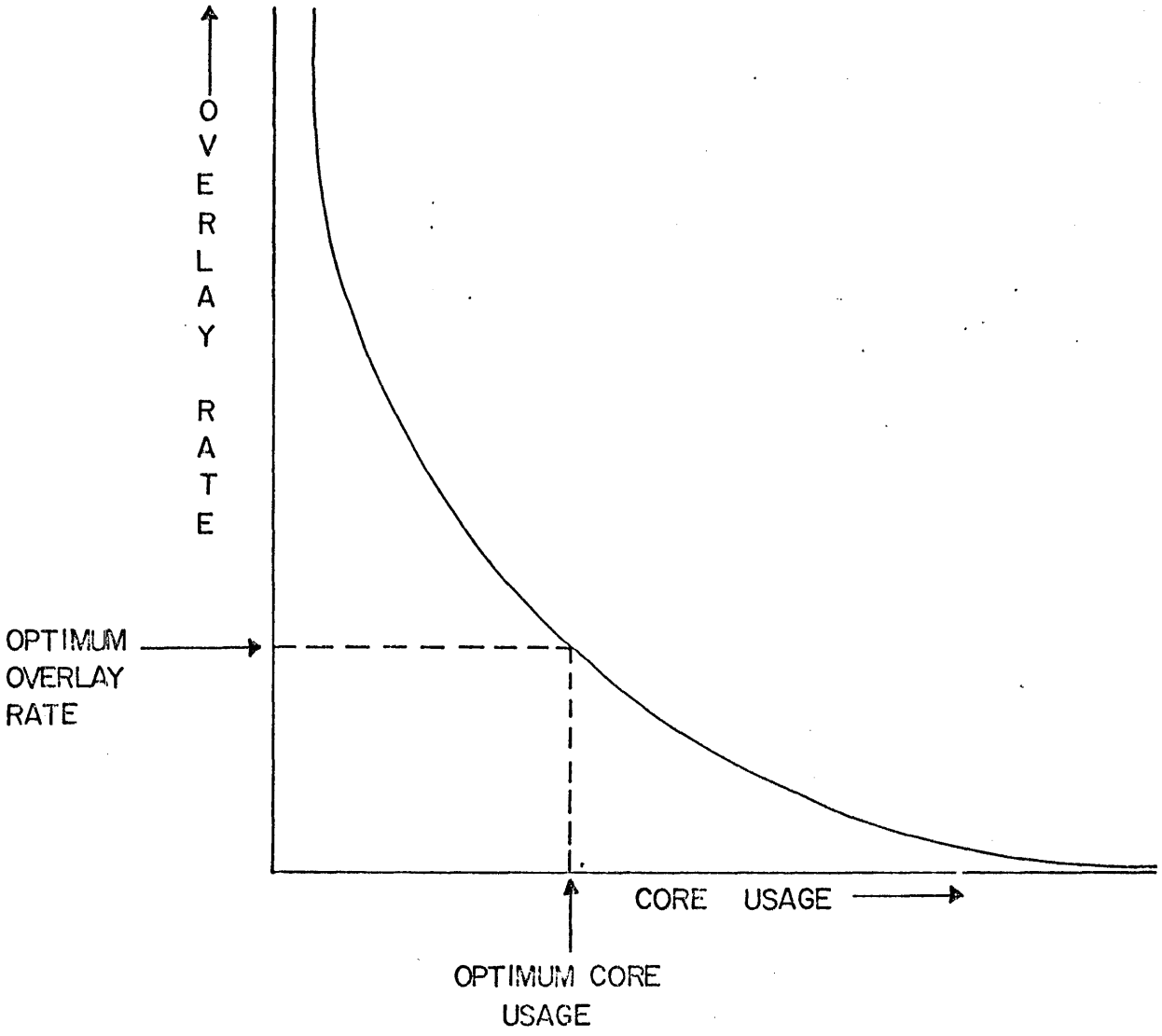

Figure 1-1. OVERLAY RATES

ESSENTIALLY, THE ABOVE SETS PRIORITY 50 PROGRAMS TO AN OVERLAY RATE EQUAL TO THE SYSTEM-S DEFAULT RATE. PRIORITY 99 PROGRAMS WILL HAVE AN OVERLAY RATE OF ZERO, WHILE PRIORITY 01 PROGRAMS WILL HAVE AN OVERLAY RATE OF TWICE THE SYSTEM-S DEFAULT RATE. THE DEFAULT OVERLAY GOAL IS ALSO THE OVERLAY RATE FOR THE CODE AND DATA SEGMENTS OF THE OPERATING SYSTEM.

THE OG MESSAGE (FOR DETAILS, SEE APPENDIX A) ALLOWS SETTING AN OVERLAY RATE FOR A PARTICULAR PROGRAM, THUS OVERRIDING THE SYSTEM-GENERATED DEFAULT RATE.

THE IMPLEMENTATION CONSISTS OF A OVERLAY CONTROL PROCEDURE WHICH CYCLICALLY "WAKES UP." THIS PROCEDURE FIRST GENERATES AN OVERLAY AMOUNT FOR EACH PROGRAM IN THE SYSTEM (INCLUDING THE OPERATING SYSTEM) BY MULTIPLYING EACH PROGRAM-S CURRENT CORE UTILIZATION BY THE PROGRAM S OVERLAY RATE. THIS PROCEDURE WILL THEN, STARTING WITH THE LEFT-OFF POINTER (DESCRIBED PREVIOUSLY), EXAMINE CONSECUTIVE SEGMENTS OF MEMORY, MOVING THE LEFT-OFF POINTER TO THE NEXT ADJACENT SEGMENT. EACH TIME AN OVERLAYABLE SEGMENT IS ENCOUNTERED, THE OVERLAY AMOUNT OF THE SEGMENT OWNER IS CHECKED. IF THIS OVERLAY AMOUNT IS ZERO, THE SEGMENT IS BYPASSED. IF THE OWNER S OVERLAY AMOUNT IS NOT ZERO, IT IS REDUCED BY THE SIZE OF THE SEGMENT; AND THE SEGMENT IS OVERLAYED. THIS OPERATION PROCEEDS UNTIL THE OVERLAY AMOUNT OF ALL PROGRAMS IN THE SYSTEM IS REDUCED TO ZERO.

THIS PROCESS OF IMPOSING A FIXED AMOUNT OF OVERLAY ON EVERY PROGRAM IN THE SYSTEM, AS WELL AS ENHANCING PROGRAM PREDICTABILITY, CAN ALSO ENHANCE THROUGHPUT IN THAT UNUSED SEGMENTS TEND TO GET OVERLAYED, THUS MAKING ROOM FOR MORE PROGRAMS. THIS IS DONE AT THE EXPENSE OF ELAPSED TIME, I.E., THE ENFORCED OVERLAY DOES SLOW DOWN AN INDIVIDUAL PROGRAM SOMEWHAT (THE AMOUNT BEING DEPENDENT ON THE OVERLAY RATE). THERE DOES EXIST, HOWEVER, CERTAIN HIGH PRIORITY PROGRAMS WHERE THE CRITERIA IS TO HAVE THESE PROGRAMS EXECUTE AS QUICKLY AS POSSIBLE. FOR THIS REASON, THE OPTION EXISTS TO TURN OFF THE OVERLAY CONTROL PROCEDURE AT ANY TIME BY SETTING THE SYSTEM OVERLAY GOAL TO ZERO. THIS WILL AUTOMATICALLY CAUSE THE

OVERLAY CONTROL PROCEDURE TO DEALLOCATE ANY MEMORY RESOURCES IT IS UTILIZING AND TERMINATE ITSELF (WHICH CAUSES THE SYSTEM TO DEALLOCATE ITS CODE AND STACK SPACE). SETTING THE OVERLAY GOAL TO NON-ZERO WILL AUTOMATICALLY REINSTATE THE OVERLAY CONTROL PROCEDURE. TURNING THE OVERLAY CONTROL PROCEDURE ON AND OFF CAN BE DONE DYNAMICALLY AT ANY TIME EVEN WHEN THERE ARE PROGRAMS RUNNING.

COMPUTING THE WORKING SET:

IT WAS PREVIOUSLY POINTED OUT THAT IT IS NOT ALWAYS POSSIBLE FOR COMPILERS TO GENERATE ACCURATE CORE ESTIMATES FOR PROGRAMS. IT WAS ALSO POINTED OUT THAT BY IMPOSING A CONSTANT AMOUNT OVERLAY ON A PROGRAM THAT AT ANY INSTANT OF TIME, THE MEMORY BEING UTILIZED BY A PROGRAM IS ITS WORKING SET. IT IS ALSO APPARENT THAT WHILE A PROGRAM-S WORKING SET WILL NOT CHANGE SIGNIFICANTLY DURING SMALL INCREMENTS OF TIME, A PROGRAM S WORKING SET MAY UNDERGO A LARGE CHANGE OVER A LONG PERIOD OF TIME. AN EFFECTIVE WORKING SET SIZE IS OF MUCH MORE INTEREST TO BOTH THE USER AND THE SYSTEM THAN THE INSTANTANEOUS WORKING SET SIZE. FOR THE USER IT PROVIDES A MORE ACCURATE COST ACCOUNTING. FOR THE SYSTEM IT PROVIDES ENHANCED AUTOMATIC SCHEDULING CAPABILITY. WE DEFINE THE EFFECTIVE WORKING SET AS THE INTEGRAL OF THE INSTANTANEOUS WORKING SET OVER THE PROGRAM-S RUNNING TIME. FOR EACH PROGRAM, WHENEVER ITS MEMORY UTILIZATION IS CHANGED, THE SYSTEM GENERATES THE FOLLOWING:

1. COREINUSE := COREINUSE + W

2. COREINTEGRAL := COREINTEGRAL - W X TIMEBASE

WHERE W IS THE SEGMENT SIZE (THE SIGN OF W IS PLUS IF THE SEGMENT IS BEING ADDED AND MINUS IF THE SEGMENT IS BEING OVERLAYED) AND TIMEBASE IS THE SUM OF A PROGRAM-S PROCESSOR AND I/O TIME. THE EFFECTIVE WORKING SET OF A PROGRAM AT ANY TIME (INCLUDING END-OF-JOB TIME) IS THEN:

WSSIZE := COREINUSE + (COREINTEGRAL/TIMEBASE)

THE VALUE OF THE CURRENT WSSIZE IS DISPLAYED ON THE CONSOLE (SEE

THE CU MESSAGE IN THE APPENDIX). AT END-OF-JOB TIME, WSSIZE IS WRITTEN OUT TO THE SYSTEM LOG (WHERE IT APPEARS AS COREUSAGE) AND ALSO IS USED TO UPDATE A PROGRAM-S CORE ESTIMATE IF THAT PROGRAM RESIDES IN THE SYSTEM DIRECTORY. ADDITIONALLY, WSSIZE IS USED IN THE CONTROL OF THE AUTOMATIC PROGRAM SUSPENSION-RESUMPTION MECHANISM DESCRIBED IN THE NEXT SECTION.

THE EFFECTIVE WORKING SET SIZE WSSIZE IS COMPUTED AT ALL TIMES WHETHER OR NOT THE WORKING SET MEMORY CONTROL IS RUNNING.

AUTOMATIC PROGRAM SUSPENSION/RESUMPTION

THE TERM "THRASHING" WAS PREVIOUSLY DEFINED. WE COULD NOW REDEFINE THRASHING AS "A DEGRADED STATE OF SYSTEM BEHAVIOR DUE TO THE SUM OF ALL PROGRAMS CURRENT WORKING SET EXCEEDING THE SYSTEM-S MAIN MEMORY CAPACITY." ONE OF THE REASONS FOR THRASHING, SPECIFIED EARLIER, WAS THE INABILITY OF COMPILERS TO PRODUCE AN ACCURATE CORE ESTIMATE FOR A PROGRAM. THIS, IN TURN, CAUSES THE SYSTEM TO ACCEPT PROGRAMS FOR EXECUTION WHICH WILL NOT "FIT" IN MEMORY. ANOTHER REASON FOR THRASHING IS THAT SOME PROGRAM-S WORKING SETS CAN CHANGE SIGNIFICANTLY OVER A PERIOD OF TIME. THE RESULTS ARE THAT AT ONE TIME ALL PROGRAMS MAY "FIT" IN MEMORY BUT SOMETIME LATER THEY WILL NOT. SINCE THRASHING DEGRADES SYSTEM PERFORMANCE, IT MUST BE AVOIDED; HOWEVER, OPINIONS VARY AS TO WHAT DEGREE OF SYSTEM-S DEGRADATION REPRESENTS EXCESSIVE THRASHING. DUE TO THIS VARIATION OF OPINION, AN OPTION EXISTS WHICH ALLOWS SETTING OF THE THRASHING DETECTION POINT.

CONSOLE A RESETTABLE SYSTEM PARAMETER, AVAILMIN, (SEE SF MESSAGE IN THE APPENDIX) ALLOWS THE USER TO CONTROL THE THRASHING POINT OF THE SYSTEM. THIS PARAMETER, EXPRESSED IN TERMS OF PERCENT OF MEMORY, CAN BE SET OR ALTERED AT ANY TIME. AFTER EACH OVERLAY PASS IN THE OVERLAY CONTROL PROCEDURE, THE AMOUNT OF AVAILABLE MAIN MEMORY IS COMPARED WITH THE AMOUNT OF MEMORY SPECIFIED BY AVAILMIN. WHEN OR IF AVAILABLE SPACE FALLS BELOW THE LEVEL SET BY AVAILMIN, THE SYSTEM WILL AUTOMATICALLY SUSPEND THE LOWEST PRIORITY PROGRAM AND ROLL OUT THIS PROGRAM-S OVERLAYABLE SEGMENTS INDICATING TO THE OPERATOR VIA A "SUSPENDED BY SYSTEM" MESSAGE WHICH PROGRAM WAS

SUSPENDED.

THE OPERATOR CAN OVERRIDE THE SYSTEM WHEN A PROGRAM IS SUSPENDED. THE <MIX-NUMBER> OK WILL CAUSE THE PROGRAM TO RESUME EXECUTION (BUT THE SYSTEM MAY IMMEDIATELY SUSPEND IT AGAIN). THE SUSPENDED PROGRAM MAY BE TERMINATED BY ENTERING <MIX-NUMBER> DS. IF IT IS MANDATORY THAT A SUSPENDED PROGRAM RESUME PROCESSING AND NOT BE SUSPENDED AGAIN, THE OPERATOR CAN INCREASE THE PROGRAM S PRIORITY. THE SYSTEM WILL AUTOMATICALLY RESUME ANY SUSPENDED PROGRAM ON A PRIORITY CHANGE.

THE UTILIZATION OF PROCESS PLUS I/O TIME AS A TIME BASE WAS CHOSEN IN THE CALCULATION OF WSSIZE SINCE THESE TIMES DO NOT CHANGE WHILE A PROGRAM IS SUSPENDED. THIS MEANS A PROGRAM-S WORKING SET WILL NOT CHANGE DURING THE TIME IT S SUSPENDED.

FOR THIS REASON, THE CRITERIA USED TO RESUME A SUSPENDED PROGRAM ARE BASED ON ITS WORKING SET SIZE (WSSIZE). PRIOR TO THE OVERLAY PASS IN THE OVERLAY CONTROL PROCEDURE, THE WORKING SET SIZE OF EACH PROGRAM SUSPENDED IS COMPARED AGAINST THE SYSTEM AVAILABLE MEMORY. THE HIGHEST PRIORITY PROGRAM WHOSE WORKING SET SIZE IS LESS THAN THE CURRENT AMOUNT OF AVAILABLE SPACE WILL BE RESUMED.

IT CAN HAPPEN THAT SUSPENDING A PROGRAM CAN RESULT IN SOME AMOUNT OF MAIN MEMORY SPACE BEING MADE AVAILABLE. RATHER THAN LET THIS SPACE REMAIN IDLE, THE SYSTEM MAY SELECT A PROGRAM FROM THE SCHEDULE QUEUE AND PLACE IT INTO EXECUTION PROVIDING THE PROGRAM-S CORE ESTIMATE INDICATES IT WILL "FIT" AND THERE IS NO MORE THEN ONE PROGRAM SUSPENDED

SINCE THE AUTOMATIC PROGRAM SUSPENSION PROCESS IS A PART OF THE OVERLAY CONTROL PROCEDURE, TURNING IT OFF (SETTING THE SYSTEM OVERLAY GOAL TO ZERO) WILL ALSO DEACTIVATE THE AUTOMATIC PROGRAM SUSPENSION PROCESS. IF OVERLAY CONTROL IS TURNED OFF WHILE JOBS ARE SUSPENDED, THESE PROGRAMS ARE RESUMED BEFORE THE PROCEDURE DEALLOCATES ITSELF.

CONTROL PROGRAMS:

THERE ARE CERTAIN TYPES OF PROGRAMS SUCH AS DATACOM MCS PROGRAMS, DATA MANAGEMENT CONTROL PROCEDURES, USER-WRITTEN CONTROL PROGRAMS, ETC., WHICH MUST NOT BE SUSPENDED OR BECOME SCHEDULED AT ANY TIME. FOR THIS REASON, A NEW OPERATOR MESSAGE HAS BEEN IMPLEMENTED (SEE APPENDIX A FOR DETAILS) WHICH ALLOWS THE OPERATOR TO MARK OR UN-MARK A PROGRAM AS A CONTROL PROGRAM. A PROGRAM MARKED AS A CONTROL PROGRAM WILL ALWAYS GO INTO IMMEDIATE EXECUTION (I.E., NEVER BECOME SCHEDULED). ADDITIONALLY, IF OVERLAY CONTROL IS TURNED ON, IT WILL NEVER BE SELECTED FOR SUSPENSION. CAUTION: INDISCRIMINATE USE OF THE CONTROL PROGRAM OPTION CAN RESULT IN SYSTEM FAILURES DUE TO "NO MEMORY" CONDITIONS.

REFERENCES

1. DENNING, PETER J., THE WORKING SET MODEL FOR PROGRAM BEHAVIOR, COMMUNICATIONS OF THE ACM, VOL. II, NUMBER 5, (MAY, 1968), 323-333.

2. DENNING, PETER J., VIRTUAL MEMORY, COMPUTING SURVEYS, VOL. 2, NO. 3, (SEPTEMBER, 1970), 153-189.

OPERATOR MESSAGES FOR OVERLAY CONTROL

1. THE SF MESSAGE

THE OPERATOR MESSAGE SF IS USED TO CONTROL AND READOUT THE PARAMETERS ASSOCIATED WITH MEMORY MANAGEMENT. IN RESPONSE TO AN SF <ETX> THE SYSTEM WILL DISPLAY ON THE 9352 SCREEN A MESSAGE IN THE FORM OF THE FOLLOWING EXAMPLE:

```
1) OLAYGOAL = 25 % PER MINUTE
2) AVAILMIN = 7% (XXXXXXX WORDS)
3) FACTOR = 100
```

THE FACTORS DISPLAYED BY THE SF<ETX> CAN BE SET BY AN SF <PARAMETER NUMBER> <PARAMETER-VALUE> <ETX> FROM THE 9352 KEYBOARD. FOR EXAMPLE:

```
SF 1 30 <ETX> WOULD SET OLAYGOAL = 30%
SF 2 5 <ETX> WOULD SET AVAILMIN = 5%
SF 3 80 <ETX> WOULD SET FACTOR = 80%
```

THE SYSTEM ALLOWS THE SETTING OF ANY COMBINATION OF ONE, MORE THAN ONE, OR ALL PARAMETERS TO BE SET WITH A SINGLE INPUT; FOR EXAMPLE:

SF 1 25 2 5 3 100 <ETX> WOULD CAUSE:

PARAMETER 1 (OLAYGOAL) TO BE SET TO 25% PER SECOND
PARAMETER 2 (AVAILMIN) TO BE SET TO 5%
PARAMETER 3 (FACTOR) TO BE SET TO 10%

ADDITIONALLY, SINCE OPERATORS ARE USED TO SETTING FACTOR BY ENTERING SF <INTEGER> <ETX>, AN SF ENTRY FOLLOWED BY A SINGLE VALUE (I.E., SF 80 <ETX>) WILL ASSUME THE PARAMETER, FACTOR, IS TO BE SET TO 80.

2. THE OG MESSAGE

A NEW MESSAGE HAS BEEN IMPLEMENTED. IT IS IN THE FORM:

<MIX-NUMBER> OG <INTEGER> <ETX>

THIS MESSAGE ALLOWS UNIQUELY SETTING THE OLAYGOAL OF EACH PROGRAM IN THE MIX. <MIX-NUMBER> OG <ETX> WILL DISPLAY THE CURRENT VALUE OF OLAYGOAL.

3. THE CU MESSAGE

THE FORMAT OF THE RESPONSE TO THE OPERATOR INPUT MESSAGE:

<MIX-NUMBER> CU <ETX>

INCLUDES A SYSTEM-MAINTAINED PARAMETER IN THE FORM:

WSSIZE = <INTEGER>

WHICH WILL BE DISPLAYED FOR BOTH THE JOB STACK AND THE SEGMENT DICTIONARY STACK. THE PARAMETER, WSSIZE, IS THE WORKING SET OF THE PROGRAM, I.E., THE SIZE IN WORDS OF THE AMOUNT OF MEMORY RESOURCES THIS PROGRAM HAS BEEN USING.

EXPLANATION OF PARAMETERS

A - OLAYGOAL

THE PARAMETER, OLAYGOAL, IS USED TO SET AN "OVERLAY IN ADVANCE" RATE. THIS OLAYGOAL, SET BY THE SF MESSAGE, IS THE OVERLAY RATE ASSIGNED TO THE MCP AND SYSTEM INTRINSICS. ADDITIONALLY, IT IS USED AS A BASE TO DERIVE A DEFAULT VALUE BASED ON PRIORITY FOR ALL PROGRAMS ENTERING THE MIX. THE OG MESSAGE CAN BE USED TO OVERRIDE THIS DEFAULT SETTING FOR A PARTICULAR PROGRAM.

THE VALUE OF OLAYGOAL REPRESENTS A PERCENTAGE OF A PROGRAM S OVERLAYABLE SPACE WHICH WILL BE REMOVED FROM CORE ON A PER MINUTE BASIS. THIS TENDS TO KEEP CORE FREE OF UNUSED SEGMENTS, THUS ALLOWING MORE PROGRAMS TO BE MULTI-PROCESSED.

B - AVAILMIN

THE SYSTEM MONITORS AVAILABLE CORE SPACE IN AN ATTEMPT TO DETERMINE WHEN THRASHING OCCURS. THE USER CAN SET A SYSTEM PARAMETER, AVAILMIN, AS A PERCENTAGE OF TOTAL CORE. THE SYSTEM MONITORS AVAILABLE SPACE; AND WHEN IT BECOMES LESS THAN AVAILMIN, IT WILL AUTOMATICALLY SUSPEND THE LOWEST PRIORITY JOB IN THE MIX.

THERE ARE SOME EXCEPTIONS, I.E., THE FOLLOWING ARE NEVER SELECTED FOR SUSPENSION:

1. ANY PROGRAM MARKED AS A CONTROL PROGRAM;
2. ANY PROGRAM WHICH HAS CALLED SORT;
3. THAT PROGRAM WHICH WOULD REDUCE THE NUMBER OF RUNNING PROGRAMS BELOW THE NUMBER OF PROCESSORS.

EVEN THOUGH A PROGRAM IS SUSPENDED, THE SYSTEM WILL CONTINUE TO OVERLAY ITS SEGMENTS. EVENTUALLY, ALL OF A PROGRAM S OVERLAY SPACE WILL BECOME AVAILABLE.

SETTING OLAYGOAL - 0 WILL CAUSE THE SYSTEM NOT TO AUTOMATICALLY SUSPEND JOBS. IF THE OLAYGOAL IS SET TO ZERO WHILE PROGRAMS ARE SUSPENDED, THE SYSTEM WILL AUTOMATICALLY RESUME THESE PROGRAMS.

USAGE CONSIDERATIONS

PRELIMINARY INDICATIONS ARE THAT THE SYSTEM PERFORMS WELL WHEN THE WORKING SET FACTORS ARE SET AS FOLLOWS:

```
OLAYGOAL = 25%
AVAILMIN = 2%
FACTOR = 100
```

HOWEVER, DUE TO VARIATIONS IN SYSTEM CONFIGURATIONS AND SYSTEM OBJECTIVES, OTHER VALUES MAY BE MORE SUITABLE.

THE CP MESSAGE:

THE OPERATOR MESSAGE, CP, CAN BE USED TO MARK A PROGRAM AS A CONTROL PROGRAM. A PROGRAM MARKED AS A CONTROL PROGRAM WILL ALWAYS BE BROUGHT INTO THE MIX (NEVER SCHEDULED) AND ONCE IN THE MIX, IT WILL NOT BE SELECTED FOR SUSPENSION. THE FORMAT OF THE MESSAGE IS:

```
CP <RESET-OPTION> <PROGRAM-NAME>
```

IF THE <RESET-OPTION> IS LEFT/ <EMPTY> THE PROGRAM WILL BE MARKED AS A CONTROL PROGRAM.

EXAMPLE:

```
(OPERATOR) CP SYSTEM/HARDCOPY
(SYSTEM) SYSTEM/HARDCOPY CONTROL PROGRAM
```

IF THE <RESET-OPTION> IS A MINUS SIGN (-), THE PROGRAM WILL BE FLAGGED AS A NON-CONTROL PROGRAM.

EXAMPLE:

```
(OPERATOR) CP - SYSTEM/RJE
(SYSTEM) SYSTEM/RJE CONTROL RESET
```

CAUTION: INDISCRIMINATE USE OF CP CAN CAUSE SYSTEM FAILURES AS TO "NO MEMORY" CONDITIONS.

D0060 MCP - "PC" KEYBOARD MESSAGE - 05-24-72

A NEW KEYBOARD INPUT MESSAGE, "PC", HAS BEEN ADDED TO THE MCP, THE

MEANING OF WHICH IS "PRINT CONFIGURATION". THE MCP WILL RESPOND WITH:

X PROCESSORS : L,M,N
Y MULTIPLEXORS : R,S,T
Z MEMORY MODS : 00-VV READY
(X, Y, Z WOULD REFLECT THE APPROPRIATE NUMBERS.)

D0061 BASIC - RELATIONAL OPERATORS IN BASIC - 07-17-72

TO AVOID AMBIGUITIES WHICH MAY OCCUR WHEN USING THE TWO-LETTER MNEMONIC RELATIONAL OPERATORS IN BASIC, THESE MNEMONICS SHOULD NOW BE PRECEDED BY A:

- BACK SLASH CHARACTER WHEN THE BASIC PROGRAM IS ENTERED FROM A REMOTE TERMINAL.
- BACK SLASH CHARACTER (0 8 2 PUNCH) WHEN ENTERED THROUGH THE CARD READER AS AN EBCDIC CARD DECK.
- BCL MULTIPLICATION OPERATOR (- 0 PUNCH) WHEN ENTERED THROUGH THE CARD READER AS A BCL CARD DECK.

HOWEVER, FOR THE USERS EASE AND MORE EFFICIENT BASIC PROGRAMS, IT IS RECOMMENDED THAT SYMBOLS EQUIVALENT TO THE TWO-LETTER MNEMONIC BE USED.

BACK SLASH EQ	=
BACK SLASH LE	≤, <=, =<
BACK SLASH LT	<
BACK SLASH GE	≥, >=, =>
BACK SLASH GT	>
BACK SLASH NE	<>, ><, ≠

WARNING: THE EBCDIC BACK SLASH CHARACTER DOES NOT TRANSLATE TO A VALID BCL CHARACTER. THEREFORE, OF A BCL "NEWTAPE" FILE IS CREATED FROM MERGED EBCDIC "CARD" AND "TAPE" FILES, ANY BACK SLASH

CHARACTERS WILL TRANSLATE TO INVALID CHARACTERS ON THE BCL NEWTAPE. IF THE BCL NEWTAPE IS LATER USED AS THE INPUT FILE "TAPE", THESE INVALID CHARACTERS WILL CAUSE SYNTAX ERRORS.

D0062 FORTRAN - IO FILE ATTRIBUTE - 07-19-72

THE "IO" FILE ATTRIBUTE MNEMONIC IS IMPLEMENTED WHICH, WHEN USED, WILL SET MYUSE (FILE ATTRIBUTE #30) TO THREE. THIS IS INTENDED TO FACILITATE OBJECT I/O VIA CANDE.

D0063 RANDOM-SERIAL FILE ATTRIBUTES - 07-23-72

THIS SYSTEM NOTE HAS BEEN CHANGED TO P0616.

D0064 SRCEDC1000 - DC1000 DISCONNECT - 07-21-72

THIS CHANGE WILL CAUSE THE DC1000 TO PERFORM A DISCONNECT WHEN REQUESTED TO DO SO BY RJE. THIS PATCH IS ONLY EFFECTIVE WHEN $ SET MODEM PROGRAM OPTION CARD IS USED WHEN ASSEMBLING SYMBOL/ SOURCEDC1000 SYMBOLIC.

D0002 COBOL - FREE FORM SOURCE INPUT - 06-14-72

D0065 FORTRAN - CROSS REFERENCE IN FORTRAN - 08-25-72

THE XREF COMPILER OPTION, WHEN SET, CAUSES (IN THE EVENT OF SUCCESSFUL COMPILATION) A CROSS REFERENCE OF ALL IDENTIFIERS USED IN THE COMPILED PROGRAM. THIS IS ACCOMPLISHED BY INITIATING SYSTEM/XREFANALYZER AT THE END OF COMPILATION, USING A FILE CREATED DURING COMPILATION THAT CONTAINS THE NECESSARY INFORMATION. THESE IDENTIFIERS ARE ARRANGED ACCORDING TO THE EBCDIC COLLATING SEQUENCE WITH LISTS OF SEQUENCE NUMBERS OF CARD IMAGES ON WHICH THE IDENTIFIER APPEARS. THE LIST OPTIONS NEED NOT BE SET TO GENERATE A CROSS REFERENCE LISTING, SYSTEM/XREFANALYZER TREATS THE FIRST REFERENCE OF THE IDENTIFIER AS A SPECIAL CASE AND SO INDICATES THIS VIA THE DECLARATION LINE ON WHICH IS THE IDENTIFIERS CLASS AND TYPE.

XREF SHOULD BE SET PRIOR TO THE FIRST SOURCE STATEMENT AND ONCE SET, IT MAY NOT BE RESET.

D0066 SYSTEM MODIFICATION - 08-10-72

THESE PROCEDURES ASSUME THAT THE "SYSTEM GENERATION" PROCESS HAS BEEN PERFORMED PREVIOUSLY, EITHER BY THE CUSTOMER OR BURROUGHS. IF THE STANDARD RELEASE IS TO BE MODIFIED, THEN ALL OCCURRENCES OF THE FOLLOWING IDENTIFIERS USED IN THE MODEL DECKS SHOULD BE REPLACED BY THE FOLLOWING COUNTERPART:

```
NEW MCP    =>   MCP
NEW        =>   SYSTEM
NEWINTS    =>   INTRINSICS
```

NOTICE THAT BURROUGHS DOES NOT PROVIDE A FILE "INFO/INTRINSICS" ON THE RELEASE TAPES. SUCH A FILE CAN BE CREATED BY COMPILING THE ESPOL GLOBALS AND USINGDUMPINFO (SEE THE SYSTEM GENERATION MODELS).

1) IF THERE ARE NO PATCHES IN THE MCP, GO TO STEP 3.

2) COMPILE SEPARATELY THE MCP PATCHES USING THE

FOLLOWING DECK SET UP AS A MODEL:

```
<I>COMPILE NEW/PROCS ESPOL LIBRARY
<I>ESPOL FILE TAPE (TITLE = SYMBOL/NEWMCP)
<I>ESPOL FILE INFO (TITLE = INFO/NEWMCP)
<I>DATA  .  ( OR BCL )
[   % START OF GLOBALS
$ LOADINFO
]   % END OF GLOBALS
$ MERGE CHECK
$ GO TO 74610000   % WRITEALABEL
% YOUR PATCHES TO WRITEALAB   74795100
$     % WRITEALABEL          74802001
                :
                :
PROCEDURE DUMMY; BEGIN LABEL AWAY; END  %%%%% LAST CARD %%%%%
```

```
<I>END.
```

THE "$ GO TO <SEQ NUMBER>" CARD SHOULD BE BLANK IN COLUMNS 73 THROUGH 80 AND THE <SEQ NUMBER> SHOULD BE THE FIRST SEQUENCE NUMBER OF THE PROCEDURE TO BE MODIFIED. THE LAST DOLLAR CARD SHOULD CONTAIN THE SEQUENCE NUMBER OF THE LAST CARD IN THE MODIFIED PROCEDURE PLUS ONE. ANY NUMBER OF "$ GO TO " CARDS AND MATCHING DOLLAR CARD DELIMITERS MAY BE PROCESSED DURING ONE COMPILATION.

IMPORTANT: IF ANY GLOBALS (SEQ NUMBERS 00000000-19990000) ARE MODIFIED OR IF ANY SEGMENT ONE PROCEDURE OR ANY SEGMENT FIVE PROCEDURE IS MODIFIED, THEN THE WHOLE MCP MUST BE RECOMPILED. EACH NEW MODIFICATION TO THE MCP GLOBALS REQUIRES ANOTHER "SYSTEM GENERATION", IN GENERAL. (NEW GLOBALS NEED NOT BE ADDED TO THE GLOBAL SECTION OF THE MCP AND MAY BE ADDED BY THE BINDER BEGINNING WITH II.3.)

3) IF THERE ARE PATCHES TO THE SORT, RECOMPILE THE SORT USING THE FOLLOWING DECK AS A MODEL:

```
<I>COMPILE NEW/SORT ESPOL LIBRARY
<I>ESPOL FILE INFO (TITLE = INFO/NEWMCP)
<I>ESPOL FILE TAPE (TITLE = SYMBOL/SORT)
<I>DATA  .  (OR BCL)
$ MERGE CHECK
% YOUR PATCHES
<I>END.
```

4) IF THERE ARE PATCHES TO MAINTENANCE, RECOMPILE IT USING THE FOLLOWING DECK AS A MODEL:

```
<I>COMPILE NEW/MAINTENANCE ESPOL LIBRARY
<I>ESPOL FILE INFO (TITLE = INFO/NEWMCP)
<I>ESPOL FILE TAPE (TITLE = SYMBOL/MAINTENANCE)
<I>DATA  .  (OR BCL)
$ MERGE CHECK
% YOUR PATCHES
<I>END.
```

4) FINALLY, BIND YOUR NEWMCP USING THE FOLLOWING DECK AS A MODEL:

<I>BIND FINAL/MCP WITH BINDER LIBRARY <I>BIND FILE HOST (TITLE = NEW/MCP) <I>DATA BIND WRITEALABEL,= FROM NEW/=,SYSTEM/=; STOP; <I> END.

EACH MCP PROCEDURE MODIFIED IN STEPS TWO, THREE, OR FOUR MUST BE EXPLICITLY LISTED ON A BIND CARD, OTHERWISE, THE BINDER WILL NOT REPLACE THE EXISTING PROCEDURES.

6) IF THERE ARE PATCHES TO THE ALGOL INTRINSICS, RECOMPILE THEM AS FOLLOWS:

```
<I>COMPILE NEW/ALGINTS ALGOL LIBRARY.
<I>ALGOL FILE TAPE (TITLE = SYMBOL/ALGOLINTRINSICS)
<I>DATA
$ MERGE CHECK
$       % END OF GLOBALS        00000999
$ GO TO 10000000   % PLIDECAT
% YOUR PATCHES TO PLIDECAT
$                  % PLIDECAT   10016001
                :
                :
PROCEDURE DUMMY; BEGIN LABEL AWAY; END %% LAST CARD %%
<I>END.
```

7) IF THERE ARE CHANGES TO THE ESPOL INTRINSICS, RECOMPILE THE APPROPRIATE PROCEDURES AS FOLLOWS:

```
<I>COMPILE NEW/ESPINTS ESPOL LIBRARY
<I>ESPOL FILE TAPE (TITLE = SYMBOL/ESPOLINTRINSICS)
<I>ESPOL FILE INFO (TITLE = INFO/NEWINTS)
<I>DATA  .  (OR BCL)
$ MERGE CHECK
[     % START OF GLOBALS
$ LOADINFO
]     % END OF GLOBALS
$ GO TO 03080000 % SUPERMON
% YOUR PATCHES TO SUPERMON
% SUPERMON        03236001
                :
```

```
                :
   PROCEDURE DUMMY; BEING LABEL AWAY; END.  % LAST CARD %%
   <I>END.
```

THE INTRINSICS INFO FILE (INFO/NEWINTS) IS NOT PROVIDED BY BURROUGHS BUT CAN BE GENERATED SIMPLY BY COMPILING THE INTRINSICS ACCORDING TO SYSTEM GENERATION PROCEDURE REGARDING ESPOL INTRINSICS.

8) IF ANY INTRINSIC MODIFICATIONS HAVE BEEN MADE IN STEPS SIX OR SEVEN, BIND A NEW SET OF INTRINSICS ACCORDING TO THE FOLLOWING:

```
   <I>BIND FINAL/INTRINSICS BINDER LIBRARY
   <I>BINDER FILE HOST (TITLE = NEW/INTRINSICS)
   <I>DATA
   BIND
   PLIDECAT
   SUPERMON
   FROM NEW/=;
   <I>END.
```

IMPORTANT: DO NOT SET THE DOLLAR OPTION "INTRINSICS".

THIS PROCESS WILL FAIL IF THE "$ SET INTRINSICS" CARD IS INCLUDED IN THE BIND DECK.

9) IF FINAL/MCP WAS MADE, CM TO FINAL/MCP ELSE CM TO NEW/MCP.

10) IF FINAL/INTRINSICS WERE CREATED, THEN CI TO FINAL/INTRINSICS. OTHERWISE, CI NEW/INTRINSICS.

SYSTEM GENERATION:

SYSTEM GENERATION NEED TAKE PLACE ONLY WHEN MCP GLOBAL DECLARATIONS ARE ALTERED. THE II.3 BINDER WILL BE ABLE TO ADD GLOBALS SO THAT THE SIMPLE ADDITION OF A GLOBAL DECLARATION MAY NOT IMPLY THAT A NEW SYSTEM MUST BE GENERATED. ONCE A SYSTEM HAS BEEN GENERATED REFLECTING THE ALTERATIONS TO THE MCP GLOBALS, THE SYSTEM MODIFICATION TECHNIQUES SHOULD BE USED FOR FURTHER DEBUGGING AND CHANGES.

1) RECOMPILE THE WHOLE MCP USING THE FOLLOWING DECK AS A MODEL:

```
<I>COMPILE HOST/MCP ESPOL LIBRARY
<I>ESPOL FILE TAPE (TITLE = SYMBOL/MCP)
<I>ESPOL FILE NEWTAPE (TITLE = SYMBOL/NEWMCP)
<I>ESPOL FILE INFO (TITLE = INFO/NEWMCP)
<I>DATA  .  (OR BCL)
$ MERGE CHECK NEWSEQERR
$ SET NEW    % FOR THE ESPOL INTRINSIC COMPILATIONS
% YOUR PATCHES TO THE MCP GLOBALS
$ DUMPINFO   % FOR SORT & MAINTENANCE
      COMPILATIONS          19990000
% YOUR PATCHES TO THE PROCEDURES TO BE
ALTERED OR ADDED
<I>END.
```

IMPORTANT: YOU MUST SET NEW AND CREATE A DISK SYMBOL FILE OF THE NEW MCP FOR USE DURING COMPILATION OF THE ESPOL INTRINSICS.

FURTHERMORE, YOU MUST DUMPINFO FOR USE DURING SORT AND MAINTENANCE COMPILATIONS.

2) COMPILE THE SORT USING THE FOLLOWING DECK AS A MODEL:

```
<I>COMPILE NEW/SORT ESPOL LIBRARY
<I>ESPOL FILE INFO (TITLE = INFO/NEWMCP)
<I>ESPOL FILE TAPE (TITLE = SYMBOL/SORT)
<I>DATA   .  (OR BCL)
$ MERGE CHECK
% YOUR PATCHES
<I>END.
```

3) COMPILE MAINTENANCE USING THE FOLLOWING DECK AS A MODEL:

```
<I>COMPILE NEW/MAINTENANCE ESPOL LIBRARY
<I>ESPOL FILE INFO (TITLE = INFO/NEWMCP)
<I>ESPOL FILE TAPE (TITLE = SYMBOL/MAINTENANCE)
<I> DATA  .  (OR BCL)
$ MERGE CHECK
% YOUR PATCHES
<I>END.
```

4) BIND A COMPLETE MCP USING AS FOLLOWS:

```
<I>COMPILE NEW/MAINTENANCE ESPOL LIBRARY
<I>ESPOL FILE INFO (TITLE = INFO/NEWMCP)
<I>ESPOL FILE TAPE (TITLE = SYMBOL/MAINTENANCE)
<I>DATA    .  (OR BCL)
$ MERGE CHECK
% YOUR PATCHES
<I>END.
```

4) BIND A COMPLETE MCP USING AS FOLLOWS:

```
<I>BIND NEW/MCP WITH BINDER LIBRARY
<I>BIND FILE HOST (TITLE = HOST/MCP)
<I>DATA
BIND = FROM NEW/=;
<I>END.
```

5) COMPILE THE ALGOL INTRINSICS:

```
<I>COMPILE INT/ALGINTS ALGOL LIBRARY
<I>ALGOL FILE TAPE (TITLE = SYMBOL/ALGOLINTRINSICS)
<I>DATA
$ MERGE CHECK
% YOUR PATCHES
<I>END.
```

6) COMPILE ALL THE ESPOL INTRINSICS USING THE FOLLOWING DECK AS A MODEL:

```
<I>COMPILE INT/ESPINTS ESPOL LIBRARY
<I>ESPOL FILE TAPE (TITLE = SYMBOL/ESPOLINTRINSICS)
<I>ESPOL FILE INFO (TITLE = INFO/NEWINTS)
<I>DATA    . (OR BCL)
$ MERGE CHECK
% YOUR PATCHES TO THE GLOBALS
$ DUMPINFO    % SAVE FOR SEPARATE COMPILATIONS    00754999
% YOUR PATCHES TO THE INDIVIDUAL INTRINSICS
<I>END.
```

7) BIND A NEW INTRINSIC FILE ACCORDING TO:

```
<I>BIND NEW/INTRINSICS BINDER LIBRARY
<I>DATA
$ SET INTRINSICS
<I>END.
```

8) OPTIONAL: REMOVE SOME JUNK FROM DISK OR COPY TO TAPE.

```
<I>COPY INT/=,HOST/= TO SAVE
<I>REMOVE INT,HOST
<I>END.
```

9) OPTIONAL BUT RECOMMENDED: SAVE THE FILES REQUIRED FOR SYSTEM MODIFICATION AS FOLLOWS:

```
<I>COPY INFO/NEWMCP,INFO/NEWINTS,NEW/MCP,
   NEW/INTRINSICS,SYMBOL/NEWMCP, - TO SAVED
   <I>END.
```

10) CM TO NEW/MCP AND CI TO NEW/INTRINSICS.

STEPS TWO AND THREE MAY BE PERFORMED IN PARALLEL.
STEPS FIVE AND SIX MAY BE PERFORMED IN PARALLEL.

D0067 PLI COMPILER GENERATION - 09-07-72

THE FOLLOWING COMPILE DECK WILL PRODUCE A NEW PL/I COMPILER:

```
<I> COMPILE SYSTEM/PL/I ALGOL LIBRARY
<I> CORE = 15000, 13000, STACK = 1500
<I> ALGOL FILE TAPE = SYMBOL/PL/I.
<I> ALGOL FILE PLDECS = SYMBOL/PL/TABLES.
<I> DATA
$ MERGE
      .
      .
      .
   PATCHES
      .
```

.
.
<I> END.

IF THE "$ MERGE" CARD IS REPLACED BY "$ MERGE DEBUG", AN INSTRUMENTED COMPILER WILL BE PRODUCED WHICH INCLUDES MANY COMPILER DEBUGGING TOOLS.

D0070 COBOL - SAME [RECORD] AREA - 08-02-72

THIS CHANGE MAKES THE WORD "RECORD" OPTIONAL IN THE "SAME RECORD/ SORT AREA" CLAUSE. WHEN NEITHER "RECORD" OR "SORT" ARE SPECIFIED, THE WORD "RECORD" IS ASSUMED.

D0071 COBOL - JUSTIFIED [RIGHT] - 08-02-72

THIS CHANGE ALLOWS THE WORD "RIGHT" TO BE OPTIONAL IN THE "JUSTIFIED" CLAUSE.

D0072 COBOL - NOTE AND ID PARAGRAPHS - 08-02-72

THIS COBOL CHANGE CAUSES THE COMPILER TO TOTALLY IGNORE ALL CHARACTERS IN "NOTE" PARAGRAPHS AND IN THE PARAGRAPHS OF THE ID DIVISION. PREVIOUS LEVELS OF THE COMPILER WERE SENSITIVE TO SPECIAL CHARACTERS AND WOULD PRODUCE ERRONEOUS FATAL ERROR MESSAGES. ALSO, THE COMPILERS DICTIONARY IS NO LONGER AFFECTED BY THESE ENTRIES.

D0073 COBOL - FOR [MULTIPLE] REEL - 08-04-72

THE WORD "MULTIPLE" IS NOW OPTIONAL IN THE "FOR MULTIPLE REEL/UNIT" CLAUSE.

D0074 COBOL - ADVANCING OPTION OF WRITE - 08-04-72

THIS COBOL CHANGE IMPLEMENTS SEVERAL NEW FEATURES FOR THE "ADVANCING" OPTION OF THE "WRITE" STATEMENT AS FOLLOWS:

1. CHANNEL <INTEGER> IS <MNEMONIC-NAME>

   THE ABOVE STATEMENT WHEN MADE IN THE SPECIAL-NAMES PARAGRAPH ASSOCIATES A <MNEMONIC-NAME> WITH A SPECIFIED CHANNEL IN THE PRINTER CARRIAGE CONTROL TAPE.. THE <INTEGER> IN THE ABOVE FORMAT MUST BE A POSITIVE INTEGER AND ONLY VALUES FROM ONE THRU 11 ARE ACCEPTABLE.

2. WRITE....ADVANCING TO.....

   THE WORD "TO" IS NOW ALLOWED AS A NOISE WORD.

3. WRITE....ADVANCING TO PAGE

   "PAGE" IS EQUIVALENT TO "CHANNEL 1".

4. WRITE....ADVANCING TO <MNEMONIC-NAME>

   IN THIS FORMAT, <MNEMONIC-NAME> MUST BE ASSOCIATED WITH A CHANNEL OF THE CARRIAGE CONTROL TAPE AS SHOWN IN (1) ABOVE.

5. THE COMPILATION OF:

   WRITE...ADVANCING CHANNEL <INTEGER> HAS BEEN SPEEDED UP CONSIDERABLY.

D0075 ESPOL - "USING PICTURE" EXTENSION - 08-02-72

A POINTER EXPRESSION IS NOW AN ACCEPTABLE ALTERNATIVE FOR THE PICTURE IDENTIFIER FOLLOWING THE KEY WORD "USING" IN A REPLACE STATEMENT. THE USER MUST GUARANTEE THE VALIDITY OF THE POINTER AND THE EDIT TABLE TO WHICH IT POINTS.

D0076 PATCH - EXECUTE OPTION - 05-05-72

A NEW PATCH CONTROL STATEMENT, "EXECUTE", HAS BEEN ADDED TO ALLOW EXECUTION OF A PROGRAM VIA A ZIP AFTER A SUCCESSFUL PATCH GENERATION. THE EXECUTE CONTROL CARDS ARE SPECIFIED ON "$*" CARDS

IN AN IDENTICAL MANNER AS FOR COMPILATIONS.

D0077 ESPOLINTRN - PROCEDURE NUMBERCONVERT - 05-05-72

THIS PATCH TO "NUMBERCONVERT" ALLOWS IT TO BE CALLED FROM AN MCP PROCEDURE WITHOUT PASSING A DESCRIPTOR TO "EBCDICFORMATTERSTUFF". IF THE CORRESPONDING PARAMETER IS TAG 0, THE INTRINSIC WILL USE "EBCDICFORMATTERSTUFF" BY DEFAULT.

D0078 ESPOLINTRN - NEW PLI INTRINSICS - 05-08-72

NEW PL/I INTRINSICS HAVE BEEN INCORPORATED FOR PICTURE PROCESSING, ARITHMETIC/BIT/CHARACTER CONVERSION, AND BIT OPERATIONS.

D0079 ALGOLINTRN - NEW INTRINSICS - 05-08-72

THIS ALGOL INTRINSICS CHANGE INCORPORATES NEW INTRINSICS TO HANDLE PL/I BUILTIN FUNCTIONS AS FOLLOWS:

VERIFY
TRANSLATE
INDEX
DECAT

D0080 ALGOL - OPTION "NOBINDINFO" - 08-11-72

A NEW DOLLAR CARD OPTION, "NOBINDINFO", HAS BEEN IMPLEMENTED. WHEN IT IS SET, THE COMPILER WILL NOT GENERATE THE PROGRAM DESCRIPTION NECESSARY FOR THE BINDER. THIS OPTION IS IGNORED WHENEVER THE PROCEDURES CAN ONLY BE USED FOR BINDING.

D0081 ESPOLINTRN - SYSTEMLOG - 05-25-72

A NEW INSTALLATION INTRINSIC, "SYSTEMLOG", IS IMPLEMENTED TO BE USED BY AN MCS TO MAKE ENTRIES IN THE SYSTEM LOG. SEE D0037 FOR DETAILS.

D0082 COMPARE - SEQUENCE COMPARE - 05-25-72

FILES MAY NOW BE COMPARED BY SEQUENCE NUMBER WHEN USING SYSTEM/COMPARE. TO COMPARE IN THIS MODE, IT IS NECESSARY TO SPECIFY THE BEGINNING SEQUENCE COLUMN AND THE SEQUENCE LENGTH SEPARATED BY "-". THIS INFORMATION IS INCLUDED ON THE DATA CARD FOLLOWING THE FILE TITLES (IN CARD COLUMN RANGE 49-80). BY DEFAULT, A RECORD-BY-RECORD COMPARISON WILL BE PERFORMED IF SEQUENCE INFORMATION IS NOT PROVIDED.

D0083 COMPARE - ERROR COUNT - 05-25-72

THE MAXIMUM NUMBER OF NON-COMPARES ALLOWED BY SYSTEM/COMPARE BEFORE TERMINATING THE CURRENT COMPARISON CAN NOW BE SPECIFIED AS AN INTEGER ON THE DATA CARD FOLLOWING THE FILE TITLES (IN CARD COLUMN RANGE 49-80). A DEFAULT VALUE OF FIVE IS USED IF AN ERROR COUNT IS NOT SPECIFIED.

D0084 ESPOLINTRN - DCSYSTEMTABLES INTRINSIC - 05-26-72

THIS PATCH IMPLEMENTS A STANDARD INSTALLATION INTRINSIC FUNCTION WHOSE IDENTIFIER IS DCSYSTEMTABLES. IT IS INTENDED TO BE USED SOLELY BY THE SYSTEM/DCSTATUS ROUTINE, DESCRIBED ELSEWHERE IN THE II.3.0 RELEASE DOCUMENTATION.

D0085 ESPOLINTRN - MATH INTRINSICS - 07-19-72

THE FOLLOWING CHANGES HAVE BEEN MADE TO MATH INTRINSICS:

1) CONSTANTS FOR EXP MODE GLOBAL;

2) LN AND LOG ARE REWRITTEN FOR GREATER ACCURACY WITH SAME SPEED;

3) MINOR SPEED IMPROVEMENTS HAVE BEEN MADE IN COS, ARCTAN, ETC.;

4) MINOR SPEED IMPROVEMENTS HAVE BEEN MADE TO SQRT, DSQRT;

5) ERF AND ERFC HAS BEEN REWRITTEN FOR GREATER ACCURACY;

6) DTAN, DSINH, DCOSH, DTANH HAVE BEEN IMPLEMENTED.

D0086 DIAGNOSTMCS - DIAGNOSTICMCS - 07-20-72

THIS IS THE INITIAL RELEASE OF LINEANALYZER MCS. FOR DOCUMENTATION, REFER TO THE B6700 TECHNICAL NEWSLETTER, TITLED "DATACOM LINE ANALYZER".

D0087 ALGOL - ALGOL FAULT DECLARATIONS - 07-18-72

THE WORD "ANYFAULT" HAS BEEN ADDED TO THE LIST OF ACCEPTABLE DECLARATIONS IN A <FAULT NAME> LIST. THIS PERMITS ERROR RECOVERY IN THE EVENT OF ANY PERMITTED ERROR CONDITION. THE SYNTAX OF THIS STATEMENT IS:

```
ON ANYFAULT
```

OR

```
ON ANYFAULT, <STATEMENT>.
```

TO FIND OUT WHICH FAULT OCCURRED, THE USER CAN ACCESS THE REAL-VALUED TASK ATTRIBUTE "HISTORY". IN HIS OWN STACK, THIS WOULD BE A REFERENCE TO "MYSELF.HISTORY".

THE CORRESPONDING FAULTS AND FAULT NUMBERS ARE AS FOLLOWS:

EXPONENTOVERFLOW	2
EXPONENTUNDERFLOW	3
INTEGEROVERFLOW	4
INVALIDOP	8
LOOP OR INSTRUCTION TIMEOUT	9
STRINGPROTECT OR SEG ARRAY	14
ZERODIVIDE	1

IF A PROGRAM UNIT CONTAINS AN ANYFAULT DECLARATION, AND IN

ADDITION, SPECIFIC FAULT DECLARATIONS, THE SPECIFIC FAULT DECLARATION WILL TAKE PRECEDENCE. THUS, IN THE CASE OF THE FOLLOWING:

```
ON ANYFAULT, GO TO L;

ON ZERODIVIDE, GO TO L1;
```

ON A DIVISION BY ZERO, THE STATEMENT "GO TO L1" WOULD BE EXECUTED; ON ALL OTHER FAULTS THE STATEMENT "GO TO L" WOULD BE EXECUTED.

D0088 BASIC - "DATA" STATEMENTS - 08-03-72

THIS PATCH INCREASES THE NUMBER OF ITEMS THAT THE USER MAY SPECIFY WITH "DATA" STATEMENTS FROM 256 TO 1,024. AS A RESULT, THE ARRAY "STRINGPOOL", INTERNAL TO THE COMPILER, MAY NOW BE COMPLETELY USED, THUS ALLOWING THE USER TO SPECIFY MORE LITERAL STRINGS IN A PROGRAM.

D0089 BINDER - SORTED STACK LISTING - 08-02-72

IF THE STACK OPTION IS SET, THE ADDRESSES WILL BE PRINTED TWICE; SORTED ONCE BY ADDRESS AND ONCE BY IDENTIFIER.

D0090 SCR - I-O MODIFIERS - 08-10-72

THE HANDLING OF I/O MODIFIERS HAS BEEN IMPROVED. SYNTAX ERRORS WILL RESULT WHEN MODIFIERS CONFLICT WITHIN THEMSELVES OR THE TYPE OF EQUIPMENT WHICH THEY WERE INTENDED FOR IS NOT SPECIFIED.

DISK ADDRESS RANGE CHECKS HAVE BEEN MOVED FROM A COMPILE TIME CHECK TO A RUN TIME CHECK.

RANGE SPECIFICATION IS NO LONGER REQUIRED ON THE IOLENGTH, OFFSET, AND SEGMENT MODIFIERS.

ANY I/O OPERATION THAT WILL RESULT IN THE "MEMINHIBIT" BIT (BIT43) OF AN IOCW BEING SET WILL NOT REQUIRE THAT A BUFFER BE SPECIFIED.

"TESTOP" HAS BEEN ADDED AS AN I/O MODIFIER WITH THIS ADDITION; THE STATUS OF DISK WRITE LOCKOUT SWITCHES MAY BE INTERROGATED BY USING

THE I/O MODIFIER SEGMENT OR OFFSET IN CONJUNCTION WITH TESTOP. ASIDE FROM THESE TWO MODIFIERS, THE "ON ERROR" CLAUSE IS PERMITTED.

"ATTENTION" HAS BEEN ADDED TO THE I/O MODIFIER LIST. THIS WILL CAUSE BIT45 TO BE SET IN THE RESPECTIVE IOCW. HARDWARE WILL RETURN BIT45 AS BIT1 AT I/O FINISH TIME. THIS COULD BE USED TO CAUSE ENTRY INTO ERROR ROUTINES.

THE WORD "MEMORY" MAY NOW BE SPELLED OUT WHEN USED IN CONJUNCTION WITH "MEMORY PROTECT" OR "MEMORY INHIBIT". HOWEVER, THE SYNTAX OF "MEM" IS STILL ACCEPTED.

THE "*" HAS BEEN IMPLEMENTED AS A MULTIPLE OPERATOR, I.E., SET <VARIABLE>=<VARIABLE>*<VARIABLE>.

THE "/" HAS BEEN IMPLEMENTED AS A DIVIDE OPERATION, I.E., SET <VARIABLE>=<VARIABLE>/<VARIABLE>.

"IOPATH", "STATUS", AND "UNITYPE" HAVE BEEN IMPLEMENTED AS <PRIMARIES>, I.E., SET I = STATUS PK 80 VIA MPX3.

"IOPATH" WILL RETURN THE PATHS AVAILABLE TO THE SPECIFIED UNIT. IF NO PATH IS AVAILABLE, A VALUE OF 0 WILL BE RETURNED. IF A PATH IS AVAILABLE, THEN THE MPX ROUTE WILL BE RETURNED IN BITS 4:4 AND A 1 WILL BE RETURNED IN BIT0. FOR EXAMPLE:

```
SET I = IOPATH UNIT R;
SET I = IOPATH DK 2 VIA MX1.
```

"STATUS" WILL RETURN THE VALUE OF THE STATUS VECTOR PERTINENT TO THE SPECIFIED UNIT AND/OR MPX. FOR EXAMPLE:

```
SET I = STATUS MT5;
SET I = STATUS CR5 VIA MPX2.
```

"UNITTYPE" WILL RETURN THE HARDWARE UNITTYPE FOR THE PERTINENT UNIT. FOR EXAMPLE:

```
SET I = UNITTYPE PK80;
SET I = UNITTYPE PR 5 VIA MX3.
```

IF A MULTIPLEXOR ROUTE IS NOT SPECIFIED IN THE ABOVE STATEMENTS,

THEN THE RESULT RETURNED WILL BE THAT RETURNED BY HARDWARE ON A GENERAL BROADCAST REQUEST.

IMPLEMENTATION IS COMPLETED OF LOGIC TO SPECIFY MULTIPLERROR ROUTES. THIS MAY BE ACCOMPLISHED IN TWO WAYS:

1) UNIT DECLARATION

```
UNIT Q = DK 5 VIA MPX3;
```

OR

2) READ DK 5 VIA MPX3....;

IF AN INVALID MULTIPLEXOR IS SPECIFIED, AN "INVALID ADDRESS" FAULT WILL OCCUR ON THE PERTINENT STATEMENT AND THE NEXT STATEMENT EXECUTED. A MULTIPLEXOR IS SPECIFIED FOR AN I/O OPERATION, BUT A PATH WILL NOT BECOME AVAILABLE WITHIN 90 SECONDS; THE PROGRAM WILL BE DS-ED UPON COMPLETION OF THE 90 SECOND TIME-OUT WITH A RUN TIME ERROR OF "IO TIME OUT".

VARIABLE UNITS AND MULTIPLEXOR ROUTES ARE NOW AVAILABLE. THIS IS ACCOMPLISHED IN THE FOLLOWING MANNER:

```
VARIABLE R, RANGE = <LOWEST UNIT NUMBER> THRU
   <HIGHEST UNIT NUMBER>;

VARIABLE M, RANGE = 1 THROUGH <HIGHEST MPX NUMBER +1>;
UNIT U = R VIA MPX M;
VARIABLE Q;
WHILE M < HIGHRANGEOF (M) DO
  BEGIN
      FOR HIGHRANGEOF(R) DO
        BEGIN
          SET Q = UNITTYPE UNIT U;
          OF Q > 0 THHEN
            DISPLAY("UNIT TYPE OF UNIT ",R," IS ",Q);
          IF R < HIGHRANGEOF(R) THEN SET R = R+1;
        END;
      SET M= M+1;
END;
```

THIS PROGRAM WILL DISPLAY THE UNITTYPE FOR ALL EQUIPMENT AND ALL MULTIPLEXORS CONNECTED TO THE GIVEN SYSTEM.

VARIABLE UNIT/MULTIPLEX DESIGNATIONS HAVE ALSO BEEN EXTENDED TO TAPE AND DISK VERIFY.

A VARIABLE UNIT OR MULTIPLEX MAY ONLY BE SPECIFIED WITHIN THE UNIT DECLARATION.

"FILE" SYNTAX HAS BEEN MODIFIED TO ALLOW SPECIFYING A SPECIFIC MULTIPLEXOR, I.E.,

```
FILE XYZ = "BADDISK/....." VIA MPX1;
```

A VARIABLE MAY ALSO BE USED AS THE MULTIPLEXOR DESIGNATOR.

ALL VARIABLES USED WITHIN THE "UNIT" DECLARATION MUST HAVE RANGE SPECIFICATIONS INCLUDED.

A "RELEASE" VERB HAS BEEN IMPLEMENTED. THE FUNCTION OF THIS VERB IS

TO RETURN A SPECIFIED UNIT TO THE SYSTEM. THE SYNTAX IS:
RELEASE UNIT R.

IF THE SPECIFIED UNIT IS NOT ASSIGNED TO SCR, THEN AN APPROPRIATE MESSAGE WILL BE DISPLAYED AND THE NEXT STATEMENT EXECUTED. IF THE UNIT IS ASSIGNED TO SCR, THEN IT IS GIVEN BACK TO THE SYSTEM IN SOME MANNER AS UNITS ARE RETURNED TO THE SYSTEM WHEN SCR GOES TO EOJ. IF THE SAME UNIT IS AGAIN REFERENCED WITHIN SCR, IT WILL THEN BE ASSIGNED, WHICH AT THIS TIME MAY CAUSE THE SCR JOB TO BE DS-ED ON A RUN TIME ERROR IF THE UNIT IS ALREADY ASSIGNED TO A PREVIOUS JOB. NO I/O OPERATIONS MAY BE IN THE VERBS "STATUS", "PATH", AND "TEST" NO LONGER ASSIGN THE SPECIFIED UNIT TO MAINTENANCE.

WHEN THE PRIMARIES "IOPATH", "STATUS", AND "UNITTYPE" ARE USED, THE SPECIFIED UNIT IS ALSO NOT ASSIGNED.

THE ABILITY TO SELECT A RESERVED UNIT WITHOUT HAVING TO SPECIFY THE SPECIFIC UNIT NUMBER HAS BEEN IMPLEMENTED. THIS PATCH WORDS IN

CONJUNCTION WITH THE KEYBOARD MESSAGE "UR". A NEW SYNTAX OF:

"UNIT <IDENTIFIER>=RESERVED <UNITTYPE>"

HAS BEEN IMLEMENTED. <UNITTYPE> MAY BE:

PR, PP, UP, CR, CP, MT.

WHEN A UNIT DECLARATION OF THE ABOVE TYPE IS USED, THE SYSTEM WILL SEARCH THE UNIT TABLE FOR THE SPECIFIED TYPE OF UNIT. UPON FINDING A UNIT OF THE REQUIRED TYPE, IT WILL THEN CHECK IF THE UNIT HAS BEEN RESERVED AND THAT THE STATUS LINES INDICATE THAT THE UNIT IS READY. IF BOTH OF THESE CONDITIONS ARE MET, THIS UNIT WILL BE THE ONE USED BY THE PROGRAM. IF THE UNIT DOES NOT MEET THESE CONDITIONS, THE SEARCH CONTINUES. IF NO UNITS ARE FOUND, AN APPROPRIATE ERROR MESSAGE WILL OCCUR.

THE RESERVED WORDS "BCL", "BINARY", AND "TRANSLATE" MAY NOW BE USED WITH DISK OPERATIONS. IF THE MODIFIERS OF "BCL", OR "BINARY" ARE NOT USED IN AN I/O STATEMENT, THEN THE DEFAULT CHARACTER SIZE IS SET TO "EBCDIC".

DEFAULT PARITY FOR MAGNETIC TAPE UNITS IS SET TO ODD PARITY IF THE WORDS "EVEN" OR "ODD" HAVE NOT BEEN SPECIFIED.

D0091 DUMPANALY - NAMES OF GLOBAL ITEMS - 07-31-72

IF MCPINFO IS PRESENT AND IF INDICATED MCP CODEFILE IS RESIDENT, THEN THIS PATCH RETRIEVES THE GLOBAL DIRECTORY OF THAT MCP, SORTS THE DATA ALPHABETICALLY BY NAME AND ADDRESS, THEN USES THIS INFORMATION TO IDENTIFY RCW-S, SIRW-S, AND MEMORY AREAS THAT BELONG TO THE MCP.

A NEW OPTION HAS ALSO BEEN ADDED TO THE ANALYZER SUCH THAT THE RETRIEVAL AND SORTING OF THE SYMBOLIC NAMES CAN BE SUPPRESSED. THE SUPRESSION MAY BE DESIRABLE WHEN THE ANALYSIS OF A DUMP MUST TAKE PLACE IN A "SHAKY ENVIRONMENT". THE ANALYZER REQUIRES FAIRLY SUBSTANTIAL HARDWARE/SOFTWARE RESOURCES TO DO THE SYMBOLIC NAMES RETRIEVAL. IF THE SYSTEM IS INADEQUATE, THE USER CAN SPECIFY "NONAMES" AND THE ANALYZER WILL SKIP ALL ATTEMPTS TO RETRIEVE AND

USE THE SYMBOLIC NAMES.

D0092 MCP - FILE ATTRIBUTES - 07-13-72

STATE ATTRIBUTE

THE FILE ATTRIBUTE, STATE, IS NOW IMPLEMENTED. CONTRARY TO PREVIOUS DOCUMENTATION, THE STATE ATTRIBUTE IS READ ONLY, REAL VALUED, AND ACCESSABLE ONLY WHEN THE FILE IS OPEN. THE VALUE RETURNED IS A COPY OF THE SOFTWARE I/O RESULT DESCRIPTOR.

THE FIELDS OF THIS WORD ARE AS FOLLOWS:

[0:1] ATTENTION - THE ERROR FIELD IS NON-ZERO.

[16:16] ERROR FIELD AS FOLLOWS:

- [4:1] - DATA ERROR (SIZE ERROR FOR VARIABLE RECORDS)
- [7:1] - PARITY ERROR
- [9:1] - EOF OR END OF PAGE
- [13:1] - BREAK ON OUTPUT - DATACOM
- [15:1] - TIMELIMIT
- [16:1] - SECURITY VIOLATION

[24:8] QUALIFICATIONS
WHY EOF OCCURRED FOR DATACOM
IF A USER OPTION APPEARS FOR THE FIRST TIME IN AN ALGOL, XALGOL, OR ESPOL PROGRAM WHILE PROCESSING
VALUES:

1 FILE ASSIGNMENT DENIED
2 FILE ASSIGNMENT POSTPONED
3 ILLEGAL USE OF FILE

[47:20] ACTUAL DATA SIZE OF THE LOGICAL RECORD IN INTMODE UNITS OR WORDS AS CONTROLLED BY THE UNITS ATTRIBUTE.

D0093 MCP - OBJECT JOB OUTPUT AND FRSN - 07-13-72

THE "WRITE" DCWRITE AND THE OBJECT JOB OUTPUT MESSAGES WILL NOW INCLUDE THE FILE RELATIVE STATION NUMBER (FRSN) IN WORD [4].[47:24] OF THE MESSAGE.

WHEN AN MCS IS PARTICIPATING IN I/O FOR THE STATION OR DOES A "RECALL MESSAGE" DCWRITE, IT WILL NOW BE ABLE TO IDENTIFY THE FILE FROM WHICH THE MESSAGE ORIGINATED.

NEW DCWRITE TYPE

STATION BREAK (TYPE = 66)

REQUIRED:

1. MESSAGE PARAMETER.

2, MSG[0].[47:8] = 66.

3. MSG[0].[23:24] = FRSN.

EXAMPLE:

```
ALLOCATE (MSG, 6);
MSG[0] := FRSN & 66 [47:7:8];
RESULT := DCWRITE (MSG);
```

SEMANTICS:

THIS DCWRITE PROVIDES THE MCS WITH THE ABILITY TO INFORM AN OBJECT JOB THAT A BREAK ON OUTPUT OCCURRED AT A STATION. THE NEXT SUBSEQUENT ATTEMPT AT OUTPUT TO THE STATION FROM THAT FILE WILL RESULT IN BREAK ERROR ACTION.

THE MINIMUM MESSAGE SIZE FOR THE "STATION BREAK" DCWRITE IS SIX (6) WORDS.

D0094 XREFANALY - NEW DECLARATIONS & INTRINSICS - 05-05-72

NEW ESPOL DECLARATIONS "FILE" AND "DIRECT FILE" ARE NOW RECOGNIZED.

D0095 PACKDIR - DISK PACK DIRECTORY LISTING - 07-31-72

THIS PROGRAM LISTS AND MAPS "NATIVE MODE" DISK PACK DIRECTORIES IN A FORMAT SIMILAR TO THE EXISTING SYSTEM/LISTDIRECTORY.

ONLY ONE DISK PACK WILL BE LISTED FOR EACH EXECUTION OF SYSTEM/PACKDIR. THE PROGRAM IS EXECUTED BY PASSING THE DESIRED UNIT NUMBER, I.E.,

RUN SYSTEM/PACKDIR (<UNIT DESIGNATE>)

D0096 ESPOLINTRN - DOUBLE PREC. MATH INTRINSICS - 08-01-72

THIS CHANGE IMPLEMENTS DARCSIN, DARCCOS, ERF, DERFC, DLGAMMA, AND DGAMMA INTRINSICS, AND CORRECTS A PROBLEM IN ARCCOS.

D0097 ESPOLINTRN - TIMING AIDS - 08-01-72

THIS CHANGE ADDS INTRINSICS TO PRINTOUT TIMINGS FROM THE FORTRAN COMPILER.

D0098 NEW ALGOL DECLARATIONS - 05-08-72

ADDITIONAL ALGOL DECLARATIONS HAVE BEEN INCORPORATED. INCLUDED ARE:

STRING ARRAYS
DIRECT STRING ARRAYS
STRING VALUE ARRAYS
TRUTHSETS
TRANSLATETABLES

D0099 XREFANALY - GRAPHIC SPELLING - 06-03-72

THIS CHANGE ADDS GRAPHIC SPELLINGS FOR NEW ESPOL ELCLASSES, STRING ARRAYS, STRING VALUE ARRAYS, TRUTHSETS, AND TRANSLATETABLES.

D0100 RJE - DIRECTED BACKUP - 06-26-72

THIS PATCH ALLOWS BACKUP FILES TO BE DIRECTED TO AN RJE STATION BY CREATING A BACKUP FILE WITH THE PREFIX RJEBD/<STATION NAME>. THE <STATION NAME> MUST BE A MAXIMUM OF 17 CHARACTERS AND CAN INCLUDE SLASHES, ALTHOUGH THE SLASH WILL NOT INDICATE A DIRECTORY BUT WILL BE TREATED AS A SPECIAL CHARACTER WITHIN A SINGLE IDENTIFIER.

D0101 MCP - SOFTWARE TRANSLATION - 09-09-72

SOFTWARE TRANSLATION HAS BEEN ADDED TO THE LOGICAL I/O SUBSYSTEM TO SUPPLEMENT THE ALREADY EXISTING HARDWARE TRANSLATION CAPABILITY. FILES WILL BE GIVEN SOFTWARE TRANSLATION USING THE FOLLOWING ALGORITHM:

```
IF FILETYPE NEQ 4 AND FILETYPE NEQ 6 THEN
  IF NOT (DIRECT I/O OR BINARY I/O) THEN
    IF PREVIOUS TO II.3.0 THE PROGRAM WAS DS-ED
      WITH "INVALID TRANSLATION" THEN *
  DO SOFTWARE TRANSLATION;
  * SEE THE FOLLOWING TABLE.
```

BOTH CHARACTER AND WORD ORIENTED I/O (UNITS ATTRIBUTE) WILL BE TRANSLATED. THE FILE ATTRIBUTES BLOCKSIZE, MAXRECSIZE, ETC., WILL STILL BE DEFINED IN LOGICAL (INTERNAL TO THE PROGRAM) UNITS, AS BEFORE, WITH THE LOGICAL I/O SUBSYSTEM USING DIFFERENT LOCAL VALUES WHEN NECESSARY.

CONSIDER CREATING A DISK FILE THAT USES WORD ORIENTED SOFTWARE TRANSLATION:

```
FILE TRANS (KIND = DISK, FILETYPE = 0, INTMODE = BCL,
   EXTMODE = EBCDIC, MAXRECSIZE = 10);
```

TEN (10) WORD BCL RECORD CONTAIN 80 CHARACTERS OF VALID DATA. THE RECORDS IN TRANS THEN MUST CONTAIN THE 80 CHARACTERS OF DATA AND STILL BE STORED IN EBCDIC WORDS. HENCE, A 10 WORD WRITE TO TRANS WILL BE TRANSLATED INTO 14 WORDS ON DISK; 80 EBCDIC CHARACTERS AND FOUR EBCDIC BLANKS TO ROUND UP TO THE WORD BOUNDARY. NOTE: IF TRANS IS LOCKED ON DISK, TO ALL THE WORLD IT WILL LOOK AS IF IT

CONTAINS 14 WORD RECORDS, 84 EBCDIC CHARACTERS OF VALID DATA IN EACH RECORD.

CHARACTER ORIENTED I/O (UNITS = TRUE) DOES NOT HAVE THE PROBLEM OF RECORD SIZE EXPANSION AND/OR CONTRACTION.

DEFAULT SOFTWARE TRANSLATION

DEFINE INTMODE:

THE INTERNAL OR LOGICAL MODE OF THE RECORDS IN A FILE.

0 - WORD MODE (BINARY OR 48-BIT)
2 - HEX CHARACTERS (4-BIT)
3 - BCL CHARACTERS (6-BIT)
4 - EBCDIC CHARACTERS (8-BIT)
5 - ASCII CHARACTERS (8-BIT)

DEFINE EXTMODE:

THE EXTERNAL OR PHYSICAL (ACTUAL) MODE OF THE RECORDS IN A FILE. IF SET, IT MAY BE OVERRIDDEN BY THE PHYSICAL MODE OF A PERMANENT FILE OR UNIT TYPE. SAME VALUES AS INTMODE, PLUS

6 - BINARY (CARD FILES - 12 BITS PER COLUMN)

SOFTWARE TRANSLATION TAKES PLACE WHENEVER THE FOLLOWING ARE TRUE:

KIND	INTMODE	EXTMODE
DISK	HEX	ASCII
	BCL	EBCDIC, ASCII
	EBCDIC	BCL, ASCII
	ASCII	HEX, BCL, EBCDIC
CONSOLE	HEX,BCL,ASCII	ALL TO EBCDIC
REMOTE	HEX,BCL,ASCII	ALL TO EBCDIC
PAPER-READER & PUNCH	HEX	BCL, EBCIDC, ASCII
	BCL	HEX, EBCDIC, ASCII

KIND	INTMODE	EXTMODE
	EBCDIC	HEX
	ASCII	HEX, BCL, EBCDIC
LINE PRINTER	ASCII	ALL TO EBCDIC
CARD READER	BCL	EBCDIC *
	ASCII	BCL, EBCDIC *
OR OR PSEUDOREADER		
CARD PUNCH	BCL	EBCDIC *
	ASCII	BCL, EBCDIC *
MAGNETIC TAPE	HEX	ASCII
	BCL	ASCII
	EBCDIC	ASCII

* ONLY VALID EXTMODES FOR CARD READER/PUNCH OR PSEUDO READERS ARE BCL, EBCDIC, AND BINARY (WITH BINARY EXCLUDED FOR PSEUDO READERS).

D0102 PLI - PLI COMPILER - 08-15-72

THE II.3 SYSTEM RELEASE INCLUDES A PRELIMINARY FIELD TEST VERSION OF THE BURROUGHS B6700/B7700 PL/I COMPILER. THE FOLLOWING CHECKLIST ATTEMPTS TO SPECIFY WHICH FEATURES ARE IMPLEMENTED, PARTIALLY IMPLEMENTED, OR NOT IMPLEMENTED. TROUBLE REPORTS SHOULD BE SUBMITTED FOR ANY FEATURE MARKED "Y" (IMPLEMENTED) WHICH FAILS TO PERFORM PROPERLY. THIS LIST IS A PARTIAL LIST OF FEATURES. NOT ALL IMPLEMENTED FEATURES ARE INCLUDED AND SOME YET TO BE IMPLEMENTED FEATURES ARE MISSING.

THE PL/I LANGUAGE MANUAL WILL SUPPLY THE DETAILS REGARDING USE OF THE LANGUAGE AND USE OF THE COMPILER.

CHECKLIST OF PL/I

FIELD TEST RELEASE FEATURES

THE FEATURES LISTED HERE ARE MARKED "Y" IF FIRST RELEASE IMPLEMENTATION IS EXPECTED AND "N" IF IT IS NOT EXPECTED. THE STATUS "R" INDICATES THAT RESTRICTED PORTIONS OF THAT FEATURE WILL BE FINISHED.

THE USAGE ABBREVIATIONS ARE:

ATT	DATA ATTRIBUTE
BIF	BUILTIN FUNCTION
CDN	CONDITION
CTS	COMPILE TIME STATEMENT
FAT	FILE ATTRIBUTE
FMT	FORMAT ITEM
FOP	FILE OPTION
IOP	I-O OPTION
KYW	KEYWORD
OPT	OPTION
PC	PICTURE CHARACTER
STM	STATEMENT

FEATURE	USAGE	STATUS	COMMENT
.	PC	Y	
:	PC	Y	
$	PC	Y	
*	PC	Y	
BACK SLASH	PC	Y	
,	PC	Y	
-	PC	Y	
% ACTIVATE	CTS	Y	
% ASSIGNMENT	CTS	Y	
% DEACTIVATE	CTS	Y	
% DECLARE	CTS	Y	
% DO	CTS	N	
% GO TO	CTS	Y	
% IF	CTS	Y	
% INCLUDE	CTS	Y	
% NULL	CTS	Y	
A	FMT	Y	
A	PC	Y	
ABBREVIATIONS OF KEYWORDS		Y	
ABNORMAL BLOCK EXIT		Y	
ABS	BIF	Y	
ACOS	BIF	Y	SINGLE PRECISION, REAL
ACTIVATE	CTS	Y	
ADD	BIF	Y	
ADDR	BIF	Y	
ADJUSTABLE BOUNDS		Y	AREA SIZE, STRING LENGTHS ALSO
AGGREGATE ARGUMENTS		Y	
AGGREGATE ASSIGNMENT		Y	
AGGREGATE EXPRESSIONS		Y	
AGGREGATE		ARRAYS	
ALIGNED	ATT	Y	
ALL	BIF	Y	

FEATURE	USAGE	STATUS	COMMENT
ALLOCATE	STM	Y	
ALLOCATION	BIF	Y	
ANY	BIF	Y	
AREA	ATT	Y	
AREA	CDN	Y	
AREA ASSIGNMENT		N	
ARGUMENTS		Y	INCLUDING CREATION OF DUMMY ARGUMENTS
ARITHMETIC CONVERSIONS		Y	INCLUDING CONVERSION TO
ARITHMETIC COMPARISONS		Y	
ARITHMETIC DEFAULTS		Y	IMPLICIT DECLARATIONS
ARITHMETICS OPERATORS		Y	ALL
ARRAY	ATT	Y	ADJUSTABLE BOUNDS
ARRAY CROSS SECTIONS		Y	EXCEPT AS PARAMETERS
ASIN	BIF	Y	SINGLE PRECISION
ASSIGNMENT OF EVENT VARIABLES		N	
ASSIGNMENT OF TASK VARIABLES		N	
ASSIGNMENT OF AREA VARIABLES		N	
ASSIGNMENT TO PICTURE VARIABLES		Y	
ASTERISK		Y	PARAMETER + RETURNS BOUNDS AND LENGTHS
ASYNCHRONOUS OPERATIONS		N	
ATAN	BIF	Y	SINGLE PRECISION
ATAND	BIF	Y	
ATANH	BIF	Y	SINGLE PRECISION
AUTOMATIC	ATT	Y	
B	FMT	N	
B	PC	Y	
BACKWARDS	FAT	N	
BASE CONVERSION		Y	

FEATURE	USAGE	STATUS	COMMENT
BASED	ATT	Y	
BB	FMT	N	
BCOLUMN	FMT	N	
BEGIN	STM	Y	
BEGINVOLUME	CDN	N	
BINARY	ATT	Y	
BINARY	BIF	Y	
BIT	ATT	Y	
BIT	BIF	Y	
BIT COMPARISON-OPERATIONS		Y	
BIT STRING CONVERSION		Y	
BITSTREAM	FAT	N	
BLINESIZE	FOP	N	
BOOL	BIF	Y	
BOUNDS OF ARRAYS		Y	(CAN BE SPECIFIED IN DECLARATIONS ONLY
BP	FMT	N	
BUFFERED	FAT	Y	IGNORED
BUILTIN	ATT	Y	INCLUDING CONTEXTUAL DECLARATION
BX	FMT	N	
BY NAME		Y	
C	FMT	N	
CALL	STM	Y	EXCEPT TASKING
CEIL	BIF	Y	
CHAR	BIF	Y	
CHARACTER	ATT	Y	
CHARACTER CONVERSIONS		Y	TO BIT AND ARITHMETIC
CHARACTER OPERATIONS		Y	
CHARACTER PICTURE		R	
CHARACTER LENGTH		Y	SPECIFY IN DEC-

FEATURE	USAGE	STATUS	COMMENT
			LARATIONS ONLY
CHARACTER SET - 48		Y	
CHARACTER SET - 60		Y	EBCDIC ONLY
CHECK	CDN	N	
CLOSE	STM	Y	
CLOSURE, MULTIPLE		Y	
COLUMN	FMT	Y	
COLLATING SEQ			EBCDIC BINARY ORDER
, (COMMA)	PC	Y	
COMPLETION	BIF	N	
COMPOUND STATEMENTS		Y	IF AND ON
COMPUTATIONAL CONDITIONS		R	SOME NOT DETECTED
CONCATENATION OPERATION		Y	
CONDITIONS		R	SEE INDIVIDUAL CONDITIONS
CONDITION B.I.F.		R	
CONDITION	CDN	Y	
CONDITION PSEUDO VARIABLES		R	
CONJG	BIF	N	
CONNECTED	ATT	Y	"UNCONNECTED" NOT HANDLED
CONSTANTS		Y	ALL PLUS CONSTANT EXPRESSION EVALUATION
CONTEXTUAL DECLARATION		Y	
CONTROLLED	ATT	Y	
CONVERSIONS		Y	ALL EXCLUDING COMPLEX
CONVERSION	CDN	Y	
COPY	IOP	Y	
COS	BIF	Y	
COSD	BIF	Y	
COSH	BIF	Y	SINGLE PRECISION

FEATURE	USAGE	STATUS	COMMENT
COUNT	BIF	Y	
COMPLEX	ATT	N	
CR	PC	Y	
CREATION OF DUMMY ARGUMENTS		Y	
CROSS SECTIONS OF ARRAYS		R	EXCEPT AS ARGUMENTS
DATA	IOP	R	NO GET DATA
DATAFIELD	BIF	N	
DATE	BIF	Y	
DB	PC	Y	
DEACTIVATE	CTS	Y	
DECIMAL	ATT	Y	
DECIMAL	BIF	Y	
DECLARATIONS		Y	ALL FORMS
DECLARE	STM	Y	COMPLETE FACTORING
DEFAULT	STM	Y	COMPLETE USER CONTROLLED DEFAULTING
DELAY	STM	Y	
DELETE	STM	N	
DEFINED	ATT	R	
DESCRIPTOR	ATT	Y	
DESCRIPTORS	ATT	Y	
DIM	BIF	Y	
DIMENSION	ATT	Y	SPECIFY IN DECLARATIONS ONLY
DIRECT	FAT	N	
DISPLAY	STM	Y	EXCEPT EVENT OPTION
DIVIDE	BIF	Y	
DO	CTS	N	NO COMPILE TIME DO
DO	STM	Y	
DO-GROUP		Y	

FEATURE	USAGE	STATUS	COMMENT
DUMMY ARGUMENTS		Y	
E	FMT	Y	EXCEPT SCALE FACTOR
EDIT	IOP	R	SEE INDIVIDUAL FORMAT ITEMS
ELSE	KYW	Y	
EMPTY	BIF	N	
END	STM	Y	INCLUDING MULTIPLE
ENDFILE	CDN	Y	
ENDPAGE	CON	Y	
ENDVOLUME	CDN	N	
ENTRY VARIABLES	ATT	N	
ENTRY CONSTANTS	ATT	Y	EXCEPT ARRAYS CLOSURE
ENTRY	STM	Y	
ENTRY POINTS		Y	PRIMARY AND SECONDARY
ENVIRONMENT	FOP	Y	
ERF	BIF	Y	SINGLE PRECISION
ERFC	BIF	Y	SINGLE PRECISION
ERROR	CDN	R	SOMETIMES RAISED
EVENT	ATT	N	NO TASKING
EXCLUSIVE	FAT	N	
EXIT	STM	R	NO TASKING
EXP	BIF	Y	
EXPRESSIONS		R	ALL SCALAR ARITHMETIC AND STRING INCLUDING POINTER QUALIFICATIONS
EXTERNAL	ATT	R	NO BINDING
F	FMT	Y	EXCEPT SCALE FACTOR
FACTORING IN DCL-S		Y	ALL ATTRIBUTES

FEATURE	USAGE	STATUS	COMMENT
			INCLUDING LEVEL
FETCH	STM	XX	NOT RQD
FILE	ATT	R	SEE INDIVIDUAL ATTRIBUTES AND FEATURES
FILE	OPT	Y	SEE INDIVIDUAL ATTRIBUTES AND FEATURES
FILE OPEN-IMPLICIT		Y	
FINISH	CDN	Y	
FIXED	ATT	Y	
FIXED	BIT	Y	
FIXEDOVERFLOW	CDN	Y	
FLOAT	ATT	Y	
FLOAT	BIF	Y	
FLOOR	BIF	Y	
FORMAT	STM	Y	
FREE	STM	Y	
FROM	FOP	Y	WRITE STATEMENT ONLY
FUNCTIONS - BUILTIN		Y	SEE INDIVIDUAL BIF-S
FUNCTIONS - USER		Y	
G	PC	N	
GENERIC	ATT	N	
GET	STM	R	LIST AND SOME EDIT
GO TO	STM	Y	
GOTO	STM	Y	
H	PC	N	
HBOUND	BIF	Y	
HIGH	BIF	Y	
I	PC	N	
I THRU N RULE		Y	MAY BE OVERRIDDEN BY DEFAULT

FEATURE	USAGE	STATUS	COMMENT
			STATEMENT
IF	STM	Y	
IGNORE	OPT	Y	
IMAG	BIF	N	
IMPLICIT DECLARATIONS		Y	
IMPLICIT OPENING OF FILES		Y	
IN	OPT	Y	
INCLUDE	CTS	Y	
INCORPORATE		N	NEVER
INDEX	BIF	Y	EXCEPT COMPILE TIME
INITIAL	ATT	Y	COMPLETE DYNAMIC CASES
INITIAL CALL		Y	
INPUT	FAT	Y	
INTERLEAVED ARRAY		R	
INTERNAL	ATT	Y	
INTERRUPT HANDLING		N	NO TASKING
INTO	OPT	Y	
INVOCATION OF PROCEDURES		Y	(EXCEPT COMPILE TIME)
IRREDUCIBLE	ATT	Y	
ISUB DEFINING	ATT	Y	
ITERATION FACTORS		Y	
K	PC	N	
KEY	CDN	N	
KEY	OPT	N	
KEYED	ATT	N	
KEYFROM	OPT	N	
KEYTO	OPT	N	
KEYWORD ABBREVIATIONS		Y	
LABEL ASSIGNMENT		Y	
LABEL	ATT	Y	(LABEL CONSTANTS ONLY)

FEATURE	USAGE	STATUS	COMMENT
LABEL PARAMETERS		Y	
LBOUND	BIF	Y	
LENGTH	ATT	Y	LENGTHS AND SIZE CAN BE SPECIFIED IN DECLARATIONS ONLY
LENGTH	BIF	Y	
LEVEL NUMBERS	ATT	Y	
LIKE	ATT	Y	
LINE	FMT	N	
LINE	OPT	N	
LINENO	BIF	N	
LINESIZE	OPT	N	
LIST	IOP	Y	
LIST-DIRECT I-O		R	EXCEPT DATA DIRECTED
LOCATE	STM	Y	
LOCATER ARGUMENTS		Y	NO TYPE COERCION
LOCATER QUALIFIER		Y	
LOCATER ASSIGNMENT		Y	
LOCKING A RECORD		N	
LOG	BIF	Y	
LOG10	BIF	Y	
LOG2	BIF	Y	
LOW	BIF	Y	
M	PC	N	
MAJOR STRUCTURE		Y	
MAJOR TASK		N	NO TASKING
MAX	BIF	Y	
MEMBER	ATT	Y	
MIN	BIF	Y	
MINOR STRUCTURES		Y	
MOD	BIF	Y	
MODE CONVERSION		Y	

FEATURE	USAGE	STATUS	COMMENT
MULTIPLE CLOSURE		Y	
MULTITASKING		N	
NAME	CDN	N	
9 (NINE)	PC	Y	
NO...	CDN	Y	
NOLOCK	OPT	N	
NORESCAN	OPT	Y	COMPILE TIME OPTION
NULL	BIF	Y	
NULL	STM	Y	
NUMERIC PICTUES		R	SEE INDIVIDUAL PICTURE CHARACTERS
OFFSET	ATT	Y	PARAMETER MUST MATCH ARGUMENT
OFFSET	BIF	Y	
ON	STM	Y	
ONCHAR	BIF	N	
ONCODE	BIF	N	
ONCOUNT	BIF	N	
1 (ONE)	PC	N	
ONFILE	BIF	N	
ONIDENT	BIF	N	
ONKEY	BIF	N	
ONLOC	BIF	N	
ONSOURCE	BIF	N	
ON-UNITS		Y	
OPEN	STM	Y	
OPTIONS	ATT	Y	
OPTIONS	OPT	Y	
OPERATORS		Y	EXCEPT ARRAYS, STRUCTURES, AND COMPLEX
ORDER	OPT	Y	
OUTPUT	FAT	Y	

FEATURE	USAGE	STATUS	COMMENT
OVERFLOW	CDN	Y	
OVERLAY DEFINING		Y	RESTRICTED OVERLAY DEFINING TO BE IMPLEMENTED
OVERPUNCHED PICTURE SIGNS		R	
P	PC	N	
P	FMT	Y	NEITHER NUMERIC OR CHARACTER
PAGE	FMT	Y	
PAGE	OPT	Y	
PAGESIZE	OPT	Y	
PARAMETERS		Y	
PARAMETER	ATT	Y	
PENDING	CDN	N	
% (PERCENT SIGN)	CTS	R	SEE INDIVIDUAL STATEMENTS
PICTURE	ATT	Y	SEE INDIVIDUAL PICTURE CHARACTERS
POINTER	ATT	Y	
POINTER	BIF	Y	
POINTER PARAMETERS		Y	
POLY	BIF	N	
POSITION	ATT	Y	LIMITED POSITION DEFINING WILL BE IMPLEMENTED
PRECISION	ATT	Y	
PRECISION	BIF	Y	
PREFIX CONDITIONS		Y	
PRINT	FAT	Y	
PRIORITY	BIF	N	NO TASKING
PROCEDURE	CTS	N	
PROCEDURE	STM	Y	
PROD	BIF	N	

NEW FEATURES AND DOCUMENTATION CHANGES

FEATURE	USAGE	STATUS	COMMENT
PROGRAMMER NAMED CONDITIONS	CDN	Y	
PSEUDO VARIABLES		Y	
PUT	STM	Y	DATA, LIST, & SOME EDIT
QUALIFIED NAMES		Y	
QUALIFICATION, LOCATER		Y	
R	FMT	Y	
R	PC	N	
RANGE	KYW	Y	IN DEFAULT STATEMENT
READ	STM	Y	EXCEPT KEYS & EVENTS
REAL	ATT	Y	
REAL	BIF	Y	
RECORD	FAT	Y	
RECORD	CDN	N	
RECORD	I	O	
RECURSIVE	ATT	Y	ONLY
REDUCIBLE	ATT	Y	IGNORED
REFER	OPT	Y	
RELEASE	STM	N	NOT REQUIRED
REMOTE FORMAT ITEM		Y	
REORDER		Y	IGNORED
REPEAT	BIF	Y	
REPEAT	SOP	Y	
REPETITION FACTOR		Y	
REPLACEMENT OF COMPILE TIME VARIABLES		Y	
REPLY	OPT	Y	
RESCAN	OPT	Y	COMPILE TIME OPTION
RETURN	STM	Y	
RETURNS	ATT	Y	BOTH ENTRY AND PROCEDURE

FEATURE	USAGE	STATUS	COMMENT
			STATEMENTS
REVERT	STM	Y	
IMPLICIT REVERTING OF CONDITIONS		Y	
REWRITE	STM	N	
ROUND	BIF	Y	
ROW MAJOR ORDER		Y	
S	PC	Y	EXCEPT DRIFTING
SCALAR ASSIGNMENT		Y	
SCALE CONVERSION		Y	
SCALE FACTOR	ATT	Y	
SCOPE ATTRIBUTES		Y	
SECONDARY	ATT	Y	IGNORED
SECONDARY ENTRY POINTS		Y	
SELECTION, GENERIC		N	
SEQUENTIAL	FAT	Y	
SET	OPT	Y	
7 (SEVEN)	PC	N	
SIGN	BIF	Y	
SIGNAL	STM	Y	
SIMPLE DEFINING	STM	R	LIMITED DEFINE CAPABILITY
SIN	BIF	Y	
SIND	BIF	Y	
SINH	BIF	Y	SINGLE PRECISION
6 (SIX)	PC	N	
60-CHARACTER SET		Y	
SIZE	ATT	Y	IN DECLARATION STATEMENTS ONLY
SIZE	CDN	N	NOT CHECKED
SKIP	FMT	Y	
SKIP	OPT	Y	
(SLASH)	PC	N	
SNAP	OPT	Y	B6700 STACK DUMP

FEATURE	USAGE	STATUS	COMMENT
SQRT	BIF	Y	
SYSTEM DEFAULT RULES		Y	
STATEMENTS		Y	SEE INDIVIDUAL STATEMENTS
STATIC	ATT	Y	
STATUS	BIF	N	NO TASKING
STERLING PICTURES		N	
STOP		R	NO TASKING
STORAGE ALLOCATION		Y	
STORAGE CLASSES		R	
STREAM	FAT	Y	
STREAM I-O STATEMENTS		R	
STRING ASSIGNMENT		Y	
STRING PARAMETERS		Y	
STRING OVERLAY DEFINING		Y	
STRING	OPT	Y	
STRINGRANGE	CDN	Y	
STRINGSIZE	CDN	Y	
- (STROKE)		Y	
STRUCTURES		Y	
STRUCTURE	ATT	Y	
SUBROUTINES		Y	
SUBSCRIPTED NAMES		Y	
SUBSCRIPTRANGE	CDN	Y	
SUBSTR	BIF	Y	AND PSEUDO VARIABLE
SUM	BIF	Y	
SYSIN		Y	DEFAULT INPUT FILE
SYSPRINT		Y	DEFAULT OUTPUT FILE
SYSTEM	OPT	Y	DEFAULT STATEMENT OPTION
T	PC	N	
TAN	BIF	Y	

FEATURE	USAGE	STATUS	COMMENT
TAND	BIF	Y	
TANH	BIF	Y	SINGLE PRECISION
TASKING		N	
TASK	ATT	N	
			STRINGS
TASK	OPT	N	
THEN	KYW	Y	
3 (THREE)	PC	N	
TIME	BIF	Y	
TITLE	FOP	Y	
TO	KYW	Y	
TRANSIENT	FAT	N	
TRANSLATE	BIF	Y	
TRANSMIT	CDN	Y	
TRUNC	BIF	Y	
2 (TWO)	PC	N	
UNALIGNED	ATT	Y	
UNBUFFERED	FAT	Y	IGNORED
UNDEFINEDFILE	CDN	N	
UNDERFLOW	CDN	Y	
UNLOCK	FOP	N	
UNLOCK	STM	N	
UNSPEC	BIF	Y	
UPDATE	FAT	N	
UPDATE	STM	N	
V	PC	Y	
VARIABLE	ATT	R	(NO ENTRY OR LABEL VARIABLES)
VERIFY	BIF	Y	
VARYING	ATT	Y	
VOLUME	OPT	N	
WAIT	STM	N	NO TASKING
WHEN	KYW	N	PARAMETER IN SECONDS
WHILE	KYW	Y	

FEATURE	USAGE	STATUS	COMMENT
WRITE	STM	R	VERY LIMITED RECORD I-O
X	FMT	Y	
X	PC	Y	
Y	PC	N	
Z	PC	Y	
ZERODIVIDE	CDN	Y	

D0103 RJE - BACKUP RESTART - 05-25-72

TO PREVENT ENDLESS RETRIES OF RJE AUTOPRINT WHEN SOME REPEATABLE ERROR OCCURRED, RJE WOULD NOT PREVIOUSLY RESPOND TO ANY AUTOPRINT REQUEST IF PRINTER/BACKUP HAD BEEN DS-ED. THIS CHANGE ALLOWS THE PRINTING TO BE RESUMED AT THE TERMINAL BY ENTERING A PB MESSAGE WITH A <EMPTY> <MIX ID>.

D0104 RJE - CARD FILES - 08-04-72

WHEN THE SAME CARD FILE DECLARATION IS USED FOR MORE THAN ONE PHYSICAL CARD FILE (I.E., MORE THAN ONE DATA DECK PER JOB WITH ONLY ONE FILE DECLARATION), THE FIRST DECK WILL BE REUSED WHEN ATTEMPTING TO READ THE SECOND AND/OR FOLLOWING DECKS. TO AVOID THE PROBLEM, THE PROGRAMMER SHOULD HAVE A SEPARATE DECLARATION FOR EACH PHYSICAL CARD FILE.

ALSO, IF THE KIND FILE ATTRIBUTE IS SET VIA A FILE CARD TO BE A CARD ERADER, A NO FILE SITUATION WILL RESULT FOR THAT CARD FILE. FOR EXAMPLE, THE CARD

<I> FILE CARD (KIND = READER)

SHOULD NOT BE USED IN A DECK SUBMITTED THROUGH RJE.

D0105 RJE - DP INPUT MESSAGE - 06-26-72

RJE WILL NO LONGER GO TO EOJ WHEN A DP MESSAGE IS RECOGNIZED. A

PROGRAM DUMP WILL BE GIVEN, THE PRESENT ACTIVE RJELINKED FILE WILL BE BE PRINTED IN THE DEBUG LISTING, AND RJE WILL CONTINUE.

D0106 RJE - TI RSC MESSAGE - 07-25-72

THE <MIX NUMBER> TI MESSAGE HAS BEEN MODIFIED TO INCLUDE THE I/O TIME FOR THE JOB IN QUESTION.

D0107 RJE - LOGON AND LOGOFF TIMES - 07-25-72

THIS PATCH CHANGES THE LOGON AND LOGOFF REMOVE SUPERVISORY CONSOLE MESSAGES TO INCLUDE THE STATION NAME, TIME, AND DATE.

D0108 RJE - SCHEDULED MESSAGES - 06-20-72

IF A JOB INITIATED THROUGH RJE DOES NOT GO TO BOJ SHORTLY AFTER INITIATION, A SCHD MESSAGE WILL NOW APPEAR AT THE RSC OF THE INITIATING TERMINAL.

D0109 FORTRAN - TIMING & DEBUGGING INFORMATION - 09-05-72

SEVERAL EXTENSIONS TO THE FORTRAN LANGUAGE HAVE BEEN MADE TO SERVE AS DEBUGGING AIDS. ALL DEBUGGING STATEMENTS CONTAIN THE CHARACTERS "DEBUG" IN COLUMNS 1-5 AND CONFORM TO THE USUAL FORTRAN STATEMENT FORMULATION RULES.

AN OPTION HAS BEEN PROVIDED TO DELETE ALL DEBUG STATEMENTS. THUS, DEBUG STATEMENTS CAN BECOME A PERMANENT PART OF THE SYMBOLIC PROGRAM, EXCLUDED WHILE THE PROGRAM IS IN A PRODUCTION FORM, AND INCLUDED WHEN THE PROGRAM IS BEING CHECKED OUT.

MONITOR

THE MONITOR STATEMENT IS USED WITHIN A SUBPROGRAM TO OBSERVE THE CHANGES IN THE SPECIFIC LIST OF ARRAY ELEMENTS AND VARIABLES. (SEE D0009 FOR DETAILS.)

DUMP

DUMP PROVIDES THE ABILITY TO TAKE SNAPSHOTS OF A LIST OF VARIABLES AND ARRAYS WHEN THE CONTROL OF THE PROGRAM PASSES TO A LABEL. (SEE D0010 FOR DETAILS.)

PROGRAM DUMP

PROGRAM DUMP CALLS ON THE MCP PROCEDURE THAT DUMPS THE CONTENTS OF THE PROGRAM-S STACK. OPTIONS ARE AVAILABLE ON THE OPTION CONTROL CARD (SEE MCP DOCUMENT) TO SELECTIVELY DUMP ARRAYS, CODE, AND/OR FILES. (SEE D0033 FOR ADDITIONAL INFORMATION.)

STATISTICS

THE STATISTICS OPTION ALLOWS THE PROGRAMMER TO SEE A FREQUENCY/ TIMING STUDY OF THE PROGRAM-S EXECUTION.

TWO OPTIONS ARE AVAILABLE: SUBPROGRAM TIMES AND COUNTS ONLY, OR TIMES AND COUNTS FOR EACH LOGICAL SEGMENT OR BLOCK IN EACH SUBPROGRAM. ANY TIME THE PROGRAM EXECUTES A "STOP" STATEMENT, A "DUMP STATISTICS" STATEMENT, OR THE PROGRAM-S EXCEPTIONEVENT (SEE MCP DOCUMENT: <MIXID> HI) IS CAUSED, THE STATISTICS ARE PRINTED ON THE FILE INDICATED IN THE STATEMENT. THE SUBPROGRAM AND BLOCK (IF APPLICABLE) THAT WAS BEING EXECUTED AT THE TIME OF THE PRINTOUT IS FLAGGED WITH AN ASTERISK TO INDICATE THAT THE TIMES FOR THIS SUBPROGRAM AND BLOCK MAY BE A BIT OFF SINCE THE TIME MEASURED IS ELAPSED TIME.

WHEN THE BLOCKS OPTION (I.E., TIME ALL BLOCKS) IS SPECIFIED, PROGRAM EXECUTION TIME INCREASES SUBSTANTIALLY (30 TO 50 PER CENT) BUT IMPORTANT, HEAVILY EXECUTED BLOCKS CAN BE LOCATED AND SOMETIMES IMPROVED.

STATISTICS ONCE OUT STAY IN EFFECT UNTIL RESET, AND THUS IS NOT LIKE OTHER FORTRAN STATEMENTS WHICH ARE ALWAYS LOCAL TO A SUBPROGRAM.

DUMP STATISTICS

THIS STATEMENT CAUSES THE ACCUMULATED TIMING/FREQUENCY STATISTICS

TO BE PRINTED.

SYNTAX:

A) COLUMNS 1-5 MUST CONTAIN DEBUG

B) CONTINUATION IS ALLOWED IN THE USUAL MANNER. ALL DEBUG STATEMENTS (EXCEPT IN COLUMNS 1-5) CONFORM TO STANDARD FORTRAN REFERENCE FORMAT.

```
MONITOR (FILE) MLIST

DUMP (FILE) CONDITION/DLIST

STATISTICS   FILE        <EMPTY>
             NO          BLOCKS
             DIAGNOSTICS

DUMP STATISTICS

PROGRAM DUMP
```

C) FILE MUST BE AN UNSIGNED INTEGER.

D) CONDITION, MLIST, DLIST AS DESCRIBED IN THE MONITOR AND DUMP OPTION NOTES.

E) "NO" MEANS TO RESET STATISTICS.

F) "DIAGNOSTICS" MEANS THAT THE SYSTEM DIAGNOSTIC FILE IS TO BE USED FOR THE STATISTICS.

G) BLOCKS SELECTS STATISTICS FOR ALL BLOCKS IN THE FORTRAN PROGRAM, <EMPTY> SELECTS STATISTICS FOR THE SUBPROGRAMS ONLY.

D0110 FORTRAN - IDENTIFIERS IN FORTRAN - 08-22-72

THE FORTRAN COMPILER DOES NOT ALLOW IDENTIFIERS LONGER THAN SIX CHARACTERS. ALL REFERENCES TO IDENTIFIERS LONGER THAN SIX CHARACTERS IN THE FORTTRAN DOCUMENT ON PAGE 4-1 AND IN THE EXAMPLES ON PAGES 4-2 AND 4-4 SHOULD BE DELETED.

THE SENTENCE ON PAGE 4-1 SHOULD BE REPLACED BY:

IDENTIFIERS LONGER THAN SIX CHARACTERS ARE NOT ALLOWED, AND THE ACTION OF THE COMPILER IS UNDEFINED.

D0111 ALGOL - COMPILE AND GO - 08-25-72

IT WILL NOT BE A SYNTAX ERROR IF A COMPILE AND GO IS ATTEMPTED ON A PROCEDURE FOR WHICH A COMPILE AND GO IS ILLEGAL. INSTEAD, NO ATTEMPT AT EXECUTION WILL BE MADE AND THE CODE FILE WILL BE LOCKED. A WARNING "EXECUTION IGNORED, CODE FILE LOCKED", WILL BE PRINTED ON THE LISTING.

PROCEDURES OR SUBPROGRAMS FOR WHICH A COMPILE AND GO IS ILLEGAL INCLUDE:

1) A PROCEDURE WITH PARAMETERS;

2) A PROCEDURE WHICH HAS GLOBAL DECLARATIONS;

3) A PROCEDURE FOR WHICH THE $ CARD OPTION "INTRINSICS" IS SET;

4) A PROCEDURE COMPILED AT LEVEL FOUR OR HIGHER;

5) ANY FORTRAN SUBROUTINE OR FUNCTIONS COMPILED SEPARATELY.

D0112 BINDER - COBOL INTRINSICS - 08-01-72

THIS SYSTEM NOTE HAS BEEN CHANGED TO P613.

D0113 REDUCING CODE FILE RQMTS-SEP - 08-25-72

BY SETTING THE FORTRAN COMPILER-S CODE FILE "AREASIZE" TO A NUMBER SMALLER THAN 180 (THE DEFAULT), SEPARATE COMPILES CAN OFTEN BE MADE TO USE SUBSTANTIALLY LESS DISK. FOR EXAMPLE, BY SETTING TO AREASIZE TO ONE:

<I> FORTRAN FILE CODE (AREASIZE = 1)

THE COMPILER IS DIRECTED TO SET THE CODE FILE-S ROW SIZE TO THE

MINIMUM (USUALLY 45). THIS IS USUALLY ENOUGH SPACE TO ACCOMMODATE MOST REASONABLE SUBPROGRAMS WITHIN A ROW. THUS, THE SUBPROGRAM OCCUPIES 45 RATHER THAN 180 SEGMENTS, YIELDING A 75 PER CENT REDUCTION IN DISK SPACE REQUIREMENTS.

BECAUSE OF BINDING RESTRICTIONS, HOWEVER, AUTOMATIC SUBPROGRAM SEGMENTATION IS NOT ALLOWED DURING SEPARATE COMPILES. IF THE COMPILER DETECTS THIS PROBLEM, IT WILL EMIT A SYNTAX ERROR FOR THE PARTICULAR SUBROUTINES THAT EXCEEDED THE ROW SIZE. ALL THAT IS NECESSARY IS THAT THE PARTICULAR SUBROUTINE BE RECOMPILED AT THE USUAL AREASIZE SPECIFICATION. BINDING WILL THEN PROCEED AS USUAL.

EXAMPLE:

```
<I> COMPILE MASTER/TEST FORTRAN LIBRARY
<I> FORTRAN FILE TAPE (TITLE = SOURCE/TEST/FORTRAN),
   CODE (AREASIZE = 1)

<I> DATA
$ MERGE SEPARATE LONG AUTOBIND
<I> END.
```

D0114 "RESERVE" & "RETURN" FUNCTION - 09-05-72

THIS IMPLEMENTATION PROVIDES THE FACILITIES OF REMOVING A SPECIFIED AREA OF THE HEAD-PER-TRACK DISK SUBSYSTEM FROM THE SYSTEM, I.E., "RESERVE", AND THE "RETURNING" OF THE RESERVED AREA TO THE SYSTEM-S AVAILABLE DISK POOL.

RESERVE:

THIS FACILITY IS INVOKED THROUGH THE KEYBOARD MESSAGE OF:

```
   RES <RESERVE SYNTAX>
```

WHERE <RESERVE SYNTAX> ::=

```
   <DISK UNIT DESIGNATE> <AREA SPECIFICATION>
      <DATA ERROR DISPOSITION> <ETX>
```

```
<DISK UNIT DESIGNATE> ::= DK <HEAD-PER-TRACK UNIT NUMBER>

<AREA SPECIFICATION> ::= <EMPTY>/
   <STORAGE UNIT RANGE> <AREA DISPOSITION>/
   <SWITCH RANGE> <AREA DISPOSITION>/
   <SEGMENT DESIGNATE>

<AREA DISPOSITION> ::=
                        AS BADDISK/
                        <EMPTY>

<STORAGE UNIT RANGE> ::= <SU DESIGNATE>/
                        <SU DESIGNATE> <STORAGE UNIT RANGE>

<SU DESIGNATE> ::= SU <NUMBER LIST>/
                        SU <NUMBER> FOR <NUMBER>
                        SU <NUMBER> THRU <NUMBER>/
                        SU <NUMBER> THRU SU <NUMBER>

<SWITCH RANGE> ::= <SWITCH DESIGNATE>/
                        <SWITCH DESIGNATE> <SWITCH RANGE>

<SWITCH DESIGNATE> ::= SWITCH <NUMBER LIST>/
                        SWITCH <NUMBER> FOR <NUMBER>
                        SWITCH <NUMBER> THRU <NUMBER>/
                        SWITCH <NUMBER> THRU SWITCH <NUMBER>

<SEGMENT DESIGNATE> ::= SEGMENT <NUMBER>/
                        SEGMENT <NUMBER> FOR <NUMBER>
                        SEGMENT <NUMBER> THRU <NUMBER>
                        SEGMENT <NUMBER> THRU SEGMENT <NUMBER>

<NUMBER LIST> ::= <NUMBER>/
                        <NUMBER> <NUMBER LIST>

<DATA ERROR DISPOSITION> ::= <EMPTY>/COPY ERRORS

<HEAD-PER-TRACK UNIT NUMBER>::= <NUMBER>

<SEPARATOR> ::= <SPACE>/,
```

EXAMPLES:

1. RES DK 5 SU 1 THRU SU 4
2. RES DK80 SWITCH 0, AS BADDISK
3. RES DK 2 SEGMENT 25478, AS BADDISK COPY ERRORS
4. RES DK 9 SU 0, SW 4, 5, 7, SW 10 THRU 12
5. RES DK 9 SEG 254777 THRU 315428 AS BADDISK

SEMANTICS:

THIS FUNCTION ALLOWS SPECIFYING AN AREA OF A HEAD-PER-TRACK DISK UNIT THAT IS TO BE PLACED IN THE IADDISK POOL OR TO BE MARKED AS A "BADDISK" FILE.

ONLY NON-IAD DISK AREAS MAY BE RESERVED.

<DISK UNIT DESIGNATE> SPECIFIES A UNIT DESIGNATE WITHIN THE HEAD-PER-TRACK DISK SYSTEM.

<AREA SPECIFICATION> SPECIFIES THE DISK AREA WITHIN THE SPECIFIED <DISK UNIT DESIGNATE> THAT IS TO BE "RESERVED". STORAGE UNIT (SU) AND SWITCH NUMBERS (SWITCH) ARE ASSUMED TO ORIGINATE AT ZERO FOR MODEL II CONTROLS (I.E., 2B DISK FILE) AND ONE FOR MODEL I CONTROLS (I.E., 1A OR 1C DISK FILE). IF <SEGMENT DESIGNATE> IS SPECIFIED, THEN ONLY ONE SEGMENT RANGE IS ALLOWED AND AN <AREA DISPOSITION> OF "AS BADDISK" IS ASSUMED.

UNLESS "AS BADDISK" OR <SEGMENT DESIGNATE> IS SPECIFIED, THE RESERVED AREA WILL BE ADDED TO THE IADDISK. IF IADDISK IS SPECIFIED THE RESERVED AREA WILL BE ADDED TO THE IADDISK SPACE AND WILL REMAIN AS SUCH UNTIL REMOVED BY A "RETURN" KEYBOARD MESSAGE OR A COLD START. IF "AS BADDISK" IS SPECIFIED, ONE OR MORE FILES WILL BE CREATED OF ONE ROW EACH POINTING WITHIN OR TO THE SPECIFIED AREA. ONE FILE WILL BE CREATED IF ALL OF THE SPECIFIED LOCK OUT SWITCH AREAS ARE "READY" AND ARE NOT "WRITE LOCK-OUT". IF ANY LOCK-OUT SWITCH AREA WITHIN THE SPECIFIED RESERVE RANGE IS NOT READY OR LOCKED OUT THEN A SEPARATE BADDISK FILE WILL BE CREATED POINTING TO EACH OF THE CONSECUTIVE AREAS OF THIS TYPE. IF A BADDISK FILE CREATED UNDER THESE REQUIREMENTS IS LATER REMOVED, THE AREAS THAT WERE ORIGINALLY FOUND TO BE NOT READY OR WRITE LOCK-OUT WILL NOT BE

RETURNED TO THE AVAILABLE DISK POOL. IN SUMMARY, SEVERAL SINGLE ROW BADDISK AREAS MAY BE CREATED IF THE SPECIFIED RESERVE AREA REFERENCES LOCK-OUT SWITCH AREAS THAT ARE NOT ALL "READY".

<DATA ERROR DISPOSITION> ALLOWS SPECIFYING AN UNCONDITIONAL TRANSFER OF NON-DIRECTORY AREAS EVEN THOUGH A DISK READ ERROR MAY OCCUR DURING THE COPY. IF ANY DISK ERROR OCCURS, A SYSTEM LOG ENTRY CONTAINING THE FILE NAME AND RECORD NUMBER IS CREATED. ALSO THE SYSTEM/MAINTLOG WILL CONTAIN AN ENTRY FOR EACH I/O BLOCK THAT CONTAINED AN ERROR. STANDARD DISK RETRY ACTION IS ENVOKED BEFORE NOTIFYING THE OPERATOR OF THE COPY PROBLEM. IF MORE THAN ONE ERROR OCCURS DURING THE COPY OF A ROW, THE OPERATOR AND LOG WILL REFLECT ONLY THE FIRST ONE OF THESE ERRORS. IF A DISK READ ERROR OCCURS AND THE COPY ERRORS OPTION HAS NOT BEEN SPECIFIED, THE RESERVE REQUEST WILL BE TERMINATED AFTER COMPLETION OF THE ABOVE NOTIFICATION PROCEDURES. OTHERWISE, THE RESERVE WILL BE ALLOWED TO CONTINUE. ANY ERROR OCCURRING ON A DISK WRITE OPERATION WILL AUTOMATICALLY ENVOKE THE GENERATION OF A BADDISK FILE WHOSE ROW SIZE WILL BE THE SAME AS THE FILE BEING COPIED AT THE TIME OF THE ERROR. THIS FILE IS CREATED IN SUCH A WAY THAT IF IT IS REMOVED, THE DISK SPACE REFERENCED WILL NOT BE RETURNED TO THE AVAILABLE DISK TABLES UNTIL A HALT/LOAD IS PERFORMED. IF A COPY ERROR OCCURS DURING THE MOVEMENT OF A ROW WITHIN A CODE FILE THE FOLLOWING ACTION WILL OCCUR.

1. IF "COPY ERRORS" AND OPTIONS HAVE NOT BEEN SPECIFIED, THE RESERVE REQUEST IS TERMINATED.

2. THE AREA IN QUESTION IS RESERVED, AND THE FILE WHICH CONTAINS THE ERROR IS REMOVED, VIA THIS ACTION, A "BADDISK" FILE WILL BE CREATED ENCOMPASSING THE AREA CONTAINING THE "READ" PROBLEM.

A RESERVE FUNCTION REQUIRES VARIED AMOUNTS OF USERS DISK SPACE AND IS, THEREFORE, SUBJECT TO CAUSING "NO USER DISK" MESSAGES. IF A "DS" RESPONSE IS SUPPLIED FOR ANY ONE OF THESE, THE "RESERVE" REQUEST WILL BE TERMINATED.

ANY TIME A "RESERVE" REQUEST IS TERMINATED, ALL SPACE COLLECTED UP

UNTIL THE TIME OF THE TERMINATION WILL BE RETURNED TO THE SYSTEM-S AVAILABLE DISK POOL.

IF A "BADDISK" FILE IS ENCOUNTERED WITHIN THE AREA SPECIFIED TO BE "RESERVED", THE AREA POINTED AT BY THE FILE IS "RESERVED" AND THE ROW ADDRESS WITHIN THE "BADDISK" FILE HEADER IS SET TO ZERO (THEREBY INHIBITING SCR ACCESS TO THIS AREA). THE AREA SPECIFIED BY THE FILE WILL BE COPIED TO ITSELF. IF AN ERROR OCCURS DURING THIS COPY, THE ACTION SPECIFIED UNDER <DATA ERROR DISPOSITION> WILL BE ENVOKED.

A FILE "RESERVE/EUNNN" IS CREATED EACH TIME A RESERVE IS REQUESTED. THIS FILE CONTAINS THE FILE NAMES FOR ANY NON-DIRECTORY AREA FOUND WITHIN THE AREA SPECIFIED TO BE RESERVED. THE FILE IS REMOVED WHEN THE RESERVE REQUEST TERMINATES.

A "RESERVE" WILL NOT BE PERFORMED ON THE FOLLOWING AREAS OF DISK AND WILL RESULT IN THE TERMINATION OF THE REQUEST.

1. THE DISK SPACE ALLOCATED FOR THE HALT/LOAD MCP.
2. BACK-UP EU MCP AREAS ASSIGNED FOR RECONSTRUCTION.
3. IADDISK AREAS.

THE "RESERVE" REQUEST WILL WAIT FOR ALL TEMPORARY FILES CONTAINING SPACE IN THE AREA TO BE RESERVED TO BE CLOSED.

A "RESERVE" REQUEST MAY CAUSE "DIRECTORY COPIER" TO BE INITIATED.

WHEN DATA AREAS ARE MOVED, THEY ARE MOVED ACCORDING TO THE CLASS DISK SPECIFICATION. THAT IS THE MOVEMENT OF CLASS 2 DATA (FOR EXAMPLE) CAN ONLY BE MOVED TO ANOTHER AREA DESIGNATED AS CLASS TYPE 2. IF A CLASSED AREA OF THE PROPER TYPE AND SIZE IS NOT FOUND, THEN A NO USERS DISK MESSAGE WILL BE DISPLAYED.

WHEN DATA HEADERS ARE MOVED, THE SYSTEM WILL REQUIRE "EXCLUSIVE" USE OF THE RESPECTIVE FILE. THIS IMPLIES THE FILE MAY NOT BE OPEN AND CANNOT BE OPENED DURING THE TIME THAT THE COPY ACTION IS TAKING PLACE. THIS REQUIREMENT HAS IMPLEMENTED A NEW RSVP MESSAGE OF "WAITING ON <FILE NAME>". THE ONLY RESPONSE TO THIS MESSAGE IS "DS". IF THIS IS ENTERED, THE RESERVE REQUEST IS TERMINATED.

OTHERWISE "RESERVE" WILL WAIT UNTIL IT CAN OBTAIN EXCLUSIVE USE OF THE APPROPRIATE FILE. THE ONLY NON-DIRECTORY DATA MOVED BY RESERVE WILL BE THAT OF THE ROW(S) OF ANY FILE THAT IS FOUND TO CONTAIN DISK ADDRESSES WITHIN THE AREA TO BE RESERVED. HOWEVER, TO ACCOMPLISH THIS, THE RESERVE FUNCTION MUST HAVE EXCLUSIVE USE OF THE ENTIRE FILE.

IF A "RESERVE" REQUEST ENCOMPASSES THE AREA THAT CONTAINS THE "SYSTEM/LOG" OR "SYSTEM/MAINTLOG", THE MESSAGE "WAITING ON <FILE NAME>" WILL BE PRODUCED. IF THE REQUEST IS TO BE ALLOWED TO CONTINUED, THE KEYBOARD MESSAGE "LR" MUST BE ENTERED.

IF EITHER DATA COMMUNICATIONS FILE OF "DC/CODE" OR "SYSTEM/NIF" IS WITHIN THE AREA TO BE RESERVED, THE FOLLOWING SPECIAL ACTION WILL OCCUR. AN "EXCLUSIVE" OPEN ACTION IS NOT REQUESTED, INSTEAD AN INTERNAL EVENT IS PROCURED DURING THE MOVEMENT OF THE ROWS WITHIN THESE FILES. ON A DISK ERROR THE FOLLOWING WILL OCCUR:

1. IF THE DATA COMMUNICATIONS SYTEM IS RUNNING, THE "RESERVE" REQUEST WILL BE TERMINATED.

2. IF "AS BADDISK" OR "COPY ERRORS" HAS NOT BEEN SPECIFIED, THE "RESERVE" REQUEST IS TERMINATED.

3. IF THE ABOVE TWO OPTIONS ARE SATISFIED, THEN THE AREA WILL BE RESERVED, AND THE FILE(S) CONTAINING THE ERROR WILL BE REMOVED.

THE FOLLOWING GENERAL ACTIVITIES OCCUR WHEN A "RESERVE" IS REQUESTED.

1. REMOVE ANY REQUESTED SPACE FROM THE AVAILABLE DISK TABLES.

2. EXAMINE THE DISK ALLOCATED FOR MEMORY DUMP AREAS AND MOVE IF NECESSARY.

3. PERFORM AN AUTOMATIC APO AND UPON COMPLETION OF COPY- ING THE DIRECTORY, RESTORE AP TO ITS PREVIOUS STATUS.

4. CHECK THE AREAS SET UP FOR ENTERUSERFILE OPERATIONS FOR BEING WITHIN THE REQUIRED AREA, AND IF SO REQUEST NEW SPACE.

5. COPY THE ENTIRE DIRECTORY INTO A NEW ARA. AT THIS TIME CHECK ALL NON-DIRECTORY HEADERS FOR ANY ROW WITHIN THEM POINTING WITHIN THE RESERVED AREA. IF A ROW IS FOUND, RECORD THE NAME OF THE FILE IN "RESERVE/EUNNN".

6. EXAMINE THE LIST OF FILES CREATED IN STEP #5. THIS FUNCTION WILL REQUEST EXCLUSIVE USE OF EACH GIVEN FILE. AFTER OBTAINING THIS USAGE, IT WILL THEN CHECK EACH ROW FOR BEING IN THE REQUESTED AREA, AND EACH ROW FOUND IS MOVED TO A NEW DISK AREA OF THE SAME CLASS.

7. CYCLE THROUGH ALL JOBS IN MIX (INCLUDING MCP STACK VECTOR) OBSERVING THE OLAY FILE DESCRIPTORS. IF ANY ADDRESSES ARE FOUND WITHIN THESE POINTING TO THE REQUESTED AREA, THE ROW IS MOVED TO A NEW AREA AND THE OLAY FILE ROW DESCRIPTOR IS UPDATED.

8. A PROCEDURE "RESERVE" HAS BEEN ADDED WHICH IS CALLED BY EACH ONE OF THE ABOVE STEPS TO DETERMINE IF ANY GIVEN DISK ADDRESS + RANGE IS TO BE RESERVED. IF IT IS, THEN THIS PROCEDURE WILL DO THE APPROPRIATE FORGETUSER DISK FUNCTIONS, RETAINING THE AREA THAT IS TO BE RESERVED. IT THEN MAKES AN ENTRY INTO AN "ABORT" FILE OF THE ADDRESS THAT HAS BEEN RESERVED. VIA THIS FILE, ANY SPACE RESERVED MAY BE RETURNED AT ANY TIME.

RETURN:

THIS FACILITY IS ENVOKED THROUGH THE KEYBOARD MESSAGE OF:

RET <DISK UNIT DESIGNATE> <RANGE> <ETX>

WHERE <RANGE> ::= <STORAGE UNIT RANGE>/<SWITCH RANGE>/ <EMPTY>

EXAMPLES:

1. RET DK 3
2. RET DK 9 SW 5 THRU 7, SW 1, 2, 4
3. RET DK 80 SU 0, SWITCH 7

SEMANTICS:

THIS FUNCTION WILL ALLOW RETURNING AREAS SPECIFIED AS "IADDISK" (EITHER VIA THE PREVIOUS MENTIONED RESERVE FUNCTION OR COLD START DECK) TO THE AVAILABLE DISK POOL. IF AN AREA IS DESIGNATED AS NOT READY (OR WRITE LOCKOUT), IT WILL NOT BE ENTERED INTO THE AVAILABLE POOL. IF A "RETURN" FUNCTION IS REQUESTED ON AN AREA THAT IS NOT FOUND TO BE DESIGNATED AS IAD DISK, AN APPROPRIATE ERROR MESSAGE WILL OCCUR AND THE RETURN REQUEST WILL BE TERMINATED.

THIS FUNCTION WILL READ THE DIRECTORY TO ENSURE THAT THERE ARE NO FILES WITHIN THE IAD DIRECTORY STRUCTURE POINTING WITH THE AREA TO BE "RETURNED". IF ANY FILE IS FOUND THE RETURN REQUEST WILL BE TERMINATED.

GENERAL INFORMATION:

ONLY ONE RESERVE OR RETURN MAY BE INITIATED AT ANY GIVEN TIME. IF AN ATTEMPT IS MADE TO START MORE THAN ONE, AN APPROPRIATE ERROR MESSAGE IS DISPLAYED.

A SEPARATOR IS REQUIRED BETWEEN ANY TWO ADJACENT SYNTAX ELEMENTS THAT ARE OF THE SAME TYPE, I.E., 1. A NUMBER FOLLOWED BY A NUMBER; OR . AN IDENTIFIER FOLLOWED BY AN IDENTIFIER.

A RESERVE/RETURN MAY BE TERMINATED VIA THE KEYBOARD MESSAGE OF <MIX INX> DS. HOWEVER THE SYSTEM ONLY EXAMINES FOR BEING TERMINATED AT CERTAIN TIMES WHICH MAY CAUSE SOME DELAY BEFORE THE ACTUAL TERMINATION OCCURS.

IF "THRU" SYNTAX IS USED THEN THE SECOND NUMBER MUST BE LARGER THAN THE FIRST.

D0115 DCPPROGEN - FULL DUPLEX - 08-28-72

INTRODUCTION

"FULL DUPLEX" DESCRIBES SOME PORTION OF A DATA TRANSMISSION LINK AS CAPABLE OF SIMULTANEOUS TRANSMISSION AND RECEPTION OF UNITS OF INFORMATION; "HALF DUPLEX" AS CAPABLE OF TRANSMISSION OR RECEPTION BUT NOT SIMULTANEOUS. THERE ARE THREE FUNCTIONAL AREAS OF THE DCP

SUBSYSTEM WHICH, PREVIOUSLY, WERE LIMITED TO A HALF-DUPLEX MODE OF OPERATION:

1. THE CLUSTER, WITH RESPECT TO EACH OF ITS LINE ADAPTERS, IS STRICTLY HALF-DUPLEX IN TERMS OF BYTES OF INFORMATION.

2. EACH LINE ADAPTER IS CONTROLLED AND MONITORED BY ONE (AT A TIME) NDL-WRITTEN READ- OR WRITE-REQUEST, WHICH IS, THEREBY, ALSO HALF-DUPLEX IN TERMS OF BYTES.

3. EACH RUNNING NDL REQUEST WAS ASSOCIATED WITH, AT MOST, ONE MESSAGE AREA, THE MESSAGE SPACE QUEUED IN MAIN MEMORY AT THE HEAD OF THE SELECTED STATION QUEUE. SO, FINALLY, THE DCP SUB-SYSTEM WAS HALF-DUPLEX IN TERMS OF MESSAGES.

IMPLEMENTATION OF A FULL-DUPLEX CAPABILITY REQUIRES SOLUTIONS IN THE THREE FUNCTIONAL AREAS:

1. THE CLUSTER INTERFACE TO A FULL-DUPLEX LINE (OR ITS MODEM) IS EFFECTED BY A "Y"-CABLE, CONNECTING THE LINE OR MODEM TO TWO LINE ADAPTERS, ASSIGNING BYTE TRANSMISSION LEADS TO ONE ADAPTER, BYTE RECEPTION TO THE OTHER.

2. CONSEQUENTLY, THERE ARE TWO SIMULTANEOUS NDL REQUESTS OR CONTROL LOGICS (ONE FOR EACH ADAPTER) RUNNING FOR THE SAME FULL-DUPLEX LINE. GENERALLY, BYTE TRANSMISSIONS AND RECEPTIONS ON A FULL-DUPLEX LINE ARE, ALTHOUGH PARALLEL, NOT COMPLETELY INDEPENDENT. THE TWO PARALLEL, SIMULTANEOUS, NDL LOGICS MUST THEN BE ABLE TO INTERACT WITH ONE ANOTHER. THE NDL/DCP FACILITY IS EXPANDED IN SEVERAL WAYS TO SERVE SUCH INTERACTION:

A. BOTH LOGICS WILL ADDRESS THE SAME TABLE SPACE. NDL TALLYS, TOGGLES, ETC., MAY THEN CONTAIN INTER-LOGIC CONTROL INFORMATION.

B. NEW NDL VERBS ARE INCLUDED BY WHICH EITHER ADAPTER-S NDL LOGIC MAY CONTROL, BE CONTROLLED BY THE OTHER.

3. ONE OF THE TWO ADAPTERS, AND ITS NDL LOGIC, WILL BE DESIGNATED AS THE "PRIMARY". AS SUCH, IT WILL CONTROL THE STATION QUEUE. THE OTHER WILL BE DESIGNATED THE "AUXILIARY".

THE AUXILIARY MAY CONTROL A SINGLE MESSAGE SPACE, NOT QUEUED IN THE STATION QUEUE. IN THIS WAY, EACH NDL LOGIC MAY HAVE A SEPARATE MESSAGE SPACE ASSOCIATED WITH IT, WITH AN INDEPENDENT, PARALLEL PATH TO THE MCP/DCP RESULT QUEUE, ENABLING FULL-DUPLEX MESSAGE TRANS- FERS.

THE NDL/DCP EXTENSIONS

DATA SPACE. THE NETWORK ASSOCIATED WITH EACH DCP OR CLUSTER EXCHANGE IS DEFINED IN MAIN SYSTEM MEMORY BY A TWO-DIMENSIONAL DOPE-VECTOR-LIKE TABLE STRUCTURE:

1. THE LINE VECTOR IS INDEXED BY CLUSTER/ADAPTER ADDRESS; THIS YIELDS A LINE DESCRIPTOR WHICH CONTAINS THE ABSOLUTE ADDRESS OF LINE TABLE.

2. THE LINE TABLE CONSISTS OF SEVERAL LINE-ASSOCIATED INFORMATION WORDS FOLLOWED BY A STATION VECTOR FOR THE STATIONS ON THE LINE. THE STATION VECTOR IS INDEXED BY STATION NUMBER AND YIELDS A STATION DESCRIPTOR WHICH CONTAINS THE ABSOLUTE ADDRESS OF STATION TABLE.

3. THE STATION TABLE CONTAINS STATION-ASSOCIATED INFORMATION.

BOTH PRIMARY AND AUXILIARY LINES HAVE SEPARATE, VALID LINE TABLES. LINE TABLES ARE EXPANDED TO CONTAIN A LINE REFERENCE TO AN ASSOCIATED LINE. THAT IS, THE LINE TABLE OF THE PRIMARY LINE WILL CONTAIN THE CLUSTER/ADAPTER ADDRESS OF THE AUXILIARY AND VICE VERSA.

THE AUXILIARY LINE TABLE CONTAINS ITS OWN INDEPENDENT REFERENCE TO A LINE CONTROL ROUTINE, DIFFERENT THAN THE PRIMARY LINE CONTROL.

THE AUXILIARY LINE TABLE CONTAINS ITS OWN ADAPTER TYPE, DELAYS AND TIMEOUT VALUES. THIS IS IMPORTANT FOR TOUCH-TONE/VOICE- RESPONSE, ETC.

THE AUXILIARY LINE TABLE CONTAINS ITS OWN LINE TALLYS AND TOGGLES. IN NDL THESE WILL BE SYNTACTICALLY DISTINGUISHED FROM THOSE OF THE PRIMARY BY A NEW NDL RESERVED WORD "AUXILIARY" OR "AUX" USED AS A PREFIXED QUALIFIER. THIS EXPANDS THE NDL CONTROLLED DATA SPACE, AS

EITHER SET OF TALLYS, TOGGLES MAY BE REFERENCED BY NDL STATEMENTS RUNNING FOR EITHER LINE.

OF THE LINE STATUS FLAGS (WRITEREADY, ACKNOWLEDGEREADY, NOT READY, SWITCHED STATUS, ETC.), ONLY AUXILIARY LINE BUSY IS VALID AS A SEPARATE ENTITY. ALL OTHERS ARE DESCRIPTIVE OF THE LINE PAIR AND ARE VALID ONLY IN THE PRIMARY LINE TABLE.

THE STATION INDEX IS VALID ONLY IN THE PRIMARY TABLE; THE STATION VECTOR IS APPENDED ONLY TO THE PRIMARY. NDL STATEMENTS EXECUTED FOR THE AUXILIARY WILL REFERENCE STATION RELATED QUANTITIES THROUGH THE PRIMARY TABLE STRUCTURE.

NDL STATEMENTS

<FORK STATEMENT> ::= <FORK LABEL>

<FORK LABEL> ::= <LABEL>

THE FORK STATEMENT MAY BE EXECUTED ON EITHER PRIMARY OR AUXILIARY LINE. IT CAUSES THE OTHER LINE TO EXECUTE THE CODE BEGINNING AT THE LABEL SPECIFIED, WHILE THE FIRST LINE CONTINUES EXECUTING IN PARALLEL.

IF THE OTHER LINE HAD ALREADY BEEN BUSY, IT IS NOT FORKED AND THE STATEMENT IS A NO-OP; OTHERWISE, BEING FORKED, IT IS AUTOMATICALLY MARKED BUSY. THE "BUSY" STATUS IS THE SAME AS THAT WHICH NDL STATEMENTS MAY REFERENCE (LINE (BUSY), AUX LINE (BUSY)) SO THE NDL CODE RUNNING FOR ONE LINE OR ANOTHER MAY RESET ITS BUSY STATUS, SO AS TO, EVEN THROUGH RUNNING, ALLOW ITSELF TO BE FORKED TO HANDLE A MORE IMPORTANT FUNCTION.

<WAIT STATEMENT> ::= WAIT/WAIT (<TIME>)/
WAIT(<TIME>:<LABEL>)

THE WAIT STATEMENT CAUSES THE LINE FOR WHICH IT WAS EXECUTED TO BE SUSPENDED UNTIL EITHER THE OTHER LINE EXECUTES A CONTINUE STATEMENT OR, IF SPECIFIED, THE ELAPSED TIME. THE WAIT STATEMENT MAY BE EXECUTED FOR EITHER THE PRIMARY OR AUXILIARY LINE AND IN EITHER A REQUEST OR CONTROL ROUTINE.

<RECEIVE ACTION PART> IS EXPANDED TO INCLUDE "CONTINUE" AS A RECEIVE ACTION.

EXAMPLE:

```
RECEIVE (1 SEC) CHARACTER [ERROR[0], CONTINUE:3]
```

SUCH A RECEIVE STATEMENT IS AS A COMPOUNDED RECEIVE AND WAIT STATEMENT, ALLOWING A LINE ALGORITHM TO RESPOND TO:

1. A BYTE RECEIVED OR ANY RELATED ERROR OCCURRENCE, SUCH AS PARITY ERROR, LOSS OF CARRIER, ETC.;

2. A TIMEOUT; OR

3. BEING CONTINUED BY THE OTHER LINE.

<CONTINUE STATEMENT> ::= CONTINUE

THE CONTINUE STATEMENT CAUSES THE OTHER LINE TO RESUME PROCESSING, PROVIDED IT HAD BEEN SUSPENDED BY A WAIT STATEMENT OR RECEIVE STATEMENT WITH A CONTINUE ACTION SPECIFIED; OTHERWISE, THE STATEMENT HAS NO AFFECT ON THE OTHER LINE. IN EITHER CASE, THE LINE ON WHICH THE CONTINUE STATEMENT WAS EXECUTED ITSELF CONTINUES WITHOUT INTERRUPTION.

<ADAPTER TYPE> WILL BE EXPANDED TO THE POSSIBILITY OF A PAIR OF ADAPTER TYPE VALUES, FOR USE IN THE MODEM, TERMINAL AND STATION DECLARATIONS. THIS ALLOWS THE SPECIFICATION OF DIFFERENT ADAPTER TYPES FOR THE PRIMARY AND AUXILIARY LINES.

<CONTROL STATEMENT> IN THE TERMINAL SECTION WILL BE EXPANDED TO ALLOW SPECIFICATION OF TWO CONTROL ROUTINES; IF THE SECOND CONTROL ROUTINE (FOR THE AUXILIARY) IS NOT SPECIFIED FOR A FULL-DUPLEX TERMINAL, THEN THE DEFAULT EQUIVALENT OF AN IDLE STATEMENT WILL BE USED.

<TERMINATE STATEMENT> WILL BE EXPANDED TO INCLUDE A SIMPLE, UNQUALIFIED "TERMINATE". AS IS TRUE OF ALL TERMINATE STATEMENTS, IT MAY BE USED ONLY IN A REQUEST. THE NEW, SIMPLE TERMINATE WILL SERVE ONLY TO EXIT FROM THE REQUEST TO THE APPROPRIATE CONTROL ROUTINE (PRIMARY OR AUXILIARY). THE EXIT WILL BE DONE WITHOUT

SUSPENSION OF THE LINE, WITH NO AFFECT ON THE READY STATUS OF THE STATION AND WITHOUT RETURNING OR DISCARDING ANY ASSOCIATED MESSAGE SPACE. THE SIMPLE TERMINATE ELIMINATES THE NEED OF THE USE OF A TERMINATE NOINPUT STATEMENT IN A WRITE REQUEST ALLOWED TO DATE.

<THE LINE TYPE STATEMENT> MAY INCLUDE THE CLAUSE

<DUPLEX: LINE IDENTIFIER>.

SUCH A TYPE STATEMENT IS TO BE USED IN THE LINE DECLARATION FOR THE "PRIMARY" LINE, AND REFERENCES THE "AUXILIARY LINE". THE LINE REFERENCED AS AUXILIARY MAY ITSELF NOT HAVE A TYPE STATEMENT, NOR MAY IT HAVE ANY STATIONS.

D0116 DCSTATUS - 06-15-72

THIS MARKS THE FIRST RELEASE OF AN ON-LINE DATA COMMUNICATIONS ANALYSIS ROUTINE IDENTIFIED AS SYSTEM/DCSTATUS. SYSTEM/DCSTATUS MAY BE "RUN" FROM THE SPO IN THE FOLLOWING MANNER:

```
RUN SYSTEM/DCSTATUS ("ALL")
```

THIS WILL PRODUCE A DUMP AND ANALYSIS OF THE DATACOM SYSTEM TABLES ON A SITE LINE PRINTER WHILE DATACOM IS RUNNING.

D0117 PRINTBINDINFO PROGRAM - 08-24-72

PRINTBINDINFO IS A PROGRAM USED TO INTERPRET AND PRINT PROGRAM DESCRIPTIONS. PROGRAM DESCRIPTIONS CONTAIN THE INFORMATION NECESSARY FOR THE BINDER TO BIND PROGRAMS TOGETHER. THE INFORMATION IS USEFUL IN CHECKING COMPATIBILITY BETWEEN DECLARATIONS IN THE HOST AND PROGRAM TO BE BOUND TO IT, IN ASSIGNING NEW SEGMENTS AND STACK LOCATIONS, AND IN CHANGING ADDRESS COUPLES IN THE CODE WHICH MIGHT REFERENCE THESE NEW STACK LOCATIONS. INFORMATION IS PRODUCED FOR BOTH THE D2 STACK, KNOWN AS THE "GLOBAL DIRECTORY", AND ANY EXTERNAL PROCEDURES WHICH HAVE BEEN DECLARED. FOR MORE INFORMATION ON PROGRAM DESCRIPTIONS, SEE THE PROGRAM BINDER MANUAL.

PRINTBINDINFO MIGHT BE USEFUL FOR 1) CHECKING COMPATIBILITY OF DECLARATIONS IN A HOST, 2) PRODUCING A SHORT SUMMARY OF ALL GLOBAL VARIABLES IN A PROGRAM, OR 3) CHECKING THE VALIDITY OF THE PROGRAM DESCRIPTION IN CASES WHERE THE BINDER SEEMS TO BE IN ERROR.

TO EXECUTE PRINTBINDINFO, THE FILE "CODE" SHOULD BE LABEL EQUATED TO THE CODE FILE FOR WHICH A PROGRAM DESCRIPTION LISTING IS DESIRED. IF INFORMATION CONCERNING ONLY CERTAIN IDENTIFIERS IS DESIRED, AND NOT THE WHOLE PROGRAM DESCRIPTION, THEN AN OPTIONAL CARD FILE NAMED "SELECTIDS" MAY BE INPUT WHICH CONTAINS THE IDENTIFIERS FOR WHICH INFORMATION IS DESIRED. THE IDENTIFIERS MAY BE PUNCHED IN FREE FORM, SEPARATED BY BLANKS (OR END OF CARD), FOR THE FULL 80 COLUMNS OF THE CARD. IF THE CARD FILE "SELECTIDS" IS NOT PRESENT, A FULL PROGRAM DESCRIPTION WILL BE OUTPUT.

EXAMPLE ONE:

THE FOLLOWING DECK WILL OUTPUT THE FULL PROGRAM DESCRIPTION CONTAINED IN THE CODE FILE "MY/FILE":

```
<I> RUN SYSTEM/PRINTBINDINFO
<I> FILE CODE (TITLE = MY/FILE)
<I> END.
```

EXAMPLE TWO:

THE FOLLOWING DECK WILL OUTPUT INFORMATION CONTAINED IN THE PROGRAM DESCRIPTION OF "MY/FILE" CONCERNING THE IDENTIFIERS "ID1" AND "ID2":

```
<I> RUN SYSTEM/PRINTBINDINFO
<I> FILE CODE (TITLE = MY/FILE)
<I> DATA SELECTIDS
   ID1    ID2
<I> END.
```

D0118 BINDER - DOLLAR OPTION NOBINDINFO - 08-25-72

THE DOLLAR OPTION "NOBINDINFO" HAS BEEN ADDED TO ALGOL, DCALGOL,

AND BINDER. INFORMATION FOR BINDING IS AUTOMATICALLY EMITTED UNLESS NOBINDINFO IS SET. IF SET WITH AUTOBINDING, NOBINDINFO IS IGNORED BY THE COMPILER AND USED BY THE BINDER. NOBINDINFO IS ALSO IGNORED FOR INTRINSICS AND ALL PROCEDURES WHICH ARE VALID ONLY FOR BINDING. INFORMATION NEEDED FOR IPC IS NOT SUPPRESSED BY THE OPTION.

D0119 DISK PACK CONFIDENCE ROUTINES - 09-05-72

THE DISK PACK VERIFY ROUTINES ARE DESIGNED TO DETECT AND HELP DIAGNOSE HARDWARE FAILURES IN THE ELECTRONIC AND MECHANICAL OPERATION OF MODELS 214, 215 AND 230 DISK PACK DRIVES. THESE ROUTINES ARE INVOKED BY MEANS OF VARIANTS IN THE VERIFY STATEMENT; THEY RUN CONCURRENTLY WITH OTHER PROGRAMS UNDER CONTROL OF THE B6700 MCP.

TESTS WHICH WRITE INFORMATION ON THE DISK PACK REQUIRE THAT THE AREA BEING TESTED BE SPECIFIED BY MEANS OF AN SCR FILE DECLARATION. THESE FILES ARE CREATED IN RESPONSE TO THE "XD" KEYBOARD MESSAGE.

THE FILE NAME FOR THE BADDISK AREA CREATED BY AN XD REQUEST WILL BE OF THE FORM <MULTI-FILE IDENTIFICATION>/<ADDRESS PART>, WHERE <MULTI-FILE IDENTIFICATION> IS THE PACK NAME FOR LABELED INTERCHANGE PACKS AND "BADDISK" FOR NATIVE-MODE PACKS; <ADDRESS PART> IS AN IDENTIFIER CONSISTING OF THE LETTERS "AD" FOLLOWED BY THE SEGMENT NUMBER OF THE FIRST SEGMENT IN THE FILE.

THESE FILES MUST REFERENCE DISK OR DISK PACK AREAS WHICH SATISFY THE FOLLOWING CONDITIONS:

1. THE FILE HAS EXACTLY ONE ROW, AND
2. THE END-OF-FILE POINTER POINTS AT THE BEGINNING OF THE FIRST SEGMENT, AND
3. THE FILETYPE IS "XDISKFILE."

THE "XD" KEYBOARD MESSAGE IS THE ONLY MEANS OF CREATING A FILE WHICH SATISFIES ALL THREE OF THESE CONDITIONS.

SYNTAX

THE XD-ED PACK AREA MAY BE REFERENCED BY AN SCR PROGRAM BY MEANS OF THE FILE DECLARATION:

<DISKPACK FILE DECLARATION>::=DISKPACK FILE <IDENTIFIER><DISKPACK FILE NAME>.

THE FORM OF THE FILE NAME DEPENDS UPON THE PACK MODE:

FOR INTERCHANGE PACKS, IT IS <PACK NAME>/AD<INTEGER>; FOR NATIVE-MODE PACKS, IT IS BADDISK/EU<INTEGER>AD<INTEGER> WHERE THE EU-NUMBER FOR NATIVE MODE PACKS IS THE NUMBER OF THE UNIT ON WHICH THE PACK WAS MOUNTED WHEN THE XD WAS EXECUTED.

THE VERIFY STATEMENT WHICH REFERENCES DISKPACKS IS OF THE FORM:

VERIFY DISKPACK <UNIT OR FILE SPECIFIER>(<DISK TEST PARAMETERS>).

ERROR PRINTOUT

THE DETECTION OF A RESULT DESCRIPTOR AND/OR DATA COMPARISON ERROR (ON READ OPERATIONS) WILL CAUSE THE RELEVANT HARDWARE INFORMATION TO BE PRINTED ON THE OUTPUT MEDIUM. THIS INFORMATION CONSISTS OF THE IOCW, AREA DESCRIPTOR, RESULT DESCRIPTOR, I/O COUNT, AND DISK PACK ADDRESS, FOLLOWED BY UP TO THIRTY WORDS OF EXPECTED DATA AND DATA READ.

THE FORMAT OF THIS PRINTOUT AND THE INTERPRETATION OF THE RESULTS IS NOT SIGNIFICANTLY DIFFERENT FROM THE ERROR PRINTOUT FOR HEAD-PER-TRACK DISK. THE CONVENTIONS USED FOR NUMBERING THE DISK PACK DRIVES, FACES, AND TRACKS CAN BE DEDUCED FROM THE ADDRESS BREAKDOWN FOR THE SMALLEST AND LARGEST ADDRESSES FOR THE GIVEN TYPE OF UNIT, WHICH IS PRINTED BEFORE INITIATING THE TEST SEQUENCE.

TEST DESCRIPTIONS

THE FOLLOWING PARAGRAPHS PROVIDE A BRIEF DESCRIPTION OF THE OPERATION OF EACH OF THE DISK PACK VERIFY TESTS.

TEST1

TEST FUNCTIONS: DATA RECORDING AND READ BACK.
TEST REQUIREMENTS: BADDISK FILE.

THE FILE AREA IS FIRST WRITTEN WITH THIRTY-WORD RECORDS CONSISTING OF ONE WORD OF DECIMAL FILE ADDRESS FOLLOWED BY TWENTY-NINE WORDS OF THE WORST-CASE BIT PATTERN. AT THE CONCLUSION OF THE WRITE PHASE, ALL SEGMENTS ARE READ BACK AND THE DATA IS CHECKED.

THE ADDRESSING PATTERN REFERENCES INCREASING SEGMENTS USING AN ADDRESS INTERLACE OF 10 SEGMENTS FOR BOTH WRITE AND READ OPERATIONS.

RESULT DESCRIPTOR ERRORS ARE REPORTED FOR BOTH PHASES, AND DATA COMPARISON ERRORS ARE PRINTED ON THE READ BACK.

TEST2

TEST FUNCTIONS: ADDRESS COINCIDENCE ON SEEK OPERATIONS;
DATA RECORDING ON WRITE.
TEST REQUIREMENTS: BADDISK FILE OF AT LEAST ONE-TRACK EXTENT.

THE BADDISK AREA IS FIRST INITIALIZED BY WRITING THE ADDRESS KEY FOLLOWED BY 29 WORDS OF THE WORST-CASE DATA PATTERN TO ENTIRE TRACKS (33-SEGMENTS PER I/O), WITH "SHORT" I/O LENGTHS FOR THE BEGINNING AND END OF THE DISK PACK AREA IF THE ADDRESS LIMITS DO NOT COINCIDE WITH TRACK BOUNDARIES.

AT THE CONCLUSION OF THIS INITIALIZATION, THE SEGMENTS ARE REFERENCED IN THE ORDER (0, N-1, N-2, 2, N-3, ..., N DIV 2, ..., N-1, 0) USING AN I/O LENGTH OF ONE WORD TO READ BACK THE ADDRESS KEY. THIS ADDRESS SEQUENCE WILL READ THE ADDRESS KEY FROM EVERY SEGMENT USING AN IMPLICIT SEEK BOTH TOWARD AND AWAY FROM THE DRIVE SPINDLE.

RESULT DESCRIPTOR ERRORS ARE REPORTED FOR BOTH WRITE AND READ OPERATIONS; AN UNEQUAL COMPARISON OF THE WRITTEN ADDRESS KEY WITH THE KEY READ BACK (WHICH MAY BE CAUSED BY EITHER ADDRESSING ERRORS OR DATA RECORDING ERRORS) PRODUCES ONE WORD OF DATA PRINTOUT FOR EACH SEGMENT WHICH FAILS THE KEY-COMPARISON TEST.

TEST3

TEST FUNCTIONS: SEGMENT CROSSOVER AND DATA RECORDING.
TEST REQUIREMENTS: BADDISK FILE OF AT LEAST ONE-TRACK EXTENT.

TEST THREE PERFORMS THE FOLLOWING COMBINATIONS OF SINGLE- AND DOUBLE-SEGMENT OPERATIONS:

1. WRITE THE ENTIRE BADDISK AREA WITH 60-WORD I/O OPERATIONS;

2. READ EACH SEGMENT BACK USING 30-WORD READS, CHECKING ADDRESS KEYS AND DATA;

3. WRITE THE BADDISK AREA AGAIN USING 30-WORD I/O OPERATIONS; AND

4. READ THE DATA BACK 60 WORDS AT A TIME AND CHECK EVERYTHING.

IN CASES ONE AND FOUR, THE DATA CONSISTS OF TWO RECORDS CONTAINING THE DECIMAL ADDRESS KEY FOLLOWED BY TWENTY-NINE WORDS OF THE WORST-CASE DATA PATTERN.

RESULT DESCRIPTOR AND DATA COMPARISON ERRORS ARE PRINTED, AS APPROPRIATE, FOR ANY FAILING OPERATION.

TEST3 PROVIDES A MEANS OF NARROWING THE DISK ADDRESS RANGE ON A SEARCH FOR SEEK AND/OR ADDRESSING ERRORS WHICH CAN ONLY BE RESOLVED TO THE NEAREST TRACK USING TEST2.

TEST4

TEST FUNCTIONS: MULTIPLEXOR TAG-TRANSFER; DISK PACK RECORDING OF RANDOM DATA PATTERNS.
TEST REQUIREMENTS: BADDISK FILE OF AT LEAST ONE-TRACK EXTENT.

THE PROGRAM LOGIC FOR DISK PACK TEST4 IS THE SAME AS THAT FOR HEAD-PER-TRACK DISK TEST14. REFER TO CHAPTER 3.

TEST5

TEST FUNCTIONS: CHARACTER-ORIENTED I/O LENGTHS USING TAG-TRANSFER.

TEST REQUIREMENTS: BADDISK FILE

DISK PACK TEST5 USES THE SAME LOGIC TO TEST THE SAME FUNCTIONS AS HEAD-PER-TRACK DISK TEST15. REFER TO CHAPTER 3.

TEST6

TEST FUNCTIONS: MULTIPLE-SEGMENT DATA RECORDING.
TEST REQUIREMENTS: BADDISK FILE OF AT LEAST FOUR SEGMENTS.

DISK PACK TEST6 IS AN ADAPTATION OF HEAD-PER-TRACK DISK TEST12 TO PACKS. THE SAME OVERLAPPED SEQUENCE OF ALL-ZEROES AND ALL-ONES RECORDS IS USED BY BOTH TESTS. REFER TO TEST12, CHAPTER 3.

TEST7

TEST FUNCTIONS: MULTIPLEXOR FORCE-TAG-3 OPERATION;
DATA RECORDING AND READ BACK.
TEST REQUIREMENTS: BADDISK FILE OF AT LEAST TWO SEGMENTS.

PROGRAM LOGIC FOR THIS TEST IS DISCUSSED IN THE TEST16 SECTION OF CHAPTER 3.

TEST8

TEST FUNCTIONS: DATA VALIDITY
TEST REQUIREMENTS: NONE

TEST8 PERFORMS READ OPERATIONS WITHOUT FIRST INITIALIZING THE PACK AREA TO A KNOWN DATA PATTERN.

THE DISK PACK AREA IS READ INTO A BUFFER ONE TRACK (33 SEGMENTS) PER I/O, WITH APPROPRIATE SHORTER I/O LENGTHS AT THE BEGINNING AND END OF THE ADDRESS RANGE, AS APPROPRIATE.

IF A RESULT DESCRIPTOR ERROR IS ENCOUNTERED, THE FAILING TRACK IS READ AGAIN ONE SEGMENT AT A TIME IN AN ATTEMPT TO NARROW THE ADDRESS RANGE OF THE ERRONEOUS SEGMENTS.

THE ERROR PRINTOUT DOES NOT SHOW ANY OF THE DATA, SINCE THIS TEST CANNOT DETERMINE WHAT THE CORRECT DATA WAS. THERE IS NO PROVISION FOR OPERATION RETRY WHEN AN ERROR IS ENCOUNTERED ON THE SINGLE-SEGMENT SEARCH PHASE.

EXAMPLES

1. VERIFY DISKPACK PK80 (SEGMENT 1234 FOR 5678, SELECT ALL);

2. UNIT PKX = PK80; VERIFY DISKPACK UNIT PKX (SEGMENT 34 THRU 56, SELECT ALL);

THESE TWO EXAMPLES DO NOT SPECIFY A BADDISK FILE; CONSEQUENTLY, TESTS WHICH PERFORM WRITE OPERATIONS WILL NOT BE RUN.

THE FIRST EXAMPLE WILL PERFORM READ-ONLY TESTS ON THE 5678 SEGMENTS BEGINNING AT SEGMENT 1234; THE SECOND EXAMPLE SPECIFIES THE FIRST AND LAST SEGMENTS TO BE TESTED AND COULD ALSO HAVE BEEN WRITTEN "SEGMENT 34 FOR 23."

3. FILE X = "A/AD1000"; VERIFY DISKPACK FILE X (SELECT ALL); THE BADDISK FILE WILL CAUSE ALL TESTS TO BE EXECUTED.

4. FILE BD = "A/AD123"; VERIFY DISKPACK FILE BD (SELECT ALL EXCEPT TEST2, TEST4); ALL TESTS EXCEPT NUMBERS 2 AND 4 WILL BE RUN ON THE ENTIRE BADDISK FILE.

5. FILE Q = "PACK/ADO"; VERIFY DISKPACK FILE Q (OFFSET 20 FOR 10, SELECT TEST1); THE TEN SEGMENT AREA BEGINNING AT THE TWENTIETH SEGMENT BEYOND THE BEGINNING OF THE BADDISK FILE WILL BE TESTED

6. FILE Z = "CHECKER/AD23"; VERIFY DISKPACK FILE Z (SEGMENT 50 FOR 100, SELECT TEST3, TEST4);

TESTS 3 AND 4 WILL BE EXECUTED WITHIN THE 100-SEGMENT AREA BEGINNING AT ABSOLUTE SEGMENT 50. IF THIS ADDRESS RANGE IS NOT WITHIN THE BOUNDS OF THE BADDISK FILE, SUITABLE RUN-TIME ERRORS WILL APPEAR ON THE OUTPUT MEDIUM, AND THE TESTS WILL NOT BE RUN.

D0120 READ-WRITE DISK VERIFY TESTS - 09-11-72

THE DISK VERIFY TESTS PERFORM READ CHECK AND WRITE/READY OPERATIONS ON BOTH MAINTENANCE AND NORMAL SEGMENTS. THE WRITE/READ TESTS WHICH REFERENCE NORMAL SEGMENTS WILL BE RUN ONLY IF THE <UNIT OR

FILE SPECIFIER> IS A FILE.

DATA PATTERNS

THE WRITE/READ TESTS USE ONE OF THREE POSSIBLE DATA PATTERNS, THE CHOICE BEING A FUNCTION OF THE TEST NUMBER. THE HEADING LINE PRINTED ON THE OUTPUT MEDIUM IDENTIFIES THE DATA BEING USED AS ONE OF THE FOLLOWING:

RANDOM

THE "RANDOM" DATA CONSISTS OF THE CONTENTS OF THE 128-WORD MCP ARRAY OF VALUES OF THE POWERS OF TEN (POTM). THIS DATA PATTERN WAS CHOSEN BECAUSE OF THE FAVORABLE REPUTATION OF B3500 DISK TESTS WHICH USE MANY CYCLES OF A RANDOM DATA PATTERN AND BECAUSE EACH WORD IN THIS ARRAY HAS A UNIQUE VALUE (THERE ARE NO REPEATS).

THE TESTS WHICH USE THIS DATA TYPICALLY SELECT THIRTY WORDS FROM THE MCP COPY OF THE ARRAY, BEGINNING AT WORD RELATIVESEGMENT MOD (128-30); IN OTHER WORDS, THE PATTERN REPEATS EVERY 98 SEGMENTS. IN ALL CASES, THE ABSOLUTE DISK ADDRESS IS PLACED IN THE FIRST WORD OF THE WRITTEN DATA IN DECIMAL IN ORDER TO DETECT AND DIAGNOSE ADDRESSING ERRORS; AT LEAST THIS FIRST WORD IS DIFFERENT IN EVERY SEGMENT REFERENCED. IN ALL CASES, THE INPUT BUFFER IS FILLED WITH ZEROES BEFORE INITIATING A DISK READ, SO THAT SHORT READS WILL BE DETECTED.

SEGMENT 5

THE NON-OVERLAYABLE MCP CODE IS USED AS A DATA PATTERN BY SEVERAL OF THE TEST ROUTINES; THE FUNCTIONAL AREAS CHECKED BY THESE TESTS ARE:

I/O LENGTH MULTIPLEXOR LOGIC, SEGMENT/ZONE/TRACK/FACE CROSSOVER EU LOGIC, AND DATA RECORDING AND ERROR DETECTION IN THE STORAGE UNIT.

THE DATA IS WRITTEN TO DISK WITHOUT MOVING IT TO AN OUTPUT BUFFER IN ORDER TO REDUCE THE SCR MEMORY REQUIREMENTS. A

CONSEQUENCE OF THIS MODE OF OPERATION IS THAT THE IOCW USED TO WRITE THIS DATA MAY BE FROM TEN TO FIFTEEN WORDS IN FRONT OF THE FIRST WORD OF PROGRAM CODE. THESE WORDS ARE OPERANDS BELONGING TO AN MCP ARRAY AND MAY CHANGE BETWEEN THE TIME THEY ARE TRANSFERRED TO DISK AND THE TIME THEY ARE READ BACK INTO AN INPUT BUFFER AND COMPARED. THE RESULT IS THAT THESE WORDS CANNOT BE CHECKED FOR CORRECT DATA TRANSMISSION ON READBACK AND WILL ONLY APPEAR IN THE DATA PRINTOUT IF THE DATA WAS TRANSFERRED TO DISK INCORRECTLY. THIS FAILURE SHOULD ALSO BE EVIDENT FROM THE TAG-3 PROGRAM CODE COMPARISON ERRORS (THE DATA IN THE "DATA READ" LINES OF OUTPUT WILL BE SHIFTED WITH RESPECT TO THE "CORRECT DATA" PRINTOUT).

THE DATA IS WRITTEN TO AND READ FROM THE DISK AREA IN SUCH A MANNER THAT ALL SEGMENTS ARE FILLED WITH A FULL THIRTY WORDS OF DATA, AND THE OPERATION IS REPEATED FIVE TIMES BY PLACING THE WRITEIOCW AT FIVE DIFFERENT LOCATIONS IN FRONT OF THE FIRST WORD 0 PROGRAM CODE IN THE WRITE PORTION OF THESE TESTS. THE REASONS FOR THIS REPETITION IS THAT EACH OF THE FIVE REPEATS WILL PUT A DIFFERENT BIT PATTERN ON ANY GIVEN PIECE OF DISK SURFACE AND, THUS, PROVIDE SOME REDUNDANCY IN CHECKING THE DISK SURFACE FOR ERROR-PRODUCING IRREGULARITIES.

WORST-CASE DATA PATTERNS

THE WORST-CASE PATTERNS WERE DESIGNED TO CHECK FOR THE FOLLOWING HARDWARE ERRORS:

1. HEAD SATURATION AND PEAK-SHIFTING: GROUPS OF FIVE ADJACENT ONE-BITS AND ZERO-BITS.

2. ALTERED FLUX DISTRIBUTION IN HEAD CAUSED BY CHIPPED HEAD GAP: ISOLATED ZERO IN A FIELD OF ONES.

3. SENSE-AMPLIFIER OVERLOAD AND RECOVERY TIME: ALTERNATE ZERO-ONE-ZERO BITS.

THIS BIT PATTERN AS RECORDED IF THE DISK SURFACE DOES NOT MATCH THE BIT PATTERN IN THE DATA WORD IN MEMORY (OR IN THE MULTI- PLEXOR D-

REGISTER) BECAUSE THE DATA IN THE D-REGISTER IS TRANSFERRED TO THE DISK CONTROL EIGHT BITS AT A TIME (MODEL I) OR 16 BITS AT A TIME (MODEL II) STARTING WITH THE MOST SIGNIFICANT BYTE; THE DATA IN THE CONTROLS 8 OR 16 BIT BUFFER IS THEN TRANSFERRED TO THE DISK SURFACE LEAST-SIGNIFICANT BIT FIRST.

THUS, THE "DATA WRITTEN" LINE OF SCR PRINTOUT AND THE BIT PATTERN ON THE DISK SURFACE FOR THE TWO METHODS OF SERIALIZATION ARE:

MODEL I:

```
DATA WORD IN MEMORY = 4 "05DF0705DF07"*
BIT PATTERN ON DISK = 1"10100000 11111011
                       11100000 10100000 11111011
                       11100000"
```

MODEL II:

```
DATA WORD IN MEMORY = 4"7C14141F1F7C"
BIT PATTERN ON DISK = 1"0010100000111110
                       1111100000101000
                       0011111011111000"
```

THE READ PORTIONS OF TESTS WHICH USE THE WORST-CASE PATTERN FILL THE INPUT BUFFER WITH THE ONE-COMPLEMENT OF THE CORRECT DATA BEFORE INITIATING THE READ. THUS, A "DATA READ"/"CORRECT DATA" PAIR OF ENTRIES WHICH ARE BIT COMPLEMENTS OF EACH OTHER INDICATES THAT THE READ DID NOT TRANSFER THE FULL AREA-DESCRIPTOR COUNT OF DATA.

ADDRESS KEYS

ALL OF THE TESTS EXCEPT THOSE WHICH USE THE SEGMENT-5 DATA PLACE THE ABSOLUTE DISK ADDRESS IN THE FIRST WORD OF DATA WRITTEN TO DISK. THIS ADDRESS CONSISTS OF TWELVE PACKED DECIMAL DIGITS (THE RESULT OF A SCALE RIGHT FINAL 12 ON THE BINARY DISK ADDRESS).

A DATA COMPARISON ERROR IN WHICH THE FIRST WORD OF DATA READ BACK DOES NOT EQUAL THE FIRST WORD OF DATA WRITTEN TYPICALLY MEANS THAT EITHER THIS ADDRESS KEY WAS DESTROYED BY A SUBSEQUENT WRITE TO THE

NEXT LOWER SEGMENT ADDRESS WHICH TRANSFERRED TOO MUCH DATA OR THAT THE ADDRESS COINCIDENCE LOGIC IN THE EU OR THE ADDRESS TRACK ON DISK ARE CAUSING A LEGITIMATE ADDRESSING ERROR.

*NOTE: THE NUMBER IN FRONT OF A QUOTE MARK IS THE NUMBER OF BITS IN EACH OF THE CHARACTERS INSIDE THE QUOTES. 1"1" IS BINARY 1, 4"1" IS HEX 1; FRAME SIZES OF 3 (OCTAL), 6 (BCL) AND 8 (EBCDIC) ARE ALSO USED OCCASIONALLY.

DATA COMPARISON

THE COMPARISON OF THE DATA READ WITH THE DATA WRITTEN IS PERFORMED USING STRING COMPARE OPERATORS (E.G., CNED) FOR THE TESTS WHICH TRANSFER OPERAND DATA; THE NON-OPERAND SEGMENT-5 DATA IS COMPARED ONE WORD AT A TIME USING THE SAME OPERATOR.

ADDRESS KEYS MAY BE COMPARED USING EITHER SAME OR CNED, DEPENDING UPON WHICH TESTS IS BEING PERFORMED.

DISK-ADDRESS SEQUENCES

THE TESTS USE ONE OF FIVE POSSIBLE ADDRESS SEQUENCES TO "COVER" THE DISK ADDRESS RANGE. THESE SEQUENCES ARE:

1. INCREASING NORMAL SEGMENTS (TESTS 2, 3, 8 AND 10). THE ENTIRE BADDISK FILE IS WRITTEN BEFORE ANY SEGMENT IS READ BACK AND CHECKED.

   A DISK ADDRESS INTERLACE OF 8 SEGMENTS IS USED FOR TESTS 3, 8 AND 10, AND AN INTERLACE OF 10 IS USED FOR THE WRITE PORTION OF TEST2. "AN INTERLACE OF 8" MEANS THAT THE SEGMENTS ARE REFERENCED IN THE FOLLOWING ORDER:

   (0, 8, 16, 24, 32, ...), (1, 9, 17, 25, 33, ...), (2, 10, 18, 26, 34, ...), (3, 11, 19, 27, 35, ...), ...,(7, 15, 23, 31, 34, ...) WITH THE "0" CORRESPONDING TO THE FIRST SEGMENT OF THE ADDRESS RANGE; THE PARENTHESIS INDICATE THAT THIS GROUP OF ADDRESSES WILL BE CONTINUED UNTIL THE NEXT MEMBER IS BEYOND THE UPPER ADDRESS LIMIT. AT THIS POINT, THIS GROUP OF PARENTHESIZED ADDRESSES IS TERMINATED AND THE NEXT GROUP IS

BEGUN.

THE SEVEN SEGMENT LATENCY IN THE INTERLACE-OF-8 CASE PROVIDES ENOUGH TIME TO HANDLE THE I/O FINISH INTERRUPT, CHECK THE RESULT DESCRIPTOR AND DATA (ON READS) AND PROCESS THE REQUEST FOR THE I/O INITIATE FOR THE NEXT ADDRESS BEFORE THAT SEGMENT HAS PASSED UNDER ITS READ/WRITE HEAD. IN THE WORST CASE (IA2 STORAGE UNIT, ZONE 1 WITH 43 SEGMENTS PER TRACK), THIS ALGORITHM WILL ACCESS FIVE SEGMENTS PER REVOLUTION; IN THE BEST CASE (IIB2/IIB4 STORAGE UNIT, ZONE 7 WITH 216 SEGMENTS/TRACK), THERE WILL BE TWENTY SEVEN "HITS" PER REVOLUTION.

2. DECREASING NORMAL SEGMENTS (TESTS 9 AND 11)

THE PURPOSE OF THESE TESTS IS TO DETERMINE WHETHER THE I/O HARDWARE IS TRANSFERRING EXACTLY THIRTY WORDS TO AND FROM DISK. BY STARTING AT THE LARGEST DISK ADDRESS AND PROGRESSING TOWARD THE SMALLEST, WE ARE ASSURED THAT ANY DATA TRANSFERRED BEYOND THE END OF THE CURRENT SEGMENT WILL SPILL OVER INTO THE ADDRESS KEY OF A SEGMENT THAT HAS ALREADY BEEN WRITTEN AND, THUS, PRODUCE A DATA COMPARISON ERROR ON READBACK.

THE NEXT DISK ADDRESS IS DETERMINED BY SUBTRACTING THE QUANTITY (SEGMENTSPERTRACK-8) FROM THE CURRENT ADDRESS, WHERE THE SEGMENTSPERTRACK IS 151 FOR IA2 STORAGE UNITS, 278 FOR IC3 AND IC4 UNITS.

ON MODEL II UNITS, THE TRACK COMPONENT OF THE DISK ADDRESS VARIES MORE RAPIDLY THAN THE ZONE COMPONENT SO THAT A FIXED INTERLACE FACTOR IS LESS ADVANTAGEOUS THAN ON MODEL I. THE BACKWARD ADDRESSING FOR MODEL II IS OPTIMIZED BY SUBTRACTING (SEGMENTSPERTRACK-8) FROM THE PREVIOUS ADDRESS, BUT THE TEST FINISHES ALL WRITES TO THE TWO 50-TRACK ZONES CONTAINING THE SAME NUMBER OF SEGMENTS PER TRACK BEFORE PROCEEDING TO THE NEXT LOWER PAIR OF ZONES. FOR EXAMPLE, IF THE ADDRESS RANGE OF THE TEST CONSISTS OF ONE DISK OF IIB2, THE BACKWARD WRITES WOULD COVER THE FOLLOWING ADDRESS LIMITS WITH THE INDICATED ADDRESS INTERLACE:

ADDRESS RANGE	INTERLACE FACTOR

FIRST:	111199 ->89600	216-8 = 208
SECOND:	89599 ->69900	197-8 = 189
	69899 ->52200	177-8 = 169
	52199 ->36400	158-8 = 150
	36399 ->22400	139-8 = 131
	22399 ->10300	121-8 = 113
LAST:	10299 ->0	103-8 = 95

3. OVERLAPPED VARIABLE-LENGTH WRITE/READ (TESTS 12 AND 13).

THE DISK ADDRESSES AND I/O LENGTHS USED BY THESE TESTS ARE DESCRIBED IN THE TEST12 SECTION BELOW. THE I/O OPERATIONS ALTERNATE BETWEEN WRITE AND READ; ADDRESS INTERLACE IS NOT USED.

4. SEGMENT-5 DATA TRANSFER (TESTS 14, 15, 16)

THE ADDRESS SEQUENCE FOR TESTS 14 AND 16 IS DISCUSSED IN THE SECTION DESCRIBING TEST 14; TEST 15 IS UNIQUE.

5. INCREASING MAINTENANCE SEGMENT ADDRESSES (TESTS 1, 4, 5 AND 6)

THE MAINTENANCE SEGMENTS ARE ACCESSED IN A NATURAL ORDER. ADDRESS INTERLACE IS NOT POSSIBLE WITH MAINTENANCE SEGMENTS; THE ADDRESS PATTERN IS NOT CONSIDERED TO BE A SIGNIFICANT TEST FACTOR.

ERROR DETECTION

THE DISK VERIFY TESTS CONTAIN PROVISIONS FOR DETECTING BOTH LONG AND SHORT READS AND WRITES. THESE PROVISIONS CONSIST OF:

1. PLACING A WORD OF ALL-ONES WITH A TAG OF 1 IMMEDIATELY AFTER THE LAST WORD THAT SHOULD BE TRANSFERRED INTO AN INPUT BUFFER, AND SPECIFYING MEMORY-PROTECT IN THE READ IOCW-S. A LONG READ WILL THEREFORE APPEAR AS A MEMORY PROTECT VIOLATION IN THE RESULT DESCRIPTOR.

2. FILLING THE INPUT BUFFER WITH THE BIT COMPLEMENT OF THE EXPECTED DATA BEFORE INITIATING A READ FOR THOSE TESTS WHICH USE THE WORST-CASE PATTERN, AND FILLING WITH OPERAND ZEROES FOR

THE TESTS WHICH TRANSFER RANDOM OR PROGRAM-CODE DATA. THE "DATA READ" PORTION OF THE ERROR PRINTOUT WILL THUS BE EITHER THE BIT COMPLE- MENT OF THE "DATA EXPECTED" OR ALL ZEROES IF THE MULTIPLEXOR HAS NOT TRANSFERRED ENOUGH DATA.

3. USING THE BACKWARD ADDRESSING DESCRIBED IN SECTION 2 TO DETECT LONG WRITES.

4. ALLOWING THE USER TO SPECIFY "SELECT TEST 8, TEST 10" IN ORDER TO INITIALIZE THE DISK WITH THE WORST-CASE PATTERN VIA TEST 8 AND THEN ATTEMPTING TO REWRITE THE SAME DISK AREA USING THE RANDOM DATA IN TEST 10. THIS SEQUENCE WILL DETECT SHORT WRITES(I.E., TOO LITTLE DATA TRANSFERRED TO DISK BY TEST 10).

ERROR HANDLING

THE DETECTION OF A RESULT DESCRIPTOR OR DATA COMPARISON ERROR WILL CAUSE CERTAIN HARDWARE INFORMATION TO APPEAR ON THE OUTPUT MEDIUM. THE FAILING OPERATION WILL BE RETRIED A MAXIMUM OF TEN TIMES. IF THE TENTH ATTEMPT IS NOT SUCCESSFUL, THESE RETRIES WILL BE TERMINATED; AND THE CURRENT TEST WILL CONTINUE WITH THE NEXT DISK ADDRESS.

THE ERROR PRINTOUT CONSISTS OF THE FOLLOWING INFORMATION:

1. THE TEST NUMBER, A PROSE DESCRIPTION OF THE OPERATION SPECIFIED BY THE IOCW (E.G., "WRITE TO NORMAL SEGMENTS"), THE RETRY NUMBER FOR THIS OPERATION AND THE CUMULATIVE I/O COUNT.

2. THE IOCW, AREA DESCRIPTOR AND RESULT DESCRIPTOR, IN HEX.

3. A PROSE DESCRIPTION OF THE RESULT DESCRIPTOR ERRORS, IF THERE ARE ANY.

4. A BREAKDOWN OF THE DISK ADDRESS INTO STORAGE UNIT, DISK FACE, ZONE AND RELATIVE SEGMENT. THE HEADING INFORMATION PRINTED OUT BEFORE THE EXECUTION OF THE FIRST VERIFY TEST INCLUDES AN ADDRESS BREAKDOWN FOR THE SMALLEST AND LARGEST ADDRESS FOR THIS TYPE OF ELECTRONICS UNIT. THESE ADDRESSES BEAR NO RELATION TO THE DISK AREA BEING TESTED; THEY ARE

INCLUDED IN THE PRINTOUT TO ALLOW THE FE TO DEDUCE WHAT ORIGIN AND LIMIT VALUES WILL BE USED FOR THE COMPONENTS OF THE DISK ADDRESS IN SUBSEQUENT ERROR PRINTOUT.

FOR EXAMPLE, AN IA2 STORAGE UNIT WILL HAVE ADDRESS BREAKDOWN PRINTOUT FOR ADDRESSES 0 AND 301999. THE ASSOCIATION OF ADDRESS 301999 WITH FACE 7 IS A HINT THAT THE FACE NUMBER REFERS TO PHYSICAL DISK FACE; SIMILARLY, THE ASSOCIATION OF ADDRESS 301999 WITH RELATIVE SEGMENT 150 SHOWS THAT THIS RELATIVE SEGMENT IS MEASURED FROM THE BEGINNING OF LOGICAL TRACK 99 IN ZONE 1 RATHER THAN THE BEGINNING OF PHYSICAL TRACK 99 IN ZONE 3.

APPENDIX CONTAINS ADDRESS TABLES WHICH USE THE SAME NUMBERING CONVENTIONS AS THE ERROR PRINTOUT FROM THE VERIFY TESTS.

5. UP TO THIRTY WORDS OF DATA FOR READ OPERATIONS, IN "CORRECT DATA"/ "DATA READ" FORMAT WITH ASTERISKS BENEATH UNEQUAL 4-BIT CHARACTERS. FOR I/O LENGTHS WHICH EXCEED THIRTY WORDS (E.G., FROM TESTS 12 OR 14), THERE MAY BE "GAPS" IN THE DATA DISPLAYED. IN ALL SUCH CASES, THE DATA NOT SHOWN DID NOT CONTAIN ANY COMPARISON ERRORS. IF THERE ARE DATA COMPARISON ERRORS BEYOND THE THIRTIETH WORD SHOWN, THESE UNEQUAL WORDS ARE COUNTED AND THE COUNT IS PRINTED BUT THE DATA ITSELF IS NOT SHOWN.

TEST DESCRIPTIONS

THE FOLLOWING PARAGRAPHS PROVIDE A MORE DETAILED DESCRIPTION OF THE PROGRAM LOGIC ASSOCIATED WITH THE INDIVIDUAL DISK CONFIDENCE TESTS.

TEST1

TEST FUNCTIONS: MAINTENANCE SEGMENT READ CHECK USING INCREASING DISK ADRESSES.

THE ADDRESSES USED FOR THIS TEST ARE THOSE OF THE MAINTENANCE SEGMENTS BELONGING TO THE TRACKS BETWEEN THE FIRST AND LAST DISK ADDRESS IN THE SPECIFIED RANGE.

THE DISK ADDRESS USED TO REFERENCE THESE MAINTENANCE SEGMENTS,

WHICH WILL ALSO APPEAR IN THE ADDRESS-BREAKDOWN PRINTOUT IF AN ERROR IS ENCOUNTERED, IS THE ADDRESS OF THE FIRST SEGMENT IN THE TRACK AND ZONE WHICH CONTAIN THE MAINTENANCE SEGMENT.

TEST 2

TEST FUNCTIONS: TWO-SEGMENT NORMAL-SEGMENT WRITE OF WORST-CASE DATA TO INCREASING DISK ADDRESSES.

REQUIREMENTS: BADDISK FILE OF AT LEAST FOUR SEGMENTS.

THE ENTIRE FILE AREA IS WRITTEN WITH SIXTY-WORD RECORDS CONSISTING OF:

1. ONE WORD OF ADDRESS KEY

2. TWENTY-NINE WORDS OF THE WORST-CASE DATA PATTERN

3. THE ONE-WORD ADDRESS KEY FOR THE SECOND SEGMENT

4. TWENTY-NINE WORDS OF THE BIT-COMPLEMENT OF THE WORST-CASE DATA PATTERN.

AT THE COMPLETION OF THESE WRITES, THE INDIVIDUAL SEGMENTS ARE READ BACK AND CHECKED USING AN I/O LENGTH OF 30 WORDS.

THE BEGINNING ADDRESS IS THEN INCREMENTED BY ONE SEGMENT AND THE ENTIRE OPERATION IS REPEATED. THIS SECOND PASS INSURES THAT EVERY SEGMENT CROSSOVER WITHIN THE RANGE OF THE FILE IS COVERED AND THAT EVERY SEGMENT (EXCEPT THE FIRST AND LAST, IF THE FILE CONTAINS AN ODD NUMBER OF SEGMENTS) IS WRITTEN WITH BOTH THE WORST-CASE PATTERN AND ITS BIT COMPLEMENT; THUS, EACH BIT POSITION ON THE DISK SURFACE IN THE LAST TWENTY-NINE WORDS OF EACH SEGMENT IS WRITTEN WITH BOTH A ONE BIT AND A ZERO BIT: THIS TEST IS THE MOST THOROUGH OF THE VERIFY ROUTINES FOR DETECTING DATA-RECORDING ERRORS.

THE WRITE PHASE USES AN ADDRESS INTERLACE OF TEN SEGMENTS AND THE READ PORTION USES AN INTERLACE OF EIGHT SEGMENTS SO THAT A TOTAL OF TWENTY-SIX REVOLUTIONS PER PHYSICAL TRACK IS REQUIRED TO COMPLETE THIS TEST SEQUENCE.

TEST3

TEST FUNCTIONS: READ-CHECK FROM NORMAL SEGMENTS USING INCREASING DISK ADDRESSES.

TEST3 USES AN ADDRESS INTERLACE OF EIGHT SEGMENTS TO PERFORM A READ CHECK OPERATION TO INCREASING NORMAL SEGMENTS. THE PRIMARY VIRTUE OF TEST3 IS THAT IT DOES NOT REQUIRE A BADDISK FILE: ABSOLUTE DISK ADDRESSES ("UNIT DKX, SEGMENT Y THRU Z") MAY ALSO BE USED.

TEST4

TEST FUNCTIONS: MAINTENANCE SEGMENT WRITE OF WORST-CASE DATA TO INCREASING DISK ADDRESSES.

THE TEST WRITES THE WORST-CASE DATA PATTERN TO ALL MAINTENANCE SEGMENTS IN THE ADDRESS RANGE. AT THE CONCLUSION OF THE WRITE PHASE, THE SAME MAINTENANCE SEGMENTS ARE READ BACK AND COMPARED WITH THE DATA WRITTEN. RESULT DESCRIPTOR ERRORS ARE REPORTED FOR BOTH WRITE AND READ OPERATIONS, AND DATA COMPARISON ERRORS ARE DISPLAYED FOR SEGMENTS WHICH WERE NOT READ BACK CORRECTLY.

TEST5

TEST FUNCTIONS: MAINTENANCE SEGMENT WRITE OF WORST-CASE DATA TO DECREASING DISK ADDRESSES.

TEST5 DIFFERS FROM TEST4 ONLY IN ITS USE OF DECREASING MAINTENANCE SEGMENT ADDRESSES. THIS TEST WILL BE REPLACED IN THE NEAR FUTURE, SINCE THE BACKWARD ADDRESSING IS OF QUESTIONABLE VALUE WHEN APPLIED TO MAINTENANCE SEGMENTS.

TEST6

TEST FUNCTIONS: MAINTENANCE SEGMENT WRITE OF RANDOM DATA TO INCREASING DISK ADDRESSES.

THE MCP POWERS-OF-TEN ARRAY (POTM) IS WRITTEN TO INCREASING MAINTENANCE SEGMENT ADDRESSES. AT THE CONCLUSION OF THE WRITES, EACH MAINTENANCE SEGMENT IS READ BACK AND COMPARED WITH

THE DATA WRITTEN. THE ONLY DIFFERENCE BETWEEN TESTS 4 AND 6 IS THE DATA WRITTEN TO THE MAINTENANCE SEGMENTS.

TEST7

TEST FUNCTIONS: MAINTENANCE SEGMENT WRITE OF WORST-CASE DATA TO DECREASING DISK ADDRESSES.

TEST7 IS A DUPLICATE OF TEST5. THE BACKWARD ADDRESSING IN MAINTENANCE SEGMENTS IS ALSO OF DUBIOUS MERIT IN TEST7.

TEST8

TEST FUNCTIONS: NORMAL SEGMENT WRITE OF WORST-CASE DATA TO INCREASING DISK ADDRESSES.

KEYED THIRTY-WORD RECORDS OF WORST-CASE DATA ARE FIRST WRITTEN TO ALL NORMAL SEGMENTS IN THE SPECIFIED BADDISK AREA USING AN ADDRESS INTERLACE OF EIGHT SEGMENTS. THESE SEGMENTS ARE THEN READ AND COMPARED WITH THE DATA WRITTEN.

THE FUNCTIONS CHECKED BY TEST8 ARE ALSO CHECKED AS A BY-PRODUCT OF TEST2-S SEGMENT CROSSOVER CHECKING. THUS, IF TEST2 HAS BEEN RUN, THERE IS LITTLE TO BE LEARNED FROM ALSO EXECUTING TEST8.

TEST9

TEST FUNCTIONS: WRITE/READ OF WORST-CASE DATA PATTERN; DECREASING NORMAL-SEGMENT ADDRESSES.

KEYED THIRTY-WORD RECORDS OF THE WORST-CASE DATA PATTERN ARE WRITTEN TO ALL SEGMENTS IN THE ADDRESS RANGE; AT THE CONCLUSION OF THE WRITE OPERATIONS, EACH SEGMENT IS READ BACK AND CHECKED. BOTH THE WRITE AND READ PORTIONS OF TEST9 USE THE BACKWARD-ADDRESSING ADDRESS INTERLACE DISCUSSED IN THE "DISK ADDRESS SEQUENCE" SECTION.

TEST10

TEST FUNCTIONS: WRITE/READ RANDOM DATA PATTERN; INCREASING

NORMAL SEGMENT ADDRESSES.

KEYED THIRTY-WORD SEGMENTS OF THE POWERS-OF-TEN "RANDOM" DATA ARE WRITTEN TO ALL NORMAL SEGMENTS IN THE SPECIFIED PORTION OF THE BADDISK FILE.

AT THE CONCLUSION OF THE WRITE PHASE, EACH SEGMENT IS READ BACK AND CHECKED. THE NORMAL SEGMENTS ARE ACCESSED IN THE FORWARD DIRECTION BY BOTH THE WRITE AND READ PORTIONS OF TEST10; AN ADDRESS INTERLACE OF EIGHT IS USED TO MINIMIZE DISK LATENCY.

TEST11

TEST FUNCTIONS: WRITE/READ RANDOM DATA PATTERN, DECREASING NORMAL SEGMENTS.

TEST11 DIFFERS FROM TEST10 ONLY IN THE DISK ADDRESSING SEQUENCE. THE ELAPSED TIME FOR TEST11 IS NEARLY TWICE AS GREAT AS FOR TEST10 SO THAT TEST11 IS RECOMMENDED ONLY IF THERE IS REASON TO SUSPECT THAT THE AMOUNT OF DATA TRANSFERRED TO DISK IS NOT CORRECT.

TEST12

TEST FUNCTIONS: DETECT DATA-RECORDING ERRORS CAUSED BY PEAK-SHIFTING.

<SYSTEM NOTE SCR #255 IN DOCUMENTATION SECTION OF II.2 SYSTEM NOTES, SECTION III SHOULD BE COPIED HERE, WITH CORRECTIONS.>

TEST13

TEST FUNCTIONS: DETECT DATA-RECORDING ERRORS ON LONG WRITES AND READS.

TEST13 USES THE WORST-CASE DATA PATTERN AND ITS BIT COMPLEMENT IN CONJUNCTION WITH THE ADDRESSING SCHEME OF TEST12 TO WRITE AND READ FROM THREE TO SIXTY SEGMENTS. THE VARIABLE LENGTH OF THE WRITES AND READS PRECLUDES USE OF ADDRESS INTERLACE; THE ADDRESSING PATTERN CAN BE SEEN FROM THE PRINTOUT OF THE ALGOL

PROGRAM IN THE TEST12 SECTION, WITH A "0" REPRESENTING A SEGMENT OF THE WORST-CASE DATA PATTERN AND A "1" REPRESENTING THE BIT COMPLEMENT OF THIS DATA.

THE ADDRESS KEY IS PLACED IN ONLY THE FIRST WORD OF THE FIRST SEGMENT OF EACH I/O IN BOTH TEST 12 AND 13.

TEST14

TEST FUNCTIONS: MULTIPLEXOR TAG-TRANSFER AND DISK DATA RECORDING.

THE MCP NON-OVERLAYABLE ("SAVE") PROGRAM CODE IS WRITTEN TO DISK BY PLACING A WRITE-IOCW IN ONE OF FIVE WORDS BELONGING TO AN ARRAY WHOSE MEMORY SPACE IS LOCATED A SHORT DISTANCE BELOW THIS CODE. AT THE CONCLUSION OF EACH WRITE, THE DATA IS READ BACK INTO A DIFFERENCE MEMORY AREA, AND THE TAG-3 PROGRAM CODE PORTION OF THE DATA IS COMPARED WITH THE WRITTEN DATA. THIS WRITE/READ/COMPARE CYCLE IS REPEATED UNTIL THE NEXT WRITE WOULD EXCEED THE BOUNDS OF THE FILE. AT THIS POINT, ANOTHER OF THE FIVE IOCW-ARRAY WORDS IS OBTAINED, AND THE ABOVE SEQUENCE IS REPEATED. THIS USE OF FIVE DIFFERENT IOCW LOCATIONS HAS THE EFFECT OF SHIFTING THE DATA ON THE DISK SURFACE TO FIVE DIFFERENT POSITIONS AND, THUS, PROVIDES REDUNDANCY IN CHECKING ANY GIVEN DISK AREA WITH DIFFERENT BIT PATTERNS.

IF MORE THAN FIVE INDEPENDENT EXECUTIONS OF MAINTENANCE ARE IN THE MACHINE SIMULTANEOUSLY, A MAXIMUM OF FIVE OF THEM MAY BE EXECUTING TEST14 OR TEST16 SIMULTANEOUSLY. ANY OTHER TEST14/ TEST16 VERIFY DISK JOBS WILL BE SUSPENDED UNTIL ONE OF ITS PREDECESSORS MAKES AN IOCW-ARRAY SLOT AVAILABLE.

THE MEMORY AREA OCCUPIED BY THE DATA IS APPROXIMATELY 7100 WORDS, WHICH OCCUPIES (7100 X 64/48)/30 = 315 SEGMENTS + 16 WORDS WHEN WRITTEN WITH TAG TRANSFER. THE DISK-ADDRESS INCREMENTING WILL MAKE THE NEXT WRITE BEGIN AT THE FIRST SEGMENT WHICH WAS NOT WRITTEN WITH A FULL THIRTY WORDS BY THE LAST WRITE. IF THE FILE DOES NOT CONSIST OF A MULTIPLE OF 316 SEGMENTS, THEN THE LAST ONE TO 314 SEGMENTS WILL NOT BE

REFERENCED.

TEST15

TEST FUNCTIONS: CHARACTER-ORIENTED I/O LENGTHS.

THE DISK ADDRESS REFERENCED BY THE IOCW FOR TEST15 OPERATIONS IS THE FIRST SEGMENT OF THE BADDISK AREA FOR ALL WRITE AND READ OPERATIONS. THE COMBINATION OF TAG-TRANSFER IOCW OPERATION AND THE AREA DESCRIPTOR LENGTHS FOR SUCCESSIVE OPERATIONS ARE DETERMINED IN SUCH A MANNER THAT THE LENGTH OF THE DATA ON THE DISK SURFACE IS USUALLY NOT AN INTEGRAL NUMBER OF 48-BIT WORDS.

THIS CHARACTER-ORIENTED I/O LENGTH SEQUENCE IS GENERATED BY USING LENGTHS OF THREE AND 23 WORDS FOR THE FIRST TWO WRITE/READ CYCLES AND BY COMPUTING ALL SUCCESSIVE LENGTHS AS THE SUM OF THE PRECEDING TWO LENGTHS MOD 991. THE FOLLOWING TABLE ILLUSTRATES THE FIRST TWELVE OF THE SIXTY I/O LENGTHS USED:

AREA DESCRIPTOR LENGTH IN WORDS	8-BIT CHARACTERS ON DISK	SEGS+CHARS ON DISK
3	24	0S+ 24C
23	184	1S+ 4C
26	208	1S+ 28C
49	392	2S+ 32C
75	600	3S+ 60C
124	992	5S+ 92C
199	1592	8S+ 152C
323	2584	14S+ 64C
522	4176	23S+ 36C
845	6760	37S+ 100C
1367 MOD 991 = 376	3008	16S+ 128C
2212 MOD 991 = 230	1840	10S+ 40C

THE MOST INTERESTING OF THESE I/O LENGTHS IS THE SECOND (23-WORD) ENTRY. AN ERROR PRINTOUT FOR THIS PARTICULAR I/O LENGTH WHICH SHOWS THAT THE LAST FOUR CHARACTERS WERE NOT TRANSFERRED TO DISK ON THE WRITE CYCLE MAY MEAN THAT 3117057A HAS NOT BEEN

INSTALLED ON A IIB DISK UNIT.

TEST16

TEST FUNCTIONS: FORCE-TAG-3 MULTIPLEXOR OPERATION AND DISK DATA RECORDING.

THE PROGRAM LOGIC OF TEST16 IS THE SAME AS THAT FOR TEST14 WITH THE FOLLOWING EXCEPTIONS:

1. THE IOCW USED TO WRITE THE CODE TO DISK DOES NOT REQUEST TAG-TRANSFER, AND

2. THE IOCW USED TO READ THE CODE BACK REQUESTS "FORCE PROGRAM TAGS" RATHER THAN "TRANSFER TAGS".

THE LENGTH OF THE DATA ON DISK IS 3/4 THAT OF TEST14 SO THAT SEVERAL MORE WRITE/READ CYCLES MAY BE REQUIRED. THE COMPENSATING ADVANTAGE IS THAT THE SIXTEEN-BIT DISK AREAS USED TO STORE THE TAGS IN TEST14 ARE OCCUPIED BY PROGRAM DATA AND CONSEQUENTLY WILL BE TESTED MORE THOROUGHLY BY THIS ROUTINE.

THE VERIFY DISK TESTS RESPOND TO A SPO INPUT OF <MIX NUMBER> HI BY DISPLAYING THE NUMBER OF THE TEST CURRENTLY RUNNING AND THE EU NUMBER AND BEGINNING ADDRESS OF THE AREA UNDER TEST.

EXAMPLES

ADDRESS-RANGE AND TEST SELECTION SYNTAX:

1. FILE BD = "BADDISK/EU033AD0000111100"; COMMENT IC4 UNIT; VERIFY DISK FILE BD (SEGMENT 111123 FOR 25, SELECT TEST2);

THE ADDRESS RANGE COVERED BY EXAMPLE 1 IS 111123 THROUGH 111147 (25 SEGMENTS) WHICH IS WITHIN THE BOUNDS OF THE BADDISK FILE. THEREFORE, THE WRITE PORTION OF TEST2 IS EXECUTABLE; AND THE TEST WILL BE RUN.

2. FILE BD = "BADDISK/EU033AD0000111100"; VERIFY DISK FILE BD (OFFSET 10 FOR 20, SELECT ALL EXCEPT TEST2);

THE BEGINNING AND ENDING ADDRESSES WILL BE 111110 THROUGH

111129 (20 SEGMENTS). THE ADDRESS RANGE IS WITHIN THE BOUNDS OF THE BADDISK FILE SO THAT ALL TESTS EXCEPT TEST2 WILL BE EXECUTED.

3. UNIT DKX = DK33; VERIFY DISK UNIT DKX (SEGMENT 4000 FOR 200, SELECT ALL EXCEPT TEST2, TEST4);

THE ADDRESS RANGE WILL BE SEGMENTS 4000 THROUGH 40199 (200 SEGMENTS); THE ABSENCE OF A BADDISK FILE SPECIFIER MEANS THAT TESTS WHICH WRITE TO NORMAL SEGMENTS WILL NOT BE RUN. THE "EXCEPT" CLAUSE WILL BY-PASS THE INVOCATION OF THE ROUTINES FOR TEST2 AND TEST4.

4. UNIIT DKX = DK33; VERIFY DISK UNIT DKX (SEGMENT 4000 THRU 40123, SELECT TEST4, TEST5, TEST6);

THE ADDRESS RANGE COVERED BY THIS VERIFY CONSISTS OF SEGMENTS 4000 THROUGH 40123 (124 SEGMENTS); THE THREE SELECTED TESTS ARE ALL CONFINED TO MAINTENANCE SEGMENTS SO THAT ALL THREE WILL BE EXECUTED.

5. VERIFY DISK DK33 (SEGMENT 20304 THRU 30405, SELECT ALL EXCEPT TEST15);

ONLY READ-CHECK AND MAINTENANCE-SEGMENT TESTS WILL BE EXECUTED IN THE 10102-SEGMENT AREA. THE EXPLICIT EXCLUSION OF TEST15 IS NOT NECESSARY BECAUSE TEST15 REQUIRES A BADDISK FILE.

APPENDIX A

THE BREAKDOWN OF DISK ADDRESSES INTO STORAGE UNIT, DISK, FACE, ZONE, AND TRACK IS COMPLICATED BY THE NON-UNIFORM CHOICE OF LOWER LIMITS FOR THESE COMPONENTS IN VARIOUS FETM-S. THE TABLES ON THE FOLLOWING PAGES USE THE SAME BASE AND LIMIT VALUES AS THE SCR ADDRESS BREAKDOWN AND CAN BE RELATED TO THE FETM NUMBERING CONVENTIONS BY MEANS OF THE FOLLOWING TABLE.

IA2 STORAGE UNITS	SU	DISK	FACE*	ZONE
SCR	0-2	1-4	1-8	1-3
DF CONTROL IV, FORM 1045028	0-4	---	0-7	0-2

IA2 STORAGE UNITS	SU	DISK	FACE*	ZONE
DFEU, FORM 1031432	0-4	1-4	1-8	1-3

* PHYSICAL FACE. FACES ARE SELECTED IN THE ORDER 1, 4, 2, 3, 5, 8, 6, 7. SEE FORM 1031432, PAGE II45.

IC3/IC4 STORAGE UNITS					
SCR	0-4	1-4	1-8		1-3
DF CONTROL IV, FORM 1045028	0-4	---	0-4		0-2
DFEU, FORM 1038866	1-5		1-8		1-3
DFSU, FORM 1038874	1-5	1-4	1-8		1-3

IIB2/IIB4 STORAGE UNITS					
SCR	1-5	1-2	1-2	0-6	00-99
MODEL II SUBSYSTEM,FORM 1047339	1-5	A-B	L-U	0-6	00-99
DF CONTROL V,FORM 1051695	1-5	0-9	1-2	0-6	00-49

IIB6 STORAGE UNITS					
SCR	1-5	1-2	1-2	0-5	00-99
MODEL II SUBSYSTEM,FORM 1047339	1-5	A-B	L-U	0-5	00-99

D0122 COBOL - COBOL SEGMENTATION - 08-18-72

AUTOMATIC SEGMENTATION OF SECTIONS MAY BE OVERRIDDEN BY SELECTIVELY COMBINING TWO OR MORE SECTIONS INTO THE SAME CODE SEGMENT BY SPECIFYING PRIORITY NUMBERS ON SECTION HEADERS AND SETTING THE DOLLAR OPTION "SECGROUP".

IF "SECGROUP" IS NOT SET, THE AUTOMATIC SEGMENTATION OF EACH SECTION WILL NOT BE AFFECTED, REGARDLESS OF ANY SPECIFICATION OF PRIORITY NUMBERS. THAT IS, SEGMENTATION WILL OCCUR AS IN THE PAST.

WHEN "SECGROUP" IS SET, CONTIGUOUS NON-DECLARATIVE SECTIONS HAVING PRIORITY NUMBERS GREATER THAN OR EQUAL TO THE PREVIOUS SECTION-S PRIORITY NUMBER WILL BE COMBINED WITH THE PREVIOUS SECTION IN THE SAME CODE SEGMENT. A SECTION HAVING NO PRIORITY NUMBER HAS A DEFAULT PRIORITY OF ZERO. SEVERAL RULES QUALIFY THIS ACTION:

1. DECLARATIVE SECTIONS MAY NOT BE COMBINED;

2. THE FIRST SECTION IN A SORT INPUT OR OUTPUT PROCEDURE UNCONDITIONALLY STARTS A NEW SEGMENT. THE LAST SECTION IN A SORT INPUT OR OUTPUT PROCEDURE UNCONDITIONALLY ENDS A SEGMENT. ALL SECTIONS IN THE RANGE OF SECTIONS NAMED AS AN IPUT OR OUTPUT PROCEDURE MAY BE COMBINED;

3. HOWEVER, SECTIONS HAVING NO PRIORITY NUMBER WILL START A NEW SEGMENT;

4. THE DISCRETION OF THE SEGMENT-LIMIT CLAUSE REMAINS IN EFFECT.

D0123 RESEQBASIC - 09-05-72

SYSTEM/RESEQBASIC IS AN ALGOL PROGRAM DESIGNED TO RESEQUENCE ALL OR PART OF A BASIC FILE, MAKING THE NECESSARY CHANGES TO THE SEQUENCE NUMBERS THAT OCCUR AS PART OF THE SYMBOLIC TEXT. SYSTEM/CANDE EXECUTES THIS PROGRAM WHEN THE VERB, RESEQ, IS REQUESTED ON A TYPE BASIC FILE; IT MAY ALSO BE RUN THROUGH THE BATCH MODE.

FOUR PARAMETERS ARE NEEDED TO EXECUTE THIS PROGRAM. THEY ARE:

1. LOWEST SEQUENCE NUMBER TO BE RESEQUENCE. SPECIFY AS 0 TO RESEQUENCE THE ENTIRE FILE.

2. HIGHEST SEQUENCE NUMBER TO BE RESEQUENCED. SPECIFY AS 99999999 TO RESEQUENCE THE ENTIRE FILE.

3. BASE FOR THE NEW SEQUENCE RANGE. MUST BE BETWEEN 1 AND 9999.

4. INCREMENT FOR THE NEW SEQUENCE RANGE. MUST BE BETWEEN 1 AND 9999.

THE SOURCE FILE, TAPE, SHOULD BE LABEL-EQUATED TO THE DISK FILE TO BE RESEQUENCED. (THIS MAY BE A B5500 BASIC FILE.) THE RESEQUENCED FILE, NEWTAPE, WILL BE CREATED AS AN EBCDIC DISK FILE; IT SHOULD ALSO BE LABEL-EQUATED TO A NEW TITLE.

NEWTAPE WILL UNCONDITIONALLY CONTAIN SEQUENCE NUMBERS IN COLUMNS 1-4, MAKING IT AVAILABLE FOR EDITING THROUGH SYSTEM/CANDE. SEQUENCE

NUMBERS WILL ALSO BE PLACED IN COLUMNS 73-80 IN NON-CANDE EXECUTIONS.

IF A FATAL ERROR OCCURS, THE FILE WILL NOT BE RESEQUENCED, AND NEWTAPE WILL NOT BE LOCKED. APPROPRIATE MESSAGES WILL BE PROVIDED.

IF A NON-FATAL WARNING OCCURS, THE FILE WILL BE RESEQUENCED, AND A LIST OF WARNINGS WILL BE PROVIDED.

EXAMPLE OF A SYSTEM/RESEQBASIC EXECUTION DECK:

```
<I> EXECUTE SYSTEM/RESEQBASIC (0,99999999,100,10)
<I> FILE TAPE = BASIC/JOB
<I> FILE  NEWTAPE = RESEQ/JOB
<I> END
```

RESEQUENCE THE COMPLETE SYMBOLIC BASIC/JOB WITH A BASE OF 100 AND AN INCREMENT OF 10. THE RESEQUENCED FILE IS TITLED RESEQ/JOB.

D0126 TC500-CANDE INTERFACE PROGRAM - 08-25-72

THIS SYSTEM NOTE HAS BEEN DELETED.

D0127 CANDEFILES - 09-05-72

SYSTEM/CANDEFILES IS A UTILITY PROGRAM INTENDED TO ASSIST INSTALLATIONS IN CONVERTING TO THE NEW MARK II.3 SYSTEM/CANDE. THE TWO FUNCTIONS PERFORMED ARE:

CREATE A <USERCODE>/<PASSWORD> ENTRY IN THE B6700 SECURITY SYSTEM WHICH WILL BE USED BY SYSTEM/CANDE TO IDENTIFY A POTENTIAL USER AND TO MAINTAIN HIS LIBRARY OF FILES.

CONVERT ANY EXISTING MARK II.2 SYSTEM/CANDE FILES TO THE NEW MARK II.3 SYSTEM/CANDE NAMING CONVENTION.

THE PROGRAM INPUT CONSISTS OF A DECK OF DATA CARDS EACH OF WHICH CONTAIN A <USERCODE> OR <USERCODE>/<PASSWORD> THAT IDENTIFIES A USER. THE <USERCODE> AND <PASSWORD> ARE IN THE FORM OF FILE IDENTIFIERS SEPARATED BY A SLASH; THE COMPLETE STRING MUST BE

TERMINATED BY A PERIOD.

FOR EACH <USERCODE>/<PASSWORD> ENTERED, AN ENTRY WILL BE MADE IN THE B6700 SECURITY SYSTEM FOR USER IDENTIFICATION.

IF A DIRECTORY TITLED <USERCODE> EXISTS ON DISK, THEN ALL FILES CONTAINED IN THAT DIRECTORY WILL BE CHANGED TO AGREE WITH THE NEW SYSTEM/CANDE NAMING CONVENTION. IF THE INPUT <USERCODE> IS SPECIFIED AS "PREFIX" THEN THE CHANGES ARE AS FOLLOWS:

OLD FILE NAME	NEW FILE NAME
PREFIX/<FILENAME>/SOURCE	(PREFIX) <FILENAME>
PREFIX/<FILENAME>/OBJECT	(PREFIX) OBJECT/<FILENAME>
PREFIX/<FILENAME1>/<FILENAME2>	(PREFIX) <FILENAME1>/<FILENAME2>

WHERE (PREFIX) IS THE STANDARD B6700 SECURITY NOTATION FOR USERCODE/PREFIX.

A PROGRAM OPTION IS PROVIDED VIA THE CONTROL STATEMENT "VALUE" TO SKIP OR PERFORM THE NAME CHANGE FOR FILES TITLED PREFIX/<FILENAME1>/<FILENAME2>. IF VALUE IS SPECIFIED AS ONE (1) THEN THE CHANGE IS PERFORMED; IF VALUE IS SET TO ZERO (OR NOT INCLUDED IN THE EXECUTION DECK), THE CHANGE IS NOT PERFORMED.

OUTPUT TO INDICATE THE DISPOSITION OF ANY FILES IN THE NAMED DIRECTORY IS PROVIDED.

NOTE:

IT IS NECESSARY TO EXECUTE THIS PROGRAM UNDER A PRIVILEGED USERCODE AND WITHOUT AN ASSOCIATED PASSWORD.

EXAMPLE:

```
<I> USER = PRIVILEGED
<I> RUN SYSTEM/CANDEFILES
<I> VALUE = 1
<I> DATA
<I> USER1/PASS1.
<I> PROJECT/CODE.
```

```
<I> DP/DP.
<I> END
```

RUN SYSTEM/CANDEFILES UNDER THE PRIVILEGED USERCODE "PRIVILEGED" WITHOUT PASSWORD, AND CREATE THE <USERCODE>/<PASSWORD>-S SPECIFIED. IF DIRECTORIES NAMED USER1, PROJECT, AND DP EXIST, CHANGE ALL REFERENCED FILES TO THE NEW MARK II.3 NAMING CONVENTION, INCLUDING THE DATA FILES OF FORM USER1/<FILENAME1>/<FILENAME2>.

D0128 BILLING ACCOUNTING LOG - 09-11-72

THIS IS PRELIMINARY DOCUMENTATION OF THE RECORD FORMATS OF THE BILLING LOG (WHICH IS TO BE RELEASED AS PART OF THE WORK-FLOW MANAGEMENT PROJECT OF II.4) AND, THEREFORE, SOME DETAILS ARE SUBJECT TO CHANGE BY THE TIME OF THAT RELEASE.

PROPOSED FORMAT OF THE B6700 BILLING/ACCOUNTING LOG

0. WORK-FLOW MANAGEMENT LOGS

   BILLING
   MAINTENANCE
   JOB FILE

1. GENERAL LOG DESIGN GOALS

2. GENERAL LOG FORMAT

3. SPECIFIC LOGICAL RECORD FORMATS

   3.1 FORMAT RECORDS

   3.2 DATA RECORDS

   3.3 DETAILED FORMATS

      3.3.0 NOISE RECORD

      3.3.1 HALT/LOAD

      3.3.2 TIME/DATE CHANGE

      3.3.3 JOB/TASK INITIATION

INTRODUCTION

THE WORK-FLOW MANAGEMENT RELEASE (II.4) WILL MAKE SEVERAL SIGNIFICANT CHANGES IN THE FORM OF SYSTEM LOGS.

LOGGING UNDER WORK-FLOW MANAGEMENT IS DIVIDED INTO THREE FUNCTIONAL AREAS; MAINTENANCE LOGGING, JOB LOGGING, AND BILLING LOGGING.

THE MAINTENANCE LOG WILL CONTAIN ENTRIES RELATING TO MALFUNCTIONING HARDWARE DEVICES; DEVICE RETRIES, ETC.

THE JOB LOG WILL CONTAIN MESSAGES RESULTING FROM THE RUNNING OF A JOB AND WILL BE USED TO PRODUCE THE JOB SUMMARY. THE BILLING LOG WILL CONTAIN THOSE ITEMS NECESSARY TO ACCURATELY ACCOUNT FOR THE SYSTEM RESOURCES USED BY THE VARIOUS JOBS. IT IS THE PURPOSE OF THIS DOCUMENT TO PRESENT A PRELIMINARY FORM OF THE BILLING LOG.

1. GENERAL LOG DESIGN GOALS

THE PARAMOUNT CONSIDERATION IN DESIGNING THE BILLING/ACCOUNTING LOG IS THE STABILITY IN THE FACE OF ERRORS. TO ACHIEVE THIS STABILITY:

A. FIXED LENGTH RECORDS ARE USED;

B. REDUNDANT INFORMATION IS PROVIDED;

C. FILE RELATIVE LINKS ARE NOT USED.

2. GENERAL LOG FORMAT

THE BILLING/ACCOUNT LOG IS A SERIAL DISK FILE. EACH B/A INFORMATION RECORD IS WRITTEN AS A SERIES OF N WORD GROUPS (N IS TO

BE DETERMINED ON THE BASIS OF EXPERIENCE). EACH SUCH GROUP CONTAINS A GROUP HEADER WORD FOLLOWED BY UP TO N-1 WORDS OF B/A INFORMATION.

THE GROUP HEADER FORMAT IS:

[47:8] GROUP COUNT.

[39:8] OF COUNT.
TOTAL GROUPS -1 IN THIS LOGICAL RECORD. THESE FIELDS GIVE: GROUP COUNT (N-1) OF (M-1).

[31:22] RESERVED FOR FUTURE SYSTEM USE.

[9:10] RECORD TYPE. TYPE OF B/A RECORD.

3. SPECIFIC LOGICAL RECORD FORMATS

IN ORDER TO MAKE THE B/A LOG FORMAT AS FLEXIBLE AS POSSIBLE, IN GENERAL. RECORDS ARE TO BE CONSIDERED AS CONTAINING RELOCATABLE DATA. IN ORDER THAT A SPECIFIC ITEM BE RETRIEVED FROM THE DATA RECORD, IT IS REFERENCED VIA THE CORRESPONDING FORMAT RECORD.

3.1 FORMAT RECORDS

FORMAT RECORDS ARE USED BY A LOG ANALYSIS PROGRAM (AND ARE NOT INCLUDED IN THE LOG ITSELF). EACH FORMAT RECORD CONTAINS INFORMATION NECESSARY TO LOCATE ACTUAL INFORMATION WITHIN THE CORRESPONDING DATA RECORDS; THAT IS, IT CONTAINS A MAP OF THE DATA RECORD. THUS, THE FORMAT RECORD VALUES ARE INDICIES INTO THE DATA RECORD.

ALL DATA RECORDS HAVE A CORRESPONDING FORMAT RECORD.

IF AN ITEM TO BE RETRIEVED FROM A DATA RECORD IS FIXED LENGTH, THE FORMAT RECORD VALUE WILL GIVE THE CORRESPONDING INDEX TO THE ITEM IN THE DATA RECORD. TYPE "WORD" INDICATES FIELD TYPE DATA.

IF A VARIABLE LENGTH ITEM IS TO BE OBTAINED, THE FORMAT VALUE REFERS TO AN ITEM IN THE DATA RECORD WHICH WILL CONTAIN THE INDEX AND LENGTH OF THE VARIABLE LENGTH ITEM (WITHIN THE DATA RECORD).

INDEXF = [19:20] (WORDS)
LENGTHF = [39:20] (WORDS)

3.2 DATA RECORDS

DATA RECORDS CONTAIN THE ACTUAL B/A INFORMATION. WHILE THERE WILL ONLY BE ONE FORMAT RECORD OF A GIVEN TYPE FOR A GIVEN LOG, THERE WILL NORMALLY BE MANY DATA RECORDS OF A PARTICULAR TYPE.

NOTE: THE DATA RECORD FORMATS ARE GENERALLY NOT GIVEN HERE. BY USE OF FORMAT INFORMATION, THE LOCATION OF ITEMS WITHIN A DATA RECORD CAN (AND PROBABLY WILL) FREELY SHIFT AROUND. B/A PROGRAMS SHOULD ALWAYS ACCESS DATA RECORDS VIA THE APPROPRIATE FORMAT INFORMATION.

3.3 DETAILED FORMATS

RECORD DESCRIPTIONS WILL DEFINE THE FORM OF THE FORMAT ENTRIES, THE FORM OF THE CORRESPONDING DATA IN THE DATA RECORD, AND THE DATA RECORD TYPE.

3.3.0 NOISE RECORD

RECORD TYPE 0 IS DEFINED TO BE A NOISE RECORD; I.E., IT CONTAINS NO VALID DATA.

3.3.1 HALT/LOAD

FORMAT TYPE = 1 EMITTED AT LOG CREATION AND DATA TYPE = 1 FOLLOWING H/L-S (CM-S, ETC).

FORMAT RECORD WORD	DATA RECORD

FORMAT RECORD WORD	DATA RECORD
0. TYPE	RECORD TYPE. (NOT AN INDEX) [9:10] RECORD LENGTH (WORDS) [39:20]
1. DATE-TIME INDEX	DATE-TIME (IF LOG ENTRY) WORD IN DATA RECORD. DATUM FORMAT IS (SINGLE WORD) DATE = [47:12]:DATE IN DAYS TIME = [35:36]:TIME OF DAY IN 2.4 US
2. SYSTEM SERIAL NUMBER INDEX	WORD
3. PROC MASK INDEX	WORD: BIT "N" TRUE INDICATES THE PRESENCE OF PROCESSOR N
4. MULTIPLEXOR INDEX	WORD: BIT "N" TRUE INDICATES THE PRESENCE OF MULTIPLEXOR N
5. MEMORY MASK A INDEX	WORD: BITS 0-31 CORRESPOND TO MEMORY MODS 0-31 AND ARE TRUE IF THE CORRESPONDING MOD IS IN USE. (PASSED H-L CONFIDENCE TEST)
6. MEMORY MASK B INDEX	WORD: BITS 0-31 CORRESPOND TO MEMORY MODS 32-63 AND ARE TRUE IF THE CORRESPONDING MOD IS IN USE.
7. MCPID INDEX	WORD: [47:32] MARK LEVEL (EBCDIC) [15:8] LEVEL NUMBER (BINARY) [7:8] PATCH NUMBER (BINARY)

FORMAT RECORD WORD	DATA RECORD
8. MCP NAME INDEX	VARIABLE LENGTH DATA: MCP NAME IN STANDARD FORM
9. INTRINSIC NAME INDEX	VARIABLE LENGTH DATA: INTRINSICS FILE NAME IN STANDARD FORM.

NOTE: FOR ALL DATA RECORDS, THE TYPE AND DATE/TIME ITEMS ARE FIXED WITHIN THE RECORD. IN THE CASE OF A HALT/LOAD DATA RECORD, ITEMS 0 THROUGH 7 WILL BE FIXED WITHIN THE DATA RECORD. THIS, OF COURSE, DOES NOT INVALIDATE USE OF THE CORRESPONDING FORMAT RECORDS.

3.3.2 TIME/DATE CHANGE

FORMAT TYPE = 3

DATA TYPE = 3

FORMAT RECORD WORD	DATA RECORD
0. TYPE	
1. DATE-TIME INDEX	FOR FORMAT, SEE 3.3.1 (AT INSTANT OF LOGGING
2. DATE-TIME INDEX	DATE, TIME CHANGED TO

3.3.3 JOB, TASK INITIATION

FORMAT TYPE = 5

DATA TYPE = 5

FORMAT RECORD WORD	DATA RECORD WORD
0. TYPE	
1. DATA-TIME INDEX	

FORMAT RECORD WORD	DATA RECORD WORD
2. JOB SERIAL NO. INDEX	SINGLE WORD. A SYSTEM SUPPLIED EQUIVALENT TO 3.3.4.9.
3. PARENT SERIAL NO. INDEX	WORD

3.3.4 JOB, TASK TERMINATION

FORMAT TYPE = 7

DATA TYPE = 7

FORMAT RECORD WORD	DATA RECORD
0. TYPE	
1. DATE-TIME INDEX	
2. JOB SERIAL NO. INDEX	WORD:
3. PROCESSOR TIME INDEX	REAL: IN MICROSECS
4. I-O TIME INDEX	REAL: IN MICROSECS
5. SAVE MEMORY SPACE-TIME INTEGRAL INDEX	REAL: IN WORD MICROSECS
6. OVERLAYABLE MEMORY SPACE-TIME INTEGRAL INDEX	REAL IN WORD - MICROSECS
7. END CONDITION (HISTORY WORD)	
8. USER CODE INDEX	VARIABLE LENGTH DATA. THE USER CODE IN STANDARD

FORMAT RECORD WORD	DATA RECORD
	FORM.
9. EXTERNAL NAME INDEX	VARIABLE LENGTH DATA. THE PROGRAMMER SUPPLIED NAME, IN STANDARD FORM.
10. CHARGE CODE INDEX	VARIABLE LENGTH DATA STANDARD FORM.

3.3.5 FILE OPEN

FORMAT TYPE = 9

DATA TYPE = 9

FORMAT RECORD TYPE	DATA RECORD
0. TYPE	
1. DATE-TIME INDEX	
2. JOB SERIAL NO.	
3. CHARGE CODE INDEX	VARIABLE LENGTH DATA. STANDARD FORM.
4. CURRENT FILE SIZE INDEX	REAL: PRIOR TO OPEN - WORDS
5. TITLE INDEX	VARIABLE LENGTH DATA - FORM.
6. OPEN INFO INDEX	WORD: [0:1] TRUE = SUCCESSFUL FALSE = UNSUCCESSFUL [7:6] OPEN MODE; INPUT, OUTPUT, ETC. [15:8] IF [0:1] = FALSE THEN OPEN ERROR HERE. [23:8] FILE TYPE

FORMAT RECORD TYPE	DATA RECORD
7. NAME QUALIFICATION	WORD: [47:10] SAVEFACTOR INDEX
8. BLOCKING INFO	WORD [47:16] WORDS PER BLOCK [8:8] NUMBER OF BUFFERS
9. UNIT INFO	WORD [47:8] PHYSICAL UNIT NO. [39:16] DENSITY-IF APPROPRIATE [8:7] KIND [0:1] SOFTWARE TRANSLATION
10. SERIAL NO.	REAL (IF APPROPRIATE)

3.3.6 FILE CLOSE

FORMAT TYPE = 11

DATA TYPE = 11

FORMAT RECORD WORD	DATA RECORD
0. TYPE	
1. DATE-TIME INDEX	
2. JOB SERIAL NO.	
3. CHARGE CODE INDEX	VARIABLE LENGTH - STANDARD FORM.
4. CURRENT FILE SIZE INDEX	REAL: I WORDS
5. TITLE INDEX	VARIABLE LENGTH - STANDARD FORM.
6. CLOSE INFO INDEX	PURGE, SAVE, ETC. (SIGN CARRIES MEANING)
7. TRANSACTION COUNT	WORD: NUMBER OF I-OS

FORMAT RECORD WORD	DATA RECORD
INDEX	SINCE OPEN
8. I-O TIME INDEX	REAL I-O TIME SINCE OPEN - USECS

APPENDIX A

AN EXAMPLE OF SIMPLE BILLING LOG ANALYSIS:

IN ORDER TO ANALYZE THE LOG, THE LOG IS READ SERIALLY (FORWARD) AND THE LOG RECORDS ARE DEBLOCKED. THE RECORD TYPE AND ITS LENGTH ARE NOW KNOWN.

AT RANDOM INTERVALS, BEGINNING OF JOB (BOJ) RECORDS WILL BE SEEN, INDICATING THE LIKELIHOOD OF FURTHER LOGGING DATA FOR THIS JOB.

AT THIS TIME, SOME FORM OF "ACCUMULATOR" IS OPENED TO BE USED TO COLLECT THE JOB INFORMATION; I.E., EACH JOB MIGHT BE REPRESENTED AS A FILE - OR BETTER YET, AS AN ARRAY ROW.

SINCE EACH RECORD CONTAINS A JOB SERIAL NUMBER, ASSOCIATION WITH THE APPROPRIATE "ACCUMULATOR" IS STRAIGHT FORWARD.

WHEN AN END OF JOB RECORD IS SEEN CORRESPONDING TO AN EXISTING BOJ, ALL INFORMATION HAS BEEN ACCUMULATED FOR THAT JOB AND BILLING MAY TAKE PLACE.

D0129 FORTRAN FORMATTED OUTPUT - 09-11-72

THE NEW FORTRAN OUTPUT FORMATTER HAS A FEW FEATURES AND AREAS OF DIFFERENCE FROM THE CURRENT II.2 FORMATTER. THEY ARE AS FOLLOWS:

1. CARRIAGE CONTROL FEATURE:

THE II.3 FORMATTER WHEN COMPILED WITH THE NEW DOLLAR OPTION "SHIFTFIRSTCHARACTER" SET PRODUCES AN INTRINSIC WHICH BEHAVES EXACTLY AS THE II.2 FORMATTER WITH RESPECT TO CARRIAGE CONTROL

(SEE PAGE 12-18 OF THE FORTRAN REFERENCE MANUAL 5000458 FOR A COMPLETE DISCUSSSION OF II.2 CARRIAGE CONTROL).

THE II.3 FORMATTER WHEN COMPILED WITH THE NEW DOLLAR OPTION RESET (IT IS RESET BY DEFAULT), PRODUCES AN INTRINSIC WITH SLIGHTLY DIFFERENT CARRIAGE CONTROL:

A. THE FIRST CHARACTER OF THE BUFFER WILL NEVER BE PRINTED. HOWEVER, IT WILL BE INTERPRETED EXACTLY AS II.2 WITH RESPECT TO CARRIAGE CONTROL EFFECT.

B. THE II.3 FORMATTER WILL ALWAYS ALLOCATE AN EXTRA CHARACTER OF BUFFER WHEN GOING TO PRINTER OR PRINTER BACKUP. THUS, A 22 WORD EBCDIC BUFFER HAS 133 CHARACTERS. THE FIRST CHARACTER IS NEVER PRINTED, AND THE SECOND THRU 133RD CHARACTERS COMPRISE THE PRINT LINE, STARTING AT PRINT POSITION ONE. THE UNPRINTED FIRST CHARACTER CONTROLS THE PRINTER CARRIAGE IN THE SAME MANNER AS II.2.

2. EW.D AND DW.D DIFFERENCES:

THE II.3 FORMATTER PRODUCES E+ XX AND D+ XX FOR THE EXPONENT PART INSTEAD OF EBXX AND DBXX, WHHEN USING THE E AND D FORMATS, RESPECTIVELY. ALSO, THE ZERO (0) TO THE LEFT OF THE DECIMAL POINT HAS BEEN DELETED, AND THUS THE FIELD WIDTH REQUIREMENTS ARE REDUCED BY ONE CHARACTER. FINALLY, THE ONLY DIFFERENCE BETWEEN EW.D AND DW.D IS THAT ONE USES AN "E" AND THE OTHER, A "D", AND THE EW.D RUNS FASTER FOR SINGLE PRECISION VALUES.

3. DOUBLE PRECISION HANDLED BY ALL FORMATS:

DOUBLE PRECISION VALUES ARE NOW HANDLED CORRECTLY BY ALL FORMAT SPECIFICATIONS. IN PARTICULAR,

A. FOR D.P VALUES EDITED UNDER THE LW SPEC-

IFICATION, THE LOWER ORDER BIT OF THE MORE SIGNIFICANT WORD IS USED TO DETERMINE THE TRUTH VALUE.

B. THE AW AND CW SPECIFIERS CAN NOW EDIT UP TO 12 CHARACTERS USING D.P. VALUES. FOR EXAMPLE:

```
  DOUBLE PRECISION D1
  DATA D1 "ABCDEFGHIJKL"/
10 FORMAT (1X,AW)
  PRINT 10,D1
```

WOULD YIELD

```
  ABCDEFGHIJKL IF W = 12
  ABCDEFGH     IF W = 8
  AB           IF W = 2
     ABCDEFGHIJKL IF W = 15
```

AND THE USE OF CW INSTEAD OF AW IN THE ABOVE WOULD YIELD

```
  ABCDEFGHIJKL IF W = 12
  EFGHIJKL     IF W = 8
  KL           IF W = 8
      ABCDEFGHIJKL IF W = 15
```

D1 IS REPRESENTED INTERNALLY AS:

HIGH HALF	LOW HALF	CHARACTERSIZE
"ABCDEF"	"GHIJKL"	EBCDIC
"00ABCDEF"	"00GHIJKL"	BCL

C. THE OW AND DW SPECIFIERS CAN NOW BE USED TO DISPLAY A D.P. VALUE - BOTH WORK FROM RIGHT TO LEFT ON THE D.P. VALUE. FOR EXAMPLE, O16 D.P. VALUE, WHILE O32 AND Z24 WOULD EDIT D.P. VALUE, WHILE O32 AND Z24 WOULD EDIT THE ENTIRE VALUE.

D. D.P. VALUES EDITED BY THE EW.D AND DW.D SPECIFIERS CAN HAVE 2, 3, OR 5 DIGIT EXPONENTS. THE MINIMUM NUMBER OF DIGITS REQUIRED IS USED. FOR EXAMPLE, THE SPECIFIER E20.10 WOULD YIELD (FOR SUITABLE INTERNAL VALUES):

```
     -.1234567891E-88
      .9876543210+876
     .3141529653E+09132
```

THE SAME HOLDS FOR D20.10.
HOLDS FOR D20.10.

4. -0 DIFFERENCES:

THE SIGN OF A VALUE IS CONSIDERED POSITIVE IF THE MANTISSA PART IS ZERO, REGARDLESS OF THE CONDITION OF THE SIGN BIT. A MINUS 0 MAY BE OUTPUT, E.G., -0.0000, BUT THIS ONLY INDICATES THAT THE FORMAT SPECIFIER DID NOT ALLOW FOR THE DISPLAY OF ALL SIGNIFICANCE, E.G., VALUE MIGHT HAVE BEEN -0.000000001325476981 32.

5. JW DIFFERENCES:

THE NEW JW SPECIFIER IGNORES THE W VALUE, AND OUTPUTS THE VALUE AS AN INTEGER CONSTANT IN THE MINIMUM FIELD NECESSARY. SINGLE OR DOUBLE PRECISION VALUES OF ANY MAGNITUDE WILL BE HANDLED CORRECTLY. FLOATING VALUES ARE ROUNDED.

6. RUN TIME ERROR MESSAGES:

IN ADDITION TO THE END-OF-FILE AND PARITY ERROR CONDITIONS, THERE ARE 14 ERROR CONDITIONS PERTAINING TO THE FORMAT/LIST ELEMENT ITSELF.

ERROR #	ERROR CONDITION
102	FORMAT WAS V SPECIFIER, AND LIST ELEMENT DID NOT PRODUCE AN F,D,E,G,Z,A,O,C,J,I,L,

ERROR #	ERROR CONDITION
	P,X, OR T. NOTE THAT ONLY THE RIGHT MOST CHARACTER OF THE LIST ELEMENT IS USED. THE LIST ELEMENT MUST BE SINGLE PRECISION.
103	FORMAT WAS V SPECIFIER OF THE FORM RV, AND THE RESULTANT SPECIFIER NEEDED A FIELD WIDTH. FOR EXAMPLE, 2V => 2I.
104	FORMAT WAS V SPECIFIER OF THE FORM RV, AND THE RESULTANT SPECIFIER NEEDED A FIELD WIDTH AND DECIMAL PLACES, E.G., 3V => 3E.
105	FORMAT WAS V SPECIFIER IF FORM RVW, AND THE RESULTANT SPECIFIER NEEDED DECIMAL PLACES, E.G., 2V * => 2F6.
106	FORMAT SPECIFIER EVALUATED TO FW.D FORM, AND D < 0
107	FORMAT SPECIFIER EVALUATED TO EW.D OR DW.D, AND D < 1
108	FORMAT WAS V SPECIFIER, AND LIST ELEMENT USED TO EVALUATE SPECIFIER TYPE (A,C,...ETC.) WAS DOUBLE PRECISION. ONLY SINGLE PRECISION IS ALLOWED.
109	FORMAT SPECIFIER EVALUATED TO GW, AND CORRESPONDING LIST ELEMENT WAS NEITHER OF TYPE

ERROR #	ERROR CONDITION
	INTEGER NOR TYPE LOGICAL (EXPRESSIONS OF TYPE INTEGER OR LOGICAL ARE EDITED UNDER GW.D AS IW OR LW, RESPECTIVELY). THEREFORE, THE DECIMAL PLACES IS CONSIDERED MISSING.
110	<UNUSED>
111	FORMAT SPECIFIER EVALUATED TO GW.D AND GW.D LOGIC CHOSE EW.D BUT D < 1.
112	FORMAT STATEMENT HAD NO FORMAT SPECIFIERS REQUIRING LIST ELEMENTS, AND FORMAT WAS USED WITH A LIST.
113	FORMAT SPECIFIER EVALUATED TO EW.D OR DW.D, AND W LEQ D
114	DYNAMIC W OR D PART OF FORMAT SPECIFIER EVALUATED TO A VALUE GREATER THAN THE MAXIMUM INTEGER ALLOWED, 549755813887
115	DYNAMIC PART OF FORMAT SPECIFIER EVALUATED TO A VALUE GREATER THAN THE MAXIMUM REAL ALLOWED, 4.31359146673*10**68.

D0130 PATCH - GUARD OPTION - 09-13-72

A NEW OPTION, GUARD, HAS BEEN IMPLEMENTED TO PROVIDE AN INDICATION OF ANY PATCHES INPUT IN A "GUARDED" PORTION OF THE HOST SYMBOLIC.

A SEQUENCE RANGE TO BE GUARDED IS INPUT ON A "$." CONTROL CARD WITH THE FORMAT

$. S1 - S2 <COMMENT>

WHERE S1 AND S2 ARE SEQUENCE NUMBER; S2 MUST BE GREATER THAN S1, AND <COMMENT> IS ANY USER SPECIFIED COMMENT.

A MAXIMUM OF 100 AREAS CAN BE GUARDED; EACH AREA IS SPECIFIED ON SEPARATE CONTROL CARDS.

ANY PATCH CARDS INSERTED IN A GUARDED AREA WILL BE SUMMARIZED AT THE END OF ANY OTHER REQUESTED OUTPUT. THE FORMAT IS

GUARD <SEQUENCE #> <GUARD COMMENT>

WHERE <SEQUENCE #> IS THE OFFENDING PATCH CARD, AND

<GUARD COMMENT> IS THE COMMENT ASSOCIATED
WITH THE GUARD CARD FOR THE GUARDED AREA.

D0131 MCP LEVEL INDICATOR FORMAT - 09-08-72

IT IS INTENDED TO CHANGE THE FORMAT OF THE LEVEL WORD FOR THE MCP ON THE II.4.0 RELEASE OF THE MCP. THIS IS NECESSARY TO PROPERLY RECORD PATCH LEVELS GREATER THAN 255. THIS CHANGE WILL AFFECT ANY USER PROGRAMS OBTAINING THIS INFORMATION FROM THE SYSTEMSTATUS INTRINSIC. THE NEW FORMAT OF THIS WORD WILL BE:

PATCH LEVEL	IN BITS	15:16
MCP LEVEL	IN BITS	23:8
MARK LEVEL	IN BITS	31:8

ALL FIELDS WILL CONTAIN BINARY INTEGERS.

D0132 FILE ATTRIBUTES - 09-13-72

SERIALNO ATTRIBUTE

THE SERIALNO ATTRIBUTE IS A READ ONLY INTEGER ATTRIBUTE WHICH RETURNS THE SERIAL NUMBER OF THE TAPE ASSIGNED TO THE FILE. IF IT IS NOT A TAPE FILE OR IF THE FILE IS NOT OPENED OR IF THE FILE IS UNLABELED, THE VALUE RETURNED IS ZERO.

D0133 FILE ATTRIBUTES - 09-11-72

THE FILE ATTRIBUTE "CARRIAGECONTROL" HAS BEEN IMPLEMENTED FOR PRINTER FILES. IT HAS THREE VALUES:

STANDARD - THE DEFAULT, NORMAL CARRIAGE CONTROL AS SPECIFIED BY THE I/O STATEMENT OR FORMAT.

CTLASA - AS FOLLOWS

CTL360 - AS FOLLOWS

NOTE: NON-STANDARD CARRIAGE CONTROL IS ONLY ALLOWED FOR CHARACTER ORIENTED (UNITS = TRUE) EBCDIC (INTMODE = EBCDIC) PRINTER FILES.

CTLASA

THE LINE IS PRINTED AFTER CARRIAGE MOTION HAS COMPLETED ACCORDING TO THE FIRST CHARACTER OF THE RECORD AS FOLLOWS:

CHARACTER	ACTION
" "(BLANK)	SINGLE SPACING
"0"	DOUBLE SPACING
"-"(MINUS)	TRIPLE SPACING
"+"	NO CARRIAGE MOTION
"1"	SKIP TO CHANNEL 1
"2"	SKIP TO CHANNEL 2
"3"	SKIP TO CHANNEL 3
"4"	SKIP TO CHANNEL 4
"5"	SKIP TO CHANNEL 5
"6"	SKIP TO CHANNEL 6
"7"	SKIP TO CHANNEL 7
"8"	SKIP TO CHANNEL 8
"9"	SKIP TO CHANNEL 9

CTL360

THE FIRST CHARACTER CONTROLS PAPER MOTION USING THE VARIOUS FIELDS IN THE CHARACTER AS FOLLOWS:

FIELD ACTION

0:1 IF THIS FIELD IS 1 THEN PRINTING OCCURS AFTER CARRIAGE MOTION.

1:1 IF THIS FIELD IS 1 THEN NO PRINTING WILL OCCUR; ONLY CARRIAGE CONTROL.

2:1 IGNORED.

6:4 THE VALUE OF THIS FIELD IS EITHER THE CHANNEL NUMBER (IF SKIPPING) OR THE LINE COUNT (IF SPACING) ACCORDING TO 7:1

7:1 IF THIS FIELD IS 1 THEN A SKIP TO CHANNEL IS THE CARRIAGE CONTROL, OTHERWISE, THE PRINTER SPACED THE NUMBER OF SPACES SPECIFIED BY 6:4.

D0134 SORT - COMPILE-TIME OPTION - 09-13-72

THIS CHANGE IMPLEMENTS A NEW $ OPTION WHICH ALLOWS THE SORT TO BE COMPILED SO THAT THE SORT RESTART CODE IS OMITTED. A MODERATE TIMING IMPROVEMENT RESULTS FROM THIS OMISSION.

TO ELIMINATE THE RESTART CODE:

USE A $ RESET RESTART OF $ POP RESTART CARD AT SEQUENCE NUMBER 90002001

RECOMPILE AND REBIND THE SORT INTO THE MCP.

NEW FEATURES AND DOCUMENTATION CHANGES

OTHER SMALL CHANGES ARE IMPLEMENTED BY THIS PATCH.

DOCUMENTS AFFECTED

DOCUMENT	SYSTEM NOTE	MARKETING NO.	MARKETING DATE
ALGOL COMPILER	D0001	5000136	06-72
ALGOL COMPILER	D0004	5000136	06-72
ALGOL COMPILER	D0013	5000136	06-72
ALGOL COMPILER	D0024	5000136	06-72
ALGOL COMPILER	D0043	5000136	06-72
ALGOL COMPILER	D0080	5000136	06-72
ALGOL COMPILER	D0118	5000136	06-72
ALGOL LANGUAGE	D0003	5000128	06-72
ALGOL LANGUAGE	D0005	5000128	06-72
ALGOL LANGUAGE	D0007	5000128	06-72
ALGOL LANGUAGE	D0008	5000128	06-72
ALGOL LANGUAGE	D0009	5000128	06-72
ALGOL LANGUAGE	D0010	5000128	06-72
ALGOL LANGUAGE	D0011	5000128	06-72
ALGOL LANGUAGE	D0018	5000128	06-72
ALGOL LANGUAGE	D0020	5000128	06-72
ALGOL LANGUAGE	D0022	5000128	06-72
ALGOL LANGUAGE	D0025	5000128	06-72
ALGOL LANGUAGE	D0035	5000128	06-72
ALGOL LANGUAGE	D0047	5000128	06-72
ALGOL LANGUAGE	D0052	5000128	06-72
ALGOL LANGUAGE	D0057	5000128	06-72
ALGOL LANGUAGE	D0087	5000128	06-72
ALGOL LANGUAGE	D0098	5000128	06-72
ALGOL LANGUAGE	D0111	5000128	06-72
BASIC COMPILER	D0001	5000383	07-71
BASIC LANGUAGE	D0003	5000383	07-71
BASIC LANGUAGE	D0051	5000383	07-71
BASIC LANGUAGE	D0061	5000383	07-71

NEW FEATURES AND DOCUMENTATION CHANGES

DOCUMENT	SYSTEM NOTE	MARKETING NO.	MARKETING DATE
BASIC LANGUAGE	D0088	5000383	07-71
BINDER INFORMATION	D0001	5000045	11-71
BINDER INFORMATION	D0003	5000045	11-71
BINDER INFORMATION	D0004	5000045	11-71
BINDER INFORMATION	D0008	5000045	11-71
BINDER INFORMATION	D0089	5000045	11-71
COBOL COMPILER	D0001	5000110	11-70
COBOL COMPILER	D0001	5000110	11-70
COBOL COMPILER	D0002	5000110	11-70
COBOL COMPILER	D0002	5000110	11-70
COBOL COMPILER	D0039	5000110	11-70
COBOL COMPILER	D0122	5000110	11-70
COBOL LANGUAGE	D0003	5000102	10-71
COBOL LANGUAGE	D0005	5000102	10-71
COBOL LANGUAGE	D0005	5000102	10-72
COBOL LANGUAGE	D0014	5000102	10-71
COBOL LANGUAGE	D0039	5000102	10-71
COBOL LANGUAGE	D0070	5000102	10-71
COBOL LANGUAGE	D0071	5000102	10-71
COBOL LANGUAGE	D0072	5000102	10-71
COBOL LANGUAGE	D0073	5000102	10-71
COBOL LANGUAGE	D0074	5000102	10-71
DUMP ANALYZER	D0091	5000334	11-71
ESPOL INFORMATION	D0001	5000094	06-72
ESPOL INFORMATION	D0007	5000094	06-72
ESPOL INFORMATION	D0008	5000094	06-72
ESPOL INFORMATION	D0012	5000094	06-72
ESPOL INFORMATION	D0015	5000094	06-72
ESPOL INFORMATION	D0017	5000094	06-72
ESPOL INFORMATION	D0019	5000094	06-72
ESPOL INFORMATION	D0021	5000094	06-72
ESPOL INFORMATION	D0022	5000094	06-72
ESPOL INFORMATION	D0023	5000094	06-72

DOCUMENT	SYSTEM NOTE	MARKETING NO.	MARKETING DATE
ESPOL INFORMATION	D0024	5000094	06-72
ESPOL INFORMATION	D0025	5000094	06-72
ESPOL INFORMATION	D0040	5000094	06-72
ESPOL INFORMATION	D0041	5000094	06-72
ESPOL INFORMATION	D0041	5000094	06-72
ESPOL INFORMATION	D0042	5000094	06-72
ESPOL INFORMATION	D0043	5000094	06-72
ESPOL INFORMATION	D0056	5000094	06-72
ESPOL INFORMATION	D0057	5000094	06-72
ESPOL INFORMATION	D0075	5000094	06-72
ESPOL INFORMATION	D0094	5000094	06-72
ESPOL INFORMATION	D0099	5000094	06-72
ESPOL INFORMATION	P0615	5000094	06-72
ESPOL INFORMATION	D0019	5000094	11-70
ESPOL INFORMATION	D0049	9000094	11-70
FORTRAN REFERENCE	D0109	5000029	11-70
FORTRAN REFERENCE	D0001	5000458	07-72
FORTRAN REFERENCE	D0002	5000458	07-72
FORTRAN REFERENCE	D0003	5000458	07-72
FORTRAN REFERENCE	D0004	5000458	07-72
FORTRAN REFERENCE	D0005	5000458	07-72
FORTRAN REFERENCE	D0062	5000458	07-72
FORTRAN REFERENCE	D0065	5000458	07-72
FORTRAN REFERENCE	D0097	5000458	07-72
FORTRAN REFERENCE	D0110	5000458	07-72
FORTRAN REFERENCE	D0111	5000458	07-72
FORTRAN REFERENCE	D0113	5000458	06-72
I-O SUBSYSTEM	D0092	5000185	07-71
MATH INTRINSICS	D0085	5000151	11-71
MATH INTRINSICS	D0096	5000151	11-71
MCP DOCUMENT	D0114	5000086	11-70
NDL	D0115	5000079	08-71
PLI INFORMATION	D0001	5000201	06-72

DOCUMENT	SYSTEM NOTE	MARKETING NO.	MARKETING DATE
PROGRAM BINDER	D0118	5000045	11-71
RJE INFORMATION	D0036	5000300	06-72
RJE INFORMATION	D0064	5000300	06-72
RJE INFORMATION	D0100	5000300	06-72
RJE INFORMATION	D0103	5000300	06-72
RJE INFORMATION	D0104	5000300	06-72
RJE INFORMATION	D0105	5000300	06-72
RJE INFORMATION	D0106	5000300	06-72
RJE INFORMATION	D0107	5000300	06-72
RJE INFORMATION	D0108	5000300	06-72
SORT INFORMATION	D0035	5000144	11-71
SYSTEM HANDBOOK	D0037	5000276	01-72
SYSTEM HANDBOOK	D0092	5000276	01-72
SYSTEM HANDBOOK	D0117	5000276	01-72
SYSTEM MISCELLANEA	D0044	5000367	04-72
XALGOL	D0013	5000128	06-72
XALGOL LANGUAGE	D0005	5000128	06-72
XALGOL LANGUAGE	D0043	5000128	06-72
XALGOL LANGUAGE	D0053	5000128	06-72
XALGOL LANGUAGE	D0055	5000128	06-72

KWIC SUBJECT INDEX

KWIC	SYSTEM NOTE	FUNCTION
AND ID PARAGRAPHS NOTE	D0072	COBOL
AND LOGOFF TIMES LOGON	D0107	RJE
AND PUNCH-BACKUP PRINTER	D0027	
AND TASK ATTRIBUTES FILE	D0025	
AND TRANSLATE TABLES TRUTHSETS	D0010	
ANSI AND B5500 COMPATIBILITY	D0014	
AREA SAME [RECORD]	D0070	COBOL
ARRAYS ALGOL CHARACTER	D0009	
ASCII CAPABILITY	D0003	
ATTRIBUTE IO FILE	D0062	FORTRAN
ATTRIBUTES FILE	D0045	MCP
ATTRIBUTES FILE	D0058	MCP
ATTRIBUTES FILE	D0092	MCP
ATTRIBUTES FILE	D0132	
ATTRIBUTES FILE	D0133	
ATTRIBUTES FILE AND TASK	D0025	
ATTRIBUTES RANDOM-SERIAL FILE	D0063	
AUTODIALOUT RJE	D0036	RJE
AUTOPRINT	D0054	MCP
BACKUP DIRECTED	D0100	RJE
BACKUP RESTART	D0103	RJE
BASIC "FOR" STATEMENT IN	D0051	BASIC
BASIC RELATIONAL OPERATORS IN	D0061	BASIC
BILLING ACCOUNTING LOG	D0128	
BINDING GLOBALS IN	D0008	
BINDING COMPILE TIME	D0004	
B5500 COMPATIBILITY ANSI AND	D0014	
B6700 DISK PACK SOFTWARE	D0028	
CALL IN ESPOL DIRECT PROCEDURE	D0040	
CANDE	D0038	
CANDEFILES	D0127	
CAPABILITY ASCII	D0003	
CARD FILES	D0104	RJE

KWIC	SYSTEM NOTE	FUNCTION
CHANGES ESPOL VECTOR MODE	D0017	
CHANGES COBOL DOCUMENTATION	D0039	
CHANGES EVENT RELATED SYNTAX	D0019	
CHARACTER ARRAYS ALGOL	D0009	
COBOL DOCUMENTATION CHANGES	D0039	
COBOL INTRINSICS	D0112	BINDER
COBOL SEGMENTATION	D0122	COBOL
CODE FILE RQMTS-SEP REDUCING	D0113	
CODES IN ALGOL STRING	D0011	
COMPARE SEQUENCE	D0082	COMPARE
COMPATIBILITY ANSI AND B5500	D0014	
COMPILATION OF COMPILERS	D0013	
COMPILE AND GO	D0111	ALGOL
COMPILE TIME BINDING	D0004	
COMPILE-TIME OPTION	D0134	SORT
COMPILER PLI	D0102	PLI
COMPILER GENERATION PLI	D0067	
COMPILERS COMPILATION OF	D0013	
CONFIDENCE ROUTINES DISK PACK	D0119	
CONFIDENCE TESTS DISK PACK	D0032	SCR
CONTROL USERS GUIDE TO MEMORY	D0059	
CONTROLLED SEGMENTATION USER	D0007	ALGOL
COPY-COMPARE	D0031	MCP
COUNT ERROR	D0083	COMPARE
CROSS REFERENCE IN FORTRAN	D0065	FORTRAN
DCP PROGRAMS LOADING	D0046	LOADER
DCSTATUS	D0116	
DCSYSTEMTABLES INTRINSIC	D0084	ESPOLINTRN
DC1000 DISCONNECT	D0064	SRCEDC1000
DEBUGGING INFORMATION TIMING &	D0109	FORTRAN
DECLARATION XALGOL FILE	D0055	XALGOL
DECLARATIONS NEW ALGOL	D0098	
DECLARATIONS ALGOL FAULT	D0087	ALGOL

KWIC	SYSTEM NOTE	FUNCTION
ESPOL TYPE TRANSFER FUNCTIONS	D0056	ESPOL
ESPOL VECTOR MODE CHANGES	D0017	
EVENT RELATED SYNTAX CHANGES	D0019	
EXECUTE OPTION	D0076	PATCH
EXTENSION "USING PICTURE"	D0075	ESPOL
FAULT DECLARATIONS ALGOL	D0087	ALGOL
FEATURES IN ESPOL NEW LABEL	D0021	
FILE AND TASK ATTRIBUTES	D0025	
FILE ATTRIBUTE IO	D0062	FORTRAN
FILE ATTRIBUTES	D0045	MCP
FILE ATTRIBUTES	D0058	MCP
FILE ATTRIBUTES	D0092	MCP
FILE ATTRIBUTES	D0132	
FILE ATTRIBUTES	D0133	
FILE ATTRIBUTES RANDOM-SERIAL	D0063	
FILE DECLARATION XALGOL	D0055	XALGOL
FILE RQMTS-SEP REDUCING CODE	D0113	
FILE SECURITY DISK PACK	D0044	MCP
FILES CARD	D0104	RJE
FOR [MULTIPLE] REEL	D0073	COBOL
FOR ALGOL GLOBALS SYNTAX	D0018	
FOR DISK PACKS LIBRARY MAINT	D0034	MCP
FORK STATEMENT ESPOL	D0041	ESPOL
FORM INPUT FREE	D0002	
FORM SOURCE INPUT FREE	D0002	COBOL
FORMAT MCP LEVEL INDICATOR	D0131	
FORMATTED OUTPUT FORTRAN	D0129	
FORTRAN IDENTIFIERS IN	D0110	FORTRAN
FORTRAN CROSS REFERENCE IN	D0065	FORTRAN
FORTRAN FORMATTED OUTPUT	D0129	
FORWARD INTERRUPT DECLARATIONS	D0020	
FREE FORM INPUT	D0002	
FREE FORM SOURCE INPUT	D0002	COBOL

NEW FEATURES AND DOCUMENTATION CHANGES

KWIC	SYSTEM NOTE	FUNCTION
FRSN OBJECT JOB OUTPUT AND	D0093	MCP
FULL DUPLEX	D0115	DCPPROGEN
FUNCTION "RESERVE" & "RETURN"	D0114	
FUNCTIONS TYPE TRANSFER	D0047	
FUNCTIONS ESPOL TYPE TRANSFER	D0056	ESPOL
GENERATION PLI COMPILER	D0067	
GLOBAL ITEMS NAMES OF	D0091	DUMPANALY
GLOBALS SYNTAX FOR ALGOL	D0018	
GLOBALS IN BINDING	D0008	
GO COMPILE AND	D0111	ALGOL
GRAPHIC SPELLING	D0099	XREFANALY
GUARD OPTION	D0130	PATCH
GUIDE TO MEMORY CONTROL USERS	D0059	
I-O MODIFIERS	D0090	SCR
I-O ESPOL	D0026	
ID PARAGRAPHS NOTE AND	D0072	COBOL
IDENTIFICATION PATCH	D0024	
IDENTIFICATION SOURCE LINE	D0001	
IDENTIFIERS IN FORTRAN	D0110	FORTRAN
IMPROVEMENTS SORT	D0035	
IN ALGOL STRING CODES	D0011	
IN BASIC "FOR" STATEMENT	D0051	BASIC
IN BASIC RELATIONAL OPERATORS	D0061	BASIC
IN BINDING GLOBALS	D0008	
IN ESPOL INTRINSICS	D0012	
IN ESPOL DYNAMIC PROCEDURES	D0015	
IN ESPOL NEW LABEL FEATURES	D0021	
IN ESPOL DIRECT PROCEDURE CALL	D0040	
IN FORTRAN IDENTIFIERS	D0110	FORTRAN
IN FORTRAN CROSS REFERENCE	D0065	FORTRAN
INDICATOR FORMAT MCP LEVEL	D0131	
INFORMATION TIMING & DEBUGGING	D0109	FORTRAN
INPUT FREE FORM	D0002	

KWIC	SYSTEM NOTE	FUNCTION
INPUT FREE FORM SOURCE	D0002	COBOL
INPUT MESSAGE DP	D0105	RJE
INTERFACE PROGRAM TC500-CANDE	D0126	
INTERRUPT DECLARATIONS FORWARD	D0020	
INTRINSIC DCSYSTEMTABLES	D0084	ESPOLINTRN
INTRINSICS NEW	D0079	ALGOLINTRN
INTRINSICS MATH	D0085	ESPOLINTRN
INTRINSICS COBOL	D0112	BINDER
INTRINSICS NEW PLI	D0078	ESPOLINTRN
INTRINSICS DOUBLE PREC. MATH	D0096	ESPOLINTRN
INTRINSICS NEW DECLARATIONS &	D0094	XREFANALY
INTRINSICS IN ESPOL	D0012	
IO FILE ATTRIBUTE	D0062	FORTRAN
ISNT OPERATOR	D0022	
ITEMS NAMES OF GLOBAL	D0091	DUMPANALY
JOB OUTPUT AND FRSN OBJECT	D0093	MCP
JOB SWAPPING	D0029	
JUSTIFIED [RIGHT]	D0071	COBOL
KEYBOARD MESSAGE "PC"	D0060	MCP
LABEL ADDRESS-EQUATION ESPOL	D0042	
LABEL FEATURES IN ESPOL NEW	D0021	
LEVEL INDICATOR FORMAT MCP	D0131	
LIBRARY MAINT FOR DISK PACKS	D0034	MCP
LINE IDENTIFICATION SOURCE	D0001	
LISTING SORTED STACK	D0089	BINDER
LISTING DISK PACK DIRECTORY	D0095	PACKDIR
LOADING DCP PROGRAMS	D0046	LOADER
LOG BILLING ACCOUNTING	D0128	
LOGGING MCS	D0037	
LOGICAL NOT ALGOL	D0052	
LOGOFF TIMES LOGON AND	D0107	RJE
LOGON AND LOGOFF TIMES	D0107	RJE
MAINT FOR DISK PACKS LIBRARY	D0034	MCP

KWIC	SYSTEM NOTE	FUNCTION
MATH INTRINSICS	D0085	ESPOLINTRN
MATH INTRINSICS DOUBLE PREC.	D0096	ESPOLINTRN
MCP LEVEL INDICATOR FORMAT	D0131	
MCS LOGGING	D0037	
MEMORY CONTROL USERS GUIDE TO	D0059	
MESSAGE TI RSC	D0106	RJE
MESSAGE DP INPUT	D0105	RJE
MESSAGE "PC" KEYBOARD	D0060	MCP
MESSAGES SCHEDULED	D0108	RJE
MODE CHANGES ESPOL VECTOR	D0017	
MODIFICATION SYSTEM	D0066	
MODIFIERS I-O	D0090	SCR
MONITOR STATEMENT	D0005	
NAMES OF GLOBAL ITEMS	D0091	DUMPANALY
NEW ALGOL DECLARATIONS	D0098	
NEW DECLARATIONS & INTRINSICS	D0094	XREFANALY
NEW INTRINSICS	D0079	ALGOLINTRN
NEW LABEL FEATURES IN ESPOL	D0021	
NEW PLI INTRINSICS	D0078	ESPOLINTRN
NOBINDINFO DOLLAR OPTION	D0118	BINDER
NOT ALGOL LOGICAL	D0052	
NOTE AND ID PARAGRAPHS	D0072	COBOL
NUMBERCONVERT PROCEDURE	D0077	ESPOLINTRN
OBJECT JOB OUTPUT AND FRSN	D0093	MCP
OF COMPILERS COMPILATION	D0013	
OF GLOBAL ITEMS NAMES	D0091	DUMPANALY
OF WRITE ADVANCING OPTION	D0074	COBOL
OPERATOR ISNT	D0022	
OPERATORS IN BASIC RELATIONAL	D0061	BASIC
OPTION GUARD	D0130	PATCH
OPTION EXECUTE	D0076	PATCH
OPTION COMPILE-TIME	D0134	SORT
OPTION ESPOL "VERSION" DOLLAR	D0049	ESPOL

KWIC	SYSTEM NOTE	FUNCTION
OPTION "NOBINDINFO"	D0080	ALGOL
OPTION NOBINDINFO DOLLAR	D0118	BINDER
OPTION OF WRITE ADVANCING	D0074	COBOL
OPTIONS REFERENCING USER	D0043	XALGOL
OUTPUT FORTRAN FORMATTED	D0129	
OUTPUT AND FRSN OBJECT JOB	D0093	MCP
PACK CONFIDENCE ROUTINES DISK	D0119	
PACK CONFIDENCE TESTS DISK	D0032	SCR
PACK DIRECTORY LISTING DISK	D0095	PACKDIR
PACK FILE SECURITY DISK	D0044	MCP
PACK SOFTWARE B6700 DISK	D0028	
PACKS LIBRARY MAINT FOR DISK	D0034	MCP
PARAGRAPHS NOTE AND ID	D0072	COBOL
PATCH IDENTIFICATION	D0024	
PICTURE" EXTENSION "USING	D0075	ESPOL
PLI COMPILER	D0102	PLI
PLI COMPILER GENERATION	D0067	
PLI INTRINSICS NEW	D0078	ESPOLINTRN
POINTER SKIP	D0057	
PREC. MATH INTRINSICS DOUBLE	D0096	ESPOLINTRN
PRINTBINDINFO PROGRAM	D0117	
PRINTER AND PUNCH-BACKUP	D0027	
PROCEDURE CALL IN ESPOL DIRECT	D0040	
PROCEDURE NUMBERCONVERT	D0077	ESPOLINTRN
PROCEDURES IN ESPOL DYNAMIC	D0015	
PROGRAM PRINTBINDINFO	D0117	
PROGRAM TC500-CANDE INTERFACE	D0126	
PROGRAMDUMP	D0033	
PROGRAMS LOADING DCP	D0046	LOADER
PUNCH-BACKUP PRINTER AND	D0027	
QUEUES UNTYPED	D0023	
RANDOM-SERIAL FILE ATTRIBUTES	D0063	
READ-WRITE DISK VERIFY TESTS	D0120	

KWIC	SYSTEM NOTE	FUNCTION
REDUCING CODE FILE RQMTS-SEP	D0113	
REEL FOR [MULTIPLE]	D0073	COBOL
REFERENCE IN FORTRAN CROSS	D0065	FORTRAN
REFERENCING USER OPTIONS	D0043	XALGOL
RELATED SYNTAX CHANGES EVENT	D0019	
RELATIONAL OPERATORS IN BASIC	D0061	BASIC
RESEQBASIC	D0123	
RESTART BACKUP	D0103	RJE
RJE AUTODIALOUT	D0036	RJE
ROUTINES DISK PACK CONFIDENCE	D0119	
ROW> ZIP WITH <ARRAY	D0053	
RQMTS-SEP REDUCING CODE FILE	D0113	
RSC MESSAGE TI	D0106	RJE
SAME [RECORD] AREA	D0070	COBOL
SCHEDULED MESSAGES	D0108	RJE
SECURITY DISK PACK FILE	D0044	MCP
SEGMENTATION COBOL	D0122	COBOL
SEGMENTATION USER CONTROLLED	D0007	ALGOL
SEQUENCE COMPARE	D0082	COMPARE
SKIP POINTER	D0057	
SOFTWARE B6700 DISK PACK	D0028	
SOFTWARE TRANSLATION	D0101	MCP
SORT IMPROVEMENTS	D0035	
SORTED STACK LISTING	D0089	BINDER
SOURCE INPUT FREE FORM	D0002	COBOL
SOURCE LINE IDENTIFICATION	D0001	
SPELLING GRAPHIC	D0099	XREFANALY
STACK LISTING SORTED	D0089	BINDER
STATEMENT DUMP	D0006	
STATEMENT MONITOR	D0005	
STATEMENT ESPOL FORK	D0041	ESPOL
STATEMENT IN BASIC "FOR"	D0051	BASIC
STATEMENTS "DATA"	D0088	BASIC

KWIC	SYSTEM NOTE	FUNCTION
STRING CODES IN ALGOL	D0011	
SWAPPING JOB	D0029	
SYNTAX CHANGES EVENT RELATED	D0019	
SYNTAX FOR ALGOL GLOBALS	D0018	
SYSTEM MODIFICATION	D0066	
SYSTEMLOG	D0081	ESPOLINTRN
TABLES TRUTHSETS AND TRANSLATE	D0010	
TASK ATTRIBUTES FILE AND	D0025	
TC500-CANDE INTERFACE PROGRAM	D0126	
TESTS DISK PACK CONFIDENCE	D0032	SCR
TESTS READ-WRITE DISK VERIFY	D0120	
TI RSC MESSAGE	D0106	RJE
TIME BINDING COMPILE	D0004	
TIMES LOGON AND LOGOFF	D0107	RJE
TIMING & DEBUGGING INFORMATION	D0109	FORTRAN
TIMING AIDS	D0097	ESPOLINTRN
TO MEMORY CONTROL USERS GUIDE	D0059	
TRANSFER FUNCTIONS TYPE	D0047	
TRANSFER FUNCTIONS ESPOL TYPE	D0056	ESPOL
TRANSLATE TABLES TRUTHSETS AND	D0010	
TRANSLATION SOFTWARE	D0101	MCP
TRUTHSETS AND TRANSLATE TABLES	D0010	
TYPE TRANSFER FUNCTIONS	D0047	
TYPE TRANSFER FUNCTIONS ESPOL	D0056	ESPOL
UNTYPED QUEUES	D0023	
USER CONTROLLED SEGMENTATION	D0007	ALGOL
USER OPTIONS REFERENCING	D0043	XALGOL
USERS GUIDE TO MEMORY CONTROL	D0059	
VECTOR MODE CHANGES ESPOL	D0017	
VERIFY TESTS READ-WRITE DISK	D0120	
WITH <ARRAY ROW> ZIP	D0053	
WRITE ADVANCING OPTION OF	D0074	COBOL
XALGOL FILE DECLARATION	D0055	XALGOL

KWIC	SYSTEM NOTE	FUNCTION
ZIP WITH <ARRAY ROW>	D0053	