EDWARD REID

EDWARD@PALEO.ORG

22 October 2002

This document was once a section of a larger one, the rest of which was discarded many years ago. I do not remember the detailed reason, though I think it was because the remainder was less interesting. The internal evidence -- particularly the dates in the table of contents -- make it apparent that the document is relative to the Mark 2.4 release. This is supported by the label on the binder from which I removed it.

CHAPTER 1

NEW FEATURES AND DOCUMENTATION CHANGES

INTRODUCTION

This chapter contains information concerning the current status of software documentation. Improvements, changes, and new features will be published in this chapter with each new software release.

The organization of the descriptions are by function rather than by software item. Thus, particular information may be found in one place rather than scattered throughout various descriptions under ALGOL, COBOL, MCP, etc.

Each description has been assigned a sequence number with the letter "D" as a prefix. The sequence numbers are monotonically increasing and are used to reference the notes in the index tables.

The first column of the "Documents Affected" table contains the name of any document that is affected by a D-Note. The second column contains the sequence number of the note that should be used to modify the publication listed in the first column. The third column is the Marketing Number which may be used to order the document from the Technical Information Organization. The fourth column contains the date of the affected publication.

The second table is a "KWIC" Subject Index. Each entry is a keyword which references the D-Note and function relative to the keyword.

TABLE OF CONTENTS

INTRODUCTION

THE FOLLOWING LIST INDICATES D NOTES THAT WERE PUBLISHED IN AN EARLIER ISSUE OF "NEW FEATURES AND DOCUMENTATION CHANGES" AND HAVE SUBSEQUENTLY BEEN INCORPORATED IN A BURROUGHS INFORMATION MANUAL. THESE NOTES MAY BE CROSSED OUT IN YOUR PERMANENT FILE AND ANY PAGES UPON WHICH ALL OF THE NOTES HAVE BEEN DELETED MAY BE ELIMINATED.

D0014
D0039
D0070
D0071
D0072
D0073
D0074
D0122

NEW FEATURES AND DOCUMENTATION CHANGES

ALGOL

D0145 ALGOL - VECTOR MODE IN ALGOL & FORTRAN - 12-04-72

VECTORMODE IN ALGOL

IF "VECTORMODE" IS SET WHEN COMPILING THE ALGOL COMPILER, ALGOL (AND DCALGOL) WILL PARSE AND EMIT VECTORMODE CODE.

THE VECTOR MODE SYNTAX IN ALGOL IS AS FOLLOWS:

DO VECTORMODE (VECTORID,[[ID=] = [VECTORID,][[ID=][VECTORID,]]
FOR COUNTVALUE) VECTOR COMPOUND STATEMENT;

WHERE [ ] INDICATES AN OPTIONAL CLAUSE.

EACH VECTORID SPECIFIES THE NAME OF THE VECTOR AND THE STARTING POINT IN THAT VECTOR. AN ASTERISK (*) CAN BE USED TO REPLACE THE LAST SUBSCRIPT TO INDICATE THE BEGINNING OF AN ARRAY ROW. ALSO, SUBSCRIPTS CAN BE EXPRESSIONS. VECTORID"S CANNOT BE SEGMENTED ARRAYS. IN ORDER TO USE AN ARRAY THAT IS LONGER THAN 1,023 WORDS IN VECTORMODE, IT MUST BE DECLARED LONG. THE ITEMS IN THE BRACKETS MAY BE EMPTY. THE CONSTRUCT ID = VECTORID, ALLOWS THE ABILITY TO USE THE SAME ARRAY WITH DIFFERENT STARTING POINTS. COUNTVALUE IS MANDATORY. IT INDICATES THE MAXIMUM NUMBER OF TIMES THE VECTOR COMPOUND STATEMENT IS EXECUTED. IT CAN BE ANY EXPRESSION.

EXPRESSIONS FOR SUBSCRIPTS AND COUNTVALUE ARE EVALUATED ONLY ONCE FROM LEFT TO RIGHT, PRIOR TO ENTERING VECTOR MODE.

EXAMPLES:

```
DO VECTORMODE (ARRID1[*], ARRID2[2, *],
   ARRID3 [K-2, I], FOR 100) BEGIN --- END;
```

```
DO VECTORMODE (ARRID1[*], AA=ARRID1[I], FOR X + A BEGIN --- END;
DO VECTORMODE (ARRID1[*], FOR N) BEGIN --- END;

DO VECTORMODE (ARRID1[K-1], RA[*], FOR N=1) BEGIN --- END;
```

VECTOR COMPOUND STATEMENT

THE COMPOUND STATEMENT THAT FOLLOWS THE DO VECTORMODE MAY TAKE THE GENERAL FORM OF AN ALGOL COMPOUND STATEMENT. HOWEVER, THE SYNTAX THAT IS PERMITTED INSIDE SUCH COMPOUND STATEMENTS IS RESTRICTED.

THE ONLY ARRAYS THAT MAY BE REFERENCED ARE THE ONE TO THREE VECTORS INDICATED IN THE DO STATEMENT. WITHIN THE VECTOR COMPOUND STATEMENT, EACH VECTORID OR ASSOCIATED ID IS REFERRED TO WITHOUT SUBSCRIPTS. THE MEANING IS ALWAYS TO FETCH FROM OR STORE INTO THE CURRENT ADDRESS CONTAINED IN THE IC MEMORY FOR THAT VECTOR.

ANY ARITHMETIC OR LOGICAL VARIABLE IN THE STACK CAN BE REFERENCED; HOWEVER, NO REFERENCE MAY BE MADE THAT MIGHT CAUSE AN INTERRUPT. THIS MEANS, IN PARTICULAR, THAT CALL BY NAME PARAMETERS, FILES, EVENTS, TASKS, AND POINTERS MAY NOT BE REFERENCED.

THE FOLLOWING ARE THE ONLY STATEMENTS PERMITTED IN A VECTOR COMPOUND STATEMENT:

1. BEGIN AND END.
2. IF
3. UNCONDITIONAL GO TO.

   THE GO TO STATEMENT IS INTERPRETED IN THE FOLLOWING MANNER. IF THE LABEL IS LOCAL TO THE VECTOR MODE BLOCK, ONLY A BRANCH FORWARD IS ALLOWED. IF THE LABEL IS OUTSIDE THE VECTOR MODE BLOCK, VECTOR MODE IS EXITED AND CODE IS EXECUTED TO BRANCH TO THAT LABEL. AS ALL LABELS INSIDE THE VECTOR COMPOUND STATEMENT ARE LOCAL, NO BRANCHING IS PERMITTED INTO THE RANGE OF THAT STATEMENT.

4. ARITHMETIC ASSIGNMENT STATEMENTS.
5. EXIT STATEMENT. THIS IS A NEW STATEMENT. IF EXECUTED,

IT MEANS TO EXIT VECTOR MODE AND CONTINUE EXECUTION WITH THE FIRST EXECUTABLE STATEMENT FOLLOWING THE VECTOR COMPOUND STATEMENT.

6. INCREMENT STATEMENT. THIS IS A NEW STATEMENT. IT TAKES THE FORM:

   INCREMENT VECTORID [, VECTORID, [VECTORID ...]];

   WHERE THE BRACKETS INDICATE OPTIONAL ITEMS.

   THE INCREMENT STATEMENT INCREMENTS THE ADDRESS OF THE VECTORS REFERENCED BY ONE (1) FOR SINGLE PRECISION ARRAYS AND TWO (2) FOR DOUBLE PRECISION ARRAYS.

      INCREMENT A, B;

   WOULD INCREMENT THE ADDRESS FOR VECTORS A AND B.

      INCREMENT A, A;

   WOULD INCREMENT THE ADDRESS FOR THE VECTOR A TWICE. IT IS MORE EFFICIENT TO INCREMENT A VECTOR ADDRESS AFTER A REFERENCE TO IT, RATHER THAN BEFORE. THERE ARE NO IMPLIED INCREMENTS IN A VECTOR MODE STATEMENT. THUS, IF NO SUCH STATEMENTS APPEAR, THE VECTOR ADDRESSES ARE NEVER INCREMENTED.

7. LABEL DECLARATION STATEMENTS.

ARITHMETIC EXPRESSIONS IN VECTOR MODE

ARITHMETIC EXPRESSIONS IN VECTOR MODE ARE STRICTLY LIMITED IN FORM. THEY MUST MEET THE FOLLOWING REQUIREMENTS.

1. PROCEDURE CALLS OF ANY SORT ARE PROHIBITED. THIS MEANS THAT ANY CALL ON AN INTRINSICS FUNCTION THAT IS NOT INLINE -- EXPRESS OR IMPLIED -- IS PROHIBITED. FOR EXAMPLE, LN MAY NOT BE CALLED. SINCE EXPONENTIATION GENERALLY CALLS AN INTRINSIC, NON-CONSTANT EXPONENTIATION IS PROHIBITED.

2. REFERENCE TO ANY POINTERS OR CHARACTER ARRAYS IS PROHIBITED THROUGHOUT VECTOR MODE AND ITS INVOCATION.

3. REFERENCE TO ANY FILE, CALL BY NAME PARAMETER AT ANY LEVEL, ARRAY OR SUBSCRIPTED VARIABLE, (OTHER THAN THE VECTORIDS THEMSELVES) IS PROHIBITED. SIMPLE VARIABLES THAT ARE AT ANY LEVEL OR ARE CALLED BY VALUE PARAMETERS MAY BE REFERENCED.

EXTENDED VECTOR MODE IN ALGOL

THIS EXTENDED SYNTAX IS ALLOWED IF "VECTORMODE" AND "EXTVECTOR" ARE SET WHEN COMPILING THE ALGOL COMPILER.

THIS OPTION "EXTVECTOR" SPECIFICALLY GIVES ANY ALGOL PROGRAMMER THE CAPACITY TO ACCESS DATA OUTSIDE HIS PROGRAM OR TO CRASH THE SYSTEM. IF A SITE MANAGER ALLOWS "EXTVECTOR" TO BE SET IN ANY COMPILER IN A SITE, ALL GUARANTEES OF SYSTEM INTEGRITY, FILE OR DATA SECURITY, AND SYSTEM STABILITY ARE SPECIFICALLY AND IRREVOCABLY VOIDED.

ALONG WITH VECTORMODE TWO ADDITIONAL FACILITIES ARE ALLOWED IN EXTVECTOR.

1. PRECEDING THE FIRST VECTORID CAN BE FROM ONE TO THREE INCREMENTS, ENCLOSED IN BRACKETS. FOR EXAMPLE:

   ```
   DO VECTORMODE ([2, I+J/K, J MOD K], A[*], B[*], C[*], FOR N
         BEGIN ... END;
   ```

   THESE INCREMENTS CAN EACH BE AN EXPRESSION AND ARE EVALUATED ONLY ONCE FOR EACH ENTRY. THEY INDICATE THE AMOUNT TO BE INCREMENTED ON THE RESPECTIVE VECTORID FOR EACH EXECUTION OF AN INCREMENT STATEMENT. ANY INCREMENTS OMITTED ARE ASSUMED TO BE ONE (1) FOR SINGLE PRECISION VECTORIDS AND TWO (2) FOR DOUBLE PRECISION VECTORIDS. IF A VECTORID IS DOUBLE PRECISION AND AN INCREMENT IS PROVIDED IT WILL BE THE PROGRAMMERS RESPONSIBILITY TO INSURE THAT IT IS A MULTIPLE OF TWO. (OTHERWISE THE LOWER HALF OF ONE PART AND THE HIGHER HALF OF ANOTHER PART WILL BE THE OPERANDS THAT ARE LOADED OR STORED INTO.)

2. READING OR SETTING THE INCREMENTS DYNAMICALLY IS ALSO ALLOWED. FOR EXAMPLE:

```
DO VECTORMODE ([2, Z+J/K, T+PDK], A[*], B[*], C[*], FOR N)
BEGIN
    A:= B * C;
    INCREMENT A, B, C;
    IF A. INCREMENT = 2 THEN A. INCREMENT:= 6;
    A:= B + C + A;
    B. INCREMENT := I + J;
    K:= C. INCREMENT;
    INCREMENT A, B, C;
END;
```

VECTORMODE IN FORTRAN

VECTOR MODE IS ALLOWED IN FORTRAN ONLY IF:

1. THE COMPILER HAS BEEN COMPILED WITH THE USER OPTION VECTORMODEISALLOWED SET.

2. THE USER HAS SET THE OPTION VECTORMODE FOR THE PORTION OF THE PROGRAM HE WANTS THE COMPILER TO TRY TO GENERATE VECTOR MODE CODE IN.

3. THE USER HAS ENCODED A LOOP IN FORTRAN THAT HAS THE FOLLOWING BEHAVIORAL CHARACTERISTICS:

   A. IT IS A DO LOOP OR A LOOP CONSTRUCTED WITH IF STATEMENTS THAT FUNCTIONS EQUIVALENTLY TO A DO LOOP.
   B. THE LOOP IS ENTERED ONLY FROM THE "NODE" IMMEDIATELY PRECEEDING THE LOOP. THAT IS, THE LOOP IS ALWAYS FALLEN INTO, NO STATEMENT BRANCHES INTO THE LOOP AND NO STATEMENT BRANCHES TO THE FIRST STATEMENT OF THE LOOP EXCEPT THE ACTUAL LOOPING AND TESTING CODE.

   C. THE INCREMENTED EXPRESSION IS A CONSTANT LESS THAN OR EQUAL TO THREE.

D. THE FINAL AND INITIAL VALUES ARE INTEGERS AND ARE CONSTANT WITH RESPECT TO THE LOOP. THAT IS, IF THE INITIAL AND FINAL VALUES ARE NOT CONSTANTS, THEN THEY MUST NOT BE ALTERED WITHIN THE LOOP.

E. THE CONTROL VARIABLE MUST NOT BE IN COMMON.

F. NO CALL, IO STATEMENTS, PAUSE, ZIP, CLOSE, OR OTHER FILE HANDLING STATEMENTS ARE ALLOWED.

G. NO FUNCTION REFERENCES ARE ALLOWED.

H. INTRINSIC CALLS INCLUDING ** ARE NOT ALLOWED UNLESS THE RESULT OF THE CALL IS INVARIANT WITH RESPECT TO THE LOOP.

I. IF ARRAYS OR COMMON ELEMENTS ARE ALTERED WITHIN THE LOOP, OR ARRAYS ARE REFERENCED WITHIN THE LOOP USING A SUBSCRIPT THAT CHANGES WITH EACH ITERATION WITHIN THE LOOP, THEN THERE MAY NOT BE MORE THAN THREE SUCH USAGES. THAT DOES NOT MEAN A LIMIT OF THREE ARRAYS:

```
   GO    10    I = 1, N
10    A(I) = A(I + 1) + A (I - 1) + B (N)
```

THE ABOVE LOOP WOULD GENERATE VECTOR MODE CODE IF THE OTHER CONDITIONS ARE MET, BECAUSE THERE ARE ONLY THREE ARRAYS BEING USED WITH VARYING SUBSCRIPTS. IF B (N) HAD BEEN REPLACED BY B (I) OR A (I + 2), THEN NO VECTOR CODE WOULD BE GENERATED. IF THE STATEMENT HAD BEEN

```
   10    B(N) = A(I) + A(I +1) + A (I + 3)
```

THEN NO VECTOR MODE WOULD BE GENERATED, BECAUSE WHILE ONLY THREE ARRAY SUBSCRIPTS WERE VARYING A FOURTH ARRAY WAS BEING ALTERED IN VALUE.

J. CALL AN ARRAY A COMMON ELEMENT BEING ALTERED OR BEING USED WITH VARYING SUBSCRIPTS A "VECTOR" ARRAY.

A VECTOR ARRAY CANNOT BE REFERENCED IN A PURELY CONDITIONAL MANNER. OTHERWISE NO VECTOR CODE IS GENERATED.

```
DO    10    I = 1, 10000
10    IF (I.LT.50)  A(I) = A(I) + 1
```

THE ABOVE EXAMPLE WOULD NOT GENERATE VECTOR CODE.

```
DO    10    I = 1, 10000
A(I) = A(I) + 1
10    IF (I.LT.50) A(I) = A(I) + 1
```

THE ABOVE EXAMPLE WOULD IF THE OTHER CONDITIONS ARE MET.

K. THE PARTIAL DERIVATIVE OF ALL OF THE SUBSCRIPT EXPRESSIONS WITH RESPECT TO THE CONTROL VARIABLE MUST BE AN INTEGER WHICH IS CONSTANT WITH RESPECT TO THE LOOP.

THIS ESSENTIALLY MEANS THAT IF THE SUBSCRIPT IS TOO COMPLICATED VECTORMODE CODE WILL NOT BE EMITTED.

```
DO    10    K = 1, N
10    A(I,J) = B(I,K)*  C(K,J) + A(I,J)
```

THE ABOVE WILL GIVE YOU VECTOR MODE, IF OTHER CONDITIONS ARE MET.

```
DO    10    K = 1, N
10    A(K*K) = B(K + K)
```

THE ABOVE WILL NOT GIVE VECTOR MODE BECAUSE OF THE SUBSCRIPT OF A. THE SUBSCRIPT OF B IS ACCEPTABLE.

L. VECTOR MODE WILL NOT BE USED FOR NON-LONG ARRAYS.

M. VECTOR MODE WILL NOT BE USED FOR FORMAL ARRAYS UNLESS LONG IS SET.

N. ONLY INNERMOST LOOPS WILL GET VECTOR MODE.

D0147 ALGOL - FORMAL PROCEDURES - 01-15-73

D0147 ALGOL - FORMAL PROCEDURES - 01-15-73

THE PARAMETER CHECKING DONE FOR FORMAL PROCEDURES HAS BEEN STANDARDIZED IN ORDER THAT ALGOL AND FORTRAN PROGRAM UNITS THAT PASS OR RECEIVE PROCEDURE PARAMETERS MAY BE BOUND TOGETHER.

WHEN USING PROCEDURE PARAMETERS STRICTLY WITHIN ALGOL, THE OVERHEAD OF THE RUN TIME PARAMETER CHECKING CODE CAN BE AVOIDED BY SPECIFICALLY DESCRIBING THE PARAMETERS OF A FORMAL PROCEDURE.

CALLS MADE ON SPECIFIED FORMAL PROCEDURES WILL HAVE ALL PARAMETER CHECKING DONE AT COMPILE TIME. THERE WILL BE NO ADDITIONAL CODE EXECUTED. ONLY NON-FORMAL AND SPECIFIED FORMAL PROCEDURES HAVING PARAMETER DESCRIPTIONS MATCHING THOSE OF THE SPECIFIED FORMAL PROCEDURE MAY BE PASSED AS ACTUAL PARAMETERS. AN UNSPECIFIED FORMAL PROCEDURE CANNOT BE PASSED AS AN ACTUAL PARAMETER TO A SPECIFIED FORMAL PROCEDURE.

THE SYNTAX FOR PROCEDURE AND FUNCTION DECLARATIONS ON PAGE 10-11 OF THE JUNE 1972 EXTENDED ALGOL LANGUAGE INFORMATION MANUAL SHOULD BE REVISED BY THESE DEFINITIONS:

1. CHANGE THE DEFINITION OF <SPECIFICATION> TO:

```
<SPECIFICATION>::= <SPECIFIER><IDENTIFIER LIST>/
                   <ARRAY SPECIFICATION>/
                   <PROCEDURE SPECIFICATION>
<PROCEDURE SPECIFICATION>::=PROCEDURE <FORMAL PROCEDURE LIST>/
              <TYPE> PROCEDURE <FORMAL PROCEDURE LIST>
<FORMAL PROCEDURE LIST>::= <FORMAL PROCEDURE SPECIFIER>/
     <FORMAL PROCEDURE LIST>,<FORMAL PROCEDURE SPECIFIER>
<FORMAL PROCEDURE SPECIFIER>::= <IDENTIFIER>/
     <IDENTIFIER> <SPECIFIED PARAMETER PART>; FORMAL
<SPECIFIED PARAMETER PART>::= ( )/
     (<FORMAL PARAMETER LIST>);<VALUE PART><SPECIFICATION PART>
```

2. REMOVE "PROCEDURE" AND "<TYPE> PROCEDURE" FROM THE LIST OF "<SPECIFIERS>"S

00147 ALGOL - FORMAL PROCEDURES - 01-15-73

EXAMPLE:

```
PROCEDURE P(Q1,Q2);
    PROCEDURE Q1();FORMAL.
              Q2(X);VALUE X; REAL X; FORMAL;
```

THE EMPTY PARENTHESIS MUST BE USED TO INDICATE NO PARAMETERS.

00158 ALGOL - PROGRAM AND PATCH ID - 10-16-72

ALGOL AND ESPOL HAVE BEEN MODIFIED TO ALLOW $ VERSION VV.CCC, WHERE VV REPRESENTS THE VERSION, THE FIRST V REPRESENTS THE MARK NUMBER, THE SECOND V REPRESENTS THE LEVEL NUMBER, AND THE CCC REPRESENTS THE CYCLE.

FOR A USER TO GAIN ACCESS TO THESE NUMBERS, TWO ADDITIONAL COMPILETIME OPTIONS HAVE BEEN IMPLEMENTED IN ALGOL AND ESPOL:

1. COMPILETIME (20) YIELDS THE VERSION AS AN INTEGER.

   A. COMPILETIME (20) DIV 10 YIELDS THE MARK NUMBER.
   B. COMPILETIME (20) MOD 10 YIELDS THE LEVEL NUMBER.

2. COMPILETIME (21) YIELDS THE CYCLE AS AN INTEGER.

WHEN COMPILING WITH SYSTEM/PATCH WITH THE MARK OPTION SET, THEN:

1. IF ON THE $# CARD THE FIRST STRING AFTER THE NOISE WORD IS A STRING OF THREE OR FEWER DIGITS, THEN

   A. THE ALGOL COMPILER WILL REPLACE THE THREE DIGIT PATCH NUMBER WITH THE STRING OF 10 CHARACTERS AS FOLLOWS: VERSION (TWO CHARACTERS), PERIOD, CYCLE (THREE CHARACTERS), PERIOD, PATCH (THREE CHARACTERS); I.E., VV.CCC.PPP.

   B. THE ESPOL COMPILER WILL REPLACE THE STRING THREE DIGIT PATCH NUMBER WITH THE STRING OF EIGHT CHARACTERS AS FOLLOWS: VERSION (2 CHARACTERS), CYCLE (3 CHARACTERS), PATCH (3 CHARACTERS). WHEN

D0158 ALGOL - PROGRAM AND PATCH ID - 10-16-72

PRINTED ON A LISTING, THE ESPOL COMPILER WILL INSERT PERIODS BETWEEN THE VERSION AND CYCLE, AND BETWEEN THE CYCLE AND PATCH NUMBER.

2. IF ON THE $# CARD THE FIRST STRING AFTER THE NOISE WORD IS A STRING OTHER THAN THREE (OR FEWER) DIGITS, THEN THE EIGHT-CHARACTER STRING WILL BE USED.

WHEN COMPILING WITH "NEW" SET, IF A $ VERSION CARD APPEARS IN THE SYMBOLIC, THEN IF THE PATCH DECK CONTAINS A $ VERSION CARD, THE NEW SYMBOLIC WILL BE UPDATED TO THE VERSION AND CYCLE ON THE LAST $ VERSION CARD IN THE PATCH DECK IF THE SEQUENCE NUMBER IS LESS THAN THE ONE ON THE SYMBOLIC.

D0183 ALGOL - POINTER EXPRESSIONS - 11-12-72

SYNTAX FOR <POINTER DESIGNATOR> IS EXPANDED TO PERMIT USING ANOTHER POINTER TO DESIGNATE THE SIZE OF A NEWLY INITIALIZED POINTER.

SYNTAX:

<POINTER DESIGNATOR>::=POINTER (<POINTER PARAMETERS>)

<POINTER PARAMETERS>::=<ARRAY PART>/<ARRAY PART>,
<CHARACTER SIZE>/<ARRAY PART>,<POINTER PRIMARY>

EXAMPLE:

```
POINTER P, Q; ARRAY A [0:5]; REAL R;
P:=POINTER (A, Q); P:=POINTER (A, CASE R OF (Q, P));
```

D0211 ALGOL - DUMPINFO AND LOADINFO IN ALGOL - 01-22-73

THE ALGOL DOLLAR CARD OPTIONS DUMPINFO AND LOADINFO HAVE BEEN MODIFIED TO FACILITATE THEIR USE WITH INTERMEDIATE LEVEL GLOBAL BINDING. THESE CHANGES ARE:

1. THE DUMPINFO AND LOADINFO OPTIONS MAY BE FOLLOWED BY

EITHER AN INTERNAL FILE NAME OR AN EXTERNAL FILE NAME TERMINATED WITH A PERIOD AND ENCLOSED IN QUOTES. THIS FILE NAME INFORMATION IS IN A FORMAT SIMILAR TO THE INCLUDE DOLLAR OPTION. THIS PERMITS SELECTIVE INFO DUMPING AT SEVERAL POINTS AND SELECTIVE INFO LOADING MORE THAN ONCE THROUGHOUT A COMPILATION.

2. DUMPINFO AND LOADINFO MUST NOW BE THE LAST OPTION APPEARING ON A DOLLAR CARD.

3. WHEN A NEW LOADINFO IS DONE ALL OLD INFO STRUCTURE IN ALGOL IS REMOVED. THUS, COMPILING DIFFERENT PORTIONS OF THE SAME PROGRAM, EVEN IF THEY OPERATE IN DIFFERENT ENVIRONMENTS, MAY NOW BE DONE IN THE SAME COMPILATION.

4. LOADINFO CHANGES ALL VARIABLES IN INFO TO BE GLOBALS AND ALL PROCEDURES ALREADY COMPILED TO BE FORWARD. THIS MEANS THAT AN INFO FILE CREATED BY A DUMPINFO DONE IMMEDIATELY BEFORE A PROCEDURE IN A NORMAL COMPILATION WILL BE SUITABLE FOR FUTURE USE AS GLOBALS IF ONE WISHES TO SEPARATELY COMPILE THAT PROCEDURE.

IN GENERAL, THE EFFECT OF THESE CHANGES IS TO CONSIDERABLY INCREASE THE NUMBER OF PLACES WHERE DUMPINFO AND LOADINFO MAY APPEAR IN ORDER TO PRODUCE AND USE AN INFO FILE SUITABLE FOR SEPARATE COMPILATION. CAUTION IS GENERALLY REQUIRED ONLY WHEN VARIABLES WITH THE SAME NAME ARE DECLARED AT DIFFERENT LEVELS; A SEPARATE COMPILATION WILL ONLY BE ABLE TO ACCESS THE LAST SUCH VARIABLE SEEN BEFORE THE LOADINFO OCCURRED.

D0218 ALGOL - CASE STATEMENT SYNTAX - 01-29-73

THE BNF IN THE ALGOL COMPILER IS IN ERROR. IT SHOULD READ AS FOLLOWS ON PAGE 9-5:

<CASE BODY>::= BEGIN<CASE COMPOUND STATEMENT>END/
BEGIN<CASE COMPOUND STATEMENT>;END/
<CASE COMPOUND STATEMENT>::= <STATEMENT>/

D0218 ALGOL - CASE STATEMENT SYNTAX - 01-29-73

<CASE COMPOUND STATEMENT>;<STATEMENT>

D0219 ALGOL - TASKVALUE - 01-15-73

THE TASK ATTRIBUTE "TASK VALUE" HAS BEEN CHANGED FROM INTEGER TO REAL TYPE.

D0220 ALGOL - INSTALLATION INTRINSICS - 02-05-73

THE INSTALLATION DOLLAR CARD OPTION HAS BEEN MODIFIED TO PERMIT GREATER SELECTIVITY IN CHOOSING INSTALLATION INTRINSICS. THE NEW FORMAT OF THIS OPTION IS:

<INSTALLATION DOLLAR OPTION>::= INSTALLATION <NUMBER-LIST>
<NUMBER-LIST>::= <NUMBER-ELEMENT>/<NUMBER-ELEMENT><NUMBER-LIST>/
<NUMBER-ELEMENT>,<NUMBER-LIST>/<EMPTY>
<NUMBER-ELEMENT>::= <INSTALL-NO1>/<INSTALL-NO1>-<INSTALL-NO2>

INSTALL-NO1 AND INSTALL-NO2 ARE UNSIGNED INTEGERS BETWEEN ONE AND 2047, INCLUSIVE. NUMBER-ELEMENTS MUST BE IN ASCENDING SEQUENCE, WITH NO NUMBER REPEATED.

EXAMPLES OF CORRECT SETTINGS ARE:

$INSTALLATION 1 2 3 4,15,22
$INSTALLATION 100-359, 400, 455-457 460-463
$INSTALLATION 100-2047
$INSTALLATION

THE INSTALLATION CARD WITH NO NUMBER-LIST IS EQUIVALENT TO THE NUMBER-LIST 100-2047.

PRIOR TO PROCESSING THE FIRST ALGOL STATEMENT IN ANY COMPILATION, THE COMPILER EXAMINES THE INTRINSICS OPTION. ONLY THE NUMBER-LIST (OR LACK THEREOF) FOLLOWING THE LAST SETTING OF THE INSTALLATION OPTION ENCOUNTERED PRIOR TO THIS TIME IS CONSIDERED.

INSTALLATION INTRINSICS LOADED ARE THOSE THAT WERE EITHER INCLUDED

IN A RANGE OR EXPLICITLY STATED ON THE LAST INSTALLATION SETTING ENCOUNTERED.

A SYNTAX ERROR WILL BE EMITTED IF THE NUMBER-LIST IS NOT IN ASCENDING SEQUENCE, IF ANY OF THE RANGES SPECIFIED OVERLAP, OR IF THE SECOND NUMBER IN A RANGE IS NOT LARGER THAN THE FIRST NUMBER. NUMBERS LARGER THAN 2047 WILL BE TREATED AS IF THEY WERE 2047.

ANY SPECIFICATION OF THE INSTALLATION OPTION THAT IS ENCOUNTERED AFTER THE FIRST STATEMENT OF THE ALGOL COMPILATION CONTINUES TO BE IGNORED.

D0252 ALGOL - "ON" STATEMENT SYNTAX - 01-29-73

THIS CHANGE EXPANDS THE CAPABILITIES OF THE <ON STATEMENT> FOR ALGOL, DCALGOL, AND ESPOL.

NEW SYNTAX:

```
<ON STATEMENT>::= <ENABLING ON STATEMENT>/<DISABLING ON STATEMENT>
<DISABLING ON STATEMENT>::= ON<FAULT LIST>
<ENABLING ON STATEMENT::= ON<FAULT LIST> <FAULT INFORMATION PART>,
                          <FAULT ACTION>/ON<FAULT LIST>
                          <FAULT INFORMATION PART>:<FAULT ACTION>
<FAULT LIST>::= <FAULT NAME>/<FAULT LIST>OR<FAULT NAME>
<FAULT NAME>::= ZERODIVIDE/EXPONENTOVERFLOW/EXPONENTUNDERFLOW/
                INVALIDINDEX/INTEGEROVERFLOW/INACTIVEQUEUE/
                MEMORYPROTECT/INVALIDOP/LOOP/MEMORYPARITY/
                SCANPARITY/INVALIDADDRESS/STACKOVERFLOW/
                STRINGPROTECT/PROGRAMMEDOPERATOR/BOTTOMOFSTACK/
                SEQUENCE/INVALIDPROGRAMWORD/STACKUNDERFLOW/
                ANYFAULT
<FAULT INFORMATION PART>::= <EMPTY>/[<FAULT STACK HISTORY>]/
                            [<FAULT STACK HISTORY>:<FAULT NUMBER>]/
                            [:<FAULT NUMBER>]
<FAULT STACK HISTORY>::= <ARRAY ROW>/<POINTER EXPRESSION>
<FAULT NUMBER>::= <VARIABLE>
```

<FAULT ACTION>::= <STATEMENT>

NOTE: INACTIVEQUEUE IS VALID ONLY IN DCALGOL. BOTTOMOFSTACK, SEQUENCE, AND STACKUNDERFLOW ARE VALID ONLY IN ESPOL.

EXAMPLES:

```
1. ON ZERODIVIDE OR INVALIDINDEX [:A[J]],
       BEGIN
       J:=J + 1; CLEANUP; GO TO L1;
       END
2. ON ANYFAULT;
3. ON MEMORYPROTECT OR LOOP : Q:= 2;
4. ON ZERODIVIDE OR INTEGEROVERFLOW;
5. ON ANYFAULT [POINTR + 2 : Z], HANDLFAULTS(Z);
6. ON EXPONENTOVERFLOW [A[*]], RECOVER(A);
7. ON ANYFAULT [:J]:
       BEGIN
       IF J= 6 THEN GO L2;
       CASE J OF
           BEGIN
            .
            .
            .
           END
       END;
```

NEW SEMANTICS:

THE NEW <FAULT LIST> REPLACES THE <FAULT NAME> IN THE OLD <ON STATEMENT>, AND ENABLES THE USER TO ARM SEVERAL FAULTS WITH RESPECT TO THE SAME <FAULT ACTION> (SEE EXAMPLES ONE AND THREE ABOVE), OR TO DISARM ONE OR MORE FAULTS AT THE SAME TIME (SEE EXAMPLE FOUR ABOVE). NOTE THAT THE OCCURRENCE OF ANY ONE OF THE FAULTS IN THE <FAULT LIST> IS SUFFICIENT TO CAUSE TRANSFER OF CONTROL TO THE <FAULT ACTION>.

THE NON-<EMPTY> <FAULT INFORMATION PART> PROVIDES THE USER WITH THE

STACK HISTORY AT THE TIME OF THE OCCURRENCE OF THE FAULT, AND/OR THE NUMBER CORRESPONDING TO THE FAULT KIND (USEFUL ONLY WHEN MORE THAN ONE FAULT IS ARMED WITH RESPECT TO THE SAME <FAULT ACTION>). THE FAULT NUMBER, WHEN INDICATED, WILL BE SET TO ONE OF THE FOLLOWING VALUES UPON OCCURRANCE OF THE CORRESPONDING FAULT:

1 ZERODIVIDE
2 EXPONENTOVERFLOW
3 EXPONENTUNDERFLOW
4 INVALIDINDEX
5 INTEGEROVERFLOW
6 INACTIVEQUEUE (DCALGOL ONLY)
7 MEMORYPROTECT
8 INVALIDOP
9 LOOP
10 MEMORYPARITY
11 SCANPARITY
12 INVALIDADDRESS
13 STACKOVERFLOW
14 STRINGPROTECT
15 PROGRAMMEDOPERATOR
16 BOTTOMOFSTACK (ESPOL ONLY)
17 SEQUENCE (ESPOL ONLY)
18 INVALIDPROGRAMWORD
19 STACKUNDERFLOW (ESPOL ONLY)

THE FORMAT OF THE STACKHISTORY IS THE STANDARD FORMAT:

SSS:AAAA:Y,#SSS:AAAA:Y,#...,#SSS:AAAA:Y.

OR

SSS:AAAA:Y#(DDDDDDDD),#...,#SSS:AAAA:Y#(DDDDDDDD).

WHERE SSS IS THE SEGMENT NUMBER, AAAA IS THE ADDRESS, Y IS THE SYLLABLE, # IS A BLANK SPACE, DDDDDDDD IS THE LINE NUMBER (ONLY PRESENT IF LINEINFO WAS SET DURING PROGRAM COMPILATION). THE PERIOD (.) ALWAYS TERMINATES THE LAST ENTRY.

THUS, IN EXAMPLE FIVE ABOVE, THE STACKHISTORY WOULD BEGIN AT POINTR + 2 AND CONTINUE UNTIL EITHER THE AREA OR THE STACKHISTORY

INFORMATION WAS EXHAUSTED.

NOTE THAT THE <FAULT STACKHISTORY> AND THE <FAULT NUMBER> ARE FIXED, WITH RESPECT TO ADDRESS, WHEN THE <ON STATEMENT> IS EXECUTED (I.E. WHEN THE FAULT IS ARMED), NOT WHEN THE FAULT OCCURS. THUS, IN THE <ON STATEMENT>

```
ON ZERODIVIDE [A[I,*]:B[J]] : Q:=B[J] + Q
```

THE <ARRAY ROW> A[I,*] IS DETERMINED BY THE VALUE OF I AT THE EXECUTION OF THE <ON STATEMENT>, AND NOT WHEN ANY ZERODIVIDE ACTUALLY OCCURS; SIMILARLY FOR THE <VARIABLE> B[J] AND J.

THE NEW FORM OF THE <ON STATEMENT>, ON<FAULT LIST><FAULT INFORMATION PART>:<FAULT ACTION>, TO BE KNOWN AS THE "GOTO" FORM, DOES NOT REQUIRE THE USER TO DO A "BAD GO TO" OUT OF THE <FAULT ACTION>; THIS "BAD GO TO" WILL HAVE BEEN PERFORMED BY THE SYSTEM. CONSEQUENTLY, PROGRAM CONTROL CAN CONTINUE FROM THE <FAULT ACTION>. FOR EXAMPLE, IN

```
   BEGIN
   ARRAY Z, Q[0:9999];
     .
     .
     .
   READ (FIL, 10000, Q);
   I:=0;
   ON ZERODIVIDE: Q[I]:=1.0E-47;
L: Z[I]:=SQRT(6.32/Q[I]);
   IF (I:=I + 1)<10000 THEN GO TO L1;
     .
     .
     .
   END;
```

THIS EXAMPLE USES THE HARDWARE TO CHECK THE VALUE OF Q[I] FOR ZERO, INSTEAD OF DOING SO EXPLICITLY (THE FORMER IS GENERALLY FASTER).

NOTE THAT THE STACK FOR THE "GOTO" CASE HAS BEEN CUT-BACK BEFORE CONTROL IS TRANSFERRED TO THE <FAULT ACTION>.

D0252 ALGOL = "ON" STATEMENT SYNTAX = 01-29-73

THE <DISABLING ON STATEMENT> DISABLES, OR DISARMS, THOSE FAULTS CORRESPONDING TO THE <FAULT NAME>S IN THE <FAULT LIST>.

IN GENERAL, THE EXECUTION OF THE <ON STATEMENT> SHOULD BE FASTER THAN PREVIOUSLY; MOREOVER, NO BLOCK EXIT IS REQUIRED TO DEACTIVATE THE ARMED FAULTS FOR THE BLOCK.

D0253 ALGOL = "WRITEAFTER" DOLLAR CARD OPTN = 02-26-73

THIS CHANGE IMPLEMENTS THE ABILITY TO WRITE AFTER CARRIAGE CONTROL. THIS IS A DOLLAR OPTION SET AROUND A FILE DECLARATION, A SWITCH FILE DECLARATION OR AN I/O STATEMENT. IF SET AROUND A FILE DECLARATION IT WILL PERTAIN TO ALL I/O STATEMENTS WHERE THAT FILE NAME EXPLICITLY APPEARS.

IF WRITEAFTER IS SET AROUND A SWITCH FILE DECLARATION, IT PERTAINS TO THOSE I/O STATEMENTS EXPLICITLY USING THE SWITCH FILE.ID. IF SET AROUND AN I/O STATEMENT, IT JUST PERTAINS TO THAT I/O STATEMENT.

D0266 ALGOL = MODEL I = 10-16-72

THIS CHANGE GENERATES MODEL II CODE BY DEFAULT. TO GENERATE MODEL I CODE $SET MODELI HAS TO BE IN THE PATCH DECK WHEN COMPILING ALGOL.

D0267 ALGOL = ACCEPT AS BOOLEAN INTRINSIC = 01-29-73

THE ACCEPT STATEMENT NOW RETURNS A BOOLEAN VALUE. IF IT IS USED IN A BOOLEAN EXPRESSION, THE PROGRAM WILL NOT STOP AND WAIT FOR A MESSAGE BUT WILL ACCEPT ONE ONLY IF THE OPERATOR HAS ALREADY TYPED ONE IN. WHEN A MESSAGE IS WAITING, THE VALUE OF THE EXPRESSION WILL BE TRUE, OTHERWISE IT WILL BE FALSE.

EXAMPLE:

```
IF ACCEPT (P) THEN
```

D0267 ALGOL - ACCEPT AS BOOLEAN INTRINSIC - 01-29-73

```
BEGIN
    REPLACE P BY "STILL RUNNING";
    DISPLAY (P);
END;
```

D0274 ALGOL - MEMORYDUMP - 03-23-73

THIS CHANGE IMPLEMENTS THE ABILITY TO CALL MEMORYDUMP IN DCALGOL. THE SYNTAX IS:

```
MEMORYDUMP(STRING);
```

WHERE STRING HAS TO BE A SIX(6) OR EIGHT(8) BIT STRING.

D0278 ALGOL - ALGOL - USR CONTROLD SEGMNTATN - 03-23-73

PARAGRAPH FOUR OF D0007 OF SYSTEM MISCELLANEA SHOULD BE MODIFIED SO AS TO READ:

ONLY PROCEDURES MAY BE INCLUDED IN A USER SEGMENT.

BACKUP

D0179 BACKUP - RECORD COUNT - 10-28-72

IF THE RECORD OPTION IS USED FOR BACKUP, THE RECORD COUNT WILL INCLUDE THE THREE HEADER RECORDS PLUS THE ONE BLANK RECORD OF THE FILE LABEL IF IT EXISTS. THUS, PRINTING RECORD 15 OF A LABELED FILE WHICH HAS GONE TO BACKUP WILL ACTUALLY PRINT RECORD 11 OF THE FILE. FOR EXAMPLE, TO OBTAIN THE FIRST THROUGH THE TWENTIETH RECORDS, THE BIAS OF FOUR IS USED (RECORD 5 24).

D0180 BACKUP - LANGUAGE KEY SPECIFIER - 10-28-72

THE ALGOL, COBOL, AND FORTRAN KEY SPECIFIERS ARE SIGNIFICANT ONLY TO PRINTER (SYMBOLIC) FILES AND THEREFORE, SHOULD NOT BE USED FOR PUNCH FILES.

D0181 BACKUP - "END" AS RANGE STOP INDICATOR - 10-28-82

THE EBCDIC STRING "END" MAY BE USED AS A POSSIBLE RANGE STOP INDICATOR (BESIDES EITHER <INTEGER> OR <"EBCDIC STRING">) WHICH IS EQUIVALENT TO SETTING THE STOP INTEGER TO 99999999.

BASIC

D0152 BASIC - DOLLAR CARD STATEMENT - 01-15-73

BASIC NOW PROVIDES THE USER WITH THE ABILITY TO CONTROL COMPILER OPTIONS WITH A BASIC PROGRAM STATEMENT.

EXAMPLE:

```
100 $ SET LINEINFO
```

THIS IS PROVIDED MAINLY FOR BASIC PROGRAMS ENTERED THROUGH CANDE WHERE ALL BASIC STATEMENTS MUST BEGIN WITH A LINE NUMBER.

SINCE THIS IS A STATEMENT AND NOT A DOLLAR CARD, THE OPTION "$", WHICH CONTROLS THE PRINTING OF DOLLAR CARDS, WILL HAVE NO EFFECT ON THESE STATEMENTS.

D0154 BASIC - "LENGTH" STRING FUNCTION - 01-15-73

THE STRING FUNCTION "LENGTH", WHICH RETURNS THE LENGTH OF A CHARACTER STRING, MAY NOW BE ABBREVIATED AS "LEN".

EXAMPLE:

```
100 A$ = "ABC"
200 L = LEN(A$)
```

LINE 200 ASSIGNS L THE VALUE OF 3.

D0162 BASIC - BASIC CHARACTER DATA EXTENSION - 12-08-72

THE HANDLING OF CHARACTER STRINGS IN BASIC HAS BEEN EXTENDED IN THE FOLLOWING WAYS:

1. THE MAXIMUM LENGTH OF A CHARACTER STRING HAS BEEN EXPANDED FROM 63 TO 255 CHARACTERS.

2. FIVE SYSTEM FUNCTIONS HAVE BEEN ADDED FOR MANIPULATING CHARACTER STRINGS. THEY ARE:

   A. VAL(S$)

      THIS FUNCTION HAS ONE ARGUMENT, S$. THIS FUNCTION RETURNS A NUMERIC CONSTANT CORRESPONDING TO THE VALUE OF S$. S$ MAY BE A STRING CONSTANT, STRING VARIABLE OR A STRING EXPRESSION, THE CHARACTERS OF WHICH FORM A VALID NUMBER.

      EXAMPLE:

```
100 A$ = "123."
200 B$ = "375E+21"
300 N = VAL(A$ + B$)
400 END
```

      AFTER THE EXECUTION OF THE ABOVE PROGRAM, N WILL HAVE THE VALUE 1.23375E+23.

   B. STR$(N)

      THIS FUNCTION HAS ONE ARGUMENT, N. THE VALUE RETURNED BY THIS FUNCTION IS A STRING CORRESPONDING TO THE VALUE OF N. N MAY BE A NUMERIC CONSTANT, ARITHMETIC VARIABLE OR ARITHMETIC EXPRESSION. THE STRING RETURNED BY STR$ IS THE SAME AS IF THE VALUE OF N HAD BEEN PRINTED WITH A "PRINT" STATEMENT.

      EXAMPLE:

```
100 S$ = STR$(123)
200 END
```

      AFTER THE EXECUTION OF THE ABOVE PROGRAM, S$ WILL HAVE THE VALUE "123".

C. EXT$(S$,S,E)

THIS FUNCTION "EXTRACTS" A DESIGNATED SEGMENT OF A CHARACTER STRING. THE ARGUMENT S IS THE NUMBER OF THE CHARACTER POSITION IN S$ AT WHICH THE EXTRACTION IS TO START AND THE ARGUMENT E IS THE NUMBER OF THE CHARACTER POSITION IN S$ AT WHICH THE EXTRACTION IS TO END. S AND E MAY BE NUMERIC CONSTANTS, ARITHMETIC VARIABLES OR ARITHMETIC EXPRESSIONS. S$ MAY BE A STRING CONSTANT, STRING VARIABLE OR STRING EXPRESSIONS.

EXAMPLE:

```
100 A$ = "1234567"
200 E$ = EXT$(A$,3,6)
300 END
```

AFTER THE EXECUTION OF THE ABOVE PROGRAM, E$ WILL HAVE THE VALUE "3456".

D. SCN(S$,O$,N,S)

THIS FUNCTION RETURNS AS A VALUE THE NUMBER OF THE CHARACTER POSITION AT WHICH A SPECIFIED OCCURRENCE OF A STRING SEGMENT OCCURS WITHIN ANOTHER STRING. THE FUNCTION RETURNS A VALUE OF ZERO IF THE SPECIFIED OCCURRENCE OF THE STRING SEGMENT DOES NOT EXIST.

S$ IS THE STRING IN WHICH THE STRING SEGMENT OCCURS AND O$ IS THE SEGMENT WHICH IS DESIRED. THE ARGUMENT N IS THE NUMBER OF THE OCCURRENCE WHICH IS DESIRED AND THE ARGUMENT S IS THE NUMBER OF THE CHARACTER POSITION IN S$ AT WHICH THE "SCAN" IS TO BEGIN. S$ AND O$ MAY BE STRING CONSTANTS, STRING VARIABLES

OR STRING EXPRESSIONS AND S AND E MAY BE NUMERIC CONSTANTS, ARITHMETIC VARIABLES OR ARITHMETIC EXPRESSIONS.

EXAMPLE:

```
100 S$ = "ABBCBBDBBEBB"
200 P$ = "BB"
300 A = SCN(S$,P$,1,1)
400 B = SCN(S$,P$,1,4)
500 C = SCN(S$,P$,2,7)
600 D = SCN(S$,P$,5,1)
700 END
```

AFTER THE ABOVE PROGRAM IS EXECUTED A WILL HAVE THE VALUE TWO, B WILL HAVE THE VALUE FIVE, C WILL HAVE THE VALUE 11 AND "D" WILL HAVE THE VALUE ZERO BECAUSE STARTING WITH THE FIRST CHARACTER OF S$ THERE IS NO FIFTH OCCURRENCE OF P$.

E. REP$(S$,O$,N$,N,S)

THIS FUNCTION RETURNS A STRING WHICH IS FORMED BY "REPLACING" SPECIFIED OCCURRENCES OF A SEGMENT OF A STRING WITH NEW STRING SEGMENTS.

THE ARGUMENT S$ IS THE SOURCE STRING IN WHICH THE "REPLACEMENTS" ARE TO OCCUR. O$ IS THE STRING SEGMENT WHICH IS TO BE REPLACED BY THE NEW SEGMENT N$. THE ARGUMENT N SPECIFIES HOW MANY OCCURRENCES OF O$ ARE TO BE REPLACED AND THE ARGUMENT S SPECIFIES THE NUMBER OF THE CHARACTER POSITION AT WHICH THE SEARCH AND REPLACEMENTS ARE TO BEGIN. AS WITH THE OTHER FUNCTIONS THE STRING ARGUMENTS MAY BE STRING CONSTANTS, STRING VARIABLES OR STRING EXPRESSIONS AND THE NUMERIC ARGUMENTS MAY BE

NUMERIC CONSTANTS, ARITHMETIC VARIABLES OR ARITHMETIC EXPRESSIONS.

IF THE ARGUMENT N (THE NUMBER OF REPLACEMENTS TO BE MADE) IS LESS THAN ZERO, THEN ALL OCCURRENCES OF O$, STARTING WITH THE STH CHARACTER IN THE STRING S$ WILL BE REPLACED. IN THIS CASE THE STRING TO BE REPLACED (I.E., O$) CANNOT BE THE NULL STRING.

IF N IS EQUAL TO ZERO NO REPLACEMENTS ARE MADE AND THE VALUE RETURNED BY REP$ IS A STRING EQUIVALENT TO S$.

WHEN N IS GREATER THAN ZERO, THEN BEGINNING WITH CHARACTER S OF S$, N OCCURRENCES OF O$ ARE REPLACED BY N$.

EXAMPLE:

```
100 A$ = ABBCBBDBBEBB"
200 B$ = "BB"
300 D$ = REP$(A$,B$,"-",1,4)
400 E$ = REP$(A$,B$,"X",-1,7)
500 F$ = REP$(A$,B$,"#",2,1)
600 END
```

AFTER EXECUTION OF THE ABOVE PROGRAM D$ WILL HAVE THE VALUE "ABBC-DBBEBB", E$ WILL HAVE THE VALUE "ABBCBBDXEX" AND "F$" WILL HAVE THE VALUE "A#C#DBBEBB".

D0163 BASIC - TIME AND DATA FUNCTIONS - 11-12-72

THE B6700 BASIC LANGUAGE NOW PROVIDES THE PROGRAMMER WITH THREE TIME FUNCTIONS AND TWO DATA FUNCTIONS WHICH CAN BE REFERENCED BY A BASIC PROGRAM.

TIME FUNCTIONS:

1. CLK$

THIS FUNCTION PROVIDES THE TIME OF DAY AS A STRING SIX CHARACTERS LONG IN THE FORM:

HH:MMB

WHERE "HH" IS HOURS, "MM" IS MINUTES, AND "B" IS A BLANK SPACE. THE TIME IS BASED ON A 24-HOUR CLOCK.

2. BCL

THIS FUNCTION RETURNS THE TIME OF DAY, BASED ON A 24-HOUR CLOCK, IN HOURS AND DECIMAL FRACTIONS OF THE HOUR. FOR EXAMPLE, A PROGRAM RUN AT 3:45 PM WHICH REFERENCES "BCL" WILL RECEIVE AS A VALUE FOR "BCL" THE NUMBER 15.75.

3. TIM

THIS FUNCTION RETURNS THE ELAPSED PROCESSOR TIME SINCE THE JOB BEGAN. THIS TIME IS EXPRESSED IN SECONDS AND DECIMAL FRACTIONS OF SECONDS.

DATE FUNCTIONS:

1. DAT$

THIS FUNCTION PROVIDES THE CURRENT DATE AS A STRING EIGHT CHARACTERS LONG IN THE FORM:

MM/DD/YY

WHERE "MM" IS THE CURRENT MONTH, "DD" IS THE CURRENT DAY, AND "YY" IS THE CURRENT YEAR.

2. IDA

THIS FUNCTION RETURNS A SIX-DIGIT INTEGER OF THE FORM:

YYMMDD

WHERE "YY" IS TWO DIGITS REPRESENTING THE YEAR, "MM" IS TWO DIGITS REPRESENTING THE MONTH, AND "DD" IS TWO DIGITS

REPRESENTING THE DAY.

ALL FIVE FUNCTIONS MAY BE USED DIRECTLY IN A "PRINT" STATEMENT (E.G., "100 PRINT DAT$"). ALSO, THE FUNCTIONS "CLK$" AND "DAT$" MAY BE ASSIGNED TO STRING VARIABLES, AND "TIM", "BCL", AND "IDA" MAY BE ASSIGNED TO NUMERIC VARIABLES.

DO164 BASIC - RESTORE & DATA STMT EXTENSIONS - 11-05-72

THE "DATA" STATEMENT IN BASIC HAS BEEN EXTENDED SO THAT NOW STRING DATA AND NUMERIC DATA ARE STORED INTERNALLY AS TWO SEPARATE DATA LISTS.

IN ADDITION, IT IS ALSO NO LONGER NECESSARY TO ENCLOSE IN QUOTES STRING CONSTANTS APPEARING IN DATA STATEMENTS. FOR UNQUOTED STRINGS IN DATA STATEMENTS LEADING BLANKS ARE IGNORED, BUT ANY BLANKS FOLLOWING THE STRING ARE NOT IGNORED. AN UNQUOTED STRING IS CONSIDERED TO BE ANY DATA LIST ELEMENT WHICH DOES NOT BEGIN WITH A QUOTE, NUMBER, +, -, ., OR @.

EXAMPLE:

```
10 READ  A, B, X$, Y$
20 PRINT "A=" A, "B=" B
30 DATA FIRST THING, 9, 17, "46"
40 PRINT "X$=" X$, "Y$=" Y$
50 END
```

WHEN COMPILED AND EXECUTED THIS PROGRAM WILL PRODUCE THE FOLLOWING OUTPUT:

```
A= 9                B= 17
X$=FIRST THING      Y$=46
```

IN THE EXAMPLE, LINE 10 CONTAINS TWO NUMERIC VARIABLES FOLLOWED BY TWO STRING VARIABLES AND THE DATA ELEMENTS IN LINE 30 ARE ORDERED WITHOUT REGARD TO WHETHER THEY ARE STRING OR NUMERIC CONSTANTS. THE ONLY SIGNIFICANT CONSIDERATION IS THAT THE NUMERIC DATA ITEMS APPEAR IN THE ORDER IN WHICH THEY ARE TO BE ASSIGNED TO THE

VARIABLES A AND B, AND THAT THE STRING DATA ITEMS APPEAR IN THE ORDER IN WHICH THEY ARE TO BE ASSIGNED TO THE STRING VARIABLES X$ AND Y$.

THE "RESTORE" STATEMENT HAS ALSO BEEN EXPANDED TO HANDLE THE TWO SEPARATE LISTS FOR DATA STATEMENTS.

THE RESTORE STATEMENT NOW HAS THE FOLLOWING FORMAT:

```
RESTORE
  OR
RESTORE *
  OR
RESTORE $
```

THE FORM "RESTORE" INSTRUCTS THE PROGRAM TO RETURN TO THE START OF THE NUMERIC DATA LIST AND TO RETURN TO THE START OF THE STRING DATA LIST FOR THE NEXT CONSTANTS TO BE ASSIGNED IN A READ STATEMENT. "RESTORE *" INSTRUCTS THE PROGRAM TO RETURN TO THE START OF THE NUMERIC DATA LIST ONLY. THE FORM "RESTORE $" INSTRUCTS THE PROGRAM TO RETURN TO THE START OF THE STRING DATA LIST ONLY.

EXAMPLE:

```
 10 DATA STRING1, 13, 76, STRING2
 20 READ A,B
 30 PRINT "AT 30 A AND B=" A B
 40 READ X$
 50 PRINT "AT 50 X$=" X$
 60 RESTORE $
 70 READ Y$
 80 PRINT "AT 80 Y$=" Y$
 90 RESTORE *
100 READ C
110 PRINT "AT 110 C=" C
120 RESTORE
130 READ A$, B$, X, Y
140 PRINT X, Y, A$, B$
150 END
```

DO164 BASIC - RESTORE & DATA STMT EXTENSIONS - 11-05-72

WHEN COMPILED AND EXECUTED, THIS PROGRAM WILL PRODUCE THE FOLLOWING OUTPUT:

```
AT 30 A AND B= 13 76
AT 50 X$= STRING1
AT 80 Y$= STRING1
AT 110 C= 13
13      76      STRING1      STRING2
```

DO177 BASIC - STRING VARIABLE NAMES - 11-05-72

BASIC NOW ACCEPTS STRING VARIABLE NAMES CONSISTING OF A LETTER, FOLLOWED BY A SINGLE DIGIT, FOLLOWED BY A DOLLAR SIGN. FOR EXAMPLE, "A9$" IS NOW A VALID STRING VARIABLE NAME IN BASIC.

NOTE: THIS APPLIES ONLY TO STRING VARIABLES AND NOT TO STRING ARRAY NAMES. STRING ARRAY NAMES MAY ONLY BE IN THE FORM "A$".

DO178 BASIC - STRING FUNCTIONS - 11-05-72

BASIC PROGRAMS MAY NOW CONTAIN USER DEFINED STRING FUNCTIONS. THE USER MUST DEFINE HIS STRING FUNCTIONS USING THE "DEF" STATEMENT.

FORMAT:

1. N DEF FNA$ = E
2. N DEF FNA$(D) = E

WHERE N IS THE LINE NUMBER OF THE DEF STATEMENT, A IS AN ALPHABETIC CHARACTER INSERTED BY THE USER, E IS AN EXPRESSION WHICH, WHEN EVALUATED, YIELDS A STRING, AND D IS A LIST OF ONE OR MORE DUMMY VARIABLES SEPARATED BY COMMAS.

A DUMMY VARIABLE CAN BE A STRING VARIABLE OR AN ARITHMETIC VARIABLE AND IS USED ONLY AS A PLACE HOLDER FOR THE VARIABLE WHICH WILL BE USED WHEN THE FUNCTION IS CALLED.

THE USER DEFINED STRING FUNCTION MAY BE USED IN A BASIC PROGRAM

D0178 BASIC - STRING FUNCTIONS - 11-05-72

WHEREVER A STRING VARIABLE CAN BE USED.

EXAMPLE:

```
100 DEF FN C$(X$,Y$,S) = X$ + SPACE(S) + Y$
200 LET A$ = "A VERY"
300 LET B$ = "GOOD BEGINNING"
400 LET F$ = FNC$(A$,B$,10)
500 END
```

AFTER THE EXECUTION OF LINE 400, F$ WILL CONTAIN THE STRING "A VERY GOOD BEGINNING".

D0182 BASIC - MULTIPLE STMT "DEF" FUNCTIONS - 11-05-72

THE "DEF" STATEMENT HAS BEEN EXPANDED SO THAT THE BASIC PROGRAMMER MAY NOW WRITE MULTIPLE STATEMENT FUNCTIONS FOR THOSE FUNCTIONS WHICH CANNOT BE EXPRESSED IN A SINGLE LINE "DEF" STATEMENT. THE FIRST STATEMENT OF A MULTIPLE LINE FUNCTION IS OF THE FORM:

1. N DEF FNL
2. N DEF FNL (D)

WHERE "N" IS THE LINE NUMBER OF THE "DEF" STATEMENT, "L" IS AN ALPHABETIC CHARACTER INSERTED BY THE USER, AND "D" IS A LIST OF ONE OR MORE DUMMY VARIABLES SEPARATED BY COMMAS.

IF THE FUNCTION RETURNS A STRING AS A VALUE FOR THE FUNCTION, THEN THE "DEF" STATEMENT MUST BE IN ONE OF THE FOLLOWING FORMS:

1. N DEF FNL$
2. N DEF FNL$ (D)

THE FIRST STATEMENT OF THE FUNCTION IS FOLLOWED BY THOSE STATEMENTS WHICH DEFINE THE FUNCTION. THE LAST STATEMENT OF THE FUNCTION MUST BE OF THE FORM:

N FNEND

WHERE "N" IS THE LINE NUMBER OF THE "FNEND" STATEMENT.

THIS STATEMENT INDICATES THE END OF THE MULTIPLE STATEMENT FUNCTION.

"DEF" STATEMENTS MAY NOT BE NESTED. FURTHERMORE, FUNCTIONS DEFINED BY "DEF" STATEMENTS CAN BE EXECUTED ONLY BY INVOKING THE FUNCTION WITH A FUNCTION CALL. SPECIFICALLY, TRANSFER OF PROGRAM CONTROL INTO THE RANGE OF A FUNCTION SUBPROGRAM BY ANY STATEMENT OTHER THAN A FUNCTION CALL IS NOT PERMITTED. ALSO, TRANSFER OF PROGRAM CONTROL FROM THE RANGE OF A FUNCTION SUBPROGRAM IS NOT PERMITTED EXCEPT BY NORMAL RETURN FROM THE SUBPROGRAM (I. E., THE EXECUTION OF A "FNEND" STATEMENT).

THIS "DEF" STATEMENT EXPANSION ALSO PROVIDES THE BASIC PROGRAMMER WITH THE ABILITY TO SPECIFY VARIABLES TO BE USED AS "LOCAL" VARIABLES IN THE FUNCTION. TO SPECIFY A VARIABLE AS BEING "LOCAL", LIST THE VARIABLE NAME IN THE "DEF" STATEMENT FOLLOWING THE FUNCTION NAME AND ARGUMENT LIST.

EXAMPLE:

```
10  FOR I = 0 TO 5
20      X = FNF (I)
30      PRINT I "FACTORIAL=" X
40  NEXT I
50  DEF FNF (N) I, X
60  IF N <> 0 THEN 90
70  FNF = 1
80  GOTO 140
90  X = 1
100 FOR I = 1 TO N
110     X = X * I
120 NEXT I
130 FNF = X
140 FNEND
150 END
```

THE EXAMPLE WILL PRODUCE THE OUTPUT:

```
0 FACTORIAL = 1
1 FACTORIAL = 1
```

```
 2 FACTORIAL = 2
 3 FACTORIAL = 6
 4 FACTORIAL = 24
 5 FACTORIAL = 120
```

LINE 50 DEFINES THE FUNCTION "F" TO HAVE ONE ARGUMENT "N" AND TWO LOCAL VARIABLES "I" AND "X". BY DEFINING "I" AND "X" TO BE LOCAL, THE USE OF THESE VARIABLES IN THE FUNCTION WILL HAVE NO EFFECT ON THE GLOBAL VARIABLES "I" AND "X" USED IN LINES 10 THROUGH 40. LINES 70 AND 130 ASSIGN VALUES TO THE FUNCTIUN "F". FOR A MULTIPLE LINE FUNCTION WHICH IS DEFINED AS A STRING FUNCTION (E. G., "100 DEF FNA$ (B$, X) C$"), THE FUNCTION VALUE ASSIGNMENT MUST SPECIFY THE FUNCTION NAME FOLLOWED BY "$" (E. G., 200 FNA$ = "THIS IS THE FUNCTION VALUE").

D0193 BASIC - DOLLAR OPTION "OLD BASIC" - 12-18-72

THE DOLLAR CARD OPTION "OLDBASIC" (RESET BY DEFAULT) HAS BEEN PROVIDED SO THAT WHEN METHODS OF HANDLING CERTAIN BASIC CONSTRUCTS ARE CHANGED IN B6700 BASIC THE USER MAY, IF HE WISHES, AVOID THESE CHANGES BY SETTING THE OPTION OLDBASIC. WHEN OLDBASIC IS SET PROGRAMS WILL COMPILE AND EXECUTE AS IF THE CHANGE HAD NEVER BEEN MADE. CURRENTLY THE CHANGES DETAILED BELOW HAVE BEEN MADE. HOWEVER, IN THE FUTURE OTHER FEATURES MAY BE IMPLEMENTED FOR WHICH LEAVING THE OLD METHOD IN THE COMPILER MAY BE DESIRABLE. IN THESE CASES SETTING THE OPTION OLDBASIC WILL CAUSE THE CONSTRUCT IN QUESTION TO BE HANDLED AS PREVIOUSLY.

1. THE HANDLING OF THE INPUT STATEMENT HAS BEEN CHANGED SO THAT NOW THE ONLY DELIMITER IN THE INPUT DATA LIST IS A COMMA. (BY SETTING OLDBASIC THE OLD METHOD OF BOTH BLANKS AND COMMAS AS DELIMITERS WILL BE USED). THIS CHANGE ALLOWS BOTH NUMERIC AND UNQUOTED STRING DATA TO CONTAIN EMBEDDED BLANKS, WHEREAS THE OLD METHOD WILL INTERPRET THE BLANKS AS DELIMITERS.

EXAMPLE:

```
100 INPUT    A,S$
200 END
WHERE THE INPUT DATA LIST IS:
    12     345,STRING VALUE,12
```

THE ABOVE EXAMPLE WITH OLDBASIC SET WOULD CAUSE THE VARIABLE, A, TO BE ASSIGNED THE VALUE 12 AND S$ TO BE ASSIGNED THE VALUE "345".

WITH OLDBASIC RESET "A" WOULD BE ASSIGNED THE VALUE 12345 AND "S$" WOULD BE ASSIGNED THE VALUE "STRING VALUE".

2. THE HANDLING OF "MAT INPUT" STATEMENTS HAS BEEN CHANGED SO THAT NOW WHEN A "MAT INPUT" STATEMENT IS EXECUTED IT DOES NOT KEEP READING DATA ELEMENTS UNTIL THE ARRAY IS COMPLETELY FULL. INSTEAD IT WILL FILL THE ARRAY, IN ROW ORDER, UNTIL IT FINDS THE END OF THE DATA LIST. THE DATA LIST CONSISTS OF ONE LINE OF INPUT FROM AN EXTERNAL FILE OR REMOTE DEVICE, OR ONE CARD OF INPUT FROM BATCH MODE (SEE #3 BELOW).

EXAMPLE:

```
100 MAT INPUT A(3,3)
200 MAT PRINT A
300 END
```

WHEN LINE 100 IS EXECUTED THE PROGRAM WILL WAIT FOR THE DATA ELEMENTS FOR MATRIX "A". IF THE INPUT LIST IS:

```
1,2,3,4
```

THEN LINE 200 WILL PRODUCE THE OUTPUT:

```
1    2    3
4    0    0
0    0    0
```

WHEN USING "MAT INPUT" WITH ONE DIMENSIONAL ARRAYS (I.E. VECTORS), IF THE NUMBER OF DATA ELEMENTS INPUT IS LESS

THAN THE SIZE OF THE VECTOR, THEN THE VECTOR WILL BE RESIZED TO BE EQUAL IN SIZE TO THE NUMBER OF ELEMENTS INPUT (SEE RELATED SYSTEM NOTE D0197).

EXAMPLE:

```
100 MAT INPUT A(15)
200 MAT PRINT A
300 END
```

WHEN LINE 100 IS EXECUTED THE PROGRAM WILL WAIT FOR THE DATA FOR MATRIX "A". IF THE INPUT IS:

```
1,2,3,4,5,6
```

THEN "A" WILL BE RESIZED TO CONTAIN ONLY THESE SIX ELEMENTS. ANY ATTEMPT TO INDEX MATRIX "A" WITH AN INDEX VALUE GREATER THAN SIX WILL PRODUCE AN "INVALID INDEX" ERROR. LINE 200 WILL PRODUCE THE OUTPUT:

```
 1   2   3   4   5
 6
```

THUS, A VECTOR CAN BE RESIZED TO AN SIZE, "N", WITH THE STATEMENT:

```
100 MAT INPUT A(N)
```

AND THEN SUPPLYING "N" DATA ELEMENTS WHEN THE STATEMENT IS EXECUTED.

BY SETTING OLDBASIC "MAT INPUT" STATEMENTS WILL CONTINUE TO EXPECT DATA UNTIL THE ARRAY IS COMPLETELY FULL, THUS PREVENTING PARTIAL FILLING OF TWO DIMENSIONAL ARRAYS AND ALSO THE RESIZING THAT MAY OCCUR WITH VECTORS.

3. THE AMPERSAND CHARACTER (I.E. "&") CAN NOW BE USED AS A CONTINUATION CHARACTER FOR DATA BEING SUPPLIED TO A "MAT INPUT" STATEMENT (SEE SYSTEM NOTE D0198 FOR AN EXAMPLE AND RULES ABOUT THE USE OF THE AMPERSAND). WHEN IT IS NOT POSSIBLE TO PUT ALL THE DATA ON ONE LINE OF INPUT (FROM REMOTE DEVICES) OR ONE CARD (FROM BATCH MODE) THE

D0193 BASIC - DOLLAR OPTION "OLD BASIC" - 12-18-72

USE OF THE AMPERSAND AS THE LAST DATA ELEMENT WILL CAUSE THE PROGRAM TO CONTINUE LOOKING FOR DATA.

BY SETTING OLDBASIC THE USE OF THE AMPERSAND AS A CONTINUATION CHARACTER IS NULLIFIED. HOWEVER, IT IS UNNECESSARY SINCE SETTING OLDBASIC WILL CAUSE "MAT INPUT" STATEMENTS TO CONTINUE READING DATA UNTIL THE ARRAY IS COMPLETELY FILLED.

D0194 BASIC - DET FUNCTION IN BASIC - 12-18-72

THE FUNCTION "DET", WHICH RETURNS THE VALUE OF THE DETERMINANT OF THE MATRIX WHICH WAS LAST INVERTED VIA THE "INV" FUNCTION, IS NOW IMPLEMENTED IN B6700 BASIC. DET HAS NO ARGUMENTS AND MAY BE USED IN AN ARITHMETIC EXPRESSION OR BE REFERRED TO DIRECTLY BY A PRINT STATEMENT.

D0195 BASIC - COTANGENT FUNCTION IN BASIC - 12-18-72

THE COTANGENT TRIGONOMETRIC FUNCTION IS NOW IMPLEMENTED IN BASIC. IT IS REFERENCED AS "COT(X)", WHERE X IS THE ANGLE, EXPRESSED IN RADIANS, WHOSE COTANGENT IS TO BE FOUND. IT MAY BE USED ANYWHERE IN A BASIC PROGRAM THAT A BASIC MATHEMATICAL FUNCTION CAN BE REFERENCED.

D0197 BASIC - NUM FUNCTION - 01-15-73

THE NUM FUNCTION IS NOW IMPLEMENTED IN B6700 BASIC. THIS FUNCTION RETURNS THE NUMBER OF DATA ELEMENTS ENTERED INTO THE LAST ARRAY WHICH WAS FILLED BY A MAT INPUT STATEMENT.

IF THE ARRAY HAS ONLY ONE DIMENSION (I.E., IT IS A VECTOR) AND THE NUMBER OF DATA ELEMENTS INPUT TO THAT VECTOR VIA THE MAT INPUT STATEMENT IS LESS THAN THE SIZE OF THE VECTOR, THEN THE VECTOR WILL

BE RESIZED TO BE AS LARGE AS THE NUMBER OF DATA ELEMENTS WHICH WERE INPUT. IN THIS CASE NUM WILL CONTAIN THE NUMBER OF DATA ELEMENTS INPUT, WHICH ALSO WILL BE THE NEW SIZE OF THE VECTOR.

EXAMPLE:

```
100 DIM A(15)
200 MAT INPUT A
300 PRINT "NUM =" NUM
400 A(7) = 10
500 END
```

WHEN THE ABOVE PROGRAM IS EXECUTED THE STATEMENT IN LINE 200 WILL WAIT UNTIL THE DATA ELEMENTS FOR THE ARRAY A HAVE BEEN SUPPLIED. IF THE DATA IS THE VALUES 1, 2, 3, 4 AND 5, THEN A(1) THROUGH A(5) WILL BE ASSIGNED THE VALUES ONE THROUGH FIVE AND A WILL BE RESIZED TO BE ONLY FIVE ELEMENTS LONG. LINE 300 WILL THEN PRODUCE THE OUTPUT:

```
NUM = 5
```

AFTER THAT LINE 400 WILL GET AN INVALID INDEX ERROR BECAUSE A IS ONLY FIVE LONG AND LINE 400 ATTEMPTED TO INDEX THE SEVENTH ELEMENT OF A. (NOTE: THE PROGRAMMER CAN RESIZE VECTOR A BACK TO 15 ELEMENTS LATER IN THE PROGRAM BY USING THE STATEMENT:

```
350 MAT INPUT A(15)
```

AND THEN WHEN LINE 350 IS EXECUTED SUPPLYING 15 DATA ELEMENTS.)

FOR TWO DIMENSIONAL ARRAYS FILLED BY A MAT INPUT STATEMENT NUM WILL CONTAIN THE NUMBER OF DATA ELEMENTS INPUT TO THE ARRAY. THE ARRAY WILL NOT BE RESIZED IF THE NUMBER OF DATA ELEMENTS INPUT IS LESS THAN THE SIZE OF THE ARRAY.

FOR BOTH TWO DIMENSIONAL AND ONE DIMENSIONAL ARRAYS, IF THE NUMBER OF DATA ELEMENTS IS MORE THAN THE SIZE OF THE ARRAY, THEN THE MAT INPUT STATEMENT WILL IGNORE THE EXCESS DATA AND NUM WILL BE EQUAL TO THE SIZE OF THE ARRAY.

BY SETTING THE OPTION "OLDBASIC" (SEE SYSTEM NOTE DO193) MAT INPUT

D0197 BASIC - NUM FUNCTION - 01-15-73

STATEMENTS WILL, AS IN THE PAST, CAUSE THE ENTIRE ARRAY TO BE FILLED AND THUS AVOID THE RESIZING OF VECTORS.

D0203 BASIC - "ASC" CHARACTER FUNCTION - 01-15-73

THE "ASC" FUNCTION IS NOW IMPLEMENTED IN B6700 BASIC. THE VALUE RETURNED BY THIS FUNCTION IS THE NUMERIC VALUE OF THE SPECIFIED EBCDIC CHARACTER. IF THE CHARACTER CANNOT BE PRINTED AN ABBREVIATION FOR THE CHARACTER MAY BE USED. THE ABBREVIATIONS WHICH THE FUNCTION WILL ACCEPT ARE NUL, SOH, STX, ETX, HT, DEL, VT, FF, CR, SO, SI, DLE, DC1, DC2, DC3, DC4, BS, CAN, EM, FS, GS, RS, US, LF, ETB, ESC, ENQ, ACK, BEL, SYN, EOT, NAK, SUB AND SP.

EXAMPLE:

```
100 PRINT "1 =", ASC(1)
200 PRINT "A =", ASC(A)
300 PRINT "LINE FEED=", ASC(LF)
400 PRINT "NUL =", ASC(NUL)
500 END
```

WHEN EXECUTED THE ABOVE PROGRAM WILL PRODUCE THE FOLLOWING OUTPUT:

```
1 = 241
A = 193
LINE FEED = 37
NUL = 0
```

D0204 BASIC - APOSTROPHE - AS COMMENT SIGN - 01-15-73

THE APOSTROPHE CHARACTER CAN NOW BE USED IN BASIC PROGRAM STATEMENTS TO INDICATE THAT THE REMAINDER OF A LINE IS A COMMENT. ANYTHING IN A BASIC STATEMENT FOLLOWING AN APOSTROPHE WILL BE TREATED AS EXPLANATORY REMARKS.

D0205 BASIC - EXPANDED "IF" SYNTAX - 01-15-73

D0205 BASIC - EXPANDED "IF" SYNTAX - 01-15-73

THE "IF" STATEMENT IN BASIC HAS BEEN EXPANDED TO ALLOW STATEMENTS OF THE FORM:

IF <BOOLEAN EXPRESSION> GO TO <LINE NUMBER>

EXAMPLE:

650 IF A<B GO TO 1100

D0206 BASIC - LIMIT DOLLAR CARD OPTION - 01-15-73

THE DOLLAR CARD OPTION "LIMIT" IS NOW IMPLEMENTED IN BASIC. THE PROPER FORMAT IS:

LIMIT <INTEGER>

THE BASIC COMPILER WILL TERMINATE COMPILATION OF A PROGRAM IF THE NUMBER OF SYNTAX ERRORS IN THE PROGRAM EQUALS OR EXCEEDS <INTEGER>. IF THE LIMIT OPTION IS NOT SPECIFIED THE DEFAULT VALUE IS 100 (10 IF CALLED FROM CANDE).

D0207 BASIC - RELATIONAL OPERATORS - 01-15-73

BASIC NOW ALLOWS RELATIONAL OPERATORS OF THE FORM - .EQ.. OTHER ACCEPTABLE OPERATORS ARE .LT., .GT., .LE., AND .NE..

EXAMPLE

100 IF A .NE. B THEN 200

D0208 BASIC - INPUT-OUTPUT STATEMENTS - 01-22-73

B6700 BASIC HAS BEEN EXTENDED SO THAT INPUT LISTS, OF "READ" AND "INPUT" STATEMENTS, AND OUTPUT LISTS, OF "PRINT" STATEMENTS, MAY NOW SPECIFY 48 ITEMS. THE PREVIOUS LIMIT WAS 16.

D0209 BASIC - "INPUT" STATEMENT IN BASIC - 01-08-73

THE "B6700 BASIC LANGUAGE MANUAL" IS IN ERROR IN ITS DESCRIPTION OF THE INPUT STATEMENT. THE MANUAL STATES, "EACH TIME AN INPUT STATEMENT IS EXECUTED, THE NEXT DATA ITEMS IN THE FILE ARE READ." THE MANUAL SHOULD READ "EACH TIME AN INPUT STATEMENT IS EXECUTED, THEN STARTING WITH THE NEXT LINE OF DATA, DATA ITEMS IN THE FILE ARE READ."

EXAMPLE:

```
100 FOR I = 1 TO 3
200 INPUT FILE NUMBERS, A(I),B(I)
300 NEXT I
400 END
```

WHERE FILE "NUMBERS" CONTAINS THE FOLLOWING THREE LINES OF DATA:

```
1, 2, 3, 4
5, 6, 7, 8, 9, 10
11, 12, 13
```

THE MANUAL INCORRECTLY IMPLIES THAT THIS PROGRAM WILL ASSIGN VALUES TO THE MATRICES A AND B SO THAT:

```
A(1) = 1      B(1) = 2
A(2) = 3      B(2) = 4
A(3) = 5      B(3) = 6
```

HOWEVER, THE CORRECT RESULTS ARE

```
A(1) = 1      B(1) = 2
A(2) = 5      B(2) = 6
A(3) = 11     B(3) = 12
```

D0221 BASIC - "PRINT" STATEMENT IMPROVEMENTS - 01-29-73

THE FOLLOWING TWO CHANGES HAVE BEEN MADE TO THE BASIC PRINT

STATEMENT:

1. EACH PRINT STATEMENT WHICH PRINTS TO A FILE WILL NOW BEGIN WRITING ITS DATA ON A NEW LINE AND CONINTUE OVER AS MANY LINES AS NEEDED (AS DOCUMENTED IN THE "B6700 BASIC LANGUAGE" MANUAL PP 4-13).

2. IF THE LAST PRINT STATEMENT (NOT TO A FILE) IN A BASIC PROGRAM IS TERMINATED WITH A COMMA OR A SEMICOLON THAT OUTPUT LIST WILL NOW BE PRINTED WHEN THE PROGRAM IS TERMINATED.

BINDER

D0156 BINDER - ALGOL TO ESPOL BINDING - 01-15-73

IT IS NOW POSSIBLE TO BIND ALGOL AND DCALGOL PROCEDURES TO THE MCP SUBJECT TO THE FOLLOWING RESTRICTIONS:

1. THE PROCEDURE MUST BE LEVEL TWO OR HIGHER.
2. IF THE PROCEDURE IS LEVEL TWO AND REFERS TO MCP GLOBALS, $INTRINSICS MUST BE SET DURING COMPILATION.
3. THE PROCEDURE MAY REFER ONLY TO DECLARED ITEMS WITHIN ITSELF OR IN THE OUTER BLOCK OF THE MCP.

OTHERWISE NORMAL BINDING PROCEDURES APPLY.

CANDE

D0186 CANDE - EXCLUDE COMMAND - 11-20-72

A NEW CANDE EDITING COMMAND, EXCLUDE, HAS BEEN IMPLEMENTED. THE SYNTAX PARALLELS THAT FOR MERGE AND REMERGE--THE KEYWORD "EXCLUDE" (MINIMUM ABBREVIATION "EXC") FOLLOWED BY A FILE NAME AND OPTIONAL SEQUENCE RANGE LIST. ANY LINE IN THE WORKFILE WITH THE SAME SEQUENCE NUMBER AS A LINE IN THE EXCLUDE FILE IS DELETED. (UNMATCHED LINES IN THE EXCLUDE FILE ARE IGNORED.) EXCLUDE MAY BE UTILIZED ALONG WITH FIND-TO-FILE TO DELETE SELECTED LINES IN A FILE.

D0236 CANDE - INCREASE MAXSTATIONS, MAXTASKS - 01-15-73

THE MAXIMUM NUMBER OF STATIONS (MAXSTATIONS) THAT MAY BE ATTACHED TO CANDE HAS BEEN INCREASED FROM 25 TO 35, AND THE MAXIMUM NUMBER OF TASKS WHICH MAY BE ACTIVE AT ONE TIME (MAXTASKS) HAS BEEN INCREASED FROM FIVE TO TEN, IN SYMBOL/CANDE AND SYSTEM/CANDE AS DISTRIBUTED. THE MAXIMUM NUMBER OF SIMULTANEOUS EDITING FUNCTIONS OR LISTINGS (MAXWORKS) REMAINS SIX. ALL THESE PARAMETERS MAY BE VARIED TO SUIT THE LOCAL INSTALLATION, BY RECOMPILING CANDE.

D0237 CANDE - PAGESKIP VARIANT - 01-15-73

CANDE NOW SETS THE PAGESKIP VARIANT RATHER THAN TOGGLE[3] TO INDICATE NEW-PAGE ACTION IN DCWRITE WRITE MESSAGES. THIS FEATURE IS USED IN SENDING FULL-PAGE OUTPUT TO SCREEN DEVICES.

D0238 CANDE - HISTRY PROCSNG FOR RSVP & SNTX - 02-05-73

D0238 CANDE - HISTRY PROCSNG FOR RSVP & SNTX - 02-05-73

CANDE WILL NOW DISPLAY RSVP MESSAGES (NO FILE, ACCEPT, ETC.) AT THE USER STATION WHEN HIS TASK IS WAITING (STED) FOR OPERATOR ATIENTION. NO EXPLICIT REPLY MECHANISM HAS BEEN PROVIDED, ALTHOUGH THE USER MAY ENTER "?DS" TO TERMINATE THE TASK. TASK.HISTORY PROCESSING HAS BEEN FURTHER MODIFIED TO ACCOUNT FOR A NEW VALUE INDICATING THAT A COMPILATION TERMINATED WITH SYNTAX ERRORS; VISIBLE OUTPUT IS NOT EFFECTED.

D0239 CANDE - LOGGING; SESSION NUMBRS; SPLIT - 02-05-73

CANDE LOGGING HAS BEEN MODIFIED IN KEEPING WITH THE NEW WORK-FLOW-MANAGEMENT LOGGING STRUCTURE. THE SYSTEMLOG INTRINSIC IS NO LONGER USED; MCSLOGGER AND DCERRORLOGGER PERFORM SIMILAR FUNCTIONS.

EACH CANDE SESSION IS ASSIGNED A NUMBER (CORRESPONDING TO JOB NUMBER FOR BATCH JOBS). A JOBFILE EXISTS FOR THE SESSION; ALL TASKS RUN IN THAT SESSION ARE LOGGED IN THAT JOBFILE. PRINTER AND PUNCH OUTPUT FROM THESE TASKS ARE GATHERED WITH THE JOBFILE. AND PROCESSED TOGETHER AT THE END OF THE SESSION. THE SESSION NUMBER BECOMES THE JOB NUMBER FOR EACH TASK IN THE SESSION. THE SESSION NUMBER IS DISPLAYED AT LOGON AND LOGOFF.

A NEW COMMAND, SPLIT (MINIMUM ABBREVIATION "SPL"), HAS BEEN ADDED TO ALLOW PROCESSING OF THE ACCUMULATED OUTPUT WITHOUT THE USER LOGGING OFF. SPLIT DIVIDES THE SESSION: THE CURRENT SESSION IS FORMALLY LOGGED OFF (CAUSING PROCESSING OF THE JOB FILE); A NEW SESSION NUMBER IS ASSIGNED AND NEW FORMAL SESSION BEGUN. NO CHANGE IS MADE IN WORKFILE, USERCODE OR CHARGECODE STATUS.

THE TIMES PRINTED AT HELLO, BYE, SPLIT OR CHARGE TIME ARE THE ACCUMULATION SINCE THE PREVIOUS SUCH TIME.

D0240 CANDE - DCP FAULT REPORTING - 02-19-73

CANDE NOW WRITES, ON A SEPARATE PRINTER BACKUP FILE, AN ANNOTATED LISTING OF THE DCP FAULT MESSAGGE WHEN A DATACOM PROCESSOR FAULT

D0240 CANDE - DCP FAULT REPORTING - 02-19-73

CONDITION IS DETECTED. THE LISTING SHOWS THE DCP NUMBER, FAULT NUMBER, AND CONTENTS OF DCP REGISTERS AND SCRATCHPAD MEMORY, APPROPRIATELY LABELED. THE INTNAME AND DEFAULT TITLE OF THE FILE IS "B."

D0275 CANDE - LOG ANALYZER - 03-07-73

THE CANDE COMMAND "LOG" NOW INVOKES SYSTEM/LOGANALYZER, RATHER THAN SYSTEM/LOGOUT. THE <INPUT SPECIFICATIONS> FOLLOWING "LOG" ARE PASSED DIRECTLY TO LOGANALYZER FOR INTERPRETATION. LOGANALYZER DIVIDES ITS PRINTER OUTPUT TO FIT THE TERMINAL WIDTH, AS SPECIFIED IN NDL.

D0276 CANDE - LINE-STATION READY - 03-23-73

SEVERAL CHANGES HAVE BEEN MADE TO MINIMIZE OCCURRANCES OR LINE-NOT-READY OR STATION-NOT-READY- SITUATIONS:

1. THE "?READY" CONTROL COMMAND NOW READIES BOTH LINE AND STATION.
2. THE ERROR COUNT IN CANDE IS RESET WHEN SWITCHED-LINE STATUS CHANGES.
3. WHEN CANDE IS STARTED, IT ATTACHES ALL STATIONS FOR WHICH CANDE IS THE MCS, MAKING LINE AND STATION READY, AND SENDING A MESSAGE TO THOSE THAT ARE CONNECTED OR UNSWITCHED.

NOTE THAT MAXSTATIONS IN CANDE MUST BE LARGE ENOUGH FOR ALL CANDE STATIONS IN THE NETWORK, SINCE ALL ARE ATTACHED IMMEDIATELY.

D0277 CANDE - BUFFER CHAOS TRAP - 03-23-73

THE TRAPS IN CANDE TO DETECT ERRORS IN TANK-BUFFER PROTOCOL HAVE

D0277 CANDE = BUFFER CHAOS TRAP = 03-23-73

BEEN MODIFIED: INSTEAD OF DIVIDING "CHAOS" OR "GOTCHA" BY ZERO, THEY PRODUCE A MEMORY DUMP (BY "CANDE:BUF CHAOS"). THE DUMP IS NOT FATAL TO THE MCP, BUT CANDE WILL BE TERMINATED FOLLOWING THE DUMP. PLEASE FORWARD BOTH MEMORY DUMP AND PROGRAM DUMPS TO TIO. LARGE SYSTEMS PLANT, WITH AN FTR. (THESE TRAPS ARE STILL CONTROLLED BY THE COMPILE-TIME $ OPTIONS PARANOID AND PEDANTIC.)

D0282 CANDE = SWAPPING = 10-30-72

HENCEFORTH, BY DEFAULT, ALL TASKS RUN BY CANDE ARE RUN IN SUBSPACE IF SWAPPER IS RUNNING. COMPILERS AND SYSTEM UTILITY ROUTINES ARE RUN WITH SUBSPACES = 1 (SWAPREENTRANT), SO THE D1 STACK IS IN NORMAL MEMORY AND THE D2 STACK IS IN A SUBSPACE. USER PROGRAMS ARE RUN WITH SUBSPACES = 2 (SWAPSTANDARD): D1 AND D2 STACKS (CODE AND DATA) ARE BOTH IN THE SUBSPACE IF THE CODE FILE IS IN A USERCODE LIBRARY; IF THE CODE FILE IS IN SYSTEMDIRECTORY THEN THE CODE IS REENTRANT AND DATA IS SWAPPED. THE EXECUTION PART OF A COMPILE-AND-GO "RUN" COMMAND HAS SUBSPACES = 3 (SWAPALL): CODE AND DATA ARE BOTH IN SUBSPACE.

A NEW SECONDARY CONTROL STATEMENT (MODIFIER) MAY BE USED TO OVERRIDE THE DEFAULT SETTINGS. THE SYNTAX AND SEMANTICS ARE:

```
SUBSPACES = 0    DO NOT USE SUBSPACE
SUBSPACES = 1    DATA IN SUBSPACE
SUBSPACES = 2    DATA IN SUBSPACE, CODE ALSO IF USER PROGRAM
SUBSPACES = 3    DATA AND CODE IN SUBSPACE
```

(THE EQUAL SIGN IS OPTIONAL.)

EXAMPLES:

```
COMPILE; C SUBSPACES = 0   [COMPILE WITHOUT SWAPPING]
E *SYSTEM/UTILITY; SUBSPACES = 1  [SWAP DATA, NOT CODE]
```

NOTE THAT SUBSPACE SETTING FOR EXECUTION MAY NOT BE SPECIFIED AT COMPILE TIME (WITH EITHER COMPILE OR RUN).

00283 CANDE - EXTEND "BRUTAL" & "PEDANTIC" - 04-03-73

THIS CHANGE CAUSES WAITANDGO TO BE CALLED EACH TIME A GETBLK IS DONE IN ORDER TO INSURE GLOBAL VARIABLES ARE "BRUTALIZED" WHENEVER A BLOCK IS GOTTEN AND THE BRUTAL OPTION IS SET. IT THEN BRUTALIZES VARIABLES BY CALLING AN INSTALLATION INTRINSIC THAT WILL SET ALL PERTINENT GLOBALS TO UNINITIALIZED VARIABLES. THIS CHANGE ALSO CAUSES THE READONLY BIT TO BE SET IN THE DIRECT ARRAYS INTO WHICH TANK BUFFERS ARE READ DEPENDING UPON THE VALUE OF THE READONLY PARAMETER IN GETBLK. THIS ACTION IS TAKEN WHEN THE PEDANTIC OPTION IS SET. TO INSURE THAT WHEN INFORMATION IS CHANGED, AN UPDATED COPY OF THE BUFFER WILL BE REWRITTEN ON DISK.

COBOL

D0224 COBOL - DOLLAR CARD PROCESSING - 11-20-72

THIS CHANGE MAKES CHANGES AND CORRECTIONS TO THE PROCESSING OF $ CARD OPTIONS.

THE OPTION NOLINK HAS BEEN DE-IMPLEMENTED SINCE THERE ARE NO LONGER ANY COMPILATION FUNCTIONS WHICH DEPEND ON ITS SETTING. SPECIFYING NOLINK IN A $ CARD WILL NOW PRODUCE A WARNING MESSAGE.

DOLLAR CARDS MAY NOW APPEAR IN ANY COMPILER INPUT. ANY INPUT IMAGE (WITH OR WITHOUT A SEQUENCE NUMBER) WHICH IS BLANK IN COLUMN SEVEN AND HAS A $ SYMBOL IN COLUMN EIGHT WILL BE RECOGNIZED AS A VALID DOLLAR CONTROL CARD (SETTING AND RESETTING OPTIONS AS SPECIFIED) AND WILL BE WRITTEN TO OUTPUT "SAVE" OR "NEWTAPE" FILES. PRIOR TO IMPLEMENTATION OF THIS CHANGE, DOLLAR CARDS WOULD NOT BE PASSED ONTO COMPILER OUTPUT AND WOULD NOT BE RELIABLY PROCESSED WHEN THEY APPEARED IN COMPILER LIBRARY FILES (VIA THE "FROM" OR "COPY"). INPUT IMAGES WITH A $ SYMBOL IN COLUMN SEVEN ARE FUNCTIONALLY EQUIVALENT TO INPUT IMAGES WHICH HAVE THE $ SYMBOL IN COLUMN EIGHT (AND ARE BLANK IN COLUMN SEVEN) EXCEPT THAT THE LATTER WILL BE WRITTEN TO OUTPUT "SAVE" AND/OR "NEWTAPE" FILES. AN INPUT IMAGE WHICH HAS THE $ SYMBOL IN BOTH COLUMNS SEVEN AND EIGHT WILL INITIALIZE ALL OPTIONS AND SET THE OPTION "$". SINCE THE $ SYMBOL WHICH FLAGGED THIS CARD AS A $ CONTROL CARD IS IN COLUMN SEVEN, IT WILL NOT BE WRITTEN TO OUTPUT FILES. A $-COLUMN EIGHT IMAGE WHICH SPECIFIES VOID, VOIDT, SAVE OR FROM WILL CAUSE A WARNING MESSAGE TO PRINT AND NO PART OF THE IMAGE WILL BE WRITTEN TO ANY COMPILER OUTPUT FILE.

A NEW OPTION, "LIBDOLLAR", HAS BEEN IMPLEMENTED ON WHOSE SETTING THE RECOGNITION OF $ CONTROL CARDS FROM A LIBRARY (WITH THE "FROM" OR "COPY") IS BASED. WHEN THE OPTION IS SET, LIBRARY $ CARDS ARE RECOGNIZED AND PROCESSED. WHEN THE OPTION IS RESET, $ CARDS FROM A

D0224 COBOL - DOLLAR CARD PROCESSING - 11-20-72

LIBRARY ARE TREATED AS COMMENTS. IF LIBDOLLAR APPEARS ON A $ CARD BROUGHT IN FROM A LIBRARY, THE OPTION IS IGNORED AND A WARNING MESSAGE IS PRINTED. THIS OPTION IS SET AUTOMATICALLY BY THE COMPILER. DOLLAR CARD INITIALIZATION ALSO CAUSES IT TO BE SET. IF THIS OPTION HAS NEVER BEEN EXPLICITLY "RESET" A "POP" WILL LEAVE THE OPTION SET.

A FATAL SYNTAX ERROR WILL BE PRODUCED IF A "FROM" OPTION APPEARS WITHIN THE IMAGES BROUGHT IN AS THE RESULT OF A "$ FROM" OR A "COPY".

A PROBLEM WHICH WOULD CAUSE INPUT IMAGES TO BE MISSING FROM THE PASS TWO LISTING HAS BEEN CORRECTED.

A PROBLEM WHICH CAUSED PASS TWO TO TERMINATE WITH AN "EOF NO LABEL" IS CORRECTED.

AN ERRONEOUS WARNING MESSAGE WHICH PRINTED ONLY WHEN THE LAST IMAGE FROM A LIBRARY WAS A $ CARD HAS BEEN REMOVED.

THE LETTER "S" NOW PRINTS ON THE RIGHT SIDE OF THE LISTING FOR IMAGES WHICH COME FROM SAVED INPUT AS A RESULT OF A "$ FROM".

D0226 COBOL - NEW ATTRIBUTES - 02-05-73

THIS CHANGE ADDS SEVEN NEW ATTRIBUTES TO THE LIST OF ALLOWABLE ATTRIBUTES. THE FILE ATTRIBUTES ADDED ARE FLEXIBLE, CURRENTBLOCK, CARRIAGECONTROL, AND TIMELIMIT. THE TASK ATTRIBUTES ARE MAXCARDS, MAXLINES, AND JOBNUMBER. FOR DETAILS OF THE USE OF THESE ATTRIBUTES REFER TO THE MCP OR INPUT-OUTPUT SUBSYSTEM MANUALS.

D0229 COBOL - <DATA-NAME> IS <MNEMONIC> - 02-19-73

THE CONSTRUCT

```
SPECIAL-NAMES.
        <DATA-NAME> IS <MNEMONIC>.
```

D0229 COBOL - <DATA-NAME> IS <MNEMONIC> - 02-19-73

WILL RESULT IN SYNTAX ERROR #0. PAGE 3-6 OF THE COBOL MANUAL IS IN ERROR IN SHOWING THIS AS VALID SYNTAX. DOCUMENTATION WILL BE REVISED.

D0247 COBOL - "CALL SYSTEM" VERB - 02-19-73

THIS CHANGE REMOVES THE RESTRICTION THAT THE DATA-NAME IN THE CALL SYSTEM VERB MUST BE A RECORD WHOSE USAGE IS DISPLAY. DISPLAY-1 RECORD AREAS ARE NOW ACCEPTABLE.

D0249 COBOL - SHORT BLOCK USE ROUTINE - 01-15-73

A USE PROCEDURE DEFINED AS

"USE AFTER RECORD SIZE ERROR ON <FILE-NAME>"

WILL BE EXECUTED FOR EACH RECORD OF A BLOCK WHICH IS LESS THAN THE SPECIFIED BLOCK SIZE. A FATAL SYNTAX ERROR WILL PRINT IF THIS USE PROCEDURE IS SPECIFIED FOR A FILE WITH VARIABLE LENGTH RECORDS OR BLOCKS OR FOR A FILE ASSIGNED TO A DEVICE OTHER THAN TAPE. DOCUMENTATION IS ALSO ADDED TO THE COMPILER.

DATACOM

DO171 MCP-DATACM - DCSYSTEMTABLES INTRINSIC - 02-19-73

THE DCSYSTEMTABLES INTRINSIC ENABLES AN ALGOL OR DCALGOL PROGRAM TO OBTAIN INFORMATION ABOUT THE CURRENT DATACOM ENVIRONMENT. IT IS PROVIDED FOR USE BY THE SYSTEM/DCSTATUS PROGRAM, AND IS THEREFORE SUBJECT TO FUTURE MODIFICATION.

DCSYSTEMTABLES IS A REAL-VALUED INTRINSIC. THE CALLING SYNTAX IS:

```
DCSYSTEMTABLES(<OPTION>,<ROW>)

<OPTION> ::=<ARITHMETIC EXPRESSION>
<ROW> ::=<ARRAY ROW>
```

EXAMPLE:

```
T := DCSYSTEMTABLES (3, ALL, *));
```

SEMANTICS:

1. THE INFORMATION SELECTED BY <OPTION> IS COPIED INTO <ROW>.
2. <ROW> IS ALWAYS RESIZED TO CONTAIN THE REQUESTED INFORMATION, AND THE PREVIOUS CONTENTS OF <ROW> IS LOST.
3. <OPTION> SELECTS THE INFORMATION AS FOLLOWS:

   0 = DCC TABLES
   1 = DCP TABLES
   2 = DCC STATION TABLE
   3 = GENERAL INFORMATION:

   A. <ROW>[0] = BIT MAS OF INITIALIZED DCP-S (I.E., IF <ROW>[0].[3:1] = 1 THEN DCP #3 HAS BEEN INITIALIZED).

   B. <ROW>[1] TO <ROW>[3] = DC FILE PREFIX.

TERMINATED BY A PERIOD.

C. <ROW>[4] = BIT MASK OF DCP-S ON-LINE.

D. <ROW>[5] = BIT MASK OF DCP-S CONFIGURED IN NDL.

4 = NIF STATION RECORD.

4. WITH THE EXCEPTION OF <OPTION> = 3, DATACOM MUST BE RUNNING WHEN THIS INTRINSIC IS INVOKED (I.E., AT LEAST ONE DCP INITIALIZED).

5. IF THE INTRINSIC IS PROPERLY CALLED AND THE REQUESTED INFORMATION IS AVAILABLE, THE VALUE RETURNED BY THE INTRINSIC (T) CONTAINS THE FOLLOWING:

A. T.[39:20] = SIZE (WORDS) OF INFORMATION = NEW SIZE OF <ROW>.

B. T.[19:20] = MEMORY ADDRESS OF THE REQUESTED MCP TABLE, IF APPLICABLE, OTHERWISE ZERO.

6. IF THE INFORMATION WAS NOT SUCCESSFULLY OBTAINED, THE VALUE RETURNED BY THE INTRINSIC INDICATES THE REASON AS FOLLOWS:

-1 = DATACOM NOT RUNNING AND
<OPTION> NEQ 3.
-2 = <OPTION> LSS 0 OR <OPTION> GTR 4.
-3 = <INVALID LSN IN <ROW> [0] (<OPTION> = 4).
-4 = INVALID NAME IN <ROW>
<OPTION> = 4).
-5 = UNKNOWN STATION (<OPTION> = 4).

7. THE OPTION $INSTALLATION MUST BE SET WHEN COMPILING THE USER PROGRAM.

8. WHEN CALLED WITH <OPTION> = 4, THE STATION DESIRED MUST BE SPECIFIED EITHER BY PLACING THE STATION NAME IN EBCDIC STARTING IN <ROW>[0], OR BY SPECIFYING THE STATION LSN IN <ROW>[0].

D0176 MCP-DATACM - SUBTRACT STATION ERROR - 10-30-72

FOR MOVE STATION DCWRITE, AN ATTEMPT TO SUBTRACT A STATION WHICH IS NOT ATTACHED WILL NOW RETURN DCWRITE ERROR #121. ON PAGE 41 OF THE DCALGOL LANGUAGE MANUAL, ERROR #121 SHOULD BE ADDED AS FOLLOWS:

121 AN ATTEMPT TO PERFORM MOVE STATION WAS MADE WHERE THE SOURCE STATION IS NOT ATTACHED TO THE REQUESTING MCS.

D0199 MCP-DATACM - UPDATE LINE DCWRITE FUNCTION - 05-30-72

THE UPDATE LINE ATTRIBUTES DCWRITE (TYPE=131) ALLOWS AN MCS TO MODIFY THE ADAPTOR TYPE AND/OR MODEM FOR A PARTICULAR LINE.

REQUIRED:

1. MESSAGE PARAMETER (MINIMUM SIZE = 9 WORDS)
2. MSG[0].[47:8] = 131.
3. MSG[7].[23:24] = DL => LINE ADDRESS
4. MSG[8].[22:7] = 0 => LEAVE ADAPTER CLASS UNCHANGED
   ≠ 0 => NEW ADAPTER CLASS
5. MSG[8].[23:1] = 0 => NEW ADAPTER REQUIRES MODEM
   = 1 => NEW ADAPTOR DIRECT CONNECT
6. MSG[8].[7:8] = 0 => LEAVE MODEM UNCHANGED
   = MODINX => INDEX INTO MESSAGE FOR NEW MODEM NAME.

SEMANTICS:

THE LINE UPDATE IS PERFORMED FOR THE SPECIFIED LINE, AND IF ANY STATIONS EXIST ON THE LINE, THE LINE IS LEFT READY. IF MSG[8].[23:1] = 1, NO MODEM NAME MAY BE SPECIFIED. IF MODINX ≠ 0, THE NEW MODEM NAME MUST BE IN EBCDIC AND BE TERMINATED BY A PERIOD. ONLY VALID, NON-DEFAULT MODEM NAMES AS SPECIFIED IN NDL ARE ALLOWED.

D0199 MCP-DATACM - UPDATE LINE DCWRITE FUNCTION - 05-30-72

EXAMPLE:

```
ALLOCATE(MSG,11);
MSG[0] := 0 & 131[47:8];
MSG[7] := 0 & 1[23:1] & DL [22:15];
MSG[8] := 0 & 2[23:8] & 9[7:8];
REPLACE POINTER(MSG[9],8) BY "NEWMODEM.";
RESULT := DCWRITE (MSG);
```

D0200 MCP-DATACM - MOVE STATION-DCWRITE FUNCTION - 05-30-72

THE MOVE/ADD/SUBTRACT STATION DCWRITE(TYPE = 130) HAS BEEN EXTENDED TO ALLOW NEW LINE ATTRIBUTES TO BE SPECIFIED IN THE MOVE AND ADD FUNCTIONS. THE NEW FORMAT OF THE MESSAGE PARAMETER IS AS FOLLOWS:

REQUIRED:

1. MESSAGE PARAMETER (MINIMUM LENGTH = 8 WORDS)
2. MSG[0].[47:8] = 130
3. MSG[0].[39:16] = VARIANT FIELD:
   .[24:1] = 0 => LEAVE STATION NOT READY AFTER MOVE
   1 => LEAVE STATION READY AFTER MOVE
   .[25:1] = 0 => DO NOT UPDATE STATION ATTRIBUTES
   1 => UPDATE STATION ATTRIBUTES AS SPECIFIED.
4. MSG[0].[23:24] = CURRENT LSN OR DLS
5. MSG[7].[23:24] = 0 => SUBTRACT STATION
   DL => DESTINATION LINE ADDRESS

OPTIONAL INFORMATION (REQUIRED IF MSG[0].[25:1] = 1):

6. MSG[8] = NEW LINE ATTRIBUTES:
   [23:8] = NEW ADAPTER TYPE NUMBER
   [15:8] = TERMINX = INDEX INTO MSG FOR NEW TERMINAL NAME
   [7:8] = MODEMINX = INDEX INTO MSG FOR NEW MODEM NAME

7. MSG[TERMINX] = NEW TERMINAL NAME FOR STATION
8. MSG[MODEMINX] = NEW MODEM NAME FOR STATION

THE NEW SEMANTICS FOR UPDATING LINE ATTRIBUTES IS AS FOLLOWS:

IF MSG[0].[25:1] = 1 AND MSG[7].[23:24] ≠ 0 THEN THE LINE ATTRIBUTES SPECIFIED IN MSG[8] WILL BE APPLIED TO THE NEW LINE. IF A PARTICULAR ATTRIBUTE FIELD IN MSG[8] IS ZERO, THE CURRENT LINE ATTRIBUTE WILL NOT BE MODIFIED. NAME ATTRIBUTES MUST BE SPECIFIED IN EBCDIC AND BE TERMINATED BY A PERIOD. THEY MUST CORRESPOND TO VALID NON-DEFAULT NAMES IN NDL.

EXAMPLE:

```
ALLOCATE (MSG,11);
MSG[0] := LSN & 130 [47:8] & 3[39:16];
MSG[7] := 0 & 1 [23:1] & DL[22:15];
MSG[8] := 0 & 89[23:8] % NEW ADAPTER TYPE
            & 9[15:8]; % NEW TERMINAL NAME INDEX
REPLACE POINTER (MSG[9],8) BY "TERMINAL29.";
RESULT := DCWRITE(MSG);
```

D0233 MCP-DATACM - DCP FAULT RESULT - 02-19-73

WHEN A DCP FAULT OCCURS, ALL CURRENTLY RUNNING MCS S WHICH HAVE INITIALIZED THEIR PRIMARY QUEUE WILL RECEIVE A DCP FAULT RESULT MESSAGE (TYPE = 20) IN THEIR PRIMARY QUEUE. THIS FAULT RESULT IS ALSO PLACED IN THE SYSTEM/SUMLOG. A DCP FAULT ALWAYS RESULTS IN TERMINATION OF THE CORRESPONDING DCC STACK. THIS NOTIFICATION CAN ONLY OCCUR IF THE DCP "STOP-ON-FAULT" SWITCH IS IN THE NEUTRAL (RUN) POSITION. THE FORMAT OF THE DCP FAULT RESULT IS AS FOLLOWS:

1. MSG[0].[47:8] = 20 (MSG CLASS = DCP FAULT)
   .[23:1] = 1
   .[22:7] = DCP NUMBER OF FAILING DCP
   .[15:16] = 0
2. MSG[2].[39:16] = 72 (SIZE OF TEXT PORTION OF MESSAGE)
3. MSG[47:24] = TIME OF MESSAGE (1/60 OF A SECOND)

D0233 MCP-DATACM - DCP FAULT RESULT - 02-19-73

4. MSG[6] = TAGS OF WORDS IN MSG[8]=>MSG[17]
   .[47:4] = TAG OF MSG[8]
   .
   .
   .
   .[11:4] = TAG OF MSG[17]
5. MSG[8].[47:24] = AA,AC,AI REGISTERS
   .[23:24] = D,Y,X REGISTERS
6. MSG[9] = SM WORD 7
7. MSG[10] = SM WORD 6
8. MSG[11] = SM WORD 5
9. MSG[12] = SM WORD 4
10. MSG[13] = SM WORD 3
11. MSG[14] = SM WORD 2
12. MSG[15] = SM WORD 1
    .[39:16] = INSTRUCTION ADDRESS REGISGER
    .[47:8] = FAULT INDEX
13. MSG[16] = SM WORD 0
    .[23:24] = MA REGISTER CONTENTS
14. MSG[17] = WORD REGISTER CONTENTS

THE VALUES OF THE FAULT INDEX CODE AND THEIR RESPECTIVE MEANINGS ARE:

0 = INITIALIZE FROM SCAN-OUT
1 = LM DATA INVALID ADDRESS
2 = LM DATA PARITY ERROR
3 = ADAPTER CLUSTER ERROR
4 = MM PROTECTED WRITE DENIED
5 = MM DATA INVALID ADDRESS
6 = MM DATA PARITY ERROR
7 = MM DATA ACCESS ERROR
8 = LM INSTRUCTION TAG ERROR
9 = LM INSTRUCTION INVALID ADDRESS
10 = LM INSTRUCTION PARITY ERROR
11 = UNEXPECTED TIMEOUT
12 = MM INSTRUCTION TAG ERROR

D0233 MCP-DATACM - DCP FAULT RESULT - 02-19-73

13 = MM INSTRUCTION INVALID ADDRESS
14 = MM INSTRUCTION PARITY ERROR
15 = MM INSTRUCTION FETCH ERROR
16 = RECURSIVE FAULT

D0234 MCP-DATACM - DATACOM ERROR LOGGING - 02-19-72

DATACOM ERROR LOGGING HAS BEEN PROVIDED AS A DCALGOL FUNCTION CALLED "DCERRORLOGGER". BY SELECTIVELY INVOKING THIS INTRINSIC, AN MCS CAN HAVE INFORMATION ABOUT HIS DATACOM ENVIRONMENT ENTERED INTO THE SYSTEM LOG.

THE DATACOM SUBSYSTEM (DCC) ALSO UTILIZES THIS INTRINSIC TO UNCONDITIONALLY RECORD MCS INITIALIZATIONS AND DCP FAULTS.

THE SYNTAX FOR CALLING DCERRORLOGGER IS:

DCERRORLOGGER(<MSG>,SIZE>)
<MSG> ::= AN ARRAY, ARRAY ROW OR MESSAGE.
<SIZE> ::= REAL EXPRESSION.

THE <MSG> IS ENTERED INTO THE SYSTEM LOG FOR THE NUMBER OF WORDS SIZE>) SPECIFIED, PREFIXED BY A 6-WORD HEADER AS DESCRIBED IN THE WORK FLOW MANAGEMENT DOCUMENT. THE MAJOR TYPE ASSIGNED TO THIS ENTRY IS FIVE, AND THE MINOR TYPE IS SET TO FOUR.

THE FORMAT OF THESE RECORDS IS ALSO DEFINED IN THE WORK FLOW MANAGEMENT DOCUMENT.

IF THE GIVEN LOG RECORD COULD NOT BE ENTERED IN THE SYSTEM LOG, DCERRORLOGGER WILL RETURN A NEGATIVE RESULT INDICATING THE SPECIFIC REASON THE REQUEST WAS DENIED. THE VALUES RETURNED BY DCERRORLOGGER AND THEIR RESPECTIVE MEANINGS ARE:

0 LOG ENTRY MODE SUCCESSFULLY
-1 <SIZE> PARAMETER WAS GREATER THAN THE TRUE SIZE OF <MSG>
-2 CALLER IS NOT A VALID MCS OR HAS NOT INITIALIZED HIS PRIMARY QUEUE.

D0234 MCP-DATACM - DATACOM ERROR LOGGING - 02-19-72

-3 A DISK ERROR OCCURRED ATTEMPTING TO LOG THE ENTRY.

EXAMPLE:

R: = DCERRORLOGGER (A[*], 6);

D0279 DATACOM - DCALGOL QUEUE TANKING - 03-23-73

WHEN INSERTING A MESSAGE TO THE HEAD OF A QUEUE, THE FOLLOWING PROBLEM CAN OCCUR: IF THE MESSAGE TO BE INSERTED IS LARGE ENOUGH TO INVOKE DISK TANKING, THE LAST MESSAGE CURRENTLY IN THE MEMORY-RESIDENT PORTION OF THE QUEUE WILL BE TANKED TO DISK BEHIND THE LAST MESSAGE IN THE QUEUE. IF THE MEMORY-RESIDENT PORTION OF THE QUEUE IS EMPTY, THEN THE MESSAGE TO BE INSERTED WILL BE TANKED BEHIND THE LAST MESSAGE IN THE QUEUE, REGARDLESS OF WHETHER IT SHOULD GO TO THE HEAD OF THE QUEUE OR NOT.

D0285 MCP-DATACM - UPDATE LINE ATTRIBUTES RESULT - 04-11-73

MESSAGE FORMAT:

MSG[0].[47:8] = 19
MSG[0].[39:40] = AS ORIGINALLY PRESENTED TO DCWRITE
MSG[1].[47:8] = 0=> UPDATE LINE SUCCESSFULLY COMPLETED
≠ 0=> INTERPRETED AS THE DCWRITE ERROR VALUES

PRAGMATICS:

AN MCS PERFORMING AN UPDATE LINE ATTRIBUTES DCWRITE (TYPE=131) WILL RECEIVE A TYPE 19 RESULT MESSAGE AT THE COMPLETION OF THE REQUEST.

D0286 MCP-DATACM - INITIALIZE PRIMARY QUEUE - 04-11-73

WHEN AN MCS PERFORMS AN INITIALIZE PRIMARY QUEUE DCWRITE (TYPE=0), INFORMATION IS RETURNED TO THE MCS VIA THE MESSAGE PARAMETER. IF

THE PASSED MESSAGE IS TOO SMALL TO CONTAIN THE INFORMATION. A NEW MESSAGE AREA WILL BE ALLOCATED IN PLACE OF THE ORIGINAL AREA. THE FORMAT OF THE MESSAGE IS AS FOLLOWS:

MSG[0] = UNCHANGED
MSG[1] = NUMBER OF THIS MCS
MSG[2] = MAXIMUM VALID LSN
MSG[3] THRU MSG[5] = UNCHANGED
MSG[6] = PREFIX OF THE CURRENT DATACOM FILE
IN EBCDIC, TERMINATED BY A PERIOD.

DATA MANAGEMENT

00254 DM6700 - DMUPDATE - 02-19-73

PURPOSE: DMUPDATE IS A ONE-TIME ONLY PROGRAM THAT CONVERTS EXISTING II.3 DATABASES TO II.4 DATABASES. IT WAS WRITTEN SO THAT THE USER WOULD NOT HAVE TO REGENERATE HIS DATA-BASES OR RECOMPILE HIS PROGRAMS THAT USE THOSE DATA-BASES.

COMPILE INFORMATION: THE FILE TAPE SHOULD BE LABEL EQUATED TO "SYMBOL/DMUPDATE" AND A $ MERGE CARD INCLUDED IN THE CARD DECK.

RUN INFORMATION: THE OBJECT JOB WILL LOOK FOR A CARD FILE WHOSE NAME IS CARD. THE CARD FORMAT IS FREEFORM AS FOLLOWS:

DATA-BASE = <DATA-BASE NAME>;

THE STEPS FOR RUNNING ARE:

1. LOAD THE DATA-BASE TO BE UPDATED.
2. RUN DMUPDATE.
3. IF NO ERRORS ARE NOTED, THE USER MAY THEN COMMENCE TO USE THE UPDATED DATA-BASE OR HE MAY DUMP IT TO A TAPE FOR LATER USE. THE USER SHOULD BE CAREFUL AND NOT FORGET TO DUMP THE ERRORS FILE AND THE ALTERED "OVERFLOW" FILE.

ACTIONS: DUMPDATE GENERATES A FILE CALLED:

DM/<DATA-BASE NAME>/ERRORS

AND ALSO GENERATES AN SDL DICTIONARY RECORD FOR THE DM/<DATA-BASE NAME>/OVERFLOW THEN CHANGES THE NAME OF THE

00254 DM6700 - DMUPDATE - 02-19-73

OVERFLOW FILE TO AN AVAILABLE STRUCTURE NUMBER. THE NEW STRUCTURE NUMBER OF OVERFLOW FILE IS THEN INSERTED INTO RECORD ZERO OF THE SDL DIRECTORY.

NOTE: NOT ALL DATA-BASES HAVE A DM/<DATA-BASE NAME>/OVERFLOW FILE. DMUPDATE MUST STILL BE RUN TO GENERATE THE DM/<DATA-BASE NAME>/ERRORS FILE. A MESSAGE WILL BE INSERTED INTO THE LOG THAT THE CHANGE DM/<DATA-BASE NAME>/OVERFLOW TO DM/<DATA-BASE NAME>/<NUMBER> DID NOT OCCUR. THIS SHOULD BE IGNORED.

00255 DM6700 - DM - REQUEST HANDLER EXECUTION - 01-19-73

REQUEST HANDLERS FOR A GIVEN DATABASE WERE PREVIOUSLY PROCESSED AS SYSTEM/DM6700. THEY ARE NOW PROCESSED AS DM <DATABASE-NAME>.

00256 DM6700 - DM - SDL IMPROVEMENTS - 02-19-73

PURPOSE:

SYMBOL/SDLS REPLACES SYMBOL/SDL/STRUCTURE AND SYMBOL/SDL/INITIALIZE. THIS WAS DONE TO IMPROVE STABILITY, DOCUMENTATION AND MAINTAINABILITY.

COMPILE INFORMATION:

SYSTEM/SDL/STRUCTURE IS COMPILED BY LABEL EQUATING THE TAPE FILE FOR DCALGOL TO "SYMBOL/SDLS" AND INCLUDING A $ SET STRUCTURE AFTER THE $ MERGE CARD IN THE CARD DECK. SYSTEM/SDL/INITIALIZE IS COMPILED IN A SIMILAR MANNER AS ABOVE WITH THE EXCEPTION OF THE $ SET STRUCTURE CARD WHICH SHOULD BE REPLACED BY A $ SET INITIALIZE.

RUN INFORMATION:

SAME AS PREVIOUS SDLS EXCEPT THAT BOTH THE SDLS NOW USE A STRICT INTERPRETATION OF SYNTAX AND SPELLING FROM THE DOCUMENTATION.

PIT FALLS:

THE PROBLEMS THAT WERE NOTED BY PEOPLE CONVERTING FROM "OLD" SDL TO "NEW" SDL WERE:

1. THE STRUCTURES FOR SYSTEM/SDL/INITIALIZE WERE UP BY ONE BECAUSE OF STRUCTURE ONE NOW BEING RESERVED FOR THE OVERFLOW FILE. SYSTEM/SDL/INITIALIZE DECKS WHICH REFERENCE STRUCTURE NUMBERS MAY BE WRONG. SINCE THE OVERFLOW FILE WILL BECOME STRUCTURE #1. IF REGENERATING UNDER 2.4, ALL STRUCTURE NUMBERS USED IN THE 2.3 SDL/INITIALIZE DECK (I.E., SET 12 BLOCKFACTOR = 10;) WILL BE OFF BY ONE SINCE STRUCTURE #1 IS NOW USED FOR ONE OVERFLOW FILE.

2. THE SPELLING AND SIZE OF KEY WORDS WAS NOT ENFORCED IN THE "OLD" SDL. HOWEVER, "NEW" SDL DOES MATCH THE DOCUMENTATION MORE CLOSELY (E.G., PREVIOUSLY SETSONG COULD HAVE BEEN SUBSTITUTED FOR SET).

NEW FEATURES:

THERE HAVE BEEN TWO NEW FEATURES ADDED TO THE SDL S. THEY ARE DEFAULT AND $ OPTIONS. DEFAULT IS HANDLED WITH THE FOLLOWING SYNTAX:

```
DEFAULT <DEFAULT OPTION>;

<DEFAULT OPTION>::= POPULATION =<NUMBER><DEFAULT OPTION>
                    /DEVICE=<DISK><DEFAULT OPTION>
                    /DUPLICATED = <STATE> <DEFAULT OPTION>
                    /<EMPTY>

<DISK>::= DISK
          /DISKPACK
<STATE>::= ON/OFF
```

<NUMBER>::= [POSITIVE INTEGER]

DEFAULTS ALLOW THE USER TO CHANGE THE NORMAL DEFAULTS TO SOMETHING MORE AGREEABLE TO HIM. THE POPULATION DEFAULT ALLOWS THE USER TO CHANGE THE VALUE OF DDL POPULATION = * FROM 10000 TO SOMETHING MORE IN LINE WITH THE SIZE OF HIS DATA-BASE. THIS POPULATION IS ONLY USED TO INITIALLY ALLOCATE SPACE AND DOES NOT IMPOSE A LIMIT TO POPULATION.

THE DEVICE DEFAULT WILL ALLOW THE USER TO SPECIFY WHERE ALL HIS DATA-BASE FILES WILL RESIDE. ANY DEVICE SPECIFIED ON INDEX AND FILE STATEMENTS IN STRUCTURE AS WELL AS ANY SET STATEMENTS FOR INITIALIZE WILL OVERRIDE THE DEFAULT CONDITION. THE VALUE OF DEFAULT DEVICE IS USED BY SYSTEM/SDL/STRUCTURE TO DETERMINE WHERE TO PLACE THE DM/<DATA-BASE NAME>/SDL FILE. IT WILL RESIDE ON THE DEVICE SPECIFIED IN THE DEVICE DEFAULT SYNTAX. DM/<DS-NAME>/ERRORS WILL ALSO BE PLACED WHEREVER THE DEFAULT DEVICE SPECIFIES.

THE ONLY FILE NOT AFFECTED BY THIS STATEMENT IS DM/<DB-NAME>/DDL. IT MUST ALWAYS RESIDE ON HEAD-PER-TRACK.

THE DUPLICATED DEFAULT WILL ALLOW THE USER TO SPECIFY THAT ALL DATA MANAGEMENT FILES (EXCEPT DM/<DATA-BASE-NAME>/DDL) ARE DUPLICATED. AGAIN THIS MAY BE OVERRIDDEN FOR SPECIFIC FILES IN THE SET STATEMENT FOR SYSTEM/SDL/INITIALIZE. THE ERROR FILE WILL BE DUPLICATED IF DUPLICATED = ON AND WILL BE A SYSTEM DUPLICATED FILE. ALL OTHER DUPLICATED FILES ARE DONE THROUGH DM6700. THE INITIAL VALUES OF THE DEFAULT ARE:

```
POPULATION = 10000
DEVICE = DISK
DUPLICATED = OFF
```

EXAMPLE:

```
?RUN SYSTEM/SDL/STRUCTURE
?DATA CARD
DEFAULT
   POPULATION = 65000
   DEVICE = DISKPACK
```

```
   DUPLICATED = ON;
(
( OTHER STRUCTURE CARDS
(
?END
```

THE EFFECT OF THIS WOULD BE:

FOR ALL SETS WITH POPULATION =*, 65000 WOULD BE USED TO CALCULATE SUCH THINGS AS AREASIZE, BLOCKING, ETC.

ALL FILES, DM/DB001/= WOULD RESIDE ON DISKPACK EXCEPT DM/DB001/DDL.

ALL FILES, DM/DB001/= WOULD BE DUPLICATED EXCEPT DM/DB001/DDL.

THE DOLLAR OPTIONS ARE SIMILAR TO THOSE OF THE B6700 COMPILERS; THEY ARE:

DEBUG	THIS SETS THE TRACE BOOLEAN WHICH WILL EVOKE THE DEBUGGING ROUTINE OF SDLS.
LIST	SETS THE LISTB BOOLEAN. IF LISTB IS FALSE, NO LISTING WILL BE GENERATED (EXCEPT FOR ERRORS). IF TRUE, THEN NORMAL LISTINGS WILL BE GENERATED.
POP	POP POPS A 48-BIT STACK WITH A FALSE.
RESET	RESET PUSHES A 48-BIT STACK WITH A FALSE.
SET	SET PUSHES A 48-BIT STACK WITH A TRUE.
SINGLE	IF THE LIST OPTION IS SET, THEN THE GENERATED LISTING WILL BE SINGLE SPACED.

(NOTE: LIST IS THE ONLY $ OPTION WITH AN INITIAL VALUE OF TRUE.)

CHANGES TO SYSTEM/SDL/STRUCTURE:

STRUCTURE NUMBER ONE IS NOW RESERVED FOR THE "OVERFLOW" FILE; THEREFORE, USER STRUCTURES NOW START AT TWO. THIS COULD CAUSE A PROBLEM IN SYSTEM/SDL/INITIALIZE IF THE SET STATEMENT IS USED; I.E., OLD CARD DECKS WITH THE SET STATEMENT IN THEM MUST BE REMADE.

FILES AND OVERFLOW FILES FOR LISTS ARE NOW BLOCKED TO A SEGMENT BOUNDARY UP TO A MAXIMUM OF 1000 WORDS. THIS WAS DONE IN AN ATTEMPT TO GET MORE EFFICIENT USE OF DISK WHEN THE DATA-BASE FILES ARE STRUCTURED. THIS MAY BE OVERRIDDEN IN SYSTEM/SDL/INITIALIZE.

CHANGES TO INITIALIZE:

THE FOLLOWING SYNTAX HAS BEEN ADDED TO THE SYNTAX FOR SET:

```
DUPLICATED = ON/OFF
DEVICE = <DISK>
<DISK>::= DISK/DISKPACK
```

DUPLICATED ALLOWS THE USER TO OVERRIDE THE DEFAULT VALUE OF DUPLICATED (SEE DEFAULT IN NEW FEATURES).

AUDIT SYNTAX SHOULD BE ALTERED AS FOLLOWS:

```
<AUDIT MEDIUM>::= DEVICE = <KIND>/<EMPTY>

<KIND>::= DISK/DISKPACK/TAPE7/TAPE9/PETAPE
```

STRUCTURE NUMBER ONE IS NOW RESERVED FOR THE OVERFLOW FILE. THEREFORE, OLD SYSTEM/SDL/INITIALIZE DECKS WILL HAVE TO BE CHANGED TO REFLECT THIS MODIFICATION.

CORRECTIONS:

THE FOLLOWING RECORDED ERRORS HAVE BEEN CORRECTED:

1. RERUNNING SYSTEM/SDL/INITIALIZE WILL NO LONGER INCREASE THE INDEX-SEQUENTIAL TABLES WITHOUT BOUND.

2. CHANGES IN BLOCK FACTOR FOR RANDOM INDEX WILL NOW BE REFLECTED IN THE FILE FOR THAT INDEX.

3. INDEXSIZE PARAMETERS OF THE INDEX STATEMENT FROM SYSTEM/SDL/STRUCTURE WOULD PREVIOUSL ONLY BE USED IF ALL WERE NON-ZERO. UNDER THE "NEW" SDL THOSE WHOSE VALUE IS NON-ZERO WILL BE USED; AND IF THE PARAMETER IS ZERO, THEN SYSTEM DEFAULT ACTION WILL OCCUR FOR THAT PARAMETER.

D0256 DM6700 - DM - SDL IMPROVEMENTS - 02-19-73

4. PREVIOUSLY IN SYSTEM/SDL/INITIALIZE A USER COULD SPECIFY AN ONLY OPTION (I.E., LIMITSONLY OR AUDITONLY, ETC.) ALONG WITH A SET STATEMENT AND THE DATA-BASE WOULD NOT BE INITIALIZED OR REINITIALIZED. UNDER THE "NEW" SDL A WARNING MESSAGE WILL BE ISSUED AND THE DATE-BASE WILL BE INITIALIZED OR REINITIALIZED.

5. IF DISJOINT ASSOCIATED SETS WITH ORDERINGS HAD CONTAINED A SET WITH MORE THAN 45 ITEMS, THE OFFSET TO THE ORDERING ITEM WOULD BE WRONG. THIS HAS BEEN CORRECTED.

D0257 DM6700 - DM - DDL EXECUTION - 02-19-73

WHILE RUNNING SYSTEM/DDL, "DDL <DATABASE-NAME>" WILL BE DISPLAYED.

D0258 DM6700 - DM - SDL EXECUTION - 02-19-73

WHILE SYSTEM/SDL/STRUCTURE IS RUNNING, A DISPLAY OF THE FORM "STR <DATABASE-NAME>" WILL APPEAR ON THE CONSOLE. THIS IS REPLENISHED AS EACH STRUCTURE IS COMPILED.

SYSTEM/SDL/INITIALIZE WILL DISPLAY A MESSAGE "INIT<DATABASE-NAME>".

D0259 DM6700 - DM - DATA COMPACTION - 02-19-73

THE DATA COMPACTION FACILITY OF DM6700 HAS BEEN IMPROVED AND THOROUGHLY TESTED. IN APPROPRIATE SITUATIONS, IT WILL YIELD A SUBSTANTIAL REDUCTION IN DISK SPACE FOR A RELATIVELY SMALL OVERHEAD.

D0260 DM6700 - GLOBAL FOR EMBEDDED SETS - 02-19-73

IF GLOBAL IS SPECIFIED FOR AN EMBEDDED SET AND IS NOT RESIDENT, THEN THE FILE IN WHICH IT RESIDES (LIST OVERFLOW FILE) WILL NOT AND

D0260 DM6700 - GLOBAL FOR EMBEDDED SETS - 02-19-73

CANNOT BE BLOCKED.

D0261 DM6700 - RANDOM STRUCTURE - 02-19-73

IF A KEY OF RETRIEVAL IS DECLARED TO BE RANDOM AT SDL TIME, IT MUST HAVE BEEN REQUIRED IN DDL AND MUST NOT HAVE BEEN IN A SET WITH POPULATION=*.

D0262 DM6700 - REQUIRED ITEMS - 02-19-73

A PREVIOUS LIMITATION WHEREBY ONLY 47 ITEMS COULD EXPLICITLY BE REQUIRED HAS BEEN REMOVED.

D0263 DM6700 - MODIFY ORDER OF DJ SET WITH IA - 02-19-73

MODIFY ORDERING OF A DISJOINT SET WITH AN EMBEDDED ASSOCIATED SET.

EXAMPLE

```
S1 SET                    S2 SET
   POP                       POP=100
   ORDERED ON A              (X ALPHA(6));
   (A ALPHA(2);              S3 SET
    B ON(6);                     POP=4
   );                           ORDERED ON A
                                (CONTAINS NUMBERS OF S1)
```

PREVIOUSLY ANY ATTEMPT TO MODIFY THE KEY OF ORDERING FOR ANY MEMBER OF A DISJOINT SET, S1, THAT HAD AN EMBEDDED SET, S3, CONTAINING S1 AND ORDERED ON THE SAME KEY AS S1 WOULD RESULT IN A STATUS OF 101. THIS RESTRICTION HAS BEEN EASED SOMEWHAT. THE KEY OF ORDERING FOR A MEMBER OF A DISJOINT SET, S1, MAY BE MODIFIED PROVIDED THAT THIS MEMBER IS NOT ALSO A MEMBER OF ANY EMBEDDED ASSOCIATED SET (I.E. S3), NOR POINTED AT VIA LINK IN ANY SET.

00264 DM6700 - CURRENT AFTER MODIFY - STORE - 02-19-73

THE STORE AFTER A MODIFY WILL NOT CHANGE THE CURRENT (I.E. CURRENT RECORD NUMBER). THIS IS A CHANGE FROM II.3.

THERE ARE TWO INSTANCES IN WHICH THIS CHANGE WILL PROVIDE DIFFERENT RESULTS FROM THE PREVIOUS SCHEME OF CHANGING THE CURRENT AFTER THE STORE AFTER A MODIFY.

1. MODIFYING THE KEY OF ORDERING

   THIS CHANGES THE LOGICAL POSITION OF A MEMBER IN A SET BUT THE CURRENT POINTER REMAINS WHERE IT WAS AFTER THE MODIFY.

EXAMPLE

```
S1 SET
    POP=*
    ORDERED ON A
    (A
     B
     .
     .
     .
    );

    ASSUME A = 5
    MODIFY NEXT OF S1.      (ASSUME CURRENT IS MEMBER #21
    MOVE 10 TO A.            AND A=5)
    STORE S1.
                            (CURRENT REMAINS #21 EVEN
                            THOUGH THE MEMBER WITH A=10 MAY
                            BE MEMBER #101)
```

2. MODIFYING A RANDOM KEY OF RETRIEVAL

   MODIFYING A RANDOM SET OF RETRIEVAL ALSO CHANGES THE LOGICAL POSITION OF MEMBER IN A SET BUT NOW THE CURRENT POINTER WILL NOT BE CHANGED AFTER THE STORE.

D0268 DM6700 - SDL - INDEX PARAMETERS - 03-07-73

THE FOLLOWING SYNTAX IS ADDED TO THE SYNTAX FOR INDEX IN SYSTEM/SDL/STRUCTURE TO BE USED AS AN ALTERNATIVE SYNTAX FOR INDEXSIZE. THEY ARE AS FOLLOWS:

FOR INDEX=SEQUENTIAL:

COARSE = <NUMBER>

THIS IS USED BY THE USER TO SET THE DESIRED COARSE TABLE SIZE (NUMBER OF ENTRIES PER COARSE TABLE)

FINE = <NUMBER>

THIS IS USED BY THE USER TO SET THE DESIRED FINE TABLE SIZE (NUMBER OF ENTRIES PER FINE TABLE).

INITIAL = <NUMBER>

THIS IS USED BY THE USER TO SET THE DESIRED INITIAL LOAD FACTOR. IT MUST BE LESS THAN OR EQUAL TO FINE. SO THIS MUST BE SET AFTER FINE IS SET.

FOR INDEX=RANDOM

NUMBER
[OR] NUMOFTABLES
[OR] NUMBEROFTABLES = <NUMBER>

THIS IS USED BY THE USER TO SET UP THE DESIRED NUMBER OF BASIC TABLES.

TABLESIZE = <NUMBER>

THIS IS USED TO SET UP THE DESIRED NUMBER OF ENTRIES IN THE BASIC TABLE.

FOR RANDOM:

BLOCK = <NUMBER>

D0268 DM6700 - SDL - INDEX PARAMETERS - 03-07-73

THIS IS USED BY THE USER TO CHANGE THE NUMBER OF BLOCKS IN THE BASIC AREA.

FOR TAG:

TAGS
[OR] BLOCK = <NUMBER>

THIS IS USED BY THE USER TO CHANGE THE NUMBER OF TAGS PER RECORD.

FOR BIT:

BLOCK = <NUMBER>

THIS IS USED BY THE USER TO CHANGE THE NUMBER OF BIT VECTORS PER RECORD.

ALL THE ABOVE ARE TYPE DEPENDENT. ERGO, TYPE MUST BE SET BEFORE ANY OF THE ABOVE ARE USED.

D0269 DM6700 - DM - DDL WARNING - 02-19-73

THE FOLLOWING DDL IS ILLEGAL BUT NOT CURRENTLY SYNTAXED AS SUCH.

```
S1 SET                    S2 SET
    POP = 100000              POP = 4000
    ORDERED ON A              ORDERED ON A DUPLICATES ALLOWED
    (                         (CONTAINS MEMBERS OF S1);
    -
    -
    -
    -
    -
    );
```

IF A DA OR IA SET IS ORDERED ON A KEY THAT IS ALSO A RETRIEVAL OR ORDERING KEY OF A DJ SET AND THE KEY DOES NOT ALLOW DUPLICATES THEN THE DA OR IA KEY MAY NOT ALLOW DUPLICATES.

D0290 DM6700 - DM - NEW STATUS - 02-16-73

A NEW STATUS HAS BEEN ADDED. STATUS = 8 INDICATES AN ATTEMPT TO PERFORM AN OPERATION ON A SUBSUMBED MEMBER THAT HAS NO CURRENT MASTER.

D0284 DM6700 - DESIGN OF RECOVERY FOR DM6700 - 04-03-73

TABLE OF CONTENTS

1. INTRODUCTION

THE WORD "RECOVERY" IN THE TITLE IS USED TO ENCOMPASS THE ENTIRE SET OF PROCEDURES FOR DATA BASE RECOVERY. IT IS ALSO USED IN THIS DOCUMENT WITH A VERY SPECIFIC MEANING INTENDED. SEVERAL OTHER WORDS ARE ALSO USED WITH SPECIFIC MEANINGS IN MIND. IN AN ATTEMPT TO AVOID SEMANTIC CONFUSION, THESE WORDS ARE HEREBY DEFINED:

AUDIT	WRITING A TRAIL OF CHANGES TO THE DATA BASE ON A SECONDARY STORAGE MEDIUM.
RECOVERY	RESTORATION OF INTEGRITY TO THE FILES OF THE DATA BASE SO THAT THEY ARE NOT LEFT IN A STATE SUCH THAT ANY OPERATION ON THEM IS PARTIALLY COMPLETE. RECOVERY MAY BE NEEDED AFTER A HALT/LOAD OR IF THE DATA BASE MONITOR IS DS-ED.
RESTART	GETTING THE APPLICATION PROGRAMS RUNNING AGAIN AT THE POINT TO WHICH THE DATA BASE HAS BEEN RECOVERED.
RECONSTRUCTION	USING THE AUDIT TRAIL AND A BACKUP DUMP TO RECONSTRUCT A PORTION OF THE DATA BASE.
RESTORATION	USING A GOOD COPY OF A DUPLICATED DATA BASE FILE TO RESTORE THE BAD COPY.

AUDIT, RECOVERY, RESTART, RECONSTRUCTION AND RESTORATION ARE PROVIDED FOR DM6700. THESE FEATURES ARE DESCRIBED IN THE FOLLOWING SECTIONS.

2. AUDIT

AUDIT IS BY DATA BASE AND NOT BY PROGRAM. THE DECISION OF WHETHER TO AUDIT OR NOT AND WHAT THE PARAMETERS OF THE AUDIT ARE TO BE ARE SPECIFIED IN THE SDL/INITIALIZE RUN. THE AUDIT STATEMENT SYNTAX IS AS FOLLOWS.

2.1. SYNTAX

```
<AUDIT STATEMENT> ::= <AUDIT TYPE><ON-LIST>;/
                      <AUDIT TYPE><OFF-LIST>;
<AUDIT TYPE> ::= AUDIT / AUDITONLY
<OFF-LIST> ::= OFF / NONE
<ON-LIST> ::= <OPTIONAL PARAMETER LIST>/
              ON <OPTIONAL PARAMETER LIST>
<OPTIONAL PARAMETER LIST> ::= <EMPTY>/<PARAMETER>/
              <PARAMETER LIST><OPTIONAL COMMA><PARAMETER>
<OPTIONAL COMMA> ::= , / <EMPTY>
<PARAMETER> ::= <DEVICE SPECIFICATION> /
                <FILESIZE SPECIFICATION> /
                <AUDIT SERIAL SPECIFICATION> /
                <IMAGES SPECIFICATION> /
                <CONTROL CYCLES SPECIFICATION> /
                <PACKNAME SPECIFICATION>
<DEVICE SPECIFICATION> ::= DEVICE = <AUDIT MEDIUM>
<PACK NAME SPECIFICATION> ::= PACKNAME = <PACK NAME>
<AUDIT MEDIUM> ::= DISK / DISKPACK / TAPE7 /
                   TAPE9 / PETAPE
<FILESIZE SPECIFICATION> ::= AREAS = <INTEGER> /
                             AREASIZE = <INTEGER>/
                             FILESIZE = <INTEGER>
<AUDIT SERIAL SPECIFICATION> ::= SERIAL = <INTEGER>
<IMAGES SPECIFICATION> ::= BEFOREIMAGES
<CONTROL CYCLES SPECIFICATION> ::= CONTROLCYCLES =
                                   <INTEGER>
```

EXAMPLES

```
AUDIT;
AUDIT DEVICE=DISK, AREAS=100, AREASIZE=1000, SERIAL=1,
      CONTROLCYCLES=10;
AUDIT OFF;
AUDITONLY DEVICE=DISKPACK, BEFOREIMAGES, FILESIZE=1000;
```

2.2. SEMANTICS

IF THE <AUDIT TYPE> IS AUDITONLY RATHER THAN AUDIT, THE AUDIT MAY BE TURNED ON OR OFF WITHOUT CAUSING INITIALIZATION OF THE DATA BASE FILES.

THE DEFAULT VALUES OF THE AUDIT PARAMETERS ARE SHOWN IN THE SECOND EXAMPLE -- THE FIRST TWO EXAMPLES ARE EQUIVALENT.

THE AUDIT TRAIL MAY BE WRITTEN TO ANY OF THE DEVICES LISTED, AND RECOVERY MAY BE ACCOMPLISHED FROM ANY OF THESE DEVICES.

AS FAR AS THE MCP FILE SYSTEM IS CONCERNED, THE AUDIT TRAIL FILE IS ONE RECORD PER BLOCK WITH A RECORD SIZE OF 30 WORDS. THIS CANNOT CURRENTLY BE VARIED. THE DATA MANAGEMENET SYSTEM DOES ITS OWN BLOCKING AND UNBLOCKING OF THE AUDIT FILE. FOR PURPOSES OF COMPUTING THE AUDIT FILE SIZE, A RECORD SIZE OF 30 WORDS IS USED. IF THE FILESIZE FORM IS USED, AREAS AND AREASIZE ARE COMPUTED SO THAT AREAS*AREASIZE=FILESIZE AND AREASIZE=10*AREAS.

WHEN AN AUDIT FILE BECOMES FULL, IT IS LOCKED, AND A NEW FILE WITH THE NEXT HIGHER SERIAL NUMBER IS CREATED AND USED AUTOMATICALLY. THE AUDIT FILE NAMES ARE:

<DATA BASE NAME>/AUDITNNNN

WHERE NNNN IS THE FOUR DECIMAL DIGIT AUDIT SERIAL NUMBER.

"9999" IS FOLLOWED BY "0000". THE AUDIT SERIAL PARAMETER SPECIFIES WHAT THE STARTING AUDIT SERIAL NUMBER WILL BE.

AFTER IMAGES OF ALTERED DATA AND STRUCTURE INFORMATION ARE ALWAYS WRITTEN TO THE AUDIT TRAIL. AT PRESENT, ONLY THE AFTER IMAGES ARE USED FOR RECOVERY AND RECONSTRUCTION. AS AN OPTION, BEFORE IMAGES MAY ALSO BE INCLUDED IN THE AUDIT TRAIL. THESE WILL BE IGNORED BY RECOVERY AND RECONSTRUCTION, BUT THE USER MIGHT BE ABLE TO USE THEM TO BACK UP HIS DATA BASE. BACKING UP OF THE DATA BASE IS NOT SUPPORTED BY DM6700 AT THIS TIME. THE POINT IN TIME

AT WHICH THE MONITOR IS FIRED UP IS A CLEAN POINT FOR THE DATA BASE AND WILL BE MARKED ON THE AUDIT TRAIL.

THE CONTROL CYCLES SPECIFICATION MAY BE USED TO CONTROL AN IMPORTANT TRADE-OFF IN THE OPERATION OF THE AUDIT. THE DATA BASE MONITOR PERIODICALLY EXAMINES THE STATE OF THE REQUEST HANDLERS AND THE QUEUE OF REQUESTS. ON EVERY NTH SCAN (AS SPECIFIED BY CONTROLCYCLES = <INTEGER>) AN AUDIT CONTROL RECORD IS WRITTEN TO THE AUDIT TRAIL. THIS INVOLVES FORCING A WRITE ON ALL BUFFERS WHICH CAN BE WRITTEN TO DISK AT THAT TIME. THE PURPOSE OF THE AUDIT CONTROL RECORD IS TO LIMIT RECOVERY EFFORT -- AT LEAST ONE AUDIT CONTROL RECORD MUST BE READ BY THE RECOVERY PROGRAM. THUS, THE MORE FREQUENTLY AUDIT CONTROL RECORDS ARE WRITTEN, THE SMALLER THE RECOVERY EFFORT, BUT THE GREATER THE OVERHEAD DURING NORMAL OPERATION OF THE DATA BASE.

THE TIME PERIOD BETWEEN SCANS IS SPECIFIED BY A NEW MONITOR LIMIT. THE SYNTAX IS:

CYCLETIME = <INTEGER>

WHERE THE UNITS OF <INTEGER> ARE SECONDS. THE CURRENT DEFAULT CYCLE TIME IS THREE SECONDS. (MONITOR LIMITS ARE SPECIFIED IN THE LIMITS OR LIMITSONLY STATEMENT IN SDL/INITIALIZE.)

2.3. PRAGMATICS

THE AUDIT TRAIL IS NOT PARTITIONED. (PARTITIONING COULD SPEED RECONSTRUCTION.) NOR IS IT POSSIBLE TO AUDIT ONLY SELECTED PARTS OF THE DATA BASE. PURE RETRIEVAL REQUESTS ARE NOT AUDITED -- ONLY CHANGES TO THE DATA BASE.

THE END OF FILE POINTER OF THE AUDIT TRAIL MUST NOT BE LOST. THEREFORE, THE AUDIT TRAIL IS PROTECTED. FOR DISK OR DISK PACK THIS MEANS THAT THE DISK IS SMEARED WITH A SPECIAL PATTERN AND THEN OVERWRITTEN. THE END OF FILE

RECORD CAN THUS BE FOUND BY PERFORMING A BINARY SEARCH. FOR TAPE PROTECTION MEANS THAT WHILE THE MCP IS COMING UP AFTER A HALT/LOAD, IT WRITES TAPE MARKS TO THE AUDIT TAPE.

THE AUDIT TRAIL IS READ AND WRITTEN WITH DIRECT I/O. AS MUCH BLOCKING IS DONE AS POSSIBLE, BUT THE LOGIC OF THE AUDIT REQUIRES A WRITE TO BE COMPLETED AT CERTAIN POINTS BEFORE FURTHER AUDITING CAN PROCEED. ONE OF THESE POINTS IS AN EXECUTION OF END-TRANSACTION BY A COBOL PROGRAM. THUS, VERY SMALL TRANSACTIONS (E.G., ONE MODIFY/STORE PAIR) WILL RESULT IN SMALL BLOCKS ON THE AUDIT TRAIL. TRANSACTIONS CANNOT BE ARBITRARILY LARGE, HOWEVER. BECAUSE ONLY A FIXED NUMBER OF BUFFERS ARE AVAILABLE. ONCE THESE ARE ALL LOCKED, A SYSTEM OVERLOAD WILL OCCUR. IT IS THE USERS RESPONSIBILITY TO DETERMINE BOTH A REASONABLE TRANSACTION SIZE AND THE MAXIMUM NUMBER OF BUFFERS AVAILABLE TO THE MONITOR.

UNLIKE OTHER DATA BASE FILES, THE PACKNAME OF THE AUDIT FILE MAY BE DIFFERENT THAN THE DATA BASE NAME. IN GENERAL, WHEN AUDITING TO PACK, ONE SHOULD AUDIT TO NAMED PACK (AS OPPOSED TO SYSTEM PACK) WHERE THE NAME IS DIFFERENT THAN THE DATA BASE NAME. THIS DECREASES THE PROBABILITY THAT DATA BASES FILES AND THE AUDIT TRAIL WILL BOTH BE LOST.

2.4. AUDIT ARCHIVE

THE DATE, TIME, SERIAL NUMBER AND PACKNAME OF EACH AUDIT FILE ARE WRITTEN IN AN AUDIT ARCHIVE WHEN THE AUDIT FILE IS CREATED. THE AUDIT ARCHIVE IS A PERMANENT DISK FILE NAMED

DM/<DATA BASE NAME>/AUDITARCHIVE.

SYSTEM/DMPRINTIT MAY BE USED TO PRINT THE AUDIT ARCHIVE IN FORMATTED OR UNFORMATTED FORM.

WHEN AN AUDIT FILE IS NEEDED AS AN INPUT FILE BY SYSTEM/

DMRECOVER, ETC., KIND IS SET TO ZERO AND PACKNAME IS SET FROM THE AUDIT ARCHIVE. SETTING KIND = 0 WILL CAUSE THE SYSTEM TO SEARCH TAPES, DISK AND DISKPACK FOR THE FILE, AND "DUP FILE" WILL RESULT IF MORE THAN ONE COPY IS FOUND. SETTING KIND = 0 HAS SOME NON-OBVIOUS CONSEQUENCES.

IF MORE THAN ONE AUDIT FILE IS MOUNTED ON TAPE DRIVES WHICH ARE ON-LINE, A "DUP FILE" WILL ALWAYS RESULT -- FOR TAPES, THE DUP FILE CONDITION BASED ONLY ON THE FIRST IDENTIFIER. TO ALLEVIATE THIS, DMRECOVER, ETC. DISPLAYS THE NAME OF THE AUDIT FILE THAT IT NEEDS BEFORE OPENING THE FILE.

KIND = 0 WILL CAUSE ONLY ONE FAMILY OF PACKS TO BE SEARCHED (AS DETERMINED BY PACKNAME). THUS, IF THE USER MOVES AN AUDIT FILE TO A FAMILY OF PACKS WHOSE NAME IS NOT EQUAL TO THE PACKNAME IN THE AUDIT ARCHIVE ENTRY, THE USER MUST CHANGE THE PACKNAME IN THE AUDIT ARCHIVE ENTRY. OTHERWISE, IT WILL NOT BE FOUND BY DMRECOVER, ETC. (NOTE: SINCE IT IS POSSIBLE TO AUDIT AND RECOVER FROM TAPE, DISK, OR PACK, AUDIT FILES NEED NOT NORMALLY BE MOVED.)

2.5. OPERATOR INTERFACE

CERTAIN ACCEPT MESSAGES MAY BE INPUT TO THE DATA BASE MONITOR FROM THE SPO AT ANY TIME. WHEN THE MONITOR MAKES ONE OF ITS PERIODIC SCANS, IT WILL MAKE THE APPROPRIATE RESPONSE IF AN ACCEPT MESSAGE WAS PREVIOUSLY INPUT FOR IT. (UNDER 11.4 PROGRAMS CAN TEST FOR THE PRESENCE OF ACCEPT MESSAGES WITHOUT BEING SUSPENDED.) THE MIX NUMBER OF THE MONITOR (NOT THE JOB NUMBER OR THE MIX NUMBER OF ONE OF THE REQUEST HANDLERS) IS ENTERED, FOLLOWED BY "AX", FOLLOWED (WITH NO INTERVENING BLANKS) BY ONE OF THE FOLLOWING MESSAGES.

AS WILL CAUSE THE MONITOR TO DISPLAY THE CURRENT AUDIT SERIAL NUMBER.

AR WILL CAUSE THE MONITOR TO DISPLAY THE CURRENT AUDIT RECORD NUMBER. FOR DISK OR DISKPACKS THIS IS THE SEGMENT NUMBER, AND FOR TAPE THIS IS THE BLOCK NUMBER.

CA WILL CAUSE THE MONITOR TO CLOSE THE CURRENT AUDIT FILE AND OPEN A NEW ONE.

ALL OTHER MESSAGES WILL CAUSE THE MONITOR TO DISPLAY "INVALID REQUEST".

3. RECOVERY

RECOVERY OF A DATA BASE IS ACCOMPLISHED BY RUNNING SYSTEM/DMRECOVER. THIS PROCEDURE IS CALLED AUTOMATICALLY BY THE DATA BASE MONITOR WHEN NECESSARY. IT MAY ALSO BE RUN BY THE USER. ITS PARAMETER IS A STRING CONSISTING OF THE NAME OF THE SDL FILE (OR THE FIRST COPY, IF DUPLICATED) FOLLOWED BY A PERIOD. DMRECOVER RECOVERS THE DATA BASE, WRITES ALL NECESSARY RESTART FILES TO DISK, AND INDICATES ITS SUCCESSFUL COMPLETION. IT ALSO KEEPS A LOG OF ANY I/O ERRORS IT ENCOUNTERS.

3.1. PRAGMATICS

THE MONITOR TURNS A BIT ON (THE DATA BASE IN USE BIT) IN RECORD ZERO OF THE SDL FILE BEFORE IT STARTS PROCESSING REQUESTS TO THE DATA BASE. IT TURNS THIS BIT OFF WHEN IT GOES TO SUCCESSFUL EOJ. THUS, WHEN THE MONITOR COMES UP AND NOTICES THAT THIS BIT IS ALREADY ON, IT KNOWS THAT EITHER A HALT/LOAD OCCURRED OR IT WAS DS-ED. IN EITHER CASE, RECOVERY IS NECESSARY, SO IT RUNS DMRECOVER. IT THEN HANGS IN A "NO FILE" FOR FILE DM/<DATA BASE NAME>/RECOVERED. DMRECOVER RECOVERS THE DATA BASE, WRITES THE RESTART FILES TO DISK, AND LOGS ANY I/O ERRORS IT ENCOUNTERS IN A DUPLICATED FILE NAMED DM/<DATA BASE NAME>/"RECOVERRS#1" AND DM/<DATA BASE NAME>/"RECOVERRS#2". SOME I/O ERRORS RESULT IN UNSUCCESSFUL RECOVERY. IF

RECOVERY IS SUCCESSFUL, IT TURNS OFF THE DATA BASE IN USE BIT AND CREATES AN EMPTY FILE DM/<DATA BASE NAME>/RECOVERED. THE CREATION OF THAT FILE CAUSES THE MONITOR TO RESUME PROCESSING. IF THE MONITOR IS DS-ED WHILE RECOVERY IS RUNNING, A CRITICAL BLOCK EXIT WILL RESULT IN RECOVERY. IF RECOVERY IS DS-ED, THERE WILL BE NO EFFECT ON THE MONITOR (I.E., THE MONITOR WILL STILL HANG IN THE "NO FILE" UNTIL RECOVERY IS SUCCESSFUL). IF RECOVERY IS DS-ED, IT MAY BE POSSIBLE TO CORRECT THE SITUATION AND RUN IT AGAIN SUCCESSFULLY.

NO MORE THAN ONE COPY OF DMRECOVER WILL RUN AT A TIME. THIS SITUATION IS DETECTED BY TESTING THE POPULATION ATTRIBUTE ON THE SDL FILE. IF IT IS GREATER THAN ONE, RECOVERY GOES TO EOJ. THUS, IF TWO COPIES OF RECOVERY SOMEHOW START UP AT THE SAME TIME, BOTH MAY GO GO EOJ. IN THIS CASE, IT WILL HAVE TO BE RUN AGAIN, ALONE.

THE RECOVERY ERRORS FILE HAS THREE WORD RECORDS WITH THE FOLLOWING FORMAT.

WORD 0

47:20 STRUCTURE NUMBER OF THE FILE IN ERROR
27:28 RECORD IN ERROR

WORD 1

16:03 TYPE OF FILE IN ERROR:
- 0 = DATA FILE (OR INDEX TABLE)
- 1 = SDL FILE
- 2 = AUDIT TRAIL
- 3 = UNUSED
- 4 = RECOVERY ERRORS FILE ITSELF

13:01 READ OR WRITE ERROR
- 0 = READ
- 1 = WRITE

12:01 COPY NUMBER IN ERROR (0 FIRST, 1 SECOND)
11:12 BLOCKFACTOR OF FILE IN ERROR

WORD 2

47:48 I-O RESULT DESCRIPTOR -- THE HARDWARE RESULT DESCRIPTOR IF THE AUDIT TRAIL; OTHERWISE, THE SOFTWARE RESULT DESCRIPTOR

3.2. STRATEGY

RECOVERY READS THE AUDIT TRAIL BACKWARDS FROM THE CURRENT END AS FAR AS NECESSARY.

DATA RECORDS

RECORDS MAY ONLY BE MODIFIED BY ONE TRANSACTION AT A TIME. RECORDS ARE NOT UNLOCKED UNTIL AFTER END-TRANSACTION HAS BEEN EXECUTED AND AUDITED. RECORDS MAY NOT BE WRITTEN TO DISK UNTIL END-TRANSACTION HAS BEEN EXECUTED AND AUDITED. THUS, RECOVERY ASSUMES THAT NO PARTIAL TRANSACTIONS ARE REFLECTED ON THE DISK. THE JOB OF RECOVERY FOR DATA RECORDS IS TO INSURE THAT ALL DISK WRITES FOR COMPLETED TRANSACTIONS HAVE OCCURRED BY WRITING ALL SUCH RECORDS USING THE AFTER IMAGES IN THE AUDIT TRAIL. IF THE WRITE HAD ALREADY OCCURRED, A SECOND WRITE WILL DO NO HARM. RECOVERY MUST REMEMBER ALL THE RECORDS IT HAS WRITTEN SO IT WONT REWRITE THEM WITH AN EARLIER VERSION.

STRUCTURE INFORMATION

TREATING THE LOCKING OF STRUCTURES THE SAME AS THE LOCKING OF DATA RECORDS WOULD RESULT IN SET LEVEL LOCKOUT, SO THIS IS NOT DONE. INSTEAD, STRUCTURES ARE LOCKED ONLY WHILE BEING CHANGED. THIS CHANGES THE RECOVERY STRATEGY. STRUCTURES ARE "RECOVERED" WHEN FIRST ENCOUNTERED IN THE AUDIT TRAIL. STRUCTURE CHANGES FOR THE LAST PARTIALLY COMPLETED TRANSACTIONS ARE UNDONE AS THEY ARE ENCOUNTERED IN THE AUDIT TRAIL.

TERMINATION

THE PURPOSE OF CONTROL RECORDS IS TO LIMIT RECOVERY EFFORT. CONTROL RECORDS ARE WRITTEN PERIODICALLY BY THE MONITOR. AT THIS TIME THE MONITORS BUFFERS ARE FLUSHED. THOSE RECORDS AND STRUCTURES THAT COULD NOT BE FLUSHED BECAUSE THEY WERE LOCKED ARE LISTED IN THE CONTROL RECORD. THE CURRENTLY ACTIVE TRANSACTION IDS ARE ALSO LISTED IN THE CONTROL RECORD.

AT LEAST ONE CONTROL RECORD MUST BE SEEN BY RECOVERY (UNLESS THE MONITOR-FIRED-UP RECORD IS ENCOUNTERED).

END-TRANSACTION OR BOJ (ACTUALLY, BOJ IS THE FIRST OPEN OF THE TRANSACTION) MUST BE ENCOUNTERED FOR ALL USER TRANSACTIONS ACTIVE DURING AND AFTER THE LAST CONTROL RECORD (OR SINCE THE MONITOR-FIRED-UP RECORD). THE FIRST TIME END-TRANSACTION IS SEEN FOR A USER TRANSACTION, A RESTART FILE IS CREATED ON DISK. IF NO END-TRANSACTION IS SEEN, BOJ WILL BE SEEN. IT IS ASSUMED THAT ALL USERS EXECUTE END-TRANSACTION PERIODICALLY. IF THIS IS NOT TRUE, RECOVERY MAY HAVE TO READ A LARGE PORTION OF THE AUDIT TRAIL TO TERMINATE.

ALL STRUCTURES AND DATA RECORDS LISTED IN THE LAST CONTROL RECORD MUST BE ACCOUNTED FOR. THEY MUST BE ENCOUNTERED IN THE AUDIT TRAIL OR MISSING FROM AN EARLIER CONTROL RECORD TO INSURE THAT THEY GET WRITTEN OR WERE WRITTEN TO DISK.

THE END OF FILE POINTERS FOR ALL THE DATA FILES MUST BE ESTABLISHED USING THE HIGHEST OPEN INFORMATION IN THE SDL STRUCTURE FILE. (THE END OF FILE OF THE SDL FILE IS CONSTANT, AND THE AUDIT TRAIL IS PROTECTED.) TO ESTABLISH THEM, A BLOCK OF ALL ZEROES IS WRITTEN AS THE NEXT BLOCK AFTER THE LAST BLOCK CONTAINING DATA.

4. RESTART

4.1. INTRODUCTION

TRANSACTIONS HAVE BEEN INTRODUCED TO PROVIDE CLEAN POINTS FOR RESTARTING DATA MANAGEMENT JOBS. THE UNDERLYING ASSUMPTION IS THAT PROGRAMS ARE CYCLIC IN NATURE, AND THAT AT THE BEGINNING OF A LOOP, ONLY A FEW IMPORTANT VARIABLES ARE NECESSARY TO PROVIDE SUFFICIENT INFORMATION TO RESTART THE PROGRAM. THESE VARIABLES WILL TYPICALLY BE FILE POSITIONS. A RESTARTABLE DATA MANAGEMENT JOB WILL BE PASSED TWO PIECES OF INFORMATION UPON RESTART: (1) THE FACT THAT IS HAS BEEN RESTARTED AND (2) EXACTLY THOSE VARIABLES WHICH THE PROGRAMMER HAS DECIDED ARE SUFFICIENT TO GET THE PROGRAM RESTARTED. THE DETAILS ARE DISCUSSED IN WHAT FOLLOWS.

4.2. TRANSACTIONS

A TRANSACTION IS A SEQUENCE OF DATA BASE OPERATIONS PERFORMED ON ONE OR MORE DATA MANAGEMENT COBOL WORKAREAS. A TRANSACTION HAS AN ID UNIQUE FROM ALL OTHER CURRENTLY ACTIVE TRANSACTIONS. THE TRANSACTION ID SERVES TO LINK RECORDS IN THE AUDIT TRAIL AND TO IDENTIFY THE LOCKER OF RECORDS. ALL DATA RECORDS AND INDEX TABLES MODIFIED BY A TRANSACTION ARE LOCKED OUT FROM OTHER TRANSACTIONS UNTIL THE TRANSACTION IS COMPLETED. THEY ARE ALSO MARKED SO THAT THEY CANNOT BE WRITTEN TO DISK UNTIL THE TRANSACTION IS COMPLETED. THERE ARE TWO TYPES OF TRANSACTIONS, DEFAULT TRANSACTIONS AND USER SPECIFIED TRANSACTIONS. ANY PARTICULAR COBOL PROGRAM USES ONE OR THE OTHER, BUT NOT BOTH.

4.2.1. DEFAULT TRANSACTIONS

DM6700 HAS BEEN USING DEFAULT TRANSACTIONS. CURRENT PROGRAMS MAY BE RUN WITHOUT CHANGE. HOWEVER, TO PROVIDE RESTARTABILITY, IT WILL BE NECESSARY TO CHANGE TO USER TRANSACTIONS. DEFAULT TRANSACTIONS ARE SINGLE DATA BASE OPERATIONS WITH THE EXCEPTION OF THE FOLLOWING PAIRS OF OPERATIONS (ON THE SAME WORKAREA) -- MODIFY/STORE,

CREATE/STORE, MODIFY/FREE AND CREATE/FREE. THESE PAIRS OF OPERATIONS ARE CONSIDERED SINGLE DEFAULT TRANSACTIONS.

THE TYPE OF TRANSACTIONS A PROGRAM IS USING MAY BE DETERMINED BY INSPECTING THE FILE CONTROL PART OF THE INPUT-OUTPUT SECTION. IF NO FILE SELECTED IS ASSIGNED TO DATA-BASE-RESTART, THEN THE PROGRAM IS USING DEFAULT TRANSACTIONS. IN THIS CASE, THE COBOL VERBS END-TRANSACTION AND ABORT-TRANSACTION WILL GIVE SYNTAX ERRORS, AND EACH SET THE PROGRAM INVOKES WILL BE ALLOCATED A UNIQUE TRANSACTION ID WHEN THE SET IS OPENED.

4.2.2. USER TRANSACTIONS

IF A FILE IS ASSIGNED TO DATA-BASE-RESTART, THEN THE COBOL PROGRAM IS USING USER SPECIFIED TRANSACTIONS. (AT MOST ONE FILE MAY BE ASSIGNED TO DATA-BASE-RESTART.) ALL SETS INVOKED BY THE PROGRAM ARE GIVEN THE SAME TRANSACTION ID WHEN OPENED. ALL OPERATIONS PERFORMED ON THE WORKAREAS FOR THESE SETS ARE PART OF THE SAME TRANSACTION, UNTIL THE TRANSACTION IS TERMINATED BY EXECUTING END-TRANSACTION OR ABORT-TRANSACTION. (THE PROGRAM MAY THEN INITIATE ANOTHER TRANSACTION.) THE PROGRAM IS EXPECTED TO EXECUTE END-TRANSACTION PERIODICALLY. IF IT DOES NOT, RECOVERY MAY TAKE MUCH LONGER THAN NECESSARY (SEE SECTION 3); AND IF THE TRANSACTION MODIFIES TOO MANY RECORDS, A SYSTEM OVERLOAD (STATUS 102) WILL OCCUR ONCE ALL THE BUFFERS ARE LOCKED. IT IS THE PROGRAMMERS RESPONSIBILITY TO KEEP THE TRANSACTIONS SIZE REASONABLE.

IF A SET IS PASSED AS A PARAMETER, ALL THE OPERATIONS ON IT BY OTHER PROGRAMS OR PROCEDURES ARE CONSIDERED AS PART OF THE TRANSACTION OF THE PROGRAM WHERE THE SET IS LOCALLY INVOKED. THAT IS, IF A SET IS INVOKED LOCALLY (AS OPPOSED TO BEING INVOKED BY REFERENCE) BY A PROGRAM, PASSING IT AS A PARAMETER DOES NOT CHANGE ITS TRANSACTION ID.

4.2.2.1. PURPOSE OF THE RESTART FILE

THE PURPOSE OF THE RESTART FILE IS TO HOLD ALL THE INFORMATION NECESSARY TO RESTART THE PROGRAM. FILE POSITIONS, LOOP COUNTERS, STATE VARIABLES, DATA MANAGEMENT KEYS OF RETRIEVAL, ETC. MAY RESIDE IN THE RECORD DESCRIPTION OF THE RESTART FILE. WHEN END-TRANSACTION IS EXECUTED, THE RECORD AREA OF THE RESTART FILE (HEREAFTER REFERRED TO AS THE RESTART AREA) IS WRITTEN ON THE AUDIT TRAIL (WHETHER THE RESTART FILE IS OPEN OR NOT). THE DATA MANAGEMENT SYSTEM WILL INSURE THAT ONLY COMPLETE TRANSACTIONS ARE PERFORMED ON THE DATA BASE. THE INTENT IS THAT THE END OF A TRANSACTION BE A CLEAN POINT FOR THE PROGRAM.

THE RESTART FILE SHOULD NOT NORMALLY BE OPENED OR WRITTEN BY THE USER. IT NEED ONLY BE OPENED AND READ WHEN THE PROGRAM IS RESTARTED. OPENING IT OR WRITING IT, HOWEVER, WILL NOT INTERFERE WITH ITS USE BY THE DATA MANAGEMENT SYSTEM.

4.2.2.2. END-TRANSACTION

THE SYNTAX IS:

END-TRANSACTION ON EXCEPTION
<STATEMENT> [ELSE <STATEMENT>]

THIS STATEMENT IS ALWAYS EXECUTED SYNCHRONOUSLY.

THE EXCEPTIONS ARE RETURNED IN TRANSACTION-STATUS (WHICH IS A COMP-1 EXPRESSION) AND ARE:

1 SET NOT OPEN. IF NONE OF THE WORKAREAS FOR THE TRANSACTION ARE CURRENTLY OPEN, THE TRANSACTION ID IS NOT ALLOCATED AND END-TRANSACTION WILL NOT BE PERFORMED.

16 OPERATIONS IN PROGRESS. ONE OR MORE (ASYNCHRONOUS) OPERATIONS ON ONE OR MORE OF THE

WORKAREAS OF THE TRANSACTION ARE STILL IN PROGRESS. END-TRANSACTION WILL NOT BE EXECUTED BECAUSE THE PROGRAM IS NOT AT A CLEAN POINT.

17 ONE OR MORE WORKAREAS OF THE TRANSACTION ARE IN THE MODIFY OR CREATE STATE. END-TRANSACTION WILL NOT BE EXECUTED BECAUSE THE PROGRAM IS NOT AT A CLEAN POINT.

18 ONE OR MORE OF THE DATA BASE MONITORS INVOLVED IN THE TRANSACTION HAVE BEEN US-ED.

NEW OPERATIONS WILL BE INHIBITED ON ANY OF THE WORKAREAS INVOLVED IN THE TRANSACTION (NEW OPERATIONS MIGHT OTHERWISE BE INITIATED IF MORE THAN ONE STACK IS INVOLVED IN THE TRANSACTION) WHILE END-TRANSACTION IS BEING PERFORMED.

4.2.2.3. ABORT-TRANSACTION

TRANSACTIONS MAY BE ABORTED WHETHER AUDIT IS ON OR OFF. THE MONITOR DOES AN IMPLICIT ABORT-TRANSACTION WHEN THE LAST SET INVOLVED IN A TRANSACTION IS CLOSED. THUS, THE DATA BASE IS LEFT AT THE END OF THE LAST COMPLETE TRANSACTION FOR THAT JOB.

UNLIKE END-TRANSACTION, ABORT-TRANSACTION WAITS FOR COMPLETION OF OPERATIONS CURRENTLY IN PROGRESS, AND DOES IMPLICIT FREES ON WORKAREAS IN THE MODIFY OR CREATE STATE. LIKE END-TRANSACTION, IT IS ALWAYS EXECUTED SYNCHRONOUSLY. THE SYNTAX IS

```
ABORT-TRANSACTION ON EXCEPTION
    <STATEMENT> [ELSE <STATEMENT>]
```

AS IS TRUE FOR END-TRANSACTION, NEW OPERATIONS FOR THE TRANSACTION ARE INHIBITED AND THE STATUS IS RETURNED IN TRANSACTION-STATUS. THE VALUES 1 AND 18 APPLY AND MEAN THE SAME AS FOR END-TRANSACTION.

ABORT-TRANSACTION DOES NOT DESTROY THE USERS CURRENTS BUT GUARANTEES THAT NONE OF THE CHANGES THE TRANSACTION HAS MADE ARE REFLECTED IN THE DATA BASE.

4.2.2.4. SUMMARY OF STATUS VALUES

THE STATUS VALUES RELEVANT TO USER TRANSACTIONS ARE:

1 NO SET OF THE TRANSACTION IS OPEN AT END-TRANSACTION OR ABORT-TRANSACTION.

16 OPERATIONS IN PROGRESS AT END-TRANSACTION.

17 ONE OR MORE WORKAREAS IN MODIFY OR CREATE STATE AT END-TRANSACTION.

18 ONE OR MORE DATA BASE MONITORS ARE DS-ED.

19 ON AN OPEN, MORE THAN 62 RESTART AREAS WOULD BE IN USE BY A SINGLE STACK.

23 ATTEMPT TO UTILIZE USER SPECIFIED TRANSACTIONS, BUT THIS VERSION OF THE DATA BASE MONITOR CAN HANDLE ONLY DEFAULT TRANSACTIONS.

102 TOO MANY BUFFERS REQUIRED.

4.3. RESTART CONVENTIONS

THE PHILOSOPHY OF DATA MANAGEMENT RESTART IS THAT JOB AND TASK RESTART ARE THE RESPONSIBILITY OF THE WORKFLOW MANAGEMENT SYSTEM (WFM), AND THAT DATA MANAGEMENT RESTART SHOULD BE INTEGRATED INTO THIS SYSTEM.

THE CURRENT WORKFLOW MANAGEMENT PHILOSOPHY IS THAT JOBS ARE ROLLED OUT AT CLEAN POINTS (I.E., NO TASKS ARE CURRENTLY RUNNING) AND RESTARTED AT THE LAST ROLLOUT POINT. THAT IS, WFM PROVIDES JOB RESTART AND THE USER PROGRAMS HIS TASK RESTART WITHIN HIS JOB IN THE WORKFLOW LANGUAGE (WFL). THIS IS ACCOMPLISHED FOR DATA MANAGEMENT

BY USING WFL TO SET THE TASKVALUE OF THE USER PROGRAM SO THAT IT CAN FIND ITS RESTART FILE.

4.3.1. RESTART FILE

4.3.1.1. NAME

THE NAME OF THE RESTART FILE IS, BY CONVENTION,

DMRESTART/<DATA BASE NAME>/<INTERNAL FILE NAME>/
<11 DECIMAL DIGITS>

WHERE THE 11 DECIMAL DIGIT FIELD IS COMPOSED OF THE FOLLOWING THREE SUBFIELDS:

<4 DIGIT ID FIELD><4 DIGIT JOB#><3 DIGIT LT#>

THE FOUR DIGIT ID FIELD IS ZERO FOR FILES PRODUCED BY DMRECOVER. THE THREE DIGIT LT# IS THE THREE DIGIT LOGICAL TASK NUMBER OF THE TASK WITHIN THE JOB. THE USER ASSIGNS THIS NUMBER OF THE TASK HIMSELF, BY USING WFL TO SET THE TASKVALUE OF HIS TASK AS DESCRIBED IN SECTION 4.3.2. IT SHOULD BE UNIQUE FOR EACH TASK OF THE JOB AND SHOULD BE IN THE RANGE ONE TO 255 INCLUSIVE.

ALL NECESSARY RESTART FILES ARE PRODUCED AUTOMATICALLY BY DMRECOVER WHEN IT IS RUN. HOWEVER, IN THE CASE WHERE A DATA MANAGEMENT JOB GETS DS-ED FOR ANY REASON BUT THE MONITOR GOES TO NORMAL EOJ, DMRECOVER WILL NOT AND MUST NOT BE RUN. IN THIS CASE, THE BUG MUST FIRST BE FIXED, THEN THE JOB MUST BE RESUBMITTED UTILIZING THE DELAYED RESTART CONVENTIONS. THE USER MUST OBTAIN HIS RESTART FILE BY RUNNING SYSTEM/GETDMRESTARTFILE.

4.3.1.2. SYSTEM-GETDMRESTARTFILE

THE INPUT FILE TO THIS PROGRAM IS IN A FILE CALLED CARD. INPUT IS FREE FORM IN COLUMNS 1-72. THE INPUT IS A LIST OF KEYWORD AND VALUE PAIRS. COMMAS MUST NOT BE USED TO SEPARATE THE PARAMETER PAIRS. BLANKS ARE USED AS THE

DELIMITER INSTEAD. THE FORM OF EACH PAIR IS <KEYWORD>=<VALUE>. THE EQUAL SIGN MUST BE PRESENT. THE KEYWORD MAY BE ABBREVIATED OR MISSPELLED OR CONTAIN IMBEDDED BLANKS OR SPECIAL CHARACTERS AS LONG AS THE FIRST FEW CHARACTERS ARE CORRECT. THE SYNTAX OF THE VALUE, HOWEVER, MUST BE EXACTLY CORRECT. FOR EXAMPLE, IF FOUR DIGITS ARE REQUIRED, THEN LEADING ZEROES, IF ANY, MUST BE PUNCHED. THE PARAMETERS MAY APPEAR IN ANY ORDER. THE PARAMETERS ARE AS FOLLOWS:

PARAMETER NAME	VALUE	REQUIRED
DATABASE NAME	= <DATA BASE NAME>	YES
INTERNAL FILE NAME	= <INTNAME OF RESTART FILE>	YES
MIX NUMBER	= <4 DIGITS>	YES
JOB NUMBER	= <4 DIGITS>	YES
DATE	= MM/DD/YY	*
JULIAN DATE	= YYDDD	*
TIME	= HH:MM:SS	*
AUDIT SERIAL NUMBER	= <4 DIGITS>	*
ID	= <4 DIGITS>	NO

(*) EITHER THE AUDIT SERIAL NUMBER IS SPECIFIED OR THE DATE AND TIME ARE BUT NOT BOTH. IF THE DATE AND TIME ARE SPECIFIED, THE AUDIT ARCHIVE IS USED TO DETERMINE THE AUDIT SERIAL NUMBER. IN ANY CASE, THE AUDIT ARCHIVE IS USED TO FIND THE PACK NAME, IF ANY.

NEITHER THE DATA BASE NAME NOR THE INTERNAL FILE NAME SHOULD BE TERMINATED WITH A PERIOD. THE DATE AND TIME SHOULD BE THOSE AT THE END OF THE TASK. THE ID FIELD, IF PRESENT, WILL BE PLACED IN THE ID FIELD OF THE RESTART FILE NAME. THE DEFAULT VALUE OF ID IS ZERO.

EXAMPLE 1

```
DATA BASE NAME = MYDB MIX-NUMBER = 0032
JOB-NUMBER = 0031
TIME = 00:01:00 DATE = 01/02/73
```

INTERNAL FILE NAME = MYRESTARTFILE

EXAMPLE 2

DATA = TESTDB AUDIT = 0003 JOB = 1234 MIX = 1236
ID = 7777 INTNAME = RSF

EXAMPLE ONE MAY PRODUCE A FILE NAMED DMRESTART/MYDB/MYRESTARTFILE/00000031LLL, IF SUCCESSFUL, WHERE LLL WAS THE LOGICAL TASK NUMBER OF THE TASK. (SEE NEXT SECTION.)

EXAMPLE TWO MAY PRODUCE A FILE NAMED DMRESTART/TESTDB/RSF/77771234LLL, WHERE LLL WAS THE LOGICAL TASK NUMBER.

ALL INPUT NEEDED BY GETDMRESTARTFILE MAY BE OBTAINED FROM THE LISTING THE SYSTEM PRODUCES WHEN THE JOB IS RUN, WITH THE POSSIBLE EXCEPTIONS OF THE DATA BASE NAME AND THE INTERNAL FILE NAME. THESE MAY, OF COURSE, BE OBTAINED FROM A LISTING OF THE PROGRAM.

GETDMRESTARTFILE WILL READ THE AUDIT FILE SPECIFIED BY THE AUDIT SERIAL NUMBER BACKWARDS UNTIL IT EITHER REACHES THE BEGINNING OR FINDS THE RESTART AREA. IF IT REACHES THE BEGINNING OF THE AUDIT FILE, IT QUITS AND PRINTS A MESSAGE INDICATING THAT THE RESTART FILE WAS NOT FOUND. IF IT ENCOUNTERS THE RESTART AREA, IT WRITES IT TO DISK AS A PERMANENT FILE, INDICATES THE TITLE IN ITS REPORT, AND THEN TERMINATES.

4.3.2. TASKVALUE

THE SIGN OF THE TASKVALUE IS USED TO INDICATE HALT/LOAD RESTART. IF THE TASKVALUE IS POSITIVE, THEN IT IS A HALT/LOAD RESTART. THE MAGNITUDE OF THE TASKVALUE IS USED TO INDICATE A NORMAL RUN OR A DELAYED RESTART. FOR A NORMAL RUN, THE ABSOLUTE VALUE OF THE TASKVALUE IS LESS THAN 256 AND IS THE LOGICAL TASK NUMBER. FOR A DELAYED RESTART, THE MAGNITUDE OF THE TASKVALUE IS GREATER THAN 255 AND IS THE ENTIRE THIRD IDENTIFIER OF THE RESTART

FILE NAME.

EXAMPLE 1

TO RUN A NORMAL JOB, THE FOLLOWING WFL PROGRAM COULD BE USED.

```
? JOB EXAMPLE1
? BEGIN
V := -1;
ON RESTART, IF V < 0 THEN V := -V;
RUN TEST1; VALUE = V;
? END JOB;
```

EXAMPLE 2

TO DO A DELAYED RESTART, ONLY ONE LINE OF THE PREVIOUS EXAMPLE NEED BE CHANGED. ASSUME THAT THE OLD JOB NUMBER WAS 9999 AND GETDMRESTARTFILE HAS BEEN RUN SUCCESSFULLY. THEN THE FOLLOWING WILL PROVIDE A DELAYED RESTART WITH FULL HALT/LOAD PROTECTION:

```
? JOB EXAMPLE1
? BEGIN
V := -9999001;
ON RESTART, IF V< 0 THEN V := -V;
RUN TEST1; VALUE = V;
? END JOB;
```

4.3.3. EXAMPLE COBOL JOB

THE FOLLOWING IS A SKELETON OF A RESTARTABLE DATA MANAGEMENT COBOL JOB. THE PROGRAM READS AN INPUT DECK OF CARDS. EACH CARD CONTAINS ENOUGH INFORMATION TO UPDATE ONE MEMBER OF THE DATA MANAGEMENT SET PERSONNEL. THE UPDATING OF ONE MEMBER IS A TRANSACTION. THE POINT OF THE RESTART CODE IS TO SKIP THE CARDS THAT HAVE ALREADY BEEN PROCESSED.

THE PROGRAM HAS FULL PROTECTION IN ALL CASES, INCLUDING HALT/ LOADS DURING A DELAYED RESTART.

EXAMPLE COBOL PROGRAM

```
IDENTIFICATION DIVISION.
     .
     .
     .
ENVIRONMENT DIVISION.
     .
     .
     .
INPUT-OUTPUT SECTION.
FILE-CONTROL.
     SELECT XXXX ASSIGN TO DATA-BASE-RESTART.
     .
     .
     .
DATA DIVISION.
FILE SECTION.
FD XXXX VALUE OF ID IS XID.
01     CARDSIN PIC 9(5) COMP.
     .
     .
     .
DATA-BASE SECTION.
01     INVOKE PERSONNEL IN COMPANYX.
     .
     .
     .
WORKING-STORAGE SECTION.
77     I PIC 9(5) COMP-1.
     .
     .
     .
01 XID.
```

```
        02 X1 PIC X(24) VALUE "DMRESTART/COMPANYX/XXXX/".
        02 X2 PIC 9(11).
        02 X3 PIC X VALUE ".".
        .
        .
        .
    PROCEDURE DIVISION.
    THEONLY SECTION.
    DUMLAB.  OPEN INPUT CARD.
        OPEN OUTPUT PRINT.
        OPEN PERSONNEL UPDATE ON EXCEPTION GO DIE.
        .
        .
        .
        IF MYSELF(TASKVALUE) > 0 GO HALT-LOADED.
        IF ABS(MYSELF(TASKVALUE)) > 255 GO DELAYED-RESTART.
        GO START-PGM.
    DELAYED-RESTART.  COMPUTE X2 = ABS(MYSELF(TASKVALUE)).
        SET XXXX (TITLE) TO XID.
        SET XXXX (FILETYPE) TO 7.
        IF XXXX (PRESENT) = VALUE FALSE
           COMPUTE I = I/0 ELSE GO CONTINUE-RESTART.
    HALT-LOADED.  COMPUTE X2 =
        1000*MYSELF(EXCEPTIONTASK(STACKNO)) +
        (MYSELF(TASKVALUE) MOD 1000).
        SET XXXX (TITLE) TO XID.
        SET XXXX (FILETYPE) TO 7.
        IF XXXX (PRESENT) = VALUE FALSE
        IF MYSELF(TASKVALUE) > 255 GO DELAYED-RESTART
           ELSE GO START-PGM.
    CONTINUE-RESTART.
        READ XXXX AT END COMPUTE I = I/0.
        CLOSE XXXX.
        MOVE CARDSIN TO I.
        MOVE 0 TO CARDSIN.
        PERFORM READ-CARD I TIMES.
```

```
    GO LOOP.

START-PGM.  MOVE 0 TO CARDSIN.
LOOP.  PERFORM READ-CARD.
    <DATA BASE OPERATIONS TO UPDATE ONE MEMBER
     OF PERSONNEL SET>
    END-TRANSACTION ON EXCEPTION GO DIE.
    GO LOOP.

EOJ.  STOP RUN.

DIE.  COMPUTE I = I/O.

READ-CARD.

    READ CARD AT END GO EOJ.
    ADD 1 TO CARDSIN.
```

4.4. PITFALLS

THE SYSTEM RESTARTS JOBS BY DEFAULT. IF THIS IS NOT DESIRED, THE JOB MUST BE CODED IN WFL SO THAT IT WONT RESTART.

THE TASKVALUE OF A TASK MUST BE TESTED AFTER THE FIRST OPEN ON A DATA MANAGEMENT SET. IF IT IS NOT, RECOVERY MAY NOT YET HAVE BEEN RUN AND HAVE CREATED THE APPROPRIATE RESTART FILES.

DIRECT I/O MUST BE USED BY THE COBOL PROGRAM TO SYNCHRONIZE ITS OUTPUT FILES WITH END-TRANSACTION. USING LOGICAL I/O WILL RESULT IN THE LOSS OF WHAT WAS IN THE CORE BUFFERS IF A HALT/ LOAD OCCURS.

IN THE RESTART CODE, WHEN POSITIONING FILES FOR RESTART, IT IS TYPICAL THAT THE COUNTERS IN THE RESTART AREA WILL BE BUMPED. ONE WAY TO GET AROUND THIS IS TO MOVE THE COUNTERS IN THE RESTART AREA TO TEMPORARY VARIABLES, ZERO OUT THE RESTART AREA, AND USE THE TEMPORARY VARIABLES FOR REPOSITIONING COUNTS.

SOME SCHEME FOR REMOVING OLD RESTART FILES MUST BE DEVISED BY THE INSTALLATION.

4.5. SUMMARY OF RESTART PROCEDURES

RESTART AFTER A HALT/LOAD IS AUTOMATIC. IF THE COBOL PROGRAM IS DS-ED, A DELAYED RESTART MUST BE PERFORMED. GETDMRESTARTFILE MAY BE USED TO OBTAIN THE RESTART FILE IN THIS CASE. IF THE MONITOR WAS ALSO DS-ED AND DMRECOVER SUBSEQUENTLY RUN, THE RESTART FILE WILL ALREADY BE THERE, BUT GETTING IT AGAIN WITH GETDMRESTARTFILE WILL DO NO HARM.

5. DUPLICATED FILES

5.1. INTRODUCTION

DM6700 NOW SUPPORTS DUPLICATED FILES FOR ALL DATA MANAGEMENT FILES EXCEPT DM/<DATABASE-NAME>/DDL, DM/<DATABASE-NAME>/AUDITARCHIVE AND THE AUDIT TRAIL.

DATA MANAGEMENT DUPLICATED FILES ARE NOT SYSTEM DUPLICATED FILES. THIS IS BECAUSE SYSTEM DUPLICATED FILES ARE NOT CURRENTLY SUPPORTED ON DISK PACK NOR WITH DIRECT I/O. THE ERRORS FILE, DM/<DATABASE-NAME>/ERRORS IS THE ONE EXCEPTION. IT IS FULLY DESCRIBED IN SECTION 7.2. AND, IF DUPLICATED, WILL BE A SYSTEM DUPLICATED FILE.

THE SYNTAX WITH WHICH DM6700 FILES (I.E., BOTH FILES AND INDEXES) ARE DUPLICATED IS INCORPORATED INTO BOTH SDL/STRUCTURE AND SDL/INITIALIZE.

IF A DM6700 FILE IS DUPLICATED, BOTH COPIES MUST RESIDE ON THE SAME DEVICE (I.E., ONE COPY CANNOT BE ON DISK PACK WHILE THE OTHER IS ON HEAD-PER-TRACK).

5.2. NAMING CONVENTION

IF A DM FILE IS TO BE DUPLICATED, ONLY ONE COPY WILL BE KEPT. THE NAMES OF DM6700 DUPLICATED FILES ARE:

DM/<DATABASE-NAME>/"<STRUCTURE-NUMBER>#1"
DM/<DATABASE-NAME>/"<STRUCTURE-NUMBER>#2"

FOR SDL THE NAMES ARE:

DM/<DATABASE-NAME>/"SDL#1"
DM/<DATABASE-NAME>/"SDL#2"

5.3. SDL INTERFACE

THERE ARE TWO WAYS IN WHICH TO DUPLICATE DM6700 FILES--USING THE DEFAULT CLAUSE, AND INDIVIDUAL SPECIFICATION BY STRUCTURE.

5.3.1. DEFAULT CLAUSE

SYNTAX FOR A "DEFAULT" STATEMENT HAS BEEN INCORPORATED INTO BOTH SDL/STRUCTURE AND SDL/INITIALIZE TO ENABLE SPECIFICATION OF DEFAULT VALUES FOR ALL STRUCTURES FOR SUCH THINGS AS DEVICE AND DUPLICATED FILES. "DEFAULT DUPLICATED = ON;" WILL CAUSE ALL DM FILES, EXCEPT THOSE MENTIONED ABOVE, TO BE DUPLICATED. IT IS POSSIBLE TO OVERRIDE THE DEFAULT VALUE GIVEN TO DUPLICATED BY INDIVIDUAL STRUCTURE USING THE SYNTAX PRESENTED BELOW. IF, FOR EXAMPLE, "DEFAULT DUPLICATED = ON;" WAS SPECIFIED AT STRUCTURE TIME, THEN SPECIFYING "DEFAULT DUPLICATED = OFF;" AT INITIALIZE TIME WOULD RESET THIS OPTION TO OFF FOR ALL FILES NOT SPECIFICALLY SET TO DUPLICATED = ON USING THE SYNTAX FOR INDIVIDUAL STRUCTURES.

IF DUPLICATED IS NOT MENTIONED IN THE DEFAULT SECTION OF SDL/STRUCTURE, THEN DUPLICATED = OFF IS ASSUMED (I.E., THE DEFAULT VALUE FOR DUPLICATED IS OFF).

5.3.2. INDIVIDUAL SPECIFICATION BY STRUCTURE

THE SYNTAX FOR MODIFYING STRUCTURE INFORMATION AT

INITIALIZE TIME HAS BEEN EXPANDED TO INCLUDE DUPLICATED = ON AND DUPLICATED = OFF.

IF THE DUPLICATED SYNTAX IS USED FOR A SPECIFIC STRUCTURE, IT WILL OVERRIDE ANY VALUE SPECIFIED FOR DUPLICATED IN THE DEFAULT SECTIONS.

NEW SYNTAX HAS ALSO BEEN ADDED TO SDL/INITIALIZE TO ENABLE ATTRIBUTES SUCH AS DEVICE AND DUPLICATED TO BE GIVEN TO THE SDL AND ERRORS FILES.

5.3.3. EXAMPLES

EXAMPLE 1:
```
?RUN SYSTEM/SDL/STRUCTURE
?DATA CARD
 DATABASE = TESTDB;
 DEFAULT DUPLICATED = ON, DEVICE = DISKPACK;
?END
```

BY SETTING DUPLICATED ON IN THE DEFAULT SECTION, THE USER HAS INDICATED THAT ALL DM FILES ARE TO BE DUPLICATED. (HE HAS ALSO SPECIFIED THAT ALL FILES RESIDE ON DISK PACK, SEE SECTION 6.)

EXAMPLE 2:
```
?RUN SYSTEM/SDL/INITIALIZE
?DATA CARD
 DATABASE = TESTDB;
 SET ERRORLOG DUPLICATED=OFF;
 SET SDL DUPLICATED=ON;
 SET 16 BLOCKFACTOR = 10, DUPLICATED = OFF,
                          AREASIZE = 5;
?END
```

IN THIS EXAMPLE, THERE IS NO DEFAULT SECTION AND THE USER HAS INDICATED THAT THE ERRORS FILE AND STRUCTURE #16 SHOULD NOT BE DUPLICATED. IN ADDITION, HE HAS REDUNDANTLY SAID THAT THE SDL FILE WILL BE DUPLICATED (I.E., SETTING DUPLICATED=ON IN THE DEFAULT SECTION OF STRUCTURE TURNED DUPLICATED ON FOR SDL).

5.4. DMPRINTIT INTERFACE

USING DMPRINTIT ONE CAN PRINT BOTH OR EITHER COPY OF A DUPLICATED FILE. (SEE SECTION 9.1.1).

TO GET A PRINTIT OF BOTH COPIES, THE "OLD" SYNTAX IS USED. THAT IS NO #1 OR #2 IS ATTACHED TO THE NAME OF THE FILE. THUS, TESTDB/0042 WILL CAUSE BOTH COPIES OF STRUCUTRE 42 TO BE PRINTED, IF IT IS DUPLICATED.

TO PRINT EITHER COPY SEPARATELY, THE EXACT TITLE IS GIVEN. THUS, TESTDB/"0042#2" WOULD RESULT IN PRINTING OF COPY NUMBER 2 FOR STRUCTURE NUMBER 42 ASSUMING STRUCUTRE 42 WAS DUPLICATED.

TESTDB/ALL WILL PRINT BOTH COPIES OF ALL FILES THAT ARE DUPLICATED.

6. DISKPACK

6.1. INTRODUCTION

SYNTAX HAS BEEN ADDED TO STRUCTURE AND INTIALIZE TO ENABLE ANY OR ALL DM FILES TO RESIDE ON A NAMED FAMILY OF DISK PACKS.

THE FAMILY NAME IS, IN ALL CASES, THE DATA BASE NAME. THUS, ALL DM FILES FOR A GIVEN DATA BASE THAT ARE TO RESIDE ON DISK PACK WILL BE PLACED ON A FAMILY OF DISK PACKS WHERE THE FAMILY NAME IS THE DATA BASE NAME. ONE OF THE IMPLICATIONS IS THAT DM FILES (EXCEPT AUDIT FILES) CANNOT RESIDE ON SYSTEM RESOURCE PACK.

DM/<DATABASE-NAME>/DDL CANNOT BE PLACED ON PACK.

DM/<DATABASE-NAME>/ERRORS CANNOT BE PLACED ON PACK IF DUPLICATED HAS BEEN SET ON. THIS IS BECAUSE THE ERRORS FILE IS A SYSTEM DUPLICATED FILE (IF DUPLICATED IS SPECIFIED) AND SYSTEM DUPLICATED FILES ARE NOT SUPPORTED ON DISK PACK.

DM/<DATABASE-NAME>/AUDITARCHIVE CANNOT BE PLACED ON DISK PACK.

6.2. SDL INTERFACE

6.2.1. DEFAULT CLAUSE

ALL DM FILES CAN BE SET TO DISK PACK OR HEAD-PER-TRACK DISK USING THE NEW DEFAULT SYNTAX IN BOTH STRUCTURE AND INITIALIZE. FOR EXAMPLE, "DEFAULT DEVICE = DISKPACK;" WILL CAUSE ALL DM FILES EXCEPT THOSE NOTED ABOVE TO RESIDE ON NAMED PACK WHERE THE FAMILY NAME IS THE DATABASE NAME. IF DEVICE IS NOT SPECIFIED IN A DEFAULT CLAUSE, THEN THE DEVICE DEFAULTS TO HEAD-PER-TRACK DISK.

6.2.2. INDIVIDUAL SPECIFICATION BY STRUCTURE

EACH FILE AND/OR INDEX CARD IN STRUCUTRE AND EACH SET STATEMENT IN INITIALIZE HAS THE DEVICE = DISK AND/OR DEVICE = DISKPACK SYNTAX INCORPORATED.

A DEVICE SPECIFICATION FOR AN INDIVIDUAL STRUCTURE OVERRIDES ANY DEVICE SPECIFICATION IN THE DEFAULT SECTION.

6.2.3. EXAMPLES

```
EXAMPLE 1:  ?RUN SYSTEM/SDL/STRUCTURE
            ?DATA CARD
             DATABASE = TESTDB;
             DEFAULT DEVICE = DISKPACK;
             INDEX ACCOUNTNO (TYPE = INDEX-SEQUENTIAL,
                              DEVICE = DISK);
             FILE SAVINGS (DEVICE = DISK);
             FILE CHECKING (DEVICE = DISKPACK);
            ?END
```

SETTING DEVICE EQUAL TO DISKPACK IN THE DEFAULT SECTION WILL CAUSE ALL DM FILES EXCEPT DDL AND THE AUDITARCHIVE

TO BE PLACED ON A SET OF PACKS WITH FAMILY NAME TESTDB.

HOWEVER THE USER HAS SPECIFIED THAT THE FILE SAVINGS AND THE INDEX SEQUENTIAL STRUCTURE FOR ACCOUNTNO SHOULD RESIDE ON HEAD-PER-TRACK. THIS INCLUDES BOTH THE FINE AND COARSE TABLES. THE FILE CHECKING WAS SPECIFICALLY PLACED ON PACK.

```
EXAMPLE 2:  ?SYSTEM/SDL/INITIALIZE
            ?DATA CARD
             DATABASE = TESTDB;
             DEFAULT DEVICE = DISK;
             SET SDL DEVICE = DISKPACK, DUPLICATED = ON;
             SET ERRORLOG DEVICE = DISKPACK;
             SET 16 BLOCKFACTOR = 11, DEVICE = DISKPACK;
            ?END
```

THE DEFAULT CLAUSE HERE WILL CAUSE THE SYSTEM TO PLACE ALL DM FILES NOT PREVIOUSLY SPECIFICALLY SET TO DISK PACK ONTO HEAD-PER-TRACK. THE DEFAULT CLAUSE AT INITIALIZE TIME DOES NOT OVERRIDE ANY DEVICE SPECIFICATIONS AT STRUCTURE TIME THAT WERE SET AS INDIVIDUAL STRUCTURES. THUS, THE FILE "CHECKING" WILL STILL BE SET TO DISK PACK EVEN THOUGH THE DEFAULT IN INITIALIZE WAS DISK.

THE USER THEN SELECTIVELY PLACES THE SDL FILE, THE ERRORS FILE AND STRUCTURE 16 BACK ONTO DISK PACK. THE SDL FILE WILL ALSO BE DUPLICATED.

WERE HE TO TRY AND SET DUPLICATED = ON FOR THE ERRORS FILE, (I.E., SET ERRORLOG DEVICE = DISKPACK, DUPLICATED = ON;), A WARNING WOULD BE PRINTED TO THE EFFECT THAT THIS FILE CANNOT RESIDE ON DISK PACK, IF DUPLICATED, AND DEVICE WOULD AUTOMATICALLY REVERT BACK TO DISK.

6.4. DMPRINTIT INTERFACE

DMPRINTIT HAS BEEN MODIFIED TO PRINT DM FILES THAT RESIDE ON DISK PACK. (SEE SECTION 9.2.2.)

7. I-O ERROR HANDLING

THIS SECTION DESCRIBES THE WAY IN WHICH DM6700 HANDLES I/O ERROR INDICATIONS IT RECEIVES FROM THE OPERATING SYSTEM, THE FORM IN WHICH SUCH ERRORS ARE TRANSMITTED BACK TO THE USER PROGRAM, AND THE MEANING OF THE NEW I/O STATUSES.

UNTIL THE 11.4 RELEASE AN I/O ERROR RESULTED IN A FAULT TERMINATION IN DM6700. I/O ERRORS ARE NOW CAPTURED AND LOGGED, AND NOTIFICATION OF SUCH IS RETURNED TO THE USER PROGRAM VIA NEW STATUSES.

7.1. ERROR FILE

7.1.1. LAYOUT OF ERROR FILE

A NEW FILE DM/<DATABASE-NAME>/ERRORS IS NOW CREATED. ITS FUNCTION IS TO KEEP TRACK BY ROW, OF WHICH COPIES OF WHICH STRUCTURES HAVE HAD READ AND/OR WRITE ERRORS.

EACH RECORD OF THE ERRORS FILE IS 45 WORDS LONG AND THE FILE IS BLOCKED TWO.

RECORD 0 CONTAINS I/O ERRORS FOR THE SDL DIRECTORY WHILE RECORD 1 CONTAINS I/O ERRORS FOR THE SECOND COPY OF SDL IF IT IS DUPLICATED (I.E., DM/<DATABASE-NAME>/"SDL#2").

RECORD 2 CONTAINS I/O ERRORS FOR STRUCTURE #1 WHILE RECORD 3 CONTAINS I/O ERRORS FOR THE SECOND COPY OF STRUCTURE #1 IF DUPLICATED.

IN GENERAL, RECORD 2XN CONTAINS A LOG OF I/O ERROR INFORMATION FOR STRUCTURE N WHILE RECORD 2XN+1 CONTAINS I/O ERROR INFORMATION FOR THE SECOND COPY OF STRUCTURE N.

7.1.2. RECORD LAYOUT - ERROR FILE

THE FORMAT OF EACH 45 WORD RECORD IS AS FOLLOWS:

WORD 0 — 47:24 - NUMBER OF WRITE ERRORS FOR THIS STRUCTURE AND COPY. 23:24 - NUMBER OF READ ERRORS FOR THIS STRUCTURE AND COPY.

WORDS 1-21 — THE ROWS WHICH HAVE RECEIVED WRITE ERRORS- ONE BIT PER ROW STARTING WITH WORD 1 BIT 47 (ROW 0) AND ENDING WITH WORD 21 BIT 0 (ROW 1008). A BIT ON (1) INDICATES THAT A RECORD IN THAT ROW HAS HAD A WRITE ERROR AND THAT ALL RECORDS IN THAT ROW ARE NOW LOCKED.

WORDS 22-42 — THE ROWS WHICH HAVE RECEIVED READ ERRORS - ONE BIT PER ROW STARTING WITH WORD 22 BIT 47 (ROW 0) AND ENDING WITH WORD 42 BIT 0 (ROW 1008). A BIT ON (1) INDICATES THAT A RECORD IN THAT ROW HAS HAD A READ ERROR BUT THE ROW IS NOT NECESSARILY LOCKED.

WORD 43 — JULIAN DATE OF THE FIRST ROW IN ERROR.

WORD 44 — TIME OF THE FIRST ROW IN ERROR. THE TIME AND DATE ARE USED TO DETERMINE WHICH AUDIT FILE TO START WITH WHEN RECONSTRUCTING.

7.1.3. DMPRINTIT INTERFACE

A FORMATTED LISTING OF THE ERRORS FILE MAY BE OBTAINED VIA DMPRINTIT USING A <DATABASE-NAME>/ERRORS CARD. THE FORMAT OF THIS PRINTOUT IS DESCRIBED IN SECTION 9.1.2.

7.2. NEW STATUSES

STATUS	MEANING
200	ROW LOCKED OUT - WRITE ERROR
201	READ ERROR - UNIT NOT READY
202	READ ERROR - READ PARITY
204	READ ERROR - DESCRIPTOR IN ERROR

STATUS 200 MEANS THAT ALL RECORDS IN A ROW ARE LOCKED OUT

TO EVERYONE. THERE WILL BE NO READS ATTEMPTED TO THIS ROW. ANY DATA MANAGEMENT OPERATION WHICH TRIES TO ACCESS A RECORD FROM THIS ROW WILL SEE THIS STATUS. THE REASON THE ROW IS LOCKED IS THAT THE DATA ON DISK OR DISK PACK IS OUT OF DATE. THE FOLLOWING EXAMPLE WILL HELP CLARIFY THE NEED FOR ROW LOCKOUT.

SUPPOSE A USER PROGRAM DOES A MODIFY, FOLLOWED BY MOVES, FOLLOWED BY A STORE, AND THAT HE RECEIVES A STATUS OF 0 FOR BOTH DM OPERATIONS. FROM HIS POINT OF VIEW, HE HAS MODIFIED SOME DATA. DM6700 MUST ALSO TREAT THE DATA AS HAVING BEEN UPDATED, BUT BECAUSE OF ITS MULTIPLE BUFFERS, THE CHANGE TO THAT DATA WILL NOT BE REFLECTED ON DISK OR PACK UNTIL THE DATA MANAGEMENT SYSTEM IS FORCED TO GIVE UP THAT BUFFER IN ORDER TO BRING IN NEW RESOURCES. WHEN DM6700 ATTEMPTS TO WRITE THIS UPDATED RECORD TO DISK AND RECEIVES A WRITE ERROR, THE ROW IS LOCKED OUT. AT THIS POINT DM6700 CALCULATES THE ROW THIS RECORD WAS IN, AND ENTERS THE FACT THAT IT ENCOUNTERED A WRITE ERROR ON THIS ROW INTO THE ERROR FILE. SINCE THE UPDATED MEMBER IS NO LONGER IN A DM6700 BUFFER AND BECAUSE DM6700 WAS UNSUCCESSFUL IN WRITING IT BACK TO DISK, ANY FUTURE ATTEMPTS TO ACCESS ANY RECORD FROM THIS ROW WILL RESULT IN A STATUS OF 200 - ROW LOCKED OUT. (THE MODIFIED DATA WILL, OF COURSE, BE WRITTEN TO THE AUDIT TRIAL, IF AUDITING.)

IF A WRITE ERROR OCCURS ON A ROW OF A DUPLICATED FILE, THEN IF THE CORRESPONDING ROW IS NOT IN ERROR IN THE OTHER COPY, A BAD STATUS WILL NOT BE PASSED BACK TO THE USER PROGRAM WHEN HE TRIES TO ACCESS A RECORD IN THAT ROW. IN EITHER CASE, HOWEVER, THE FACT THAT A WRITE ERROR HAS OCCURRED IS LOGGED IN THE ERROR FILE. STATUSES 201, 202 AND 204 ARE READ ERRORS. ALL READ ERRORS ARE ALSO LOGGED IN THE ERROR FILE BUT A READ ERROR DOES NOT CAUSE ROW LOCKOUT. THE DATA ON DISK IS NOT OUT OF DATE, IT JUST CANNOT BE READ.

WITH DUPLICATED FILES. A READ ERROR FROM A ROW IN ONE COPY WILL NOT BE RETURNED TO THE USER PROGRAM AS SUCH UNLESS THE CORRESPONDING ROW IN THE OTHER COPY HAS EITHER A READ OR WRITE ERROR. IF THE CORRESPONDING ROW HAS A WRITE ERROR, THEN A STATUS OF 200 WILL BE RETURNED. IF THE CORRESPONDING ROW HAD A READ ERROR, THE APPROPRIATE READ ERROR WOULD BE RETURNED.

THE SYSTEM MAINTENANCE LOG CAN BE USED IN CONJUNCTION WITH THE ERROR FILE TO DETERMINE SUCH THINGS AS PHYSICAL DISK ADDRESSES, DATES AND TIMES OF ERRORS.

FOR PURPOSES OF RECONSTRUCTION AND RESTORATION, IT IS IMPORTANT TO BE COGNIZANT OF ENTRIES IN THE ERROR FILE. THIS MAY BE DONE VIA USER PROGRAMS GIVING SOME SORT OF NOTIFICATION UPON RECEIVING SUCH ERROR STATUSES OR BY PERIODIC FORMATTED LISTINGS OF THE ERROR FILE VIA DMPRINTIT. SECTION 8 GIVES A DETAILED EXPLANATION OF HOW AND WHEN THE ERROR FILE IS USED IN CONJUNCTION WITH RECONSTRUCTION AND RESTORATION.

8. RECONSTRUCTION AND RESTORATION

8.1. INTRODUCTION

RECONSTRUCTION AND RESTORATION PERFORM TWO FUNCTIONS.

THE FIRST IS TO BRING DATA IN A ROW LOCKED OUT BECAUSE OF WRITE ERRORS UP TO DATE AND TO PLACE THIS NEW, UPDATED ROW IN ANOTHER PHYSICAL LOCATION.

THE SECOND FUNCTION IS TO MOVE ROWS WHICH HAVE HAD READ ERRORS TO OTHER PHYSICAL LOCATIONS.

RECONSTRUCTION AND RESTORATION PERFORM BOTH FUNCTIONS. THE DIFFERENCES BETWEEN THEM ARE A MATTER OF TIME AND SPACE.

8.2. RESTORATION

RESTORATION REQUIRES A GOOD COPY OF THE ROW FROM THE OTHER COPY OF THE DUPLICATED DATA BASE FILE. IT COPIES THE GOOD ROW TO A GOOD ROW OF ANOTHER FILE (EITHER ONE INTERNAL TO RESTORATION OR A USER SPECIFIED BACKUP) AND THEN EXCHANGES THE NEWLY CREATED GOOD ROW WITH THE BAD ROW. AT THIS POINT, THE PARTICULAR ROW IS VALID AND UP TO DATE ON BOTH COPIES OF THE DUPLICATED DATA MANAGEMENT FILE.

8.3. RECONSTRUCTION

RECONSTRUCTION DOES NOT HAVE AN UP TO DATE COPY OF THE BAD ROW AS RESTORATION DOES. THUS ITS FIRST TASK WHEN RECONSTRUCTING A ROW LOCKED OUT BECAUSE OF WRITE ERRORS, IS TO BRING THE CORRESPONDING ROW OF A PREVIOUS BACKUP COPY OF THE FILE UP TO DATE USING THE AUDIT TRAIL. ONCE THE BACKUP ROW IS UP TO DATE, RECONSTRUCTION EXCHANGES THIS ROW WITH THE BAD ROW JUST AS RESTORATION DOES.

SELECTION OF THE STARTING AUDIT FILE IS CRITICAL. THE DATE AND TIME THAT THE STARTING AUDIT FILE WAS CREATED MUST BE BEFORE THE DATE AND TIME OF THE FIRST ERROR FOR A GIVEN STRUCTURE. IF THE STARTING AUDIT FILE WAS CREATED AFTER THE FIRST ERROR, MODIFICATIONS TO DATA IN ANY OF THE BAD ROWS MAY HAVE BEEN MADE IN THE INTERIM AND WOULD THUS BE LOST -- AFTER IMAGES FOR THESE CHANGES WOULD BE IN AN EARLIER AUDIT FILE.

CONSIDER THE FOLLOWING SEQUENCE OF EVENTS:

TUESDAY - DB/AUDIT0001 CREATED.

RECORD IN ROW 56 FOR STRUCTURE 12 MODIFIED AND STORED SUCCESSFULLY. A WRITE ERROR OCCURS ATTEMPTING TO WRITE THE PREVIOUSLY MODIFIED RECORD. (I.E., COPY OF RECORD ON DISK IS NOW OUT OF DATE).

DB/AUDIT0001 FULL.
DB/AUDIT0002 CREATED.

ASSUMING THE ABOVE WRITE ERROR WAS THE FIRST ERROR FOR STRUCTURE 12, THE DATE AND TIME WILL HAVE BEEN ENTERED INTO THE ERROR FILE. IF DB/AUDIT0002 WERE CHOSEN AS THE STARTING AUDIT FILE, THE RECORD IN ROW 56 WOULD BE OUT OF DATE AFTER RECONSTRUCTION FINISHED. THUS, UNLESS OVERRIDDEN BY A USER SPECIFIED STARTING AUDIT SERIAL NUMBER, RECONSTRUCTION WILL SEARCH THE AUDIT ARCHIVE FOR AN AUDIT FILE THAT WAS CREATED BEFORE THE DATE AND TIME OF THE FIRST ERROR THAT APPEARS IN THE ERROR FILE.

8.4. DMROWRECOVERY - USER INTERFACE

8.4.1. INTRODUCTION

THE USER INTERFACE TO RECONSTRUCTION AND RESTORATION IS THROUGH THE PROGRAM SYSTEM/DMROWRECOVERY.

THE BASIC INTENT OF DMROWRECOVERY IS THAT THE DATA BASE ADMINISTRATOR BE ABLE TO TELL DM6700 TO "REBUILD" A STRUCTURE FOR A DATA BASE, BUT THAT DM6700 DETERMINE VIA THE ERROR FILE WHICH ROWS SHOULD BE RESTORED AND WHICH NEED TO BE RECONSTRUCTED TO COMPLETELY REBUILD ALL BAD ROWS FOR THAT STRUCTURE OF THE DATA BASE. THUS IT IS NOT POSSIBLE TO REBUILD ONLY SOME ROWS OR SPECIFIC ROWS OF A STRUCTURE WITHOUT REBUILDING ALL ITS BAD ROWS.

8.4.2 REMOVE

THE REMOVE COMMAND GIVES THE USER THE ABILITY TO REMOVE FROM THE ERROR FILE THE INDICATION THAT A ROW OR ROWS OF A DATA MANAGEMENT FILE HAD INCURRED A READ ERROR. THUS ANY SUBSEQUENT RECONSTRUCTION AND/OR RESTORATION (I.E., RUN OF DMROWRECOVERY) WILL HAVE NO EFFECT UPON THIS ROW.

THIS COMMAND WAS INCORPORATED FOR THE CIRCUMSTANCE IN WHICH READ ERRORS HAD BEEN DETECTED BECAUSE OF A UNIT NOT READY. SUCH AN ERROR WOULD BE RETURNED TO THE APPLICATION PROGRAM AS A STATUS OF 201. HOWEVER, IT IS

REASONABLE FOR A DATA BASE ADMINISTRATOR TO RECOGNIZE THAT THESE READ ERRORS WERE ONLY A RESULT OF A UNIT NOT READY AND SINCE THE DATA IS NOT BUT OF DATE AND CAN NOW BE READ (UNIT MADE READY) HE SHOULD RESET THOSE READ ERRORS.

SINCE THE REMOVE ONLY PERTAINS TO READ ERRORS, NO INTEGRITY CAN BE LOST SINCE THE WORST THAT CAN HAPPEN IS TO TURN OFF THE INDICATION THAT A ROW WHICH HAD A READ PARITY WAS BAD. THE ROW WILL SIMPLY NOT BE RESTORED OR RECONSTRUCTED UNLESS IT IS TURNED BACK ON EITHER BY THE DATA BASE ADMINISTRATOR OR VIA ANOTHER READ PARITY RECEIVED FROM DM6700.

THERE IS NO WAY TO REMOVE A WRITE ERROR FROM THE ERROR FILE.

8.4.3. INSERT

THE INSERT COMMAND DOES THE OPPOSITE OF THE REMOVE COMMAND. IT ENABLES THE DATA BASE ADMINISTRATOR TO SET READ ERROR BITS FOR ANY ROW OR ROWS.

THIS COMMAND IS NEEDED IN A SITUATION WHERE, SAY, A STORAGE UNIT WERE TO FAIL. GIVEN ENOUGH ACTIVITY AND TIME, DM6700 WOULD ATTEMPT TO ACCESS OR WRITE ONTO THIS STORAGE UNIT AND WOULD APPROPRIATELY RECORD THE FACT THAT IT WAS UNABLE TO DO SO. THUS GIVEN SUCH A SITUATION, THE DATA BASE ADMINISTRATOR NEEDS A WAY TO TELL DM6700 THAT FOR VARIOUS STRUCTURES, CERTAIN OF ITS ROWS ARE IN ERROR.

8.4.4. REBUILD

THIS COMMAND SIGNALS DM6700 THAT, FOR A GIVEN DATA BASE AND STRUCTURE, THERE ARE ROWS IN ERROR AND OF THE DESIRE TO REBUILD ALL BAD ROWS. DM6700 WILL DECIDE WHICH ROWS (IF ANY) NEED TO BE REBUILT.

8.4.5. INSTALLATION ALLOCATED DISK AND DISK PACK

ADDITIONAL CONSIDERATION MUST BE GIVEN TO FILES WHICH RESIDE ON INSTALLATION ALLOCATED DISK OR DISK PACK.

IN PARTICULAR, EACH BAD ROW MUST BE MAPPED INTO A ROW OF A BACKUP FILE. THIS DIFFERS FROM NON-IAD DM FILES IN THAT DMROWRECOVERY WILL ALLOCATE ITS OWN BACKUP ROW FOR RESTORATION PURPOSES IF A BACKUP IS NOT SPECIFIED BY THE USER. WITH IAD FILES, HOWEVER, THERE MUST BE A ONE FOR ONE CORRESPONDENCE BETWEEN BAD ROWS AND GOOD BACKUP ROWS. THE SYNTAX FOR ACCOMPLISHING THIS ONE TO ONE MAPPING IS GIVEN IN 8.4.6.

AN ADVANTAGE WHICH RESULTS FROM HAVING TO PROVIDE A ONE TO ONE MAPPING OF GOOD TO BAD ROWS IS THAT IAD SPECIFIED BACKUP FILES NEED ONLY HAVE AS MANY ROWS AS THERE ARE BAD ROWS IN THE DM FILE.

EXAMPLE:

CONSIDER THE FOLLOWING SITUATION. STORAGE UNIT 32 CRASHES DURING THE NIGHT. ALL OF DM/TESTDB/0045 AND THREE ROWS OF DM/TESTDB/0027 WERE ON THIS SU AS IAD FILES. THE FOLLOWING IS THE IAD MAPPING FOR BOTH FILES.

DM/TESTDB/0027

ROW	SU	ADDRESS
0	32	100
1	32	105
2	32	110
3	34	100
4	34	105

DM/TESTDB/0045

ROW	SU	ADDRESS
0	32	3010
1	32	3110

SUPPOSE ALSO THAT ROW 4 OF DM/TESTDB/0027 WAS LOCKED OUT FROM A PREVIOUS WRITE ERROR. THIS PARTICULAR INSTALLATION TAKES BACKUP DUMPS BY SU SO THAT RECONSTRUCTING A SU INVOLVES RECONSTRUCTING FROM A BACKUP COPY OF SU32 ON RESERVE SU48. THE ERROR IN ROW 4 PRESENTS AN ADDITIONAL PROBLEM, HOWEVER, SINCE THE

INSTALLATION WANTS TO KEEP THAT ROW ON SU34 AND NOT ON THE BACKUP SU48. THUS A SIX ROW IAD FILE DM/TESTDB/IADBACKUP MUST BE CREATED.

DM/TESTDB/IADBACKUP

ROW	SU	ADDRESS
0	48	100
1	48	105
2	48	110
3	48	3010
4	48	3110
5	34	200

THUS TO REBUILD THE BAD ROWS FROM BOTH THE FILES, THE FOLLOWING CARDS WOULD BE USED WITH DMROWRECOVERY:

```
?RUN SYSTEM/DMROWRECOVERY
?DATA CARD
DATABASE = TESTDB
STRUCTURE 45
     INSERT ROWS 0,1
     REBUILD ALL ROWS
          BACKUP IS (DM/TESTDB/IADBACKUP)
               IAD CORRESPONDENCE IS 3=0, 4=1
          AUDIT = 1;
STRUCTURE 27
     INSERT ROWS 0=2
     REBUILD
          BACKUP IS (DM/TESTDB/IADBACKUP)
               IAD CORRESPONDENCE IS 0=0,1=1,2=2,5=4
               ;
?END
```

FOR STRUCTURE 45, AN INDICATION THAT ROWS 0 AND 1 ARE BAD MUST FIRST BE INSERTED INTO THE ERROR FILE SINCE THIS STRUCTURE HAD NO ERRORS PRIOR TO THE CRASHING OF SU32. SINCE THERE WERE NO ERRORS, IT IS NOT POSSIBLE FOR DMROWRECOVERY TO FIND IN THE AUDIT ARCHIVE THE PROPER

AUDIT FILE WITH WHICH TO START. THUS AN AUDIT SERIAL NUMBER MUST BE SPECIFIED.

FOR STRUCTURE 27 NOTE THAT NO AUDIT SERIAL NUMBER IS NECESSARY, SINCE ROW 4 WAS IN ERROR PRIOR TO SU32 CRASHING. THUS THE DATE, AND TIME AT WHICH THE ERROR OCCURRED IS RECORDED IN THE ERROR FILE AND DMROWRECOVERY CAN DETERMINE WHAT IT IS. HAD ROW 4 HAD A READ INSTEAD OF WRITE ERROR AND THE DATA BASE ADMINISTRATOR WANTED ONLY TO REBUILD THE SU, THEN THAT ROW COULD HAVE BEEN REMOVED FROM THE ERROR FILE WITHOUT COMPROMISING THE INTEGRITY OF THE DATABASE. (NOTE: THE RECORD THAT ENCOUNTERED THE READ ERROR WILL STILL BE IN ERROR.)

8.4.6. SYNTAX

THE SYNTAX FOR DMROWRECOVERY IS:

```
<DMROWRECOVERY REQUEST> ::= <DATABASE NAME SPEC>
                              <STRUCTURE ROW RECOVERY LIST>
<DATABASE NAME SPEC> ::= <DATABASE NAME> /
        <DATABASE DESIGNATOR><EQUALS PART><DATABASE NAME>
<DATABASE DESIGNATOR> ::= DATABASE / DATA-BASE
<EQUALS PART> ::= <EMPTY> / =
<STRUCTURE ROW RECOVERY LIST> ::=
                          <STRUCTURE ROW RECOVERY REQUEST>/
                            <STRUCTURE ROW RECOVERY LIST>;
                           <STRUCTURE ROW RECOVERY REQUEST>
<STRUCTURE ROW RECOVERY REQUEST> ::= <STRUCTURE SPEC>
                                       <ERROR FILE STATEMENT>/
                          <STRUCTURE SPEC><REBUILD STATEMENT>/
                 <STRUCTURE SPEC><ERROR FILE STATEMENT>
                 <REBUILD STATEMENT>
<STRUCTURE SPEC> ::= SDL / <STRUCTURE NUMBER> / STRUCTURE
                          <STRUCTURE NUMBER>
<STRUCTURE NUMBER> ::= <UNSIGNED INTEGER>
<ERROR FILE STATEMENT> ::= <REMOVE STATEMENT> /
                              <INSERT STATEMENT>
```

```
<REMOVE STATEMENT> ::=
    REMOVE <ERROR FILE PART> BOTH
        <COPIES PART><ALL SPEC> /
    REMOVE <ERROR FILE PART> BOTH
        <COPIES PART><ROW SPECS>
        <AND PART><FILE SPECS PART> /
    REMOVE <ERROR FILE PART><DUPLICATE SPECS>
        <ROW SPECS PART><AND PART><FILE SPECS PART>
<ERROR FILE PART> ::= <EMPTY> / FROM ERROR FILE/
                      INTO ERROR FILE
<COPIES PART> ::= <EMPTY> / COPIES
<ALL SPEC> ::= ALL / ALL ROWS
<ROW SPECS> ::= <ROW PART><ROW NUMBER SPECS>
<ROW PART> ::= <EMPTY> / ROW / ROWS
<ROW NUMBER SPECS> ::= <ROW NUMBER> /
     <ROW NUMBER><THRU SPEC> /
     <ROW NUMBER SPECS><COMMA PART><ROW NUMBER> /
     <ROW NUMBER SPECS><COMMA PART>
     <ROW NUMBER><THRU SPEC>
<THRU SPEC> ::= THRU <ROW NUMBER> / - <ROW NUMBER>
<COMMA PART> ::= <EMPTY> / ,
<AND PART> ::= <EMPTY> / AND
<FILE SPECS PART> ::= <EMPTY>
     <DUPLICATE SPECS><ROW SPECS PART><AND PART> /
     <FILE SPECS PART><FILE SPECS PART>
<DUPLICATE SPECS> ::= <EMPTY> /
                      <COPY PART> ONE /
                      <COPY PART> TWO
<COPY PART> ::= <EMPTY> / COPY
<ROW SPECS PART> ::= <ALL SPEC> / <ROW SPECS>
<INSERT STATEMENT> ::=
     INSERT <ERROR FILE PART> BOTH
          <COPIES PART><ROW SPECS>
          <AND PART><FILE SPECS PART> /
     INSERT <ERROR FILE PART><DUPLICATE SPECS>
          <ROW SPECS><AND PART><FILE SPECS PART>
```

```
<REBUILD STATEMENT> ::= REBUILD /
       REBUILD <ALL PART> BACKUP <FILE PART><TITLE SPECS>
<ALL PART> ::= <EMPTY> / <ALL SPEC>
<FILE PART> ::= <EMPTY> / FILE
<TITLE SPECS> ::= <TITLE><IAD PART><AUDIT PART>/
     <DUPLICATE TITLE SPECS><TITLE><IAD PART><AUDIT PART>
<TITLE> ::= (<FILE TITLE IN DISPLAY FORM>)
<IAD PART> ::= <EMPTY> /
                IAD <CORRESPONDENCE PART>
                <IAD NUMBER SPECS>
<CORRESPONDENCE PART> ::= <EMPTY> / CORRESPONDENCE IS
<IAD NUMBER SPECS> ::=
     <IAD ROW NUMBER> = <FILE ROW NUMBER> /
     <IAD NUMBER SPECS><COMMA PART><IAD ROW NUMBER> =
          <FILE ROW NUMBER>
<AUDIT PART> ::= <EMPTY>/
                          AUDIT <SERIAL PART><NUMBER PART>
                                = <AUDIT FILE NUMBER>
<SERIAL PART> ::= <EMPTY> / SERIAL
<NUMBER PART> ::= <EMPTY> / NUMBER
<DUPLICATE TITLE SPECS> ::=
     <FOR PART><COPY PART> ONE <IS PART><TITLE>
     <IAD PART><COMMA PART><FOR PART><COPY PART>
     TWO <IS PART>
<FOR PART> ::= <EMPTY> / FOR
<IS PART> ::=  <EMPTY> / IS
```

8.4.7. EXAMPLES

EXAMPLE 1

```
DATA-BASE = T0047
    SDL
         REMOVE FROM ERROR FILE
              COPY ONE ROWS 2, 5, 7 THRU 11
              COPY TWO ALL ROWS
         AND REBUILD
```

```
                    BACKUP FILE FOR COPY ONE IS
                                (DM/T0047/"SDL#1"),
                            FOR COPY TWO IS
                                (DM/T0047/"SDL#2")
          ;

          STRUCTURE 2
               INSERT INTO ERROR FILE
                    COPY ONE ROW 13 AND COPY TWO ROWS 9-18
               REBUILD ALL ROWS
                    BACKUP FILE
                         FOR COPY ONE IS (DM/T0047/"0002#1")
                              IAD CORRESPONDENCE IS 1=5,2=6,6=0
                         FOR COPY TWO IS (DM/T0047/"0002#2")
                              IAD CORRESPONDENCE IS 1=1,3=3,4=5
                    AUDIT SERIAL NUMBER = 2;
```

EXAMPLE 2

```
T0047
    2
          INSERT
               ONE 13 TWO 9-18
          REBUILD ALL
          BACKUP
                    ONE (DM/T0047/"0002#1")
                         IAD 1 = 5, 2 = 6, 6 = 0
                    TWO (DM/T0047/"0002#2")
                         IAD 1=1, 3=3, 4=5
               AUDIT = 2;
```

8.4.8. PRAGMATICS

EACH <STRUCTURE ROW RECOVERY REQUEST> IS A SEPARATE REQUEST TO THE DATA BASE MONITOR. THESE REQUESTS ARE HANDLED THE SAME AS NORMAL REQUESTS (E.G., FIND, MODIFY, ETC.) FROM USER COBOL PROGRAMS. ONE OF THE CONSEQUENCES

IS THAT THE AUDIT TRAIL MUST BE SCANNED ONCE FOR EACH STRUCTURE THAT NEEDS TO BE RECONSTRUCTED.

ONCE DMROWRECOVERY HAS PASSED THE DATA BASE MONITOR ALL ITS REQUESTS, IT GOES TO EOJ WITHOUT WAITING FOR A REPLY FROM THE MONITOR.

8.4.9. DOLLAR OPTIONS

THE INPUT TO SYSTEM/DMROWRECOVERY IS A CARD FILE LABELED "CARD".

THE PROGRAM HAS DOLLAR OPTIONS AS FOLLOWS:

DEBUG DUMPS SCANNER ARRAY AND, IF THE STATEMENT IS ERROR FREE, THE CONTENTS OF THE MESSAGE ARRAY.

LIST IS OFF BY DEFAULT AND PRODUCES A LISTING OF THE INPUT CARDS IF TURNED ON.

SINGLE IF SET, IT SINGLE SPACES THE LISTING.

SYNTAX IF SET, THEN THE REQUESTS ARE NOT PASSED ON TO SYSTEM/DM6700.

THESE OPTIONS MAY BE SET, RESET AND POPPED. THE RUN MUST BE COMPLETELY ERROR FREE BEFORE ANY REQUESTS ARE PASSED ON TO SYSTEM/DM6700.

9. DMPRINTIT

9.1. DUPLICATED FILES

IN THE CASE OF DM DUPLICATED FILES, DMPRINTIT WILL PRINT BOTH COPIES OF THE FILE, WHETHER THE OPTION "ALL" IS SET (I.E., <DB NAME>/ALL) OR WHETHER A PARTICULAR STRUCTURE IS REQUESTED (I.E., <DB NAME>/0004). A SINGLE COPY WILL BE PRINTED IN RESPONSE TO THE FOLLOWING TYPE OF INPUT: <DB NAME>/"0004#2" OR <DB NAME>/"0004#1".

INDICATION THAT DUPLICATED IS ON OR OFF IS PRINTED BOTH IN THE HEADING OF EACH FORMATTED STRUCTURE AND IN THE SDL

REPORT FOR THOSE STRUCTURES SUBJECT TO DUPLICATION (LINKS, LISTS AND RANDOM INDEX ARE NOT SUBJECT TO DUPLICATION).

9.1.2. ERRORS FILE

A FORMATTED CHART OF THE ERRORS FILE (DM/<DB NAME>/ERRORS) MAY BE OBTAINED WITH THE FOLLOWING SYNTAX: <DB NAME>/ERRORS. IF THERE ARE NO WRITE OR READ ERRORS PERTAINING TO THE NAMED DATA BASE, A MESSAGE WILL BE PRINTED OUT TO THAT EFFECT.

IF ERROR INFORMATION EXISTS, THEN THREE KINDS OF TABLES WILL BE PRINTED:

1. FOR EACH STRUCTURE WHICH CONTAINS ERRORS (BY COPY, IF DUPLICATED) THE FOLLOWING INFORMATION IS PRINTED: STRUCTURE NUMBER OR "SDL", COPY NUMBER (IF PERTINENT), DATE AND TIME OF THE FIRST ERROR, THE TYPE (WHETHER WRITE OR READ) THE NUMBER OF ERRORS, AND THE NUMBERS OF THE ROWS(AREAS) WHICH ARE IN ERROR.

2. RECONSTRUCTION AND RESTORATION INFORMATION FOLLOWS EACH STRUCTURE WITH ERRORS. THIS INFORMATION DISTINGUISHES THOSE ROWS WHICH REQUIRE RECONSTRUCTION FROM THOSE WHICH REQUIRE RESTORATION.

3. A FINAL SUMMARY STATEMENT IS MADE WHICH NAMES THE AUDIT FILE WHICH PRECEDES THE DATE AND TIME OF THE FIRST ERROR.

9.1.3. BADROW CHECK

WITH THE IMPLEMENTATION OF THE ERRORS FILE, DMPRINTIT WILL NOW CHECK (IN FORMATTED MODE ONLY) WHETHER OR NOT A RECORD WAS IN A BAD ROW (AREA). THIS PREVENTS THE PRINTING OF GARBAGE INFORMATION. INSTEAD, THE MESSAGE

"RECORD IN BAD ROW" IS PRINTED.

9.1.4 AUDITARCHIVE

A FORMATTED CHART OF THE AUDITARCHIVE (DM/<DB NAME>/AUDITARCHIVE) MAY BE OBTAINED WITH THE FOLLOWING SYNTAX: <DB NAME>/AUDITARCHIVE. THE FILE IS READ IN REVERSE ORDER AND PRINTS THE FOLLOWING INFORMATION: RECORD NUMBER, DATE, TIME, AUDIT SERIAL NUMBER, AND PACKNAME ("PACK" IMPLIES SYSTEM RESOURCE DISKPACK).

9.1.5. AUDIT REPORT IN SDL

FORMATTED SDL NOW INCLUDES AUDIT INFORMATION ABOUT THE DATA BASE. IF AUDIT HAS BEEN TURNED ON, DMPRINTIT FORMATS THE FOLLOWING RECORD: NUMBER OF AREAS IN AUDIT FILE, WHETHER BEFOREIMAGES BIT IS ON OR OFF, NUMBER OF CONTROLCYCLES, WHETHER DATA BASE-IN-USE BIT IS TRUE OR FALSE, THE AUDIT DEVICE, PACKNAME (IF THE DEVICE IS DISK PACK), AND THE AUDIT SERIAL NUMBER.

9.2. OTHER CHANGES

9.2.1. ALPHA-HEX MODES

THIS FEATURE PROVIDES FOR THE PRINTING OF DATA FILES IN THE ALPHANUMERIC MODE. THE FOLLOWING MODE CARDS ARE NOW POSSIBLE: ALPHA, NO ALPHA, ALPHA1, NO ALPHA1, HEX, NO HEX. BY DEFAULT, DMPRINTIT WILL WORK AS BEFORE (I.E., NO ALPHANUMERICS, ONLY HEX).

ALPHA CAUSES THE ALPHANUMERIC EQUIVALENT OF THE DATA TO BE PRINTED ON THE RIGHT SIDE OF THE PAGE. ALPHA1 CAUSES THE ALPHANUMERICS TO BE PRINTED BENEATH EACH HEX LINE. ALPHA AND ALPHA1 CAUSE BOTH THE ABOVE TO OCCUR. NO HEX ELIMINATES THE HEX LINE AND PRINTS ONLY THE ALPHANUMERICS. HEX RETURNS TO THE NORMAL MODE OF HEX OUTPUT ONLY.

9.2.2. DISKPACK

CHANGES ENABLING DMPRINTIT TO HANDLE FILES ON DISK PACK HAVE BEEN MADE. THE SDL REPORT NOW INCLUDES THE DEVICE AND THE PACKNAME (IF THE FILE IS ON DISK PACK).

FOR DMPRINTIT TO ACCESS FILES STORED ON NAMED PACK WHERE THE PACK NAME IS NOT EQUAL TO THE DATA BASE NAME (FOR EXAMPLE, AUDIT FILES). THE FOLLOWING "MODE" CARDS ARE USED: "USERPACK <PACKNAME>" AND "NO USERPACK" (TO RESET THE OPTION).

9.2.3. ANYFILENAME

THE CAPACITY TO PRODUCE A HEX DUMP OF ANY FILE (WITH AT LEAST A TWO-LEVEL TITLE) HAS BEEN REIMPLEMENTED. THIS IS PARTICULARLY USEFUL IN PRINTING THE CONTENTS OF THE AUDIT FILES.

9.2.4. DEAD LIST ELEMENTS

DEAD LIST ELEMENTS ARE NO LONGER PRINTED.

9.2.5. CARDD TO CARD

THE CARD READER FILE FORMERLLY KNOWN AS CARDD HAS BEEN RENAMED CARD FOR CONSISTENCY WITH OTHER PORTIONS OF THE DM SYSTEM.

9.2.6. "DISK" SYNTAX DEIMPLEMENTATION

THE NOISE WORD "DISK" AS INPUT HAS BEEN DEIMPLEMENTED (I.E., DISK <DB NAME>/0027 IS NO LONGER VALID).

9.2.7. IMPROPER DATA ERROR

AN ADDITIONAL ERROR MESSAGE WILL CATCH SOME OF THE POSSIBLE SYNTAX ERRORS OF THE USER (FOR EXAMPLE, FILENAME TOO LONG, USE OF DEIMPLEMENTED "DISK" SYNTAX, MISSING SLASH).

9.2.8. SDL REPORT

SEVERAL ADDITIONS AND MODIFICATIONS HAVE BEEN MADE TO THE SDL REPORT.

1. IF THE POPULATION OF A SET WAS * (I.E., VARIABLE), THEN MAXIMUM POPULATION WILL PRINT AS "VARIABLE" INSTEAD OF "NOT SPECIFIED".

2. "RANDOM INDEX" FOLLOWED BY "PRESENT" OR "NONE" REPLACES "PRIMF INDEX" WITH "RANDOM" OR "NONE".

3. DEVICE FIELDS ARE NOW INDICATED BY THE WORD MNEMONIC (I.E., DISK OR DISKPACK VERSUS 1 OR 17) RATHER THAN BY INTETERS AND HAVE BEEN INCLUDED WITH EACH STRUCTURE RELATED TO A DEVICE.

4. LINK COUNT ITEM NUMBER AND THE DUNT-BLOCK-THIS-FILE BIT ARE RECENT ADDITIONS TO THE SDL REPORT ON A FILE.

9.2.9. COLUMNIZATION IN INDEX-SEQUENTIAL

THE UNALIGNED COLUMN PROBLEM FOR THE INDEX-SEQUENTIAL TABLES HAS BEEN CORRECTED.

D0265 DMPRINTIT - DMPRINTIT - CARD FILE - 02-19-73

THE CARD FILE FOR SYSTEM/DMPRINTIT HAS BEEN CHANGED FROM CARDD TO CARD.

EXAMPLE:

```
?RUN SYSTEM/DMPRINTIT
?DATA CARD
 .
 .
 .
 .
?END
```

DCALGOL

D0149 DCALGOL - DCALGOL QUEUE ATTRIBUTES - 01-15-73

SIMILAR TO TASK AND FILE ATTRIBUTES, QUEUE ATTRIBUTES MAY BE APPLIED TO ANY QUEUE OR QUEUE ARRAY ELEMENT. ILLEGAL ATTRIBUTE REFERENCES, SUCH AS ATTEMPTING TO SET A READ-ONLY ATTRIBUTE, WILL RESULT IN A SYNTAX ERROR OR RUN-TIME ERROR MESSAGE DISPLAYED ON THE CONSOLE AND ENTERED IN THE SYSTEM LOG. IN THE LATTER CASE, THE PROGRAM WILL NOT BE TERMINATED. THE FOLLOWING IS A DESCRIPTION OF QUEUE ATTRIBUTES AND THEIR MEANINGS:

1. QACTIVE. BOOLEAN. READ/WRITE.

   THIS ATTRIBUTE RETURNS THE CURRENT ACTIVE STATE OF A QUEUE. SETTING THIS ATTRIBUTE TRUE EXPLICITLY ACTIVATES THE QUEUE. SETTING IT FALSE DEACTIVATES THE QUEUE AND FLUSHES ANY MESSAGES CURRENTLY IN THE QUEUE. THIS ATTRIBUTE IS INITIALLY FALSE.

2. QMEMORYLIMIT. INTEGER. READ/WRITE.

   THIS DEFINES THE MAXIMUM VALUE THAT THE ATTRIBUTE QMEMORYSIZE (EXPLAINED BELOW) MAY ACHIEVE BEFORE DISK TANKING IS INVOKED. THE MAXIMUM VALUE OF THIS ATTRIBUTE IS 2**16-1 OR 65535. SETTING THIS ATTRIBUTE TO ZERO CAUSES ALL MESSAGES TO BE TANKED. QMEMORYLIMIT MAY BE CHANGED AT ANYTIME WITH THE FOLLOWING RESULTS:

   1. IF THE VALUE IS INCREASED, NO ATTEMPT WILL BE MADE TO DETANK ANY MESSAGES IN ORDER TO RAISE QMEMORYSIZE TO THE NEW LIMIT.

   2. IF THE VALUE IS DECREASED, NO ATTEMPT WILL BE MADE TO TANK ANY MESSAGES IN ORDER TO LOWER QMEMORYSIZE TO THE NEW LIMIT.

3. QMEMORYSIZE. INTEGER. READ-ONLY.

THIS ATTRIBUTE REFLECTS THE CURRENT SIZE IN WORDS OF THE RESIDENT PORTION OF A QUEUE. THIS IS THE SUM OF THE SIZES OF EACH COMPLETE MESSAGE AREA IN THE QUEUE PLUS ONE WORD (A LINK WORD) FOR EACH MESSAGE.

4. QSIZE. INTEGER. READ-ONLY.

QSIZE IS THE SUM OF THE SIZES OF EACH MESSAGE IN THE QUEUE BOTH IN MEMORY AND ON DISK. THIS INCLUDES THE MESSAGE AREAS ONLY AND NOT THE MESSAGE LINK WORDS.

5. QMESSAGECOUNT. INTEGER. READ-ONLY.

THIS IS THE TOTAL NUMBER OF MESSAGES IN THE QUEUE, INCLUDING ANY WHICH HAVE BEEN TANKED.

6. QUSERCOUNT. INTEGER. READ-ONLY.

THIS REPRESENTS THE NUMBER OF INDEPENDENT USERS OF A QUEUE. QUSERCOUNT IS INCREASED BY ONE WHEN THE QUEUE IS PASSED "BY VALUE" TO A PROCEDURE, AND DECREASED BY ONE WHEN EXITING THAT PROCEDURE. QUSERCOUNT CAN ALSO BE ALTERED WITH THE ATTACH FUNCTION. GIVEN THE EXAMPLE:

```
ATTACH(Q1,Q2);
```

Q1.QUSERCOUNT IS REDUCED BY ONE
Q2.QUSERCOUNT IS INCREASED BY ONE

7. QTANK. BOOLEAN. READ/WRITE.

QTANK IS TRUE IF A PORTION OF THE QUEUE IS CURRENTLY RESIDENT ON DISK, AND IS FALSE IF THE ENTIRE QUEUE IS IN MEMORY. SETTING THIS ATTRIBUTE TO TRUE CAUSES ALL MESSAGES CURRENTLY IN MEMORY TO BE TANKED TO DISK. SETTING QTANK TO FALSE IS AN ERROR.

8. QINSERTEVENT. EVENT. READ-ONLY.

THIS EVENT IS CAUSED EACH TIME A MESSAGE IS INSERTED

INTO A QUEUE VIA THE INSERT OR COMBINE FUNCTION, AND IS RESET WHEN THE LAST MESSAGE IS REMOVED FROM THE QUEUE VIA THE REMOVE, HOLD OR FLUSH FUNCTION. THIS EVENT MAY BE USED AS ANY OTHER EVENT. IT SHOULD BE NOTED THAT UTILIZING QINSERTEVENT IS FAR MORE EFFICIENT THAN INVOKING THE HOLD INTRINSIC. THE FOLLOWING CODE EXAMPLE IS EQUIVALENT TO HOLD(MSG,Q);

```
WHILE REMOVE(MSG,Q) = 0 DO
WAIT(Q.QINSERTEVENT);
```

IT IS NOW POSSIBLE TO CONSTRUCT MORE COMPLEX HOLD STATEMENTS;

```
CASE WAIT((2),Q.QINSERTEVENT,E)-1 OF
   BEGIN
        ; % 2 SECONDS HAPPENED
        S := REMOVE(MSG,Q);
        ; % EVENT "E" HAPPENED
   END;
```

9. QHEADSIZE. INTEGER. READ-ONLY.

THIS IS THE SIZE OF THE FIRST MESSAGE IN THE QUEUE, OR ZERO IF THE QUEUE IS EMPTY.

DCPPROGEN

D0161 DCPPROGEN - INHIBIT SYNC EDIT - 11-12-72

THIS PATCH ADDS A NEW BIT VARIABLE, "SYNCS", WHICH MAY BE SET, RESET, OR TESTED. THE BIT DESIGNATED IS BIT ZERO OF THE TYPEFIELD (THE INHIBIT SYNC EDIT BIT). NOTE: ASYNCHRONOUS LINE REQUEST SETS WILL TREAT BIT TESTS AS FALSE AND WILL NO-OP BIT SETS OR RESETS FOR THIS BIT VARIABLE.

D0192 DCPPROGEN - DIALOUT ERROR RESULTS - 12-08-72

FOR ERROR RESULTS AS A RESULT OF DIAL OUT REQUESTS, THE FOLLOWING INFORMATION IS GIVEN:

1. MSG[0].[39:16] CONTAINS AC AI AT TIME OF DIALOUT ERROR.
2. MSG[1].[39:8] CONTAINS THE REASON FOR TERMINATION.

    1 => DLO TRUE
    2 => ERROR TRANSMITTING DIGITS
    3 => ERROR TRANSMITTING EON
    4 => TIME OUT WAITING FOR COS => TRUE
    5 => UNEXPECTED INTERRUPT WAITING FOR COS => TRUE
    6 => UNEXPECTED INTERRUPT WAITING FOR CC => TRUE
    7 => TIME OUT WAITING FOR CC => TRUE.

DCSTATUS

00250 DCSTATUS - SYSTEM DCSTATUS - 02-19-73

INTRODUCTION

THE SYSTEM/DCSTATUS PROGRAM ALLOWS ONE TO PERFORM AN ANALYSIS OF THE CURRENT RUN-TIME STATE OF THE B6700 DATACOM SUBSYSTEM. THE PROGRAM MAKES USE OF THE DCSYSTEMTABLES INTRINSIC TO GAIN ACCESS TO TABLES MAINTAINED BY THE MCP (AND DCP), AND TO PERFORM A RUN-TIME SNAP SHOT ANALYSIS OF THEM. AS SUCH, THE PROGRAM FORMS AN EXAMPLE OF THE WAY IN WHICH THE DCSYSTEMTABLES INTRINSIC MAY BE USED.

SELECTION OF OUTPUT OPTIONS

THE PROGRAM MUST BE SUPPLIED WITH AN OPTION LIST WHICH SPECIFIES THOSE ELEMENTS OF THE DATACOM SUBSYSTEM WHICH ARE REQUIRED TO BE ANALYZED. THE OPTIONS ARE SPECIFIED ON A HIERARCHIAL BASIS, I.E.

(A) ANALYSIS FOR A STATION ONLY.
(B) ANALYSIS FOR ALL STATIONS ON A LINE.
(C) ANALYSIS FOR ALL LINES ON A CLUSTER.
(D) ANALYSIS FOR ALL CLUSTERS ON A DCP.
(E) ANALYSIS FOR ALL DCP S

THE HIERARCHIAL ITEM IS SELECTED BY USING THE KEY WORDS ALL, DCP, CLUSTER, LINE, OR STATION. EACH HIGHER ORDER ITEM IS INCLUSIVE OF ALL LOWER ORDER ITEMS. THUS IF A CLUSTER IS SPECIFIED, THEN THE ANALYSIS IS PERFORMED FOR ALL STATIONS ON ALL LINES UN THAT CLUSTER.

ANALYSIS OUTPUT FROM THE PROGRAM WILL NORMALLY BE SENT TO A SITE LINE PRINTER HOWEVER, THE PROGRAM WILL DETECT IF THE PRINTER FILE IS LABEL EQUATED TO REMOTE. FOR THIS CASE, THE OUTPUT FORMAT IS MODIFIED SO THAT THE OUTPUT WILL FIT A 72 CHARACTER LINE WIDTH.

SYNTAX OF OPTIONS

<OPTIONS>::=<OPTION LIST>
<OPTION LIST>::=<SUBSYSTEM SPECIFICATION>/
<SUBSYSTEM SPECIFICATION>; <OPTION LIST>
<SUBSYSTEM SPECIFICATION>::= ALL/<DCP DESIGNATE>/TABLES/
<CLUSTER DESIGNATE>/<LINE DESIGNATE>/
<STATION DESIGNATE>/<TERMINAL DESIGNATE>

<DCP DESIGNATE>::=DCP<DCP NUMBER>
<DCP NUMBER> ::=<UNSIGNED INTEGER>
<CLUSTER DESIGNATE>::=CLUSTER <DCP NUMBER>,<CLUSTER NUMBER>
<CLUSTER NUMBER>::=<UNSIGNED INTEGER>.

<LINE DESIGNATE>::=LINE <DCP NUMBER> , <LINE NUMBER>/
LINE <DCP NUMBER> , <CLUSTER NUMBER>,<LINE NUMBER>
<LINE NUMBER>::=<UNSIGNED INTEGER>

<STATION DESIGNATE>::= STATION <1SN> <NDL OPTION>.
<1SN ::= <EMPTY>/<LOGICAL STATION NUMBER>
<NDL OPTION>::= <EMPTY>/NDL
<TERMINAL DESIGNATE>::= TERMINAL <REMOTE TYPE INDEX>
<REMOTE TYPE INDEX>::= <EMPTY>/<UNSIGNED INTEGER>.
<TABLES> ::= TABLES

SEMANTICS:

<TABLES>OPTION

THIS WILL PRODUCE A RAW HEXADECIMAL DUMP OF THE DATACOM CONTROLLER AND DCP LINE AND STATION TABLES. TAG BITS ARE OMITTED FROM THE DUMP.

<ALL>OPTION

THIS WILL PRODUCE A COMPLETE ANALYSIS OF THE DATACOM NETWORK. THE TABLES ARE DUMPED, AN ANALYSIS OF THE LINE AND STATION TABLES TOGETHER WITH AN ANALYSIS OF EACH REMOTE TYPE. ALL OTHER OPTIONS ARE SUBSETS OF THIS OPTION.

<DCP DESIGNATE>OPTION

THIS WILL PRODUCE A FULL ANALYSIS OF THE DESIGNATED DCP S LINES AND STATIONS.

<CLUSTER DESIGNATE>OPTION

THIS WILL PRODUCE A FULL ANALYSIS OF THE DESIGNATED CLUSTER S LINES AND STATIONS.

<LINE DESIGNATE>OPTION

THIS WILL PRODUCE A FULL ANALYSIS OF THE DESIGNATED LINE AND ITS STATIONS.

<STATION DESIGNATE>OPTION

IF <NDL OPTION> IS EMPTY, THEN AN ANALYSIS OF THE TABLES MAINTAINED BY THE DATACOM CONTROLLER IS PRODUCED. IF <LSN> IS EMPTY, THEN THE ANALYSIS IS PRODUCED FOR ALL STATIONS. THIS WILL INCLUDE THOSE STATIONS WHICH ARE NOT CURRENTLY ASSIGNED TO A LINE, AND WILL THUS NOT BE ANALYZED UNDER THE DCP, LINE OR CLUSTER OPTIONS. IF NDL IS SPECIFIED, THEN THE ANALYSIS IS PRODUCED FROM THE NETWORK INFORMATION FILE, RATHER THAN FROM IN-CORE TABLES AND THE DCPCODE FILE, AND CONTAINS THE NDL DECLARATIONS FOR THAT STATION. THESE MAY NOT BE THE CURRENT ATTRIBUTES OF THE STATION, BECAUSE MODIFICATIONS MAY BE MADE AT RUN-TIME VIA DCWRITES.

<TERMINAL DESIGNATE>OPTION

THE NDL SPECIFICATIONS OF TERMINALS IS PRODUCED BY THIS OPTION. THIS INFORMATION IS RETRIEVED FROM THE NETWORK INFORMATION FILE.

THE <REMOTE TYPE INDEX> IS THE INDEX USED BY THE DATACOM CONTROLLER INTO A TABLE WHICH DESCRIBES EACH TERMINAL SPECIFIED IN NDL. IN PHYSICAL TERMS, TERMINALS ARE NUMBERED IN THE SEQUENCE IN WHICH THEY APPEAR IN THE NDL SPECIFICATION OF THE NETWORK.

RUNNING INSTRUCTIONS

THE PROGRAM IS COMPILED AS A PROCEDURE WITH AN ARRAY AS A PARAMETER.

THE OPTION LIST IS PASSED TO THE PROGRAM VIA THIS ARRAY.

TO COMPILE THE PROGRAM:-

```
? COMPILE SYSTEM/DCSTATUS DCALGOL LIBRARY
? DCALGOL FILE TAPE (TITLE = SYMBOL/DCSTATUS)
? DATA
$ MERGE INSTALLATION
$ SET LEVEL 2
(THE $ CARDS)
?END
```

TO EXECUTE THE PROGRAM:-

```
? EXECUTE SYSTEM/DCSTATUS ("<OPTIONS>")
? END
```

EXAMPLES:

```
? EXECUTE SYSTEM/DCSTATUS ("LINE 0, 2; CLUSTER 0, 4")
? END
```

THIS WILL CAUSE OUTPUT TO BE SENT TO A SITE LINE PRINTER.

```
? EXECUTE SYSTEM/DCSTATUS ("ALL")
? FILE LINE (TITLE = M332, KIND = REMOTE)
? END
```

THIS WILL CAUSE THE OUTPUT TO BE SENT TO A REMOTE STATION NAMED M332.

MISCELLANEOUS INFORMATION

SYSTEM/DCSTATUS MAY BE CALLED DIRECTLY FROM SYSTEM/MCSII USING THE DP CONTROL STATEMENT. OUTPUT MAY BE DIRECTED TO A SITE LINE PRINTER OR TO THE REMOTE DEVICE ON WHICH THE COMMAND WAS ENTERED.

SYNTAX.

IF SITE IS USED, ANY NUMBER OF COMMANDS MAY BE ENTERED

SIMULTANEOUSLY OUTPUT BEING DIRECTED TO A SITE LINE PRINTER. IF REMOTE IS USED, THEN OUTPUT WILL BE DIRECTED TO THE REMOTE DEVICE ON WHICH THE DP COMMAND WAS ENTERED. IT MUST BE NOTED THAT IN REMOTE MODE, ONLY ONE COPY OF DCSTATUS MAY BE RUN AT A TIME. CARE SHOULD BE TAKEN WHEN USING THE REMOTE OPTION, AS SYSTEM/DCSTATUS WILL NECESSARILY PRODUCE VOLUMINOUS OUTPUT FOR THE ALL, DCP AND CLUSTER OPTIONS, AND IT IS NOT POSSIBLE TO TERMINATE A PROGRAM FROM A REMOTE WHEN USING SYSTEM/MCSII.

IT SHOULD BE NOTED THAT THE DATACOM STATUS INTRINSIC DOES NOT LOCK THE VARIOUS MCP TABLES THAT IT ACCESSES. IT IS THEREFORE POSSIBLE THAT THE CONTENTS OF THE TABLES MAY CHANGE WHILE THE INTRINSIC IS RUNNING. IN PARTICULAR, SOME FLAGS MAINTAINED IN THE TABLES ARE SET IN A TRANSIENT MANNER. IT WILL THEREFORE BE A COINCIDENCE ONLY THAT A RUN OF DCSTATUS CATCHES THEM SET.

SYSTEM/DCSTATUS WILL ONLY RETURN MEANINGFUL INFORMATION IF DATACOM IS INITIALIZED. IF DATACOM IS NOT INITIALIZED THEN DCSTATUS OUTPUT WILL INDICATE SUCH A SITUATION.

IF, HOWEVER SYSTEM OPTION NUMBER 12, AUTODC IS SET TRUE, THEN A RUN OF SYSTEM/DCSTATUS WILL CAUSE AN IMMEDIATE INITIALIZATION OF THE DATACOM TABLES, AND INFORMATION RETURNED WILL BE MEANINGFUL. USING THIS MODE OF OPERATION, NO DCP S WILL BE FIRED UP, AND TO START DATACOM ACTIVITY A FURTHER DC [DCP NUMBER] MUST BE ENTERED ON THE SPO. NOTE ALSO THAT THE DATACOM TABLES WILL REMAIN INITIALIZED AFTER THE DCSTATUS RUN. TO RETURN THIS MEMORY AREA TO THE SYSTEM, A DCP MUST BE FIRED UP, AND THEN DS-ED.

ESPOL

D0140 ESPOL - EVENTS IN ESPOL - 10-16-72

AN EVENT MAY BE DECLARED "BY VALUE" IN A QUEUE DECLARATION. IN WHICH CASE THE EVENT ITSELF (A DOUBLE PRECISION WORD) IS INSERTED IN THE QUEUE STRUCTURE.

THE EVENT INTRINSIC MAY BE USED ANY PLACE IN THE SYNTAX THAT AN EVENT MAY BE USED. THE SYNTAX IS:

EVENT(<SUBSCRIPTED VARIABLE>)

THE <SUBSCRIPTED VARIABLE> MAY BE OF TYPE WORD, REAL, BOOLEAN, OR DOUBLE. THIS FUNCTION CAUSES AN "XTND" TO BE DONE ON THE INDEXED DESCRIPTOR.

D0166 ESPOL - NON-SAVE DECK OUTPUT - 10-23-72

THIS PATCH ALLOWS NON-SAVE SEGMENTS AND ARRAYS TO BE PUNCHED VIA THE $ DECK OPTION.

THESE CARDS (IN STACKER 2) ARE PUNCHED IN EBCDIC, WITHOUT TAGS, UP TO 12 WORDS/CARDS. THEY ARE PRECEDED BY A WORD WHICH HAS MOM-ADDR. IN [15:16], INDEX OF THE FIRST WORD IN [31:16], LENGTH OF THE CARD IN [43:12], AND 4"F" IN [47:4].

D0241 ESPOL - ESPOL DOLLAR OPTIONS - 11-06-72

ESPOL DOLLAR OPTIONS MAY BE SET EQUAL TO AN <OPTION EXPRESSION>. PREVIOUSLY, ONLY USER OPTIONS, OPTIONALLY PRECEDED BY A "NOT", WERE RECOGNIZED.

THE SYNTAX FOR THIS IS AS FOLLOWS:

00241 ESPOL - ESPOL DOLLAR OPTIONS - 11-06-72

```
SET <OPTION> = <OPTION EXPRESSION>
<OPTION EXPRESSION>::= <OPTION SECONDARY>/
                       <OPTION EXPRESSION><BOOLEAN OPERATOR>
                            <OPTION SECONDARY>
<OPTION SECONDARY>::= <OPTION PRIMARY>/NOT <OPTION PRIMARY>
<OPTION PRIMARY>  ::= <OPTION>/
                       <COMPILE TIME VARIABLE>
                       <RELATIONAL OPERATOR><UNSIGNED INTEGER>/
                       (<OPTION EXPRESSION>)
<OPTION>::= <STANDARD OPTION>/<USER OPTION>

FOR EXAMPLE:
   $ SET KLUDGE = EXPERIMENTAL OR OPTM AND NOT FUNNY
   $SET OMIT = NOT(OPTM EQV SUPEROP)
```

THE STANDARD RULES OF PRECEDENCE FOR BOOLEAN OPERATIONS ARE OBSERVED. THE RELATIONAL OPERATORS MUST BE THE SPECIAL CHARACTER VARIETY. THE "DECLARATION" AND USE OF USER OPTIONS ARE NO LONGER ASSOCIATED WITH THE BLOCK STRUCTURE OF ESPOL. "NO OPTION ACTION" $ CARDS, I.E., $ CARDS WHICH USED TO CAUSE STANDARD OPTIONS TO BE CLEARED, NOW CAUSE ALL OPTIONS TO BE CLEARED.

00242 ESPOL - PROCEDURE DECLARATION - 11-06-72

THE SYNTAX FOR CONTROLLING THE SEGMENTATION AND STATE OF PROCEDURES HAS BEEN EXTENDED. ON PAGE 7-29 OF THE JUNE 1972 B6700/B7700 ESPOL INFORMATION MANUAL, REVISE THE SYNTAX OF PROCEDURE DECLARATION TO:

```
<PROCEDURE DECLARATION>::= <SAVE PART><PROCEDURE TYPE>PROCEDURE
                           <PROCEDURE HEADING><PROCEDURE BODY>
<SAVE PART>::= <EMPTY>/SAVE/SAVE 1/<RESIDENCY PART><STATE PART>
<RESIDENCY PART>::= <EMPTY>/RESIDENT/INITIALIZATION
<STATE PART>::= <EMPTY>/CONTROL
```

THE "RESIDENT" DECLARATOR INHIBITS SEGMENTATION. "INITIALIZATION" IMPLIES RESIDENCY IN SEGMENT ONE. THE <RESIDENCY PART> HAS NO EFFECT ON STATE. THUS:

D0242 ESPOL - PROCEDURE DECLARATION - 11-06-72

"SAVE" IS EQUIVALENT TO "RESIDENT CONTROL"
"SAVE 1" IS EQUIVALENT OT "INITIALIZATION CONTROL"

D0243 ESPOL - EXPONENTIATION & MULTIPLICATN - 12-04-72

THE SYMBOLS FOR EXPONENTIATION AND MULTIPLICATION HAVE BEEN CHANGED TO THOSE OF ALGOL. ONLY THE MEANING OF THE ASTERISK HAS BEEN CHANGED; THE OLD MULTIPLY SIGN IS STILL ACCEPTED FOR MULTIPLICATION. THE NEW DOLLAR OPTION "OLDEXPO" MAY BE USED TO CAUSE THE ATERISK TO BE RECOGNIZED AS THE EXPONENTIATION SYMBOL.

SYMBOL	MEANING
×	MULTIPLICATION
*	MULTIPLICATION (IF OLDEXPO IS RESET)
*	EXPONENTIATION (IF OLDEXPO IS SET)
**	EXPONENTIATION

D0244 ESPOL - THE WORD INTRINSIC "STFF" - 12-04-72

A LABEL WHICH HAS HAD A PCW GENERATED FOR IT CAN BE USED AS AN ARGUMENT IN THE STFF INTRINSIC.

D0245 ESPOL - POINTER EXPRESSIONS - 12-11-72

THE DEFINITION OF <POINTER PARAMETERS> ON PAGE 9-20 OF THE JUNE 1972 ESPOL LANGUAGE MANUAL SHOULD BE EXPANDED AS FOLLOWS:

```
<POINTER PARAMETERS>::= <ARRAY PART>/<ARRAY PART>,
                      ( <CHARACTER SIZE>/<STRING>
<CHARACTER SIZE>::= 4/6/8/*/<POINTER EXPRESSION>
```

THE <STRING> MUST BE LONGER THAN 48 BITS. ITS MAXIMUM INTERNAL CHARACTER SIZE DETERMINES THE CHARACTER SIZE OF THE READ-ONLY POINTER.

D0245 ESPOL - POINTER EXPRESSIONS - 12-11-72

THE <POINTER EXPRESSION> PART OF <CHARACTER SIZE> IS USEFUL FOR SETTING THE CHARACTER SIZE OF ONE POINTER TO THAT OF ANOTHER. THIS FEATURE HAS BEEN AVAILABLE FOR SOME TIME, THOUGH UNDOCUMENTED.

D0270 ESPOL - WRITEAFTER DOLLAR OPTION - 03-07-73

A "WRITEAFTER" DOLLAR OPTION HAS BEEN PROVIDED IN ESPOL. IT IS INTENDED TO BE USED FOR ESPOL INSTALLATION INTRINSICS THAT ARE CALLED BY FORTRAN PROGRAMS. IF THE OPTION IS SET, ALL NON-DIRECT I/OS TO PRINTER FILES WILL BE DONE IN THE FORTRAN MANNER, I.E., SPACING IS DONE BEFORE PRINTING. (SEE D0253 FOR ALGOL.)

ESPOL INTRINSICS

D0150 ESPOLINTRN - NEW FORMATTER - 09-05-72

THIS NOTE REPLACES P0505 IN THE MARK II.3 SOFTWARE IMPROVEMENTS DOCUMENT AND REPLACES D0129 IN THE MARK II.2 SYSTEM MISCELLANEA.

A NEW SET OF FORMATTING INTRINSICS HAS BEEN WRITTEN FOR FORTRAN, WITH NECESSARY CHANGES MADE IN THE FORMATTER, FREEFIELD INTRINSICS, BINARY I/O INTRINSIC, AND FORTRAN COMPILER. THESE CHANGES HAVE BEEN MADE IN A MANNER THAT SHOULD BE TRANSPARENT TO USERS, BUT RESULT IN FASTER I/O EXECUTION.

THE GREATEST IMPROVEMENT SHOULD BE NOTICED IN FORMATTED I/O. BINARY I/O PREVIOUSLY WAS CHANGED TO DECREASE EXECUTION TIME. MORE INTENSIVE STUDY WILL BE MADE OF THE FREEFIELD INTRINSIC ON FUTURE RELEASES. IN ADDITION, IT IS ANTICIPATED THAT ALGOL WILL BE ADOPTED TO THE NEW I/O FORMATTING INTRINSIC ON FUTURE RELEASES.

THE FORMATTING INTRINSICS ARE REDESIGNED TO BE LIST DRIVEN RATHER THAN FORMAT DRIVEN. STORAGE REQUIRED IS TAKEN IN THE STACK WHENEVER POSSIBLE. THERE ARE SEPARATE INTRINSICS FOR INPUT AND OUTPUT BECAUSE THE INTRINSICS ARE SPECIFIC TO FORTRAN AND BECAUSE THEY ARE SPECIALIZED TO INPUT OR OUTPUT. THE NUMBER OF TEST DECISIONS REQUIRED HAS BEEN SIGNIFICANTLY REDUCED.

ON INPUT:

1. DOUBLE PRECISION IS FULLY IMPLEMENTED FOR ALL FORMAT PHRASES.
2. "O" AND "Z" FORMATTING IS DONE AS IT IS IN THE DATA STATEMENT.
3. IN "I","O","Z","D","E", AND "F" FORMATS, LEADING, TRAILING AND IMBEDDED SPACES ARE CONSIDERED ZEROS ON

INPUT.

4. THE MEANING OF THE ERROR NUMBERS FOR INPUT ARE:

FORMAT ERROR

201 - INVALID DATA FOR "I" FORMAT.
202 - INVALID DATA FOR "O" FORMAT.
203 - INVALID DATA FOR "Z" FORMAT.
204 - INVALID DATA FOR "L" FORMAT.
205 - INVALID DATA FOR "E","F", OR "D" FORMAT.
206 - MISSING DECIMAL PLACES SPECIFICATION- "V" FORMAT
207 - MISSING WIDTH SEPCIFICATION-"V" FORMAT
208 - INVALID PHRASE CODE-"V" FORMAT
209 - INPUT VALUE TOO LARGE FOR LIST ELEMENT
210 - ATTEMPTED TO EXCEED THE END OF THE INPUT RECORD

5. IF A DATA-ERROR LABEL IS PRESENT ON THE READ STATEMENT THE PROGRAM WILL NOT BE DSED, BUT WILL CONTINUE TO THE DATA-ERROR LABEL WHERE THE FORMAT ERROR NUMBER WILL BE IN FIELD[24:8] OF THE I/O RESULT WORD.

ON OUTPUT:

THE NEW FORTRAN OUTPUT FORMATTER HAS A FEW FEATURES AND AREAS OF DIFFERENCE FROM THE CURRENT II.3 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 THIS 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 IN 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 E XX AND D XX, WHEN 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 DIFFERENCES BETWEEN EW.D AND DW.D ARE 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 SPECIFICATION, 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 = 2
       ABCDEFGHIJKL   IF W = 15
```

D1 IS REPRESENTED INTERNALLY AS:

HIGH HALF	LOW HALF	CHARACTERSIZE
"ABCDEF"	"GHIJKL"	EBCDIC
"00ABCDEF"	"00GHIJKL"	BCL

C. THE OW AND ZW 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 AND Z12 WOULD EDIT THE LOW HALF OF A D.P. VALUE, WHILE O32 AND Z24 WOULD EDIT BOTH HALVES, AND O20 AND Z15 WOULD EDIT THE LOWER ORDER 12 BITS OF THE HIGH HALF AND THE ENTIRE LOW HALF.

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 E21.10 WOULD YIELD (FOR SUITABLE INTERNAL VALUES):

```
     -.1234567891E-88
      .9876543210+876
    .3141529653E+09132
```

THE SAME HOLDS FOR D21.10, WITH A "D" USED FOR THE "E".

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., THE VALUE MIGHT HAVE BEEN -0.00000000132547698132.

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 OUTPUT 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, 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 OF THE 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

ERROR #	ERROR CONDITION
	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 INTEGER NOR TYPE LOGICAL (EXPRESSIONS OF TYPE INTEGER OR LOGICAL ARE EDITED UNDER GW.D AS IW OR LW, RESPECTIVELY). THEREFORE, THE DECIMAL PLACES ARE 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.
116	ATTEMPTED RECURSION: EVALUATION OF A LIST ELEMENT CAUSED A READ/WRITE/CLOSE ON THE CURRENT FILE.
117	RECORD OVERFLOW: ATTEMPTED TO FORMAT BEYOND END-OF-RECORD.

D0198 ESPOLINTRN - MAT INPUT CHAR CONTINUATION - 01-15-73

IN BASIC THE AMPERSAND CHARACTER (I.E., "&") CAN NOW BE USED TO INDICATE CONTINUATION OF DATA BEING SUPPLIED TO A "MAT INPUT" STATEMENT.

EXAMPLE:

```
100 DIM A(15)
200 MAT INPUT A
300 END
```

WHEN THE PROGRAM ABOVE IS EXECUTED THE "MAT INPUT" STATEMENT WILL CAUSE THE PROGRAM TO WAIT UNTIL THE DATA FOR MATRIX "A" HAS BEEN SUPPLIED. IF ALL THE DATA ELEMENTS CAN NOT BE SPECIFIED IN ONE LINE (FROM REMOTE DEVICES) OR ONE CARD (FROM BATCH MODE) THEN USING THE AMPERSAND WILL INDICATE TO THE PROGRAM THAT MORE DATA IS TO BE SUPPLIED. THE PROGRAM WILL THEN CONTINUE TO WAIT FOR MORE DATA. FOR THE EXAMPLE ABOVE THE DATA MIGHT BE:

```
1, 2, 3, 4,&
5, 6, 7,&
8, 9
```

THE PROGRAM WILL FIRST READ 1,2,3 AND 4 INTO A(1), A(2), A(3) AND A(4). WHEN IT SEES "&" IT LOOKS FOR MORE DATA ON THE NEXT INPUT RECORD AND ASSIGNS 5,6 AND 7 TO A(5), A(6), AND A(7). THE PROGRAM SEES ANOTHER "&" AND LOOKS FOR MORE DATA ASSIGNING 8 TO A(8) AND 9 TO A(9). THERE IS NO AMPERSAND AFTER 9 SO "A" NOW CONTAINS ONLY NINE ELEMENTS. (SEE SYSTEM NOTE D0197 ABOUT RESIZING OF VECTORS WITH "MAT INPUT" STATEMENTS.)

THE AMPERSAND MAY BE USED TO CONTINUE STRING DATA FOR STRING ARRAYS FILLED BY "MAT INPUT" STATEMENTS AS WELL AS NUMERIC DATA AS IN THE EXAMPLE. IN EITHER CASE THE AMPERSAND MUST BE PRECEDED BY A COMMA OR BE THE FIRST ITEM ON THE INPUT LINE.

SETTING THE OPTION "OLDBASIC" (SEE SYSTEM NOTE D0193) WILL VOID THE USE OF THE AMPERSAND AS A CONTINUATION CHARACTER. HOWEVER, IN THIS

D0198 ESPOLINTRN - MAT INPUT CHAR CONTINUATION - 01-15-73

MODE THE USE OF A CONTINUATION CHARACTER IS NOT NECESSARY SINCE "MAT INPUT" STATEMENTS WILL CONTINUE TO LOOK FOR DATA UNTIL THE ARRAY IS COMPLETELY FILLED (WHEN "OLDBASIC" IS SET).

D0271 ESPOLINTRN - K AND $ FORMAT MODIFRS-FORTRAN - 03-07-73

THIS CHANGE IMPLEMENTS THE K AND $ FORMAT MODIFIERS FOR FORTRAN OUTPUT FORMATTING:

1. PLACING A $ NEXT TO AN EDITING-PHRASE CAUSES A $ TO BE INSERTED ADJACENT (ON THE LEFT) TO EDITED DATUM. FOR EXAMPLE, 17.34 EDITED UNDER $F10.2 WOULD APPEAR AS ####$ 17.34 (WHERE THE # STANDS FOR A BLANK). IF THE $ CANNOT FIT WITHIN THE FIELD, ASTERISKS ARE INSERTED.

2. PLACING THE LETTER K NEXT TO AN EDITING PHRASE CAUSES COMMAS TO BE INSERTED BETWEEN DIGIT-TRIPLES TO THE LEFT OF THE DECIMAL POINT. FOR EXAMPLE, -1234.56 EDITED UNDER KF10.2 WOULD APPEAR AS #-1,234.56; 1234567 UNDER KI10 WOULD APPEAR AS #1,234,567; AND -987654 UNDER 5PKE20.5 WOULD APPEAR AS ###-98,765.40000E+01 (WHERE # STANDS FOR A BLANK).

NOTE THAT BOTH THE K AND $ MAY BE USED TOGETHER, E.G., K$F10.2 WITH 1234.56 WOULD PRODUCE #$1,234.56 (WHERE # STANDS FOR A BLANK); $KF10.2 IS A VALID ALTERNATE FORM.

THESE TWO PHRASE MODIFIERS MUST BE ADJACENT TO THE EDITING PHRASE (OR ADJACENT TO A MODIFIER ADJACENT TO THE PHRASE), E.G., 2K810.3 P8K$G20.6 ARE VALID BUT $2J5 IS IMPROPER.

THE MODIFIERS MAY NOT BE USED FOR INPUT AND ALL SUCH USE CAUSES FORMAT ERROR #208 (INVALID PHRASE). THEY MAY BE USED TO MODIFY ANY PHRASE CODE WHICH EDITS A DATUM, INCLUDING V.

D0280 ESPOLINTRN - BLANK FIELD ON FORMATTED INPUT - 03-28-73

THE INPUT FORMATTING INTRINSIC MAY BE MODIFIED TO GENERATE AN INTERNAL MINUS ZERO FOR A BLANK INPUT FIELD FOR I,O,Z, F,E,D AND G PHRASES. THIS IS DONE BY COMPILING THE INPUT FORMATTER WITH THE $ OPTION MINUSZEROFORBLANKFIELD SET, AND REPLACING THE INPUT FORMATTER IN THE INTRINSIC FILE BY BINDING.

FORTRAN

D0146 FORTRAN - FORTRAN OPTIMIZATION - 10-16-72

A FORTRAN COMPILER THAT ATTEMPTS TO OPTIMIZE THE CODE EMITTED BY PROGRAM FLOW ANALYSIS AND OTHER MEANS IS NOW AVAILABLE. INVOCATION OF THIS OPTIMIZATION IS BY USE OF THE DOLLAR CARD OPTION OPT IN THE STANDARD RELEASE FORTRAN COMPILER.

OPT CURRENTLY MAY HAVE THREE VALID VALUES: -1, 0, AND 1. THE VALUE OF OPT MAY BE SET BY FOLLOWING THE WORD OPT BY AN OPTIONAL EQUAL SIGN AND ONE OF THESE NUMBERS. ALTHOUGH THE VALUE OF OPT MAY BE CHANGED AT ANY TIME, A CHANGE FROM ZERO OR -1 TO 1 OR, 1 TO ANY LOWER VALUE TAKES EFFECT ONLY AT THE THE BEGINNING OF A PROGRAM UNIT. CHANGING BETWEEN OPT VALUES OF 0 AND -1 IN EITHER DIRECTION HAS EFFECT IMMEDIATELY.

AN OPT VALUE OF 1 WILL INVOKE OPTIMIZATION. TO SET THIS VALUE, THE FOLLOWING EXAMPLE DOLLAR CARD COULD BE USED:

```
$ OPT = 1
```

THE DEFAULT VALUE FOR OPT IS 0; THE VALUE -1 HAS BEEN DESCRIBED PREVIOUSLY. IT IS PRIMARILY INTENDED FOR NON-STANDARD PROGRAMS THAT RELY UPON SIDE EFFECTS AND IS NOT DESIGNED FOR GENERAL USAGE.

SUBPROGRAMS ELIGIBLE FOR OPTIMIZATION

NOT ALL PROGRAM UNITS ARE ELIGIBLE FOR OPTIMIZATION (OPT=1), AND THOSE THAT ARE NOT WILL BE FLAGGED. THE FOLLOWING PROGRAM UNIT FEATURES DISQUALIFY OPTIMIZATION:

1. A PROGRAM UNIT CONTAINING ONE OR MORE ENTRY STATEMENTS.
2. A PROGRAM UNIT WHICH EQUIVALENCES ONE OR MORE INTEGER, REAL, OR LOGICAL VARIABLES OR ARRAYS TO ONE OR MORE DOUBLE PRECISION OR COMPLEX VARIABLE OR ARRAY.

3. A FUNCTION SUBPROGRAM WHICH HAS A LABEL AS A FORMAL PARAMETER (DOES NOT APPLY TO SUBROUTINES). ALSO, ANY PROGRAM UNIT WHICH REFERENCES A FUNCTION THAT HAS A LABEL AS A PARAMETER.

4. A PROGRAM UNIT THAT CONTAINS A NAMELIST.

5. A PROGRAM UNIT WHICH PASSES A SUBPROGRAM AS AN ARGUMENT TO SOME SUBPROGRAM.

SINCE OPT MAY BE CHANGED BETWEEN 1 AND 0 ON A PROGRAM UNIT BASIS, MERELY BECAUSE A PROGRAM CONTAINS SUBROUTINES THAT ARE INELIGIBLE FOR OPTIMIZATION DOES NOT MEAN THAT THE PROGRAM AS A WHOLE WILL NOT BENEFIT, POSSIBLY SIGNIFICANTLY, FROM SETTING OPT OT 1 FOR THOSE PROGRAM UNITS WHERE PERMITTED.

OPTIMIZATION FEATURES

VARIABLES WHICH HAVE BEEN DECLARED WITH THE OPTION OWN SET WILL CONTINUE TO WORK CORRECTLY UNDER OPTIMIZATION; BUT FOR VARIABLES (NOT ARRAYS), SETTING "OWN" WILL NOT FACILITATE THE BEST CODE. THEREFORE, A NEW DOLLAR CARD OPTION "OWNARRAYS" HAS BEEN ADDED TO THE FORTRAN COMPILER. WHEN SET, THIS OPTION TREATS ALL ARRAYS AS IF THEY HAD BEEN DECLARED WITH "OWN" SET, BUT MAKES ALL SIMPLE VARIABLES LOCAL. IT IS RECOMMENDED THAT OWNARRAYS BE USED WITH FORTRAN, OPTIMIZED OR NOT, WHERE POSSIBLE.

IF THE FORTRAN COMPILER IS COMPILED WITH THE USER OPTION SETOWNARRAYS SET, THE OPTION OWNARRAYS WILL BE SET BY DEFAULT IN ALL FORTRAN COMPILATIONS AND MUST BE EXPLICITLY RESET BY THE INDIVIDUAL USER IF NOT DESIRED. THE DOLLAR CARD OPTION GRAPH IS EXPLICITLY DESIGNED FOR USE WITH OPTIMIZATION. THIS PRODUCES A ROUGH FLOW CHART OF THE PROGRAM UNIT AND CAN ASSIST PROGRAMMERS IN MAKING FULLER USE OF THE OPTIMIZED COMPILER.

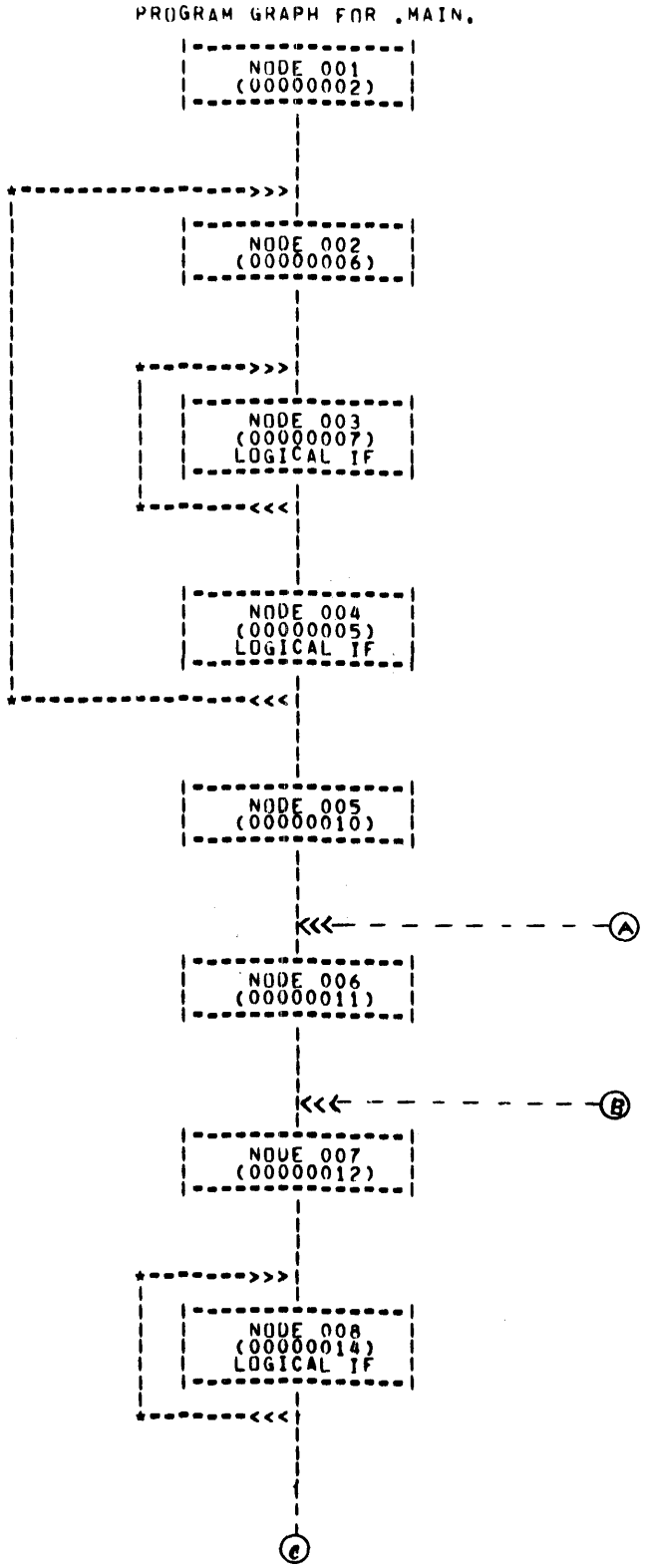
AS DESCRIBED ELSEWHERE, AUTOMATIC VECTORMODE CODE CAN BE EMITTED BY THE OPTIMIZING COMPILER, PROVIDED THE DOLLAR CARD OPTION VECTORMODE IS SET.

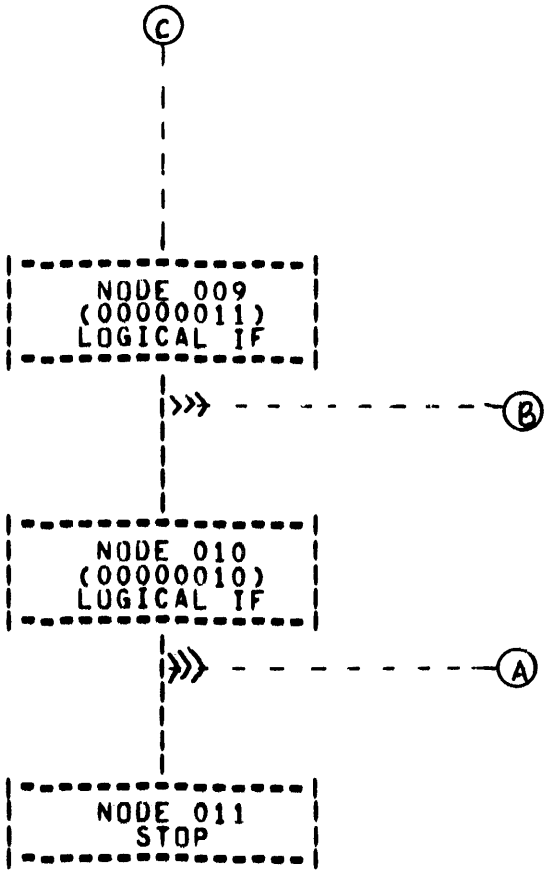
THE MONITOR AND DUMP FEATURES DO NOT WORK WITH OPTIMIZATION;

HOWEVER TRACE, STATISTICS, PROGRAM DUMP, AND DUMP STATISTICS DO.

INFORMATION OF THE USE AND PROBLEMS WITH THE USE OF OPTIMIZED FORTRAN WILL BE GREATLY APPRECIATED.

ON THE FOLLOWING PAGES IS AN EXAMPLE OF A SHORT OPTIMIZED PROGRAM.


PROGRAM GRAPH FOR .MAIN.
NODE 001
(00000002)
NODE 002
(00000006)
NODE 003
(00000007)
LOGICAL IF
NODE 004
(00000005)
LOGICAL IF
NODE 005
(00000010)
A
NODE 006
(00000011)
B
NODE 007
(00000012)
NODE 008
(00000014)
LOGICAL IF
C


C
NODE 009
(00000011)
LOGICAL IF
B
NODE 010
(00000010)
LOGICAL IF
A
NODE 011
STOP

```
       B6700/B7700   F O R T R A N   C O M P I L A T I O N   M A R K   2.3.017     MONDAY, 02/19/73     04:31 PM
$$LIST SINGLE STACK CODE FREF OPT=1                                                  C  000;0000:5
$SET GRAPH                                                                           C  000;0000:5
      DIMENSION A(50,50), B(50,50), C(50,50)                                         C  00000001
                                                                                      START OF SEGMENT 002
      N=10                                                                           C  00000002
      N=50                                                                           C  00000003
      C(1,1)=0                                                                       C  00000004
      DO 10 I=1,N                                                                    C  00000005
      DO 10 J=1,N                                                                    C  00000006
      A(I,J)=SIN(.4)                                                                 C  00000007
      B(I,J)=SQRT(2.0)                                                               C  00000008
10    CONTINUE                                                                       C  00000009
      DO 200 II=1,N                                                                  C  00000010
      DO 200 JJ=1,N                                                                  C  00000011
      C(II,JJ)=0                                                                     C  00000012
      DO 100 K=1,N                                                                   C  00000013
      C(II,JJ)=C(II,JJ)+A(II,K)*B(K,JJ)                                              C  00000014
100   CONTINUE                                                                       C  00000015
200   CONTINUE                                                                       C  00000016
      STOP                                                                           C  00000017
      END                                                                            C  00000018
```

```
STMT:00000005 NODE:0004    002:0016:0  VALC  (3,009) = I          3009                (3,00F)
                           0016:2  ONE                           B1
                           0016:3  ADD                           B0
STMT:00000005 NODE:0004    002:0016:4  NAMC  (3,009) = I          7009
                           0017:0  STON                          B9
STMT:00000005 NODE:0004    002:0017:1  LTR   50                   B232
                           0017:3  GRTR                          BA
                           0017:4  BRFL  000E:0                  A00000E
STMT:00000010 NODE:0005    002:0018:1  ONE                       B1                  (3,00F) = II
STMT:00000010 NODE:0006    002:0018:2  PUSH                      B4
                           0018:3  NOOP                        FF
                           0018:4  NOOP                        FF
                           0018:5  NOOP                        FF
STMT:00000011 NODE:0006    002:0019:0  ONE                       B1
                           0019:1  NAMC  (3,003) = JJ           7003
                           0019:3  STOD                        B8
STMT:00000012 NODE:0006    002:0019:4  VALC  (3,00F) = II       300F
                           001A:0  LTR   51                     B233
                           001A:2  SUBT                        B1
                           001A:3  NAMC  (3,00C) = I,0002       700C
                           001A:5  STOD                        B8
STMT:00000014 NODE:0007    002:001B:0  VALC  (3,003) = JJ       3003                (3,010)
                           001B:2  LTR   50                     B232
                           001B:4  MULT                        B2
STMT:00000014 NODE:0007    002:001B:5  DUPL                      B7
                           001C:0  VALC  (3,00C) = I,0002       300C
                           001C:2  ADD                         B0
                           001C:3  NAMC  (3,005) = C            7005
                           001C:5  INDX                        A6
                           001D:0  NAMC  (3,00D) = R,0003       700D
                           001D:2  OVRD                        BA
STMT:00000012 NODE:0007  * 001D:2  OVRN                        BB
                           001D:3  ZERO                        B0
                           001D:4  STOD                        B8
STMT:00000013 NODE:0007    002:001D:5  NAMC  (3,00A) = I,0000    700A
                           001F:1  LOAD                        BD
                           001F:2  NAMC  (3,002) = K            7002
                           001F:4  STOD                        B8
STMT:00000014 NODE:0007    002:001F:5  LTR   51                   B233
                           001F:1  SUBT                        B1
                           001F:2  NAMC  (3,00E) = I,0004       700E
                           001F:4  STOD                        B8
STMT:00000014 NODE:0008    002:001F:5  NOOP                      FF
STMT:00000014 NODE:0008    002:0020:0  VALC  (3,002) = K        3002
                           0020:2  LTR   50                     B232
                           0020:4  MULT                        B2
                           0020:5  VALC  (3,00C) = I,0002       300C
                           0021:1  ADD                         B0
                           0021:2  NAMC  (3,007) = A            7007
                           0021:4  NXLV                        AD
                           0021:5  VALC  (3,002) = K            3002
                           0022:1  VALC  (3,00E) = I,0004       300E
                           0022:3  ADD                         B0
                           0022:4  NOOP                        FF
                           0022:5  NAMC  (3,006) = B            7006
                           0023:1  NXLV                        AD
                           0023:2  MULT                        B2
                           0023:3  VALC  (3,00D) = R,0003       300D
                           0023:5  ADD                         B0
                           0024:0  NAMC  (3,00D) = R,0003       700D
                           0024:2  STOD                        B8
STMT:00000014 NODE:0008    002:0024:3  NAMC  (3,002) = K        7002
                           0024:5  STAR  (LINKED)               A40000
                           0025:2  BRTR  0020:0                 A10020
                           0025:5  NVLD                        FF
                      0024:5       STAR  0026:0                 A40026
STMT:00000011 NODE:0009    002:0026:0  VALC  (3,003) = JJ       3003                (3,010)
                           0026:2  ONE                         B1
                           0026:3  ADD                         B0
STMT:00000011 NODE:0009    002:0026:4  NAMC  (3,003) = JJ       7003
                           0027:0  STON                        B9
STMT:00000011 NODE:0009    002:0027:1  LTR   50                   B232
                           0027:3  GRTR                        BA
                           0027:4  BRFL  001B:0                 A0001B
STMT:00000010 NODE:0010    002:002B:1  VALC  (3,00F) = II       300F                (3,010)
                           002B:3  ONE                         B1
                           002B:4  ADD                         B0
STMT:00000010 NODE:0010    002:002B:5  NAMC  (3,00F) = II       700F
                           0029:1  STON                        B9
STMT:00000010 NODE:0010    002:0029:2  LTR   50                   B232
                           0029:4  GRTR                        BA
```

CODE LISTING FOR .MAIN.

```
STMT:00000018 NODE:0001   002:0000:0  ZERO                    R0                      (3,002) = K
STMT:00000018 NODE:0001   002:0000:1  ZERO                    R0                      (3,003) = IJ
STMT:00000018 NODE:0001   002:0000:2  ZERO                    R0                      (3,004) = J
STMT:00000018 NODE:0001   002:0000:3  LIT6  2500              R309C4                  (3,005) = C
                               0001:0  MKST                    AE
                               0001:1  NAMC  (0,007)           4007
                               0001:3  NAMC  (3,005)           7005
                               0001:5  STFF                    AF
                               0002:0  ONE                     R1
                               0002:1  LTA   16                R210
                               0002:3  ENTR                    AB
STMT:00000018 NODE:0001   002:0002:4  LIT6  2500              R309C4                  (3,006) = B
                               0003:1  MKST                    AE
                               0003:2  NAMC  (0,007)           4007
                               0003:4  NAMC  (3,006)           7006
                               0004:0  STFF                    AF
                               0004:1  ONE                     R1
                               0004:2  LTA   16                R210
                               0004:4  ENTR                    AB
STMT:00000018 NODE:0001   002:0004:5  LIT6  2500              R309C4                  (3,007) = A
                               0005:2  MKST                    AE
                               0005:3  NAMC  (0,007)           4007
                               0005:5  NAMC  (3,007)           7007
                               0006:1  STFF                    AF
                               0006:2  ONE                     R1
                               0006:3  LTA   16                R210
                               0006:5  ENTR                    AB
STMT:00000018 NODE:0001   002:0007:0  LIT8  40000000 00010000 RFFFFFFFFFFF 800000001000 (3,008)
                               0009:0  LTA   6                 R206
                               0009:2  STAG                    9584
STMT:00000004 NODE:0001   002:0009:4  ZERO                    R0
                               0009:5  NAMC  (3,005) = C       7005
                               000A:1  INDX                    A6
                               000A:2  ZERO                    R0
                               000A:3  STOD                    R8
STMT:00000005 NODE:0001   002:000A:4  ONE                     R1                      (3,009) = I
STMT:00000006 NODE:0001   002:000A:5  LIT8  68771905537       RF 001003200001         (3,00A) = I.0000
                               000C:0  LTA   4                 R204
                               000C:2  STAG                    9584
STMT:00000007 NODE:0001   002:000C:4  ZERO                    R0                      (3,00B) = I.0001
STMT:00000012 NODE:0001   002:000C:5  ZERO                    R0                      (3,00C) = I.0002
STMT:00000014 NODE:0001   002:000D:0  ZERO                    R0                      (3,00D) = R.0003
STMT:00000014 NODE:0001   002:000D:1  ZERO                    R0                      (3,00E) = I.0004
STMT:00000014 NODE:0002   002:000D:2  PUSH                    R4
                               000D:3  NOOP                    FF
                               000D:4  NOOP                    FF
                               000D:5  NOOP                    FF
STMT:00000006 NODE:0002   002:000F:0  NAMC  (3,00A) = I.0000  700A
                               000F:2  LOAD                    R0
                               000F:3  NAMC  (3,004) = J       7004
                               000F:5  STOD                    R8
STMT:00000007 NODE:0002   002:000F:0  VALC  (3,009) = I       3009
                               000F:2  LTA   51                R233
                               000F:4  SUBT                    A1
                               000F:5  NAMC  (3,00B) = I.0001  700B
                               0010:1  STOD                    R8
STMT:00000007 NODE:0003   002:0010:2  NOOP                    FF
                               0010:3  NOOP                    FF
                               0010:4  NOOP                    FF
                               0010:5  NOOP                    FF
STMT:00000008 NODE:0003   002:0011:0  VALC  (3,004) = J       3004                    (3,00F)
                               0011:2  LTA   50                R232
                               0011:4  MULT                    A2
                               0011:5  VALC  (3,00B) = I.0001  300B
                               0012:1  ADD                     A0
STMT:00000007 NODE:0003   002:0012:2  DUPL                    A7
                               0012:3  NAMC  (3,007) = A       7007
                               0012:5  INDX                    A6
                               0013:0  VALC  (1,006)           2006 = 26B1D8750257
                               0013:2  STOD                    R8
STMT:00000008 NODE:0003   002:0013:3  NAMC  (3,006) = B       7006
                               0013:5  INDX                    A6
                               0014:0  VALC  (1,005)           2005 = 2616A09E667F
                               0014:2  STOD                    R8
STMT:00000008 NODE:0003   002:0014:3  NAMC  (3,004) = J       7004
                               0014:5  STBR  (LINKED)          A40000
                               0015:2  BRTR  0011:0            A10011
                               0015:5  NVLO                    FF
                          0014:5       STBR  0016:0            A40016
```

```
                                        0029:5  BREL   0019:0                  A00019
STMT:00000010 NODE:0011    002:002A:2   MKST                                   AF
                                        002A:3  NAMC   (0,00A)                 400A
                                        002A:5  ENTR                           AB
                                        002B:0  EXIT                           A3
                      *****  LOCAL VARIABLES  *****
REAL     ARRAY     A        REL. ADDR. = 3:0007.  SIZE = 02500
REAL     ARRAY     B        REL. ADDR. = 3:0006.  SIZE = 02500
REAL     ARRAY     C        REL. ADDR. = 3:0005.  SIZE = 02500
N        IS NOT REFERENCED IN THIS ROUTINE
INTEGER  VARIABLE  I        REL. ADDR. = 3:0009
INTEGER  VARIABLE  J        REL. ADDR. = 3:0004
INTEGER  VARIABLE  II       REL. ADDR. = 3:000F
INTEGER  VARIABLE  JJ       REL. ADDR. = 3:0003
INTEGER  VARIABLE  K        REL. ADDR. = 3:0002
INTEGER  VARIABLE  T.0000   REL. ADDR. = 3:000A
INTEGER  VARIABLE  T.0001   REL. ADDR. = 3:000B
INTEGER  VARIABLE  T.0002   REL. ADDR. = 3:000C
REAL     VARIABLE  R.0003   REL. ADDR. = 3:000D
INTEGER  VARIABLE  T.0004   REL. ADDR. = 3:000E
                                        002B:1  NVLD                           FF
                                        002B:2  NVLD                           FF
                                        002B:3  NVLD                           FF
                                        002B:4  NVLD                           FF
                                        002B:5  NVLD                           FF
                                                                  SEGMENT 002 IS 002C LONG
```

D0153 FORTRAN - "FIRST" DOLLAR CARD OPTION - 10-30-72

THIS PATCH IMPLEMENTS THE FORTRAN COMPILE TIME OPTION GETFIRSTDOLLARCARDFROMSPO. IF SET THE COMPILER ACCEPTS THE FIRST DOLLAR CARD FROM THE OPERATORS CONSOLE VIA AN "AX" INPUT MESSAGE. THIS SHOULD BE RESET FOR NORMAL OPERATIONS.

D0157 FORTRAN - "OWNARRAYS" OPTION - 01-15-73

A NEW DOLLAR-CARD OPTION "OWNARRAYS" HAS BEEN ESTABLISHED IN FORTRAN. WITH THIS OPTION SET ALL ARRAYS (AND ONLY ARRAYS UNLESS OWN IS ALSO SET) WILL BE TREATED AS OWN ARRAYS. HENCE SPACE WILL NOT BE GIVEN UP UPON EXIT OR RETURN FROM THE SUBPROGRAM NOR REINITIALIZED UPON RE-ENTRY.

D0160 FORTRAN - TRUNCATED IDENTIFIERS - 02-19-73

THIS PATCH WILL CAUSE A WARNING, "IDENTIFIER TRUNCATED TO 6 CHARACTERS" FOR ALL IDENTIFIERS REQUIRING SUCH TRUNCATION. PREVIOUSLY THE WARNING WAS ISSUED ONLY IF THE FIRST OCCURENCE CAUSED TRUNCATION. FOR EXISTING PROGRAMS IN WHICH THIS ERROR FREQUENTLY OCCURS THE DOLLAR OPTION "SUPRS" CAN BE SET TO ELIMINATE THE WARNING MESSAGES.

D0165 FORTRAN - CORE TO CORE DATA TRANSFER - 11-06-72

CORE TO CORE DATA TRANSFER CONSTRUCT HAS BEEN ADDED TO FORTRAN INPUT/OUTPUT STATEMENTS.

ADDITIONAL FORMS OF THE READ STATEMENT ARE:

```
READ (A, F) M
READ (A, F, R) M
READ (A, F)
READ (A, F, R)
READ (A) M
READ (A, R) M
```

ADDITIONAL FORMS OF THE WRITE STATEMENT ARE:

```
WRITE (A, F)
WRITE (A, F, R) M
WRITE (A, F)
WRITE (A, F, R)
WRITE (A) M
WRITE (A, R) M
```

WHERE "A" IS AN ARRAY IDENTIFIER; "F" IS A FORMAT NUMBER, AN ARRAY IDENTIFIER, OR A SLASH (/); AND "R" IS A RESULT CLAUSE LIST.

THESE FORMS ALLOW TRANSFER OF DATA BETWEEN ONE AREA OF STORAGE AND ANOTHER. TRANSFER TO AND FROM THE ARRAY IS ALWAYS BEGUN AT THE LOW END OF STORAGE MOVING TOWARD THE HIGH END UNDER THE CONTROL OF THE FORMAT SPECIFIED.

EXAMPLE:

```
DIMENSION A(2)
READ /, REPEAT, WIDTH, DEC
WRITE (A, 1) REPEAT, WIDTH, DEC
1 FORMAT ("(", I1, "F", I2, ".", I2, ")")
```

AFTER THE WRITE STATEMENT "A" WOULD CONTAIN Z4DF6C640F44B, Z40F25D404040 WHEN 6,4,2 WERE THE VALUES OF REPEAT, WIDTH, DEC RESPECTIVELY. THIS EXAMPLE MIGHT BE USED TO DYNAMICALLY CONSTRUCT FORMAT SPECIFICATIONS.

U0228 FORTRAN - DOLLAR CARD CHARACTER OPTIONS - 12-04-72

COMPILER DOLLAR CARD SYNTAX HAS BEEN EXTENDED TO INCLUDE THE

FOLLOWING OPTIONS:

CHARS = N: THIS OPTION SETS TO N THE NUMBER OF CHARACTERS OF A STRING (INCLUDING HOLLERITH) WHICH WILL BE STORED IN ONE WORD. PREVIOUSLY, THIS WAS CONSTANT AT SIX EBCDIC (OR ASCII) CHARACTERS FOR FORTRAN AND SIX BCL CHARACTERS FOR XFORTRAN. STRING CHARACTERS WILL BE STORED LEFT JUSTIFIED IN THE FIELD OF N RIGHT MOST BYTES OF A WORD OF ALL ZEROS WITH BLANK FILL ON THE RIGHT. WITH THE BCL OPTION (SEE BELOW) SET, N HAS A MAXIMUM VALUE OF 8 OTHERWISE 6.

EXAMPLE

```
$ SET CHARS = 4
REAL A(2)/"ABCDEF"/
```

WILL RESULT IN A(1) = Z0000C1C2C3C4, A(2) = Z0000C5C64040.

THE DEFAULT SETTING IS SIX EBCDIC CHARACTERS. NOTICE THAT WITH CHARS < 6, VARIABLES INITIALIZED TO ALL BLANKS WILL NO LONGER SET THE SIGN OF THE MANTISSA (46:1). HENCE, IF X IS SUCH A VARIABLE X ≠ "ABS(X).

BCL: THIS OPTION IMPLIES THAT ALL STRINGS (HOLLERITH OR PROPER) ARE TO BE USED AS BCL CHARACTERS. IF "CHARS" HAS NOT BEEN EXPLICITLY SET, THEN THE NUMBER OF CHARACTERS PER WORD WILL BE SET TO EIGHT. BCL AND ASCII OPTIONS CANNOT BE SET CONCURRENTLY; SHOULD THIS CONDITION OCCUR, A SYNTAX ERROR WILL BE GIVEN.

B5500: SAME AS B5700.

B5700: USE OF THIS OPTION ALLOWS COMPLETE COMPATIBILITY WITH BURROUGHS B5700 FORTRAN PROGRAMMING LANGUAGE. THE BCL OPTION WILL BE SET AND THE ASCII OPTION WILL BE RESET. IF CHARS HAS NOT BEEN EXPLICITLY USED, THEN THE CHARACTERS PER WORD WILL BE SET TO SIX.

THE ABOVE COMPILE TIME OPTIONS ELIMINATE THE NEED FOR A SEPARATE B5700 COMPATIBLE COMPILER; AND HENCEFORTH, SYSTEM/XFORTRAN COMPILER WILL NOT BE MAINTAINED. THIS PATCH ALSO ALLOWS COMPLETE INTERCHANGEABILITY BETWEEN APOSTROPHE AND QUOTE FOR EBCDIC PROPER STRINGS. THEIR USE AS DELIMITERS CANNOT BE MIXED.

D0228 FORTRAN - DOLLAR CARD CHARACTER OPTIONS - 12-04-72

EXAMPLE

REAL A(2)/"AB- -C""DEFGH-"/YIELDS A(1) = ZC1C27D7DC37F,
A(2) = ZC4C5C6C7C87D BUT "ABCDEF- IS AN INVALID LITERAL.

(NOTE THAT THE DASH HAS BEEN USED FOR THE APOSTROPHE.)

NOTE: APOSTROPHE REMAINS AN ILLEGAL BCL CHARACTER SO THAT ALTHOUGH IT MAY DELIMIT A BCL PROPER STRING, AN OCCURRENCE WITHIN SUCH A LITERAL WILL BE INTERPRETED AS A BCL INVALID CHARACTER; I.E., A QUESTION MARK.

ANOTHER FEATURE OF THIS PATCH IS THAT THE GRAPHIC ASSOCIATED WITH THE CARD CODE (0, 12-ZONE) IS IDENTICAL TO THE GRAPHIC FOR CARD CODE (8, 6, 12-ZONE) WHICH IS AN EBCDIC PLUS ("+").

D0231 FORTRAN - TRACE STATEMENT - 02-19-73

A NEW DEBUG STATEMENT HAS BEEN ADDED TO FORTRAN, THE TRACE STATEMENT. LIKE THE STATISTICS STATEMENT, THIS STATEMENT HAS EFFECT ACROSS SUBPROGRAM BOUNDARIES. THE SYNTAX IS:

DEBUG TRACE (F) T
DEBUG TRACE (F,C) T

WHERE

T IS THE TRACE LIST AND
F AND C ARE FILE NUMBERS OR <EMPTY>.

"DEBUG" MUST BE IN COLUMNS 1-5. COLUMN SIX MUST BE BLANK. THE TRACE STATEMENT MAY BE CONTINUED TO SUCCEEDING CARDS.

TRACE LISTS

T IS A LIST OF ELEMENTS SEPARATED BY COMMAS. ELIGIBLE ENTRIES ARE:

DUMP
NODUMP
PROGRAMDUMP

NOPROGRAMDUMP
CALLS
NONE
S
#S

WHERE S IS DEFINED TO BE ANY OF THE FOLLOWING:

ALL
IO
GO
GO TO
READ
WRITE
BACKSPACE
REWIND
PRINT
PUNCH
=
CALL
CONTINUE

STATEMENT ACTION

THIS STATEMENT ALLOWS THE TRACING OF A PROGRAMS FLOW BY PRINTING OUT ALL SEQUENCE NUMBERS AND STATEMENT NUMBERS OF THE SPECIFIED STATEMENTS AND SUBPROGRAM CALLS. COMPARATOR ACTION IS ALSO ALLOWED TO DETECT DEVIATIONS IN PREDETERMINED PROGRAM FLOW, AND TO HAVE PROGRAM DUMPS TAKEN SHOULD A MISMATCH ARISE.

"F" IS THE OUTPUT FILE. IF "F" IS <EMPTY> THEN THE PREVIOUS VALUE OF "F" IS CARRIED OVER. IF "F" HAS NEVER BEEN SPECIFIED THEN THE TRACING ACTION IS SUPPRESSED. "C" IS THE COMPARATOR FILE. IF "C" IS NOT SPECIFIED BY USE OR THE FIRST FORM OF THE STATEMENT, THEN THE TRACE OUTPUT IS WRITTEN ON THE FILE "F". "F" MAY BE A DISK OR TAPE FILE, AND IS AUTOMATICALLY LOCKED AT THE END OF THE PROGRAM. IF THE COMPARATOR FILE IS SPECIFIED, THEN INSTEAD OF WRITING A TRACE RECORD TO "F", A RECORD IS READ FROM "C", AND COMPARED TO THE

TRACE RECORD THAT WOULD HAVE BEEN OUTPUT TO "F" HAD "C" NOT BEEN SPECIFIED. IF THEY COMPARE AS IDENTICAL, NO ACTION IS TAKEN. IF THEY DO NOT COMPARE, THEN A MESSAGE IS WRITTEN TO "F", AND IF DUMP OR PROGRAM DUMP WAS SPECIFIED, A PROGRAM DUMP IS TAKEN.

THE LIST "T" IS USED TO SPECIFY THE TRACING ACTION. DUMP OR PROGRAM DUMP SPECIFY THAT PROGRAM DUMPS ARE TO BE TAKEN IF THE COMPARISON WITH "C" RECORDS FAILS. NODUMP OR NOPROGRAMDUMP RESETS THIS SPECIFICATION. NODUMP IS THE ASSUMED SETTING. NONE OF THESE OPTIONS HAVE ANY MEANING IF THE COMPARITOR MODE HAS NOT BEEN REQUESTED.

"CALLS" TRACES ALL SUBROUTINE AND FUNCTION ENTRIES AND EXITS. INTRINSIC CALLS ARE NOT TRACED.

"NONE" RESETS ALL REQUESTS PRECEDING IT. OTHERWISE EACH "T" ELEMENT IS PROCESSED ADDITIVELY.

THE SPECIFICATION "S" IS USED TO INDICATE THAT TRACING IS DESIRED ON AN EXECUTABLE STATEMENT OR CLASS OF EXECUTABLE STATEMENTS. IF PRECEEDED BY A POUND SIGN (#) THEN ONLY STATEMENTS OF THIS CLASS WITH STATEMENT NUMBERS ARE TRACED.

ALL - INDICAGES ALL EXECUTABLE STATEMENTS (AND SETS CALLS)
IO - INDICATES ALL I/O STATEMENTS
GO, GO TO - INDICATES GO TO STATEMENTS
PRINT, PUNCH, READ, WRITE - INDICATE THE STATEMENT NAMED
CALL, CONTINUE, BACKSPACE, REWIND - INDICATE THE STATEMENT NAMED
= - INDICATES ALL ASSIGNMENT AND DO STATEMENTS

FOLLOWING IS AN EXAMPLE OF A SAMPLE PROGRAM WITH A TRACE STATEMENT.

```
      B6700/B7700   F O R T R A N   C O M P I L A T I O N   M A R K   2.3.017    MONDAY, 02/19/73    04:35 PM
$SET $ LIST SINGLE OWNARRAYS      FREE                                                    C  000:000015
 $SET OPT=1                                                                  00001000   C  000:000015
DEBUG TRACE(6) #ALL                                                          00002000   C
                                                                                         START OF SEGMENT 002
                                                                                             FIB IS 0005 LONG
1       I=1                                                                  00003000   C
2       I=2                                                                  00004000   C
        I=I+1                                                                00005000   C
3       I=3                                                                  00006000   C
4       I=4                                                                  00007000   C
        CALL X                                                               00008000   C
123     CALL X                                                               00009000   C
456     STOP                                                                 00010000   C
        END                                                                  00011000   C
 $SET OPT=0                                                                  00012000   C
                                                                                         SEGMENT 002 IS 0030 LONG


                                                                                         START OF SEGMENT 004
        SUBROUTINE X                                                         00013000   C  004:000010
123     I=1                                                                  00014000   C  004:000010
456     I=456                                                                00015000   C  004:000614
        I=789                                                                00016000   C  004:000010
1000    I=1000                                                               00017000   C  004:000E10
        RETURN                                                               00018000   C  004:001510
        END                                                                  00019000   C  004:001513
                                                                                         SEGMENT 004 IS 001A LONG


                                                                                         START OF SEGMENT 005
                                                                                         SEGMENT 005 IS 000D LONG
NO ERRORS DETECTED.  NUMBER OF CARDS = 21.
COMPILATION TIME = 4 SECONDS ELAPSED.  0.52 SECONDS PROCESSING.
D2 STACK SIZE = 5 WORDS.  FILESIZE = 78 WORDS.  ESTIMATED CORE STORAGE REQUIREMENT = 211 WORDS.
TOTAL PROGRAM CODE = 125 WORDS.  ARRAY STORAGE = 0 WORDS.
NUMBER OF PROGRAM SEGMENTS = 4.  NUMBER OF DISK SEGMENTS = 16.
PROGRAM CODE FILE = TRACE/TRACE, COMPILED ON 02/18/73



STMT#   1       SEQ=00003000
STMT#   2       SEQ=00004000
STMT#   3       SEQ=00006000
STMT#   4       SEQ=00007000
ENTER X         SEQ=00008000
STMT#   123     SEQ=00014000
STMT#   456     SEQ=00015000
STMT#   1000    SEQ=00017000
EXIT  X         SEQ=00018000
STMT#   123     SEQ=00009000
ENTER X         SEQ=00009000
STMT#   123     SEQ=00014000
STMT#   456     SEQ=00015000
STMT#   1000    SEQ=00017000
EXIT  X         SEQ=00009000
STMT#   456     SEQ=00010000
```

D0231 FORTRAN - TRACE STATEMENT - 02-19-73

FORTRAN EXAMPLE

D0232 FORTRAN - FILE IDENTIFIER - 02-19-73

THIS PATCH IMPLEMENTS THE USE OF PROPER STRINGS AND HOLLERITH STRINGS WHEN SPECIFYING THE EXTERNAL IDENTIFIER WITHIN A FILE CARD.

EXAMPLE:

```
FILE 10 = "EBCDIC-NAME", RECORD = 100
FILE 11 = "AB"/2HCD, RECORD = 100
```

NOTE THERE IS NO AMBIGUITY BETWEEN HOLLERITH STRINGS AND IDENTIFIERS CREATED UNDER PREVIOUS SYNTAX SINCE IDENTIFIERS WHICH ARE NOT STRINGS ARE NOT ALLOWED TO BEGIN WITH NUMERIC CHARACTERS.

D0230 FORTRAN - FORTRAN DOLLAR CARD OPTIONS - 02-19-73

A NUMBER OF NEW DOLLAR CARD OPTIONS HAVE BEEN ADDED TO FORTRAN AND SOME EXISTING OPTIONS HAVE BEEN REVISED. THIS NOTE ATTEMPTS TO COLLATE ALL OF THE NEW DOLLAR CARD OPTIONS. UNLESS EXPLICITLY NOTED, ALL OF THESE OPTIONS MAY BE EXPLICITLY OR IMPLICITLY SET, RESET, OR POPPED FOLLOWING THE PROCEDURES FOR EXISTING OPTIONS; AND ARE RESET BY DEFAULT.

OWNARRAYS: MAY ONLY BE ALTERED EXPLICITLY. WHEN SET, THIS OPTION TREATS ALL ARRAYS AS IF THEY WERE DELCARED AS "OWNS" BUT DOES NOT AFFECT THE STATUS OF SIMPLE VARIABLES. THIS IS DESCRIBED IN GREATER DETAIL IN THE OPTIMIZATION SYSTEM NOTE, D0146.

($): DOLLAR CARDS (WITH THE DOLLAR SIGN IN COLUMN ONE) WILL BE LISTED IF LIST IS SET. IF THE FIRST $ SIGN ON THE CARD IS IN COLUMN TWO, IT WILL BE LISTED ACCORDING TO THE SAME RULES AS A SOURCE CARD.

D0230 FORTRAN - FORTRAN DOLLAR CARD OPTIONS - 02-19-73

GRAPH, VECTORMODE: OPTIMIZATION OPTIONS THAT ARE SIGNIFICANT ONLY IF OPTIMIZING. GRAPH PROVIDES ADDITIONAL LISTING; VECTORMODE WILL PERMIT EMITTING VECTORMODE CODE WHEN FEASIBLE. THIS IS DEXCRIBED IN GREATER DETAIL IN THE OPTIMIZATION SYSTEM NOTE, D0146.

B7700: AN OPTION THAT IS SIGNIFICANT ONLY IF OPTIMIZING. THIS INHIBITS SOME CONSTRUCTS THAT, WHILE OPTIMAL FOR THE B6700, ARE NOT PREFERRED ON THE B7700. ALL CODE EMITTED, WITH OR WITHOUT THIS OPTION, WILL RUN ON EITHER MACHINE. THIS ONLY AFFECTS EMITTING OPTIMAL CONTRUCTS.

END: WHEN ENCOUNTERED, SIGNALS END OF FILE ON THE PARTICULAR MEDIUM WHERE FOUND. THIS OPTION IS NOT IMPLICITLY SET, RESET, OR POPPED. IT IS AN INDEPENDENT OPTION, HANDLED SIMILARLY TO $ PAGE IN NOT AFFECTING OTHER OPTIONS.

B5500,B5700,BCL,ASCII,EBCDIC,CHARS=N: THESE OPTIONS ASSIST INCONVERTIBILITY OF PROGRAMS. THEY ARE DESCRIBED IN THE CHARACTERS PER WORD SYSTEM NOTE, D0228.

D0287 FORTRAN - STATISTICS IN FORTRAN - 04-11-73

IT SHOULD BE NOTED IN THE SYNTAX SECTION OF SYSTEM NOTE D0109 THAT TO GET STATISTICS, DIAGNOSTICS OR THE FILE NUMBER MUST BE ENCLOSED IN PARENTHESES.

INPUT-OUTPUT

D0185 MCP-1-0 = "WRITESPO" PROCEDURE = 12-04-72

1. THIS PROCEDURE ASSUMES WHAT ITS NAME IMPLIES: A WRITE OF DATA TO A CONSOLE IS NEEDED. THE PARAMETERS ARE:

```
(U, WDS, DIRARAY, IOFINISHEVENT)      % ALL CALL-BY-VALUE
                                        EXCEPT THE EVENT
```

WHERE

U = UNIT # OF CONSOLE

WDS = NUMBER OF WORDS TO BE WRITTEN

(DATA STRING MAY HAVE IMBEDDED "ETX", IN WHICH CASE IT WILL TERMINATE THE WRITE. NOTE THAT [19:3] MAY CONTAIN A VALUE OF ZERO THROUGH FIVE TO INDICATE CHARACTERS IN ADDITION TO THE NUMBER OF WORDS IN [16:17].)

DIRARAY = DIRECT ARRAY

(CONTAINS DATA TO BE WRITTEN AND DIRARAY.IOMASK HAS APPROPRIATE MASK.)

IOFINISHEVENT = EVENT CAUSED WHEN I/O IS FINISHED

2. WRITESPO IS A BOOLEAN PROCEDURE WHICH RETURNS:

FALSE = AOK (I.E., WRITE IS "IN MOTION")

TRUE = COULD NOT/WOULD NOT ATTEMPT WRITE

```
11:8 = 34 MEANS "U" IS BAD
     = 40 MEANS "DIRARAY" IS NOT DIRECT ARRAY
     = 41 MEANS "DIRARAY" HAS I/O IN PROCESS
     = 42 MEANS IOFINISHEVENT DECLARED LATER
          THAN DIRARAY
```

D0185 MCP-1-0 = "WRITESPO" PROCEDURE = 12-04-72

3. CALLER CAN USE DIRECT I/O (ARRAY) ATTRIBUTES:

   A. TO SET UP "IOMASK", OR RETRIEVE IT

   B. TO CHECK FOR "IOCOMPLETE"

   C. TO RETRIEVE "IORESULT" (RAW WORD)

   D. TO CHECK "IOERRORTYPE" (SEE I/O SUBSYSTEM MANUAL PP. 4-23)

   E. TO RETRIEVE "IOTIME" (IN 2.4 USECS).

4. CONTROLLER CAN TRANSMIT DATA TO ANY CONSOLE IF A USER IS CONNECTED VIA CONSOLE FILES. OUTPUT CAN BE "MIXED".

D0191 MCP-1-0 = FILE ATTRIBUTES = TIMELIMIT = 12-15-72

THE DATACOM FILE (KIND = REMOTE) ATTRIBUTE, "TIMELIMIT", HAS BEEN IMPLEMENTED. TIMELIMIT IS A POSITIVE REAL NUMBER IN UNITS OF SECONDS. IT CAN BE BOTH SET AND READ. THE ATTRIBUTE CAN BE SET VIA LABEL EQUATION, IN THE FILE DECLARATION, AT RUN TIME WITH THE FILE EITHER OPENED OR CLOSED, OR IN READ OR WRITE STATEMENTS. IT CAN BE READ ONLY WHILE THE FILE IS OPEN.

THE ACTION WHEN TIMELIMIT IS GREATER THAN ZERO IS AS FOLLOWS:

ON A READ STATEMENT, IF NO INPUT IS RECEIVED WITHIN TIMELIMIT SECONDS, THE READ STATEMENT IF TERMINATED WITH A TIMELIMIT ERROR.

ON A WRITE STATEMENT, IF NO BUFFER BECOMES AVAILABLE WITHIN TIMELIMIT SECONDS, THE WRITE STATEMENT IS TERMINATED WITH A TIMELIMIT ERROR.

A TIMELIMIT ERROR IS REPORTED BY THE LOGICAL I/O RESULT DESCRIPTOR HAVING THE ATTENTION BIT [0:1] AND BIT [15:1] TURNED ON.

SYNTAX FOR SETTING TIMELIMIT IN AN ALGOL I/O STATEMENT:

D0191 MCP-1-0 - FILE ATTRIBUTES - TIMELIMIT - 12-15-72

ADD TO THE <RECORD NUMBER OR CARRIAGE CONTROL> DEFINITION (SEE 9-21 IN ALGOL LANGUAGE DOCUMENT).

[TIMELIMIT <ARITHMETIC EXPRESSION>]

EXAMPLE:

WRITE(FILEID [TIMELIMIT 25.3],12,ARRAYROW);

D0196 MCP-1-0 - FILE ATTRIBUTES - CURRENTBLOCK - 12-19-72

THE FILE ATTRIBUTE, "CURRENTBLOCK", IS A READ ONLY INTEGER WHICH IS ACCESSABLE ONLY WHEN THE FILE IS OPEN. THE VALUE RETURNED IS THE SIZE OF THE BLOCK CURRENTLY IN USE, IN LOGICAL UNITS (I.E., INTMODE UNITS IF IT IS A CHARACTER ORIENTED FILE, OTHERWISE WORDS).

D0235 MCP-1-0 - FILE ATTRIBUTES - 02-19-73

LINENUM

THE ATTRIBUTE LINENUM INDICATES THE CURRENT LINE NUMBER OF THE LOGICAL PAGE, DEFINED BY THE PAGESIZE ATTRIBUTE. IT IS ONLY MEANINGFUL FOR PRINTER FILES WHERE THE PAGESIZE ATTRIBUTE HAS BEEN SET GREATER THAN ZERO. LINENUM CAN BE SET TO ANY VALUE BETWEEN ZERO AND 255.

USE OF THE "[LINE <AEXP>]" FORM OF THE <RECORD NUMBER OR CARRIAGE CONTROL> PART CAUSES THE LOGICAL I/O SUBSYSTEM TO SPACE FORWARD TO THE LOGICAL LINE EQUAL TO <AEXP> AND SETS LINENUM TO <AEXP>. (IF <AEXP> IS GREATER THAN OR EQUAL TO PAGESIZE, OR IF <AEXP> IS LESS THAN LINENUM (IMPLYING A BACKSPACE) AN END OF PAGE RESULT IS RETURNED. A SKIP TO CHANNEL ONE, I.E., "[SKIP 1]" RESETS LINENUM TO ONE. SETTING LINENUM GREATER THAN OR EQUAL TO PAGESIZE (WHEN IT IS GREATER THAN ZERO) WILL CAUSE AN END OF PAGE RESULT ON THE NEXT WRITE STATEMENT.

EVERY SERIAL WRITE STATEMENT INCREMENTS LINENUM (WHILE PAGESIZE IS

D0235 MCP-I-O - FILE ATTRIBUTES - 02-19-73

GREATER THAN ZERO). A SERIAL WRITE STATEMENT WHICH RESULTS IN END OF PAGE WILL HAVE BEEN PRINTED. AN END OF PAGE DOES NOT CAUSE THE NEXT WRITE STATEMENT TO START ON THE TOP OF A PHYSICAL PAGE. ANY SPECIAL ACTION AFTER THE END OF PAGE RESULT MUST BE DONE BY THE PROGRAM. THE VALUE OF LINENUM AFTER THE END OF PAGE IS ONE.

AN END OF PAGE RESULT IS THE SAME AS AN END OF FILE RESULT.

PAGESIZE

THE ATTRIBUTE PAGESIZE INDICATES THE NUMBER OF LINES ON A LOGICAL PAGE OF A PRINTER FILE. PAGESIZE CAN BE SET OR INTERROGATED AT ANYTIME. PAGESIZE CAN HAVE A VALUE BETWEEN 0 AND 255-INCLUSIVE. WHEN PAGESIZE IS NON-ZERO THE LOGICAL I/O SUBSYSTEM MAINTAINS THE LINENUM AND PAGE ATTRIBUTES. WHENEVER THE VALUE OF LINENUM BECOMES GREATER THAN OR EQUAL TO PAGESIZE, THE WRITE STATEMENT WILL RETURN AN END OF PAGE RESULT (THAT IS THE SAME AS END OF FILE) AND LINENUM WILL BE RESET TO 1. IF PAGESIZE IS RESET TO ZERO WHILE THE PRINTER FILE IS OPEN, PAGE AND LINENUM WILL SUBSEQUENTLY BE IGNORED. PREVIOUS TO II-4 THIS CAUSED A FATAL ERROR.

PAGESIZE IS ALSO A DATACOM FILE (KIND=REMOTE) ATTRIBUTE WITH A MORE RESTRICTED MEANING. PAGESIZE IS READ ONLY, THE FILE MUST BE OPEN TO ACCESS IT, AND IT REQUIRES A RELATIVE STATION NUMBER AS AN INDEX. AS A DATACOM FILE ATTRIBUTE IT RETURNS THE NUMBER OF LINES ON A PAGE AS SPECIFIED IN THE NDL DESCRIPTION OF THE STATION.

D0282 I-O - CARRIAGE CONTROL VALUES - 03-23-73

THE FIRST PARAGRAPH OF D0133 IN THE SYSTEM MISCELLANEA HAS BEEN CHANGED TO READ AS FOLLOWS:

THE FILE ATTRIBUTE "CARRIAGECONTROL" HAS BEEN IMPLEMENTED FOR PRINTER FILES. IT HAS THREE VALUES:

STANDARD - 0 - THE DEFAULT, NORMAL CARRIAGE CONTROL AS SPECIFIED BY THE I/O STATEMENT OR FORMAT.

CTLASA - 1

CTL360 - 2

NOTE: NON-STANDARD CARRIAGE CONTROL IS ONLY ALLOWED FOR CHARACTER ORIENTED (UNITS = TRUE) EBCDIC (INTMODE = EBCDIC) PRINTER FILES.

MCP

D0172 MCP - TIME SLICING AND CODE SWAPPING - 10-16-72

THE MCP WILL NOW PLACE CODE WITHIN THE SUBSPACE IF THE TASK ATTRIBUTE HAS SUBSPACE SET TO THREE. MEMORY ASSIGNED TO A SUBSPACE TASK WILL NOW BE INCREASED UP TO THE SIZE OF THE SUBSPACE WHENEVER A TASK EXCEEDS THE CURRENTLY ALLOTTED SPACE.

TIME SLICING IS IMPLEMENTED AS FOLLOWS:

ONE OF THE GOALS OF THE SUBSPACE TASK EXECUTION OPTION IS TO ALLOW A LARGE NUMBER OF BURST-ORIENTED TASKS TO RUN WITHOUT FREEZING MEMORY RESOURCES DURING THEIR DORMANT PERIODS. MEMORY IS FREED BY SWAPPING THE DORMANT TASK TO DISK SO THAT ITS MEMORY RESOURCES ARE AVAILABLE FOR USE BY ANOTHER SUCH TASK. BECAUSE A LARGE NUMBER OF TASKS ARE BIDDING FOR A SOMEWHAT SMALLER MEMORY RESOURCE, TASKS WHICH FOR SOME REASON DISCONTINUE THEIR BURST-ORIENTATION FOR SOME DURATION OF TIME MUST HAVE AN ARTIFICIAL BURST RATE IMPOSED UPON THEM. THIS IS TIME SLICING AND IS NECESSARY TO ENSURE THAT ALL TASKS WILL HAVE AN OPPORTUNITY TO USE THE SUBSPACE MEMORY RESOURCE. THERE ARE FIVE CASES WHEN A TASK OPERATING WITHIN A SUBSPACE WILL BE SWAPPED OUT TO DISK:

A. WHENEVER THE TASK ATTEMPTS TO OBTAIN INPUT FROM THE DATACOM SUBSYSTEM AND THE REQUIRED INPUT IS NOT YET AVAILABLE;

B. WHENEVER THE OUTPUT BUFFERS FOR FILES BEING DIRECTED TO THE DATACOM SUBSYSTEM ARE FILLED AND THE TASK ATTEMPTS MORE SUCH OUTPUT;

C. WHENEVER THE TASK HAS BEEN SUSPENDED;

D. WHENEVER THE TIME SLICE ALLOCATED THE TASK HAS EXPIRED;

E. WHENEVER THE TASK EXCEEDS THE SUBSPACE SIZE ALLOCATED TO

IT ON ITS PREVIOUS SWAP-IN.

TASKS ARE RETURNED TO MEMORY (WITH RELOCATION SO THAT IT IS NOT REQUIRED THAT THE IDENTICAL MEMORY SPACE FROM WHICH A TASK WAS SWAPPED BE AVAILABLE IN ORDER TO SWAP THE TASK BACK TO CORE) WHENEVER THEY ARE CAPABLE OF UTILIZING THE PROCESSOR AND THERE IS ADEQUATE MEMORY AVAILABLE FOR THE SWAP-IN. THERE ARE TWO PRIORITY LEVELS FOR SELECTING READY-TO-RUN TASKS FOR SWAP-IN.

A. DEMAND SWAPPING

TASKS WHICH ARE NEW TO THE SYSTEM, WHICH HAVE RECEIVED THE DATACOM INPUT FOR WHICH THEY ARE WAITING, TASKS FOR WHICH THE DATACOM SUBSYSTEM HAS OUTPUT AT LEAST ONE HALF OF THE DATA WHICH EXCESS ORIGINALLY CAUSED THE SWAP-OUT, AND TASKS WHICH HAVE BEEN AWAKENED FROM SWAP-OUT SUSPENSION, WILL BE CONSIDERED AHEAD OF A TASK WHICH WAS SWAPPED BECAUSE OF TIME SLICE EXPIRATION. THE EXCEPTION TO THIS IS THAT WHENEVER THE NUMBER OF TASKS INSERTED IN FRONT OF A SLICE JOB EXCEEDS ITS SLICE NUMBER BY TWO, THAT TASK WILL REVERT TO DEMAND STATUS. WITHIN DEMAND STATUS, TASKS ARE ORDERED IN A FIRST-IN FIRST-SWAPPED ORDER.

B. TIME SLICED TASKS

TASKS SWAPPED OUT BECAUSE THEY EXCEED THEIR TIME SLICE WILL BE SWAPPED INTO AVAILABLE MEMORY ONLY IF THERE ARE NO DEMAND STATUS SWAP REQUESTS WHICH CAN BE SATISFIED.

IT SHOULD BE NOTED THAT PRIORITY IS NOT A DIRECT CONSIDERATION IN THE SWAPPING ALGORITHM; INSTEAD, IT APPEARS IN THE FORMULA USED TO COMPUTE TIME SLICES. THE TIME SLICE ALLOCATED A TASK IS EXPRESSED IN TERMS OF BOTH ELAPSED TIME AND PROCESSOR TIME. BEFORE ALLOCATING A PROCESSOR TO A SWAPPABLE JOB, ITS PROCESSOR AND ELAPSED TIME SLICES ARE CHECKED. IF EITHER IS EXCEEDED, A NEW SLICE IS COMPUTED AS PER THE FORMULA BELOW, AND THE JOB IS SWAPPED OUT. WHEN A JOB IS SWAPPED OUT DUE TO A DEMAND CONDITION (I.E., DATACOM INPUT REQUIRED), ITS SLICE NUMBER IS RESET TO ZERO.

EACH TIME A TASK IS SWAPPED BECAUSE OF TIME SLICE EXCEEDED, THE

SLICE NUMBER IS INCREMENTED BY ONE. THIS NUMBER IS SUBJECT TO A MAXIMUM VALUE AS SPECIFIED IN THE SWAPDISK FILE, WITH A SYSTEM IMPOSED MAXIMUM OF 256.

THE FORMULA FOR COMPUTING A TIME SLICE IS:

$$T = (N * K1 + C + P + 8) * K2 + M * 416667$$
$$E = T * R$$

WHERE

T IS THE PROCESSOR TIME SLICE. UNITS ARE 2.4 USEC.

E IS THE ELAPSED TIME SLICE.

N IS THE SLICE NUMBER (THE MAXIMUM VALUE FOR N IS OBTAINED FROM WORD 5 OF THE FIRST RECORD OF SWAPDISK. IF THAT WORD IS ZERO, A DEFAULT VALUE OF SEVEN WILL BE USED.).

C IS THE CORE SPACE USED BY THE TASK IN CHUNKS.

M IS THE MINIMUM TIME SLICE IN SECONDS. THIS NUMBER IS OBTAINED FROM WORD 4 OF THE SWAPDISK. A DEFAULT VALUE OF ONE SECOND WILL BE USED IF THIS WORD IS ZERO.

K1 IS 4.

K2 IS 50000.

P IS PRIORITY.

R IS THE RATIO OF ELAPSED TIME TO PROCESSOR TIME. THIS NUMBER IS OBTAINED FROM WORD 6 OF SWAPDISK. IF THE VALUE OF WORD 6 IS 0, THEN 2 IS USED; OTHERWISE, WORD 6 MUST BE GREATER THAN OR EQUAL TO 1.

D0184 MCP - MULTIPLE MCP CODE FILES - 11-06-72

D0184 MCP = MULTIPLE MCP CODE FILES = 11-06-72

THIS CHANGE PROVIDES THE ABILITY TO HAVE MULTIPLE MCP CODE FILES (WHICH ARE HALT/LOAD CAPABLE) WITHOUT RUNNING DIRECTORY RECONSTRUCTION. WHEN MULTIPLE MCPS ARE PRESENT (WITH OR WITHOUT RECONSTRUCTION), AND AN IRRECOVERABLE ERROR OCCURS ON THE MCP CODE FILE, AN AUTOMATIC SWITCH TO BACKUP MCP WILL BE PERFORMED. IRRECOVERABLE ERRORS ON THE MCP WITH NO BACKUP CODE FILES WILL RESULT IN A DEFUNCT "MCP CODE ERR" BEING DISPLAYED.

BACKUP MCPS WITHOUT RECONSTRUCTION ARE ESTABLISHED IN TWO WAYS:

1. AT COLD START TIME VIA THE "BACKUPEUS" CARD OR "EU <UNIT NO> BACKUP EU" AND NOT SPECIFYING A DIRECTORY COPY FREQUENCY VIA THE "FREQUENCY" CARD;

2. RESERVE WITH RES DK <UNIT NO> AS MCP SYNTAX. DATA FILES ARE MOVED OFF THE SPECIFIED AREA AND AN MCP COPIED THERE.

HALT/LOADS FROM VARIOUS UNITS ARE POSSIBLE AND NO DIRECTORY RECONSTRUCTION WILL OCCUR UNLESS THE USER HAS SPECIFIED SUCH VIA THE COLD START "FREQUENCY" CARD OR RESERVED AN ALTERNATE RECONSTRUCTION CAPABLE EU VIA THE "RES DK <UNIT NO> AS ALTERNATE".

COLD STARTS AND "CM=S" WILL COPY THE NEW MCP TO ALL EU=S DESIGNATED TO HAVE MCP CODE FILES.

D0210 MCP = WORK FLOW MANAGEMENT = 01-15-73

THIS PATCH BEGINS THE IMPLEMENTATION OF WORK FLOW MANAGEMENT. THE DOCUMENTATION IS CONTAINED IN WORK FLOW MANAGEMENT USERS MANUAL VOLUME I AND WORK FLOW MANAGEMENT USERS MANUAL VOLUMN II.

D0227 MCP = MCS LOGGING = 02-05-73

A NEW DCALGOL INTRINSIC "MCSLOGGER" ALLOWS AN MCS TO MAKE ENTRIES INTO THE SYSTEM LOG. THIS INTRINSIC REPLACES THE PREVIOUS "SYSTEMLOG" INSTALLATION INTRINSIC. MCSLOGGER IS A REAL INTRINSIC WHICH HAS ONE PARAMETER, AN ARRAY, WHICH CONTAINS THE INFORMATION

TO BE LOGGED. THE FORMAT OF THE ARRAY IS DESCRIBED IN THE WORK FLOW MANAGEMENT DOCUMENT.

WITH THE EXCEPTION OF LOG-ON ENTRIES, THE CONTENTS OF THE PASSED ARRAY IS NOT MODIFIED, AND THE FIRSTWORD OF THE ARRAY (ARRAY[0]) MUST CONTAIN A VALID JOB NUMBER.

FOR LOG-ON, MCSLOGGER WILL SUPPLY A UNIQUE JOB NUMBER IN WORD ZERO OF THE ARRAY. THIS JOB NUMBER MUST BE USED FOR ALL FUTURE LOGGING FOR THIS USER, TERMINAL, ETC.

IF THE GIVEN ENTRY COULD NOT BE ENTERED IN THE LOG, MCSLOGGER WILL RETURN A NEGATIVE RESULT INDICATING THE SPECIFIC REASON THE REQUEST WAS DENIED. VALUES RETURNED BY MCSLOGGER AND THEIR RESPECTIVE MEANINGS ARE:

0 => ENTRY LOGGED SUCESSFULLY

-1 => MEANS EITHER:

1. ARRAY TOO SMALL TO CONTAIN THE INFORMATION (I.E., LENGTH FIELDS IN LINK WORDS WERE IN ERROR)

2. THE INFORMATION REQUIRED MORE THAN 256 WORDS.

-2 => THE CALLER IS NOT A VALID MCS,OR HAS NOT INITIALIZED ITS PRIMARY QUEUE.

-3 => A DISK PARITY OCCURED WHILE ENTERING THE RECORD IN THE LOG.

-4 => EITHER THE MAJOR TYPE OF THE ENTRY WAS NOT FOUR (MCS RECORD), OR THE MINOR TYPE WAS INVALID (LEQ ZERO OR GTR FOUR).

-5 => THE ENTRY WAS NOT LOG-ON AND WORD ZERO OF THE ARRAY DID NOT CONTAIN A VALID JOB NUMBER.

EXAMPLE:

```
RESULT:= MCSLOGGER (JOBINFO[J,*]);
```

D0248 MCP - TASK ATTRIBUTE "HISTORY" - 02-19-73

D0248 MCP - TASK ATTRIBUTE "HISTORY" - 02-19-73

IT SHOULD BE NOTED THAT IN THE MARK II.3 SYSTEM RELEASE THE VALUES RETURNED BY TASK.HISTORY WERE CHANGED RADICALLY. THERE HAS BEEN SOME FURTHER CHANGE IN MARK II.4. IT IS EXPECTED THAT THESE VALUES WILL BE IN A CONSTANT STATE OF FLUX AS THEY ARE USED HEAVILY BY THE MCP FOR PROCESS CONTROL AND ONLY INCIDENTALLY BY MCS FOR INFORMATION GATHERING. ANY USER PROGRAM WHICH USES THESE VALUES SHOULD BE PREPARED FOR MINOR CHANGES WITH EACH SYSTEM RELEASE. CURRENT VALUES MAY BE FOUND BY CONSULTING THE LIST OF DEFINES STARTING AT SEQUENCE NUMBER 05027200 IN THE MCP LISTING.

D0273 MCP - FORMMESSAGE ASSIGNED LP - 01-22-73

A FORMS MESSAGE CAN NOW BE ASSIGNED TO A LINE PRINTER USING THE CONTROL MESSAGE: FORM LP<UNITNUMBER> <STRING LESS THAN OR EQUAL TO 15 CHARACTERS>ETX. THE CONTROL MESSAGE, FORM LP<UNITNUMBER>ETX, RETURNS THE FORMMESSAGE ASSIGNED, IF IT EXISTS. IF A LINEPRINTER IS "FM-ED", ANY PRINTER FILE OPENED WITH THE SAME FORMMESSAGE WILL BE AUTOMATICALLY ASSIGNED TO THE "FM-ED" LINEPRINTER WITHOUT OPERATOR NOTIFICATION OR INTERVENTION. THE LINEPRINTER WHILE "FM-ED" IS NOT AVAILABLE TO A PRINTER FILE WHICH DOES NOT HAVE THE SAME (OR ANY) FORMMESSAGE.

THE CONTROL MESSAGE CL WILL UNASSIGN THE FORMMESSAGE.

MCSII

00251 MCSII - NEW FEATURES FOR SYSTEM MCSII - 02-19-73

1. LOGICALACK

A LOGICALACK STATUS ITEM FOR A STATION HAS BEEN IMPLEMENTED.

LOGICALACK MAY BE SET AT ATTACH TIME OR BY USE OF THE ALTER CONTROL STATEMENT. SETTING LOGICALACK CAUSES MCSII TO EXECUTE A SET LOGICALACK DCWRITE FOR THE SPECIFIED STATION AND SUBSEQUENTLY AUTOMATICALLY LOGICALLY ACKNOWLEDGE THAT STATION WHENEVER A TERMINATE LOGICALACK STATEMENT IS EXECUTED BY THE DCP FOR THAT STATION.

EXAMPLE:

```
ATTACH TC5AA, READY, ENABLED, LOGICALACK;

ALTER TC5BB LOGICALACK = TRUE;
```

SYNTAX:

```
<STATUS ITEM>::= ENABLED/READY/NONALL/MONERR/CONTROL/NOHELLO/
                 NOACK/AUTOERR/LOGICALACK
<MODEM SPECIFICATION>::=<EMPTY>/MODEM<MODEM ID>
<MODEM ID>::=<MODEM IDENTIFIER>
```

SEMANTICS:

THE ATTRIBUTE LIST MAY BE SPECIFIED IN ANY ORDER OR MAY BE <EMPTY>. WHERE AN <EMPTY> OPTION IS SPECIFIED, THAT ATTRIBUTE OR ATTRIBUTES WILL NOT BE UPDATED FOR THAT STATION.

EXAMPLE:

```
MOVE STATION TC5AA TO (0,0,3) READY ADAPTER 9 MODEM M2W
                              TERMINAL TC5TYPE2;
```

II. UPDATE CONTROL STATEMENT

THE NEW CONTROL STATEMENT UPDATE ENABLES THE USER TO INSTRUCT MCSII TO CHANGE THE ATTRIBUTES OF A LINE BY MAKING USE OF THE UPDATELINE DCWRITE.

SYNTAX:

<UPDATE STATEMENT> ::=UPDATE<LINE ADDRESS><LINE ATTRIBUTE LIST>
<LINE ADDRESS>::=(<DCPNUMBER>,<CLUSTER NUMBER>,<LINE NUMBER>)
<LINE ATTRIBUTE LIST>::= <ADAPTER CLASS SPECIFICATION>
<MODEM SPECIFICATION>
<ADAPTER CLASS SPECIFICATION>::=ADAPTER<ADAPTER CLASS><MODE PART>/
<EMPTY>
<ADAPTER CLASS>::=<UNSIGNED INTEGER>
<MODE PART>::=(MODEM)/(DIRECT)

SEMANTICS:

THIS CONTROL STATEMENT ALLOWS THE USER TO SPECIFY THE ADAPTER CLASS AND/OR MODEM FOR A PASSIVE LINE WHERE PASSIVE IS UNDERSTOOD TO MEAN A LINE WHICH HAS NO CURRENT STATION ATTACHMENT; TO EXECUTE AN UPDATELINE FOR AN ACTIVE LINE WILL RESULT IN A DCWRITE ERROR.

EXAMPLE.

```
UPDATE(0,3,12) ADAPTER 2 (MODEM) MODEM MI200;
```

III. MOVE CONTROL STATEMENT

EXTENSIONS TO THE MOVE CONTROL STATEMENT HAVE BEEN IMPLEMENTED TO MAKE USE OF THE NEW FEATURES OF DYNAMIC RECONFIGURATION.

FOR THE MOVE STATEMENT, THREE NEW OPTIONS WILL CAUSE MCSII TO INSTRUCT THE DCC TO CHANGE THE ADAPTER, MODEM AND TERMINAL ATTRIBUTES FOR THE SPECIFIED STATION PRIOR TO THE LOGICAL MOVE.

SYNTAX:

<MOVE STATION STATEMENT>::=MOVE STATION <STATION ID><TO PART>
<READY PART><STATION ATTRIBUTE LIST>
<STATION ATTRIBUTE LIST>::= <EMPTY>/UPDATE<ADAPTER SPECIFICATION>

<TERMINAL SPECIFICATION><MODEM SPECIFICATION>
<ADAPTER SPECIFICATION>::=<EMPTY>/ADAPTER<ADAPTER TYPE>
<ADAPTER TYPE>::=<UNSIGNED INTEGER>
<TERMINAL SPECIFICATION>::=<EMPTY>/TERMINAL<TERMINAL ID>
<TERMINAL ID>::=<TERMINAL IDENTIFIER>

IV. DP CONTROL STATEMENT

A NEW CONTROL STATEMENT, DP, ALLOWS THE USER TO CALL THE SYSTEM/ DCSTATUS PROGRAM FOR MCSII.

SYNTAX:

<DUMP STATEMENT>::= DP<OUTPUT PART>(<OPTIONS LIST>)
<OUTPUT PART::=SITE/REMOTE

SEMANTICS:

THE DP STATEMENT CAUSES SYSTEM/MCSII TO PROCESS SYSTEM/DCSTATUS. THE SITE OUTPUT PART WILL CAUSE THAT PROGRAM TO OUTPUT ITS ANALYSIS TO A LINE PRINTER OF THE SITE, THE REMOTE OPTION WILL CAUSE THEOUTPUT TO BE SENT TO THE REMOTE CALLER. THE <OPTIONS> AVAILABLE ARE DEFINED IN THE SYSTEM/ DCSTATUS DOCUMENTATION.

```
DP REMOTE (STATION 4; TERMINAL 2; LINE 1, 1, 12);
```

V. RELEASE CONTROL STATEMENT

MCSII WILL NOW ACCEPT CONTROL OF STATIONS FROM ANOTHER MCS AND A NEW CONTROL STATEMENT RELEASE ALLOWS THE USER TO SPECIFY THAT A STATION CURRENTLY UNDER MCSII S CONTROL MAY BE PASSED TO ANOTHER MCS.

SYNTAX:

<RELEASE STATEMENT>::=RELEASE <STATION ID> TO <MCS ID>
<MCS ID>::=<MCS NAME>/MCS NUMBER>
<MCS NAME::=<MCS IDENTIFIER>
<MCS NUMBER>::=<UNSIGNED INTEGER>

SEMANTICS:

MCSII WILL ACCEPT CONTROL OF A STATION FROM ANOTHER MCS. THE STATUS OPTIONS, READY AND ENABLED, WILL BE SET TO THE VALUES OF THE STATION STATUS FROM THE SYSTEM, INDICATED IN THE PASSING MESSAGE I. E., IF THE STATION IS NOT READY WHEN IT IS RECEIVED IT WILL BE LEFT NOT READY BY MCSII UNTIL AN ALTER STATEMENT TO CHANGE THAT ITEM IS MADE BY THE USER.

THE RELEASE CONTROL STATEMENT WILL UNCONDITIONALLY PASS THE SPECIFIED STATION TO THE SPECIFIED MCS. IF THAT MCS IS NOT CURRENTLY RUNNING THE SYSTEM WILL ATTEMPT TO FIRE IT UP.

EXAMPLE:

```
RELEASE TC5AA TO SYSTEM/CANDE;
```

VI. SM SPO COMMAND

IT IS NOW POSSIBLE TO ENTER MCSII CONTROL COMMANDS DIRECTLY FROM THE SPO. THIS HAS BEEN IMPLEMENTED USING THE SM - SEND TO MCS - SPO COMMAND.

SYNTAX:

```
<MIX INDEX> SM: <MCSII CONTROL STATEMENT>
<MIX INDEX> ::= MIX NUMBER OF MCSII STACK
<MCSII CONTROL STATEMENT>::= <CONTROL STATEMENT BLOCK>
```

REFER TO THE RELEVANT DOCUMENTATION OF MCSII FOR THE SYNTAX OF <CONTROL STATEMENT BLOCK>

EXAMPLE.

```
713 SM : ATTACH M332; ALTER M332 ENABLE = TRUE
713 SM : TO ALL DATACOM FINISHES IN TEN MINUTES
```

IF A SYNTAX ERROR IS DETECTED IN THE CONTROL STATEMENT, THEN THE MESSAGE ***SITE ERR*** WILL BE DISPLAYED. IF A DCWRITE ERROR IS DETECTED, THEN THE MESSAGE ***DCWRITE ERROR*** WILL BE DISPLAYED.

IN BOTH CASES, THE PRINTER FILE WILL BE OPENED AND THE RELEVANT SYNTACTIC ERROR INFORMATION WILL BE WRITTEN TO THE FILE.

NDL

D0167 NDL - INITIALIZE RETRY - 10-23-72

THIS PATCH ADDS THE ABILITY TO RESET THE RETRY BYTE TO ITS INITIAL VALUE. SYNTAX IS:

```
INITIALIZE RETRY
```

D0168 NDL - BYTE VARIABLE - 10-23-72

THIS PATCH ADDS A NEW BYTE VARIABLE TO THE EXISTING LIST: IR. THIS BYTE VARIABLE IS USED TO INTERROGATE THE CLUSTER IR REGISTER. IT IS ALSO NOW POSSIBLE TO TREAT A BYTE VARIABLE AS A BITVARIABLE BY DESIGNATING ONE BIT OF THAT BYTE VARIABLE:

```
<BIT VARIABLE>::= <BYTE VARIABLE> [<BIT DESIGNATOR>]
```

EXAMPLE:

```
TALLY [1][8] = TRUE
   IF IR[9] THEN

AUX (LINE(TALLY[0])) [15] =
   LINE (TOG[0]).
```

THE NEW READ ONLY BYTE VARIABLE, IR, WILL ENABLE THE USER TO INTERROGATE THE "INPUT REGISTER" OF THE DCP. IF IR IS STORED INTO ANOTHER BYTE VARIABLE ONLY BITS 0-7 ARE STORED. IR BIT DESIGNATORS MAY BE 0-9.

D0169 NDL - SWITCH GO TO STATEMENT - 10-23-72

THE SWITCH GO TO STATEMENT ALLOWS THE USER TO BRANCH TO A LABEL

D0169 NDL - SWITCH GO TO STATEMENT - 10-23-72

DEPENDING ON THE VALUE OF THE SPECIFIED BYTE VARIABLE.

<SWITCH GO TO STATEMENT>::=GO TO <BYTE VARIABLE>,
(<LABEL LIST>).

<LABEL LIST>::=<LABEL>/<LABEL>,<LABEL LIST>

EXAMPLE:

GO TO TALLY[0], (7, 10, 1).

THE ABOVE STATEMENT WOULD EFFECTIVELY GENERATE THE FOLLOWING:

IF TALLY[0] EQUALS 0 THEN GO TO 7.
IF TALLY[0] EQUALS 1 THEN GO TO 10.
IF TALLY[0] EQUALS 2 THEN GO TO 1.

IF THE VALUE OF THE <BYTE VARIABLE> IS OUT OF RANGE, PROCESSING WILL CONTINUE AT THE NEXT STATEMENT.

THE COMMA IMMEDIATELY FOLLOWING THE BYTE VARIABLE IS OPTIONAL.

D0170 NDL - CARRIAGE CONTROL - 10-30-72

THIS PATCH DOES THE FOLLOWING:

1. DELETES THE VARIANT TOGGLE "SKIPLINES" WHICH WAS NOT IMPLEMENTED CORRECTLY AND WAS MISLEADING;

2. ADDS VARIANT TOGGLE "SKIP" WHICH IS SET BY LOGICAL I/O OR MCS-S TO INDICATE SKIP TO CHANNEL ACTION;

3. ADDS VARIANT TOGGLE "SPACE" WHICH IS SET BY LOGICAL I/O OR MCS-S TO INDICATE SPACE LINES ACTION;

4. ADDS VARIANT TOGGLE "TAB" WHICH MAY BE SET BY MCS TO INDICATE TABULATION;

5. ADDS BYTE VARIABLE "SKIP CONTROL" WHICH CONTAINS THE NUMBER OF SPACES OR THE CHANNEL NUMBER TO SKIP TO. IT CAN BE SET BY LOGICAL I/O OR MCS-S.

D0272 NDL = NDL & DCPPROGEN UNITE = 10-30-72

D0272 NDL = NDL & DCPPROGEN UNITE = 10-30-72

NDL NOW CALLS DCPPROGEN AS A PROCEDURE RATHER THAN RUNNING IT AS A PROGRAM. THUS, DCPPROGEN MUST BE BOUND TO NDL PRIOR TO EXECUTION. ONLY ONE PRINTER FILE IS NOW PRODUCED BY NDL/DCPPROGEN AND SYNTAX ERRORS IN EITHER PORTION WILL CAUSE "SNTX" NOTIFICATION. SYSTEM/DCPCODE AND SYSTEM/NIF ARE NOT LOCKED IF SYNTAX ERRORS OCCUR IN EITHER PHASE. SYSTEM/REQUESTIMAGE IS NEVER LOCKED.

THE METHOD OF CREATING SYSTEM/NDL HAS NOW CHANGED. AFTER COMPILING SYSTEM/DEPPROGEN AND SYSTEM/NDL WITH ALGOL THE BINDER MUST BE RUN TO BIND SYSTEM/DCPPROGEN INTO SYSTEM/NDL. WHEN BINDING, A STACK CARD ?STACK=2000 MUST BE PUT IN THE CONTROL DECK TO AVOID STACK OVERFLOW. THUS A PROPER BIND DECK WOULD BE:

```
?BIND SYSTEM/NDL BINDER LIBRARY
?BINDER FILE HOST = SYSTEM/NDL
?STACK=2000
?DATA
BIND DEPPROGEN FROM SYSTEM/=;STOP
?END
```

PACKDIR

D0138 PACKDIR = 09-25-72

PART I: HOW TO USE PACKDIR

PACKDIR CAN BE USED TO LIST DISKPACK AND NATIVE MODE DISK DIRECTORIES MUCH IN THE WAY LISTDIRECTORY LISTS HEAD-PER-TRACK FILES. THE PARAMETER PASSED TO PACKDIR SPECIFIES THE DIRECTORY TO BE LISTED AND THE FORMAT OF THE LISTING.

INPUT PARAMETER

THE INPUT SHOULD BE A STRING OF CHARACTERS ENCLOSED IN QUOTES. THIS STRING IS MADE UP OF "KEYWORD SUBSTRINGS" (CONTAINING LETTERS, DIGITS, SLASHES, AND EQUAL SIGNS) THAT ARE SEPARATED BY BLANKS AND/OR COMMAS. THAT IS, A PARTICULAR KEY PHRASE MAY NOT CONTAIN ANY EMBEDDED BLANKS BUT PHRASES MAY BE SEPARATED BY AN ARBITRARY NUMBER OF BLANKS AND COMMAS. THE KEY PHRASES CAN OCCUR IN ANY ORDER ON THE RUN CARD.

THE FOLLOWING KEY PHRASES ARE ACCEPTED:

MAP — SPECIFIES THAT THE DISK CHECKERBOARD (A DISPLAY OF ALLOCATED AND AVAILABLE SEGMENTS) IS TO BE PRINTED. DEFAULT IS NOMAP.

NOMAP — SUPPRESSES THE CHECKERBOARD LISTING (DEFAULT VALUE).

DISK — SPECIFIES THAT THE DIRECTORY TO BE LISTED IS ON A HEAD-PER-TRACK DISK UNIT (I.E., KIND = DISK). DEFAULT IS ON DISK PACK (KIND = PACK).

NAME = ABC/DEF/.../XYZ
THIS INDICATES THE NAME OF THE DIRECTORY TO BE LISTED. IN THE CASE OF DISK PACKS, THE FIRST QUALIFIER IS USED FOR THE PACK NAME. IF THE WHOLE PACK IS TO BE

LISTED, JUST CODE NAME = PACKNAME.

ABC/DEF/.../XYZ

THIS IS THE SAME AS NAME = ABC/DEF/.../XYZ, ONLY THE NAME CANNOT BE A RESERVED KEYWORD (SEE ABOVE).

OUTPUT LISTING

BASICALLY, THE SAME FOUR LISTINGS PROVIDED BY LISTDIRECTORY ARE PRODUCED BY PACKDIR. HOWEVER, CERTAIN ADDITIONAL DIAGNOSTIC OUTPUTS ARE AVAILABLE WITH PACKDIR.

MAIN LISTINGS: THE FIRST TWO LISTINGS ARE THE MAIN OUTPUT PRODUCED. THE FIRST LISTING IS A DISPLAY OF ALL THE FILE NAMES IN THE DIRECTORY GIVEN BY NAME=. THIS LISTING IS FORMATTED AS A TREE STRUCTURE TO INDICATE THE VARIOUS SUBLEVELS OF THE FILES IN THE DIRECTORY. IT IS PRINTED IN THE SAME ORDER AS THE FILES OCCUR IN THE DIRECTORY. THE SECOND LISTING INDICATES LOCATIONS ON THE HEADER AND ALL THE ROWS OF EACH FILE GIVEN IN THE FIRST LISTING. THE "LOCATION" INCLUDES THE UNIT NUMBER AND SEGMENT ADDRESS. UNFORTUNATELY, THE "UNIT NUMBER" PRINTED FOR THE HEADERS IS ALWAYS A ONE (THE BASE PACK INDEX NUMBER) RATHER THAN THE ACTUAL ELECTRONICS NUMBER.

MAP LISTINGS: THESE TWO LISTINGS (PRODUCED ONLY IF MAP IS SPECIFIED) GIVE AN INDICATION OF HOW MUCH SPACE IS AVAILABLE AND HOW MUCH SPACE IS IN USE ON THE VOLUMES BEING MAPPED. IF NAME= SELECTS ONLY A SUBDIRECTORY, THESE LISTINGS ARE MISLEADING. FURTHERMORE, NO ACCOUNT IS TAKEN OF THE VOLUME LABELS ON THE DISK PACKS. FINALLY, THE HEADERS ARE INCORRECTLY INDICATED AS BEING ON UNIT ONE, CAUSING THE CHECKERBOARD TO BE INCORRECTLY ANALYZED.

THE USE OF THE OPERATOR COMMAND DIR

THE OPERATOR CAN INVOKE ANY ONE OF THREE DIRECTORY LISTING PROGRAMS WITH THE KEYBOARD COMMAND DIR. IF HE SIMPLY INPUT DIR WITH NO PARAMETERS, LISTDIRECTORY IS CALLED TO LIST THE MASTER HEAD-PER-TRACK (HPT) DIRECTORY. IN ORDER TO LIST THE DIRECTORY OF AN

"INTERCHANGE DISK PACK, HE SHOULD INPUT:

DIR IC, PKNN

OR

DIR IC, PACKNAME

IN THIS CASE, LISTPACK IS CALLED. IT IS IMPORTANT THAT "IC" APPEAR FIRST, WITHOUT ANY IMBEDDED BLANKS. ANY OTHER INPUT CAUSES KEYIN TO PLACE QUOTE MARKS AROUND THE PARAMETER AND PASS THE ENTIRE STRING TO PACKDIR. NOTICE THAT THE OPERATOR SHOULD NOT INPUT THE QUOTES.

PART II: LOGIC DESCRIPTION OF PACKDIR

PACKDIR WAS DERIVED FROM LISTDIRECTORY, AND ITS BASIC FLOW IS MUCH THE SAME. FILES USED BY PACKDIR ARE:

LINE - ALL PRINTED OUTPUT IS PRODUCED ON THIS FILE WITH A RECORDSIZE OF 132 CHARACTERS (22 WORDS).

INFO - AN INTERMEDIATE WORK FILE THAT IS USED TO STORE THE HEADER AND ROW LOCATIONS. INFO IS ACCESSED SEQUENTIALLY WITH A RECORD SIZE OF 30 CHARACTERS (5 WORDS) AND IS PURGED AT THE END OF A JOB.

D - A FILE THAT IS USED TO OPEN THE MASTER DIRECTORY AND ALL THE FILES AND SUBFILES IN THAT DIRECTORY.

FLOW OF PACKDIR

AFTER THE INPUT PARAMETER STRING IS ANALYZED, PACKDIR OPENS THE SELECTED DIRECTORY AND MAKES UP TO FOUR PASSES:

FIRST PASS - DURING THIS PASS, EACH FILE IS OPENED, ITS ROW AND HEADER LOCATIONS ARE SAVED ON INFO, AND ITS NAME DISPLAYED ON LINE. THE MAIN ROUTINE FOR THE FIRST PASS IS PROCEDE, WHICH IS CALLED RECURSIVELY TO HANDLE SUBFILES. (SINCE THERE IS NO WAY, CURRENTLY, TO SIMPLY READ THE HEADERS OF FILES ON DISK PACKS, AS

LISTDIRECTORY DOES FOR HPT DISK FILES, PACKDIR MUST OPEN ALL THE FILES TO OBTAIN THE ROW INFORMATION.)

SECOND PASS — DURING THIS PASS, THE ROW LOCATIONS ARE READ BACK FROM INFO, DISPLAYED ON LINE AND, IF MAP IS SELECTED, INPUT TO THE SORT. THE MAIN ROUTINES FOR THIS PASS ARE IF, WHICH READS THE RECORDS FROM INFO AND SENDS THEM TO THE SORT, AND AREAMAP, WHICH PRINTS THE HEADER AN ROW ADDRESSES.

THIRD PASS — (ONLY IF MAPPING) DURING THIS PASS, WHICH IS THE OUTPUT OF THE SORT ON ROW LOCATIONS, "AREAS THAT CAN BE MADE AVAILABLE WITH THE REMOVAL OF ONE FILE" ARE DISPLAYED ON LINE AND THE SORTED ROW LOCATIONS ARE SAVED ON INFO. (UNFORTUNATELY, THESE ARE MISLEADING BECAUSE THE HEADER AND ROW UNIT NUMBERS ARE INCOMPATIBLE.) THE ROUTINE OP RETRIEVES THE RECORDS FROM THE SORT, SAVES THEM ON INFO, AND CALLS PRINT TO PRODUCE THE REPORT.

LAST PASS — (ONLY IF MAPPING) DURING THIS PASS, THE SORTED ROW LOCATIONS ARE READ BACK FROM INFO, AND THE ROUTINE LISTIT IS CALLED TO PRINT THE CHECKERBOARDS OF THE VARIOUS UNITS (THESE AGAIN ARE WRONG BECAUSE OF THE UNIT PROBLEM).

PL/I

D0142 PLI - PLI IO IMPROVEMENTS - 02-14-73

THERE HAS BEEN A GENERAL IMPROVEMENT OF THE STREAM I/O FEATURES IN PL1. A FULL EXPLANATION OF THE FUNCTION OF MOST OF THESE FEATURES CAN BE OBTAINED FROM THE PL/I LANGUAGE INFORMATION MANUAL (5000201).

OPEN STATEMENT

THE FOLLOWING OPEN OPTIONS HAVE BEEN IMPLEMENTED:

1. PAGESIZE (<SCALAR-EXPRESSION>)
2. LINESIZE (<SCALAR-EXPRESSION>)

   WHERE THE EXPRESSION REPRESENTS LINE SIZE, IN CHARACTERS, FOR A STREAM OUTPUT FILE.

A NEW OPEN OPTION HAS BEEN ADDED FOR STREAM OUTPUT FILES:

TAB (<SCALAR-EXPRESSION>, ...)

WHERE THE <SCALAR-EXPRESSION> LIST, DEFINES THE TAB COLUMNS TO BE USED FOR LIST AND DATA DIRECTED PUT STATEMENTS, OR FOR TAB FORMAT ITEMS.

THE EXPRESSIONS MUST BE ASCENDING, POSITIVE AND LESS THAN THE FILE LINE SIZE. ALL ERRONEOUS EXPRESSIONS WILL BE IGNORED.

EXAMPLE:

```
OPEN FILE (SYSPRINT) LINESIZE (132) TAB (20,40,105);
```

IF USER TAB SETTINGS ARE NOT SUPPLIED, DEFAULT TABS WILL BE USED (1, 25, 49, 73, 47, 121, ...).

BUILTIN FUNCTION

THESE STREAM-IO-RELATED BUILTIN FUNCTIONS ARE NOW IMPLEMENTED:

1. COUNT (<FILE-NAME>)
2. LINENO (<FILE-NAME>)

EDIT-DIRECTED FORMATS

THE FOLLOWING FORMAT ITEMS HAVE BEEN IMPLEMENTED:

1. LINE (<SCALAR-EXPRESSION>)

   WHERE THE EXPRESSION REPRESENTS THE NEXT LINE NUMBER TO BE ADVANCED TO.

2. B-FORMAT ITEM (BIT-STRING FORMAT)
3. P-FORMAT ITEM (PICTURE FORMAT)
4. TAB-FORMAT ITEM

   WHICH CAUSES MOVEMENT TO THE NEXT USER OR DEFAULT TAB SETTING, RELATIVE TO THE CURRENT COLUMN.

DATA-DIRECTED IO

GET-DATA AND PUT-DATA STATEMENTS ARE NOW FULLY IMPLEMENTED.

TYPE CONVERSION

COMPLETE DATA TYPE CONVERSIONS ARE NOW PERFORMED FOR DATA-LIST AND STREAM ITEMS.

EXAMPLE:

```
DCL CS CHAR(6) INITIAL ('0123.4');
   PUT EDIT(CS) (F(10));
```

BIT STRINGS ARE NOW ALLOWED IN INPUT DATA STREAMS.

WARNING:

THE DEFAULT "UNITS" ATTRIBUTE FOR STREAM FILES WILL BE CHARACTERS. THEREFORE A FILE DECLARATION FOR A PRINT FILE:

```
   DCL FILE (PRINT) ENV (MAXRECSIZE=22);
```

WILL NOT PRODUCE THE DESIRED RESULT THE USER SHOULD SPECIFY UNITS = -WORDS- OR MAXRECSIZE = 132, TO GET THE PROPER RECORD SIZE. (NOTE: THE DASH HAS BEEN USED FOR THE SINGLE QUOTE.)

D0143 PLI - PLI BINDING - 02-19-73

PL/I TO PL/I BINDING IS NOW IMPLEMENTED. THE BINDING PROCESS WILL GENERALLY CONSIST OF BINDING A GROUP OF EXTERNAL PROCEDURES TO A "HOST" PROCEDURE. COMMUNICATION BETWEEN THE PROCEDURES IS PERFORMED THROUGH COMMON EXTERNAL DECLARATIONS WITHIN THE PROCEDURES AND PARAMETERS. ANY PL/I PROCEDURE WITHOUT PARAMETERS MAY BE DESIGNATED AS THE "HOST" PROCEDURE.

TO BIND A SEPARATE PROCEDURE TO A HOST, THE SEPARATE PROCEDURE MUST BE DECLARED EXTERNAL IN THE HOST.

```
HOST:PROC;
   DCL SEPARATE ENTRY (CHAR(*)) EXTERNAL;
   DCL CHAR CHAR(8) INIT(-ABCDEFGH-);
   CALL SEPARATE(CHAR);
END HOST;

SEPARATE:PROC(C);
   DCL C CHAR(*);
   PUT LIST(C);
END SEPARATE;
```

(NOTE THAT THE DASH HAS BEEN USED FOR THE SINGLE QUOTE)

IF THE NAME OF THE CODE FILE OF "HOST" IS HOST/HOST AND THE NAME OF THE CODE FILE OF "SEPARATE" IS SEPARATE/SEPARATE THEN THE BIND DECK IS:

```
   ?BIND BOUND WITH BINDER LIBRARY
   ?BINDER FILE HOST = HOST/HOST
   ?DATA
   BIND SEPARATE FROM SEPARATE/SEPARATE;
    STOP
```

```
?END
```

WHEN "BOUND" IS RUN, THE RESULT IS THE CHARACTER STRING -ABCDEFGH- ON THE SYSPRINT PRINT FILE.

RESTRICTIONS

1. ANY EXTERNAL ENTRY CONSTANT PASSED A PARAMETER CAN ONLY BE BOUND TO A HOST. (THE PROCEDURE "SEPARATE" IS ONLY VALID WHEN BOUND. IF THE CODE FILE SEPARATE/SEPARATE IS RUN, AN INVALID OPERATOR WILL RESULT.) ONLY A PROCEDURE WITH NO PARAMETERS MAY BE USED AS A HOST.

2. THE FIRST EXTERNAL ENTRY CONSTANT MUST BE SPECIFIED IN THE BIND DECK.

EXAMPLE:

```
HH:PROC;
   DCL SEP1 ENTRY EXT,
       SEP2 ENTRY (CHAR(*))EXT;
   CALL SEP2(-ABCDEFG-);
END HH;
SEP1:PROC;
SEP2:ENTRY(C);
   DCL C CHAR(*);
   PUT LIST(C);
END SEP1;
```

(NOTE THAT THE DASH HAS BEEN USED FOR THE SINGLE QUOTE)

ASSUMING THE CODE FILE NAME FOR HH IS H, THE BIND SHOULD BE AS FOLLOWS:

```
?BIND B BINDER LIBRARY
?BIND FILE HOST = H
?DATA
   BIND SEP1 FROM S;
?END
```

WHEN B IS RUN, THEN THE RESULT WILL BE THE CHARACTER STRING

"ABCDEFG" ON SYSPRINT.

HANDLING OF STATIC EXTERNAL:

IF A VARIABLE IS DECLARED STATIC EXTERNAL IN BOTH THE HOST AND THE SEPARATE PROCEDURE, THE INITIAL VALUES IN THE HOST ARE THE ONES USED WHEN BOUND. IF A VARIABLE IS DECLARED STATIC EXTERNAL IN ONLY THE SEPARATE PROCEDURE, THE INITIAL VALUES OF THAT VARIABLE ARE USED.

EXAMPLE

```
HOST:PROC;

   DCL A(4) STATIC EXTERNAL
       INIT (1,2,3,4),
       SEPARATE ENTRY EXTERNAL;
   CALL SEPARATE;
END HOST;
SEPARATE:PROC;
   DCL A(4) STATIC EXTERNAL INIT(5,6,7,8),
       A1(4) STATIC EXTERNAL INIT(9,10,11,12,);
   PUT DATA (A,A1);
END SEPARATE;
```

WHEN BOUND AND RUN THE RESULT WILL BE

```
   A(1) = 1,  A(2) = 2,   A(3) = 3,   A(4) = 4,
  A1(1) = 9, A1(2) = 10, A1(3) = 11, A1(4) = 12;
```

NOTE THAT ANY EXTERNAL STATIC, CONTROLLED OR BASED VARIABLE SHOULD BE INITIALIZED BEFORE IT IS USED IN A DECLARATION:

EXAMPLE:

```
   DCL
   1 S1(X) STATIC,
      2 A(X) INIT(B(1),B(2),B(3),B(4)),
   1 S2 STATIC,
      2 B(2*X) INIT(C(1),C(2),C(3),C(4)),
   1 S3 STATIC,
```

D0143 PLI - PLI BINDING - 02-19-73

```
   2 C(2*X) INIT(1,2,3,4),
X STATIC INIT(2);
```

SHOULD BE DECLARED IN THIS ORDER:

```
DCL
X STATIC INIT(2),
1 S3 STATIC,
   2 C(2*X) INIT(1,2,3,4),
1 S2 STATIC,
   2 B(2*X) INIT(C(1),C(2),C(3),C(4)),
1 S1(X) STATIC,
   2 A(X) INIT(B(1),B(2),B(3),B(4));
```

VARIABLES WHOSE ORDER OF DECLARATION WILL CAUSE THE PROGRAM TO RUN INCORRECTLY WHEN BOUND WILL GET A LEVEL THREE ERROR MESSAGE.

NOBINDINFO CONTROL CARD OPTION

IF NOBINDINFO IS SET, THE COMPILER WILL NOT PUT INFORMATION FOR THE BINDER IN THE CODE FILE. NOBINDINFO IS RESET BY DEFAULT.

D0155 PLI - PACKED PICTURES - 01-15-73

A NEW PICTURE CLASSIFICATION HAS BEEN ADDED TO THE PLI LANGUAGE. THE CLASSIFICATION IS PACKED PICTURES AND ALLOWS THE FOLLOWING PICTURE CHARACTERS IN A PICTURE ATTRIBUTE:

- H SPECIFIES THAT THE ASSOCIATED POSITION WILL CONTAIN A 4-BIT PACKED DECIMAL DIGIT
- S SPECIFIES THAT THE ASSOCIATED POSITION WILL CONTAIN A PACKED 4-BIT SIGN (EITHER 1101 OR 1100)
- V SPECIFIES THE IMPLIED PACKED DECIMAL POINT

EXAMPLE:

```
DCL P1 PICTURE "HHHVHHS";
```

D0155 PLI - PACKED PICTURES - 01-15-73

THE PRECISION OF P1 IS (5,2) AND P1 WILL OCCUPY SIX 4-BIT BYTES INTERNALLY.

A USER SHOULD USE PACKED PICTURES INSTEAD OF THE DECIMAL FIXED ATTRIBUTES TO INSURE THE CORRECT INTERNAL REPRESENTION FOR NON-BURROUGHS PL1 PROGRAMS.

D0174 PLI - ENTRY VARIABLES IMPLEMENTED - 10-30-72

ENTRY VARIABLES ARE NOW IMPLEMENTED. AT THIS TIME, THERE IS NO PARAMETER CHECKING.

D0175 PLI - EXPLICIT ATTRIBUTE IMPLEMENTED - 10-30-72

A DEFAULT DECLARATION OF DFT (NOT EXPLICIT) ERROR WILL CAUSE AN ERROR MESSAGE TO BE GENERATED FOR ANY IDENTIFIER NOT EXPLICITLY DECLARED.

D0225 PLI - PLI FILE DECLARATIONS - 12-18-73

FILE DECLARATIONS IN PL/I HAVE BEEN EXPANDED AND IMPROVED.

PL/I LANGUAGE FILE ATTRIBUTES

SYNTAX CHECKING OF PL/I FILE ATTRIBUTES HAS BEEN EXTENDED. THE FOLLOWING ATTRIBUTES HAVE NOT BEEN IMPLEMENTED AND A SYNTAX ERROR WILL BE GENERATED IF AN ATTEMPT IS MADE TO SPECIFY THEM:

BACKWARDS, KEYED, EXCLUSIVE, UPDATE, DIRECT

ATTEMPTING TO SET CONFLICTING PL/I ATTRIBUTES EXCEPT EXCLUSIVE WILL NOW GENERATE A SYNTAX ERROR.

THE FOLLOWING IS A LIST OF PL/I FILE ATTRIBUTES AND THE ATTRIBUTES THAT DO NOT CONFLICT WITH THEM.

STREAM	INPUT, OUTPUT PRINT
RECORD	INPUT, OUTPUT, UPDATE, SEQUENTIAL, DIRECT, BACKWARDS, KEYED
INPUT	STREAM, RECORD, SEQUENTIAL, DIRECT, BACKWARDS, KEYED
OUTPUT	STREAM, RECORD, SEQUENTIAL, DIRECT, PRINT, KEYED
UPDATE	RECORD, SEQUENTIAL, DIRECT, KEYED
SEQUENTIAL	RECORD, INPUT, OUTPUT, UPDATE, BACKWARDS, KEYED
DIRECT	INPUT, OUTPUT, UPDATE, KEYED
BACKWARDS	RECORD, INPUT, SEQUENTIAL
PRINT	STREAM, OUTPUT
KEYED	RECORD, INPUT, OUTPUT, UPDATE, SEQUENTIAL, DIRECT

PL/I SYSTEM FILE ATTRIBUTES

FILE OPTIONS OR ENVIRONMENT SECTION CURRENT SYNTAX:

1. THE CURRENT SYNTAX FOR KIND, BUFFERS, SPACE, MAXRECSIZE AND SAVEFACTOR WILL STILL BE RECOGNIZED.
2. BLOCKING WILL NO LONGER BE RECOGNIZED. BLOCKSIZE SHOULD BE USED.
3. THE SYNTAX FOR TITLE HAS BEEN CHANGED. SEE TITLE UNDER THE NEW SYNTAX.

INSIDE A FILE OPTION OR ENVIRONMENT DECLARATION A SYSTEM FILE ATTRIBUTE HAS ONE OF FOUR SPECIFICATIONS.

1. NUMERIC REQUIRES A DECIMAL INTEGER-CONSTANT GREATER THAN ZERO SPECIFICATION, E.G. MAXRECSIZE = 10, AREAS = 2,...
2. MNEMONIC REQUIRES A CHARACTER-STRING MNEMONIC ENCLOSED IN SINGLE QUOTES, E.G. KIND = 'DISK', MYUSE = '

OUT=,...

3. STRING REQUIRES A CHARACTER STRING ENCLOSED IN SINGLE QUOTES, E.G. FORMMESSAGE = 'USE FOR 1040',...

4. SUBTITLE IS A STRING VALUED ATTRIBUTE THAT HAS SOME SYNTAX RESTRICTIONS. SUBTITLES WITH SLASHES MUST BE ENCLOSED IN DOUBLE QUOTES AND DOUBLE QUOTES MAY ONLY APPEAR IN PAIRS ENCLOSING AN ENTIRE SUBTITLE, E.G. TITLE = 'A/"B/C"/D',

5. NULL VALID ONLY FOR ATTRIBUTES WITH THE MNEMONICS OF 'TRUE' AND 'FALSE'. THE ATTRIBUTE WILL BE SET TO TRUE. E.G. OPTIONAL,...IS THE SAME AS OPTIONAL ='TRUE',...

(NOTE THAT THE DASH HAS BEEN USED FOR THE SINGLE QUOTE.)

SEMANTICS:

```
<FILE OPTIONS DECLARATION>::=
   OPTIONS(<INITIAL SYSTEM FILE ATTRIBUTE LIST>)
<FILE ENVIRONMENT DECLARATION>::=
   ENVIRONMENT(<INITIAL SYSTEM FILE ATTRIBUTE LIST>)
<INITIAL SYSTEM FILE ATTRIBUTE LIST>::=
   <INITIAL SYSTEM FILE ATTRIBUTE>/
   <INITIAL SYSTEM FILE ATTRIBUTE LIST>,
         <INITIAL SYSTEM FILE ATTRIBUTE>
<INITIAL SYSTEM FILE ATTRIBUTE>::=
   <INTEGER VALUE SYSTEM FILE ATTRIBUTE> = <INTEGER-CONSTANT>/
   <STRING VALUED SYSTEM FILE ATTRIBUTE> = <STRING>/
   <REAL VALUED SYSTEM FILE ATTRIBUTE> = <INTEGER-CONSTANT>/
   <BOOLEAN VALUED SYSTEM FILE ATTRIBUTE> = '<BOOLEAN MNEMONIC>'/
   <BOOLEAN VALUED SYSTEM FILE ATTRIBUTE>/
   DENSITY = '<DENSITY MNEMONIC>'/
   PARITY = '<PARITY MNEMONIC>'/
   KIND = '<KIND OPTION>'/
   LABELTYPE = '<LABELTYPE MNEMONIC>'/
   EXTMODE = '<EXTMODE MNEMONIC>'/
```

```
   PROTECTION = -<PROTECTION MNEMONIC>-/
   MYUSE = -<MYUSE MNEMONIC>-/
   OTHERUSE = -<OTHERUSE MNEMONIC>-/
   SPEED = -<SPEED MNEMONIC>-/
   DIRECTION = -<DIRECTION MNEMONIC>-/
   SIZEMODE = -<SIZEMODE MNEMONIC>-/
   CARRIAGECONTROL = -<CARRIAGECONTROL MNEMONIC>-/
   FILEKIND = -<FILEKIND MNEMONIC>-/
   UNITS = -<UNITS MNEMONIC>-/UNITS/
   SECURITYTYPE = -<SECURITYTYPE MNEMONIC>-/
   SECURITYUSE = -<SECURITYUSE MNEMONIC>-
```

(PLEASE NOTE THAT THE DASH HAS BEEN USED FOR THE SINGLE QUOTE.)

```
<INTEGER VALUED SYSTEM FILE ATTRIBUTE>::=
   REEL/DATE/CYCLE/VERSION/SAVEFACTOR/FILETYPE/BLOCKSIZE/
   MAXRECSIZE/MINRECSIZE/AREASIZE/AREAS/BUFFERS/SIZEOFFSET/
   SIZE2/  PAGESIZE/PAGE/LINENUM/COPIES/UNITNO/AREACLASS/
   LASTSTATION
<REALVALUED SYSTEM FILE ATTRIBUTE>::=
   TIMELIMIT
<BOOLEAN VALUED SYSTEM FILE ATTRIBUTE>::=
   OPTIONAL/FLEXIBLE/OPEN/SINGLEPACK/CYLINDERMODE/CODEFILE/
   INTERCHANGE/IC/DUPLICATED/READCHECK
<STRING VALUED SYSTEM FILE ATTRIBUTE>::=
   TITLE/PACKNAME/FORMMESSAGE/FORMESSAGE/INTNAME/
   SECURITYGUARD/FAMILY
<DENSITY MNEMONIC>::=
   HIGH/MEDIUM/LOW/SUPER
<PARITY MNEMONIC>::=
   STANDARD/NONSTANDARD
<KIND OPTION>::=
   <DEVICE>/
   <BACKUP DEVICE><BACKUP OPTION><BACKUP UNIT>/
   <BACKUP DEVICE><BACKUP OPTION>/
   <BACKUP OPTION><BACKUP UNIT>/
   <BACKUP OPTION>
```

```
<DEVICE>::=
   <DISPLAY MNEMONIC>/
   <REMOTE MNEMONIC>/
   <PAPER READER MNEMONIC>/
   <PAPER PUNCH MNEMONIC>/
   <CARD READER MNEMONIC>/
   <DISK PACK MNEMONIC>/
   <CARD PUNCH MNEMONIC>/
   <PRINTER MNEMONIC>/
   <DISK MNEMONIC>/
   <TAPE MNEMONIC>
<BACKUP DEVICE>::= <CARD PUNCH MNEMONIC>/
   <PRINTER MNEMONIC>
<BACKUP UNIT>::= <DISK MNEMONIC>/<TAPE MNEMONIC>
<DISPLAY MNEMONIC>::= SPO/SPOS/DISPLAY
<REMOTE MNEMONIC>::= REMOTE/DC
<PAPER READER MNEMONIC>::= PAPER/PAPERREADER/PTR/PAPER READER
<PAPER PUNCH MNEMONIC>::= PAPERPUNCH/PTP/PAPER PUNCH
<CARD READER MNEMONIC>::= READER/READERS
<DISK PACK MNEMONIC>::= DISKPACK/DISPACKS/PACK/PACKS
<CARD PUNCH MNEMONIC>::= PUNCH/PUNCHES/CP
<PRINTER MNEMONIC>::= PRINTER/PRINTERS/LP
<DISK MNEMONIC>::= DISK/SERIAL
<TAPE MNEMONIC>::= TAPE/TAPES/TAPE9/TAPE9S/PETAPE/PETAPES
<BACKUP OPTION>::= BACKUP/BACK UP
<STRING>::= <CHARACTER STRING CONSTANT>
<BOOLEAN MNEMONIC>::= TRUE/FALSE
<LABELTYPE MNEMONIC>::= STANDARD/OMITTED/OMITTEDEOF
<EXTMODE MNEMONIC>::= SINGLE/DOUBLE/HEX/BCL/EBCDIC/ASCII
<PROTECTION MNEMONIC>::= TEMPORARY/SAVED/PROTECTED
<MYUSE MNEMONIC>::= CLOSED/IN/OUT/IO
<OTHERUSE MNEMONIC>::= SECURED/IN/OUT/IO
<SPEED MNEMONIC>::= FAST/MEDIUMFAST/MEDIUMSLOW/SLOW
<DIRECTION MNEMONIC>::= FORWARD/REVERSE
<SIZEMODE MNEMONIC>::= <EXTMODE MNEMONIC>
<CARRIAGECONTROL MNEMONIC>::= STANDARD/CTLASA/CTL360
```

<FILEKIND MNEMONIC>::=

ALGOLCODE / ALGOLSYMBOL / BACKUPDISK / BASICCODE / BASICSYMBOL / BINDERSYMBOL / BOUNDCODE / COBOLCODE / COBOLSYMBOL / CODEFILE / COMPILERCODEFILE / CONTROLDECK / DATA / DCALGOLCODE / DCALGOLSYMBOL / DIRECTORY / ESPOLCODE / ESPOLSYMBOL / FORTRANCODE / FORTRANSYMBOL / GUARDFILE / INTRINSICFILE / JOVIALCODE / JOVIALSYMBOL / LIBRARYCODE / MCPCODEFILE /PLICODE / PLISYMBOL / RECONSTRUCTIONFILE / SEQDATA / SYSTEMDIRECTORY / SYSTEMDIRFILE / VERSIONDIRECTORY / XALGOLCODE / XALGOLSYMBOL / XDISKFILE / XFORTRANCODE / XFORTRANSYMBOL

<UNITS MNEMONIC>::= <BOOLEAN MNEMONIC>/WORDS/CHARACTERS
<SECURITYTYPE MNEMONIC>::= PRIVATE/CLASSA/CLASSB/CLASSC
<SECURITY USE MNEMONIC>::= SECURED/IN/OUT/IO/READONLY/
WRITEONLY/READWRITE

THE COMPILER WILL SET SOME DEFAULT ATTRIBUTE SPECIFICATIONS FOR A FILE DECLARATION. THESE MAY BE OVERRIDDEN BY SPECIFYING THEM IN THE FILE OPTIONS LIST.

1. SAVEFACTOR IS SET TO ONE
2. UNITS IS SET TO CHARACTERS
3. MYUSE IS SET TO OUT IF A FILE IS DECLARED OUTPUT OR PRINT.
4. MYUSE IS SET TO IN IF A FILE IS DECLARED INPUT
5. MAXRECSIZE WILL BE SET IF NO FILE OPTIONS LIST IS SPECIFIED. IT WILL BE SET TO 132 IF THE FILE NAME IS SYSPRINT OR THE FILE IS DECLARED PRINT, OTHERWISE IT WILL BE SET TO 80.
6. AREAS WILL BE SET TO 20 AND AREASIZE TO 1000 IF KIND HAS BEEN SPECIFIED TO BE DISK.
7. IF KIND HAS NOT BEEN SPECIFIED AN ATTEMPT WILL BE MADE TO ASSIGN IT A VALUE.
   IF FILE NAME IS SYSPRINT, KIND WILL BE SET TO REMOTE IF THE COMPILE IS A REMOTE ENTRY, OTHERWISE IT WILL BE SET TO PRINTER.
   IF THE FILE NAME IS SYNIN, KIND WILL BE SET TO REMOTE IF

D0225 PLI - PLI FILE DECLARATIONS - 12-18-73

THE COMPILE IS A REMOTE ENTRY, OTHERWISE IT WILL BE SET TO READER.
IF THE FILE IS DECLARED PRINT THEN KIND WILL BE SET TO PRINTER.
IF THE FILE IS DECLARED OUTPUT BUT NOT PRINT THEN KIND WILL BE SET TO DISK.
IF KIND IS SPECIFIED IN THE FILE OPTIONS LIST, THE COMPILER SYNTAXES IT TO SEE THAT IT AGREES WITH OTHER SPECIFICATIONS.

A. IF A FILE IS DECLARED PRINT, KIND MUST BE SET TO PRINTER.
B. IF A FILE IS DECLARED INPUT, KIND MAY NOT BE SET TO A DEVICE APPROPRIATE ONLY FOR OUTPUT.
C. IF A FILE IS DECLARED OUTPUT, KIND MAY NOT BE SET TO DEVICE APPROPRIATE ONLY FOR INPUT.

SYSTEM FILE ATTRIBUTE INTMODE

INTMODE IS UNCONDITIONALLY SET TO EBCDIC. ATTEMPTING TO SPECIFY IT IN A FILE OPTIONS LIST WILL GENERATE AN ERROR OF LEVEL ZERO AND THE SPECIFICATION WILL BE IGNORED.

D0246 PLI - SEPARATE COMPILES - 02-19-73

PROCEDURES MAY NOW BE COMPILED SEPARATELY IN PLI. THE CONTROL CARD OPTION "MULTIPLE", WHEN SET, CAUSES THE LAST NAME OF THE PROGRAM FILE NAME TO BE REPLACED BY THE NAME OF THE MAIN ENTRY POINT OF THE PROCEDURE. IF SEPARATE IS NOT SET, THE NAME OF THE FIRST PROCEDURE WILL BE THE PROGRAM FILE NAME AND ALL SUBSEQUENT PROCEDURES WILL HAVE THE LAST NAME OF THE PROGRAM FILE NAME REPLACED BY THE NAME OF THE MAIN ENTRY POINT OF THE PROCEDURE. IF NEW, NEW1 OR NEW2 IS SET, ALL PROCEDURES WILL BE CONTAINED IN A SINGLE NEW SYMBOLIC FILE. PROCEDURES MUST BE SEPARATED BY A CARD WITH A QUESTION MARK "?" IN TEXT COLUMN ONE.

EXAMPLE:

```
<I>COMPILE A/B/C WITH PLI LIBRARY;
<I>DATA
   "SET MULTIPLE"
   ONE: PROCEDURE;
   END ONE;
   ?
$TWO: PROCEDURE;
   END $TWO;
<I>END.
```

THE TWO CODE FILES ARE CALLED A/B/ONE AND A/B/"$TWO".

Predict sort
II.4 System Misc.

PREDICTSORT

00148 PREDICTSORT - SORT TIMING PREDICTOR - 01-15-72

SORT TIMING PREDICTION ROUTINES

A SORT UTILITY PROGRAM WRITTEN IN FORTRAN HAS BEEN INCLUDED IN THE MARK 11.4 RELEASE UNDER THE SOURCE NAME OF "SYMBOL/PREDICTSORT" AND OBJECT NAME OF "SYSTEM/PREDICTSORT". THE OUTPUT OF THIS PROGRAM IS A PREDICTION OF THE RESOURCES REQUIRED (PROCESSOR AND INPUT-OUTPUT) TO ACCOMPLISH THE SPECIFIED SORT. INPUT TO THE PROGRAM IS A RELATIVELY COMPREHENSIVE SET OF PARAMETERS THAT SPECIFY THE SORT TO BE DONE.

ACCURACY OF PROGRAM

SINCE THE B6700 SORT IS A SOPHISTICATED PROCESS, PREDICTION OF ANTICIPATED RESULTS IS, BY NECESSITY, COMPLEX. PREDICTION RESULTS SHOULD NOT BE EXPECTED TO BE 100% ACCURATE, BUT IT IS DESIRED AND EXPECTED TO BE WITHIN 5% ACCURACY FOR THE MAJORITY OF TEST CASES. A GENERAL TENDENCY TOWARD OVERESTIMATION RATHER THAN UNDERESTIMATION HAS BEEN BUILT INTO THE PROGRAM. PROCESSOR TIME ESTIMATES ARE BASED UPON A 5-10 CLOCK AND 1.2 MICROSECOND MEMORY. I/O TIME ESTIMATES ARE BASED UPON THE SPEED OF THE SPECIFIED DEVICES AND ASSOCIATED PROCESSOR ESTIMATE. ELAPSED TIME ESTIMATES ARE BASED UPON MAXIMUMS OF THE ASSOCIATED I/O AND PROCESSOR ESTIMATES. ALL PREDICTIONS ARE BASED UPON THE SORT RUNNING IN AN OPTIMAL ENVIRONMENT WHERE THE ONLY CONTENTION FOR MEMORY OR ACCESS TO A I/O DEVICE IS FOR AND BY THE SORT ITSELF. ELAPSED TIME ESTIMATES DO NOT ANTICIPATE OVERLAPPING OF I/O EXCEPT WHERE DISK IS BEING OVERLAPPED WITH TAPE FOR ITS SORTS. PLEASE NOTE THAT THERE IS NO ACCOUNTING OF ANY TYPE OF TAPE REEL SWITCHING OR ANY KIND OF ACTIVITY THAT MAY REQUIRE OPERATOR INTERVENTION.

INPUT PARAMETERS

THE USE OF NAMELIST WAS CHOSEN FOR EASE OF INPUT AND THE USER SHOULD REFER TO THE B6700 FORTRAN LANGUAGE MANUAL FOR INFORMATION REGUARDING THE USE OF NAMELIST. A WORD OF CAUTION IS IN ORDER TO EMPHASIZE THE NEED FOR CORRECT SPELLING OF PARAMETER NAMES AND THE USE OF PUNCTUATION (COMMAS AND EQUAL SIGNS). THE NAME OF THE NAMELIST RECORD IS "INFO" AND ALL ELEMENTS IN THE LIST ARE SIMPLE VARIABLES.

MORE THAN ONE SET OF INPUT DATA CAN BE FED (ONE AT A TIME) INTO THE PROGRAM. WHEN INPUT SETS ARE STACKED IN THIS FASHION, EACH SUCCEEDING SET WILL REPLACE ONLY THE VARIABLES SPECIFIED WITHIN THE SPECIFIC SUCCESSOR SET. ONCE A PARAMETER HAS BEEN ASSIGNED A VALUE, IT WILL RETAIN THAT VALUE UNTIL A SUBSEQUENT ASSIGNMENT REPLACES IT. ALL VALUES ARE INITIALIZED TO ZERO EXCEPT THE FOLLOWING:

SEQ IS INITIALIZED TO 50.0
FELAP, FIO, FPROC ARE INITIALIZED TO 1.0.

THE NAMES AND USE OF INPUT PARAMETERS ARE:

COMPAR (A REAL POSITIVE VALUE WHERE 1.0 EQUALS ONE MICROSECOND) THIS IS THE AVERAGE AMOUNT OF PROCESSOR TIME (IN MICROSECONDS) REQUIRED FOR ONE INVOCATION OF THE COMPARE PROCEDURE. THIS MUST ALSO INCLUDE PROCEDURE ENTRY AND EXIT TIME.

RSIZE (AN INTEGER VALUE WHERE ONE EQUALS ONE WORD AT LEAST ONE AND AT MOST 65535) THIS IS THE SIZE (IN WORDS) OF THE SORT RECORD. IN COBOL THIS IS THE SIZE OF THE SD WHILE IN ALGOL IT IS THE SAME AS THE RECORD LENGTH. NOTE THAT RECORD SIZE MUST BE IN WORDS AND NOT IN CHARACTERS.

RECS (AN INTEGER VALUE WHERE ONE EQUALS ONE RECORD AND WITH A VALUE OF AT LEAST ONE) THE EXACT NUMBER OF RECORDS TO BE SORTED.

DSIZE (A NON-NEGATIVE INTEGER VALUE WHERE ONE EQUALS ONE WORD) THIS IS THE AMOUNT OF DISK (IN WORDS) THAT WILL BE

SPECIFIED FOR THE SORT.

TAPES (A NON-NEGATIVE INTEGER VALUE WHERE ONE EQUALS ONE WORKTAPE) THIS IS THE NUMBER OF TAPES THAT WILL BE SPECIFIED FOR THE SORT.

CSIZE (A NON-NEGATIVE INTEGER VALUE WHERE ONE EQUALS ONE WORD) THIS IS THE AMOUNT OF MEMORY (IN WORDS) THAT WILL BE SPECIFIED FOR THE SORT.

RESTAR (A NON-NEGATIVE INTEGER VALUE LESS THAN 33) THE CORRECT VALUES THIS PARAMETER CAN TAKE ON ARE THE SAME AS THE RESTART PARAMETER USED FOR INVOCATION OF THE SORT. THIS PROGRAM, HOWEVER, DOES NOT ATTEMPT PREDICTIONS THAT INVOLVE RESTARTING PREVIOUSLY INCOMPLETE SORTS.

SEQ (A NON-NEGATIVE REAL VALUE BETWEEN ZERO AND 100) THIS PARAMETER IS USED TO SPECIFY THE RELATIVE SEQUENCE INHERENT IN THE INPUT DATA. THIS PROGRAM WILL ASSUME THAT THE SEQUENCE SPECIFIED WILL BE UNIFORM THROUGH ALL OF THE DATA. THE IMPORTANCE OF THIS PARAMETER SHOULD NOT BE MINIMIZED BECAUSE OF THE DIRECT RELATIONSHIP UPON THE RESULTS OBTAINED. USERS SHOULD BE AWARE THAT THE MARK 2.3 AND SUCCESSOR SORTS (INVOLVING THE USE OF SORT DISK) WILL ATTEMPT TO TAKE ADVANTAGE OF INHERENT FILE SEQUENCE TO THE EXTENT OF SWITCHING BETWEEN ASCENDING AND DESCENDING COMPARISONS. THIS GENERALLY RESULTS IN AN EFFECTIVE SEQUENCE OF AT LEAST 40% AND CAN EASILY IMPROVE THE SEQUENCE TO APPROACH 100%. THIS PROGRAM ASSUMES THE SORT IN USE WILL BE AT LEAST AT THE MARK 2.3 LEVEL.

ELAPIN (A NON-NEGATIVE REAL VALUE WHERE 1.0 EQUALS ONE MICROSECOND) THE AVERAGE AMOUNT OF ELAPSED TIME (IN MICROSECONDS) THAT OCCURS AS A RESULT OF AN ATTEMPT BY THE SORT TO "READ" ONE INPUT RECORD. THIS PARAMETER REPRESENTS TIME NOT INCURRED AS EITHER PROCESSOR OR I/O BUT AS TIME WAITING FOR AN EVENT TO OCCUR THAT IS EXTERNAL TO THE SORTING PROGRAM PROPER.

ELAPO (A NON-NEGATIVE REAL VALUE WHERE 1.0 EQUALS ONE MICROSECOND) THIS PARAMETER IS SIMILAR TO ELAPIN EXCEPT ITS USE IS FOR OUTPUT RECORDS (RETURNED TO USER BY SORT).

PROCIN (A NON-NEGATIVE REAL VALUE WHERE 1.0 EQUALS ONE MICROSECOND) THIS PARAMETER IS USED TO SPECIFY THE AVERAGE AMOUNT OF PROCESSOR TIME (IN MICROSECONDS) INCURRED TO READ ONE INPUT RECORD. IF AN INPUT PROCEDURE IS TO BE USED, IT IS THE AMOUNT OF PROCESSOR TIME REQUIRED BY THAT PROCEDURE. IF A FILE IS TO BE USED BY AN INPUT PROCEDURE OR PASSED TO THE SORT, THE PROCESSOR TIME USED BY THE SYSTEM TO READ THE RECORD MUST BE INCLUDED IN THE PARAMETER VALUE. IN ANY CASE ALL NECESSARY PROCEDURE ENTRY AND EXIT TIMES MUST BE INCLUDED.

PROCO (A NON-NEGATIVE REAL VALUE WHERE 1.0 EQUALS ONE MICROSECOND) THIS PARAMETER IS USED TO SPECIFY THE AVERAGE AMOUNT OF PROCESSOR TIME (IN MICROSECONDS) INCURRED TO WRITE ONE OUTPUT RECORD FROM THE SORT TO THE USERS OUTPUT FILE OR OUTPUT PROCEDURE. THE USE OF THIS VALUE IS SIMILAR TO PROCIN EXCEPT THAT IT IS APPLIED TO OUTPUT RATHER THAN INPUT.

UIOIN (A NON-NEGATIVE REAL VALUE WHERE 1.0 EQUALS ONE MICROSECOND) THIS PARAMETER IS USED TO SPECIFY THE AVERAGE AMOUNT OF I/O TIME TO READ ONE USER INPUT RECORD. THIS PARAMETER SHOULD BE THE SAME AS THE I/O TIME THAT WOULD BE LOGGED BY THE SYSTEM TO ACCOMPLISH THE SPECIFIC I/O.

UIOOUT (A NON-NEGATIVE REAL VALUE WHERE 1.0 EQUALS ONE MICROSECOND) THIS PARAMETER IS SIMILAR TO UIOIN EXCEPT THAT ITS USE IS APPLIED TO THE USER OUTPUT FILE RATHER THAN INPUT.

IA2 (NON-NEGATIVE REAL VALUES NO MORE THAN 100 WHERE IA2
IC3 THROUGH II86 REPRESENT A DISK TYPE --SEE B6700 HARDWARE
IC4 HANDBOOK, SECTION 7-- AND THE SUM OF ALL DISK TYPES IS NO
II82 MORE THAN 100.0) THIS PARAMETER IS USED TO SPECIFY THE

1184 PERCENT OF DSIZE TO BE USED BY THE PARTICULAR DISK TYPE.
1186 IN ACTUAL USE DSIZE IS FIRST CONVERTED TO THE NUMBER OF DISK ROWS AVAILABLE AND THE ROWS ARE THEN APPORTIONED ACCORDING TO THE PERCENTAGE VALUE, TO THE VARIOUS DISK TYPES. SELECTION BETWEEN DISK TYPES IS BASED UPON DISK SPEED PROCEEDING FROM HIGHEST SPEED (SMALLEST AVERAGE ACCESS) TO LOWEST SPEED.

TAPE0 (AN INTEGER VALUE AT LEAST ONE AND AT MOST 11, EXCEPT
TAPE1 THAT ZERO IS USED TO SPECIFY AN UNUSED TAPE) THIS
TAPE2 PARAMETER IS USED TO SPECIFY THE KIND OF TAPE UNIT THAT
TAPE3 WILL BE USED FOR EACH SORT TAPE. ONE UNIT MUST BE
TAPE4 ASSIGNED A VALUE FOR EACH TAPE SPECIFIED BY THE TAPES
TAPE5 PARAMETER. THE MINIMUM NUMBER OF TAPES THAT MAY BE
TAPE6 SPECIFIED IS THREE AND THE MAXIMUM IS EIGHT. THE PROGRAM
TAPE7 WILL EXPECT ASSIGNMENTS BEGINNING WITH TAPE0 AND CONTINUING SEQUENTIALLY THROUGH TAPE N WHERE N IS THE NUMBER OF TAPES SPECIFIED BY THE PARAMETER TAPES(MINUS ONE). THE TAPE UNIT TYPES REPRESENTED BY THE INTEGER ONE THROUGH 11 ARE:

1	7 TRACK	200 BPI	18KC	90IPS
2	7 TRACK	556 BPI	50KC	90IPS
3	7 TRACK	800 BPI	72KC	90IPS
4	7 TRACK	200 BPI	24KC	120IPS
5	7 TRACK	556 BPI	66KC	120IPS
6	7 TRACK	800 BPI	96KC	120IPS
7	9 TRACK	800 BPI	72KB	90IPS
8	9 TRACK	800 BPI	96KB	120IPS
9	PE	1600 BPI	144KB	90IPS
10	PE	1600 BPI	192KB	120IPS
11	PE	1600 BPI	240KB	150IPS

FOOLAP (A NON-NEGATIVE REAL VALUE AT MOST 1.0) THIS PARAMETER IS USED TO EXPRESS THE PERCENTAGE OF DISK I/O THAT WILL BE IN PROCESS SIMULTANEOUSLY. IN ACTUAL PRACTICE THE ELAPSED TIME OF AN I/O BOUND SORT MAY BE LESS THAN THE TOTAL I/O TIME BECAUSE OF THE ABILITY TO ACCOMPLISH

SIMULTANEOUS I/O REQUESTS. SINCE THIS PROGRAM DOES NOT CONSIDER I/O SIMULTANFITY, THIS PARAMETER WAS PROVIDED AS A MEANS OF EXPRESSING THAT CONDITION. PLEASE NOTE THAT THIS PARAMETER APPLIES ONLY TO DISK I/O. THE CONTENT OF THIS PARAMETER WILL BE MULTIPLIED BY THE DISK I/O TIME AND THE RESULT WILL BE USED TO REDUCE THE ELAPSED TIME. A VALUE OF 1.0 REPRESENTS 100% AND A VALUE OF 0.5 REPRESENTS 50%.

FTOLAP (A NON-NEGATIVE REAL VALUE AT MOST 1.0) THIS PARAMETER IS USED TO EXPRESS THE PERCENTAGE OF TAPE I/O THAT WILL BE IN PROCESS SIMULTANEOUSLY. USE OF THIS PARAMETER IS SIMILAR TO FDOLAP EXCEPT THAT IT APPLIES TO TAPE I/O RATHER THAN DISK.

FELAP (A NON-NEGATIVE REAL VALUE) THIS IS A GENUINE FUDGE FACTOR THAT IS INITIALLY SET TO 1.0. AFTER ELAPSED TIMES HAVE BEEN COMPUTED THEY WILL BE MULTIPLIED BY THIS PARAMETER. A VALUE LARGER THAN 1.0 WILL INCREASE THE ELAPSED TIMES AND A VALUE LESS THAN 1.0 WILL DECREASE ELAPSED TIMES. IT IS PROVIDED AS A OPTIONAL MEANS OF REFINING THE OUTPUT OF THIS PROGRAM.

FIO (A NON-NEGATIVE REAL VALUE) THE USE OF THIS PARAMETER IS SIMILAR TO FELAP EXCEPT THAT IT WILL BE USED TO MULTIPLY I/O TIME. IT IS INITIALLY SET TO 1.0.

FPROC (A NON-NEGATIVE REAL VALUE) THE USE OF THIS PARAMETER IS SIMILAR TO FELAP EXCEPT THAT IT WILL BE USED TO MULTIPLY PROCESSOR TIME. IT IS INITIALLY SET TO 1.0.

DBSIZE (AN INTEGER VALUE AT LEAST ZERO AND AT MOST 65535) THIS PARAMETER IS USED TO OVERRIDE THE COMPUTED BUFFER SIZE (DISK BUFFERS ONLY) IN A FASHION SIMILAR TO LABEL EQUATION PERMITTED BY THE SORT. IF THIS PARAMETER IS NON-ZERO, IT WILL BE USED FOR COMPUTING ESTIMATES BY THIS PROGRAM.

DEBUG (A NON-NEGATIVE REAL VALUE) THIS PARAMETER IS A DEBUGGING

TOOL FOR THIS PROGRAM. A VALUE BETWEEN 0.0 AND 1.0 WILL CAUSE CERTAIN DEBUGGING INFORMATION TO BE PRINTED AND A VALUE GREATER THAN 1.0 WILL CAUSE PRINTING OF ADDITIONAL DEBUGGING INFORMATION.

MODIFICATION OF PARAMETERS AND OUTPUT

PARAMETER VALUES MAY BE MODIFIED BY THE PROGRAM IN SOME CASES, HOWEVER, WHEN THIS OCCURS NOTIFICATION OF THE CHANGE WILL BE PRINTED. FATAL ERRORS WILL ALSO BE PRINTED AND THE ATTEMPT WILL BE MADE TO FIND ALL SAID ERRORS IN ANY SINGLE SCAN OF AN INPUT SET. THE OUTPUT FORMAT IS SIMILAR TO THE FORMAT USED BY THE PROGRAM "SYSTEM/SORTSTAT". A NEW AREA OF INFORMATION HAS BEEN ADDED THAT CONTAINS INFORMATION ABOUT THE AMOUNT OF WORKFILE STORAGE SPACE REQUIRED FOR THE SORT. THE ESTIMATE PRODUCED FOR DISK WORK SPACE WILL MUST LIKELY BE SUFFICIENT, HOWEVER THERE IS A POSSIBILITY THAT THE ESTIMATE REPRESENTS AN OPTIMAL CONDITION THAT WOULD NOT OCCUR IN ACTUAL PRACTICE WITH DATA THAT IS NOT UNIFORM IN SEQUENCE OR IN FACT IN A DIFFERENT SEQUENCE. THE AMOUNT OF TAPE SPACE REQUIRED ASSUMES NO LOSS OF TAPE DUE TO IMPERFECT RECORDING SURFACES AND ESTIMATES ONLY THE TOTAL AMOUNT OF SPACE AND NOT THE NUMBER OF REEL SWITCHES THAT MIGHT OCCUR. IT IS LIKELY THAT THIS GENERAL AREA WILL BE EXPANDED IN THE FUTURE.

SAMPLE DATA

EXAMPLES OF CARD DECKS TO COMPILE OR RUN THIS PROGRAM ARE AS FOLLOWS (WHERE <B> REPRESENTS A BLANK COLUMN):

A. TO COMPILE

```
<I>COMPILE SYSTEM/PREDICTSORT WITH FORTRAN TO LIBRARY
<I>FORTRAN FILE TAPE (TITLE = SYMBOL/PREDICTSORT)
<I>DATA
<END>
```

B. TO RUN

```
<I>RUN SYSTEM/PREDICTSORT
```

<I>DATA FILES

<B>&INFO<B>

<ONE OR MORE PARAMETER CARDS>

<B>&END

<I>END

PARAMETER VALUES MAY ALSO BE PLACED ON THE &INFO CARD OR THE &END CARD.

RJE

D0159 RJE - RUN JOBS WITH PARAMETERS - 12-04-72

THIS PATCH ALLOWS A "RUN" OR "EXECUTE" CARD RUN THROUGH RJE TO HAVE A QUOTED STRING AS A SINGLE PARAMETER. EXAMPLE:

<I> RUN SYSTEM/DCSTATUS ("ALL")$ END.

D0173 RJE - DP MESSAGE TO RJE - 12-04-72

THIS PATCH ALLOWS THE DP MESSAGE AT EITHER THE REMOTE CONSOLE OR THE CENTRAL SITE (VIA THE SM CONSTRUCT) TO BE OPTIONALLY FOLLOWED BY DUMP OPTIONS DESIRED. THE OPTIONS ALLOWED ARE "ARRAYS", "BASE", "CODE", "FILES", OR <INTEGER>. COMMAS MAY BE USED TO SEPARATE TO OPTIONS IF DESIRED.

D0187 RJE - PRIORITY OF RJE BACKUPS - 11-20-72

THIS PATCH CAUSES THE AUTOPRINT ROUTINE OF RJE TO RUN AT A PRIORITY FIVE HIGHER THAN RJE ITSELF. IN NO CASE WILL THE PRIORITY OF BACKUPS BE HIGHER THAN 99.

D0188 RJE - FORMMESSAGE THROUGH RJE - 11-20-72

THIS PATCH IMPLEMENTS THE FORMMESSAGE FUNCTION FOR PRINTFILES PRINTED AT AN RJE TERMINAL. THE NOTIFICATION OF FORMS REQUIRED WILL BE <MIX NUM> <FORMMESSAGE STRING>. THE PRINTING JOB WILL WAIT UNTIL THE REMOTE OPERATOR RESPONDS <MIX NUM> FM <OPTIONAL STRING>. ONLY ONE FILE WILL BE PRINTED AND A "PB" MESSAGE WILL CAUSE PRINTING TO CONTINUE. THIS ALLOWS REMOVAL OF ANY SPECIAL FORMS

D0188 RJE - FORMMESSAGE THROUGH RJE - 11-20-72

WHICH HAVE BEEN USED.

D0189 RJE - VALUE AND BDNAME - 11-20-72

THIS PATCH IMPLEMENTS "VALUE" AND "BDNAME" CONTROL CARDS IN RJE. A RUN TIME "VALUE" OR "BDNAME" CANNOT BE SET AT COMPILE TIME. RJE ALWAYS SETS BOBASE. SO IF USED, BDNAME IS EFFECTIVE.

D0190 RJE - SS BETWEEN RJE STATIONS - 11-20-72

THIS PATCH ALLOWS RJE USERS TO SEND MESSAGES TO OTHER RJE STATIONS REMOTE SUPERVISOR CONSOLES. THE SYNTAX IS <STA NUMBER> SS <TEXT>. THE TEXT WILL APPEAR ON THE STATION GIVEN BY <STA NUMBER> AS, "#SS FROM <ORIGINATING STA NUMBER>: <TEXT>".

D0201 RJE - SM MESSAGE IMPLEMENTATION - 11-20-72

RJE WILL NOW ACCEPT SM MESSAGES FROM THE CENTRAL SITE CONSOLE. THE BASIC FORM OF AN SM MESSAGE IS <MCS MIX NUMBER>SM:<TEXT>.

RJE REQUIRES THAT THE <TEXT> BE IN ONE OF TWO BASIC FORMS, <KEY WORD><OPTIONAL TEXT> OR <LSN><KEY WORD><OPTIONAL TEXT>. <KEY WORD>S ARE TWO LETTER MNEMONICS. THE FOLLOWING DISCUSSION REFERS ONLY TO SM MESSAGES WHICH MUST BE ENTERED AT THE CENTRAL SITE CONSOLE.

<KEY WORD>S REQUIRING LSN.

PH IF AN AUTOMATIC CALLBACK WAS ATTEMPTED BUT UNSUCESSFUL, THE PH MESSAGE WILL SIGNAL RJE TO TRY DIALING AGAIN. NOTE THAT AN ATTEMPT TO DIAL MUST HAVE ALREADY OCCURRED. THIS RESTRICTION AVOIDS ERRONEOUSLY OR CAPRICIOUSLY CONNECTING SWITCHED LINES.

SU <OPTION> - SU SETS THE <OPTION> SPECIFIED. <OPTION> CAN BE EITHER LOGON OR USER. IF USER IS SET, ALL DECKS ORIGINATING

FROM THAT STATION MUST BE PRECEDED BY A USER CARD. IF LOGON IS SET, A LOGON SEQUENCE AS DESCRIBED IN THE RJE MANUAL IS REQUIRED. IF LOGON IS SET AND USER IS RESET ALL DECKS ORIGINATING FROM THAT STATION WHICH DO NOT HAVE A USER CARD WILL BE RUN UNDER THE USERCODE AS GIVEN BY THE LOGON SEQUENCE. IF LOGON IS RESET, NO LOGON SEQUENCE IS REQUIRED, AND NO DEFAULT USERCODE WILL BE USED. IF A STATION IS ACTIVE AND NOT LOGON WHEN THE CENTRAL SITE OPERATOR SETS LOGON, THE STATION WILL BE LOGGED OFF AND REQUIRED TO LOGON.

THE SETTINGS OF THESE TWO OPTIONS WILL BE REMEMBERED ACROSS DIFFERENT RUNS OF RJE. HOWEVER, IF RJELINKED FILE IS REMOVED, SETTINGS WILL RETURN TO THE DEFAULT OF LOGON = SET AND USER = RESET.

NOTE THAT WHEN RUNNING UNDER A USERCODE, LIBRARY MAINTENANCE CANNOT BE PERFORMED ON FILES NOT UNDER THAT USERCODE DIRECTORY UNLESS IT IS A PRIVILEDGED USER. TO FIND FILES UNDER A USERCODE THROUGH RJE, USE PD USERCODE/<USER>/<FILE NAME>, WHERE <FILE NAME> CAN BE AN EQUALS SIGN. PD <FILE NAME> WILL SEARCH THE SYSTEM DIRECTORY.

RO <OPTION> - THE RO MESSAGE RESETS THE ABOVE OPTIONS.

SV THE SV MESSAGE SAVES THE SPECIFIED STATION. THE STATION IS MADE NOT READY, AND ERROR RECOVERY IS DISCONTINUED. SEE SYSTEM NOTE P0840.

RY THE RY MESSAGE MAKES THE STATION, AND LINE ASSOCIATED WITH THAT STATION, READY. THIS WOULD APPROPRIATELY BE PERFORMED ON A PREVIOUSLY SAVED STATION. SEE PRECEDING SV MESSAGE DISCUSSION.

RE <OPTIONAL TO><MCS NAME> - THE RE MESSAGE ALLOWS THE CENTRAL SITE OPERATOR TO RELEASE AN RJE STATION AS GIVEN BY THE <LSN> TO ANOTHER MCS. AN EXAMPLE WOULD BE:

31 RE TO SYSTEM/CANDE

NOTE THAT ONLY THE STATION SPECIFIED IS RELEASED. IF, FOR

EXAMPLE, LSN 31 WERE A REMOTE SUPERVISORY CONSOLE THE PREVIOUS STATEMENT WOULD RELEASE ONLY THE REMOTE SUPERVISORY CONSOLE; THE PRINTER, CARD READER AND DC1000 WOULD FUNCTION NORMALLY. IF THE LSN RELEASED IS A PRINTER OR REMOTE SUPERVISORY CONSOLE, NO FURTHER OUTPUT TO THAT LSN WILL BE ATTEMPTED.

<KEY WORD> ALLOWING OPTIONAL LSN.

WH THE WH MESSAGE WITHOUT LSN PROVIDES THE LSN, STATE (I.E. ACTIVE, INACTIVE, SAVED) AND STATION NAME OF EACH STATION KNOWN TO RJE. IF AN LSN IS PROVIDED, THE STATION NAME, USERCODE (IF APPLICABLE), LOGON REQUIREMENTS, USER CARD REQUIREMENTS OR DEFAULT USERCODE USE, SWITCHED STATUS, STATE AND PHONE NUMBER (IF APPLICABLE) ARE GIVEN FOR THE SPECIFIED STATION. THE INFORMATION IS PROVIDED VIA SEQUENTIAL DISPLAYS ON AN ASYNCRONUOUSLY RUNNING TASK.

<KEY WORD>S NOT ALLOWING LSN.

DP <DUMP OPTIONS> - PROVIDES A RJE LINKED FILE PRINTOUT AND A PROGRAM DUMP. IT IS ONLY APPLICABLE WHEN RJE IS COMPILED WITH DEBUG SET. <DUMP OPTIONS> ARE BASE, ARRAYS, FILES, CODE OR AN <INTEGER>. OPTIONS MAY BE SEPARATED BY COMMAS OR BLANKS. IF NO OPTIONS ARE SPECIFIED THE OPTIONS USED WILL BE AS APPEARED ON THE OPTION CARD WHEN RJE WAS COMPILED. FOR A DISCUSSION OF THE FUNCTION OF THE <DUMP OPTIONS> SEE DOCUMENTATION ON PROGRAMDUMP INTRINSIC.

HI THIS MESSAGE CAUSES THE CURRENT DEBUG LISTING TO BE PRINTED. CONSECUTIVE HI MESSAGES WILL PRINT ONLY THE DEBUG LISTING NOT PREVIOUSLY PRINTED.

QT THIS MESSAGE WILL CAUSE RJE TO GO TO EOJ IN AN ORDERLY FASHION. IN PARTICULAR, STATIONS WILL BE LOGGED OFF SO THAT CORRECT ACCOUNTING CAN BE PROVIDED. THIS TECHNIQUE SHOULD BE USED IN PREFERENCE TO DS-ING RJE.

SS <STRING> - THIS IS A BROADCAST OF THE <STRING> TO ALL ACTIVE RJE STATIONS. THE MESSAGE WILL BE PRINTED ON THE RSC AS, #SS

ALL:<STRING>.

D0202 RJE - DP OPTIONS - 11-20-72

THIS PATCH IMPLEMENTS THE OPTIONS BASE, ARRAYS, FILES, CODE AND <INTEGER> TO THE DP REMOTE SUPERVISORY CONSOLE (RSC) INPUT MESSAGE. THE OPTIONS FOLLOW THE MNEMONIC DP AND MAY BE SEPARATED BY EITHER BLANKS OR COMMAS. THE DP MESSAGE IS ONLY ALLOWED WHEN RJE IS COMPILED WITH DEBUG SET. IF NO OPTIONS ARE SPECIFIED THE OPTIONS AS APPEARED ON THE OPTION CARD WHEN RJE WAS COMPILED WILL BE USED. NOTE THAT THIS FUNCTION PARALLELS THE SM "DP" MESSAGE DESCRIBED IN D0201.

FOR A DISCUSSION OF THE FUNCTIONS OF THE VARIOUS OPTIONS SEE DOCUMENTATION ON PROGRAMDUMP INTRINSIC.

D0212 RJE - MULTISTATION RJE TERMINALS - 01-15-73

THIS PATCH CHANGES RJE TO TREAT EACH PERIPHERAL AND ITS ASSOCIATED DC1000 AS DISTINCT STATIONS GROUPED BY LINE. THE US REMOTE SUPERVISORY CONSOLE (RSC) MESSAGE WILL NOW SHOW THE CORRECT LSN FOR THE PERIPHERALS FOR THE ORIGINATING DC1000. THE <RJE MIX#> SM:<LSN>WH WILL ALSO SHOW THESE LSNS. ALL PRINTER BACKUPS WILL BE GROUPED BY THE DC1000 LSN.

IN GENERAL, WHEN THE REMOTE TERMINAL IS BEING THOUGHT OF AS ONE UNIT THE LSN OF THE DC1000 IS THE PROPER LSN TO USE. (E.G., WHEN DIRECTING BACKUP FILES TO RJE TERMINALS.)

IN MOST CASES THE USER NEED NOT BE CONCERNED THAT EACH PERIPHERAL HAS A SEPARATE LSN.

D0213 RJE - STOP BY RSVP AND ACCEPT - 01-15-73

THIS PATCH CAUSES RJE TO PRINT ON THE REMOTE SUPERVISORY CONSOLE

D0213 RJE - STOP BY RSVP AND ACCEPT - 01-15-73

(RSC) ANY ACTIVE RSVP MESSAGE. THIS INCLUDES NO FILE, DUP FILE, ACCEPT, ETC.

D0214 RJE - AX RSC INPUT MESSAGE - 01-22-73

THIS PATCH ALLOWS THE RJE USER TO RESPOND TO JOBS STOPPED BECAUSE OF AN ACCEPT STATEMENT. THE FORMAT OF THE REPLY IS: <MIX#>AX<TEXT>. THE <TEXT> WILL BE PASSED TO THE JOB DESCRIBED BY <MIX#>.

D0215 RJE - PRINT FILE SEARCHING - 01-29-73

THIS PATCH CHANGES RJE TO UTILIZE THE GINSERTEVENT SO THAT RJE IS MORE EVENT DRIVEN. IN PARTICULAR, THE RJE ALGORITHM FOR PRINT FILES IS DIFFERENT. RJE WILL LOOK FOR PRINT FILES FOR A STATION WHEN A JOB COMPLETES OR A PB REMOTE SUPERVISORY CONSOLE (RCS) INPUT MESSAGE FROM THAT STATION IS RECOGNIZED. WHEN APPROXMATELY 30 SECONDS HAVE ELAPSED RJE SEARCHES FOR PRINT FILES FOR ALL STATIONS. THIS ELAPSED TIME IS INCREASED AS RJE BECOMES BUSIER.

D0216 RJE - RSC OUTPUT TO INACTIVE STATION - 01-29-73

THIS PATCH CAUSES MESSAGES SENT TO THE REMOTE SUPERVISORY CONSOLE (RSC) TO BE THROWN AWAY WHEN A SWITCHED STATION IS INACTIVE.

D0222 RJE - RE RSC MESSAGE - 01-15-73

THIS PATCH IMPLEMENTS THE RE REMOTE SUPERVISORY CONSOLE INPUT MESSAGE. THE FORMAT IS:

<LSN>RE <OPTIONAL TO><MCS NAME>

THE <LSN> MUST BE AN LSN ASSIGNED TO THE ORIGINATING TERMINAL.

EXAMPLE:

D0222 RJE - RE RSC MESSAGE - 01-15-73

31 RE TO SYSTEM/CANDE.

RJE WILL ALSO NOW RECOGNIZE A STATION RELEASED TO IT BY ANOTHER MCS.

D0223 RJE - LOGANALYZER THROUGH RJE - 02-05-73

THIS PATCH CHANGES RJE TO INTERFACE WITH SYSTEM/LOGANALYZER. ALTHOUGH REMOTE SUPERVISORY CONSOLE INPUT IS ALLOWED, A DECK FORMAT IS REQUIRED. E.G. ?LOG MIX 1721;END. FOR THE VARIOUS MESSAGES ALLOWED SEE THE DOCUMENTATION ON SYSTEM/LOGANALYZER.

SCR

D0281 SCR - EU MAINTENANCE - 03-23-73

THE REQUIREMENT FOR THE RY EU FACILITY HAS BEEN OBVIATED BY THE DISK RESERVE AND RETURN FACILITIES IN CONJUNCTION WITH IMPROVEMENTS IN SCR.

THE PURPOSE OF RY EU WAS TO PERMIT ROTATING OUT OF SYSTEM A DISK EU AND ITS STORAGE MODULES TO PERFORM MAINTENANCE. TO ACCOMPLISH THIS USING THE RY EU FACILITY REQUIRED THE FOLLOWING STEPS:

1. HALT SYSTEM.
2. CABLE IN A SUBSTITUTE EU PLUS STORAGE MODS AS MIRROR IMAGE OF SUBSYSTEM BEING ROTATED OUT.
3. WHILE THE SYSTEM IS DOWN, RUN SOME OFF-LINE ROUTINE WHICH WOULD BLINDLY COPY THE SUBSYSTEM BEING ROTATED OUT TO THE SUBSTITUTE SUBSYSTEM.
4. SWAP UNIT NUMBERS OF THE SUBSTITUTE AND ORIGINAL EUS.
5. TAKE ORIGINAL SUBSYSTEM OFF-LINE, PUT SUBSTITUTE SUBSYSTEM ON-LINE, AND HALT/LOAD THE SYSTEM.
6. AT THIS POINT THE SYSTEM IS UP AGAIN AND THE ORIGINAL SUBSYSTEM MAY BE TESTED AND REPAIRED OFF-LINE.
7. WHEN THE OFF-LINE SUBSYSTEM IS REPAIRED, IT MAY BE TESTED BY TURNING THE SUBSYSTEM ON-LINE AND KEYING IN RY EU ON THAT UNIT NUMBER. THE RY WOULD ALLOCATE A SERIES OF SPECIAL BADDISK FILES WHICH MAPPED ALL OF THE DISK ON THAT EU.
8. ON-LINE MAINTENANCE COULD THEN BE RUN ON THESE BADDISK FILES TO TEST AND VERIFY THE REPAIRED SUBSYSTEM.

9. TO RETURN THIS DISK TO THE SYSTEM REQUIRED THAT ALL OF THE BADDISK FILES BE MOVED AND THE SYSTEM BE HALT/LOADED.

TO ACCOMPLISH THIS USING CURRENT SOFTWARE:

1. RESERVE EU TO BE PM-ED AS BADDISK.

2. PERFORM MAINTENANCE, TESTING, AND REPAIRS IN ON-LINE OR OFF-LINE MODE AS REQUIRED.

3. WHEN THE DISK SUBSYSTEM HAS BEEN SATISFACTORILY PM-ED AND VERIFIED, THE SUBSYSTEM CAN BE RETURNED TO THE SYSTEM BY RESERVE <OK>, COPY ERRORS AND THEN PERFORMING A RETURN ON THAT EU.

IF IT IS NECESSARY TO ADD A DISK SUBSYSTEM TO THE SYSTEM, THIS CAN BE ACCOMPLISHED BY MAKING THE EU READY (AFTER A H/L), RESERVEING IT AS IAD AND THEN RETURNING IT.

NOTE: THE USE OF RES AND RET ELIMINATES THE NEED FOR TAKING THE SYSTEM DOWN OR DOING ANY HALT/LOADS.

MISCELLANEOUS

D0137 ARITHMETIC TEST AE4 - 09-22-72

INTRODUCTION:

THE AMOUNT OF CODE EXECUTED BY THE AE3 PROGRAM IN DETERMINING THE CORRECT ANSWER AND CHECKING THE MACHINE RESULT WITH THE ARITHMETIC INTERPRETER RESULT CAUSES THE EXECUTION FREQUENCY OF THE MACHINE DOUBLE-PRECISION OPERATOR TO BE TOO LOW FOR AE3 TO BE USEFUL AS A DIAGNOSTIC AID IN TROUBLE SHOOTING INTERMITTENT HARDWARE PROBLEMS.

THE AE4 PROGRAM IS INTENDED TO BE THE DIAGNOSTIC HALF OF A DETECT AND DIAGNOSE PAIR OF PROGRAMS IN WHICH AE3 MAY BE USED AS THE DETECT HALF. THIS PROGRAM ACCEPTS INPUT FROM THE CONSOLE SPECIFYING THE A, X, B, Y REGISTER CONTENTS, THE OPERATOR TO BE APPLIED TO THESE OPERANDS, AND THE CORRECT RESULT. IT THEN APPLIES THE SPECIFIED OPERATOR TO THESE OPERANDS AND PERFORMS A BIT COMPARISON OF THE RESULT WITH THE (PRESUMED) CORRECT ANSWER. A RUNNING COUNT OF THE NUMBER OF EXECUTIONS OF THE OPERATOR AND NUMBER OF MACHINE ANSWERS NOT EQUAL TO THE SPECIFIED CORRECT ANSWER MAY BE ACCUMULATED IN ORDER TO DETERMINE THE FREQUENCY OF FAILURE.

INPUT PARAMETERS:

THE REGISTER CONTENTS AND OPERATOR MNEMONIC ARE SPECIFIED BY MEANS OF "AX" REPLIES TO CONSOLE QUERIES. THE PROGRAM WILL DISPLAY A QUESTION OF THE FORM "WHAT-S A = ?" AND WAIT FOR THE RESPONSE. THIS RESPONSE IS EXPECTED TO BE OF THE FORM <MIX NUMBER> AX <IDENTIFIER> = <VALUE><ETX>. THE <IDENTIFIER> USED IN SUCCESSIVE QUERIES IS: A, X, B, Y, OPERATOR MNEMONIC, FIRSTWORD OF ANS, AND SECONDWORD OF ANS; THE <VALUE> FOR OPERATOR MNEMONIC MAY BE ADD, SUBT, MULT, DIVD, IDIV OR RDIV, AND THE <VALUE> FOR ALL OTHER QUERIES IS EXPECTED TO BE EITHER:

D0137 ARITHMETIC TEST AE4 - 09-22-72

1. A STRING OF ONE TO 16 ZEROES;
2. A STRING OF EXACTLY 12 HEX DIGITS;
3. A STRING OF EXACTLY 16 OCTAL DIGITS, OR
4. AN ASTERISK.

THE ASTERISK MEANS THAT THE VALUE OF THIS PARAMETER IS NOT TO BE CHANGED AND IS USED WITH THE HI INTERRUPT DISCUSSED BELOW.

A COMPLETE DIALOG SPECIFYING THAT 1.0 X 1.0 = 1.0 IS TO BE TESTED WOULD LOOK LIKE THIS:

```
QUERY:      <MIX-#> ACCEPT: WHAT-S A = ?
RESPONSE:      <MIX-#> AX A = 1141000000000000 <ETX>
QUERY:      <MIX-#> ACCEPT: WHAT-S X = ?
RESPONSE:      <MIX-#> AX X = 0 <ETX>
QUERY:      <MIX-#> ACCEPT: WHAT-S B = ?
RESPONSE:      <MIX-#> AX B = 1141000000000000 <ETX>
QUERY:      <MIX-#> ACCEPT: WHAT-S OPERATOR MNEMONIC?
RESPONSE:      <MIX-#> AX OP = MULT <ETX>
QUERY:      <MIX-#> ACCEPT: WHAT-S FIRSTWORD OF ANS?
RESPONSE:      <MIX-#> AX F = 1141000000000000 <ETX>
QUERY:      <MIX-#> ACCEPT: WHAT-S SECONDWORD OF ANS?
RESPONSE:      <MIX-#> AX S = 0 <ETX>
```

ALL OF THE NON-ZERO NUMBERS IN THE EXAMPLE ARE GIVEN IN OCTAL BUT COULD ALSO HAVE BEEN ENTERED IN HEX AS 261000000000.

PROGRAM OPERATION:

AFTER THE INPUT DIALOG HAS FINISHED, THE PROGRAM REPEATEDLY APPLIES THE DESIGNATED OPERATOR TO THE A/X-B/Y REGISTER CONTENTS AND PERFORMS A BIT COMPARISON OF THE MACHINE ANSWER WITH THE FIRSTWORD AND SECONDWORD CONSOLE INPUTS, WHICH ARE SUPPOSED TO BE THE CORRECT ANSWER. IF THESE TWO WORDS ARE NOT EXACTLY EQUAL, AN ERROR COUNTER IS INCREMENTED. THIS ERROR COUNTER CAN BE INTERROGATED BY MEANS OF <MIX-#>DT3<ETX>; THE FREQUENCY OF FAILURE MAY BE DETERMINED BY SAMPLING THE COUNT OF THE TOTAL NUMBER OF EXECUTIONS OF THE DOUBLE-PRECISION OPERATOR BY MEANS OF <MIX-#>DT4<ETX>.

D0137 ARITHMETIC TEST AE4 = 09-22-72

IF THERE IS ANY UNCERTAINTY CONCERNING WHETHER THE PROGRAM QUERIES WERE ANSWERED CORRECTLY, A LISTING FOR THE CURRENT SET OF PARAMETERS CAN BE OBTAINED BY PERFORMING:

1. A PD BD=<ETX>, AND
2. A ?PB D <NUMBER>; END <ETX>

WHERE THE <NUMBER> IS THE MIX NUMBER OF THE PROGRAM (INCLUDING LEADING ZEROES), AS INDICATED BY THE PD REQUEST.

THE CURRENT TEST MAY BE ABORTED AND A NEW SET OF PARAMETERS SPECIFIED BY ENTERING <MIX=#>HI<ETX>. THE PROGRAM WILL ENTER THE CONSOLE QUERY MODE, ASKING "WHAT=S = ?" AS IF IT HAS JUST BEEN INITIATED. THE DIFFERENCE IS THAT THE REGISTER CONTENTS WILL NOT BE CHANGED FROM THOSE THAT WERE BEING USED IN THE INTERRUPTED TEST, SO CHANGES AND CORRECTIONS MAY BE ENTERED. WHEN ALL SUCH DESIRED CHANGES HAVE BEEN MADE, A REPLY OF <MIX=#>AX END<ETX> TO THE CURRENT ACCEPT INPUT REQUEST WILL TERMINATE CONSOLE REQUESTS AND BEGIN EXECUTION OF THE DOUBLE OPERATOR WITH THE NEW PARAMETERS.

PROGRAM OPTIONS:

THE EXECUTION FREQUENCY OF THE DOUBLE-PRECISION OPERATOR MAY BE INCREASED BY RECOMPILING THE SYMBOL/TEST/AE4 SYMBOLIC FILE USING THE FOLLOWING DECKS:

```
<I> COMPILE TEST/AE4 UCALGOL LIBRARY
<I> ALGOL FILE TAPE (TITLE = SYMBOL/TEST/AE4, KIND = DISK).
<I> DATA
$ MERGE CHECK IGNOREERRORS
$ SET LIST % THIS CARD IS OPTIONAL
<I> END.
```

THE EFFECT OF THE IGNOREERRORS DOLLAR OPTION IS TO ELIMINATE THE COUNTING OF ERRORS AND OPERATOR EXECUTIONS. WITH THIS DOLLAR OPTION SET, THE CONSOLE INPUT ROUTINES WILL ALSO NOT REQUEST INPUT OF THE FIRSTWORD AND SECONDWORD OF THE CORRECT ANSWER.

PROGRAM LIMITATIONS:

D0137 ARITHMETIC TEST AE4 = 09-22-72

THE MACHINE RESULT FOR THE OPERATION AND THE CONSOLE INPUT FOR THE PRESUMED CORRECT ANSWER ARE COMPARED BY MEANS OF THE SAME OPERATOR. THUS, VALUES WHICH WOULD BE EQUAL UNDER ARITHMETIC COMPARISON (I.E., OCTAL 1141000000000000 = OCT 1130100000000 = OCT 1120010000000000 = ... = 0000000000000001) ARE NOT EQUAL UNDER THE SAME OPERATOR: THE FIRSTWORD AND SECONDWORD OF THE ANSWER ENTERED IN THE PARAMETER SPECIFICATIONS MUST BE THE EXACT BIT PATTERNS THAT A CORRECTLY OPERATING ARITHMETIC UNIT WOULD GENERATE.

THE PROGRAM ASSUMES THAT THE "CORRECT ANSWER" ENTERED IN THE CONSOLE DIALOG IS INDEED CORRECT AND MAKES NO ATTEMPT TO VALIDATE THIS INPUT. IF THIS RESULT IS INCORRECT OR IS ENTERED INCORRECTLY, SPURIOUS ERROR INDICATIONS WILL RESULT.

THERE IS ALSO NO ATTEMPT MADE TO DETERMINE WHETHER THE GIVEN A/X=B/Y OPERATOR COMBINATION WILL PRODUCE AN EXPONENT UNDERFLOW, EXPONENT OVERFLOW, OR INTEGER OVERFLOW RESULT, AND THERE IS NO ATTEMPTED RECOVERY FOR THESE HARDWARE INTERRUPTS AND THE JOB WILL BE TERMINATED IF ANY OF THESE FAULTS OCCUR.

D0144 BINDING INTERMED LEVEL GLOBALS = 02-19-73

BINDING WITH INTERMEDIATE GLOBALS

THE FACILITY HAS BEEN ADDED TO FORTRAN, ALGOL, AND THE BINDER TO BIND AND REPLACE PROCEDURES OR SUBPROGRAMS COMPILED AT ANY DEGREE OF NESTING WITHIN AN ALGOL PROGRAM. THIS FACILITY CAN BE USED WITH THE REVISED DUMPINFO AND LOADINFO IN ALGOL (SEE D0211 FOR A DESCRIPTION). THE OBJECTIVE IS TO FACILITATE ALGOL-FORTRAN COMMUNICATION AT ALL NESTING DEPTHS AS WELL AS DEBUGGING ALGOL PROGRAMS.

THE FORMAT IN ALGOL OR FORTRAN IS TO COMPILE THE PROCEDURE OR SUBPROGRAM AT THE LEVEL AT WHICH IT IS INTENDED TO RUN. THIS IS DONE BY SETTING THE LEVEL N DOLLAR CARD OPTION AT COMPILE TIME. (THE LEVEL THAT AN ALGOL PROCEDURE IS INTENDED TO RUN AT MAY BE FOUND BY COUNTING THE NESTING DEPTH IN FROM THE OUTER BLOCK OF THE

PROGRAM AND ADDING TWO. THUS A GLOBAL PROCEDURE RUNS AT LEVEL 3, A PROCEDURE NESTED INSIDE OF A GLOBAL PROCEDURE RUNS AT LEVEL 4, ETC.). THIS PROCEDURE MAY THEN BE BOUND INTO A HOST WHERE IT HAS BEEN DECLARED EXTERNAL OR REPLACE A PROCEDURE ALREADY IN THE HOST. ALL VARIABLES THAT ARE GLOBAL TO THE SEPARATELY COMPILED PROCEDURE MUST BE DECLARED WITHIN BRACKETS PRIOR TO THE COMPILATION. PREVIOUSLY, ONLY TRUE GLOBAL VARIABLES APPEARED WITHIN THE BRACKETS; NOW ALL THOSE VARIABLES WHICH CONSTITUTE THE ENTIRE ENVIRONMENT OF THE SEPARATELY COMPILED PROCEDURE (INCLUDING THOSE AT INTERMEDIATE LEVELS) MUST APPEAR.

BINDER SYNTAX

BINDER SYNTAX FOR REPLACING OR BINDING A PROCEDURE DECLARED AT ANY LEVEL IS THE SAME AS FOR BINDING GLOBAL PROCEDURES; MERELY USE THE BIND INSTRUCTION OR THE USUAL DEFAULTS. IT IS NOW POSSIBLE HOWEVER TO HAVE SEVERAL REPLACEABLE OR EXTERNAL PROCEDURES WITH THE SAME NAME, SO THE BIND STATEMENT SYNTAX HAS BEEN EXTENDED TO INCLUDE PROCEDURE QUALIFICATION.

QUALIFIERS MAY BE USED IN EACH OF THE FOLLOWING THREE STATEMENTS:

```
BIND <PROCEDURE IDENTIFIER><QUALIFIERS><FILE PART>;
EXTERNAL <PROCEDURE IDENTIFIER><QUALIFIERS>;
USE <IDENTIFIER> FOR <IDENTIFIER><QUALIFIERS>;
<QUALIFIERS>::= OF <PROCEDURE IDENTIFIER><QUALIFIERS>/
                <EMPTY>
```

FOR EXAMPLE,

```
BIND P;
```

WILL ATTEMPT TO BIND EVERY PROCEDURE NAMED P WITHIN THE HOST; IF MORE THAN ONE IS DECLARED IN THE PROGRAM (WHETHER OR NOT THE PROCEDURE IS ACTUALLY THERE OR JUST DECLARED EXTERNAL), THE BINDER WILL TRY TO BIND ALL OF THEM. IF THERE IS A PROCEDURE P CONTAINED WITHIN Q AND A PROCEDURE P CONTAINED WITHIN R, THE STATEMENT

```
BIND P OF Q;
```

WILL ATTEMPT TO BIND ONLY THE PROCEDURE P LOCATED WITHIN Q.

ONLY THE QUALIFICATION NECESSARY TO UNIQUELY IDENTIFY, WITHOUT OMITTING ANY NESTING LEVEL, THE PROCEDURE TO BE BOUND IS REQUIRED; IN THE EVENT THERE IS MORE THAN ONE WAY TO SO UNIQUELY IDENTIFY THE PROCEDURE, ANY OF THEM WILL WORK.

IN THE EVENT THAT THE BINDER ATTEMPTS TO BIND A PROCEDURE AND DOES NOT FIND THE CODE FILE OF THE PROCEDURE (BASED ON STANDARD BINDER DEFAULTS FOR CODE FILE TITLES), THE PROCEDURE WILL NOT BE BOUND AND A WARNING WILL BE GIVEN.

IF THE SEPARATE PROCEDURE CONTAINS A PARAMETER OR TYPE MISMATCH WITH ITS DECLARATION IN THE HOST, OR THE PROCEDURE IS COMPILED AT AN INCOMPATIBLE LEVEL, THE BINDER WILL ISSUE A WARNING MESSAGE AND DISCONTINUE BINDING THE PROCEDURE. (PREVIOUSLY, THE BINDER WOULD ISSUE ERROR MESSAGES AND NOT LOCK THE CODE FILE IF THE ABOVE CONDITIONS AROSE. NOW, HOWEVER, IT SUCCESSFULLY "BACKS OUT" OF BINDING THE PROCEDURE SO THE CODE FILE CAN BE LOCKED.) TRUE ERRORS WILL STILL RESULT FROM MISMATCHES BETWEEN GLOBALS.

BINDING PROCEEDS FROM THE MOST GLOBAL LEVEL TO THE LEAST GLOBAL LEVEL. FOR EXAMPLE, IF P IS CONTAINED IN Q, THE BINDER INSTRUCTIONS

```
BIND P; BIND Q;
```

WILL RESULT FIRST IN Q BEING BOUND AND THEN P BEING BOUND INTO Q.

THE MEANING OF QUALIFIER IN USE STATEMENTS IS SHOWN BY THE FOLLOWING EXAMPLE:

TWO SEPARATELY COMPILED PROCEDURES P AND Q EACH REFERENCE A VARIABLE X WHICH IS GLOBAL TO THE PROCEDURES. "X OF P" SHOULD ACTUALLY BE REFERENCING A VARIABLE NAMED Y IN THE HOST WHILE "X OF Q" SHOULD BE MATCHED UP TO A VARIABLE NAMED "X" IN THE HOST. THE USE STATEMENT: "USE Y FOR X OF P;" WILL RESULT IN ONLY THE VARIABLE "X" REFERENCED BY P MATCHING UP TO A "Y". IF "Y" DID NOT EXIST IN THE HOST, THE "X OF P" WOULD BE ADDED AS A NEW GLOBAL WITH A NAME OF "Y".

ALGOL EXAMPLE

CONSIDER THE FOLLOWING ALGOL PROGRAM:

```
BEGIN
REAL X; FILE F;
PROCEDURE Q;
BEGIN
REAL Y;
PROCEDURE R; EXTERNAL;
R;
IF Y=3 THEN X:=X+1;
END Q;
Q;
WRITE(F,/,X);
END.
```

A SAMPLE PROCEDURE R TO BE BOUND INTO THIS PROGRAM COULD APPEAR AS FOLLOWS:

```
$ SET LEVEL 4
[REAL X,Y; FILE F;]
PROCEDURE R;
BEGIN
Y:=X+3;
WRITE (F,/,Y);
END.
```

THE BIND STATEMENT TO BIND INTO THE HOST COULD BE SIMPLY

```
BIND R; (OR BIND R FROM CODE FILE NAME).
```

THE RESULTING PROGRAM WOULD HAVE THE FOLLOWING OUTPUT:

```
3
1
```

FORTRAN-ALGOL EXAMPLE

COMMON BLOCKS MAY BE DECLARED AT INTERMEDIATE LEVELS IN ALGOL AND A

FORTRAN PROGRAM UNIT MAY THEN REFERENCE THEM. FOR EXAMPLE, CONSIDER THE FOLLOWING ALGOL PROGRAM:

```
BEGIN
FILE F;
PROCEDURE P;
BEGIN
REAL ARRAY A[1:50];
DOUBLE ARRAY B[1] = A;
PROCEDURE Q; EXTERNAL;
Q;
WRITE (F,/,A[1], B[10]);
END P;
P;
WRITE (F,"GOODBYE">);
END.
```

IT IS DESIRED TO BIND THE FOLLOWING FORTRAN SUBROUTINE INTO THIS PROGRAM:

```
$ SET LEVEL 4
SUBROUTINE Q
COMMON /BLK/AA(50)
DOUBLE PRECISION BB(25)
EQUIVALENCE (AA,BB)
AA(1)=44
BB(10)=3456
RETURN
END
```

THEN THE USUAL USE STATEMENT (SEE D0008) MAY BE USED TO BIND IN THE COMMON BLOCK. ONLY THE BLOCK WILL NOW BE DECLARED INSIDE OF P RATHER THAN GLOBALLY. THE BINDER INSTRUCTIONS COULD BE AS FOLLOWS:

```
BIND Q;
USE A FOR /BLK/;
```

THE RESULTING OUTPUT SHOULD BE 44, 356, AND GOODBYE, IN THAT ORDER.

DUPLICATE ID EXAMPLE

CONSIDER THE FOLLOWING ALGOL PROGRAM

```
BEGIN
PROCEDURE P; BEGIN END;
PROCEDURE Q;
BEGIN
PROCEDURE P; BEGIN END;
P;
END;
Q;
P;
END.
```

ASSUME THAT A FILE CALLED "SEP/P" WAS CREATED BY COMPILING A PROCEDURE P SEPARATELY AT LEVEL FOUR. (I.E., THE LEVEL IS COMPATIBLE WITH THE PROCEDURE P NESTED IN Q IN THE ABOVE EXAMPLE, BUT IS NOT COMPATIBLE WITH THE GLOBAL P.) THE BINDER INSTRUCTION

```
BIND P FROM SEP/P;
```

WILL, AS STATED BEFORE, CAUSE THE BINDER TO REPLACE ALL PROCEDURES NAMED P. HOWEVER, WHEN THE BINDER ATTEMPTS TO REPLACE THE GLOBAL P, AN "INCOMPATIBLE LEVEL ERROR" WILL RESULT AND THE BINDER WILL DISCONTINUE BINDING P AT THAT POINT. LATER IT WILL SUCCESSFULLY REPLACE THE P NESTED IN Q AND EVENTUALLY WILL LOCK THE CODE FILE. THE RESULTS WOULD HAVE BEEN THE SAME IF THE INSTRUCTIONS HAD BEEN

```
EXTERNAL P;
BIND P OF Q;
```

EXCEPT THAT NO ATTEMPT WOULD HAVE BEEN MADE TO BIND THE GLOBAL P.

PRECEDENCE OF INPUT STATEMENTS

IN CASES WHERE MORE THAN ONE STATEMENT MIGHT APPLY TO AN IDENTIFIER, THE FOLLOWING RULES APPLY:

1. THE MOST QUALIFIED IDENTIFIER WHICH FITS THE ENVIRONMENT OF THE GIVEN IDENTIFIER WILL APPLY.

2. IF TWO STATEMENTS ARE EQUALLY QUALIFIED THEN AN EXTERNAL STATEMENT WILL TAKE PRECEDENCE OVER A BIND STATEMENT WHICH, IN TURN, WILL TAKE PRECEDENCE OVER A CBIND STATEMENT USED IN AUTOBIND. (USE STATEMENTS DO NOT CONFLICT WITH BIND, EXTERNAL, OR CBIND STATEMENTS AND THEREFORE THIS SECOND PRECEDENCE RULE DOES NOT APPLY TO USE STATEMENTS.)

EXTERNAL STATEMENT

THE MEANING OF THE EXTERNAL STATEMENT HAS BEEN EXPANDED SOMEWHAT TO MEAN THAT THE PROCEDURE IS TO BE LEFT AS FOUND BY THE BINDER. (I.E., IF THE PROCEDURE IS EXTERNAL, LEAVE IT EXTERNAL; IF IT IS PRESENT, DO NOT TRY TO REPLACE IT.) THUS THE STATEMENTS:

```
BIND P;
EXTERNAL P OF Q;
```

WILL RESULT IN THE BINDER ATTEMPTING TO BIND OR REPLACE ALL PROCEDURES NAMED P EXCEPT THOSE NESTED IN PROCEDURES NAMED Q.

THE STATEMENTS

```
EXTERNAL P;
BIND P OF Q;
```

WILL RESULT IN THE BINDER ATTEMPTING TO BIND OR REPLACE ALL PROCEDURES NAMED P WHICH ARE NESTED IN PROCEDURE NAMED Q AND NO OTHER.

ADDING GLOBALS

IF A GLOBAL APPEARS IN A SEPARATELY COMPILED PROCEDURE THAT IS NOT DECLARED IN THE HOST, THE BINDER WILL, IF ABLE, ADD THE GLOBAL TO THE HOST. THE GLOBAL WILL BE ADDED IN AS A TRUE GLOBAL; I.E., AT THE OUTERMOST BLOCK OF THE HOST.

COMPILE TIME BINDING

THESE FEATURES FOR INTERMEDIATE LEVEL GLOBAL BINDING HAVE ALSO BEEN

D0144 BINDING INTERMED LEVEL GLOBALS - 02-19-73

ADDED TO AUTOBIND IN THE COMPILERS. THUS, A HIGHER LEVEL PROCEDURE MAY BE COMPILED SEPARATELY AND BOUND INTO A HOST WITH THE SAME AUTOMATIC FEATURES AS GLOBAL PROCEDURES. FOR MORE DETAILS ON AUTOBIND, SEE D0004.

D0217 COMPILATION OF COMPILERS - 01-29-73

THIS NOTE REPLACES D0013 IN THE SYSTEM MISCELLANEA.

TABLES FOR THE ALGOL, ESPOL, AND XALGOL COMPILERS ARE GENERATED BY A TABLE GENERATION PROGRAM WHICH CREATES A NEW SYMBOLIC FILE OF THE COMPILER. THEREFORE, AS IN THE PAST, THERE IS NO NEED TO USE THE "INCLUDE" DOLLAR OPTION WHEN COMPILING ANY OF THESE COMPILERS.

DOCUMENTS AFFECTED

DOCUMENT	SYSTEM NOTE	MARKETING NO.	MARKETING DATE
ALGOL COMPILER	D0158	5000136	06-72
ALGOL COMPILER	D0183	5000136	06-72
ALGOL LANGUAGE	D0144	5000128	06-72
ALGOL LANGUAGE	D0145	5000128	06-72
ALGOL LANGUAGE	D0147	5000128	06-72
ALGOL LANGUAGE	D0156	5000128	06-72
ALGOL LANGUAGE	D0191	5000128	06-72
ALGOL LANGUAGE	D0211	5000128	06-72
ALGOL LANGUAGE	D0218	5000128	06-72
ALGOL LANGUAGE	D0219	5000128	06-72
ALGOL LANGUAGE	D0220	5000128	06-72
ALGOL LANGUAGE	D0252	5000128	06-72
ALGOL LANGUAGE	D0253	5000128	06-72
ALGOL LANGUAGE	D0266	5000128	06-72
ALGOL LANGUAGE	D0267	5000128	06-72
BASIC LANGUAGE	D0152	5000383	07-71
BASIC LANGUAGE	D0154	5000383	07-71
BASIC LANGUAGE	D0162	5000383	07-71
BASIC LANGUAGE	D0163	5000383	07-71
BASIC LANGUAGE	D0164	5000383	07-71
BASIC LANGUAGE	D0177	5000383	07-71
BASIC LANGUAGE	D0178	5000383	07-71
BASIC LANGUAGE	D0182	5000383	07-71
BASIC LANGUAGE	D0193	5000383	07-71
BASIC LANGUAGE	D0194	5000383	07-71
BASIC LANGUAGE	D0195	5000383	07-71
BASIC LANGUAGE	D0197	5000383	07-71
BASIC LANGUAGE	D0203	5000383	07-71
BASIC LANGUAGE	D0204	5000383	07-71
BASIC LANGUAGE	D0205	5000383	07-71

DOCUMENTS AFFECTED

DOCUMENT	SYSTEM NOTE	MARKETING NO.	MARKETING DATE
BASIC LANGUAGE	D0206	5000383	07-71
BASIC LANGUAGE	D0207	5000383	07-71
BASIC LANGUAGE	D0208	5000383	07-71
BASIC LANGUAGE	D0209	5000383	07-71
BASIC LANGUAGE	D0221	5000383	07-71
BINDER	D0144	5000045	11-71
BINDER	D0156	5000045	11-71
CANDE LANGUAGE	D0186	5000318	10-72
CANDE LANGUAGE	D0237	5000318	10-72
CANDE LANGUAGE	D0238	5000318	10-72
CANDE LANGUAGE	D0239	5000318	10-72
CANDE LANGUAGE	D0275	5000318	10-72
CANDE OPERATION	D0236	5000615	10-72
CANDE OPERATION	D0240	5000615	10-72
CANDE OPERATION	D0276	5000615	10-72
CANDE OPERATION	D0277	5000615	10-72
CANDE OPERATION	D0282	5000615	10-72
CANDE OPERATION	D0283	5000615	10-72
COBOL REFERENCE	D0224	5000656	02-73
COBOL REFERENCE	D0226	5000656	02-73
COBOL REFERENCE	D0229	5000656	02-73
COBOL REFERENCE	D0247	5000656	02-73
COBOL REFERENCE	D0249	5000656	10-71
DATA MANAGEMENT	D0254	5000235	01-73
DATA MANAGEMENT	D0255	5000235	01-73
DATA MANAGEMENT	D0256	5000235	01-73
DATA MANAGEMENT	D0257	5000235	01-73
DATA MANAGEMENT	D0258	5000235	01-73
DATA MANAGEMENT	D0259	5000235	01-73
DATA MANAGEMENT	D0260	5000235	01-73
DATA MANAGEMENT	D0261	5000235	01-73
DATA MANAGEMENT	D0262	5000235	01-73
DATA MANAGEMENT	D0263	5000235	01-73

DOCUMENTS AFFECTED

DOCUMENT	SYSTEM NOTE	MARKETING NO.	MARKETING DATE
DATA MANAGEMENT	D0264	5000235	01-73
DATA MANAGEMENT	D0265	5000235	01-73
DATA MANAGEMENT	D0268	5000235	01-73
DATA MANAGEMENT	D0269	5000235	01-73
DATA MANAGEMENT	D0284	5000235	01-73
DATA MANAGEMENT	D0290	5000285	01-73
DATACOM FUNCTIONAL	D0279	5000060	12-70
DCALGOL LANGUAGE	D0149	5000052	08-71
DCALGOL LANGUAGE	D0176	5000052	08-71
DCALGOL LANGUAGE	D0192	5000052	08-71
DCALGOL LANGUAGE	D0199	5000052	08-71
DCALGOL LANGUAGE	D0200	5000052	08-71
DCALGOL LANGUAGE	D0220	5000052	08-71
DCALGOL LANGUAGE	D0252	5000052	08-71
DCALGOL LANGUAGE	D0274	5000052	08-71
DCALGOL LANGUAGE	D0285	5000052	08-71
DCALGOL LANGUAGE	D0286	5000052	08-71
ESPOL LANGUAGE	D0140	5000094	06-72
ESPOL LANGUAGE	D0156	5000094	06-72
ESPOL LANGUAGE	D0158	5000094	06-72
ESPOL LANGUAGE	D0166	5000094	06-72
ESPOL LANGUAGE	D0241	5000094	06-72
ESPOL LANGUAGE	D0242	5000094	06-72
ESPOL LANGUAGE	D0243	5000094	06-72
ESPOL LANGUAGE	D0244	5000094	06-72
ESPOL LANGUAGE	D0245	5000094	06-72
ESPOL LANGUAGE	D0252	5000094	06-72
ESPOL LANGUAGE	D0270	5000094	06-72
FORTRAN REFFRENCE	D0144	5000458	07-72
FORTRAN REFERENCE	D0146	5000458	07-72
FORTRAN REFERENCE	D0153	5000458	06-72
FORTRAN REFERENCE	D0157	5000458	06-72
FORTRAN REFERENCE	D0160	5000458	06-72

DOCUMENTS AFFECTED

DOCUMENT	SYSTEM NOTE	MARKETING NO.	MARKETING DATE
FORTRAN REFERENCE	D0165	5000458	07-72
FORTRAN REFERENCE	D0198	5000458	06-72
FORTRAN REFERENCE	D0228	5000458	07-72
FORTRAN REFERENCE	D0230	5000458	07-72
FORTRAN REFERENCE	D0231	5000458	07-72
FORTRAN REFERENCE	D0232	5000458	07-72
FORTRAN REFERENCE	D0271	5000458	06-72
FORTRAN REFERENCE	D0280	5000458	06-72
I-O SUBSYSTEM	D0185	5000185	07-71
I-O SUBSYSTEM	D0273	5000185	07-71
MARK 2.3. SW IMPR	D0150	5000367	10-72
MCP	D0167	5000086	12-71
MCP	D0171	5000086	11-70
MCP	D0172	5000086	12-71
MCP	D0184	5000086	12-71
MCP	D0196	5000086	12-71
MCP	D0227	5000086	11-70
MCP	D0233	5000086	11-70
MCP	D0234	5000086	11-70
MCP	D0235	5000086	11-70
MCSII USERS GUIDE	D0251	5000219	09-71
NDL	D0161	5000078	08-71
NDL	D0168	5000078	08-71
NDL	D0169	5000078	08-71
NDL	D0170	5000078	08-71
NDL	D0272	5000078	08-71
NDL	P0908	5000078	08-71
PL/I LANGUAGE	D0142	5000201	10-72
PL/I LANGUAGE	D0143	5000201	10-72
PL/I LANGUAGE	D0155	5000201	10-72
PL/I LANGUAGE	D0174	5000201	10-72
PL/I LANGUAGE	D0175	5000201	10-72
PL/I LANGUAGE	D0225	5000201	10-72

DOCUMENTS AFFECTED

DOCUMENT	SYSTEM NOTE	MARKETING NO.	MARKETING DATE
PL1 LANGUAGE	D0246	5000201	10-72
RJE	D0159	5000300	06-72
RJE	D0173	5000300	06-72
RJE	D0187	5000300	06-72
RJE	D0188	5000300	06-72
RJE	D0189	5000300	06-72
RJE	D0190	5000300	06-72
RJE	D0201	5000300	06-72
RJE	D0202	5000300	06-72
RJE	D0212	5000300	06-72
RJE	D0213	5000300	06-72
RJE	D0214	5000300	06-72
RJE	D0215	5000300	06-72
RJE	D0216	5000300	06-72
RJE	D0222	5000300	06-72
RJE	D0223	5000300	06-72
SORT PROGRAM	D0148	5000144	11-72
SYSTEM HANDBOOK	D0180	5000276	01-72
SYSTEM HANDBOOK	D0181	5000276	01-72
SYSTEM MISCELLANEA	D0150	5000367	10-72
SYSTEM MISCELLANEA	D0180	5000367	10-72
SYSTEM MISCELLANEA	D0181	5000367	10-72
SYSTEM MISCELLANEA	D0217	5000367	10-02
SYSTEM MISCELLANEA	D0248	5000367	10-72
SYSTEM MISCELLANEA	D0278	5000367	10-72
SYSTEM MISCELLANEA	D0282	5000367	10-72
SYSTEM MISCELLANEA	D0287	5000367	10-72

KWIC	SYSTEM NOTE	FUNCTION
<DATA-NAME> IS <MNEMONIC>	D0229	COBOL
<MNEMONIC> <DATA-NAME> IS	D0229	COBOL
$ FORMAT MODIFRS-FORTRAN K AND	D0271	ESPOLINTRN
"ASC" CHARACTER FUNCTION	D0203	BASIC
"BRUTAL" & "PEDANTIC" EXTEND	D0283	CANDE
"CALL SYSTEM" VERB	D0247	COBOL
"DEF" FUNCTIONS MULTIPLE STMT	D0182	BASIC
"END" AS RANGE STOP INDICATOR	D0181	BACKUP
"FIRST" DOLLAR CARD OPTION	D0153	FORTRAN
"HISTORY" TASK ATTRIBUTE	D0248	MCP
"IF" SYNTAX EXPANDED	D0205	BASIC
"INPUT" STATEMENT IN BASIC	D0209	BASIC
"LENGTH" STRING FUNCTION	D0154	BASIC
"OLD BASIC" DOLLAR OPTION	D0193	BASIC
"ON" STATEMENT SYNTAX	D0252	ALGOL
"OWNARRAYS" OPTION	D0157	FORTRAN
"PEDANTIC" EXTEND "BRUTAL" &	D0283	CANDE
"PRINT" STATEMENT IMPROVEMENTS	D0221	BASIC
"STIF" THE WORD INTRINSIC	D0244	ESPOL
"WRITEAFTER" DOLLAR CARD OPTN	D0253	ALGOL
"WRITESPO" PROCEDURE	D0185	MCP-I-O
ACCEPT STOP BY RSVP AND	D0213	RJE
ACCEPT AS BOOLEAN INTRINSIC	D0267	ALGOL
AE4 ARITHMETIC TEST	D0137	
AFTER MODIFY - STORE CURRENT	D0264	DM6700
ALGOL & FORTRAN VECTOR MODE IN	D0145	ALGOL
ALGOL - USR CONTROLD SEGMNTATN	D0278	ALGOL
ALGOL DUMPINFO AND LOADINFO IN	D0211	ALGOL
ALGOL TO ESPOL BINDING	D0156	BINDER
ANALYZER LOG	D0275	CANDE

KWIC SUBJECT INDEX

KWIC	SYSTEM NOTE	FUNCTION
APOSTROPHE - AS COMMENT SIGN	D0204	BASIC
ARITHMETIC TEST AE4	D0137	
ASSIGNED LP FORMMESSAGE	D0273	MCP
ATTRIBUTE "HISTORY" TASK	D0248	MCP
ATTRIBUTE IMPLEMENTED EXPLICIT	D0175	PL1
ATTRIBUTES NEW	D0226	COBOL
ATTRIBUTES FILE	D0235	MCP-I-O
ATTRIBUTES DCALGOL QUEUE	D0149	DCALGOL
ATTRIBUTES - CURRENTBLOCK FILE	D0196	MCP-I-O
ATTRIBUTES - TIMELIMIT FILE	D0191	MCP-I-O
ATTRIBUTES RESULT UPDATE LINE	D0285	MCP-DATACM
AX RSC INPUT MESSAGE	D0214	RJE
BACKUPS PRIORITY OF RJE	D0187	RJE
BASIC DET FUNCTION IN	D0194	BASIC
BASIC "INPUT" STATEMENT IN	D0209	BASIC
BASIC COTANGENT FUNCTION IN	D0195	BASIC
BASIC CHARACTER DATA EXTENSION	D0162	BASIC
BASIC" DOLLAR OPTION "OLD	D0193	BASIC
BDNAME VALUE AND	D0189	RJE
BETWEEN RJE STATIONS SS	D0190	RJE
BINDING PL1	D0143	PL1
BINDING ALGOL TO ESPOL	D0156	BINDER
BINDING INTERMED LEVEL GLOBALS	D0144	
BLANK FIELD ON FORMATTED INPUT	D0280	ESPOLINTRN
BLOCK USE ROUTINE SHORT	D0249	COBOL
BOOLEAN INTRINSIC ACCEPT AS	D0267	ALGOL
BUFFER CHAOS TRAP	D0277	CANDE
BYTE VARIABLE	D0168	NDL
CARD CHARACTER OPTIONS DOLLAR	D0228	FORTRAN
CARD FILE DMPRINTIT -	D0265	DMPRINTIT
CARD OPTION LIMIT DOLLAR	D0206	BASIC
CARD OPTION "FIRST" DOLLAR	D0153	FORTRAN
CARD OPTIONS FORTRAN DOLLAR	D0230	FORTRAN

KWIC	SYSTEM NOTE	FUNCTION
CARD OPTN "WRITEAFTER" DOLLAR	D0253	ALGOL
CARD PROCESSING DOLLAR	D0224	COBOL
CARD STATEMENT DOLLAR	D0152	BASIC
CARRIAGE CONTROL	D0170	NDL
CARRIAGE CONTROL VALUES	D0282	I-O
CASE STATEMENT SYNTAX	D0218	ALGOL
CHAOS TRAP BUFFER	D0277	CANDE
CHAR CONTINUATION MAT INPUT	D0198	ESPOLINTRN
CHARACTER DATA EXTENSION BASIC	D0162	BASIC
CHARACTER FUNCTION "ASC"	D0203	BASIC
CHARACTER OPTIONS DOLLAR CARD	D0228	FORTRAN
CODE FILES MULTIPLE MCP	D0184	MCP
CODE SWAPPING TIME SLICING AND	D0172	MCP
COMMAND EXCLUDE	D0186	CANDE
COMMENT SIGN APOSTROPHE - AS	D0204	BASIC
COMPACTION DM - DATA	D0259	DM6700
COMPILATION OF COMPILERS	D0217	
COMPILERS COMPILATION OF	D0217	
COMPILES SEPARATE	D0246	PL1
CONTINUATION MAT INPUT CHAR	D0198	ESPOLINTRN
CONTROL CARRIAGE	D0170	NDL
CONTROL VALUES CARRIAGE	D0282	I-O
CONTROLD SEGMNTATN ALGOL - USR	D0278	ALGOL
CORE DATA TRANSFER CORE TO	D0165	FORTRAN
CORE TO CORE DATA TRANSFER	D0165	FORTRAN
COTANGENT FUNCTION IN BASIC	D0195	BASIC
COUNT RECORD	D0179	BACKUP
CURRENT AFTER MODIFY - STORE	D0264	DM6700
CURRENTBLOCK FILE ATTRIBUTES<-	D0196	MCP-I-O
DATA COMPACTION DM -	D0259	DM6700
DATA EXTENSION BASIC CHARACTER	D0162	BASIC
DATA FUNCTIONS TIME AND	D0163	BASIC
DATA STMT EXTENSIONS RESTORE &	D0164	BASIC

KWIC SUBJECT INDEX

KWIC	SYSTEM NOTE	FUNCTION
DATA TRANSFER CORE TO CORE	D0165	FORTRAN
DATACOM ERROR LOGGING	D0234	MCP-DATACM
DCALGOL QUEUE ATTRIBUTES	D0149	DCALGOL
DCALGOL QUEUE TANKING	D0279	DATACOM
DCP FAULT REPORTING	D0240	CANDE
DCP FAULT RESULT	D0233	MCP-DATACM
DCPPROGEN UNITE NDL &	D0272	NDL
DCSTATUS SYSTEM	D0250	DCSTATUS
DCSYSTEMTABLES INTRINSIC	D0171	MCP-DATACM
DCWRITE FUNCTION UPDATE LINE	D0199	MCP-DATACM
DDL EXECUTION DM -	D0257	DM6700
DDL WARNING DM -	D0269	DM6700
DECK OUTPUT NON-SAVE	D0166	ESPOL
DECLARATION PROCEDURE	D0242	ESPOL
DECLARATIONS PLI FILE	D0225	PLI
DESIGN OF RECOVERY FOR DM6700	D0284	DM6700
DET FUNCTION IN BASIC	D0194	BASIC
DIALOUT ERROR RESULTS	D0192	DCPPROGEN
DJ SET WITH IA MODIFY ORDER OF	D0263	DM6700
DM - DATA COMPACTION	D0259	DM6700
DM - DDL EXECUTION	D0257	DM6700
DM - DDL WARNING	D0269	DM6700
DM - NEW STATUS	D0290	DM6700
DM - REQUEST HANDLER EXECUTION	D0255	DM6700
DM - SDL EXECUTION	D0258	DM6700
DM - SDL IMPROVEMENTS	D0256	DM6700
DMPRINTIT - CARD FILE	D0265	DMPRINTIT
DMUPDATE	D0254	DM6700
DM6700 DESIGN OF RECOVERY FOR	D0284	DM6700
DOLLAR CARD CHARACTER OPTIONS	D0228	FORTRAN
DOLLAR CARD OPTION LIMIT	D0206	BASIC
DOLLAR CARD OPTION "FIRST"	D0153	FORTRAN
DOLLAR CARD OPTIONS FORTRAN	D0230	FORTRAN

KWIC	SYSTEM NOTE	FUNCTION
FAULT RESULT DCP	D0233	MCP-DATACM
FEATURES FOR SYSTEM MCSII NEW	D0251	MCSII
FIELD ON FORMATTED INPUT BLANK	D0280	ESPOLINTRN
FILE DMPRINTIT - CARD	D0265	DMPRINTIT
FILE ATTRIBUTES	D0235	MCP-I-O
FILE ATTRIBUTES - CURRENTBLOCK	D0196	MCP-I-O
FILE ATTRIBUTES - TIMELIMIT	D0191	MCP-I-O
FILE DECLARATIONS PLI	D0225	PLI
FILE IDENTIFIER	D0232	FORTRAN
FILE SEARCHING PRINT	D0215	RJE
FILES MULTIPLE MCP CODE	D0184	MCP
FLOW MANAGEMENT WORK	D0210	MCP
FOR DM6700 DESIGN OF RECOVERY	D0284	DM6700
FOR EMBEDDED SETS GLOBAL	D0260	DM6700
FOR RSVP & SNTX HISTRY PROCSNG	D0238	CANDE
FOR SYSTEM MCSII NEW FEATURES	D0251	MCSII
FORMAL PROCEDURES	D0147	ALGOL
FORMAT MODIFRS-FORTRAN K AND $	D0271	ESPOLINTRN
FORMATTED INPUT BLANK FIELD ON	D0280	ESPOLINTRN
FORMATTER NEW	D0150	ESPOLINTRN
FORMMESSAGE ASSIGNED LP	D0273	MCP
FORMMESSAGE THROUGH RJE	D0188	RJE
FORTRAN STATISTICS IN	D0287	FORTRAN
FORTRAN DOLLAR CARD OPTIONS	D0230	FORTRAN
FORTRAN OPTIMIZATION	D0146	FORTRAN
FORTRAN VECTOR MODE IN ALGOL &	D0145	ALGOL
FUNCTION NUM	D0197	BASIC
FUNCTION "ASC" CHARACTER	D0203	BASIC
FUNCTION "LENGTH" STRING	D0154	BASIC
FUNCTION UPDATE LINE UCWRITE	D0199	MCP-DATACM
FUNCTION MOVE STATION-UCWRITE	D0200	MCP-DATACM
FUNCTION IN BASIC DET	D0194	BASIC
FUNCTION IN BASIC COTANGENT	D0195	BASIC

KWIC SUBJECT INDEX

KWIC	SYSTEM NOTE	FUNCTION
FUNCTIONS STRING	D0178	BASIC
FUNCTIONS TIME AND DATA	D0163	BASIC
FUNCTIONS MULTIPLE STMT "DEF"	D0182	BASIC
GLOBAL FOR EMBEDDED SETS	D0260	DM6700
GLOBALS BINDING INTERMED LEVEL	D0144	
GO TO STATEMENT SWITCH	D0169	NDL
HANDLER EXECUTION DM - REQUEST	D0255	DM6700
HISTRY PROCSNG FOR RSVP & SNTX	D0238	CANDE
I MODEL	D0266	ALGOL
IA MODIFY ORDER OF DJ SET WITH	D0263	DM6700
ID PROGRAM AND PATCH	D0158	ALGOL
IDENTIFIER FILE	D0232	FORTRAN
IDENTIFIERS TRUNCATED	D0160	FORTRAN
IMPLEMENTATION SM MESSAGE	D0201	RJE
IMPLEMENTED ENTRY VARIABLES	D0174	PL1
IMPLEMENTED EXPLICIT ATTRIBUTE	D0175	PL1
IMPROVEMENTS PL1 10	D0142	PL1
IMPROVEMENTS DM - SDL	D0256	DM6700
IMPROVEMENTS "PRINT" STATEMENT	D0221	BASIC
INACTIVE STATION RSC OUTPUT TO	D0216	RJE
INCREASE MAXSTATIONS, MAXTASKS	D0236	CANDE
INDEX PARAMETERS SDL -	D0268	DM6700
INDICATOR "END" AS RANGE STOP	D0181	BACKUP
INHIBIT SYNC EDIT	D0161	DCPPROGEN
INITIALIZE PRIMARY QUEUE	D0286	MCP-DATACM
INITIALIZE RETRY	D0167	NDL
INPUT BLANK FIELD ON FORMATTED	D0280	ESPOLINTRN
INPUT CHAR CONTINUATION MAT	D0198	ESPOLINTRN
INPUT MESSAGE AX RSC	D0214	RJE
INPUT-OUTPUT STATEMENTS	D0208	BASIC
INSTALLATION INTRINSICS	D0220	ALGOL
INTERMED LEVEL GLOBALS BINDING	D0144	
INTRINSIC DCSYSTEMTABLES	D0171	MCP-DATACM

KWIC	SYSTEM NOTE	FUNCTION
INTRINSIC ACCEPT AS BOOLEAN	D0267	ALGOL
INTRINSIC "STFF" THE WORD	D0244	ESPOL
INTRINSICS INSTALLATION	D0220	ALGOL
IO IMPROVEMENTS PLI	D0142	PLI
IS <MNEMONIC> <DATA-NAME>	D0229	COBOL
ITEMS REQUIRED	D0262	DM6700
JOBS WITH PARAMETERS RUN	D0159	RJE
K AND $ FORMAT MODIFRS-FORTRAN	D0271	ESPOLINTRN
KEY SPECIFIER LANGUAGE	D0180	BACKUP
LANGUAGE KEY SPECIFIER	D0180	BACKUP
LEVEL GLOBALS BINDING INTERMED	D0144	
LIMIT DOLLAR CARD OPTION	D0206	BASIC
LINE ATTRIBUTES RESULT UPDATE	D0285	MCP-DATACM
LINE DCWRITE FUNCTION UPDATE	D0199	MCP-DATACM
LINE-STATION READY	D0276	CANDE
LOADINFO IN ALGOL DUMPINFO AND	D0211	ALGOL
LOG ANALYZER	D0275	CANDE
LOGANALYZER THROUGH RJE	D0223	RJE
LOGGING MCS	D0227	MCP
LOGGING DATACOM ERROR	D0234	MCP-DATACM
LOGGING; SESSION NUMBRS; SPLIT	D0239	CANDE
LP FORMMESSAGE ASSIGNED	D0273	MCP
MAINTENANCE EU	D0281	SCR
MANAGEMENT WORK FLOW	D0210	MCP
MAT INPUT CHAR CONTINUATION	D0198	ESPOLINTRN
MAXSTATIONS, MAXTASKS INCREASE	D0236	CANDE
MAXTASKS INCREASE MAXSTATIONS,	D0236	CANDE
MCP CODE FILES MULTIPLE	D0184	MCP
MCS LOGGING	D0227	MCP
MCSII NEW FEATURES FOR SYSTEM	D0251	MCSII
MEMORYDUMP	D0274	ALGOL
MESSAGE RF RSC	D0222	RJE
MESSAGE AX RSC INPUT	D0214	RJE

KWIC	SYSTEM NOTE	FUNCTION
MESSAGE IMPLEMENTATION SM	D0201	RJE
MESSAGE TO RJE DP	D0173	RJE
MODE IN ALGOL & FORTRAN VECTOR	D0145	ALGOL
MODEL I	D0266	ALGOL
MODIFRS=FORTRAN K AND $ FORMAT	D0271	ESPOLINTRN
MODIFY = STORE CURRENT AFTER	D0264	DM6700
MODIFY ORDER OF DJ SET WITH IA	D0263	DM6700
MOVE STATION=DCWRITE FUNCTION	D0200	MCP=DATACM
MULTIPLE MCP CODE FILES	D0184	MCP
MULTIPLE STMT "DEF" FUNCTIONS	D0182	BASIC
MULTIPLICATN EXPONENTIATION &	D0243	ESPOL
MULTISTATION RJE TERMINALS	D0212	RJE
NAMES STRING VARIABLE	D0177	BASIC
NDL & DCPPROGEN UNITE	D0272	NDL
NEW ATTRIBUTES	D0226	COBOL
NEW FEATURES FOR SYSTEM MCSII	D0251	MCSII
NEW FORMATTER	D0150	ESPOLINTRN
NEW STATUS DM =	D0290	DM6700
NON=SAVE DECK OUTPUT	D0166	ESPOL
NUM FUNCTION	D0197	BASIC
NUMBRS; SPLIT LOGGING; SESSION	D0239	CANDE
ON FORMATTED INPUT BLANK FIELD	D0280	ESPOLINTRN
OPERATORS RELATIONAL	D0207	BASIC
OPTIMIZATION FORTRAN	D0146	FORTRAN
OPTION "OWNARRAYS"	D0157	FORTRAN
OPTION LIMIT DOLLAR CARD	D0206	BASIC
OPTION WRITEAFTER DOLLAR	D0270	ESPOL
OPTION "FIRST" DOLLAR CARD	D0153	FORTRAN
OPTION "OLD BASIC" DOLLAR	D0193	BASIC
OPTIONS DP	D0202	RJE
OPTIONS ESPOL DOLLAR	D0241	ESPOL
OPTIONS FORTRAN DOLLAR CARD	D0230	FORTRAN
OPTIONS DOLLAR CARD CHARACTER	D0228	FORTRAN

KWIC SUBJECT INDEX

KWIC SUBJECT INDEX

KWIC	SYSTEM NOTE	FUNCTION
RECORD COUNT	D0179	BACKUP
RECOVERY FOR DM6700 DESIGN OF	D0284	DM6700
RELATIONAL OPERATORS	D0207	BASIC
REPORTING DCP FAULT	D0240	CANDE
REQUEST HANDLER EXECUTION DM -	D0255	DM6700
REQUIRED ITEMS	D0262	DM6700
RESTORE & DATA STMT EXTENSIONS	D0164	BASIC
RESULT DCP FAULT	D0233	MCP-DATACM
RESULT UPDATE LINE ATTRIBUTES	D0285	MCP-DATACM
RESULTS DIALOUT ERROR	D0192	DCPPROGEN
RETRY INITIALIZE	D0167	NDL
RJE DP MESSAGE TO	D0173	RJE
RJE FORMMESSAGE THROUGH	D0188	RJE
RJE LOGANALYZER THROUGH	D0223	RJE
RJE BACKUPS PRIORITY OF	D0187	RJE
RJE STATIONS SS BETWEEN	D0190	RJE
RJE TERMINALS MULTISTATION	D0212	RJE
ROUTINE SHORT BLOCK USE	D0249	COBOL
RSC INPUT MESSAGE AX	D0214	RJE
RSC MESSAGE RF	D0222	RJE
RSC OUTPUT TO INACTIVE STATION	D0216	RJE
RSVP & SNTX HISTRY PROCSNG FOR	D0238	CANDE
RSVP AND ACCEPT STOP BY	D0213	RJE
RUN JOBS WITH PARAMETERS	D0159	RJE
SDL - INDEX PARAMETERS	D0268	DM6700
SDL EXECUTION DM -	D0258	DM6700
SDL IMPROVEMENTS DM -	D0256	DM6700
SEARCHING PRINT FILE	D0215	RJE
SEGMNTATN ALGOL - USR CONTROLD	D0278	ALGOL
SEPARATE COMPILES	D0246	PL1
SESSION NUMBRS; SPLIT LOGGING;	D0239	CANDE
SET WITH IA MODIFY ORDER OF DJ	D0263	DM6700
SETS GLOBAL FOR EMBEDDED	D0260	DM6700

KWIC SUBJECT INDEX

KWIC	SYSTEM NOTE	FUNCTION
SHORT BLOCK USE ROUTINE	D0249	COBOL
SIGN APOSTROPHE - AS COMMENT	D0204	BASIC
SLICING AND CODE SWAPPING TIME	D0172	MCP
SM MESSAGE IMPLEMENTATION	D0201	RJE
SNTX HISTRY PROCSNG FOR RSVP &	D0238	CANDE
SORT TIMING PREDICTOR	D0148	PREDICTSORT
SPECIFIER LANGUAGE KEY	D0180	BACKUP
SPLIT LOGGING; SESSION NUMBRS;	D0239	CANDE
SS BETWEEN RJE STATIONS	D0190	RJE
STATEMENT TRACE	D0231	FORTRAN
STATEMENT DOLLAR CARD	D0152	BASIC
STATEMENT SWITCH GO TO	D0169	NDL
STATEMENT IMPROVEMENTS "PRINT"	D0221	BASIC
STATEMENT IN BASIC "INPUT"	D0209	BASIC
STATEMENT SYNTAX "ON"	D0252	ALGOL
STATEMENT SYNTAX CASE	D0218	ALGOL
STATEMENTS INPUT-OUTPUT	D0208	BASIC
STATION ERROR SUBTRACT	D0176	MCP-DATACM
STATION RSC OUTPUT TO INACTIVE	D0216	RJE
STATION-DCWRITE FUNCTION MOVE	D0200	MCP-DATACM
STATIONS SS BETWEEN RJE	D0190	RJE
STATISTICS IN FORTRAN	D0287	FORTRAN
STATUS DM - NEW	D0290	DM6700
STMT "DEF" FUNCTIONS MULTIPLE	D0182	BASIC
STMT EXTENSIONS RESTORE & DATA	D0164	BASIC
STOP BY RSVP AND ACCEPT	D0213	RJE
STOP INDICATOR "END" AS RANGE	D0181	BACKUP
STORE CURRENT AFTER MODIFY -	D0264	DM6700
STRING FUNCTION "LENGTH"	D0154	BASIC
STRING FUNCTIONS	D0178	BASIC
STRING VARIABLE NAMES	D0177	BASIC
STRUCTURE RANDOM	D0261	DM6700
SUBTRACT STATION ERROR	D0176	MCP-DATACM

KWIC SUBJECT INDEX

KWIC	SYSTEM NOTE	FUNCTION
SWAPPING	D0282	CANDE
SWAPPING TIME SLICING AND CODE	D0172	MCP
SWITCH GO TO STATEMENT	D0169	NDL
SYNC EDIT INHIBIT	D0161	DCPPROGEN
SYNTAX EXPANDED "IF"	D0205	BASIC
SYNTAX "ON" STATEMENT	D0252	ALGOL
SYNTAX CASE STATEMENT	D0218	ALGOL
SYSTEM DCSTATUS	D0250	DCSTATUS
SYSTEM MCSII NEW FEATURES FOR	D0251	MCSII
SYSTEM" VERB "CALL	D0247	COBOL
TANKING DCALGOL QUEUE	D0279	DATACOM
TASK ATTRIBUTE "HISTORY"	D0248	MCP
TASKVALUE	D0219	ALGOL
TERMINALS MULTISTATION RJE	D0212	RJE
TEST AE4 ARITHMETIC	D0137	
THROUGH RJE FORMMESSAGE	D0188	RJE
THROUGH RJE LOGANALYZER	D0223	RJE
TIME AND DATA FUNCTIONS	D0163	BASIC
TIME SLICING AND CODE SWAPPING	D0172	MCP
TIMELIMIT FILE ATTRIBUTES -	D0191	MCP-I-O
TIMING PREDICTOR SORT	D0148	PREDICTSORT
TRACE STATEMENT	D0231	FORTRAN
TRANSFER CORE TO CORE DATA	D0165	FORTRAN
TRAP BUFFER CHAOS	D0277	CANDE
TRUNCATED IDENTIFIERS	D0160	FORTRAN
UNITE NDL & DCPPROGEN	D0272	NDL
UPDATE LINE ATTRIBUTES RESULT	D0285	MCP-DATACM
UPDATE LINE DCWRITE FUNCTION	D0199	MCP-DATACM
USE ROUTINE SHORT BLOCK	D0249	COBOL
USR CONTROLD SEGMNTATN ALGOL -	D0278	ALGOL
VALUE AND BDNAME	D0189	RJE
VALUES CARRIAGE CONTROL	D0282	I-O
VARIABLE BYTE	D0168	NDL

KWIC SUBJECT INDEX

KWIC		SYSTEM NOTE	FUNCTION
VARIABLE NAMES	STRING	D0177	BASIC
VARIABLES IMPLEMENTED	ENTRY	D0174	PL1
VARIANT	PAGESKIP	D0237	CANDE
VECTOR MODE IN ALGOL & FORTRAN		D0145	ALGOL
VERB	"CALL SYSTEM"	D0247	COBOL
WARNING	DM - DDL	D0269	DM6700
WITH IA MODIFY ORDER OF DJ SET		D0263	DM6700
WITH PARAMETERS	RUN JOBS	D0159	RJE
WORD INTRINSIC "STFF"	THE	D0244	ESPOL
WORK FLOW MANAGEMENT		D0210	MCP
WRITEAFTER DOLLAR OPTION		D0270	ESPOL