B-1700 COBOL COMPILER DOCUMENTATION

DECEMBER, 1973

DESIGN FEATURES.

SOURCE LANGUAGE SELECTION.

IN THE EARLY DESIGN STAGES OF THE COBOL COMPILER, CONSIDERATION WAS GIVEN TO IMPLEMENTING PROPER SUBSETS OF THE COBOL SOURCE LANGUAGE AS SPECIFIED IN THE USA STANDARD COBOL MANUAL. IT WAS FOUND THAT IMPLEMENTATION OF A SUBSET HAD THE FOLLOWING DISADVANTAGES:

1. THE SUBSETS SPECIFIED WERE VERY LIMITED WITH MANY RESTRICTORS. EG: NUMBER OF UNIQUE DATA-NAMES, NO NESTED REDEFINES, ETC.

2. THE LANGUAGE ELEMENTS SEEMED TO BE ARBITARILY SELECTED ON THE BASIS OF REDUNDANCY RATHER THAN DIFFICULTY OF IMPLEMENTATION.

3. THE SUBSETS SPECIFIED WERE DIFFICULT TO USE. E.G. THE USER HAD TO REMEMBER IF HE COULD SAY "ZERO" OR "ZEROES".

4. ALTHOUGH UPWARD COMPATABILITY TO B3500 COBOL COULD BE ASSURED, COMPATABILITY IN THE OTHER DIRECTION WAS NOT SO SAFE.

5. THE COST OF DEVELOPMENT, MAINTAINENCE, AND CHECK OUT OF A MULTIPLICITY OF COMPILERS SEEMED UNREASONABLE.

IT WAS THEREFORE DECIDED THAT THE HIGH LEVEL OF EACH LANGUAGE MODULE WOULD BE IMPLEMENTED WITH AN EYE TOWARD COMPATABILITY WITH THE B3500 LANGUAGE EXTENSIONS. AN ATTEMPT TO "SHOEHORN" THE COMPILER INTO SMALL MEMORY MACHINES WOULD THEN BE CONSIDERED WHEN THE IMPLEMENTATION REQUIREMENTS WERE KNOW.

MULTI-PHASE COMPILATION.

THE ADVANTAGES OF "MULTI-PASS" OR MULTI-PHASE COMPILATION ARE:

1. BETTER UTILIZATION OF SMALL MEMORY STORES. THE ORIGINAL TEXT IS TRANSFORMED TO A MORE CONVENIENT FORM BY PASSING IT AGAINST A PART OF THE COMPILER. THE TRANSFORMED TEXT IS PASSED TO THE NEXT PHASE VIA AN INTERMEDIATE WORK FILE. IN THIS WAY THE DATA IS MANAGED IN A SOMEWHAT SEQUENTIAL MANNER, MINIMIZING RANDOM OVERLAYING AS IN A DATA PAGING OR VITUAL MEMORY MANAGEMENT SYSTEM. IN THE SAME WAY CODE OVERLAYS ARE MINIMIZED BECAUSE ONLY A PART OF THE CODE IS INVOKED TO TRANSFORM ALL THE DATA FOR THE PHASE.

2. IT IS POSSIBLE FOR ONE PHASE TO BE CODED AND CHECKED OUT BEFORE A PRIOR PHASE BY BUILDING THE INTERMEDIATE FILES REQUIRED WITH ANOTHER PROGRAM. THIS IS ESPECIALLY IMPORTANT WHEN SIMULATING WHERE THE COST OF SIMULATING ALL PRIOR PHASES IS PROHIBITIVE.

THE DISADVANTAGES OF MULTI-PHASE COMPILATION ARE:

1. THE FORMAT OF EACH TOKEN , AS IT APPEARS TO EACH PASS, MUST BE WELL DEFINED. IF TOKEN FORMATS CHANGE, THERE ARE MANY PLACES THAT CAN BE AFFECTED.

2. EXTRA CODE IS REQUIRED IN EACH PHASE TO GET AND PUT THE TOKENS.

3. IF A LARGE MEMORY IS AVAILABLE, THERE IS NO EASY WAY TO COMBINE PHASES. E.G. LABEL AND DATA-NAME QUALIFICATION RESOLUTION COULD BE COMBINED IF THERE WAS SPACE FOR THE INTERMEDIATE TABLES.

4. A CERTAIN FIXED OVERHEAD IS REQUIRED FOR OPENING AND CLOSING OF THE INTERMEDIATE FILES EVEN FOR A MINIMUM SOURCE LANGUAGE DECK.

DEBUGGING CAPABILITIES.

A VARIETY OF DEBUGGING OUTPUT HAS BEEN INCORPORATED IN THE COBOL COMPILER IN ORDER TO MINIMIZE THE DEBUGGING ERROR DETECTION AND ERROR CORRECTION TIME. THIS OUTPUT IS PRINTED WHEN THE FOLLOWING DOLLAR CARD RESERVED WORDS ARE USED:

<NO> PARSE: INITIAL PARSING

<NO> DICT: DICTIONARY PROCESSING

<NO> DNQUAL: DATA-NAME QUALIFICATION RESOLUTION

<NO> LQUAL: LABEL QUALIFICATION RESOLUTION

<NO> MOIGE: MERGE(NOTE: "MERGE" HAS ANOTHER MEANING)

<NO> DATSYN: DATA DIVISION SYNTAX CHECKING

<NO> EXPLODE: EXPLODE

<NO> PROSYN: PROCEDURE DIVISION SYNTAX CHECKING

<NO> CODEGEN: CODE GENERATION

<NO> FIXUP: FIXUP THE CODEFILE

WHEN THIS OPTION IS USED, A FORMATTED TOKEN AS SEEN BY THE GET AND PUT PROCEDURES FOR THAT PHASE IS PRINTED. IN ADDITION, THE NAME OF EACH PROCEDURE AND ANY PERTINANT VARIABLES IS PRINTED UPON ENTERING THE PROCEDURE. MOST RECURSIVE PROCEDURES ALSO MONITOR THEIR EXIT POINTS FOR EASE OF DEBUGGING. IT IS POSSIBLE TO SEE EACH PROCESS, FOR THE DURATION OF SEVERAL CARDS, BY TURNING MONITOR ON OR OFF FOR THE DESIRED PHASE(S).

DYNAMIC MEMORY.

WHEN DESIGNING A COMPILER FOR MANY MEMORY CONFIGURATIONS, IT IS IMPORTANT TO EFFECTIVELY UTILIZE ADDITIONAL MEMORY IF IT IS AVAILABLE. THE COBOL COMPILER ACCOMPLISHES THIS BY MANAGING THE DYNAMIC SPACE DECLARED AT COMPILE TIME.

A CERTAIN MINIMUM SPACE IS REQUIRED FOR BUILDING LISTS AND TABLES DURING THE DIFFERENT PHASES. ASSOCIATED WITH THIS MINIMUM SPACE ARE LIMITS E.G. THE NUMBER OF DATA-NAMES, PROCEDURE-NAMES, ETC. IF A PARTICULAR SOURCE PROGRAM EXCEEDS ANY OF THESE LIMITS, MORE SPACE MUST BE DEDICATED BY INCREASING THE DYNAMIC SPACE AND RECOMPILING. AN ATTEMPT HAS BEEN MADE TO MAKE THESE RESTRICTIONS "REASONABLE".

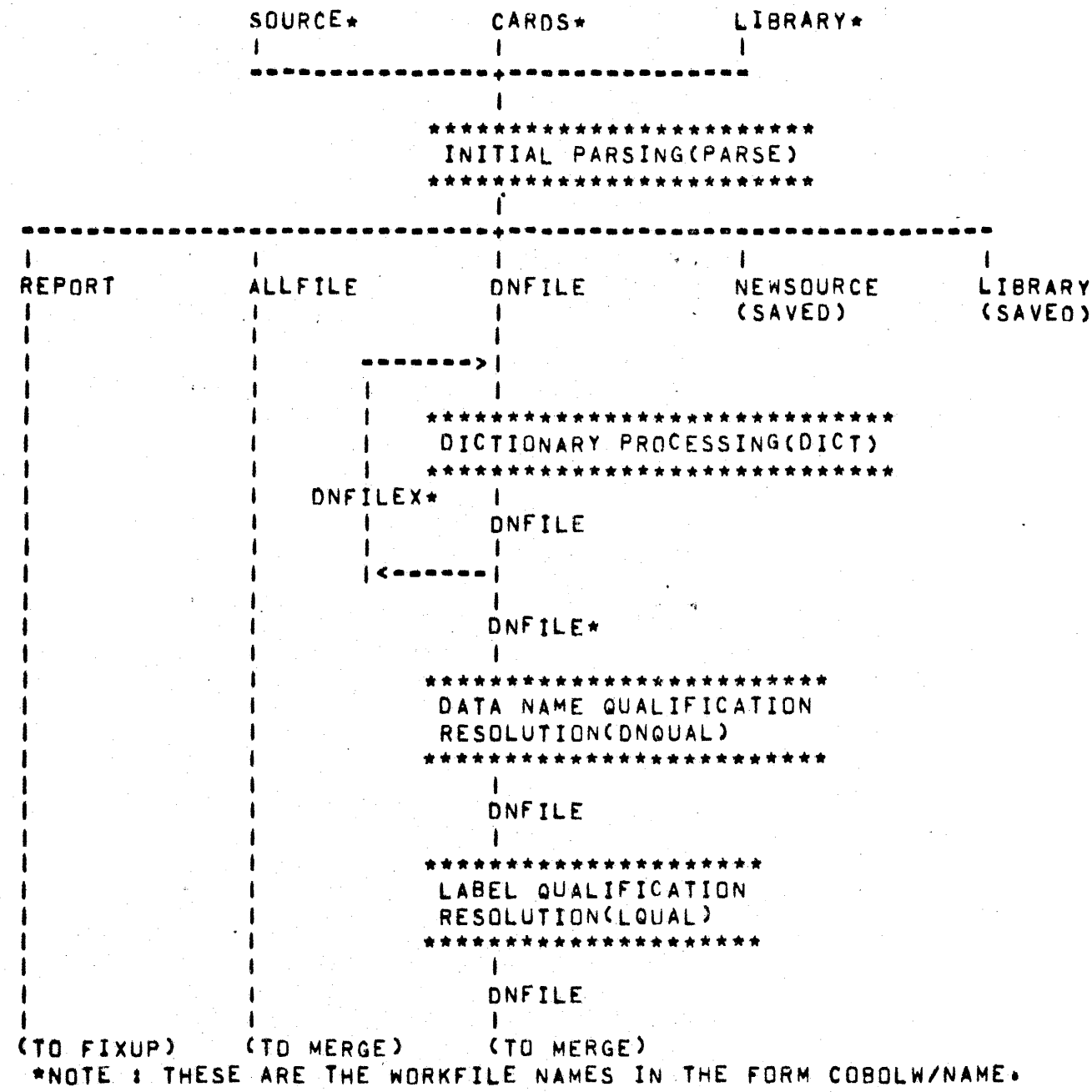
IF MORE SPACE IS AVAILABLE, THE COMPILER IS DESIGNED TO USE THAT SPACE FOR A SIGNIFICANT SPEED GAIN.

INTERMEDIATE FILE DESIGN.

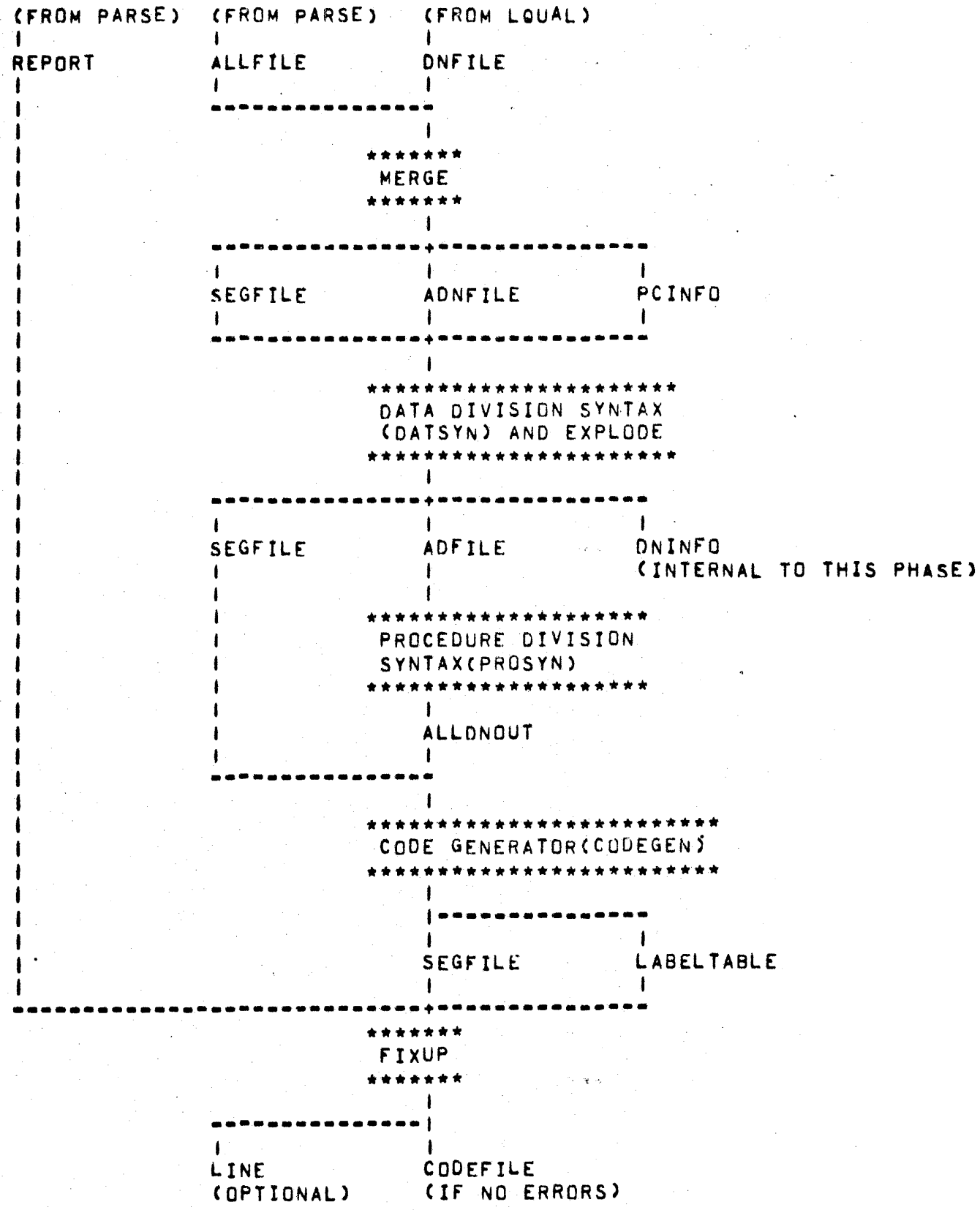
IN ORDER TO MINIMIZE THE INTERMEDIATE FILE SIZES, INFORMATION ABOUT THE ORIGINAL TEXT IS DISTRIBUTED TO SEVERAL FILES WHICH ARE ORDERED TO EACH OTHER. THE INDIVIDUAL FILES ARE THEN PROCESSED WITHOUT THE NEED TO COPY EXTRANEOUS INFORMATION. WHEN ALL OF THE TRANSFORMATIONS ARE COMPLETE THE FILES ARE MERGED.

CARE MUST BE TAKEN TO PRESERVE THE ORIGINAL ORDERING OF THE FILES.

THE DIAGRAM BELOW SHOWS THE FLOW OF THE INTERMEDIATE FILES FROM ONE PHASE TO ANOTHER. THE FIRST RECORD OF THE SEGFILE IS USED TO PASS GLOBAL DATA BETWEEN PHASES E.G. THE DONT.GENERATE.CODE FLAG, THE ABORT FLAG IN CASE OF A DRASTIC ERROR, ETC. THIS ASPECT OF THE SEGFILE IS NOT SHOWN.



*NOTE : THESE ARE THE WORKFILE NAMES IN THE FORM COBOLW/NAME.


(FROM PARSE)
(FROM PARSE)
(FROM LQUAL)
REPORT
ALLFILE
DNFILE
MERGE
SEGFILE
ADNFILE
PCINFO
DATA DIVISION SYNTAX
(DATSYN) AND EXPLODE
SEGFILE
ADFILE
DNINFO
(INTERNAL TO THIS PHASE)
PROCEDURE DIVISION
SYNTAX(PROSYN)
ALLDNOUT
CODE GENERATOR(CODEGEN)
SEGFILE
LABELTABLE
FIXUP
LINE
(OPTIONAL)
CODEFILE
(IF NO ERRORS)

INITIAL PARSING (PARSE).

INPUT FILES.

CARDS(READER):	SOURCE CARD IMAGES OR PATCH CARDS.
SOURCE(OPTIONAL DISK):	SOURCE CARD IMAGES TO WHICH PATCHES MAY BE APPLIED.
LIBRARY(OPTIONAL DISK):	SOURCE CARD IMAGES MERGED WITH THE PRIMARY FILE WHEN A COPY STATEMENT IS ENCOUNTERED.

OUTPUT FILES.

NEWSOURCE(OPTIONAL DISK):	SOURCE CARD IMAGES TO WHICH PATCH CARDS HAVE BEEN APPLIED.
LIBRARY(OPTIONAL DISK):	NEW LIBRARY FILE(S) CREATED WHEN "L" SPECIFIED IN COLUMN 7.
REPORT(DISK):	ALL CARD IMAGES PROCESSED INCLUDING DOLLAR CARDS, LIBRARY CARDS AND PATCH CARDS. THIS FILE IS USED TO PRINT THE OPTIONAL LISTING OR ERROR REPORT.
ALLFILE(DISK):	CONTAINS ALL CONSTANT INFORMATION ABOUT EACH TOKEN PROCESSED. THIS BECOMES THE CONTROLLING FILE DURING THE MERGE PHASE.
DNFILE(DISK):	CONTAINS PICTURE STRINGS AND ALL VARIABLE NAMES ISOLATED. E.G. SECTION-NAMES, PARAGRAPH-NAMES, AND DATA-NAMES.

GENERAL FUNCTIONS.

1. MERGE SOURCE LANGUAGE INPUTS INCLUDING LIBRARY CARD IMAGES.
2. SCAN AND ISOLATE BASIC SYMBOLS E.G. WORDS, INTEGERS, NON-NUMERIC LITERALS, NUMBERS, ETC.

3. LOOK UP RESERVED WORDS IN A RESERVED WORD LIST.

4. LOOK UP WORDS IN THE OPTIONAL COPY REPLACING LIST OR THE OPTIONAL SPECIAL NAMES LIST.

5. VERIFY THAT ALL FOUR DIVISIONS ARE PRESENT AND IN THE PROPER ORDER. IF THIS MUCH IS NOT CORRECT, THE ABORT FLAG IS SET TO BYPASS THE REGULAR PROCESSING AND SHORTCUT TO THE FIXUP PHASE.

RESERVED WORD LOOKUP.

IN ORDER TO MINIMIZE THE MEMORY REQUIRED TO HOLD THEM, THE COBOL RESERVED WORDS ARE ORGANIZED INTO TWO PRIMARY LISTS:

1. IDENTIFICATION THRU WORKING-STORAGE.

2. PROCEDURE DIVISION RESERVED WORDS.

THE WORDS ARE ARRANGED IN SUBLISTS BY LENGTH, E.G. "IS", "BY", "OF", ETC. APPEAR WITH OTHER TWO CHARACTER RESERVED WORDS. A GUESS AS TO THE FREQUENCY OF USE IS APPLIED TO EACH SUBLIST WITH THE MORE FREQUENT WORDS APPEARING AT THE BEGINNING OF THE SUBLIST.

WHEN A MATCH IS FOUND BY SEARCHING THE SUBLIST SEQUENTIALLY, THE RESERVED WORD KEY AND CATEGORY IS PROVIDED. NOISE WORDS, NOTE SENTENCES, AND NOTE PARAGRAPHS ARE DELETED WHEN DETECTED.

SPECIAL FUNCTIONS.

THERE ARE CERTAIN SPECIAL FUNCTIONS ACCOMPLISHED IN THIS PHASE. THESE APPEAR BELOW ACCORDING TO THE DIVISION, SECTION, OR PARAGRAPH HEADER ASSOCIATED WITH THE FUNCTION.

MONITOR SENTENCE.

1. VERIFY THAT A FILE-NAME IS DECLARED.

2. IDENTIFY AND MARK DATA-NAMES AND PROCEDURE-NAMES TO BE MONITORED.

3. PASS THESE NAMES TO THE MERGE PHASE AS NON-NUMERIC LITERALS ON THE ALLFILE.

4. IF THE WORD "ALL" APPEARS AS THE PROCEDURE-NAME LIST, THE NAME OF EACH SECTION AND PARAGRAPH IS PASSED TO THE MERGE PHASE AS A NON-NUMERIC LITERAL ON THE ALLFILE.

DATE-COMPILED.

1. SCAN TO THE END OF THE SENTENCE.

2. BUILD AN IMAGE CONTAINING DATE AND TIME AND OUTPUT IT TO THE REPORT FILE.

OBJECT-COMPUTER.

1. VERIFY AND SAVE THE MEMORY SIZE VALUE(IN CHARACTERS) FOR THE FIXUP PHASE.

2. SAVE SEGMENT-LIMIT VALUE.

3. SAVE DATA SEGMENT-LIMIT VALUE.

SPECIAL-NAMES.

1. SAVE THE CURRENCY SIGN VALUE.

2. SAVE THE DECIMAL-POINT IS COMMA VALUE.

3. BUILD THE MNEMONIC NAMES LIST.

DATA DIVISION.

1. ASSIGN AN OCUR NUMBER TO EACH FILE-NAME AND DATA-NAME DECLARED. THIS NUMBER BEGINS WITH 1 AND INCREASES BY 1 (0 IS RESERVED FOR ERROR REPORTING). THIS OCUR BECOMES THE INTERNAL REPRESENTATION FOR THE DATA-NAME AFTER THE DICTIONARY PROCESSING PHASE. FILLER ENTRIES ARE NOT ASSIGNED AN OCUR UNLESS THEY ARE 01 FILLER ... ENTRIES OF A FILE.

2. OUTPUT A DUMMY FD TO CORRESPOND TO THE WORKING-STORAGE SECTION HEADER. THIS DUMMY FD IS REQUIRED FOR THE DATA-NAME QUALIFICATION RESOLUTION PHASE.

FILE SECTION.

1. PARSE THE FILE-NAME OF AN FD OR SD DECLARATION. SET LEVEL = 0 AND THE APPROPRIATE FLAGS TRUE.

2. DROP THE DATA RECORD(S) CLAUSE AND ITS OPERANDS.

3. MARK THE OPERANDS OF THE LABEL RECORD(S) CLAUSE.

4. PARSE THE ITEM DESCRIPTIONS OF THE FILE RECORD (SEE WORKING-STORAGE SECTION).

WORKING-STORAGE SECTION.

1. PARSE ITEM DESCRIPTIONS.

2. ISOLATE AND VERIFY PROPER LEVEL INDICATOR.

3. RECOGNIZE AND MARK FILLER ENTRIES.

4. MARK THE OPERANDS OF A REDEFINES OR RENAMES CLAUSE.

5. RECOGNIZE A PICTURE DECLARATION AND CONTROL THE SCANNER TO ISOLATE A PC STRING.

6. MARK INDEX NAMES.

7. MARK ALL ITEMS THAT ARE NOT "CORRESPONDING" CANDIDATES. E.G. ENTRIES WITH FILLER SPECIFIED, LEVEL = 66, 77, OR 88, ENTRIES CONTAINING AN OCCURS OR REDEFINES CLAUSE. THESE ARE NOT CONSIDERED FOR SELECTION OF CORRESPONDING PAIRS.

PROCEDURE DIVISION.

1. RECOGNIZE AND MARK SECTION NAMES AND PARAGRAPH NAMES AND ASSIGN THEM A LABEL OCUR NUMBER. THIS NUMBER BEGINS WITH 1 AND INCREASES BY 1 FOR EACH NEW SECTION OR PARAGRAPH (0 IS RESERVED FOR ERROR REPORTING). THIS LABEL OCUR BECOMES THE INTERNAL NUMBER FOR THE NAME AFTER THE DICTIONARY PROCESSING PHASE.

2. ASSIGN A SEGMENT NUMBER TO EACH SECTION BASED ON SEGMENT-LIMIT AND PRIORITY NUMBER DECLARATIONS. THESE SEGMENT NUMBERS ARE SEQUENTIAL STARTING WITH 0.

3. RECOGNIZE GO TO PARAGRAPHS AND MARK THEM AS LEGAL OPERANDS OF AN ALTER STATEMENT.

4. IDENTIFY AND MARK OPERANDS OF A GO TO, ALTER, PERFORM, OR A PERFORM ... THRU ... STATEMENT.

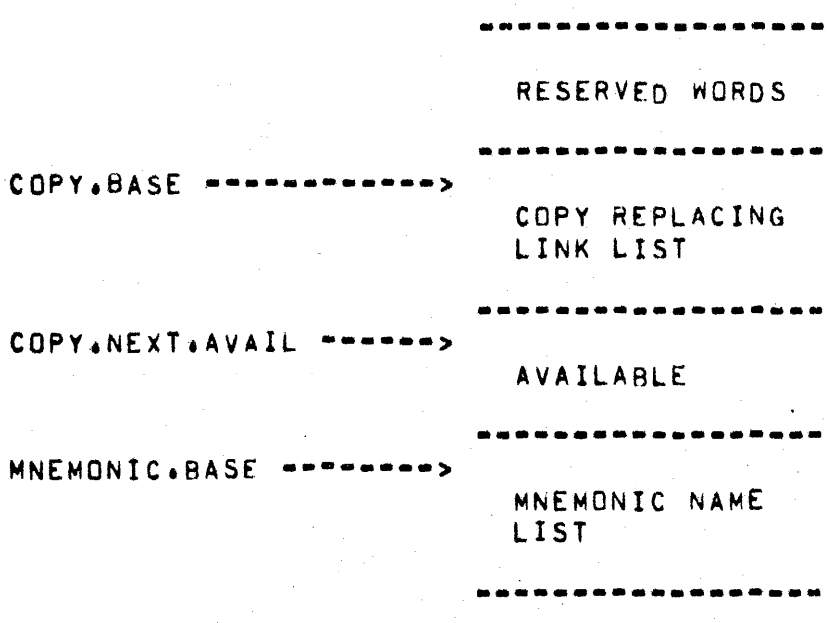
5. MARK OPERANDS OF OTHER SPECIAL I/O VERBS AS PROCEDURE NAMES.

6. RECOGNIZE AND MARK OPERANDS OF A CORRESPONDING VERB (E.G.

MOVE CORRESPONDING A TO B).

DYNAMIC MEMORY.

THE DYNAMIC MEMORY FOR PARSE IS UTILIZED AS FOLLOWS:



IF COPY.NEXT.AVAIL EVER MEETS MNEMONIC.BASE THEN AN ERROR MESSAGE IS ISSUED WITH A REQUEST TO RECOMPILE USING MORE MEMORY.

DICTIONARY PROCESSING (DICT).

INPUT FILES.

DNFILE: CONTAINS ALL PICTURE STRINGS, DECLARED VARIABLE NAMES AND REFERENCES TO THE VARIABLES.

OUTPUT FILES.

DNFILE: ALL PICTURE STRINGS AND VARIABLES ARE REDUCED TO AN OCUR.

GENERAL FUNCTIONS.

1. BUILD A DICTIONARY OF DECLARED DATA-NAMES AND PROCEDURE-NAMES.

2. LOOK UP REFERENCES TO THE DECLARED VARIABLES AND SUBSTITUTE AN OCUR NUMBER AND A SAME NAME OCUR NUMBER FOR THE SYMBOL STRING. THE SAME NAME OCUR NUMBER POINTS TO ANY PRIOR OCCURENCE OF THIS NAME IN A LINK LIST FASHION. IF THE SAME NAME OCUR = THE OCUR THEN THE LAST ELEMENT OF THE LIST HAS BEEN REACHED OR THE NAME IS UNIQUE.

3. ASSIGN A PICTURE OCUR NUMBER TO EACH NEW PICTURE STRING THAT WAS DECLARED. A PICTURE STRING IS DISTINGUISHED FROM OTHER VARIABLE NAMES BY ADDING A BLANK TO THE END OF THE STRING AND INCREASING THE SYMBOL LENGTH BY 1 IN THE SCANNING PROCESS(PARSE).

4. THESE FUNCTIONS ARE DONE ON AN ITERATIVE BASIS. WHEN THE FIRST ATTEMPT IS MADE TO ADD A NAME TO THE DICTIONARY AND IT WILL NOT FIT, A DICTIONARY.FULL.FLAG IS SET TO 1 AND THAT TOKEN IS MARKED AS UNPROCESSED ON THE OUTPUT FILE. WHEN A NAME IS LOOKED UP IN THE DICTIONARY AND IS FOUND, IT IS MARKED AS PROCESSED ON THE OUTPUT FILE AND FURTHER PROCESSING OF THAT TOKEN IS INHIBITED FOR THE DURATION OF THE ITERATIONS. IF THE NAME IS NOT FOUND, AND THE DICTIONARY.FULL.FLAG IS 1 THE TOKEN IS MARKED AS UNPROCESSED ON THE OUTPUT FILE, OTHERWISE AN OCUR NUMBER OF 0 IS RETURNED TO INDICATE "UNIDENTIFIED NAME" AND THE TOKEN IS MARKED AS PROCESSED. THIS CONTINUES UNTIL ALL VARIABLE NAMES HAVE BEEN ADDED TO THE DICTIONARY.

DICTIONARY SEARCH.

ASSOCIATED WITH EACH VARIABLE NAME IS A STACKHEAD NUMBER. THIS NUMBER IS CREATED IN PARSE BY APPLYING A TRANSFORMATION TO THE CHARACTER STRING COMPRISING THE NAME. IT IS USED AS AN INDEX INTO A STACKHEAD ARRAY TO FIND THE BEGINNING OF THE LINK LIST IN WHICH THIS NAME SHALL APPEAR. A LINK OF 0 INDICATES END OF THE LIST OR AN EMPTY LIST.

TWO STACKHEAD ARRAYS ARE MAINTAINED IN THE LOOKUP PROCESS, ONE FOR DATA-NAMES AND THE OTHER FOR PICTURE STRINGS AND LABELS (SECTION-NAMES AND PARAGRAPH-NAMES). THE PICTURE STRINGS ARE DISTINGUISHED FROM THE LABELS BECAUSE THEY HAVE A BLANK CHARACTER APPENDED TO THE END.

DYNAMIC MEMORY.

THE STACKHEAD ARRAYS AND LINK LISTS ARE MAINTAINED IN THE AVAILABLE DYNAMIC MEMORY. ONE OR MORE ITERATIONS THROUGH THE FILE MAY BE ELIMINATED BY INCREASING THE SIZE OF DYNAMIC FOR THIS PHASE.

EACH ITERATION CONSISTS OF TWO PASSES.

PASS I.

1. COPY ALL UNPROCESSED TOKENS OF THE ENVIRONMENT DIVISION TO THE OUTPUT FILE.

2. ADD ALL DECLARED DATA-NAMES AND LABELS TO THE DICTIONARY.

3. LOOK UP THE OPERANDS OF A REDEFINES OR RENAMES CLAUSE. THIS SOLVES THE PROBLEM OF IMPLIED QUALIFICATION FOR DUPLICATE DECLARATIONS IN THAT THE LAST MENTIONED OCCURENCE NUMBER WILL BE FOUND RATHER THAN THE FINAL OCCURENCE OF THAT NAME.

4. LOOK UP ALL DATA-NAMES OF THE PROCEDURE DIVISION. THE OCUR NUMBER FOUND WILL POINT TO THE FINAL OCCURENCE OF THE SAME NAME LINK LIST.

5. COPY UNPPOCESSED LABELS TO THE OUTPUT FILE.

PASS II.

1. LOOK UP UNPROCESSED NAMES OF THE ENVIRONMENT DIVISION.

2. COPY UNPROCESSED DATA-NAMES TO THE OUTPUT FILE.

3. LOOK UP LABEL REFERENCES OF THE PROCEDURE DIVISION. IF THE LABEL IS A DUPLICATE, THE OCUR NUMBER OF THE FINAL DECLARATION WILL BE FOUND.

4. IF THE DICTIONARY.FULL.FLAG = 1 THEN ITERATE TO PASS I.

DATA-NAME QUALIFICATION RESOLUTION (DNQUAL).

INPUT FILES.

DNFILE: CONTAINS A SAMENAME OCUR FOR DECLARED DATA-NAMES AND AN UNRESOLVED OCUR FOR DATA-NAME REFERENCES AND QUALIFIERS.

OUTPUT FILES.

DNFILE: CONTAINS RESOLVED OCUR FOR DATA-NAME REFERENCES.

GENERAL FUNCTIONS.

1. STARTING AT THE DATA DIVISION OF THE DNFILE (READ ONLY), BUILD AN EXPLICIT DATA-NAME TABLE CONTAINING THE DATA-NAME FLAGS (DN.FLAGS), THE SAME NAME OCUR (SNAME), AND THE SCOPE OCUR FOR EACH EXPLICIT DATA-NAME. THE GROUP FLAG IS SET AT THIS TIME.

2. RE-READ THE ENTIRE DNFILE AND WRITE THE RESOLVED OCUR FOR REFERENCED DATA-NAMES. THE QUALIFIER TOKENS ARE DROPPED AT THIS TIME.

3. SET THE LABEL RECORD OPERAND FLAG FOR ALL ELEMENTS OF A LABEL RECORD.

4. SELECT CORRESPONDING PAIRS. THE PAIRS ARE DELIMITED BY A CORRESPONDING SENTINEL SO THAT THE FILE CAN BE PROCESSED PROPERLY IN THE MERGE PHASE.

5. UPDATE THE DATA-NAME REFERENCE COUNT BASED ON THE RESOLVED OCUR.

6. SET THE MONITORED FLAG OF THE DATA-NAMES LISTED IN THE MONITOR STATEMENT.

7. READ THE DNFILE AND WRITE A NEW DNFILE WITH THE UPDATED DN.FLAGS POSTED TO THE EXPLICIT DATA-NAME TOKENS.

8. WHEN THE PROCEDURE DIVISION IS FOUND ON THE INPUT FILE, BEGIN THE LABEL QUALIFICATION RESOLUTION PHASE(LQUAL).

SCOPE OF AN ENTRY.

THE SCOPE OF EACH EXPLICIT DATA-NAME IS CALCULATED FROM ITS LEVEL NUMBER WHEN BUILDING THE DATA-NAME TABLE. THE SCOPE OF A GROUP ITEM IS THE LAST ELEMENTARY ITEM OF THE GROUP. THE SCOPE OF AN ELEMENTARY ITEM OR AN INDEX NAME OR A RENAMES ENTRY, OR A CONDITION NAME IS ITSELF.

EXAMPLE:

```
  OCUR   SNAME   SCOPE
  ----   -----   -----
  (01)     01      11     01 A.
  (02)     02      08       02 B.
  (03)     03      03         88 C VA 1, 5 THRU 10.
  (04)     04      04           03 D PC X.
  (05)     05      08           03 E.
  (06)     06      06             04 F PC X.
  (07)     07      07             04 G PC X.
  (08)     08      08               88 H VA 20, 30.
  (09)     09      09       02 I PC X.
  (10)     10      10     66 J RENAMES D THRU F.
  (11)     11      11     66 K RENAMES B.
  (12)     12      14     01 X.
  (13)     02      13       02 B PC X.
  (14)     04      14       02 D PC X.
```

THE SAME NAME AND SCOPE ATTRIBUTES ARE USED IN RESOLVING REFERENCES TO DUPLICATE NAMES AND IN THE SELECTION OF CORRESPONDING PAIRS FOR A CORRESPONDING VERB.

DYNAMIC MEMORY.

THE EXPLICIT DATA-NAME TABLE IS BUILT IN DYNAMIC MEMORY. EACH TABLE ENTRY OCCUPIES 34 BITS. IF THE ENTIRE TABLE CAN NOT BE CONTAINED, AN ERROR MESSAGE IS ISSUED WITH A REQUEST TO RECOMPILE USING MORE MEMORY.

LABEL QUALIFICATION RESOLUTION (LQUAL).

INPUT FILES.

DNFILE: CONTAINS UNRESOLVED LABEL REFERENCES.

OUTPUT FILES.

DNFILE: CONTAINS A UNIQUE LABEL OCUR FOR REFERENCED SECTIONS OR PARAGRAPHS.

GENERAL FUNCTIONS.

1. STARTING WITH THE PROCEDURE DIVISION OF THE DNFILE (THE FILE WAS LEFT OPEN IN DNQUAL), BUILD AN EXPLICIT LABEL TABLE CONTAINING THE LABEL FLAGS, SAME NAME OCUR, SEGMENT NUMBER, AND SECTION OCUR ASSOCIATED WITH THE PARAGRAPH IF ANY. THE FILE IS WRITTEN DURING THIS PROCESS.

2. READ THE FILE AND WRITE THE RESOLVED OCUR FOR REFERENCED LABELS. THE LABEL QUALIFIERS ARE DROPPED AT THIS TIME.

3. MARK THE MONITORED LABELS AND THE TERMINAL PARAGRAPH OF A PERFORM RANGE.

4. UPON ENCOUNTERING AN ALTER OPERAND, VERIFY THAT IT IS A GO TO PARAGRAPH AND MARK IT AS ALTERED.

5. WHEN THE END OF FILE IS REACHED, ASSIGN A SEQUENTIAL TERMINAL PARAGRAPH NUMBER TO EACH TERMINAL PARAGRAPH OF A PERFORM RANGE.

6. READ AND WRITE THE DNFILE WITH THE UPDATED LABEL INFORMATION POSTED TO EACH EXPLICIT REFERENCE OR LABEL REFERENCE TO IT.

DYNAMIC MEMORY.

THE EXPLICIT LABEL TABLE IS BUILT IN DYNAMIC MEMORY. EACH ENTRY OCCUPIES 37 BITS. IF THE ENTIRE TABLE CAN NOT BE CONTAINED, AN ERROR MESSAGE IS ISSUED WITH A REQUEST TO RECOMPILE USING MORE MEMORY.

MERGE.

INPUT FILES.

ALLFILE: CONTAINS CONSTANT INFORMATION ABOUT EACH TOKEN PROCESSED IN PARSE.

DNFILE: CONTAINS UNIQUE OCUR NUMBER FUR EACH DATA-NAME AND LABEL REFERENCED.

OUTPUT FILES.

ADNFILE: CONTAINS MERGED TOKENS OF THE ALLFILE AND THE DNFILE.

PCFILE: CONTAINS PCINFO ENTRY FOR EACH UNIQUE PICTURE STRING OF THE DATA DIVISION.

SEGFILE: EDIT MASKS GENERATED BY THE PICTURE ANALYZER, AND SYMBOLS TO BE PRINTED WHEN MONITORING ARE ADDED TO THIS FILE.

GENERAL FUNCTIONS.

THE PRIMARY FUNCTION OF THIS PHASE IS TO MERGE THE ALLFILE AND THE DNFILE GIVING THE ADNFILE. IN ADDITION THE FULLOWING FUNCTIONS ARE PERFORMED.

1. THE DATA-NAME SYMBOLS TO BE PRINTED WHEN MONITORING ARE WRITTEN TO THE SEGFILE, AND THE MONITORED OCUR VS THE MEMORY ADDRESS ARE PUT INTO A TABLE. A TABLE OF OCUR NUMBERS VS THE LABEL SYMBOLS TO BE PRINTED IS PLACED IN ANOTHER TABLE.

2. ANALYZE UNIQUE PICTURE STRINGS (THE PICTURE OCUR OF THE CURRENT PC IS GREATER THAN THE LAST ONE ANALYZED) AND CREATE A PCINFO ENTRY ON THE PCFILE. THIS ENTRY CONTAINS PC ATTRIBUTES LIKE SIZE, SCALE, CLASS, ETC. IF AN EDIT MASK IS REQUIRED IT IS WRITTEN TO THE SEGFILE AND ITS MEMORY ADDRESS IS POSTED TO THE PCINFO ENTRY.

3. A TABLE OF CONDITION-NAME OCUR NUMBERS VS THE ASSOCIATED VALUE LIST IS CREATED FOR EACH LEVEL 88 ENTRY OF THE DATA DIVISION.

4. WHEN A MONITORED DATA-NAME IS SEEN, ITS OCUR IS LOOKED UP IN THE MONITOR TABLE AND THE MONITOR ADDRESS IS POSTED TO THAT ENTRY.

5. WHEN A MONITOR LABEL LIST ELEMENT IS ENCOUNTERED, THE SYMBOL TO BE PRINTED IS PROVIDED AS A NON-NUMERIC LITERAL FOLLOWING THAT TOKEN.

6. WHEN A CONDITION-NAME IS REFERENCED, THE PARENTHESIZED TEXT THAT IS EQUIVALENT TO THE DESIRED TEST IS PROVIDED FROM THE CONDITION-NAME TABLE. NOTE: THE PARENTHESES ARE NECESSARY IN THE CASE WHERE THE CONDITION-NAME IS NEGATED.

DYNAMIC MEMORY.

THE MONITOR SYMBOL TABLES AND THE CONDITION-NAME TABLE ARE MAINTAINED IN DYNAMIC MEMORY. IF THEY CAN NOT BE ENTIRELY CONTAINED, AN ERROR MESSAGE IS ISSUED WITH A REQUEST TO RECOMPILE USING MORE MEMORY.

DATA DIVISION SYNTAX CHECK (DATSYN).

INPUT FILES.

PCFILE: CONTAINS A PCINFO ENTRY FOR EACH UNIQUE PC STRING THAT WAS DECLARED.

ADNFILE: PROCESSED AS A READ ONLY FILE UP TO THE PROCEDURE DIVISION.

OUTPUT FILES.

DNINFO: CONTAINS A DNINFO ENTRY FOR EACH EXPLICIT DATA-NAME EXCLUDING FILLER ENTRIES. THIS ENTRY CONTAINS ATTRIBUTES SUCH AS USAGE, ADDRESS, LENGTH, NUMBER OF SUBSCRIPTS REQUIRED, BLANK WHEN ZERO, ETC.

SEGFILE: CONTAINS DATA DIVISION TOKENS, E.G. CARD TOKEN WITH ASSOCIATED ADDRESS INFORMATION TO BE PRINTED, VALUES TO WHICH THE WORKING-STORAGE VARIABLES SHOULD BE INITIALIZED, ERROR OR WARNING MESSAGES, ETC.

GENERAL FUNCTIONS.

1. PERFORM A DETAILED SYNTAX CHECK OF THE SOURCE PROGRAM UP TO THE PROCEDURE DIVISION.

2. SAVE THE FILE ATTRIBUTES SPECIFIED IN THE FILE-CONTROL PARAGRAPH. THESE ATTRIBUTES ARE HELD IN THE FD.INFO TABLE.

3. SAVE THE "SAME RECORD AREA ..." ATTRIBUTE OF THE I-O-CONTROL PARAGRAPH IN THE FD.INFO TABLE. THE MULTI-FILE TAPE ID AND THE MULTI-FILE PACK ID ATTRIBUTES ARE ALSO SAVED IN THE FD.INFO TABLE.

4. COMBINE THE FD OR SD ATTRIBUTES DECLARED IN THE FILE SECTION WITH THOSE IN THE FD.INFO TABLE. THE SET OF COMBINED INFORMATION IS USED TO CREATE THE FILE PARAMETER BLOCK (FPB) FOR EACH FILE DECLARED.

5. PERFORM A DETAILED SYNTAX CHECK OF EACH DECLARED DATA-NAME AND ALLOCATE MEMORY FOR EACH ENTRY. A COP INDEX (COPX) IS ASSIGNED AT THIS TIME. THESE ATTRIBUTES, E.G. USAGE, BLANK

WHEN ZERO, NUMBER OF SUBSCRIPTS, ADDRESS, COPX, ETC., ARE SAVED IN THE DNINFO FILE AND CAN BE RETRIEVED BY USING THE DATA-NAME OCUR NUMBER AS THE KEY.

6. PARSE THE "VALUE IS ..." CLAUSE OF THE WORKING-STORAGE ENTRIES. A TOKEN FOR INITIALIZING MEMORY AT RUN TIME IS ISSUED TO THE SEGFILE.

7. CREATE THE PSEUDO DATA SEGMENT DICTIONARY FOR DATA SEGMENTATION.

8. WHEN THE PROCEDURE DIVISION IS FOUND, GATHER DATA SEGMENTS, FINALIZE SEGMENT-NUMBER AND DISPLACEMENT, BUILD THE COP TABLE, AND BEGIN THE EXPLODE PHASE.

DATA SEGMENTATION.

WHEN ALLOCATING STORAGE FOR DATA, A PSEUDO DATA DICTIONARY ENTRY IS BUILT FOR EACH DATA SEGMENT CANDIDATE, E.G. EACH FILE RECORD WORK AREA AND EACH WORKING-STORAGE RECORD THAT IS NOT REDEFINED. THE NON-CONTIGUOUS ITEMS (LEVEL = 77) ARE ASSIGNED TO DATA SEGMENT 0. SPECIAL CONSIDERATION IS GIVEN TO THESE ITEMS IN THAT SEGMENT 0 IS ALWAYS PRESENT AND A PRESENCE CHECK BY THE COBOL INTERPRETER IS NOT NECESSARY.

WHEN THE PROCEDURE DIVISION IS FOUND, AN ATTEMPT IS MADE TO GATHER DATA SEGMENTS ACCORDING TO THE DATA SEGMENT-LIMIT VALUE SPECIFIED IN THE OBJECT-COMPUTER PARAGRAPH.

IF DATA SEGMENT-LIMIT = 0 THEN ALL DATA IS ASSIGNED TO SEGMENT 0 AND NO FURTHER ACTION IS NECESSARY.

IF DATA SEGMENT-LIMIT IS NOT = 0 THEN EACH FILE RECORD WORK AREA IS ASSIGNED TO A NEW SEGMENT AND THE WORKING-STORAGE RECORDS ARE GATHERED AS FOLLOWS:

1. IF A CANDIDATE IS GREATER THAN DATA SEGMENT-LIMIT THEN ASSIGN IT TO A NEW SEGMENT.

2. IF A CANDIDATE WILL FIT WITH THE CANDIDATES GATHERED SO FAR (E.G. THE COMBINED SIZE DOES NOT EXCEED DATA SEGMENT-LIMIT) THEN ASSIGN IT TO THE CURRENT DATA SEGMENT.

3. IF A CANDIDATE WILL NOT FIT IN THE CURRENT SEGMENT THEN BEGIN GATHERING TO A NEW SEGMENT.

THIS METHOD TENDS TO GATHER SMALL CANDIDATES TO A DATA SEGMENT SIZE THAT IS CLOSE TO THE DESIRED DATA SEGMENT-LIMIT.

DYNAMIC MEMORY.

DYNAMIC MEMORY CONTAINS SEVERAL TABLES WHICH ARE MANAGED AS FOLLOWS:

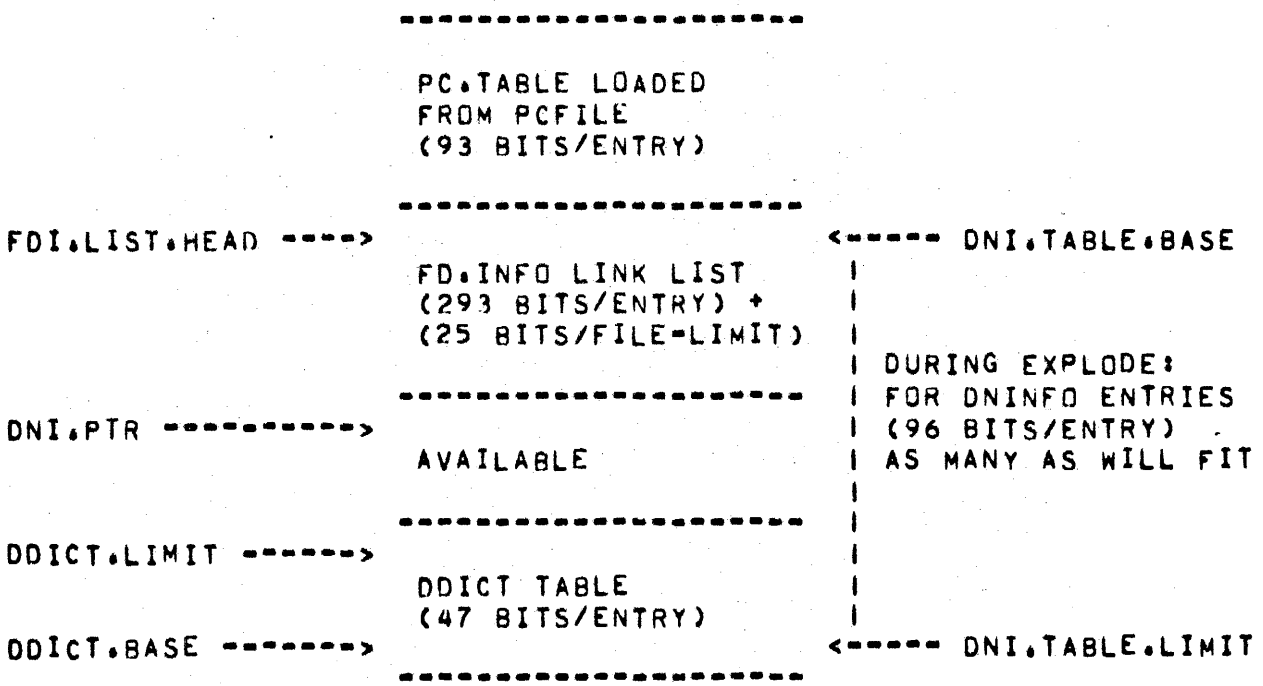
1. THE PC.TABLE IS LOADED FROM THE PCFILE INTO THE FIRST PART OF DYN.WA AT THE BEGINNING OF DATSYN.

2. FDI.LIST.HEAD CONTAINS A LINK TO THE FIRST FILE INFO ENTRY (FDI.INFO) LINK LIST. THERE IS ONE FD.INFO ENTRY PER SELECT(ED) FILE.

3. THE DATA DICTIONARY TABLE IS BUILT AT THE END OF DYN.WA AND GOES TOWARD THE FD.INFO TABLE. THERE IS ONE DDICT ENTRY FOR EACH DATA SEGMENT CANDIDATE. DDICT(0) IS RESERVED FOR THE LABEL RECORD WORK AREA OF THE USE PROCEDURES.

4. IF DYN.PTR EVER MEETS DDICT.LIMIT THEN AN ERROR MESSAGE IS ISSUED WITH A REQUEST TO RECOMPILE USING MORE MEMORY.

5. AT THE BEGINNING OF EXPLODE THE FD.INFO LINK LIST AND THE DATA DICTIONARY TABLE ARE REPLACED BY AS MANY DNINFO FILE ENTRIES AS WILL FIT. THE OVERFLOW ENTRIES ARE RETRIEVED ON A RANDOM BASIS FROM THE DNINFO FILE.



EXPLODE.

INPUT FILES.

ADNFILE: READY TO READ THE PROCEDURE DIVISION.

OUTPUT FILES.

ADNFILE: TOKEN REFERENCES ARE REPLACED BY A COPY OF THEIR ATTRIBUTES.

GENERAL FUNCTIONS.

1. READ THE DNINFO FILE AND LOAD DYNAMIC MEMORY WITH AS MANY ENTRIES AS IT WILL HOLD.

2. EXPAND TOKENS TO INCLUDE ALL THE KNOWN ATTRIBUTES FOR THAT TOKEN. E.G. A DATA-NAME OCUR IS USED TO INDEX THE DNINFO TABLE, TO WHICH THE PC ATTRIBUTES ARE ADDED FROM THE PC.TABLE. SUBSCRIPT FACTORS AND TABLE BOUND INFORMATION IS ISSUED AT THIS TIME.

3. EXPAND SPECIAL REGISTER REFERENCES TO LOOK LIKE DATA-NAME REFERENCES (TALLY, TODAYS-DATE, SW1, ... , SW8, ETC.).

4. OUTPUT TOKEN INFORMATION ABOUT DECLARED FILE-LIMITS.

5. PARSE THE USE SENTENCE OF A USE PROCEDURE.

6. PARSE SORT STATEMENTS WITH PARTICULAR ATTENTION TO THE "USING..." AND "GIVING..." CLAUSES.

PROCEDURE DIVISION SYNTAX CHECKING (PROSYN).

INPUT FILES.

ADNFILE: EXPLODED TOKENS INCLUDING A COPY OF THEIR ATTRIBUTES.

OUTPUT FILES.

ADNFILE: CONTAINS TOKENS THAT HAVE BEEN SYNTAX CHECKED, REARRANGED, AND SIMPLIFIED FOR CODE GENERATION.

GENERAL FUNCTIONS.

1. PERFORM A DETAILED SYNTAX CHECK OF EACH PROCEDURE DIVISION CONSTRUCT. ANY ERROR OR WARNING MESSAGES ISSUED ARE MERGED WITH PRIOR MESSAGES.

2. GET OPERANDS AND THEIR SUBSCRIPTS OR INDEXES AND AUTOMATICALLY STACK THEM FOR USE BY EACH CALLER. OPTIMIZE LITERAL SUBSCRIPTS AND INDEXES WHEN THE OPERAND IS PUT TO THE OUTPUT FILE (SEE "SUBSCRIPT AND INDEX OPTIMIZATION"). THE LOOK-AHEAD FEATURE OF THE GET PROCEDURE CAN BE INVOKED BY SETTING A FLAG. IN THIS MODE, TOKENS ARE PRESENTED TO THE CALLER ONE AT A TIME UNTIL THE FLAG IS RESET AND NORMAL OPERATION RESUMES WHERE IT LEFT OFF. THIS MODE IS ESPECIALLY USEFUL FOR OPTIMIZING, E.G. THE ROUNDED, SIZE ERROR, AND MULTIPLE RECEIVING FIELD REQUESTS ARE ENCODED AS VARIATIONS WITH THE ARITHMETIC VERBS.

3. CHANGE STATEMENTS TO A SIMPLER FORM, E.G. CORRESPONDING PAIRS OF A MOVE CORRESPONDING STATEMENT ARE PUT OUT AS SEPARATE MOVE STATEMENTS.

4. TRANSFORM ARITHMETIC EXPRESSIONS AND BOOLEAN EXPRESSIONS TO THEIR PARENTHESIS FREE POLISH EQUIVALENT BY APPLYING THE OPERATOR PRECIDENCE RULES. AN ATTEMPT IS MADE TO PRODUCE EQUIVALENT STRINGS FOR ARITHMETIC EXPRESSIONS AND THEIR ARITHMETIC STATEMENT COUNTERPARTS.

5. SUPPLY THE IMPLIED SUBJECT AND RELATIONAL OPERATOR TO ABREVIATED CONDITIONS.

DYNAMIC MEMORY.

THE OPERAND STACK IS MAINTAINED IN DYNAMIC MEMORY. IF IT CAN NOT BE ENTIRELY CONTAINED, AN ERROR MESSAGE IS ISSUED WITH A REQUEST TO RECOMPILE WITH MORE MEMORY.

CODE GENERATOR (CODEGEN).

INPUT FILES.

ADNFILE: CONTAINS THE PROCEDURE DIVISION TOKENS THAT HAVE BEEN SIMPLIFIED FOR CODE GENERATION.

OUTPUT FILES.

SEGFILE: CODEGEN TOKENS ARE APPENDED TO THE TOKENS PRODUCED BY THE PRIOR PHASES.

LABELTABLE: CONTAINS EXPLICIT LABEL ATTRIBUTES AND IMPLICIT LABEL ATTRIBUTES (E.G. BRANCH POINTS OF A CONDITION) USED FOR GENERATING THE CORRECT BRANCH ADDRESSES.

GENERAL FUNCTIONS.

1. CHECK THE DONT.GENERATE.CODE FLAG WHICH IS SET BY ANY PRIOR PHASE THAT DETECTED A SYNTAX ERROR. IF IT HAS BEEN SET, COPY THE TOKENS NEEDED TO PREPARE THE ERROR REPORT ONTO THE SEGFILE AND BEGIN THE FIXUP PHASE. IF NO ERRORS HAVE BEEN DETECTED, GENERATE THE REQUIRED CODE.

2. GENERATE CODE FOR THE EXPONENTIATE INTRINSIC. THIS CODE IS GENERATED ONLY WHEN NEEDED (FOR CERTAIN SIMPLE CASES THE CODE IS EMITTED IN-LINE).

3. EMIT CODE FOR FILE-LIMIT CHECKING.

4. EMIT CODE TO ANALYZE THE REASON FOR A USE PROCEDURE BEING INVOKED.

5. GENERATE CODE FOR THE MONITOR INTRINSIC.

6. UPDATE THE LABELTABLE FILE FOR EACH SECTION, PARAGRAPH, AND IMPLICIT BRANCH THAT IS EMITTED.

DYNAMIC MEMORY.

A GET.TOKEN PROCEDURE AUTOMATICALLY STACKS THE OPERANDS AND THEIR SUBSCRIPTS FOR USE BY EACH CALLER. THE OPERAND STACK IS MAINTAINED IN DYNAMIC MEMORY. IF IT CAN NOT BE ENTIRELY MAINTAINED, AN ERROR MESSAGE IS ISSUED WITH A REQUEST TO RECOMPILE WITH MORE MEMORY.

FIXUP

INPUT FILES.

REPORT:	CONTAINS CARD IMAGES NEEDED FOR LISTING. NOT OPENED IF NO LIST SPECIFIED AND THERE ARE NO ERRORS.
SEGFILE:	CONTAINS VARIOUS INFORMATION FROM PRECEDING PASSES WHICH IS USED TO BUILD THE CODEFILE AND SUPPLY ADDITION DATA FOR THE LISTING.
LABELTABLE:	USED TO FINALIZE BRANCH ADDRESSES. NOT PRESENT IF THERE ARE SYNTAX ERRORS.

OUTPUT FILES.

CODEFILE:	CONTAINS OBJECT PROGRAM ACCORDING TO MCP SPECIFICATIONS. NOT PRESENT IF THERE ARE SYNTAX ERRORS.
LINE:	USED FOR LISTING AND COMPILER DEBUGGING OUTPUT.

GENERAL FUNCTIONS.

FIXUP HAS 2 PRIMARY JOBS: PRODUCE A CODEFILE AND A LISTING. NO CODEFILE IS PRODUCED IF THERE ARE SYNTAX ERRORS. FIXUP IS SENSITIVE TO $ CARDS. A DEFAULT $ CARD IS THE FIRST TOKEN FIXUP SEES (AS PART OF THE DATSYN TOKENS IN THE SEGFILE) UNLESS THE USER HAS A $ CARD AT THE FRONT OF HIS DECK. THE DEFAULT $ CARD GIVES "LIST".

IF THE LIST OPTION IS CURRENTLY OFF AND THERE IS AN ERROR OR WARNING, THE APPROPRIATE SOURCE CARD IS PRINTED.

EVERYTHING FIXUP DOES IS VERY SPECIALIZED AND DETAILED. IT KNOWS HOW THE MCP EXPECTS A CODEFILE TO BE FORMATTED, HOW THE COBOL INTERPRETER WANTS MEMORY LAID OUT FOR A COBOL PROGRAM, AND OTHER THINGS TOO STRANGE AND WONDEROUS TO REVEAL TO MORTAL MAN.

HOUSEKEEPING.

THE COBOL INTERPRETER PERMITS 6 DIFFERENT CONTAINER SIZES TO BE VARIABLE IN LENGTH. CODEGEN DETERMINES 2 OF THESE: SEGB (SPECIFIES SIZE FOR THE DATA SEGMENT NUMBER PORTION OF DATA ADDRESSES) AND COPXB (SIZE FOR COPX). FIXUP DETERMINES 4 OF THESE:

1. BDISPB (SIZE FOR BRANCH ADDRESSES).

2. DISPB (SIZE FOR DATA ADDRESS DISPLACEMENT).

3. LENB (SIZE FOR DATA LENGTH).

4. COPB (SIZE FOR COP ENTRY...COPB:SEGB+DISPB+LENB+4).

FIXUP MUST ALSO BUILD THE ALTER TABLE. THE DIGIT SIZE OF AN ENTRY = (8+BDISPB+3)/4.

FILL DYNAMIC LABEL TABLE AREA.

AT THE BEGINNING OF FIXUP THE DYNAMIC AREA AVAILABLE (IF ANY) IS SEQUENTIALLY LOADED WITH AS MANY LABELTABLE ENTRIES AS WILL FIT, EACH ENTRY BEING RESOLVED TO ITS BADDR FORMAT (33 BITS) ON THE FLY. OVERFLOW ENTRIES ARE RETRIEVED RANDOMLY IN THEIR ORIGINAL FORM FROM THE LABELTABLE FILE AND RESOLVED BY EXCEPTION.

BUILD DATA DICTIONARY.

FIXUP USES THE DNDICTTABLE (BUILT BY DATSYN) IN THE SEGFILE TO BUILD THE CODEFILE DATA DICTIONARY. DATA SEGMENT 0 SIZE REFLECTS ONLY THE USERS DATA. SINCE THE MCP (MALE CHAUVINIST PIG) EXPECTS DATA SEGMENT 0 TO BE A PICTURE OF WHAT WILL BE IN MEMORY, FIXUP MUST UPDATE DSEG0 TO INCLUDE:

1. EDIT TABLE (8 CHARACTERS).

2. COP TABLE.

3. SPECIAL REGISTERS (SW1..SW8, TALLY, DATE, TIME, TODAYS-DATE, TODAYS-TIME, TODAYS-NAME).

4. CONSTANT POOL (DATANAME MONITOR SYMBOLS, EDIT MASKS, TRANSLATION TABLES, FILE LIMITS).

5. TRASH AREA (INTERMEDIATE RESULTS).

6. ALTER TABLE (IF ANY).

7. STACK.

DATA SEGMENT PORTIONS OF THE CODEFILE ARE SET TO ALL 0 BITS. THIS IS DONE BECAUSE UPON THE FIRST ACCESS OF A DATA SEGMENT, THE MCP READS

IT OFF DISK IF IT HAS BEEN ALLOCATED SPACE IN THE CODEFILE.

BUILD TRANSLATION TABLES.

EITHER ASCII-TO-EBCDIC AND/OR EBCDIC-TO-ASCII TABLE IS CREATED ONLY IF APPROPRIATE TRANSLATION INSTRUCTION IS GENERATED. THE TABLE(S) IS LOCATED AS THE LAST PART OF THE CONSTANT POOL.

PROCESS MERGE TOKENS FROM SEGFILE.

THE ONLY VALID TOKEN IS TMEMVALUE. THIS IS FOR INITIALIZING THE CONSTANT POOL PORTION OF THE CODEFILE TO MONITOR DATA-NAME SYMBOLS AND EDIT MASKS.

BUILD CODE DICTIONARY.

THE PSEUDO CODE DICTIONARY (PCD) PRODUCED BY CODEGEN CONTAINS A SUMMARY OF STATIC CODE AND NUMBER OF VARIABLE LENGTH CONTAINERS FOR EACH LOGICAL PROGRAM SEGMENT. BY NOW FIXUP HAS CALCULATED THE CONTAINER SIZES AND MULTIPLIES THESE BY THE PCD NUMBERS AND ADDS THE PCD STATIC CORE TO BUILD THE FINAL CODE DICTIONARY IN THE CODEFILE.

IN ADDITION "MARKER" RECORDS ARE WRITTEN ON THE SEGFILE. THESE TELL THE BEGINNING CODEFILE DISK SEGMENT ADDRESS FOR EACH LOGICAL PROGRAM SEGMENT.

PROCESS DATSYN TOKENS.

THE REPORT FILE IS USED TO PRODUCE A LISTING (IF APPLICABLE) AND TCARDADR TOKENS ARE FETCHED TO APPLY DATA ADDRESSES TO THE LISTING. TMEMVALUE TOKENS ARE PROCESSED TO IMPLEMENT THE "VALUE IS" CLAUSE.

PROCESS CODEGEN TOKENS.

IN ORDER TO ALIGN PRINTED CODE AND CODE ADDRESSES ON THE LISTING, CODEGEN TOKENS ARE PROCESSED SERIALLY. THE USER MAY HAVE CODE SEGMENTATION MIXED SO THAT PIECES OF A LOGICAL PROGRAM SEGMENT ARE SCATTERED THROUGHOUT THE PROGRAM.

FIXUP EMITS CODE FOR THE CURRENT PROGRAM SEGMENT. IF A NEW SEGMENT OCCURS, FIXUP USES THE "MARKER" RECORDS TO REMEMBER WHERE THE CODE FOR A SEGMENT LEFT OFF ON THE CODEFILE.

THE LABELTABLE FILE IS ACCESSED WHENEVER A TBADDR TOKEN IS ENCOUNTERED. THE ROUTINE LABEL.FIXUP TAKES A LABELTABLE OCUR FROM THE TOKEN AND USES IT AS AN INDEX INTO THE LABELTABLE TO FINALIZE A BRANCH ADDRESS.

FILE LIMIT VALUES ARE SET UP BY CODEGEN AS TMEMVALUE TOKENS AND ARE INITIALIZED ON THE CODEFILE AT THIS POINT.

CLEANUP.

1. PRINT THE CODE DICTIONARY.
2. BUILD THE FILE PARAMETER BLOCKS.
3. PRINT THE DATA DICTIONARY.
4. BUILD AND PRINT THE PATH DICTIONARY (DATA MANAGEMENT).
5. BUILD AND PRINT THE COP TABLE.
6. BUILD AND PRINT THE RUN STRUCTURE.
7. FINALIZE THE PROGRAM PARAMETER BLOCK.
8. BUILD AND PRINT THE ALTER TABLE.
9. PRINT THE PROGRAM PARAMETER BLOCK.
10. PRINT THE SUMMARY INFORMATION.

GLOSSARY OF ALLFILE AND DNFILE TOKENS

00=TINTGR	INTEGER EG: 123
00=TEDC	ENV.DIV.BLOCK CODE (DNFILE ONLY)
01=TREAL	REAL NUMBER EG: 12.3
01=TDDC	DATA DIVISION BLOCK CODE (DNFILE ONLY)
02=TNNLIT	NON-NUMERIC LIT EG: "ABC"
02=TPDC	PROC.DIV.BLOCK CODE (DNFILE ONLY)
03=TXDN	EXPLICIT DATA NAME EG: 01 DATA-NAME.
03=TFILEREC	EG: READ FILEREC
04=TXSECN	EXPLICIT SECTION NAME EG: XSECN SECTION.
05=TXPAR	EXPLICIT PARAGRAPH NAME EG: XPAR.
06=TFIXUP	FIXUP THE INDICATED OCUR ACCORDING TO THE CODE.
07=TWORD	WORD EG: MOVE 1 TO WORD.
08=TWMON	MONITOR WORD OPERAND
09=TCSPL	CORRESPONDING LEFT EG: MOVE CORR TCSPL TO TCSPR
10=TCSPR	CORRESPONDING RIGHT EG: MOVE CORR TCSPL TO TCSPR
11=TQUAL	WORD QUALIFIER EG: MOVE 1 TO WORD OF QUAL
12=TLABL	LABEL EG: GO TO LABL.
13=TALTER	ALTER(ED) OPND EG: ALTER TALTER TO PROCEED TO P2.
14=TPERF	PERFORM OPERAND EG: PERFORM PERF.
15=TPTHRU	PERFORM THRU OPERAND EG: PERFORM P1 THRU PTHRU.
16=TLMON	MONITOR LABEL OPERAND
17=TLQUAL	LABEL QUALIFIER EG: GO TO P1 OF LQUAL
18=TRESWD	RESERVED WORD EG: GO, MOVE, ADD.
19=TEOB	END-OF-BLOCK MARKER-FOR UNBLOCKING
20=TEOF	END-OF-FILE MARKER-FOR UNBLOCKING
21=TDOT	PERIOD
22=TPC	PICTURE
23=TDOLAR	DOLLAR CARD
24=TERROR	ERROR OR WARNING MESSAGE
25=TCARD	CARD MARKER I.E. SOURCE CARD ENCOUNTERED
26=TPROCESSED	PROCESSED DICTIONARY ENTRIES (DNFILE ONLY)
27=TRDFNS	REDEFINES/RENAMES OPERAND EG: 02 X REDEFINES RDFNS
28=TLABELREC	OPERAND OF "LABEL RECORDS ARE" CLAUSE
29=TCSPS	CORRESPONDING SENTINEL-DELIMITS A CSPL,CSPR PAIR
30=TSTATESTOP	STATEMENT DELIMITER
31=TINDEX	"INDEXED BY" OPERAND EG: 02 DN INDEXED BY TINDEX
32=TSUBS	SUBSCRIPT FOR PREVIOUS OPERAND..CREATED IN EXPLODE
33=TBOOSTOP	BOOLEAN PRIMARY DELIMITER..CREATED IN REARRANGE
34=TARITHOP	ARITHMETIC OPERATOR EG: ADD0, DIV3
35=TINDSECT	FOR INITIALIZING ALTERED GO TO PARS OF IND.SECTIONS
36=TMINIMUM	ESTABLISHES MINIMUM SCALE AND LEFT PART FOR INTERMEDIATE RESULTS OF AN ARITHMETIC EXPRESSION

GLOSSARY OF SEGFILE TOKENS

00=TCOPX	COP TABLE INDEX...IF 0 THEN INLINE TCOP FOLLOWS
01=TLIT	LITERAL OF 4 OR 8 BIT UNITS
02=TCOP	INLINE COP
03=TDADDR	ADDRESS EVENTUALLY RELATIVE TO DSEG0
04=TOP	S-LANGUAGE OPERATOR
05=TBADDR	LABEL.TABLE OCUR CHANGED INTO BRANCH ADR BY FIXUP
06=TBITS	BIT STRING TO INSERT INTO CODE...GENERAL USE
07=TSUBORINX	INLINE COP FOR SUBSCRIPTS OR INDEXES
08=TSEGLINK	LINKS SECTIONS BELONGING TO THE SAME CODE SEGMENT
09=TFACTOR	FACTOR FOR SUBSCRIPTED INLINE COPS
10=TBOUND	BOUND FOR SUBSCRIPTED INLINE COPS
11=TALTERINDEX	POINTER RELATIVE TO ALTER TABLE BASE
12=TOPNDX	SETS LIT FLAG TO 0...COP TABLE INDEX
13=TLITFIX	MAKE A LIT RELATIVE TO A CERTAIN BASE(FOR I/O USE)
14=TSTOP	SIGNALS CHANGE IN CODE SEGMENTS
15=	UNUSED
16=	UNUSED
17=TMEMVALUE	FOR FIXUP...INITIALIZES DATA SEGMENT VALUE IN CODEF
18=TCARDADR	DATA ADR TO PRINTOUT WITH CARD (CREATED IN DATSYN)
19=TEOB	END-OF-BLOCK MARKER-FOR UNBLOCKING
20=TEOF	END-OF-FILE MARKER-FOR UNBLOCKING
23=TDOLLAR	DOLLAR CARD
24=TERROR	ERROR OR WARNING MESSAGE
25=TCARD	CARD MARKER I.E. SOURCE CARD ENCOUNTERED
26=TCOMMLITFIX	MAKE A LIT RELATIVE TO A CERTAIN BASE(FOR I/O USE)

MISCELLANEOUS

NOTE:
URS = FIELD FOR ANY USE YOU WISH.
** = TYPE NOT PRESENT IN THIS PASS.
-- = TYPE PERMANENTLY DROPPED IN PREVIOUS PASS.
USAGE

000=RESERVED
001=CMP
010=DISPLAY
011=MIXED
100=ASCII
101=INDEX
110=UNDEFINED
111=CMP-3

DATA NAME FLAGS (DNFLAGS) (10)

REFERENCE COUNT	LABEL RECORD OPERAND	SORT FILE NAME	GROUP FLAG	NOT A CORR OPND	COND NAME	FILLER	MONITORED	FILE NAME
(2)	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)

REFERENCE COUNT(TO MINIMIZE NO. OF COP ENTRIES):
0:NO REFERENCES
1:ONE REFERENCE
2:TWO OR MORE REFERENCES

LABEL FLAGS (6)

THIS SECT CONTAINS ALTERED GO TO PAR	MONITORED	OBJECT OF AT LEAST ONE ALTER	PERFORMED TERMINAL PARAGRAPH	EXPLICIT GO TO PAR	49FLAG
(1)*	(1)	(1)	(1)	(1)	(1)*

* ON IN SECTION & ALL ITS PARAGRAPHS
49FLAG:1=PRIORITY GTR 49
EXPLICIT GO TO PAR FLAG IS SET IN QUALIFICATION RESOLUTION (III)

URS IN LITERAL TOKENS (4)

FORMAT	ALL	SIGNED	BINARY
(1)	(1)	(1)	(1)

FORMAT:0=8BIT, 1=4BIT BINARY:1=LIT IS 24BIT BINARY

BASIC COP (53)

ADDRESS (31)	LENGTH (LOGICAL SIZE) (18)	SUB FLAG (1)	DATA TYPE (2)	ASCII FLAG (1)

DATA TYPE

00=UNSIGNED 4 BIT
01=UNSIGNED 8 BIT
10=SIGNED 4 BIT
11=SIGNED 8 BIT

COP FACTORS (80)

#SUBS (2)	FACTOR1 (18)	FACTOR2 (18)	FACTOR3 (18)	SUBSCRIPT BOUNDS (24)

FPB INFO (80)

UNUSED (37)	FILE LABEL TYPE (4)	FILE HARDWARE (7)	FILE ACCESS (4)	FILE LIMITS FLAG (1)	FPB NUMBER (9)	FILE MAX REC LEN (18)

PCINFO (74)

MASK ADR (24)	MAPSZ (CORE SZ) (18)	SCALE (F)	SIGN (3)	CLASS (3)	BZ (1)	CP (1)

SIGN	CLASS
0=U	0=NUMERIC INTEGER
1=S	1=NUMERIC REAL
2=J	2=NE
3=K	3=AN
4=E	4=AB
	5=AE

MON INFO (30)

MONITOR SYMBOL ADR. (24)	MON SYM LGTH (6)

ALLFILE: OUTPUT FROM INITIAL PARSING (I)

```
00=TINTGR                (29+(LENGTH*8))
 ------------------------------------------------------
  TYPE   COLUMN   SCALE   BYTE LENGTH   EBCDIC LITERAL
  (6)    (7)      (8)     (8)           (255 MAX)
 ------------------------------------------------------
01=TREAL                 SEE 00=TINTGR
02=TNNLIT                SEE 00=TINTGR
03=TXDN                  (13)
 --------------
  TYPE   COLUMN
  (6)    (7)
 --------------
04=TXSECN                **  % SECTIONS ARE MARKED AS SUCH ON DNFILE BUT
                             % AS TXPAR ON ALLFILE
05=TXPAR                 SEE 03=TXDN
06=TFIXUP                **
07=TWORD                 SEE 03=TXDN
08=TWMON                 SEE 03=TXDN
09=TCSPL                 SEE 03=TXDN
10=TCSPR                 SEE 03=TXDN
11=TQUAL                 **
12=TLABL                 SEE 03=TXDN
13=TALTER                SEE 03=TXDN
14=TPERF                 SEE 03=TXDN
15=TPTHRU                SEE 03=TXDN
16=TLMON                 SEE 03=TXDN
17=TLQUAL                **
18=TRESWD                (29)
 ----------------------------
  TYPE   COLUMN   KEY (IN HEX)
  (6)    (7)      (16)
 ----------------------------
19=TEOB                  (6)
 ------
  TYPE
  (6)
 ------
20=TEOF                  SEE 19=TEOB
21=TDOT                  SEE 03=TXDN
22=TPC                   SEE 03=TXDN
23=TDOLAR                (36)
 -------------
  TYPE   FLAGS
  (6)    (30)
 -------------
```

```
24=TERROR                (24)
 ----------------------------
  TYPE   COLUMN   WARN   ERROR#
  (6)    (7)      (1)    (10)
 ----------------------------
25=TCARD                 SEE 19=TEOB
26=TPROCESSED            **
27=TRDFNS                SEE 03=TXDN
28=TLABELREC             SEE 03=TXDN
29=TCSPS                 **
30=TSTATESTOP            **
31=TINDEX                **
32=TSUBS                 **
33=TBOOSTOP              **
34=TARITHOP              **
35=TINDSECT              **
```

DNFILE: OUTPUT FROM INITIAL PARSING (II)

```
00=TEDC                 (6)
 ------
  TYPE
  (6)
 ------
01=TDDC                 SEE 00=TEDC
02=TPDC                 SEE 00=TEDC
03=TXDN                 (56+(LENGTH*8))
 ------------------------------------------------------------------
  TYPE  LEVEL  DNFLAGS  DNOCUR  STACKHEAD#  SYMBOL LENGTH  SYMBOLS
  (6)   (16)   (10)     (12)    (7)          (5)            (30 MAX)
 ------------------------------------------------------------------
  NOTE : LEVEL=2 CHARACTERS

04=TXSECN               (50+(LENGTH*8))
 ------------------------------------------------------------------
  TYPE  49     SEG#  CURRENT  LABEL  STACKHEAD#  SYMBOL LENGTH  SYMBOLS
        FLAG       SECTION  OCUR
                   OCUR
  (6)   (1)    (7)   (12)     (12)   (7)          (5)            (30 MAX)
 ------------------------------------------------------------------
  SEG# IS SET BY THIS PASS..SYMBOL SCRAMBLE DONE BY THIS PASS

05=TXPAR                SEE 04=TXSECN
06=TFIXUP               (23)
 ------------------------
  TYPE  OCUR  FIXUP CODE
  (6)   (12)  (5)
 ------------------------
  MARK SUCH THINGS AS:
    A. NOT CORRESPONDING OPERAND.
    B. GO TO PAR.

07=TWORD                (18+(LENGTH*8))
 ------------------------------------------
  TYPE  STACKHEAD#  SYMBOL LENGTH  SYMBOLS
  (6)   (7)         (5)            (30 MAX)
 ------------------------------------------
08=TWMON                SEE 07=TWORD
09=TCSPL                SEE 07=TWORD
10=TCSPR                SEE 07=TWORD
11=TQUAL                SEE 07=TWORD
12=TLABL                SEE 07=TWORD
13=TALTER               SEE 07=TWORD
14=TPERF                SEE 07=TWORD
15=TPTHRU               SEE 07=TWORD
16=TLMON                SEE 07=TWORD
17=TLQUAL               SEE 07=TWORD
18=TRESWD               **
```

```
19=TEOB                 (6)
 ------
  TYPE
  (6)
 ------
20=TEOF                 SEE 19=TEOB
21=TDOT                 **
22=TPC                  (30+(LENGTH*8))
 ------------------------------------------------------
  TYPE   PC OCUR   STACKHEAD#   SYMBOL LENGTH   SYMBOLS
  (6)    (12)      (7)          (5)             (31 MAX)
 ------------------------------------------------------
```

A BLANK IS INCLUDED AT THE END OF THE PICTURE TO PROVIDE A CONVENIENT DELIMITER FOR PICTURE PROCESSOR AND TO KEEP PICTURES UNIQUE FROM LABEL NAMES IN DICTIONARY PROCESSING PASS.

```
23=TDOLAR               (36)
 -------------
  TYPE   FLAGS
  (6)    (30)
 -------------
24=TERROR               **
25=TCARD                **
26=TPROCESSED           **
27=TRDFNS               SEE 07=TWORD
28=TLABELREC            SEE 07=TWORD
29=TCSPS                SEE 00=TEDC
30=TSTATESTOP           **
31=TINDEX               **
32=TSUBS                **
33=TBOOSTOP             **
34=TARITHOP             **
35=TINDSECT             **
```

DNFILE: OUTPUT FROM DICTIONARY PROCESSING (II)

```
00=TEDC                 (6)
 ------
  TYPE
  (6)
 ------
01=TDDC                 SEE 00=TEDC
02=TPDC                 SEE 00=TEDC
03=TXDN                 (44)
 ------------------------------------
  TYPE  LEVEL  DNFLAGS  SAMENAME OCUR
  (6)   (16)   (10)     (12)
 ------------------------------------
  FILLER: FILLER FLAG=1
  WORKING-STORAGE SECTION: ENTRY=0

04=TXSECN               (38)
 ------------------------------------------------
  TYPE  49FLAG  SEG#  SECTION OCUR  SAMENAME OCUR
  (6)   (1)     (7)   (12)          (12)
 ------------------------------------------------
05=TXPAR                SEE 04=TXSECN
06=TFIXUP               (23)
 -------------------------
  TYPE  OCUR  FIXUP CODE
  (6)   (12)  (5)
 -------------------------
07=TWORD                (18)
 -------------------------
  TYPE  UNRESOLVED OCUR
  (6)   (12)
 -------------------------
08=TWMON                SEE 07=TWORD
09=TCSPL                SEE 07=TWORD
10=TCSPR                SEE 07=TWORD
11=TQUAL                SEE 07=TWORD
12=TLABL                SEE 07=TWORD
13=TALTER               SEE 07=TWORD
14=TPERF                SEE 07=TWORD
15=TPTHRU               SEE 07=TWORD
16=TLMON                SEE 07=TWORD
17=TLQUAL               SEE 07=TWORD
18=TRESWD               **
19=TEOB                 SEE 00=TEDC
20=TEOF                 SEE 00=TEDC
21=TDOT                 **
```

```
22=TPC                  (23+(LENGTH*8))
 -------------------------------
  TYPE  OCUR  LENGTH  SYMBOLS
  (6)   (12)  (5)     (31 MAX)
 -------------------------------
  PICTURES HAVE BEEN LOOKED UP AND NOW OCUR = UNIQUE PICTURE. SYMBOLS
  ARE RETAINED FOR CONVENIENCE FOR PICTURE PROCESSING PASS.

23=TDOLAR               (36)
 -------------
  TYPE  FLAG$
  (6)   (30)
 -------------
24=TERROR               **
25=TCARD                **
26=TPROCESSED           (26)
 -------------
  TYPE  #BITS
  (6)   (20)
 -------------
  AT  BEGINNING OF EACH BUFFER = USED TO INDICATE HOW MUCH DATA WAS
  PREVIOUSLY PROCESSED IN THIS ITERATIVE PASS (WHICH ALLOWS THE
  DICTIONARY TO OVERFLOW) - THIS TYPE IS DROPPED IN THE NEXT PASS.

27=TRDFNS               SEE 07=TWORD
28=TLABELREC            SEE 07=TWORD
29=TCSPS                SEE 00=TEOC
30=TSTATESTOP           **
31=TINDEX               **
32=TSUBS                **
33=TBOOSTOP             **
34=TARITHOP             **
35=TINDSECT             **
```

DNFILE: OUTPUT FROM QUALIFICATION RESOLUTION (III)

```
00=TEDC                 DROPPED
01=TDDC                 (6)
 ------
  TYPE
  (6)
 ------
02=TPDC                 DROPPED
03=TXDN                 (56)
 ------------------------------------------
  TYPE  LEVEL  DNFLAGS  SAMENAME OCUR  SCOPE
  (6)   (16)   (10)     (12)           (12)
 ------------------------------------------
  DNFLAGS REF.CNT. HAS BEEN UPDATED
04=TXSECN               (31)
 ---------------------------------------
  TYPE  LABEL FLAGS  SEG#  SAMENAME OCUR
  (6)   (6)          (7)   (12)
 ---------------------------------------
05=TXPAR                SEE 04=TSECN
06=TFIXUP               DROPPED
07=TWORD                (28)
 ------------------------------
  TYPE  RESOLVED OCUR  DNFLAGS
  (6)   (12)           (10)
 ------------------------------
  DNFLAGS NEEDED FOR THINGS LIKE SPECIAL PROCESSING OF CONDITION NAMES

08=TWMON                SEE 07=TWORD
09=TCSPL                SEE 07=TWORD
10=TCSPR                SEE 07=TWORD
11=TQUAL                DROPPED
12=TLABL                (43)
 ------------------------------------------------
  TYPE  LABEL FLAGS  SEG#  SECTION OCUR  RESOLVED OCUR
  (6)   (6)          (7)   (12)          (12)
 ------------------------------------------------
13=TALTER               (51)
 ------------------------------------------------------------
  TYPE  LABEL FLAGS  SEG#  SECTION OCUR  RESOLVED OCUR  OBJECT TIME
                                                       ALTER INDEX
  (6)   (6)          (7)   (12)          (12)          (8)*
 ------------------------------------------------------------
  * FOR QUAL.RES. USE ONLY
```

```
14=TPERF                (51)
 ------------------------------------------------------------------------
  TYPE  LABEL FLAGS  SEG#  SECTION OCUR  RESOLVED OCUR  TERM PAR #
  (6)   (6)          (7)   (12)          (12)           (8)
 ------------------------------------------------------------------------
  TERM PAR# CANNOT BE THE OCUR BECAUSE OF 8 BIT LIMIT
15=TPTHRU               SEE 14=TPERF
16=TLMON                SEE 12=TLABL
17=TLQUAL               DROPPED
18=TRESWD               **
19=TEOB                 SEE 01=TDDC
20=TEOF                 SEE 01=TDDC
21=TDOT                 **
22=TPC                  (23+(LENGTH*8))
 ---------------------------
  TYPE  OCUR  LENGTH | SYMBOLS
  (6)   (12)  (5)    | (31 MAX)
 ---------------------------
23=TDOLAR               DROPPED
24=TERROR               (17)
 -------------------
  TYPE  WARN  ERROR#
  (6)   (1)   (10)
 -------------------
25=TCARD                **
26=TPROCESSED           DROPPED
27=TRDFNS               (52)
 ------------------------------------------------------
  TYPE  RESOLVED OCUR  DNFLAGS  SAMENAME OCUR  SCOPE
  (6)   (12)           (10)     (12)           (12)
 ------------------------------------------------------
28=TLABELREC            SEE 07=TWORD
29=TCSPS                SEE 01=TDDC
30=TSTATESTOP           **
31=TINDEX               **
32=TSUBS                **
33=TBOOSTOP             **
34=TARITHOP             **
35=TINDSECT             **
```

PCINFO: OUTPUT FROM MERGE (IV)

(91)

MASK ADDR (24)	MAPSZ (CORE SZ) (18)	LENGTH (LOGICAL SZ) (18)	SCALE (F)	SIGN (1)	CLASS (3)	BZ (1)	CP (1)	ERROR FLAG (1)

MAPSZ=CORE SIZE..LOGICAL SIZE= # UNITS TO ACCEPT FROM SOURCE..
IF ERRORFLAG=1 THEN MASK ADR. = ERROR# AND SCALE = CARD COLUMN..
MASK IS IN CONSTANT POOL..LENGTH LATER BECOMES PART OF BASIC COP..
OTHER INFO LATER BECOMES PCINFO PORTION OF ELEMENTARY TOKEN.

SIGN

0=US UNSIGNED (4 OR 8 BIT)
1=SS HIGH ORDER DIGIT (4 OR 8 BIT)
2=JS HIGH ORDER DIGIT OF LAST UNIT (4 OR 8 BIT)
3=KS HIGH ORDER CHAR (8 BIT)
4=ES +,-,CR,DB (8 BIT)

CLASS

0=INTEGER	(4 OR 8 BIT)	PC 999
1=REAL	(4 OR 8 BIT)	PC 999V99 (NUMERIC SCALED)
2=NE	(8 BIT)	PC $$,$$$ OR 999.99+
3=AN	(8 BIT)	PC XX9A
4=AB	(8 BIT)	PC AAA
5=AE	(8 BIT)	PC XXBAA

ALLDNFILE: OUTPUT FROM MERGE (IV)

```
00=TINTGR              (33+(LENGTH*8))
 ------------------------------------------------------------
  TYPE  COLUMN  URS  SCALE  BYTE LENGTH  EBCDIC LITERAL
  (6)   (7)     (4)  (8)    (8)          (255 MAX)
 ------------------------------------------------------------
01=TREAL               SEE 00=TINTGR
02=TNNLIT              SEE 00=TINTGR
03=TXDN                (63)
 ------------------------------------------------------------
  TYPE  COLUMN  LEVEL  DNFLAGS  SAMENAME OCUR  SCOPE
  (6)   (7)     (16)   (10)     (12)           (12)
 ------------------------------------------------------------
04=TXSECN              (42)
 ------------------------------------------------------------
  TYPE  COLUMN  URS  LABEL FLAGS  SEG#  SAMENAME OCUR
  (6)   (7)     (4)  (6)          (7)   (12)
 ------------------------------------------------------------
  IF MONITOR FLAG=1 THEN FOLLOWS A NON-NUMERIC LITERAL TOKEN.

05=TXPAR               SEE 04=TXSECN
06=TFIXUP              --
07=TWORD               (69)
 ------------------------------------------------------------------------
  TYPE  COLUMN  URS  DNOCUR  DNFLAGS  MONITOR SYMBOL ADR.  MON SYM LGTH
  (6)   (7)     (4)  (12)    (10)     (24)                 (6)
 ------------------------------------------------------------------------
08=TWMON               DROPPED
09=TCSPL               SEE 07=TWORD
10=TCSPR               SEE 07=TWORD
11=TQUAL               --
12=TLABL               (54)
 ------------------------------------------------------------
  TYPE  COLUMN  URS  LABEL FLAGS  SEG#  SECT OCUR  LABEL OCUR
  (6)   (7)     (4)  (6)          (7)   (12)       (12)
 ------------------------------------------------------------
  URS:
   ---------------------------------
    DUMMY TLABL CREATED FOR  UNUSED
    A. GO TO.
    (1)                      (3)
   ---------------------------------
13=TALTER              (62)
 ------------------------------------------------------------------------
  TYPE  COLUMN  URS  LABEL FLAGS  SEG#  SECT OCUR  LABEL OCUR  ALTER IN
  (6)   (7)     (4)  (6)          (7)   (12)       (12)        (8)
 ------------------------------------------------------------------------
```

```
14=TPERF                (62)
 ------------------------------------------------------------------------
  TYPE  COLUMN  URS  LABEL FLAGS  SEG#  SECT OCUR  LABEL OCUR  TERM PAR#
  (6)   (7)     (4)  (6)          (7)   (12)       (12)        (8)
 ------------------------------------------------------------------------
15=TPTHRU               SEE 14=TPERF
16=TLMON                DROPPED
17=TLQUAL               --
18=TRESWD               (29)
 ---------------------------
  TYPE  COLUMN  KEY (IN HEX)
  (6)   (7)     (16)
 ---------------------------
19=TEOB                 (6)
 ------
  TYPE
  (6)
 ------
20=TEOF                 SEE 19=TEOB
21=TDOT                 (13)
 -------------
  TYPE  COLUMN
  (6)   (7)
 -------------
22=TPC                  (25)
 ----------------------
  TYPE  COLUMN  PC OCUR
  (6)   (7)     (12)
 ----------------------
23=TDOLAR               (36)
 ------------
  TYPE  FLAG$
  (6)   (30)
 ------------
24=TERROR               (24)
 ---------------------------
  TYPE  COLUMN  WARN  ERROR#
  (6)   (7)     (1)   (10)
 ---------------------------
25=TCARD                SEE 19=TEOB
26=TPROCESSED           --
27=TRDFNS               (63)
 ------------------------------------------------------------------------
  TYPE  COLUMN  URS  RESOLVED OCUR  DNFLAGS  SNAM OCUR  SCOPE
  (6)   (7)     (4)  (12)           (10)     (12)       (12)
 ------------------------------------------------------------------------
28=TLABELREC            SEE 07=TWORD
29 TCSPS                DROPPED
30=TSTATESTOP           **
31=TINDEX               **
32=TSUBS                **
33=TBOOSTOP             **
```

34=TARITHOP
35=TINDSECT

**
THIS TOKEN PUT ON SEGFILE AT SPECIFIC LOCATION
AWAITING PROSYN PROCESSING...SEE SEGFILE LAYOUT

DNINFO: (INTERNAL FILE) DATA DIVISION SYNTAX (V)

(96)

LEVEL	JUST	BWZ	OCCURS VALUE	COPX	PC FLG	LGTH	ADR.	USAGE	#SUBS
(7)	(1)	(1)	(18)	(12)	(1)	(18)	(33)	(3)	(2)

IF PC.FLG THEN LGTH=PCINFO INDEX.
IF LEVEL=50 THEN LGTH=MOM OCUR.
IF #SUBS NEQ 0 THEN SUBSCRIPTED & #SUBS=1,2,OR 3.

ASCII FLAG

0=EBCDIC
1=ASCII

IF ENTRY=0 THEN ENTRY=WORKING STORAGE ELSE ENTRY=FD:

(96)

LEV	SD	FD	UN USED	01 OCUR	FILE LABEL TYPE	FILE HARD	FILE ACCESS	FILE LIMITS	FPB NUM	MAX REC
(7)	(1)	(1)	(32)	(12)	(4)	(7)	(4)	(1)	(9)	(18)

ALLDNFILE: OUTPUT FROM EXPLODE (VI)

```
00=TINTGR                (33+(LENGTH*8))
 ------------------------------------------------------
  TYPE  COLUMN  URS  SCALE  BYTE LENGTH  EBCDIC LITERAL
  (6)   (7)     (4)  (8)    (8)          (255 MAX)
 ------------------------------------------------------
01=TREAL                 SEE 00=TINTGR
02=TNNLIT                SEE 00=TINTGR
03=TFILEREC              (281)
 ------------------------------------------------------------------------------
  TYPE  COL  URS  COPX  BASIC  FPB   PC    DN     JUST  BWZ  MON   USAGE
                        COP    INFO  INFO  FLAGS              INFO
  (6)   (7)  (4)  (12)  (53)   (80)  (74)  (10)   (1)   (1)  (30)  (3)
 ------------------------------------------------------------------------------

04=TXSECN                (42)
 ------------------------------------------------
  TYPE  COLUMN  URS  LABEL FLAGS  SEG#  SAMENAME OCUR
  (6)   (7)     (4)  (6)          (7)   (12)
 ------------------------------------------------
05=TXPAR                 SEE 04=TXSECN
06=TFIXUP                --
07=TWORD                 (281)
 ------------------------------------------------------------------------------
  TYPE  COL  URS  COPX  BASIC  COP   PC    DN     JUST  BWZ  MON   USAGE
                        COP    FACTS INFO  FLAGS              INFO
  (6)   (7)  (4)  (12)  (53)   (80)  (74)  (10)   (1)   (1)  (30)  (3)
 ------------------------------------------------------------------------------
08=TWMON                 --
09=TCSPL                 SEE 07=TWORD
10=TCSPR                 SEE 07=TWORD
11=TQUAL                 --
12=TLABL                 (54)
 ------------------------------------------------
  TYPE  COLUMN  URS  LABEL FLAGS  SEG#  SECTION  LABEL
                                       OCUR     OCUR
  (6)   (7)     (4)  (6)          (7)   (12)     (12)
 ------------------------------------------------
13=TALTER                (62)
 ------------------------------------------------------------------------
  TYPE  COLUMN  URS  LABEL FLAGS  SEG#  SECTION  LABEL  ALTER TABLE
                                       OCUR     OCUR   INDEX
  (6)   (7)     (4)  (6)          (7)   (12)     (12)   (8)
 ------------------------------------------------------------------------
14=TPERF                 (62)
 ------------------------------------------------------------------------
  TYPE  COLUMN  URS  LABEL FLAGS  SEG#  SECTION  LABEL  TERM PAR#
                                       OCUR     OCUR
  (6)   (7)     (4)  (6)          (7)   (12)     (12)   (8)
 ------------------------------------------------------------------------
```

```
15=TPTHRU               SEE 14=TPERF
16=TLMON                --
17=TLQUAL               --
18=TRESWD               (29)
 ----------------------------
  TYPE   COLUMN  KEY (IN HEX)
  (6)    (7)     (16)
 ----------------------------
19=TEOB
 ------
  TYPE
  (6)
 ------
20=TEOF                 SEE 19=TEOB
21=TDOT                 (13)
 -------------
  TYPE   COLUMN
  (6)    (7)
 -------------
22=TPC                  DROPPED
23=TDOLAR               (36)
 ------------
  TYPE   FLAGS
  (6)    (30)
 ------------
24=TERROR               (24)
 ----------------------------
  TYPE   COLUMN  WARN   ERROR#
  (6)    (7)     (1)    (10)
 ----------------------------
25=TCARD                SEE 19=TEOB
26=TPROCESSED           --
27=TRDFNS               --
28=TLABELREC            --
29=TCSPS                --
30=TSTATESTOP           **
31=TINDEX               SEE 07=TWORD (IMPLICIT LEVEL 50)
32=TSUB                 **
33=TBOOSTOP             **
34=TARITHOP             **
35=TINDSECT             **
```

ALLDNFILE: OUTPUT FROM PROCEDURE DIVISION SYNTAX CHECK (VII)

```
00=TINTGR              (26+((LENGTH*SIGNED)*(4 OR 8)))
 ----------------------------------
  TYPE  URS  SCALE  LENGTH  LITERAL
  (6)   (4)  (8)    (8)     (255 MAX)
 ----------------------------------
01=TREAL               SEE 00=TINTGR
02=TNNLIT              SEE 00=TINTGR
03=TFILEREC            (271)
 ------------------------------------------------------------------
  TYPE  URS  COPX  BASIC  FPB    PCINFO  DNFLAGS  JUST  BWZ  MON
                   COP    INFO                              INFO
  (6)   (4)  (12)  (53)   (80)   (74)    (10)     (1)   (1)  (30)
 ------------------------------------------------------------------
04=TXSECN              (23)
 ----------------------------
  TYPE  URS  LABEL FLAGS  SEG#
  (6)   (4)  (6)          (7)
 ----------------------------
  LABEL OCUR NEED NOT BE IN TOKEN SINCE IT CAN BE COMPUTED FROM
  A SEQUENTIAL EXPLICIT LABEL COUNTER. THE ALTER TABLE INDEX CAN ALSO
  BE COMPUTED TO GIVE TO THE CODE GENERATOR.

05=TXPAR               SEE 04=TXSECN
06=TFIXUP              --
07=TWORD               (271)
 ------------------------------------------------------------------
  TYPE  URS  COPX  BASIC  COP      PCINFO  DNFLAGS  JUST  BWZ  MON
                   COP    FACTORS                             INFO
  (6)   (4)  (12)  (53)   (80)     (74)    (10)     (1)   (1)  (30)
 ------------------------------------------------------------------
  NOTE: FOR DATE, TIME, TODAYS-DATE, ETC, A TWORD IS CREATED (INTERNAL
    TO PROSYN) WITH SUBF=0, #SUBS NEQ 0, AND FACTOR1 TELLS WHICH:
    1=DATE, 2=TIME, 3=TODAYS-DATE, 4=TODAYS-NAME
    URS
   ------------------------------------------
    INDEXED.DN  COP RELATIVE INDICATOR  INDEX
    (1)         (2)                     (1)
   ------------------------------------------
    INDEXED.DN =THIS IS AN INDEXED DATANAME (DN INDEXED BY...)
    COP RELATIVE INDICATOR:
       SET INTERNAL TO CODE/GENERATOR
       0=USE SEG# & DISP IN BASIC COP
       1=DISP IS RELATIVE TO REUSEABLE TRASH BASE DISP
    INDEX = 77 C USAGE INDEX.

08=TWMON               --
09=TCSPL               --
10=TCSPR               --
11=TQUAL               --
```

```
12=TLABL                (47)
 ------------------------------------------------------
  TYPE  URS  LABEL FLAGS  SEG#  SECTION OCUR  LABEL OCUR
  (6)   (4)  (6)          (7)   (12)          (12)
 ------------------------------------------------------
  SEARCH ROUTINE IN PROSYN USES LOW ORDER BIT OF URS TO TELL CONDITION
  ROUTINE IN CODEGEN TO USE NEXT SENTENCE OCUR (NS.OCUR). NS.OCUR IS SET
  BY SEARCH ROUTINE IN CODEGEN.

13=TALTER               (55)
 ------------------------------------------------------------------
  TYPE  URS  LABEL FLAGS  SEG#  SECTION OCUR  LABEL OCUR  ALTER INDEX
  (6)   (4)  (6)          (7)   (12)          (12)        (8)
 ------------------------------------------------------------------
14=TPERF                (55)
 ------------------------------------------------------------------
  TYPE  URS  LABEL FLAGS  SEG#  SECTION OCUR  LABEL OCUR  TERM PAR#
  (6)   (4)  (6)          (7)   (12)          (12)        (8)
 ------------------------------------------------------------------
15=TPTHRU               DROPPED (ALL PERFORMS= 14=TPERF)
16=TLMON                --
17=TLQUAL               --
18=TRESWD               (30)
 ----------------
  TYPE  URS  KEY
  (6)   (8)  (16)
 ----------------
    IF KEY=ALPHABETIC OR NUMERIC THEN
    URS
   --------------------
    NOT CLASS  UNUSED
    (1)        (7)
   --------------------
    IF KEY=RELATIONAL OP (LSS,LEQ,NEQ,EQL,GEQ,GTR) THEN
    URS
   -------------------------
    UNUSED  IMPLIED SUBJECT
    (7)     (1)
   -------------------------
19=TEOB                 (6)
 ------
  TYPE
  (6)
 ------
20=TEOF                 SEE 19=TEOB
21=TDOT                 DROPPED (CHANGED TO RESERVED)
22=TPC                  --
23=TDOLAR               (36)
 -------------
  TYPE  FLAGS
  (6)   (30)
 -------------
24=TERROR               (24)
 ---------------------------
```

```
  TYPE   COLUMN   WARN   ERROR#
  (6)    (7)      (1)    (10)
 ----------------------------
25=TCARD                SEE 19=TEOB
26=TPROCESSED           --
27=TRDFNS               --
28=TLABELREC            --
29=TCSPS                DROPPED
30=TSTATESTOP           SEE 19=TEOB
  GENERAL USE IS TO DELIMIT A STATEMENT FOR CODEGEN BECAUSE THE CODEGEN
  CONTROL ROUTINE ALWAYS DOES A GET FOR A RESERVED WORD. EXAMPLE OF USE
  OF TSTATESTOP: THE MOVE VERB MAY HAVE MULTIPLE RECEIVING FIELDS AND
  MUST KEEP SCANNING IN PROSYN UNTIL IT GETS A NON-RECEIVING FIELD TOKEN
  . PUTTING  OUT THE TSTATESTOP MAKES SCANNING IN CODEGEN EASIER. ALSO
  USED IN PROSYN BY SORT, ACCEPT/DISPLAY, DUMP, SET, STOP RUN, AND
  DATA MANAGEMENT.

31=TINDEX               SEE 07=TWORD   % IMPLICIT LEVEL 50
32=TSUB                 (75)
 ----------------------------
  TYPE   URS   COPX   BASIC COP
  (6)    (4)   (12)   (53)
 ----------------------------
33=TBOOSTOP             SEE 19=TEOB
34=TARITHOP             (19)
 ------------------------------------------------------------------------
  TYPE   URS   OPERATOR   OPERATOR   ROUND   SIZE    MULTIPLE RECEIVING
                          CLASS              ERROR   FIELDS
  (6)    (4)   (4)        (2)        (1)     (1)     (1)
 ------------------------------------------------------------------------
    IF DIV3 THEN:
    URS
   ---------------------------------
    REMAINDER ROUNDED   REMAINDER   UNUSED
    (1)                 (1)         (2)
   ---------------------------------
    IF STORE THEN
    URS
   ---------------------------------------------
    UNUSED   EMIT.BOF(1) FOR ** OR / IN COMPUTE
    (3)      (1)
   ---------------------------------------------
    OPERATOR:
       0=ADD,1=SUB,2=:MUL,3=DIV,4=EXP,5=MOD,6=STORE,7=STOREMOD,
       8=DELETE-TOP-OF-STACK,9=CHANGE-SIGN
    OPERATOR CLASS:
       EG: ADD0,ADD1,ADD2,ADD3
           SUB0,SUB1,SUB2,SUB3
           DIV1,DIV2,DIV3,
           MUL1,MUL2

35=TINDSECT             (33)
 ------------------------------------------------
  TYPE   TXPAR ALTER INDEX   TLABL OCUR   TLABL SEG
```

```
 (6)   (8)                  (12)        (7)
 ------------------------------------------------
36=TMINIMUM
 ------------------------------------------------
  TYPE  URS  MINIMUM SCALE  MINIMUM LEFT PART
  (6)   (4)  (8)            (8)
 ------------------------------------------------
```

USED BY COMPUTE TO ESTABLISH MINIMUM SCALE & LEFT PART FOR INTERMEDIATE RESULTS

LABELTABLE: OUTPUT FROM CODEGEN (VIII).

(96 BIT ENTRY...15 PER DISK SEGMENT)
TYPE=3 (LINK)

TYPE	UNUSED	LABEL.OCUR
(2)	(77)	(17)

TYPE=0,1,2 (0=PLAIN ADR,1=AND ADR,2=THEN ADR)

TYPE	UNUSED	SEG#	CODE	#BADDRS	#LENBS	#DISPS	#DADDRS
(2)	(13)	(7)	(22)	(13)	(13)	(13)	(13)

NOTE:#DISPS IS COUNT OF DISPS THAT ALSO HAVE A SEG AS PART OF THE ADDRESS. #DADDRS HAVE DISP BUT NO SEG.

SEGFILE: FROM ALL PRIOR PHASES

NOTE: MERGE, DATSYN, AND CODEGEN EACH HAVE A DISCRETE AREA FOR THEIR TOKENS. 18=TCARDADR IS EMITTED ONLY BY DATSYN. MOST OF THESE TOKENS WERE DESIGNED FOR USE BY CODEGEN.

S E G F I L E H E A D E R (DISK SEGMENT 0...2 DISK SEGMENTS)
SEE COMPILER LISTING FOR CONTENTS

P S E U D O C O D E D I C T I O N A R Y (11 DISK SEGMENTS)
(144 BIT ENTRY...10 PER DISK SEGMENT)
USED TO COLLECT CODE TOGETHER AS USER SPECIFIED

LINK HEAD (20)	LINK TAIL (20)	UNUSED (30)	CODE (22)	#BADDRS (13)	#LENBS (13)	#DISPS (13)	#DADDRS (13)

T I N D S E C T E N T R Y T A B L E (1 DISK SEGMENT)
(10 BIT ENTRY...100 ENTRIES)

NUMBER OF ENTRIES FOR THIS CODE SEGMENT (10)

T I N D S E C T T O K E N S (100 DISK SEGMENTS)
(27 BIT ENTRY...53 PER DISK SEGMENT)

TXPAR (8)	ALTER INDEX	TLABL OCUR (12)	TLABL SEG# (7)

NOTE THAT THIS SETUP LIMITS EACH USER INDEPENDENT CODE SEGMENT TO 53 ALTERABLE GOTO PARAGRAPHS

P A T H D I C T I O N A R Y (DATA MANAGEMENT)

M E R G E T O K E N S
SEE DOCUMENTATION ON TOKENS

D A T S Y N T O K E N S
SEE DOCUMENTATION ON TOKENS

D N D I C T T A B L E
(47 BIT ENTRY...30 PER DISK SEGMENT)

SEG (7)	DISP (21)	LENGTH (18)	ALLOCATE.DISK.ON.CODEFILE (1)

SEG=DATA DICTIONARY INDEX
LENGTH=BYTE LENGTH

```
  INITIALIZE: 1=WORKING STORAGE, FILE LABELS, ETC.
              0=OTHER (FD 01 RECORDS, ETC.)
  LAST DNDICTTABLE ENTRY IS TERMINATOR SET TO ALL 0 BITS
**********************************************************
C O P   T A B L E
  (55 BIT ENTRY...26 PER DISK SEGMENT)
  BASIC COP (TYPE=0)
  ------------------------------------------
   TYPE  SEG  DISP  LENGTH  SUBF  DTYPE  ASCII
   (2)   (7)  (24)  (18)    (1)   (2)    (1)
  ------------------------------------------
  NOTE:IN FINAL COP TABLE:
  IF 1 SUB THEN 1 EXTRA COP TABLE ENTRY...
  IF SUBF=1 THEN FOLLOWED BY
  SUBSCRIPT ENTRY (TYPE=1)
  IF 2 OR 3 SUBS THEN 2 EXTRA COP TABLE ENTRIES.
  ------------------------------------------
   TYPE  #SUBS  UNUSED  BOUND  FACT1  UNUSED
   (2)   (2)    (1)     (24)   (24)   (2)
  ------------------------------------------
  AND IF #SUBS>0 THEN FOLLOWED BY
  SUBSCRIPT ENTRY #2 (TYPE=2)
  ------------------------------------------
   TYPE  UNUSED          FACT2  FACT3  UNUSED
   (2)   (3)             (24)   (24)   (2)
  ------------------------------------------
**********************************************************
F P B   (2 DISK SEGMENTS)
  ONE ENTRY PER FILE DECLARED IN PROGRAM
  ----------------------------------------
   FPB (AS MCP EXPECTS IT)  OTHER FILE INFO
   (1 SEG)                  (1 SEG)
  ----------------------------------------
**********************************************************
A L T E R   T A B L E
  (19 BIT ENTRY...75 PER DISK SEGMENT)
  ----------------------
   TXPAR SEG  TLABL OCUR
   (7)        (12)
  ----------------------
**********************************************************
C O D E G E N   T O K E N S
  SEE BELOW
```

```
00=TCOPX                (29)
 ---------------------------------------------
  TYPE  COPX  SUBFLAG  #SUBS  SUBBED.DN  SEG#
  (6)   (12)  (1)      (2)    (1)        (7)
 ---------------------------------------------
01=TLIT                 (16+((LENGTH+SIGN)*(4 OR 8)))
 ----------------------------
  TYPE  LTYPE  LGTH  LSYMB
  (6)   (2)    (8)   (255 MAX)
 ----------------------------
  LTYPE
  -----
  0=4 BIT UNSIGNED
  1=8 BIT
  2=4 BIT SIGNED
  3=RESERVED
02=TCOP (INLINE)        (64)
 ------------------------------------------------------------
  TYPE  BASIC COP  #SUBS  SUBBED.DN  COP RELATIVE INDICATOR
  (6)   (53)       (2)    (1)        (2)
 ------------------------------------------------------------
    SUBBED.DN:
        IF SUBF AND SUBBED.DN THEN SUBSCRIPTED DATANAME
        IF SUBF AND NOT SUBBED.DN THEN INDEXED DATANAME

03=TDADDR               (32)
 ----------------------------------------
  TYPE  TDADDR RELATIVE INDICATOR  DADDR
  (6)   (2)                        (24)
 ----------------------------------------
    TDADDR RELATIVE INDICATOR:
        00=RESERVED
        01=DATA SEG 0 RELATIVE
        10=RELATIVE TO TRASH

04=TOP                  (18)
 --------------
  TYPE  OP CODE
  (6)   (12)
 --------------
05=TBADDR               (39)
 ------------------------
  TYPE  LABEL.TABLE OCUR
  (6)   (33)
 ------------------------
06=TBITS                (14+LENGTH)
 ----------------------------
  TYPE  BIT LENGTH  BIT STRING
  (6)   (8)         (255 MAX)
 ----------------------------
```

```
07=TSUBORINX          (61)
 ------------------------------------
  TYPE   BASIC COP   COP.REL.INDICATOR
  (6)    (53)        (2)
 ------------------------------------
08=TSEGLINK           (34)
 ----------------------------------------
  TYPE   DISK ADR OF NEXT SECTION   SEG#
         TO BE PLACED IN THIS
         PHYSICAL CODE SEG
  (6)    (20)                       (8)
 ----------------------------------------
09=TFACTOR            (24)
 --------------
  TYPE   FACTOR
  (6)    (18)
 --------------
10=TBOUND             (30)
 --------------------
  TYPE   TABLE BOUND
  (6)    (24)
 --------------------
11=TALTERINDEX        (14)
 --------------------
  TYPE   ALTER INDEX
  (6)    (8)
 --------------------
  NOTE: SIZE OF CONTAINER FOR ALTER TABLE IS NOT KNOWN. THIS IS A
    POINTER RELATIVE TO ALTER TABLE BASE(USED BY GOPAR OPERATOR).

12=TOPNDX             SEE 00=TCOPX
13=TLITFIX            (28+(LENGTH*(4 OR 8)))
 ------------------------------------------------------------------
  TYPE   LTYPE   LENGTH   BIT.DISP   TOADDR RELATIVE INDICATOR   LITERAL
  (6)    (2)     (8)      (10)       (2)                        (160 MAX)
 ------------------------------------------------------------------
  NOTE: LENGTH DOES NOT INCLUDE BIT.DISP OR TOADDR RELATIVE INDICATOR.
    BIT.DISP IS BIT LOCATION IN THE LIT TO BE FIXED UP. THE LENGTH OF
    THE AREA TO BE FIXED UP IS ASSUMED TO BE 24.
  NOTE: FOR I/O TYPE VERBS THE ADR IS CHANGED IN FIXUP TO BE BASE REG.
    RELATIVE.  IE: INCLUDE COP TABLE & EDIT MASK.

14=TSTOP              SEE 19=TEOB
  NOTE: TERMINATES A PORTION OF CODE WHEN CODEGEN ENCOUNTERS A TXSECN
    WITH SEG# NEQ SEG# OF LAST TXSECN.
15                    **
16                    **
```

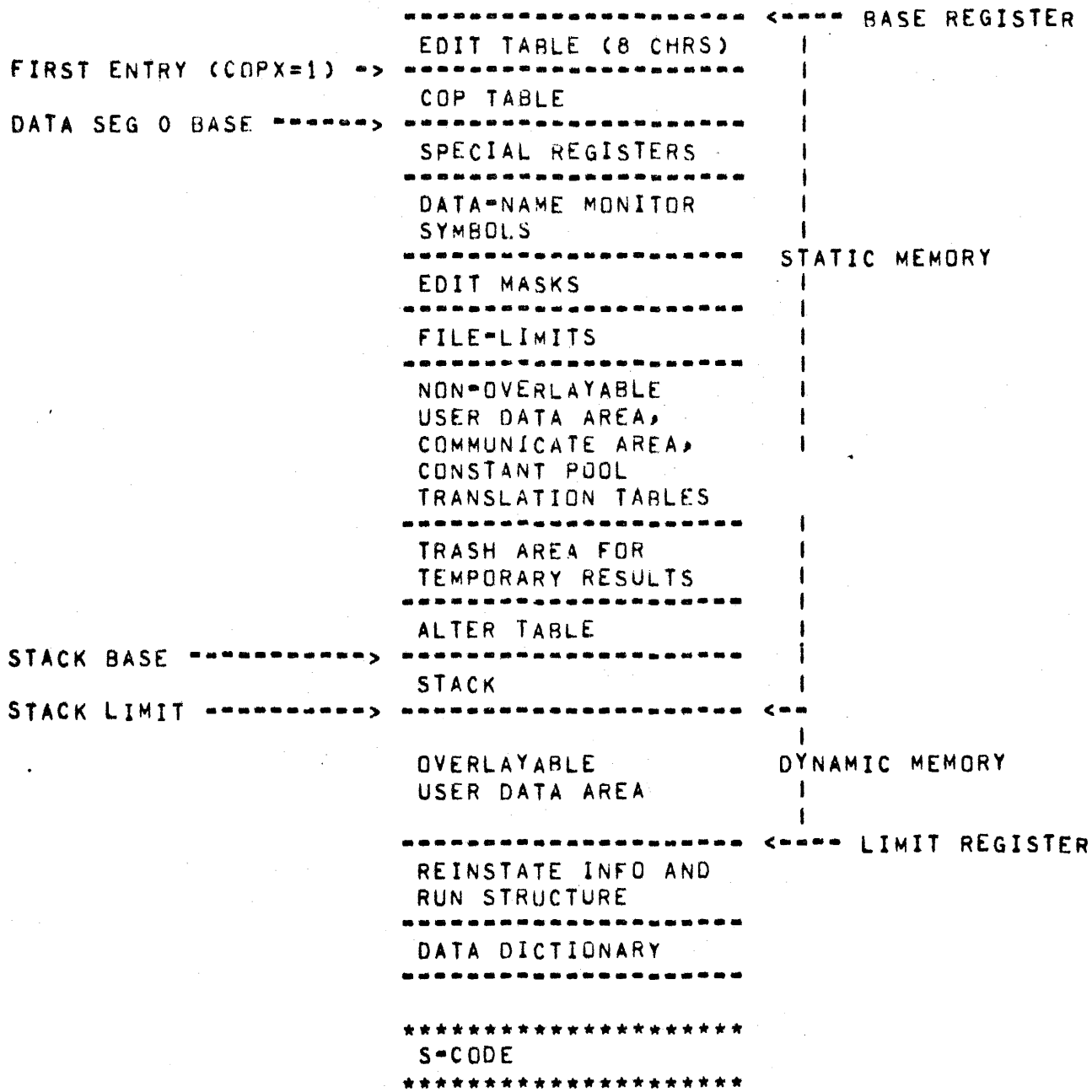
```
17=TMEMVALUE            (70+(LENGTH*(4 OR 8)))
 ------------------------------------------------------------------------
  TYPE  ASCII  LTYPE  LIT     ALL   RIGHT  DNDICT  DIGIT  DEST  LSYMB
        FLAG          LENGTH  FLAG  JUST   TABLE   DISP   SIZE
                                             INDEX
  (6)   (1)    (1)    (8)     (1)   (1)    (10)    (24)   (18)  (160 MAX
 ------------------------------------------------------------------------
  NOTE: MAPS "VALUE" CLAUSE ONTO CODEFILE SO INITIAL VALUES ARE LOADED
    AT RUN TIME.
  DNDICTTABLE INDEX OF 0 INDICATES DSEGO

18=TCARDADR             (51)
 ----------------------------------------
  TYPE  DNDICTTABLE INDEX  DIGIT DISP  COPX
  (6)   (9)                (24)        (12)
 ----------------------------------------
  NOTE: TO GIVE AN ADDRESS ASSOCIATED WITH A CARD TO BE PRINTED ON THE
    REPORT.

19=TEOB                 (6)
 ------
  TYPE
  (6)
 ------
20=TEOF                 SEE 19=TEOB
23=TDOLAR               (36)
 ------------
  TYPE  FLAG$
  (6)   (30)
 ------------
24=TERROR               (24)
 --------------------------
  TYPE  COLUMN  WARN  ERROR#
  (6)   (7)     (1)   (10)
 --------------------------
25=TCARD                SEE 19=TEOB
26=TCOMMLITFIX          (28+(LENGTH*(4 OR 8)))
 ------------------------------------------------------------------------
  TYPE  LTYPE  LENGTH  FIX.CTNUM TDADDR RELATIVE INDICATOR  LITERAL
  (6)   (2)    (8)     (10)      (2)                       (160 MAX)
 ------------------------------------------------------------------------
  FIELDS ARE ANALOGOUS TO THOSE FOR TLITFIX EXCEPT FOR FIX.CTNUM. THE
  LITERAL IS ASSUMED TO BE A COMMUNICATE MESSAGE WHICH    IN WHICH CT.1
  THRU CT.10 MAY NEED TO BE FIXED UP. EACH BIT OF FIX.CTNUM CORRESPONDS
  TO A CT. WHAN THE I**TH BIT OF FIX.CTNUM IS ON, FIXUP WILL FIX THE
  CT.I (THE I**TH CT).
```

CODEFILE: OUTPUT FROM FIXUP (IX).

DISK ADR	CONTENTS
0	PROGRAM PARAMETER BLOCK (PPB)
1	RUN STRUCTURE
2	DATA DICTIONARY
	DATA (THE VARIABLE DSEG0.DISK.ADR POINTS TO HERE)
	CODE DICTIONARY
	CODE
	FILE PARAMETER BLOCK(S) (FPB)
	PATH DICTIONARY (DATA MANAGEMENT)

COBOL S-MEMORY ALLOCATION.

A TYPICAL COBOL PROGRAM LAYOUT IN MEMORY IS AS FOLLOWS:



THE SPECIAL REGISTERS ARE ALLOCATED AS FOLLOWS:

```
SW1             PC 9 CMP
 .
 .
 .
SW8             PC 9 CMP
TALLY           PC 9(5) CMP
JULIAN-DATE     PC 9(5) CMP NOTE: YYDDD
TIME            PC 9(7) CMP NOTE: HHMMSST
TODAYS-DATE     PC 9(6) CMP NOTE: MMDDYY
TODAYS-NAME     PC X(9)
DM-STATUS       PC 9(6) CMP
```

MONITOR STATEMENT

THE MONITOR STATEMENT MUST PRECEDE "IDENTIFICATION DIVISION". SYNTAX = MONITOR <FILE-NAME> [DEPENDING] (<DATA-NAME LIST> : <LABEL LIST>).

THE <LABEL LIST> MAY CONSIST OF ONLY "ALL" WHICH IMPLIES EVERY PROGRAM LABEL.

SYMBOLS FOR MONITORED DATA-NAMES ARE PUT IN DATA SEGMENT ZERO. THIS IS BECAUSE IT IS LIKELY THAT A DATA-NAME WILL BE A RECEIVING FIELD MORE THAN ONCE AND THIS METHOD SAVES CARRYING THE SYMBOLS AS A LITERAL IN THE CODE SEGMENTS.

SYMBOLS FOR MONITORED PROGRAM LABELS ARE SET UP AS LITERALS IN THE CODE WHERE THE LABEL EXPLICITLY OCCURS.

A DATSYN ROUTINE (BUILD.COPS) IS RESPONSIBLE FOR ASSIGNING A MONITOR BUFFER OF 132 CHARACTERS IN DATA SEGMENT ZERO. IT ALSO BUILDS A MONITOR COP AS COP # 1 UNLESS "NOCOP" IS SPECIFIED.

AT CODE SEGMENT ZERO, DISPLACEMENT ZERO, A ROUTINE IS EMITTED WHICH CONSISTS OF A MVN, COMMUNICATE, AND XIT. THIS ACHIEVES A WRITE ON THE USER SPECIFIED LINE PRINTER FROM THE MONITOR BUFFER. THE ROUTINE IS ENTERED AFTER THE MONITOR BUFFER HAS BEEN SET UP AT VARIOUS POINTS IN THE PROGRAM. MOTIVATION FOR THE ROUTINE IS TO SAVE CODE SPACE.

MONITOR TOKENS

EG: MONITOR PRINT (MASTER, XXX : PAR1).

MERGE INPUT		MERGE OUTPUT		
MONITOR	(TRESWD)	MONITOR	(TRESWD)	
PRINT	(TWORD)	PRINT	(TWORD)	
MASTER	(TWMON)	.		
XXX	(TWMON)	.		
PAR1	(TLMON)	.		
		PAR1	(TXPAR)	WHERE THE LABEL APPEARS
		"PAR1"	(TNNLIT)	
		.		
		MASTER	(TWORD)	WITH MONITOR LENGTH AND MONITOR SYMBOL ADDRESS

DATSYN OUTPUT

```
MONITOR (TRESWD)
PRINT   (TFILEREC) WITH MONITOR BUFFER ADDRESS
PERIOD  (TDOT)
```

EG: MONITOR DEPENDING PRINT (A, B, C : ALL).

MERGE INPUT

```
MONITOR (TRESWD)
DEPEND  (TRESWD)
PRINT   (TWORD)
A       (TWMON)
B       (TWMON)
C       (TWMON)
```

NO INDICATION OF "ALL" BECAUSE TNNLITS APPEAR AFTER EACH EXPLICIT LABEL.

SUBSCRIPT AND INDEX OPTIMIZATION.

RULES FOR SUBSCRIPTS.

1. LITERAL SUBSCRIPTS CAN NOT BE SIGNED. THEREFORE, THEY CAN NOT GENERATE TOO LOW AN ADDRESS.

2. THE BOUND IS THE MAXIMUM DIGIT DISPLACEMENT OF THE LAST VALID ENTRY (WHICH IS ANY LENGTH AND MAY BE IN DIGITS OR CHARACTERS).

3. THE INTERPRETER CHECKS EACH SUBSCRIPT LEQ 0. IF SO, THEN ERROR. AFTER ALL GATHERING AND MULTIPLYING OF SUBSCRIPTS IS DONE, THIS IS CHECKED AGAINST BOUNDS AND IF > BOUNDS THEN ERROR.

4. IF ALL SUBSCRIPTS FOR A DATANAME ARE LITERALS, PROSYN CHECKS FOR BOUNDS ERROR. OTHERWISE, EVEN THOUGH A LITERAL MAY BE LARGER THAN THE CORRESPONDING OCCURS VALUE IN THE DATA DIVISION DECLARATION, WE DONT CHECK (THE VALUE OF ANOTHER SUBSCRIPT MAY OFFSET THIS VALUE AND BRING THE TOTAL WITHIN BOUNDS).

RULES FOR INDEXES.

1. DATA-NAME(INDEX + LITERAL) : THE DATA-NAME ADDRESS IS OPTIMIZED AND THE LITERAL DROPPED.

2. DATA-NAME(INDEX - LITERAL) : IF THE NEGATIVE LITERAL PLUS THE DATA-NAME ADDRESS ARE GTR 0, WE OPTIMIZE. OTHERWISE, WE DONT KNOW IF THE INDEX WILL MAKE THE FINAL ADDRESS WITHIN BOUNDS...SO WE DONT OPTIMIZE AND WE LET THE INTERPRETER CATCH ANY ERRORS.

GLOBAL.DOLLAR FORMAT

BIT	RESERVED WORD FOR $ CARD	
0	LIST	
1	DOUBLE	1=DOUBLE,0=SINGLE (DEFAULT)
2	SUPPRESS	
3	SPEC	
4	CONTROL	
5	CODE	
6	ANSI	(VS. DEFAULT B3500 COBOL WHICH ASSOCIATES "ELSE" WITH SIZE ERROR CLAUSE AND NORMAL I/O AT END AND INVALID KEY)
7	PARSE	(MONITOR)
8	DICT	(MONITOR)
9	DNQUAL	(MONITOR)
10	LQUAL	(MONITOR)
11	MOIGE	(MONITOR)
12	DATSYN	(MONITOR)
13	EXPLODE	(MONITOR)
14	PROSYN	(MONITOR)
15	CODEGEN	(MONITOR)
16	FIXUP	(MONITOR)
17	DEBUG	(NO DEBUG YIELDS NO CODE FOR COBOL MONITOR OR DUMP...DEBUG IS DEFAULT)
18	ERRMESS	(GIVES LISTING OF ALL COBOL ERRORS)
19	NOCOP	
20	REFERENCE	(PRODUCES MONITOR CODE FOR VARIABLES IN COBOL MONITOR LIST AND REFERENCED AS OTHER THAN RECEIVING FIELD... EG: IF A=B.)
21	TIME	
22	EXAMINE	(ONLY IF S"OP PRESENT IN INTERPRETER)
23 - 28	UNDEFINED	
29	RESERVED	

USE OF DYNAMIC

I. PARSE
   A. RESERVED WORDS
      1. MONITOR DATA DIVISION = 14936 BITS.
      2. PROCEDURE DIVISION = 10944 BITS.
   B. COPY REPLACING A BY B....
      1. A REQUIRES 14+(8*LENGTH OF A BITS).
      2. B REQUIRES:
         A. B IS WORD 14+(8*LENGTH OF B BITS).
         B. B IS QUALIFIED WORD.
            14+(8*LENGTH B)+
            14+(8*LENGTH OF EACH QUALIFIER)+
            22 FOR EACH IN/OF.
         C. B IS A LITERAL
            22+(8*LENGTH OF B).
      3. LINKS REQUIRE 24 BITS PER REPLACING ENTRY (I.E. EACH A BY B).
   C. MNEMONIC NAMES
         24+(8*LENGTH OF MNEMONIC SYMBOL) FOR EACH
         +16 BITS FOR AN END OF LIST

II. DATA NAME QUALIFICATION
   EACH EXPLICIT DATA NAME REQUIRES 34 BITS (MAXIMUM OF 5000 ENTRIES).
   BUT OVERALL MAX OF 65K BITS NOW = 1928.

III. LABEL QUALIFICATION
   EACH EXPLICIT PARAGRAPH/SECTION NAME REQUIRES 37 BITS MAX. OF 4000
   ENTRIES) = 1783.
   MAXIMUM OF 50 ALTERED GO TO PARAGRAPHS.

IV. MERGE
   A. CONDITION NAMES EXPANSION.
      EACH 88 VALUE SPECIFIED REQUIRES 33 BITS IN ADDITION TO THE
      LENGTH OF THE VALUE CHARACTER STRING ITSELF.
      ADD 53 BITS OVERHEAD.
   B. MONITOR FUNCTION.
      EACH DATA-NAME MONITORED REQUIRES 42 BITS.
      EACH LABEL MONITORED REQUIRES 18 BITS IN ADDITION TO THE
      CHARACTER STRING OF THE LABEL ITSELF.

V. DATSYN
   A. PICTURES.
      EACH UNIQUE PICTURE REQUIRES 93 BITS.
   B. FD'S.
      EACH FD REQUIRES 293 BITS.
   C. DATA SEGMENTATION.
      EACH GROUP OF DECLARATIONS (FILE RECORD AREAS, 77'S, 01'S)
      WHICH ARE CANDIDATES FOR A DATA SEGMENT REQUIRES 47 BITS FOR
      TEMPORARY PSEUDO DATA DICTIONARY.

1ST 3 PASSES -- PROCEDURE DETAILS

```
GENERAL DESCRIPTION OF INITIAL PARSING :
OVERALL THE PASS IN BROKEN INTO TWO PARTS
1. PARSING OF MONITOR THRU DATA DIVISION.
2. PARSING OF THE PROCEDURE DIVISION.
INVOLVED IN DOING THE PARSE ARE PROCEDURES FOR
1. MERGING SOURCE LANGUAGE INPUTS (INCLUDING LIBRARY COPY)
2. SCANNING AND ISOLATING SYMBOLS (RESERVED, DATA NAMES, LABELS ETC)
3. WRITING OUTPUT FILES (ALLFILE,DNFILE, REPORT,LIBRARY,NEW SOURCE)
4. LOOKING UP RESERVED WORDS
5. COPYING SUBSCRIPT LISTS
6. REPLACING ITEMS IN COPY
7. BASIC SYNTAX CHECKING FOR
   A. OVERALL :
    1. THAT ALL DIVISINS ARE PRESENT AND IN PROPER ORDER
    2. FIRST ITEM IS A RESERVED WORD (EXCLUDING $ CARDS)
   B. MONITOR :
    1. HAS A FILE NAME
    2. LEFT AND RIGHT PARENS PRESENT
    3. STMT TERMINATED BY A PERIOD
   C. ID  DIVISION :
       EXCEPT FOR PGM-ID THE COMPILER FINDS THE NEXT
       FIXED WORD IN COLUMNS 8-11
       ALSO SUPPLIES DATE-COMPILED
   D. ENVIRONMENT DIVISION :
    1.  OBJECT-COMPUTER
      A.  VERIFIES AND SAVES MEMORY SIZE
      B.  VERIFIES AND SAVES CODE AND DATA SEGMENT
    2.  SPECIAL-NAMES :
      A. SAVES CURRENCY SYMBOL
      B. SETS DECIMAL-PT  COMMA
      C. BUILDS A LIST OF MNEMONIC NAMES
    3. FILE-CONTROL:
       JUST COPIES
    4. I-O-CONTROL :
       COPIES IT
   E. DATA DIVISION :
    1. ISOLATES FILE SECTION -ITS FD SD AND RECORDS FROM
       W-STORAGE SECTION
    2. FD, SD AND W-STOR GO TO DN FILE WITH LEVEL "00"
        AND EXCEPT FOR W-S THE FD FLAG IS SET TRUE
     FILE SECTION
      1.  EACH ENTRY MUST START WITH "FD" OR "SD"
      2.  A FILENAME MUST BE PRESENT
      3.  EVERYTHING FROM FD TO A PERIOD IS COPIED
          A. EXCEPT DATA AND LABEL RECORDS
             DATA REC CLAUSE DROPPED
             LABEL RECORD - TLABELREC
      4.  FOR THE RECORD DESCRIPTIONS :
          EACH MUST START WITH AN INTEGER LEVEL -ERROR IF NOT,
```

```
            ERROR IF LEVEL IS "00",
            FILLER OR TWORD MUST FOLLOW LEVEL -ERROR IF NOT,
            ERROR IF A FIXED NAME FOUND BEFORE A PERIOD IS FOUND,
         WORKING-STORAGE
           SAME AS FOR FILE RECORDS
      F. PROCEDURE DIVISION
         1. IF DECLARATIVES:
          A. ERROR IF NO PERIOD FOLLOWING "DECLARATIVES",
          B. ERROR IF SECTION NAME MISSING,
          C. ERROR IF NO END DECLARATIVES,
          D. ERROR IF NO USE STMT,
          E. IF END DECL FOUND BUT NOT "DECL" THEN ERROR,
          D. END DECL FOLLOWED BY PERIOD,
          E. IF "END" IN COL 8-11 - CONSIDERED "END DECL",
         2. GENERAL:
          A
          A. ERROR IF FIRST ITEM NOT A PARAGRAPH OR SECTION NAME
              OR DECLARATIVES,
          B. ERROR IF NO PARAG. FOLLOWING SECTION,
          C. LAST INPUT MUST BE TERMINATED BY A PERIOD,
          D. "EXIT" MUST BE FOLLOWED BY A PERIOD,
          E. PRIORITY GTR 99 -ERROR,
          F. MISSING PERIOD AFTER "SECTION" AND PARAGRAPH,
          G. FOR THOSE VERBS REFERENCING LABELS -ERROR IF
              LABEL NOT TWORD OR TINTEGER,
          H. "FROM" OR "TO" MISSING ON SUBTRACT, ADD, MOVE, ALTER
          I. FILE NAME MISSING FROM READ, SELECT
      GENERAL PROCEDURE DIVISION PROCESSING INVOLVES
         1. RECOGNIZE AND TYPING EXPLICIT SECTIONS AND PARAG.
         2. RECOGNIZE THOSE VERBS THAT HAVE LABEL REFERENCES AND
            MARKING THE LABELS
         3. SUPPLYING A SEGMENT NUMBER TO OUTPUT TOKENS(TXSECN,TXPAR)
         4. FOR THOSE VERBS WHERE LABELS ARE REQUIRED - VERIFY THAT
            A LABEL IS PRESENT AND IS AN INTEGER OR WORD
         5. IF DECLARATIVES IS END DECLAR. PRESENT AND VICE-VERSE
         6. RECOGNIZING "CORRESPONDING" AND MARKING ITS OPERANDS

PROCEDURE DOLLAR;
  1. SCANS A $-CARD & RETURNS DIAGNOSTICS
  2. LOOKS UP CONTROL WORDS (DOLL.WORDS ARRAY)
  3. SETS GLOBAL.DOLLAR BOOLEANS ACCORDINGLY.
  4. OUTPUTS TO REPORT FILE (TO PROVIDE FOR ERROR PRINTING & "CONTROL")
  5. BUMPS CARD.MARKER.CT

ROUTINE FEED;
   1. COLLATES INPUT FROM CARDS, TAPE/DISK & LIBRARY
   2. WRITES NEW TAPE/DISK IF REQUESTED
   3. PERFORMS SEQUENCE CHECK IF REQUESTED (DEFAULT=ON)
   4. REPORTS EXPLICIT LABELS (CLABLF)
   5. PURGES BLANKS BEYOND COL 12 OF CONTINUATION CARDS
   6. WRITES REPORT FILE
   7. MAINTAINS CARD-MARKER COUNT
   8. WRITES NEW LIBRARY IF REQUESTED
   9. VOIDS TAPE/DISK RECORDS IF REQUESTED
```

10. RESEQUENCES OUTPUT IF REQUESTED

```
PROCEDURE DEBLANK;
        1. SKIPS BLANK COLUMNS OF THE CARD IMMAGE

PROCEDURE GET.WORD VARYING;
        1. ISOLATES A TOKEN WHICH CONFORMS TO"WORD" SPELLING RULES

PROCEDURE GET.NUMBER VARYING;
        1. ISSOLATES NUMBERS OR HYPHEN-NUMBERS OR WORDS
        2. PRIOR TO INITIAL ENTRY SCLAS=NUMBER.C

PROCEDURE GET.NNLIT VARYING;
        1. ISOLATES STRINGS THAT START WITH A QUOTE
        2. INCLUDES FINAL QUOTE WITH THE STRING AND RETURN NNLIT.C
        3. IF END.OF.CARD ENCOUNTERED RETURNS QUOTE.C

PROCEDURE GET.UNDIGIT.LIT VARYING;
        1. ISSOLATES AN UNDIGIT LITERAL

PROCEDURE GET.NEXT.TOKEN;
        1. ISSOLATES A NEW TOKEN ON THE CARD IMMAGE AND SAVES THE
         SYMBOL IN ISYMBOL IF NECESSARY
        2. IF SCLAS IS OTHER THAN NEW.TOKEN.C THEN ISOLATES THE
         CONTINUED TOKEN AND CONCATONATES IT WITH ISYMBOL
         WHEN NECESSARY
        3. REPORTS THE SCAN CLASS (SCLAS) OF THE TOKEN

PROCEDURE SCAN;
        1. GETS A TOKEN FROM THE CARD IMMAGE
        2. SAVES THE CHARACTER VALUE(NCV) FOR ANYONE INTERESTED
        3. DEBLANKS THE CARD IMMAGE FOLLOWING THE TOKEN
        4. LOOKS FOR A CONTINUATION CARD("-" IN COLUMN 7)

PROCEDURE NOT.FOUND;
        1. VERIFIES THAT THE SYMBOL LENGTH LEQ 30 AND DOESNT END IN "-"
        2. CHECK FOR REPLACING CANDIDATES
        3. CHECKS FOR APPEARENCE IN COLUMNS 8-11

PROCEDURE GET.REAL.LIT;
        1. COMBINES THE INTEGER AND FRACTION PARTS OF A REAL
          LITERAL
        2. ACCOUNTS FOR DECIMAL-POINT IS COMMA DECLARATION
        3. ALLOWS FOR MULTIPLE DECIMAL POINTS
        4. ASSUMES THAT TLGTH= LENGTH OF THE INTEGER PART
          AND THAT THE FRACTIONAL PART IS READY TO BE SCANNED

PROCEDURE SCANNER;
        1. COMBINES TOKENS FROM SCAN INTO LARGER CONSTRUCTS
        2. LOOKS UP TWORD IN RESERVED WORD LIST AND SKIPS NOISE
          WORDS,NOTE SENTENCES,AND NOTE PARAGRAPHS
        3. DETECTS THE DECIMAL POINT OF A REAL NUMBER
        4. DETECT ADJACENT QUOTES IN A NNLIT
        5. DETECTS "**" AS A SPECIAL CASE
```

```
        6. TRANSLATES UNDIGITS A-F INTO THEIR HEX EQUIVALENTS

PROCEDURE SCTRL;
        1. CALL ON RPLGET IF A COPY...REPLACING IS IN PROGRESS
        2. IF DOTFLG WAS SET ON A PRIOR CALL THEN A PERIOD IS
           SUPPLIED THIS CALL
        3. CALLS SCANNER FOR NEXT TOKEN
        4. CHECK FOR LIBRARY END-OF-FILE

PROCEDURE MAKALL.RETURN VARYING;
    WRITE TOKEN AND CAUSE A RETURN IN CALLING PROCEDURE

PROCEDURE TIME.TO.QUIT VARYING;
    DETERMINES END OF SUBSCRIPT LIST AND RETURNS

PROCEDURE MKSUBS;
    WRITES SUBSCRIPTS COMBINING ANY SIGNS WITH THE FOLLOWING
    INDEXES ON RELATIVE INDEXING

PROCEDURE SEARCH.MNEMONICS(ADDIT)VARYING;
    SEARCHES MNEMONIC NAME PORTION OF DYN.WA-IF ADDING IT ADDS TO THE
    LIST-ADDING BEGINS AT END OF COPY LIST

ROUTINE MAKLIB;
    HANDLES CHANGING TO CREATING LIBRARY FILES AND
    BACK AGAIN.
    THREE CASES HANDLED
    1. JUST STARTING NEW LIB
    2. FINISHING LIB
    3. FINISHING PREVIOUS LIB AND STARTING ANOTHER BY HAVING
       LIB NAME ON PREVIOUS LIB S TERMINATING "L" CARD
       I.E.    L  "LIB1"
               .
               .
               L  "LIB2"

PROCEDURE MAKLIB.SCAN VARYING;
    TO CAUSE GETTING  OF LIBRARY FILE NAME

PROCEDURE COPYSCAN.NOT.PERIOD VARYING;
    SPEIAL CALL ON SCAN BY COPY IS TO PREVENT THE DEBLANKING
    AND CALL ON FEED ON THE BREAK CHARACTER
    THIS ALLOWS COPYING TO START ON THE IMMEDIATE NEXT CARD

PROCEDURE CHECK.COPY.LIST(ADDIT);
    SEARCHES THE DYN.WA FOR A MATCH ON THE OP.A  IN "REPLACING OP.A BY
    OP.B" IF ADDIT THEN DUPLICATES CAUSES ERRORS
          IF NOT ADDIT THEN SCTRL WANTS THE TEXT TO BE PLACED
          IN APPROPRIATE AREAS(TYPE,SCALE,ISYMBOL.LEN,ISYMBOL,SKEY)
          A CALL ON RETURN.COPY.TEXT DOES IT

PROCEDURE RETURN.COPY.TEXT;
    SETS UP SCTRL VARIABLES FROM COPY REPLACING INFO AS SCTRL
    WOULD IF IT HAD ISOLATED THE TOKEN FROM THE INPUT
```

```
PROCEDURE ERRURS.IN.OP.B VARYING;
    SYNTAX CHECKS OP.B -RECURSES TO HANDLE QUALIFIERS AND SUBSCRIPTS

PROCEDURE PLACE.OPB.IN.LIST VARYING;
    IF SPACE AVAILABLE IN DYN.WA THIS PLACES OPB INTO DYN.WA

PROCEDURE SUBSCRIPT.ERRORS VARYING;
  SYNTAX CHECKS SUBSCRIPTS OF OPB

PROCEDURE REPLACING.ERRORS VARYING;
    SYNTAX CHECK OF "REPLACING  OP.A BY OP.B" PAIRS-IF ANY ERRORS IT
    RETURNS A 1.

PROCEDURE COPY.PROC VARYING;
    TO SYNTAX COPY STATEMENT AND CAUSE THE SAVING OF REPLACING
    ITEMS IN A LIST

PROCEDURE INVOKE.PROC VARYING;
         1. VERIFIES COPY NOT NESTED
         2. VERIFIES LEVEL="01" IN THE DATA-BASE SECTION
         3. OPENS THE DATASET LIBRARY WHOSE NAME IS
           "ODL.DBNAME"/DATASET.NA4E

PROCEDURE DB.LIT.TO.ISYMBOL;
         1. BUILDS ISYMBOL DESCRITOR AND FILLS IT WITH "DATA-BASE"

PROCEDURE PUT.DB.TQUAL;
         1. PROVIDES "DATA-BASE" AS A TQUAL ON THE DNFILE

PROCEDURE PUT.COMPONENT.LIST;
         1. PARSES A COMPONENT LIST
         2. SUPPLIES "DATA-BASE" AS A TQUAL FOR EACH COMPONENT-NAME

PROCEDURE OUTPUT.FIXUP;
     TO OUTPUT A FIXUP TOKEN ON ITEMS NOT QUALIFYING FOR CORRESPONDING

PROCEDURE NTRY.ERROR(ERRNO) VARYING;
  . OUTPUTS ERROR AND RETURNS 0-TOUT,1-FD,SD,DB,FIXED NAME
```

MERGE PROCEDURE DETAILS

```
TINTGR TYPE 0
    VARIABLE LENGTH
   2. MERGE
   3. MERGE
   4. MERGE
                          *
TDDC  TYPE 1
   3.  DROP ON ONFILE (AT START OF BLOCK)
TREAL TYPE 1  (VARIABLE LENGTH)
   2. MERGE ALLFILE ONLY
   3. MERGE ALLFILE ONLY
   4. MERGE ALLFILE ONLY
                          *
TNNLIT TYPE 2
    VARIABLE LENGTH
   1. MERGE ALLFILE ONLY, MAY BE DROPPED, MAX LENGTH IS 30 CHAR
   2. MERGE ALLFILE ONLY
   3. MERGE ALLFILE ONLY
   4. MERGE ALLFILE ONLY
                          *
TXDN  TYPE 3
   3. MERGE & BUILD CONDITION NAME TABLE IF CONDITION FLAG
                          *
TXSECT   TYPE 4
   4. MERGE
         IF LMONITOR AND NOT MONITOR=ALL CHECK MONITOR FLAG
          AND PUT MONITOR SYMBOL
                          *
TXPAR    TYPE 5
   4. MERGE
         IF LMONITOR AND NOT MONITOR=ALL CHECK MONITOR FLAG
          AND PUT MONITOR SYMBOL
                          *
TFIXUP   TYPE 6              (DOES NOT OCCUR)
                          *
TWORD    TYPE 7
   1. MERGE  (FILE NAME)
   2. MERGE
   3. MERGE
   4. MERGE  - IF COND FLAG LOOK UP & EXPAND CONDITION NAME
         ALSO MAY GET  ( ) TWORD,TINTGR TERROR,SIZE ERROR,ROUNDED.
         CREATE TERROR  IF  INVALID THRU OPERANDS.
         IF WORD MONITOR CHECK MONITOR FLAG, LOOK UP AND
          PUT MONITOR SYMBOL MEMORY ADDRESS
TWMON    TYPE 8
   1.  BUILD MONITOR TABLE  - DONT OUTPUT, DROP
                          *
TCSPL    TYPE 9
TCSPR    TYPE 10
```

```
   4. MERGE 2 FILES,  TRICKY IF ERRORS.
      PAIRS OCCUR ON DNFILE & NOT ON  ALLFILE.
      SAVE SUBSCRIPTS FROM ALLFILE AND EXPAND.
      MAY OR MAY NOT BE DELIMITED BY  TCSPS.
                          *
TQUAL     TYPE 11               (DOES NOT  OCCUR)
                          *
TLABL     TYPE 12
    4. MERGE
                          *
TALTER    TYPE 13
    4. MERGE
                          *
TPERF     TYPE 14
    4. MERGE
                          *
TPTHRU    TYPE 15
    4. MERGE
                          *
TLMON     TYPE 16
    1. BUILD MONITOR TABLE - DONT OUTPUT, DROP.
                          *
TLQUAL    TYPE 17         (DOES NOT OCCUR)
                          *
TRESWD    TYPE 18
    DOES NOT OCCUR ON DNFILE
   1. BUILD MONITOR TABLE  & MERGE
   2. MERGE
   3. MERGE
   4. MERGE
                          *
TEOB      TYPE 19
    TRANSPAREN TO CONTROL ROUTINES
   1. -
   2. -
   3. -
   4. -
                          *
TEOF      TYPE 20
   4.   CLOSE FILES AND PREPARE FOR NEXT PASS
                          *
TDOT      TYPE 21
   1. MERGE ALLFILE ONLY
   2. MERGE ALLFILE ONLY
   3. MERGE ALLFILE ONLY - DROP IF FOLLOWS COND.NAME ENTRY
   4. MERGE ALLFILE ONLY
                          *
TPC       TYPE 22
   3. OUTPUT MASK TO SEGFILE.
      MAKE PCINFO FILE & OUTPUT
                          *
TDOLAR    TYPE 23
    TRANSPAREN TO CONTROL ROUTINES,
    RESET GLOBAL DOLLAR FLAGS
```

```
   1. MERGE ALLFILE ONLY
   2. MERGE ALLFILE ONLY
   3. MERGE ALLFILE ONLY
   4. MERGE ALLFILE ONLY
                          *
TERROR   TYPE 24
    ALLFILE ERRORS MAY OR MAY NOT MATCH DNFILE TOKEN.
    DNFILE  ERRORS MATCH ALLFILE TOKEN.  EXCEPT IN "CORR"
    OUTPUT ALL ERRORS TO ADNFILE
    1. MERGE FROM EITHER FILE
    2. MERGE FROM EITHER FILE
    3. MERGE FROM EITHER FILE
    4. MERGE FROM EITHER FILE
         IF"CORR"  COMPLICATIONS MAY ARISE
         CREATE ERROR IF  INVALID  PICTURE  OR  CONDITION NAME.
                          *
TCARD    TYPE 25
    MUST  BE KEPT IN CORRECT SEQUENCE WITH ERRORS
   1. MERGE ALLFILE ONLY
   2. MERGE ALLFILE ONLY
   3. MERGE ALLFILE ONLY
   4. MERGE ALLFILE ONLY
                          *
TPROCESSED    TYPE 26        (DOES NOT  OCCUR)
                          *
TRDFNS   TYPE 27
    3. MERGE
                          *
TLABELREC  TYPE  28
    3. MERGE
                          *
TCSPS    TYPE 29
    4. DELIMITS "CORR" SERIES ON DNFILE ONLY  - DROPPED
                          *
 TYPE  30 AND ABOVE DO NOT OCCUR IN  PASS 4.
                          *

    C O R R E S P O N D I N G   C O M B I N A T I O N S
  ALL        DN                   ALL        DN
  ***        ***                  ***        **
  MOVE                            MOVE
  CORRESP                         CORRESP
  TCSPL(A)   TCSPL(A)             TCSPL(A)   TERROR
  TERROR                          TCSPR(B)   TCSPR(B)
  ..........................................
  MOVE                            MOVE
  CORRESP                         CORRESP
  TCSPL(A)   TCSPL(A)             TCSPL(A)   TCSPL(A)
  TCSPR(B)   TCSPR(B)             TCSPR(B)   TERROR
             TCSPL(C OF A)
             TCSPR(C OF B)
             TCSPL(D OF A)
             TCSPR(D OF B)
             TCSPS
```

```
*************************************************
MOVE                              MOVE
CORRESP                           CORRESP
TCSPL(A)  TCSPL(A)                TCSPL(A)   TERROR
TCSPR(B)  TCSPR(B)                TCSPR(B)   TERROR
          TERROR
          TERROR
*************************************************

PROCEDURE PUT;
    PUTS PRE-FORMMATED TOKEN FROM WORKAREA TO ADNBUFF

PROCEDURE DYN.LIM.CHECK(DESC) BIT(48);
     ASSURE DESC IS WITHIN THE LIMITS OF DYNAMIC MEMORY

PROCEDURE PUT.TO.WA;
    PUTS TOKEN FROM ALLBUF TO WORKAREA & COMBINES ON TOKEN IF APPLICABLE
    FORMATS TOKENS READY TO MOVE TO ADNBUF

PROCEDURE MCTBL.SETUP(P1,P2,P3);
    1. INITIALIZES THE MASK CONTROL TABLE

ROUTINE  PC.SETUP;
    1. INITIALIZES THE MASK CONTROL TABLE WHICH IS CONSTRUCTED
      AS FOLLOWS:
      CLI BIT(3)   CLASS INDEX USED TO SET A CLASS FLAG IN CLASS.WA
      MA  BIT(4)   MASK ADDRESS TO BE EMITED
      MC  BIT(5)   MASK CONTROL USED AS A CASE VARIABLE AND FOR
                   PRECIDENCE CHECKING I.E.<+><$><Z> IS PROPER
    2. INITIALIZES THE CLASS TABLE WHICH IS INDEXED BY THE CLASS.WA
      THE CLASS.WA BITS TAKE ON THE FOLLOWING MEANING:
        0:DUMMY
        1:NUMERIC EDIT  SPECIFIED I.E. + - $ Z * . ,
        2:X SPECIFIED
        3:A SPECIFIED
        4:9 SPECIFIED
        5:B SPECIFIED
        6:INSERT SPECIFIED I.E. 99/99/99
        7:0 SPECIFIED
    3. IF DECIMAL POINT IS COMMA WAS SPECIFIED THEN THE MASK CONTROL
      ENTRY FOR "," AND "."ARE INTERCHANGED
    4. IF CURRENCY IS LITERAL WAS SPECIFIED THEN THE MASK CONTROL ENTRY
      FOR "$" IS INTERCHANGED WITH THE ENTRY FOR THAT LITERAL

PROCEDURE PCERROR(PCERROR.NO);
    1. PROCESSES ERRORS IN A PICTURE STRING

PROCEDURE MAKMSK(INSTR);
    1. EMITS A MASK OPERATOR AND MASK ADDRESS
    2. RETURNS IF NO MORE ROOM IN MSKWA OR IF RPT=0

ROUTINE  GETPCHR;                                  GET A PICTURE CHARACTER
    1. GETS A PICTURE CHARACTER                   *
    2. CHECKS FOR REPEATED CHARACTER I.E. 999
```

3. CHECKS FOR REPLICATION FACTOR I.E. 9(3)
4. UPDATES MAP SIZE COUNTER
5. GETS THE MASK CONTROL TABLE ENTRY AS A FUNCTION
   OF THE ERCDIC CHARACTER OBTAINED
6. SETS A CLASS BIT USING THE CLI INDEX OF MCTRL ENTRY OBTAINED

ROUTINE UPSIZE;
1. UPDATES THE LOGICAL SIZE COUNTER(SIZE)
2. SAVES PREVIOUS MASK CONTROL

ROUTINE MOVIT;
1. EMITS A MOVE DIGITS OPERATOR
2. UPDATES LOGICAL SIZE

ROUTINE FLTCHK;
1. CHECKS THE FLOAT TOGGLE
2. EMITS AN INSERT FLOAT OR A MOVE DIGITS

ROUTINE INSRT;
1. CHECKS THE SUPPRESS TOGGLE
2. EMITS AN INSERT SUPPRESS OR
3. EMITS AN INSERT UNCONDITIONALLY AND
4. SAVES MCTRL AND SETS INUFLG FOR SIZE=0

DATSYN PROCEDURE DETAILS

```
PROCEDURE DATSYN.EXPLODE;
  ************  MANAGEMENT OF DYNAMIC WORK AREA (DYN.WA) **************
        1. THE PC.TABLE IS LOADED FROM THE PCFILE INTO THE FIRST PART
           OF DYN.WA AT THE BEGINNING OF DATSYN
        2. FDI.LIST.HEAD CONTANS A LINK TO THE FIRST FILE INFO ENTRY
           (FDI.INFO) LINK LIST -ONE ENTRY PER SELECT(ED) FILE
        3. THE DATA DICTIONARY IS BUILT BEGINNING AT THE END OF DYN.WA
           AND GOING TOWARD THE FD.INFO TABLE-ONE ENTRY FOR EACH DATA
           SEGMENT CANDIDATE WITH DDIC[0] BEING RESERVED FOR THE
           LABEL RECORD WORK AREA IN USE PROCEDURES.
        4. IF DYN.PTR EVER MEETS DDICT.LIMIT AN OUT-OF-MEMORY
           CONDITION EXISTS.
        5. AT THE BEGINNING OF EXPLODE THE FDI.INFO LINK LIST AND THE
           DATA DICTIONARY TABLE ARE REPLACED BY AS MANY DNINFO FILE
           ENTRIES AS WILL FIT WITH THE OVERFLOW HANDLED ON A RANDOM
           RETRIVEAL BASIS
                        DYN.WA
                        ----------------------
                           PC.TABLE LOADED
                           FROM PCINFO FILE
   ---------------      ----------------------
    FDI.LIST.HEAD ---->                          <----- DNI.TABLE.BASE
     (24)                  FD.INFO LINK
   ---------------         LIST                          FOR DNINFO FILE
                                                         ENTRIES DURING
                        ----------------------           EXPLODE
   DYN.PTR ----------->                                  (AS MANY AS WILL
                           AVAILABLE                      FIT)
                        ----------------------
   DDICT.LIMIT ------->
                           DDICT TABLE
   DDICT.BASE -------->                          <----- DNI.TABLE.LIMIT
                        ----------------------
                        ----------------------

PROCEDURE GET.PC.INFO VARYING;
        1. VERIFIES THAT THE PC.INFO FOR A PC.OCUR IS IN THE PC.TABLE
        2. MOVES IT TO PC.INFO
        3. SETS SIZE TO CORE.SIZE

PROCEDURE FIND.DNINFO(OCUR); FORMAL OCUR VARYING;
        1. FINDS THE BLOCK CONTAINING THE DNINFO FOR THIS OCUR
        2. IF THE BLOCK IS NOT IN MEMORY THEN THE CURRENT BLOCK
           IS WRITEN AND THE REQUIRED BLOCK IS READ

PROCEDURE GET.DNINFO(OCUR);
        1. GETS A RECORD FROM THE DNINFO FILE

PROCEDURE  GET;
```

```
          1. GETS A TOKEN FROM THE ALLDN FILE INTO THE ADN.WA
          2. USING ADN.TABLE(TYPE) FOR UNBLOCKING CRITERIA AND ACTION
             TO BETAKEN, TOKENS OF INTEREST WILL BE RETURNED TO THE
             CALLER AND OTHERS WILL BE WRITTEN TO THE OUTPUT FILE

PROCEDURE MAK.FWD.REFERENCE(FWD.REF);
          1. CREATES A FORWARD REFERENCE AT FWD.REF
          2. ALLOWS A TWORD OR TINTGR(FOR LITERAL FILE-LIMITS)
          3. SETS FWD.TYPE AS FOLLOWS:
             0:TWORD THAT NEEDS TO BE FIXED UP(FWD.DADDR=TWORD.OCUR)
             1:TINTGR FILE LIMIT CONVERTED TO 24 BITS BINARY

PROCEDURE ITS.A.SORTER VARYING;
          1. CHECKS FOR A SORTER DEVICE
               (CHECKS HDWR.INFO SUBSCRIPT)

PROCEDURE PARSE.FWD.REFS(REQUIRED.SIZE,ERRMSG,FWD.REF);
          1. PARSES FORWARD REFERENCE
          2. CHECKS TO SEE IF THE REFERENCE IS AN UNSIGNED,NUMERIC,
             ELEMENTARY,COMPUTATIONAL ITEM OF THE REQUIRED SIZE
          3. SAVES THE SEGMENT INDEX AND DISPLACEMENT AND MARKS THE
             TYPE AS A FIXED-UP TWORD

PROCEDURE CHECK.FWD.FILE.LIMIT(FILE.LIMIT);
          1. CHECKS CURRENT ITEM AGAINST FORWARD REFERENCE FILE LIMIT
             DATANAMES

PROCEDURE PARSE.SPECIAL.KEY VARYING;
          1. PARSES ACTUAL KEYS FOR REMOTE AND SORTER DEVICES

PROCEDURE CHECK.FORWARD.LIST;
          1. CHECKS CURRENT ITEM AGAINST THE FORWARD REFERENCE
             LIST BUILT IN THE ENVIRONMENT DIVISION(E.G. SELECT...
             ACTUAL KEY IS DATANAME...)

PROCEDURE PUT.DNINFO;
          1. CREATES THE DNINFO FILE
          2. CALLS ON FIND.DNINFO TO GET THE REQUIRED BLOCK

PROCEDURE MAK.DNINFO;
          1. MAKES A DNINFO ENTRY FOR EACH EXPLICIT DATANAME THAT IS
             NOT A FILLER

PROCEDURE ASSIGN.COPX;
          1. ASSIGNS THE COP INDEX

PROCEDURE DOOT VARYING;
          1. VERIFIES THE PROPER CLASS FOR JUST AND BWZ
             DECLARATIONS OF AN ELEMENTARY ITEM
          2. IGNORES BWZ CLAUSE IF IT IS DONE BY THE EDIT MASK
          3. RETURNS 2 TO SIGNAL EXIT

PROCEDURE IGNORE VARYING;
          1. RECOVERS TO A FIXED NAME AND RETURNS 2 (EXIT)
```

```
          2. RECOVERS TO A DATA CLAUSE AND RETURNS 0

PROCEDURE ERP.IGNORE(ERRMSG) VARYING;
          1. OUTPUTS THE ERROR MESSAGE
          2. RETURNS(IGNORE)

PROCEDURE GET.TINTGR(CANT.BE.ZERO) VARYING;
          1. RETURNS THE BINARY VALUE OF THE CURRENT INTEGER TOKEN
          2. GETS PAST THE LITERAL
          3. TRUNCATES TO 8 DIGITS(MOD 16,777,215)

PROCEDURE GET.TINTGR.RANGE VARYING;
          1. PARSES ... INTEGER-1[TO INTEGER-2] CLAUSE
          2. RETURNS INTEGER-1 OR INTEGER-2
          3. SETS FPB.VARIABLE AND VERIFIES THAT FPB.HDWR?TDISK

PROCEDURE DOCCURS VARYING;
          1. CHECKS THE LEVEL FOR 77 OR 01
          2. CHECKS FOR VARIABLE LENGTH SPECS

PROCEDURE FIGURATIVE.CONSTANT;
          1. ENCODES FIGURATIVE CONSTANTS
          2. CHANGES TYPE TO TINTGR/TNNLIT

PROCEDURE VALUE.ERROR VARYING;
          1. CHECKS  ADN TOKENS FOR LEGAL VALUE DECLARATION
          2. HANDLES (SIGNED) LITERALS,FIGURATIVE CONSTANTS,
             ALL<LITERAL> AND "DATE-COMPILED"
          3. BUILDS AN ENCODED TINTGR/TREAL/TNNLIT

PROCEDURE DVALUE VARYING;
          1. PROCESSES THE VALUE CLAUSE OF A DATA DESCRIPTION
          2. IGNORES IT AS DOCUMENTATION IF IN THE FILE SECTION
          3. SAVES THE VALUE IN LIT.INFO

PROCEDURE HARDWARE.CHECK VARYING;
          1. VERIFIES THAT THE DECLARED HARDWARE WILL ALLOW
             4 BIT DATA

PROCEDURE DINDEXED VARYING;
          1. PROCESSES INDEX NAMES AND WRITES THEM TO THE DNINFO FILE
          2. THE OCUR OF THE ASSOCIATED TABLE IS PUT INTO THE
             GRP.LENGTH OF THIS ENTRY
          3. REPORTS THE NUMBER OF INDEX NAMES PROCESSED.
          4. MAINTAINS THE OCUR OF THE FIRST AND LAST INDEX NAME FOR
             SUBSEQUENT ADDRESS ASSIGNMENT

PROCEDURE DATA.CLAUSES VARYING;
          1. PROCESSES THE CLAUSES OF A DATA DESCRIPTION
          2. VERIFIES THAT BZ,JUST,SYNC NOT DECLARED FOR GROUP ITEM

PROCEDURE GET.TXDN;
          1. GETS AN EXPLICIT DATANAME AND ITS DATA CLAUSES
          2. RETURNS FOR LEVEL=0, A "." OR A FIXED NAME
```

```
        3. SYNTAX CHECK LEVEL NUMBERS WITH SPECIAL HANDLING OF
           66,77 AND 88
        4. DELIVERS THE TXDN AND ITS ATTRIBUTES TO A HOLD AREA

PROCEDURE GET.REN.OPND;
        GETS A RENAMES OPERAND AND VERIFIES THAT:
        1. IT IS IN THE CURRENT RECORD
        2. THE "THRU" OPERAND IS NOT SUBORDINATE TO THE "FROM" OPERAND
        3. ITS LEVEL?00,01,66,77,OR 88
        4. ITS A SIMPLE VARIABLE

PROCEDURE RENAMES.ERP(ERRNO);
        1. OUTPUTS THE ERROR MESSAGE
        2. RECOVERS TO A TXDN, A FIXED NAME OR A "."
        3. GETS PAST THE "."

PROCEDURE RENAMES.THRU;
        1. SYNTAX CHECKS THE THRU PART OF A LEVEL 66 ENTRY
        2. BUILDS A GROUP DNINFO ENTRY

PROCEDURE LEVEL66;
        1. SYNTAX CHECKS A LEVEL 66 ENTRY
        2. BUILDS A DNINFO ENTRY

PROCEDURE RENAME;
        1. CONTROLS SYNTAX CHECKING OF LEVEL 66 ENTRIES
        2. PUTS THE ELEMENTARY OR GROUP ITEM TO THE DNINFO FILE

PROCEDURE MOD.2;
        1. ADJUSTS THE LOCATION COUNTER TO A 0 MOD 2 ADDRESS
        2. OUTPUTS A"FILLER ADDED" WARNING MESSAGE IF NECESSARY

PROCEDURE ALLOCATE(GRP.USAGE,GRP.LOCALS.PTR);
        1. THIS RECURSIVE PROCEDURE ALLOCATES MEMORY FOR LEVEL 77
           ENTRIES AND RECORD DESCRIPTIONS
        2. PARSES USAGE DECLARATIONS AND COPIES GROUP USAGE TO ITS
           SUBORDINATES
        3. PARSES ENTRIES(HAVING A SAME NAME) FOR ILLEGAL DUPLICATES
        4. FINDS THE DIGIT SIZE OF ELEMENTARY ITEMS AND ADJUSTS TO
           AN EVEN DIGIT ADDRESS AND SIZE FOR GROUP ITEMS
        5. PARSES REDEFINED OPERANDS OF OTHER THAN AN 01 ENTRY
        6. PARSES VALUE DECLARATIONS
        7. PARSES FORWARD REFERENCES DECLARED IN THE ENVIRONMENT
           DIVISION AND THE FILE SECTION
        8. GETS PAST LEVEL 88 ENTRIES WHICH HAVE ALREADY BEEN PARSED
        9. RECURSES FOR INCREASING LEVEL NUMBERS
        10. SAVES THE USAGES OF SUBORDINATE ENTRIES FOR SUBSEQUENT
            REPORTING TO THE GROUP ENTRY
        11. FINDS GROUP ITEM SIZES
        12. ALLOCATES SPACE FOR SUBSCRIPTED VARIABLES
        13. VERIFIES THE CORRECT SIZE FOR REDEFINING AREAS
        14. ITERATES FOR SAME LEVELS
        15. RETURNS FOR LEVELS LEQ 01
        16. NOTE:A VALUE STACK TEMPLATE AND STK.PTR INDEX IS USED TO
```

```
          REFERENCE LOCAL DATA IN ORDER TO SAVE NAME.STACK SPACE

PROCEDURE FIND.DIGIT.SIZE;
        1. FINDS THE DIGIT SIZE OF THE CURRENT DATA ITEM
        2. VERIFIES THAT NO DATA CLAUSES HAVE BEEN SPECIFIED FOR
           AN INDEX DATA ITEM
        3. ADJUSTS THE LOCATION COUNTER TO A CHARACTER ADDRESS FOR
           GROUP ITEMS
        4. VERIFIES THAT A PICTURE WAS SPECIFIED FOR ELEMENTARY ITEMS
        5. CHECKS FOR PROPER CLASS AND SIGN SPECS ON COMP ITEMS

PROCEDURE REDEFINED.OPERAND;
        1. VERIFIES THAT THE REDEFINED OCUR=LAST OCUR OR THE OCUR OF
           THE ORIGINALLY REDEFINED AREA
        2. VERIFIES THAT THE LEVELS ARE THE SAME
        3. CHECKS FOR SUBSCRIPTED OPERAND
        4. BUMPS A REDEFINES COUNTER
        5. RESETS THE LOCN COUNTER AND CHECKS FOR A GROUP ITEM ADDRESS
           NOT CHARACTER ADJUSTED

PROCEDURE VALUE.PARSE;
        1. PARSES THE VALUE DECLARATION OF THE WS.SECTION
        2. CHECKS FOR SUBSCRIPTED OR REDEFINED AREAS
        3. CHECKS FOR VALUE DECLARATION ON PRIOR GROUP ITEM
        4. SAVES THE DECLARED VALUE OF A GROUP ITEM
        5. CALLS MAK.VALUE FOR AN ELEMENTARY ITEM

PROCEDURE CHECK.FOR.DUPLICATES;
        1. PARSES THE FOLLOWING:
           01 A.
             02 B.
               03 C.
                 04 D ...
                 04 E ...
                 04 D ...  CANT BE QUALIFIED
                 04 B ...  APPEARS TO QUALIFY ITSELF
        2. THIS IS ACCOMPLISHED BY SAVING A POINTER TO PARENT DATA
           AS A LOCAL TO THE RECURSIVE PROCEDURE

PROCEDURE APPLY.GROUP.USAGE;
        1. COPIES DECLARED GROUP USAGE TO SUBORDINATES
        2. CHECKS FOR CONFLICTS

PROCEDURE CHECK.REDEFINED.SIZE;
        1. CHECKS REDEFINED SIZE AGAINST THE SIZE OF REDEFINED OPERAND
        2. SPECIAL CONSIDERATION IS GIVEN TO COMP ITEMS
        3. UPDATES RDFNS.COUNT

PROCEDURE CHECK.FORWARD.GROUP.ITEM;
        1. CHECKS LOCUR AGAINST THE FORWARD LIST
           LIST BUILT FOR ACTUAL KEY.FILE LIMITS
           AND VA OF ID. PRESENTLY ONLY CHECKS THE ACTUAL KEY

PROCEDURE FIXUP.GRP.SIZE;
```

```
          1. FINDS THE GROUP SIZE OF AN ITEM AND FIXES IT UP ON THE
             DNINFO FILE
          2. CHECKS FOR GROUP VALUE DECLARATION

PROCEDURE FDI.SEARCH(FD.OCUR,ADD.IT);
          1. SEARCHES THE FD.INFO LIST FOR A MATCHING FD OCUR
          2. ADDS THE FD.OCUR TO THE LIST UPON REQUEST
          3. WHEN A DUPLICATE SELECT CLAUSE IS SPECIFIED,A NEW LIST
             ELEMENT IS CREATED BUT ONLY THE FIRST SELECT WILL BE
             RETRIEVED ON LOOKUP
          4. IF A NEW LIST ENTRY IS TO BE MADE BUT THERE IS NO MORE
             ROOM, THE NEW LIST ELEMENT WILL OVERWRITE THE LAST
             LIST ELEMENT
          5. FDI.LIST.HEAD CONTAINS THE VALUE.STACK ADDRESS OF THE
             FIRST ENTRY OF THE LINK LIST. IF =0 THEN THE LIST IS EMPTY.
          6. IT IS IMPORTANT THAT FDI.NEXT BE THE FIRST FIELD OF
             FD.INFO. THIS ALLOWS FDI.LIST.HEAD TO BE HANDLED THE SAME
             AS FDI.NEXT. (SEE FD.INFO TEMPLATE)

PROCEDURE ITS.A.TAPE VARYING;
          1. CHECKS FOR A TAPE DEVICE
             (CHECKS HDWR.INFO SUBSCRIPT)

PROCEDURE ITS.A.DISK VARYING;
          1. CHECKS FOR A DISK DEVICE
             (CHECKS HDWR.INFO SUBSCRIPT)

PROCEDURE FILE.CONTROL;
          1. PARSES THE SELECT STATEMENTS OF THE FILE-CONTROL PARAGRAPH
          2. BUILDS AN FDINFO TABLE ENTRY IN SELECT(ED) ORDER SO THAT
             FD DECLARATION  CAN BE ADDED TO ITS CORRESPONDING SELECT

PROCEDURE IGNORE VARYING;
          1. RECOVERS TO A NON-ZERO CATAGORY
          2. RETURNS 0

PROCEDURE ERP.IGNORE(ERRMSG) VARYING;
          1. OUTPUTS THE ERROR MESSAGE
          2. RETURNS(IGNORE)

PROCEDURE ASSIGN VARYING;
          1. PARSES THE ASSIGN CLAUSE
          2. VERIFIES THAT HARDWARE NEQ SPO
          3. VERIFIES THAT SD FILES ARE ASSIGNED TO DISK

PROCEDURE RESERVE VARYING;
          1. PARSES RESERVE ... ALTERNATE AREAS

PROCEDURE MAK.FILE.LIMIT;
          1. PARSES A FILE-LIMIT OPERAND
          2. GETS SPACE AT THE END OF THE FDINFO ENTRY
          3. UPDATES FDI.PTR AND FILE.LIMIT.TABLE,BOUND

PROCEDURE FILE.LIMITS VARYING;
```

```
          1. PARSES FILE-LIMITS ARE...
          2. ADDS THE FILE-LIMIT PAIRS TO THE END OF THE FDINFO ENTRY
          3. A FILE-LIMIT=0 INDICATES THE END OF THE LIST
          4. PROCESSES "THRU END" AS @FFFFFE@

PROCEDURE ACCESS.MODE VARYING;
          1. PARSES ACCESS MODE IS ...
          2. INSISTS ON SEQUENTIAL FOR SD FILE

PROCEDURE ACTUAL.KEY VARYING;
          1. PARSES ACTUAL KEY IS DATANAME

PROCEDURE DOT VARYING;
          1. VERIFIES THAT A HARDWARE DEVICE WAS SPECIFIED
          2. VERIFIES THAT ACTUAL KEY WAS SPECIFIED FOR A RANDOM FILE
          3. IF FILE LIMITS ARE PRESENT FOR ANYTHING OTHER
             THAN A DISK FILE  FDI.FILE.LIMITS.FLG  IS RESET
             AND NO SPACE FOR THE FILE LIMITS IS RESERVED IN
             FILE.LIMIT.TABLE.BOUND
          4. RETURNS
            0: TKEY.CAT=SELECT
            2: OTHER

PROCEDURE FILCTL.CLAUSES VARYING;
          1. PARSES THE FILE CONTROL CLAUSES
          2. RETURNS
             0: RECOVER
             1: GET; RECOVER;
             2: RETURN

PROCEDURE ERP.SKIPIT(ERRMSG) VARYING;
          1. SKIPS A SELECT DECLARATION AFTER OUTPUTING THE ERROR
          2. RETURNS:
            0:IF STOPPED ON "SELECT"
            1:IF STOPPED ON A FIXED-NAME NEQ "FILE"

PROCEDURE SSELECT;
          1. PARSES THE"SELECT"DECLARATION OF THE FILE-CONTROL
          2. DELIVERS THE SELECT ATTRIBUTES TO THE FD.INFO TABLE
          3. RECOVERS FROM MISSING "."

PROCEDURE I.O.CONTROL;
          1. PARSES THE RERUN, SAME [RECORD] AREA AND
             MULTIPLE FILE CONTAINS DECLARATIONS OF THE
             I-O-CONTROL PARAGRAPH
          2. THE MULTIPLE FILE DECLARATION IMPLIES A "SAME AREA"
             DECLARATION FOR THE FILE-NAME LIST AND NEED NOT BE
             DECLARED SEPARATELY
          3. THE SAME AREA AND SAME RECORD AREA LISTS ARE MAINTAINED
             IN FD OCUR ORDER WITH THE LAST LIST ELEMENT CONTAINING
             THE MAXIMUM SIZES FOR ALL THE LIST MEMBERS

PROCEDURE PROPER.I.O.CONTROL.CLAUSE VARYING;
          1.RETURNS 1 IF THE CURRENT TOKEN IS AN O.K. I.O.CONTROL TOKEN
```

```
PROCEDURE IOC.IGNORE(ERRMSG);
        1. RECOVERS TO AN I-O-CONTROL CLAUSE AFTER OUTPUTING THE ERROR

PROCEDURE GOT.A.FILE.NAME VARYING;
        1. PARSES A FILE-NAME LIST AND SEARCHES FDINFO FOR THE FILE
        2. PUTS AN ERROR IF NO FILE-NAME AND FD.COUNT=0
        3. RETURN:
           0:NO FILE-NAME
           1:FILE-NAME FOUND

PROCEDURE MAK.SAME.AREA;
        1. BUILDS A LINK LIST OF FILE-NAMES APPEARING IN A
           SAME[RECORD]AREA FOR ... DECLARATION OR A
           MULTIPLE FILE TAPE ... DECLARATION
        2. ALL ELEMENTS OF A GIVEN FILE-NAME LIST POINT TO THE FDINFO
           ENTRY OF THE FILE-NAME IN THAT LIST THAT HAS THE LOWEST
           OCUR(LOW.FD.OCUR.PTR)
        3. REDUNDANT FILE-NAMES ARE ALLOWED
        4. IF A FILE-NAME APPEARS ON MORE THAN ONE LIST THE EFFECT IS
           THE SAME AS IF ALL THE FILES APPEARED ON THE SAME LIST
        5. THE HI.FD.OCUR ENTRY CONTAINS THE MAXIMUM RECORD WORK AREA
           SIZE OF ALL ITS LIST ELEMENTS

PROCEDURE RERUN.PARSE;
        1. PARSES ... RERUN ON TAPE EVERY INTEGER RECORDS OF FILE-NAME

PROCEDURE MULTIPLE.PARSE;
        1. PARSES ... MULTIPLE FILE TAPE "FILE-ID" CONTAINS
           FILE-NAME POSITION 1 ...

PROCEDURE SAME.PARSE;
        1. PARSES ... SAME [RECORD] AREA FOR FILE-NAME...

PROCEDURE APPLY.PARSE;
        PARSES ... APPLY  MICR (AND/OR) OCR  ON FILE-NAME...

PROCEDURE MONITOR.ON.DICT;
        1. PRINTS DDICT.REC[DDICT.PTR]

PROCEDURE GET.DDICT.SPACE;
        1. GETS SPACE FOR A DN DICTIONARY TABLE ENTRY FROM THE END
           OF THE DYNAMIC WORK AREA(DYN.WA) AND PROCEEDING TOWARD THE
           DYN.PTR
        2. IF NO ROOM THEN THE LAST GOOD SPACE IS REUSED
        3. UPDATES SEG.INX.COUNT AND SETS DDICT.PTR
        4. DDICT.SEG CONTAINS THE FOLLOWING FOR SEGMENT GATHERING:
           0: ALLOCATE TO SEGMENT 0(E.G. 77 ENTRIES OF WS-SECTION)
           1: FORCE TO A NEW SEGMENT(E.G. FILE-RECORD WORK AREAS)
           2: MAY GATHER TO A NEW SEGMENT(E.G. 01 S OF WS-SECTION)

PROCEDURE ALLOCATE.AN.01;
        1. ALLOCATES A RECORD DESCRIPTION OR A GROUP OF 77 ENTRIES
        2. FINDS THE MAXIMUM(LOCN,LOCN.MAX)
```

```
PROCEDURE FILE.SECTION;
        1. PARSES THE FILE DECLARATIONS OF THE FILE SECTION
        2. SELECTS THE LARGEST RECORD SIZE FOR STORAGE ALLOCATION
        3. CREATES FPB AND LABEL RECORD TOKENS ON THE SEGFILE
        4. UPDATES SAME AREA AND SAME RECORD AREA LISTS WITH MAXIMUM
           SIZES

PROCEDURE CHK.FFILE;
        1. PARSES THE FILE CONTAINS ... ATTRIBUTES FOR DISK FILES
        2. MAKES FPB.RCDS.AREA A MULTIPLE OF FPB.RCDS.BLOCK

PROCEDURE PROPER.FD.CLAUSE VARYING;
        1. RETURNS 1 IF CURRENT TOKEN IS AN O.K. FD TOKEN

PROCEDURE IGNORE VARYING;
        1: RECOVERS TO A PROPER.FD.CLAUSE OR FIXED NAME

PROCEDURE ERP.IGNORE(ERRMSG) VARYING;
        1. OUTPUTS THE ERROR MESSAGE
        2. RETURNS IGNORE

PROCEDURE FFILE VARYING;
        1. PARSES THE FILE CONTAINS ... CLAUSE OF AN FD

PROCEDURE RRECORD VARYING;
        1. PARSES THE RECORD CONTAINS ... CLAUSE OF AN FD
        2. IS FOR DOCUMENTATION ONLY WITH THE SIDE EFFECT OF
           FPB.VARIABLE!1 FOR AN INTEGER RANGE

PROCEDURE BBLOCK VARYING;
        1. PARSES THE BLOCK CONTAINS ... CLAUSE OF AN FD
        2. SAVES THE BLOCKING FACTOR IN FPB.RCDS.BLOCK IF "RECORDS"
           IS SPECIFIED, ELSE IN FPB.MAX.BLOCK.SIZE

PROCEDURE LABEL.RECORDS VARYING;
        1. PARSES THE LABEL RECORDS ... OF AN FD DECLARATION

PROCEDURE RECORDING.MODE VARYING;
        1. PARSES RECORDING MODE IS ... OF AN FD DECLARATION

PROCEDURE IID VARYING;
        1. PARSES THE VALUE OF ID IS ...

PROCEDURE FD.CLAUSES VARYING;
        1. PARSES THE FD CLAUSES
        2. RETURN
           0:RECOVERS:
           1:GET;RECOVER;
           2:RETURN

PROCEDURE GET.AN.FD;
        1. PARSES AN FD ... DECLARATION
        2. COMBINES SELECT ... DECLARATIONS TO BUILD THE FPB
```

```
PROCEDURE ASSIGN.SEG.INX;
        1. SETS THE FDI.SEG.INX TO THE SEG.INX OF ITS PREDICESSOR
           IN A SAME.AREA LIST OR ASSIGNS A NEW SEGMENT INDEX

PROCEDURE FIXUP.DN.DICT(SEGX,LGTH);
        1. FIXES UP DDICT.LENGTH(SEGX) WITH THE MAX(DDICT.LENGTH,LGTH)

PROCEDURE FD.01.PARSE;
        1. PARSES 01 ... OF A FILE
        2. CHECKS FOR DUPLICATE RECORD NAME
        3. CHECKS FOR REDEFINES ... OF THE FILE RECORD

PROCEDURE ITS.A.LABEL.RECORD;
        1. ASSIGNS DDICT[0] AS THE WORK AREA FOR THE RECORD
        2. KEEPS TRACK OF THE MAXIMUM RECORD SIZE

PROCEDURE ITS.A.RECORD;
        1. SAVES THE FIRST 01 OCUR
        2. KEEPS TRACK OF THE MAXIMUM RECORD SIZE OF THE FILE
        3. IF RECORDING MODE IS ASCII, FORCES USAGE IS ASCII

PROCEDURE FILE.RECORDS;
        1. PARSES FILE.RECORDS AND FILE.LABEL.RECORDS
        2. VERIFIES THAT FILE.01.OCUR?0

PROCEDURE FIXUP.FILE.SIZES;
        1. FIXES UP THE SIZES IN DDICT AND DNINFO
        2. FINDS FPB.BLOCK.SIZE AND FPB.RECORD.SIZE
        3. PUTS THE FPB AND FPB TRAILER TO THE SEGFILE

PROCEDURE PARSE.FILE.DESCRIPTION;
        1. PARSES A FILE DESCRIPTION AND ITS FILE.RECORDS
        2. HANDLES ERROR RECOVERY FOR A RECORD DESCRIPTION THAT HAS
           NO FPB ASSOCIATED WITH IT

PROCEDURE WS.SECTION;
        1. PARSES THE WORKING-STORAGE SECTION
        2. SELECTS THE LARGEST REDEFINED RECORD FOR STORAGE ALLOCATION
        3. PAYS SPECIAL ATTENTION TO LEVEL 77 ENTRIES

PROCEDURE MAK.DN.DICT;
        1. UPDATES THE DATA DICTIONARY ENTRY FOR A DATA SEGMENT
           CANDIDATE AND GETS SPACE FOR THE NEXT ONE

PROCEDURE CHECK.01.REDEFINED;
        1. CHECKS FOR REDEFINED 01 OF WORKING-STORAGE SECTION
        2. INHIBITS REDEFINITION OF AN 01 FILLER
        3. UPDATES SEGMENT INDEX FOR RECORDS WITH NO REDEFINES CLAUSE
        4. VERIFIES THAT THE REDEFINED OCUR=LAST RECORD OCUR OR
           THE OCUR OF A PRIOR RECORD OF A CHAIN
        5.  RESETS THE LOCATION COUNTER AND SETS RDFNS.COUNT WHEN
           NECESSARY

PROCEDURE DATSYN.WINDUP;
```

```
          1. GETS FPB ADDRESS OF THE MONITOR FILE IF NECESSARY
          2. FIXES UP FILE-LIMITS AND ACTUAL KEY ADDRESSES IN THE FPB
          3. FIXES UP FPB.WA, AND DNINFO ADDRESSES FOR FPBS ON THE
             SAME [RECORD] AREA LISTS

PROCEDURE FILL.HDWR.TABLE(P1,P2,P3,P4);
          1. INITIALIZES THE HARDWARE TABLE

PROCEDURE FILL.ADN.TABLE(P1,P2);
          1. INITIALIZED THE ALLDN FILE CONTROL TABLE

PROCEDURE DATSYN.SETUP;
          1. BUILDS PC.INFO TABLE AT THE BEGINNING OF THE
             DYNAMIC WORK AREA
          2. INITIALIZES THE ALLDN FILE CONTROL TABLE AS FOLLOWS:
           ADN.TSIZE BIT(7)  TOKEN SIZE FOR UNBLOCKING
           ADN.CASEV BIT(4)  CASE STATEMENT VARIABLE
          3. READS THE FIRST BLOCK OF THE ALLDN FILE
          4. OPENS THE DNINFO FILE
          5. OPENS THE SEG FILE
```

EXPLODE..ADDITIONAL INFO

```
          I N I T I A L     E X P L O D E     O U T P U T
  IF MONITOR THEN
    MONITOR
    [DEPENDING]
    TFILEREC
    TDOT
  IF FILE LIMITS THEN
    FILE.LIMITS (RESERVED WORD)
    TINTGR FOR FILE.LIMIT.TABLE.BOUND..URS=BINARY LEN=3 LIT=BINARY BOUND
    FOR EACH FILE IN FDI.TABLE THAT HAS FILE LIMITS:
      TFILEREC
      FROM LIMIT (TWORD PC 9(8) CMP) OR  (TINTGR URS=BINARY LEN=3)
      THRU LIMIT (TWORD PC 9(8) CMP) OR  (TINTGR URS=BINARY LEN=3)
    TDOT
    PROCEDURE DIVISION
  IF DECLARATIVES THEN
    DECLARATIVES

PROCEDURE  BUILD.SPECIAL.REG;
                DIGIT   DSEGO
    REGISTER    LENGTH  ADR     DATATYPE
    ..........  ......  .....   ............
    SW1         1       0       4BIT UNSIGNED
     .
     .
    SW8         1       7       4BIT UNSIGNED
    TALLY       5       8       4BIT UNSIGNED
    DATE        5       13      4BIT UNSIGNED (JULIAN YY DDD)
    TIME        7       18      4BIT UNSIGNED (HHMMSST T=TENTH)
    TODAYS-DATE 6       25      4BIT UNSIGNED (MMDDYY)
    TODAYS-NAME 18      31      8BIT ALPHA

PROCEDURE CREATE.TWORD(OCCUR); FORMAL OCCUR VARYING;
    FOR USE BY FILE.LIMITS & FORWARD REFERENCES OF:
    1. VALUE OF ID IS DATANAME. 2. ACTUAL KEY IS DATANAME.

PROCEDURE USE.SYNTAX;
    TO SYNTAX USE STATEMENTS:
    1.VERIFY FORMAT
    2.CHECK FOR DUPLICATES ASSIGNED TO A FILE
    3.VERIFY HARDWARE
    4.VERIFY LABEL RECORDS CLAUSE PRESENT FOR A FILE
    5.PLACE IN THE FPB TAILER THE EXPLICIT LABEL OCURS OF
      THE USE RTNES FOR EACH FILE

PROCEDURE MAK.DNINFO.TABLE;
          1. READS AS MANY DNINFO ENTRIES INTO THE DNI.TABLE AS WILL
            FIT AND UPDATES DNI.TABLE.OCUR.MAX
```

PROSYN PROCEDURE DETAILS

```
ROUTINE PUT.FROM.INPUT;
  MOVES TOKENS FROM ALLOFILE BUFFER TO CODEFILE BUFFER

PROCEDURE PUT.OPERAND(A); FORMAL A VARYING;
    OPERAND=LIT,TINDEX,TWORD,TCSPL,TCSPR,OR TFILEREC

PROCEDURE FILL.OP.STACK VARYING;
  MOVES NEXT TOKEN INTO NEXT AVAILABLE OPERAND STACK POSITION

PROCEDURE GET.IT VARYING;
  BREAKS OUT NEXT TOKEN.TYPE & MOVES TOKEN TO OPERAND.STACK IF
  APPLICABLE. IF SUBSCRIPTED TWORD TOKEN, SUBSCRIPT
  TOKENS ARE PUT INTO ADJACENT OPERAND.STACK ENTRIES. 12/70 KH

PROCEDURE HOMOGENEOUS.LIT(A,C) VARYING; FORMAL (A,C) VARYING;
    CHECKS A LIT TO SEE IF COMPRISED ENTIRELY OF SAME CHARACTER.

PROCEDURE READ.WRITE.SYNTAX VARYING;
    TO  SYNTAX  READ  WRITE RELEASE RETURN  SEEK
    ADVERB.CNTRL OF  FORMAT
   (1)  IS KEYWORD VALID FOR THE VERB
   (3)  PRECEDENCE OF THE KEYWORD
   (3)  WHAT IS EXPECTED TO FOLLOW
    NEED  TAKES ON 1 OF 6 VALUES DEPENDING ON THE ITEM RECEIVED ON
          INPUT
    0 -  A TWORD MUST FOLLOW
    1 -  A TWORD OR INTEGER MUST FOLLOW
    2 -  A TWORD,INTEGGR OR APPROPRIATE KEYWORD MUST FOLLOW
    3 -  A RESERVED WORD-VERB OR APPROPRIATE ADVERB MUST FOLLOW
    4 -  A VERB MUST FOLLOW
    5 -  AN APPROPRIATE KEYWORD MUST FOLLOW

    A- OPERAND STK PTR  TO TFILEREC
    B- OPERAND STK PTR  TO FROM/INTO DATA NAME
    C- OPERAND STK PTR  TO WRITE SPACING INTEGER OR DATA NAME
    F- OPERAND STK PTR  TO FILE INFO

PROCEDURE WHAT.GOT FIXED;
   BASED ON TYPE AND KEY
   PROVIDE A CASE VARIABLE VALUE  OUTER IOCONTROL
   0 - TWORD
   1 - TINTGR
   2 - KEYWORD
   3 - VERB
   4 - EVERYTHING ELSE

PROCEDURE SORT.VERB;
    OUTPUT TO GENRAT FOR SORT
    TRESWD  = SORT
```

```
      URS(1)     INPUT PROCEDURE PRESENT
    TPERF      OF INPUT PROC IF PRESENT
    TFILEREC   OF  SORT FILE
      WITH URS(2) = 0  DS/DP (END)
                  = 1  PURGE
                  = 2  RUN

    SERIES OF
      TRESWD  = ASCENDING/DESCENDING
     SERIES OF
      TWORD   KEYS
    TSTATESTOP
    ONE OF
    TFILEREC  = USING FILE
      WITH URS(2) 0= LOCK
                  1= RELEASE
                  2= PURGE
      OR
    TPERF      OF   INPUT PROC. (DUPLICATION)

    ONE OF
    TFILEREC  = GIVING
      WITH URS(2) 0=LOCK
                  1=RELEASE      OR
    TPERF      OF   OUTPUT PROC

PROCEDURE OPEN.CLOSE VARYING;
  VALID.ADVERBS TABLE ENTRIES
 0        NO              OPEN  CLOSE
 1        PURGE                 CLOSE
 2        REEL                  CLOSE
 3        REVERSED        OPEN
 4        REWIND          OPEN  CLOSE
 5        ACCESS          OPEN
 6        RELEASE               CLOSE
 7        LOCK            OPEN  CLOSE
 8 WAS REMOVE OUT ON 02/01/73
 9        PUNCH           OPEN
 10       PRINT           OPEN
 11       INTERPRET       OPEN
 12       STACKER         OPEN
 13       HERE                  POSSIBLY
 14       PACK                  POSSIBLY
 15       CODEFILE        OPEN  CLOSE
 16       CRUNCH                CLOSE
 17       ROLLOUT               CLOSE

PROCEDURE SET.VERB;
    VALID SET COMBINATIONS
       (TO)
    OBJ  I SUBJ
    ---    ----
    DN   I IN
    IDI  I IDI
```

```
    IDI  | IN
    IN   | LIT2
    IN   | DN
    IN   | IDI
    IN   | IN

    (UP/DOWN BY)
    OBJ  | SUBJ
    ---    ----
    IN   | LIT1
    IN   | DN

    DN   = DATA NAME = ELEMENTARY INTEGER
    IDI  = INDEX DATA ITEM
    IN   = INDEX NAME
    LIT1= INTEGER LITERAL = NON-ZERO, LEQ 6 DIGITS
    LIT2= UNSIGNED LIT1 (CAN BE +)

  REARRANGING DONE FOR CONVENIENCE OF CODEGEN

    SET,VERB, SUBJ, OBJ, [OBJ, ...], TSTATESTOP

    FOR BOTH FORMATS THE SUBJECT IS REPOSITIONED AHEAD OF THE OBJECT(S),
    "TO", "UP BY" & "DOWN BY" ARE DROPPED, AND TSTATESTOP IS ADDED

    NOTE = FORMAT ENCODED IN URS OF SET VERB ON OUTPUT
   ----------------------
   ,OPTION,SERIES,UNUSED,
   , (2)  , (1)  , (5)  ,
   ----------------------
    00 DOWN BY
    01 UP BY
    10 TO

    APPROACH: 1. LOOK AHEAD TO "TO" ETC, COUNTING OBJECT OPERANDS, DOING
                 GROSS SYNTAX CHECKING ON THE WAY
              2. SYNTAX CHECK THE SUBJECT OPERAND
              3. SET APPLICABLE OPTIONS IN URS, OUTPUT VERB, OUTPUT SUBJ
              4. DETAIL SYNTAX CHECK & OUTPUT EACH OBJ
              5. RECOVER TO NEXT VERB/PAR

PROCEDURE PERFORM,VERB;
  REARRANGING DONE FOR CONVENIENCE OF CODEGEN
    FORMAT 1 (VANILLA) = NONE
    FORMAT 2 (TIMES)   = DROP "TIMES"
    FORMAT 3 (UNTIL)   = DROP "UNTIL", CONDITION SUPPLIES BOOSTOPS, THEN
    FORMAT 4 (VARYING) = DROP "VARYING","FROM","BY","AFTER",
              CONDITIONS HANDLED AS IN 3. INSERT COPIES OF OPERANDS OF
    "VARYING" & "FROM" CLAUSES FOLLOWING TPERF = FOR INITIALIZING CODE.

    NOTE = FORMAT ENCODED IN URS OF TPERF ON OUTPUT
    ---------------
    OPTION  VARYING
     (2)    COUNT(2)  #BITS
```

```
 0-3      1-3     VALUES
 ---------------

NO TSTATESTOPS REQUIRED

VALID PERFORM VARYING COMBINATIONS
    "FROM" CLAUSE   (VARYING OBJ FROM SUBJ)
OBJ | SUBJ
---   ----
DN  | LIT
DN  | DN
*DN | IN
 IN | DN
*IN | IDI
 IN | IN
*IN | LIT

     NOTE - IDI | IN  IS NOT PERMITTED SINCE IDI COULD NOT THEN BE
            LEGALLY INCREMENTED IN THE 'BY' PHRASE, ACCORDING TO
            'SET' RULES.
    "BY" CLAUSE     (VARYING OBJ ... BY SUBJ)
OBJ | SUBJ
---   ----
DN  | LIT
DN  | DN
*IN | LIT
      3. USED TO GET AUTOMATIC OUTPUT FROM CERTAIN INPUT TOKENS.
*IN | DN

 DN  - DATA NAME - ELEMENTARY NUMERIC
*IN  - INDEX NAME (REQUIRES BOTH OPERANDS SATISFY SET VERB RULES)
 IDI - INDEX DATA ITEM
 LIT - NUMERIC LITERAL

PROCEDURE EXAMINE,VERB;
                  *** INPUT ***                    *** OUTPUT ***
        ---------------          ---------------          ---------------
  CASE  I EXAMINE       I  CASE  I EXAMINE       I  CASE  I EXAMINE      I
        I TWORD         I        I TWORD         I        I TWORD        I
        I TALLYING      I        I REPLACING     I    1   I TWORD-TINTGRI
    1   I ALL,LEADING   I    3   I ALL,FIRST,    I  2 + 3 I TWORD-TINTGRI
        I UNTIL FIRST   I        I LEADING,      I
        I TWORD-TINTGR  I        I UNTIL FIRST   I
                                 I TWORD-TINTGR  I
        I REPLACING     I        I BY            I
    2   I BY            I        I TWORD-TINTGR  I
        I TWORD-TINTGR  I
   IN ADDITION TO TWORD-TINTGR ABOVE A FIGURATIVE CONSTANT MAY BE USED
```

CODEGEN PROCEDURE DETAILS

REFERENCES TO FPB TRAILER OCCUR IN
1. DATSYN;FIXUP.FILE.SIZES
2. EXPLODE;USER ROUTINES
3. CODEGEN;IOVERBS ASSOCIATED WITH FILES
4. FIXUP;BUILD FPBS

PROCEDURE SEE.TOKEN.OUT;
1. CALLED ONLY BY PUT.TOKEN.
2. CALLED ONLY IF TRACE.CODEGEN.
3. MONITORS ON LINE PRINTER TOKENS IN CODEBUFFER (A MISNOMER), THE OUTPUT BUFFER FOR THE SEGFILE.
4. USED FOR DEBUGGING.

PROCEDURE SEE.TOKEN.IN;
1. CALLED BY ANYONE.
2. CALLED ONLY IF TRACE.CODEGEN.
3. MONITORS ON LINE PRINTER TOKENS IN OPERAND STACK THAT ARE IN INPUT FORMAT.
4. USED FOR DEBUGGING.

PROCEDURE FORCE.SEGFILE(T);
1. FINISHES OFF THE CURRENT SEGFILE BUFFER WITH TEOF OR TEOB.
2. SETS UP FOR NEW BUFFER.

PROCEDURE PUT.FROM.INPUT;
1. MOVES TOKENS FROM ALLONFILE BUFFER TO CODEFILE BUFFER.
2. CALLED BY GET.IT.

ROUTINE FILL.OPERAND.STACK;
1. CALLED BY GET.IT.
2. MOVES INPUT TOKEN TO OP.STACK[AOS].

ROUTINE GET.IT;
1. GETS NEXT INPUT TOKEN.
2. MOVES TOKEN TO OP.STACK IF APPLICABLE.

PROCEDURE GET.OPERAND;
1. CALLS GET.IT TO PLACE TOKEN IN STACK.
2. IF TOKEN IS A TWORD AND SUBSCRIPTED THEN MOVE SUBSCRIPT TOKENS INTO ADJACENT OP.STACKS ENTRIES.

PROCEDURE GET.TRASH(L,SCL,S,F,A);
1. L LENGTH IN UNITS.
2. SCL SCALE.
3. S SIGN (W,S,J,K).
4. F FORMAT. 1 FOR 8BIT, 0 FOR 4BIT.
5. A ASCII. 1 FOR ASCII, 0 FOR EBCDIC.
6. BUILDS TWORD TOKEN IN OP.STACK[AOS].
7. TOKEN HAS INLINE COP FOR NEXT AVAILABLE TRASH AREA.

```
          8. CALLER MUST EITHER SAVE AOS BEFORE CALL OR USE LOI AFTER.

PROCEDURE MAKE.TWORD (      WRELADR,L,SCL,S,F,A);
    LIKE GET.TRASH & GET.C.POOL BUT YOU TELL IT THE ADDRESS
    IF A=1 THEN ASCII ELSE EBCDIC, IF F=1 THEN 8 BIT ELSE 4 BIT,
    BUILDS TWORD IN NEXT AVAIL OPERAND STACK LOCATION.
    CALLER MUST SAVE AVAIL.OP.STACK BEFORE CALLING THIS PROCEDURE.

PROCEDURE FIND.POWER(N);
          1. FINDS THE NUMBER OF BITS NECESSARY TO HOLD A GIVEN NUMBER.

PROCEDURE CHANGESIGN(A);
          1. EMITS CODE TO CHANGE SIGN OF AN OPERAND.

PROCEDURE BUILD.INT (NEG,LIT      ); FORMAL (NEG,LIT) VARYING;
    PUTS TINTGR IN OP.STACK[AOS]
    IF NEG = 1 MAKES IT A MINUS LIT
    LIT MUST BE A HEX BIT STRING I.E.  73 = @(4)73@,

PROCEDURE FIXUP.N(N,NSEG,NDISP); FORMAL (N,NSEG,NDISP) VARYING;
    FIXES UP "N" FOR MULTIPLE MOVES & COMPARES

PROCEDURE RELATIVE.INDEX(I) VARYING;FORMAL.VALUE I VARYING;
     DETERMINES IF AN INDEXED TWORD HAS RELATIVE INDEXING,
         E.G.      A ( I, J+1, K-3 )
    EXPECTING EITHER TINDEX OR TINDEX FOLLOWED BY SIGNED TINTGR - -
    - - UP TO 3 INTERMIXED SINGLES OR PAIRS

PROCEDURE REL.INDEX.CODE (I);  FORMAL.VALUE I VARYING;
                                                              KOMP
     EMITS CODE TO RESOLVE RELATIVE INDEXES AND COMPRESSES OP.STACK

PROCEDURE MOVESIGN(A,B);
          1. CALLED BY MOVEGEN FOR NUMERIC TO NUMERIC MOVES.
          2. SENDING AND/OR RECEIVING IS J OR K SIGN.
          3. SENDING MAY BE A LITERAL.

PROCEDURE PREPARE.LIT(K,A,S);
          1. K  1 IS K-SIGN, 0 IS OTHER.
          2. A  1 IS ASCII,  0 IS EBCDIC.
          3. S  1 IS MINUS,  0 IS PLUS.
          4. CREATES A LITERAL OF PLUS OR MINUS AND PUTS LIT TOKEN
             INDEX IN L.

PROCEDURE SCALELIT(A,B); FORMAL (A,B) VARYING;
    A=OPERAND.STACK ADDRESS OF INTEGER OR REAL LIT TOKEN.
    B=OP.STACK ADR OF TWORD OR INTEGER OR REAL LIT TOKEN.

PROCEDURE REALLY.SCALELIT(A,F.LGTH,F.SCALE,B.SIGNED);
          1. A IS INDEX OF LIT TOKEN.
          2. SCALES A LITERAL ACCORDING TO F.SCALE. WILL RIGHT TRUNCATE
             OR ZERO FILL SCALE IF NECESSARY.
          3. LEFT TRUNCATES THE LIT IF NECESSARY.
          4. CALLER MUST EITHER SAVE AOS BEFORE CALL OR USE LOI AFTER.
```

```
          5. ORIGINAL TOKEN MAY BE 4 OR 8 BIT.
          6. FINAL TOKEN FORMAT IS 4 BIT.
          7. IF LIT IS SIGNED AND SIGN IS 8BIT MINUS OR 4BIT @D@ THEN
             SIGN IS PRESERVED ELSE DROPPED.

PROCEDURE HOMOGENEOUS.LIT(A,C) VARYING;
    CHECKS A LIT TO SEE IF    COMPRISED ENTIRELY OF SAME CHARACTER.

PROCEDURE E.LIT.TO.A.LIT(E);
    TRANSLATES EBCDIC LITERAL INTO ASCII LITERAL IN NEXT AVAIL OPERAND.

PROCEDURE DO.EDIT(MOI) VARYING; FORMAL.VALUE MOI VARYING;
  PROCEDURE MOVE.DIGITS;
    SINCE REPEAT OPERATOR HANDLES UP TO 16 DIGITS AT A TIME
    THIS PROCEDURE WILL HANDLE THOSE DATA ITEMS EXCEEDING 16 DIGITS
  END MOVE.DIGITS;
  PROCEDURE BUILD.MASK;
    DETERMINES THE MASK FOR AN EDTE
    PLACES THE MASK IN THE OPERAND STACK BEGINNING AT AOS
    IF A DATA ITEM USES ALL POSSIBLE EDIT SYMBOLS IT WOULD APPEAR AS:
       -67890043.67952
    THE MASK STRING IS DETERMINED BY THE ALGORITHM:
    INM(MIN1) CAT MVD(LOGICALSIZE-SCALE-1) CAT INU(DPT) CAT MVD(SCALE-1)
  END BUILD.MASK;
   ..........................................................................
    IF DATA ITEM IS A J OR K SIGN THE FOLLOWING STEPS ARE TAKEN
      1. GET.TRASH IS CALLED  - TO RESERVE DEST SPACE FOR A MOVE
      2. MOVEGEN   -  WILL CHANGE SIGN TO S.SIGN
    MONITOR WILL DETERMINE IF AN EDIT IS NECESSARY
    IF SO IT WILL:
      1. CALL GET.TRASH   -  TO RESERVE DEST SPACE FOR AN EDTE
      2. DETERMINE THE MASK FOR THE EDTE
END DO.EDIT;

PROCEDURE MONITER(MONITER.OPERAND.INDEX);
     MONITOR CODING FOR DATA NAMES OR INDEXES
     INPUT: ALLDNFILE FROM PROCEDURE SYNTAX CHECK
     OUTPUT: SEGFILE  -- OPERATORS USED: CAT N, COPX1,OPND1,....,OPNDN
                                         EDTE OPND1,COPX1,MASK
     EXAMPLES OF MONITOR AT EXECUTION:  DATAVALUE = -XXXXXX.....XXXXX
                                        GOPHER/HOLE (5,6,9) = XXXX.XXX
     TOKEN FROM ALLDNFILE:
      TYPE URS COPX BASIC COP FACTORS PCINFO DNFLG JUST BWZ M.ADR M.LGTH
      (6)  (4) (12)   (53)     (80)    (74)  (10)  (1)  (1) (24)  (6)

              * * * OPERAND   STACK * * *
      BEFORE  GET.TRASH       AFTER  GET.TRASH
      *-------------* | MOI   *-------------* | MOI
      *             *         *             *
      *  TWORD      *         *   TWORD     *
      *             *         *             *
      *------------ | AOS     *------------ | LOI
      *             *         *             *
      *             *         *   TWORD     *
```

```
      *------------*          *------------- | AOS
      *            *          *              *
      *            *          *              *
     MONITOR CODING FOR PARAGRAPHS AND SECTIONS
     INPUT: ALLONFILE FROM PROCEDURE SYNTAX CHECK
     OUTPUT: SEGFILE  -- OPERATORS USED: CAT N,COPX1,OPND1,....,OPNDN
                                         INC1 COPX
     EXAMPLE OF MONITOR AT EXECUTION:  PARAGIRAFFE (XXXXXX)

     TOKEN FROM ALLONFILE:
      TYPE URS LABEL FLAGS  SEG #
      (6)  (4)     (5)       (7)

     FOLLOWED BY TNNLIT:
      TYPE URS SCALE LENGTH LITERAL
      (6)  (4)  (8)   (8)    (30 MAX)

PROCEDURE SZ.ERR.2(DEST); FORMAL DEST VARYING;
    USED FOR ACCUMULATING SIZE ERROR INFO BY SETTING OVERFLOW TOGGLE FOR
    "ADD A TO B, B1, B2...BN ON SE S1."
    SE2 IS TURNED OUT FOR ADD A TO B...ADD A TO BN-1
    SE3 IS TURNED OUT FOR ADD A TO BN.
PROCEDURE CHECK.REMAIN;
    /* ORIGINALLY THE REMAINDER WAS TAKEN DIRECTLY FROM THE DIVIDEND
    FIELD AFTER THE DIVIDE FOR THE QUOTIENT (CAUSE THATS WHERE THE
    MACHINE LEAVES IT). HOWEVER, IF THE QUOTIENT IS TO BE ROUNDED, THE
    REMAINDER CAN CHANGE SINCE 1 EXTRA DIGIT OF ACCURACY IS REQUIRED.
    EG :
    DIVIDE 16 INTO 247.2 GIVING PC99V9          = 15.4       REMAIN = .8
    DIVIDE 16 INTO 247.2 GIVING PC99V9 ROUNDED = 15.45(+.5) REMAIN =  0
    THEREFORE WE NOW DO 2 DIVIDES IF THE QUOTIENT IS TO BE ROUNDED.
    ALSO, ALA B3500 COBOL :
    "THE REMAINDER IS CARRIED TO THE SAME DEGREE OF ACCURACY AS DEFINED
    IN THE PICTURE OF THE QUOTIENT AND ALL EXTRA POSITIONS ARE FILLED
    WITH ZEROS."                                                    */

PROCEDURE SET.TRASH(A) VARYING; FORMAL A VARYING;
  SIRED BY ARITH.EXP OUT OF GAS...TAKES CARE OF NUTTY COBOL PC OF 99PP &
  PP99...GETS CORRECT SIZE AND CREATES 4 BIT S.SIGN TOKEN.

PROCEDURE ARITH.EXPRESSION VARYING;
  RETURNS OP.STACK INDEX OF OPERAND (WHICH MAY BE A TRASH AREA FOR
  INTERMEDIATE RESULTS). ARITH EXPS ARE DELIMITED BY A RESEVED WORD OR
  TROUSTOP. THE BEGINNING OPERAND FOR ARITH.EXP TO LOOK AT IS IN
  LAST.OPERAND.INDEX (SET BY FILL.OPERAND.STACK).

PROCEDURE  IO.EMIT.FIX.MVN (LA,LB);
  LA =OPND STACK PTR TO MSG TLITFIX LITERAL
  LB =OPND STACK PTR TO MSG AREA

PROCEDURE BUILD.4BIT.LIT(ITEM,LIT.TOK);
  FORMATS A 4BIT TINTGR (UNSIGNED UNLESS PASSED A MINUS FIXED ITEM) IN
   OPERAND STACK, POINTED TO BY PASSED LIT.TOK
  MAXIMUM OF 6 DIGITS - USED PRIMARILY BY INDEX-NAME HANDLERS
```

```
PROCEDURE IO.FILE.INFO(REC.NUM,STK.PTR,RW);
    RW    0= READ FILE
          1= WRITE FILE
    REC.NUM= FPB NUMBER
    STK.PTR= PLACE IN STACK FOR INFO-NOT POINTER

PROCEDURE IOVERBS;
    PROCEDURES  FOR IO  STATEMENTS WITH FILE REFERENCES
    OPEN,CLOSE,READ,WRITE,SEEK,SELECT,CONTROL, SORT
    RETURN,RELEASE

PROCEDURE  IO.BUILD.PRINT;
   NOTE  IOVERBS GLOBAL  A  IS PTR TO FILE REC
                         B  IS PTR TO TWORD OR TINTGR
         THAT SPECIFIES  SPACING VALUE -PROSYN PROVIDES A DEFAULT
         OF  BEFORE 1 LINE - ALSO ALL LITS ARE CONVERTED TO THEIR
         24 BIT BINARY EQUIVALENT IN PROSYN

PROCEDURE  WHAT.IS.SPACING BIT(4);
    DETERMINE SPACING  VALUE FOR COMM TO PRINTER

PROCEDURE IO.SET.ID;
   GEN  CODE  TO  SET  FPB  FILE ID
   CODE EMITTED
    MVN COMM.LIT,COMM.MSG.AREA
    CUMM(ACCESS.FPB),COMM.MSG.AREA       READ FPB
    MVA  PACK.ID,FPB.PACK.ID
    MVA  FILE.NAME,FPB.AREA
    MVN COMM.LIT,COMM.MSG.AREA
    COMM(ACCESS.FPB),COMM.MSG.AREA      WRITE FPB

PROCEDURE GEN.FILE.LIMIT.RTNES;
    TO GENERATE FILE LIMIT SAVE AND CHECK RINES
    A "FILE LIMIT" VERB IS OUTPUT FROM DATSYN FOLLOWED BY AN INTEGER
    ARRAY BOUND AND  TFILEREC AND ALL FILE LIMITS
    FOR  EACH FILE
    TWO RTNES ARE EMITTED AS NEEDED- 1 FOR SEQUENTIAL FILES AND
    1 FOR RANDOM - TO CHECK THE FILE LIMITS
    THERE IS A GENERAL FILE LIMIT WORK AREA RESERVED AND
    1 SPECIFIC AREA FOR EACH FILE WITH FILE LIMITS
    WHEN A LIMIT IS TO BE CHECKED THE FILE AREA IS MOVED TO
    THE GENERAL AREA
    INCLUDED IN THE AREA ARE
    1.    A 6 DIGIT FPB NUMBER
          USED ON SEQ FILES FOR POSITIONING
    2.    A  6 DIGIT FILE LIMIT INDEX INTO AN ARRAY OF ALL LIMITS
    3.    A  6 DIGIT RECORD COUNT/ACTUAL KEY
    4.    A  1 DIGIT OPEN FLAG
             0= NOT OPEN
    FOR EACH FILE A RTNE IS EMITTED TO "CAPTURE" LIMITS AT OPEN

    THE AREA SET ASIDE FOR EACH FILE DEPENDS ON ACCESS:
    RANDOM :
   (6)  FILE LIMIT INDEX
```

```
   (6)  RECORD #
   (1)  OPEN BOOLEAN

    SEQUENTIAL:
   (6)  FILE #
   (6)  FILE LIMIT INDEX
   (6)  RECORD #
    FILE LIMITS STATEMENT CREATED FROM FILES WITH FILE LIMITS
    THE ORDER OF THE STMT IS:
 (1)VERB 5022
 (2)TINTGR LEN=3,SCL=0,FOR=8 - BINARY AMOUNT OF SPACE NEEDED FOR THE
     ARRAY (A LIMIT IS A FROM OR THRU VALUE)- #OF DIGITS
 (3)N  COMBINATIONS OF
       TFILEREC OF FILE AND
       M COMBINATIONS OF
          TWORD/TINTGR   FROM LIMIT
          TWORD/TINTGR   THRU LIMIT
          (TINTGR CONVERTED TO BINARY VALUE-LEN=3,FOR=8)
 (4) TDOT

PROCEDURE  CHNG.TO.INDEXED;
    TO CHANGE FL.ARRAY  FROM A NON SUBSCRIPTED  ITEM
      TO AN INDEXED ITEM WITH PROPER FLAGS AND FACTORS

PROCEDURE EMIT.RAN.CK.RTN;
  TO EMIT CODE TO CHECK FILE LIMITS ON A RANDOM FILE
    CODE  GENERATED IS :
    OFL(0)
    ZRO  OPEN.FLAG,=,L2
 L1 CPN  END-LIST,FROM [INDEX],NEQ,L3
    ZRO  REC#,EQL,L2
    OFL(1)
 L2 XIT
 L3 CPM L4,REC#,LSS,FROM[INDEX],GTR,THRU[INDEX],0
    XIT
 L4 INC 12, INDEX
    BUN  L1

PROCEDURE EMIT.SEQ.CK.RTN;
    TO GENERATE THE CODE FOR CHECKING SEQUENTIAL FILE LIMITS

    SEQUENTIAL FILE LIMIT CHECK ROUTINE CODE
    CPN  REC#,FL.ARY[INDEX],LEQ +L3     REC# TO THRU LIMIT
    INC  FL.ARY.ENTRY.LEN, INDEX         THRU-BUMP TO NEXT FROM
    CPN  END.LIST.LIT,FL.ARY[INDEX],NEQ, +L1      EOL TO FROM
    CAT 2, FILE.MSG.AREA,VERB,FPB#,ADVERB --POSN TO EOF
    MVN  END.LIST.LIT, REC#
     COMM
    XIT

 L1 MVN FL.ARRAY[INDEX],REC#   REC# OF ARY TO FILE LIMIT WORK AREA
    CAT  3,FILE.MSG.AREA,VERB,FPB#,ADVERB,REC# OF FL WORK AREA
    INC  FL.ARY.ENTRY.LEN,INDEX
```

```
 L2 COMM FILE.MSG.AREA
 L3 XIT

PROCEDURE IO.SORT;
    OUTPUT FROM SORT:
    MVN  OPEN.LIT, MSG.AREA     FIRST 5 INSTRUCTIONS PRESENT ONLY IF
    COMM MSG.AREA               INPUT PROCEDURE SPECIFIED. OPEN AND
    ALTER DADDR,ACON            PRESENT IF PROC INDEPENDENT OVERLAY
    NTR  INPUT.PROCEDURE        CLOSE IS FOR SORT FILE
    MVN  CLOSE.LIT,MSG.AREA
    COMM MSG.AREA
    MVN  LIT, KEY.TABLE         SORT KEY DESC. TO TRASH
    MVN  LIT, SORT.INFO.TABLE   OTHER SORT INFO TO TRASH
    MVN  TLITFIX,MSG.AREA       SORT COMM. LIT  (TO SORT KEY INFO)
    MVN  TLITFIX,MSG.AREA       SORT.KEY DESC TRASH ADDR,IN AND OUT FILES
    COMM MSG.AREA

    MVN  OPEN.LIT,MSG.AREA      LAST 5 INSTRUCTIONS PRESENT ONLY IF
    COMM MSG.AREA               OUTPUT PROCEDURE SPECIFIED
    ALTER DADDR,ACON            PRESENT IF PROC INDEPENDENT OVERLAY
    NTR  OUTPUT PROCEDURE
    MVN  CLOSE.LIT,MSG.AREA
    COMM MSG.AREA
PROCEDURE TWORD.TINDEX(W,I) VARYING;
                                        USED BY INDEX HANDLER      KOMP
    APPROACH - SUBTRACT 1 FROM W GIVING T1
               MULTIPLY T1 BY I'S FACTOR GIVING T2
               W NOW POINTS TO T2, I IS UNDISTURBED
PROCEDURE TINDEX.TINDEX(I,J) VARYING;
                                        USED BY INDEX HANDLER      KOMP
    APPROACH - DIVIDE J BY ITS FACTOR GIVING T1
               MULTIPLY T1 BY I'S FACTOR GIVING T2
               I NOW POINTS TO T2, J IS UNDISTURBED
         ALSO OPTIMIZES CASES WHERE I'S FACTOR IS A MULTIPLE OF
         J'S FACTOR, AVOIDING DIVIDE CODE

PROCEDURE DECLARATIVEHOUSEKEEPING;
   DECLARATIVES CAUSES PARITY AND LABEL USE RTN. ANALYZERS
   TO BE EMITTED FOR ANY FILE THAT
   1. DOES NOT HAVE A SPECIFIC USE RTN ASSIGNED
   2. FOR PARITY - ONLY HARDWARE DEVICES THAT RETURN PARITY
   3. FOR LABEL  - TAPE ONLY
   THE ANALYZER DETERMINES DYNAMICALLY HOW THE FILE WAS OPENED
   AND CAUSES THE APPROPRIATE USE RTN TO BE EXECUTED

PROCEDURE EXAMINE;
   THIS PROCEDURE WILL BE THE ONE CALLED IN THE EXAMINE IF SET
   ON THE DOLLAR CARD.  REQUIRES THE EXAMINE S-OP IN THE INTERPRETER

         ************   TYPES  OF  EXAMINE   ************
                    --  EXAMINE DATANAME  --
    TALLYING   (ALL)          (LITERAL-1)   [REPLACING BY LITERAL-2]
               (LEADING)      (DATANAME-1)                DATANAME-2]
               (UNTIL FIRST)
```

```
    REPLACING  (ALL)          (LITERAL-3)   BY  (LITERAL-4)
               (LEADING)      (DATANAME-3)      (DATANAME-4)
               (FIRST)
               (UNTIL FIRST)

PROCEDURE EXAMINE.BY.COMPARE;
   THIS PROCEDURE WILL BE THE ONE CALLED IF THE EXAMINE IS NOT SET
   ON THE DOLLAR CARD.  DOES NOT REQUIRE THE EXAM S-OP IN INTE-PRETER

PROCEDURE SUBSCRIPTED.FIELD;
    CALCULATES THE BEGINNING ADR IF A SUBSCRIPTED ITEM IS BEING EXAMINED
    MULTIPLYS THE ITEM:S FACTOR TIMES THE VALUE IN THE SUBSCRIPT AT
    EXECUTION TIME.  THIS RESULT MINUS THE FACTOR IS THE BEGINNING ADR

PROCEDURE FIRST.OPTION;
   HANDLES THE EXAMINE A REPLACING "FIRST" B

PROCEDURE LABEL.PROCESSOR;
    FUNCTIONS - GENERAL
      1. FIXUP LABEL TABLE FOR THIS OCUR (WHICH IS MAINTAINED BY THE
                                          GLOBAL COUNTER - EXPLICIT.LABEL
      2. EMIT TERMINAL-PARAGRAPH-OF-PERFORM-RANGE EXIT CODE IF REQUIRED
      3. IF EXPLICIT-GO-TO-PARAGRAPH EMIT GOPAR OR BUN CODE
      4. CALL MONITOR CODE GENERATOR IF REQUIRED
      5. CHANGE TXPAR/TXSECN TO TXLABEL
  - - - - - - - - WHEN CHANGING SEGMENTS
      6. EMIT "FALL-THRU" CODE (BUN
      7. FIXUP PSEUDO-CODE DICTIONARY & TSEGLINK
      8. EMIT TSTOP,  WRITE CODEBUFFER ONTO SEGFILE, EMIT NEW TSEGLINK
  - - - - - - - - MISC
      9. EMIT "SYSTEM" EXIT FOR TERMINAL PARAGRAPH OF A USE SECTION
     10. SET FIRST.EXECUTABLE FOR FIRST EXPLICIT LABEL

  EXAMPLES OF REQUIRED CODE FOR CONTROL-TRANSFERRING OPERATIONS
  FIXED SECTION 0.                        * OVERLAY SECTION 50.
  P1.                                     *       ALTR(I4,I6)
                                          *       ALTR(I6,+G)
                                          *       GOPAR(0)
       PERFORM I2.                        * I1.
          ALTR(I0,I2)                     *     ALTER I4 TO I5.
          PERF(K1,OVERLAY)                *       ALTR(I4,I6)
                                          *
       GO TO I3.                          *     ALTER I6 TO I7.
          ALTR(I0,I3)                     *       ALTR(I6,I8)
          BUN(OVERLAY)                    * I2.
                                          *       - - -
  P2. GO TO P4.                           *         PXIT(K1)
          GOPAR(P2).                      * I3.
                                          *       - - -
  P3. GO TO.                              * I4. GO TO I6.
          GOPAR(P3)     P3 INITIALLY      *         GOPAR(I4)
       .F COMM(ABEND)    POINTS TO +F     * I5.
  P4. ALTER P3 TO I3.                     *       - - -
          ALTR(P3, +A)                    * I6. GO TO.
```

```
      BUN(+B       NEXT SEQL SOURCE*          GOPAR(I6)      INITIALLY
  .A  ALTR(I0,I3)                  *      .G COMM(ABEND)  POINTS TO +G
      BUN(OVERLAY)                 * I7.
  GO TO P1, P4, I7 DEPENDING ON X.*        LAST SOURCE STATEMENT
  .B GOTO(COPX,3,+D,P1,P4,+C)      *       IN THIS SECTION
  .C ALTR(I0,I7)                   *      - - -
     BUN(OVERLAY)                  *         BUN(+E)  NEXT SEQL SOURCE
  .D              NEXT SEQL SOURCE *
 FOR EXAMPLES OF REQUIRED CODE SEE LABEL.PROCESSOR

PROCEDURE GOTODEP (A); FORMAL A VARYING;
      SPECIAL STACK ENTRY - INTERNAL TO GOTODEP ONLY - NEEDED TO EMIT

PROCEDURE PERFORM;
 VALID PERFORM VARYING COMBINATIONS & IMPLIED OPERATIONS
 NOTE - WHERE INDEXING IS USED, SET.GEN IS CALLED BY PERFORM
       *VARYING SUBJ FROM OBJ*   EQUIV   *SET SUBJ TO OBJ
 OBJ   SUBJ
 ---   ----
 DN    LIT      MOVE
 DN    DN       MOVE     * IN THESE CASES, IF ANY AFTER CLAUSES,
 DN    IN       SET.TO   * ALL "FROM" SUBJ-S MUST BE SAVED INITIALLY
 IN    DN       SET.TO   * IN CONSTANT POOL.
 IN    IDI      MOVE     * NOT REQUIRED FOR LITERAL SUBJ S OR
 IN    IN       SET.TO   * WHERE ONLY SINGLE VARYING CLAUSE.
 IN    LIT      SET.TO

       *VARYING SUBJ . . . BY OBJ* EQUIV  *SET SUBJ UP BY OBJ*
 OBJ   SUBJ
 ---   ----
 DN    LIT      ADD(LIT,DN);
 DN    DN       ADD(DN,DN);
 IN    LIT      SET.GEN(UP.BY,LIT,IN);
 IN    DN       SET.GEN(UP.BY,DN,IN);

 DN  -DATA NAME
 IN  -INDEX NAME
 IDI -INDEX DATA ITEM

PROCEDURE SET.GEN;
 VALID SET COMBINATIONS & IMPLIED OPERATIONS
    (TO)
 OBJ   SUBJ
 ---   ----
 DN    IN       DIV(F,IN,DN); INC1(DN);
 IDI   IDI      MOVE
 IDI   IN       MOVE
 IN    LIT      PRE-DEC1 LIT; PRE-MULT BY F; MVN(NEWLIT,IN);
 IN    DN       SUB(1,DN,T); MULT(F,T,IN);
 IN    IDI      MOVE
 IN    IN       MOVE   *(FACTORS EQUAL)
 IN    INN      DIV(FF,INN,T); MULT(F,T,IN);
  (UP/DOWN BY)
 OBJ   SUBJ
```

```
    ---    ----
    IN     LIT      PRE-MULT LIT BY F; INC/DEC(NEWLIT,IN);
    IN     DN       MULT(F,DN,T); INC/DEC/(T,IN);

    DN  -DATA NAME
    IDI -INDEX DATA ITEM
    IN  -INDEX NAME
    INN -INDEX NAME HAVING DEFFERENT FACTOR
    LIT -INTEGER LITERAL

PROCEDURE SEND.RECEIVE;
    A - MESSAGE
    B - FROM/INTO IDENTIFIER
    C - QUEUE NAME
    SEND ID-1 [FROM ID-2] TO ID-3/LITERAL
         ON INVALID-REQUEST S1
         ON Q-FULL S2
    RECEIVE ID-1 [INTO ID-2] FROM ID-3/LITERAL
         ON INVALID-REQUEST S1
         ON Q-EMPTY S2
         MOVE ID-2 TO ID-1          IF FROM
         MOVE COMM.LIT,MSG AREA
         MOVE ID-3, TRASH           IF NOT IN SEG 0 OR IS LITERAL
         MAKP ID-3/TRASH,MSG AREA
         MAKP ID-2,MSG AREA         IF NOT IN SEG 0
         COMM
         MOVE ID-1,ID-2             IF RECEIVE INTO
         FETCH                      PRESENT IF CONDITION OPTIONS
         CPN 1,MSG,NEQ,+A
         S1
    .A   CPN 2,MSG,NEQ,+B
         S2
    .B

PROCEDURE FIXUP.CAT.LEN ;
  TO UPDATE THE LENGTH OF THE RECEIVING FIELD ON A CAT
  THE LOCATION GIVEN IS THAT OF N IN THE INST. AND
              THE RECEIVING FIELD HAS AN IN LINE  COP

PROCEDURE ALTERED.LIT FIXED;
   TO SET LENGTH OF LIT TO PROPER SIZE,CHANGE TO TNNLIT
   AND RETURN THE LENGTH
   LIT IS IN 8 BIT FORMAT

PROCEDURE ALTERED.WORD FIXED;
   TO SET LOGICALIIZE , ZERO COPX IF NECESSARY , CHANGE REAL TO INTEGR
    AND RETURN LOGICALSIZE

PROCEDURE X.QUALIFIES(I) VARYING;
  TO DETERMINE WHETHER  AN OPERAND QUALIFIES AS AN OPND OF  A CAT

PROCEDURE  IO.ADD.TO.CAT;
   TO CONTINUE CAT
```

```
PROCEDURE OPTIMAL.DISP  FIXED;
   TO DETERMINE WHETHER AN INTERMEDIATE TRASH AREA REQ ON DISPLAY

PROCEDURE IO.BUILD.DISP.COMM;
   TO BUILD A DISPLAY COMMUNICATE
   A  IS A PTR TO THE TOKEN OF THE  ITEM TO BE DISPLAYED

PROCEDURE SEP.ITEM;
   TO DEVELOPE A MOVE OF A SINGLE ITEM TO BE DISPLAYED
   INTO THE APROPRIATE TRASH AREA

PROCEDURE DM.VERBS;
     GENERATE CODE TO EFFECT THE MCP COMMUNICATE CORRESPONDING TO
      A DATA MANAGEMENT VERB.  ENTER ON VERB TRESWD.

PROCEDURE DO.SWITCH6;
    INDICATES MONITOR DEPENDING SPECIFIED.
    WILL CHECK TO SEE IF SWITCH 6 = ZERO OR IF IT'S GREATER.

PROCEDURE MONITOR.INITIAL;
    FOLLOWING PROCEDURE SETS UP THE WRITE SUBROUTINE FOR MONITOR
    CHECKS TO SEE IF MONITOR DEPENDING HAS BEEN SPECIFIED
    SETS UP THE LITERAL THAT IS MOVED TO THE RECORD MSG AREA
```

DATA MANAGEMENT

THE DM ADDITIONS TO THE COBOL COMPILER WERE COMPLETED IN OCTOBER, 1973. THE DEVELOPMENT WAS IN 4 STAGES:

1. UNDERSTANDING WHAT DM IS AND HOW TO USE IT.
2. DOCUMENTATION OF THE NECESSARY COMPILER CHANGES.
3. IMPLEMENTATION.
4. TESTING.

THE FOLLOWING IS AN EXPLANATION OF THE FLOW OF DATA MANAGEMENT CONSTRUCTS THROUGH THE COMPILER. THIS IS THE DOCUMENT THAT WAS USED FOR STEP 2.

DATA-BASE SECTION DECLARATIONS

DATASET

```
<LEVEL-NUMBER> <DATASET-NAME> DATASET DDL-NUMBER <DDL #>
  [RETRIEVAL/ORDERING KEY <KEY-NAME> DDL-NUMBER <DDL #>
  (<COMPONENT-NAME-1> [<COMPONENT-NAME-2>] ...)] ... .
```

SUBSET

```
<LEVEL-NUMBER> <SUBSET-NAME> SUBSET DDL-NUMBER <DDL #>
  TO <DATASET-NAME> DDL-NUMBER <DDL #>
  [(<COMPONENT-NAME-1> [<COMPONENT-NAME-2>] ...)].
```

FORMAT OF THE DDL-NUMBER

THE <DDL#> CONSISTS OF A STRUCTURE NUMBER
FOLLOWED BY THE TIME AND DATE WHEN THE DATASET WAS CREATED.
THE FORMAT FOR THE DDL-NUMBER IS :

```
SSS HH:MM:SS MM/DD/YY
```

THIS STRING OF CHARACTERS IS ENCODED INTO A
A 64 BIT PATH DICTIONARY ENTRY BY MERGE

```
01 PATH.DICT.FORMAT            BIT(64),
  02 LOCK.BIT                  BIT(1),
  02 MEDIA.BIT                 BIT(1),
  02 BEEN.OPENED               BIT(1),
  02 EXPLICIT.OPEN             BIT(1),
  02 STR.NUMBER                BIT(12),
  02 DM.TIME                   BIT(17),
    03 DM.HOUR                 BIT(5),
    03 DM.MIN                  BIT(6),
    03 DM.SEC                  BIT(6),
  02 DM.DATE                   BIT(16),
    03 DM.MONTH                BIT(4),
    03 DM.DAY                  BIT(5),
    03 DM.YEAR                 BIT(7);
```

DMSTATUS REGISTER

THE DMSTATUS REGISTER IS 6 DIGITS LONG
AND WILL BE SET BY EACH DM COMMUNICATE.
A VALUE OF ZERO INDICATES NO EXCEPTION.

```
PARSE- ADD "DMSTATUS" TO THE RESERVE WORD LISTS.
       TRANSLATE THE SYNTAX DMSTATUS (<EXCEPTION NAME>) INTO
       (DMSTATUS = <LITERAL>). <EXCEPTION NAME>'S ARE RESERVED WORDS
```

ONLY IN THIS CONTEXT. EACH CORRESPONDS TO AN EXCEPTION NUMBER,
<LITERAL>.

MERGE= ASSIGN MEMORY IN DSEG0

EXPLODE= EXPAND REFERENCES TO LOOK LIKE A TWORD
AND SET CONDITIONS NAME FLAG SO DMSTATUS WILL ALWAYS BE A
FULL RELATION IN PROSYN. THIS ALSO PREVENTS DMSTATUS FROM
BEING USED OUTSIDE A CONDITION.

PARSE
-----

1. WHEN DATA-BASE SECTION IS DECLARED
   - VERIFY THAT IT FOLLOWS THE FILE SECTION AND PRECEEDS
     THE WORKING-STORAGE SECTION
   - OUTPUT A DUMMY TXDN WITH LEVEL=0(LIKE WS-SECTION)
     AND SYMBOL="DATA-BASE"
   - SAVE THIS DB.SECTION OCUR AS A GLOBAL
   - DATABASE.SECTION.FLAG:=1
   - VERIFY THE DATA-BASE SECTION TO CONSIST ONLY OF
     "01 DS-NAME INVOKE ..." ENTRIES AND "DB ... "ENTRIES

2. WHEN DB IS DECLARED
   - SAVE THE DATABASE NAME
   - OUTPUT THE DATABASE NAME AS A TNNLIT
   - OUTPUT A TXDN

3. WHEN INVOKE IS DECLARED
   - VERIFY DATABASE.SECTION.FLAG=1 AND LEVEL=01
   - OPEN #DATA-BASE/DATASET.NAME(LIKE A COPY FROM LIBRARY)
     WHERE DATA-BASE IS THE FIRST 9 CHARACTERS OF THE DATABASE-NAME
   - SCAN PAST THE 01 DATASET-NAME OF THE SOURCE IMAGE
     AND SUBSTITUTE THE INVOKING DATASET-NAME
   - MARK THE SCANNED SOURCE IMAGES WITH "*"

4. WHEN DATASET IS DECLARED
   - VERIFY DATABASE.SECTION.FLAG=1
   - PARSE THE DDL-NUMBER
   - ASSIGN AN OCUR TO EACH KEY-NAME WHERE
     LEVEL:=IF ORDERING THEN 51 ELSE 52
   - MARK COMPONENT-NAMES AS TWORD WITH "DATA-BASE" AS A TQUAL

5. WHEN SUBSET IS DECLARED
   - VERIFY DATABASE.SECTION.FLAG=1
   - PARSE THE DDL-NUMBER
   - LEVEL:=53
   - DELETE THE TARGET DATASET-NAME
   - PARSE THE TARGET DDL-NUMBER
   - OUTPUT THE TARGET COMPONENT-NAME AS TWORD WITH
     "DATA-BASE" AS A TQUAL

6. WHEN END OF DATA-BASE SECTION IS FOUND
   DATABASE.SECTION.FLAG:=0

7. WHEN FIND/MODIFY IS SPECIFIED
   OUTPUT THE COMPONENT-NAMES OF THE SELECTION
   EXPRESSION WITH "DATA-BASE" AS TQUAL

DICTIONARY PROCESSING

1. NO CHANGE

QUALIFICATION RESOLUTION

1. ITEMS WITH LEVEL OF 51, 52, OR 53 HAVE
   SCOPE ONLY OF THEMSELVES

2. THE DB.SECTION OCUR INCLUDES ALL
   DATABASE ITEMS IN ITS SCOPE

3. INHIBIT ERROR REPORTING FOR A TWORD
   WITH THE DB.SECTION OCUR AS ITS QUALIFIER
   - IF UNIDENTIFIED NAME THEN OCUR:=0
   - IF INSUFFICIENT QUALIFICATION THEN
     OCUR:=LAST OF THE DUPLICATES

MERGE

NOTES:
- NO CONDITION NAMES WILL BE CONTAINED IN A DATASET DESCRIPTION
  THEREFORE THE BUILDING OF THE
  CONDITION-NAME TABLE DOES NOT CONFLICT
  WITH THE BUILDING OF THE PATH INFO TABLE

DATA.BASE SECTION PROCESSING

1. FOR EACH 01 DATASET-NAME INVOKE.NO:=BUMP INVOKE.COUNT

2. BASED ON THE STRUCTURE.NUMBER OF THE DDL-NUMBER
   ASSIGN A UNIQUE PATH.NO AND BUILD A CORRESPONDING
   PATH.DICTIONARY.ENTRY IN THE PATH.DICTIONARY BLOCK
   ON THE SEGFILE WHICH CONTAINS THE UNIQUE DDL-NUMBER
   (START THE PATH.NO AT 1)

3. BUILD A TABLE RELATING THE DATASET OCUR TO ITS
   KEY-NAMES AND THEIR COMPONENT-NAMES
   AND CONTAINING THE INVOKE.NO AND PATH.NO AND THE TARGET PATH.NO
   NOTE: IF THE TARGET DATASET OF A SUBSET IS NOT INVOKED
       THEN IGNORE THAT PATH AND ITS COMPONENTS WHEN BUILDING THE TABLE
       E.G ITS COMPONENT OCUR=0

4. DROP ALL ATTRIBUTES(E.G. DDL-NUMBER,
   COMPONENT-NAMES, ETC.) WHEN PARSING
   PATH DECLARATIONS

5. FOR EACH DB, SAVE DATA BASE NAME (TNNLIT) IN A TABLE
   WITH ITS OCUR. DROP TNNLIT.

   EXAMPLE 1

```
 02 STUDENTS DATASET DDL-NUMBER 38
    ORDERING KEY STUDNAMES DDL-NUMBER 71
      (LASTNAME, FIRSTNAME).
```

ALL	DN	ADNFILE
TXDN	TXDN(STUDENTS,LEVEL=02)	TXDN(STUDENTS,LEVEL=02)
DATASET		DATASET
DDL-NUMBER		
TINTGR(38)		
TXDN	TXDN(STUDNAMES,LEVEL=51)	TXDN(STUDNAMES,LEVEL=51)

```
DDL-NUMBER
TINTGR(71)
(
TWORD          TWORD(LASTNAME)
TWORD          TWORD(FIRSTNAME)
)
.
```

EXAMPLE 2

```
 02 COURSE SUBSET DDL-NUMBER 8
    TO UNIV-COURSES DDL-NUMBER 14
      (DEPARTMENT, LEVEL).
```

ALL	DN	ADNFILE
TXDN DDL-NUMBER TINTGR(8) TO DDL-NUMBER TINTGR(14) (TWORD TWORD) .	TXDN(COURSE,LEVEL=53) TWORD(DEPARTMENT) TWORD(LEVEL)	TXDN(COURSE,LEVEL=53)

PROCEDURE DIVISION PROCESSING

GENERAL

1. PARSE THE DM.VERBS WITH PARTICULAR ATTENTION TO THE RELATIONSHIP OF DATASET-NAMES AND THEIR PATHS

2. OUTPUT THE INVOKE.NO AND PATH.NO AS TINTGR TOKENS FOLLOWING REFERENCES TO PATH NAMES

3. THE CT.ADVERB BITS FOR THE DM.VERBS ARE ARRANGED AS FOLLOWS:

BIT	MEANING
1	DISTINGUISHES INSERT FROM STORE OR REMOVE FROM DELETE.
2	RECREATE

```
3       DMSTATUS FORMAT
          0:BINARY
          1:DECIMAL 4 BIT
4       "ON EXCEPTION" SPECIFIED
5       UPDATE
6       MODIFY
7-11    SELECTION EXPRESSION TYPE
        0:NEXT
        1:PRIOR
        2:FIRST
        3:LAST
        4:NEXT AND "AT" CLAUSE
        5:CURRENT
        6:"AT" CLAUSE
12      PATH TYPE
        0:KEY
        1:DATASET

NOTE: NO SYNTAX CHECKING WILL BE DONE
  BY MERGE FOR THE EXCEPTION CLAUSE.
  IF THE CLAUSE IS SPECIFIED IT WILL BE
  OUTPUT AS IS
```

OPEN

```
    OPEN UPDATE <DATABASE-NAME>
      [ON EXCEPTION <STATEMENT> [ELSE <STATEMENT>]]

    CT.VERB=6
    CT.OBJECT=NOT USED
    CT.ADVERB=
      BIT 3  = DMSTATUS FORMAT
        0:BINARY
        1:DECIMAL 4 BIT
      BIT 4  ="ON EXCEPTION"
      BIT 5  ="UPDATE"
    CT.1=BIT LENGTH OF DM-STATUS REGISTER
    CT.2=BASE RELATIVE ADDRESS OF DM-STATUS REGISTER
    CT.3=BASE RELATIVE ADDRESS OF DATA BASE NAME
    CT.4=LENGTH OF DATA BASE NAME IN BITS

  MERGE OUTPUT
  ------------

  TRESWD  OPEN.DM
  TNNLIT  DATABASE NAME
  TRESWD  UPDATE
          <EXCEPTION CLAUSE>
```

1. VERIFY THAT IT IS A DATABASE-NAME

2. CHANGE OPEN TO DM.OPEN BEFORE OUTPUT(NEW VERB)

3. LOOK UP DATABASE NAME IN TABLE BY OCUR, OUTPUT THE NAME

CLOSE

```
    CLOSE <DATABASE-NAME>
      [ON EXCEPTION <STATEMENT> [ELSE <STATEMENT>]]

    CT.VERB=7
    CT.OBJECT=NOT USED
    CT.ADVERB=
      BIT 3  =DMSTATUS FORMAT
        0:BINARY
        1:DECIMAL 4 BIT
      BIT 4  ="ON EXCEPTION"
    CT.1=BIT LENGTH OF DM-STATUS REGISTER
    CT.2=BASE RELATIVE ADDRESS OF DM-STATUS REGISTER
    CT.3=BASE RELATIVE ADDRESS OF DATA BASE NAME
    CT.4=LENGTH OF DATA BASE NAME IN BITS

  MERGE OUTPUT
  ------------

  TRESWD  CLOSE.DM
  TNNLIT  DATA BASE NAME
          <EXCEPTION CLAUSE>

1. VERIFY THAT IT IS A DATABASE-NAME

2. CHANGE CLOSE TO DM.CLOSE BEFORE OUTPUT(NEW VERB)

3. LOOK UP DATA BASE NAME IN TABLE BY OCUR, OUTPUT THE NAME
```

```
CREATE/RECREATE
---------------
    CREATE/RECREATE <DATASET-NAME>
      [ON EXCEPTION <STATEMENT> [ELSE <STATEMENT>]]

    CT.VERB=18
    CT.OBJECT=INVOKE.NO, PATH.NO
    CT.ADVERB=
      BIT 3  =DMSTATUS FORMAT
        0:BINARY
        1:DECIMAL 4 BIT
      BIT 4  ="ON EXCEPTION"
      BIT 8  ="RECREATE"
    CT.1=BIT LENGTH OF DM-STATUS REGISTER
    CT.2=BASE RELATIVE ADDRESS OF DM-STATUS REGISTER
    CT.3=BIT LENGTH OF THE DATASET RECORD WORK AREA
    CT.4=BASE RELATIVE ADDRESS OF THE DATASET RECORD WORK AREA

  MERGE OUTPUT
  ------------

  TRESWD  CREATE/RECREATE
  TINTGR  INVOKE.NUM OF DATASET-NAME
  TINTGR  PATH.NUM OF DATASET-NAME
  TWORD   DATASET-NAME
          <EXCEPTION CLAUSE>

1. VERIFY THE DATASET-NAME

2. OUTPUT THE DATASET-NAME AND PATH INFO
```

STORE

```
    STORE <DATASET-NAME>
      [ON EXCEPTION <STATEMENT> [ELSE <STATEMENT>]]

    CT.VERB=16
    CT.OBJECT=INVOKE.NO, PATH.NO
    CT.ADVERB=
      BIT 3  =DMSTATUS FORMAT
        0:BINARY
        1:DECIMAL 4 BIT
      BIT 4  ="ON EXCEPTION"
    CT.1=BIT LENGTH OF DM-STATUS REGISTER
    CT.2=BASE RELATIVE ADDRESS OF DM-STATUS REGISTER
    CT.3=BIT LENGTH OF THE DATASET RECORD WORK AREA
    CT.4=BASE RELATIVE ADDRESS OF THE DATASET RECORD WORK AREA

  MERGE OUTPUT
  ------------

  TRESWD  STORE
  TINTGR  INVOKE.NUM OF DATASET-NAME
  TINTGR  PATH.NUM OF DATASET-NAME
  TWORD   DATASET-NAME
          <EXCEPTION CLAUSE>

1. VERIFY THE DATASET-NAME

2. OUTPUT THE DATASET-NAME AND PATH INFO
```

DELETE

```
    DELETE <DATASET-NAME>
      [ON EXCEPTION <STATEMENT> [ELSE <STATEMENT>]]

    CT.VERB=17
    CT.OBJECT=INVOKE.NO, PATH.NO
    CT.ADVERB=
      BIT 3  =DMSTATUS FORMAT
        0:BINARY
        1:DECIMAL 4 BIT
      BIT 4  ="ON EXCEPTION"
    CT.1=BIT LENGTH OF DM-STATUS REGISTER
    CT.2=BASE RELATIVE ADDRESS OF DM-STATUS REGISTER
    CT.3=BIT LENGTH OF THE DATASET RECORD WORK AREA
    CT.4=BASE RELATIVE ADDRESS OF THE DATASET RECORD WORK AREA

  MERGE OUTPUT
  ------------

  TRESWD  DELETE
  TINTGR  INVOKE.NUM OF DATASET-NAME
  TINTGR  PATH.NUM OF DATASET-NAME
  TWORD   DATASET-NAME
          <EXCEPTION CLAUSE>

1. VERIFY THE DATASET-NAME

2. OUTPUT THE DATASET-NAME AND PATH INFO
```

FREE

```
    FREE <DATASET-NAME>
      [ON EXCEPTION <STATEMENT> [ELSE <STATEMENT>]]

    CT.VERB=21
    CT.OBJECT=INVOKE.NO, PATH.NO
    CT.ADVERB=
      BIT 3  =DMSTATUS FORMAT
        0:BINARY
        1:DECIMAL 4 BIT
      BIT 4  ="ON EXCEPTION"
    CT.1=BIT LENGTH OF DM-STATUS REGISTER
    CT.2=BASE RELATIVE ADDRESS OF DM-STATUS REGISTER

  MERGE OUTPUT
  ------------

  TRESWD  FREE
  TINTGR  INVOKE.NUM OF DATASET-NAME
  TINTGR  PATH.NUM OF DATASET-NAME
  TWORD   DATASET-NAME
          <EXCEPTION CLAUSE>

1. VERIFY THE DATASET-NAME

2. OUTPUT THE DATASET-NAME AND PATH INFO
```

INSERT/REMOVE

```
    INSERT <DATASET-NAME> INTO <SUBSET-NAME>
      [ON EXCEPTION <STATEMENT> [ELSE <STATEMENT>]]

    REMOVE CURRENT FROM <SUBSET-NAME>
      [ON EXCEPTION <STATEMENT> [ELSE <STATEMENT>]]

    CT.VERB=16 FOR INSERT, 17 FOR REMOVE
    CT.OBJECT=INVOKE.NO, PATH.NO OF SUBSET-NAME
    CT.ADVERB=
      BIT 1   =ON
      BIT 3   =DMSTATUS FORMAT
        0:BINARY
        1:DECIMAL 4 BIT
      BIT 4   ="ON EXCEPTION"
    CT.1=BIT LENGTH OF DM-STATUS REGISTER
    CT.2=BASE RELATIVE ADDRESS OF DM-STATUS REGISTER
    CT.3=INVOKE.NO, PATH.NO OF DATASET-NAME

  MERGE OUTPUT
  ------------
  TRESWD  INSERT/REMOVE
  TINTGR  INVOKE.NUM OF DATASET-NAME
  TINTGR  PATH.NUM OF DATASET-NAME
  TWORD   DATASET-NAME OR TRESWD CURRENT
  TINTGR  INVOKE.NUM OF SUBSET-NAME
  TINTGR  PATH.NUM OF SUBSET-NAME
  TWORD   SUBSET-NAME
          <EXCEPTION CLAUSE>

1. VERIFY THE DATASET-NAME TO BE A
   PROPER TARGET DATASET OF THE SUBSET-NAME
   AND OUTPUT IT WITH ITS PATH INFO

2. OUTPUT THE TARGET DATASET-NAME AND ITS PATH INFO

3. CT.VERBS ARE LIKE THOSE FOR STORE AND DELETE.
   THEY ARE DISTINGUISHED BY BIT 1 OF CT.VERB.

4. FOR REMOVE, INVOKE NUM AND PATH NUM OF DATASET ARE ZERO.
```

FIND/MODIFY

CURRENT PATH FORMAT

```
FIND/MODIFY [<DATASET-NAME> VIA]
  <DATASET-NAME>/<SUBSET-NAME>/<ORDER-NAME>/<RETRIEVAL-NAME>
  [ON EXCEPTION <STATEMENT> [ELSE <STATEMENT>]]
```

ORDERED ACCESS FORMAT

```
FIND/MODIFY [<DATASET-NAME> VIA]
  NEXT/PRIOR/FIRST/LAST
  <DATASET-NAME>/<SUBSET-NAME>/<ORDER-NAME>
  [ON EXCEPTION <STATEMENT> [ELSE <STATEMENT>]]
```

RANDOM ACCESS FORMAT

```
FIND/MODIFY [DATASET-NAME VIA]
  [NEXT] <SUBSET-NAME>/<ORDER-NAME>/<RETRIEVAL-NAME>
  AT <COMPONENT-NAME-1> = <DATA-NAME-1>/<LITERAL-1>
  [AND <COMPONENT-NAME-2> = <DATANAME-2>/<LITERAL-2>]...
  [ON EXCEPTION <STATEMENT> [ELSE <STATEMENT>]]
```

NOTE: THE "<DATASET-NAME> VIA" CLAUSE IS MANDATORY
WHEN THE PATH IS A <SUBSET-NAME>

```
CT.VERB=15
CT.OBJECT=INVOKE,NO, PATH,NO OF THE PATH-NAME
CT.ADVERB=
  BIT 3  =DMSTATUS FORMAT
    0:BINARY
    1:DECIMAL 4 BIT
  BIT 4  ="ON EXCEPTION"
  BIT 6  ="MODIFY"
  BIT 7-11=TYPE OF SELECTION EXPRESSION
    0:NEXT
    1:PRIOR
    2:FIRST
    3:LAST
    4:NEXT AND "AT" CLAUSE
    5:CURRENT
    6:"AT" CLAUSE
  BIT 12 =TYPE OF PATH
    0:KEY
    1:DATASET

CT.1=BIT LENGTH OF DM-STATUS REGISTER
CT.2=BASE RELATIVE ADDRESS OF DM-STATUS REGISTER
```

```
    CT.3=BIT LENGTH OF THE DATASET RECORD WORKAREA
    CT.4=BASE RELATIVE ADDRESS OF THE DATASET RECORD WORK AREA
    CT.5=BASE RELATIVE ADDRESS OF SEARCH KEY(CAT OF COMPONENT NAMES)
    CT.6=INVOKE.NO, PATH.NO OF DATASET-NAME

  MERGE OUTPUT
  ------------

  TRESWD  FIND/MODIFY
  TINTGR  INVOKE.NUM OF DATASET-NAME
  TINTGR  PATH.NUM OF DATASET-NAME
  TWORD   DATASET-NAME
  TINTGR  INVOKE.NUM OF THE PATH
  TINTGR  PATH.NUM OF THE PATH
  TWORD   PATH NAME
 [TRESWD  NEXT/FIRST/PRIOR/LAST]
          <SELECTION EXPRESSION>
          <EXCEPTION CLAUSE>

WHERE <SELECTION EXPRESSION> IS:
  TWORD   COMPONENT-NAME
  TRESWD  "="
  TWORD/LITERAL
      .
      .
      .

1. VERIFY THE PATH NAME TO BE A PROPER PATH
   TO THE DATASET-NAME AND OUTPUT THE PATH INFO

2. OUTPUT THE TARGET DATASET-NAME AND ITS PATH INFO
   (SUPPLY THE PARENT OCUR AS A TWORD TOKEN
   IF THE DATASET-NAME IS OMITTED)

3. VERIFY THE COMPONENT-NAME LIST TO BE IN THE PROPER ORDER WITH NO
   NAMES MISSING FROM THE LIST

4. THE TYPE OF THE PATH NAME IS DATASET IF BIT 7 OF THE DNFLAGS
   IS ON IN THE PATHNAME.  THIS BIT IS USED AS THE FILLER FLAG IN
   THE DATA DIVISION.
```

DATSYN
------

1. WHEN DATASET IS DECLARED ALLOCATE A SEPARATE WORK AREA FOR IT

2. WHEN AN ENTRY WITH LEVEL=51, 52, OR 53 IS DECLARED,
   TREAT IT LIKE AN 88

3. VERIFY THE UNIQUENESS OF THE 01 DATASET-NAMES WITHIN A DB
   AND DATA BASE NAMES

EXPLODE
----

1. CHANGE TWORD TO TINDEX FOR ITEMS WITH
   LEVEL=51 THRU 53(DONE IN PROCEDURE RESOLVE.FACTORS)
   IN ORDER TO RESTRICT THE USE OF PATH NAMES TO
   THE DM.VERBS

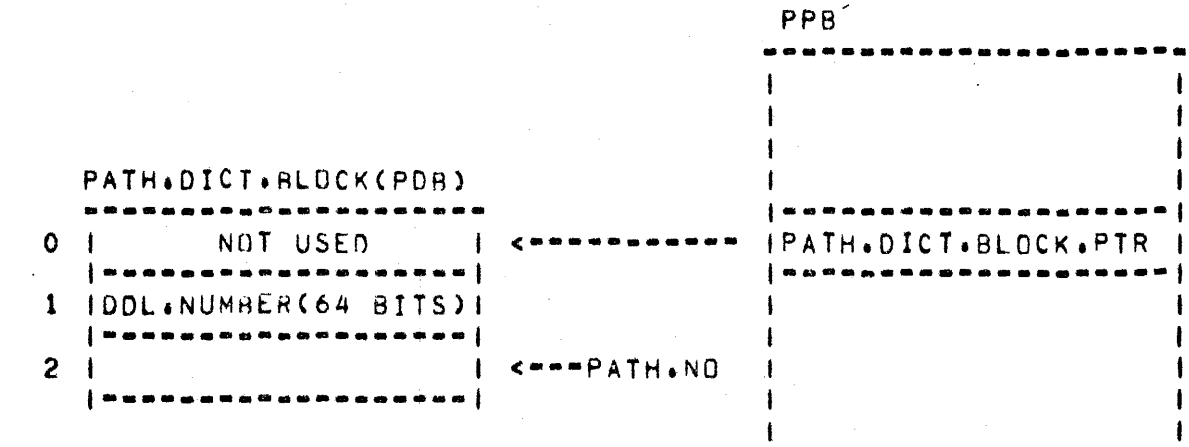
PROSYN PROCESSING
----

1. PARSE THE "ON EXCEPTION ..." CLAUSE OF THE DM.VERBS

CODEGEN
----

1. GENERATE COMMUNICATES CORRESPONDING TO DM VERBS

FIXUP PROCESSING



1. THE COMPILER ASSIGNS A PATH NUMBER TO EACH UNIQUE PATH NAME DECLARED IN THE DATA-BASE SECTION (E.G. DATASET-NAME, SUBSET-NAME, ORDER-NAME, RETRIEVAL-NAME)

2. THE PATH DICTIONARY BLOCK(PDB) CONTAINS A PATH DICTIONARY ENTRY (PDE) FOR EACH PATH NAME. THE PDE CONTAINS THE 64 BIT DDL-NUMBER ASSOCIATED WITH THAT PATH NAME.

3. THE PATH.NO AND INVOKE.NO ARE USED TO IDENTIFY THE DESIRED PATH TO A DATASET RECORD. THE PATH.NO IS USED AS AN INDEX INTO THE PATH.DICTIONARY TO GET THE DDL.NUMBER. THE INVOKE.NO IS USED TO IDENTIFY THE INFORMATION ABOUT A DATASET RECORD WHERE THAT DATASET HAS BEEN INVOKED MORE THAN ONCE.

SAMPLE DATA-BASE SECTION

```
                                                    OCUR  INV#  PATH#  TGT#  CN  CN
 DATA-BASE SECTION.                                  01
 DB UNIVERSITIES.                                    02
 01 STUDENT INVOKE UNIV-PERSONNEL.
*01 UNIV-PERSONNEL DATASET DDL-NUMBER 88             03    01    01     01
*    ORDERING KEY PERSNAMES DDL-NUMBER 73           04    01    02     01   25  27
*      (LASTNAME,FIRSTNAME)
*    RETRIEVAL KEY PERSMAJORS DDL-NUMBER 82         05    01    03     01   34
*      (MAJOR)
*    RETRIEVAL KEY PERSAGES DDL-NUMBER 2            06    01    04     01   29
*      (AGE).
*   02 NAME.                                        07
*     03 LASTNAME PC X(15).                         08
*     03 INITIAL PC X.                              09
*     03 FIRSTNAME PC X(10).                        10
*   02 SEX PC 9 CMP.                                11
*   02 AGE PC 99 CMP.                               12
*   02 CORSE-NOS OCCURS 4.                          13
*     03 DEPT PC XX.                                14
*     03 NMBR PC 999 CMP.                           15
*   02 COURSE SUBSET DDL-NUMBER 8                   16    01    05     06   41  42
*      TO UNIV-COURSES DDL-NUMBER 14
*        (DEPARTMENT,LEVEL).
*   02 MAJOR PC X(4).                               17
*   02 SALARY PC 9(5)V99 CMP.                       18
*   02 ADVISOR SUBSET DDL-NUMBER 16                 19    01    07     01
*      TO UNIV-PERSONNEL DDL-NUMBER 88.
 01 FACULTY INVOKE UNIV-PERSONNEL.
*01 UNIV-PERSONNEL DATASET DDL-NUMBER 88             20    02    01     01
*    ORDERING KEY PERSNAMES DDL-NUMBER 73           21    02    02     01   25  27
*      (LASTNAME,FIRSTNAME)
*    RETRIEVAL KEY PERSMAJORS DDL-NUMBER 82         22    02    03     01   34
*      (MAJOR)
*    RETRIEVAL KEY PERSAGES DDL-NUMBER 2            23    02    04     01   29
*      (AGE).
*   02 NAME.                                        24
*     03 LASTNAME PC X(10).                         25
*     03 INITIAL PC X.                              26
*     03 FIRSTNAME PC X(10).                        27
*   02 SEX PC 9 CMP.                                28
*   02 AGE PC 99 CMP.                               29
*   02 CORSE-NOS OCCURS 4.                          30
*     03 DEPT PC XX.                                31
*     03 NMBR PC 999 CMP.                           32
*   02 COURSE SUBSET DDL-NUMBER 8                   33    02    05     06   41  4
*      TO UNIV-COURSES DDL-NUMBER 14
*        (DEPARTMENT,LEVEL).
```

```
 *   02 MAJOR PC X(4).                                  34
 *   02 SALARY PC 9(5)V99 CMP.                          35
 *   02 ADVISOR SUBSET DDL-NUMBER 16                    36   02   07   01
 *      TO UNIV-PERSONNEL DDL-NUMBER 88.
  01 U-COURSES INVOKE UNIV-COURSES.
 *01 UNIV-COURSES DATASET DDL-NUMBER 14                 37   03   06   06
 *    ORDERING KEY DEPTLEVELS DDL-NUMBER 32            38   03   08   06   41   42
 *      (DEPARTMENT,LEVEL)
 *    RETRIEVAL KEY CLASS-SZ DDL-NUMBER 34             39   03   09   06   50
 *      (CLASS-SIZE)
 *    RETRIEVAL KEY DEPTS DDL-NUMBER 12                40   03   10   06   41
 *       (DEPARTMENT).
 *   02 DEPARTMENT PC XX.                               41
 *   02 LEVEL PC 999 CMP.                               42
 *   02 PROFESSOR SUBSET DDL-NUMBER 33                  43   03   11   01
 *       TO UNIV-PERSONNEL DDL-NUMBER 88.
 *   02 BOOKS DATASET DDL-NUMBER 7.                     44   03   12   12
 *     03 TITLE PC X(60).                               45
 *     03 AUTHOR PC X(30).                              46
 *   02 DAYS-OF-WEEK PC XXX.                            47
 *   02 BUILDING PC 999 CMP.                            48
 *   02 ROOM PC XX.                                     49
 *   02 CLASS-SIZE PC 99 CMP.                           50
 *   02 STUDENTS DATASET DDL-NUMBER 38                  51   03   13   13
 *      ORDERING KEY STUDNAMES DDL-NUMBER 71            52   03   14   13   53   5
 *        (LAST-NAME,FIRST-NAME).
 *     03 LAST-NAME PC X(15).                           53
 *     03 FIRST-NAME PC X(10).                          54
```

PATH DICTIONARY TABLE

PATH.NO	STRUCTURE.NO
0	NOT USED
1	88
2	73
3	82
4	02
5	08
6	14
7	16
8	32
9	34
10	12
11	33
12	07
13	38
14	71

GENERAL COMMENTS

1. THE SOURCE IMAGES WITH AN "*" ARE PROVIDED BY THE DASDL COMPILER

AS A LIBRARY FILE TO THE COBOL COMPILER

2. THE INVOKED NAME IS THE NAME OF THAT FILE AND
IS NOT A PROPER DATANAME OF THE DATSET
(E.G. "*UNIVERSIT"/"UNIV-PERSO")
- THE NAME "STUDENT" REPLACES THE WORD "UNIV-PERSONNEL" AT OCUR=3
- THE NAME "UNIV-PERSONNEL" IS DELETED WHEN IT APPEARS AS THE
TARGET DATASET-NAME OF ADVISOR AT OCCUR=19 AND OCUR=36

3. "DATA-BASE" SECTION APPEARS TO BE A FILE NAME(OCUR=1)
WHOSE SCOPE INCLUDES ALL DATA.BASE DATANAMES(SCOPE=54)

4. COMPONENT NAMES APPEAR TO BE QUALIFIED BY
"DATA-BASE" AND MUST NOT BE QUALIFIED WHEN REFERENCED
- THIS SPECIAL NAMING CONVENTION ALLOWS UNIDENTIFIED
COMPONENT-NAMES IN THE CASE WHERE THE TARGET DATASET-NAME
OF A SUBSET IS NOT INVOKED
- WHEN DUPLICATE COMPONENT NAMES EXIST THEN THE OCUR OF
LAST OF THE DUPLICATES WILL BE RETURNED
(E.G. LASTNAME APPEARS AT OCUR=8 AND OCUR=25)
- DDL WILL GUARANTEE THE UNIQUENESS OF COMPONENT-NAMES
IN THE DATABASE

5. THE INVOKE NUMBER STARTS AT 1 AND IS BUMPED FOR
EACH INVOKE SPECIFIED

6. THE PATH.NO IS ASSIGNED BY SEQUENTIALLY
SEARCHING THE PATH.DICTIONARY.TABLE FOR A
MATCHING STRUCTURE NUMBER. IF A MATCH IS NOT
FOUND THEN THE NEW STRUCTURE NUMBER IS ADDED TO
THE END OF THE TABLE. THE INDEX INTO THE TABLE
IS THE PATH.NO. NOTE THAT THE DDL-NUMBER IN THE EXAMPLE CONTAINS
ONLY THE STRUCTURE NUMBER(THE TIME AND DATE PART
HAS BEEN OMITED FOR CLARITY)

PATH TABLE AND ALGORITHM FOR ITS CONSTRUCTION

```
DECLARE
1 PATH.TABLE  REMAPS BASE,
  2 PATH.OCUR              BIT(12),
  2 LAST.ENTRY.LINK        BIT(24),
  2 PATH.TYPE              BIT(3),
  2 INVOKE.NUM             BIT(8),
  2 PATH.NUM               BIT(8),
  2 TARGET.PATH.NUM        BIT(8),
  2 COMP.OCUR(MAX.COMP) BIT(12),
1 TARGET.STACK (16),
  2 TS.LEVEL               BIT(8),
  2 TS.PATH                BIT(8);

-FIND A PATHNAME
    TXDN FOLLOWED BY "DATASET" OR TXDN WITH LEVEL NUMBER 51-53.
    STOP ON "WORKING-STORAGE SECTION" OR "PROCEDURE DIVISION".
```

```
*FILL IN PATH.OCUR

*FILL IN PATH.TYPE
    DETERMINE TYPE FROM LEVEL NUMBER

         LEVEL      TYPE
         -----      ----
         01          0    DATASET-NAME
         02-49       1    DATASET-NAME(NESTED)
         51          2    ORDER-NAME
         52          3    RETRIEVAL-NAME
         53          4    SUBSET-NAME

*IF LEVEL = 01 THEN CURRENT INVOKE NUMBER GETS BUMPED.

*FILL IN INVOKE.NUM

*FILL IN PATH.NUM
    DETERMINE FROM LOOKING UP DDL# IN PATH DICTIONARY; IF NOT
    FOUND, ADD NEW ENTRY.

*ADJUST TARGET STACK
    POP STACK UNTIL LEVEL ON TOP IS LESS THAN CURRENT LEVEL (OR
    STACK EMPTY).  IF CURRENT LEVEL < 50 (DATASET) THEN PUSH NEW ENTRY.

*FILL IN TRGT.PATH.NUM
    IF TARGET DDL# IS EXPLICIT, FIND FROM PATH DICTIONARY, ADDING A
    NEW ENTRY IF NECESSARY.  IF TARGET IS IMPLICIT THEN USE PATH
    ON TOP OF TARGET STACK.

 * FILL IN AN OCUR FOR EACH COMPONENT-NAME WHICH
   FOLLOWS IN THE PARENTHESIZED LIST. ADD AN EXTRA
   OCUR OF ZERO , TERMINATING THE LIST.
```

TABLE OF CONTENTS:

ALPHABETIC INDEX: