<table>
<thead>
<tr>
<th>SPACE 5</th>
<th>AUTHORS...</th>
<th>FT000040</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>STANLEY HAGSTROM</td>
<td>FT000050</td>
</tr>
<tr>
<td></td>
<td>FRANKLIN PROSSER</td>
<td>FT000060</td>
</tr>
<tr>
<td></td>
<td>STEPHEN YOUNG</td>
<td>FT000070</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FT000080</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>EJECT</td>
<td>ON</td>
<td>FT000170</td>
</tr>
<tr>
<td>PCC</td>
<td>OFF</td>
<td>FT000230</td>
</tr>
<tr>
<td>COUNT</td>
<td>15000</td>
<td>FT000180</td>
</tr>
<tr>
<td>LBL</td>
<td>IFSTR</td>
<td>FT000190</td>
</tr>
<tr>
<td>NOCRS</td>
<td></td>
<td>FT000200</td>
</tr>
<tr>
<td>TITLE</td>
<td></td>
<td>FT000210</td>
</tr>
<tr>
<td>PCC</td>
<td>OFF</td>
<td>FT000220</td>
</tr>
<tr>
<td>SPACE 9</td>
<td>FT000270</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>FT000280</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>FT000290</td>
<td></td>
</tr>
<tr>
<td>FASTRAN COMPILER</td>
<td>FT000300</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>FT000310</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>FT000320</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>FT000330</td>
<td></td>
</tr>
<tr>
<td>INDIANA UNIVERSITY</td>
<td>FT000340</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>FT000350</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>FT000360</td>
<td></td>
</tr>
<tr>
<td>BLOOMINGTON, INDIANA</td>
<td>FT000360</td>
<td></td>
</tr>
<tr>
<td>SPACE</td>
<td>3</td>
<td>FT000370</td>
</tr>
<tr>
<td>-------</td>
<td>---</td>
<td>----------</td>
</tr>
<tr>
<td>*</td>
<td>*</td>
<td>FT000380</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FT000390</td>
</tr>
<tr>
<td>*</td>
<td></td>
<td>FT000400</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FT000410</td>
</tr>
<tr>
<td>*</td>
<td></td>
<td>FT000420</td>
</tr>
</tbody>
</table>
**TTL INTRODUCTION -- GENERAL COMMENTARY**

* FASTRAN IS A FORTRAN II COMPILER WRITTEN FOR THE IBM

* 709/7090/7094 SERIES OF COMPUTERS. FASTRAN STANDS FOR FAST

* FORTRAN. IT WAS DESIGNED AND WRITTEN AT INDIANA UNIVERSITY

* IN LATE 1963 AND EARLY 1964 BY STANLEY HAGSTROM, FRANKLIN

* PROSSER, AND STEPHEN YOUNG. THE PROJECT WAS UNDERTAKEN FOR TWO

* REASONS. FIRST, WE WERE DISSATISFIED WITH THE COMPILATION SPEED

* OF IBM'S FORTRAN II COMPILER. SECOND, WE WISHED TO GAIN SOME

* EXPERIENCE IN COMPILER-WRITING TECHNIQUES. THE ORIGINAL GOAL

* WAS TO PRODUCE A COMPILER WHICH WOULD COMPILE

* PROGRAMS QUICKLY, PROBABLY AT THE EXPENSE OF OBJECT PROGRAM

* LENGTH AND EFFICIENCY.

* THE GOAL OF FAST COMPILATION WAS READILY ACHIEVED. HOWEVER,

* TO OUR PLEASANT SURPRISE, THE RESULTING OBJECT PROGRAMS WERE

* IN GENERAL SHORTER THAN IBM'S, AND THE EXECUTION TIMES DID

* NOT COMPARE TOO UNFAVORABLY, EVEN IN THE EARLY VERSIONS.

* REM

* IN A SENSE, FASTRAN IS AN OUTGROWTH OF AN EARLIER EFFORT AT

* INDIANA UNIVERSITY BY DAVID KURN AND ONE OF THE AUTHORS (SHI)

* TO PRODUCE A LOAD-AND-GO COMPILER (CALLED FLAT AND BASED ON A

* GENERALIZATION OF THE FORTRAN IV LANGUAGE) DURING THE SUMMER OF

* 1962. THIS PROJECT, ALTHOUGH EVENTUALLY ABANDONED, SHOWED

* CLEARLY HOW TO ACHIEVE HIGH COMPILERS SPEEDS WITH REASONABLE

* OBJECT CODE EFFICIENCY. MANY OF THE TECHNIQUES USED IN FLAT

* HAVE ALSO BEEN USED IN FASTRAN, INCLUDING THE METHOD OF SCANNING

* STATEMENTS AND PROCESSING ARITHMETIC EXPRESSIONS. IN FACT,

* INITIALLY SOME SECTIONS OF THE FLAT CODE WERE TAKEN OVER BODILY

* TO FASTRAN. IT IS DOUBTFUL IF WITHOUT THIS EARLIER EXPERIENCE

* ON FLAT WE COULD HAVE AVOIDED THE MANY PITFALLS AWAITING THE

* COMPILER WRITER AND HAVE SO EASILY MET OUR ORIGINAL GOALS.

* REM

* IT IS ESTIMATED THAT FROM START TO FINISH REQUIRED ABOUT NINE

* MAN-MONTHS OF EFFORT FOR THE FIRST VERSION OF FASTRAN, WITH

* MAYBE THREE MAN-MONTHS MORE REQUIRED FOR REFINEMENTS.

* REM

* FASTRAN IS A TWO-PASS IN-CORE COMPILER THAT USES A SINGLE SCRATCH

* TAPE IN THE COMPILATION OF LARGE PROGRAMS. FASTRAN ACCEPTS THE

* COMPLETE FORTRAN II LANGUAGE, INCLUDING HOLLERITH LITERALS AND

* DOUBLE PRECISION AND COMPLEX ARITHMETIC. ALSO, A FLOATING

* ROUNDED FEATURE IS IMPLEMENTED, AND PAGE SPACING, EJECTING, AND

* SUBTITLE IS PERMITTED. FASTRAN MAY OPERATE IN A MONITOR

* SYSTEM SUCH AS FMS, OR IT MAY BE EXECUTED AS AN FMS JOB, WITH

* SOURCE PROGRAMS AS THE DATA. (IN THIS LAST MODE, CLEARLY NO

* OBJECT PROGRAM EXECUTION IS POSSIBLE FOLLOWING COMPIALTION.)

* REM

* OUR AIM HAS BEEN TO MAINTAIN COMPATIBILITY WITH IBM'S FMS-II

* VERSION OF FORTRAN II. FASTRAN PRODUCES AN OBJECT PROGRAM

* BINARY DECK WHICH IS COMPLETELY ACCEPTABLE TO THE BSS LOADER.

* THE STANDARD FMS LIBRARY IS UTILIZED, AND FASTRAN'S CALLS TO

* THESE LIBRARY ROUTINES ARE IDENTICAL TO THOSE OF FORTRAN II.

* THE FASTRAN AND FORTRAN II BINARY DECKS ARE COMPLETELY INTER-

* CHANGABLE AS FAR AS SYSTEM FUNCTIONS ARE CONCERED. FORTRAN II

* SOURCE LANGUAGE DEBUGGING MAY BE USED WITH FASTRAN-COMPILED

* PROGRAMS. TO DO THIS, A SYMBOL TABLE MUST BE REQUESTED, JUST

* AS IN IBM'S FORTRAN II.

* REM

* FASTRAN IS WRITTEN TO OPERATE USING THE INDIANA UNIVERSITY

* INPUT/OUTPUT SYSTEM, IOS. THIS SYSTEM AUTOMATICALLY PROVIDES

* BLOCKING AND BUFFERING FOR ALL INPUT/OUTPUT. A BRIEF DESCRIPTION

* OF IOS, AND A DISCUSSION OF FASTRAN'S INPUT/OUTPUT REQUIREMENTS

* ARE PROVIDED IN LATER PAGES. WITH THIS INFORMATION, THE TASK

* FT001020
OF IMPLEMENTING FASTRAN WITH ANOTHER I/O SYSTEM IS GENERALLY NOT DIFFICULT. NOTICE THAT WITHOUT BLOCKING AND BUFFERING OF TAPES, FASTRAN WILL BE TAPE-BOUND, AND THE COMPILATION SPEEDS WILL SUFFER.

REM

IF FASTRAN IS USED WITH AN ADEQUATE INPUT/OUTPUT SYSTEM (SUCH AS I/O), COMPIIATION SPEEDS ARE OBTAINED THAT ARE 10 TO 100 TIMES FASTER THAN THE IBM FORTRAN II UNDER FMS II. AN 'AVERAGE' COMPILE TIME FACTOR IS ABOUT 13, BUT VERY SHORT PROGRAMS RESULT IN QUITE LARGE TIME FACTORS, AND IN GENERAL THE SHORTER THE PROGRAM, THE LARGER THE RATIO OF FORTRAN II VS FASTRAN COMPILATE TIMES. THIS TENDT IS DUE MAINLY TO THE SUBSTANTIAL OVERHEAD TIME IN IBM'S COMPILER. NEVERTHLESS, FOR LARGE PROGRAMS, THE TIME FACTOR APPEARS TO APPROACH 10, SO FASTRAN MAY BE CONSERVATIVELY SAID TO RUN AT A BASIC SPEED ABOUT TEN TIMES FASTER THAN THE IBM COMPILER.

REM

OUR EXPERIENCE SHOWS THAT THE OBJECT CODE PRODUCED BY FASTRAN AVERAGES ABOUT TEN PERCENT SHORTER THAN THAT PRODUCED BY IBM'S FORTRAN II. THIS FIGURE IS BASED ON A CONSIDERATION OF NUMBER OF INSTRUCTIONS, WORKING CELLS, AND CONSTATS PRODUCED.

REM

THE EXECUTION TIME OF FASTRAN-PRODUCED OBJECT PROGRAMS IS GENERALLY SLIGHTLY LONGER THAN FOR IBM'S VERSIONS. THIS IS ATTRIBUTABLE MAINLY TO THE LACK OF INDEXING OPTIMIZATION IN THIS VERSION OF FASTRAN. THE MOST USEFUL FORMS OF INDEXING IMPROVE-MENT ARE FORTUNATELY EASY TO IMPLEMENT IN FASTRAN, AND THIS IS CURRENTLY BEING DONE. ODDLY ENOUGH, WE ANTICIPATE THAT SINCE IMPROVED INDEXING WILL FURTHER SHORTEN THE OBJECT CODE, FASTRAN COMPILATIONS SHOULD SPEED UP A BIT...

REM

FASTRAN IS AN IN-CORE COMPILER. IT CAN REMAIN IN CORE THROUGH THE COMPIIATION OF A NUMBER OF CONSECUTIVE FORTRAN PROGRAMS. THIS ALLOWS A VERY SMALL SYSTEM OVERHEAD TIME. THERE ARE NO EXTRANEOUS PASSES OVER THE MATERIAL TO BE COMPILED. THERE IS ONE PASS OVER THE SOURCE DECK, AND A SECOND AND FINAL PASS OVER THE INTERMEDIATE MATERIAL TO PRODUCE THE BINARY DECK. SHORT AND MEDIUM-LENGTH PROGRAMS ARE COMPILED IN CORE, AND NO SCRATCH TAPE IS USED (I.E. AN INTERNAL FILE IS MAINTAINED). ON LONGER PROGRAMS, A SINGLE INTERMEDIATE TAPE IS WRITTEN, REMOVED, AND THEN READ SEQUENTIALLY. THERE IS NO DELAY DURING THE REWIND, SINCE THE INTERNAL FILE IS PROCESSED AT THIS TIME. THE COMPIIATION DOES NOT PROCEED THROUGH A SYMBOLIC ASSEMBLY PHASE, THUS ELIMINATING ADDITIONAL TIME-CONSUMING OPERATIONS (THIS IS OF COURSE DONE AT THE QUESTIONABLE EXPENSE OF AN ASSEMBLY LISTING OPTION).

IT IS PERHAPS APPROPRIATE AT THIS POINT TO STRESS THAT FASTRAN IS BASICALLY A ONE-PASS COMPILER, THE SECOND PASS BEING REQUIRED SOLELY BY THE BSS BINARY DECK FORMAT (I.E. PROGRAM CARD FIRST AND NO PROVISION FOR CHAINED BACKFILLS, ETC.).

REM

ALL SOURCE PROGRAM SYMBOLS AND LABELS ARE PLACED IN A SYMBOL TABLE (SYMTAB) WHOSE SIZE IS 4096 WORDS. THE METHOD OF ASSIGNING A SYMBOL OR LABEL TO A LOCATION IN SYMTAB IS SUCH THAT NO LENGTHY SEARCHES OR SORTS ARE NECESSARY. A SCRAMBLED SORT TECHNIQUE IS USED, WHEREBY THE BITS IN A BCD SYMBOL OR LABEL ARE CONCENTRATED INTO A 12-BIT FIELD (2**12=4096). THIS IS DONE BY MULTIPLYING THE BCD SYMBOLIC NAME BY A SIMPLE STANDARD INTEGER, AND EXTRACTING A TWELVE-BIT FIELD FROM THE RESULT. THE POINTER THUS OBTAINED IS NORMALLY THE SYMTAB LOCATION.
OF THE SYMBOL OR LABEL, IF HOWEVER THE LOCATION IS ALREADY
OCCUPIED BY A DIFFERENT SYMBOL (UNLIKELY IN MOST PROGRAMS), THEN
THE SYMBOL IS ASSIGNED TO THE NEXT AVAILABLE LOCATION. TO
LOCATE A SYMBOL IN SYMTAB, THE SAME COMPUTATION IS PERFORMED.
IF THE SYMBOL IS NOT AT THE COMPUTED LOCATION, A SEQUENTIAL
SEARCH IS PERFORMED UNTIL EITHER THE ITEM IS LOCATED, OR AN
EMPTY CELL IS ENCOUNTERED. (THE EMPTY CELL MEANS OF COURSE
THAT THE SYMBOL IS NOT IN THE TABLE, AND IT MAY BE INSERTED
IF DESIRED IN THE LOCATED EMPTY CELL).
THE TECHNIQUE IS VERY RAPID FOR SPARSELY-FILLED SYMBOL TABLES.
IT BECOMES LESS EFFICIENT AS THE TABLE FILLS UP, SINCE MORE AND
LONGER SEQUENTIAL SEARCHES WILL BE PERFORMED. HOWEVER, A VERY
LARGE PROGRAM IS REQUIRED TO FILL THE TABLE MORE THAN SPARSELY.
EACH SYMTAB WORD HAS A COUNTERPART IN ANOTHER 4096-WORD TABLE;
EQUIV. EQUIV CONSISTS PRIMARILY OF FLAG BITS DESCRIBING THE
CORRESPONDING SYMTAB ENTRY, AND AN ADDRESS POINTER WHICH MAY BE
USED TO DEFINE THE VALUE OF THE ENTITY, OR TO POINT INTO A
GENERAL POOL AREA WHERE MOST ADDITIONAL INFORMATION IS STORED.

THE FORMAT OF THE EQUIV TABLE ENTRY IS DESCRIBED NOW IN GENERAL
TERMS. THE ACTUAL BIT CONFIGURATIONS FOR THE VARIOUS FLAGS AND
FIELDS ARE FOUND IN THE REAR OF THE LISTING, UNDER 'ABSOLUTE
SYMBOL DEFINITIONS'.

THE PREFIX OF AN EQUIV WORD HOLDS THE MODE OF THE ITEM. THE
MODE IS REPRESENTED BY A THREE-BIT FIELD, AND NOT AS INDIVIDUAL
BITS. THE FOUR MACHINES ARE MLABEL (LABEL ON EXECUTABLE STATEMENT),
NAME, MSTRG (FORMAT STRING LABEL), MINTG (INTEGER VARIABLE OR FUNCTION
NAME), AND MREAL (REAL VARIABLE OR FUNCTION NAME).

BITS 3-14 HOLD BIT FLAGS DESCRIBING THE STATUS OF THE SYMTAB
ITEM. NORMALLY EACH BIT STANDS FOR A SINGLE ITEM OF INFORMATION. IF NO CONFU-
SION CAN RESULT, THE BIT FLAGS AND THEIR INTERPRETATIONS ARE...

BVARB... SYMBOL HAS APPEARED AS A VARIABLE IN AN EXECUTABLE
STATEMENT, AND THIS MUST BE DEFINED IN THE PROGRAM.
BLHSX... SYMBOL OR LABEL HAS BEEN DEFINED (HAS A 'LEFT-HAND
VALUE').
BDoub... SYMBOL WAS USED AT SOME POINT AS A DOUBLE PRECISION
OR A COMPLEX VARIABLE, AND MUST HAVE DOUBLE STORAGE.
BARGF... SYMBOL IS A FORMAL SUBPROGRAM PARAMETER.
BARRY... SYMBOL APPEARS IN A DIMENSION STATEMENT, ADDRESS FIELD
OF EQUIV WORD POINTS TO THE COMMON/DIMENSION/EQUIVALENCE
INFORMATION IN THE POOL AREA.

BCOMN... SYMBOL APPEARS IN A COMMON STATEMENT, ADDRESS FIELD
OF EQUIV WORD POINTS TO POOL AREA.
BEUQt... SYMBOL APPEARS IN ONE OR MORE EQUIVALENCE STATEMENTS,
ADDRESS FIELD OF EQUIV WORD POINTS TO POOL AREA.
BASSN... SYMBOL APPEARS AS THE VARIABLE IN AN ASSIGN OR
ASSIGNED GO TO STATEMENT. ADDRESS FIELD OF EQUIV WORD POINTS
TO ASSIGN LISTS IN POOL AREA.
BINF... SYMBOL IS USED AS AN ARITHMETIC STATEMENT FUNCTION
NAME.
BEXTF... SYMBOL IS USED AS AN EXTERNAL NAME.
BLEBF... EXTERNAL SYMBOL NAME IS CURRENTLY OF LIBRARY FUNCTION
TYPE, I.E. ENDS IN 'F'.
BPATH... LABEL IS IN THE PATH OF FLOW OF THE PROGRAM, I.E. THERE IS A WAY FOR THE PROGRAM FLOW TO REACH THIS STATEMENT.
* BIT 15-20 HOLD THE SUBPROGRAM ARGUMENT NUMBER, IF BAR GT IS ON.
* A MAXIMUM OF 64 FORMAL PARAMETERS CAN BE HANDLED.
* BIT 21-35, THE ADDRESS FIELD, HOLD A SYMBOL OR LABEL DEFINITION, OR IF CERTAIN EQUIV FLAGS ARE ON, POINT TO FURTHER INFORMATION IN THE POOL AREA OR ELSEWHERE.
* ASIDE FROM SYMTAB AND EQUIV, THERE ARE VERY FEW FIXED-LENGTH TABLES IN FASTRAN. THIS ALLLOWS MAXIMUM FLEXIBILITY IN THE USE OF AVAILABLE STORAGE. CONSEQUENTLY, THE LIKELIHOOD OF 'OVERFLOW' OF THE COMPILER IS QUITE REMOTE.
* A BRIEF SURVEY OF THE COMPILATION PROCESS IS GIVEN HERE. MORE DETAILED INFORMATION IS TO BE FOUND IN DESCRIPTIONS OF THE INDIVIDUAL SECTIONS IN THE BODY OF THE LISTING. THE COMPILATION IS LOGICALLY DIVIDED INTO FIVE SECTIONS... INITIALIZATION, PASS 1, POST PASS 1, PASS 2, AND POST PASS 2.
* INITIALIZATION IS VERY SHORT, BEING CONCERNED WITH SETTING VARIOUS CELLS TO THEIR INITIAL VALUES.
* PASS 1 IS THE PRIMARY PASS OVER THE SOURCE DECK. THE SOURCE PROGRAM IS LISTED; THE STATEMENTS ARE SCANNED AND INTERPRETED, THE SYMBOL TABLE IS FORMED, VARIOUS LISTS ARE GENERATED, AND MANY TYPES OF ERRORS ARE DETECTED. INTERMEDIATE OBJECT CODE IS PREPARED, TO BE PROCESSED IN PASS 2 WHEN THE SCAN OF THE SOURCE DECK IS COMPLETE.
* POST PASS 1 PROCESSES THE SYMBOL TABLE, DETECTS ALL REMAINING ERRORS (ERRORS INVOLVING RELATIONS BETWEEN THE VARIOUS ITEMS), COMPUTES THE OBJECT PROGRAM LENGTH AND PUTS OUT THE BINARY DECK PROGRAM CARD, DEFINES ALL VARIABLES. IF THIS PROGRAM IS A SUBROUTINE OR FUNCTION, ITS PROLOGUE IS FORMED HERE.
* IF AN OBJECT PROGRAM SYMBOL TABLE HAS BEEN REQUESTED, IT IS FORMED AND PUT OUT PRIOR TO THE PROGRAM CARD.
* POST PASS 1 IS ENTERED ONLY IF PASS 1 WAS FREE FROM ERRORS.
* PASS 2 PROCESSES THE INTERMEDIATE OBJECT CODE, PRODUCING THE THE FINAL CODE AND THE REMAINDER OF THE BINARY DECK.
* POST PASS 2 PUTS OUT THE OBJECT PROGRAM STORAGE MAP, AND TERMINATES THE COMPI. PASS 2 AND POST PASS 2 ARE ENTERED ONLY FOR ERROR-FREE COMPILATIONS.
A FEW WORDS ON FASTARAN'S ERROR DETECTION PHILOSOPHY ARE IN ORDER. MOST ERRORS, INCLUDING TYPOGRAPHICAL AND SYNTACTICAL ERRORS, ARE CAUGHT DURING PASS 1, THE PRIMARY PASS OVER THE SYMBOLIC DECK. WE FEEL THAT DIAGNOSTIC MESSAGES ARE MOST INFORMATIVE WHEN THEY ARE ISSUED ALONG WITH THE ERRONEOUS STATEMENT. THEREFORE, SOURCES PROGRAM IMMEDIATELY FOLLOWING THE STATEMENT IN ERROR. THERE ARE SOME ERRORS, NAMLY INVOLVING INTER-STATEMENT RELATIONS, THAT CANNOT BE DETECTED UNTIL THE ENTIRE DECK HAS BEEN READ. THESE THEN ARE GIVEN IMMEDIATELY FOLLOWING THE END CARD. ALL ERROR MESSAGES ARE SIGNALLED BY AN OBVIOUS SERIES OF ASTERISK AT THE LEFT SIDE OF THE PAGE.

THE DIAGNOSTIC MESSAGES ARE AS INFORMATIVE AS POSSIBLE WITHOUT UNDULY SACRIFICING COMPILATION SPEED, AND WITHOUT UNDULY CHANGING THE CONFUSION OF THE PROGRAMMER. IN RESERVED WORD (FIXED FORMAT) STATEMENTS, ONE CAN BE RATHER SPECIFIC WITHOUT RISK OF CONFUSING THE SUBMITTER OF THE PROGRAM. HOWEVER, SPECIFIC DIAGNOSTICS ARE IN GENERAL AVOIDED IN ARITHMETIC STATEMENTS, SINCE WE FEEL THAT TOO OFTEN A SPECIFIC MESSAGE DOES NOT IN FACT IDENTIFY THE PROGRAMMER'S DIFFICULTY, BUT INSTEAD CAUSES HIM MOMENTARY PUZZLEMENT. THEREFORE, A COMMON ARITHMETIC STATEMENT ERROR MESSAGE IS SIMPLY 'INVALID FORM OF STATEMENT'.

IN PASSING WE NOTE THAT FASTARAN DETECTS MANY TYPES OF ERRORS THAT IBM'S FORTRAN II WILL NOT CATCH. AMONG THESE ARE... CONSISTENCY ERRORS IN ASSIGN/ASSIGNED-GO-TO USAGE, UNDEFINED VARIABLES, RELATIVE CONSTANT ERRORS, ERRORS IN THE CONSISTENCY OF ARITHMETIC STATEMENT FUNCTION USAGES, AND NONSENSE SUCH AS '75 GO TO 75'.

ALSO, A NUMBER OF VALID SITUATIONS INVOLVING FORTRAN II 'JOHNNY- COME-LATELY' FEATURES, SUCH AS MOLLERITH LITERALS AND DOUBLE PRECISION AND COMPLEX ARITHMETIC, ARE HANDLED CORRECTLY AND CONSISTENTLY IN FASTARAN, WHILE FORTRAN II COMPILES ERRONEOUSLY OR INCONSISTENTLY. (DETAILS WILL BE SUPPLIED UPON REQUEST).
THE LAYOUT OF CORE STORAGE IN THE FASTRAN COMPILER IS AS FOLLOWS.
THE CORE LOCATIONS OF THE VARIOUS SECTIONS ARE OF COURSE SUBJECT TO CHANGE. MOST LOCATIONS ARE NOT IMPORTANT, AND ARE NOT SHOWN.
<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>CODE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>REM</td>
<td>MONITOR CELLS, ETC</td>
</tr>
<tr>
<td>00144</td>
<td>REM</td>
<td>INPUT/OUTPUT SYSTEM (INDIANA UNIVERSITY IOS)</td>
</tr>
<tr>
<td>10000</td>
<td>REM</td>
<td>BEGINNING OF FASTRAN COMPILER</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>ONE-TIME INITIALIZATION WHEN COMPILER ENTERS CORE</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>INITIALIZATION FOR EACH COMPILATION</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>INITIAL CARD PROCESSING</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>BURSTING OF CARD</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>STATEMENT LABEL AND MODE</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>DETERMINATION OF STATEMENT TYPE</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>STATEMENT TERMINATION ROUTINE</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>PASS 1 STATEMENT PROCESSORS</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>ARITHMETIC STATEMENT PROCESSING</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>COMPILER</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>CODE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>FT003110</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FT003130</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FT003140</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FT003150</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FT003160</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FT003170</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FT003180</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FT003190</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FT003200</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FT003210</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FT003220</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FT003230</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FT003240</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FT003250</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FT003260</td>
</tr>
</tbody>
</table>
INDIANA UNIVERSITY FASTRAN (FAST FORTRAN II) COMPILER

SPACE 5
SPACE 5
AUTHORS...

STANLEY HAGSTROM
FRANKLIN PROSSER
STEPHEN YOUNG

RESEARCH COMPUTING CENTER, INDIANA UNIVERSITY
BLOOMINGTON, INDIANA

1964

COPYRIGHT INDIANA UNIVERSITY FOUNDATION 1964
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>EJECT</td>
<td></td>
<td>FT000170</td>
</tr>
<tr>
<td>PCC</td>
<td>ON</td>
<td>FT000180</td>
</tr>
<tr>
<td>COUNT</td>
<td>15000</td>
<td>FT000190</td>
</tr>
<tr>
<td>LBL</td>
<td>1FSTR</td>
<td>FT000200</td>
</tr>
<tr>
<td>NOCRS</td>
<td></td>
<td>FT000210</td>
</tr>
<tr>
<td>TITLE</td>
<td></td>
<td>FT000220</td>
</tr>
<tr>
<td>PCC</td>
<td>OFF</td>
<td>FT000230</td>
</tr>
</tbody>
</table>
INTRODUCTION

FASTRAN COMPILER

RESEARCH COMPUTING CENTER

INDIANA UNIVERSITY

BLOOMINGTON, INDIANA

GENERAL COMMENTARY
INTRODUCTION -- GENERAL COMMENTARY

FASTRAN IS A FORTRAN II COMPILER WRITTEN FOR THE IBM

709/7090/7094 SERIES OF COMPUTERS. FASTTRAN STANDS FOR FAST

FORTRAN. IT WAS DESIGNED AND WRITTEN AT INDIANA UNIVERSITY

IN LATE 1963 AND EARLY 1964 BY STANLEY HAGSTROM, FRANKLIN

PRES, AND STEPHEN YOUNG. THE PROJECT WAS UNDERTAKEN FOR TWO

REASONS. FIRST, WE WERE DISSATISFIED WITH THE COMPILATION SPEED

OF IBM'S FORTRAN II COMPILER. SECOND, WE WISHED TO GAIN SOME

EXPERIENCE IN COMPILER-WRITING TECHNIQUES. THE ORIGINAL GOAL

WAS TO PRODUCE A COMPILER WHICH WOULD COMPILE

PROGRAMS QUICKLY, PROBABLY AT THE EXPENSE OF OBJECT PROGRAM

LENGTH AND EFFICIENCY.

THE GOAL OF FAST COMPILATION WAS REALIZED. HOWEVER,

TO OUR PLEASANT SURPRISE, THE RESULTING OBJECT PROGRAMS WERE

IN GENERAL SHORTER THAN IBM'S, AND THE EXECUTION TIMES DID

NOT COMARE TOO UNFAVORABLY, EVEN IN THE EARLY VERSIONS.

REM

IN A SENSE, FASTTRAN IS AN OUTGROWTH OF AN EARLIER EFFORT AT

INDIANA UNIVERSITY BY DAVID KURN AND ONE OF THE AUTHORS (SH)

TO PRODUCE A LOAD-AND-GO COMPILER (CALLED FLAT AND BASED ON A

GENERALIZATION OF THE FORTRAN IV LANGUAGE) DURING THE SUMMER OF

1962. THIS PROJECT, ALTHOUGH EVENTUALLY ABANDONED, SHOWED

CLEARLY HOW TO ACHIEVE HIGH COMPILERS SPEEDS WITH REASONABLE

OBJECT CODE EFFICIENCY. MANY OF THE TECHNIQUES USED IN FLAT

HAVE ALSO BEEN USED IN FASTTRAN, INCLUDING THE METHOD OF SCANNING

STATEMENTS AND PROCESSING ARITHMETIC EXPRESSIONS. IN FACT,

INITIALLY SOME SECTIONS OF THE FLAT CODE WERE TAKEN OVER BODILY

TO FASTTRAN. IT IS DOUBTFUL IF WITHOUT THIS EARLIER EXPERIENCE

ON FLAT WE COULD HAVE AVOIDED THE MANY PITFALLS AWAITING THE

COMPILER WRITER AND HAVE SO EASILY MET OUR ORIGINAL GOALS.

IT IS ESTIMATED THAT FROM START TO FINISH REQUIRED ABOUT NINE

MAN-MONTHS OF EFFORT FOR THE FIRST VERSION OF FASTTRAN, WITH

MAYBE THREE MAN-MONTHS MORE REQUIRED FOR REFINEMENTS.

REM

FASTRAN IS A TWO-PASS IN-CORE COMPILER THAT USES A SINGLE SCRATCH

TAPE IN THE COMPILATION OF LARGE PROGRAMS. FASTTRAN ACCEPTS THE

COMPLETE FORTRAN II LANGUAGE, INCLUDING HOLLERITH LITERALS AND

DOUBLE PRECISION AND COMPLEX ARITHMETIC. ALSO, A FLOATING

ROUNDING FEATURE IS IMPLEMENTED, AND PAGE SPACING, EJECTING, AND

SUBTITLING IS PERMITTED. FASTTRAN MAY OPERATE IN A MONITOR

SYSTEM SUCH AS FMS, OR IT MAY BE EXECUTED AS AN FMS JOB, WITH

SOURCE PROGRAMS AS THE DATA. (IN THIS LAST MODE, CLEARLY NO

OBJECT PROGRAM EXECUTION IS POSSIBLE FOLLOWING COMPILATION.)

REM

OUR AIM HAS BEEN TO MAINTAIN COMPATIBILITY WITH IBM'S FMS-II

VERSION OF FORTRAN II. FASTTRAN PRODUCES AN OBJECT PROGRAM

BINARY DECK WHICH IS COMPLETELY COMPATIBLE TO THE BSS LOADER.

THE STANDARD FMS LIBRARY IS UTILIZED, AND FASTTRAN'S CALLS TO

THESE LIBRARY ROUTINES ARE IDENTICAL TO THOSE OF FORTRAN II.

THE FASTTRAN AND FORTRAN II BINARY DECKS ARE COMPLETELY INTER-

CHANGABLE AS FAR AS SYSTEM FUNCTIONS ARE CONCERNED. FORTRAN II

SOURCE LANGUAGE DEBUGGING MAY BE USED WITH FASTTRAN-COMPILRED

PROGRAMS. TO DO THIS, A SYMBOL TABLE MUST BE REQUESTED, JUST

AS IN IBM'S FORTRAN II.

REM

FASTRAN IS WRITTEN TO OPERATE USING THE INDIANA UNIVERSITY

INPUT/OUTPUT SYSTEM, IUS. THIS SYSTEM AUTOMATICALLY PROVIDES

BLOCKING AND BUFFERING FOR ALL INPUT/OUTPUT. A BRIEF DESCRIPTION

OF IUS, AND A DISCUSSION OF FASTTRAN'S INPUT/OUTPUT REQUIREMENTS

ARE PROVIDED IN LATER PAGES. WITH THIS INFORMATION, THE TASK

REM
OF IMPLEMENTING FASTRAN WITH ANOTHER I/O SYSTEM IS GENERALLY NOT DIFFICULT. NOTICE THAT WITHOUT BLOCKING AND BUFFERING OF TAPES, FASTRAN WILL BE TAPE-BOUND, AND THE COMPILATION SPEEDS WILL SUFFER.

REM

IF FASTRAN IS USED WITH AN ADEQUATE INPUT/OUTPUT SYSTEM (SUCH AS I/O), COMPIILATION SPEEDS ARE OBTAINED THAT ARE 10 TO 100 TIMES FASTER THAN THE IBM FORTRAN II UNDER FMS II. AN 'AVERAGE' COMPILATION TIME FACTOR IS ABOUT 13, BUT VERY SHORT PROGRAMS RESULT IN QUITE LARGE TIME FACTORS, AND IN GENERAL THE SHORTER THE PROGRAM, THE LARGER THE RATIO OF FORTRAN II VS FASTRAN COMPILATE. THIS TREND IS DUE MAINLY TO THE SUBSTANTIAL OVERHEAD TIME IN IBM'S COMPILER. NEVERTHLESS, FOR LARGE PROGRAMS, THE TIME FACTOR APPEARS TO APPROACH 10, SO FASTRAN MAY BE CONSERVATIVELY SAID TO RUN AT A BASIC SPEED ABOUT TEN TIMES FASTER THAN THE IBM COMPILER.

REM

OUR EXPERIENCE SHOWS THAT THE OBJECT CODE PRODUCED BY FASTRAN AVERAGES ABOUT TEN PERCENT SHORTER THAN THAT PRODUCED BY IBM'S FORTRAN II. THIS FIGURE IS BASED ON A CONSIDERATION OF NUMBER OF INSTRUCTIONS, WORKING CELLS, AND CONSTANTS PRODUCED.

REM

THE EXECUTION TIME OF FASTRAN-PRODUCED OBJECT PROGRAMS IS GENERALLY SLIGHTLY LONGER THAN FOR IBM'S VERSIONS. THIS IS ATTRIBUTABLE MAINLY TO THE LACK OF INDEXING OPTIMIZATION IN THIS VERSION OF FASTRAN. THE MOST USEFUL FORMS OF INDEXING IMPROVEMENT AREafortunately rather easy to implement in FORTRAN, AN THIS IS CURRENTLY BEING DONE. ODDLY ENOUGH, WE ANTICIPATE THAT SINCE IMPROVED INDEXING WILL FURTHER SHORTEN THE OBJECT CODE, FASTRAN COMPILATIONS SHOULD SPEED UP A BIT...

REM

FASTRAN IS AN IN-CORE COMPILER. IT CAN REMAIN IN CORE THROUGH THE COMPIILATION OF A NUMBER OF CONSECUTIVE FORTRAN PROGRAMS. THIS ALLOWS A VERY SMALL SYSTEM OVERHEAD TIME. THERE ARE NO EXTRANEOUS PASSES OVER THE MATERIAL TO BE COMPIILE. THERE IS ONE PASS OVER THE SOURCE DECK, AND A SECOND AND FINAL PASS OVER THE INTERMEDIATE MATERIAL TO PRODUCE THE BINARY DECK. SHORT AND MEDIUM-LENGTH PROGRAMS ARE COMPIILE IN CORE, AND NO SCRATCH TAPE IS USED (I.E., AN INTERNAL FILE IS MAINTAINED). ON LONGER PROGRAMS, A SINGLE INTERMEDIATE TAPE IS WRITTEN,rewound, and then read sequentially. THERE IS NO DELAY DURING THE Rewind, since the internal file is processed at this time. THE COMPILATION DOES NOT PROCEED THROUGH A SYMBOLIC ASSEMBLY PHASE, THUS ELIMINATING ADDITIONAL TIME-CONSUMING OPERATIONS (THIS IS OF COURSE DONE AT THE QUESTIONABLE EXPENSE OF AN ASSEMBLY LISTING OPTION).

REM

IT IS PERHAPS APPROPRIATE AT THIS POINT TO STRESS THAT FASTRAN IS BASICALLY A ONE-PASS COMPILER, THE SECOND PASS BEING REQUIRED SOLELY BY THE BSS BINARY DECK FORMAT (I.E. PROGRAM CARD FIRST AND NO PROVISION FOR CHAINED BACKFILLS, ETC.).

REM

ALL SOURCE PROGRAM SYMBOLS AND LABELS ARE PLACED IN A SYMBOL TABLE (SYMTAB) WHOSE SIZE IS 4096 WORDS. THE METHOD OF ASSIGNING A SYMBOL OR LABEL TO A LOCATION IN SYMTAB IS SUCH THAT NO LENGTHY SEARCHES OR SORTS ARE NECESSARY. A SCRABBLED SORT TECHNIQUE IS USED, WHEREBY THE BITS IN A BCD SYMBOL OR LABEL ARE CONCENTRATED INTO A 12-BIT FIELD (2*12=4096). THIS IS DONE BY MULTIPLYING THE BCD SYMBOLIC NAME BY A SIMPLE STANDARD INTEGER, AND EXTRACTING A TWELVE-BIT FIELD FROM THE RESULT. THE POINTER THUS OBTAINED IS NORMALLY THE SYMTAB LOCATION.
OF THE SYMBOL OR LABEL. IF HOWEVER THE LOCATION IS ALREADY OCCUPIED BY A DIFFERENT SYMBOL (UNLIKELY IN MOST PROGRAMS), THEN THE SYMBOL IS ASSIGNED TO THE NEXT AVAILABLE LOCATION. TO LOCATE A SYMBOL IN SYMTAB, THE SAME COMPUTATION IS PERFORMED.

IF THE SYMBOL IS NOT AT THE COMPUTED LOCATION, A SEQUENTIAL SEARCH IS PERFORMED UNTIL EITHER THE ITEM IS LOCATED, OR AN EMPTY CELL IS ENCOUNTERED. (THE EMPTY CELL MEANS OF COURSE THAT THE SYMBOL IS NOT IN THE TABLE, AND IT MAY BE INSERTED IF DESIRED IN THE LOCATED EMPTY CELL).

THE TECHNIQUE IS VERY RAPID FOR SPARSELY-FILLED SYMBOL TABLES. IT BECOMES LESS EFFICIENT AS THE TABLE FILLS UP, SINCE MORE AND LONGER SEQUENTIAL SEARCHES WILL BE PERFORMED. HOWEVER, A VERY LARGE PROGRAM IS REQUIRED TO FILL THE TABLE MORE THAN SPARSELY.

EACH SYMTAB WORD HAS A COUNTERPART IN ANOTHER 4096-WORD TABLE, EQUIV. EQUIV CONSISTS PRIMARILY OF FLAG BITS DESCRIBING THE CORRESPONDING SYMTAB ENTRY, AND AN ADDRESS POINTER WHICH MAY BE USED TO DEFINE THE VALUE OF THE ENTITY, OR TO POINT INTO A GENERAL POOL AREA WHERE MOST ADDITIONAL INFORMATION IS STORED. THE FORMAT OF THE SYMTAB ENTRY IS DESCRIBED NOW IN GENERAL TERMS. THE ACTUAL BIT CONFIGURATIONS FOR THE VARIOUS FLAGS AND TERMS ARE FOUND IN THE REAR OF THE LISTING, UNDER "ABSOLUTE SYMBOL DEFINITIONS".

THE PREFIX OF AN EQUIV WORD HOLDS THE MODE OF THE ITEM. THE MODE IS REPRESENTED BY A THREE-BIT FIELD, AND NOT AS AN INDIVIDUAL BITS. THE FOUR MODES ARE MLABL (LABEL ON EXECUTABLE STATEMENT), MSTRG (FORMAT STRING LABEL), MINTG (INTEGER VARIABLE OR FUNCTION NAME), AND MREAL (REAL VARIABLE OR FUNCTION NAME).

BITS 3-14 HOLD BIT FLAGS DESCRIBING THE STATUS OF THE SYMTAB ITEM. NORMALLY EACH BIT STANDS FOR A SINGLE ITEM OF INFORMATION, HOWEVER, SOME BITS ARE USED FOR TWO OR MORE TASKS, IF NO CONFU- SION CAN RESULT. THE BIT FLAGS AND THEIR INTERPRETATIONS ARE...

BVARB... SYMHO HAS APPEARED AS A VARIABLE IN AN EXECUTABLE STATEMENT, AND MUST BE DEFINED IN THE PROGRAM.

BLHSX... SYMBOL OR LABEL HAS BEEN DEFINED (HAS A "LEFT-HAND VALUE").

BDOUB... SYMBOL WAS USED AT SOME POINT AS A DOUBLE PRECISION OR A COMPLEX VARIABLE, AND MUST HAVE DOUBLE STORAGE.

BANGT... SYMBOL IS A FORMAL SUBPROGRAM PARAMETER.

BAARY... SYMBOL APPEARS IN A DIMENSION STATEMENT. ADDRESS FIELD OF EQUV WORD POINTS TO THE COMMON/DIMENSION/EQUIVALENCE INFORMATION IN THE POOL AREA.

BCOMN... SYMBOL APPEARS IN A COMMON STATEMENT. ADDRESS FIELD OF EQUV WORD POINTS TO POOL AREA.

BREQUF... SYMBOL APPEARS IN ONE OR MORE EQUIVALENCE STATEMENTS. ADDRESS FIELD OF EQUV WORD POINTS TO POOL AREA.

BASSN... SYMBOL APPEARS AS THE VARIABLE IN AN ASSIGN OR ASSIGNED GO TO STATEMENT. ADDRESS FIELD OF EQUV WORD POINTS.

TO ASSIGN LISTS IN POOL AREA.

BINTF... SYMBOL IS USED AS AN ARITHMETIC STATEMENT FUNCTION NAME.

BEXTF... SYMBOL IS USED AS AN EXTERNAL NAME.

BLIB... EXTERNAL SYMBOL NAME IS CURRENTLY OF LIBRARY FUNCTION TYPE, I.E. ENDS IN 'F'.

BPATH... LABEL IS IN THE PATH OF FLOW OF THE PROGRAM, I.E. THERE IS A WAY FOR THE PROGRAM FLOW TO REACH THIS STATEMENT.
REM BITS 15-20 HOLD THE SUBPROGRAM ARGUMENT NUMBER, IF BARGT IS ON. FT002230
REM A MAXIMUM OF 64 FORMAL PARAMETERS CAN BE HANDLED. FT002240
REM BITS 21-35, THE ADDRESS FIELD, HOLD A SYMBOL OR LABEL DEFINITION, FT002250
REM OR IF CERTAIN EQUIV FLAGS ARE ON, POINT TO FURTHER INFORMATION FT002260
REM IN THE POOL AREA OR ELSEWHERE. FT002270
REM ASIDE FROM SYMAB AND EQUIV, THERE ARE VERY FEW FIXED-LENGTH FT002280
REM TABLES IN FASTRAN. THIS ALLOWS MAXIMUM FLEXIBILITY IN THE USE FT002290
REM OF AVAILABLE STORAGE. CONSEQUENTLY, THE LIKELIHOOD OF FT002300
REM 'OVERFLOW' OF THE COMPILER IS QUITE REMOTE. FT002310
REM A BRIEF SURVEY OF THE COMPILATION PROCESS IS GIVEN HERE. MORE FT002320
REM DETAILED INFORMATION IS TO BE FOUND IN DESCRIPITIONS OF THE FT002330
REM INDIVIDUAL SECTIONS IN THE BODY OF THE LISTING. THE COMPILATION FT002340
REM IS LOGICALLY DIVIDED INTO FIVE SECTIONS... INITIALIZATION, FT002350
REM PASS 1, POST PASS 1, PASS 2, AND POST PASS 2. FT002360
REM INITIALIZATION IS VERY SHORT, BEING CONCERNED WITH SETTING FT002370
REM VARIOUS CELLS TO THEIR INITIAL VALUES. FT002380
REM PASS 1 IS THE PRIMARY PASS OVER THE SOURCE DECK. THE SOURCE FT002390
REM PROGRAM IS LISTED, THE STATEMENTS ARE SCANNED AND INTERPRETED, FT002400
REM THE SYMBOL TABLE IS FORMED, VARIOUS LISTS ARE GENERATED, AND MANY FT002410
REM TYPES OF ERRORS ARE DETECTED. INTERMEDIATE OBJECT CODE IS FT002420
REM PREPARED, TO BE PROCESSED IN PASS 2 WHEN THE SCAN OF THE SOURCE FT002430
REM DECK IS COMPLETE. FT002440
REM POST PASS 1 PROCESSES THE SYMBOL TABLE, DETECTS ALL REMAINING FT002450
REM ERRORS (ERRORS INVOLVING RELATIONS BETWEEN THE VARIOUS FT002460
REM ITEMS), COMPUTES THE OBJECT PROGRAM LENGTH AND PUTS OUT THE FT002470
REM BINARY DECK PROGRAM CARD, DEFINES ALL VARIABLES. IF THIS PROGRAMFT002480
REM IS A SUBROUTINE OR FUNCTION, ITS PROLOGUE IS FORMED HERE. FT002490
REM IF AN OBJECT PROGRAM SYMBOL TABLE HAS BEEN REQUESTED, IT IS FT002500
REM FORMED AND PUT OUT PRIOR TO THE PROGRAM CARD. FT002510
REM POST PASS 1 IS ENTERED ONLY IF PASS 1 WAS FREE FROM ERRORS. FT002520
REM PASS 2 PROCESSES THE INTERMEDIATE OBJECT CODE, PRODUCING THE FT002530
REM THE FINAL CODE AND THE REMAINDER OF THE BINARY DECK. FT002540
REM POST PASS 2 PUTS OUT THE OBJECT PROGRAM STORAGE MAP, AND FT002550
REM TERMINATES THE COMPILATION. PASS 2 AND POST PASS 2 ARE FT002560
REM ENTERED ONLY FOR ERROR-FREE COMPILATIONS. FT002570
REM A FEW WORDS ON FASTRAN'S ERROR DETECTION PHILOSOPHY ARE IN ORDER. FT002580
REM MOST ERRORS, INCLUDING TYPOGRAPHICAL AND SYNTACTICAL ERRORS, ARE FT002590
REM CAUGHT DURING PASS 1; THE PRIMARY PASS OVER THE SYMBOLOGICAL DECK. FT002600
REM WE FEEL THAT DIAGNOSTIC MESSAGES ARE MOST INFORMATIVE WHEN THEY FT002610
REM ARE ISSUED ALONG WITH THE ERRONEOUS STATEMENT. THEREFORE, FT002620
REM MOST ERROR MESSAGES IN FASTRAN WILL APPEAR IN THE LISTING OF THE FT002630
REM SOURCE PROGRAM IMMEDIATELY FOLLOWING THE STATEMENT IN ERROR. FT002640
REM THERE ARE SOME ERRORS, NAMELY INVOLVING INTER-STATEMENT FT002650
REM RELATIONS, THAT CANNOT BE DETECTED UNTIL THE ENTIRE DECK HAS FT002660
REM BEEN READ. THESE THEN ARE GIVEN IMMEDIATELY FOLLOWING THE END FT002670
REM CARD. ALL ERROR MESSAGES ARE SIGNALLED BY AN OBVIOUS SERIES FT002680
REM OF ASTERISKS AT THE LEFT SIDE OF THE PAGE. FT002690
REM THE DIAGNOSTIC MESSAGES ARE AS INFORMATIVE AS POSSIBLE WITHOUT FT002700
REM UNDULY SACRIFICING COMPILATION SPEED, AND WITHOUT UNDULY CHANGING FT002710
THE CONFUSION OF THE PROGRAMMER, IN RESERVED WORD (FIXED FORMAT)
STATEMENTS, ONE CAN BE RATHER SPECIFIC WITHOUT RISK OF CONFUSING
THE SUBMITTER OF THE PROGRAM, HOWEVER, SPECIFIC DIAGNOSTICS ARE
IN GENERAL AVOIDED IN ARITHMETIC STATEMENTS, SINCE WE FEEL THAT
TOO OFTEN A SPECIFIC MESSAGE DOES NOT IN FACT IDENTIFY THE
PROGRAMMER'S DIFFICULTY, BUT INSTEAD CAUSES HIM MOMENTARY
PUZZLEMENT. THEREFORE, A COMMON ARITHMETIC STATEMENT ERROR
MESSAGE IS SIMPLY 'INVALID FORM OF STATEMENT'.

REM
IN PASSING WE NOTE THAT FASTRAN DETECTS MANY TYPES OF ERRORS THAT
IBM'S FORTRAN II WILL NOT CATCH. AMONG THESE ARE...
CONSISTENCY ERRORS IN ASSIGN/ASSIGNED-GO-TO USAGES, UNDEFINED
VARIABLES, RELATIVE CONSTANT ERRORS, ERRORS IN THE CONSISTENCY
OF ARITHMETIC STATEMENT FUNCTION USAGES, AND NONSENSE SUCH AS
'75 GO TO 75'.
REM

ALSO, A NUMBER OF VALID SITUATIONS INVOLVING FORTRAN II 'JOHNNY-
COME-LATELY' FEATURES, SUCH AS HOLLERITH LITERALS AND DOUBLE PRE-
CISION AND COMPLEX ARITHMETIC, ARE HANDLED CORRECTLY AND
CONSISTENTLY IN FASTRAN, WHILE FORTRAN II COMPiles ERRONEOUSLY
OR INCONSISTENTLY. (DETAILS WILL BE SUPPLIED UPON REQUEST).
THE LAYOUT OF CORE STORAGE IN THE FASTRAN COMPILER IS AS FOLLOWS.

THE CORE LOCATIONS OF THE VARIOUS SECTIONS ARE OF COURSE NOT SHOWN. MOST LOCATIONS ARE NOT IMPORTANT, AND ARE SUBJECT TO CHANGE.

SPACE 3

00000 REM MONITOR CELLS, ETC
0 0144 REM INPUT/OUTPUT SYSTEM (INDIANA UNIVERSITY I0S)
10000 REM BEGINNING OF FASTRAN COMPILER
           REM INITIALIZATION OF COMPILER
           REM ONE-TIME INITIALIZATION WHEN COMPILER ENTERS CORE
           REM INITIALIZATION FOR EACH COMPILATION
           REM INITIAL CARD PROCESSING
           REM BURSTING OF CARD
           REM STATEMENT LABEL AND MODE
           REM DETERMINATION OF STATEMENT TYPE
           REM STATEMENT TERMINATION ROUTINE
           REM PASS 1 STATEMENT PROCESSORS
           REM ARITHMETIC STATEMENT PROCESSING
           REM COMPILED Routines
           REM SCANNER
           REM SUBSCRIPT SCANNER AND COMPILER
           REM CODE KEY-WORD GENERATOR
           REM OBJECT PROGRAM CODE GENERATORS
           REM ARITHMETIC STATEMENT UTILITY ROUTINES
           REM PASS 1 UTILITY ROUTINES
           REM POST PASS 1
           REM ASSIGN VARIABLE AND LABEL CHECKING
           REM EQUIVALENT VARIABLE STORAGE ASSIGNMENT
           REM LENGTH OF PROLOGUE
           REM PRIMARY SYMBOL TABLE PASS
           REM COMMON VARIABLE STORAGE ASSIGNMENT
           REM LENGTH OF OBJECT PROGRAM
           REM OUTPUT OF ERRORS FOUND IN SYMBOL TABLE PASS
           REM OBJECT PROGRAM SYMBOL TABLE OUTPUT, IF REQUESTED
           REM BINARY DECK LABEL ROUTINE
           REM PROGRAM CARD OUTPUT
           REM TRANSFER VECTOR AND PROLOGUE OUTPUT ROUTINES
           REM POST PASS 1 UTILITY ROUTINES
           REM PASS 2... OUTPUT REMAINDER OF BINARY DECK
           REM TEXT
           REM CONSTANTS
           REM FORMAT STATEMENTS
           REM POST PASS 2... OBJECT PROGRAM MAP
           REM COMPILED TERMINATION ROUTINE
           REM UTILITY ROUTINES
           REM ERROR ROUTINE AND DIAGNOSTIC MESSAGES
           REM COMPILER TABLES
           REM FLAGS AND WORKING CELLS
           REM (END OF FASTRAN SYMBOLIC DECK)
           REM LITERALS
           REM ARRAY 'COLUMN' (666 CELLS LONG)
           REM POOL (ABOUT 5200 CELLS LONG, BACKWARD TO END OF COLUMN)
           REM EQUIV (4096 CELLS LONG, BACKWARD IN CORE)
           REM SYMTAB (4096 CELLS LONG, BACKWARD IN CORE)
THE FASTRAN COMPILER UTILIZES A GENERALIZED INPUT/OUTPUT SYSTEM DEVELOPED AT INDIANA UNIVERSITY...IOS. THIS I/O SYSTEM RESIDES IN LOW-CORE AND CONSISTS OF A SERIES OF ROUTINES FOR THE BUFFERED READING AND WRITING OF BLOCKED TAPE RECORDS, ALONG WITH NORMAL NON-DATA OPERATIONS (WGF, REW, ETC.). IN ADDITION, IOS HANDLES CERTAIN SPECIAL FUNCTIONS SUCH AS ON-LINE PRINTING, READING OF 716 CLOCK AND LOADING OF SYSTEM TAPE RECORDS. THE DATA CHANNEL TRAP FEATURE IS USED BY IOS TO CONTROL CHANNEL ACTIVITY.

CALLS TO IOS ARE EFFECTED BY I/O MACROS (E.G., BUFFER, READ, WRITE, BACKR, WEOF, ETC.). THESE MACROS ARE BUILT INTO THE MACHINE LANGUAGE ASSEMBLER EMPLOYED AT INDIANA UNIVERSITY (A SPECIALIZED VERSION OF IBM'S FAP ASSEMBLER).

IN ORDER THAT ANOTHER INSTALLATION CAN ASSEMBLE FASTRAN, THIS VERSION OF FASTRAN HAS BEEN MODIFIED. ALL IOS MACRO CALLS HAVE BEEN EXPANDED IN-LINE, AND THE MACRO NAMES NULLIFIED (E.G., 'READ OPSYN NULL'). THE PERTINENT TRANSFER ADDRESSES TO THE IOS ROUTINES ARE DECLARED IN BOOL STATEMENTS (E.G., 'READ BOOL 242'). THIS MATERIAL IS LOCATED AFTER THE MACRO DEFINITION SECTION OF THE COMPILER. ALL CALLS TO IOS ROUTINES ARE BRACKETED IN THE LISTING BY CARDS HAVING 'IOSXX' AND 'ENDXX' IN THE LOCATION FIELDS (XX IS A NUMBER FROM 01 TO 99). THESE SEQUENCES MAY EASILY BE FOUND BY LOCATING THE IOSXX ENTRIES IN THE SYMBOLIC REFERENCE TABLE.
THE FOLLOWING IS A BRIEF SUMMARY OF THE EXPANSION OF I/O MACROS.

ALL I/O MACROS ARE OF THE GENERAL FORM

SYMBOL NAME A(1), A(2), ..., A(N-1), A(N)

SPACE 2

THE MACHINE LANGUAGE EXPANSION OF THIS MACRO IS,

SYMBOL S1L IDX

TXL NAME,,(NORMAL RETURN)

XYZ A(1),,A(2)

XYZ A(N),,0 OR A(N-1),,A(N)

WHERE,

X = O IFF NOT LAST ARGUMENT PAIR

Y = O IFF ADDRESS ARGUMENT NOT INDIRECTLY Addressed

Z = O IFF DECREMENT ARGUMENT NOT INDIRECTLY Addressed

(NORMAL RETURN) = LOCATION + 1 OF LAST WORD OF MACRO

UNNECESSARY ARGUMENTS INCLUDED IN AN I/O MACRO ARE IGNORED

... WILL NOT CAUSE ASSEMBLY OR EXECUTION ERRORS.

FIRST BLANK IN ARGUMENT STRING TERMINATES THE VARIABLE FIELD.

AN ASTERISK (*) AS THE FIRST CHARACTER OF AN ARGUMENT

SPECIFIES INDIRECT ADDRESSING (ALL INDEX REGISTERS ASSUMED TO

CONTAIN ZERO).

ETC CARD(S) FOR CONTAINING LONG MACRO ARGUMENT LISTS (CARD

PRECEDING ETC CARD MUST HAVE COMMA AS LAST NON-BLANK CHAR.).
EJECT

THE FOLLOWING IS A BRIEF DESCRIPTION OF THE VARIOUS IOS MACROS

USED BY FASTRAN,

SPACE 2

BUFFERS LOC(1), NO(1), SIZE(1), TAPE(1),...

TO ASSIGN BUFFER SPACE TO VARIOUS TAPES. BUFFERS MUST BE

AT LEAST TWO WORDS LONGER THAN LARGEST PHYSICAL RECORD TO

BE READ (OR WRITTEN) WITH THESE BUFFERS. BUFFERS CAN

BE RE-DEFINED AT ANY TIME. TWO BUFFERS PER TAPE IS

GENERALLY OPTIMUM.

REM

XMODER LOC

CONTROL TO LOC IF READ GIVEN IN WRONG MODE. NO DATA

TRANSMISSION OCCURS.

REM

XRDN LOC

CONTROL TO LOC IF READ OR WRITE REDUNDANCY.

REM

XEOF LOC

CONTROL TO LOC IF END-OF-FILE READ.

REM

READ TAPE, MODE, LOC(1), CNT(1), ... LOC(N), CNT(N)

TO SCATTER READ ONE LOGICAL RECORD FROM TAPE. THIS IS A

TRANSMIT TYPE READ IN THAT DATA IS MOVED FROM THE BUFFER

TO THE LOCATIONS SPECIFIED IN THE READ MACRO.

REM

READ TAPE, MODE, L(LOC), L(COUNT)

SET DECREMENT OF LOC TO LOCATION OF FIRST WORD OF THE

NEXT LOGICAL RECORD. THIS IS A NON-TRANSMIT TYPE READ IN

THAT NO DATA IS MOVED FROM THE BUFFER. DECREMENT OF

LOCATION COUNT WILL CONTAIN NUMBER OF WORDS IN THE RECORD

JUST LOCATED.

REM

WRITE TAPE, MODE, LOC(1), CNT(1), ... LOC(N), CNT(N)

TO GATHER WRITE ONE LOGICAL RECORD ON TAPE. TRANSMIT

TYPE WRITE.

REM

BACKR TAPE, COUNT

TO BACKSPACE COUNT LOGICAL RECORDS ON TAPE.

REM

WEOF TAPE

TO WRITE AN END-OF-FILE ON TAPE (BUFFER(S) FLUSHED

AUTOMATICALLY).

REM

REWIND TAPE

TO REWIND TAPE.

REM

PRINT LOC(1), CNT(1), ... LOC(N), CNT(N)

EACH PAIR OF PARAMETERS Represents A LINE TO BE PRINTED

ON THE ON-LINE PRINTER (LOCATION AND COUNT). FIRST

CHARACTER USED AS SPACE CONTROL (A LA FORTRAN II).

REM

RCLK LOC

TO READ 716 PRINTER CLOCK INTO STORAGE, CONVERT TIME TO

SIX BCD CHARACTERS AND STORE THIS IN LOC AND LOGICAL

ACUMULATOR (LOC IS AN OPTIONAL ARGUMENT). SENSE EXIT 10 FOR

AND ECHO IMPULSES AT THE 716 ARE USED TO OBTAIN THIS CLOCK

READING.
**THE IOBASE PSEUDO-OPERATION APPEARING RIGHT AFTER THE MACRO**
**DEFINITIONS IN THE ASSEMBLY LISTING OF FASTRAN IS REQUIRED BY**
**THE ASSEMBLER TO PIN-POINT THE LOCATION OF A CRITICAL TRANSFER**
**VECTOR LOCATED IN IOS.**

**FASTRAN REQUIRES FOUR TAPES DURING NORMAL OPERATION AND BUFFER**
**SPACE FOR THESE TAPES IS DEFINED WITHIN FASTRAN. ALL TAPES HAVE**
**LOGICAL TAPE NUMBERS AND IOS IS RESPONSIBLE FOR ESTABLISHING**
**THE LOGICAL/PHYSICAL TAPE EQUIVALENCES. THE FOLLOWING TABLE**
**SUMMARIZES THE TAPES USED BY FASTRAN,**

<table>
<thead>
<tr>
<th>SYMBOLIC</th>
<th>LOGICAL</th>
<th>PHYSICAL</th>
<th>SYMBOLIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAME</td>
<td>NUMBER</td>
<td>NUMBER</td>
<td>BUFFER</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MINIAP</td>
<td>5</td>
<td>A2</td>
<td>BUFMIT</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MLSTAP</td>
<td>6</td>
<td>A3</td>
<td>BUFMLT</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSSBIN</td>
<td>7</td>
<td>B4</td>
<td>BUFBIN</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYSUT1</td>
<td>2</td>
<td>B2</td>
<td>BUFUT1</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SPACE** 2

**NOTE:**
**IF NOT RUN UNDER THE MONITOR, SYSSBIN = MLSTAP = LOGICAL TAPE**
**NUMBER 9 = PHYSICAL TAPE A5. THAT IS, THE BCD LIST OUTPUT AND**
**BINARY OBJECT DECK ARE PLACED ON TAPE 9 (= A5).**
* EJECT
* IF AN INSTALLATION IS INTERESTED IN USING A SYSTEM OTHER THAN
* IOS (E.G. IOCS) TO PERFORM THE I/O REQUIREMENTS OF THE COMPILER
* THE FOLLOWING STEPS COULD BE TAKEN,

REM

1. LOCATE ALL OF THE IOS MACROS IN FASTRAN. THIS CAN BE
   DONE BY USING THE SYMBOLIC REFERENCE TABLE ... ALL IOS
   MACROS ARE NAMED IOSXX (XX = 01, 02, ...).

REM

2. DELETE THOSE MACROS WHICH DO NOT APPLY TO THE NEW SYSTEM
   ... FOR EXAMPLE, THE XEOF FUNCTION MAY BE HANDLED BY EACH
   CALL TO THE READ ROUTINE IN THE NEW SYSTEM.

REM

3. REPLACE ALL OTHER MACROS WITH CALLS TO THE NEW SYSTEM TO
   PERFORM A GIVEN FUNCTION ...

REM

NOTE-

THE FIRST CARD OF EACH FASTRAN COMPILATION AND THE FIRST CARD
OF EACH STATEMENT THAT FOLLOWS A COMMENTS CARD, AN ASTERISK
(*) CARD OR A SUBTITLING CARD IS READ WITH A NORMAL (TRANSMIT-
TYPE) READ. ALL OTHER SOURCE CARDS ARE READ WITH A LOCATE
READ MACRO. THIS, OF COURSE, IS DESIGNED TO ELIMINATE THE
BUFFER-TO-WORK AREA MOVE REQUIRED BY THE TRANSMIT-TYPE READ.

SPACE 3

A COPY OF AN IOS USER'S MANUAL WILL BE SENT UPON REQUEST.
INTRODUCTION -- FASTRAN/FORTRAN INCOMPATIBILITIES

FASTRAN/FORTRAN II INCOMPATIBILITIES.

SPACE 4

THERE ARE SOME FEATURES OF FASTRAN WHICH ARE INCOMPATIBLE WITH

(FOR DIFFERENT FROM) THE IBM FORTRAN II. THESE ARE LISTED

AND DISCUSSED BELOW. MUST ARE RATHER INCONSEQUENTIAL.

HOWEVER, SOME ARE OF CONSIDERABLE SIGNIFICANCE, AND THE

INCOMPATIBILITY WAS INTENTIONALLY CREATED. THE LIST BELOW

IS ROUGHLY IN THE ORDER OF DECREASING IMPORTANCE.

REM
REM
REM

THERE ARE NO RELATIVE CONSTANTS AT ALL IN FASTRAN. ALL VARIABLES

HAVE A STORAGE ASSIGNMENT, AND STORAGE IS ALWAYS CORRECT AND

UP-TO-DATE. THIS IS A HIGHLY DESIRABLE FEATURE, IN OUR OPINION,

AS RELATIVE CONSTANTS ARE A CONSIDERABLE NUISANCE TO THE

PROGRAMMER. SOME RELCON 'ERRORS' IN FORTRAN II CAN BE SO

WELL DISGUISED THAT MANY HOURS OR DAYS ARE SPENT IN DEBUGGING.

THE 'INCORRECT' PROGRAM. ONE SIGNIFICANT RESULT OF DISPENSING

WITH THE RELCON PROBLEM IS THAT IN FASTRAN, A DO-TYPE INDEX

IN AN I/O LIST MAY NOT BE THE SAME VARIABLE NAME AS A

CURRENT DO-STATEMENT INDEX, SINCE THE SAME STORAGE LOCATION

IS ASSIGNED TO BOTH. THIS IS CAUGHT IN FASTRAN AS AN ERROR...

NO INCORRECT CODE IS PRODUCED.

REM
REM

FIXED POINT EXPRESSIONS INVOLVING A SUCCESSION OF MULTIPLY

AND DIVIDE OPERATIONS (E.G. I*J/K OR I*J/K*L) ARE NOT

REARRANGED, AS IS DONE IN 7090 FORTRAN II. IN FASTRAN,

THE CODE GENERATED IS EFFECTIVELY THAT DUE TO A LEFT-TO-RIGHT

SCAN OF THE EXPRESSION.

REM
REM

UNDEFINED VARIABLES AND STATEMENT LABELS IN THE SOURCE

PROGRAM CAUSE AN ERROR MESSAGE. A VARIABLE USED IN THE PROGRAM

IS UNDEFINED IF IT DOES NOT HAVE ONE OR MORE OF THE FOLLOWING

PROPERTIES...

IN A COMMON STATEMENT, EQUIVALENCED TO A COMMON VARIABLE,

IN AN INPUT LIST, ON THE LEFT-HAND SIDE OF AN EQUAL SIGN,

IN A SUBROUTINE OR FUNCTION STATEMENT ARGUMENT LIST, IN THE

ARGUMENT LIST OF A CALL STATEMENT OR A CALL TO AN EXTERNAL

FUNCTION (FORTRAN OR LIBRARY), USED AS A DO-INDEX OR AN

INDEX IN AN INPUT/OUTPUT LIST, USED AS AN ASSIGN VARIABLE.

STATEMENT LABELS ARE UNDEFINED IF THEY DO NOT APPEAR IN THE

LABEL FIELD OF AN EXECUTABLE STATEMENT. (ALSO, OF COURSE ALL

FORMAT STATEMENT LABELS MUST BE DEFINED).

REM
REM

NO LIST OR LIST8 FEATURE IS IMPLEMENTED.

REM
REM

IMPLIED MULTIPLICATION IS NOT RECOGNIZED IN FASTRAN.

REM
REM

ALL FORMS OF LITERALS WORK. TERMINAL BLANKS ARE OKAY.

LITERALS IN BOOLEAN STATEMENTS WORK IF THE CHARACTER COUNT IS

IN OCTAL. (LITERALS IN BOOLEAN STATEMENTS DO NOT WORK AT ALL

IN IBM FORTRAN II).

REM
REM

ERROR MESSAGES ARE ISSUED IN-LINE WHENEVER POSSIBLE. I.E.

THEY ARE ISSUED IMMEDIATELY WHEN THE ERROR IS DIAGNOSED.

REM
REM

STATEMENTS LIKE A=0, A=0.3, I=0, ETC. CAUSE STZ INSTRUCTIONS
**TO BE GENERATED.**

**SOURCE PROGRAM DOUBLE PRECISION FLOATING POINT CONSTANTS MAY BE MORE ACCURATE THAN IN FORTRAN II (BY ONE BINARY BIT), DUE TO JUDICIOUS Rounding IN THE FASTRAN COMPILER NUMBER PROCESSOR.**

**THE COLLECTION OF ASSIGN AND ASSIGNED GO TO STATEMENTS IN THE SOURCE PROGRAM IS CHECKED FOR CONSISTENCY. ASSIGN STATEMENT, VICE VERSA. THE STATEMENT LABEL IN EACH ASSIGN STATEMENT MUST APPEAR IN THE LABEL LIST OF AN ASSIGNED GO TO FOR THE SAME ASSIGN VARIABLE.**

**THE VALUE OF A VARIABLE SUBSCRIPT, EXCLUSIVE OF ITS ADDEND, NEED NOT BE LESS THAN OR EQUAL TO THE CORRESPONDING ARRAY DIMENSION, AS IS REQUIRED IN FORTRAN II FOR CORRECT CODE.**

**F-CARDS MUST PRECEDE IN THE SOURCE PROGRAM THE USAGES OF THEIR EXTERNAL NAMES.**

**THE STANDARD ERROR OPTION IS NOT AVAILABLE IN FASTRAN COMPILED PROGRAMS.**

**NO CHECKING FOR VALIDITY OF FORMAT STATEMENTS IS DONE IN FASTRAN.**

**SINCE HPR MAY NOT SAFELY BE USED TO HALT THE COMPUTER WHEN THE DATA CHANNEL TRAP IS BEING USED, HTP MUST BE USED INSTEAD IN OBJECT PROGRAM CODE FOR THE 'STOP N' AND 'PAUSE N' STATEMENTS.**

**THE STORAGE REGISTER ADDRESS IS UNAVAILABLE FOR THE CONSTANT N. THEREFORE, FASTRAN PLACES THE CONSTANT N (OR ZERO) IN THE ADDRESS OF BOTH THE ACCUMULATOR AND THE MQ.**

**THE COMPILER USES THE FACT THAT THE DOUBLE PRECISION ADD, SUBTRACT, MULTIPLY, AND DIVIDE LIBRARY ROUTINES LEAVE THE RESULT IN BOTH THE PSEUDO ACCUMULATOR (OR PSEUDO MQ) AND ALSO IN THE ACTUAL AC+MQ. (THIS IS EASILY ALTERED, IF AN INSTALLATION CANNOT ACCEPT THIS OBJECT PROGRAM CODE SHORTCUT).**

**SUCH INCONSISTENCIES AS '5 GO TO 5' OR '5 IF (A) 5,6,7' ARE CAUGHT AS ERRORS IN FASTRAN. IN OTHER WORDS, A STATEMENT MAY NOT REFER TO ITSELF.**

**UNDER FASTRAN, THERE IS NO SCAN FOR MONITOR * CONTROL CARDS WITHIN A FORTRAN PROGRAM, AND THUS THEY ARE NOT LISTED ON-LINE OR INTERPRETED. ( * CARDS ARE LEGAL IN A FASTRAN PROGRAM. THEY ARE TREATED AS COMMENTS).**

**COMMENT AND ASTERISK CARDS MAY NOT IMMEDIATELY PRECEDE A STATEMENT CONTINUATION CARD. BLANK CARDS ARE ACCEPTABLE ANYWHERE.**

**THE DOUBLE PRECISION AND COMPLEX BUILT-IN FUNCTION 'FIXF' IS NOT ALLOWED (ERROR MESSAGE IF USED). THIS IS BECAUSE THE IBM FORTRAN II COMPILER GENERATES INCORRECT CODE FOR THIS FUNCTION.**

**THE LIBRARY FUNCTION 'XLOC' IS BUILT-IN IN FASTRAN.**

**THE USE OF A PHYSICAL TAPE DESIGNATION SUCH AS A4 IN CALL CHAIN (3,A4) IS NOT ALLOWED. HOWEVER, THE MONITOR CALL CHAIN (3,A4) IS NOT ALLOWED. HOWEVER, THE MONITOR CALL CHAIN (3,A4) IS NOT ALLOWED.**
CONTROL CARD ** CHAIN (3,A4)** IS PROCESSED BY THE MONITOR, AND THUS WOULD BE ACCEPTABLE.

THE MONITOR END CARD OPTIONS MAY APPEAR ON THE END CARD, BUT THEY ARE IGNORED BY FATRAN.

FREQUENCY STATEMENTS ARE IGNORED.

A FUNCTION STATEMENT MUST HAVE ARGUMENTS, AND THE FUNCTION NAME MUST BE DEFINED WITHIN THE FUNCTION SUBPROGRAM.

FATRAN ALLOWS STATEMENT LABELS FROM 1 THROUGH 99999.

THE READ DRUM AND WRITE DRUM STATEMENTS ARE NOT ACTIVATED IN THIS VERSION OF FATRAN.
ADDITIONAL STATEMENTS ALLOWED IN FASTRAN

* SPACE 3
* IN ADDITION TO THE INCOMPATIBILITIES, THERE ARE SOME EXTENSIONS
* TO THE LANGUAGE AVAILABLE FOR USE.
* SPACE 2

**** THE FASTRAN STATEMENT 'FLOATING ROUND' WILL CAUSE ROUNDED
* OF ALL SUBSEQUENT FASTRAN-COMPILED FLOATING POINT ARITHMETIC.
* NOTE THAT THIS IMPROVES ACCURACY, AND INCREASES THE LENGTH OF
* THE OBJECT PROGRAM. THE FEATURE MAY BE TURNED OFF BY ISSUING
* A 'FLOATING ROUND OFF' STATEMENT.
* REM

**** A PAGE SUBTITLING OPTION IS ALLOWED. A CARD WITH A $-SIGN
* IN COLUMN 1 INDICATES A PAGE SUBTITLE. THIS CARD CAUSES

A PAGE EJECT, AND SUBTITLING COMMENCES ON THE NEXT PAGE, COLUMNS
2-72 BEING USED AS THE SUBTITLE. THIS SUBTITLE IS IN EFFECT
UNTIL SUERSEDEED BY ANOTHER $-CARD. TO SUSPEND SUBTITLING,
USE A BLANK $-CARD. (NOTE THAT THE MAIN PAGE TITLE MAY NOT
BE ALTERED).
REM

**** THE CHARACTER $ IN COLUMN 1 OR THE WORD EJECT IN COLUMNS 1-5
* CAUSES A PAGE EJECT BEFORE PRINTING THE NEXT LINE OF THE
* SOURCE PROGRAM LISTING. CONSECUTIVE $ OR EJECT CARDS GIVE ONLY
* ONE EJECT.
* REM

**** SPACING IN THE SOURCE PROGRAM LISTING MAY BE EFFECTED BY A
* CARD HAVING AN $ IN COLUMN 1 OR THE WORD SPACE IN COLUMNS 1-5.
* AN INTEGER MUST APPEAR BEGINNING IN COLUMN 7, DENOTING THE
* NUMBER OF LINES TO BE SKIPPED. IF THE SPACING CAUSES A PAGE
**EJECT, SPACING IS NOT CONTINUED ON THE NEW PAGE. TO SKIP ONE LINE, USE A COMPLETELY BLANK CARD.**

**SPACE CARDS IMMEDIATELY FOLLOWING AN EJECT CARD OR A SUBTITLE CARD ARE IGNORED.**

REM

**** AN OBJECT PROGRAM SYMBOL TABLE MAY BE REQUESTED BY INCLUDING A FASTRAN STATEMENT "SYMBOL TABLE" ANYWHERE IN THE SOURCE DECK. THIS STATEMENT TURNS ON THE SYMBOL TABLE BIT (BIT 31) IN THE FMS MONITOR FLAG BOX AT LOCATION 18. FASTRAN LATER INTERROGATES THIS BIT TO DETERMINE IF A SYMBOL TABLE SHOULD BE OUTPUT. NOTE THAT THIS IS ALSO AN FMS MONITOR FUNCTION (* SYMBOL TABLE). REM

**** AN ON-LINE LISTING MAY BE OBTAINED BY ISSUING A FASTRAN STATEMENT "ONLINE". OUTPUT WILL THEN COMMENCE BOTH ON-LINE AND OFF-LINE AND CONTINUE UNTIL THE END OF THE COMPILATION OR UNTIL THE ON-LINE FEATURE IS TERMINATED BY AN "OFFLINE" FASTRAN STATEMENT. NOTE THAT ANY NORMAL ON-LINE COMMENTARY (SUCH AS "SOURCE PROGRAM ERROR") WILL APPEAR INTERSPERSED AMONG THE LINES OF THE SOURCE PROGRAM LISTING.
INTRODUCTION -- OBJECT PROGRAM STORAGE LAYOUT

OBJECT PROGRAM STORAGE LAYOUT

THE DETAILED ORDER OF THE VARIOUS SECTIONS OF THE OBJECT PROGRAM RESULTING FROM A FASTRAN COMPILATION IS AS FOLLOWS (BEGINNING AT LOCATION ZERO)... REM

TRANSFER VECTOR

LIBRARY REFERENCES
NON-LIBRARY REFERENCES
PROLOGUE (FOR SUBPROGRAMS ONLY)
INDEX REGISTER RESTORING INSTRUCTIONS
FINAL SUBPROGRAM EXIT
INDEX REGISTER SAVING INSTRUCTIONS

(THIS IS THE SUBPROGRAM'S ENTRY POINT)
BODY OF PROLOGUE (SETTING OF ADDRESS IN TEXT)
TRANSFER AROUND ARITH. STMNT. FCNS. IF THERE ARE ANY ARITHMETIC STATEMENT FUNCTIONS
(IN ORDER OF THEIR APPEARANCE IN THE SOURCE PROGRAM)
FLOATING POINT TRAP INITIALIZATION (FOR MAIN PROGRAM ONLY)
CAL (FPT)
SLW 0
STZ 77462
TEXT (CODE GENERATED FOR EXECUTABLE STATEMENTS)
CONSTANT STORAGE
STRING (FORMAT) STORAGE
WORKING STORAGE
LOW-CORE (NON-COMMON) VARIABLE STORAGE
NON-DIMENSIONED AND DIMENSIONED VARIABLES INTERMIXED REM
REM *
REM *
VAST EMPTINESS
REM *
REM *
COMMON STORAGE
<table>
<thead>
<tr>
<th>TTL FMS II EDITOR CONTROL CARD NO. 1</th>
<th>FT007760</th>
</tr>
</thead>
<tbody>
<tr>
<td>THE FOLLOWING &quot;INSTRUCTIONS&quot; WILL CAUSE GENERATION OF THE PROGRAM</td>
<td>FT007770</td>
</tr>
<tr>
<td>CARD REQUIRED BY THE FMS-II EDITOR TO EDIT FASTRAN INTO THE IOS/FMS-II SYSTEM</td>
<td>FT007780</td>
</tr>
<tr>
<td>REM</td>
<td>FT007790</td>
</tr>
<tr>
<td>FASTRAN IS TOO BIG TO BE ACCEPTED AS ONE RECORD BY THE EDITOR. THEREFORE THERE IS ANOTHER CONTROL CARD GENERATED SOMEWHERE BEFORE LOCATION 30000</td>
<td>FT007800</td>
</tr>
<tr>
<td>REM</td>
<td>FT007810</td>
</tr>
<tr>
<td>THESE TWO CARDS MUST BE PULLED OFF IF FASTRAN IS NOT BEING RUN UNDER THIS MONITOR SYSTEM</td>
<td>FT007820</td>
</tr>
<tr>
<td>SPACE 3</td>
<td>FT007830</td>
</tr>
<tr>
<td>ABS</td>
<td>FT007840</td>
</tr>
<tr>
<td>5</td>
<td>FT007850</td>
</tr>
<tr>
<td>IORT</td>
<td>FT007860</td>
</tr>
<tr>
<td>PRGORG,,BEGRC2-PRGORG</td>
<td>FT007870</td>
</tr>
<tr>
<td>TXI</td>
<td>FT007880</td>
</tr>
<tr>
<td>BEGINY,,30</td>
<td>FT007890</td>
</tr>
<tr>
<td>ENTRY PT,,PROGRAM NUMBER</td>
<td>FT007900</td>
</tr>
<tr>
<td>ABS</td>
<td>FT007910</td>
</tr>
<tr>
<td>FT007920</td>
<td></td>
</tr>
</tbody>
</table>
* TTL MACRO DEFINITIONS

* REM SPACE 9
  * * * * * * * * *

* REM FASTRAN COMPILER

* REM RESEARCH COMPUTING CENTER

* REM INDIANA UNIVERSITY

* REM BLOOMINGTON, INDIANA
  * * * * * * * * *

* REM MACRO DEFINITIONS

* * * * * * * * *

* REM MACRO DEFINITIONS

* * * * * * * * *
MACROS USED TO FORM AND COMMUNICATE OBJECT-TIME INSTRUCTION
INFORMATION TO CITBLD.

REM

CITBLD MACRO A
TSX CITBLD,4,A
CITBLD END
SPACE 2

CODEN MACRO A
TSX CITBLD,4,N
CAL A
CODEN END
SPACE 2

CODEP MACRO A
CAL A
ACL PCOUNT
TEX CITBLD,4,P
CODEP END
SPACE 2

CODETV MACRO A
TEX OUTPTV,4
SUBROUTINE NAME.
PZE A
CODETV END
SPACE 2

CODEC MACRO A, B, C
CAL A
IFF 1,C,XRFLAG
ORS XRFLAG
TEX CITBLD,4,B
CODEC END
SPACE 2

CODEO MACRO A, B, C
ORA A
IFF 1,C,XRFLAG
ORS XRFLAG
TEX CITBLD,4,B
CODEO END
SPACE 2

CODECO MACRO A, B, C, D
CAL A
ORA B
IFF 1,D,XRFLAG
ORS XRFLAG
TEX CITBLD,4,C
CODECO END
SPACE 2

MACRO FOR COMMUNICATING WITH THE GENERAL CODE ROUTINE.
REM

CODE MACRO OP,A
TSX CODE,4
PZE OP,A
CODE END
SPACE 2

SPECIAL PURPOSE MACROS TO COMMUNICATE WITH ROUTINES

WHICH WILL USE THE GENERAL CODE ROUTINE.

THESE MACROS CAUSE THE GENERATION OF TWO OBJECT-TIME
INSTRUCTIONS. THEY ARE USED BY THE DOUBLE PRECISION AND
COMPLEX ARITHMETIC SECTIONS OF THE COMPILER.
REM

CUDLD MACRO A
MACRO TO LOAD THE AC AND MQ WITH OPERAND
TSX   CODLD,4
PZE   CLA,A

CODLD END
SPACE 2

CODLAC MACRO A
TSX CODLAC,4
PZE CLA,A

CODLAC END
SPACE 2

COSTO MACRO A
TSX COSTO,4
PZE STO,A

COSTO END
SPACE 2

CODSTR MACRO A
TSX CODSTR,4
PZE STR,A

CODSTR END
SPACE 2

COSTZ MACRO A
TSX COSTZ,4
PZE STZ,A

CODSTZ END
SPACE 2

SETUP MACRO A
TSX SETUP,4
PZE ('A')

SETUP END
SPACE 2

CODSUB MACRO A
TSX CODSUB,4
PZE CLS,A

CODSUB END
SPACE 2

CODLMQ MACRO A
TSX CODLMQ,4
PZE CLA,A

CODLMQ END
SPACE 2

* MACRO TO RECORD THE MODE AND REGISTER LOCATION OF A TEMPORARY
* RESULT DURING GENERATION OF ARITHMETIC CODE.
* FIRST ARGUMENT IS THE MODE (REAL, INTEGER). SECOND ARGUMENT
* IS THE LOCATION (AC, MQ, PAC, PMQ). IF THIRD ARGUMENT IS
* PRESENT, FLOATING ROUNDING IS PERFORMED BY ENTERING THE
* ROUNDX ROUTINE PRIOR TO ENTERING THE RESULT ROUTINE.
* REM
RESULT MACRO A,B,C
IFF 1,C
TSX RESULT,4,A+8+B
IFF 0,C
TSX ROUNDX,4,A+8+B

RESULT END
SPACE 2

* DEBUGGING MACRO. PRINTS TYPE OF STATEMENT PROCESSOR IDENTIFIED.
* REM
XREM MACRO A
TRA 'L1
BCI 'A'
TSX LIST,4
PZE 'L1,10
* REMK END
* SPACE 2
* DEBUGGING MACRO DISABLED IN THIS VERSION. TO MAKE THIS
* MACRO FUNCTIONAL, REPLACE IT WITH A 'REMARK OPSYN XREM' CARD.
* REM
* REMK MACRO A
* EQU *
* REMK END
* SPACE 2
* MACRO TO FORM THE PROCESSOR KEY WORD. GENERATED IN THE
* RESERVED WORD TEST. B, C, AND D ARE ONE-BIT FLAGS.
* A GIVES THE ENTRY POINT TO THE PROCESSOR.
* REM
* SPRSW MACRO A,B,C,D,E
* CAL ++2
* TRA PROCX1
* SPRSW END
* VFD 1/B, 1/C, 1/D, 015/E, 3/15/A
* SPACE 2
* MACRO TO RECORD A SOURCE PROGRAM ERROR. COMMUNICATION WITH THE
* ERROR ROUTINE IS VIA AN STR TRAP. ADDRESS OF THE STR HAS NAME
* OF THE ERROR MESSAGE, AND DECREMENT HAS THE POINT TO WHICH
* CONTROL IS TO BE RETURNED WHEN THE ERROR ROUTINE IS FINISHED.
* REM
* ERROR MACRO A,B
* STR FORM*A,B
* ERROR END
* SPACE 2
* MACRO TO RECORD A MACHINE ERROR. USES THE ERROR ROUTINE.
* REM
* MACER MACRO STR MCR
* MACER END
* SPACE 2
* MACRO TO TEST THE NOCODE FLAG. NOCODE IS TURNED ON BY THE
* ERROR ROUTINE TO INDICATE A SOURCE PROGRAM ERROR. IF AN ERROR
* HAS BEEN ENCOUNTERED, CONTROL LEAVES THE CURRENT PROCESSOR
* AND GOES TO SKEND.
* REM
* NOCODE MACRO
* ZET NOCODE
* TRA SKEND
* NOCODE END
* SPACE 2
* POST PASS 1 ERROR MACRO USED TO ACCUMULATE SYMBOLS IN ERROR
* FOR LATER PRINTING.
* REM
* ERRORY MACRO FLAG
* TSX ERRORY,4,FLAG
* ERRORY END
* SPACE 2
* MACRO TO SAVE INDEX REGISTERS.
* REM
* SAVE MACRO Q
* SXA Q,4
* SXA Q+1,2
* SXA Q+2,1
* SAVE END
* SPACE 2
* MACRO TO RESTORE INDEX REGISTERS
RESTOR MACRO
 请求这个页码。

AXT **,4
AXI **,2
AXT **,1

RESTOR END

* MACRO TO COMMUNICATE WITH THE SKIP ROUTINE. ARGUMENT IS A
* SYMBOL REPRESENTING ONE BCD CHARACTER (SIX BITS). RETURN FROM
* SKIP IS TO 1,4 IF NEXT CHARACTER IN COLUMN IS NOT EQUAL TO
* THE ARGUMENT, AND TO 2,4 IF THE NEXT CHARACTER IS EQUAL TO
* THE ARGUMENT.

REM

SKIP MACRO A
TSX SKIP,4,A

SKIP END

* MACRO TO COMMUNICATE WITH THE SKIP1 ROUTINE, SAME AS SKIP
* MACRO EXCEPT RETURNS ARE REVERSED.

REM

SKIP1 MACRO A
TSX SKIP1,4,A

SKIP1 END

* MACRO USED TO BUILD THE OPERATOR RANK TABLE.

REM

RANK MACRO DP,RANK,OPKEY,TYPE,ENTRY

DP EQU $-RANK+1

OPKEY VFD 7/OP,7/RANK,1/TTYPE,2/1,4/15/ENTRY

RANK END

* MACROS USED TO BUILD THE GENERATOR KEY TABLES.

REM

KEY MACRO A,B,OP,D,E,F

VFD 7/OP,3/A,2/B,3/D,2/E,4/,15/F

KEY END

SPACE 2

KEY1 MACRO OP,M,SUB
KEY 0,0,OP,M,VS,SUB*1
KEY 0,0,OP,M,AC,SUB*2
KEY 0,0,OP,M,MQ,SUB*3

KEY1 END

SPACE 2

KEY2 MACRO M1,OP,M2,SUB
KEY M1,VS,OP,M2,VS,SUB*1
KEY M1,VS,OP,M2,AC,SUB*2
KEY M1,VS,OP,M2,MQ,SUB*3
KEY M1,AC,OP,M2,VS,SUB*4
KEY M1,MQ,OP,M2,VS,SUB*5

KEY2 END

SPACE 2

KEY3 MACRO M1,OP,M2,SUB
KEY M1,VS,OP,M2,VS,SUB*1
KEY M1,VS,OP,M2,AC,SUB*2
KEY M1,VS,OP,M2,MQ,SUB*3

KEY3 END

SPACE 2

KEY4 MACRO OP,M,SUB
KEY 0,0,OP,M,VS,SUB*1
KEY 0,0,OP,M,AC,SUB*2
<table>
<thead>
<tr>
<th>KEY</th>
<th>END</th>
<th>2</th>
<th>FT010520</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPACE</td>
<td></td>
<td></td>
<td>FT010530</td>
</tr>
<tr>
<td>*</td>
<td>MACRO</td>
<td>USED TO BUILD THE OPEN FUNCTION KEY TABLE.</td>
<td>FT010540</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FT010550</td>
</tr>
<tr>
<td>TAB3</td>
<td>MACRO</td>
<td>A</td>
<td>FT010560</td>
</tr>
<tr>
<td></td>
<td>EQU</td>
<td>*LIBTAB+64</td>
<td>FT010570</td>
</tr>
<tr>
<td></td>
<td>VFD</td>
<td>7/A,011/BINTF+BOPNF</td>
<td>FT010580</td>
</tr>
<tr>
<td></td>
<td>BCI</td>
<td>1,A</td>
<td>FT010590</td>
</tr>
<tr>
<td>TAB3</td>
<td>END</td>
<td></td>
<td>FT010600</td>
</tr>
<tr>
<td>SPACE</td>
<td>2</td>
<td></td>
<td>FT010610</td>
</tr>
</tbody>
</table>
* THE FOLLOWING MATERIAL IS NEEDED TO IMPLEMENT THIS VERSION OF
* FASTTRAN, WHICH DOES NOT DEPEND ON INDIANA UNIVERSITY'S VERSION
* OF THE FAP ASSEMBLER FOR THE MEANING OF THE IOS CALLS.
* SEE THE INTRODUCTION FOR MORE DETAILS ON THE INPUT/OUTPUT
* SCHEME IN FASTTRAN.

** DECLARE ALL IU-FAP IOS OPERATIONS TO BE NULL

```
IOBASE  OPSYN  NULL
READ   OPSYN  NULL
WRITE  OPSYN  NULL
REWIND OPSYN  NULL
WEOF   OPSYN  NULL
BACKR  OPSYN  NULL
PRINT  OPSYN  NULL
RCLK   OPSYN  NULL
XEOF   OPSYN  NULL
XRN    OPSYN  NULL
XMDEER OPSYN  NULL
BUFFER OPSYN  NULL
CLOSE  OPSYN  NULL
LOAD   OPSYN  NULL
LREAD  OPSYN  NULL
```

* DEFINE THE LOCATIONS OF THE ROUTINES IN IOS

```
REX  BOOL  241
READ  BOOL  242
WRITE  BOOL  243
REWIND  BOOL  244
WEOF  BOOL  246
BACKR  BOOL  247
PRINT  BOOL  253
RCLK  BOOL  256
XEOF  BOOL  257
XRN  BOOL  260
XMDEER  BOOL  262
BUFFER  BOOL  263
CLOSE  BOOL  264
LOAD  BOOL  266
LREAD  BOOL  303
```

* THE FOLLOWING IOS CALL IS NEEDED IN THE IU VERSION OF FASTTRAN
* TO LOCATE THE ORIGIN OF IOS MACROS FOR FAP.

```
IOBASE  161  BASE FOR ALL IOS MACROS
```
<table>
<thead>
<tr>
<th>TTL</th>
<th>COMPILER ORIGIN, ETC.</th>
<th>FT01090</th>
<th>PAGE 42</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRGORG</td>
<td>PRGORG</td>
<td>FT01100</td>
<td></td>
</tr>
<tr>
<td>ORG</td>
<td>PRGORG</td>
<td>FT01110</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FT01120</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FT01130</td>
<td></td>
</tr>
<tr>
<td>DEFINE TSX INSTRUCTION TO ALLOW A DECREMENT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TSX</td>
<td>OPD 007415160002</td>
<td>FT01140</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>FT01150</td>
<td></td>
</tr>
</tbody>
</table>
* Manual Entry for Error Stop

* In Case of an Unexpected Stop During Debugging of the Compiler, the Operator May Transfer Manually to Location 10000 Octal.

* The Symbol Table Will Be Dumped Onto the Output Tape, If a Space is Requested (If LSTSYM is On). Then an On-Line Message Will Tell the Operator to Take a Core Dump.

WHAMMY

ZEF  LSTSYM  List Symbol Table Only If Requested

IOS01  EQU  *

WEOF  *MLSTAP  Entry in Case of Execution Halt To

STL  IOEX

BRA  WEOF,,+2

MTW  MLSTAP

END01  REM  *

IOS02  EQU  *

BACKR  *MLSTAP,,1  Back Over the E-O-F

STL  IOEX

BRA  BACKR,,+2

MTW  MLSTAP,,1

END02  REM  *

IOS03  EQU  *

PRINT  DUMPRQ,,4,,DUMPRQ+4,,DUMPRQ+4,,DUMPRQ+4,,DUMPRQ+4,,

STL  IOEX

BRA  PRINT,,+5

PZE  DUMPRQ,,4

PZE  DUMPRQ+4,,1

PZE  DUMPRQ+4,,1

MZE  DUMPRQ+4,,1

END03  REM  *

HTR  *

DUMPRQ  BCI  4, Now Take Core Dump

BCI  1,0

FT011160
FT011170
FT011180
FT011190
FT011200
FT011210
FT011220
FT011230
FT011240
FT011250
FT011260
FT011270
FT011280
FT011290
FT011300
FT011310
FT011320
FT011330
FT011340
FT011350
FT011360
FT011370
FT011380
FT011390
FT011400
FT011410
FT011420
FT011430
FT011440
FT011450
FT011460
FT011470
FT011480
TTL  INITIALIZATION
*  THE FOLLOWING ARE THE FIRST INSTRUCTIONS EXECUTED AFTER FASTRAN
*  IS LOADED AND ARE EXECUTED ONE TIME ONLY. THEY HAVE THE
*  FOLLOWING FOUR FUNCTIONS,
*  1. SORT GENERATOR KEY TABLES FOR,
   *  A. REAL AND INTEGER
   *  B. BOOLEAN
   *  C. DOUBLE PRECISION
   *  D. COMPLEX
   REM
*  2. DEFINE BUFFERS FOR,
   *  A. MINOTAP = SYSTEM INPUT TAPE
   *  B. MLSSTAP = SYSTEM OUTPUT TAPE
   *  C. SYSBIN = SYSTEM (BINARY) LOAD TAPE
   *  D. SYSUT1 = SYSTEM SCRATCH TAPE NO. 1
   REM
*  3. DEFINE MODE ERROR AND END-OF-FILE RETURNS
   REM
*  4. SET LOCATION 2 = TRA ERROR AND SET LOCATION 8 FOR
   FLOATING POINT TRAP.
   SPACE 2
*  ON COMPLETION OF THIS PRE-INITIALIZATION SECTION, CONTROL IS
*  GIVEN TO THE MAIN INITIALIZATION SECTION TO INITIALIZE FOR
*  THIS COMPILATION.
   SPACE 3
   REM
   REM
   REM
   REM
ICS04  EQU *
BEGINY  LOAD 31,31
BEGINY  STL  IDEX
BEGINY  BRA  LOAD,,**3
       PZE  MZE  31
END04  REM
BEGINZ  CAL =6
       SLW  MLSTAP
       CAL =7
       IF THIS IS MONITORED RUN
       SLW  SYSBIN
ICS05  EQU *
REWINd  SYSTP1
       START A1 REWINDING
       STL  IDEX
       BRA  REWIND,,**2
       MZE  SYSTP1
END05  REM
BEGINX  EQU *
       INITIAL ENTRY POINT FOR COMPILER
       AXT  4,1
       SORT THE GENERATOR KEY TABLES
       ONCE1  CAL  SORTK0,1
       GET AN INFORMATION WORD
       STA  ONCE+1
       STA  ONCE+2
       REM
       ONCE  CALL  SORT,......
       REM
       CAL  GENDES+1,1
       GET LENGTH OF ONE OF THE KEY TABLES
       ARS  6
       AND DETERMINE THE POWER-OF-TWO LENGTH
       ORA  021700000000
       OF THIS TABLE. THIS INFORMATION
FAD =021700000000 IS USED BY THE GEN ROUTINE'S
SUB =020000003000 BINARY SEARCH SECTION.
ARS 9+18 SHIFT COUNT INTO ADDRESS
STA GENBIN+1,1 SAVE IN ADDRESS OF BINER CELL
PAX *4 GET POINTER TO THE BINER TABLE
CAL BLOWER ,4 DECREMENT OF THIS TABLE WORD HAS THE
STD GENBIN+1,1 NEXT LOWER POWER OF TWO THAN THE TABLE
REM LENGTH, SET INTO THE BINER CELL.

TIX ONCE1,1,1
REM

I0506 EQU *
BUFFER BUFBIN,2,246,*SYSBIN
STL IOEX
BRA BUFFER**,++3
PZE BUFBIN**,2
MZE 240,,SYSBIN
REM
END06
IUS07 EQU *
BUFFER BUFUT1,2,179,SYSTUI
STL IOEX
BRA BUFFER**,++3
PZE BUFUT1**,2
MZE 179,,SYSUTI
REM
END07
I0508 EQU *
BUFFER BUFM1,2,302,MINTAP
STL IOEX
BRA BUFFER**,++3
PZE BUFM1**,2
MZE 302,,MINTAP
REM
END08
I0509 EQU *
BUFFER BUFMLT,2,302,*MLSTAP
STL IOEX
BRA BUFFER**,++3
PZE BUFMLT**,2
MZE 302,,MLSTAP
REM
END09
I0510 EQU *
XMODER GETMER
STL IOEX
BRA XMODER,,++2
MZE GETMER
REM
END10
I0511 EQU *
XEOF GETEOF
STL IOEX
BRA XEOF,,++2
MZE GETEOF
REM
END11
REM
LDQ ERROR .INITIALIZE EXECUTION ERROR SUBROUTINE
STQ 2
REM
CAL (FPI) . INITIALIZE THE FLOATING TRAP CELL
SLW 8
REM
STZ 0 TO RESET POSSIBLE TAG IN CELL 0
SLF TURN OFF THE LIGHTS
<table>
<thead>
<tr>
<th>REM</th>
<th>AXT</th>
<th>POOL-COLUMN-665 4</th>
<th>LENGTH OF THE POOL REGION</th>
<th>FT012880</th>
</tr>
</thead>
<tbody>
<tr>
<td>STZ</td>
<td>POOL+1 4</td>
<td>ZERO OUT THE POOL AREA THE FIRST TIME</td>
<td>FT012900</td>
<td></td>
</tr>
<tr>
<td>TIX</td>
<td>-1</td>
<td>START</td>
<td>FT012910</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FT012920</td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td></td>
<td></td>
<td>FT012930</td>
<td></td>
</tr>
<tr>
<td>SPACE</td>
<td>2</td>
<td>INFORMATION WORDS FOR TABLE SORTS</td>
<td>FT012940</td>
<td></td>
</tr>
<tr>
<td>PZE</td>
<td>KEYTB0,KEYTB0+LKEYTB0-1</td>
<td></td>
<td>FT012950</td>
<td></td>
</tr>
<tr>
<td>PZE</td>
<td>KEYTB1,KEYTB1+LKEYTB1-1</td>
<td></td>
<td>FT012960</td>
<td></td>
</tr>
<tr>
<td>PZE</td>
<td>KEYTB2,KEYTB2+LKEYTB2-1</td>
<td></td>
<td>FT012970</td>
<td></td>
</tr>
<tr>
<td>PZE</td>
<td>KEYTB3,KEYTB3+LKEYTB3-1</td>
<td></td>
<td>FT012980</td>
<td></td>
</tr>
<tr>
<td>SORTKO EQU</td>
<td>*</td>
<td></td>
<td>FT013000</td>
<td></td>
</tr>
</tbody>
</table>
The following instructions make up the normal initialization section of the Fastran compiler. This section is executed prior to the compilation of a program.

**EJECT**

**SPACE** 3

**REM**

**START** EQU *

**SWI** 6

**TRA** ++3  NO TIME IF SSW 6 UP

**10S12** EQU *

**RCLOCK** GET TIME IF SSW 6 DOWN

**STL** IOEX

**BRA** RCLOCK,,++1

**END12** REM

**SLN** 4  TURN ON CONSOLE WATCHER'S LIGHT

**REM**

**INITIALIZE THE SYMBOL TABLE**

**AXT** LSYMB+1,4

**STZ** SYMTAB+1,4

**TIX** ++1,4,1

**REM**

**INITIALIZE THE GETCH ROUTINE**

**STL** GETR1  SEQUENCE INVOLVED IN ONE-TIME TITLING SECT.

**NOP** GET1  THIS CELL IS USED IN A 'TRA**

**REM**

**10S13** EQU *

**REWIND** SYSUT1  REWIND THE SCRATCH TAPE

**STL** IOEX

**BRA** REWIND,,++2

**MZE** SYSUT1

**END13** REM

**LXA** NXTLOC+4  INITIALIZE THE POOL AREA TO ZERO

**TXL** ++3,4,0  SKIP OUT IF NO POOL WAS USED

**STZ** POOL+1,4

**TIX** ++1,4,1

**REM**

**INITIALIZE ALL COMPILER FLAG CELLS TO ZERO.**

**AXT** TRIGR2-TRIGR4  ZERO OUT BLOCK OF SWITCHES

**STZ** TRIGR2,4

**TIX** ++1,4,1

**REM**

**CAL** ALLSVN  INITIALIZE ASTACK+1 TO ALL ONES

**SLW** ASTACK+1

**CAL** =0200500000000000  INITIALIZE THE '7/9' PUNCH...

**SLW** 9LEFT  AFTER ZEROING OUT THE CELL.

**REM**

**ZSD** WORKS

**ZSD** GETWR1

**ZSD** FRESHR1

**REM**

**CAL** CICTL9

**SLW** CICTLH

**AXT** CRTI-1,4  INITIALIZE CICTLD ROUTINE FOR CIT1

**SXA** CTLDC1+4

**SXA** GETLCC+4

**AXT** CRTFLG1-1,4

**SXA** CTCDC1+4

**SXA** CTCDD1+4
SXA   CTLOCC,4   FT013610
SXA   CTLOCD,4   FT013620
SXA   CTLOCE,4   FT013630
SXA   GETLCA,4   FT013640
SXA   GETLCB,4   FT013650
ZSA   STSH1      FT013660
ZSD   TBUND-3    FT013670
ZSD   TBUND-2    FT013680
ZSD   TBUND-1    FT013690
ZSD   TBUND      FT013700
REM               FT013710
AXT   1,1         X  FT013720
SXA   GETFG1,1    X  FT013730
SXA   GETFG2,1    X  FT013740
SXA   PROLOG+2,1  FT013750
SXA   CTBL1,1     FT013760
SXA   CTBL2,1     FT013770
SXA   NXTLOC,1    FT013780
REM               FT013790
AXT   6,1         X  FT013800
SXA   GETFG2+1,1  X  FT013810
REM               FT013820
AXT   20,1        FT013830
SXA   STSH1-1,1   FT013840
REM               FT013850
AXT   36,1        FT013860
SXA   CTBL2+1,1   FT013870
REM               FT013880
AXT   EQUabl-INITFG,4 EQUIVALENCE MASTER DIRECTOR CELL FT013890
PXA   4           FT013900
SLW   INITFG      FT013910
REM               FT013920
REM               INITIALIZE FLAG CELLS PERTAINING TO FT013930
REM               CLOSED AND RESERVED LIBRARY NAMES FT013940
AXT   LSLNTB,4   FT013950
STZ   SLNTAB+LSLNTB,4 FT013960
TIX   #-1,4,2     FT013970
REM               FT013980
SXA   MAP22,4     SET MAP22 ADDRESS TO 2. FT013990
REM               FT014000
TRA   STARTT      FT014010
<table>
<thead>
<tr>
<th>TTL</th>
<th>PASS 1 - DESCRIPTION</th>
<th>FT014020</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>REM</td>
<td>FT014030</td>
</tr>
<tr>
<td>SPACE 9</td>
<td></td>
<td>FT014040</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>FT014050</td>
</tr>
<tr>
<td></td>
<td>FASTRAN COMPILER</td>
<td>FT014060</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>FT014070</td>
</tr>
<tr>
<td></td>
<td>RESEARCH COMPUTING CENTER</td>
<td>FT014080</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>FT014090</td>
</tr>
<tr>
<td></td>
<td>INDIANA UNIVERSITY</td>
<td>FT014100</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>FT014110</td>
</tr>
<tr>
<td></td>
<td>BLOOMINGTON, INDIANA</td>
<td>FT014120</td>
</tr>
<tr>
<td>SPACE 3</td>
<td></td>
<td>FT014130</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>FT014140</td>
</tr>
<tr>
<td></td>
<td>PASS 1</td>
<td>FT014150</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>FT014160</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FT014170</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FT014180</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FT014190</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FT014200</td>
</tr>
</tbody>
</table>
* EJECT
* BRIEF DESCRIPTION OF PASS 1 STRUCTURE.
* SPACE 3
* PASS 1 PERFORMS THE PRIMARY PASS OVER THE SOURCE DECK.
* THE SECTIONS OF PASS 1 ARE LISTED BELOW WITH A BRIEF STATEMENT
* OF THEIR FUNCTIONS. MOST ROUTINES HAVE THEIR OWN DETAILED
* DESCRIPTIONS GIVEN IN THE BODY OF THE LISTING.
* REM
* GETCH... READS AND LISTS SOURCE CARDS. COMMENTS CARDS IG OR *)
* ARE DETECTED HERE AND INGORED AFTER LISTING. OTHER CARDS ARE
* BURST WITH BLANKS DELETED FOR EASIER SCANNING. THIS ROUTINE
* CONTROLS PAGE TITLING AND SUBTITLING FORMATS, AND DETECTS A
* POSSIBLE BINARY DECK CARD LABEL.
* REM
* GETLBL... SCANS THE LABEL FIELD OF A CARD, DETECTING ANY
* COLUMN 1 CHARACTER AND ANY STATEMENT LABEL.
* REM
* PROCXX... RESERVED WORD TEST AND DETERMINATION OF STATEMENT TYPE.
* (AN ARITHMETIC STATEMENT IS DETECTED BY DEFAULT, I.E. IT DOES
* NOT CONFORM TO A RESERVED TYPE.)
* REM
* PROCXI... TRANSACTS MISCELLANEOUS BUSINESS PRIOR TO ENTERING
* THE STATEMENT PROCESSOR DETERMINED IN PROCXX. SEVERAL ERRORS
* ARE DETECTED HERE. IF OBJECT CODE ON A PREVIOUS STATEMENT WAS
* DELAYED IN ORDER TO SEE THE CURRENT CARD LABEL, IT IS NOW PUT
* OUT. THE STATEMENT LABEL (IF ANY) IS ENTERED INTO SYMTAB.
* REM
* SKEND... ENTERED FROM ALL STATEMENT PROCESSES AT THE END OF
* THEIR PROCESSING. AMONG OTHER MATTERS, ANY DO-STATEMENTS WHICH
* TERMINATE AT THE CURRENT STATEMENT WILL HAVE THEIR APPROPRIATE
* OBJECT CODE PRODUCED NOW.
* REM
* (STATEMENT PROCESSES FOR ALL RESERVED WORD STATEMENTS).
* REM
* IARITH... ARITHMETIC STATEMENT PROCESSING. THIS PROCESSOR
* UTILIZES A LARGE NUMBER OF AUXILIARY SUBROUTINES. SEE THEIR
* DESCRIPTIONS IN LATER PAGES.
* REM
* (PASS 1 UTILITY PROGRAMS).
* REM
GETCH ROUTINE...

REM

THIS ROUTINE READS SOURCE CARDS OFF THE MONITOR INPUT TAPE,
LISTS THEM ON THE OUTPUT TAPE, AND FOR NON-COMMENTS, NON-*,
OR NON-$ CARDS, BURSTS EACH STATEMENT ONE CHARACTER PER WORD.
IF A STATEMENT CONSISTS OF MORE THAN ONE CARD, THE ENTIRE
STATEMENT WILL BE BURST AT ONCE. COLUMNS 73-80 OF THE FIRST
CARD AND COLUMNS 1-6 AND 73-80 OF ALL SUCCEEDING CARDS WILL
BE IGNORED. NOTE THAT COLUMNS 1-6 (WITH COLUMN 6 BLANKED)
OUT) OF THE FIRST CARD OF A STATEMENT ARE CONSIDERED TO BE
A PART OF THE STATEMENT. TRAILING BLANKS WILL BE DELETED
BEFORE THE CARD IS BURST. THE BURST CARD IS STORED STARTING
WITH 'COLUMN' AND GOING TOWARD HIGH CORE. EACH CHARACTER
OCCUPIES BITS P1-5 OF A SEPARATE WORD IN COLUMN, WITH THE
NEXT FIVE CHARACTERS OF THE STATEMENT FILLING OUT THE WORD.
A MAXIMUM OF 666 CHARACTERS (10 CARDS + 6 LABEL CHARACTERS) IS
ALLOWED. AN OCTAL 77 IS ENTERED AS THE LAST CHARACTER ON
THE CARD FOR USE BY THE SCANNER.
UPON ENTERING THE ROUTINE FOR THE FIRST TIME, THE TITLE FOR EACH
PAGE OF THE OUTPUT LISTING WILL BE SET UP IN THE REGION
TITLE (ALSO DATE AND PAGE COUNT). IF THE TITLE CARD HAS
$*, # OR @ IN COLUMN 1, THE CARD IS NOT LISTED ON THE OUTPUT
TAPE. SUBTITLING IS PERMITTED BY MEANS OF A
CARD CONTAINING A $-SIGN IN COLUMN 1. IN THIS CASE, COLUMNS
2-72 OF $-SIGN CARDS WILL BE LISTED IN WORDS 4-15 OF THE
SECOND LINE OF EACH PAGE. $-SIGN CARDS CAUSE A PAGE EJECT
EXCEPT AS THE SECOND CARD OF THE SOURCE DECK. ALSO $-SIGN
CARDS WILL NOT BE LISTED ON THE OUTPUT TAPE. TO TURN OFF
SUB-TITLING, USE A BLANK $-SIGN CARD.

BINARY DECK LABELING IS EFFECTED BY THE USUAL FORTRAN RULES,
I.E., IF THE FIRST NON-$ CARD (OR NON-$ CARD IN ESSPAN) IS A
COMMENTS CARD WITH AT LEAST ONE NON-Blank CHARACTER IN COLUMNS
2-72, THESE SIX CHARACTERS ARE USED AS THE BINARY DECK LABEL. IF
THIS CONDITION IS NOT MET, THE SUBROUTINE OR FUNCTION NAME
IS USED, OR FOR A MAIN PROGRAM, THE CHARACTERS 'MAIN'.
ARE USED (THIS LAST IS DIFFERENT FROM FORTRAN).

STARTN

NZT ERHERE DID ERROR OCCUR IN LAST STATEMENT
TRA GETCH NO
STZ ERHERE YES. RESET FLAG CELL
TSX LIST,4 SKIP A LINE ON THE OUTPUT TAPE
BLANKS,,1
REM
GETCH ZET ENFILP HAS AN END-OF-FILE BEEN ENCOUNTERED
ERROR 37,FINISH NO END CARD FOUND. ERROR
STZ GETR4 TURN OFF FLAG INDICATING NON-VOID STATEMENT
REM UN CASE EOF IS ENCOUNTERED BEFORE NEXT
REM EXECUTABLE STATEMENT.
GETR ZET GETRFL
TRA GETR3 NO READ
REM
IOS14 EQU *
GETR5 READ MINTAP,DEC,CARD,14
GETR5 STL IOEX
BRA READ,,+3
PZP MINTAP,,DEC
MZE CARD,,14

FT014610
FT014620
FT014630
FT014640
FT014650
FT014660
FT014670
FT014680
FT014690
FT014700
FT014710
FT014720
FT014730
FT014740
FT014750
FT014760
FT014770
FT014780
FT014790
FT014800
FT014810
FT014820
FT014830
FT014840
FT014850
FT014860
FT014870
FT014880
FT014890
FT014900
FT014910
FT014920
FT014930
FT014940
FT014950
FT014960
FT014970
FT014980
FT014990
FT015000
FT015010
FT015020
FT015030
FT015040
FT015050
FT015060
FT015070
FT015080
FT015090
FT015100
FT015110
FT015120
FT015130
FT015140
FT015150
FT015160
FT015170
FT015180
FT015190
FT015200
END14
STL GETRFL
TRA GETRI
STA **3
AXT 0,4
CAL **4
SLW CARD+14,1
TXI **+1,4,-1
TIX **-3,1,1

GETR1
TRA * **
REM THIS SECTION SETS UP PAGE TITLING. 1ST CARD OF THE SOURCE DECK WILL BE USED AS A TITLE\nREM IF IT ALSO HAPPENS TO BE A COMMENTS OR * CARD, IT WILL NOT GET LISTED.

GETL
STL LIST
STZ L18
REM INITIALIZE FOR PAGE EJECT (LINE COUNT)\nREM INITIALIZE PAGE COUNT TO ZERO
STZ GETIB
CAL CARD
REM BLANK OUT COLUMN 1 IF C, *, OR $.
LGR 30
REM DO NOT LIST THESE CARDS IF TITLES.
LAS =H00000C
TRA ++2
TSX GET6+2,6
REM COMMENTS CARD, GO CHECK FOR LABELING\nLAS =H00000*\nTRA ++2
TRA GETIC * CARD\nLAS =H00000S\nREM IS IT A SPACE STATEMENT.
TRA ++2
TRA GET5 YES. IGNORE IT ON FIRST CARD.
LAS =H00000E\nREM IS IT EJECT.
TRA ++2
TRA GET5 YES. IGNORE IT ON FIRST CARD\nERA =H00000S\nREM IS IT A SUB-TITLE CARD.
TZG GETIC YES. IGNORE SUB-TITLING AND USE AS TITLE
STL GETIB\nREM NOT A COMMENTS, * OR $ CARD
STL LABFLG\nREM DECK LABELING NO LONGER POSSIBLE

GETIA
AXT 12,6
CAL CARD+12,4
SLW TITLE+13,4
TIX **-2,4,1
REM
CAL DBETB\nREM GET THE MONITOR DATE CELL
PAm 24\nREM SHIFT OUT ALL BUT THE MONTH
ANAm =077\nREM KILL ALL BUT THE LAST CHARACTER
PAX 1,4
REM LFT 10000\nREM WAS THERE A TENS DIGIT.
TXI **+1,4,10\nREM YES
CAL MONTHs4\nREM GET THE NAME OF THE MONTH
SLW TITLE+15\nREM PUT INTO THE TITLE LINE
REM
ZAC\nREM
LGL 6\nREM SHIFT IN THE TENS DIGIT OF THE DAY
TNZ **2\nREM IS IT ZERO.
CAL BLANKS\nREM YES
LGL 6\nREM OTHER DIGIT
ALS 6\nREM
ORA =H 00\nREM INSERT TRAILING BLANK
TITLE+14

BRING IN THE YEAR

PUT YEAR INTO TITLE LINE

GET1E

GETR1

NO SUBTITLE. RESET 1 TIME TITLING SWITCH

GET1B

LIST TITLING CARD

GET5

DO NOT LIST. RETURN FOR NEW CARD

GETIC

COMMENT, * OR $ CARDS

GET1B

TURN OFF LISTING OF THESE CARDS

CAL CARD

BLANK OUT COLUMN 1

Ana =07777777777

Ora =H 00000

SLW CARD

GET1A

CAL CARD

CHECK FOR SUB-TITLING CARD

30

LAS =H00000

IS IT SPACE.

**2

NO

TRA GETSPC

YES. GO PROCESS THE SPACING

LAS =H00000E

IS IT EJECT.

**2

NO

TRA GET12

YES. GO SET FOR EJECT

ERA =H00000

TZE GETTIL

SUB-TITLE. GO FIX UP THE TITLING

REM

CAL CARD+13

SAVE LOOK-AHEAD AND BLANK 'EM OUT

SLW LASTL

X

Ana =07777777777

Ora =H000

SLW CARD+13

X

TSX LIST+4

LIST THIS SOURCE CARD

CARD+14

CAL CARD

LGR 30

LAS =H00000

IS IT COMMENTS

**2

NO

TSX GET6+4

YES. GO CHECK FOR DECK LABELING

LAS =H00000

IS IT *

**2

NO

TRA GET5

YES

STL GET4

STL LABFLG

DECK LABELING NO LONGER POSSIBLE

AXT CARD+12,1

SXA CARD1,1

REM

CHECK IS IT A CONTINUATION CARD

REM

EOS15

EQU *

GET2A

LREAD MINTAP,DEC,ERASE,GETRFL

LOCATE NEXT SOURCE CARD

GET2A

IOEX

BRA LREAD++,++3

PZE MINTAP,DEC

UE ERASE,,GETRFL

END15

REM

REM

REM

REM
LXD ERASE,4
SXA GETR2,4
AXT 0,4
CAL GETR2 IS COL. 6 NON-ZERO
LGR 30
LAS =H0000C
TRA ++2
TRA GET3
LAS =H00000*
TRA ++2
TRA GET3
LAS =H00000$ IS THIS A SUBTITLE CARD.
TRA ++2 NO
TRA GET3 YES, STATEMENT IS NOW COMPLETE
LAS =H0000S IS IT SPACE.
TRA ++2 NO
TRA GET3 YES, TERMINATE THIS STATEMENT HERE.
LAS =H000CE IS IT EJECT.
TRA ++2
TRA GET3 YES, TERMINATE THIS STATEMENT HERE
LGL 30
ANA =077
TZE GET3 NO, THEREFORE NOT A CONTINUATION CARD.
LAS =H00000
TRA ++2
TRA GET4 COL 6 IS BLANK, GO TEST FOR BLANK CARD
REM CONTINUATION CARD
CAL CARD1
ADD =11
STA CARD1
AXT 11,1
TXI ++1,4,-1
6 TR2 CARD1
SLW ++,1
TIX =3,1,1 SAVE LOOK-AHEAD AND BLANK 81-84
AXC 13+,4
CAL* GETR2 X
SLW LASTLA X
ANA =077770000000 X
ORA =H00 X
SLW* GETR2 X
CAL GETR2
STA ++2
TSX LIST,4 LIST CARD JUST READ
+++1,14
TRA GET2A TRY AGAIN FOR A CONT. CARD
REM
CAL* GETR2 TEST FOR COMPLETELY BLANK CARD
ERZ BLANKS
TNZ GET3 SKIP OUT IF NON-BLANK WORD.
TXI ++1,4,-1
TXM ++,4,-12 QUIT WHEN OUT OF WORDS
TRA CARD1+2 ALL BLANK, GO LIST AND CONTINUE
SPACE 2
REM
GET 6
ZET LABFLG CHECK IF WE CAN GET A DECK LABEL
TRA 3,4 IS LABELING STILL POSSIBLE.
CAL CARD NO
LQ CARD+1 COMPLETELY KILL THE 'C'
LGL 7
LGR =H ARE COLUMNS 2-7 BLANK.

LAS **2 NO

CAL =0 YES. SET LABEL TO ZERO

Get7

SLW LABEL ESTABLISH A LABEL

STL LABFLG NO MORE LABELING IS POSSIBLE FROM 'C' CARDS

TRA* 3,4 RETURN

SPACE 2

REM

Get11

SLW SBTL SET THE SUBTITLING FLAG ON

AXT 12,4 WORDS ON THE CARD

CAL BLANKS APPEND BLANK TO COLUMN 1

GGL 30 RESTORE THE WORD

REM

Get11 SLW SUBTL+15,4 MOVE CARD IMAGE INTO SUBTITLE BUFFER

ERA BLANKS AND NOTE ANY NON-BLANK WORD IN THE CARD

TZE **2

SLN 1 SET LIGHT IF NON-BLANK WORD

CAL *13,4 GET NEXT WORD

TIX GETT1,4,1

REM

SLT 1 ANY NON-BLANK WORDS.

Get2

SLT SBTL NO. SET THE SUBTITLING FLAG OFF

TRA GET5 GO GET NEXT CARD.

REM

Get13 CAL CARD+1 PROCESS SPACING REQUIREMENTS

TSX BCDFIX+1,4 GO GET THE COUNT

ARS 18 INTO ADDRESS (IGNORE ERRORS IN THE COUNT)

PAX 1,4

ADD LIST PINES SO FAR ON THIS PAGE

CAS LPERPG IS IT TOO MUCH.

TRA GET2 YES. EJECT

TRA GET2

TSX LIST+1,4 SKIP LINES UNTIL THE COUNT IS EXHAUSTED

PZE BLANKS+1,1

TIX #2,4,1

TRA GET5 RETURN FOR ANOTHER CARD

SPACE 2

REM

Get12

STL MODERR EXECUTED ONLY IF MODE ERROR OR END-OF-FILE

TRA **2

REM

GetED

STL ENFILF

NBT GETR4

ERROR 37,FINISH NO END CARD FOUND. ERROR

REM

Get3

LXA CARD+1,4 SCAN OFF TRAILING BLANKS

TIX **1,4,CARD

PXA 1,4

PAC 1,4

CAL =H

Get3a

LAS CARD-1,4

TRA **2

TXI #2,4,1

TXL GETCH+1,4 BLANK CARD. GO GET ANOTHER CARD

TXH GET5+1,4,2 LABEL ONLY. IS AN ERROR

LDQ =07676767676

STQ CARD,4
<table>
<thead>
<tr>
<th>Instruction</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>PXA</td>
<td>F017610</td>
</tr>
<tr>
<td>PAC</td>
<td>F017620</td>
</tr>
<tr>
<td>TXI</td>
<td>F017630</td>
</tr>
<tr>
<td>PXX</td>
<td>F017640</td>
</tr>
<tr>
<td>ACL</td>
<td>F017650</td>
</tr>
<tr>
<td>ACL</td>
<td>F017660</td>
</tr>
<tr>
<td>STA</td>
<td>F017670</td>
</tr>
<tr>
<td>AXT</td>
<td>F017680</td>
</tr>
<tr>
<td>TOV</td>
<td>F017690</td>
</tr>
<tr>
<td>CAL</td>
<td>F017700</td>
</tr>
<tr>
<td>LDQ</td>
<td>F017710</td>
</tr>
<tr>
<td>STG</td>
<td>F017720</td>
</tr>
<tr>
<td>SXX</td>
<td>F017730</td>
</tr>
<tr>
<td>GET3B</td>
<td>F017740</td>
</tr>
<tr>
<td>LDQ</td>
<td>F017750</td>
</tr>
<tr>
<td>GET3B3</td>
<td>F017760</td>
</tr>
<tr>
<td>CRQ</td>
<td>F017770</td>
</tr>
<tr>
<td>TXH</td>
<td>F017780</td>
</tr>
<tr>
<td>ALS</td>
<td>F017790</td>
</tr>
<tr>
<td>ORA</td>
<td>F017800</td>
</tr>
<tr>
<td>GET3B4</td>
<td>F017810</td>
</tr>
<tr>
<td>***</td>
<td>F017820</td>
</tr>
<tr>
<td>TXI</td>
<td>F017830</td>
</tr>
<tr>
<td>TIX</td>
<td>F017840</td>
</tr>
<tr>
<td>GET3B2</td>
<td>F017850</td>
</tr>
<tr>
<td>AXT</td>
<td>F017860</td>
</tr>
<tr>
<td>TIX</td>
<td>F017870</td>
</tr>
<tr>
<td>TRA</td>
<td>F017880</td>
</tr>
<tr>
<td>GET3B0</td>
<td>F017890</td>
</tr>
<tr>
<td>TRA</td>
<td>F017900</td>
</tr>
<tr>
<td>GET3B5</td>
<td>F017910</td>
</tr>
<tr>
<td>TRA</td>
<td>F017920</td>
</tr>
<tr>
<td>GET3B6</td>
<td>F017930</td>
</tr>
<tr>
<td>SLW</td>
<td>F017940</td>
</tr>
<tr>
<td>SCANTT</td>
<td>F017950</td>
</tr>
<tr>
<td>DUP</td>
<td>F017960</td>
</tr>
<tr>
<td>PZE</td>
<td>F017970</td>
</tr>
<tr>
<td>DUP</td>
<td>F017980</td>
</tr>
<tr>
<td>PZE</td>
<td>F017990</td>
</tr>
<tr>
<td>DUP</td>
<td>F018000</td>
</tr>
<tr>
<td>PZE</td>
<td>F018010</td>
</tr>
<tr>
<td>DUP</td>
<td>F018020</td>
</tr>
<tr>
<td>SCANTC</td>
<td>F018030</td>
</tr>
<tr>
<td>EQU</td>
<td>F018040</td>
</tr>
<tr>
<td>GET5</td>
<td>F018050</td>
</tr>
<tr>
<td>ERROR</td>
<td>F018060</td>
</tr>
<tr>
<td>CARD2</td>
<td>F018070</td>
</tr>
<tr>
<td>PZE</td>
<td>F018080</td>
</tr>
<tr>
<td>TITLE</td>
<td>F018090</td>
</tr>
<tr>
<td>BCI</td>
<td>F018100</td>
</tr>
<tr>
<td>BSS</td>
<td>F018110</td>
</tr>
<tr>
<td>BLANK</td>
<td>F018120</td>
</tr>
<tr>
<td>BLANKS</td>
<td>F018130</td>
</tr>
<tr>
<td>BSS</td>
<td>F018140</td>
</tr>
<tr>
<td>BCI</td>
<td>F018150</td>
</tr>
<tr>
<td>BSS</td>
<td>F018160</td>
</tr>
<tr>
<td>REM</td>
<td>F018170</td>
</tr>
<tr>
<td>REM</td>
<td>F018180</td>
</tr>
<tr>
<td>GETCHX</td>
<td>F018190</td>
</tr>
<tr>
<td>EQU</td>
<td>F018200</td>
</tr>
</tbody>
</table>
**TTL P A S S 1 - STATEMENT LABEL AND MODE**

* GETLBL ROUTINE...

**REM**

* ROUTINE TO SCAN OFF THE LABEL IN COLUMNS 1-5. ROUTINE ALSO
  LOOKS AT COLUMN 1 FOR MODE DESIGNATIONS F, B, D, I.
* ROUTINE SETS CELL MODFLG TO 1, 2, 3, 4 ACCORDING AS COLUMN 1
* IS B, D, I, F, RESPECTIVELY. IF COLUMN 1 IS NOT ANY OF THE
  ABOVE, MODFLG IS ZEROED OUT.

**REM**

* THE PROCESSED LABEL WILL APPEAR IN THE CELL 'CARD' AT THE END
* SCAN. BLANKS AND LEADING ZEROES WILL BE DROPPED.
* THE LABEL IS NOT PUT INTO SYMTAB YET.

**SPACE 3**

GETLBL CAL CARD

ARS 30
AXT 0,1
LAS =O60
TRA ++2
TRA LABL1 BLANK IN COL 1
LAS =012
TRA ++3
TRA ++2
TRA LABL1 COL 1 IS NUMERIC
LAS =H00000F
TRA ++2
TXI LABL4,1,4 COL 1 IS F
LAS =H000008
TRA ++2
TXI LABL4,1,1 COL 1 IS B
LAS =H00000D
TRA ++2
TXI LABL4,1,2 COL 1 IS D
LAS =H00000I
ERROR 80,SKEND INVALID CHARACTER OR LETTER IN COL 1
ERROR 80,SKEND

LABL4 CAL =07777777777
ANA CARD
ORA =H 00000
SLW CARD

**REM**

LABL1 CAL CARD GET LABEL IMAGE

STZ CARD ZERO OUT LABEL CELL
ARS 6 KILL COLUMN 6
REM
LAS =HU IS LABEL BLANK.
TRA ++2 NO.
TRA GETLBCX-1 YES.
REM
LGR 30 NO. PUT IN MQ
AXC 36,4 SET TERMINAL SHIFT
AXT 5,2 NUMBER OF POSITIONS TO TEST

**REM**

LABL8 ZAC

LGL 6 SHIFT IN CHARACTER
LAS =060 IS IT BLANK.
TRA ++4 NO. BAD CHARACTER
TRA LABL2 YES. SKIP.
REM
LAS =012 IS IT NIN-NUMERIC.
<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRA</td>
<td><strong>1</strong></td>
<td>60</td>
</tr>
<tr>
<td>ERROR</td>
<td>1,SKEND</td>
<td></td>
</tr>
<tr>
<td>NZT</td>
<td>CARD</td>
<td></td>
</tr>
<tr>
<td>TZE</td>
<td>LABL2</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ORA</td>
<td>CARD</td>
<td></td>
</tr>
<tr>
<td>ALS</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>SLW</td>
<td>CARD</td>
<td></td>
</tr>
<tr>
<td>TXI</td>
<td><strong>1,4,6</strong></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LABL2</td>
<td>TIX</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAL</td>
<td>CARD</td>
<td></td>
</tr>
<tr>
<td>TZE</td>
<td>LABL3</td>
<td></td>
</tr>
<tr>
<td>ARS</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>LDQ</td>
<td>=H</td>
<td></td>
</tr>
<tr>
<td>LGL</td>
<td>0,4</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLW</td>
<td>CARD</td>
<td></td>
</tr>
<tr>
<td>SXA</td>
<td>MODFLG,1</td>
<td></td>
</tr>
<tr>
<td>GETLBX</td>
<td>EQU</td>
<td></td>
</tr>
</tbody>
</table>
TTL PASS 1 - DETERMINE STATEMENT TYPE
RESERVED WORD TEST AND DETERMINATION OF STATEMENT TYPE.

* CONTROL PASSES TO THIS ROUTINE AFTER ANY LABEL AND/OR COLUMN
* 1 MODE HAS BEEN SCANNED.

* IF THERE IS AN F IN COLUMN 1, AN IMMEDIATE EXIT IS TAKEN TO THE
* F-CARD PROCESSOR. IF NOT, THE FIRST CHARACTER IN THE FIRST
* WORD OF 'COLUMN' IS USED TO BRANCH ON A TABLE OF TRANSFERS.
* EACH INITIAL LETTER WHICH MAY BEGIN A RESERVED WORD HAS ITS
* OWN SUBSEQUENT TEST SECTION. THERE, IT IS DETERMINED IF
* THIS STATEMENT BEGINS WITH A RESERVED WORD. IF IT DOES NOT,
* THEN THE STATEMENT IS AN ARITHMETIC SUBSTITUTION STATEMENT.

* IN ANY EVENT, THE CELL PROCSW IS FORMED, TELLING THE
* ENTRY POINT TO THE CORRECT STATEMENT PROCESSOR, AND TELLING
* IF THIS IS AN EXECUTABLE, TRANSFER, DO, CALL, AND/OR END-TYPE
* STATEMENT. CONTROL THEN PASSES TO THE PASS 1 DRIVER, WHERE
* SOME INITIAL BUSINESS IS HANDLED PRIOR TO PROCESSING THE
* STATEMENT.

SPACE 3

PROCXX LXA MODFLG,4 GET THE MODE FLAG FOR THIS STATEMENT
TXH PROCXX,4,3 IS IT AN F-CARD.
REM
AXT 0,2
CAL COLUMN
ARS 30
PAC 4 KEY WORD TEST
TRA **1,4 KEY WORD FIRST LETTER TABLE
REM
DUP 1,17
STR FORM11,SKEND
TRA TA1.0 A 21
TRA TB2.0 B 22
TRA TC3.0 C 23
TRA TD4.0 D 24
TRA TE5.0 E 25
TRA TF6.0 F 26
TRA TG7.0 G 27
TRA PARIH H 30
TRA TI8.0 I 31
DUP 1,7
STR FORM11,SKEND
TRA PARIH J 41
TRA PARIH K 42
TRA TL13.0 L 43
TRA PARIH M 44
TRA PARIH N 45
TRA TG19.0 U 46
TRA TP9.0 P 47
TRA PARIH Q 50
TRA TR10.0 R 51
DUP 1,8
STR FORM11,SKEND
TRA TS11.0 S 62
TRA PARIH T 63
TRA PARIH U 64
TRA PARIH V 65
TRA TM12.0 W 66
TRA PARIH X 67
<table>
<thead>
<tr>
<th>Code</th>
<th>Action</th>
<th>Parameter</th>
<th>Value</th>
<th>Code</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRA</td>
<td>PARITH</td>
<td>Y</td>
<td>70</td>
<td>FT019630</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>PARITH</td>
<td>Z</td>
<td>71</td>
<td>FT019640</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DUP</td>
<td></td>
<td>1,6</td>
<td></td>
<td>FT019650</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STR</td>
<td></td>
<td>FORM11,SKEND</td>
<td></td>
<td>FT019660</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
* ROUTINE TO CHECK IF A POSSIBLE IF, FORMAT, END, OR CALL
* STATEMENT IS REALLY AN ARITHMETIC STATEMENT (I.E. HAS A
* VALID EQUAL SIGN FOLLOWING A POSSIBLE SUBSCRIPT EXPRESSION).

CHKSUB
SXA  EXCKSB,4
LDI 2  PREPARE TO LATER RESTORE THE ERROR TRAP
AXT CHKERR,4  ALTER THE ERROR TRAP CELL TO RETURN TO
SXA 2,4  THIS ROUTINE INSTEAD OF THE ERROR ROUTINE.
SXA SCN85,4  ALTER A CELL IN SCN8CD TO GIVE ERROR RETURN.
REM
TSX SCN8CD,4  NOW SCAN OFF ANY VALID CONSTANT OR
REM SYMBOL, TO AVOID POSSIBLE HOLLERITH
REM
SK1 RPCHAR  LITERAL, WHICH WILL BE CAUGHT AS AN ERROR.
REM
TXI *-1,2,2,-1  IS THE BREAK CHARACTER A RIGHT PAREN.
REM
AXC PARITH-1,1  NO. RETURN FOR ANOTHER SCAN
SXA SKIP1 EQCHAR  GET A POSSIBLE TRANSFER POINT
SXA EXCKSB,1  IS THE NEXT CHARACTER AN EQUAL SIGN.
REM
CHKERR STI 2  SET THE PROPER TRANSFER POINT
AXT SCN86,4  RESTORE THE ERROR TRAP CELL
SXA SCN85,4  RESTORE THE ORIGINAL ADDRESS IN SCN8CD CELL
REM
EXCKSB AXT **,4  NO. RESTORE XR4 AND RETURN
TRA 1,4  PROCESSOR HAS BEEN FOUND
EJECT

PROCX2 REMARK (F-CARD PROCESSOR)
SPRSE IFCARD, NOSEQ, NOTRA, NODO
<table>
<thead>
<tr>
<th>Column</th>
<th>Value</th>
<th>FT</th>
</tr>
</thead>
<tbody>
<tr>
<td>TAI0</td>
<td>EJECT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CAL</td>
<td>=HASSIGN</td>
</tr>
<tr>
<td></td>
<td>ERA</td>
<td>COLUMN,2</td>
</tr>
<tr>
<td></td>
<td>TNZ</td>
<td>PARITH</td>
</tr>
<tr>
<td></td>
<td>TSX</td>
<td>BREAK,4</td>
</tr>
<tr>
<td></td>
<td>ERA</td>
<td>=077</td>
</tr>
<tr>
<td></td>
<td>TNZ</td>
<td>PARITH</td>
</tr>
<tr>
<td></td>
<td>REMARK</td>
<td>(ASSIGN STATEMENT PROCESSOR)</td>
</tr>
<tr>
<td></td>
<td>SPRSW</td>
<td>1ASSNX,XEQ,NOTRA,NODD</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>FT019960</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FT019970</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FT019980</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FT019990</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FT020000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FT020010</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FT020020</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FT020030</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FT020040</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>EJECT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Th2.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAL</td>
<td>=HBACKSP</td>
<td>IS IT BACKSPACE</td>
</tr>
<tr>
<td>ERA</td>
<td>COLUMN,2</td>
<td></td>
</tr>
<tr>
<td>INZ</td>
<td>PARITH</td>
<td>NO</td>
</tr>
<tr>
<td>CAL</td>
<td>=HACE</td>
<td></td>
</tr>
<tr>
<td>ERA</td>
<td>COLUMN+6,2</td>
<td></td>
</tr>
<tr>
<td>ANA</td>
<td>=0777777000000</td>
<td></td>
</tr>
<tr>
<td>INZ</td>
<td>PARITH</td>
<td>NO AGAIN</td>
</tr>
<tr>
<td>REMARK</td>
<td>(BACKSPACE STATEMENT PROCESSOR)</td>
<td></td>
</tr>
<tr>
<td>SPRSW</td>
<td>1BSRXX,XEQ,NOTRA,NODO</td>
<td></td>
</tr>
</tbody>
</table>
EJECT
CAL =HCALL
ERA COLUMN,2
ANA =0777777770000
TNZ T3.1
TXI ++1,2,-4
TSX BREAK,4
LAS =077
TRA ++2
TRA T3.3
ERA =H0000001
TNZ PARITH NO, ARITHMETIC STATEMENT
REM
TXL T3.3,2,-9
LXD FIRSTF,4
TXL ++2,4,0
TXL T3.3,2,-8
TSX CHKSUB,4

T3.3 REMARK (CALL STATEMENT PROCESSOR)
SPRSW ICALLX,XEQ,NOTRA,NODG,BCALL

T3.1 CAL =HCOMMON
ERA COLUMN,2
TNZ T3.2
TXI ++1,2,-6
TSX BREAK,4
LAS =077
TRA ++3
ERA =H00000,
TNZ PARITH NO
REMARK (COMMON STATEMENT PROCESSOR)
SPRSW ICOMNX,NOXEQ,NOTRA,NODG

T3.2 CAL =HCONTIN
ERA COLUMN,2
TNZ PARITH NO
CAL =HUC
ERA COLUMN+6,2
ANA =0777700000000
TNZ PARITH
REMARK (CONTINUE STATEMENT PROCESSOR)
SPRSW ICNTX,XEQ,NOTRA,NODG
<table>
<thead>
<tr>
<th>ID</th>
<th>Instruction</th>
<th>Description</th>
<th>Address</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>T04.0</td>
<td>EJECT</td>
<td>IS IT DIMENSION</td>
<td>FT020560</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>CAL =HIDIMENS</td>
<td></td>
<td>FT020570</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ERA COLUMN,2</td>
<td></td>
<td>FT020580</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TNZ T4.1</td>
<td></td>
<td>FT020590</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CAL =HION</td>
<td></td>
<td>FT020600</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ERA COLUMN+6,2</td>
<td></td>
<td>FT020610</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ANA =07777770000000</td>
<td></td>
<td>FT020620</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TNZ PARITH</td>
<td></td>
<td>FT020630</td>
<td></td>
</tr>
<tr>
<td></td>
<td>REMARK (DIMENSION STATEMENT PROCESSOR)</td>
<td></td>
<td>FT020640</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPRSW DIMX,NOSEQ,NODA</td>
<td></td>
<td>FT020650</td>
<td></td>
</tr>
<tr>
<td>T4.1</td>
<td>CAL =HDO</td>
<td></td>
<td>FT020660</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ERA COLUMN,2</td>
<td></td>
<td>FT020670</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ANA =07777000000000</td>
<td></td>
<td>FT020680</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TNZ PARITH</td>
<td></td>
<td>FT020690</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TXI **1,2,-2</td>
<td></td>
<td>FT020700</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TSX BREAK,4</td>
<td></td>
<td>FT020710</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ERA =H00000=</td>
<td></td>
<td>FT020720</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TNZ PARITH</td>
<td></td>
<td>FT020730</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TSX BREAK,4</td>
<td></td>
<td>FT020740</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ERA =H00000,</td>
<td></td>
<td>FT020750</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TNZ PARITH NOT A DO</td>
<td></td>
<td>FT020760</td>
<td></td>
</tr>
<tr>
<td></td>
<td>REMARK (DO STATEMENT PROCESSING)</td>
<td></td>
<td>FT020770</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPRSW PDO,SEQ,NODA,DO</td>
<td></td>
<td>FT020780</td>
<td></td>
</tr>
</tbody>
</table>
EJECT
CAL =HEND       IS IT END
ERA COLUMN,2   FT020790
ANA =0777777000000  FT020800
TNZ T5.2
AXC 3,2
SKIP 63      USUALLY THIS IS END OF STATEMENT
TRA T5.4    AHA, NOT SO THIS TIME...

T5.3 REMARK (END STATEMENT PROCESSOR)
SPRSW PEND,NOSEQ,NOTRA,NODC,BEND

T5.4 SKIP LPCHAR       IS THERE A LEFT PAREN.
TRA PARITH       NO. ARITHMETIC STATEMENT
TSX CKSUB,4     YES, GO CHECK IF THIS IS ARITHMETIC
TRA T5.3       OH, HO... THIS IS ONE OF THOSE RIDICULOUS
REM  END STATEMENTS WITH THE MONITOR OPTIONS.
REM  THE OPTIONS ARE, OF COURSE, IGNORED.

T5.0 CAL =HENDFIL
ERA COLUMN,2    IS IT ENDFILE
TNZ T5.1
CAL =HE
ERA COLUMN+6,2
ANA =07700000000000  FT020990
TNZ PARITH     NO
REMARK (END FILE STATEMENT PROCESSOR)
SPRSW IEOFXX,XXEQ,NOTRA,NODD

T5.2 CAL =HEQUIVA   IS IT EQUIVALENCE
ERA COLUMN,2
TNZ PARITH
CAL =HELENCE
ERA COLUMN+6,2
ANA =07777777700
TNZ PARITH
REMARK (EQUIVALENCE STATEMENT PROCESSOR)
SPRSW IEOVX,NOSEQ,NOTRA,NODD
EJECT
CAL =HFFORMAT
ERA COLUMN,2
INZ T6.1
AXC 6,2
SKIP LPCHAR IS NEXT CHARACTER A LEFT PAREN.
TRA PARITH NO
ZET FORFLG HAS ARRAY NAMED 'FORMAT' BEEN ENCOUNTERED.
TSX CHKSUB,4 YES. GO CHECK IF THIS IS ARITHMETIC STATE.
REMARK (FORMAT STATEMENT PROCESSING)

T6.1
SPRSW PFMT,NOEQ,NOTRA,NODD
CAL =HFREQUE
ERA COLUMN,2
INZ T6.2
CAL =HNZ
ERA COLUMN+6,2
ANA =077777700000
INZ PARITH
REMARK (FREQUENCY STATEMENT PROCESSING)
SPRSW 1FREQX,NOEQ,NOTRA,NODD

T6.2
CAL =HFUNCTION IS IT FUNCTION
ERA COLUMN,2
INZ TF6.0
CAL =HON
ERA COLUMN+6,2
ANA =077777700000
INZ PARITH
REMARK (FUNCTION STATEMENT PROCESSING)
SPRSW PFUNCT,NOEQ,NOTRA,NODD
EJECT
CAL =HIF(SEN
ERA COLUMN+2
TNZ T8.2
CAL =HSELIG
ERA COLUMN+6,2
TNZ T8.1
CAL =HT
ERA COLUMN+12,2
ANA =077000000000
TZE PIFSL TO IS(SENSE LIGHT) PROCESSOR
ERROR 15,SKEND SYMBOL TOO LONG
REM

T8.1
CAL =HSESWT
ERA COLUMN+6,2
TNZ PARITH
CAL =HCH
ERA COLUMN+12,2
ANA =077770000000
TZE PIFSW TO IF (SENSE SWITCH) PROCESSOR
ERROR 15,SKEND SYMBOL TOO LONG
REM

T8.2
CAL =HIFI
ERA COLUMN+2
ANA =077777000000
TNZ T8.3
AXC 3,2 SET THE COLUMN POINTER
ZET IFFLAG HAS AN ARRAY NAMED 'IF' BEEN ENCOUNTERED.
TSX CHKSUB,4 YES, GO CHECK IF THIS IS AN ARITH. STATE.
REMARK (IF STATEMENT PROCESSOR)
SPRISW 11FXXX,XEQ,TRANS,NOOD
REM

T8.3
CAL =HIFACCW
ERA COLUMN+2
TNZ T8.4
CAL =HMULATO
ERA COLUMN+6,2
TNZ PARITH
CAL =HRQVERF
ERA COLUMN+12,2
TNZ ERR015 SYMBOL TOO LONG
CAL =HLOW
ERA COLUMN+18,2
ANA =077777000000
TZE PIFAC TO IF ACCUMULATOR OVERFLOW PROCESSOR
ERROR 15,SKEND SYMBOL TOO LONG

T8.4
CAL =HIFQUOT
ERA COLUMN+2
TNZ T8.5
CAL =HINTOV
ERA COLUMN+6,2
TNZ PARITH
CAL =HERFLOW
ERA COLUMN+12,2
TZE PIFMQ TO IF QUOTIENT OVERFLOW PROCESSOR
ERROR 15,SKEND SYMBOL TOO LONG

T8.5
CAL =HIFDIVI
ERA COLUMN+2
TNZ PARITH
CAL =HDECHEC
<table>
<thead>
<tr>
<th>Error Code</th>
<th>Description</th>
<th>Code 1</th>
<th>Code 2</th>
<th>Code 3</th>
<th>Code 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>T0.5A</td>
<td>ERROR 15, SKEND SYMBOL TOO LONG</td>
<td>REMARK (IF DIVIDE CHECK STATEMENT PROCESSOR)</td>
<td>SPRSW IFDIV, XEQ, TRANS, NODO</td>
<td>SPRSW IFSLT, XEQ, TRANS, NODO</td>
<td>SPRSW IFSWT, XEQ, TRANS, NODO</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
EJECT
CAL =HPAUSE
ERA COLUMN,2
ANA =0777777777700
TNZ T9.1
TXI ++1,2,-5
TSX BREAK,4
ERA =077
TNZ PARITH
REMARK (PAUSE STATEMENT PROCESSING)
SPRSW 1PAUSE,xEQ,NOTRA,NODO

T9.1
CAL =HPRINT
ERA COLUMN,2
ANA =0777777777700
TNZ T9.2
TXI ++1,2,-5
TSX BREAK,4
LAS =077
MACER MACHINE ERROR
TRA ++3
ERA =H000000
TNZ PARITH
REMARK (PRINT STATEMENT PROCESSOR)
SPRSW 1PRINT,xEQ,NOTRA,NODO

T9.2
CAL =HPUNCH
ERA COLUMN,2
ANA =0777777777700
TNZ PARITH
TXI ++1,2,-5
TSX BREAK,4
LAS =077
MACER MACHINE ERROR
TRA ++3
ERA =H000000
TNZ PARITH
REMARK (PUNCH STATEMENT PROCESSOR)
SPRSW 1PUNCH,xEQ,NOTRA,NODO
<table>
<thead>
<tr>
<th>Line</th>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1</td>
<td>TXI 1,2,-4</td>
<td>FREEZE, 4</td>
</tr>
<tr>
<td>10.2</td>
<td>ERA COLUMN,2</td>
<td>READ COLUMN,2</td>
</tr>
<tr>
<td>10.3</td>
<td>ERA COLUMN+6,2</td>
<td>READ COLUMN+6,2</td>
</tr>
<tr>
<td>10.4</td>
<td>ERA COLUMN+12,2</td>
<td>READ COLUMN+12,2</td>
</tr>
<tr>
<td>10.5</td>
<td>ERA COLUMN+18,2</td>
<td>READ COLUMN+18,2</td>
</tr>
<tr>
<td>10.6</td>
<td>ERA COLUMN+24,2</td>
<td>READ COLUMN+24,2</td>
</tr>
<tr>
<td>10.7</td>
<td>ERA COLUMN+30,2</td>
<td>READ COLUMN+30,2</td>
</tr>
<tr>
<td>10.8</td>
<td>ERA COLUMN+36,2</td>
<td>READ COLUMN+36,2</td>
</tr>
<tr>
<td>10.9</td>
<td>ERA COLUMN+42,2</td>
<td>READ COLUMN+42,2</td>
</tr>
<tr>
<td>10.10</td>
<td>ERA COLUMN+48,2</td>
<td>READ COLUMN+48,2</td>
</tr>
<tr>
<td>10.11</td>
<td>ERA COLUMN+54,2</td>
<td>READ COLUMN+54,2</td>
</tr>
<tr>
<td>10.12</td>
<td>ERA COLUMN+60,2</td>
<td>READ COLUMN+60,2</td>
</tr>
</tbody>
</table>

**Remark:**
- (READ TAPE STATEMENT PROCESSOR)
- (READ DRUM STATEMENT PROCESSOR)
<table>
<thead>
<tr>
<th>TNZ</th>
<th>PARITH</th>
<th>FT023470</th>
</tr>
</thead>
<tbody>
<tr>
<td>REMARK</td>
<td>(REWIND STATEMENT PROCESSOR)</td>
<td>FT023480</td>
</tr>
<tr>
<td>SPRSW</td>
<td>1REWXX, XEQ, NOTRA, NODO</td>
<td>FT023490</td>
</tr>
</tbody>
</table>
EJECT

T311.0
CAL =HSENSEL
ERA COLUMN,2
TNZ T311.1
CAL =HIGHT
ERA COLUMN+6,2
ANA =0777777770000
TNZ PARITH
REMARK (SENSE LIGHT STATEMENT PROCESSOR)
SPRSW ISENSE,XEQ,NOTRA,NODD

T311.1
CAL =HSUBROU
ERA COLUMN,2
TNZ T311.2
CAL =HTINE
ERA COLUMN+6,2
ANA =0777777770000
TNZ PARITH
REMARK (SUBROUTINE STATEMENT PROCESSOR)
SPRSW PSUBR,NOXEQ,NOTRA,NODD

T311.2
CAL =HSTOP
ERA COLUMN,2
ANA =0777777770000
TNZ TS11.3
TXI ++1,2,-4
TSX BREAK,4
ERA =077
TNZ PARITH
REMARK (STOP STATEMENT PROCESSOR)
SPRSW ISTOPX,XEQ,TRANS,NODD
<table>
<thead>
<tr>
<th>Tw12</th>
<th>CAL</th>
<th>#HWRITE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ERA</td>
<td>COLUMN,2</td>
</tr>
<tr>
<td>ERA</td>
<td>T12.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CAL</td>
<td>#HAP</td>
</tr>
<tr>
<td>ERA</td>
<td>COLUMN+6,2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ANA</td>
<td>#07777700000</td>
</tr>
<tr>
<td>REMARK</td>
<td>(WRITE TAPE STATEMENT PROCESSOR)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPRSW</td>
<td>1WRTPX,XEQ,NOTRA,NODD</td>
</tr>
<tr>
<td>T12.1</td>
<td>CAL</td>
<td>#HWRITE</td>
</tr>
<tr>
<td>ERA</td>
<td>COLUMN,2</td>
<td></td>
</tr>
<tr>
<td>TNZ</td>
<td>T12.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CAL</td>
<td>#HRUM</td>
</tr>
<tr>
<td>ERA</td>
<td>COLUMN+6,2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ANA</td>
<td>#07777700000</td>
</tr>
<tr>
<td>REMARK</td>
<td>(WRITE TAPE STATEMENT PROCESSOR)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPRSW</td>
<td>1WRQX,XEQ,NOTRA,NODD</td>
</tr>
<tr>
<td>T12.2</td>
<td>CAL</td>
<td>#HWRITE</td>
</tr>
<tr>
<td>ERA</td>
<td>COLUMN,2</td>
<td></td>
</tr>
<tr>
<td>TNZ</td>
<td>PARITH</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CAL</td>
<td>#HUPUTT</td>
</tr>
<tr>
<td>ERA</td>
<td>COLUMN+6,2</td>
<td></td>
</tr>
<tr>
<td>TNZ</td>
<td>PARITH</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CAL</td>
<td>#HAP</td>
</tr>
<tr>
<td>ERA</td>
<td>COLUMN+12,2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ANA</td>
<td>#07777700000</td>
</tr>
<tr>
<td></td>
<td>TNZ</td>
<td>T8.5A</td>
</tr>
<tr>
<td>REMARK</td>
<td>(WRITE OUTPUT TAPE STATEMENT PROCESSOR)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPRSW</td>
<td>1WOTXX,XEQ,NOTRA,NODD</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td></td>
</tr>
<tr>
<td>PARITH</td>
<td>REMARK</td>
<td>(ARITHMETIC STATEMENT PROCESSOR)</td>
</tr>
<tr>
<td>SPRSW</td>
<td>1ARITH,XEQ,NOTRA,NODD</td>
<td></td>
</tr>
</tbody>
</table>
* EJECT PROCESsING FOR NON-FORTRAN STATEMENTS  
  SPACE 2  
  T13.0 CAL =HLISTPA IS IT PASS!  
  ERA COLUMN,2  
  TNZ T13.2 NO  
  CAL =HSSIOFF  
  ERA COLUMN+6,2  
  TNZ T13.4  
  STZ LPASSI  
  REMARK (TURN OFF LISTING OF PASS 1)  
  TRA T16.2  
  T13.4 ANA =0777777000000  
  TNZ PARITH  
  STL LPASSI TURN ON LISTING OF PASS 1  
  REMARK (TURN ON LISTING OF PASS1)  
  TRA T16.2  
  T13.2 CAL =HLISTSY IS IT LISTSYMBOLTABLE  
  ERA COLUMN,2  
  TNZ PARITH  
  CAL =HMBOLTA  
  ERA COLUMN+6,2  
  TNZ PARITH  
  CAL =HBLE  
  ERA COLUMN+12,2  
  ANA =0777777000000  
  TNZ T8.5A  
  REMARK (TURN ON SYMBOL TABLE LIST FLAG)  
  STL LSTSYM  
  TRA T16.2  
  T15.0 CAL =HONLINE IS IT ONLINE  
  ERA COLUMN,2  
  TNZ T15.1  
  TXI ++1,2,-6  
  SKIP EOS  
  TRA PARITH  
  REMARK (PRINT OUTPUT ON LINE ALSO)  
  STL PRIQQ  
  TRA T16.2  
  T15.1 CAL =HOFFLIN IS IT OFF LINE  
  ERA COLUMN,2  
  TNZ PARITH  
  CAL =HE  
  ERA COLUMN+6,2  
  ANA =0770000000000  
  TNZ PARITH  
  REMARK (TURN OFF ON-LINE LISTING)  
  STZ PRIQQ  
  TRA T16.2  
  T16.0 CAL =HFLOAT1  
  ERA COLUMN,2  
  TNZ PARITH  
  CAL =HNGKOUN  
  ERA COLUMN+6,2  
  TNZ PARITH  
  CAL =HDUFF  
  ERA COLUMN+12,2  
  ANA =077777770000  
  CAL T16.1  
  STZ ROUNDF
<table>
<thead>
<tr>
<th>Line</th>
<th>Instruction</th>
<th>Comment</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>116.2</td>
<td>REMARK (FLOATING ROUNging OFF)</td>
<td></td>
<td>FT024740</td>
</tr>
<tr>
<td>116.1</td>
<td>ANA 0770000000000</td>
<td></td>
<td>FT024750</td>
</tr>
<tr>
<td></td>
<td>TNS PARITH</td>
<td></td>
<td>FT024760</td>
</tr>
<tr>
<td></td>
<td>STL ROUNDf</td>
<td></td>
<td>FT024770</td>
</tr>
<tr>
<td></td>
<td>REMARK (FLOATING ROUNding ON)</td>
<td></td>
<td>FT024780</td>
</tr>
<tr>
<td></td>
<td>TRA TI6.2</td>
<td></td>
<td>FT024790</td>
</tr>
<tr>
<td>1511.3</td>
<td>CAL =HSYMBol</td>
<td></td>
<td>FT024800</td>
</tr>
<tr>
<td></td>
<td>ERA COLUMN,2</td>
<td></td>
<td>FT024810</td>
</tr>
<tr>
<td></td>
<td>TNS PARITH</td>
<td></td>
<td>FT024820</td>
</tr>
<tr>
<td></td>
<td>CAL =HTABLE</td>
<td></td>
<td>FT024830</td>
</tr>
<tr>
<td></td>
<td>ERA COLUMN+6,2</td>
<td></td>
<td>FT024840</td>
</tr>
<tr>
<td></td>
<td>ARS 6</td>
<td></td>
<td>FT024850</td>
</tr>
<tr>
<td></td>
<td>TNS PARITH</td>
<td></td>
<td>FT024860</td>
</tr>
<tr>
<td></td>
<td>REMARK (SYMBOL TABLE REQUESTED)</td>
<td></td>
<td>FT024870</td>
</tr>
<tr>
<td></td>
<td>CAL 020</td>
<td></td>
<td>FT024880</td>
</tr>
<tr>
<td></td>
<td>ORS 18 SET MONITOR BIT ON</td>
<td></td>
<td>FT024890</td>
</tr>
<tr>
<td></td>
<td>SPRSW SKEND,NOSEQ,NOTRA,NODD</td>
<td></td>
<td>FT024900</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FT024910</td>
</tr>
<tr>
<td>Line</td>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>PROCL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>PROCG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>ENTER HERE VIA SPRSW MACROD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>PAI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>LFT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>BEND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>IS IT THE END STATEMENT.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>TRA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>PEND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>YES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>STA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>LBL5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>SET EXIT TO APPROPRIATE PROCESSOR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>NZT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>DOFLAG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>WAS PREVIOUS STATEMENT A DO.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>TRA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>++4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>NO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>LNT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>BXEQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>YES IT WAS. IS THIS STATEMENT EXECUTABLE.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>REM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>ERROR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>65,LBL3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>NON-EXECUTABLE STATEMENT NOT ALLOWED AS FIRST STATEMENT FOLLOWING A DO.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>REM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>STZ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>DOFLAG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>TURN OFF DO INDICATION</td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>REM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>LNT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>BXEQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>IS THIS STATEMENT EXECUTABLE.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>REM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>LBL3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>NO, SKIP OVER ANY WAITING TRANSFER CODING UNTIL NEXT EXECUTABLE STATEMENT IS ENCOUNTERED.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>REM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>THIS IS AN EXECUTABLE STATEMENT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>NZT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>CARD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>IS THERE A LABEL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>TRA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>LBL4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>NO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>LBL10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>LDQ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>CARD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>PROCESS THE LABEL.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>REM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>LOCATE,4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>52</td>
<td>MULTIPLE DEFINITION.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>53</td>
<td>TSX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>54</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>LOOK IT UP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>PAX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>57</td>
<td>BLHSX+MSTRG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>58</td>
<td>HAS LABEL PREVIOUSLY BEEN DEFINED EITHER AS A LABEL OR A FORMAT NAME.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>59</td>
<td>REM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>LBL8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>61</td>
<td>SIL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>62</td>
<td>IS PRECEDING STATEMENT A TRANSFER CMD FLAG AS DEFINED</td>
<td></td>
<td></td>
</tr>
<tr>
<td>63</td>
<td>NZT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>TRAFLG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>65</td>
<td>NO. THEREFORE PATH OF FLOW COMES IN FROM ABOVE.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>66</td>
<td>REM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>67</td>
<td>STI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>68</td>
<td>EQUIV,4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>69</td>
<td>SAVE LABEL POINTER IN CARD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>70</td>
<td>SLW</td>
<td></td>
<td></td>
</tr>
<tr>
<td>71</td>
<td>CARD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>72</td>
<td>ZET</td>
<td></td>
<td></td>
</tr>
<tr>
<td>73</td>
<td>TRAFLG</td>
<td></td>
<td></td>
</tr>
<tr>
<td>74</td>
<td>WAS PRECEDING STATEMENT A TRANSFER.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>75</td>
<td>TSX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>76</td>
<td>LBL11,4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>77</td>
<td>YES. GO TO POST-PROCESSOR ROUTINE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>78</td>
<td>NZT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>79</td>
<td>CARD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>IS THERE A LABEL.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>81</td>
<td>TRA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>82</td>
<td>++4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>83</td>
<td>NO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>84</td>
<td>LAX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>85</td>
<td>CARD,4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>86</td>
<td>YES. GET THE EQUIV POINIER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>87</td>
<td>CAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>88</td>
<td>PCOUNT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>89</td>
<td>DEFINE THE LABEL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>90</td>
<td>STA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>91</td>
<td>EQUIV,4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>92</td>
<td>LD1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>93</td>
<td>PROCSW</td>
<td></td>
<td></td>
</tr>
<tr>
<td>94</td>
<td>NZT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>95</td>
<td>DOPNTR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>96</td>
<td>ARE THERE ACTIVE DO'S</td>
<td></td>
<td></td>
</tr>
<tr>
<td>97</td>
<td>TRA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>98</td>
<td>LBL4A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>99</td>
<td>NO</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>REM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>LBL4A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>102</td>
<td>LAX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>103</td>
<td>DOPNTR,4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>104</td>
<td>NZT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>105</td>
<td>LBL4A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>106</td>
<td>REM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>107</td>
<td>LAX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>108</td>
<td>DOPNTR,4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>109</td>
<td>CAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>110</td>
<td>CARD</td>
<td></td>
<td></td>
</tr>
<tr>
<td>111</td>
<td>ERA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>112</td>
<td>DOSTAK,4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>113</td>
<td>TNZ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>114</td>
<td>LBL4A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>115</td>
<td>REM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>116</td>
<td>LAX</td>
<td></td>
<td></td>
</tr>
<tr>
<td>117</td>
<td>DOPNTR,4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>118</td>
<td>TNZ</td>
<td></td>
<td></td>
</tr>
<tr>
<td>119</td>
<td>LBL4A</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* This is terminal statement of a do.
LFT BDO+BRANS
ERROR 66, SKEND
REM TURN ON DO TERMINATION IN SKEND
STL DOTRFM
TRA LBL4A
REM
LBL4 ZET TRAFLG
ERROR 67, ++1
REM IS PRECEDING STATEMENT A TRANSFER
LbL4A STZ TRAFLG
LBL5 TRA **
REM NO. GO TO PROCESSOR
LbL11 TRA* TRAFLG
REM
LBL3 LXA PROC$W, 4
REM STATEMENT IS NOT EXECUTABLE... IS IT
TXL ++2, 4, PFMT-1
REM A FORMAT STATEMENT
TXL LBL6, 4, PFMT
REM YES, IT IS FORMAT
STZ CARD
REM CLEAR OUT ANY LABEL
TRA* LBL5
REM NOW VALIDATE FORMAT LABEL
LBL6 NZT CARD
REM IS THERE A LABEL
ERROR 60, SKEND
LDQ CARD
TSX LOCATE, 4
REM LOCATE LABEL IN S.T.
PAX #4
TMI LBL7
REM FORMAT NAME IS NEW
LFT BLHSX
REM LABEL NOT NEW. CHECK HOW IT GOT IN SYMBOL TABLE,
ERROR 58, SKEND
REM LABEL ALREADY DEFINED EITHER AS A STRAIGHT LABEL OR AS A STRING, FORMAT)
LbL7 SIL BLHSX+MSTRG
REM LABEL IS NEW. FLAG NAME AS STRING AND TURN ON DEFINED INDICATION
STI EQUIV, 4
TRA* LBL5
REM GO TO PROCESSOR
TTL PASS 1 - STATEMENT TERMINATION

* SKEND ROUTINE.
  * SKEND IS THE TERMINATION ROUTINE FOR ALL THE STATEMENT
  * PROCESSORS EXCEPT 'END'.
  * A TEST IS MADE TO DETERMINE IF THE POOL AREA HAS OVERFLOWED.
  * THE CELL 'FIRSTF' IS UPDATED. THE ADDRESS HOLDS THE NUMBER
  * OF STATEMENTS PROCESSED, AND THE DECREMENT HOLDS THE NUMBER
  * OF EXECUTABLE STATEMENTS.
  * IF THIS STATEMENT TERMINATES ONE OR MORE DO LOOPS, THE TER-
  * MINATION CODE IS PUT OUT.
  * IF THIS IS A CALL OR TRANSFER-TYPE STATEMENT, THE INDREG
  * ROUTINE IS ENTERED, WHICH RELEASES ALL NON-DO-INDEX-
  * CONTAINING XRS.
  * WORKING CELLS ARE RELEASED, AND THE ROUTINE THEN RETURNS
  * TO PROCESS THE NEXT CARD.

**SPACE** 3

**SKEND** LXA NXTLOC,4 TEST FOR POOL OVERFLOW
**TXL** 2,4,POOL-COLUMN-667
**STR** MSG100,,FINISH *****TEMPORARY RETURN LOCATION

**REM**
**CAL** FIRSTF GET STATEMENT COUNTER
**ADD** =1 BUMP
**LDI** PROCSW
**LFB** BKQ COUNT EXECUTABLE STATEMENTS IN DECREMENT
**ADD** =01000000
**STD** FIRSTF RESTORE STATEMENT COUNTER

**REM**
**ZET** DORIFM IS THIS STATEMENT THE TERMINATION OF A DO
**TSX** DORIFM,4 YES. CLOSE OUT ALL DO LOOPS TERMINATED BY

**REM**
**STZ** DORIFM THIS STATEMENT
**REI** DORIFM RESET THE DO TERMINATION FLAG

**REM**
**LDI** PROCSW GET PROCESSOR SWITCH
**LFB** BIRANS+BCALL WAS THIS A TRANSFER OR A CALL STATEMENT.
**TSX** INDREG,4 YES. CLEAR OUT THE NON-DO INDEX REGISTERS

**REM**
**LFI** BKEQ ALTER THE FLAG CELL IF A EXECUTABLE STATEMENT
**STZ** RASCAL
**LFT** BCALL NON-ZERO CELL IF A CALL STATEMENT
**STL** RASCAL THIS FLAG USED BY END PROCESSOR ONLY

**REM**
**STZ** WRKCEL RESET WORKING STORAGE
**TRA** STARIT GO GET NEXT CARD
<table>
<thead>
<tr>
<th>Scramble</th>
<th>Description</th>
<th>FT Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTL</td>
<td>PASS 1 PROCESSOR - SUBROUTINE</td>
<td>FT026360</td>
</tr>
<tr>
<td>SUBROUTINE NAME (A,B,C,...,Z)</td>
<td></td>
<td>FT026370</td>
</tr>
<tr>
<td>REM</td>
<td>ENTRY... PSUBR</td>
<td>FT026380</td>
</tr>
<tr>
<td>SPACE 3</td>
<td>PSUBR</td>
<td>FT026390</td>
</tr>
<tr>
<td>STZ</td>
<td>FNCTFL FLAG AS A SUBROUTINE</td>
<td>FT026400</td>
</tr>
<tr>
<td>AXC 10,2</td>
<td></td>
<td>FT026410</td>
</tr>
<tr>
<td>TRA</td>
<td>PFUNCT+2 TO FUNCTION PROCESSOR</td>
<td>FT026420</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FT026430</td>
</tr>
</tbody>
</table>
PAS 1 PROCESSOR - FUNCTION

FUNCTION NAME(A,B,C,...,Z)  
RE:  
ENTRY...  PFUNCT  
THE FUNCTION STATEMENT MUST BE THE FIRST STATEMENT OF THE  
PROGRAM. THAT IS, CELL FIRSTF MUST BE ZERO. THE NAME OF THE  
ROUTINE IS OBTAINED, CHECKED FOR VALIDITY, AND STORED IN CELL  
SBNAME.  
RE:  
ALL FUNCTIONS MUST HAVE AT LEAST ONE ARGUMENT. EACH ARGUMENT IS  
ENTERED IN SYMTAB FLAGGED AS A PARAMETER WITH MODE DETERMINED  
BY THE FIRST LETTER. A COUNT OF THE NUMBER OF PARAMETERS IS  
KEPT IN CELL NOARGS.  
RE:  
TO DISTINGUISH WHETHER THIS IS FUNCTION OR SUBROUTINE CODING  
TEST CELL FNCTFL WHICH WILL BE NON-ZERO FOR FUNCTIONS.  

SPACE 3  
PFUNCT  
STL FNCIFL FLAG AS A FUNCTION  
AXC 8,2 GET FUNCTION NAME  
ZET FIRSTF IS THIS THE 1ST STATEMENT OF PROGRAM  
ERROR 52,SKEND NO. SUBPROGRAM DECLARATION COMES TOO LATE  
TSX SCNBID,4  
SLW SBNAME SAVE THE NAME  
NZT LABEL HAS DECK LABEL ALREADY BEEN ASSIGNED.  
SLW LABEL NO  
NZT FNCIFL  
TRA PFUNCT1 SUBROUTINE CODING. OMIT NAME TEST  
CAL SCNB1 VALIDATE THE FUNCTION NAME  
STD 21  
LAC SCNB1-1,4  
TXI 1,4,1 NO. CHARACTERS IN XR4  
TXL PFUNCT1,4,3 NAME LESS THAN 4 LETTERS  
REM IS LAST LETTER F.  
TXI 1,2,1 TEST FOR TERMINAL F  
SKIP FCHAR LAST CHAR F  
TXI 2,2,1 LAST LETTER WAS F  
REM  
NAME IS A VALID FUNCTION NAME. DO NOT ENTER NAME IN SYMTAB NOW,  
BUT WAIT FOR ITS APPEARANCE ON THE LEFT HAND SIDE OF A SUBSTITU-  
TION STATEMENT. THEN AS A PART OF THE END STATEMENT PROCESSING,  
TEST THAT FUNCTION NAME IS IN SYMTAB WITH BLHXS FLAG BIT ON.  

PFUNCT1 STZ NOARGS GET ARGUMENTS  
SKIP1 LPCHAR  
TRA PFUNCT4  
NZT FNCIFL SUBROUTINE CODING. MAY BE NO ARGUMENTS  
TRA PFUNCT2 FUNCTION MUST HAVE AT LEAST ONE ARGUMENT  
ERROR 7,SKEND  
PFUNCT3 TSX SCNBID,4  
XCL TSX LOCATE,4  
TPL PFUNCT3 SYMBOL ALREADY IN SYMTAB. ERROR  
PAX 1,4  
CAL SYMTAB,4 DETERMINE MODE DD ARGUMENT  
LAS =04777777777777  
SIL MREAL  
TRA PFUNCT5  
LAS =03777777777777  
SIL MINTG
TRA  PFUNC5  FT027040
SIL  MREAL  FT027050
PFUNC5  S1L  BARGT+BLSX  APPEND PARAMETER FLAG AND DEFINITION FLAG  FT027060
STI  EQUIV,4  FT027070
CLA  NOARGS  FT027080
ADD  =1  FT027090
STO  NOARGS  FT027100
ALS  15  FT027110
ORS  EQUIV,4  FT027120
REM  IS THIS THE LAST ARGUMENT  FT027130
SK1P  CMCHAR  FT027140
TRA  PFUNC4  FT027150
SK1P  RPCHAR  FT027160
TRA  PFUNC3  FT027170
REM  NO MORE ARGUMENTS  FT027180
PFUNC2  SK1P  63  FT027190
TRA  PFUNC3  FT027200
PFUNC6  STZ  RTNFLG  TURN RETURN STATEMENT INDICATION OFF  FT027210
TRA  SKEND  FT027220
* TTL PROCESSOR - RETURN
* RETURN
* REM
* ENTRY... PRTN
* VALID ONLY IN SUBROUTINE OR FUNCTION CODING.
* RETURN STATEMENT MUST APPEAR AT LEAST ONCE IN A FUNCTION
* SUBPROGRAM. RTNFLG CONTAINS A COUNT OF RETURN STATEMENTS.
* REM
* THIS STATEMENT CAUSES A TRANSFER TO THE END OF THE TEXT,
* WHERE THE ACCUMULATOR WILL BE LOADED (FOR FUNCTIONS). THEN
* A TRANSFER BACK UP TO THE PROLOGUE EXIT SEQUENCE IS GENERATED.
* SPACE 3
* PRTN NZT SBNAME IS THIS SUBROUTINE
* ERROR 53,SKEND MAIN PROGRAM
* CAL MODFLG FUNCTION. IS COL 1 OF THIS STATEMENT
* NZT RTNFLG THE SAME AS ON ALL OTHER RETURN STATEMENTS
* SLW RTNMOD ENCOUNTED SO FAR.
* LAS RTNMOD
* TRA **2 NO
* TRA **2 YES
* ERROR 54,**1
* CLA RTNFLG STEP RETURN STATEMENT COUNT BY 1
* ADD =1
* AXI RTNFLG
* STO RTNFLG
* AXI PRTN3,4 SET TRAFLG FOR POSSIBLE TRANSFER CODING
* SXA TRAFLG,4 TO JUMP TO END OF TEXT IN CASE NEXT CARD
* REM IS NOT AN END CARD.
* TRA SKEND
* REM
* REM
* REM
* ENTER HERE IF TRAFLG IS ON. TEST IF THIS STATEMENT IS AN END
* STATEMENT. IF SO OMIT THE TRA INSTR. NORMALLY GENERATED BY
* THE RETURN STATEMENT.
* PRTN3 SXA PRTN3X,4
* LXA PROCSTW,4
* TXL PRTN4,4,PEND-1
* TXH PRTN4,4,PEND
* PRTN3X AXI **,4 STATEMENT IS END.
* TRA 1,4 STATEMENT NOT END.
* REM
* PRTN4 CAL TRAEND NOT END.
* TSX CITBLD,4,R
* TRA PRTN3X
* REM
* TRAEND TRA EQUIV-LOCEND
* FT027230
* FT027240
* FT027250
* FT027260
* FT027270
* FT027280
* FT027290
* FT027300
* FT027310
* FT027320
* FT027330
* FT027340
* FT027350
* FT027360
* FT027370
* FT027380
* FT027390
* FT027400
* FT027410
* FT027420
* FT027430
* FT027440
* FT027450
* FT027460
* FT027470
* FT027480
* FT027490
* FT027500
* FT027510
* FT027520
* FT027530
* FT027540
* FT027550
* FT027560
* FT027570
* FT027580
* FT027590
* FT027600
* FT027610
* FT027620
* FT027630
* FT027640
* FT027650
* FT027660
* FT027670
ENTRY...  ICALLX
REM

A SUBROUTINE MAY NOT CALL ITSELF. THE NAME IN THE CALL STATEMENT
MAY NOT BE USED AS A VARIABLE. IT IS LEGAL FOR THE NAME TO BE
USED ALSO AS A FUNCTION NAME, EITHER FORTRAN OR LIBRARY.

THE ARGUMENT LIST IS PROCESSED BY THE COMPIL ROUTINE IN A MANNER
VERY SIMILAR TO PROCESSING A FORTRAN FUNCTION CALL.

IF THERE IS NO ARGUMENT LIST, THE CODE TSX NAME,4 IS GENERATED
HERE AND NOW.
SPACE 3

ICALLX

AXC 4,2
TSX SCNBOD,4  GET SUBROUTINE NAME
TMI ERROR8  NOT SYMBOLIC
REM
LAS SBNAME  COMPARE WITH THIS PROGRAM'S NAME
TRA **2  PROGRAM MAY NOT CALL ITSELF
REM
SLW FORM69  SAVE NAME OF SUBROUTINE
XCL
SLW ASTACK  SAVE EQUV POINTER
PAX 4  NEW ITEM IN SYMTAB
TMI ICALL2  OLD ITEM IN SYMTAB
REM
LFI BEXTF+BART  OLD ITEM. MUST BE EXTERNAL OR ARGUMENT
LFT BVX+BBARY+BCOMM+BEQV
ERROR 69,SKEND  MULTIPLE USAGE OF SUBROUTINE NAME
REM
RIL 700000+BLIBF  RESET MODE BITS AND LIBRARY FLAG
REM

ICALL2
CAL FORM69  GET BCD NAME
LAS =H100000  DETERMINE A 'MODE'. THIS IS NECESSARY TO
TRA **1  FAKE OUT LATER SECTIONS OF COMPILER.
LAS =HNI1111
REM
SIL BEXTF+MREAL  MARK AS EXTERNAL AND REAL
TRA **2  MARK AS FIXED AND EXTERNAL
REM

ICALL1
STI EQUIV,4  SET EQUIV WORD
CAL COLUMN,2  GET NEXT CHARACTER
ERA =H1
REM
AS 30
TNZ 1CALL4  NOT A LEFT PAREN
REM
CAL ASTACK  SET ASTACK WORD. DO NOT MOVE
STA ASTACK
REM
TSX CMPLCL,4  PROCESS THE CALLING SEQUENCE
TRA SKEND
REM
REM

1CALL4
SKIP 63  MUST BE END OF STATEMENT
ERROR 7,SKEND
REM
TSX PUTCL2,4  RESTORE CELL 2
LFT BAR-G  IS IT AN ARGUMENT.
REM
<table>
<thead>
<tr>
<th>TRA</th>
<th>1CALL3</th>
<th>YES</th>
<th>FT028280</th>
</tr>
</thead>
<tbody>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FT028290</td>
</tr>
<tr>
<td>CODECO</td>
<td>ASTACK, TSX4,T,XRFLAG</td>
<td>NO ARGUMENTS IN CALL STATEMENT</td>
<td>FT028300</td>
</tr>
<tr>
<td>TRA</td>
<td>SKEND</td>
<td>CODE IS... TSX NAME,4</td>
<td>FT028310</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FT028320</td>
</tr>
<tr>
<td>1CALL3</td>
<td>CAL</td>
<td>ASTACK</td>
<td>FT028330</td>
</tr>
<tr>
<td>TSX</td>
<td>PROLOG,4</td>
<td>MAKE PROLOGUE ENTRIES</td>
<td>FT028340</td>
</tr>
<tr>
<td>CODEC</td>
<td>TSX4,N,XRFLAG</td>
<td>CODE IS... TSX **,4</td>
<td>FT028350</td>
</tr>
<tr>
<td>TRA</td>
<td>SKEND</td>
<td></td>
<td>FT028360</td>
</tr>
</tbody>
</table>
* TTL PASS 1 PROCESSOR - END
* END STATEMENT PROCESSOR
* REM ENTRY... PEND
* REM IF THIS IS A SUBPROGRAM, A RETURN SEQUENCE IS GENERATED HERE
* FOR FUNCTIONS, AND FOR SUBROUTINES WHICH CONTAIN A
* RETURN STATEMENT. THE CELL 'LOCEND', WHICH
* IS USED BY THE RETURN STATEMENTS, IS NOW SET TO THE CURRENT
* VALUE OF PCOUNT.
* REM IF THE LAST EXECUTABLE STATEMENT WAS NOT A TRANSFER-TYPE,
* THEN AN ERROR MESSAGE IS ISSUED.
* REM
* PEND NZT TRAFLG LAST EXECUTABLE STATEMENT MUST BE A
* ZET RASCAL TRANSFER OR A CALL.
* TRA **2 ERROR 33,**3 PROGRAM RUNS OFF THE END
* REM ZET TRAFLG WAS THE LAST EXEC. STATEMENT A TRANSFER.
* TSX LBL11,4 YES. GO TO THE POST-PROCESSOR ROUTINE
* REM TO TURN OUT ANY WAITING TRANSFER CODING.
* REM
* PEND4 NZT SBNAME IS IT A SUBPROGRAM.
* TRA PEND99 NO
* CLA PCOUNT SET THE VALUE OF THE END CARD ENTRY
* STO LOCEND FOR RETURN STATEMENTS
* NZT FNCTFL IF A FUNCTION, THEN RTNFLG MUST BE ON.
* TRA PEND15 NOT A FUNCTION
* NZT RTNFLG
* ERROR 55,**1
* REM
* LDQ SBNAME GET THE NAME OF THIS FUNCTION
* TSX LOCATE,4 LOOK IT UP IN SYMBOL TABLE
* PAX ^4 PRESERVE THE POINTER
* TPL PEND6 OLD ITEM IN THE TABLE
* STZ SYMTAB,4 REMOVE THE SYMBOL FROM TABLE
* ERROR 56,PEND15 AND PUT OUT AN ERROR MESSAGE
* REM
* PEND6 LNT BLHSX HAS NAME BEEN DEFINED.
* ERROR 56,PEND15 NO.
* LFT BARGI+BARRY+BEXIF+BINTF
* ERROR 85,PEND15 ILLEGAL USE OF THE NAME
* REM
* SXA PEND10,4 SAVE THE POINTER
* CAL RTNMOO GET MODE OF RETURNS
* LAS =1
* TRA PEND13 DOUBLE OR COMPLEX
* TRA PEND14 BOOLEAN
* REM
* CAL PEND10 GET EQUIV POINTER FOR NAME
* CODEO CLAV CODE IS... CLA NAME
* REM
* PEND15 CAL TRA GET A TRANSFER OP CODE
* ZET RTNFLG IS THERE A RETURN STATEMENT IN THIS PROGRAM?
* TSX CITBLD,4,X YES. PUT OUT THE TRA TO RETURN SEQUENCE
* REM
* PEND99 STZ LPASS1 KILL LISTING OF THIS MESS
* TSX CITBLD,4,END PASS ON THE END FLAG TO CITBLD
<table>
<thead>
<tr>
<th>TRA</th>
<th>PPASS1</th>
<th>GO NOW TO POST PASS 1</th>
<th>FT028970</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPACE</td>
<td></td>
<td></td>
<td>FT028980</td>
</tr>
<tr>
<td>PEND14</td>
<td>CAL</td>
<td>PEND10 GET POINTER TO NAME</td>
<td>FT028990</td>
</tr>
<tr>
<td>CODEO</td>
<td>CAL,V</td>
<td>CODE IS... CAL NAME</td>
<td>FT029000</td>
</tr>
<tr>
<td>TRA</td>
<td>REM</td>
<td>PEND15</td>
<td>FT029010</td>
</tr>
<tr>
<td>PEND13</td>
<td>CAL</td>
<td>PEND10 GET POINTER TO NAME</td>
<td>FT029030</td>
</tr>
<tr>
<td>CODEO</td>
<td>CLA,V</td>
<td></td>
<td>FT029040</td>
</tr>
<tr>
<td>CAL</td>
<td>REM</td>
<td>PEND10</td>
<td>FT029050</td>
</tr>
<tr>
<td>CODEO</td>
<td>LDQ,G</td>
<td></td>
<td>FT029060</td>
</tr>
<tr>
<td>CODEN</td>
<td>ST0777</td>
<td></td>
<td>FT029070</td>
</tr>
<tr>
<td>CODEN</td>
<td>STQ776</td>
<td></td>
<td>FT029080</td>
</tr>
<tr>
<td>TRA</td>
<td>REM</td>
<td>PEND15</td>
<td>FT029090</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FT029100</td>
</tr>
</tbody>
</table>
ENTRY... 1FCARD

REM

REM

1FCARD

AUC 0,2  RESET POINTER TO COLUMN

TSX SCNBCC,4  GET NAME

TMI ERRD7  NOT A SYMBOLIC NAME

REM

SLW FORM69  SAVE NAME IN ERROR MESSAGE CELL

XCL

TSX LOCATE,4  PUT NAME IN SYMBOL TABLE

TMI 1FCAR1  NEW SYMBOL

REM

LFI BVARB+BCOMN+BARRY+BEQV ALL FLAGS MUST BE OFF

ERROR 69,1FCAR2

LFT BARGT IS IT AN UNCOMMITTED ARGUMENT.

SIL BEXITF YES, TURN ON EXTERNAL FUNCTION FLAG

LNI BEXITF EXTERNAL FUNCTION FLAG MUST NOW BE ON

ERROR 69,1FCAR2

TRA 1FCAR2

REM

1FCAR1

PAX 4  GET EQUIV POINTER

CAL FORM69 GET BCD NAME OF FUNCTION

LAS =H100000 DETERMINE THE MODE, THIS IS NECESSARY

TRA ++1 TO KEEP LATER SECTIONS OF COMPILER

LAS =HN1111 FROM LOUSING UP.

SIL BEXITF+MREAL MARK AS EXTERNAL AND REAL

TRA ++2

SIL BEXITF+MIN1G MARK AS EXTERNAL AND INTEGER

STL EQUIV,4  RESET EQUIV WORD

REM

1FCAR2

SK1P CMCHAR TEST FOR COMMA

TRA 1FCARD+1 COMMA FOUND, RETURN FOR ANOTHER NAME

REM

SK1P 63 TEST FOR END OF STATEMENT

ERROR 3,SKEND NOT EOS, ERROR

TRA SKEND
COMMON A,B,C,...
REM
ENTRY... ICOMNX
AN ENTRY IS MADE IN DMTBL FOR EACH COMMON VARIABLE.
The FIRST WORD OF THE DMTBL ENTRY CONTAINS THE VALUE OF THE CURRENT COMMON COUNTER IN THE DECREMENT, AND EITHER THE EQUTBL OR EQUIV TABLE POINTER IN THE ADDRESS, DEPENDING IF BEQU.
IS ON OR NOT. IF THIS IS A NEW DMTBL ENTRY, THREE WORDS ARE RESERVED TO ALLOW FOR POSSIBLE FUTURE DIMENSION OR ASSIGN INFO.
STORAGE IS ALREADY AVAILABLE IN DMTBL FOR THIS VARIABLE.
IF IT HAS PREVIOUSLY APPEARED IN A DIMENSION STATEMENT OR IN EITHER AN ASSIGN OR ASSIGNED GO TO STATEMENT.
THE EQUIV WORD ADDRESS GETS THE DMTBL POINTER POSITION.
The FLAG BITS BCOMM AND BLHSX ARE TURNED ON.
A SYMBOL MAY NOT APPEAR MORE THAN ONCE IN A COMMON DECLARATION,
AND ALL SYMBOLS MUST BE NON-SUBSCRIPTED NAMES.
REM
NOTE... DPOINT AND EPOINT ARE SYNONYMOUS WITH NXTLOC, AND DMTBL AND EQUTBL ARE SYNONYMOUS WITH POOL.
SPACE 3
ICOMNX
AXC 6,2
TSX SCNBD,4
IPL
ERROR 2,SKNED
REM
SLW FORM69
XCL
TSX LOCATE,4
PAX ,1
TPL ICOMN1
CAL FORM69
LAS =H100000
TRA ++1
LAS =H10000
SIL MREAL
TRA ++2
SIL MIN16
PXA ,1
REMN
LXA DPOINT,4
STA DMTBL,4 (DMTBL) ADDRESS = EQUIV POINTER
REM
CLA DPOINT
ADD =3
STA DPOINT
PXA ,4
REM
SIL BCOMM+BLHSX
STA EQUIV,1
REM
CAL CPPOINT
ADD =1817
STA CPPOINT
REM
STO DMTBL,4 (DMTBL) DECREMENT = CPPOINT
REM
SIL CMCHAR
CHECK FOR COMMA
<table>
<thead>
<tr>
<th>TRA</th>
<th>ICOMNX+1</th>
<th>COMMA FOUND. RETURN FOR ANOTHER SYMBOL</th>
<th>FT030080</th>
</tr>
</thead>
<tbody>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FT030090</td>
</tr>
<tr>
<td>SKIP1</td>
<td>63</td>
<td>NOT A COMMA. MUST BE END OF STATEMENT</td>
<td>FT030100</td>
</tr>
<tr>
<td>TRA</td>
<td>SKEND</td>
<td>OKAY.</td>
<td>FT030110</td>
</tr>
<tr>
<td>ERROR</td>
<td>5,SKEND</td>
<td></td>
<td>FT030120</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FT030130</td>
</tr>
<tr>
<td>ICOMN1</td>
<td>LFI</td>
<td>BCOMN OLD SYMBOL IS COMMON BIT ON.</td>
<td>FT030140</td>
</tr>
<tr>
<td>TRA</td>
<td>ICOMN6</td>
<td>YES. ERROR</td>
<td>FT030150</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FT030160</td>
</tr>
<tr>
<td>LFI</td>
<td>BEXIF+BINTF</td>
<td>MAY NOT BE MARKED AS SUBPROGRAM NAME</td>
<td>FT030170</td>
</tr>
<tr>
<td>ERROR</td>
<td>69,ICOMN2</td>
<td></td>
<td>FT030180</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FT030190</td>
</tr>
<tr>
<td>LFI</td>
<td>BARRY+BASSN</td>
<td>NO. IS STORAGE ALREADY AVAILABLE.</td>
<td>FT030200</td>
</tr>
<tr>
<td>TRA</td>
<td>ICOMN3</td>
<td>YES.</td>
<td>FT030210</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FT030220</td>
</tr>
<tr>
<td>LFI</td>
<td>BEQUV</td>
<td>NO. IS EQUIVALENCE BIT ON.</td>
<td>FT030230</td>
</tr>
<tr>
<td>PIA</td>
<td>YES. (DIMTL) ADDRESS = EQUtbl POINTER</td>
<td>FT030240</td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>ICOMN4</td>
<td></td>
<td>FT030250</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FT030260</td>
</tr>
<tr>
<td>ICOMN3</td>
<td>PIA</td>
<td></td>
<td>FT030270</td>
</tr>
<tr>
<td>PAX</td>
<td>4</td>
<td></td>
<td>FT030280</td>
</tr>
<tr>
<td>TRA</td>
<td>ICOMN5</td>
<td></td>
<td>FT030290</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FT030300</td>
</tr>
<tr>
<td>ICOMNO</td>
<td>CAL</td>
<td>SYMIAB,1 GET BCD NAME OF SYMBOL IN ERROR</td>
<td>FT030310</td>
</tr>
<tr>
<td>SLW</td>
<td>FORM27</td>
<td>PUT IN ERROR MESSAGE</td>
<td>FT030320</td>
</tr>
<tr>
<td>ERROR</td>
<td>27,ICOMN2</td>
<td>PUT OUT MESSAGE AND RETURN</td>
<td>FT030330</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FT030340</td>
</tr>
</tbody>
</table>
**TIL**  PASS 1 PROCESSOR - DIMENSION

* DIMENSION A(I,J,K), B(I,J), C(I), ....

* REM

* ENTRY... 1DIMNX

* AN ENTRY IN THE DIMENSION STACK DIMTBL IS MADE FOR EACH
* VARIABLE ENCOUNTERED IN A DIMENSION STATEMENT. THREE
* WORDS OF STORAGE IN DIMTBL ARE ALREADY AVAILABLE IF THE
* VARIABLE HAS PREVIOUSLY APPEARED IN A COMMON STATEMENT.

* IF NOT, A TWO OR THREE WORD ENTRY IS MADE, DEPENDING IF THE
* ARRAY IS ONE-DIMENSIONAL OR MORE THAN ONE-DIMENSIONAL.

* WORD TWO CONTAINS THE TOTAL LENGTH SPECIFIED BY THE
* SUBSCRIPTS (ADDRESS FIELD), THE FIRST DIMENSION (DECREMENT
* FIELD), AND THE NUMBER OF DIMENSIONS (PREFIX BITS).

* IF NEEDED, WORD THREE, IF NECESSARY, CONTAINS DIMENSIONS TWO (ADDRESS
* FIELD) AND THREE (DECREMENT FIELD).

* REM

* IF BEQV IS ON, THE EQUIV BLT pointer (located in the address
* OF THE EQUIV ENTRY) IS PUT IN THE ADDRESS OF THE WORD ONE
* OF THE DIMTBL ENTRY. IF BEQV IS OFF, THE EQUIV TABLE
* POSITION IS PUT IN THE DIMTBL ADDRESS. IN EITHER EVENT,

* THE ADDRESS OF THE EQUIV TABLE ENTRY IS SET TO THE
* DIMTBL POSITION. BARRY IS TURNED ON.

* REM

* IF COLUMN 1 CONTAINS A 'D' OR 'I', THE BIT BDOWN IS TURNED ON,
* INDICATING THAT THIS ARRAY WILL RECEIVE A DOUBLE-LENGTH STORAGE
* ASSIGNMENT. A SYMBOL MAY NOT APPEAR MORE
* THAN ONCE IN A DIMENSION STATEMENT, AND EACH SYMBOL MUST
* BE SUBSCRIPTED. AN ASSIGN VARIABLE MAY NOT APPEAR IN A
* DIMENSION STATEMENT.

* AN ARRAY MUST APPEAR IN A DIMENSION STATEMENT PRIOR TO ITS
* FIRST USAGE IN AN EXECUTABLE STATEMENT.

* REM

* NOTE... DPOINT AND EPOINT ARE SYNONYMOUS WITH NXTLOC, AND
* DIMTBL AND EQUITBL ARE SYNONYMOUS WITH POOL.

**1DIMNX**

* SPACE 3

**AXC** 9,2  
RESET POINTER

**REM**

**1DIMNY**  
**TSX** SCN8CD,4  
GET ARRAY NAME

**TMI** ERROR8  
INVALID ARRAY NAME

**SLW** P  
SAVE BCD NAME

**REM**

**SKIP** LPCHAR  
TEST FOR LEFT PAREN

**ERROR** 9,SKEND  
NO PAREN

**REM**

**LDQ** =1817  
SET MULTIPLIER

**STQ** *.P+6  
SET MULTIPLIER

**AXT** 1,1  

**REM**

**1DIMN1**  
**TSX** SCN8CD,4  
GET DIMENSION

**TMI** *.4  
IS SUBSCRIPT A NUMBER.

**REM**

**ERROR** 17,*.1  
NO. (ZERO SUBSCRIPT CAUGHT BY SCN8CD).

**CAL** =1817  
FAKE A SUBSCRIPT

**TRA** 10IMN5  

**REM**

**TSX** BCDFIX,4  
CONVERT TO FIXED POINT NUMBER

**SLW** *.P+5,1  
SAVE DIMENSION

**XCA**

**TOV** *.1  
TURN OFF OVERFLOW LIGHT

**MPY** *.P+6  
COMPUTE TOTAL STORAGE REQUIRED
<p>| ALS  | 19 | TRY TO TURN ON OVERFLOW | FTO30950 |
| TND  | *+2 | CHECK FOR OVERFLOW | FTO30960 |
| ERROR | 22, *+3 | ARRAY TOO BIG FOR CORE | FTO30970 |
| REM  | 2 | RESTORE TO DECREMENT | FTO30980 |
| IDIMN5 | STD | SIZE OF ARRAY | FTO30990 |
| REM  | CMCHAR | TEST FOR COMMA (MORE SUBSCRIPTS) | FTO31000 |
| TRA  | *+3 | NOT A COMMA | FTO31010 |
| TXH  | ERR018, 1 , 2 | TOO MANY DIMENSIONS | FTO31020 |
| TXI  | IDIMN1 , 1 , 1 | BUMP COUNT AND LOOP BACK | FTO31030 |
| STZ  | *P+4, 1 | SET STOP CELL | FTO31040 |
| SKIP | RCP CHAR | TEST FOR RIGHT PAREN | FTO31050 |
| ERROR | 9, SKEND | NO PAREN | FTO31060 |
| PDX  | *1 | GET NUMBER OF DIMENSIONS | FTO31070 |
| LDQ  | *P+6 | GET TOTAL SIZE OF ARRAY | FTO31080 |
| RQL  | 3 | MOVE INTO POSITION | FTO31090 |
| LGL  | 15 | SET PREFIX AND ADDRESS | FTO31100 |
| ORA  | *P+4 | MOVE FIRST DIMENSION INTO DECREMENT | FTO31110 |
| SLW  | *P+6 | SAVE FIRST WORD IN DIMENSION DECLARATION | FTO31120 |
| REM  | *P | TEST EACH ARRAY NAME TO SEE IF IT | FTO31130 |
| LAS  | HFORMAT | IS 'FORMAT' OR 'IF', THIS IS TO SPEED | FTO31140 |
| TRA  | *+2 | THE IF AND FORMAT PROCESSORS | FTO31150 |
| STL  | FORFLG | FORMAT NAME FOUND, SET THE FLAG | FTO31160 |
| LAS  | HIF | | FTO31170 |
| TRA  | *+2 | | FTO31180 |
| STL  | IFFLAG | IF ARRAY FOUND, SET THE FLAG | FTO31190 |
| XCL  | | | FTO31200 |
| REM  | LOCATE, 4 | PUT IN SYMTAB | FTO31210 |
| REM  | | | FTO31220 |
| STZ  | *P+8 | SET FLAG FOR OLD SYMBOL | FTO31230 |
| IPL  | *+2 | IS IT OLD SYMBOL | FTO31240 |
| STL  | *P+8 | NEW SYMBOL, SET FLAG | FTO31250 |
| REM  | *P | ARE ANY OF THESE FLAGS ON | FTO31260 |
| PAX  | 4 | | FTO31270 |
| LFT  | BARRY+BVARB+BEXIF+BINIF | ARE ANY OF THESE FLAGS ON | FTO31280 |
| TRA  | IDIM13 | YES, ERROR | FTO31290 |
| REM  | | | FTO31300 |
| SIL  | BARRY | NO, TURN ON ARRAY FLAG | FTO31310 |
| CAL  | *P | GET BCD NAME OF ARRAY | FTO31320 |
| LAS  | HI00000 | NEW SYMBOL, SET MODE BITS | FTO31330 |
| TRA  | *+1 | | FTO31340 |
| LAS  | HI00000 | FLOATING POINT | FTO31350 |
| TRA  | *+3 | FIXED POINT | FTO31360 |
| SIL  | MINTG | | FTO31370 |
| TRA  | IDIM17 | YES | FTO31380 |
| REM  | | | FTO31390 |
| LDQ  | MODFLG | GET COLUMN 1 FLAG CELL | FTO31400 |
| RQL  | 34 | MOVE BIT 34 INTO MQ SIGN POSITION | FTO31410 |
| TQP  | IDIM17 | IS IT '0' OR '1' | FTO31420 |
| SIL  | BDUB | YES, SET BDUB BIT | FTO31430 |
| REM  | | | FTO31440 |
| IDIM17 | ZET | IS IT NEW SYMBOL | FTO31450 |
| TRA  | IDIM12+2 | YES | FTO31460 |
| REM  | | | FTO31470 |</p>
<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Rtn. Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>LFT BCOMN</td>
<td>OLD SYMBOL. IS IT IN COMMON.</td>
<td>F031550</td>
</tr>
<tr>
<td>TRA 1DIM10</td>
<td>YES.</td>
<td>F031560</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>F031570</td>
</tr>
<tr>
<td>LNT BEQV</td>
<td>NO. IS IT IN EQUIVALENCE STACK.</td>
<td>F031580</td>
</tr>
<tr>
<td>TRA 1DIM12</td>
<td>NO.</td>
<td>F031590</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>F031600</td>
</tr>
<tr>
<td>PIA</td>
<td>YES.</td>
<td>F031610</td>
</tr>
<tr>
<td>1DIMN8</td>
<td>GET DIMTBL POINTER</td>
<td>F031620</td>
</tr>
<tr>
<td>LXA DPOINT,1</td>
<td>GET DIMTBL POINTER IN ADDRESS</td>
<td>F031630</td>
</tr>
<tr>
<td>STA DIMTBL,1</td>
<td>PUT CORRECT POINTER IN ADDRESS</td>
<td>F031640</td>
</tr>
<tr>
<td>PXA</td>
<td>RESTORE EQUIV ENTRY</td>
<td>F031650</td>
</tr>
<tr>
<td>1DIM11</td>
<td>EQUIV,4</td>
<td>F031660</td>
</tr>
<tr>
<td>STA EQUIV,4</td>
<td>PUT CORRECT POINTER IN ADDRESS</td>
<td>F031670</td>
</tr>
<tr>
<td>TXI</td>
<td>BUMP DIMTBL POINTER</td>
<td>F031680</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>F031690</td>
</tr>
<tr>
<td>1DIMN9</td>
<td>INSERT CODE WORD IN DIMTBL</td>
<td>F031700</td>
</tr>
<tr>
<td>CAL .P+6</td>
<td>BUMP DIMTBL POINTER</td>
<td>F031710</td>
</tr>
<tr>
<td>SLW DIMTBL,1</td>
<td>PUT IN DIMENSION STACK WITH ZERO PREFIX</td>
<td>F031720</td>
</tr>
<tr>
<td>TXI</td>
<td>BUMP DIMTBL POINTER</td>
<td>F031730</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>F031740</td>
</tr>
<tr>
<td>NZT .P+3</td>
<td>IS THERE MORE THAN ONE DIMENSION.</td>
<td>F031750</td>
</tr>
<tr>
<td>TRA 1DIMN6</td>
<td>NO.</td>
<td>F031760</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>F031770</td>
</tr>
<tr>
<td>ARS 18</td>
<td>SHIFT INTO ADDRESS</td>
<td>F031780</td>
</tr>
<tr>
<td>QRA .P+3</td>
<td>COMBINE WITH THE SECOND DIMENSION</td>
<td>F031790</td>
</tr>
<tr>
<td>SLW DIMTBL,1</td>
<td>PUT IN DIMENSION STACK WITH ZERO PREFIX</td>
<td>F031800</td>
</tr>
<tr>
<td>TXI</td>
<td>BUMP DIMTBL POINTER</td>
<td>F031810</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>F031820</td>
</tr>
<tr>
<td>1DIMN6</td>
<td>IS COMMON FLAG ON.</td>
<td>F031830</td>
</tr>
<tr>
<td>LNT BCOMN</td>
<td>NO. UPDATE DIMTBL POINTER</td>
<td>F031840</td>
</tr>
<tr>
<td>SXX DPOINT,1</td>
<td>CHECK FOR FURTHER ENTRIES</td>
<td>F031850</td>
</tr>
<tr>
<td>REM</td>
<td>GET NEXT ENTRY</td>
<td>F031860</td>
</tr>
<tr>
<td>1DIM14 SKIP</td>
<td>END OF STATEMENT REQUIRED</td>
<td>F031870</td>
</tr>
<tr>
<td>1DIMN7</td>
<td></td>
<td>F031880</td>
</tr>
<tr>
<td>TRA</td>
<td>SKEND</td>
<td>F031890</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>F031900</td>
</tr>
<tr>
<td>IDIM12 LNT</td>
<td>NOT IN EQUIVALENCE. IS IT A PARAMETER.</td>
<td>F031910</td>
</tr>
<tr>
<td>1DIM15</td>
<td>NO. ERROR. SHOULD NOT BE AN OLD SYMBOL</td>
<td>F031920</td>
</tr>
<tr>
<td>TRA</td>
<td>YES. TREAT AS NEW SYMBOL.</td>
<td>F031930</td>
</tr>
<tr>
<td>PXA .K</td>
<td></td>
<td>F031940</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>F031950</td>
</tr>
<tr>
<td>1DIM15</td>
<td>GET BCD NAME OF SYMBOL IN ERROR</td>
<td>F031960</td>
</tr>
<tr>
<td>CAL SYMTAB,4</td>
<td>PUT IN THE ERROR MESSAGE</td>
<td>F031970</td>
</tr>
<tr>
<td>SLW FORM19+6</td>
<td>PUT OUT MESSAGE AND RETURN</td>
<td>F031980</td>
</tr>
<tr>
<td>SLW FORM19+6</td>
<td>PUT OUT MESSAGE AND RETURN</td>
<td>F031990</td>
</tr>
<tr>
<td>ERROR 19,1DIM12+2</td>
<td>PUT OUT MESSAGE AND RETURN</td>
<td>F032000</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>F032010</td>
</tr>
<tr>
<td>IDIM10</td>
<td>COMMON VARIABLE</td>
<td>F032020</td>
</tr>
<tr>
<td>PIA .1</td>
<td>GET DIMTBL POINTER</td>
<td>F032030</td>
</tr>
<tr>
<td>TRA 1DIM11</td>
<td>GET DIMTBL POINTER</td>
<td>F032040</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>F032050</td>
</tr>
<tr>
<td>1DIM13 CAL</td>
<td>GET BCD NAME OF ARRAY IN ERROR</td>
<td>F032060</td>
</tr>
<tr>
<td>SLW FORM21</td>
<td>PUT NAME INTO THE ERROR MESSAGES</td>
<td>F032070</td>
</tr>
<tr>
<td>SLW FORM28+2</td>
<td>PUT NAME INTO THE ERROR MESSAGES</td>
<td>F032080</td>
</tr>
<tr>
<td>SLW FORM69</td>
<td>PUT NAME INTO THE ERROR MESSAGES</td>
<td>F032090</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>F032100</td>
</tr>
<tr>
<td>LFT BARRY</td>
<td>IS NAME ALREADY IN DIMTBL AS ARRAY.</td>
<td>F032110</td>
</tr>
<tr>
<td>ERROR 21,**1</td>
<td>YES.</td>
<td>F032120</td>
</tr>
<tr>
<td>LFT BEXTF+BINTF</td>
<td></td>
<td>F032130</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F032140</td>
</tr>
<tr>
<td>ERROR</td>
<td>LFT</td>
<td>NAME USED AS VARIABLE AND FUNCTION IS VARIABLE IN ASSIGN-TYPE STATEMENTS.</td>
</tr>
<tr>
<td>-------</td>
<td>-----</td>
<td>------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ERROR</td>
<td>28</td>
<td>YES</td>
</tr>
<tr>
<td>TRA</td>
<td>1DIM14</td>
<td>RETURN FOR NEXT ARRAY IN THE LIST</td>
</tr>
</tbody>
</table>
THE EQUIVALENCE TABLE EQUTBL IS COMPOSED OF 'CHAINS', WHICH
IN TURN ARE FORMED FROM ONE OR MORE 'STACKS'. A CHAIN CONTAINS
ALL SYMBOLS RELATED TO EACH OTHER THROUGH EQUIVALENCE
DECLARATIONS. A NEW STACK IS FORMED FROM THE NEW ITEMS IN A
SINGLE EQUIVALENCE DECLARATION (THE PARENTHETICAL EXPRESSION
IN THE EQUIVALENCE STATEMENT). A SYMBOL APPEARS NO MORE THAN
ONCE IN EQUTBL.

A STACK IS FORMED AS ITEMS ARE PROCESSED FROM A DECLARATION LIST. THE STACK ENTRIES ARE ONE WORD PER SYMBOL. THE ADDRESS CONTAINS THE SYMBOL TABLE POINTER, AND THE DECREMENT CONTAINS THE SUBSCRIPT (ONE, IF NO SUBSCRIPT IS GIVEN). EACH STACK IS BEGUN AND ENDED WITH A CONTROL WORD CONTAINING A MINUS SIGN.

THESE WORDS ARE USED TO DELIMIT A STACK AND, WHEN NECESSARY, TO LINK THIS STACK WITH OTHERS TO FORM A CHAIN. THE LOCATION OF EACH NEWLY BUILT STACK (UNLESS IT IS LINKED TO A FORMER STACK) IS RECORDED IN THE ADDRESS OF A DIRECTOR WORD, WHICH IS STORED IN THE DECREMENT OF 'INITFG' POINTS TO THE FIRST OF THESE POOL. THE DECREMENT OF 'INITFG' POINTS TO THE FIRST OF THESE DIRECTOR WORDS, AND THE DECREMENT OF EACH DIRECTOR POINTS TO THE NEXT ONE. THE LAST ONE HAS A ZERO DECREMENT.

IF A SYMBOL ENCOUNTERED IN A LIST IS ALREADY IN AN EQUTBL CHAIN (NOT THE CURRENT CHAIN), THE DIFFERENCE IN OLD AND CURRENT SUBSCRIPTS (DEL) IS COMPUTED. IF DEL IS NEGATIVE, THE MAGNITUDE OF DEL IS ADDED TO EACH MEMBER OF THE OLD CHAIN, AND DEL IS SET TO ZERO. IF DEL IS POSITIVE, DEL IS ADDED TO EACH MEMBER (IF ANY) OF THE CURRENT CHAIN, AND DEL IS SET EQUAL TO DEL. DEL IS AN AMOUNT TO BE ADDED AUTOMATICALLY TO THE SUBSCRIPTS OF ALL FUTURE ITEMS IN THE CURRENT EQUIVALENCE LIST. THE OLD CHAIN IS THEN LINKED TO THE CURRENT STACK (IF ANY), FORMING ONE LONGER CHAIN. IF, IN THE CURRENT LIST, A SECOND LINKAGE TO THE PRESENT CHAIN (CONSISTING OF THE OLD CHAIN(S) AND THE CURRENT STACK LINKED TOGETHER) IS FOUND, THE OLD SUBSCRIPT IS COMPARED WITH THE EFFECTIVE CURRENT SUBSCRIPT (SUBSCRIPT PLUS DEL). IF THESE NUMBERS ARE NOT EQUAL, AN INCONSISTENCY ERROR IS RECORDED.

THE LINKING PROCESS MUST CONNECT STACKS IN THE SAME ORDER AS THEY WERE FORMED. OTHERWISE, THE CORRECT ORDER OF VARIABLES IN COMMON MIGHT BE ALTERED.
THE DECREMENT FIELD OF THE TOP CONTROL WORD OF A STACK IS USED TO LINK THIS STACK WITH A STACK BUILT EARLIER IN EQUTBL. SIMILARLY, THE ADDRESS FIELD OF THE BOTTOM CONTROL WORD OF A STACK IS USED TO LINK TO A LATER STACK.

THE LOCATION OF A SYMBOL'S ENTRY IN EQUTBL IS RECORDED IN THE ADDRESS OF THE EQUIVALENT WORD, UNLESS THE SYMBOL HAS ALREADY APPEARED IN A COMMON, DIMENSION, ASSIGN, OR ASSIGNED GO TO STATEMENT. IN THE LATTER CASES, THE EQUTBL POINTER IS PLACED IN THE ADDRESS OF THE FIRST WORD IN THE COMMON/DIMENSION/ASSIGN ENTRY. In either event, the flag BEQV is turned ON in the EQUIVALENT TABLE ENTRY.

A SUBPROGRAM PARAMETER MAY NOT APPEAR IN AN EQUIVALENCE DECLARATION.

NOTE... DPOINT AND EPOINT ARE SYNONYMOUS WITH NXLLOC, AND
* DIMTB1 AND EQU1BL ARE SYNONYMOUS WITH POOL.
* ROUTINE FOR SCANNING EQUIVALENCE STATEMENTS.
* 
* IEUV1X AXC 11,2 RESET POINTER
* SKIP LPCHAR CHECK FOR LEFT PAREN
* ERROR 9,SKEND NO PAREN
* REM
* IEUV1X STZ PRIOR RESET FLAG
* STZ LINKER
* REM
* IEUV1X TSX SCNBOD,4 GET NAME
* TMI ERROR8 NOT A VARIABLE
* SLW *P SAVE BCD NAME
* REM
* IEUV1X LDQ =1817 SUBSCRIPT IS ONE
* STQ *P+1
* SKIP LPCHAR CHECK FOR SUBSCRIPT
* TRA STOEQU NO PAREN, ERGO NO SUBSCRIPT GIVEN
* REM
* IEUV1X TSX SCNBOD,4 GET SUBSCRIPT
* TPL ERROR17 MUST BE NUMERICAL
* TSX BCDFIX,4 CONVERT TO FIXED POINT NUMBER
* SLW *P+1 SAVE SUBSCRIPT
* REM
* IEUV1X SKIP RPCHAR CHECK FOR RIGHT PAREN
* ERROR 9,SKEND NO PAREN FOR SUBSCRIPT
* REM
* IEUV1X TRA STOEQU PROCESS EQUIVALENCE OF SYMBOL
* REM
* IEUV1X SKIP RPCHAR CHECK FOR RIGHT PAREN
* ERROR 9,SKEND NO ! EITHER. ERROR.
* REM
* IEUV1X ZET LINKER HAS A LINKAGE BEEN FOUND
* TRA IEU62 YES. NO DIRECTOR WORD NEEDED
* REM
* IEUV1X LXA EPOIN1,T,4 NO. GET NEXT POOL LOCATION FOR DIRECTOR
* CAL TOP GET TOP OF STACK JUST FINISHED
* STA EQU1BL,6 SET STACK LOCATION IN DIRECTOR WORD
* LXA INITFG,1 GET LAST DIRECTOR LOCATION
* SXA INITFG,4 SET NEW DIRECTOR LOCATION
* PXD ,4
* STD EQU1BL,1 SET LINKER IN PREVIOUS DIRECTOR WORD
* TX1 **1,4+1 BUMP EQU1BL POINTER
* SXA EPOIN1,T,4
* REM
* IEUV1X SKIP 63 MAY BE END OF STATEMENT
* TRA SKEND YES
* REM
* CAL COLUMN,2 NO. CHECK FOR ,1 CHARACTERS
* ERA =H,1
* ARS 24 BAD FORMAT
* INZ ERROR7
* REM
* TX1 IEUV3,X,-2 RETURN FOR ANOTHER SET OF EQUIVALENCES
* ROUTINE TO HANDLE EQUIVALENCE TABLE ENTRIES.

* STOEQU
  LDQ  *P   GET BCD SYMBOL
  TSX  LOCATE,4  PUT IN SYMTAB
  SLW  *P+2  SAVE SYMTAB TABLE POINTER
  TPL  IEOQ20  OLD SYMBOL. MODE ALREADY SET.
  REM

* NEW SYMBOL. SET MODE BITS AND BEQV FLAG.
  CAL  *P  NEW SYMBOL. FIND MODE.
  LAS  =HI00000
  TRA  ++1
  LAS  =HNI1111
  SIL  MREAL+BEQV FLOATING POINT
  TRA  ++2
  SIL  MINIG+BEQV FIXED POINT
  TRA  IEOQ26
  REM
  REM
  REM
  REM
  IEOQ20
  LFT  BARGT+BEXTF+BINTF
  TRA  IEOQ40  ILLEGAL FLAG
  REM
  IEOQ47
  LFT  BEQV  IS SYMBOL ALREADY IN EQUIVALENCE TABLE.
  TRA  IEOQ21  YES. GO TO SECTION 1.
  REM
  SIL  BEQV  NO. TURN ON BEQV FLAG
  REM

* NEW EQULBL ENTRY.

  IEOQ26
  LXA  EPOINT,4  SET EQULBL POINTER
  NZI  PRIOR  IS PRIOR FLAG ON
  TRA  IEOQ23  NO.
  REM
  TXI  ++1,4,-1  DROP POINTER
  CLA  *P+2  YES. GET SYMTAB POINTER
  ZET  LINKER  IS LINKER FLAG ON.
  ADD  DEL1  YES. INCREASE SUBSCRIPT
  REM
  IEOQ25
  ADD  *P+1  ADD IN SUBSCRIPT
  SLW  EQULBL,4  STORE EQULBL ENTRY
  REM
  LXA  *P+2,1  GET SYMTAB POINTER
  STI  EQUV,1  RESTORE EQUV WORD
  LFT  BCOMM+BARRY+BASSN  IS SYMBOL IN POOL ALREADY.
  TRA  IEOQ51  YES.
  REM
  PXA  ;4  NO.
  STA  EQUV,1  SET ADDRESS OF EQUV WORD TO EQULBL ENTRY
  TXI  IEOQ52,4,1
  REM

  IEOQ51
  PIA
  PAX  ;1  GET DIMITBL POINTER
  PXA  ;4
  STA  DIMITBL,1  SET EQULBL POINTER IN DIMITBL ENTRY
  TXI  ++1,4,1
  REM

  IEOQ52
  CLS  =0  SIGN BIT FOR BOTTOM LINK WORD
  STO  EQULBL,4  SET LINK WORD
  SXA  LO,4  SET LO = EPOINT
  TXI  ++1,4,1  BUMP POINTER
SXATERMFG,4 \( \) SET CURRENT TERMINATION POINT OF EQUATION
SXAEPINT,4 \( \) RESTORE POINTER
TRAL1EUQV1+1 \( \) RETURN

REM

1EUQ4DPA \( \) \text{ILLEGAL FLAG. GET SYMTAB POINTER}
CALSYMINT,4 \( \) \text{GET BCD NAME OF VARIABLE IN ERROR}
SLWFORM25+4 \( \) \text{PUT NAME INTO ERROR MESSAGE}
SLWFORM69
LFTBARGT
ERROR25,**1 \( \) \text{CANNOT BE ARGUMENT}
LFT \text{BEXIT}+\text{BINTE}
ERROR69,**1 \( \) \text{CANNOT BE SUBPROGRAM NAME}
TRA1EUQV1+1
REM

REM

* NEW EQUATION ENTRY. PRIOR FLAG OFF.

1EUQ23STL \( \) PRIOR \( \) \text{TURN ON PRIOR FLAG}
LXATERMFG,1 \( \) GET CURRENT END OF EQUIVALENCE STACKS
SXD **+2,1 \( \) \text{SKIP IF THIS IS THE BEGINNING OF ENTRIES}
TNX **+3,1,1 \( \) \text{TEST EPOINT VERSUS TERMFG}
TXH **+2,4,** \( \) \text{EQUAL. REDUCE EPOINT POINTER}
TXI **+3,4,-1 \( \) \text{FLAG CELL}
REM
CLS =0 \( \) \text{FLAG CELL}
STOEQUTBL,4 \( \) \text{FLAG CELL FOR TOP OF THIS STACK}
SXATOP,4 \( \) \text{SET TOP OF THIS NEW STACK}
REM
NZILINKER \( \) \text{IS LINKER FLAG ON.}
TXI1EUQ25-3,4,1 \( \) \text{NO.}
REM

1EUQ24LXALO,1 \( \) \text{YES. GET LO POINTER}
PXDPX,1 \( \) \text{PUT IN DECREMENT}
STD \text{EQUABLE,4 \( \) \text{SET (TOP) DECREMENT = LO}
REM
CLA \( \) \text{TOP}
STA \text{EQUABLE,1 \( \) \text{SET (LO) ADDRESS = TOP}
REM
CLA \text{P+2 \( \) \text{GET EQUABLE POINTER}
ORA \text{DELL \( \) \text{INSERT SUBSCRIPT INCREMENT IN NEW EQUABLE WDFTOQ350}
TXI1EUQ25,4,1 \( \text{BUMP EPOINT AND TRANSFER}
REM
REM
* SECTION 1. SYMBOL ALREADY IN EQUATION.
REM

1EUQ21PAX \( \) \text{GET A POINTER FROM THE EQUABLE WORD}
LFT \text{BCOMN+BARRY+BASSN \( \) \text{IS SYMBOL IN POOL ALREADY.}
CALDIMTBL,4 \( \) \text{YES. GET WORD CONTAINING EQUABLE POINTER}
REM
PAX \( \) \text{EQUABLE POINTER}
NZILINKER \( \) \text{IS LINKER FLAG ON.}
TRA1EUQ29 \( \) \text{NO.}
REM
SXA \text{P+4,4 \( \) \text{YES. SAVE EQUABLE POINTER}
TXI **+1,4,-1 \( \text{SCAN UPWARD TO TOP OF STACK}
1EUQ53CLA \text{EQUABLE,4 \( \) \text{GET TABLE ENTRY}
TPL **-2 \( \text{NOT YET A LINK WORD}
REM
ANA \text{0777770000000 \( \text{SAVE DECREMENT ONLY}
TZE1EUQ54 \text{TEST FOR ZERO DECREMENT
PDX ,4               RESET POINTER FOR NEXT SEGMENT OF STACK     
TXI 1EQU53,4,1, -1  CONTINUE THE SCAN                      
REM FTO34580        
REM FTO34590        
REM FTO34600        
1EQU54 PXA ,4        GET TOP OF THIS STACK               
ERA HI              COMPARE TOP OF STACK WITH HI          
REM FTO34610        
REM FTO34620        
REM FTO34630        
REM FTO34640        
REM FTO34650        
REM FTO34660        
REM FTO34670        
REM FTO34680        
REM FTO34690        
REM FTO34700        
REM FTO34710        
REM FTO34720        
REM FTO34730        
REM FTO34740        
REM FTO34750        
REM FTO34760        
REM FTO34770        
REM FTO34780        
REM FTO34790        
REM FTO34800        
REM FTO34810        
REM FTO34820        
REM FTO34830        
REM FTO34840        
REM FTO34850        
REM FTO34860        
REM FTO34870        
REM FTO34880        
REM FTO34890        
TRA 1EQU38          
REM FTO34900        
REM FTO34910        
REM FTO34920        
REM FTO34930        
REM FTO34940        
REM FTO34950        
REM FTO34960        
REM FTO34970        
1EQU39 LXA T3,1      RESET POINTER                       
TXI **1,4,1        SCAN DOWN FROM T3                     
CLA EQUtbl,4       GET ENTRY                             
TMI 1EQU35          LINK WORD FOUND                       
REM FTO34980        
REM FTO34990        
REM FTO35000        
REM FTO35010        
REM FTO35020        
CLA DEL             GET DEL                              
TMI **3             SKIP IF MINUS                         
ADD EQUtbl,4       BUMP SUBSCRIPT                        
STO EQUtbl,4       LINK WORD                            
TXI 1EQU38+2,4,1   LINK WORD, TEST ADDRESS              
REM FTO35030        
REM FTO35040        
REM FTO35050        
REM FTO35060        
REM FTO35070        
REM FTO35080        
1EQU35 PXA ,1        LINK WORD, TEST ADDRESS              
PXA ,1              SAVE ADDRESS ONLY                      
XCA                 SAVE LINK WORD                         
TXH 1EQU36,1,0     IS LINK ADDRESS ZERO.                
REM FTO35090        
REM FTO35100        
REM FTO35110        
REM FTO35120        
REM FTO35130        
REM FTO35140        
TRA 1EQU32          YES. SET 'LU' AND RETURN             
REM FTO35150        
REM FTO35160        
* THIS ENTRY LINKS TWO DIFFERENT OLD STACKS IN EQUtbl.
* TI RECORDS THE NEXT ENTRY INTO THE 'OTHER' STACK.
* WHEN ONE OF THE CHAINS HAS BEEN COMPLETELY PROCESSED, TI WILL BE SET TO ZERO.
* T3 IS USED TO TEMPORARILY HOLD A RESTARTING POINT IN ONE OF THE STACKS.
* LXA ,P4,1          RESTORE EQUtbl POINTER
CLA EQUtbl,1       GET TABLE ENTRY
ANA =077770000000   KILL ADDRESS FIELD
SUA ,P1            LESS NEW SUBSCRIPT
SUA DEL1           LESS DEL1
STO DEL            NEW DEL
REM FTO346660       
REM FTO34670        
REM FTO34680        
REM FTO34690        
REM FTO34700        
REM FTO34710        
REM FTO34720        
* LXA ,P4,1          RESTORE EQUtbl POINTER
CLA EQUtbl,1       GET TABLE ENTRY
ANA =077770000000   KILL ADDRESS FIELD
SUA ,P1            LESS NEW SUBSCRIPT
SUA DEL1           LESS DEL1
STO DEL            NEW DEL
REM FTO346660       
REM FTO34670        
REM FTO34680        
REM FTO34690        
REM FTO34700        
REM FTO34710        
REM FTO34720        
* LXA ,P4,1          RESTORE EQUtbl POINTER
CLA EQUtbl,1       GET TABLE ENTRY
ANA =077770000000   KILL ADDRESS FIELD
SUA ,P1            LESS NEW SUBSCRIPT
SUA DEL1           LESS DEL1
STO DEL            NEW DEL
REM FTO346660       
REM FTO34670        
REM FTO34680        
REM FTO34690        
REM FTO34700        
REM FTO34710        
REM FTO34720
LEQU37
CLA T1
CLA T1
STA EQU$BL,4
LXI T1+1
PXD
STD EQU$BL,1
SXA T3-1
STQ T1
REM CLS DEL
DEL = - DEL
REM
REM
LEQU36
NZI T1
HAS ONE STACK BEEN COMPLETED YET.
TRA 1EQU39
YES.
TRA 1EQU39
REM
REM
PXZ ;1
GET NON-ZERO LINK ADDRESS
LAS T1
COMPARE WITH T1
TRA 1EQU37
PZE UNREACHABLE POINT
REM
REM
LEQU39
PAX ;1
RESET POINTER TO NEXT STACK
TXI 1EQU38+2,4,1
REM
REM
REM
* FIRST LINKAGE WITH EARLIER STACKS.*
REM
REM
LEQU23
STL LINKER
TURN ON LINKER FLAG
CAL EQU$BL,4
GET OLD ENTRY
ANA =077777000000
SAVE DECREMENT
SUB .P+1
SUBTRACT NEW SUBSCRIPT
CHS
STO DEL
SET NEW DEL
SLW DEL1
DEL1 = POSITIVE DEL
TMI **2
SKIP IF DEL IS MINUS
STZ DEL1
SET DEL1 = ZERO
SXA .P+3,4
SAVE ENTRY POINTER
REM
REM
LEQU41
CLA EQU$BL,4
SCAN UPWARD TO TOP OF THIS STACK
IMI 1EQU42
IS IT LINK WORD.
REM
REM
CLA DEL
NO.
TMI **3
NO ACTION IF DEL IS MINUS
ADD EQU$BL,4
MODIFY SUBSCRIPT
STO EQU$BL,4
TXI 1EQU41,4,-1
REM
REM
LEQU42
ANA =077777000000
LINK WORD FOUND. IS IT TOP OF STACK
TZE **3
CONTINUE THE SCAN. TOP OF STACK NOT YET.
TXI 1EQU41,4,-1
REM
REM
SXA HI,4
HI = TOP OF STACK
LXA .P+3,4
SCAN DOWN FROM ENTRY. RELOAD POINTER
TXI **1,4,1
REM
REM
LEQU43
CLA EQU$BL,4
LINK WORD FOUND.
REM
REM
CLA DEL
**IMI**

**ADD** EQU8BL, 4  
**STO** EQU8BL, 4  
**TXI** 1EQU43, 4, 1

**REM**

**LEQU44 ANA** =077777 LINK WORD FOUND. SAVE ADDRESS FIELD  
**TZE** ++3 END OF STACK FOUND.  
**PAX** 4 CONTINUE THE SCAN DOWNWARD  
**TXI** 1EQU43, 4, 1  
**REM**

**ZET** PRIOR END OF STACK FOUND. IS PRIOR FLAG ON.  
**TRA** 1EQU45 YES.  
**REM**

**LEQU32 SXA** LO, 4 NO. LO = BOTTOM OF STACK  
**TRA** 1EQUV1+1 RETURN  
**REM**

**LEQU45 CLA** TOP EQU8BL, 4 PRIOR ON. (BOT OF STACK) ADDRESS = TOP  
**STA** EQU8BL, 4  
**PXD** 4 (TOP) DECREMENT = BOTTOM OF STACK  
**LXA** TOP, 4  
**STD** EQU8BL, 4  
**REM**

**CLA** DEL RETURN IF NO BUMPING OF NEW STACK IS NEEDED  
**LEQU46 TPL** 1EQUV1+1  
**REM**

**LEQU49 TXI** ++1, 4, 1 STEP DOWN 'PRIOR' STACK  
**IMI** 1EQUV1+1 BOTTOM OF STACK. RETURN  
**ADD** DELI ADD DELI TO SUBSCRIPT  
**STO** EQU8BL, 4  
**TXI** 1EQU48, 4, 1  
**REM**

*SECOND LINK WITH THIS STACK. CHECK FOR CONSISTENCY.*

**LEQU31 LXA** .P++4, 1 RESTORE EQU8BL POINTER  
**CLA** DELI  
**ADD** .P+1 ADD SUBSCRIPT  
**ERA** EQU8BL, 1  
**ANA** =077777000000 KILL THE ADDRESS FIELD  
**TZE** 1EQUV1+1 RETURN. FAIR WEATHER.  
**REM**

**CAL** .P GET BCD NAME OF GUILTY SYMBOL  
**SLW** FORM26+10 PUT NAME INTO THE ERROR MESSAGE  
**ERROR** 26, 1EQUV1+1 PUT OUT MESSAGE AND RETURN  
**REM**

**FT035770**

**FT035780**

**FT035790**

**FT035800**

**FT035810**

**FT035820**

**FT035830**

**FT035840**

**FT035850**

**FT035860**

**FT035870**

**FT035880**

**FT035890**

**FT035900**

**FT035910**

**FT035920**

**FT035930**

**FT035940**

**FT035950**

**FT035960**

**FT035970**

**FT035980**

**FT035990**

**FT036000**

**FT036010**

**FT036020**

**FT036030**

**FT036040**

**FT036050**

**FT036060**

**FT036070**

**FT036080**

**FT036090**

**FT036100**

**FT036110**

**FT036120**

**FT036130**

**FT036140**

**FT036150**

**FT036160**

**FT036170**

**FT036180**

**FT036190**

**FT036200**
<table>
<thead>
<tr>
<th>TTL</th>
<th>PASS 1 PROCESSOR - FREQUENCY</th>
</tr>
</thead>
<tbody>
<tr>
<td>FREQUENCY:</td>
<td>FT036210</td>
</tr>
<tr>
<td>REM</td>
<td>FT036220</td>
</tr>
<tr>
<td>ENTRY...</td>
<td>IFREQX</td>
</tr>
<tr>
<td>THIS STATEMENT IS IGNORED.</td>
<td>FT036240</td>
</tr>
<tr>
<td>SPACE 3</td>
<td>FT036250</td>
</tr>
<tr>
<td>IFREQX</td>
<td>SKEND</td>
</tr>
<tr>
<td>THIS STATEMENT IS JUST A NO-OP.</td>
<td>FT036260</td>
</tr>
<tr>
<td></td>
<td>FT036270</td>
</tr>
<tr>
<td>Instruction</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>TTL CONTINUE</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
</tr>
<tr>
<td>ENTRY... ICNTX</td>
<td></td>
</tr>
<tr>
<td>THIS STATEMENT IS TREATED AS A NO-OP.</td>
<td></td>
</tr>
<tr>
<td>SPACE 3</td>
<td></td>
</tr>
<tr>
<td>ICNTX AXC 8,2</td>
<td>RESET POINTER</td>
</tr>
<tr>
<td>SKIP 63</td>
<td>END OF STATEMENT EXPECTED</td>
</tr>
<tr>
<td>ERROR 3, SKEND</td>
<td></td>
</tr>
<tr>
<td>IRA SKEND</td>
<td></td>
</tr>
</tbody>
</table>
**TTL** PASS 1 PROCESSOR - GO TO

* GO TO N
  
  **ENTRY** 1GOTOX
  
  * THE LABEL N IS PLACED IN THE SYMBOL TABLE AFTER IT IS CHECKED
  AGAINST THE CURRENT CARD LABEL. THE LABEL N IS NOT
  
  * PERMITTED TO EQUAL THE CURRENT CARD LABEL. THE PATH-OF-FLOW
  * FLAG 'BPATH' IS TURNED ON FOR THE LABEL N. THE GENERATION
  * OF THE CODE 'TRA N' IS DELAYED UNTIL THE LABEL ON THE
  * NEXT EXECUTABLE STATEMENT IS FOUND. IF THIS LABEL IS
  * THE SAME AS N, NO CODE IS GENERATED.
  
  **SPACE** 3

1GOTOX

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AXC</td>
<td>4,2</td>
<td>RESET POINTER</td>
</tr>
<tr>
<td>AXI</td>
<td>1TRANS,4</td>
<td>SET ENTRY FOR LABEL ROUTINE</td>
</tr>
<tr>
<td>SXA</td>
<td>TRAFLG,4</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TSX</td>
<td>SCNBCD,4</td>
<td>GET LABEL N</td>
</tr>
<tr>
<td>IPL</td>
<td>ERROR1</td>
<td>NOT A LABEL</td>
</tr>
<tr>
<td>SLW</td>
<td>*P</td>
<td>SAVE BCD LABEL</td>
</tr>
<tr>
<td>SKIP</td>
<td>63</td>
<td>END OF STATEMENT EXPECTED</td>
</tr>
<tr>
<td>ERROR</td>
<td>3,SKEND</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLS</td>
<td>*P</td>
<td>GET BCD LABEL. SET 'REACHABLE LABEL' FLAG</td>
</tr>
<tr>
<td>TSX</td>
<td>STOLBL,4</td>
<td>PUT N IS SYMTAB. MINUS SIGN</td>
</tr>
<tr>
<td>SLW</td>
<td>1TRAN2</td>
<td>SAVE POINTER TO N</td>
</tr>
<tr>
<td>LAS</td>
<td>CARD</td>
<td>COMPARE WITH CURRENT LABEL</td>
</tr>
<tr>
<td>TRA</td>
<td>**2</td>
<td>INVALID TRANSFER</td>
</tr>
<tr>
<td>ERROR</td>
<td>6,**1</td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>SKEND</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* ROUTINE TO TEST IF LAST TRANSFER INSTRUCTION IS NEEDED.
* POINTER HAS BEEN SAVED IN CELL 1TRANS.
* ENTRY TO THIS ROUTINE WAS SET INTO TRAFLG BY PREVIOUS
* TRANSFER STATEMENT. EXIT FROM THIS ROUTINE IS TO 1,4.

1TRANS

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ZET</td>
<td>NOCODE</td>
<td>IS CODE STILL NECESSARY.</td>
</tr>
<tr>
<td>TRA</td>
<td>1,4</td>
<td>NO. RETURN</td>
</tr>
<tr>
<td>CAL</td>
<td>1TRANS2</td>
<td>GET POINTER</td>
</tr>
<tr>
<td>LAS</td>
<td>CARD</td>
<td>TEST AGAINST CURRENT LABEL</td>
</tr>
<tr>
<td>TRA</td>
<td>**2</td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>1,4</td>
<td>NO TRANSFER INSTRUCTION NEEDED. RETURN.</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SXA</td>
<td>ITTRAN3,4</td>
<td>CODE IS... TRA N</td>
</tr>
<tr>
<td>CODEO</td>
<td>TRA,R</td>
<td></td>
</tr>
</tbody>
</table>

1ITRAN3

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AXI</td>
<td>**,4</td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>1,4</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
* COMPUTED GO TO (N1, N2, ... NM), I

* ENTRY...

* EACH LABEL N1, N2, ... NM IS PUT IN THE SYMBOL TABLE AND THE
  * PATH-OF-FLOW FLAG BPATH IS TURNED ON. NONE OF THE LABELS MAY
  * BE THE CURRENT CARD LABEL. THE VARIABLE I MUST BE
  * FIXED POINT. IF I IS NOT A PARAMETER, THE CODE 'LDC',#4'
  * IS GENERATED. IF I IS A SUBPROGRAM PARAMETER (I.E. BARGT
  * IS ON), THE PROCESSOR TRANSFERS TO THE PROLOG ROUTINE.
  * ON RETURN, THE CODE 'LDC',#4' IS GENERATED.
  * NEXT AN INSTRUCTION 'TRA',#4' IS PUT OUT. THIS IS FOLLOWED
  * BY A LIST OF TRANSFERS 'TRA',#I' IN THE SAME ORDER AS THEY
  * APPEAR IN THE STATEMENT.

SPACE 3

ICOMGO
AXC 9,2
RESET POINTER

XLA 1,TRAN,4
SET EXIT FOR SUBSEQUENT LABEL ROUTINE

REM

AXC 1,4
SET 'GO TO' COUNTER

ICOMGI
XSA *P,4
SAVE COUNTER

TSX SCNB,4
GET N

TMI **2
IS IT A LABEL?

ERROR 1,ICOMG6
NO. ERROR

REM

TSX STOVL,4
PUT N(I) IN SYMTAB. MINUS SIGN.

LAS CARD
COMPARE WITH CURRENT LABEL

TRA **2
IMPROPER TRANSFER

ERROR 6,**1

REM

LXA *P,4
RESTORE COUNTER

SLW COLUMN,4
SAVE N POINTER

REM

ICOMG6
CAL COLUMN,2
TEST FOR COMMA

ERA =H,

ARS 30

TNZ **3

TXI **1,2,-1

TXI ICOMGI,4,-1
COMMA FOUND

REM

SXD ICOMG2,4
SAVE NUMBER OF LABELS

CAL COLUMN,2
TEST FOR },, CHARACTERS

ERA =H1,

ARS 24

TNZ ERROR11
IMPROPER PUNCTUATION

TXI **1,2,-2

REM

TSX SCNB,4
GET I

TMI ERROR8
NOT A VARIABLE

SLW FORM,69
SAVE BCD NAME

REM

LAS =H100000
TEST FOR FIXED POINT

TRA **1

LAS =H000000

TRA **1

ERROR 12,SKEND

REM

TSX SYMSTO,4
PUT I IN SYMTAB.

SLW *P,4
SAVE POINTER

REM
<table>
<thead>
<tr>
<th>Instruction</th>
<th>Meaning</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>LFT</td>
<td>BEXTF+8INTF</td>
<td>FT037470</td>
</tr>
<tr>
<td>ERROR</td>
<td>69,SKEND</td>
<td>FT037480</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FT037490</td>
</tr>
<tr>
<td>SKIP</td>
<td>63</td>
<td>FT037500</td>
</tr>
<tr>
<td>ERROR</td>
<td>3,SKEND</td>
<td>FT037510</td>
</tr>
<tr>
<td>NCODE</td>
<td>TEST CODE FLAG</td>
<td>FT037520</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FT037530</td>
</tr>
<tr>
<td>LNT</td>
<td>BARGI</td>
<td>FT037540</td>
</tr>
<tr>
<td>TRA</td>
<td>1COMG5</td>
<td>FT037550</td>
</tr>
<tr>
<td>CAL</td>
<td>.P+1</td>
<td>FT037560</td>
</tr>
<tr>
<td>TSX</td>
<td>PROLOG,4</td>
<td>FT037570</td>
</tr>
<tr>
<td>CODEC</td>
<td>LG4,N</td>
<td>FT037580</td>
</tr>
<tr>
<td>TRA</td>
<td>1COMG4</td>
<td>FT037590</td>
</tr>
<tr>
<td>TRA</td>
<td>1COMG5</td>
<td>FT037600</td>
</tr>
<tr>
<td>1COMG5</td>
<td>CODECO</td>
<td>FT037610</td>
</tr>
<tr>
<td>1COMG4</td>
<td>CODECO</td>
<td>FT037620</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FT037630</td>
</tr>
<tr>
<td>1COMG4</td>
<td>CODECO</td>
<td>FT037640</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FT037650</td>
</tr>
<tr>
<td>AXC</td>
<td>1,2</td>
<td>FT037660</td>
</tr>
<tr>
<td>1COMG3</td>
<td>CODECO</td>
<td>FT037670</td>
</tr>
<tr>
<td>1COMG2</td>
<td>TXL</td>
<td>FT037680</td>
</tr>
<tr>
<td>TXI</td>
<td>1COMG3,2,-1</td>
<td>FT037690</td>
</tr>
</tbody>
</table>
TIL PASS 1 PROCESSOR - ASSIGN

ASSIGN LABEL TO NAME

REM

ENTRY... IASNX

THE ASSIGN VARIABLE MUST BE FIXED POINT, AND MAY NOT APPEAR IN
A DIMENSION STATEMENT. IF THE SYMBOL 'NAME' HAS
NOT ALREADY APPEARED IN AN ASSIGN OR ASSIGNED GO TO STATEMENT
(i.e. BENNS AND BGO TO ARE BOTH OFF), A THREE WORD ENTRY IN
THE ASSIGN TABLE IS MADE. THESE WORDS SHARE STORAGE IN THE
'POOL' AREA WITH A POSSIBLE THREE-WORD COMMON TABLE ENTRY.

WORD ONE CONTAINS THE EQUV TABLE POINTER (OR THE EQUVBL POINTER IF BEQV IS ON). THE DECREMENT OF THIS WORD IS AVAILABLE TO HOLDFT037810

WORD TWO HAS THE POSSIBLE COMMON INFORMATION. WORD TWO HAS THE

VARIABLE'S EQUV TABLE POINTER IN THE ADDRESS FIELD.

WORD THREE HAS THE LABEL'S EQUV TABLE POINTER IN THE ADDRESS.

THE ADDRESS OF THE EQUV ENTRY FOR THE VARIABLE CONTAINS

THE ASSIGN TABLE ENTRY POINTER.

THE DECREMENT FIELD OF THE SECOND WORD OF THE PREVIOUS ASSIGN TABLE (IF ANY) IS SET TO THE POINTER TO THE SECOND WORD OF
THIS CURRENT ASSIGN TABLE. ALSO THE CELL ASNLCE, WHICH
HOLDS THE BEGINNING OF THE LAST-BUILT STACK, IS SET TO THIS
SAME VALUE.

REM

IF BASSN OR BGO TO IS ON, THE ASSIGN TABLE IS SEARCHED TO
SEE IF THE LABEL ALREADY APPEARS IN THIS STACK. IF IT DOES,
AND IF THE ENTRY HAS A MINUS SIGN, BPATH IS TURNED ON, SINCE
THE LABEL HAS APPEARED IN BOTH ASSIGN AND GO TO STATEMENTS.

IF THE LABEL DOES NOT APPEAR IN THE STACK, THEN THE LABEL
POINTER IS ADDED TO THE STACK WITH A PLUS SIGN, AND THE
LOCATION OF THE NEW ENTRY IS PLACED IN THE DECREMENT OF THE
PREVIOUS ENTRY.

IN ALL CASES, THE FLAG BASSN IS TURNED ON IN THE EQUV TABLE
ENTRY FOR THE VARIABLE 'NAME'.

REM

THE CODE 'AXT LABEL, 4' IS PUT OUT. IF NAME IS NOT A PARAMETER,
'SX NAME, 4' IS GENERATED. IF NAME IS A PARAMETER, THE
PROLOG ROUTINE IS ENTERED. ON RETURN, THE CODE 'SX A **, 4'
IS PUT OUT.

SPACE 3

1ASNX AXC 6,2

STZ .P=1

REM

TSX SCNBO, 4

REM

TLP ERROR1

REM

SPP DON'T TURN ON BPATH.

REM

TSX STOB, 4

REM

SLW .P

REM

CAL COLUMN, 2

REM

ERA =HTO

REM

ARS 24

REM

TNZ ERROR7

REM

TXI **1, 2, -2

REM

TSX SCNBO, 4

REM

TMI ERRORB

REM

LAS =HI00000

REM

TRA **1
LAS = HNIL

TRA *1

ERROR 12,SKEND NOT FIXED POINT

REM

TSX SYMSTG,4 PUT NAME IN SYMBOL TABLE

SLW P+2 SAVE EQUIV POINTER

CALOVF CAL EQUIV-OVERFL ADDRESS POINTS TO PSEUDO-EQUIV WORD

STZOVF STZ EQUIV-OVERFL

REM

REM PSEUDO-EQUIV WORD TO GET COMMON RELOCATION

OVERFL VFD 018/MREAL+BCOMN,3/015/00000

PAX ,4 GET EQUIV POINTER

REM

LFT BARRY+BEXTF+BINF

TRA 1ASS15 ILLEGAL FLAG

REM

REM

IASS13 SKIP 63 END OF STATEMENT EXPECTED

ERROR 3,SKEND

REM

LXA P2+4 RESTORE XR4

PIA

LFT BASSN

TRA 1ASSN1 NO.

REM

REM

SIL BASSN+4LHSX YES, TURN ON ASSIGN AND DEFINITION FLAGS

AXT 1ASS10+1,1 SET TRANSFER ADDRESS

SXA 1ASS10,1

REM

REM

* PRODUCE INITIAL ENTRIES FOR A NEW ASSIGN STACK

IASS11 STI EQUIV,4 RESTORE EQUIV WORD

LFT BCOMN IS SYMBOL IN COMMON.

TRA 1ASSN9 YES, TRANSFER

REM

LXA NXTLOC,1 GET NEXT AVAILABLE LOCATION IN POOL

PXA ,4 GET EQUIV TABLE POINTER

LFT BEQIV IS SYMBOL IN EQUIVALENCE STACK.

PIA YES, GET EQUIV POINTER

STA POOL,1 SET ASSIGN STACK ADDRESS

REM

REM

PX A ,1 GET POOL POINTER

STA EQUIV,4 SET EQUIV WORD ADDRESS

TXI **1,1,3 BUMP NXTLOC

SXA NXTLOC,1

REM

REM

IASSN9 ALS 18 MOVE ASSIGN TABLE POINTER TO DECREMENT

ADD =1817 BUMP

NZT ASNPFLG IS THIS THE FIRST ASSIGN ENTRY.

TRA 1ASS12 YES.

REM

LXD ASNLCF,1 GET POINTER TO BEGINNING OF PREVIOUS STACK

STD POOL,1 SET LINKER TO CURRENT STACK IN FORMER STACK

REM

IASS14 STD ASNLCF BEGINNING OF LAST-BUILT STACK

PDX ,1 GET START OF THIS STACK

PX A ,4 GET EQUIV TABLE POINTER

SLW POOL,1 SET IN ASSIGN TABLE

IASS10 TRA ** EXIT POINT FOR ASSIGNED GO TO ALSO

REM

REM

CAL TASSNX SET THE ASSIGN STATEMENT FLAG IN THE STACK

ORS POOL,1
<table>
<thead>
<tr>
<th>CAL</th>
<th>.P</th>
<th>GET LABEL POINTER</th>
<th>F038900</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLW</td>
<td>POOL-1,1</td>
<td>PUT IN NEXT WORD IN ASSIGN ENTRY</td>
<td>F038910</td>
</tr>
<tr>
<td>TRA</td>
<td>IASSN6</td>
<td></td>
<td>F038920</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>F038930</td>
</tr>
<tr>
<td>IASS12</td>
<td>STD</td>
<td>ASNFLG</td>
<td>F038940</td>
</tr>
<tr>
<td>TRA</td>
<td>IASS14</td>
<td>FIRST ASSIGN TABLE ENTRY, SET FLAG CELL</td>
<td>F038950</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>F038960</td>
</tr>
<tr>
<td>IASSN1</td>
<td>SIL</td>
<td>BASSN+BLHSX</td>
<td>F038970</td>
</tr>
<tr>
<td>TRA</td>
<td>IASS14</td>
<td>ASSIGN STACK ALREADY PRESENT, TURN ON FLAG</td>
<td>F038980</td>
</tr>
<tr>
<td>STI</td>
<td>EQUIV,4</td>
<td>RESTORE EQUIV WORD</td>
<td>F038990</td>
</tr>
<tr>
<td>PAX</td>
<td>.1</td>
<td></td>
<td>F039000</td>
</tr>
<tr>
<td>TXI</td>
<td>++1,1,2</td>
<td></td>
<td>F039010</td>
</tr>
<tr>
<td>CAL</td>
<td>TASSNX</td>
<td>PUT FLAG IN THE STACK</td>
<td>F039020</td>
</tr>
<tr>
<td>ORS</td>
<td>POOL+1,1</td>
<td></td>
<td>F039030</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>F039040</td>
</tr>
<tr>
<td>IASSN3</td>
<td>CAL</td>
<td>POOL,1</td>
<td>F039050</td>
</tr>
<tr>
<td>STD</td>
<td>.P+1</td>
<td>GET ELEMENT IN STACK</td>
<td>F039060</td>
</tr>
<tr>
<td>TRA</td>
<td></td>
<td>SAVE LINKER</td>
<td>F039070</td>
</tr>
<tr>
<td>ERA</td>
<td>.P</td>
<td>IS LABEL ALREADY IN STACK.</td>
<td>F039080</td>
</tr>
<tr>
<td>ANA</td>
<td>=07777</td>
<td>SAVE ADDRESS</td>
<td>F039090</td>
</tr>
<tr>
<td>TZE</td>
<td>IASSN4</td>
<td>YES, LEAVE LOOP</td>
<td>F039100</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>F039110</td>
</tr>
<tr>
<td>NZT</td>
<td>.P+1</td>
<td>NO, IS THIS END OF STACK.</td>
<td>F039120</td>
</tr>
<tr>
<td>TRA</td>
<td>++3</td>
<td>YES.</td>
<td>F039130</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>F039140</td>
</tr>
<tr>
<td>LXD</td>
<td>.P+1,1</td>
<td>NO.</td>
<td>F039150</td>
</tr>
<tr>
<td>TRA</td>
<td>IASSN3</td>
<td>GET NEXT LABEL IN STACK</td>
<td>F039160</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>F039170</td>
</tr>
<tr>
<td>CAL</td>
<td>NXTLOC</td>
<td>END OF STACK</td>
<td>F039180</td>
</tr>
<tr>
<td>ALS</td>
<td>18</td>
<td>SET LINKER TO THIS LABEL</td>
<td>F039190</td>
</tr>
<tr>
<td>STD</td>
<td>POOL,1</td>
<td></td>
<td>F039200</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>F039210</td>
</tr>
<tr>
<td>IASSN2</td>
<td>LXA</td>
<td>NXTLOC,1</td>
<td>F039220</td>
</tr>
<tr>
<td>CAL</td>
<td>.P</td>
<td>GET LABEL POINTER</td>
<td>F039230</td>
</tr>
<tr>
<td>SLW</td>
<td>POOL,1</td>
<td>PUT IN ASSIGN STACK</td>
<td>F039240</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>F039250</td>
</tr>
<tr>
<td>TXI</td>
<td>++1,1,1</td>
<td>BUMP NXTLOC</td>
<td>F039260</td>
</tr>
<tr>
<td>SXXA</td>
<td>NXTLOC,1</td>
<td></td>
<td>F039270</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>F039280</td>
</tr>
<tr>
<td>IASSN6</td>
<td>STI</td>
<td>.P+3</td>
<td>F039290</td>
</tr>
<tr>
<td>CODEC</td>
<td>AXTn,R,XRFLAG</td>
<td>CODE IS... AXT LABEL,4</td>
<td>F039300</td>
</tr>
<tr>
<td>CODEC</td>
<td>PXA4,N</td>
<td>CODE IS... PXA 0,4</td>
<td>F039310</td>
</tr>
<tr>
<td>CAL</td>
<td>.P+2</td>
<td>GET EQUIV POINTER</td>
<td>F039320</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>F039330</td>
</tr>
<tr>
<td>LDI</td>
<td>.P+3</td>
<td>RESTORE INDICATORS</td>
<td>F039340</td>
</tr>
<tr>
<td>LFT</td>
<td>BARGT</td>
<td>IS IT A PARAMETER</td>
<td>F039350</td>
</tr>
<tr>
<td>TRA</td>
<td>1ASSN5</td>
<td>YES.</td>
<td>F039360</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>F039370</td>
</tr>
<tr>
<td>CODEC</td>
<td>SLW,V</td>
<td>CODE IS... SLW NAME,4</td>
<td>F039380</td>
</tr>
<tr>
<td>TRA</td>
<td>SKEND</td>
<td></td>
<td>F039390</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>F039400</td>
</tr>
<tr>
<td>IASSN5</td>
<td>TSX</td>
<td>PROLOG,4</td>
<td>F039410</td>
</tr>
<tr>
<td>CODEC</td>
<td>SLW,N</td>
<td>CODE IS... SLW **,4</td>
<td>F039420</td>
</tr>
<tr>
<td>TRA</td>
<td>SKEND</td>
<td></td>
<td>F039430</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>F039440</td>
</tr>
<tr>
<td>IASSN4</td>
<td>CLA</td>
<td>POOL,1</td>
<td>F039450</td>
</tr>
<tr>
<td>TPL</td>
<td>1ASSN6</td>
<td>IS SIGN PLUS.</td>
<td>F039460</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>F039470</td>
</tr>
<tr>
<td>PAX</td>
<td>.4</td>
<td>GET EQUIV POINTER</td>
<td>F039480</td>
</tr>
<tr>
<td>CAL</td>
<td>BPATHW</td>
<td></td>
<td>F039490</td>
</tr>
<tr>
<td>ORS</td>
<td>EQUIV,4</td>
<td>TURN ON BPATH FLAG IN EQUIV WORD</td>
<td>F039500</td>
</tr>
<tr>
<td>REM</td>
<td>IASS15</td>
<td>CAL</td>
<td>SYMTAB,4</td>
</tr>
<tr>
<td>--------</td>
<td>---------</td>
<td>-------</td>
<td>----------</td>
</tr>
<tr>
<td>SLW</td>
<td>FORM28+2</td>
<td>PUT NAME INTO ERROR MESSAGE</td>
<td>FT039510</td>
</tr>
<tr>
<td>SLW</td>
<td>FORM69</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LFT</td>
<td>BARRY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ERROR</td>
<td>28,**1</td>
<td>MAY NOT BE ARGUMENT</td>
<td>FT039540</td>
</tr>
<tr>
<td>LFT</td>
<td>BEXTF+BINTF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ERROR</td>
<td>69,**1</td>
<td>MAY NOT BE SUBPROGRAM NAME</td>
<td>FT039560</td>
</tr>
<tr>
<td>TRA</td>
<td>IASS13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>BPATHW VFD</td>
<td>018/BPATH</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**TIL PASS 1 PROCESSOR - ASSIGNED GO TO**

* GO TO NAME, (N1, N2, ..., NM)

* REM

* ENTRY... IAGOTO

* THE ASSIGN STACK IS BUILT AS DESCRIBED IN THE ASSIGN STATEMENT.

* IF A LABEL IN THE 'GO TO' LIST IS ALREADY IN THE STACK WITH A

* PLUS SIGN, BPATH IS TURNED ON, SINCE IT HAS NOW APPEARED IN

* BOTH ASSIGN AND GO TO STATEMENTS. THE SIGN IS MADE MINUS.

* NEW LABELS IN THE STACK ARE GIVEN MINUS SIGNS.

* NONE OF THE LABELS IN THE GO TO LIST ARE ALLOWED TO EQUAL

* THE CURRENT CARD LABEL.

* REM

* THE FLAG BGOTO IS TURNED ON IN THE EQUIV TABLE ENTRY

* FOR THE VARIABLE.

**SPACE 3**

* IAGOTO

* AXC 4,2
    * RESET POINTER

* SIZ *P+1
    * INITIALIZE TEMPORARY CELL

* STZ *P+4
    * INITIALIZE ERROR CELL

* REM

* AXT 1TRANS+1,4
    * SET NO-OP EXIT FROM NEXT CARD PROCESSOR

* SXA TRAFLG+4
    * SET ERROR CELL ON.

* REM

* TSX SCNBDG,4
    * GET VARIABLE NAME

* THL ERROR8
    * NOT A VARIABLE

* REM

* LAS =H10000
    * CHECK FOR FIXED POINT

* TRA *P+1
    * NOT FIXED POINT. SET ERROR CELL ON.

* LAS =HNI1111

* STL *P+4

* ERROR 12, SKEND

* REM

* TSX SYMSTO,4
    * PUT NAME IN SYMBOL TABLE

* PAX 4
    * GET EQUIV TABLE POINTER

* SLW *P+2
    * SAVE EQUIV POINTER

* LFT BARRY+BEXTF+BINFT

* IRA 1AG014
    * ILLEGAL FLAG

* REM

**IAG011 PIA**

* LFT BASSN
    * IS THE ASSIGN FLAG OFF.

* TRA 1AG03
    * NO. ASSIGN STACK IS ALREADY PRESENT

* REM

* SIL BASSN
    * OFF. BUILD NEW ASSIGN STACK

* AXT *S+3,1
    * SET EXIT POINT FROM SECTION OF CODE

* SXA 1ASS10,1
    * IN ASSIGN STATEMENT PROCESSOR.

* TRA 1 ASS11
    * GO TO THIS SECTION OF CODE

* REM

* TXI *S+1,1
    * RETURN FROM OTHER CODE AT THIS POINT

* CAL =0777777
    * SET LABEL ENTRY TO WIERD END-OF-STACK

* SLW POOL,1
    * CONDITION

* SIZ *P+3
    * ZERO ONLY FOR FIRST LABEL IN A STACK.

* REM

**IAG012 CAL**

* TGOTOX
    * PUT FLAG INTO THE STACK

* ORS POOL+1,1

* REM

* CAL COLUMN,2
    * CHECK FOR ,

* ERA =H1

* ARS 24
    * INVALID PUNCTUATION

* TNZ ERROR7

* SXA 1AG015,1
    * SAVE START OF LABEL LIST

* TXI *S+1,2,-2
    * BUMP POINTER
REM 1AG0T4 TSX SCNBD4 GET LABEL
TIP ERROR1 NOT A LABEL
REM
SSP DON'T TURN ON BPATH IN STOLBL.
TSX STOLBL4 PUT IN SYMBOL TABLE
SLW 4 SAVE LABEL POINTER
LST IS IT SAME AS CURRENT CARD.
TRA ++2 NO.
ERROR 6,++1 YES, ERROR
REM
AXT ++1 RESET POINTER TO START OF LABEL LIST
LXA .P,4 GET EQUIV POINTER
ZET .P,4 TEST ERROR CONDITION
TRA 1AGOT6 ERROR FLAG ON. SKIP TABLE BUILDING
REM
CLA POOL,1 GET LABEL ENTRY IN STACK
STD .P,1 SAVE LINKER
ERA .P COMPARE WITH CURRENT LABEL POINTER
ANA =077777 SAVE ADDRESS POINTER
TZE 1AGOT7 SAME.
REM
NZT .P,1 NOT THE SAME. IS THIS END OF STACK.
TRA ++3 YES.
REM
LXD .P,1,1 NO. GET LINKER
TRA 1AGOT8
REM
CLS .P GET LABEL POINTER. MINUS SIGN
NZT .P,3 IS THIS THE FIRST LABEL ENTRY IN THE STACK.
TRA 1AGOT9 YES.
REM
LXA NXTLOC,4 GET POINTER TO NEXT OPEN POSITION IN STACK
STO POOL,4 SET LABEL ENTRY INTO STACK
REM
PXO .4 GET POINTER
STD POOL,1 SET LINKER INTO PRIOR LABEL ENTRY IN STACK
TXI ++1,4,1 BUMP NXTLOC
SXA NXTLOC,4 RESTORE NXTLOC
REM
1AGOT6 SKIP1 CMCHAR CHECK FOR COMMA
TRA 1AGOT4 COMMA FOUND. RETURN FOR ANOTHER LABEL
REM
CAL COLUMN,2 NO COMMA. CHECK FOR ) AND EOS
ARS 24
REM
ERA =03477 BAD PUNCTUATION
TNZ ERROR3
REM
ZET .P,4 TEST ERROR FLAG
TRA SKEND FLAG ON. QUIT.
REM
CAL .P,2 GET EQUIV POINTER
LST BARG IS VARIABLE A PARAMETER.
TRA 1AG010 YES.
REM
CODEO TRAIV CODE IS... TRA* NAME
TRA SKEND
REM
1AG010 TSX PROLOG,4 PREPARE PROLOG TABLE ENTRY
CODEC  TRA13N  CODE IS...  TRA* **  FT040810
TRA  SKEND  EXIT  FT040820

REM  FT040830

1AG07  CLS  POOL,1  LABEL FOUND IN STACK  FT040850
TPL  1AG076  SKIP IF LABEL HAD MINUS SIGN  FT040860
REM  FT040870
SIL  BPATH  TURN ON BPATH IN EQUIV WORD  FT040880
REM  FT040890
STL  EQUIV,4  RESTORE EQUIV WORD  FT040900
1AG079  POOL,1  SET SIGN OF LABEL ENTRY MINUS  FT040910
STL  ,P+3  THESE 2 INST. USED FOR 1ST LABEL IN STACK  FT040920
TRA  1AG076  FT040930
REM  FT040940
1AG03  STL  ,P+3  OLD ASSIGN STACK. TURN ON STACK FLAG  FT040950
PAX  ,1  PUT STACK POINTER IN INDEX  FT040960
TXI  1AG012,1,2  BUMP STACK POSITION TO FIRST LABEL ENTRY  FT040960
REM  FT040970
1AG014  CAL  SYMTAB,4  ILLEGAL FLAG. GET BCD NAME OF SYMBOL  FT040980
SLW  FORM28+2  PUT NAME INTO ERROR MESSAGE  FT040990
SLW  FORM69  FT041000
LFT  BARRY  FT041010
ERROR  28,**+1  MAY NOT BE ARRAY  FT041020
LFI  BEXTF+BINTF  FT041030
ERROR  69,**+1  MAY NOT BE SUBPROGRAM NAME  FT041040
TRA  1AG011  FT041050
TTL PROCESSOR - DO
DO STATEMENT PROCESSOR

ENTRY... PDO

PRODUCES DO CODING AS FOLLOWS...

FOR DO N I=M1,M2,M3 WE GET
1. LXD M3,4 V FLAG
2. SXX 2,4 S FLAG
3. LXD M2,4 V FLAG
4. SXX ***+4 P FLAG
5. STZ 1 V FLAG
6. LXD M1,4 V FLAG
7. TRAS ***+2 + FLAG
8. *1 TXH .3,4,** OR M2 S FLAG
9. SXX 1,4 V FLAG
...
10. LXD 1,4 V FLAG
11. *2 TXI .1,4,** OR M3 P FLAG
13. *3 BSS 0

IF M3=CONSTANT OR IS OMITTED, OMIT 1 AND 2 AND CHANGE 11 TO
TXI .1,4,M3
IF M2=CONSTANT OMIT 3 AND 4 AND COMPILE 8 AS TXH .3,4,M2
IF M1=CONSTANT CHANGE 6 TO AXT M1,4

FORMAT OF THE DOSTACK IS ...

WORD 3 POINTER TO TERMINAL LABEL
WORD 2 PZE POINTER FOR INDEX,T,PRIORITY FOR .2 PS. CELL OR CONSTANT
WORD 1 PZE PCOUNT FOR .1,,POINTER FOR .3 SP. CELL

WHERE T=0 DENOTES M3=CONSTANT
AND T=1 DENOTES M3 NOT = CONSTANT

PDO
STL DOFLAG FLAG AS A DO SO THAT ON NEXT STATEMENT
REM CHECK CAN BE MADE THAT A NON-EXECUTABLE
REM STATEMENT DOES NOT FOLLOW A DO.
LXA DOPNTR,1
TXI **+1,1,3
TXH PDO1,1,LDOSTK DO STACK OVERFLOW
SXA DOPNTR,1
AXC 2,2
TSX SCNBC0,4
TPL PDO3 ERROR SINCE LABEL NOT FOUND
STZ DOSTAK+1,1 CLEAR OUT 2ND WORD OF DO STACK ENTRY
XCL IS LABEL ALREADY IN SYMTAB AS DEFINED
TSX LOCATE,4
SLW DOSTAK,1 SAVE LABEL IN DO-STACK
TIM PDO4 NO
LFT BLH5X+MSTRG IT'S NOT A NEW LABEL IN FACT.
ERROR 73,PD02 IT IS ALREADY DEFINED. POP UP DO STACK
REM LABEL NOT ENCLOSED YET. DO NOT ENTER IT.
PDO4 TSX SCNBC0,4 GET INDEX OF LOOP
TSX SYMSTA,4 ENTER INDEX IN SYMTAB IF NECESSARY. IF
REM IT IS AN INDEX VARIABLE. FLAG AS DEFINED
LFT BARG IS INDEX AN ARGUMENT
ERROR 59,PD02 YES. NOT ALLOWED
DATA DOSTAK+1,1 SAVE POINTER FOR INDEX

* IS INDEX OF THIS DO USED FOR INDEX IN ANY DO WITHIN WHICH
* THIS DO IS NESTED. IF SO, NOT ALLOWED. EXIT TO P002
* AND IGNORE THIS DO.

SX A ++1,1
AX I ++1,4,-3
T XL ++5,4,0
L AS DOSTAK+1,1
T X I *-2,4,-3
E R R O R B6,P002
T X I *-4,4,-3

S K I P E Q C H A R
E R R O R 60,P002
T S X S C N B C D,4
T M I P D O 9
T S X S Y M S T B,4
R E M
R I R 700000
R I L 7
S T I M I V A R
S T A M I V A R
S T L M I F L A G
S K I P C M C H A R
E R R O R 2,P002
R E M
P D O 9 A T S X S C N B C D,4
T M I P D O 8
T S X S Y M S T B,4
R I R 700000
R I L 7
S T I M 2 V A R
S T A M 2 V A R
S T L M 2 F L A G
S K I P C M C H A R
T R A P D O 1 5
R E M
P D O 8 A T S X S C N B C D,4
T M I P D O 7
T S X S Y M S T B,4
R I R 700000
R I L 7
S T I M 3 V A R
S T A M 3 V A R
C O D E L X D 4,M 3 V A R
T S X N E X T I L P,4
A L S 1 8
S T D D O S T A K+1,1
A R S 1 8
O R A S X D 4
T S X C I T B L D,4,R
I R A P D O 6
R E M
P D O 7 T S X B C D F I X,4
P D O 7 A O R A =0100000
S T D D O S T A K+1,1
S T T D O S T A K+1,1
T R A P D O 6

REM
STZ M2FLAG
TSX BXDFIX,4
SLW M2VAR
TRA PD08A-2
REM
STZ M1FLAG
TSX BXDFIX,4
ARS 18
SLW M1VAR
TRA PD09A-2
REM
SKIP 63
MUST BE END OF STATEMENT
REM
NZI M2FLAG
IS M2 A VARIABLE.
REM
TRA PD010
CODE LXD4,M2VAR
YES. TURN OUT INSTRUCTIONS 3 AND 4
CAL SXD44
ACL PCOUNT
TSX CITBLD,4,P
REM
PDD010
CAL DOSTAK+1,1
ANA =077777
ORA STZ
TSX CITBLD,4,V
NZI M1FLAG
IS M1 A CONSTANT
TRA PD011A
YES. PUT OUT AXT INSTEAD
CODE LXD4,M1VAR
PDD011A
CAL TRA2
ACL PCOUNT
TSX CITBLD,4,P
TRA PD011B
PDD011B
CAL M1VAR
M1 A CONSTANT. TURN OUT THE AXT
ORA AXT4
TSX CITBLD,4,N
TRA PD011C
REM
PDD011C
CAL PCOUNT
STA DOSTAK+2,1
TSX NEXITP,4
ALS 18
STO DOSTAK+2,1
ARS 18
NZI M2FLAG
IS M2 A VARIABLE
ORA M2VAR
NO. OR IN THE CONSTANT DECREMENT
ORA TXH4
ORS XRFLAG
TSX CITBLD,4,R
CAL DOSTAK+1,1
ANA =077777
ORA SXD4
TSX CITBLD,4,V
TRA SKEND
*
<table>
<thead>
<tr>
<th>PDL2</th>
<th>REM</th>
<th>BUMP DO STACK BACK DOWN</th>
<th>FT042860</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LXA</td>
<td>DOPNTR, 4</td>
<td>FT042870</td>
</tr>
<tr>
<td></td>
<td>TXI</td>
<td>++1, 4, -3</td>
<td>FT042880</td>
</tr>
<tr>
<td></td>
<td>SXA</td>
<td>DOPNTR, 4</td>
<td>FT042890</td>
</tr>
<tr>
<td></td>
<td>TRA</td>
<td>SKEND</td>
<td>FT042900</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IGNORE THIS DO</td>
<td></td>
</tr>
</tbody>
</table>
* EJECT
* DO TERMINATION ROUTINE
* REM
* DO TERMINATION ROUTINE. TURNS OUT THE LXD 2,4 AND TXI .1,4,** OR DT M2
* M3 FOR DO LOOP, THEN LOOKS IN DOSTACK TO SEE IF ANY OTHER DO'S
* TERMINATE ON THIS SAME LABEL STATEMENT. IF SO THESE DO LOOPS
* ARE ALSO HANDLED IN THE SAME WAY.
* REM
* SPECIAL CONSTANT CELLS .2 AND .3 ARE DEFINED HERE.
* SPACE 3
* DOTERM SXA DOTRMX,4
* SXA DOTRMX+1,1
* LXA DOPNTR,1
* DOTRM CAL DOSTAK+1,1
* PAX ,4 GET THE EQUIV POINTER
* LDI EQUIV,4 AND GET THE EQUIV WORD
* RIL BDUX TURN OFF THE DO INDEX FLAG BIT
* STI EQUIV,4 AND RESET THE EQUIV WORD
* REM
* ANA =077777
* ORA LX04 TURN OUT THE LXD 1,4
* TSX CITBLC,4,V
* LDI DOSTAK+1,1
* RFT 100000
* TRA DOTRM2 INCREMENT IS A CONSTANT
* REM
* REM NOW DEFINE SPECIAL CONSTANT .2
* REM
* CAL DOSTAK+1,1
* PDX 4
* CAL PCOUNT
* SLW EQUIV,4
* REM
* CAL DOSTAK+2,1
* ANA =077777
* TRM4 RXA TX14 TURN OUT THE TXI .1,4,**
* TSX CITBLC,4,P
* CAL DOSTAK+2,1
* PDX 4
* CAL PCOUNT
* SLW EQUIV,4
* TXI **+1,**-3
* SXA DOPNTR,1
* TXL DOTRMX,1,0 IS DO STACK EMPTY
* CAL DOSTAK-3,1 NO
* ERA DOSTAK,1
* TZE DOTRM1 NEXT DO ALSO TERMINATES HERE
* DOTRM AXT **,4
* AXT **,1
* IRA 1,4
* REM
* REM CODING TO TURN OUT TXI .1,4,*M2
* REM
* DOTRM2 CAL DOSTAK+1,1
* ANA =077777000000
* SLW DOTRM3
* CAL DOSTAK+2,1
* ANA =077777
* ORA DOTRM3
* IRA DOTRM4
*
* TTL PASS 1 PROCESSOR - IF
  IF (X) N1,N2,N3
  * ENTRY... 1IFXXX
  * THE 'IF' STATEMENT PROCESSOR TRANSFERS INITIALLY TO THE
  * COMPIL ROUTINE TO GENERATE THE CODE FOR EVALUATING THE
  * PARENTHETICAL ALGEBRAIC OR LOGICAL EXPRESSION. ON RETURN
  * FROM COMPIL, EACH LABEL IS PLACED IN THE SYMBOL TABLE AFTER
  * IT IS TESTED AGAINST THE CURRENT CARD LABEL. NONE OF THE
  * LABELS N1, N2, N3 MAY EQUAL THE CURRENT CARD LABEL.
  * THE PATH-OF-FLOW FLAG 'PATH' IS TURNED ON FOR EACH LABEL.
  * THE GENERATION OF THE CODE IS DELAYED UNTIL THE LABEL ON THE
  * NEXT EXECUTABLE STATEMENT IS OBTAINED. THEN THE ROUTINE
  * ITRIFF IS ENTERED, AND THE NECESSARY CODE PUT OUT.
  * SPACE 3
  
  1IFXXX
  * AXC 2,2  RESET POINTER TO PAREN
  TSX CMPLIF,4  PROCESS ARITHMETIC STATEMENT. RETURN
  REM WITH POINTER AT N1.
  AXT ITRIFF,4  SET ENTRY FOR LABEL ROUTINE
  SXA TRAFLG,4  SET LOOING INDEX
  AXT 3,1  SET LOOING INDEX
  REM
  
  1IFXX2
  * TSX SCNCHD,4  GET N(I)
  TPL ERROR1 NOT A LABEL
  REM
  TSX STOLBL,4  PUT N(I) IN SYMTAB. MINUS SIGN.
  SLW .STT+4,1  SAVE POINTER IN SOME AVAILABLE CELLS
  LAS CARD  COMPARE WITH CURRENT LABEL
  TRA +2  IMPROPER TRANSFER
  ERROR 6, +1  IMPROPER TRANSFER
  REM
  INX 1IFXX1,1,1  ALL DONE
  REM
  SKIP CMCHAR CHECK FOR COMMA
  ERROR 2,SKEND NO COMMA
  TRA 1IFXX2
  REM
  
  1IFXX1
  * SKIP 63  END OF STATEMENT EXPECTED
  ERROR 3,SKEND QUIT FOR NOW
  TRA SKEND QUIT FOR NOW
  SPACE 8  
  * ROUTINE TO GENERATE CODE FOR THE IF STATEMENT. THIS SECTION.
  * IS ENTERED FROM THE PASS 1 DRIVER AFTER THE LABEL ON THE NEXT
  * EXECUTABLE STATEMENT HAS BEEN OBTAINED.
  * ONLY NECESSARY CODE IS PRODUCED.
  * SPACE 3
  
  ITRIFF
  * SXA ITRIFX,4  SAVE XR4
  CAL .STT+1  GET MINUS LABEL
  ERA .STT+3  COMPARE WITH PLUS LABEL
  TNZ ITRIF1  NOT EQUAL
  CAL .STT+2  SAME, GET ZERO LABEL
  ERA CARD  IS IT SAME AS CURRENT STATEMENT LABEL.
  TNZ ITRIF2  NO
  CAL TNZ  YES
  ERA  ITRIF5+1
  REM
  
  ITRIF2  CODECO IZER,STT+2,R  CODE IS... IZER N2
  CAL .STT+1  GET MINUS LABEL
  ERA  IS IT SAME AS CURRENT CARD LABEL.
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TZE</td>
<td>ITRIFX</td>
<td>YES. EXIT</td>
<td>FT044110</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FT044120</td>
</tr>
<tr>
<td>ITRIF5</td>
<td>CODECO</td>
<td>TRA,..STT+1,R</td>
<td>CODE IS... TRA N1</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FT044140</td>
</tr>
<tr>
<td>ITRIFX</td>
<td>AXT</td>
<td>**,4</td>
<td>FT044150</td>
</tr>
<tr>
<td>TRA</td>
<td>1,4</td>
<td>FT044160</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FT044170</td>
</tr>
<tr>
<td>ITRIF1</td>
<td>CODECO</td>
<td>TZE,..STT+2,R</td>
<td>CODE IS... TZE N2</td>
</tr>
<tr>
<td>CAL</td>
<td>..STT+3</td>
<td>GET PLUS LABEL</td>
<td>FT044190</td>
</tr>
<tr>
<td>ERA</td>
<td></td>
<td>CARD</td>
<td>FT044200</td>
</tr>
<tr>
<td>TNZ</td>
<td>ITRIF3</td>
<td>NOT SAME AS CURRENT LABEL</td>
<td>FT044210</td>
</tr>
<tr>
<td>CAL</td>
<td>TMI</td>
<td>SAME. CODE IS... TMI N1</td>
<td>FT044220</td>
</tr>
<tr>
<td>TRA</td>
<td>ITRIF5+1</td>
<td></td>
<td>FT044230</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FT044240</td>
</tr>
<tr>
<td>ITRIF3</td>
<td>CAL</td>
<td>TPL</td>
<td>CODE IS... TPL N3</td>
</tr>
<tr>
<td>ORA</td>
<td>..STT+3</td>
<td></td>
<td>FT044260</td>
</tr>
<tr>
<td>TRA</td>
<td>ITRIF2+2</td>
<td></td>
<td>FT044270</td>
</tr>
<tr>
<td>Instruction</td>
<td>Location</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>----------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ILPAS1 PROCP</td>
<td>KFT044280</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IF DIVIDE CHECK N1, N2</td>
<td>FT044290</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>FT044300</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENTRY... 11FDIV</td>
<td>FT044310</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEITHER N1 NOR N2 MAY EQUAL THE CURRENT CARD LABEL.</td>
<td>FT044320</td>
<td></td>
<td></td>
</tr>
<tr>
<td>THE INITIAL CODE GENERATED IS 'DCT' AND 'TRA N1'. THE CODE</td>
<td>FT044330</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'TRA N2' IS DELAYED AND IS GENERATED ONLY IF THE NEXT EXECUTABLE</td>
<td>FT044340</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STATEMENT IS NOT LABELED N2.</td>
<td>FT044350</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPACE 3</td>
<td>FT044360</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11FDIV AXG 13Z</td>
<td>FT044370</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STZ ,P ZERO FLAG</td>
<td>FT044380</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRA 11FAC1</td>
<td>FT044390</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
* TTL PASS 1 PROCESSOR - IF QUOT. OVERFLOW N1, N2 FT044400
  IF QUOTIENT OVERFLOW N1, N2 FT044410
* REM FT044420
  ENTRY... 11FQUO FT044430
* PROCESSING FOR THIS STATEMENT IS IDENTICAL TO THAT FOR THE FT044440
  'IF ACCUMULATOR OVERFLOW' STATEMENT. FT044450
  SPACE 3 FT044460
  11FQUO AXC 18,2 FT044470
  RESET POINTER FT044480
  TRA **2
ENTRY...
NEITHER N1 NOR N2 MAY EQUAL THE CURRENT CARD LABEL.
IF N1 = N2, THE CODE 'STZ OVFLOW' IS GENERATED, WHERE 'OVFLOW'
IS CELL 77462 OCTAL. IF N1 IS NOT EQUAL TO N2, THE CODE
'CAL OVFLOW', 'STZ OVFLOW', 'TNZ N1' IS GENERATED. THE
GENERATION OF THE FINAL INSTRUCTION 'TRA N2' IS DELAYED UNTIL
IT IS ASCERTAINED THAT THE NEXT EXECUTABLE STATEMENT IS NOT
Labeled N2.
SPACE 3

11FACC AXC 21,2
STL ,P
REM
11FAC1 AXT 1TRANS,4
SXA TRAFLG,4
REM
TSX SCNBAD,4
TPL ERROR1
REM
TSX STDLBL,4
LAS CARD
TRA ++2
ERROR 6,++1
REM
SLW ,P+1
SKIP CMCHARG
REM
ERROR 2,SKEND
REM
TSX SCNBAD,4
TPL ERROR1
REM
SLW ,P+3
REM
TSX STDLBL,4
LAS CARD
TRA ++2
ERROR 6,++1
REM
SLW ,P+2
REM
SKIP 63
ERROR 3,SKEND
ERROR 3,SKEND
REM
CAL ,P
REM
TNZ 11FAC2
CODEC DCT,N
CODECO TRA,P+1,R
TRA SKEND
REM
11FAC2 CAL ,P+1
ERA ITRAN2
REM
11FAC4 CODEC CALQVF,D
CODEC STZOVF,D
CODECO TNZ,P+1,R
TO 0444490
PAGE 128
* IF ACCUMULATOR OVERFLOW N1, N2
* ENTRY...
* NEITHER N1 NOR N2 MAY EQUAL THE CURRENT CARD LABEL.
* IF N1 = N2, THE CODE 'STZ OVFLOW' IS GENERATED, WHERE 'OVFLOW'
* 'CAL OVFLOW', 'STZ OVFLOW', 'TNZ N1' IS GENERATED. THE
* GENERATION OF THE FINAL INSTRUCTION 'TRA N2' IS DELAYED UNTIL
* IT IS ASCERTAINED THAT THE NEXT EXECUTABLE STATEMENT IS NOT
* Labeled N2.
* SPACE 3
11FACC AXC 21,2
STL ,P
REM
11FAC1 AXT 1TRANS,4
SXA TRAFLG,4
REM
TSX SCNBAD,4
TPL ERROR1
REM
TSX STDLBL,4
LAS CARD
TRA ++2
ERROR 6,++1
REM
SLW ,P+1
SKIP CMCHARG
REM
ERROR 2,SKEND
REM
TSX SCNBAD,4
TPL ERROR1
REM
SLW ,P+3
REM
TSX STDLBL,4
LAS CARD
TRA ++2
ERROR 6,++1
REM
SLW ,P+2
REM
SKIP 63
ERROR 3,SKEND
ERROR 3,SKEND
REM
CAL ,P
REM
TNZ 11FAC2
CODEC DCT,N
CODECO TRA,P+1,R
TRA SKEND
REM
11FAC2 CAL ,P+1
ERA ITRAN2
REM
11FAC4 CODEC CALQVF,D
CODEC STZOVF,D
CODECO TNZ,P+1,R
TO 0444490
PAGE 128
* IF ACCUMULATOR OVERFLOW N1, N2
* ENTRY...
* NEITHER N1 NOR N2 MAY EQUAL THE CURRENT CARD LABEL.
* IF N1 = N2, THE CODE 'STZ OVFLOW' IS GENERATED, WHERE 'OVFLOW'
* 'CAL OVFLOW', 'STZ OVFLOW', 'TNZ N1' IS GENERATED. THE
* GENERATION OF THE FINAL INSTRUCTION 'TRA N2' IS DELAYED UNTIL
* IT IS ASCERTAINED THAT THE NEXT EXECUTABLE STATEMENT IS NOT
* Labeled N2.
* SPACE 3
11FACC AXC 21,2
STL ,P
REM
11FAC1 AXT 1TRANS,4
SXA TRAFLG,4
REM
TSX SCNBAD,4
TPL ERROR1
REM
TSX STDLBL,4
LAS CARD
TRA ++2
ERROR 6,++1
REM
SLW ,P+1
SKIP CMCHARG
REM
ERROR 2,SKEND
REM
TSX SCNBAD,4
TPL ERROR1
REM
SLW ,P+3
REM
TSX STDLBL,4
LAS CARD
TRA ++2
ERROR 6,++1
REM
SLW ,P+2
REM
SKIP 63
ERROR 3,SKEND
ERROR 3,SKEND
REM
CAL ,P
REM
TNZ 11FAC2
CODEC DCT,N
CODECO TRA,P+1,R
TRA SKEND
REM
11FAC2 CAL ,P+1
ERA ITRAN2
REM
11FAC4 CODEC CALQVF,D
CODEC STZOVF,D
CODECO TNZ,P+1,R
TTL PROCESSOR - IF SENSE SWITCH

* IF (SENSE SWITCH 1) N1, N2
  REM
  ENTRY... ILFSWT
  THE NUMBER I IS CHECKED TO ASSURE THAT IT IS BETWEEN 1 AND 6
  INCLUSIVE. THE LABELS N1 AND N2 ARE TESTED TO DETERMINE
  THAT NEITHER IS EQUAL TO THE CURRENT CARD LABEL. N1 AND N2
  ARE PUT IN THE SYMBOL TABLE AND THE PATH-OF-FLOW FLAG 'BPATH'
  IS TURNED ON. THE CODE 'SWT I' IS GENERATED, FOLLOWED BY
  'TRA N2'. THE GENERATION OF THE FINAL INSTRUCTION 'TRA N1'
  IS DELAYED UNTIL THE LABEL ON THE NEXT EXECUTABLE STATEMENT
  IS FOUND. IF THIS LABEL IS THE SAME AS N1, NO CODE IS
  GENERATED.
  SPACE 3

ILFSWT AXA 16,2
STZ P
TRA **3

RESET POINTER AT N1
ZERO FLAG

FTO45110
FTO45120
FTO45130
FTO45140
FTO45150
FTO45160
FTO45170
FTO45180
FTO45190
FTO45200
FTO45210
FTO45220
FTO45230
FTO45240
FTO45250
FTO45260
FTO45270
ENTRY... 1IFS1

* THE NUMBER I MUST BE FROM 1 TO 4 INCLUSIVE. THE FIRST CODE
* GENERATED IS 'SLT 1'. THE OTHER DETAILS ARE THE SAME AS GIVEN
* IN THE SENSE SWITCH DESCRIPTION ABOVE.
SPAC 3

1IFS1
AXC 15,2  RESET POINTER
STL ,P  NON-ZERO FLAG
REM
AXT 1TRANS,4  SET ENTRY FOR LABEL ROUTINE
SXA TRAFL,4  
REM
CAL COLUMN-1,2  CHECK FOR CLOSING PARENTHESIS
ERA =H) 
AR 30  NO PARENTHESIS
INZ ERROR7
REM
CAL COLUMN-2,2  GET DIGIT
AR 30  
INZ ++2  ZERO NOT ALLOWED

1IFS2 ERROR 5,1IFS1
REM
LDQ =6  FOR SWITCH
GET ,P  TEST FLAG
LDQ =4  FOR LIGHT
TLQ 1IFS2  IMPROPER DIGIT
SLW ,P+1  SAVE DIGIT
REM
1IFS1
TSX SCNBD,4  GET N1
TPL ERROR 1 NOT A LABEL
SLW ,P+2  SAVE BCD N1
REM
TSX STOLBL,4  PUT N1 IN SYMTAB. MINUS SIGN
LASCARD  COMPARE WITH CURRENT LABEL
TRA ERROR ++2  IMPROPER TRANSFER
REM
SLW ITRAN2  SAVE POINTER FOR N1
SK1 CMCH  TEST FOR COMMA
ERROR 2,SKEND
REM
TSX SCNBD,4  GET N2
TPL ERROR 1 NOT A LABEL
REM
TSX STOLBL,4  PUT N2 IN ST. MINUS SIGN
LASCARD  COMPARE N2 WITH CURRENT LABEL
TRA ERROR ++2  IMPROPER TRANSFER
REM
SLW ,P+5  SAVE N2 POINTER
REM
SK 63  END OF STATEMENT EXPECTED
ERROR 3,SKEND
NOCODE  TEST CODE FLAG
REM
CAL SLT  FOR LIGHT. CODE IS... SLT 1
NZT ,P  TEST FLAG
CAL SWT  FOR SWITCH. CODE IS... SWT 1
<table>
<thead>
<tr>
<th>CODECO</th>
<th>.P+1,N</th>
<th>FT045880</th>
</tr>
</thead>
<tbody>
<tr>
<td>REM</td>
<td></td>
<td>FT045890</td>
</tr>
<tr>
<td>CODECO</td>
<td>TRA,.P+5,R</td>
<td>CODE IS... TRA N2</td>
</tr>
<tr>
<td>TRA</td>
<td>SKEND</td>
<td>FT045910</td>
</tr>
<tr>
<td>ENTRY</td>
<td>1SENSE</td>
<td>1SENSE</td>
</tr>
<tr>
<td>-------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>SENSE LIGHT I</td>
<td>ENTRY... 1SENSE</td>
<td>THE NUMBER I MUST BE FROM 0 TO 4 INCLUSIVE. THE CODE</td>
</tr>
<tr>
<td>IS 'SLF I'.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPACE 3</td>
<td>AXC 11,2</td>
<td>SET POINTER AT END OF STATEMENT</td>
</tr>
<tr>
<td>SKIP 63</td>
<td>ONLY ONE DIGIT ALLOWED</td>
<td></td>
</tr>
<tr>
<td>ERROR 5, SKEND</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>CAL COLUMN-2,2</td>
<td>CHECK SIZE OF DIGIT</td>
</tr>
<tr>
<td>ARS 30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDQ =4</td>
<td>TLQ ERRORS5</td>
<td>GREATER THAN 4, OR NOT A DIGIT</td>
</tr>
<tr>
<td>REM</td>
<td>CODE0 SLF,N</td>
<td>CODE IS... SLN I</td>
</tr>
<tr>
<td>TRA SKEND</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

PAGE 133
<table>
<thead>
<tr>
<th>ENTRY</th>
<th>1STOPX</th>
</tr>
</thead>
<tbody>
<tr>
<td>STOP I</td>
<td></td>
</tr>
<tr>
<td>TTL</td>
<td>PASS 1 PROCESSOR - STOP</td>
</tr>
</tbody>
</table>

**REMARKS**

- **ENTRY**: 1STOPX

**NOTES**

- The number 1 must be absent (i.e., zero) or be an octal number from 0 to 7777. The number is put in the constant.
- The address fields of both the AC and the MQ. The code 'MTR #1' is put out.

**I-STOPX**

<table>
<thead>
<tr>
<th>AXC</th>
<th>4,2</th>
<th>reset pointer for 'STOP'</th>
</tr>
</thead>
<tbody>
<tr>
<td>STL</td>
<td>*P</td>
<td>non-zero flag</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AXT</td>
<td>TRANS+1,4</td>
<td>set exit for subsequent label routine</td>
</tr>
<tr>
<td>SXT</td>
<td>TRAFLG,4</td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>*+3</td>
<td></td>
</tr>
</tbody>
</table>
* PAUSE 1

* ENTRY... IPAUSE

* THE TREATMENT IS THE SAME AS FOR THE STOP STATEMENT, EXCEPT

* THAT THE FINAL CODE IS 'HTR ++1'. THE HPR OPERATION CANNOT BE

* USED BECAUSE OF POSSIBLE COMPLICATIONS WITH THE

* DATA CHANNEL TRAP.

SPACE 3

1PAUSE

AXC 5,2    RESET POINTER FOR 'PAUSE'

STZ .P     ZERO FLAG

REM

SKIP 63    TEST IF OCTAL NUMBER PRESENT

TRA IPAUS3 NUMBER PRESENT

CODEC LGR74,N NUMBER NOT PRESENT. CODE IS... LGR 74

TRA IPAUS4

REM

IPAUS3

ZAC

LDQ COLUMN,2 GET NUMBER. TEST FOR OCTAL

AXT 6,2 SET FOR SIX BITES

LGL 3

TRA ++3

IPAUS2

ARS 3 DROP 'BCD' PART OF NUMBER

LGL 6 SHIFT IN NEXT BCD DIGIT

PAI

RFT 7 TEST HIGH-ORDER THREE BITS

TRA ++3 ALL BITS NOT OFF

TIX IPAUS2,2,1 RETURN

ERROR 13,SKEND NUMBER TOO LONG

REM

LGL 3

PAI

RNT 77 TEST FOR END OF STATEMENT

ERROR 4,SKEND NOT AN OCTAL NUMBER

ARS 6 JUSTIFY OCTAL NUMBER

NOCODE

REM

IPAUS1

TSX ASGON1,4 PUT OCTAL NUMBER IN CONSTANT TABLE

SLW .P+1

CODEO CAL,C CODE IS... CAL OCTAL

CODECO LOQ,.P+1,C LOQ OCTAL

REM

IPAUS4

CAL PCOUNT GET PCOUNT

NZT .P IS IT 'STOP'?

ADD =1 NO. CODE IS... HTR ++1

TSX CITBLD,4,P YES. CODE IS... HTR *

TRA SKEND
* TEMPORARY FORMAT STATEMENT PROCESSOR. DOES NOT CHECK FORMAT STRING FOR VALIDITY

* SPACE 3

PFMT CAL NXTLOC SET POOL POINTER INTO ADDRESS OF EQUIVALENT

STA EQUIV,4

AXT -1,1

PAX ,4

TXI ++1,4,1

* DROP ANY LEADING WORDS OF BLANKS AND WORD FORMAT

AXI 6,2

LDQ CARD,1

ZAC

LGL 6

ERA =H000001

TZE ++3

TIX *-4,2,1

TI X *-7,1,-1

ORA =H (1

TXN **3,2,1

LGL 6

TRA *-2

PFMT1 SLW POOL,4 FIRST NON-BLANK WORD FOUND. STASH WORD IN POOL

REM

TXI ++1,1,-1

CAL CARD,1

LAS =0767676767676

TXI PFMT1,4,1

TRA **2

TXI PFMT1,4,1

PX A ,4 END OF FORMAT STRING. SET NO. WORD IN STR

REM INTO POOL

SUB NXTLOC

LXA NXTLOC,1

STA POOL,1

TXI **1,4,1

SXA NXTLOC,4 RESET POOL POINTER TO NEXT AVAILABLE LOCATION.

TRA SKEND IN POOL
**TIL PASS I PROCESSOR — BACKSPACE**

* BACKSPACE I

REM

* ENTRY... 1BSRXX

* SEE THE WRITE OUTPUT TAPE STATEMENT DESCRIPTION FOR THE

* TREATMENT OF THE LOGICAL TAPE NUMBER I AND THE TRANSFER VECTOR

* ENTRY. THE INITIALIZATION ROUTINE NAME IS (BST).

* THE APPROPRIATE 'CAL ----' CODE IS GENERATED, FOLLOWED BY

* A 'TSX (BST);,4' INSTRUCTION.

SPACE 3

1BSRXX

AXT (BST),4 GET SUBROUTINE ENTRY

AXC 9,2 RESET POINTER

REM

1BSRX1 SXA 1RIT99,4 SAVE ENTRY

STL .P+3 FLAG3 NON-ZERO FOR NON-DATA-TRANSMIT

TRA 1RITX2 PROCESSING CONTINUED IN RIT SECTION
<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>TTL PROCESSOR - RE WIND</td>
</tr>
<tr>
<td>*</td>
<td>RE WIND 1</td>
</tr>
<tr>
<td>*</td>
<td>REM</td>
</tr>
<tr>
<td>*</td>
<td>ENTRY... 1REWXX</td>
</tr>
<tr>
<td>*</td>
<td>THE TREATMENT IS THE SAME AS FOR BACKSPACE, EXCEPT THAT THE EXTERNAL ROUTINE CALLED BY THE TSX OPERATION IS (RWT).</td>
</tr>
<tr>
<td>1REWXX</td>
<td>AXT (RWT),4 ENTRY FOR RE WIND</td>
</tr>
<tr>
<td>AXC</td>
<td>6,2</td>
</tr>
<tr>
<td>TRA</td>
<td>1BSRX1</td>
</tr>
<tr>
<td>Field</td>
<td>Value</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>TTL</td>
<td>END FILE I</td>
</tr>
<tr>
<td></td>
<td>REM</td>
</tr>
<tr>
<td></td>
<td>ENTRY... 1EOFXX</td>
</tr>
<tr>
<td></td>
<td>THE TREATMENT IS THE SAME AS FOR BACKSPACE, EXCEPT THAT THE EXTERNAL ROUTINE CALLED BY THE TSX OPERATION IS (EFT).</td>
</tr>
<tr>
<td>SPACE</td>
<td>3</td>
</tr>
<tr>
<td>1EOFXX</td>
<td>AXT (EFI), 4 ENTRY FOR END OF FILE</td>
</tr>
<tr>
<td></td>
<td>AXC 7,2</td>
</tr>
<tr>
<td></td>
<td>TRA 1BSRX1</td>
</tr>
</tbody>
</table>
PASS 1 PROCESSOR - READ

* READ FMT, LIST

* ENTRY... IREADX


* STATEMENT: THE INITIALIZATION CODE IS 'TSX (CSH),4'.

* THE TERMINATION CODE IS 'TSX (RTN),4'.

SPACE 3

IREADX

AXT (CSH),4 INITIALIZATION ENTRY FOR READ
AXT (RTN),1 TERMINATION ENTRY FOR INPUT
STL *P+6 FLAG6 NON-ZERO FOR INPUT
AXC 4,2
REM

IREA1

SXA 1RIT11+1,1 SET TERMINATION ROUTINE NAME
STZ *P+3 FLAG3 IS ZERO FOR DATA-TRANSMIT
STZ *P+4 FLAG4 IS ZERO FOR BCD
REM

SXA **2,4 SET TRANSFER VECTOR NAME
TSX OUTPV,4
REM

PZE ** CODE IS... TSX (XXX),4
TRA 1RITX7 PROCESSING CONTINUED IN RIT SECTION
* PRINT FMT, LIST
  
  REM

* ENTRY... 1PRINT

* THE TREATMENT IS THE SAME AS FOR THE READ STATEMENT, EXCEPT
* THAT THE INITIALIZATION ROUTINE NAME IS (SPH), AND THE LIST IS

* AN OUTPUT LIST.

  SPACE 3

  1PRINT AXT (SPH),4  INITIALIZATION ENTRY FOR PRINT
  AXT (FIL),1  TERMINATION ENTRY FOR OUTPUT
  STZ P+6  FLAG6 IS ZERO FOR OUTPUT
  AXc 5,2
  TRA 1READ1
* TTL PASS 1 PROCESSOR - PUNCH
  PUNCH FMT, LIST
  REM
  ENTRY... 1PUNCH
  * THE TREATMENT IS THE SAME AS FOR THE READ STATEMENT, EXCEPT
  THAT THE INITIALIZATION ROUTINE NAME IS (SCH), AND THE LIST IS
  ** AN OUTPUT LIST.
  SPACE 3
  1PUNCH AXT (SCH),4 INITIALIZATION FOR PUNCH
  TRA 1PRINT+1
# TSSL PROCESSOR - READ TAPE

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ TAPE I, LIST</td>
<td></td>
<td>FT047980</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FT047990</td>
</tr>
<tr>
<td>ENTRY... IRDTPX</td>
<td></td>
<td>FT048000</td>
</tr>
<tr>
<td>THE TREATMENT IS THE SAME AS FOR THE READ INPUT TAPE STATEMENT, EXCEPT THAT THERE IS NO FORMAT STATEMENT NAME, AND THE EXTERNAL SUBROUTINE NAMES ARE (TSB) FOR INITIALIZATION AND (RLR) FOR TERMINATION.</td>
<td></td>
<td>FT048020</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FT048030</td>
</tr>
<tr>
<td>IRDTPX AXT (TSB),4</td>
<td>INITIALIZATION FOR READ TAPE</td>
<td>FT048040</td>
</tr>
<tr>
<td>AXT (RLR),1</td>
<td>TERMINATION POINT FOR READ TAPE</td>
<td>FT048050</td>
</tr>
<tr>
<td>STL .P+6</td>
<td>FLAG6 NON-ZERO FOR INPUT</td>
<td>FT048060</td>
</tr>
<tr>
<td>AXC 8,2</td>
<td></td>
<td>FT048070</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FT048080</td>
</tr>
<tr>
<td>IRDIP1 SXA 1RIT99,4</td>
<td>SAVE TRANSFER VECTOR ENTRIES</td>
<td>FT048090</td>
</tr>
<tr>
<td>SXA 1RIT11,1</td>
<td></td>
<td>FT048100</td>
</tr>
<tr>
<td>STZ .P+3</td>
<td>FLAG3 IS ZERO FOR DATA-TRANSMIT</td>
<td>FT048110</td>
</tr>
<tr>
<td>STL .P+4</td>
<td>FLAG4 IS NON-ZERO FOR BINARY</td>
<td>FT048120</td>
</tr>
<tr>
<td>TRA 1RITX2</td>
<td>PROCESSING CONTINUED IN RIT SECTION</td>
<td>FT048130</td>
</tr>
</tbody>
</table>


* TTL PROCESSOR - WRITE TAPE
* WRITE TAPE I, LIST
* REM ENTRY... 1WRTPX
* THE TREATMENT IS THE SAME AS FOR THE WRITE OUTPUT TAPE STATEMENT,
* EXCEPT THAT THERE IS NO FORMAT STATEMENT NAME, AND THE
* EXTERNAL SUBROUTINE NAMES ARE (STB) FOR INITIALIZATION AND
* (WLR) FOR TERMINATION.
* (WLR) FOR TERMINATION.
SPACE 3
1WRTPX AXT (STB)*4 INITIALIZATION FOR WRITE TAPE
AXT (WLR)*1 TERMINATION FOR WRITE TAPE
STZ P+6 FLAG6 IS ZERO FOR OUTPUT
AXC 9,2
TRA 1RDTP1
WRITE OUTPUT TAPE I, FMT, LIST

ENTRY...  I1WOTXX

IF I IS A NUMBER, IT IS CONVERTED TO A BINARY DECREMENT INTEGER AND IS PUT IN THE CONSTANT TABLE. THE CODE 'CAL INTEGER' IS GENERATED.
IF I IS A FIXED POINT VARIABLE NAME, IT IS PUT IN THE SYMBOL TABLE. IF THE SYMBOL IS NOT A PARAMETER (SUBPROGRAM ARGUMENT), THE CODE 'CAL VARIABLE' IS GENERATED. IF I IS A PARAMETER (I.E. BARGT IS ON), THE PROLOG ROUTINE IS ENTERED. ON RETURN, THE CODE 'CAL ***' IS PUT OUT.

THE NAMES OF THE EXTERNAL SUBROUTINES FOR INITIALIZATION, (STH), AND TERMINATION, (FIL), ARE PUT IN THE TRANSFER VECTOR LIST.
THE INITIALIZATION CODE 'TSX (STH),4' IS GENERATED.

IF FMT IS A STATEMENT LABEL, IT IS PUT IN THE SYMBOL TABLE. IT MUST NOT ALREADY BE IN THE TABLE UNLESS IT IS FLAGGED AS A STRING (I.E. THE MSTRG BITS ARE ON). FOR A NEW SYMBOL TABLE ENTRY, THE MSTRG BITS ARE TURNED ON. THE CODE 'PZE STRNG' IS TURNED OUT.

IF FMT IS A VARIABLE NAME, IT IS PUT IN THE SYMBOL TABLE. THE NAME MUST ALREADY BE IN THE TABLE FLAGGED AS AN ARRAY. IF FMT IS NOT A SUBROUTINE PARAMETER, THE CODE 'PZE VARIABLE' IS PUT OUT. IF FMT IS A PARAMETER, THE PROLOG ROUTINE IS ENTERED. ON RETURN, THE CODE 'PZE ***' IS PUT OUT.

THE PROCESSOR NEXT TRANSFERS TO THE OUTPUT LIST PROCESSOR TO GENERATE THE OUTPUT CODE. ON RETURN, THE TERMINATION CODE 'TSX (FIL),4' IS PUT OUT.

SPACE  3

I1WOTXX AXT (STH),4  INITIALIZATION ENTRY FOR BCD WRITE
AXT (FIL),1  TERMINATION FOR BCD WRITE
STZ .P+6  FLAGG IS ZERO FOR OUTPUT
AXC 15+2
IRA 1RITX1
**TLPPASS1**  
**READ INPUT TAPE 1, FMT, LIST**  
**ENTRY... IRTXX**  
**THE TREATMENT IS THE SAME AS FOR THE WRITE OUTPUT TAPE**  
**STATEMENT, EXCEPT THAT THE LIST IS AN INPUT LIST AND THE**  
**EXTERNAL TERMINATION ROUTINE NAME IS (RTN).**  
**SPACE 3**

**1RTXX**  
**AXT (TSH),4**  
**AXT (RTN),1**  
**STL .P+6**  
**AXC 13,2**  
**REM**  
**1RTX1**  
**SXA 1RII99,4**  
**SXA 1RII11+1,1**  
**STZ .P+3**  
**STZ .P+4**  
**REM**  
**1RTX2**  
**TSX SCN8CD,4**  
**TPL 1RII13**  
**REM**  
**1RTX3**  
**TSX BCDFIX,4**  
**TSX ASCONI,4**  
**Coded CALC CODE IS... TAP NUMBER**  
**REM**  
**1RTX4**  
**TSX OUTPTV,4**  
**REM**  
**PZE **  
**REM**  
**NXT .P+3**  
**TRA 1RII14**  
**SKIP 63**  
**ERROR 3,SKEND**  
**TRA SKEND**  
**REM**  
**1RTX5**  
**ZET .P+4**  
**TRA 1RII15**  
**REM**  
**TSX LOCATE,4**  
**SLW .P+7**  
**TPL 1RII16**  
**REM**  
**PAX 4**  
**SIL MSTRG,4**  
**REM**  
**1RTX7**  
**TSX SCN9CD,4**  
**TPL 1RIT10**  
**REM**  
**XCL**  
**TSX LOCATE,4**  
**SLW .P+7**  
**TPL 1RII16**  
**REM**  
**PAX 4**  
**SIL MSTRG,4**  
**RA**  
**REM**  
**1RTX8**  
**IL MSTRG**  
**LFT 700000**  
**ERRROR 24,1RII15**  
**REM**  
**CODEC .P+7,F**  
**REM**
REM   SKIP   CMCHAR   TEST FOR COMMA
TRA   IRT13   NO COMMA
REM
AXT   POOL-ISTKFG,4   RESET ISTACK ROUTING ADDRESS
SXK   ISTKFG,4   FOR USE BY THE LIST PROCESSOR
STZ   REGIST   RESET REGISTER INDICATION
TSX   IOLIST,4   GO TO LIST PROCESSOR
TRA   IRT17   ON RETURN, SKIP END OF STATEMENT TEST
REM

IRIT13 SKIP   63   MUST BE END OF STATEMENT
ERROR 3,SKEND
REM
IRIT17 NIT   **3   IS THERE A TERMINATION ENTRY.
TRA   SKEND   NO.
REM
IRIT11 TSX   OUTPTV,4   PUT NAME IN TRANSFER VECTOR
PZE **   CODE IS... TSX (XXX),4
TRA   SKEND
REM
*
* PROCESSING FOR VARIABLE TAPE NUMBER
IRITX3 LAS   =HI00000   TEST FOR FIXED POINT
TRA   **1
LAS   =HO00000
TRA   **1
ERROR 12,IRITX4   NOT FIXED POINT
REM
SLW   FORM69   PUT IN SYMTAB
TSX   SYMSTD,4   PUT IN SYMTAB
REM
LFT   BEXITF+BINTF   MAY NOT BE SUBPROGRAM NAME
ERROR 69,**1
REM
LNT   BARGT   IS IT SUBROUTINE ARGUMENT.
TRA   IRTIX9   NO. REGULAR VARIABLE
REM
TSX   PROLOG,4   YES. FIX UP PROLOGUE
CODEC   CAL,V   CODE IS... CAL **
TRA   IRT12
REM
IRITX9 CODEEO   CAL,V   CODE IS... CAL TAPE. NUMBER
TRA   IRT12
REM
*
* PROCESSING FOR VARIABLE FORMAT STATEMENT SPECIFICATION
IRIT10 XCL   LOCATE,4   FIND SPOT FOR VARIABLE FORMAT IN SYMTAB
TSX   BARKY   IS IT DIMENSIONED.
LNT   14,IRIT16   NO. PUT OUT MESSAGE AND RETURN
REM
PAX **
SIL   BVARB   MARK AS VARIABLE IN EXECUTABLE STATEMENT
STI   EQUIV,4   RESTORE EQUIV WORD
REM
LNT   BARGT   IS IT SUBROUTINE ARGUMENT.
TRA   IRT15
REM
TSX   PROLOG,4   YES. FIX UP PROLOGUE
CODEC   PZE,N   CODE IS... PZE **
***NOTE*** READ AND WRITE DRUM ARE DEACTIVATED IN THIS

* VERSION OF FASTRAN
REM

REORGI EQU * POINT OR REORIGIN TO DEACTIVATE DRUM
SPACE 3

* WRITE DRUM I, J, LIST
REM

* ENTRY... 1WDRMX
* THE DRUM NUMBER I AND THE DRUM LOCATION J MUST EACH BE A
* FIXED POINT CONSTANT OR A FIXED POINT VARIABLE.
* THE CODE 'CAL I', 'TSX (SDR),4', 'CAL J', AND 'LDA' IS
* TURNED OUT. THE TREATMENT OF I AND J AND THE TRANSFER
* VECTOR ENTRY IS SIMILAR TO THAT DISCUSSED IN THE WRITE
* OUTPUT TAPE STATEMENT DESCRIPTION.
* THE DRUM LIST PROCESSOR IS ENTERED TO GENERATE THE OUTPUT
* CODE SEQUENCES.
SPACE 3

1WDRMX AXC 9,2
AXT (SDR),4 GET ENTRY TO WRITE DRUM ROUTINE
TRA 1WDRML
* INPUT PROCESSOR - READ DRUM

READ DRUM I, J, LIST

REM
ENTRY... 1RD RX

* THE PROCEDURE IS SIMILAR TO WRITE DRUM, EXCEPT THAT THE
EXTERNAL SUBROUTINE NAME IS (DRS), AND THE LIST IS TREATED
* AS AN INPUT LIST.

SPACE 3

1RD RX AXC 8,2
   AXT (DRS)

REM

1RD RXM1 SXX 1RD RXM4, S
   SAVE TRANSFER VECTOR ENTRY
   TSX SCNB C0,4
   GET DRUM NUMBER I
   TPL 1RD RXM2
   VARIABLE
   
REM
   TSX BCD FIX,4
   CONSTANT
   TSX ASCONL,4
   PUT IN CONSTANT TABLE
   CODEO CAL+C
   CODE IS... CAL I
   REM
   
1RD RXM5 SKIP CMCHAR
   CHECK FOR COMMA
   ERROR 2,SKEND
   REM
   TSX OUTPTV,4
   PUT ENTRY IN TRANSFER VECTOR
   CODE IS... TSX (XXX)
   * PZE
   REM
   TSX SCNB C0,4
   GET DRUM LOCATION J
   TPL 1RD RXM2
   VARIABLE
   
REM
   TSX BCD FIX,4
   CONSTANT
   TSX ASCONL,4
   PUT IN CONSTANT TABLE
   CODEO CAL+C
   CODE IS... CAL J
   CODEC LDA+N
   CODE IS... LDA
   REM
   SKIP1 CMCHAR
   CHECK FOR POSSIBLE COMMA
   TRA 1RD R MX
   TO DRUM LIST PROCESSOR
   REM
   
1RD RXM6 SKIP 63
   END OF STATEMENT REQUIRED
   ERROR 3,SKEND
   TRA SKEND
   REM

* PROCESSING OF VARIABLE DRUM NUMBER AND VARIABLE DRUM ADDRESS

* 1RD RXM2 LAS =H100000
   TEST FOR FIXED POINT
   TRA ++1
   LAS =H000000
   TRA ++1
   ERROR 12,SKEND
   NOT FIXED POINT
   
REM
   TSX SYMST0,4
   OKAY.. PUT IN SYMTAB
   LNT BART
   TEST FOR SUBROUTINE ARGUMENT
   TRA 1RDRM3
   NOT AN ARGUMENT
   
REM
   TSX PROLOG,4
   SUBROUTINE ARGUMENT. FIX UP PROLOG.
   CODEC CAL+N
   CODE IS... CAL **
   TRA 1RDRM5
   REM
   
1RDRM3 CODEO CAL+V
   CODE IS... CAL X
   TRA 1RDRM5
   SPACE 3
<table>
<thead>
<tr>
<th>ICDRM</th>
<th>TRA</th>
<th>SKEND</th>
<th>***** TEMPORARY DRUM LIST PROCESSOR *****</th>
<th>FT050750</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPACE</td>
<td>3</td>
<td></td>
<td></td>
<td>FT050760</td>
</tr>
<tr>
<td>ORG</td>
<td>REORG1</td>
<td>TO DELETE DRUM CODE FROM THIS VERSION</td>
<td>FT050770</td>
<td></td>
</tr>
</tbody>
</table>
COMMENTARY ON I/O LIST PROCESSOR -

SPACE 2

ON ENTRY TO I/O LIST THE COLUMN POINTER IS POSITIONED AT THAT

CHARACTER IN THE I/O STATEMENT FOLLOWING THE COMMA AFTER THE

FORMAT STATEMENT NUMBER(BCD) OR AFTER THE TAPE NUMBER (BINARY).

REM

THE I/O LIST PROCESSOR IS BROKEN INTO TWO PARTS. THE FIRST PART

(BEGINNING AT I/O LIST 1) BUILDS A STACK AND P STACK AND

AND THE SECOND PART (I/O LIST 2) PROCESSES THESE TWO LISTS AND

GENERATES THE OBJECT CODE.

SPACE 2

I/O LIST PROCESSOR - PART 1-

I/O LIST ITEMS, LEFT AND RIGHT PARENTHESES CAUSE A STACK

ENTRIES TO BE GENERATED. ASTACK ENTRIES FOR LIST ITEMS ARE

HANDLED LIKE ITEMS IN AN ARITHMETIC STATEMENT. A LEFT PAREN.

IN AN I/O LIST WILL CAUSE THE FOLLOWING ASTACK ENTRY TO BE

GENERATED,

MTH (PAREN. LEVEL),0

IF INDEXING IS INVOLVED IN AN I/O LIST THEN THE ASTACK ENTRY

FOR THE LEFT PAREN. WHICH BEGINS THE RANGE OF THE INDEX IS,

MTH (PAREN. LEVEL), PSTACK PTR. FOR INDEX)

A RIGHT PARENTHESIS THAT DOES NOT FOLLOW INDEXING

SPECIFICATIONS IN AN I/O LIST DOES NOT CAUSE GENERATION OF

AN ASTACK ENTRY. RIGHT PARENS. FOLLOWING INDEXING SPEC.

CAUSE THE FOLLOWING ASTACK ENTRY TO BE GENERATED,

MIW (PSTACK INDEX PTR.),0

ENTRIES IN P STACK ARE MADE ONLY WHEN INDEXING IS INVOLVED

(I.E. I = 11,12,13). WHENEVER INDEXING IS ENCOUNTERED FIVE

P STACK WORDS ARE GENERATED AS FOLLOWS,

MTH (ASTACK PTR. FOR THIS PAREN. LEVEL),0

INDEXING VARIABLE (1)

INITIAL INDEX VALUE (11)

FINAL INDEX VALUE (12)

INDEX INCREMENT (13)

DURING EXECUTION OF PART 1, LOCATIONS (CURPTR-1), (CURPTR-2),

AND (CURPTR-3) CONTAIN THE A STACK POINTERS FOR PAREN.

LEVELS 1, 2, AND 3. PARENTHESIS NESTING CANNOT BE GREATER

THAN 3. LOCATION I/OListView CONTAINS THE CURRENT PAREN. LEVEL.

SPACE 2

I/O LIST PROCESSOR - PART 2-

THE SECOND PART OF THE I/O LIST PROCESSOR SIMPLY USES THE

A STACK AND P STACK ENTRIES GENERATED BY PART 1 AND

OUTPUT THE OBJECT CODE ON C I T USING THE CODE AND CODEN

ROUTINES. ITEMS IN INPUT LISTS ARE FLAGGED AS BEING DEFINED

BY TURNING ON B L H S X IN E Q U I T. THE (SLO) OR (SLI)

SEQUENCES ARE GENERATED FOR NON-SUBSCRIPTED ARRAY ITEMS.

SPACE 2

CONSIDER THE FOLLOWING EXAMPLE,

REM

DIMENSION A(100),B(10,100)

READ INPUT TAPE 5,100,C,(D),(B(I,J),I=1,10),A(J),J=J1,100,3

THE FOLLOWING A STACK ENTRIES WOULD BE GENERATED BY
PART 1.

ENTRY

REMARKS

C
MTH 1,,0
NORMAL ASTACK ENTRY
ENTRY FOR LEFT PAREN.:

D
MTH 1,,6
NORMAL ASTACK ENTRY
ADDR. = LEVEL 1

MTH 2,,1
LEVEL,,PSTACK POINTER
NORMAL ASTACK ENTRY

B(I,J)
MTW 1,,0
ADDR. = PSTACK POINTER
GENERATED BY RIGHT
PAREN. FOLLOWING INDEX
ON I.

A(I)
MTH 6,,0
NORMAL ASTACK ENTRY
ADDR. = PSTACK POINTER
GENERATED BY RIGHT
PAREN. FOLLOWING INDEX
ON J.

SPACE 2

THE FOLLOWING P S T A C K ENTRIES WOULD BE GENERATED BY
THE TWO INDEXING PARAMETERS I AND J.

ENTRY

REMARKS

MTH 2,,5
ADDR. = PAREN. LEVEL
DECR. = ASTACK POINTER
FIRST INDEX VAR.

I
1
INITIAL VALUE

10
FINAL VALUE

1
INCREMENT

MTH 1,,4
ADDR. = PAREN. LEVEL
DECR. = ASTACK POINTER
SECOND INDEX VAR.

J
J1
INITIAL VALUE: THIS
AN ASTACK-TYPE ENTRY.

100
FINAL VALUE

3
INCREMENT

SPACE 2

THE OBJECT CODE GENERATED BY THIS READ STATEMENT WOULD BE,

CALS =05000000
TSX \$15\$H,4
PZE (LOC. FORMAT 100)
STR C
STR D
LXO J1,4
X SXO J,4
AXT 1,4
Y SXO 1,4
LDQ =10
MPY J
LDQ =1
* STO WORK1
  MPY I
  ADD WORK1
  ALS 17
  STO WORK1
  STR
  LXD WORK1,4
  STQ B+1,4
  LXD I,4
  TXI ++1,4+1
  TXL Y,4,10
  STR
  LXD J,4
  STQ A+1,4
  LXD J,4
  TXI ++1,4+3
  TXL X,4,100
  TSX $(RTN),4
  RESTORE TO NORMAL
<table>
<thead>
<tr>
<th>IOLIST</th>
<th>EXIOLS,4</th>
<th>SAVE XR4 FOR RETURN</th>
<th>FT052190</th>
</tr>
</thead>
<tbody>
<tr>
<td>SXA</td>
<td></td>
<td>FT052200</td>
<td></td>
</tr>
<tr>
<td>STZ</td>
<td>DBLMOD</td>
<td>NEEDED FOR 'SCAN' TO WORK OKAY</td>
<td>FT052210</td>
</tr>
<tr>
<td>STZ</td>
<td>PSTACK</td>
<td>INITIALIZE FIRST PSTACK ENTRY</td>
<td>FT052220</td>
</tr>
<tr>
<td>STZ</td>
<td>ASTACK</td>
<td>FIRST ASTACK ENTRY</td>
<td>FT052240</td>
</tr>
<tr>
<td>STZ</td>
<td>IOLVL</td>
<td>PAREN LEVEL INDICATOR</td>
<td>FT052250</td>
</tr>
<tr>
<td>STZ</td>
<td>NODGAR</td>
<td>INITIALIZE NO. DO PARAMS TO ZERO</td>
<td>FT052260</td>
</tr>
<tr>
<td>AXT</td>
<td>0,1</td>
<td>XR1 IS RUNNING ASTACK POINTER</td>
<td>FT052270</td>
</tr>
<tr>
<td>TSX</td>
<td>SKIP,4,EOS</td>
<td>IS NEXT END-OF-STATEMENT CHARACTER</td>
<td>FT052290</td>
</tr>
<tr>
<td>TRA</td>
<td>++2</td>
<td>NO, SKIP NEXT INSTR.</td>
<td>FT052290</td>
</tr>
<tr>
<td>TRA</td>
<td>EXIOLS</td>
<td>YES, EXIT -- NO LIST</td>
<td>FT052290</td>
</tr>
<tr>
<td>IOLST1</td>
<td>TSX</td>
<td>SKIP,4,LPCHAR</td>
<td>IS NEXT CHARACTER</td>
</tr>
<tr>
<td>TRA</td>
<td>IOLST2</td>
<td>NO, GO TO IOLST2</td>
<td>FT052320</td>
</tr>
<tr>
<td>LXA</td>
<td>IOLVL,4</td>
<td>GET CURRENT PAREN LEVEL</td>
<td>FT052330</td>
</tr>
<tr>
<td>TXH</td>
<td>IOLSTA,4,2</td>
<td>IF LEVEL THREE, ERROR</td>
<td>FT052340</td>
</tr>
<tr>
<td>TXI</td>
<td>++1,4,1</td>
<td>INCREASE LEVEL BY 1</td>
<td>FT052350</td>
</tr>
<tr>
<td>SXA</td>
<td>IOLVL,4</td>
<td>SAVE IN IOLVL</td>
<td>FT052360</td>
</tr>
<tr>
<td>PXA</td>
<td>,4</td>
<td>BUILD ASTACK ENTRY</td>
<td>FT052370</td>
</tr>
<tr>
<td>ORA</td>
<td>=0700000000000</td>
<td>MTH LEVEL NO.,0</td>
<td>FT052380</td>
</tr>
<tr>
<td>SLW</td>
<td>ASTACK,1</td>
<td>SET ASTACK ENTRY</td>
<td>FT052390</td>
</tr>
<tr>
<td>PXA</td>
<td>,1</td>
<td>INDICATE LAST ASTACK POS, THIS LEVEL</td>
<td>FT052400</td>
</tr>
<tr>
<td>STA</td>
<td>CURPTR,4</td>
<td>SAVE THIS IN CURPTR VECTOR</td>
<td>FT052410</td>
</tr>
<tr>
<td>IOLST1</td>
<td>IOLST1,1</td>
<td>BUMP ASTACK PTR. AND LOOK FOR</td>
<td>FT052420</td>
</tr>
<tr>
<td>IOLST2</td>
<td>TSX</td>
<td>SCAN,4</td>
<td>SCAN OFF NEXT LIST ELEMENT</td>
</tr>
<tr>
<td>ERROR</td>
<td>48,SKEND</td>
<td>E-O-S IMPOSSIBLE</td>
<td>FT052440</td>
</tr>
<tr>
<td>TRA</td>
<td>++2</td>
<td>CONSTANT --- GO PUT IT OUT</td>
<td>FT052450</td>
</tr>
<tr>
<td>ERKOR</td>
<td>48,SKEND</td>
<td>MISPLACED OPERATOR</td>
<td>FT052460</td>
</tr>
<tr>
<td>SLW</td>
<td>ASTACK,1</td>
<td>NO, SAVE THIS IN ASTACK</td>
<td>FT052470</td>
</tr>
<tr>
<td>PXA</td>
<td>,1</td>
<td>BUMP ASTACK POINTER</td>
<td>FT052480</td>
</tr>
<tr>
<td>STZ</td>
<td>ASTACK,1</td>
<td>KEEP NEXT ASTACK ENTRY CLEAR</td>
<td>FT052490</td>
</tr>
<tr>
<td>CAL</td>
<td>COLUMN,2</td>
<td>OBSERVE NEXT CHAR. IN STATEMENT</td>
<td>FT052500</td>
</tr>
<tr>
<td>ANA</td>
<td>=0770000000000</td>
<td>X</td>
<td>FT052510</td>
</tr>
<tr>
<td>LAS</td>
<td>=H=00000</td>
<td>IS IT =</td>
<td>FT052520</td>
</tr>
<tr>
<td>TRA</td>
<td>++2</td>
<td>NO</td>
<td>FT052530</td>
</tr>
<tr>
<td>TXI</td>
<td>IOLST6,2,-1</td>
<td>YES, GO TO IOLST6</td>
<td>FT052540</td>
</tr>
<tr>
<td>LAS</td>
<td>=H=00000</td>
<td>NO, IS IT</td>
<td>FT052550</td>
</tr>
<tr>
<td>TRA</td>
<td>++2</td>
<td>NO</td>
<td>FT052560</td>
</tr>
<tr>
<td>TXI</td>
<td>IOLST4,2,-1</td>
<td>YES, GO TO IOLST4</td>
<td>FT052570</td>
</tr>
<tr>
<td>LAS</td>
<td>=H=00000</td>
<td>NO, IS IT</td>
<td>FT052580</td>
</tr>
<tr>
<td>TRA</td>
<td>++2</td>
<td>NO</td>
<td>FT052590</td>
</tr>
<tr>
<td>TXI</td>
<td>IOLST1,2,-1</td>
<td>YES, GO TO IOLST1</td>
<td>FT052600</td>
</tr>
<tr>
<td>LAS</td>
<td>=0770000000000</td>
<td>IS IT END-OF-STATEMENT</td>
<td>FT052610</td>
</tr>
<tr>
<td>TRA</td>
<td>++2</td>
<td>NO</td>
<td>FT052620</td>
</tr>
<tr>
<td>TRA</td>
<td>IOLSTL</td>
<td>GO TO PART 2 OF IOLST PROCESSOR</td>
<td>FT052630</td>
</tr>
<tr>
<td>LAS</td>
<td>=H=00000</td>
<td>NO, IS IT</td>
<td>FT052640</td>
</tr>
<tr>
<td>ERROR</td>
<td>47,SKEND</td>
<td>INVALID LIST</td>
<td>FT052650</td>
</tr>
<tr>
<td>TXI</td>
<td>++2,2,-1</td>
<td>YES, GO TO IOLST9</td>
<td>FT052660</td>
</tr>
<tr>
<td>TRA</td>
<td>=2</td>
<td>NO, AN ERROR</td>
<td>FT052670</td>
</tr>
<tr>
<td>IOLST9</td>
<td>LXA</td>
<td>IOLVL,4</td>
<td>PROCESS RIGHT PARENTHESIS</td>
</tr>
<tr>
<td>TXL</td>
<td>IOLST8,4,0</td>
<td>ERROR IF LEVEL ALREADY ZERO</td>
<td>FT052690</td>
</tr>
<tr>
<td>TXI</td>
<td>++1,4,1</td>
<td>DECREASE LEVEL BY 1</td>
<td>FT052700</td>
</tr>
<tr>
<td>SXA</td>
<td>IOLVL,4</td>
<td>SAVE IN IOLVL</td>
<td>FT052710</td>
</tr>
<tr>
<td>CAL</td>
<td>COLUMN,2</td>
<td>GET NEXT CHARACTER OF STATEMENT</td>
<td>FT052720</td>
</tr>
<tr>
<td>ANA</td>
<td>=0770000000000</td>
<td>X</td>
<td>FT052730</td>
</tr>
<tr>
<td>TRA</td>
<td>IOLST3</td>
<td>GO LOOK FOR, E-O-S-</td>
<td>FT052740</td>
</tr>
<tr>
<td>IOLST4</td>
<td>LDI</td>
<td>ASTACK+1,1</td>
<td>PROCESS LEFT PARENTHESIS</td>
</tr>
<tr>
<td>LNT</td>
<td>BARRY</td>
<td>IS THIS AN ARRAY</td>
<td>FT052760</td>
</tr>
<tr>
<td>ERROR</td>
<td>47,SKEND</td>
<td>SUBSCRIPTED VAR. NOT AN ARRAY</td>
<td>FT052770</td>
</tr>
<tr>
<td>TXI</td>
<td>++1,1,1</td>
<td>JUSTIFY THE INDEX</td>
<td>FT052780</td>
</tr>
</tbody>
</table>
TSX INDEX,4
  INDEX,4
  RESTORE THE INDEX
  FT052790
  FT052800
  FT052810
  FT052830
  FT052840

TRA IOLST4-3
  GO LOOK FOR , E-D-S )
  FT052850

LOI ASTACK+1,1
  PROCESS =
  FT052860

STZ ASTACK+1,1
  PICK-UP INDEX VARIABLE
  FT052870

TXI **1,1,-1
  CLEAR INDEX VAR. FROM ASTACK
  FT052880

SXA APTR+1
  SAVE ASTACK POINTER
  FT052890

ILL MINTG
  IS THIS A FIXED POINT INTEGER
  FT052895

LFT 700000
X

ERROR 46,SKEND
  INDEX VAR. NOT FIXED POINT
  FT052900

NXT IOLVL
  CHECK THAT LEVEL IS NON-ZERO
  FT052910

ERROR 46,SKEND
  INDEXING AT LEVEL ZERO
  FT052920

LXA IOLVL+4
  GET ASTACK POINTER FOR THIS LEVEL
  FT052930

CAL CURPTR+4
X

ORA =07000000000000
X

LXA PPTR,1
  GENERATE PSTACK ENTRY
  FT052940

SLW PSTACK,1
  MTH ASTACK PTR. THIS LEVEL,,0
  FT052950

PAX ,4
  SET DECREMENT OF ASTACK ENTRY TO
  FT052960

TXI **1,1
  PSTACK POINTER
  FT052970

PKX X

STD ASTACK,4
  PUT THIS IS ASTACK ENTRY
  FT052980

STI PSTACK,1
  X
  FT053000

IOLST7 TXI DOARG,4
  GO SCAN OFF INDEXING PARAMETERS
  FT053010

TXI **1,1
  BUMP PSTACK POINTER
  FT053020

SLW PSTACK,1
  STORE PARAMETERS IN PSTACK
  FT053030

CLA NODOAR
  BUMP NO. INDEX PARAMS BY 1
  FT053040

ADD =1
X

STD NODOAR
X

TSX SKIP,4,RPCCHAR
IS NEXT CHARACTER )

TRA IOLST8
  NO, MAKE SURE IT IS ,

CAL =2
  DETERMINE NO. DO PARAMETERS
  FT053060

LAS NODOAR
  ARE THERE TWO PARAMETERS
  FT053090

ERROR 46,SKEND
  NO, NOT ENOUGH PARAMETERS
  FT053100

TRA IOLST5
  YES, GO SET THIRD TO 1
  FT053110

CAL =3
X

LAS NODOAR
  ARE THERE THREE PARAMETERS
  FT053120

LOI NODOAR
YES

TRA IOLST5+3
  YES, GO TO IOLST5+3
  FT053130

ERROR 46,SKEND
  TOO MANY INDEXING PARAMETERS
  FT053140

CAL =1
  SET THIRD INDEXING PARAMETER TO 1
  FT053150

TXI **1,1,1
  BUMP PSTACK POINTER
  FT053170

SLW PSTACK,1
  SET THIS PARAMETER IN PSTACK
  FT053180

TXI **1,1
  FT053190

SXA PPTR,1
  SAVE PSTACK PTR. FOR NEXT TIME AROUND
  FT053200

TXX **1,1,-5
  FT053210

TXX X

LXA APTR,1
  RESTORE XR1 TO ASTACK POINTER
  FT053220

ORA =06000000000000
X

SLW ASTACK,1

TXI **1,1

STZ NODOAR
  RESET NO. DO PARAMS TO ZERO
  FT053230

TRA IOLST9
  GO CONTINUE LIST PROCESSING
  FT053240

IOLST8 TXS SKIP,*,CMCHAR
  IS NEXT CHARACTER ,
  FT053250

ERROR 46,SKEND
  COMMA EXPECTED IN INDEXING PORTION
  FT053260

TRA IOLST7
  YES, SKIP THE , AND GO FOR NEXT PARAM
  FT053270

DOARG SCA EXDOAR,4
  SAVE XR4
  FT053280

TSX SCNBGD,4
  SCAN OFF NEXT SYMBOL OR CONSTANT
  FT053290

TMI DOARG1
  TRANSFER IF IT WAS A CONSTANT
  FT053300

TSX SYMSTB,4
  ENTER IN SYMTAB AND VERIFY FIXED PI.
  FT053310

RIL 7
  TURN OFF SUBR. ARG. BITS
  FT053320
RIR 700000
STI DOARG2
STA DOARG2
CAL DOARG2
EXDOAR AXT...4
TRA 1,4
DDARGI TSX BCDFIX,4
ARS 18
TRA EXDOAR
IOLSTA ERROR 48,SKEND
IOLSTB ERROR 48,SKEND
IOLSTL ZET IOLVL
ERROR 48,SKEND
SXD **5,1
AXT 0,1
CAL ASTACK,1
TNZ IOLSTZ
TXI **1,1,1
TXL *=3,1,...
EXIOLS AXT...4
TRA 1,4
IOLSTZ PAI
LNT 700000
TRA IOLSTQ
LFT 077777
TRA **2
TXI EXIOLS-1,1,1
PDX ,2
STZ IOLT11
STZ IOLT12
STZ IOLT13
STZ IOLT14
LDI PSTACK-3,2
LFT 777777
TRA **3
STI IOLT11
TRA IOLSTM
STI SAVE
TSX NEXILP,4
SLW IOLT13
CODE LXD4,SAVE
CAL IOLT13
ORA SXD4
TSX CITEBLD,4,R
LDI PSTACK-2,2
LFT 777777
TRA **3
STI IOLT12
TRA IOLSTM
STI SAVE
TSX NEXILP,4
SLW IOLT14
CODE LXD4,SAVE
CAL IOLT14
ORA SXD4
TSX CITEBLD,4,R
LDI PSTACK-1,2
LFT 777777
TRA IOLSTO
PIA GENERATE AXT (11),4
<table>
<thead>
<tr>
<th>ORA</th>
<th>AXT4</th>
<th>X</th>
<th>FTO53990</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSX</td>
<td>CIBL0D,4,N</td>
<td>X</td>
<td>FTO54000</td>
</tr>
<tr>
<td>IRA</td>
<td>IOLSTP</td>
<td>X</td>
<td>FTO54010</td>
</tr>
<tr>
<td>IOLSTP</td>
<td>IOLSTP</td>
<td>X</td>
<td>FTO54020</td>
</tr>
<tr>
<td>CODE</td>
<td>LXD4,SAVE</td>
<td>GENERATE LXD (11),4</td>
<td>FTO54030</td>
</tr>
<tr>
<td>CAL</td>
<td>IOLTL13</td>
<td>SET UP PSTACK AS FOLLOWS (THIS INDEX)</td>
<td>FTO54040</td>
</tr>
<tr>
<td>ALS</td>
<td>18</td>
<td>MTH (ASTACK PTR.),,,0</td>
<td>FTO54050</td>
</tr>
<tr>
<td>ORA</td>
<td>PCOUNT</td>
<td>(POOL PTR. 12),,0</td>
<td>FTO54060</td>
</tr>
<tr>
<td>SLW</td>
<td>PSTACK-3,2</td>
<td>PZE (I2),,(I3)</td>
<td>FTO54070</td>
</tr>
<tr>
<td>LDQ</td>
<td>IOLTL14</td>
<td>(1)</td>
<td>FTO54080</td>
</tr>
<tr>
<td>STQ</td>
<td>PSTACK-2,2</td>
<td></td>
<td>FTO54090</td>
</tr>
<tr>
<td>SLW</td>
<td>PSTACK-2,2</td>
<td>PZE (PCOUNT),,(POOL PTR. (I3))</td>
<td>FTO54110</td>
</tr>
<tr>
<td>SLW</td>
<td>SAVE</td>
<td>GENERATE SXD (11),4</td>
<td>FTO54120</td>
</tr>
<tr>
<td>PAX</td>
<td>4</td>
<td>INDICATE INDEXING VARIABLE IN IOLIST</td>
<td>FTO54130</td>
</tr>
<tr>
<td>LDI</td>
<td>EQUIV,4</td>
<td>X</td>
<td>FTO54140</td>
</tr>
<tr>
<td>STI</td>
<td>EQUIV,4</td>
<td>X</td>
<td>FTO54150</td>
</tr>
<tr>
<td>TSX</td>
<td>IOLTL14</td>
<td>X</td>
<td>FTO54160</td>
</tr>
<tr>
<td>CODE</td>
<td>SXD4,SAVE</td>
<td>X</td>
<td>FTO54170</td>
</tr>
<tr>
<td>CAL</td>
<td>IOLTL11</td>
<td>X</td>
<td>FTO54180</td>
</tr>
<tr>
<td>ALS</td>
<td>18</td>
<td>X</td>
<td>FTO54190</td>
</tr>
<tr>
<td>ORA</td>
<td>IOLTL12</td>
<td>X</td>
<td>FTO54200</td>
</tr>
<tr>
<td>SLW</td>
<td>PSTACK-1,2</td>
<td>X</td>
<td>FTO54210</td>
</tr>
<tr>
<td>IOLSTQ</td>
<td>IOLSTQ</td>
<td>X</td>
<td>FTO54220</td>
</tr>
<tr>
<td>TXI</td>
<td>EXIOLS-1,1</td>
<td>NOW GO GET NEXT ASTACK ENTRY</td>
<td>FTO54230</td>
</tr>
<tr>
<td>LMT</td>
<td>600000</td>
<td>IS THIS A RIGHT PAREN ENTRY</td>
<td>FTO54240</td>
</tr>
<tr>
<td>IRA</td>
<td>IOLSTR</td>
<td>NO, GO PROCESS THIS LIST ITEM</td>
<td>FTO54250</td>
</tr>
<tr>
<td>PAX</td>
<td>2</td>
<td>X</td>
<td>FTO54260</td>
</tr>
<tr>
<td>CAL</td>
<td>PSTACK-3,2</td>
<td>X</td>
<td>FTO54270</td>
</tr>
<tr>
<td>SLW</td>
<td>SAVE</td>
<td>X</td>
<td>FTO54280</td>
</tr>
<tr>
<td>CODE</td>
<td>LXD4,SAVE</td>
<td>X</td>
<td>FTO54290</td>
</tr>
<tr>
<td>CAL</td>
<td>PSTACK-4,2</td>
<td>X</td>
<td>FTO54300</td>
</tr>
<tr>
<td>ALS</td>
<td>3</td>
<td></td>
<td>FTO54310</td>
</tr>
<tr>
<td>PDX</td>
<td>4</td>
<td></td>
<td>FTO54320</td>
</tr>
<tr>
<td>TXL</td>
<td>**3,4,0</td>
<td></td>
<td>FTO54330</td>
</tr>
<tr>
<td>CAL</td>
<td>PCOUNT</td>
<td>X</td>
<td>FTO54340</td>
</tr>
<tr>
<td>SLW</td>
<td>EQUIV,4</td>
<td>X</td>
<td>FTO54350</td>
</tr>
<tr>
<td>CAL</td>
<td>PSTACK-2,2</td>
<td>X</td>
<td>FTO54360</td>
</tr>
<tr>
<td>ANA</td>
<td>=077777700000</td>
<td>X</td>
<td>FTO54370</td>
</tr>
<tr>
<td>IRA</td>
<td>TXI14</td>
<td>X</td>
<td>FTO54380</td>
</tr>
<tr>
<td>ACL</td>
<td>PCOUNT</td>
<td>X</td>
<td>FTO54390</td>
</tr>
<tr>
<td>TSX</td>
<td>CIBL0D,4,P</td>
<td>OUT GOES THE TXI</td>
<td>FTO54400</td>
</tr>
<tr>
<td>CAL</td>
<td>PSTACK-1,2</td>
<td>X</td>
<td>FTO54410</td>
</tr>
<tr>
<td>TZE</td>
<td>**4</td>
<td></td>
<td>FTO54420</td>
</tr>
<tr>
<td>PAX</td>
<td>4</td>
<td></td>
<td>FTO54430</td>
</tr>
<tr>
<td>CAL</td>
<td>PCOUNT</td>
<td>X</td>
<td>FTO54440</td>
</tr>
<tr>
<td>SLW</td>
<td>EQUIV,4</td>
<td>X</td>
<td>FTO54450</td>
</tr>
<tr>
<td>CAL</td>
<td>PSTACK-2,2</td>
<td>NOW PUT OUT THE TXL</td>
<td>FTO54460</td>
</tr>
<tr>
<td>ANA</td>
<td>=0777777</td>
<td>X</td>
<td>FTO54470</td>
</tr>
<tr>
<td>ALS</td>
<td>18</td>
<td>X</td>
<td>FTO54480</td>
</tr>
<tr>
<td>SLW</td>
<td>SAVE</td>
<td>X</td>
<td>FTO54490</td>
</tr>
<tr>
<td>CAL</td>
<td>PSTACK-4,2</td>
<td>X</td>
<td>FTO54500</td>
</tr>
<tr>
<td>ANA</td>
<td>=0777777</td>
<td>X</td>
<td>FTO54510</td>
</tr>
<tr>
<td>ORA</td>
<td>SAVE</td>
<td>X</td>
<td>FTO54520</td>
</tr>
<tr>
<td>ORA</td>
<td>TXL4</td>
<td>X</td>
<td>FTO54530</td>
</tr>
<tr>
<td>TSX</td>
<td>CIBL0D,4,P</td>
<td>X</td>
<td>FTO54540</td>
</tr>
<tr>
<td>TXI</td>
<td>EXIOLS-1,1</td>
<td>GO CONTINUE ASTACK PROCESSING</td>
<td>FTO54550</td>
</tr>
<tr>
<td>IOLSTR</td>
<td>SLW</td>
<td>SAVE TO OUTPUT LIST ITEMS</td>
<td>FTO54560</td>
</tr>
<tr>
<td>PAX</td>
<td>4</td>
<td>GET EQUIV POINTER</td>
<td>FTO54570</td>
</tr>
<tr>
<td>NZT</td>
<td>IOFALG</td>
<td>IS THIS INPUT OR OUTPUT</td>
<td>FTO54580</td>
</tr>
</tbody>
</table>
### TTL PASS 1 - ARITHMETIC STATEMENTS

**ENTRY... 1ARITH**

*The processor is entered by default via the reserved word tests.*

*In other words, statements failing to meet requirements for reserved words are considered to be arithmetic statements.*

*The compil routine is entered immediately, and the statement processed. The statement must, of course, contain an equal sign.*

*If cell 2 has been saved (for dp and ca statements), it is restored prior to exiting from this processor.*

<table>
<thead>
<tr>
<th>IARITH AXC 0,2</th>
<th>Set pointer to first letter of statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSX COMPIL,4</td>
<td></td>
</tr>
<tr>
<td>NZI EQU LFG</td>
<td>Did an equal sign occur</td>
</tr>
<tr>
<td>ERROR 60,SKEND</td>
<td>NO = SIGN. THEREFORE NOT A SUBSTITUTION</td>
</tr>
<tr>
<td>REM</td>
<td>STATEMENT</td>
</tr>
<tr>
<td>TSX PUTCL2,4</td>
<td>RESTORE CELL 2</td>
</tr>
<tr>
<td>TRA SKEND</td>
<td></td>
</tr>
</tbody>
</table>
THE ARITHMETIC STATEMENT COMPILER

* COMPILE SUBROUTINE FOR ARITHMETIC SEQUENCES
* ENTRIES... COMPIL (ARITHMETIC STATEMENTS)
* CMPLIF (IF STATEMENTS)
* CMPLCL (CALL STATEMENTS)

REM

THIS ROUTINE IS THE HEART OF ARITHMETIC STATEMENT PROCESSING.
IT IS ENTERED FROM THE SUBSTITUTION STATEMENT PROCESSOR, THE
IF-STATEMENT PROCESSOR, AND THE CALL STATEMENT PROCESSOR.

ELEMENTS OF THE STATEMENT (OPERATORS AND OPERANDS) ARE SCANNED
OFF BY THE SCAN ROUTINE AND PLACED INTO A STACK (THE ARITHMETIC
STACK, ASTACK) ACCORDING TO CERTAIN RULES OF OPERATOR
HEIRARCHY.  SEE THE DESCRIPTION OF THE OPERATOR RANK TABLE
FOR THE ORDER AND HEIRARCHY OF OPERATORS.

THE GENERATION OF OBJECT PROGRAM CODE IS DELAYED AS LONG
AS POSSIBLE; I.E. THE STACK IS BUILT UNTIL AN OPERATOR IS
ENCOUNTERED WHICH CANNOT BE PLACED IN THE STACK BECAUSE OF THE
OPERATOR HEIRARCHY RULES.  CODE IS THEN PRODUCED, LAST-ENTERED
ITEMS FIRST, AND THE STACK SHORTENED UNTIL A POINT IS REACHED
THAT ALLOWS THE WAITING OPERATOR TO BE ENTERED INTO THE STACK.
STACK BUILDING COMMENCES AS BEFORE.  WHEN THE
END-OF-STATEMENT IS REACHED, CODE IS PRODUCED UNTIL THE
STACK IS CLEARED.

THE FORM OF AN ASTACK ENTRY IS AS FOLLOWS...  FOR OPERATORS,
THE OPERATOR KEY (FROM OPERATOR RANK TABLE) IS USED.  FOR
SYMBOLIC OPERANDS, THE EQUIV WORD IS USED, WITH THE SUBROUTINE
ARGUMENT BITS (BITS 15-20) CLEARED OUT, AND WITH THE EQUIV
POINTER IN THE ADDRESS FIELD.  CONSTANT OPERANDS, AND TEMPORARY
OPERANDS PRODUCED AT INTERMEDIATE STAGES IN THE CODE GENERATION,
HAVE ONLY THE CORRECT MODE BITS (PREFIX) SET ON.

OPERATORS ARE DISTINGUISHED FROM OPERANDS BY BIT 16...
BIT 16 = 0 OPERAND
BIT 16 = 1 OPERATOR

CONSTANT OPERANDS ARE DISTINGUISHED FROM OTHERS BY BIT 17...
BIT 17 = 0 VARIABLE OR INTERMEDIATE RESULT
BIT 17 = 1 CONSTANT

AN INTERMEDIATE RESULT HELD IN WORKING STORAGE WILL HAVE THE
HIGH-ORDER BIT OF THE ADDRESS (BIT 21) ON, AND THE REMAINDER
OF THE ADDRESS WILL BE THE WORKING CELL POINTER.

A SUBSCRIPTED VARIABLE WILL HAVE A NON-ZERO TAG FIELD, AND
THE ADDRESS WILL POINT TO THE SUBSCRIPT STACK (ISTACK) ENTRY.
BIT 15 IS USED IN PROCESSING OF ARITHMETIC STATEMENT FUNCTIONS
TO INDICATE A DUMMY ARGUMENT.

SPACE

CMPLCL CAL =1 ENTRY FOR CALL STATEMENT
SLW CALLCFG SET SUBPROGRAM LEVEL TO 1
IRA **2

CMPLIF STZ CALLCFG ENTRY FOR IF STATEMENT. ZERO CALL FLAG
STL EXITFG ON FOR CALL AND IF STATEMENTS
IRA **3
ENTRY FOR ARITHMETIC STATEMENTS

MODFLG, 4  GET MODE OF THIS STATEMENT

GENDES, 4  GET DESCRIPTOR FOR BINARY SEARCH IN GEN

GEN98     SET ADDRESS OF 'EQUAL COMPARE' INSTRUCTION

BAND      ORIGIN OF TABLE FOR THIS STATEMENT

BA        LENGTH OF TABLE

GENBIN, 4  GET POWER-OF-TWO LENGTH OF TABLE

BPOWER+1  AND THE RAISE/LOWER TABLE POINTER

18       AND PUT INTO THE BINSER INSTRUCTIONS.

MODFLG    GET MODE

1       GET A POSSIBLE DP OR CA BIT INDICATOR

DBLMOD    SET THE DBLMOD FLAG

EQLFLG    SET THE EQLFLG

REGIST    SET THE REGIST

REGIST+1  SET THE REGIST+1

REGFLG    SET THE REGFLG

PCNT      SET THE PCNT

ARITFN    SET THE ARITFN

ARFLAG    SET THE ARFLAG

SUBFLG    SET THE SUBFLG

CXFLAG    SET THE CXFLAG

SAVL2     SET THE SAVL2

SETUP2    SET THE SETUP2

HFLAG     SET THE HFLAG

POOL-ISTKFG, 4  INITIALIZE THE ISTACK DIRECTOR ADDRESSES

ISTKFG, 4  PUT INTO ADDRESS OF FLAG WORD

ISTKFG, 4  PUT INTO ADDRESS OF FLAG WORD

-1, 1    PUT INTO ADDRESS OF FLAG WORD

TSX       SCAN, 4  TO SCANNER TO GET NEXT ITEM

EOEPRO    END OF STATEMENT ENCASED.

STACK     CONSTANT ENCOUNTERED. EQUIVALENT

WILL BE IN LAC.

CMP1     OPERATOR ENCOUNTERED. EQUIVALENT WILL BE

IN LAC.

FXI      OPERAND (SYMBOLIC) ENCOUNTERED. EQUIV.

STACK     OPERAND (SYMBOLIC) ENCOUNTERED. EQUIV.

LSTK     STACK ITEM IN ARITHMETIC STACK

GETNXT, 1, LASTAK  GO FOR MORE, UNLESS STACK OVERFLOWS

MSG100, 5, KSEND  ARITHMETIC STACK OVERFLOW

6        OPERATOR PROCESSOR

STA      SAVE OP EQUIV TEMPORARILY AND GO TO

SYMTS    OPERATOR PROCESSOR.

**       OPERATOR PROCESSOR.
* PLUS PROCESSOR
  * PLUSP LDI ASTACK,1 IS THE TOS AN OPERATOR.
  LFI BOPRX
  FKA GETNXT PLUS IS UNARY. WE IGNORE IT.
  REM
  ADIE1 CAL SYMST GET OPERATOR LEVEL OF OPERATOR TO BE ADDED
  LDQ ADDXXX GET INSTRUCTION TO BE INSERTED
  ANA 03760000000
  SLW CMP1 ADDXXX+1 SET 'EQUAL CONDITION' INSTRUCTION
  STQ ADDXXX+1
  ADIST2 CAL ASTACK+1,1 GET PREVIOUS OPERATOR KEY
  ANA 03760000000
  LAS CMP1 ADDXXX TRA (PROC3 LEVELS OF OPS PERMIT STACKING
  REM OF NEW OPERAND
  ** ** INSTRUCTION IS INSERTED HERE.
  TSX GEN+4 CANNOT ADD. GENERATE CODE AND RECYCLE FOR
  TRA ADIST2 ANOTHER TEST.
  *
  * MUST GENERATE CODE IF OPERATOR LEVEL OF OP TO BE ADDED IS GREATER
  * THAN THAT OF TOP OPERATOR IN STACK, OR CONVERSELY, IF TOP
  * OPERATOR OF STACK HAS A LOWER LEVEL THAN OPERATOR TO BE ADDED.
  * THE ACTION TAKEN FOR EQUAL OPERATOR LEVELS VARIES FROM OP TO OP.
  * THE PERTINENT TRANSFER INSTRUCTION IS SET FOR EACH TYPE OF OP.
  SPACE 6 MINUS PROCESSOR
  *
  MINUSP LDI ASTACK,1 MUST DECIDED IF MINUS IS BINARY OR UNARY.
  LNT BOPRX IS TOS AN OPERATOR
  TXL **2,1,-1 BINARY MINUS UNLESS STACK IS EMPTY
  TXI STACK+1,1,-1 UNARY MINUS. STACK IT.
  REM
  REM CHANGE MINUS KEY
  REM SCANNER ALWAYS RETURNS THE UNARY MINUS
  REM EQUIVALENT. MUST BE REPLACED BY BINARY
  REM MINUS EQUIVALENT (NEGKEY) IF MINUS IS
  REM BINARY.
  CAL NEGKEY
  SLW SYMST
  MINUS1 LDI MINXXX GET INSTRUCTION TO BE INSERTED
  TRA ADTEST+2
  REM
  MINXXX TSX STKPR,4
  SPACE 6 = SIGN PROCESSOR
  *
  EQUALP NZT EQUALFG HAS EQUAL SIGN APPEARED ALREADY.
  ZET EXITFG NO. IS THIS AN IF OR CALL STATEMENT.
  ERROR 45,SKEND YES
  TXH 0THRUP+1,1,-2 ERROR IF STACK IS EMPTY
  REM
  STL EQUALFG TURN ON EQUAL SIGN FLAG
  REM
  LDI ASTACK,1 GET SYMBOL
  LFI BOPRX+B CNST CANNOT BE OPERATOR OR CONSTANT
  ERROR 84,SKEND
  REM
  LUX SCANSB,4 GET EQUIV ADDRESS FOR PREVIOUS SYMBOL
| LD | EQUIV,+ | GET EQUIV WORD | FT057040 |
| SL | BLHSX  | TURN ON THE DEFINITION FLAG FOR SYMBOL | FT057050 |
| ST | EQUIV,+ | RESTORE | FT057060 |
| TR | ADTEST+1 | | FT057070 |
|    | SPACE 6 | | FT057080 |
|    | * | DIVIDE PROCESSOR | FT057090 |
|    | * | TREATMENT IS SAME AS OTHER ARITHMETIC OPERATORS, | FT057100 |
|    | * | EXCEPT THAT TWO CONSECUTIVE DIVIDE OPERATORS CANNOT | FT057110 |
|    | * | BE ALLOWED INTO THE STACK (THIS RESULTS IN INCORRECT | FT057120 |
|    | * | COMPILATION). | FT057130 |
| REM | DIVIDP TXH | OTHROP+1,1,-2 | FT057140 |
|    | TRA | MINUS1 | FT057150 |
|    | SPACE 6 | | FT057160 |
|    | * | MULTIPLY PROCESSOR | FT057170 |
|    | OTHROP TXL | ADTEST+1,1,-2 | FT057180 |
|    | ERROR | STACK NOT EMPTY | FT057190 |
|    | REM | EXPRESSION MAY NOT START WITH ONE OF THESE OPERATORS | FT057200 |
|    | SPACE 6 | | FT057210 |
|    | * | EXPONENTIAl PROCESSOR | FT057220 |
|    | EXPONP TXH | OTHROP+1,1,-2 | FT057230 |
|    | * | MAY NOT BEGIN AN EXPRESSION | FT057240 |
|    | LDQ | EXPXXX | FT057250 |
|    | TRA | ADTEST+2 | FT057260 |
|    | EXPXXX ERROR | 74,SKEND | FT057270 |
|    | * | CONSECUTIVE ** OPS ARE ILLEGAL | FT057280 |
|    |    | | FT057290 |
|    |    | | FT057300 |
|    |    | | FT057310 |
PAREN PROCESSING...CURRENT PAREN COUNT IS KEPT IN CELL PCT2. THE MEANING OF A LEFT PAREN IS DETERMINED BY THE ITEM IMMEDIATELY PRECEDING IT IN THE ARITHMETIC STACK.

THERE ARE ESSENTIALLY 8 TYPES OF PARENS...

- TYPE 0 - NESTING PAREN ONLY
- TYPE 1 - EXTERNAL LIBRARY FUNCTION PAREN
- TYPE 2 - OPEN (INTERNAL) FUNCTION PAREN
- TYPE 3 - ARITHMETIC STATEMENT FUNCTION PAREN
- TYPE 4 - EXTERNAL FORTRAN FUNCTION PAREN
- TYPE 5 - CALL STATEMENT PAREN
- TYPE 6 - IF STATEMENT PAREN
- TYPE 7 - SUBSCRIPT PAREN (DOES NOT GET INTO ASTACK)

THE TYPE THAT A LEFT PAREN IS WILL BE FOUND IN THE PAREN STACK (PSTACK), AS FOLLOWS...

- PZE ASTACK POINTER OF ITEM BEFORE PAREN,,TYPE
- PCNT Serves also as the pointer for PSTACK.
- PNFLAG Holds the same as the current PSTACK entry

IN ADDITION THE PAREN TYPE IS ALSO ENTERED IN THE ADDRESS FIELD OF THE ASTACK EQUIVALENT OF THE PAREN.

LEFT PAREN PROCESSING

LPARNS

LDI ASTACK,1 GET CURRENT STACK ENTRY
LFT BOPRX BIT ON DUE TO OPERATOR, OR DUE TO THE
TRA *+3 ALL-BIT ASTACK+1 WORD (ENCOUNTERED FOR
REM IF AND CALL STATEMENTS)
LFT BARRY IS IT AN ARRAY
TRA (PROC4 YES. PROCESS SUBSCRIPT.
REM
LXI PCNT,4 REMEMBER LOCATION IN ASTACK OF
TXI *+1,4,1 ITEM BEFORE LEFT PAREN FOR LATER CLASSIFICATION.
SXA PCNT,4
STZ PSTACK,4
TXH (PROC5,1,-2 IF STACK IS EMPTY, IT MUST BE IF OR CALL
PX 1)
STA PSTACK,4 SET ASTACK POINTER IN PSTACK ADDRESS FOR
REM ALL BUT 'IF' (UNNECESSARY) AND 'CALL'
REM (ALREADY PROPERLY SET TO ZERO).
REM
NOW CLASSIFY TYPE OF PAREN ACCORDING TO NATURE OF TOP OF
STACK

AXT 0,4 SET THE PAREN TYPE FLAG INDICATOR
LFT BOPRX AN OPERATOR, SO PAREN IS MERELY NEXTING
TRA (PROC1 SPACE 6 ITEM PRECEDING PAREN IS AN OPERAND OF SOME
REM KIND. MAY BE EXTERNAL FUNCTION (LIBRARY OR
REM OTHERWISE), AN ARITHMETIC STATEMENT (INTER-
REM FUNCTION, AN ARRAY, OR AN OPEN INTERNA
REM L FUNCTION.
REM
REM LFT BINTF IS IT AN INTERNAL FUNCTION (OPEN OR
REM ARITHMETIC STATEMENT.)
TRA (PROC2 YES
REM SPACE 6
ERROR 7: SKEND
REM 5:4 PAREN TYPE FIVE FOR CALL STATEMENT
NZT CALLGF ON FOR CALL STATEMENT
TXI (PROC1, 1, 1) PAREN TYPE 6 FOR IF STATEMENT
REM (PROC1, 1, 1) SUBROUTINE NAME WAS SNUG INTO ASTACK
REM BY ICALLX ROUTINE. NOW ADJUST THE
REM ASTACK POINTER.
SPACE 6
COMMA PROCESSOR
*
* THIS SECTION DECIDES IF COMMA IS ...
* A) AT PAREN LEVEL ZERO IN WHICH CASE MUST BE END OF
* EXPRESSION. VALID FOR I/O LISTS ONLY.
* B) A POSSIBLE COMPLEX CONSTANT IF PAREN BEFORE THIS
* COMMA IS MERELY NEXING.
* C) END OF A SUBPROGRAM ARGUMENT IF PAREN BEFORE
* THIS COMMA IS OF TYPE 1, 2, 3, 4, OR 5.
COMMAP NZT PCNT
TRA ADTEST+1 I/O LIST. TREAT AS END OF EXPRESSION.
REM
LXD PNFLAG, 4 GET TYPE OF PREVIOUS PAREN
TXH ADTEST+1, 4, 0 STACK IF PAREN NOT NESTING
REM STL CFLAG POSSIBLE COMPLEX CONSTANT.
TRA (PROC3 SET THE FLAG AND STACK THE COMMA
SPACE 6
CLOSE PARENT PROCESSING
*
CLOSE PARENS OF THE FOLLOWING TYPES...
1. AT PAREN LEVEL ZERO. ERROR
2. AS MATCHING PAREN FOR NESTING LEFT PAREN, IF CFLAG IS F058850
ON, EXPRESSION MUST BE COMPLEX CONSTANT. OTHERWISE, CLOSE F058850
PAREN SIMPLY FORCES GENERATION OF CODE.
3. INDICATE END OF CALLING SEQUENCE FOR INTERNAL AND
EXTERNAL FUNCTIONS, AND SUBROUTINE CALLS.
4. INDICATES END OF ARITHMETIC EXPRESSION IN
IF STATEMENT.
RPARNP ZET CFLAG HAS A POSSIBLE COMPLEX CONSTANT SITUATION
TRA CXXCH YES. CHECK IF TOP OF STACK IS A COMPLEX
REM CONSTANT OR IF IT IS AN ERROR
NZT PCNT
ERROR 91, 8KEND TOO MANY RIGHT PARENS
REM REM NOW GENERATE CODE UNTIL FORM (A OR ,A IS
REM REACHED.
TSX STKPR, 4 DUMP ASTACK
RPNI CAL ASTACK+1, 1
LAS LPNKEY
TRA ++2
TRA RPNI
LAS CMAKEY
TRA ++2
TRA RPNI
TSX GEN+, 4 GENERATE
TRA RPNI
REM
LDC  PNFLAG,4
TRA*  *+1,4
TRA  )PROC0  0 - MATCHING PAREN MERELY NESTING
TRA  )PROC1  1 - EXTERNAL LIBRARY FUNCTION
TRA  )PROC2  2 - OPEN (INTERNAL) FUNCTION
TRA  )PROC3  3 - ARITHMETIC STATEMENT FUNCTION
TRA  )PROC4  4 - EXTERNAL FORTRAN FUNCTION
TRA  )PROC5  5 - CALL STATEMENT
TRA  )PROC6  6 - IF STATEMENT
SPACE 6

REM  NESTING PAREN
REM
)PROC0  CAL  ASTACK,1
SLW  ASTACK+1,1
PXA  *1
TXI  *+1,1,-1
ERA  REGIST+1
TNZ  )2,1
SXA  REGIST+1,1
TRA  )2,1
SPACE 6

*  LIBRARY FUNCTION PROCESSING

)PROC1  TXI  *+1,1,-1
SXD  )1,1,1
SXD  )1,4,1
SXD  )1,6,1
SXD  )1,11,1
STZ  ARFLAG
LXA  PNFLAG,1
FVI  *+1,1,2
REM  DBLMD
TRA  )1,19
CAL  REGIST
LBT  )1,19
TRA  )1,19
REM
CAL  ASTACK,1
ERA  REGIST+1
TZE  )1,1

11.19  TSX  STORE,4
TSX  )1,22,4
REM
CAL  MODFLG
LAS  =1
TRA  )1,18
TRA  )1,8
CAL  CLA,A
REM
TXH  )1,2,1,**
TXI  *+1,1,2
TXH  )1,3,1,**
AXT  TOPCOR-1,4
SXA  STOCEL,4
REM
TXI  *+1,1,2
TSX  )1,22,4
CODE  LDQ,A
REM
CAL  STOCEL

FT059120  FT059130
FT059140  FT059150
FT059160  FT059170
FT059180  FT059190
FT059200  FT059210
FT059220  FT059230
FT059240  FT059250
FT059260  FT059270
FT059280  FT059290
FT059300  FT059310
FT059320  FT059330
FT059340  FT059350
FT059360  FT059370
FT059380  FT059390
FT059400  FT059410
FT059420  FT059430
FT059440  FT059450
FT059460  FT059470
FT059480  FT059490
FT059500  FT059510
FT059520  FT059530
FT059540  FT059550
FT059560  FT059570
FT059580  FT059590
FT059600  FT059610
FT059620  FT059630
FT059640  FT059650
FT059660  FT059670
FT059680  FT059690
FT059700  FT059710
11.6
CODE
STQ,N
TLX
11.5,1,**
STORE OFF THIS ARGUMENT
RETURN IF MORE ARGUMENTS

11.3
REM
LXA
PNFLAG,1
GET ASSTACK POINTER TO TOP OF FUNCTION STACK
FT059770
TXI
*1,1,4
BUMP TO THE MQ ARGUMENT
FT059780
TSX
11.22,4
GO LOAD UP THE CODE ROUTINE CELLS A,A+1
CODE
LDU,A
RE
LXA
PNFLAG,1
CAL
ASTACK,1
LDI
ASTACK,1
ANA
=077777
SAVE ONLY EQUIV POINTER
LFI
BARGI
IS IT A SUBPROGRAM PARAMETER.
TRA
11.7
YES.
RE
CODE
TSX*,T,XRFLAG
TRA
11.6+2
CODE IS... TSX FUNCTION,4
PUT OUT RESULT
REM
TSX
PROLOG,4
MAKE PROLOG ENTRY
CODEC
TSX4,N,XRFLAG
TRA
15.6+2
CODE IS... TSX **,4
REM
CAL,A
TRA
11.1
LOGICAL STATEMENT. CODE IS... CAL A
SPACE
2
11.18
TSX
PUTCL2,4
RESTORE CELL 2
CAL
=0100000
SLW
STOCEL
TRA
11.10
INITIALIZE HIGH-CORE POINTER
REM
TSX
11.22,4
GO LOAD UP THE CODE ROUTINE CELLS A, A+1
REM
LFI
/MINTG*700000
TRA
11.12
REAL (DP OR CA) VARIABLE
CODE
CLA,A
INTEGER
CAL
STOCEL
SUB
=1
GET POINTER
LOWER BY 1
CODE
STQ,N
CAL
STOCEL
SUB
=2
LOWER BY 2
SLW
STOCEL
LFI
/MINTG*700000
TRA
11.14
NOT AN INTEGER
CODE
ST2,N
ZERO THE OTHER HIGH-CORE LOCATION
TXH
11.2,1,**
QUIT WHEN ALL ARGUMENTS DONE
TI
11.10,1,2
BUMP ASSTACK POINTER AND RETURN FOR MORE
REM
CAL
PCOUNT
FT05920
FT059730
FT059740
FT059750
FT059760
FT059770
FT059780
FT059790
FT059800
FT059810
FT059820
FT059830
FT059840
FT059850
FT059860
FT059870
FT059880
FT059890
FT059900
FT059910
FT059920
FT059930
FT059940
FT059950
FT059960
FT059970
FT059980
FT059990
FT060000
FT060010
FT060020
FT060030
FT060040
FT060050
FT060060
FT060070
FT060080
FT060090
FT060100
FT060110
FT060120
FT060130
FT060140
FT060150
FT060160
FT060170
FT060180
FT060190
FT060200
FT060210
FT060220
FT060230
FT060240
FT060250
FT060260
FT060270
FT060280
FT060290
FT060300
FT060310
TSX CITBD,4,P
TRA 11.21

11.22 LD1 ASTACK,1
GET THE FIRST ARGUMENT
LFT BEXTB+BINF
ARGUMENT CANNOT BE SUBPROGRAM NAME
ERROR 76,SKEND
STI A
SET WORDS FOR CODE ROUTINE
SXA A+1,1
TRA 1,4
SPACE 6

PROC2
TSX OPNGEN,4
PUT OUT THE CODE FOR BUILT-IN FUNCTION
LXA PCNT,4
REMOVE PAREN FROM STACK
TXI **1,4,-1
LOWER PCNT
SXA PCNT,4
CAL PSTACK,4
RESET PNFLAG TO CURRENT PAREN TYPE
SLW PNFLAG
TRA GETNXT
EXIT AND CONTINUE THE SCAN
SPACE 6

* ARITHMETIC STATEMENT FUNCTION PROCESSING
REM
* THE CALLING SEQUENCE FOR ARITHMETIC STATEMENT FUNCTIONS IS
* THE SAME AS FOR FORTRAN FUNCTIONS, EXCEPT THAT ANY DOUBLE-LENGTH
* ARGUMENTS WILL HAVE TWO ENTRIES IN THE CALLING SEQUENCE.
* THE SECOND ENTRY HAS THE ADDRESS OF THE SECOND (LOW-ORDER)
* PART OF THE ARGUMENT.
REM

PROC3
STL ARFLAG
FLAG THAT THIS IS INTERNAL FUNCTION CALL
LXA PNFLAG,4
GET START OF THIS FUNCTION STRING
CAL ASTACK,4
AND GO GET THE FUNCTION KEY FROM STACK
PAX ,4
EQUIV POINTER
CAL EQUIV,4
EQUIV WORD
PAX ,4
ARITH. STATEMENT FUNCTION STACK POINTER
TXI **1,4,1
BUMP TO SECOND WORD
SXD 15,35,4
SET DECREMENTS FOR LATER TESTS
CAL POOL,4
MODE IS IN ADDRESS OF SECOND STACK WORD
ERA MODFLG
COMPARE WITH THIS STATEMENT'S MODE
ANA =077777
TZE 15,30
SAME M nODES. OKAY
ERROR 35,SKEND
INCONSISTENT M nODES
SPACE 6

* CALL AND FORTRAN FUNCTION PAREN PROCESSING
REM

PROC4
EQU *
ENTRY FOR FORTRAN FUNCTION PAREN

PROC5
EQU *
ENTRY FOR CALL STATEMENT PAREN
REM
LXA CALLFG,4
REDUCE THE SUBPROGRAM COUNT BY 1
TXI **1,4,-1
SXA CALLFG,4
STZ ARFLAG
NOT AN ARITH. STATEMENT FUNCTION
REM

15.30 TSX STORE,4
STORE OFF ANY REGISTER RESULT
NZT ARFLAG
SKIP IF THIS IS A CALL TO AN INTERNAL FN...
TSX PUTCL2,4
RESTORE CELL 2
REM
SXD 15,1,1
SET END OF STACK IN THE TESTING CELLS
SXD 15,4,1
STZ ARGCNT
ZERO THE NUMBER OF ARGUMENTS
STZ SCPELL
ZERO THE SPECIAL PCOUNT INDICATOR
REM
LXA PNFLAG,1
BEGINNING OF SUBPROGRAM STRING IN ASTACK
TXI +1,1,2 BUMP TO FIRST ARGUMENT
TXH 15,2,1,** QUIT IF OUT OF ARGUMENTS
REM
LDI ASTACK,1 GET ARGUMENT
TSX ARGMP,4 BUMP ARGUMENT COUNTER BY 1
REM
LNT BCNST SKIP IF THE ARGUMENT IS A CONSTANT
RFT 40000 OR IN WORKING STORAGE
TXI 15.13,1,2
REM
PIA GET EQUIV POINTER OR ISTACK POINTER
PAX +4
RFT 700000 IS VARIABLE SUBSCRIPTED.
CAL POOL-1,4 YES, GET EQUIV POINTER
PAX +4
CAL LHSWD GET A DECREMENT BLHSX BIT
NZT ARFLAG SKIP IF THIS IS A CALL TO AN INTERNAL FN.
ORS EQUIV,4 PUT INTO THE EQUIV WORD
REM
RFT 700000 IS ITEM SUBSCRIPTED.
TRA 15.17 YES
LNT BARIT IS THIS ARG FROM AN INTERNAL FUNCTION...
TXI 15.13,1,2 NO
REM
PIA
ANA =077777 ARGUMENT IS ALSO AN ARGUMENT IN THIS
CODEO CAL4,N CODE IS... CAL ARG,N,4
ZET SPCELL HAS A SPECIAL CELL ALREADY BEEN OBTAINED.
TRA ++3
TSX NEXTP,4
SLW SPCELL
REM
CAL ARGCNT GET ARGUMENT NUMBER IN THIS FUNCTION
TSX CITBLN,4,Q PUT OUT AS A CONSTANT ADDEND
CODEO STA,SPCELL,R CODE IS... STA ARGCNT+SPCELL
REM
ZET DBLMOD CHECK IF THIS IS A DOUBLE-LENGTH ARGUMENT
NZT ARFLAG AND IF THIS IS ALSO AN ARITH. STATE. FN.
TXI 15.11,1,2 NO
REM
LNT MREAL YES, IT IS, IS THIS A REAL ARGUMENT.
TXI 15.11,1,2 NO
REM
PIA
ACL =1 BUMP ARG NO BY 1
ANA =077777 SAVE ONLY EQUIV POINTER
CODEO CAL4,N CODE IS... CAL ARG+1,4
REM
TSX ARGMP,4 BUMP ARGUMENT COUNTER BY 1
TSX CITBLN,4,Q PUT OUT AS CONSTANT ADDEND
CODEO STA,SPCELL,R CODE IS... STA ARGCNT+1+SPCELL
TXI 15.11,1,2 RETURN FOR ANOTHER ARGUMENT
15.17 PIA GET THE ISTACK POINTER
PAX +4
CAL POOL,4 IF EITHER POOL WORD HAS A MINUS SIGN,
ORA POOL-2,4 THEN THE SUBSCRIPT INVOLVES A VARIABLE.
PBT
TXI 15.13,1,2 NO MINUS SIGNS. IGNORE
REM
15.11    STI    A
       SXA    A+1,1
       CODE    PXA4,A
       REM    SUBSCRIPT
       CAL    PCOUNT
       SUB    =1
       CODEO    SUB,P
       REM    SUBSCRIPT INVOLVING VARIABLE
       TSX    NEXILP,4
       SLW    SPCELL
       REM    SPECIAL CELL ALREADY BEEN OBTAINED.
       TRA    15.11
       REM    GET A POOL POINTER (RELATIVE TO EQUIV)
       REM    SAVE THE POINTER. THIS CELL IS USED TO
       REM    RETAIN THE DEFINITION OF THE
       REM    BEGINNING OF THE CALLING SEQUENCE. IT
       REM    REFERS TO EQUIV AS ITS BASE, AND NEEDS
       REM    THE CIT FLAG 'R'.

15.12    LNI    MREAL
       TXI    15.11,2
       LXA    A,4
       CAL    POOL-1,4
       PAX   ,4
       CAL    EQUIV,4
       CAL    POOL-1,4
       ANA   =07777
       CODEO    NOP,N
       REM    PUT OUT THE ADDITIVE CONSTANT
       REM    STA,SPCELL,R
       REM    STA SPECIAL,CELL+ARGCNT
       TSX    CITBLN,4,Q
       CODEO    STA,SPCELL,R
       REM    YES. IS THIS ARGUMENT REAL.
       REM    NO
       REM    GET ISTACK POINTER
       REM    2ND ISTACK WORD
       REM    GET EQUIV POINTER
       REM    EQUIV WORD
       REM    GET DIMTL POINTER
       REM    GET LENGTH OF THIS ARRAY
       REM    SAVE EQUIV POINTER
       REM    CODE IS... NOP LENGTH.OF.ARRAY
       CAL    PCOUNT
       SUB    =1
       CODEO    ADD,P
       REM    RETURN FOR ANOTHER ARGUMENT
       TSX    ARGBMP,4
       CODEO    STA,SPCELL,R
       TXI    15.11,2
       ZET    DBLMOD
       NZT    ARFLAG
       TRA    15.11
       LNI    MREAL
       TRA    15.11
       TSX    ARGBMP,4
       TRA    15.11
       REM    SECOND PASS. PREPARE THE CALLING SEQUENCE
       REM    IS THIS AN ARITH. STATE. FN. DEFINITION.
       REM    NO
REM ARGCNT YES, SAVE XR4
ADD =1
TSX CITBLN, 4, Q PUT OUT ADDEND
CODECO SXA4, SCPEL, R CGDE IS... SXA END OF LIST + 1, 4
REM)
LXA SPCELL, 4
TXL 15.9, 4, 0 TRA IF NO PRELIMINARY CODE WAS PRODUCED
REM
CAL PCOUNT DEFINE THE SPECIAL CELL AS THE BEGINNING
STA EQUV, 4 OF THE CALLING SEQUENCE.
REM
LXA PNFLAG, 1 GET ASTACK POINTER
CAL ASTACK, 1
STZ ARGCNT RESET THE ARGUMENT COUNTER
ZET ARFLAG IS THIS A CALL TO AN INTERNAL FUNCTION.
TRA 15.14 YES
REM
LDI ASTACK, 1 GET FLAGS
LFT BARGT IS IT A SUBROUTINE PARAMETER.
TRA 15.8 YES
REM
ANA =077777 CODEO TSX4, T, XRFLAG CODE IS... TSX SUBPROGRAM, 4
REM TXI **1.1, 2 BUMP ASTACK POINTER
TXH 15.0, 1, ** QUIT WHEN ALL ARGUMENTS DONE
LDI ASTACK, 1
STI A
REM
TSX ARGCMP, 4 BUMP ARGUMENT COUNTER BY 1
REM
NZT ARFLAG CONTINUE TO PROCESS THE ARGUMENTS IF THIS
TRA 15.38 IS NOT AN ARITH. FN. CALL
REM
LXA ARGCNT, 4 GET NUMBER OF THIS ARGUMENT
TXI **1.4, ** BUMP BY ORIGIN OF THE STACK
P/A POOL, 4 COMPARE THIS MODE WITH DEFINED MODE
ARS 33
TZE **2 OKAY
ERROR 36, SKEND DIFFERENT MODES, ERROR
REM
RFT 700000 IS ARGUMENT SUBSCRIPTED.
TRA 15.5 YES, CODE MAY HAVE ALREADY BEEN INSERTED
REM
LFT BARIIT IS THIS ARG OF AN INTERNAL FUNCTION.
TRA 15.20 YES
REM
SXA A+1.1
ZET DBLIND IS THIS A DOUBLE-LENGTH STATEMENT,
NZT ARFLAG AND ARE WE CALLING AN ARITH. STATE. FN...
TRA 15.19 NO
REM
LNT MREAL YES, IS THIS A REAL ARGUMENT.
TRA 15.19 NO
REM
CAL TSX YES, PREPARE TO PUT OUT BOTH TSX INSTRUCTIONS
SLW C0D31 2ND INSTRUCTION FOR THE CODE ROUTINE
STL SPCDFG SIGNAL CODE ROUTINE FOR 2 INSTRUCTIONS
<table>
<thead>
<tr>
<th>Label</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSX</td>
<td>+1,4</td>
<td>Go put out the code</td>
</tr>
<tr>
<td>PZE</td>
<td>TSX,+A</td>
<td></td>
</tr>
<tr>
<td>TSX</td>
<td>ARGMP,+4</td>
<td>Bump argument counter by 1</td>
</tr>
<tr>
<td>TXI</td>
<td>+4,1,2</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**15.19**
CODE | TSX,+A | CODE IS... TSX argument |
| TXI   | +4,1,2 | |
| REM   | | |

**15.5**
LXA | +4,1 | GET ISTACK POINTER |
CAL   | POOL,+4 | IF EITHER ISTACK WORD HAS A MINUS SIGN; |
ORA   | POOL-2,+4 | THE CODE HAS ALREADY BEEN PRODUCED. |
PBI   | | |
TRA   | +5.42 | CONSTANT SUBSCRIPT. PUT OUT CODE |
REM   | | |
**15.20**
CODE | TSX | NO ARGUMENT ADDRESS. CODE IS... TSX ** |
REM   | | |
ZET   | DBLMOD | |
NZI   | ARFLAG | |
TXI   | +4,1,2 | IS IT REAL |
REM   | | NO. GO DO NEXT ARGUMENT |
TXI   | +4,1,2 | |
REM   | | |
**15.0**
LXD   | PNFLAG,+1 | GET PARENT TYPE |
TXI   | +5.714 | TRA IF CALL STATEMENT |
REM   | | |
CAL   | AX=4 | PREPARE TO POSSIBLY RESTORE XR4 IN THE CODE |
ZET   | ARITIFN | IS THIS IN AN ARITH. STATEMENT FUNCTION... |
TXS   | CIBL0,+4,N | YES. CODE IS... AX= **4 |
REM   | | |
NZI   | ARFLAG | IS THIS A CALL TO AN ARITH ST FN... |
TRA   | +5.36 | NO |
REM   | | |
**15.37**
LXA   | ARGCNT,+4 | BUMP BY ORIGIN OF STACK +1 |
TXI   | +5.142 | |
REM   | | GET POINTER TO 2ND STACK WORD |
LXD   | +4 | |
REM   | | |
ERA   | POOL,+4 | COMPARE ARGCNT+ORIGIN+1 WITH ORIGIN+LENGTH+1 |
ARS   | +4 | |
REM   | | OKAY |
TZE   | +2 | DIFFERENT NUMBER OF ARGUMENTS |
REM   | | |
**15.36**
AXC   | +12.1-1,4 | SET EXIT POINT FOR RESULT ROUTINE |
SXA   | GENXIT,+4 | |
LXA   | PNFLAG,+1 | FUNCTION PAREN. RECORD RESULT IN |
CAL   | ASTACK,+1 | ASTACK, AND CONTINUE TO PROCESS STATEMENT. |
ALS   | +2 | MOVE PERTINENT MODE BIT INTO P-BIT |
PBI   | | BIT IS ON IF REAL RESULT |
RESULT | INTEGR,AC | |
RESULT | REAL,AC | |
REM   | | |
**15.7**
SKIP  | 63 | END OF STATEMENT EXPECTED FOR CALL |
ERROR | 3,SKEND | |
TRA   | CMPLX | |
REM   | | |
**15.8**
ANA   | =077777 | SAVE EQUIV POINTER |
TXS   | PROLOG,+4 | MAKE PROLOGUE ENTRIES |
CODEC  TSX4,N,XFLAG      CODE IS... TSX **,4
TI 15.4,1,2  FT063320
REM  FT063330

15.14  PAX 4  PUT OUT TSX TO INTERNAL FUNCTION  FT063340
CAL EQUV,4  GET THE EQUIV WORD  FT063360
PAX 4  AND GET THE ARITH. STATE. FN STACK POINTER  FT063370
CAL POOL,4  DEFINITION OF FUNCTION  FT063380
ANA # 077777  FT063390

CODEO TSX4,P,XFLAG    CODE IS... TSX INTERNAL.FUNCTION,4  FT063400
TXI 15.4,1,2  FT063410
REM  FT063420

ARGSMP CAL ARGCNT  SUBROUTINE TO BUMP ARGUMENT COUNTER BY 1  FT063430
ADD =1  FT063440
SLW ARGCNT  FT063450
TRA 1,4  FT063460
REM  FT063470

LHSXWD VFD 018/BLHSX  FT063480
SPACE 6  FT063490

* TERMINATION OF IF STATEMENT COMPIL  FT063500
REM  FT063510

JPROC6 TSX PLACE,4  PUT RESULT INTO THE AC  FT063520
TSX PUTCLZ,4  RESTORE CELL 2  FT063530
LXA CMPLX,4  FT063540
TRA 1,4  RELOAD XR4 AND RETURN  FT063550
<table>
<thead>
<tr>
<th>NAME</th>
<th>CODE</th>
<th>Meaning</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>EJECT</td>
<td></td>
<td>FINAL PROCESSING FOR ARITHMETIC STATEMENT FUNCTION DEFINITIONS.</td>
<td>FT063560</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ENTERED VIA THE CODE GENERATORS AT END-OF-STATEMENT PROCESSING.</td>
<td>FT063570</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REM</td>
<td>FT063580</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASFEND</td>
<td>FT063590</td>
</tr>
<tr>
<td>ZET</td>
<td>SUBFLG</td>
<td>WAS THERE ANY SUBSCRIPTING.</td>
<td>FT063600</td>
</tr>
<tr>
<td></td>
<td>90,* +1</td>
<td>YES, NOT ALLOWED</td>
<td>FT063610</td>
</tr>
<tr>
<td>LXA</td>
<td>ARITFN,4</td>
<td>START OF THE STACK</td>
<td>FT063620</td>
</tr>
<tr>
<td>CAL</td>
<td>POOL-1,4</td>
<td>SECOND STACK WORD (HAS END OF STACK POINT)</td>
<td>FT063630</td>
</tr>
<tr>
<td>STD</td>
<td>ASFEN1</td>
<td>SET TRANSFER INSTRUCTION DECREMENT</td>
<td>FT063640</td>
</tr>
<tr>
<td>ARS</td>
<td>18</td>
<td>MOVE INTO ADDRESS</td>
<td>FT063650</td>
</tr>
<tr>
<td>SUB</td>
<td>ARITFN</td>
<td>GET LENGTH OF STACK+1</td>
<td>FT063660</td>
</tr>
<tr>
<td>ORA</td>
<td>TRA04</td>
<td>OR IN THE TRANSFER INSTRUCTION</td>
<td>FT063670</td>
</tr>
<tr>
<td>TSX</td>
<td>CITBLD,4,N</td>
<td>PUT OUT THE RETURN CODE</td>
<td>FT063680</td>
</tr>
<tr>
<td>CAL</td>
<td>PCOUNT</td>
<td>SET LARSTS TO THE CURRENT LENGTH OF ARITHMETIC STATEMENT FUNCTIONS.</td>
<td>FT063690</td>
</tr>
<tr>
<td>SLW</td>
<td>LARSTS</td>
<td>ARITHMETIC STATEMENT FUNCTIONS.</td>
<td>FT063700</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REM</td>
<td>FT063710</td>
</tr>
<tr>
<td>LXA</td>
<td>ARITFN,4</td>
<td>GET START OF STACK</td>
<td>FT063720</td>
</tr>
<tr>
<td>TXI</td>
<td>*1,4,2</td>
<td>BUMP TO START OF ARGUMENTS</td>
<td>FT063730</td>
</tr>
<tr>
<td>ASFEN2</td>
<td></td>
<td>CHANGE ALL ARGUMENTS FROM THEIR BCD</td>
<td>FT063740</td>
</tr>
<tr>
<td>TZE</td>
<td>ASFEN1-1</td>
<td>NAMES TO THEIR 'EQUIV-LIKE' MODE</td>
<td>FT063750</td>
</tr>
<tr>
<td>LDI</td>
<td>=0</td>
<td>REPRESENTATIONS.</td>
<td>FT063760</td>
</tr>
<tr>
<td>LAS</td>
<td>=H10000D</td>
<td></td>
<td>FT063770</td>
</tr>
<tr>
<td>TRA</td>
<td>*1</td>
<td></td>
<td>FT063780</td>
</tr>
<tr>
<td>LAS</td>
<td>=HNL411</td>
<td></td>
<td>FT063790</td>
</tr>
<tr>
<td>SIL</td>
<td>MREAL</td>
<td></td>
<td>FT063800</td>
</tr>
<tr>
<td>TRA</td>
<td>*2</td>
<td></td>
<td>FT063810</td>
</tr>
<tr>
<td>SIL</td>
<td>MINTG</td>
<td></td>
<td>FT063820</td>
</tr>
<tr>
<td>STI</td>
<td>POOL,4</td>
<td></td>
<td>FT063830</td>
</tr>
<tr>
<td>TXI</td>
<td>*1,4,1</td>
<td>BUMP TO NEXT ARGUMENT</td>
<td>FT063840</td>
</tr>
<tr>
<td>ASFEN1</td>
<td>TXL</td>
<td>ASFEN2,4,** CONTINUE IF MORE ARGUMENTS</td>
<td>FT063850</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REM</td>
<td>FT063860</td>
</tr>
<tr>
<td>TSX</td>
<td>CITBLD,4</td>
<td>TELL CITBLD THAT THIS FUNCTION IS OVER</td>
<td>FT063870</td>
</tr>
<tr>
<td>LXA</td>
<td>LWORKS,4</td>
<td>UPDATE THE GETWRK ROUTINE TO RESERVE AND</td>
<td>FT063880</td>
</tr>
<tr>
<td>SXD</td>
<td>GETWR1,4</td>
<td>REMEMBER ANY WORKING CELLS USED IN</td>
<td>FT063890</td>
</tr>
<tr>
<td>SXD</td>
<td>FREWR1,4</td>
<td>THIS ARITHMETIC STATEMENT FUNCTION CODE.</td>
<td>FT063900</td>
</tr>
<tr>
<td>TRA</td>
<td>SKEND</td>
<td>QUIT</td>
<td>FT063910</td>
</tr>
</tbody>
</table>
* EJECT *
** ROUTINE TO CHECK VALIDITY OF A COMPLEX CONSTANT **

REM

CAL MODFLG GET COLUMN 1 MODE
ERA =3 IS IT A COMPLEX STATEMENT.
TNZ ERROR7 NO. BAD FORM SOMEWHERE
REM
LDI ASTACK,1 GET CURRENT ASTACK WORD
LNT MREAL+BCNST BITS MUST BE ON
CAL ASTACK+1,1 GET EITHER A MINUS OR A COMMA
LAS UNEGKY IS IT MINUS.
TRA ++2 NO
TXI CXCHK1,1,-1 YES. BUMP POINTER AND GO
CAL ASTACK+2,1 GET FIRST PART KEY WORD
LNT MREAL+BCNST BITS MUST BE ON
ERROR 43,SKEND
CAL ASTACK+3,1 GET EITHER A MINUS OR THE LEFT PAREN
LAS UNEGKY IS IT MINUS.
TRA ++2 NO
TXI CXCHK4,1,-1 YES. BUMP POINTER AND GO
ERA LPNKEY MUST BE THE LEFT PAREN
INZ CXCHK2 ERROR IF NOT
CAL CX1 GET THE TWO PARTS OF THE CONSTANT
LDQ CX2
TSX ASCON2,4 GO PUT IN CONSTANT TABLE
ORA REALCT OR MODE AND BCNST TO THE CONSTANT POINTER
LXA PNFLAG,1 RESET ASTACK POINTER
TI X ++1,1,1 ADJUST ASTACK POINTER TO POINT AT PAREN
SLW ASTACK+1 PUT CONSTANT WORD INTO ASTACK
STZ CXFLAG ZERO OUT THE COMPLEX CONSTANT INDICATOR
TRA J2,1 ADJUST PSTACK AND QUIT
CAL ASTACK+2,1 SET SIGN OF SECOND PART MINUS
STO CX2
CAL ASTACK+1,1 GET THE NEXT ASTACK WORD
TRA CXCHK3
CAL ASTACK+3,1 GET THE NEXT ASTACK WORD
TRA CXCHK5
REALCT VFD 018/MREAL+BCNST
- EJECT END OF STATEMENT. PROCESS THE STACK
  
  REM
  
  EDEPRO TXL COMPLX,1,0 IS STACK EMPTY
  TSX STKPRT,4 ********** DUMP ASTACK
  REM
  
  TSX GEN,4 NO
  TXH *-1,1,0 GENERATE UNTIL ONLY ONE ITEM REMAINS
  REM
  
  CMPLX RESTOR
  TRA 1,4
* ROUTINE TO DUMP THE CONTENTS OF ASTACK ON THE LIST TAPE

REM

STKPRT IRA 1,4 ALTER THIS INSTRUCTION TO GET ASTACK

REM PRINT-OUT.

SAVE EXSTKP

TSX LIST,4

PON XYZXY,4

TXI **1,1,1

LDQ ASTACK+1,1

TSX CINSTR,4

SLW EOEPI+3

STQ EOEPI+4

TSX LIST,4

EOEPI,5

TIX *-6,1,1

EXSTKP RESTOR

TRA 1,4 RETURN

REM

EOEPI 3,

PZE

PZE

REM

XYZXY BCI 3,

BCI 1,ASTACK
* THE SCAN ROUTINE IS DIVIDED INTO SEVERAL SECTIONS... THE
* SYMBOLIC NAME SCANNER, THE NUMBER SCANNER, THE HOLLERITH
* LITERAL SCANNER, AND THE OPERATOR SCANNERS.
* ENTRY... ISA SCAN,4
* EXIT... IRA ... RETURN HERE IF END OF STATEMENT
* IRA ... RETURN HERE IF A CONSTANT
* IRA ... RETURN HERE IF OPERATOR
* ... ... RETURN HERE IF A SYMBOLIC OPERAND
* SPACE 2
* ALL CONSTANTS ENCOUNTERED WILL BE ENTERED INTO THE CONSTANT
* TABLE CONTAB, AND THE CONSTANT EQUIVALENT WILL BE RETURNED TO
* THE LOGICAL ACCUMULATOR ON EXITING FROM SCAN. FORMAT OF THE
* CONSTANT EQUIVALENT IS...
* REM
* BITS S,1-2 MODE
* BITS 3-16 NOT USED
* BIT 17 =1. INDICATES A CONSTANT. THIS BIT IS
* USED TO Flag ENTRIES IN THE ASTACK ONLY. THIS BIT WILL BE OFF FOR VARIABLES IN
* ASTACK.
* BITS 18-20 NOT USED
* BITS 21-35 POINTER ADDRESS TO CONTAB.
* SPACE 2
* FOR SYMBOLIC OPERANDS AN ENTRY WILL BE MADE IN SYMTAB IF THE
* SYMBOL IS NOT ALREADY THERE. THE MODE WILL BE SET... IF THE
* SYMBOL IS ALREADY IN SYMTAB AND THE NAME DOES NOT END IN -F, IT
* WILL BE ASSUMED TO BE CORRECT, AND THE EQUIVALENT WILL BE
* RETURNED TO THE LOGICAL ACCUMULATOR ON EXITING FROM THE SCAN.
* THE EQUIVALENT FOR SYMBOLIC OPERANDS WILL HAVE THE FORM...
* REM
* BITS S,1-14 SAME AS IN SYMTAB.
* BITS 15-20 ALL ZERO. IF THESE BITS ARE NEEDED,
* THEY MUST BE OBTAINED FROM EQUIV.
* BITS 21-35 EQUIV POINTER. ALLOWS ACTUAL SYMBOL
* EQUIVALENT TO BE OBTAINED. THIS ADDRESS
* MAY BE CHANGED TO POINT TO ISTACK IF THE
* VARIABLE IS SUBSCRIPTED.
* REM
* NOTE THAT THE ARGUMENT NUMBER FOR PARAMETERS IS NOT AVAILABLE
* IN THE ASTACK EQUIVALENT.
* SPACE 2
* OPERATORS RETURN AN EQUIVALENT ALSO TO THE LOGICAL ACCUMULATOR
* BEFORE EXITING FROM SCAN. THIS EQUIVALENT IS TAKEN FROM THE
* OPERATOR RANK TABLE (Q.V., AND ALSO THE DEFINITION OF THE
* RANK MACRO).
* REM
* THE ORDER OF THE SECTIONS OF THE SCANNER IS IMPORTANT. THEY
* MUST BE... SYMBOLIC, NUMBER, OPERATORS. THIS THEN AFFORDS
* A CONVENIENT RAPID TEST, USED IN MANY PARTS OF FASTRAN, TO
* DETERMINE IF A CHARACTER IS NUMERIC, ALPHABETIC, OR SPECIAL.
* THE TEST IS MADE BY CONVERTING ON THE SCAN TABLE, AND DETERMINING
* IF THE RESULTING LOCATION IS BELOW (LETTER), EQUAL, OR ABOVE
* (SPECIAL) THE ENTRY FOR NUMERAL (SCAN1).
SCAN | SXA | SCAX+1 | FT065380  
SCAN | SXA | SCAX+1,4 | FT065390  
SCAN | LDQ | COLUMN+2 | PICK UP CHARACTER AND CLASSIFY | FT065400  
SCAN | CRQ | SCANT,1,1 | FT065410  
SCAN | ++,1,1 | FT065420  
SCAN | ** | FT065430  
REM | | | SCAN CONVERSION TABLE | FT065440  
SCANT | DUP | 1,10 | FT065450  
SCAN1 | 00-11 NUMBERS | FT065460  
SCAN2 | 12 ERROR | FT065470  
SCAN13 | 13 = | FT065480  
SCAN2 | 14 ' (ERROR) | FT065490  
DUP | 1,9 | FT065500  
SCAN2 | 15-17 ERROR | FT065510  
SCAN20 | 20 + | FT065520  
DUP | 1,3 | FT065530  
SCAN5 | 21-31 LETTERS | FT065540  
SCAN2 | 32 ERROR | FT065550  
SCAN33 | 33 . | FT065560  
SCAN34 | 34 ) | FT065570  
DUP | 1,9 | FT065580  
SCAN2 | 35-37 ERROR | FT065590  
SCAN40 | 40 - | FT065600  
DUP | ** | FT065610  
SCAN5 | 41-51 LETTERS | FT065620  
SCAN2 | 52 ERROR | FT065630  
SCAN2 | 53 $ | FT065640  
SCAN54 | 54 * | FT065650  
DUP | 1,8 | FT065660  
SCAN2 | 55 - 57 ERROR | FT065670  
SCAN60 | 60 BLANK | FT065680  
SCAN61 | 61 / | FT065690  
DUP | 1,8 | FT065700  
SCAN5 | 62 - 71 LETTERS | FT065710  
SCAN2 | 72 ERROR | FT065720  
SCAN73 | 73 , | FT065730  
SCAN74 | 74 | FT065740  
SCAN2 | 75 ERROR | FT065750  
SCAN2 | 76 ERROR | FT065760  
SCAN77 | 77 END OF STATEMENT | FT065770  
REM | | | FT065780
<table>
<thead>
<tr>
<th>SCANX</th>
<th>AXT</th>
<th>**,1</th>
<th>FT066390</th>
</tr>
</thead>
<tbody>
<tr>
<td>AXT</td>
<td>**,4</td>
<td></td>
<td>FT066400</td>
</tr>
<tr>
<td>TRA</td>
<td>4,4</td>
<td></td>
<td>FT066410</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FT066420</td>
</tr>
<tr>
<td>SCAN5K LFI</td>
<td>BVAR8+BCOMN+BEQV</td>
<td>ALL FLAGS MUST BE OFF</td>
<td>FT066430</td>
</tr>
<tr>
<td>ERROR</td>
<td>69,SKEND</td>
<td></td>
<td>FT066440</td>
</tr>
<tr>
<td>SIL</td>
<td>BEXISTF</td>
<td>MARK IT AS AN EXTERNAL FUNCTION</td>
<td>FT066450</td>
</tr>
<tr>
<td>TRA</td>
<td>SCAN5N</td>
<td></td>
<td>FT066460</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FT066470</td>
</tr>
<tr>
<td>SCAN5S NZT</td>
<td>DBLMOD</td>
<td>IS IT DOUBLE OR COMPLEX</td>
<td>FT066480</td>
</tr>
<tr>
<td>TRA</td>
<td>SCAN5T</td>
<td>NO</td>
<td>FT066490</td>
</tr>
<tr>
<td>LFI</td>
<td>BARRY</td>
<td>IF BARRY IS ON, BDOUB MUST ALREADY BE ON</td>
<td>FT066500</td>
</tr>
<tr>
<td>TRA</td>
<td>BDOUB</td>
<td></td>
<td>FT066510</td>
</tr>
<tr>
<td>TRA</td>
<td>**,2</td>
<td></td>
<td>FT066520</td>
</tr>
<tr>
<td>ERROR</td>
<td>49,**,2</td>
<td>SINGLE ARRAY USED IN DOUBLE CONTEXT</td>
<td>FT066530</td>
</tr>
<tr>
<td>SIL</td>
<td>BDOUB</td>
<td>TURN ON THE DOUBLE LENGTH FLAG</td>
<td>FT066540</td>
</tr>
<tr>
<td>TRA</td>
<td>SCAN5T</td>
<td></td>
<td>FT066550</td>
</tr>
<tr>
<td>EJECT</td>
<td>TEST SYMBOL FOR ARITHMETIC STATEMENT FUNCTION ARGUMENT</td>
<td>FT066560</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-------------------------------------------------------</td>
<td>----------</td>
<td></td>
</tr>
<tr>
<td>SCANAE</td>
<td>SCANT1,$4 REMOVE NON-ESSENTIAL CHARACTERS</td>
<td>FT066580</td>
<td></td>
</tr>
<tr>
<td>ORA</td>
<td>SCANT2,$4 AND ADD BLANKS</td>
<td>FT066590</td>
<td></td>
</tr>
<tr>
<td>LXA</td>
<td>ARITFN,1 GET START OF STACK</td>
<td>FT066600</td>
<td></td>
</tr>
<tr>
<td>TXI</td>
<td>++1,1,1</td>
<td>FT066610</td>
<td></td>
</tr>
<tr>
<td>SXD</td>
<td>SCANA1,1 SAVE IN FUTURE INSTRUCTION</td>
<td>FT066620</td>
<td></td>
</tr>
<tr>
<td>TXI</td>
<td>++1,1,1</td>
<td>FT066630</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FT066640</td>
<td></td>
</tr>
<tr>
<td>SCANAH</td>
<td>POOL,1 TEST IF THIS SYMBOL IS AN ARGUMENT</td>
<td>FT066650</td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>++2</td>
<td>FT066660</td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>SCANAG YES. MATCHES A MEMBER OF THE STACK</td>
<td>FT066670</td>
<td></td>
</tr>
<tr>
<td>TXI</td>
<td>++1,1,1</td>
<td>FT066680</td>
<td></td>
</tr>
<tr>
<td>SCANAF</td>
<td>SCANA1,1,** RETURN FOR MORE IF NOT THROUGH THE STACK</td>
<td>FT066690</td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>SCAN5L RETURN TO MAIN SCANNER. NOT AN ARGUMENT</td>
<td>FT066700</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FT066710</td>
<td></td>
</tr>
<tr>
<td>SCANA1</td>
<td>LDI =0 DETERMINE THE MODE OF THE ARGUMENT</td>
<td>FT066720</td>
<td></td>
</tr>
<tr>
<td>LAS</td>
<td>=H100000 DETERMINE THE MODE OF THE ARGUMENT</td>
<td>FT066730</td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>++1</td>
<td>FT066740</td>
<td></td>
</tr>
<tr>
<td>LAS</td>
<td>=HNI(((1) REAL MODE + ARGUMENT BIT</td>
<td>FT066750</td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>++2</td>
<td>FT066760</td>
<td></td>
</tr>
<tr>
<td>SCANA1</td>
<td>SIL MINIT+BARIT FIXED MODE + ARGUMENT BIT</td>
<td>FT066770</td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>**2</td>
<td>FT066780</td>
<td></td>
</tr>
<tr>
<td>SIL</td>
<td>BARIT TURN ON THE ARGUMENT BIT</td>
<td>FT066790</td>
<td></td>
</tr>
<tr>
<td>STI</td>
<td>SCAN5B RESET THE SAVE CELL</td>
<td>FT066800</td>
<td></td>
</tr>
<tr>
<td>TXI</td>
<td>++1,1,1,** LOWER BY START OF STACK, THUS GETTING</td>
<td>FT066810</td>
<td></td>
</tr>
<tr>
<td>SXA</td>
<td>SCANS5B,1 THE ARGUMENT NUMBER</td>
<td>FT066820</td>
<td></td>
</tr>
<tr>
<td>CAL</td>
<td>SCANS5B</td>
<td>FT066830</td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>SCANX EXIT THE SCANNER</td>
<td>FT066840</td>
<td></td>
</tr>
</tbody>
</table>
REM FUNCTION NAME ENCOUNTERED
REM
SCAN5H  ERR015,4,7  IS NAME TOO LONG (GREATER THAN 7 CHAR.)
SXA  SCAN5Q,4  SAVE LENGTH OF SYMBOL
SLW  FORM69  SAVE BCD NAME OF SYMBOL
TXL  SCAN5M,4,6  IS IT LESS THAN SEVEN CHARACTERS.
REM
SKIP  LPCHAR  NO LEFT PAREN MUST BE NEXT
ERROR  15,SKEND  FTP066920
CAL  FORM69  FTP066940
TXI  SCAN5U,2,1  RESTORE THE POINTER AND CONTINUE
REM
SCAN5M  ANA  SCANT1,4  KILL THE NON-ESSENTIAL BITS OF SYMBOL
ORA  SCANT2,4  REPLACE BY BLANKS
SLW  FORM69  SAVE BCD NAME OF SYMBOL
REM
XCL  LOCAIE,4  FIND IN SYMTAB
STO  SCAN5B  SAVE EQUIV POINT AND ACCUM SIGN
STA  SCAN5N  SET EQUIV ADDRESS
LFT  BARRY  IS IT AN ARRAY.
ERROR  20,**+1  YES. NOT ALLOWED
REM
SKIP1  LPCHAR  IS THERE A LEFT PAREN NEXT.
TXI  SCAN5R,2,1  YES. RESTORE THE POINTER AND GO
LFT  BINF  NO PAREN. IS FUNCTION BIT ON.
ERROR  57,SKEND  YES. NOT ALLOWED WITHOUT PAREN
TRA  SCAN5C  TREAT AS REGULAR VARIABLE
REM
SCAN5R  CLA  SCAN5B  GET THE 'LOCATE' SIGN INDICATION
TPL  **+4  SKIP IF A OLD SYMBOL
PAX  +4  GET EQUIV POINTER
STZ  SYMTAB,4  ZERO OUT INDICATIONS OF -F SYMBOL
REM
CAL  FORM69  GET BCD NAME OF SYMBOL
SCAN5Q  AXT  **+4  REGAIN THE LENGTH OF SYMBOL
TXI  **+1,4,-1  PREPARE TO DELETE THE TERMINAL F
ANA  SCANT1,4  FTP067220
ORA  SCANT2,4  FTP067240
SCAN5U  SLW  FORM23+3  FTP067250
SLW  FORM69  FTP067260
NZT  DBLMOD  IS IT DOUBLE PRECISION OR COMPLEX.
IRA  SCAN5P  FTP067270
REM
LXD  SCAN5A,1  DOUBLE-LENGTH STATEMENT. MUST ADD A
TXL  SCAN5P,1,0  'D' OR 'I' TO NAME... EXCEPT FOR
XCL  SCAN5,4  FTP067290
SXA  SCAN51,4  FTP067310
TSX  LOCATE,4  FTP067320
LFT  BINF  FTP067330
TRA  SCAN5O  FTP067340
REM
PL  SCAN51  IF NOT A NEW SYMBOL, SKIP
PAX  +4  FTP067350
STZ  SYMTAB,4  FTP067360
SCAN5I  AXT  **+4  FTP067370
TXH  ERR015,4,5  FTP067390
LDI  MODFLG  FTP067400
LDQ  FORM69  FTP067410
A 'I' IS INSERTED FOR THE 'I', AND A '4'
CAL =1 FOR THE 'D'. POST-PASS1 CHANGES THESE. FT067450
RNT 1 IS THIS A COMPLEX STATEMENT. FT067460
CAL =4 NO. SET UP THE DP LEADING BIT FT067470
LGL 30 RESTORE THE NEW NAME TO THE ACCUM FT067480
REM FT067490
REM SEARCH BUILT-IN FUNCTION TABLE FT067500
SCA5P AXT LOPENT,4 LENGTH OF OPEN FUNCTION TABLE FT067510
LAS LIBMAB+LOPENT+1,4 NAME FROM TABLE FT067520
TRA **2 NOT FOUND YET FT067530
TRA ROPEN OPEN FUNCTION LOCATED FT067540
REM FT067550
TIX *-3,4,2 FT067560
REM FT067570
XCL FT067580
TSX LOCATE*,4 FT067590
SCA5O STO SCAN5B SAVE EQUIV POINTER AND ACCUM SIGN FT067600
LFT BARRY+BVARB+BCOMM+BEOVU ALL FLAGS MUST BE OFF FT067610
ERROR 69,SKEND FT067620
REM FT067630
PAX ,4 GET EQUIV POINTER FT067640
RIL 700000 RESET THE MODE BITS FT067650
CAL FORM69 GET BCD NAME OF SYMBOL FT067660
ERA =HX FT067670
ARS 30 FT067680
TZE SCAN5F FIXED MODE FT067690
SIL MREAL SET REAL MODE BITS FT067700
TRA SCAN5G FT067710
REM FT067720
SCA5F SIZ MTING INTEGR MODE BITS FT067730
ZET DBLMOD IS IT DOUBLE-LENGTH STATEMENT. FT067740
ERROR 34,**1 FT067750
REM FT067760
SCA5G LXD SCAN5A,1 IS THIS THE START OF A STATEMENT. FT067770
TCL SCANAA,1,0 IF SO, GO TO ARITH STATEMENT FN. PROCESSOR FT067780
REM FT067790
LNT BINTF IF OFF, THEN TURN ON BEXTF+BLIBF FT067800
REM FT067810
SIL BEXIF+BLIBF NO. DEFINE AS EXTERNAL LIBRARY FUNCTION FT067820
TRA SCAN5E-1 FT067830
REM FT067840
ROPEN CAL LIBTAB+LOPENT,4 GET KEY WORD FOR THIS FUNCTION FT067850
LXD SCAN5A,4 SCAN5A,4 GET key word for this function FT067860
FXH SCAX,4,0 OPERATOR (BUILT-IN) ROUTINE NAME FT067870
ERROR 88,SKEND OPEN FUNCTION CANNOT BE DEFINED BY PROGRAM FT067880
* ARITHMETIC STATEMENT FUNCTION DEFINITION.
  REM 067890
  * THE EQUIV ENTRY FOR AN ARITHMETIC STATEMENT FUNCTION HAS THE
  * ARITH. STATE. FN. STACK LOCATION IN THE ADDRESS, AND BINFN IS
  * TURNED ON. THE MODE OF THE FUNCTION IS SET IN THE PREFIX.
  REM 067910
  * THE ARGUMENTS AND OTHER INFORMATION FOR EACH FUNCTION ARE KEPT
  * IN A STACK IN POOL. THE STACK IS INITIALLY FORMED AS FOLLOWS...
  * THE FIRST WORD CONTAINS THE EQUIV ENTRY WITH THE ADDRESS ALTERED
  * TO CONTAIN THE DEFINITION OF THE FUNCTION (CURRENT VALUE OF
  * PCOUNT). THE SECOND WORD HAS THE COLUMN 1 MODE IN ITS ADDRESS,
  * AND THE POOL POINTER TO THE END OF THIS STACK IN THE DECREMENT.
  * SUBSEQUENT WORDS CONTAIN THE BCD NAMES OF THE DUMMY ARGUMENTS.
  * IF AN ARGUMENT IS REAL AND THE COLUMN 1 MODE IS '0' OR 'I', THEN
  * ONE EXTRA CELL IS SKIPPED TO AID IN LATER FORMING CALLING
  * SEQUENCES TO THIS FUNCTION.
  REM 068010
  * THE FLAG CELL ARITFN CONTAINS THE LOCATION OF THE STACK
  * IN ITS ADDRESS.
  REM 068030
  * AS THE STATEMENT IS SCANNED, THE LIST OF DUMMY ARGUMENTS IS
  * SEARCHED WHENEVER A VARIABLE NAME IS ENCOUNTERED.
  * WHEN A DUMMY NAME IS LOCATED, THE STACK ENTRY IS SET TO
  * CONTAIN A BARI BIT (BIT 15).
  * THIS WILL CAUSE THE CODE ROUTINE TO PICK UP SUCH ARGUMENTS
  * FROM THE CALLING SEQUENCE INDIRECTLY (WHENEVER POSSIBLE).
  REM 068130
  * AFTER THE STATEMENT IS PROCESSED, THE DUMMY ARGUMENT NAMES IN
  * THE STACK ARE REPLACED WITH EQUIV-LIKE MODE DESIGNATIONS.
  REM 068150
  SCANAA CLA SCAN5B GET THE RETURNED WORD FROM LOCATE ROUTINE
  TMI ++2 ERROR IF ALREADY IN SYMTAB
  ERROR 78,S,KEND
  SIL BINFN SET ARITHMETIC FUNCTION BIT ON
  STI SCAN58 RESTORE THE SAVE CELL
  CAL NXTLOC GET POOL POINTER
  STA SCAN58 SET SAVE CELL ADDRESS TO START OF STACK
  CAL SCAN58
  SLW EQUIV,4 RESTORE THE EQUIV WORD
  REM 068250
  LDI PROCWS GET THE PROCESSOR SWITCH
  RIL BXEO TURN OFF THE EXECUTABLE STATEMENT BIT
  STI PROCWS RESTORE SWITCH
  REM 068300
  LXD FIRSTF,1 DETERMINE THAT NO EXECUTABLE STATEMENT
  TXL ++2,1,0 HAS ALREADY OCCURRED.
  ERROR 89,++1
  Llx NXTLOC,1 GET POINTER TO START OF STACK
  SXA ARITFN,1 SAVE POINTER AND MARK THIS AS AN ARITH FN.
  SLW POOL,1 SET DESCRIPTOR WORD IN THE STACK
  CAL PCOUNT DEFINE THE FUNCTION, AND SAVE THE
  STA POOL,1 DEFINITION IN THE FIRST STACK ADDRESS
  CAL MODFLG SAVE THE MODE IN THE SECOND STACK ADDRESS
  STA POOL-1,1
  TXI ++1,1,2 BUMP POINTER
  TXI ++1,2,-1 SKIP THE LEFT PAREN (ALREADY CHECKED FOR)
  REM 068400
  SCANAC TSX SCNBCO,4 GET AN ARGUMENT
<table>
<thead>
<tr>
<th>TMI</th>
<th>ERROR8</th>
<th>MUST BE SYMBOLIC</th>
<th>FT068470</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLW</td>
<td>POOL,1</td>
<td>SAVE IN THE STACK</td>
<td>FT068480</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FT068490</td>
</tr>
<tr>
<td>NIT</td>
<td>DBLMD</td>
<td>IS THIS A DOUBLE-LENGTH STATEMENT.</td>
<td>FT068500</td>
</tr>
<tr>
<td>TRA</td>
<td>SCANAJ</td>
<td>NO</td>
<td>FT068510</td>
</tr>
<tr>
<td>LAS</td>
<td>=H100000</td>
<td>YES. IF THIS IS A REAL VARIABLE,</td>
<td>FT068520</td>
</tr>
<tr>
<td>TRA</td>
<td>++1</td>
<td>THEN SKIP A CELL IN THE STACK</td>
<td>FT068530</td>
</tr>
<tr>
<td>LAS</td>
<td>=HNI(!)</td>
<td>SO THAT THE CALLING SEQUENCE WILL</td>
<td>FT068540</td>
</tr>
<tr>
<td>TXI</td>
<td>++2,1,1</td>
<td>COME OUT RIGHT.</td>
<td>FT068550</td>
</tr>
<tr>
<td>NOP</td>
<td></td>
<td></td>
<td>FT068560</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FT068570</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FT068580</td>
</tr>
<tr>
<td>SCANJ</td>
<td>SKIPL1</td>
<td>CMCHAR IS IT A COMMA.</td>
<td>FT068590</td>
</tr>
<tr>
<td>TXI</td>
<td>SCANAC,1,1</td>
<td>YES. GO GET ANOTHER ARGUMENT</td>
<td>FT068600</td>
</tr>
<tr>
<td>CAL</td>
<td>COLUMN,2</td>
<td>GET NEXT TWO CHARACTERS, AND</td>
<td>FT068610</td>
</tr>
<tr>
<td>ERA</td>
<td>=H)</td>
<td>DETERMINE THAT THEY ARE =</td>
<td>FT068620</td>
</tr>
<tr>
<td>ARS</td>
<td>24</td>
<td></td>
<td>FT068630</td>
</tr>
<tr>
<td>TNZ</td>
<td>ERROR7</td>
<td></td>
<td>FT068640</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FT068650</td>
</tr>
<tr>
<td>LXA</td>
<td>SCAN5B,4</td>
<td>GET THE START OF STACK</td>
<td>FT068660</td>
</tr>
<tr>
<td>SXD</td>
<td>SCANAF,1</td>
<td>SET TEST INSTRUCTION DECREMENT</td>
<td>FT068670</td>
</tr>
<tr>
<td>PXD</td>
<td>1</td>
<td>TERMINATION OF STACK</td>
<td>FT068680</td>
</tr>
<tr>
<td>STD</td>
<td>POOL-1,4</td>
<td>SAVE END-OF-STACK IN 2ND WORD DECREMENT</td>
<td>FT068690</td>
</tr>
<tr>
<td>TXI</td>
<td>++1,1,1</td>
<td>BUMP POOL POINTER</td>
<td>FT068700</td>
</tr>
<tr>
<td>SXA</td>
<td>NXILOC,1</td>
<td>ALL DONE. RESET THE POOL POINTER</td>
<td>FT068710</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FT068720</td>
</tr>
<tr>
<td>LDI</td>
<td>POOL,4</td>
<td>GET THE DEFINITION WORD FROM STACK</td>
<td>FT068730</td>
</tr>
<tr>
<td>RIL</td>
<td>BINIF</td>
<td>RESET THE INTERNAL FUNCTION BIT</td>
<td>FT068740</td>
</tr>
<tr>
<td>SFI</td>
<td>ASTACK</td>
<td>PUT INTO TH FIRST ASTACK WORD</td>
<td>FT068750</td>
</tr>
<tr>
<td>CAL</td>
<td>ASKEY</td>
<td>GET OPERATOR KEY FOR ARITH. ST. FUNCTIONS</td>
<td>FT068760</td>
</tr>
<tr>
<td>SLW</td>
<td>ASTACK-1</td>
<td>PUT INTO SECOND ASTACK WORD</td>
<td>FT068770</td>
</tr>
<tr>
<td>AXI</td>
<td>1,1</td>
<td>SET ASTACK POINTER TO PROPER VALUE</td>
<td>FT068780</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FT068790</td>
</tr>
<tr>
<td>TSX</td>
<td>CITBLE,4</td>
<td>TELL CITBLD OF START OF ARITH. STATE. FN.</td>
<td>FT068800</td>
</tr>
<tr>
<td>TXI</td>
<td>GETNAT,2,-2</td>
<td>GO TO COMPIL ROUTINE, AND SKIP THE )=</td>
<td>FT068810</td>
</tr>
</tbody>
</table>
EJECT

* NUMBER PROCESSOR...

* THIS ROUTINE IS ENTERED WHEN THE INITIAL CHARACTER SCANNED
  * IS A DIGIT. IF THE COLUMN 1 MODE IS 'B', THE OCTAL CONSTANT SCAN 6830
  * SECTION IS ENTERED, OTHERWISE, ANY INTEGER PART OF THE NUMBER
    * IS SCANNED OFF AND COMPUTED. WHEN A NON-DIGIT CHARACTER IS
  * ENCOUNTERED, THIS INTEGER SECTION IS EXITED. IF THE CHARACTER
  * IS NOT A DECIMAL POINT OR AN 'H', THE INTEGER IS STASHED IN CONTAB
  * AS A FIXED POINT DECREMENT INTEGER.

REM

* IF THE CHARACTER IS AN 'H', THE HOLLERITH LITERAL SCANNER IS
  * ENTERED, WITH THE COMPUTED INTEGER BEING THE CHARACTER COUNT.

REM

* IF THE CHARACTER AFTER THE INTEGER IS A DECIMAL POINT, THE
  * FLOATING POINT SCANNER IS ENTERED. THIS SECTION IS ALSO
  * ENTERED WHEN THE ORIGINAL SCAN IDENTIFIES A DECIMAL POINT.
  * THE FRACTIONAL PART (IF ANY) OF THE NUMBER IS SCANNED OFF.
  * THE BINARY EQUIVALENT OF EACH DIGIT OF THE FRACTION IS COMPUTED
  * BY MULTIPLYING IT BY THE APPROPRIATE FULL WORD FIXED POINT
  * POWER OF 10 (IN DOUBLE PRECISION), AND ADDING THE RESULT TO
  * THE PREVIOUS FRACTION COMPUTATION. IF THE INTEGER PART IS
  * ZERO, ANY LEADING FRACTIONAL ZEROS ARE COUNTED, AND THE
  * FRACTION COMPUTATION BEGINS WITH THE FIRST NON-ZERO DIGIT.
  * AS IF IT WERE THE DIGIT REPRESENTING 10**-1.
  * IF AN 'E' IS ENCOUNTERED, THE EXPONENT IS SCANNED OFF. THIS
  * EXPONENT IS COMBINED WITH THAT ONE GENERATED BECAUSE OF ANY
  * LEADING FRACTIONAL ZEROS. WHEN THE END OF THE CONSTANT IS
  * ENCOUNTERED, THE INTEGER AND FRACTION ARE COMBINED AND
  * NORMALIZED (FIRST NON-ZERO BIT IN POSITION 9), THE RESULTING
  * DOUBLE PRECISION CONSTANT IS MULTIPLIED (DOUBLE PRECISION) BY
  * THE APPROPRIATE POWER OF 10 AS SPECIFIED BY THE PREVIOUSLY
  * COMPUTED EXPONENT. (A FLOATING POINT TRAP MAY OCCUR HERE...)
  * ANY REGISTER INVOLVED IS SET TO ZERO.
  * IF THIS IS A SINGLE PRECISION OR COMPLEX STATEMENT, THE HIGH-
  * ORDER PART OF THE CONSTANT IS ROUNDED AND STASHED IN CONTAB.
  * IF DOUBLE PRECISION, BOTH PARTS OF THE CONSTANT ARE STASHED.

SPACE 2

SCAN1 CAL MODFLG GET THE COLUMN 1 FLAG
ERAS =1 IS IT A BOOLEAN STATEMENT.
TZE SCNOCY YES, NUMBER MUST BE OCTAL

REM

SCAN2 CLA COLUMN,2
ARS 30
SIO INTG

REM

SCAN3 STZ EXPON
STZ DECP TRIST RTM DOUBLE PRECISION FLAG
STZ FRAC OUT
STZ FRAC+1
STZ EXPON

REM

SCAN4 RDI **+1,2-1 GET NEXT CHARACTER
CAL COLUMN,2
ARS 30
SLW CHAR
LAS =10
TRA SCAN1B *GREATER THAN 10
MACER MACHINE ERROR, OCTAL 12 FROM DECIMAL TAPE

REM

CLA INTG ANOTHER DIGIT
SCANIK SLW
SXA **2,4
LAS =10
TRA **
MACER ADDRESS SET PREVIOUSLY
CLA EXPN
ALS 2
ADD EXPN
ALS 1
ACL CHAR
STO EXPN
AXI SCAN1J,4 GET PROPER TRANSFER ADDRESS
SCANIG TXI **3,2,-1
CAL COLUMN, 2
ARS 30
TRA SCANIK
REM
STO EXPN
TRA SCANIG
REM
SCANIH CLS =0 AFTER E
REM
SCANIF ERROR 72,**1
LGR 74 ZERO EVERYTHING
TRA SCAN1Y PUT OUT A ZERO CONSTANT
REM
REM (FPT). TRA **1 CELL 8 TRA INTO FPT ROUTINE
LDI 0 GET THE TRAP WORD
LFT 4 IS IT OVERFLOW.
ERROR 72,EXFPT-2 YES. PRINT MESSAGE AND CONTINUE
LFT 2 IS IT AQ AND MQ.
ZAC YES. ZERO THE ACCUM
LDQ =0 ZERO THE MQ
EXFPT TRA* 0 RETURN
REM
SPACE 5 OCTAL CONSTANT SCANNER
REM
SCNOCT STZ INTG ZERO THE CONSTANT HOLDER
SXA SCNOCT, 2 SAVE THE COLUMN POINTER
AXI 13, 4
CAL COLUMN, 2
TXI **1,2,-1
LGR 33 INZ SCNOCL NOT AN OCTAL CHARACTER
REM
CAL INTG
LGL 3 MOVE DIGIT INTO NUMBER
SLW INTG RESTORE THE PARTIAL INTEGER
TIX SCNOCT+2, 4, 1 ERROR 72,SKEND OCTAL NUMBER TOO BIG FOR COMPUTER
REM
SCNOCL LGL 3 RESTORE THE WHOLE CHARACTER
LAS =H00000H IS IT AN H.
TRA **2 NO
TRA SCNOCL YES. HOLLERITH LITERAL
LAS =060 IS IT A BLANK.
TRA **3 NO. PREPARE TO EXIT
TRA SCNOCL+2 YES. SKIP THE BLANK
LAS =012 IS THIS AN ILLEGAL OCTAL DIGIT.
CAL INTG NO. PREPARE TO EXIT
<table>
<thead>
<tr>
<th>TXI</th>
<th>SCANIN+1,2,1</th>
<th>PUT IN CONTAB AS A 'REAL' NUMBER</th>
<th>FT071810</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR</td>
<td>4,SKEND</td>
<td>ILLEGAL OCTAL DIGIT. ERROR</td>
<td>FT071820</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SCNOC2</td>
<td>AXI</td>
<td>**,2</td>
<td>FT071830</td>
</tr>
<tr>
<td>TRA</td>
<td>SCANIT</td>
<td>HOLLERITH LITERAL. RESTORE THE COLUMN</td>
<td>FT071840</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>POINTER AND RETURN FOR LITERAL SCAN</td>
<td>FT071850</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FT071860</td>
</tr>
</tbody>
</table>
EJECT

HOLLERITH LITERAL SCANNER.

* THIS SCANNER IS ENTERED WHEN A FIXED POINT INTEGER IS FOLLOWED
* BY THE CHARACTER H. SINCE THE FORM OF THE STATEMENT AS IT
* APPEARS IN 'COLUMN' CONTAINS NO BLANKS, AND SINCE THE LITERAL
* MAY HAVE BLANK CHARACTERS IN IT, IT IS NECESSARY TO SCAN THE
* ORIGINAL STATEMENT (IN 'COLUMN') TO OBTAIN THE LITERAL.
* THE COUNT OF NON-BLANK CHARACTERS THUS FAR ENCOUNTERED
* (DOWN THROUGH THE H) IS THE COLUMN POINTER (XR 2). THIS SAME
* NUMBER OF NON-BLANK CHARACTERS IS SCANNED OFF THE CARD ARRAY.
* THE CORRECT NUMBER OF HOLLERITH CHARACTERS IS THEN SCANNED
* FROM CARD, AND PACKED TO THE LEFT TO FORM CONSTANT WORD(S).
* IF THIS IS NOT IN A SUBPROGRAM CALLING SEQUENCE, ONLY ONE
* WORD IS STASHED IN THE CONSTANT TABLE. IF THIS IS IN A CALLING
* SEQUENCE, ALL THE SCANNED CHARACTERS ARE STASHED (6 TO A WORD) INTO
* CONTAB. IN THIS LATTER CASE, A WORD OF ALL SEVENS (OCTAL)
* IS ADDED TO THE END OF THE LITERAL.
* INFORMATION IS SAVED TO ENABLE A RESTART OF THE SCAN FROM
* THIS LITERAL, IN CASE ANOTHER LITERAL IS ENCOUNTERED IN THIS
* SAME STATEMENT.
* NOTE THAT LITERALS IN DOUBLE PRECISION AND COMPLEX STATEMENTS
* ARE USUALLY MEANINGLESS, BUT ALLOWED NEVERTHELESS. ALSO,
* WHILE LITERALS IN BOOLEAN STATEMENTS WILL NOT WORK CORRECTLY
* IN IBM'S FORTRAN COMPILER, THEY WILL FUNCTION CORRECTLY IN
* FASTRAN.

SPACE 2

SCANH SXD SCANH+4,2 IS THIS THE FIRST LITERAL OF STATEMENT.
ZET HFLAG NO. PICK UP THE SCAN WHERE WE LEFT OFF
TRA SCANH2
REM
STL HFLAG YES. FIRST TIME IN STATEMENT
AXT 0,1 SET NON-BLANK CHARACTER COUNT TO ZERO
AXC 1,2 SET FOR FIRST WORD OF STATEMENT
AXT 6,4 SET BYTE COUNT TO 6
LDQ CARD,+2 GET STATEMENT WORD
REM

SCANH0 ZAC
LGL 6 IS IT A NON-BLANK CHARACTER.
ERA =H0000 IS IT A NON-BLANK CHARACTER.
TZE SCANH3
REM
TXI *+1,1,-1 YES. UP THE NON-BLANK CHARACTER COUNT
SCANH4 TAL SCANH1,1,0,0 HAVE CORRECT NUMBER OF NON-BLANK
REM CHARACTERS BEEN FOUND YET.
SCANH3 TIX SCANH0,4,1 NO. IS THIS WORD USED UP.
TXI SCANH0-2,2,-1 YES. GET NEXT WORD ON CARD
REM
SCANH1 SAVE SCANH2 SAVE MQ AND XRS FOR POSSIBLE RESTART
STQ SCANH1
REM
REM
AXT COLUMN,1 NOW SCAN OFF THE LITERAL
SXH SCANHA,1 PUT 'COLUMN' INTO INSTRUCTION ADDRESS
LXH INT6,1 GET NUMBER OF HOLLERITH CHARACTERS NEEDED
REM
SCANHC CAL CARD,+2 GET CURRENT WORD FROM CARD
LDQ CARD,+1,2 ALSO GET NEXT WORD
XEK SCANHB,4 LEFT-JUSTIFY THE LITERAL
**3,1,5  SKIP IF LITERAL FILLS WORD (OR MORE)   FT072470
REX                    FT072480
ANR  SCANT1,1  SAVE ONLY PERTINENT CHARACTERS  FT072490
ORA  SCANT2,1  ADD TERMINAL BLANKS  FT072500
REM                    FT072510
SCANHA  **  SAVE WORDS IN THE COLUMN REGION  FT072520
CAL  **-1  BUMP ADDRESS OF THE STORE WORD  FT072530
ACL  **=1  RESTORE THE ADDRESS  FT072540
STA  **=1  RETURN IF MORE LITERAL CHARACTERS NEEDED  FT072550
REM                    FT072560
TXI  **=1,2,-1  BUMP *CARD* WORD COUNTER BY 1  FT072570
TXI  **=1,2,-1  GET COUNTER FOR NUMBER OF LITERAL WORDS  FT072580
REM                    FT072590
LAC  SCANHA,4  GET COUNT OF ALL BUT THE WORD  FT072600
XAC  SCANHA,4  GET COUNTER FOR NUMBER OF LITERAL WORDS  FT072610
REM                    FT072620
LXD  SCANHF,2  RETURN FOR ANOTHER WORD IF NECESSARY  FT072630
AXI  SCANHF,2  RETURN FOR ANOTHER WORD IF NECESSARY  FT072640
REM                    FT072650
SCANHF  ZAC  COUNT NUMBER OF NON-BLANK CHARACTERS IN LIT  FT072660
LGL  6  INITIALIZATION  FT072670
ERA  =H00000  SHIFTS IN A CHARACTER  FT072680
REM                    FT072690
TZE  **=2  SKIPPING BLANKS  FT072700
TXI  **=1,2,-1  ADJUST THE COLUMN POINTER  FT072710
TXI  **=1,2,-1  ADJUST THE COLUMN POINTER  FT072720
REM                    FT072730
TIX  SCANHF,1  RETURN IF NOT DONE WITH THIS WORD  FT072740
REM                    FT072750
CAL  COLUMN  CAN THIS BE IN A SUBPROGRAM ARGUMENT LIST?  FT072760
REM                    FT072770
TRA  SCANIN+1  NO, GO STASH THE LITERAL  FT072780
REM                    FT072790
LXA  INTG,1  YES, POSSIBLY. GET NUMBER OF CHARACTERS  FT072800
REM                    FT072810
TXH  SCANHD,1  IF MORE THAN ONE WORD, GO USE THE  FT072820
REM                    FT072830
LDQ  AL LVN  BRUTE-FORCE STASHING ROUTINE  FT072840
REM                    FT072850
XCL  ASCON2,4,REAL  TERMINAL WORDS FOR SUBROUTINE ARGUMENTS  FT072860
REM                    FT072870
SUB  =1  EXCHANGE TO EFFECT CORRECT STASHING  FT072880
REM                    FT072890
TRA  SCAN1Z+1  GO STASH CONSTANT  FT072900
REM                    FT072910
SCANTH  XA  NOCONT,4  LONG LITERAL, BRUTE-FORCE STASHING USED  FT072920
REM                    FT072930
SCANHV  LXA  NOCONT,4  GET PRESENT NUMBER OF CONSTANTS IN TABLE  FT072940
REM                    FT072950
SA  SCANHV,2  SAVE THE COLUMN POINTER  FT072960
AXT  0,2  PREPARE STACK WORD  FT072970
REM                    FT072980
CAL  COLUMN,2  PREPARE THE ASTACK WORD  FT072990
SLW  CONTAB,4  GET A WORD OF THE LITERAL  FT073000
REM                    FT073010
TXI  **=1,4,1  PUT INTO THE CONSTANT TABLE  FT073020
REM                    FT073030
TXH  SCANHH,4,NOCNST  PUT OUT MSG  FT073040
REM                    FT073050
CAL  NOCONT  ADDRESS OF ACCUM GETS THE START OF STRING  FT073060
AXT  **1,2  PUT INTO THE CONSTANT TABLE  FT073070
REM                    FT073080
REM AXT 0,4  
RESET TABLE IF FULL FT073070
STR MSG100,,SCANHG-1 ERROR MESSAGE FT073090
SPACE 2 FT073100
REM SHIFT TABLE FT073110
LGL 6 FT073120
LGL 12 FT073130
LGL 18 FT073140
LGL 24 FT073150
LGL 30 FT073160
LGL 36 FT073170
SCANHB EQU * FT073180
REM FT073190
SCANH2 RESTOR ENTER HERE FOR ALL BUT FIRST TIME IN STATE FT073200
LDQ SCANHT FT073210
TRA SCANH3 FT073220
EJECT
OPERATOR SCANNERS

SPACE 3 = SIGN PROCESSING

REM

SCAN13 CAL EQLKEY
SCAN3A TXI +1,2,-1
LX A SCAX+1
TRA 3,4

SPACE 3 = PROCESSING

REM

SCAN20 CAL PLUSKY
TRA SCAN3A

SPACE 3 = PROCESSING

REM

SCAN33 LDG COLUMN+1,2 GET NEXT CHARACTER
CRQ SCAN1,1
TXI ERO79,1,SCAN1-1 NON-NUMERIC ERROR
TXH ERO79,1,SCAN1
SIZ INTG
CAL MODFLG
ERA =1
TZE ERO79
TXI SCAN1X,2,1 ADJUST COLUMN POINTER SO THAT THE
REM NUMBER SCANNER WILL CATCH THE DECIMAL POINT

SPACE 3 ) PROCESSING

REM

SCAN34 CAL RPNKEY
TRA SCAN3A

SPACE 3 - PROCESSING

REM

SCAN40 CAL UNEGKY
TRA SCAN3A

SPACE 3 = PROCESSING

REM

SCAN54 CAL COLUMN+1,2 = SIGN
ERA =H*
ARS 30
TZE **3
CAL MPRKEY = ONLY
TRA SCAN3A
REM
CAL EXPRKEY **
TXI SCAN3A,2,-1

SPACE 3 BLANK PROCESSING

REM

SCAN60 TXI SCAN3A,2,-1
SPACE 3 ) PROCESSING

REM

SCAN61 CAL DIVKEY
TRA SCAN3A

SPACE 3 = PROCESSING

REM

SCAN73 CAL CMAC0EY
TRA SCAN3A

SPACE 5 ) PROCESSING

REM

SCAN74 CAL LPNKEY
TRA SCAN3A

SPACE 5 END-OF-STATEMENT PROCESSING

REM
<table>
<thead>
<tr>
<th>SCAN77 LXA</th>
<th>SCANX,1</th>
<th>FT073830</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRA</td>
<td>1,4</td>
<td>FT073840</td>
</tr>
<tr>
<td>SPACE</td>
<td>6</td>
<td>FT073850</td>
</tr>
<tr>
<td>*</td>
<td>ILLEGAL FIRST CHARACTER</td>
<td>FT073860</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FT073870</td>
</tr>
<tr>
<td>SCAN2 ERROR</td>
<td>77,SKEND</td>
<td>FT073880</td>
</tr>
</tbody>
</table>
**TIL PASS 1 - SUBSCRIPT SCANNER**

This routine scans a subscript expression and prepares a
summary of the data needed to compute the subscript.

**Calling sequence...**

TSX 1INDEX,4

Where XR2 points to the beginning of the subscript proper.

XR1 has the astack pointer.

**REM**

1INDEX

SXA 1INDEX1,4

SXA 1INDEX1+1,1

ENT - STT,4 ZER0 THE WORKING CELLS

STZ -ENT,4

TIX *-1,4,1

REM

CAL ASTACK,1 GET SYMBOL REFERENCE

STA 1INDEX16 SAVE POINTER FOR POSSIBLE USE LATER

PAX ,4

CAL EQUIV,4 EQUIV POINTER

PAX ,4

REM

CAL POOL-1,4 GET DIMBBL INFORMATION

ARS 18

STA DIM1 FIRST DIMENSION

ARS 15

STA DIMN NUMBER OF DIMENSIONS

REM

CAL POOL-2,4 GET NEXT DIMENSION, IF ANY

ARS 18

STA DIM3 SECOND DIMENSION

REM

AXT 0,1 XR1 BUMPED FOR EACH NEW SEGMENT

1INDEX4 TSX SCNBD,4 GET VARIABLE OR CONSTANT

TMI 1INDEX2 CONSTANT

REM

1INDEX3 LAS =H100000 CHECK FOR FIXED POINT

TRA ++1

LAS =HN(0)

ERROR 12,**2 NOT FIXED POINT

MACER MACHINE ERROR

REM

TSX SYMSIQ,4 FIX UP SYMBOL

SLw VARB,1 SAVE EQUIV POINTER

STI VIND,1 SAVE EQUIV WORD

REM

CAL COLUMN,2 GET BREAK CHARACTER

IXI ++1,2,-1

ARS 30

STZ ASIGN

LAS =H00000+ IS IT PLUS SIGN.

TRA ++2

TRA 1INDEX5 YES

REM

STL ASIGN NO. INDICATE A POSSIBLE MINUS SIGN

LAS =H00000- IS IT A MINUS SIGN.

TRA ++2

TRA 1INDEX5 YES

REM

1INDEX6 LAS =H00000) IS IT CLOSING PAREN.
TRA ++2
TXI 1INDX3,1,1 YES, SCAN COMPLETED
REM
ERA =H00000, NO, IS IT A COMMA.
INZ 1IND16 ILLEGAL FORM OF SUBSCRIPT
TXI 1INDX4,1,1 RETURN FOR ANOTHER SUBSCRIPT
REM
1INDX5 TSX SCNBDC,4 GET CONSTANT
TPL 1IND16 ERROR, NOT A CONSTANT
REM
TSX BCDFIX,4 CONVERT CONSTANT
ZET ASIGN SET SIGN OF CONSTANT
REM
SSM ARS 18
1INDX7 STO CADD,1 ADDITIVE CONSTANT
REM
CAL COLUMN,2 GET BREAK CHARACTER
ARS 30
TXI 1INDX6,2,-1 GO TEST BREAK CHARACTER
REM
1INDX2 TSX BCDFIX,4 CONSTANT MULTIPLIER
ARS 18
XCL SAVE THIS CONSTANT
REM
CAL COLUMN,2 GET BREAK CHARACTER
ARS 30
LAS =H00000* IS IT AN ASTERISK.
TRA ++2
TXI ++3,2,-1 YES
XCL GET THE CONSTANT AGAIN
TRA 1INDX7
REM
STQ CMLT,1 CONSTANT IS A MULTIPLIER
TSX SCNBDC,4 ASTERISK, GET VARIABLE
TPL 1IND13 PROCESS REST OF EXPRESSION
TRA 1IND16 NOT A VARIABLE, ILLEGAL FORM
REM
1INDX3 ITXL 1IND14-1,1,1 ANY SINGLY-DIMENSIONED SUBSCRIPT ALLOWED
PXA +1
SUB DIMN CHECK FOR CORRECT DIMENSIONING OF SUBSCRIPT
TZE 1IND14 CORRECT NUMBER OF DIMENSIONS
REM
1IND16 AXT **,4 GET SYMTAB POINTER
CAL SYMTAB,4 GET BCD NAME OF OFFENDING ARRAY
SLW FORM+2+6 PUT INTO ERROR MESSAGE
ERROR 42,SKEND
REM
SXA DIMN,1 FOR PSEUDO-SINGLY DIMENSIONED ARRAYS
1IND14 LQW =0 COMPUTE THE CONSTANT ADDEND
CAL =2
LAS DIMN HOW MANY DIMENSIONS.
TRA 1INDX8 ONE DIMENSION
TRA 1INDX9 TWO DIMENSIONS
REM
CLS =1 THREE DIMENSIONS
ADD CADD-2 THIRD ADDITIVE CONSTANT
XCA
MPY DIM2 TIMES SECOND DIMENSION
REM
1INDX9 XCA
SUB  =1  SECOND ADDITIVE CONSTANT  FT075090
ADD  CADD-1  FIRST ADDITIVE CONSTANT  FT075150
XCA  FT075100
MPY  DIM1  TIMES FIRST DIMENSION  FT075120
REM  FT075130
IINDX8  XCA  FT075140
ADD  CADD  TAKE COMPLEMENT FOR POSITIVE CONSTANT ADD.  FT075160
PAC  ++2  FT075170
TPL  4  FOR MINUS CONSTANT ADDEND  FT075180
PIX  **1,4,1  ADD ONE FOR ARRAYS  FT075190
SUM  CONADD,4  CONSTANT ADDEND  FT075200
REM  FT075210
CAL  =2  GET CONSTANT MULTIPLIERS FOR ISTACK  FT075220
LAS  DIMN  FT075230
TRA  IIND10  ONE DIMENSION  FT075240
TRA  IIND11  TWO DIMENSIONS  FT075250
REM  FT075260
NZT  VIND-2  THREE DIMENSIONS. IS THERE A 3RD VARIABLE.  FT075270
TRA  IIND11  NO.  FT075280
REM  FT075290
LDQ  DIM2  YES  FT075300
MPY  DIM1  FT075310
ZET  CMLT-2  IS THERE A THIRD MULTIPLIER.  FT075320
MPY  CMLT-2  YES  FT075330
STQ  SAV3  SAVE THIRD CONSTANT  FT075340
REM  FT075350
IIND11  NZT  VIND-1  IS THERE A SECOND VARIABLE.  FT075360
TRA  IIND10  NO.  FT075370
REM  FT075380
LDQ  DIM1  YES  FT075390
ZET  CMLT-1  IS THERE A SECOND MULTIPLIER.  FT075400
MPY  CMLT-1  YES  FT075410
STQ  SAV2  SAVE SECOND CONSTANT  FT075420
REM  FT075430
IIND10  NZT  VIND  IS THERE A FIRST VARIABLE.  FT075440
TRA  IBUILD  NO. PROCEED TO BUILD ISTACK  FT075450
CAL  CMLT  IS THERE A FIRST MULTIPLIER.  FT075460
TNZ  ++2  FT075470
CAL  =1  NO MULTIPLIER EXPLICITLY GIVEN  FT075480
SLW  SAV1  NOW GO BUILD THE ISTACK ENTRY  FT075490
REM  FT075500
* EJECT
* THIS SECTION BUILDS THE ISTACK ENTRY.
* THE FORM OF THE ENTRY IS...
* FIRST WORD ADDRESS HAS THE ASTACK POINTER. THE
* DECREMENT MAY CONTAIN THE ADDRESS OF THE NEXT UNUSED
* SIX-WORD ISTACK ENTRY.
* SECOND WORD IS THE ORIGINAL ASTACK WORD.
* THIRD WORD ADDRESS IS THE CONSTANT ADDEND.
* FOR A SUBSCRIPT WHICH REQUIRES NO EXTRA CODE, THE
* THIRD WORD DECREMENT HAS THE EQUIV POINTER (IF ANY)
* FOR THE SINGLE VARIABLE INVOLVED (THIS ISTACK WORD HAS A
* MINUS SIGN).
* IF CODE MUST BE GENERATED TO COMPUTE THE SUBSCRIPT,
* THE THIRD WORD DECREMENT HAS ZERO UNTIL THE CODE IS
* GENERATED, AT WHICH TIME IT WILL RECEIVE THE NUMBER
* OF THE WORKING CELL USED TO COMPUTE AND HOLD THE
* SUBSCRIPT. THE ISTACK WORD HAS A PLUS SIGN FOR THIS CASE
* IF THE SUBSCRIPT HAS ONLY A CONSTANT ADDEND, THE
* DECREMENT OF THE THIRD WORD IS ZERO (PLUS SIGN).
* ALSO IF CODE MUST BE GENERATED TO COMPUTE THE SUBSCRIPT,
* THE SIGN OF THE FIRST ISTACK WORD IS MADE MINUS.
* ONE OR MORE OF THE NEXT THREE WORDS MAY CONTAIN
* THE LINEAR COEFFICIENTS (IN DECREMENTS) FOR THE
* SUBSCRIPT VARIABLES (EQUIV POINTERS IN ADDRESSES).
* ANY WORDS NOT USED ARE ZEROED OUT. THEY ARE STILL A
* PART OF THE ISTACK ENTRY.
* THE ASTACK ENTRY GETS THE ADDRESS OF THE ISTACK ENTRY IN ITS
* ADDRESS FIELD, AND A NON-ZERO TAG FIELD.
* THE ISTACK ENTRIES ARE LINKED TOGETHER SO THAT THEY MAY BE
* RE-USED FOR EACH STATEMENT. THE LINKING DIRECTOR IS THE CELL
* ISTKFG, WHICH WILL CONTAIN THE START OF ISTACKS IN ITS ADDRESS
* AND THE ADDRESS OF THE LAST-USED (IN THIS STATEMENT) ISTACK ENTRY.
* IN ITS DECREMENT. THIS CELL IS INITIALIZED TO ZERO AT THE
* BEGINNING OF EACH COMPILATION. ON ENTRY TO THE COMPIL ROUTINE,
* THE DECREMENT IS SET EQUAL TO THE ADDRESS.

IBUILD CAL CMUL IF THESE THREE WORDS ARE ZERO, THE
ACL VIND-2 SUBSCRIPT IS JUST A VARIABLE IN THE
ACL VIND-1 FIRST DIMENSION POSSIBLY PLUS A CONSTANT.
SLW ASIGN
REM
LXA ISTKFG,1 GET ADDRESS OF LAST-USED ISTACK ENTRY
CAL POOL,1 GET FIRST WORD OF ENTRY
PDX IBLD2,4,0 SKIP IF NON-ZERO
TXH
REM
LXA NXTLOC,4 NO MORE AVAILABLE SPACE. GET NEW SPACE
PXD 4
STD POOL,1 SET LINKER INTO PREVIOUS ISTACK ENTRY
TXI **1,4.6 BUMP NXTLOC
SXA NXTLOC,4 RESET NXTLOC
TXI **1,4.6 AND RESTORE THE POINTER
REM
IFLD2 SXA ISTKFG,4 SET THE FLAG TO CURRENT ISTACK ENTRY USED
LXA LIINDX1+1,1 RESTORE XR1 (ASTACK POINTER)
LDQ ASTACK,1 GET ASTACK WORD

FT075510
FT075520
FT075530
FT075540
FT075550
FT075560
FT075570
FT075580
FT075590
FT075600
FT075610
FT075620
FT075630
FT075640
FT075650
FT075660
FT075670
FT075680
FT075690
FT075700
FT075710
FT075720
FT075730
FT075740
FT075750
FT075760
FT075770
FT075780
FT075790
FT075800
FT075810
FT075820
FT075830
FT075840
FT075850
FT075860
FT075870
FT075880
FT075890
FT075900
FT075910
FT075920
FT075930
FT075940
FT075950
FT075960
FT075970
FT075980
FT075990
FT076000
FT076010
FT076020
FT076030
FT076040
FT076050
FT076060
FT076070
FT076080
FT076090
FT076100
<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>STQ</td>
<td>POOL-1,4 PUT INTO SECOND ISTACK WORD</td>
<td>FT076110</td>
</tr>
<tr>
<td>PZA</td>
<td>1,4 PUT POINTER INTO ACCUM</td>
<td>FT076120</td>
</tr>
<tr>
<td>STA</td>
<td>ASTACK,1 SET ASTACK POINTER TO ISTACK</td>
<td>FT076130</td>
</tr>
<tr>
<td>COM</td>
<td>ASTACK,1 NON-ZERO TAG INTO ASTACK</td>
<td>FT076140</td>
</tr>
<tr>
<td>SFT</td>
<td>ASTACK,1 GET ASTACK POINTER</td>
<td>FT076150</td>
</tr>
<tr>
<td>STA</td>
<td>POOL,4 PUT IN ADDRESS OF FIRST ISTACK WORD</td>
<td>FT076160</td>
</tr>
<tr>
<td>STP</td>
<td>POOL,4 ERASE ANY PREVIOUS SIGN BIT</td>
<td>FT076170</td>
</tr>
<tr>
<td>ZAC</td>
<td>ASIGN DETERMINE IF ANY CODE IS NEEDED</td>
<td>FT076180</td>
</tr>
<tr>
<td>ZET</td>
<td>ASIGN YES CODE IS NEEDED</td>
<td>FT076190</td>
</tr>
<tr>
<td>TRA</td>
<td>1BUIL1 YES CODE IS NEEDED.</td>
<td>FT076200</td>
</tr>
<tr>
<td>CAL</td>
<td>VIND GET A POSSIBLE FIRST VARIABLE</td>
<td>FT076210</td>
</tr>
<tr>
<td>TZE</td>
<td>++3 IF ZER SUBSCRIPT IS ONLY A CONSTANT</td>
<td>FT076220</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FT076230</td>
</tr>
<tr>
<td>CLS</td>
<td>VARB GET VARIABLE EQUIV POINTER</td>
<td>FT076240</td>
</tr>
<tr>
<td>ALS</td>
<td>18 TO DECREMENT</td>
<td>FT076250</td>
</tr>
<tr>
<td>1BUIL1</td>
<td>ORA CONADD INSERT THE CONSTANT ADDEND</td>
<td>FT076260</td>
</tr>
<tr>
<td>STU</td>
<td>POOL-2,4 PUT C.A. AND POINTER INTO ISTACK 3RD WORD</td>
<td>FT076270</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FT076280</td>
</tr>
<tr>
<td>ZET</td>
<td>ASIGN IS CODE NEEDED.</td>
<td>FT076290</td>
</tr>
<tr>
<td>TRA</td>
<td>1BUIL2 YES CONTINUE BUILDING ISTACK</td>
<td>FT076300</td>
</tr>
<tr>
<td>IINDXI</td>
<td>AXT **,4</td>
<td>FT076310</td>
</tr>
<tr>
<td>AXT</td>
<td>**,1</td>
<td>FT076320</td>
</tr>
<tr>
<td>TRA</td>
<td>1,4</td>
<td>FT076330</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FT076340</td>
</tr>
<tr>
<td>1BUIL2</td>
<td>PZA 1</td>
<td>FT076350</td>
</tr>
<tr>
<td>PAX</td>
<td>1</td>
<td>FT076360</td>
</tr>
<tr>
<td>CLS</td>
<td>POOL,1 SET SIGN MINUS IN FIRST POOL WORD.</td>
<td>FT076370</td>
</tr>
<tr>
<td>STO</td>
<td>POOL,1 THIS INDICATES THAT CODE MUST BE GENERATED</td>
<td>FT076380</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FT076390</td>
</tr>
<tr>
<td>STZ</td>
<td>POOL-3,1 TO COMPUTE THE SUBSCRIPT</td>
<td>FT076400</td>
</tr>
<tr>
<td>STZ</td>
<td>POOL-4,1</td>
<td>FT076410</td>
</tr>
<tr>
<td>STZ</td>
<td>POOL-5,1</td>
<td>FT076420</td>
</tr>
<tr>
<td>TXI</td>
<td>**1,1,3 GET FIRST MULTIPLIER</td>
<td>FT076430</td>
</tr>
<tr>
<td>CAL</td>
<td>SAV1 GET FIRST MULTIPLIER</td>
<td>FT076440</td>
</tr>
<tr>
<td>TZE</td>
<td>1BUIL3 SHIFT INTO DECREMENT</td>
<td>FT076450</td>
</tr>
<tr>
<td>ALS</td>
<td>18 MOVE INTO DECREMENT</td>
<td>FT076460</td>
</tr>
<tr>
<td>TSX</td>
<td>ASCON1,4 PUT CONSTANT INTO CONSTANT TABLE</td>
<td>FT076470</td>
</tr>
<tr>
<td>ALS</td>
<td>18 MOVE INTO DECREMENT</td>
<td>FT076480</td>
</tr>
<tr>
<td>ORA</td>
<td>VARB ADD IN EQUIV POINTER FOR FIRST VARIABLE</td>
<td>FT076490</td>
</tr>
<tr>
<td>SLW</td>
<td>POOL,1 MAKE ISTACK (POOL) ENTRY</td>
<td>FT076500</td>
</tr>
<tr>
<td>TXI</td>
<td>**1,1,1 BUMP POINTER POINTER</td>
<td>FT076510</td>
</tr>
<tr>
<td>I BUIL3</td>
<td>CAL SAV2</td>
<td>FT076520</td>
</tr>
<tr>
<td>IZE</td>
<td>1BUIL4</td>
<td>FT076530</td>
</tr>
<tr>
<td>ALS</td>
<td>18 MOVE INTO DECREMENT</td>
<td>FT076540</td>
</tr>
<tr>
<td>TSX</td>
<td>ASCON1,4 PUT CONSTANT INTO CONSTANT TABLE</td>
<td>FT076550</td>
</tr>
<tr>
<td>ALS</td>
<td>18 MOVE INTO DECREMENT</td>
<td>FT076560</td>
</tr>
<tr>
<td>ORA</td>
<td>VARB-1 OR IN NEXT EQUIV POINTER</td>
<td>FT076570</td>
</tr>
<tr>
<td>SLW</td>
<td>POOL,1 PUT INTO ISTACK</td>
<td>FT076580</td>
</tr>
<tr>
<td>TXI</td>
<td>**1,1,1</td>
<td>FT076590</td>
</tr>
<tr>
<td>I BUIL4</td>
<td>CAL SAV3</td>
<td>FT076600</td>
</tr>
<tr>
<td>ALS</td>
<td>18 MOVE INTO DECREMENT</td>
<td>FT076610</td>
</tr>
<tr>
<td>TSX</td>
<td>ASCON1,4 PUT CONSTANT INTO CONSTANT TABLE</td>
<td>FT076620</td>
</tr>
<tr>
<td>ALS</td>
<td>18 MOVE INTO DECREMENT</td>
<td>FT076630</td>
</tr>
<tr>
<td>ORA</td>
<td>VARB-1 OR IN NEXT EQUIV POINTER</td>
<td>FT076640</td>
</tr>
<tr>
<td>SLW</td>
<td>POOL,1 PUT INTO ISTACK</td>
<td>FT076650</td>
</tr>
<tr>
<td>TXI</td>
<td>**1,1,1</td>
<td>FT076660</td>
</tr>
</tbody>
</table>

This document appears to be a listing of assembly code instructions with addresses provided for each instruction.
<table>
<thead>
<tr>
<th>TZE</th>
<th>1INDX1</th>
<th>FT076710</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALS</td>
<td>18</td>
<td>FT076720</td>
</tr>
<tr>
<td>TSX</td>
<td>ASCON1,4</td>
<td>FT076730</td>
</tr>
<tr>
<td>ALS</td>
<td>18</td>
<td>FT076740</td>
</tr>
<tr>
<td>ORA</td>
<td>VARB-2</td>
<td>ADD IN THIRD EQUV POINTER</td>
</tr>
<tr>
<td>SLW</td>
<td>POOL,1</td>
<td>PUT INTO ISTACK</td>
</tr>
<tr>
<td>TRA</td>
<td>1INDX1</td>
<td>FT076770</td>
</tr>
<tr>
<td>EJECT</td>
<td>ROUTINE TO PRODUCE CODE NECESSARY TO EVALUATE A SUBSCRIPT.</td>
<td>FT076780</td>
</tr>
<tr>
<td>-------</td>
<td>----------------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>ENTRY IS...</td>
<td>CAL ISTACK,POINTER</td>
<td>FT076810</td>
</tr>
<tr>
<td></td>
<td>TSX ICODEX,4</td>
<td>FT076820</td>
</tr>
<tr>
<td></td>
<td>ON RETURN, THE DECREMENT OF THE THIRD ISTACK WORD HAS THE</td>
<td>FT076830</td>
</tr>
<tr>
<td></td>
<td>WORKING CELL POINTER WHERE THE RESULT OF THE COMPUTATION</td>
<td>FT076840</td>
</tr>
<tr>
<td></td>
<td>IS STORED, IF ANY CODE WAS PRODUCED.</td>
<td>FT076850</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>FT076860</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>FT076870</td>
</tr>
<tr>
<td></td>
<td>ICODEX SAVE EXICOD</td>
<td>FT076880</td>
</tr>
<tr>
<td></td>
<td>STZ ASGN INITIALIZE FLAG CELL</td>
<td>FT076890</td>
</tr>
<tr>
<td></td>
<td>PAX 2 GET ISTACK POINTER</td>
<td>FT076900</td>
</tr>
<tr>
<td></td>
<td>CLA POOL,2 GET FIRST ISTACK WORD</td>
<td>FT076910</td>
</tr>
<tr>
<td></td>
<td>TPL EXICOD EXIT IF NO CODE IS NEEDED</td>
<td>FT076920</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>FT076930</td>
</tr>
<tr>
<td></td>
<td>TXI **1,2,3 BUMP POINTER AND SET DECREMENT OF TEST INST</td>
<td>FT076940</td>
</tr>
<tr>
<td></td>
<td>SXD L1ND24,2</td>
<td>FT076950</td>
</tr>
<tr>
<td></td>
<td>TXI **1,2,2 SET POINTER TO BOTTOM OF STACK</td>
<td>FT076960</td>
</tr>
<tr>
<td></td>
<td>TSX STORE,4 MUST STORE OFF ANY CURRENT REGISTER</td>
<td>FT076970</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>FT076980</td>
</tr>
<tr>
<td></td>
<td>TSX GETWRK,4 GET A WORKING CELL</td>
<td>FT076990</td>
</tr>
<tr>
<td></td>
<td>SLW TEMPX SAVE</td>
<td>FT077000</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>FT077010</td>
</tr>
<tr>
<td></td>
<td>11ND15 CAL POOL,2 GET MULTIPLIER</td>
<td>FT077020</td>
</tr>
<tr>
<td></td>
<td>TXE 11ND19+1 MO CODE HERE</td>
<td>FT077030</td>
</tr>
<tr>
<td></td>
<td>PAX .1 GET EQUIV POINTER</td>
<td>FT077040</td>
</tr>
<tr>
<td></td>
<td>ARS 18 MOVE CONTAB POINTER INTO ADDRESS</td>
<td>FT077050</td>
</tr>
<tr>
<td></td>
<td>CODEO LDQ,C CODE IS... LDQ CONSTANT</td>
<td>FT077060</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>FT077070</td>
</tr>
<tr>
<td></td>
<td>NZT ASGN HAS CODE ALREADY BEEN PRODUCED.</td>
<td>FT077080</td>
</tr>
<tr>
<td></td>
<td>TRA 11ND27 NO</td>
<td>FT077090</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>FT077100</td>
</tr>
<tr>
<td></td>
<td>CODECO STD,TEMPX,W CODE IS... STO TEMPORARY</td>
<td>FT077110</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>FT077120</td>
</tr>
<tr>
<td></td>
<td>11ND27 LDI EQUIV,1 GET EQUIV WORD FOR VARIABLE</td>
<td>FT077130</td>
</tr>
<tr>
<td></td>
<td>PXX .1 EQUIV POINTER</td>
<td>FT077140</td>
</tr>
<tr>
<td></td>
<td>LEI BARGT IS IT A PARAMETER</td>
<td>FT077150</td>
</tr>
<tr>
<td></td>
<td>TRA 11ND18 YES</td>
<td>FT077160</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>FT077170</td>
</tr>
<tr>
<td></td>
<td>CODEO MPY,V CODE IS... MPY VARIABLE</td>
<td>FT077180</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>FT077190</td>
</tr>
<tr>
<td></td>
<td>11ND25 NZT ASGN HAS CODE ALREADY BEEN PRODUCED.</td>
<td>FT077200</td>
</tr>
<tr>
<td></td>
<td>TRA 11ND19 NO</td>
<td>FT077210</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>FT077220</td>
</tr>
<tr>
<td></td>
<td>CODECO ADD,TEMPX,W CODE IS... ADD TEMPORARY</td>
<td>FT077230</td>
</tr>
<tr>
<td></td>
<td>11ND19 STL ASGN SIGNAL THAT CODE HAS BEEN PRODUCED</td>
<td>FT077240</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>FT077250</td>
</tr>
<tr>
<td></td>
<td>11ND24 TXL <strong>2,2,</strong> DECREMENT WAS SET EARLIER</td>
<td>FT077260</td>
</tr>
<tr>
<td></td>
<td>TXI 11ND15,2,-1 LOWER POINTER AND RETURN</td>
<td>FT077270</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>FT077280</td>
</tr>
<tr>
<td></td>
<td>11ND22 CODEC ALS17,N CODE IS... ALS 17</td>
<td>FT077290</td>
</tr>
<tr>
<td></td>
<td>CODECO STD,TEMPX,W CODE IS... STO TEMPORARY</td>
<td>FT077300</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>FT077310</td>
</tr>
<tr>
<td></td>
<td>CAL TEMPX WORKING CELL POINTER</td>
<td>FT077320</td>
</tr>
<tr>
<td></td>
<td>ALS 10 SHIFT INTO DECREMENT</td>
<td>FT077330</td>
</tr>
<tr>
<td></td>
<td>STD POOL+1,2 SET 3RD ISTACK WORD DECREMENT TO WRK PTR</td>
<td>FT077340</td>
</tr>
<tr>
<td></td>
<td>EXICOD RESTOR</td>
<td>FT077350</td>
</tr>
<tr>
<td></td>
<td>TRA 1,4</td>
<td>FT077360</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>FT077370</td>
</tr>
<tr>
<td>REM</td>
<td>1IND18</td>
<td>TSX</td>
</tr>
<tr>
<td>-----</td>
<td>--------</td>
<td>-----</td>
</tr>
<tr>
<td>CODEC</td>
<td>MPY,N</td>
<td>CODE IS... MPY **</td>
</tr>
<tr>
<td>TRA</td>
<td>1IND25</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
TIL ARITHMETIC CODE KEY GENERATOR

* GEN ROUTINE. THIS ROUTINE BUILDS THE CODE GENERATOR KEY WORD
* FOR THE SEQUENCE 'OPERAND/OPERATOR/OPERAND' OR 'OPERATOR/OPERAND'
* THIS ROUTINE PROCESSES ALL OPERATORS EXCEPT PARENS AND COMMA.
* NOTE THAT THE 'OPERATOR/OPERAND' CASE MAY ONLY HAVE A UNARY
* MINUS AS THE OPERATOR.

REM

* THE FORM OF THE GENERATED KEY WORD IS...
* BITS P,1-6 OPERATOR KEY CODE
* BITS 7-9 MODE OF LEFT-HAND OPERAND, IF ANY
* BITS 10-11 REGISTER LOCATION OF THIS OPERAND
* BITS 12-14 MODE OF RIGHT-HAND OPERAND
* BITS 15-16 REGISTER LOCATION OF THIS OPERAND
* BITS 17-35 NOT USED

REM

* AFTER THE KEY WORD IS FORMED BY THE GEN ROUTINE, THE
* APPROPRIATE GENERATOR KEY TABLE IS
* SEARCHED (BINARY SEARCH). IF THE PROPER KEY IS NOT FOUND,
* AN ERROR MESSAGE IS ISSUED.
* IF THE KEY IS FOUND, THE ROUTINE TRANSFERS TO THE ADDRESS
* SPECIFIED IN THE ADDRESS FIELD OF THE LOCATED KEY. THE PROPER
* CODE IS THEN PUT OUT.

REM

* THE APPLICABLE GENERATOR KEY TABLE IS ESTABLISHED AT THE
* BEGINNING OF THE COMPILE ROUTINE ACCORDING TO THE COLUMN
* I MODE OF THE STATEMENT.
* THE ASTACK ENTRIES FOR THE LEFT-HAND AND RIGHT-HAND OPERANDS
* ARE PLACED IN THE CELLS 'A' AND 'B' FOR USE BY THE CODE ROUTINE.
* ALSO, THE ASTACK POINTERS ARE PLACED IN THE CELLS 'A+1' AND 'B+1'.

SPACE 3

GEN STZ AUXFG1
SXA GENXIT,4 INDICATES ENTRY NOT FROM PLACE OR STORE
LDQ =0 ZERO THE MQ
REM
CAL ASTACK,1 B OPERAND ENTRY
PAI
LFI BEXTF+BINTF
TRA GEN8
SLL B
SXA B+1,1 SET CELLS FOR THE CODE ROUTINE
REM
RQL 34 SKIP TWO BITS FOR B-OPERAND REGISTER
ARS 33 MODE OF B OPERAND
LGR 3 BUILD KEY WORD
NZI AUXFG1 SKIP IF NOT FROM PLACE OR STORE ROUTINES
TXI GEN10,1,-2 BUMP ASTACK POINTER TO THE 'A' OPERAND
CAL AUXILX GET OP KEY FOR PLACE AND STORE SEQUENCES
LGR 12 FORM THE KEY WORD
TRA GEN11
REM
AUXILX VFD
REM 24/,7/AUXIL

GEN10 CAL ASTACK,1 GET THE 'A' OPERAND
PAI
SLW A SET CELLS FOR CODE ROUTINE
SXA A+1,1
REM
ARS 19 GET OPERATOR FLAG INTO LOW-ORDER BIT
LBT IS IT AN OPERATOR.
TRA GEN1 NO
<table>
<thead>
<tr>
<th>REM</th>
<th>**1,1,1</th>
<th>YES. (MUST BE UNARY MINUS CODE SEQUENCE)</th>
<th>F078020</th>
</tr>
</thead>
<tbody>
<tr>
<td>REM</td>
<td>RESTORE THE ASTACK POINTER</td>
<td>F078040</td>
<td></td>
</tr>
<tr>
<td>SIZ</td>
<td>A+1</td>
<td>DELETE A-OPERAND INDICATIONS</td>
<td>F078050</td>
</tr>
<tr>
<td>RQL</td>
<td>31</td>
<td>ZERO OUT THE A OPERAND BITS</td>
<td>F078060</td>
</tr>
<tr>
<td>CAL</td>
<td>ASTACK,1</td>
<td>GET THE OPERATOR</td>
<td>F078070</td>
</tr>
<tr>
<td>GEN2</td>
<td>ARS</td>
<td>OPERATOR KEY</td>
<td>F078080</td>
</tr>
<tr>
<td>LGR</td>
<td>7</td>
<td>BUILD KEY WORD</td>
<td>F078090</td>
</tr>
<tr>
<td>GEN1</td>
<td>STQ</td>
<td>REGIST</td>
<td>F078110</td>
</tr>
<tr>
<td>NRT</td>
<td>GEN6</td>
<td>TEMPORARY SAVE</td>
<td>F078120</td>
</tr>
<tr>
<td>REM</td>
<td>GEN5</td>
<td>SKIP IF NOTHING IN A REGISTER CURRENTLY</td>
<td>F078130</td>
</tr>
<tr>
<td>TRA</td>
<td>REGIST+1</td>
<td>GET STACK POINTER FOR OPERAND IN REGISTER</td>
<td>F078140</td>
</tr>
<tr>
<td>LAS</td>
<td>B+1</td>
<td>IS IT SAME AS B-OPERAND.</td>
<td>F078150</td>
</tr>
<tr>
<td>GEN11</td>
<td>TRA</td>
<td>GEN3</td>
<td>F078160</td>
</tr>
<tr>
<td>GEN6</td>
<td>GEN5</td>
<td>SAME AS A-OPERAND.</td>
<td>F078170</td>
</tr>
<tr>
<td>TZE</td>
<td>GEN4</td>
<td>YES</td>
<td>F078180</td>
</tr>
<tr>
<td>REM</td>
<td>STOSAV</td>
<td>SAVE THE GEN6 WORD TEMPORARILY</td>
<td>F078190</td>
</tr>
<tr>
<td>STQ</td>
<td>STORE+4</td>
<td>MUST PUT REGISTER RESULT IN WORKING STORE</td>
<td>F078200</td>
</tr>
<tr>
<td>CAL</td>
<td>STOSAV</td>
<td>RESET THE GEN6 WORD FOR THE ORIGINAL CODE</td>
<td>F078210</td>
</tr>
<tr>
<td>SLW</td>
<td>GEN6</td>
<td>FORM COMPETED KEY WORD</td>
<td>F078220</td>
</tr>
<tr>
<td>GEN3</td>
<td>GEN5</td>
<td>GET CODE FOR REGISTER</td>
<td>F078230</td>
</tr>
<tr>
<td>REM</td>
<td>REGIST</td>
<td>GET CODE FOR REGISTER</td>
<td>F078240</td>
</tr>
<tr>
<td>ALX</td>
<td>19</td>
<td>MOVE INTO POSITION FOR B-OPERAND KEY</td>
<td>F078250</td>
</tr>
<tr>
<td>TRA</td>
<td>**3</td>
<td>F078260</td>
<td></td>
</tr>
<tr>
<td>GEN4</td>
<td>GEN6</td>
<td>MOVE INTO POSITION FOR A-OPERAND KEY</td>
<td>F078270</td>
</tr>
<tr>
<td>ALX</td>
<td>24</td>
<td>F078280</td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>GEN6</td>
<td>FORMED</td>
<td>F078290</td>
</tr>
<tr>
<td>GEN5</td>
<td>SXA</td>
<td>BEXIT+1</td>
<td>F078300</td>
</tr>
<tr>
<td>BEXIT</td>
<td>AXT</td>
<td>**4</td>
<td>F078310</td>
</tr>
<tr>
<td>AXT</td>
<td>**4</td>
<td>GET POWER OF 2 LENGTH OF TABLE</td>
<td>F078320</td>
</tr>
<tr>
<td>BPOWER</td>
<td>AXI</td>
<td>**1</td>
<td>F078330</td>
</tr>
<tr>
<td>REM</td>
<td>AXT</td>
<td>INITIALIZE LOWER TABLE POINTER</td>
<td>F078340</td>
</tr>
<tr>
<td>REM</td>
<td>AXT</td>
<td>INITIALIZE LOWER TABLE POINTER</td>
<td>F078350</td>
</tr>
<tr>
<td>REM</td>
<td>BAX</td>
<td>BR,4,**</td>
<td>F078360</td>
</tr>
<tr>
<td>B7</td>
<td>BXH</td>
<td>BR,4,**</td>
<td>F078370</td>
</tr>
<tr>
<td>CAL</td>
<td>=07777777000000</td>
<td>SKIP IF OUTSIDE THE TABLE</td>
<td>F078380</td>
</tr>
<tr>
<td>BAND</td>
<td>ANA</td>
<td>**4</td>
<td>F078390</td>
</tr>
<tr>
<td>REM</td>
<td>GEN6</td>
<td>AND WITH A WORD FROM THE TABLE</td>
<td>F078400</td>
</tr>
<tr>
<td>LAS</td>
<td>GEN6</td>
<td>F078410</td>
<td></td>
</tr>
<tr>
<td>TIX</td>
<td>BLOWER,1,1</td>
<td>MASK</td>
<td>F078420</td>
</tr>
<tr>
<td>TRA</td>
<td>**2</td>
<td>F078430</td>
<td></td>
</tr>
<tr>
<td>GEN1</td>
<td>TIX</td>
<td>BLOWER,1,1</td>
<td>F078440</td>
</tr>
<tr>
<td>GEN1</td>
<td>TIX</td>
<td>BLOWER,1,1</td>
<td>F078450</td>
</tr>
<tr>
<td>TRA</td>
<td>**2</td>
<td>ELEMENT ABOVE KEY. LOWER POINTER</td>
<td>F078460</td>
</tr>
<tr>
<td>B</td>
<td>TIX</td>
<td>BLOWER,1,1</td>
<td>F078470</td>
</tr>
<tr>
<td>REM</td>
<td>BEXIT</td>
<td>**4</td>
<td>F078480</td>
</tr>
<tr>
<td>BEXIT</td>
<td>AXT</td>
<td>**1</td>
<td>F078490</td>
</tr>
<tr>
<td>ERA</td>
<td>GEN6</td>
<td>RESTORE XRI</td>
<td>F078500</td>
</tr>
<tr>
<td>GEN98</td>
<td>TZE</td>
<td>**4</td>
<td>F078510</td>
</tr>
<tr>
<td>REM</td>
<td>TZE</td>
<td>TEST TO SEE IF EQUALITY FOUND</td>
<td>F078520</td>
</tr>
<tr>
<td>REM</td>
<td>TZE</td>
<td>IF EQUALITY FOUND, GO PUT OUT THE CODE.</td>
<td>F078530</td>
</tr>
<tr>
<td>REM</td>
<td>TZE</td>
<td>DEPENDS ON GENERATOR KEYS HAVING ZERO TAGS</td>
<td>F078540</td>
</tr>
<tr>
<td>REM</td>
<td>ERROR</td>
<td>7SKEND</td>
<td>F078550</td>
</tr>
<tr>
<td>REM</td>
<td>ERROR</td>
<td>NOT EQUAL. ERROR</td>
<td>F078560</td>
</tr>
<tr>
<td>GENXII</td>
<td>AXT</td>
<td>**4</td>
<td>F078570</td>
</tr>
<tr>
<td>REM</td>
<td>AXT</td>
<td>RETURN</td>
<td>F078580</td>
</tr>
<tr>
<td>GEN1</td>
<td>AXT</td>
<td>RETURN</td>
<td>F078590</td>
</tr>
<tr>
<td>RQL</td>
<td>34</td>
<td>BUILD THE KEY WORD</td>
<td>F078600</td>
</tr>
<tr>
<td>LGR</td>
<td>3</td>
<td>ZERO OUT TWO BITS</td>
<td>F078610</td>
</tr>
</tbody>
</table>
GEn9
CAL ASTACK=1,1 GET OPERATOR
LFT BEXITF+BINTF FLAGS MUST BE OFF
TRA **2
IRA GEN2
REM
GEn8
PIA REGAIN THE ASTACK WORD
REM
PAX ,4 POINTER
RFTool0000 IS IT SUBSCRIPTED.
CAL POOL-1,4 YES. GET WORD WITH EQUIV POINTER IN IT
PAX ,4 GET EQUIV POINTER FOR SURE THIS TIME
CAL SYMTAB,4 GET BCD NAME OF OFFENDING SYMBOL
SLW FORM69 PUT INTO ERROR MESSAGE
REM
ERROR 69,8END
REM
REM
PZE KEYTB3+LKYTB3,,LKYTB3 COMPLEX ARITHMETIC KEYS
PZE KEYTB2+LKYTB2,,LKYTB2 DOUBLE PRECISION KEYS
PZE KEYTB1+LKYTB1,,LKYTB1 BOOLEAN KEYS
REM
GEndE PZE KEYTB0+LKYTB0,,LKYTB0 REAL AND INTEGER KEYS
REM
BSS 4 HOLDS POWER-OF-TWO LENGTHS OF THE KEY
REM TABLES IN DECREMENT, AND STARTING VALUE
REM OF RAISE-LOWER TABLE POINTER IN ADDRESS.
GenBIN EQU *-1 SET DURING PRE-INITIALIZATION.
REM
SPACE 2
* LISTS USED IN THE BINARY SEARCH OVER THE KEY TABLES
REM
TXI BA,4,16384 TABLE FOR LOWERING THE SEARCH POINT.
TXI BA,4,8192
TXI BA,4,4096
TXI BA,4,2048
TXI BA,4,1024
TXI BA,4,512
TXI BA,4,256
TXI BA,4,128
TXI BA,4,64
TXI BA,4,32
TXI BA,4,16
TXI BA,4,8
TXI BA,4,4
TXI BA,4,2
TXI BA,4,1
GLower XEC *+1
REM
TXI BA,4,-16384 TABLE FOR Raising THE SEARCH POINT
TXI BA,4,-8192
TXI BA,4,-4096
TXI BA,4,-2048
TXI BA,4,-1024
TXI BA,4,-512
TXI BA,4,-256
TXI BA,4,-128
TXI BA,4,-64
TXI BA,4,-32
TXI BA,4,-16
TXI BA,4,-8
TXI BA,4,-4
TXI BA,4,-2
TXI BA,4,-1
BRaise XEC *+1
<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPEN</td>
<td>FUNCTION KEY GENERATOR</td>
<td>FT079220</td>
</tr>
<tr>
<td>RTN</td>
<td>ROUTINE TO GENERATE THE CODE KEY WORD FOR OPEN</td>
<td>FT079230</td>
</tr>
<tr>
<td>(BUILT-IN)</td>
<td></td>
<td>FT079240</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FT079250</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FT079260</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FT079270</td>
</tr>
<tr>
<td>SXA</td>
<td>OPN1, 4</td>
<td>SAVE RETURN POINT</td>
</tr>
<tr>
<td>LDQ</td>
<td>=0</td>
<td>ZERO THE MQ</td>
</tr>
<tr>
<td>CAL</td>
<td>ASTACK, 1</td>
<td>GET OPERAND</td>
</tr>
<tr>
<td>PAI</td>
<td></td>
<td>SAVE THE ASTACK WORD</td>
</tr>
<tr>
<td>LFI</td>
<td>BEXTF+BINTF</td>
<td>FLAGS MUST BE OFF</td>
</tr>
<tr>
<td>TRA</td>
<td>GEN8</td>
<td></td>
</tr>
<tr>
<td>SLW</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>SXA</td>
<td>B+1, 1</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ARS</td>
<td>33</td>
<td></td>
</tr>
<tr>
<td>LGR</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>TXI</td>
<td>**1, 1,-2</td>
<td>BUMP ASTACK POINTER TO NEXT ITEM</td>
</tr>
<tr>
<td>CAL</td>
<td>ASTACK, 1</td>
<td>IS IT THE FUNCTION CODE YET.</td>
</tr>
<tr>
<td>PBT</td>
<td></td>
<td>NO</td>
</tr>
<tr>
<td>RQ1</td>
<td>31</td>
<td></td>
</tr>
<tr>
<td>STZ</td>
<td>A+1</td>
<td>DELETE ANY A-OPERAND INDICATIONS</td>
</tr>
<tr>
<td>TRA</td>
<td>OPN1</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>PAI</td>
<td>SAVE THE ASTACK WORD</td>
</tr>
<tr>
<td>SLW</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>SXA</td>
<td>A+1, 1</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>ARS</td>
<td>33</td>
</tr>
<tr>
<td>RQ1</td>
<td>34</td>
<td></td>
</tr>
<tr>
<td>LGK</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>LXA</td>
<td>PNFLAG, 4</td>
<td>GET KEY WORD AND LOSE HIGH-ORDER BIT</td>
</tr>
<tr>
<td>CLA</td>
<td>ASTACK, 4</td>
<td>GET KEY WORD AND LOSE HIGH-ORDER BIT</td>
</tr>
<tr>
<td>LAS</td>
<td>=1586</td>
<td>IS IT MAX OR MIN FUNCTION.</td>
</tr>
<tr>
<td>TXI</td>
<td>NOTMAX, 1,-2</td>
<td>NO, RESTORE POINTER</td>
</tr>
<tr>
<td>REM</td>
<td>TRA</td>
<td>**1</td>
</tr>
<tr>
<td>ARS</td>
<td>30</td>
<td>MAX OR MIN FUNCTION</td>
</tr>
<tr>
<td>STA</td>
<td>MAXPTR</td>
<td>MOVE KEY NUMBER AND DIVIDE BY TWO</td>
</tr>
<tr>
<td>CAL</td>
<td>ASTACK+2, 1</td>
<td>SAVE IN DIRECTOR WORD</td>
</tr>
<tr>
<td>PBT</td>
<td></td>
<td>GET 'NEXT' ARGUMENT FROM ASTACK</td>
</tr>
<tr>
<td>TRA</td>
<td>**2</td>
<td>NOT THE OPERATION CODE</td>
</tr>
<tr>
<td>TXI</td>
<td>OPN1, 1,-2</td>
<td>NO MORE ARGUMENTS, GO SET EXIT POINT</td>
</tr>
<tr>
<td>REM</td>
<td>CAL</td>
<td>ASTACK, 4</td>
</tr>
<tr>
<td>AXC</td>
<td>OPN1, 4</td>
<td>GET RETURN POINT FOR MAX AND MIN CODE</td>
</tr>
<tr>
<td>TRA</td>
<td>OPN1+1</td>
<td>GO PUT OUT CODE</td>
</tr>
<tr>
<td></td>
<td>NOTMAX</td>
<td>ASTACK, 1</td>
</tr>
<tr>
<td>PBT</td>
<td>ERROR</td>
<td>7, SKEND</td>
</tr>
<tr>
<td></td>
<td>OPN1</td>
<td>**4, 4</td>
</tr>
<tr>
<td>SXT</td>
<td>GENXIT, 4</td>
<td>SET SO 'RESULT' ROUTINE EXITS PROPERLY</td>
</tr>
<tr>
<td>TRA</td>
<td>GEN9</td>
<td>GO PUT OUT CODE</td>
</tr>
</tbody>
</table>

The above code is a portion of a BASIC program designed to generate a key word for an OPEN routine. The program includes various instructions and labels for handling different aspects of the OPEN function, such as saving return points, checking function codes, and managing the ASTACK word.
TTL ARITHMETIC CODE GENERATORS

* SUBROUTINES CALLED BY THE GENERATORS

SPACE 3

REM ROUTINE TO OUTPUT THE SEQUENCE
REM UFA =0233000000000
REM FAD =0233000000000

INTXX SXA INTXX,4
NZZT FLAG11 HAS OCTAL 233000000000 BEEN ASSIGNED YET.
TSX SET.1,4 NO. GO ASSIGN THE CONSTANT
CAL UFA11
TSX CITBLD,4,C
CAL FAD11
TSX CITBLD,4,C

INTXX AXI **,4
TRA 1,4
SPACE 5

REM ROUTINE TO OUTPUT THE SEQUENCE
REM UFA =0233000000000
REM LRS
REM ANA =0377777
REM LLS
REM ALS 18

REM

XINTXX SXA XINTXX,4
NZZT FLAG11
TSX SET.1,4
NZZT FLAG12
TSX SET.2,4
CAL UFA11
TSX CITBLD,4,C
GODEN LRS
CAL ANA12
TSX CITBLD,4,C
GODEN LLS
GODEN ALS18

XINTXX AXI **,4
TRA 1,4
SPACE 5

REM ROUTINE TO OUTPUT THE SEQUENCE
REM LRS 18
REM ORA =0233000000000
REM FAD =0233000000000

REM

FLOXXX SXA FLOXX,4
NZZT FLAG11
TSX SET.1,4
CODEN LRS18
CAL ORA11
TSX CITBLD,4,C
CAL FAD11
TSX CITBLD,4,C

FLOXX AXI **,4
TRA 1,4
SPACE 6

REM ROUTINE TO ASSIGN EXPONENT TO STORAGE

FLOXX AXI **,4
TRA 1,4
SPACE 6

REM ROUTINE TO ASSIGN EXPONENT TO STORAGE

SIT1 SXA ESST1,4
CAL =0233000000000
TSX ASCON1,4 ASSIGN CONSTANT TO STORAGE
STA FADD11 SET THE ADDRESSES OF THE PERTINENT INSTRUCTIONS
STA UFA11
STA FLAG11 SET NONZERO FLAG

EXSET1 AXT **,4
TRA 1,4
REM
SET.1 EQU SET11
SPACE 6 ROUTINE TO ASSIGN 0377777 TO STORAGE
REM
SET2 SXA EXSET1,4 GET INTEGER MASK BITS
CAL =0377777 PUT INTO STORAGE
TSX ASCON1,4 PUT POINTER INTO INSTRUCTION
STA ANA12
STL FLAG12 SET NONZERO FLAG
TRA EXSET1
REM
SET.2 EQU SET12
SPACE 6 ROUTINE TO OUTPUT A CALL ENTRY TO AN EXTERNAL SUBROUTINE
REM
OUTPTV SXA OUTPTX,4 TSX OUTPTV,4 NAME OF SUBROUTINE
REM
OUTPTX AXT **,4
TRA 2,4
REM
SPACE 6 ROUTINE TO DETECT CASES LIKE I = 0 AND TURN OUT A STZ.
REM
STSC1 STL DVSTFG DOUBLE PRECISION ENTRY
TRA **2
REM
STSC1 STZ DVSTFG SINGLE PRECISION AND BOOLEAN ENTRY
LDI 8 GET STACK WORD
LNT 8GCNST IS THIS A CONSTANT
TRA 2,2 NO RETURN
REM
LX A 8,4 YES GET THE CONTAB POINTER
ZET CONTAB,4 IS CONSTANT ZERO.
TRA 2,2 NO RETURN
REM
ZET DVSTFG IS THIS DOUBLE-LENGTH.
TRA STSC1 YES
REM
CODE STZ,A NO CODE IS... STZ A
TRA 1,2 RETURN, SKIPPING OTHER CODE
REM
STSC1 CLA 1,2 GET THE CONTROL WORD
TMJ **,3 SKIP IF THIS IS THE D = I CASE
ZET CONTAB+1,4 IS SECOND PART ZERO.
TRA 2,2 NO RETURN
REM
FT080360 PAGE 214
* ROUTINE TO TURN OUT IN-LINE CODE
* FOR EXPONENTIATION SEQUENCES... R**2 THROUGH R**7.

REM

R EXP C1 LDI B  GET THE ASTACK WORD FOR THE EXPONENT
   LNT BCNST  IS IT A CONSTANT
   TRA 1,4  NO
   REM
   SXA EXREX,4  SAVE XR4
   LXA B,4  GET THE CONTAB POINTER
   CLA CONTAB,4  AND GET THE CONSTANT
   TMI EXREX  QUIT IF NEGATIVE (SHOULDN'T HAPPEN)
   PDC 4  SIZE OF EXPONENT COMPLEMENTED
   TIXL EXREX,4,-8  POWER GREATER THAN 7, OR ZERO
   TXH GENXIT,4,-2  R**1 NO CODE NECESSARY
   REM
   SXA REXP C4,4  SAVE EXPONENT (COMPLEMENTED)
   CAL REGIST  GET REGISTER LOCATION OF ARGUMENT
   ANA =02  SAVE ONLY THE MQ BIT, IF ANY
   SLW T1  SAVE
   TIXL STORE,4  STORE OFF THE ARGUMENT
   CAL ASTACK,1  AND RESET THE 'A' WORD AGAIN
   SLW A  (ASTACK MAY HAVE BEEN CHANGED BY STORE)
   REM
   SET T1  WAS ARGUMENT IN THE MQ.
   TRA **3  YES. DON'T LOAD THE MQ AGAIN
   CODE LDQ,A  SQUARE THE NUMBER
   CODE FMP,A  SQUARE THE NUMBER
   REM
   R EXP C4 AXT **,4  REGAIN THE EXPONENT (COMPLEMENTED)
   TXH REXP C9,4,-3  IS IT POWER 2.
   CODEN XCA  NO CONTINUE
   REM
   LXA REXP C4,4  REGAIN EXPONENT
   TRA **,-2,4  BRANCH
   TRA REXP3
   TRA REXP4
   TRA REXP5
   TRA REXP6
   REM
   REXP7 CODE FMP,A  R**7
   CODEN XCA
   REM
   REXP5 CODEN STQ777  R**5
   STL T1  SET AN AVAILABLE FLAG CELL
   TRA REXP6+1  SKIP
   REM
   REXP6 STL T1  R**6 SET FLAG CELL
   CODE FMP,A  XCA
   CODEN XCA
   ZET T1  IS THIS POWER 6.
   TRA **3  NO
   REM
   REXP4 CODEN STQ777  R**4
   CODEN FMP777
   REM
   REXP3 CODE FMP,A  R**3
<table>
<thead>
<tr>
<th>RESULT</th>
<th>REAL, AC, ROUND</th>
<th>FT081560</th>
</tr>
</thead>
<tbody>
<tr>
<td>REM</td>
<td></td>
<td>FT081570</td>
</tr>
<tr>
<td>XREX</td>
<td>AXT **, 4</td>
<td>FT081580</td>
</tr>
<tr>
<td></td>
<td>RESTORE XR4 AND RETURN</td>
<td>FT081590</td>
</tr>
<tr>
<td>TRA</td>
<td>2, 4</td>
<td></td>
</tr>
</tbody>
</table>
EJECT

- CODE GENERATORS -

REM

CODE CLA,B
REM

ABS01 ABSF(B)
ABS02 SSP
RESULT REAL,AC
ABS03 XCA
TRA ABS02
REM

XABS01 CLA,B
XABS02 SSP
RESULT INTEGR,AC
XABS03 XCA
TRA XABS02
REM

XABS01 INF(B)
XABS02 INTXXX,4
RESULT REAL,AC
XABS03 XCA
TRA INT02
REM

INT01 CLA,B
INT02 INTXXX,4
RESULT REAL,AC
INT03 XCA
TRA INT02
REM

INT01 INF(B)
INT02 INTXXX,4
RESULT REAL,AC
INT03 XCA
TRA INT02
REM

FLOAT01 CLA,B
FLOAT02 INTXXX,4
RESULT REAL,AC
FLOAT03 XCA
TRA FLOAT02
REM

FLOAT01 FIXF(A)
FLOAT02 INT01
FLOAT03 INT02
REM

XFIX01 XFIX02 XFIX03
SPACE 6
LDI B XLOCFI OR XLOCF(R)
LFT $VARB+BARGT MUST BE ONE OR THE OTHER
TRA ++2

ERROR 7,SKEND
REM

RT 700000 IS IT SUBSCRIPTED
TRA XLOC0A
YES.
XLOC0B CODE AXT4,B
TRA XLOC0C
REM

XLOC0A LXA B,4 SEE IF SUBSCRIPT IS ONLY A CONSTANT ADDEND
CAL POOL,4
ORA POOL-2,4
PBT
TRA XLOC0B
REM

CODE PXA4,B
CAL  PCOUNT
SUB  =1
CODED SUB,P
CODEN PAX4

XLOCOC CODED PXD4
RESULT INTEGRAC

SPACE 6

MXMIN1 CODE CLA,A CODE GENERATORS FOR ALL MAX AND MIN'S
MXMIN4 CODE LDQ,B
MXMINA CODEP TLQ2
CODEN XCA
REM

CLA ASTACK,1 ROUTING SEQUENCE
MXPTR **,4 GET PREVIOUSLY SET DIRECTOR
TPL MAXINT,4 IF NOT KEY WORD, GO TO INTERMEDIATE CODE
CAL XCA PREPARE FOR POSSIBLE CODE SEQUENCE
TRA MAXTRM,4 GO TO TERMINAL SEQUENCE
REM

MXMIN2 CODE LDQ,A
TRA MXMINA
MXMIN3 CODE CLA,A
TRA MXMINA
MXMIN5 CODE CLA,B
TRA MXMINA
REM

MAXINT EQU *
RESULT REAL, MQ MIN1
RESULT INTEGR, MQ MIN0
RESULT INTEGRAC MAX0
RESULT REAL, AC MAX0
RESULT REALAC MAX1
RESULT INTEGR, MQ XMIN0
RESULT REAL, MQ XMIN1
RESULT REALAC XMAX1
RESULT INTEGRAC XMAX0
REM

MAXTRM EQU *
TRA SIGNOC MIN1
TSX CIBLD,4,N MINO. PUT OUT XCA
TSX FLOXXX,4 MAX0. FLOAT THE INTEGER
RESULT REAL, AC MAX1
RESULT INTEGR, MQ XMIN0
TSX CIBLD,4,N XMIN1. PUT OUT XCA
TSX XINTX,4 XMAX1. FIX THE REAL RESULT
RESULT INTEGRAC XMAX0
SPACE 6

XSIGN1 CODE CLA,A XSIGNF1,J
XSIGN4 CODE LDQ,B
XSIGNA CODEN LLS
RESULT INTEGRAC

XSIGN2 CODE LDQ,A
XSIGNB CODEN LRS
RESULT INTEGR, MQ

XSIGN3 CODE CLA,A
TRA XSIGNA
XSIGN5 CODE CLA,B
TRA XSIGNB
SPACE 2

SIGN01 CODE CLA,A SIGNF(A,B)
SIGN04 CODE LDQ,B
SIGN0A CODEN LLS
CODE

MODA1
CODE FAD, A

MODA3
CODE TNZ2

MOD01
EQU MOD02

MOD02
EQU MOD02

MOD04
EQU MOD02

MOD05
EQU MOD02

SPACE 2

REM

IEQ101
TSX STZ TST, 2

PZE IEQIA

CAL A SKIP OUT IF A = B

ERR B

TZE IEQIB

CODE CLA, B

IEQ102
CODE STO, A

IEQIA

LDI A

TSX INDXT, 4 TEST FOR VALID CHANGE OF INDEX OR SUBSCRIPT

IEQ10
TXL CMPLX, 1, 0

ERROR 84, SKEND

REM

IEQ103
CODE STQ, A

TRA IEQIA

REM

IADD01
CODE CLA, A

IADD04
CODE ADD, B

RESULT INTEG, AC

IADD02
CODE ADD, B

RESULT INTEGR, AC

IADD03
CODE XCA

TRA IADD02

IADD05
CODE XCA

TRA IADD04

REM

ISUB01
CODE CLA, A

ISUB04
CODE SUB, B

RESULT INTEGR, AC

ISUB02
CODE CHS

CODE ADD, A

RESULT INTEGR, AC

ISUB03
CODE XCA

TRA ISUB02

ISUB05
CODE XCA

TRA ISUB04

REM

IMPY01
CODE LDQ, A

IMPY05
CODE MPY, B

IMPYA
CODE ALS17

RESULT INTEGR, AC

IMPY03
CODE MPY, A

TRA IMPYA

IMPY02
CODE XCA

TRA IMPY03

IMPY04
CODE XCA

TRA IMPY05
<table>
<thead>
<tr>
<th>CODE</th>
<th>LABEL</th>
<th>REMARK</th>
<th>FUNCTION</th>
<th>REGISTERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>RADD01</td>
<td>CODE</td>
<td>REM</td>
<td>R+R</td>
<td>CLA, A</td>
</tr>
<tr>
<td>RADD04</td>
<td>CODE</td>
<td>REM</td>
<td>R+R</td>
<td>FAD, B</td>
</tr>
<tr>
<td>RESULT</td>
<td>REAL, AC, ROUND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RADD02</td>
<td>CODE</td>
<td>REM</td>
<td>R+R</td>
<td>FAD, A</td>
</tr>
<tr>
<td>RESULT</td>
<td>REAL, AC, ROUND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RADD03</td>
<td>CODE</td>
<td>REM</td>
<td>R+R</td>
<td>XCA</td>
</tr>
<tr>
<td>TRA</td>
<td>RADD02</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RADD05</td>
<td>CODE</td>
<td>REM</td>
<td>R+R</td>
<td>XCA</td>
</tr>
<tr>
<td>TRA</td>
<td>RADD04</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RSUB01</td>
<td>CODE</td>
<td>REM</td>
<td>R-R</td>
<td>CLA, A</td>
</tr>
<tr>
<td>RSUB04</td>
<td>CODE</td>
<td>REM</td>
<td>R-R</td>
<td>FSB, B</td>
</tr>
<tr>
<td>RESULT</td>
<td>REAL, AC, ROUND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RSUB02</td>
<td>CODE</td>
<td>REM</td>
<td>R-R</td>
<td>CHS</td>
</tr>
<tr>
<td>TRA</td>
<td>RADD02</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RSUB03</td>
<td>CODE</td>
<td>REM</td>
<td>R-R</td>
<td>XCA</td>
</tr>
<tr>
<td>TRA</td>
<td>RSUB02</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RSUB05</td>
<td>CODE</td>
<td>REM</td>
<td>R-R</td>
<td>XCA</td>
</tr>
<tr>
<td>TRA</td>
<td>RSUB04</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RNEG01</td>
<td>CODE</td>
<td>REM</td>
<td>-R</td>
<td>CLS, B</td>
</tr>
<tr>
<td>RESULT</td>
<td>REAL, AC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RNEG03</td>
<td>CODE</td>
<td>REM</td>
<td>-R</td>
<td>XCA</td>
</tr>
<tr>
<td>RESULT</td>
<td>REAL, AC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RNEG02</td>
<td>CODE</td>
<td>REM</td>
<td>-R</td>
<td>CHS</td>
</tr>
<tr>
<td>RESULT</td>
<td>REAL, AC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RMPY01</td>
<td>CODE</td>
<td>REM</td>
<td>R*R</td>
<td>LDG, A</td>
</tr>
<tr>
<td>RMPY05</td>
<td>CODE</td>
<td>REM</td>
<td>R*R</td>
<td>FMP, B</td>
</tr>
<tr>
<td>RESULT</td>
<td>REAL, AC, ROUND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RMPY02</td>
<td>CODE</td>
<td>REM</td>
<td>R*R</td>
<td>XCA</td>
</tr>
<tr>
<td>RMPY03</td>
<td>CODE</td>
<td>REM</td>
<td>R*R</td>
<td>FMP, A</td>
</tr>
<tr>
<td>RESULT</td>
<td>REAL, AC, ROUND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RMPY04</td>
<td>CODE</td>
<td>REM</td>
<td>R*R</td>
<td>XCA</td>
</tr>
<tr>
<td>TRA</td>
<td>RMPY05</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RDIV01</td>
<td>CODE</td>
<td>REM</td>
<td>R/R</td>
<td>CLA, A</td>
</tr>
<tr>
<td>RDIV04</td>
<td>CODE</td>
<td>REM</td>
<td>R/R</td>
<td>FDP, B</td>
</tr>
<tr>
<td>RESULT</td>
<td>REAL, MQ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RDIV02</td>
<td>CODE</td>
<td>REM</td>
<td>R/R</td>
<td>STO, 777</td>
</tr>
<tr>
<td>RDIVA</td>
<td>CODE</td>
<td>REM</td>
<td>R/R</td>
<td>CLA, A</td>
</tr>
<tr>
<td>CODEN</td>
<td>FDP, 777</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESULT</td>
<td>REAL, MQ</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RDIV03</td>
<td>CODE</td>
<td>REM</td>
<td>R/R</td>
<td>STO, 777</td>
</tr>
<tr>
<td>TRA</td>
<td>RDIVA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RDIV05</td>
<td>CODE</td>
<td>REM</td>
<td>R/R</td>
<td>XCA</td>
</tr>
<tr>
<td>TRA</td>
<td>RDIV04</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REXP01</td>
<td>CODE</td>
<td>REM</td>
<td>R**R</td>
<td>CLA, A</td>
</tr>
<tr>
<td>REXP04</td>
<td>CODE</td>
<td>REM</td>
<td>R**R</td>
<td>LDG, B</td>
</tr>
<tr>
<td>REXP A</td>
<td>CODE</td>
<td>REM</td>
<td>R**R</td>
<td>TSX</td>
</tr>
<tr>
<td>OUTPTV, 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RESULT</td>
<td>REAL, AC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REXP02</td>
<td>CODE</td>
<td>REM</td>
<td>R**R</td>
<td>LDG, A</td>
</tr>
<tr>
<td>CODEN</td>
<td>XCA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FNAME</td>
<td>RNAME</td>
<td>FNAME</td>
<td>RNAME</td>
<td></td>
</tr>
</tbody>
</table>
RESULT

B * B
RESULT
REAL,AC

BANDO1 CODE CAL,B
BANDO2 CODE ANA,A
BANDO4 CODE ANA,B
SPACE 2

ARITH, STATE, FN... B = B

ASFR5 TRA ASFEND

SPACE 8

DABS01 COOLD B DABSFA)
DABS02 CODEN SSP
CODEN LRS

CA8SOA CODEN S0777
CODEN STQ776
RESULT REAL, PAC

DABS03 CODEN CLA775
CODEN LDQ774
TRA DABS02

SPACE 2

DFLOAT1 CODE CLA,B DFLOATF(I)

DFLOAT2 TSX FLOXXX,4
IRA DABS0A

DFLOAT3 CODEN XCA
TRA DFLOAT2

SPACE 2

DFIX01 STR MSGFIX,,SKEND ***PUT OUT COMPLAINT AGAINST IBM

DFIX02 EQU DFIX01
DFIX03 EQU DFIX01

6

MSGFIX BCI 6, IBM'S 'FIXF' GENERATES NONSENSE C0DF

PAGE 224
DEXPR1 CODLAC  A
DEXPR4 CODLMQ  B
DEXPR6 TSX   OUTPV,4

RESULT  DEXPI3
         REAL,AC
DEXPR5 TSX   MQ,AC,4
TRA    DEXPR4
DEXPR2 TSX   AC,MQ,4
DEXPR3 CODLAC  A
TRA    DEXPRA
SPACE   2
DEXPI1 CODLAC  A
         D ** I
DEXPI4 CODE   LDQ,B
DEXPIA TSX   OUTPV,4
         DEXPI2

RESULT  REAL,AC
DEXPI5 TSX   MQ,AC,4
TRA    DEXPI4
DEXPI2 CODEN  ST0775
DEXPI8 CODLAC  A
CODEN   LDQ775
TRA    DEXPIA
DEXPI3 CODEN  ST0775
TRA    DEXPI8
SPACE   2
DEQD01 TSX   STZTSO,2
         D = D
PZE    IEQIB
CODLD  B
DEQD0A CODSTO  A
TRA  CMPLY
DEQD02 ZET   REGFLG
TRA  DEQD0A
CODEN  CLA777
CODEN   LDQ776
TRA  DEQD0A
DEQD03 ZET   REGFLG
TRA  DEQD0A
CODEN  CLA775
CODEN   LDQ774
TRA  DEQD0A
SPACE   2
DEQI01 TSX   STZTSO,2
         D = I
MZE    IEQIB
CODE   CLA,B
DEQI02 TSX   FLOXXX,4
TRA  DEQD0A
DEQI03 CODEN  XCA
TRA  DEQI02
SPACE   2
I EQD01 EQU  IEQRO1
         I = D
I EQD0A EQU  IEQRO2
I EQD02 ZET   REGFLG
TRA  IEQD0A
CODEN  CLA777
TRA  IEQD0A
I EQD03 ZET   REGFLG
TRA  IEQD0A
CODEN  CLA775
TRA  IEQD0A
<table>
<thead>
<tr>
<th>SPACE</th>
<th>ASFD0</th>
<th>CODLAC</th>
<th>ASFD1</th>
<th>ARITH, STATE, FN...</th>
<th>D = D</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>B</td>
<td>ASFEND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRA</td>
<td></td>
<td>ASFD2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ISX</td>
<td>MQ,AC,4</td>
<td>ASFD3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRA</td>
<td>ASFEND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPACE</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASFEND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CODE</td>
<td>CLA,B</td>
<td>ASFD1</td>
<td>ARITH, STATE, FN...</td>
<td>D = 1</td>
</tr>
<tr>
<td></td>
<td>CODEN</td>
<td>ST0777,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CODEN</td>
<td>STQ776,</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRA</td>
<td>ASFEND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPACE</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASFEND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRA</td>
<td>ASFD12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPACE</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASFEND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRA</td>
<td>ASFD1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPACE</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASFEND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRA</td>
<td>ASFD1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPACE</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASFEND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRA</td>
<td>ASFD1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPACE</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASFEND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRA</td>
<td>ASFD1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPACE</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASFEND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRA</td>
<td>ASFD1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPACE</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASFEND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRA</td>
<td>ASFD1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPACE</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASFEND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRA</td>
<td>ASFD1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPACE</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASFEND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRA</td>
<td>ASFD1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPACE</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASFEND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRA</td>
<td>ASFD1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPACE</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASFEND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRA</td>
<td>ASFD1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPACE</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASFEND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRA</td>
<td>ASFD1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPACE</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASFEND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRA</td>
<td>ASFD1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPACE</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASFEND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRA</td>
<td>ASFD1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPACE</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASFEND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRA</td>
<td>ASFD1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPACE</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>ASFEND</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TRA</td>
<td>ASFD1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPACE</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
* EJCT
* THE FOLLOWING ARE SOME ABYSMAL TEMPORARY CODE SEQUENCES FOR
* C + C     DSIGNF    ISIGNF
* C - C     DSIGNF    ISIGNF
REM
CADD01 CODLAC A C + C
CADD1 CODLD B
CADD0A CODDEN STQ774
CODDEN FAD777
CADD0B CODDEN STQ777
CODDEN CLA776
CADD0D CODDEN FAD774
CADD0E CODDEN STQ776
RESULT REAL,AC
CADD02 CODLD A
TRA CADD0A
CADD03 CODLD A
CADD0C CODDEN STQ776
CODDEN FAD775
TRA CADD0B
CADD05 CODLD B
TRA CADD0C SPACE 6
CSUB01 GODSUB B C - C
TRA CADD02
CADD02 CODLD A
CODDEN STQ774
CODDEN FSB777
CODDEN STQ777
CODDEN CLS776
TRA CADD0D
CADD03 CODLD A
CODDEN STQ776
CODDEN FSB775
CODDEN STQ777
CODDEN CLS774
CODDEN FAD776
TRA CADD0D
CADD04 CODLD B
CODDEN STQ774
CODDEN CHS
CODDEN FAD777
TRA CSUB0A
CSUB05 CODLD B
CODDEN STQ776
CODDEN CHS
CODDEN FAD775
TRA CSUB0B SPACE 6
DSIGN1 CODLAC A
DSIGN4 CODLD B
CODDEN STQ775
CODDEN CLA776
DSIGNB CODDEN LLS
CODDEN STQ776
CODDEN CLA777
CODDEN LDQ775
CODDEN LLS
CODDEN STQ777
RESULT REAL,AC
DSIGN2 CODLD A DSIGNF(A,B) AND ISIGNF(A,B)
<table>
<thead>
<tr>
<th>CODEN</th>
<th>ST0774</th>
<th>FT088450</th>
</tr>
</thead>
<tbody>
<tr>
<td>CODEN</td>
<td>LDQ777</td>
<td>FT088460</td>
</tr>
<tr>
<td>CODEN</td>
<td>LLS</td>
<td>FT088470</td>
</tr>
<tr>
<td>CODEN</td>
<td>ST0777</td>
<td>FT088480</td>
</tr>
<tr>
<td>CODEN</td>
<td>CLA774</td>
<td>FT088490</td>
</tr>
<tr>
<td>CODEN</td>
<td>LDQ776</td>
<td>FT088500</td>
</tr>
<tr>
<td>DSIGNA</td>
<td>CODEN</td>
<td>LLS</td>
</tr>
<tr>
<td>TRA</td>
<td>CADDOE</td>
<td>FT088520</td>
</tr>
<tr>
<td>DSIGN3</td>
<td>COOLD</td>
<td>A</td>
</tr>
<tr>
<td>CODEN</td>
<td>STQ776</td>
<td>FT088540</td>
</tr>
<tr>
<td>CODEN</td>
<td>LDQ775</td>
<td>FT088550</td>
</tr>
<tr>
<td>CODEN</td>
<td>LLS</td>
<td>FT088560</td>
</tr>
<tr>
<td>CODEN</td>
<td>ST0777</td>
<td>FT088570</td>
</tr>
<tr>
<td>CODEN</td>
<td>CLA776</td>
<td>FT088580</td>
</tr>
<tr>
<td>CODEN</td>
<td>LDQ774</td>
<td>FT088590</td>
</tr>
<tr>
<td>TRA</td>
<td>DSIGNA</td>
<td>FT088600</td>
</tr>
<tr>
<td>DSIGN5</td>
<td>COOLD</td>
<td>B</td>
</tr>
<tr>
<td>CODEN</td>
<td>ST0777</td>
<td>FT088620</td>
</tr>
<tr>
<td>CODEN</td>
<td>CLA774</td>
<td>FT088630</td>
</tr>
<tr>
<td>TRA</td>
<td>DSIGNB</td>
<td>FT088640</td>
</tr>
</tbody>
</table>
ITL ARITHMETIC UTILITY PROGRAM - CODE EFT088650

* CODE ROUTINE.
* REM
* THIS ROUTINE IS ENTERED DURING PROCESSING OF ARITHMETIC
* STATEMENTS VIA COMPIL, GEN, AND ULTIMATELY THE CODE
* GENERATORS. ONE OR MORE OBJECT INSTRUCTIONS ARE GENERATED.
* REM
* THE ENTRY IS...
* TSX CODE$4
* PZE OP$STACK

WHERE 'OP' IS THE ADDRESS OF THE OPERATION CODE, AND STACK IS THE LOCATION OF A WORD CONTAINING A COPY OF THE APPROPRIATE ASSTACK

ENTRY. THE CORRESPONDING ASSTACK POINTER MUST BE IN THE ADDRESS FIELD OF STACK+1.

REM

THE ENTRY IN STACK MAY DESCRIBE A CONSTANT, VARIABLE,
ARITHMETIC STATEMENT FUNCTION ARGUMENT, OR AN INTERMEDIATE RESULT HELD IN WORKING STORAGE.

IF A VARIABLE IS A PARAMETER, PROLOG IS ENTERED.

IF THE TAG FIELD OF THE STACK WORD IS NON-ZERO, THE VARIABLE IS SUBSCRIPTED. THE ISTACK POINTER IS THEN THE ADDRESS FIELD OF STACK. CODE IS GENERATED TO LOAD XR4 WITH A PREVIOUSLY COMPUTED QUANTITY AND/OR SUPPLY A CONSTANT ADDEND TO CITBLD OR PROLOG.

REM

THIS ROUTINE IS EQUIPPED TO PROCESS TWO INSTRUCTIONS SUPPLIED BY THE DP AND CA CODE DRIVERS. THE FLAG SPCDFG IS THE ROUTING KEY FOR THIS MODE OF OPERATION.

REM

CODE
STZ SPCDFG
SXA EXCODE$4
SXA EXCODE+1,1
CAL 1,4
STA COD1
PDC 4
CAL 4
STZ C0D6
SIZ C0D4
REM
RFT 700000
TRA COD5
ANA =07777
ORA **
ORS XRFLAG
REM
LFT 8CNS3
TRA COD3
REM
RFI 40000
TRA COD2
REM
LFT BEXTF
TRA COD12
REM
LFT BARGT
TRA COD7
REM

PAGE 230
LFI BARIT IS IT AN ARITH. STATE. FUNCTION ARGUMENT
TRA COD14 YES FT089260
REM FT089270
STA COD31 SET EQUIV POINTER INTO SECOND INSTRUCTION FT089280
ZSD COD37 NO CONSTANT ADDEND HERE FT089290
TSX CITBLD,4,V FT089300
TRA COD47+1 FT089310
REM FT089320
COD2 ANA 077777777777 WORKING STORAGE. REMOVE HIGH-ORDER BIT
STA COD31 SAVE WORKING POINTER IN SECOND INSTRUCTION FT089330
TSX FREWrk,4 FREE UP THIS WORKING CELL FT089340
TSX CITBLD,4,W PUT OUT THE CODE FT089350
NZT SPDCFG TEST THE ROUTING FLAG FT089360
TRA EXCODE NO SECOND INSTRUCTION NEEDED FT089370
CAL COD31 GET THE SECOND INSTRUCTION FT089380
ACL =1 BUMP THE WORKING CELL POINTER FT089390
STA COD2+2 GO PUT OUT THIS CODE FT089400
REM FT089410
TRA COD2+2 GO PUT OUT THIS CODE FT089420
COD3 STA COD31 CONSTANT. SAVE POINTER IN 2ND INSTRUCTION FT089430
TSX CITBLD,4,C PUT OUT THE CODE FT089440
NZT SPDCFG TEST THE ROUTING FLAG FT089450
TRA EXCODE NO MORE TO DO FT089460
CAL COD31 GET THE SECOND INSTRUCTION FT089470
SUB =1 BUMP THE CONTAB POINTER DOWN BY ONE FT089480
TSX CITBLD,4,C PUT OUT THE CODE FT089490
REM FT089500
FRCME AXT **,4 FT089510
AXT **,1 FT089520
TRA 2,4 RETURN FT089530
REM FT089540
REM FT089550
REM FT089560
COD7 STA COD4 PROLOGUE REQUIRED. SAVE POINTER FT089570
ANA 077777 SAVE ONLY ADDRESS FT089580
ZSD COD40 NO CONSTANT ADDEND HERE FT089590
TRA COD42 FT089600
REM FT089610
REM FT089620
REM FT089630
REM FT089640
COD5 PAX 1,1 PROCESS SUBSCRIPTING FOR THIS VARIABLE FT089650
CLA POOL-2,1 GET ISTACK POINTER FT089660
ARS 18 ISTACK ENTRY FT089670
TMI COD8 GET POINTER FT089680
LDQ POOL,1 POINTER IS TO SYMTAB FT089690
TQP COD1U GETTHE FIRST ISTACK WORD FT089700
REM SKIP IF PLUS SIGN (ONLY CONSTANT ADDEND)
REM LXD4 OR IN THE OP CODE FT089710
ORM XRFLAG FT089720
TSX FREWrk,4 FREE UP THE WORKING CELL FT089730
TSX CITBLD,4,W CODE IS... LXD WORKING,4 FT089740
REM FT089750
REM FT089760
COD10 STL COD6 SET FLAG TO 'LOAD XR' STATUS FT089770
CLA POOL-2,1 GET THE CONSTANT ADDEND FROM ISTACK ENTRY FT089780
PAX 1,4 FT089790
SXD COD37,4 SET THE 'BUMPING' INSTRUCTIONS FT089800
SXD COD40,4 FT089810
REM FT089820
LDT POOL-1,1 GET ASTACK WORD FT089830
LFT BARGT FT089840
TRA      COD11               SUBPROGRAM PARAMETER
REM
PXA      +4                   GO PROCESS THE CONSTANT ADDEND
REM
CAL      POOL-1,1             SECOND ISTACK ENTRY
REM
ANA      =077777              SAVE POINTER
STA      COD31                PUT INTO SECOND INSTRUCTION
ORA*     COD01                OR IN THE OP CODE

COD38    ZET                 SKIP IF NO XR WAS LOADED
ORA       #4820               TAG OF 4
ORS       XRFLAG              INSTRUCTION SET BY CATEST ROUTINE

***    ***                    TEST THE ROUTING FLAG
TRA      EXCODE               NO MORE CODE NEEDED HERE
REM
CAL      =077777               ADDEND OF MINUS ONE
LNT      BARRY                SKIP IF NOT AN ARRAY
TRA      COD37+2              GET THE EQUIV POINTER
REM
LXA      COD31,4               GET THE EQUIV WORD
CAL      EQUIV,4               AND GET THE EQUIV WORD
PAX       +4                   AND GET THE DIMTBL POINTER
CAL      POOL-1,4             AND GET THE ARRAY LENGTH
PAC       +4                   COMPLEMENTED

COD37    TXI       +1,4,**   BUMP BY A POSSIBLE CONSTANT ADDEND
PXA       +4                   PUT TOTAL ADDEND INTO ACCUM
TSX      CATEST,4             GO PROCESS THE CONSTANT ADDEND
REM
CAL      COD31                GET THE SECOND INSTRUCTION
STZ      SPCDFG               RESET THE ROUTING FLAG
TRA      COD38                GO PUT OUT THE CODE AGAIN
REM

COD8     PAX       +4                   GET EQUIV WORD
ORA       LXD4                OR IN THE OPERATION CODE
ORS       XRFLAG              INSTRUCTION SET BY CATEST ROUTINE
LDI      EQUIV,4              GET THE ARRAY LENGTH
LFT       BARGT               IS IT AN ARGUMENT.
TRA      COD9                YES
REM
TSX      CITBLD,4,V            CODE IS... LXD VARIABLE,4
TRA      COD10-1
REM
TSX      PROLOG,4             PUT OUT PROLOGUE
CAL      LXD4                GET THE OPERATION CODE
TSX      CITBLD,4,N            CODE IS... LXD **,**
TRA      COD10-1
REM
TSX      CITBLD,4,T            CODE IS... (OP) TRANSFER VECTOR
TRA      EXCODE               GET ORIGINAL ASTACK WORD
REM
CAL      POOL-1,1             GET ORIGINAL ASTACK WORD
STA      COD4                CONSTANT ADDEND
ORA       COD4                EQUIV POINTER
PXD       +4                   PUT OUT PROLOGUE AND ADDEND
REM
CAL*     COD1                SKIP IF NO XR WAS LOADED
ZET      COD6                SKIP IF NO XR WAS LOADED
ORA  =4820  TAG OF 4
ORA  XRFQALG
TSX  CITBLD,4,N  CODE IS... (OP) **
N2I  SPCDFG  TEST THE ROUTING FLAG
TRA  EXCODE  NO MORE WORK TO DO HERE
REM  
CAL  =077777080000  DECREMENT ADDEND OF -1
LNT  BARRY  SKIP IF NOT AN ARRAY
TRA  COD4O+2
LX4  COD4,4  GET THE EQUIV POINTER
CAL  EQUIV,4  AND GET THE EQUIV WORD
PAX  4  AND GET THE DIMBL POINTER
CAL  POOL-1,4  AND GET THE ARRAY LENGTH
PAC  4  COMPLEMENTED
REM  
COD4O  TXI  **1,4,**  BUMP BY A POSSIBLE CONSTANT ADDEND
PAD  4
ORA  COD4  GET THE EQUIV POINTER
TX4  PROLG5,4  GO PUT OUT THE PROLOGUE
REM  
CAL  CODD1  GET THE SECOND INSTRUCTION
STA  SPCDFG  RESET THE ROUTING FLAG
TRA  COD41
REM  
COD41  STA  COD31  SAVE ARGUMENT POINTER IN SECOND INSTRUCTION
ORA  =060400000  ADD IN XR4 AND INDIRECT ADDRESS BITS
TX4  CITBLD,4,N  PUT OUT THE CODE
REM  
NZI  SPCDFG  IS THERE A SECOND INSTRUCTION.
TRA  EXCODE  NO
CAL  COD31  YES. GET THE INSTRUCTION
ACL  =060400000  ADD IN XR4, INDIRECT BITS, AND BUMP ADDRESS
TX4  CITBLD,4,N  PUT OUT THE SECOND INSTRUCTION
TRA  EXCODE  
SPACE  3  ROUTINE TO PUT OUT A CONSTANT ADDEND FOR A VARIABLE.
REM  
REM  USES SPECIAL CITBLD FLAGS 'D' FOR CA OF +1 AND 'G' FOR CA OF -1. ANY OTHER NON-ZERO CA IS GIVEN TO CITBLN
REM  WITH FLAG 'Q'.
REM  
CATEST  SXX  EXCATS,4
PAX  4  GET THE CA
TAL  EXCATS-2,4,0  FOR NO CONSTANT ADDEND
LDQ  CITX0
TXL  EXCATS-1,4,1  FOR CA OF +1.
LDQ  CITXG
TXH  EXCATS-1,4,-2  FOR CA OF -1.
TXS  CITBLN,4,Q  FOR OTHER CA'S. GIVE THE CA TO CITBLN
LDQ  CITXV  INSTRUCTION FOR THIS CASE
REM  
REM  SET THE INSTRUCTION FOR THE CODE ROUTINE
REM  
CAXT  AXT  **4
TRA  1,4
REM  
CITXV TSX  CITBLD,4,V  STRAIGHT VARIABLE RELOCATION
CITXO TSX  CITBLD,4,D  VARIABLE RELOCATION AND CA OF +1.
CITXG TSX  CITBLD,4,G  VARIABLE RELOCATION AND CA OF -1.
* TIL ARITHMETIC UTILITY FOR DP AND CA
ROUTINE TO RESTORE CELL 2

PUICL2 CLA SAVCL2 GET THE CELL-2 FLAG
TPL 1,4 CELL 2 IS NOT NOW SAVED
SXA EXPUT2,4
TSX FREWRK,4 FREE UP THE WORKING CELL
CODEO LDQW
CODEN STQ2 RESTORE CELL 2
STZ SAVCL2 RESET THE POINTER
STZ SETUP2 RESET THE ROUTINE NAME CELL

EXPUT2 AXT **,4
TRA 1,4
SPACE 6

* ROUTINE TO ESTABLISH OBJECT-TIME LINKAGES TO DP AND CA Routines

SETUP SXA EXSETP,4
CAL SAVCL2 SKIP IF CELL 2 IS STILL SAVED
TNZ SETUP1
REM
CODEN LDQ2 GET CELL 2
TSX GETWRK,4 GET A WORKING CELL
SSM SET SIGN MINUS AND SAVE WRKCEL POINTER
STO SAVCL2
CODEO STQW
REM

SETUP1 LXA EXSETP,4
CAL 1,4 GET THE ROUTINE TO BE LOADED
LAS SETUP2 IS IT SAME AS ONE ALREADY LOADED.
TRA **+2 NO
TRA EXSETP+1 YES. ALL DONE
REM
SLW SETUP2 SAVE NAME OF ROUTINE IN CELL
STL 1,4 MARK USAGE IN SPECIAL NAME TABLE
ORA LDQ
TSX CITBLD,4,S GET THE TRANSFER VECTOR ENTRY
CODEO STQ2 PUT INTO CELL 2
REM
EXSETP AXT **,4
TRA 2,4
SPACE 6

* ROUTINE TO PUT PSEUDO-AC INTO PSEUDO-MQ

AC+MQ SXA EXACMQ,4
ZET REGFLG SKIP IF AC IS ALREADY IN THE AC+MQ
TRA AC+MQ1
REM
CODEN GLA777
CODEN LDQ776
AC+MQ1 CODEN STQ775
CODEO STQ774
REM

EXACMQ AXT **,4
TRA 1,4
SPACE 6

* ROUTINE TO PUT PSEUDO-MQ INTO PSEUDO-AC

MQ.AC SXA EXACMQ,4
ZET REGFLG SKIP IF PSEUDO-MQ IS ALREADY IN AC+MQ

*
<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRA</td>
<td>MQ.AC1</td>
<td>FT091620</td>
</tr>
<tr>
<td>REM</td>
<td>MQ.AC1</td>
<td>FT091630</td>
</tr>
<tr>
<td>CODEN</td>
<td>CLA775</td>
<td>FT091640</td>
</tr>
<tr>
<td>CODEN</td>
<td>LDQ774</td>
<td>FT091650</td>
</tr>
<tr>
<td>MQ.AC1</td>
<td>CODEN STQ777</td>
<td>FT091660</td>
</tr>
<tr>
<td>CODEN</td>
<td>STQ776</td>
<td>FT091670</td>
</tr>
<tr>
<td>TRA</td>
<td>EXACMQ</td>
<td>FT091680</td>
</tr>
<tr>
<td>SPACE</td>
<td></td>
<td>FT091690</td>
</tr>
<tr>
<td>* ROUTINE TO LOAD THE PSEUDO-AC FROM STORAGE</td>
<td>FT091700</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FT091710</td>
</tr>
<tr>
<td>CDDLAC</td>
<td>SXA EXCODE+1,1 SAVE XR1 IN CODE ROUTINE</td>
<td>FT091720</td>
</tr>
<tr>
<td>AXC</td>
<td>CDDLAC1-2,1 GET RETURN POINT FROM CODE ROUTINE</td>
<td>FT091730</td>
</tr>
<tr>
<td>REM</td>
<td>CDDLAC2</td>
<td>FT091740</td>
</tr>
<tr>
<td>CAL</td>
<td>LDQ GET OP CODE FOR SECOND INSTRUCTION</td>
<td>FT091750</td>
</tr>
<tr>
<td>SLW</td>
<td>COD31 SAVE IN CODE CELL</td>
<td>FT091760</td>
</tr>
<tr>
<td>SXA</td>
<td>EXCODE+1 SET RETURN POINT IN CODE ROUTINE EXIT SEQ</td>
<td>FT091770</td>
</tr>
<tr>
<td>SXA</td>
<td>EXSPCD+4 SAVE XR4</td>
<td>FT091780</td>
</tr>
<tr>
<td>STL</td>
<td>SPCDFG SET THE SPECIAL CODE ROUTING FLAG</td>
<td>FT091790</td>
</tr>
<tr>
<td>TRA</td>
<td>COD32 GO TO CODE ROUTINE</td>
<td>FT091800</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FT091810</td>
</tr>
<tr>
<td>SPACE</td>
<td></td>
<td>FT091820</td>
</tr>
<tr>
<td>* ROUTINE TO PUT OUT TWO STZ INSTRUCTIONS</td>
<td>FT091830</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FT091840</td>
</tr>
<tr>
<td>CODSTZ</td>
<td>CAL STZ</td>
<td>FT091850</td>
</tr>
<tr>
<td>TRA</td>
<td>CODSTR+I</td>
<td>FT091860</td>
</tr>
<tr>
<td>SPACE</td>
<td></td>
<td>FT091870</td>
</tr>
<tr>
<td>* ROUTINE TO LOAD THE PSEUDO-MQ FROM STORAGE</td>
<td>FT091880</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>CODLMQ</td>
<td>FT091890</td>
</tr>
<tr>
<td>SXA</td>
<td>EXCODE+1,1</td>
<td>FT091900</td>
</tr>
<tr>
<td>AXC</td>
<td>CODLMQ1-2,1</td>
<td>FT091910</td>
</tr>
<tr>
<td>TRA</td>
<td>CDDLAC2</td>
<td>FT091920</td>
</tr>
<tr>
<td>SPACE</td>
<td></td>
<td>FT091930</td>
</tr>
<tr>
<td>* ROUTINE TO PUT NEGATIVE OF STORAGE INTO PSEUDO-AC</td>
<td>FT091940</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>CODSUB</td>
<td>FT091950</td>
</tr>
<tr>
<td>SXA</td>
<td>EXCODE+1,1</td>
<td>FT091960</td>
</tr>
<tr>
<td>AXC</td>
<td>CODSUB1-2,1</td>
<td>FT091970</td>
</tr>
<tr>
<td>TRA</td>
<td>CDDLAC2</td>
<td>FT091980</td>
</tr>
<tr>
<td>SPACE</td>
<td></td>
<td>FT091990</td>
</tr>
<tr>
<td>* ROUTINE TO OUTPUT THE STR AND PZE INSTRUCTIONS</td>
<td>FT092000</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>CODSTR</td>
<td>FT092010</td>
</tr>
<tr>
<td>CAL</td>
<td>PZE GET SECOND INSTRUCTION</td>
<td>FT092020</td>
</tr>
<tr>
<td>SLW</td>
<td>COD31 SAVE SECOND INSTRUCTION FOR CODE ROUTINE</td>
<td>FT092030</td>
</tr>
<tr>
<td>STL</td>
<td>SPCDFG</td>
<td>FT092040</td>
</tr>
<tr>
<td>TRA</td>
<td>CODE+1</td>
<td>FT092050</td>
</tr>
<tr>
<td>SPACE</td>
<td></td>
<td>FT092060</td>
</tr>
<tr>
<td>* ROUTINE TO LOAD THE AC+MQ FROM STORAGE</td>
<td>FT092070</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>CODLD</td>
<td>FT092080</td>
</tr>
<tr>
<td>CAL</td>
<td>LDQ</td>
<td>FT092090</td>
</tr>
<tr>
<td>TRA</td>
<td>CODSTR+1</td>
<td>FT092100</td>
</tr>
<tr>
<td>SPACE</td>
<td></td>
<td>FT092110</td>
</tr>
<tr>
<td>* ROUTINE TO STORE THE AC+MQ IN VARIABLE STORAGE</td>
<td>FT092120</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>CODSTO</td>
<td>FT092130</td>
</tr>
<tr>
<td>CAL</td>
<td>STQ</td>
<td>FT092140</td>
</tr>
<tr>
<td>TRA</td>
<td>CODSTR+1</td>
<td>FT092150</td>
</tr>
<tr>
<td>SPACE</td>
<td></td>
<td>FT092160</td>
</tr>
<tr>
<td>* RE-ENTRY POINTS FROM THE CODE ROUTINE</td>
<td>FT092170</td>
<td></td>
</tr>
<tr>
<td>CODSUB1</td>
<td>CODEN LRS RE-ENTRY FROM CODE ROUTINE</td>
<td>FT092180</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FT092190</td>
</tr>
<tr>
<td>CDDLAC1</td>
<td>CODEN STQ777 RE-ENTRY FROM CODE ROUTINE</td>
<td>FT092200</td>
</tr>
<tr>
<td>CODEN</td>
<td>STQ776</td>
<td>FT092210</td>
</tr>
</tbody>
</table>
**RESULT ROUTINE.**

**TYPICAL CALL IS VIA THE 'RESULT' MACRO...**

**ENTRY FOR FLOATING ROUND OPTION.**

**NORMAL RESULT ENTRY.**

**GET CALLING INSTRUCTION**

**DECREMENT TO ADDRESS**

**SAVE LOCATION CODE FOR RESULT**

**GET THE CALLING INSTRUCTION**

**INITIALIZE REGFLG**

**IF AC AND MQ CONTAIN VALID RESULTS,**

**IN ADDITION TO A PSEUDO-REG, SET THE FLAG**

**SAVE THE MODE INDICATION OF THE RESULT**

**POSITION THE MODE BITS INTO PREFIX**

**MODIFY ASTACK**

**RETURN TO GEN ROUTINE**
<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTL</td>
<td>A R I T H M E T I C U T I L I T Y - S T O R E</td>
<td>FT092820</td>
</tr>
<tr>
<td></td>
<td>ROUTINE TO STORE THE ACCUMULATOR OR MQ IN TEMPORARY STORAGE.</td>
<td>FT092830</td>
</tr>
<tr>
<td></td>
<td>P U T S NUMBER OF WORKING CELL INTO ADDRESS OF CURRENT ASTACK WORD,</td>
<td>FT092840</td>
</tr>
<tr>
<td></td>
<td>WITH A 1-BIT IN THE HIGH ORDER POSITION OF THE ADDRESS.</td>
<td>FT092850</td>
</tr>
<tr>
<td>REM</td>
<td>IS IT ALREADY IN STORAGE.</td>
<td>FT092860</td>
</tr>
<tr>
<td>STR</td>
<td>1,4 YES</td>
<td>FT092870</td>
</tr>
<tr>
<td>STZ</td>
<td>AUXFLG SET FLAG TO 'STORE' MODE</td>
<td>FT092880</td>
</tr>
<tr>
<td>SXA</td>
<td>EXSTOR+4 ZERO THE MQ</td>
<td>FT092890</td>
</tr>
<tr>
<td>LOQ</td>
<td>=0 ZERO THE MQ</td>
<td>FT092900</td>
</tr>
<tr>
<td>CAL</td>
<td>REGIST</td>
<td>FT092910</td>
</tr>
<tr>
<td>LGR</td>
<td>2 REGISTER OF THIS STORABLE WORD</td>
<td>FT092920</td>
</tr>
<tr>
<td>LXA</td>
<td>REGIST+1,4 GET ASTACK POINTER</td>
<td>FT092930</td>
</tr>
<tr>
<td>CAL</td>
<td>ASTACK+4 GET ASTACK WORD</td>
<td>FT092940</td>
</tr>
<tr>
<td>ARS</td>
<td>33 SHIFT MODE INTO POSITION</td>
<td>FT092950</td>
</tr>
<tr>
<td>LGR</td>
<td>3</td>
<td>FT092960</td>
</tr>
<tr>
<td>CAL</td>
<td>AUXILX GET 'OP' CODE FOR STORE OPERATION</td>
<td>FT092970</td>
</tr>
<tr>
<td>LGR</td>
<td>12 ZERO A-OPERAND BITS AND MOVE IN THE OP CODE</td>
<td>FT092980</td>
</tr>
<tr>
<td>STQ</td>
<td>GEN6 SET THE KEY WORD IN ITS CELL</td>
<td>FT092990</td>
</tr>
<tr>
<td>TIA</td>
<td>GEN5 GO NOW TO THE GEN ROUTINE</td>
<td>FT093000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FT093010</td>
</tr>
<tr>
<td>Instruction</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------------------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>TIL</td>
<td>Arithmetic utility - place</td>
<td></td>
</tr>
<tr>
<td>*</td>
<td>Routine to place current stack word into the accumulator</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Used by IF statement processor only.</td>
<td></td>
</tr>
<tr>
<td>STL</td>
<td>SET FLAG TO 'PLACE' MODE</td>
<td></td>
</tr>
<tr>
<td>STA</td>
<td>FLAG TO THE GEN ROUTINE</td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>GO FIND PROPER CODE SEQUENCE</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
AUXI1 CODE CLA,B REM GENXIT
TRA GENXIT
REM
AUXI2 NZT AUXFLG REM GENXIT
TRA AUXIL9 FOR STORE MODE
TRA GENXIT
REM
AUXI3 CODE CAL,B REM BOOLEAN
TRA GENXIT
REM
AUXI5 STZ AUXFLG REM GENXIT
TRA AUXIL4+1 SET FLAG TO 'MQ' MODE
REM
AUXI4 STL AUXFLG REM GENXIT
TRA AUXIL8 SET FLAG TO 'AC' MODE
REM NZT AUXFLG FOR STORE MODE
TRA AUXIL8
REM ZET REGFLG RESULT ALREADY IN AC
TRA GENXIT
REM CAL CLA777 GET CODE
REM NZT AUXFLG
REM CLA775 GET CODE FOR MQ LOAD
CAL CLA775 CODE TO LOAD THE ACCUM
TRA GENXIT
REM
AUXI6 ZET AUXFLG REM GENXIT
TRA GENXIT RESULT ALREADY IN ACCUM
REM TSX PREPWK,4 GO PREPARE THE WORKING CELL
REM CODED STO,W STORE THE ACCUMULATOR
EXSTOR AXT #4
TRA 1,#4
REM
AUXI7 ZET AUXFLG REM GENXIT
TRA GENXIT RESULT ALREADY IN ACCUM
REM TSX PREPWK,4 GO PREPARE THE WORKING CELL
REM CODED SLW,W STORE OFF THE BOOLEAN ACCUMULATOR
TRA EXSTOR
REM
AUXI9 TSX PREPWK,4 REM EXSTOR GO PREPARE THE WORKING CELL
RE CODED STO,W STORE OFF THE MQ
TRA EXSTOR
REM
AUXI8 ZET REGFLG REM AUX110 SKIP IF ACCUM AND MQ ALREADY HAVE DATA
TRA AUX110
REM CAL CLA775 REM AUXFLG
REM ZET AUXFLG
REM ACL =2 MODIFY INSTRUCTION FOR THE PSEUDO-AC
REM TSX CITBLD,4,N PUT OUT CODE TO LOAD THE ACCUM
REM CAL LDQ774 REM AUXFLG
REM ZET AUXFLG
REM ACL =2 REM CITBLD,4,N PUT OUT CODE TO LOAD THE MQ
REM
AUXI10 TSX PREPWK,4,#2 REM AUX111 PREPARE TWO WORKING CELLS
STA AUX111 SAVE THE WORKING CELL POINTER
<table>
<thead>
<tr>
<th>CODEO</th>
<th>STO,W</th>
<th>STORE OFF THE ACCUMULATOR</th>
<th>FT093700</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAL</td>
<td>AUXI11</td>
<td>REGAIN THE WORKING CELL POINTER</td>
<td>FT093710</td>
</tr>
<tr>
<td>ADD</td>
<td>=1</td>
<td>BUMP THE POINTER</td>
<td>FT093720</td>
</tr>
<tr>
<td>TSX</td>
<td>CITBLD,4,W</td>
<td>STORE OFF THE MQ</td>
<td>FT093730</td>
</tr>
<tr>
<td>TRA</td>
<td>EXSTOR</td>
<td></td>
<td>FT093740</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FT093750</td>
</tr>
<tr>
<td>AUXI11</td>
<td>STQ</td>
<td>**</td>
<td>CODE WORD FOR STORE ROUTINE</td>
</tr>
<tr>
<td>SPACE</td>
<td>4</td>
<td></td>
<td>FT093770</td>
</tr>
<tr>
<td>PREPWK</td>
<td>SXA</td>
<td>EXPREP,4</td>
<td>ROUTINE TO PREPARE A WORKING CELL</td>
</tr>
<tr>
<td>CAL</td>
<td>=4</td>
<td>GET CALLING INSTRUCTION</td>
<td>FT093790</td>
</tr>
<tr>
<td>STD</td>
<td>=1</td>
<td>PUT NUMBER OF WORDS WANTED INTO INSTRUCTION</td>
<td>FT093800</td>
</tr>
<tr>
<td>TSX</td>
<td>GETWRK,4,**</td>
<td>GO GET THE WORKING CELLS</td>
<td>FT093810</td>
</tr>
<tr>
<td>DRA</td>
<td>=1B21</td>
<td>ADD IN A HIGH-ORDER BIT FOR ASTACK</td>
<td>FT093820</td>
</tr>
<tr>
<td>LXA</td>
<td>REGIST+1,4</td>
<td>GET ASTACK POINTER</td>
<td>FT093830</td>
</tr>
<tr>
<td>STA</td>
<td>ASTACK,4</td>
<td>MODIFY ASTACK</td>
<td>FT093840</td>
</tr>
<tr>
<td>ANA</td>
<td>=037777</td>
<td>KILL THE HIGH-ORDER BIT</td>
<td>FT093850</td>
</tr>
<tr>
<td>STZ</td>
<td>REGIST</td>
<td>RE-INITIALIZE THE REGISTER CELLS</td>
<td>FT093860</td>
</tr>
<tr>
<td>STZ</td>
<td>REGIST+1</td>
<td></td>
<td>FT093870</td>
</tr>
<tr>
<td>EXPREP</td>
<td>AXT</td>
<td>**,4</td>
<td>FT093880</td>
</tr>
<tr>
<td>TRA</td>
<td>1,4</td>
<td></td>
<td>FT093890</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>FT093900</td>
</tr>
</tbody>
</table>
ALGEBRAIC ARITHMETIC UTILITY PROGRAM - GETWRK

ROUTINE TO OBTAIN ONE OR TWO WORKING CELLS FROM THE WORKING

POOL. CALLING SEQUENCE IS...

ISX GETWRK,4,XX

WHERE XX IS 0 IF ONE CELL IS WANTED, AND 2 IF TWO

CONSECUTIVE CELLS ARE WANTED.

ADDRESS OF LOGICAL ACCUMULATOR WILL CONTAIN THE POINTER

TO THE FIRST OF THESE CELLS.

AT PRESENT, ONLY 36 CELLS ARE AVAILABLE.

SPACE 2

GETWRK SXA EXWRK,4
CAL +4 GET THE CALLING INSTRUCTION
ANA =07000000 SAVE ONLY THE LOWER DECREMENT
SLW GETWR7
AXT GETWR3,4 PREPARE TO SET UP TRANSFER ADDRESS
TZE **2
AXT GETWR4,4 TWO CELLS WANTED
SXN GETWR5,4
REM CAL WRKCEL GET WORKING CELL KEY WORD
SSM
AXT 1/4 INITIALIZE XR4
LGR 1 MOVE NEXT BIT INTO MQ SIGN
GETWR5 TQP ** ADDRESS SET FROM ABOVE
TXI **1/4,1 BUMP COUNTER
GETWR2 TXL *-3/4,36 CONTINUE THE SEARCH IF NOT OUT OF BITS
STR MSG100,,SKEN ER ERROR MESSAGE
REM GETWR4 LBF IS AN ADJACENT SECOND BIT AVAILABLE.
TRA **2 YES. GO TO CHANGE
TXI GETWR2,4,1 NO. RETURN TO LOOK AGAIN
ORA =1 TURN ON THE BIT
REM GETWR3 LRS 0 TURN ON THE BIT TO MARK IT IN USE
SXN **1/4,4 PUT SHIFT COUNT INTO INSTRUCTION
LGR ** RE-ADJUST THE KEY WORD
SLW WRKCEL AND RESTORE IT
REM GETWR1 TXI **1/4 BUMP BY LENGTH OF CELLS RESERVED FOR
REM ARITH. STATE. FUNCTIONS.
ZET GETWR7
TXI **1/4,1 BUMP LENGTH IF TWO CELLS OBTAINED
WORKS SHOULDN'T BE BUMPED.
SXN **3/4,4 YES. UPDATE THE TEST INSTRUCTION
SXN LWORKS,4 AND UPDATE LWORKS
REM ZET GETWR7
TXI **2/4,-2 ADJUST POINTER FOR TWO-CELL CASE
TXI **1/4,-1 ADJUST POINTER FOR THE ONE-CELL CASE
PXA +4 PUT POINTER INTO ADDRESS OF ACCUM
REM EXWRK AXT **4
TRA 1/4
**Routine to Release One Working Cell from Service.**

**Calling Sequence is...**

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SPACE 2</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FREWRK SXA</td>
<td>EXWRK,4</td>
<td></td>
</tr>
<tr>
<td>SLW GETWR7</td>
<td>SAVE POINTER</td>
<td></td>
</tr>
<tr>
<td>PAC ,4</td>
<td><strong>1,4,</strong>* Bump by Number of Cells Reserved by**</td>
<td></td>
</tr>
<tr>
<td>FREWR1 TXI</td>
<td><strong>1,4,</strong>* Bump by Number of Cells Reserved by**</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>ARITH. STATE. FUNCTIONS.</td>
<td></td>
</tr>
<tr>
<td>CAL</td>
<td>WRKCEL</td>
<td></td>
</tr>
<tr>
<td>LGR  1,4</td>
<td>SHIFT KEY WORD</td>
<td></td>
</tr>
<tr>
<td>LRS  0</td>
<td>KILL THE APPROPRIATE BIT</td>
<td></td>
</tr>
<tr>
<td>LGL  1,4</td>
<td>SHIFT BACK INTO PROPER POSITION</td>
<td></td>
</tr>
<tr>
<td>SLW WRKCEL</td>
<td>RESTORE THE KEY WORD</td>
<td></td>
</tr>
<tr>
<td>CAL</td>
<td>GETWR7</td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>EXWRK</td>
<td></td>
</tr>
</tbody>
</table>

---

The text above describes the process of releasing a working cell from service, including the calling sequence, instructions, and their descriptions.
TTL PASS 1 UTILITY PROGRAM - ASCON1 FT094630
* CONSTANT STACKING ROUTINES... ASCON1 AND ASCON2 FT094640
  REM FT094650
* ROUTINES TO ENTER ONE- AND TWO-WORD CONSTANTS INTO THE CONSTANT TABLE.
  REM FT094670
* ENTRY FOR ONE-WORD CONSTANT...
  REM FT094680
* CAL CONSTANT FT094690
* TSX ASCON1,4 FT094700
  REM FT094710
* ENTRY FOR TWO-WORD CONSTANT
  CAL FIRST.PART FT094720
  LDQ SECOND.PART FT094730
  TSX ASCON2,4 FT094740
  REM FT094750
  REM FT094770
* CONTAB GOES BACKWARD IN MEMORY, WITH INDEXING STARTING AT ZERO.
  NOCONT = NUMBER OF CONSTANTS CURRENTLY IN THE TABLE.
  NOCNST = MAXIMUM ALLOWABLE LENGTH OF CONTAB.
  REM FT094780
* IF CONSTANT TABLE OVERFLOW OCCURS, THE CURRENT LENGTH (NOCONT) IS RESET TO ZERO, AND AN ERROR MESSAGE ISSUED.
  SPACE 4 FT094790
  ASCON1 SXA ASCX,4 FT094900
  LXAX NOCONT,4 FT094910
  TXL ASC2,4,0 TABLE EMPTY FT094920
  LAS CONTAB+1,4 FT094930
  TRA ++2 FT094940
  TXI ASCX-1,4,-1 CONSTANT FOUND. ADJUST POINTER FT094950
  TI RX-3,4,1 FT094960
  LXAX NOCONT,4 NOT IN TABLE. ADD IT. FT094970
  ASC2 TXI ++1,4,1 FT094980
  TXH ASC1,4,NOCNST FT094990
  ASC3 SLW CONTAB+1,4 TABLE NOT FULL FT095000
  SXA NOCONT,4 FT095010
  TXI ++1,4,-1 FT095020
  PXA +4 FT095030
  ASCX AX T **,4 FT095040
  TRA 1,4 FT095050
  ASC1 AX T 1,4 CONSTANT TABLE FULL. RESET IT. FT095060
  STR MSG100,,ASC2 FT095070
  ASCON2 SXA ASCX,4 FT095080
  XCL FT095090
  LXAX NOCONT,4 FT095100
  TRA ASC5 FT095110
  LAS CONTAB+1,4 FT095120
  TRA ++2 FT095130
  ASC5 TI X-3,4,1 FT095140
  TXH ASC1,4,NOCNST TABLE FULL FT095150
  LXAX NOCONT,4 FT095160
  SLW CONTAB+2,4 FT095170
  ASC6 TXI ++1,4,2 FT095180
  TXH ASC1,4,NOCNST TABLE FULL FT095190
  SXA NOCONT,4 FT095200
  SLW CONTAB+2,4 FT095210
<table>
<thead>
<tr>
<th>Command</th>
<th>Instruction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STQ</td>
<td>CONTAB+1,4</td>
<td></td>
</tr>
<tr>
<td>TXI</td>
<td>ASCX-1,4,-1</td>
<td>ADJUST POINTER AND RETURN</td>
</tr>
<tr>
<td>ASC4</td>
<td>XCL</td>
<td></td>
</tr>
<tr>
<td>LAS</td>
<td>CONTAB,4</td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>*+2</td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>ASCX-1</td>
<td>RETURN</td>
</tr>
<tr>
<td>XCL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>ASC5</td>
<td></td>
</tr>
<tr>
<td>TTL</td>
<td>PASS 1 UTILITY PROGRAM - NEXTLP</td>
<td>FT095310</td>
</tr>
<tr>
<td>-----</td>
<td>---------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td></td>
<td>ROUTINE TO GET NEXT LOCATION IN POOL</td>
<td>FT095320</td>
</tr>
<tr>
<td></td>
<td>SPACE 3</td>
<td>FT095330</td>
</tr>
<tr>
<td>NEXTLP</td>
<td>CAL</td>
<td>NXTLOC</td>
</tr>
<tr>
<td></td>
<td>ADD</td>
<td>=1</td>
</tr>
<tr>
<td></td>
<td>SLW</td>
<td>NXTLOC</td>
</tr>
<tr>
<td></td>
<td>ACL</td>
<td>++2</td>
</tr>
<tr>
<td></td>
<td>TRA</td>
<td>1,4</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>PZE</td>
<td>LSYM TB</td>
</tr>
</tbody>
</table>
* TIL PASS 1 UTILITY PROGRAM - PROLOG
* THIS SUBROUTINE ENTERS A WORD INTO THE PROLOGUE TABLE
* (PRLTBL). ENTRIES INTO THIS TABLE ARE MADE DURING PASS 1.
* SPACE 2
* CALLING SEQUENCE,
* REM
* CAL (ARGUMENT)
* TSX PROLGS,4
* *
* WHERE,
* REM
* ARGUMENT (3-17) = INCREMENT TO BUMP SUBR. ARG. LOCATION
* ARGUMENT (21-35) = POINTER IN SYMTAB FOR THIS ARGUMENT'S ENTRY
* SPACE 2
* ENTRY AT PROLOG FORCES INCREMENT TO BE ZERO
* SPACE 3
* PROLOG SXA =077777 SAVE ONLY ADDRESS
* PROLGS EXPROL,4 SAVE XR4
* AXC **,4 GET -1(PRLTBL) POINTER
* TXL PROL1,4-PRLNCH-1 ERROR IF PRLTBL FULL ALREADY
* PAX ;4 GET EQUIV POINTER
* XCL ;4 SAVE ARGUMENT IN MQ
* CAL EQUIV,4 GET THIS ENTRY'S EQUIV WORD
* ARS 15 SET ACC(P,1-35) TO,
* RQL 3 ARGUMENT NO. (P,1-5)
* LGL 15 DELTA (6-20)
* ALS 15 PCOUNT (21-35)
* ACL PCOUNT STORE THIS NEW ENTRY IN PROLOGUE
* XEC PROLOG+2 RESET XR4 BEFORE STORING
* SLW PRLTBL-1,4 TABLE (PRLTBL)
* LXA PROLOG+2,4 BUMP PRLTBL POINTER
* TXI **+1,4,1 X
* SXA PROLOG+2,4
* EXPROL AXT **,4 RESTORE XR4
* IRA 1,4 AND EXIT
* REM
* "THE FOLLOWING MACRO IS EXECUTED IFF A CALL IS MADE TO ENTER A WORD INTO PRLTBL AND THE TABLE IS FULL."
* PROLI STR MSG100,,EXPROL
THIS SUBROUTINE IS RESPONSIBLE FOR INSERTING ENTRIES INTO THE
COMPILED INSTRUCTION TABLE (CITBL) AND CIT FLAG (CITFLG).
TABLE. EACH CIT ENTRY HAS ASSOCIATED WITH IT ONE AND ONLY ONE
FLAG BYTE IN CITFLG. THIS CIT IS 6 TIMES LONGER THAN
CITFLG.

NOW CIT IS BROKEN INTO TWO HALVES, CIT1 AND CIT2. IF
CIT1 IS FILLED, ENTRIES ARE MADE IN CIT2. IF SUFFICIENT
ENTRIES ARE MADE TO FILL CIT2 THEN IT (CIT2) IS DUMPED ON
SYSUT1 ALONG WITH ITS FLAG TABLE (CITFLG).
SUCCESSIVE ENTRIES ARE PUT INTO CIT2. THUS, ONCE FILLED,
CIT1 AND CITFLG1 ARE NOT DISTURBED AND ONLY CIT2 AND
CITFLG2 BLOCKS ARE DUMPED ONTO SYSUT1. THUS, IN PASS 2
CIT1 IS PROCESSED WHILE SYSUT1 IS REWINDING.

WHEN A CIT2 BLOCK IS FULL EXCEPT FOR ITS LAST ENTRY, A FLAG
OF 76 (OCTAL) IS INSERTED INTO CITFLG2 AND THESE TWO BLOCKS
ARE DUMPED ONTO SYSUT1 AND CITBLD IS REINITIALIZED TO
START FILLING CIT2 AGAIN WITH SUBSEQUENT ENTRIES.
A FLAG OF 77 (OCTAL) INDICATES THAT CITBLD IS TO TRUNCATE
THE CURRENT CIT AND CITFLG BLOCKS. IN THIS CASE THE 76
FLAG IS OMITTED.

CALLING SEQUENCE,

ENTRY AT CITBLD WILL BUMP PCOUNT BY ONE AND ENTRY AT
CITBLN WILL NOT BUMP PCOUNT.

AN ENTRY TO CITBLN IS MADE AT THE BEGINNING OF EACH
ARITHMETIC STATEMENT FUNCTION AND AN ENTRY AT CITBLF IS
MADE AT THE END OF EACH ARITHMETIC STATEMENT.

ENTRY WITH NO STEP OF PCOUNT
ENTRY WITH STEP OF PCOUNT BY 1
IS THIS END OF ALL A.S.F.
YES
DID ANY A.S.F. EXIST
NO
IS PASS 1 OUTPUT TO BE LISTED.
NO
LIST INSTRUCTION, PCOUNT, AND CITBLD FLAG
NO
CITBL,
TSX CINST,4
<table>
<thead>
<tr>
<th>MZE</th>
<th>SYSUT1</th>
<th>FT097680</th>
</tr>
</thead>
<tbody>
<tr>
<td>REWIND</td>
<td>SYSUT1</td>
<td>FT097690</td>
</tr>
<tr>
<td>STL</td>
<td>IOEX</td>
<td>FT097700</td>
</tr>
<tr>
<td>BRA</td>
<td>REWIND,,++2</td>
<td>FT097710</td>
</tr>
<tr>
<td>MZE</td>
<td>SYSUT1</td>
<td>FT097720</td>
</tr>
<tr>
<td>END19</td>
<td>REM</td>
<td>FT097730</td>
</tr>
<tr>
<td>TRA</td>
<td>EXCIT</td>
<td>FT097740</td>
</tr>
<tr>
<td>REM</td>
<td>RETURN</td>
<td>FT097750</td>
</tr>
<tr>
<td>REM</td>
<td>... TO HANDLE END OF ARITH. STMT. FCNS.</td>
<td>FT097760</td>
</tr>
<tr>
<td>C1TBL6</td>
<td>STZ C1TBL6</td>
<td>ENTRY FOR START OF A. S. F.</td>
</tr>
<tr>
<td>STL</td>
<td>C1TBL7</td>
<td>FT097780</td>
</tr>
<tr>
<td>TRA</td>
<td>1,4</td>
<td>FT097790</td>
</tr>
<tr>
<td>REM</td>
<td>FT097800</td>
<td></td>
</tr>
<tr>
<td>C1TBLF</td>
<td>STL C1TBL6</td>
<td>ENTRY FOR END OF A. S. F.</td>
</tr>
<tr>
<td>TRA</td>
<td>1,4</td>
<td>FT097820</td>
</tr>
<tr>
<td>REM</td>
<td>FT097830</td>
<td></td>
</tr>
<tr>
<td>C1TBLG</td>
<td>STZ C1TBL6</td>
<td>CONTROL HERE IF END OF A. S. F.</td>
</tr>
<tr>
<td>LOQ</td>
<td>C1TBL6</td>
<td>FT097850</td>
</tr>
<tr>
<td>STQ</td>
<td>C1TBLH</td>
<td>FT097860</td>
</tr>
<tr>
<td>ZET</td>
<td>SNAME</td>
<td>FT097870</td>
</tr>
<tr>
<td>TRA</td>
<td>0,4</td>
<td>FT097880</td>
</tr>
<tr>
<td>SLW</td>
<td>C1TB15</td>
<td>FT097890</td>
</tr>
<tr>
<td>CAL</td>
<td>PCOUNT</td>
<td>FT097900</td>
</tr>
<tr>
<td>ACL</td>
<td>=3</td>
<td>FT097910</td>
</tr>
<tr>
<td>SLW</td>
<td>PCOUNT</td>
<td>FT097920</td>
</tr>
<tr>
<td>STL</td>
<td>(FPT)</td>
<td>FT097930</td>
</tr>
<tr>
<td>SXX</td>
<td>++2,4</td>
<td>FT097940</td>
</tr>
<tr>
<td>TXS</td>
<td>C1TB1N,4,H</td>
<td>GO OUTPUT 'H' FLAG FOR CITPRO</td>
</tr>
<tr>
<td>AXI</td>
<td>...4</td>
<td>FT097960</td>
</tr>
<tr>
<td>CAL</td>
<td>C1TB15</td>
<td>FT097970</td>
</tr>
<tr>
<td>TRA</td>
<td>0,4</td>
<td>FT097980</td>
</tr>
<tr>
<td>SPACE</td>
<td>3</td>
<td>FT097990</td>
</tr>
<tr>
<td>C1TBL8</td>
<td>TRA C1TBLH+4</td>
<td>RESTORE THE ACCUM</td>
</tr>
<tr>
<td>C1TBL9</td>
<td>ZET C1TBL6</td>
<td>FT098010</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FT098020</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FT098030</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FT098040</td>
</tr>
</tbody>
</table>
TIL PASS I UTILITY PROGRAM - SCNB4D

* ROUTINE TO SCAN A STATEMENT, AND RETURN ONLY VARIABLE NAMES
* AND STATEMENT LABELS IN BCD. RESULT WILL BE IN THE LOGICAL
* ACCUMULATOR. THE ACCUM SIGN WILL BE SET MINUS IF THE ITEM
* FOUND IS A LABEL, AND WILL BE SET PLUS IF IT IS A VARIABLE NAME.
* ENTRY...
* XR 2 MUST BE POINTING TO THE STARTING POSITION IN COLUMN 3.
* REM
* THERE IS AN ERROR RETURN IN THE ROUTINE IN CASE THE FIRST
* CHARACTER TO BE SCANNED IS A BREAK CHARACTER, OR IF A LABEL IS
* GREATER THAN 99999 OR ZERO, OR IF A SYMBOL IS TOO LONG.
* REM
* SPACE 3
* SCNB4D
* SXA SCNB4C,4
* SXA SCNB4C+1,1
* LDQ COLUMN,2
* CRQ SCANT1,1
* TXL **+3,1,SCAN1-1 LETTER
* TXL SCNB2,1,SCAN1 ERROR IF NOT NUMERAL
* ERROR 81,SKEND
* REM VARIABLE - EXIT ON BREAK CHARACTER
* REM
* SXD SCNB1,2
* CAL COLUMN,2
* TXI **+1,2i-1
* LDQ COLUMN,2
* CRQ SCANT1,1
* TXL *-3,1,SCAN1
* SXA **+1,2
* AXC **,**
* SCNB1 TXI **+1,4,**
* TXH SCNB3,4,6
* ANA SCANT1,4
* ORA SCANT2,4 APPEND TRAILING BLANKS
* SCNB4 CX AXT **,**
* AXT **,**
* TRA 1,4
* SCNB3 ERROR 15,SKEND
* REM
* SCNB2 SXD SCNB4,2 ALL NUMERIC LABEL SCANNER
* CAL COLUMN,2
* TXI **+1,2i-1
* LDQ COLUMN,2
* CRQ SCANT1,1
* SCNB5 TXL **+2,1,SCAN1-1
* TXL *-4,1,SCAN1
* SCNB6 SXA **+1,2 BREAK CHARACTER OR LETTER FOUND
* AXC **,**
* SCNB4 TXI **+1,4,**
* TXH SCNB7,4,5
* ANA SCANT1,4
* TNZ **+2
* ERROR 82,SKEND
* REM
* ORA SCANT2,4
* LDQ =H
* LGR 6
* TNZ *-1
* XCL
* SSM
<table>
<thead>
<tr>
<th>TRA</th>
<th>SCNBX</th>
<th>FT098650</th>
</tr>
</thead>
<tbody>
<tr>
<td>REM</td>
<td></td>
<td>FT098660</td>
</tr>
<tr>
<td>SCNB7</td>
<td>ERROR</td>
<td>70, SKEND</td>
</tr>
</tbody>
</table>
TTL PASS 1 UTILITY PROGRAM - LOCATE

ENTRY...

LDQ SYMBOL

TSX LOCATE,4

RETURN...

REM

POSITION OF THE SYMBOL IN THE TABLE IS RETURNED IN THE ADDRESS

OF THE ACCUMULATOR. THE ACCUM SIGN IS SET MINUS IF THE SYMBOL

WAS NOT ALREADY IN THE TABLE.

REM

THE SYMBOL TABLE OPERATES USING A SCRAMBLED SORT TECHNIQUE.

THE BCD SYMBOLS THEMSELVES ARE STORED IN A REGION CALLED

SYMtab, WHILE THEIR EQUIVALENTS ARE STORED IN A REGION

CALLED EQUIV. BOTH THESE REGIONS GO BACKWARD IN MEMORY,

WITH INDEXING STARTING AT ZERO. (NOTE THAT THE FIRST CELL

IN SYMTAB IS NOT AVAILABLE).

SPACE 3

LOCATE STQ LOCAT1

MPY =0400400400

ANA =07777

SXA LOCATX,4

PAX *4

CAL LOCAT1

TXH *2,4,0

TXI *1,4,1

LOCAT5 NZI SYMTAB,4 IS THIS SPOT EMPTY

TRA LOCAT3 YES GO ENTER SYMBOL

LAS SYMTAB,4 NO SCAN FORWARD FOR SYMBOL.

TXI *-3,4,1

TRA LOCAT2 SYMBOL FOUND.

TXI *-5,4,1

LOCAT3 SLW SYMTAB,4 ENTER THE SYMBOL

STZ EQUIV,4

TXH LOCAT4,4,LSYMTH-1 IS THIS THE BOTTOM OF THE TABLE.

PXA *4 NO

SSM *4 MARK SYMBOL AS NOT IN

LDI EQUIV,4

LOCATX AXT *4

TRA 1,4

LOCAT2 PXA,4

LDI EQUIV,4

LXA LOCATX,4

TRA 1,4

LOCAT4 AXT 1,4

STZ SYMTAB-LSYMTH

TRA LOCAT5
<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>EQUATION</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>SYMSTD</td>
<td>ROUTINE TO ENTER AN INTEGER VARIABLE INTO SYMTAB.</td>
</tr>
<tr>
<td>4</td>
<td>SYMXA</td>
<td>SYMSTX,4</td>
</tr>
<tr>
<td>1</td>
<td>XCL</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>TSX</td>
<td>LOCATE,4</td>
</tr>
<tr>
<td>4</td>
<td>SYMSTY</td>
<td>MINTG+BVARB</td>
</tr>
<tr>
<td>4</td>
<td>PAX</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>STI</td>
<td>EQUIV,4</td>
</tr>
<tr>
<td>4</td>
<td>LFT</td>
<td>BEXTF+BINF</td>
</tr>
<tr>
<td>4</td>
<td>TRA</td>
<td>SYMER</td>
</tr>
<tr>
<td>4</td>
<td>AXT</td>
<td>**,4</td>
</tr>
<tr>
<td>1</td>
<td>REM</td>
<td>ROUTINE TO ENTER INTEGER VARIABLE IS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SYMTAB. CHECK IF INTEGER MODE WITH ERROR EXFT099310</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EXIT IF NOT.</td>
</tr>
<tr>
<td></td>
<td>SYMSTB</td>
<td>STZ SYMSII</td>
</tr>
<tr>
<td></td>
<td>TRA</td>
<td>**2</td>
</tr>
<tr>
<td></td>
<td>SYMSTA</td>
<td>STL SYMSI1</td>
</tr>
<tr>
<td></td>
<td>SXL</td>
<td>SYMSTX,4</td>
</tr>
<tr>
<td></td>
<td>LAS</td>
<td>=H000000</td>
</tr>
<tr>
<td></td>
<td>TRA</td>
<td>**1</td>
</tr>
<tr>
<td></td>
<td>LAS</td>
<td>=H000000</td>
</tr>
<tr>
<td></td>
<td>NOP</td>
<td>12,PD02</td>
</tr>
<tr>
<td></td>
<td>ERROR</td>
<td>XCL</td>
</tr>
<tr>
<td></td>
<td>TSX</td>
<td>LOCATE,4</td>
</tr>
<tr>
<td></td>
<td>ZET</td>
<td>SYMSTI</td>
</tr>
<tr>
<td></td>
<td>SXL</td>
<td>BLH5X+8DDIX</td>
</tr>
<tr>
<td></td>
<td>TRA</td>
<td>SYMSTY</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SYMER</td>
<td>CAL SYMTAB,4</td>
</tr>
<tr>
<td></td>
<td>SLW</td>
<td>FORM69</td>
</tr>
<tr>
<td></td>
<td>ERROR</td>
<td>69,SKEND</td>
</tr>
<tr>
<td>Instruction</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>STO STOLBL</td>
<td>Save AC sign</td>
<td></td>
</tr>
<tr>
<td>XCL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TSX</td>
<td>Locate, 4</td>
<td></td>
</tr>
<tr>
<td>PAX</td>
<td>Label 4</td>
<td></td>
</tr>
<tr>
<td>LDQ</td>
<td>STOLBL 2</td>
<td></td>
</tr>
<tr>
<td>TQP</td>
<td>Label reachable</td>
<td></td>
</tr>
<tr>
<td>SIL</td>
<td>BPATH YES</td>
<td></td>
</tr>
<tr>
<td>STI</td>
<td>EQUIV 4</td>
<td></td>
</tr>
<tr>
<td>LFT</td>
<td>IS IT ANYTHING BUT A LABEL</td>
<td></td>
</tr>
<tr>
<td>ERROR</td>
<td>63, **1 YES</td>
<td></td>
</tr>
<tr>
<td>STOLBX</td>
<td>AXT **, 4</td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>1, 4</td>
<td></td>
</tr>
</tbody>
</table>
**P S A S U T I L I T Y P R O G R A M - S K I P**

* Routines to test the next character of statement against a given character supplied in the decrement.
* Calling sequences to the two routines are:

<table>
<thead>
<tr>
<th>REM</th>
<th>TSX SKIP, OCTAL VALUE OF CHARACTER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TRA *** RETURN IF CHARACTERS DO NOT MATCH</td>
</tr>
<tr>
<td></td>
<td>*** *** RETURN IF CHARACTERS MATCH</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>REM</th>
<th>TSX SKIP1, OCTAL VALUE OF CHARACTER</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TRA *** RETURN IF CHARACTERS MATCH</td>
</tr>
<tr>
<td></td>
<td>*** *** RETURN IF CHARACTERS DO NOT MATCH</td>
</tr>
</tbody>
</table>

**Skip**

<table>
<thead>
<tr>
<th>CAL</th>
<th>0,4</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALS</td>
<td>12</td>
</tr>
<tr>
<td>ERA</td>
<td>COLUMN,2</td>
</tr>
<tr>
<td>ARS</td>
<td>30</td>
</tr>
<tr>
<td>TNZ</td>
<td>1,4</td>
</tr>
<tr>
<td>TXI</td>
<td>++1,2,-1</td>
</tr>
<tr>
<td>TRA</td>
<td>2,4</td>
</tr>
<tr>
<td>SPACE</td>
<td>4</td>
</tr>
</tbody>
</table>

**Skip1**

<table>
<thead>
<tr>
<th>CAL</th>
<th>0,4</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALS</td>
<td>12</td>
</tr>
<tr>
<td>ERA</td>
<td>COLUMN,2</td>
</tr>
<tr>
<td>ARS</td>
<td>30</td>
</tr>
<tr>
<td>TNZ</td>
<td>2,4</td>
</tr>
<tr>
<td>TXI</td>
<td>++1,2,-1</td>
</tr>
<tr>
<td>TRA</td>
<td>1,4</td>
</tr>
</tbody>
</table>

*Page 257*
TIL PASS 1 UTILITY PROGRAM - BREAK
REM THIS ROUTINE GETS THE NEXT BREAK CHARACTER. INITIALLY
REM XR2 SHOULD BE SET AT 1ST CHARACTER TO BE EXAMINED, ROUTINE FT100120
REM WILL EXIT WITH XR2 POINTING TO CHARACTER FOLLOWING
REM THAT WHICH CAUSED THE EXIT. ROUTINE EXITS WITH BCD
REM VALUE OF BREAK CHARACTER IN LAC RIGHT JUSTIFIED WITH
REM LEADING ZEROS.
REM
BREAK SXA BREAKX,1
LDQ COLUMN,2
CRQ SCAN,1,1
TXH ++2,1,SCAN1+1
TXI --3,2,-1
CAL COLUMN,2
ARS 30
TXI ++1,2,-1
BREAK AXI **,1
TRA 1,4
TTL P A S S 1 - I N D E X C H E C K I N G
THE FOLLOWING ROUTINES HANDLE INDEXING IN FASTRAN.

* THIS ROUTINE IS ENTERED WHENEVER AN INTEGER VARIABLE IS CHANGED
* IN A SUBSTITUTION STATEMENT, OR IS THE IMPLIED DO PARAMETER
* IN AN INPUT/OUTPUT LIST, OR IS READ IN VIA AN INPUT LIST.
* REM
* A TEST IS MADE TO DETERMINE IF THIS VARIABLE IS CURRENTLY A
* DO INDEX. IF SO, THIS IS AN ERROR.
* REM
* ENTRY... LDI SOME.FORM.OF.EQUIV.WORD.HAVING.FLAG.BITS
* TSX INDXT,4
* (RETURN)
* REM
* NOTE THAT THIS ROUTINE WILL BE ENLARGED WHEN BETTER INDEXING
* IS INSTALLED IN FASTRAN.

INDXT SPACE 3
LFT BDOIX IS THIS A DO INDEX
ERROR 87,++1 YES, ERROR
TRA 1,4 RETURN
<table>
<thead>
<tr>
<th>TITLE</th>
<th>POST PASS 1 - DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>FT100480 FT100490</td>
</tr>
<tr>
<td>REM</td>
<td>FT100500 FT100510</td>
</tr>
<tr>
<td>SPACE 9</td>
<td>FT100520 FT100530</td>
</tr>
<tr>
<td>REM</td>
<td>FT100540 FT100550</td>
</tr>
<tr>
<td>REM</td>
<td>FT100560 FT100570</td>
</tr>
<tr>
<td>REM</td>
<td>FT100580 FT100590</td>
</tr>
<tr>
<td>REM</td>
<td>FT100600 FT100610</td>
</tr>
<tr>
<td>REM</td>
<td>FT100620 FT100630</td>
</tr>
<tr>
<td>REM</td>
<td>FT100640 FT100650</td>
</tr>
<tr>
<td>REM</td>
<td>FT100660</td>
</tr>
</tbody>
</table>
EJECT
ON ENTRY TO POST PASS 1, THE COMPLETE SOURCE PROGRAM HAS BEEN
SCANNED AND THE COMPILED INSTRUCTION TABLE (C I T) AND THE
ASSOCIATED FLAG TABLE (C I T F L G) HAVE BEEN GENERATED AND, IF
OF SUFFICIENT LENGTH, RESIDE ON SYSUT1.

REM

THE FOLLOWING IS AN OVERALL PICTURE OF POST PASS 1. (DETAILED
COMMENTS PRECEDE EACH SECTION).

SPACE 3

POST PASS 1 PERFORMS THE FOLLOWING FUNCTIONS IN THE INDICATED
ORDER,

SPACE 2

PPASS1- LIST SYMBOL TABLE AND POOL AREA IFF 'LSTSYM' NON-ZERO
AND GO TO 'FINISH' IFF 'NOCODE' NON-ZERO.

REM

ASNASN- VERIFY ASSIGNED GO TO USAGE.

REM

ASEQUX- DEFINE VARIABLES APPEARING IN EQUIVALENCE STATEMENTS.

REM

TVECTL- COUNT NUMBER OF LIBRARY ROUTINES USED IN OBJECT

REM

PROLEN- CALCULATE LENGTH OF OBJECT TIME PROLOGUE.

REM

SYPASS- CHECK VALIDITY OF SYMBOL TABLE ENTRIES.

REM

ASCNN2- ASSIGN COMMON STORAGE FOR OBJECT PROGRAM

REM

OBJLEN- CALCULATE LENGTHS OF VARIOUS SECTIONS OF THE OBJECT

REM

OUTUND- OUTPUT UNDEFINED LABELS, STRINGS (FORMATS),

REM

TSTPP2- GO TO 'FINISH' IFF 'NOCODE' NON-ZERO.

REM

TSTPSI- PUNCH SYMBOL TABLE IF NECESSARY

REM

CDLABL- SET-UP FOR BINARY DECK LABELLING.

REM

OUTPDC- WRITE PROGRAM CARD ON SYSBIN.

REM

STVECT- STASH TRANSFER VECTOR INTO BINARY CARD IMAGES.

REM

GENPRL- GENERATE OBJECT TIME PROLOGUE.

SPACE 5

THIS IS THE END OF POST PASS 1. AT THIS POINT PASS 2 IS ENTERED.
### TTL POST PASS 1 - PPASS1

* POST PASS 1 IS ENTERED AT THIS POINT. A COUPLE OF TESTS ARE MADE BEFORE POST PASS 1 PROPER IS ENTERED.

| Space | 3 |

<table>
<thead>
<tr>
<th>PPASS1</th>
<th>ZET</th>
<th>LSTSYM</th>
<th>ARE SYMTAB AND POOL TO BE OUTPUT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TSX</td>
<td>STDUMP,4</td>
<td>YES, OUTPUT SYMTAB AND POOL</td>
</tr>
<tr>
<td>ZET</td>
<td>NOCODE</td>
<td>NO, WAS FATAL ERROR DETECTED IN PASS</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>FINISH</td>
<td>YES, EXIT THIS COMPILATION</td>
<td></td>
</tr>
<tr>
<td>SLF</td>
<td>NO, CONTINUE POST PASS 1 PROCESSING</td>
<td>TURN OFF THE CONSOLE WATCHER'S LIGHT</td>
<td></td>
</tr>
</tbody>
</table>

FT101140  FT101150  FT101160  FT101170  FT101180  FT101190  FT101200  FT101210  FT101220  FT101230
<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>TTL</td>
<td>POST PASS 1 - ASNASN</td>
</tr>
<tr>
<td>2</td>
<td>*</td>
<td>POST PASS 1 PROCESSOR FOR ASSIGN VARIABLES AND LABELS</td>
</tr>
<tr>
<td>3</td>
<td>*</td>
<td>ENTRY... ASNASN</td>
</tr>
<tr>
<td>4</td>
<td>*</td>
<td>THIS PROCESSOR IS ENTERED AT THE BEGINNING OF POST PASS 1</td>
</tr>
<tr>
<td>5</td>
<td>*</td>
<td>PROCESSING. THE ASSIGN TABLES ARE SCANNED. FOR THE</td>
</tr>
<tr>
<td>6</td>
<td>*</td>
<td>VARIABLES, BOTH TASSN AND TGOTO MUST BE ON.</td>
</tr>
<tr>
<td>7</td>
<td>*</td>
<td>ALL LABEL ENTRIES MUST HAVE MINUS SIGNS, INDICATING THAT THEY</td>
</tr>
<tr>
<td>8</td>
<td>*</td>
<td>APPEAR IN AN ASSIGNED GO TO LIST.</td>
</tr>
<tr>
<td>9</td>
<td>REM</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>REM</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>ASNASN</td>
<td>N2T</td>
</tr>
<tr>
<td>12</td>
<td>TRA</td>
<td>ASEQUX</td>
</tr>
<tr>
<td>13</td>
<td>REM</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>ST2</td>
<td>15,4</td>
</tr>
<tr>
<td>15</td>
<td>AXT</td>
<td>15,4</td>
</tr>
<tr>
<td>16</td>
<td>REM</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>LXD</td>
<td>ASNFLG,1</td>
</tr>
<tr>
<td>18</td>
<td>CAL</td>
<td>POOL,1</td>
</tr>
<tr>
<td>19</td>
<td>STD</td>
<td>ASNFLG</td>
</tr>
<tr>
<td>20</td>
<td>REM</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>PAX</td>
<td>2</td>
</tr>
<tr>
<td>22</td>
<td>PAI</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>RNI</td>
<td>TASSN+TGOTO</td>
</tr>
<tr>
<td>24</td>
<td>TRA</td>
<td>ASNAS2</td>
</tr>
<tr>
<td>25</td>
<td>REM</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>ASNAS1</td>
<td>TXI</td>
</tr>
<tr>
<td>27</td>
<td>CLA</td>
<td>POOL,1</td>
</tr>
<tr>
<td>28</td>
<td>SLW</td>
<td>+P+1</td>
</tr>
<tr>
<td>29</td>
<td>PDX</td>
<td>1</td>
</tr>
<tr>
<td>30</td>
<td>TMI</td>
<td>ASNAS4</td>
</tr>
<tr>
<td>31</td>
<td>REM</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>ZET</td>
<td>+P</td>
</tr>
<tr>
<td>33</td>
<td>TRA</td>
<td>ASNAS5</td>
</tr>
<tr>
<td>34</td>
<td>REM</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>CAL</td>
<td>SYMTAB,2</td>
</tr>
<tr>
<td>36</td>
<td>SLW</td>
<td>FORM29+13</td>
</tr>
<tr>
<td>37</td>
<td>TSK</td>
<td>LIST,4</td>
</tr>
<tr>
<td>38</td>
<td>PZE</td>
<td>BLANK,1</td>
</tr>
<tr>
<td>39</td>
<td>REM</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>ASNAS6</td>
<td>AXT</td>
</tr>
<tr>
<td>41</td>
<td>CAL</td>
<td>=H</td>
</tr>
<tr>
<td>42</td>
<td>SLW</td>
<td>COLUMN+16,4</td>
</tr>
<tr>
<td>43</td>
<td>TIX</td>
<td>1,-1,4,1</td>
</tr>
<tr>
<td>44</td>
<td>REM</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>AXT</td>
<td>14,4</td>
</tr>
<tr>
<td>46</td>
<td>ZET</td>
<td>+P</td>
</tr>
<tr>
<td>47</td>
<td>TRA</td>
<td>ASNAS4</td>
</tr>
<tr>
<td>48</td>
<td>REM</td>
<td></td>
</tr>
<tr>
<td>49</td>
<td>STL</td>
<td>+P+1,2</td>
</tr>
<tr>
<td>50</td>
<td>ASNAS5</td>
<td>LXA</td>
</tr>
<tr>
<td>51</td>
<td>CAL</td>
<td>SYMTAB,2</td>
</tr>
<tr>
<td>52</td>
<td>SLW</td>
<td>COLUMN+16,4</td>
</tr>
<tr>
<td>53</td>
<td>TIX</td>
<td>ASNAS4,4,1</td>
</tr>
<tr>
<td>54</td>
<td>REM</td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>ASNAS7</td>
<td>TSX</td>
</tr>
<tr>
<td>56</td>
<td>TRA</td>
<td>ASNAS6</td>
</tr>
<tr>
<td>57</td>
<td>REM</td>
<td></td>
</tr>
</tbody>
</table>

*FTI101240* *PAGE 263*
<table>
<thead>
<tr>
<th>ASNAS4 TXH</th>
<th>ASNAS1+1,1,0</th>
<th>IS THIS ALL THE LABELS IN THIS TABLE.</th>
<th>FT101840</th>
<th>FT101850</th>
</tr>
</thead>
<tbody>
<tr>
<td>TXL</td>
<td>ASNAS7,4,13</td>
<td>WRITE LINE IF BUFFER HAS INFORMATION IN IT</td>
<td>FT101860</td>
<td>FT101870</td>
</tr>
<tr>
<td>TRA</td>
<td>ASNASN</td>
<td>YES, RETURN FOR NEXT TABLE</td>
<td>FT101880</td>
<td>FT101890</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASNAS2 CAL</td>
<td>SYMTAB,2</td>
<td>GET BCD NAME OF GUILTY VARIABLE</td>
<td>FT101900</td>
<td>FT101910</td>
</tr>
<tr>
<td>SLW</td>
<td>FORM30+3</td>
<td>PUT NAME INTO ERROR MESSAGES</td>
<td>FT101920</td>
<td>FT101930</td>
</tr>
<tr>
<td>SLW</td>
<td>FORM31+4</td>
<td></td>
<td>FT101940</td>
<td>FT101950</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TSX</td>
<td>LIST,4</td>
<td>PUT OUT BLANK LINE</td>
<td>FT101960</td>
<td>FT101970</td>
</tr>
<tr>
<td>PZE</td>
<td>BLANK,,1</td>
<td></td>
<td>FT101980</td>
<td>FT101990</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RFT</td>
<td>TASSN</td>
<td>IS IT TASSN THAT IS OFF.</td>
<td>FT102000</td>
<td>FT102010</td>
</tr>
<tr>
<td>ERROR</td>
<td>30,ASNASN</td>
<td>NO, THUS TGOTO IS OFF.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ERROR SPACE</td>
<td>31,ASNASN</td>
<td>YES, TASSN IS OFF. PRINT MESSAGE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TASSNX VFD</td>
<td>18/018/TASSN</td>
<td>WHOLE-WORD REPRESENTATIONS OF FLAGS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TGOTOX VFD</td>
<td>18/018/TGOTO</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
* ENTRY... ASEQX
  * THIS ROUTINE ASSIGNS THE SYMBOLS IN THE EQUIVALENT CHAINS TO COMMON OR VARIABLE STORAGE.
  * IS TREATED IN ITS TURN AS FOLLOWS...
  * THE STACK IS SCANNED TO FIND THE LARGEST DEL (DELMAX).
  * AND TO DETERMINE IF ANY ELEMENT IS IN COMMON. THE STACK IS THEN SCANNED AGAIN. IF ANY ELEMENT OF THE CHAIN IS IN COMMON, THE FLAG BITS BCMM AND BLHSM ARE TURNED ON FOR EACH ELEMENT IN THIS CHAIN.
  * VARIABLES ARE ASSIGNED TO STORAGE (COMMON OR LOW-CORE VARIABLE) BY THE EXPRESSION...
  * LOCATION = CURRENT LCOMMN (OR LVARS) + DELMAX + DELMAX.
  * THE OVERHANG (HANG) IS COMPUTED FROM THE EXPRESSION...
  * HANG = LENGTH + DELMAX - DELMAX.
  * THE MAXIMUM OVERHANG (HANGMX) TO DATE IS FIGURED FROM...
  * HANGMX = MAXIMUM(HANG, HANGMX).
  * AT THE END OF THIS STACK, THE APropriate COUNTER (LCOMMN OR LVARS) IS INCREASED BY HANGMX. THE PROCESSOR THEN RETURNS TO TREAT THE NEXT EQUIVALENCE CHAIN.
  * THE LOCATION ASSIGNED TO EACH VARIABLE IS PLACED IN THE ADDRESS FIELD OF THE EQUIV TABLE WORD. NOTE THAT THE DIMTBLI Pointer FOR ARRAYS IS LOST AT THIS POINT.

RES

ASEQX
ZET  NOCODE  IS PROGRAM IN ERROR.
TRA  EXASEQ  YES, DON'T PROCESS EQUIVALENCE STACKS.

REM

ASEQ6
STZ  HANGMX
STZ  DELMAX
STZ  COMFLG
STZ  GTCCELL
REM

ASEQ7
LXD  INIFFG,4  GET POINTER TO STACK DIRECTOR
TXX  **2,4,0  IS THIS END OF EQUIVBL STACKS.
REM
TRA  EXASEQ

REM

CAL  POOL,4  NO. GET DIRECTOR TO NEXT STACK
SLW  INIFFG
LXA  INIFFG,4  GET EQUIVBL POSITION OF TOP OF STACK
REM

LDI  EQUIVL,4  GET TOP OF STACK LINKER WORD
LNT  300000  HAS THIS STACK ALREADY BEEN PROCESSED.
TXI  **2,4,1  NO. BUMP POINTER.
TRA  ASEQ7  YES. SKIP TO NEXT STACK
REM

ASEQ2
CLA  EQUIVL,4  GET ENTRY FROM EQUIVBL
TMI  ASEQ1  LINK WORD FOUND
PAX  ,1  GET EQUIV TABLE POINTER
ARS  18  SHIFT DEL TO ADDRESS
LQD  DELMAX
TLQ  **2
XCA  DELMAX
STA  DELMAX
REM
LDI  EQUIV,1  GET EQUIV ENTRY

PAGE 265
LFT  BCOMN  IS COMMON FLAG ON.
      COMFLG  YES, SET COMFLG ON
LFT  BLHSX  IS THIS VARIABLE DEFINED.
       DEFINE  YES, TURN ON THE DEFINE FLAG
       ASEQU2,4,1  RETURN FOR ANOTHER ENTRY
REM  ASEXQ9  GET ENTRY
       ASEQU3  LINK WORD FOUND.
REM  PAX 1  SET ENTRY TO EQUIV TABLE
       ARS 18  SHIFl DEL TO ADDRESS
       SUB  DELMAX  DEL - DELMAX
       SLW  ORIGIN  ORIGIN = DELMAX - DEL
REM  LDI  EQUIV,1  GET EQUIV TABLE ENTRY
LFT  BARRY  IS IT AN ARRAY.
TRA  ASEQU5  YES.
REM  CAL 1  NO.
ASEQ4  LFT  BDOUB  RE-ENTRY FROM ARRAY LOOP.
       ALS 1  DOUBLE LENGTH IF BDOUB IS ON
       ADD ORIGIN  LENGTH + ORIGIN = HANG
       LDQ DELMAX
       TLQ ++2  TEST AC AGAINST MQ
       XCA  PUT LARGER IN AC
REM  LDQ  HANGMX  HANG VERSUS HANGMX
       TLQ ++2  PUT LARGER IN AC
       XCA  HANGMX = LARGEST HANG
REM  CLA*  CTCELL  GET PROPER COUNTER
       ADD ORIGIN  OVERALL RELOCATABLE ORIGIN OF SYMBOL
REM  ZEI  COMFLG  SKIP IF NOT IN COMMON STACK
       BCOMN+BLHSX  TURN ON COMMON BIT AND DEFINITION BIT.
       DEFINE  IS THIS STACK DEFINED.
       BLHSX  YES, TURN ON THE EQUIV DEFINED BIT
       STA EQUIV,1  RESTORE ENTRY
       STA EQUIV,1  NO, SET DEFINITION OF ENTRY IN EQUIV TABLE
REM  TIXI  ASEQU9,4,1  RETURN FOR ANOTHER ENTRY
REM  ASEQU5  PIA 2  ARRAY, GET LENGTH FROM DIMTBL
       PAX  CAL  DMTBL-1,2
       ANA =07777  SAVE ONLY ADDRESS
       TRA  ASEQU4
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASEQU3</td>
<td>PAX 14</td>
<td>FT103220</td>
</tr>
<tr>
<td></td>
<td>ASEQU8,4,0 TRANSFER IF THIS IS THE END OF THE CHAIN</td>
<td>FT103230</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>FT103240</td>
</tr>
<tr>
<td></td>
<td>CAL -3B2 NOT YET END OF STACK</td>
<td>FT103250</td>
</tr>
<tr>
<td></td>
<td>STP EQUBL,4 SET 'USED STACK' PREFIX BITS IN LINK WORD</td>
<td>FT103260</td>
</tr>
<tr>
<td></td>
<td>TXI ASEQU9,4,1 CONTINUE PROCESSING THIS STACK</td>
<td>FT103270</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>FT103280</td>
</tr>
<tr>
<td>ASEQU8</td>
<td>CLA HANGMX UPDATE THE COMMON OR VARIABLE COUNTER</td>
<td>FT103290</td>
</tr>
<tr>
<td></td>
<td>ADD* CTCELL</td>
<td>FT103300</td>
</tr>
<tr>
<td></td>
<td>STO* CTCELL</td>
<td>FT103310</td>
</tr>
<tr>
<td></td>
<td>TRA ASEQU6 RETURN FOR NEXT CHAIN</td>
<td>FT103320</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td>FT103330</td>
</tr>
<tr>
<td>EXASEQ</td>
<td>EQU *</td>
<td>FT103340</td>
</tr>
<tr>
<td>Instruction</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>TTL</td>
<td>POST PASS 1 - TVECTL</td>
<td>FT103350</td>
</tr>
<tr>
<td>*</td>
<td>THE FOLLOWING ROUTINE SCANS THE LIBRARY NAME TABLE (SLNTAB)</td>
<td>FT103360</td>
</tr>
<tr>
<td>*</td>
<td>TO DETERMINE WHICH LIBRARY ROUTINES ARE USED BY THE OBJECT</td>
<td>FT103370</td>
</tr>
<tr>
<td>*</td>
<td>PROGRAM. A LINEAR SEARCH IS MADE THROUGH SLNTAB AND EACH</td>
<td>FT103380</td>
</tr>
<tr>
<td>*</td>
<td>ROUTINE USED IS DEFINED AS A TRANSFER VECTOR ENTRY, THE FIRST</td>
<td>FT103390</td>
</tr>
<tr>
<td>*</td>
<td>HAVING THE DEFINITION 1, THE SECOND 2, ETC. THIS DEFINITION</td>
<td>FT103400</td>
</tr>
<tr>
<td>*</td>
<td>GOVERNS THE ORDER IN WHICH THE ELEMENTS OF THE TRANSFER VECTOR</td>
<td>FT103410</td>
</tr>
<tr>
<td>*</td>
<td>WILL APPEAR IN THE OBJECT PROGRAM. NOTICE THAT THIS ROUTINE</td>
<td>FT103420</td>
</tr>
<tr>
<td>*</td>
<td>COUNTS LIBRARY ROUTINES ONLY ... EXTERNAL (NON-LIBRARY)</td>
<td>FT103430</td>
</tr>
<tr>
<td>*</td>
<td>REFERENCES ARE COUNTED LATER ON IN POST PASS 1 (SPAS).</td>
<td>FT103440</td>
</tr>
<tr>
<td>SPACE</td>
<td>3</td>
<td>FT103450</td>
</tr>
<tr>
<td>TVECTL</td>
<td>AXT 0,1</td>
<td>FT103460</td>
</tr>
<tr>
<td></td>
<td>XR1 HAS NUMBER OF ROUTINES USED</td>
<td></td>
</tr>
<tr>
<td></td>
<td>XR4 IS TABLE POINTER</td>
<td>FT103470</td>
</tr>
<tr>
<td>AXT</td>
<td>-2,4</td>
<td>FT103480</td>
</tr>
<tr>
<td>TVEL1</td>
<td>SLNTAB-2,4</td>
<td>FT103490</td>
</tr>
<tr>
<td></td>
<td>IS THIS ROUTINE USED IN TEXT</td>
<td>FT103500</td>
</tr>
<tr>
<td>TVE2</td>
<td>TVEL2,1,1</td>
<td>FT103510</td>
</tr>
<tr>
<td>TXL</td>
<td>++2,4,-LSLNTB</td>
<td>FT103520</td>
</tr>
<tr>
<td>TXI</td>
<td>TVEL1,4,-2</td>
<td>FT103530</td>
</tr>
<tr>
<td></td>
<td>OTHERWISE BUMP POINTER AND GO BACK</td>
<td></td>
</tr>
<tr>
<td>SXA</td>
<td>LTVECT,1</td>
<td>FT103540</td>
</tr>
<tr>
<td></td>
<td>SAVE THIS (PARTIAL) COUNT IN LTVECT</td>
<td></td>
</tr>
</tbody>
</table>
* TTL POST PASS 1 - PROLEN
* THE FOLLOWING ROUTINE CALCULATES THE LENGTH OF THE OBJECT TIME
* PROLOGUE TO BE GENERATED FOR THIS PROGRAM.
* SPACE 2
* THIS VALUE IS DEPENDENT ON,
* REM
* 1. INDEX REGISTER SAVING REQUIREMENTS
* 2. THE CONTENTS OF THE PROLOGUE TABLE (P R L T B L)
* 3. THE PRESENCE OR ABSENCE OF ARITHMETIC STATEMENT FUNCTIONS
* SPACE 2
* THE PROLOGUE TABLE IS FIRST SORTED IN ORDER THAT ALL PROLOGUE
* TABLE ENTRIES FOR EACH ARGUMENT ARE TOGETHER (TO MINIMIZE LENGTH
* OF PROLOGUE).
* SPACE 2
* FORMAT OF P R L T B L ENTRIES,
* REM
* BITS (S,1-5) = SUBROUTINE ARGUMENT NUMBER
* (6-20) = ARGUMENT INCREMENT (DELTANO)
* (21-35) = LOCATION (RELATIVE 0) OF TEXT TO BE INITIALIZED
* SPACE 2
* SEE G E N P R L FOR A DISCUSSION OF THE ACTUAL LAYOUT OF THE
* OBJECT TIME PROLOGUE.
* SPACE 3
* PROLEN AXI 0,4 XR4 IS USED TO CONTAIN LPROLGN
* NZT SUBR IS THIS A SUBROUTINE COMPILATION
* TRA PROLA NO, EXIT THIS SECTION
* STZ ERASE4 ERASE4 = 0 IFF NO XR*S YET AT ALL
* LXA PROLOG+2,1 SET XR1 TO LENGTH OF PRLTB
* LXA PROLOG+2,2 SET XR2 TO LENGTH OF PRLTB
* TXX PROL9,1,1 GO GET XR USAGE IF PRLTB EMPTY
* SXX ++2,1 SAVE LENGTH OF PRLTB
* AXI PRLTB-1,2,1 CALCULATED ADDR. OF LAST ENTRY IN
* TXX ++3,1 PRLTB
* SXX ++3,1 SAVE THIS ADDR. IN SORT CALL SEQ.
* CALL SORT,PRLTB,... SORT PRLTB
* TXX ++1,1,1 XR1 IS LAST PRLTB LOC. + 1
* SXX PROL2,1 SVAE THIS IN PROL2
* STZ ERASE ERASE = ARG. NO. AND DELTA
* STZ ERASE1 ERASE1 = ARG. NO.
* STZ ERASE2 ERASE2 = DELTA
* STZ ERASE3 ERASE3 = 0 IFF NO XR*S YET THIS ARG.
* AXI 0,4 RESET XR4 AT BEGINNING
* TXX ++1,2,-1 SET XR2 TO LENGTH OF PRLTB
* CAL ++2 pick-up PRLTB ENTRY
* AXA ARGDEL MASK OUT PCOUNT
* LAX ERASE IS THIS SAME ARG. NO. AND DELTA
* TRA PROL4 NO, GO TO PROL4
* TRA PROL3 YES, GO TO PROL3
* MACER PRLTB OUT OF SORT
* REM ...
* PROL3 TXX ++1,4,1 BUMP LENGTH OF PROLOGUE BY 1
* TXX PROL2,2,1 STEP THROUGH PRLTB
* TRA PROL8 AT END OF PRLTB, GO GET XR NEEDS
* REM ...
* PROL4 SLX ERASE SAVE NEW ARG. NO. AND DELTA IN ERASE
* ANA ARGBIT MASK OUT DELTA
LAS ERASE1 COMPARE WITH LAST ARG. NO. FTI04140
TRI PROL7 GO TO PROL7 IF NEW ARG. NO. FTI04150
TRI PROL5 GO TO PROL5 IFF SAME ARG. NO. FTI04160
MACER PRLTBL OUT OF SORT FTI04170
REM \* TO HANDLE CHANGE IN DELTA ONLY FTI04180
REM FTI04190
REM FTI04200
PROL5 NZI ERASE3 HAVE XR*S BEEN USED FOR THIS ARG. NO. FTI04210
TXI **1,4,1 NO, BUMP LPROLG BY 1 FTI04220
CAL* PROL2 SET EARSE2 TO THIS NEW DELTA FTI04230
ANA DELBIT X FTI04240
SLW ERASE2 X FTI04250
STL ERASE3 INDIcate XR USED FOR THIS ARGUMENT FTI04260
PROL6 TXI PROL3+1,4,2 BUMP LPROLG BY 2 AND STEP THRU PRLTBL FTI04270
REM \* TO HANDLE CHANGE IN ARG. NO. FTI04280
REM FTI04290
REM FTI04300
PROL7 ZET ERASE3 WAS XR USED FOR LAST ARGUMENT FTI04310
STL ERASE4 YES, TURN EARSE4 ON FTI04320
SLW ERASE1 SAVE NEW ARGUMENT NUMBER FTI04330
XEC PROL2 GET THIS PRLTBL ENTRY AGAIN FTI04340
ANA DELBIT SAVE THIS DELTA IN EASE2 FTI04350
SLW ERASE2 X FTI04360
STZ ERASE3 INDICATE NO XR*S THIS ARGUMENT YET FTI04370
ZET ERASE2 FL IS THIS DELTA ZERO FTI04380
TXI PROL6+1,4,2 NO, THEN BUMP LPROLG BY 2 FTI04390
TRA PROL6 YES, GO TO PROL6 AND BUMP LPROLG FTI04400
REM \* TO HANDLE COUNTING OF XR SAVING REQUIREMENTS FTI04410
REM FTI04420
REM FTI04430
PROL8 ZET ERASE3 TO HANDLE SPECIAL CASE WHERE LAST ARG FTI04440
STL ERASE4 USED XR BUT NO OTHER ARGS, DID FTI04450
PROL9 SXA LPBODY,4 SET LBODY TO LENGTH OF PROLOGUE BODY FTI04460
NZI RTNFLG DID A 'RETURN' STATEMENT OCCUR FTI04470
TRA PROLA-2 NO, DO NOT SAVE XR*S IN PROLOGUE FTI04480
CAL* XRFLAG GET TEXT XR NEEDS FTI04490
ZET ERASE4 OR IN PROLOGUE XR NEEDS FTI04500
ORA TGBIT X FTI04510
SLW XRFLAG FTI04520
PAI COUNT NUMBER OF TOTAL XR*S REQUIRED FTI04530
AXT 0,2 BUILD THIS COUNT IN XR2 FTI04540
RFI 400000 X FTI04550
TXI **1,2,1 X FTI04560
RFT 200000 X FTI04570
TXI **1,2,1 X FTI04580
RFT 100000 X FTI04590
TXI **1,2,1 X FTI04600
SXA NOXRS,2 SAVE THIS COUNT IN NOXRS FTI04610
TXL PROLA-3,2,0 GO TO PROLA-3 IF NO XR*S NEED SAVING FTI04620
PXD +2 OTHERWISE BUMP LPROLG BY 2 FOR EACH FTI04630
ALS 1 XR THAT NEEDS SAVING. FTI04640
STD **1 ONE FOR THE SXA AND ONE FOR THE AXI FTI04650
TXI **1,4,\ldots X FTI04660
TXI **1,4,1 BUMP LPROLG BY 1 FOR THE TRA (N),4 FTI04670
ZET LARSTS BUMP LPROLG BY 1 IF THIS IS A SUBR. FTI04680
TXI **1,4,1 AND THERE WERE ARITH. SIMNT. FCNS. FTI04690
PROLA SXA LPROLG,4 SAVE THIS FINAL COUNT IN LPROLG FTI04700
**TIL POST PASS I - SYMPASS**

* THE FOLLOWING ROUTINE CONSTITUTES THE POST PASS I SYMBOL TABLE

* (SYMTAB / EQU IV) PASS. THIS ROUTINE HAS TWO MAIN

* FUNCTIONS,

REM FTI04750

* 1. DEFINE LOW CORE VARIABLE STORAGE, FORMATS AND EXTERNAL

* TRANSFER VECTOR ENTRIES AND,

* 2. COLLECT ALL UNEDEFINED LABELS, UNEDEFINED STRINGS, UNEDEFINED

* VARIABLES AND STATEMENTS WITH NO PATH OF FLOW FOR LATER

* OUTPUT.

REM FTI04770

SPACE 3 FTI04810

* EACH EQU IV ENTRY IS EXAMINED FOR WHICH THE ASSOCIATED

* SYMTAB ENTRY IS NON-ZERO. THE FOLLOWING ACTIONS ARE TAKEN

* FOR THE VARIOUS EQU IV ENTRIES POSSIBLE,

SPACE 2 FTI04820

* EXTERNAL SYMBOL - A. EXTERNAL SYMBOLS ARE CHAINED TOGETHER

* USING THE DECREMENT OF THE EQU IV WORD.

REM FTI04830

* B. THE LENGTH OF THE TRANSFER VECTOR

* (LTVECT) IS BUMPED BY ONE AND THE

* SYMBOL IS GIVEN THIS DEFINITION (IN

* ADDRESS OF EQU IV WORD).

REM FTI04850

* C. IF THE FIRST CHARACTER OF THE SYMBOL IS

* '4' THIS CHARACTER IS CHANGED TO 'D'. IF

* THE FIRST CHARACTER IS '1' IT IS CHANGED

* TO 'I'.

REM FTI04870

* STATEMENT LABEL - A. IF THE STATEMENT LABEL IS UNEDEFINED IT IS

* STACKED WITH OTHER UNEDEFINED LABELS.

REM FTI04880

* B. IF THE STATEMENT LABEL HAS NO PATH OF

* FLOW TO IT THIS LABEL IS STACKED WITH

* OTHER LABELS OF THIS TYPE.

REM FTI04890

* FORMAT (STRING) - A. IF THE FORMAT IS UNEDEFINED IT IS STACKED

* WITH OTHER UNEDEFINED FORMATS.

REM FTI04900

* B. IF NOT UNEDEFINED, THIS STRING IS CHAINED

* TO PREVIOUS STRINGS USING THE DECREMENT

* OF FIRST POOL ENTRY FOR THIS STRING.

REM FTI04910

* C. THE STRING IS DEFINED (IN ADDRESS OF

* EQU IV) TO BE (LENGTH OF STRINGS SO

* FAR (LSTRIN)) + (LENGTH OF THIS

* STRING) - 1.

REM FTI04920

* D. LSTRIN IS BUMPED BY THE LENGTH OF

* THIS STRING.

REM FTI04930

* SUBROUTINE ARGUMENT - A. NO ACTION TAKEN BY SYMPASS.

REM FTI04940

* INTEGER VARIABLE - A. IF THE VARIABLE IS UNEDEFINED AND

* BVARB IS ON (INDICATING THAT IT WAS

* USED IN A VARIABLE CONTEXT) THIS VARIABLE LEFT

* IS STACKED WITH OTHER UNEDEFINED VARIABLES.

REM FTI04950

* B. IF DEFINED OR BVARB OFF, A TEST FOR

* FOR THE APPEARANCE OF THIS VARIABLE IN AN...

REM FTI04960

* EQUIVALENCE STATEMENT IS MADE.
C. SUCCESSIVE TESTS ARE MADE FOR THE VARIABLE'S APPEARANCE IN A COMMON
AND WHETHER OR NOT IT IS AN ARRAY
AND WHETHER OR NOT IT IS A '2-CELL'
COMPLEX. IN ANY CASE, THE VARIABLE IS DEFINED APPROPRIATELY IN THE ADDRESS OF IT'S EQUIV ENTRY.

REAL VARIABLE - A. SAME ACTION AS THAT FOR INTEGER VARIABLE.
NONE OF THESE - A CATASTROPHIC ERROR MESSAGE IS ISSUED.

SPACE 3

SYPASS AXT 102,1 ZERO OUT THE ERROR CELLS
STZ CARD+102,1
TIX *=1,1
REM
CAL LVARBS SET IVARBS TO VALUE-1 OF LVARBS ON
SUB *=1 ENTRY TO SYMBOL TABLE PASS
SLW IVARBS
AXT LSIM,1 SET XR1 TO LENGTH OF SYMBOL TABLE
ZEI SYMAB,1 IS THIS SYMBOL TABLE ENTRY ZERO
TRA ++3 NO, GO EXAMINE EQUIV ENTRY
SPASSI TIX *=2,1,1 YES, TIX THROUGH SYMBOL TABLE
TRA ASCMN2 GO GET LCOMMN
STZ ERASE1 INITIALIZE DOUBLE/COMPLEX SWITCH
LDI EQUIV,1 PUT EQUIV ENTRY IN INDICATORS
IIL 700000 INVERT LEFT-MOST 3 BITS
LFT BEXF IS THIS AN EXTERNAL SYMBOL
TRA SPASS4 YES, GO PROCESS IT
LNT /MLABL*70000 NO, IS THIS A LABEL
TRA SPASSA NO, GO SEE IF IT IS A STRING
LNT BLHSX YES, HAS THIS LABEL BEEN DEFINED
ERRORY 0 UNDEFINED LABEL
LNT BPATH YES, IS THIS LABEL IN PATH OF FLOW
ERRORY 3 NO PATH OF FLOW
TRA SPASS1 YES, GO CONTINUE EQUIV PROCESSING
REM
SPASSA LNT /MSTRG*70000 IS THIS A STRING
TRA SPASSB NO, GO SEE IF IT IS A SUBR. ARGUMENT
LNT BLHSX YES, HAS THE STRING BEEN DEFINED
ERRORY 1 UNDEFINED STRING
PIA ACC = EQUIV ENTRY
PAX ,4 XR4 = POOL POINTER FOR THIS STRING
CAL POOL,4 SET ERASE = LENGTH OF THIS STRING
SLW ERASE X
LXA STSM,1 GET POINTER TO LAST STRING ENCOUNTERED
TXL ++3,2,0 TRANSFER IFF THIS IS FIRST STRING
PXD ,4 OTHERWISE CHAIN THIS STRING TO LAST
STD POOL,2 X
CAL LSTRIN GET LENGTH OF STRINGS SO FAR
ACL ERASE DEFINE THIS STRING
SUB *=1 DEF. IS LSTRIN+LEN-1
STA EQUIV,1 THIS IS DEFINITION OF THIS STRING
STD POOL,4 SET LAST DECR. IN CHAIN TO ZERO
SXA STSM,4 SAVE POOL POINTER FOR NEXT LINK
TXH ++2,2,0 TRANSFER IF THIS IS OTHER THAN FIRST
SXA STSM,4 SAVE POOL POINTER OF FIRST STRING
ADD *=4
SLW LSTRIN X
<table>
<thead>
<tr>
<th>TRA</th>
<th>SPASS1</th>
<th>GO CONTINUE PROCESSING</th>
<th>FT105910</th>
</tr>
</thead>
<tbody>
<tr>
<td>LFI</td>
<td>BARGT</td>
<td>IS THIS A SUBROUTINE ARGUMENT</td>
<td>FT105920</td>
</tr>
<tr>
<td>TRA</td>
<td>SPASSE</td>
<td>YES, GO CHECK IF ALSO IN COMMON</td>
<td>FT105930</td>
</tr>
<tr>
<td>LNT</td>
<td>/MINTG=700000</td>
<td>NO, IS THIS AN INTEGER VARIABLE</td>
<td>FT105940</td>
</tr>
<tr>
<td>TRA</td>
<td>SPASSD</td>
<td>NO, GO SEE IF REAL VARIABLE</td>
<td>FT105950</td>
</tr>
<tr>
<td>LFT</td>
<td>BLHSX</td>
<td>YES, HAS IT BEEN DEFINED</td>
<td>FT105960</td>
</tr>
<tr>
<td>TRA</td>
<td>$++3</td>
<td></td>
<td>FT105970</td>
</tr>
<tr>
<td>LFT</td>
<td>BVARB</td>
<td></td>
<td>FT105980</td>
</tr>
<tr>
<td>ERRORY</td>
<td>2</td>
<td>UNDEFINED VARIABLE</td>
<td>FT105990</td>
</tr>
<tr>
<td>LFI</td>
<td>BEQUV</td>
<td>YES, DOES IT APPEAR IN AN EQUIVALENC</td>
<td>FT106000</td>
</tr>
<tr>
<td>TRA</td>
<td>SPASSG</td>
<td>YES, GO SEE IF COMMON VARIABLE</td>
<td>FT106010</td>
</tr>
<tr>
<td>LNT</td>
<td>BCOMM</td>
<td>NO, DOES IT APPEAR IN COMMON STMNT</td>
<td>FT106020</td>
</tr>
<tr>
<td>TRA</td>
<td>SPASSZ</td>
<td>NO, GO SEE IF IT IS AN ARRAY</td>
<td>FT106030</td>
</tr>
<tr>
<td>SPASSF</td>
<td>ERASE</td>
<td>YES, SAVE THE INDICATORS</td>
<td>FT106040</td>
</tr>
<tr>
<td>LOI</td>
<td>EQUIV,1</td>
<td>RESET INDICATORS TO EQUIV ENTRY</td>
<td>FT106050</td>
</tr>
</tbody>
</table>
POST PASS 1 PROCESSOR FOR NON-EQUIVALENCE COMMON VARIABLES

ENTRY... ASCMNX AND ASCMN2

THIS PROCESSOR IS USED AFTER THE VARIABLES IN THE
EQUIVALENCE STACK EQUTBL HAVE BEEN ASSIGNED TO STORAGE.
AND DURING THE POST PASS 1 SYMBOL TABLE SCAN.
THIS ROUTINE IS ENTERED AT LOCATION ASCMNX FOR EACH NON-
EQUIVALENCE COMMON VARIABLE. THE CORRESPONDING ENTRY IN DMTBL
(FIRST WORD) IS STACKED IN THE COMMON STACK COMTBL (LOCATED
OVER 'COLUMN' BEYOND THE LITERALS).

AT THE END OF THE SYMBOL TABLE SCAN, THE ROUTINE IS AGAIN
ENTERED AT LOCATION ASCMN2, AND COMTBL IS SORTED ON THE VALUE
OF THE COMMON POINTER CPOINT (LOCATED IN THE
DECREMENTS). EACH ENTRY IN COMTBL, IN ORDER, IS ASSIGNED
TO COMMON STORAGE. ON EXIT, THE COMMON COUNTER LCOMM
HAS THE ADDRESS (RELATIVE TO TOP OF COMMON) OF THE NEXT
AVAILABLE STORAGE CELL.

THE STORAGE ASSIGNMENT (DEFINITION) OF THE VARIABLE IS
PLACED IN THE EQUIV WORD ADDRESS. NOTE THAT A POSSIBLE
POINTER TO THE DMTBL WILL BE LOST AT THIS POINT.

ASCMNX PIA

GET E Q U I V W O R D

PAX +2
SET POINTER

CAL DIMTBL,2
GET COMMON INFORMATION FROM DMTBL

REM

LXA COMCNT,2
GET COMMON TABLE POSITION POINTER

TXI ++1,2,1
BUMP

SXA COMCNT,2
RESTORE POINTER

REM

TXL ++2,2,LCOLMN TEST FOR TABLE OVERFLOW

REM

STL MSG100,,FINISH

REM

SLW COLUMN+666,2 PUT COMMON ENTRY INTO THE COMTBL

REM

IRA SPASS1
EJECT
LNT  BARRY  IS THIS AN ARRAY  FT06440
PIA  SPASS3  NO, GO SET EQUIV ENTRY AND BUMP LVARBS FT06480
PAX  SPASS3  YES, SET XR2 TO DIMITL ENTRY LOCATION FT06490
CAL  LVARBS  GET CURRENT LVARBS  FT06500
ACL  DIMITL-1,2  LVARBS + ARRAY SIZE  FT06520
LFT  BDOUB  IS IT A DOUBLE-LENGTH VARIABLE?  FT06530
ACL  DIMITL-1,2  YES. DOUBLE THE STORAGE ALLOCATION  FT06540
STK  LVARBS  RECORD THE NEW LVARBS  FT06550
SUB  =1  REDUCE THE COUNT  FT06560
STA  EQUIV,1  DEFINE THE ARRAY IN EQUIV  FT06570
TRA  SPASS1  GO CONTINUE EQUIV PROCESSING  FT06580
CAL  LVARBS  GET NUMBER LOW-CORE VARIABLES SO FAR  FT06590
ACL  ONE  BUMP LVARBS BY 1  FT06600
LFT  BDOUB  IS THIS A 2-CELL VARIABLE  FT06610
ACL  ONE  YES, BUMP LVARBS BY 1  FT06620
TRA  SPASSK  FT06630

SPASS4
LNT  BARGT  IS THIS EXTERNAL SYMBOL AN ARGUMENT.  FT06640
TRA  SPASS1  YES. SKIP OUT  FT06650
LXX  TVS1,4  CHAIN EXTERNAL REFERENCES THRU EQUIV  FT06660
TXL  *3,4,0  TRANSFER IF THIS IS FIRST EXTERNAL  FT06670
FXD  1  PUT CURRENT EQUIV POINTER INTO  FT06680
STD  EQUIV,4  DECREMENT OF LAST EQUIV ENTRY FOR  FT06690
REM  EXTERNAL  FT06700
CAL  LTVECT  SET ADDRESS OF THIS EQUIV ENTRY  FT06710
ACL  ONE  BUMP LTVECT BY ONE  FT06720
SLW  EQUIV,1  TO T.V. ELEMENT NUMBER  FT06730
SXA  TVS1,1  SAVE CURRENT EQUIV POINTER FOR NEXT  FT06740
STA  LTVECT  FT06750
TXH  *2,4,0  SAVE POOL PTR. OF FIRST T.V. ELEMENT  FT06760
SXA  TVS2,1  FT06770
LDI  SYMTAB,1  IF FIRST CHARACTER OF EXTERNAL NAME  FT06780
LFT  700000  IS 4 OR 1 CHANGE THIS CHARACTER TO  FT06790
TRA  SPASS1  'D' OR 'I' RESPECTIVELY  FT06800
SIL  200000  X  FT06810
LFT  10000  X  FT06820
SIL  100000  X  FT06830
STI  SYMTAB,1  X  FT06840
TRA  SPASS1  GO CONTINUE EQUIV PROCESSING  FT06850

SPASSD
LNT  /MREAL*700000  IS THIS A REAL VARIABLE  FT06860
ERROR  4  INVALID EQUIV ENTRY  FT06870
TRA  SPASSC  YES, GO PROCESS THIS REAL VARIABLE  FT06880
REM  FT06890

SPASSG
LFT  BCOMN  FT06900
TRA  SPASS1  FT06910
PIA  =077777  FT06920
ANA  SSM  FT06930
ADD  IVARBS  FT06940
STA  EQUIV,1  FT06950
TRA  SPASS1  FT06960

SPASSE
LFT  BCOMN  IS THE ARGUMENT IN COMMON.  FT06980
TRA  SPASSF  YES. GO ENTER INTO THE COMMON STACK  FT06990
TRA  SPASS1  NO. NO FURTHER PROCESSING NECESSARY  FT07000
* TTL  POST PASS 1 - ASCMN2
* PASS OVER EQUIV TABLE COMPLETED. SORT THE COMMON STACK.
* ASSIGN STORAGE TO COMMON VARIABLES. STORAGE DEFINITION IS
* PLACED IN ADDRESS OF EQUIV WORD. NOTE THAT THE DIMTBL POINTER
* FOR ARRAYS IS LOST AT THIS POINT.

ASCMN2
NZT COMCNT WERE ANY COMMON VARIABLES PUT IN TABLE.
TRA OBJJEN NO, GO TO OBJJLEN
REM
REM
REM
LAC COMCNT,1 GET NUMBER OF ITEMS IN COMMON TABLE
TXI ++1,1,COLUMN+666 GET ABSOLUTE ADDRESS OF END OF TABLE
SXU ++2,1
REM
CALL SORT,***,COLUMN+666-1 SORT THE COMMON TABLE
REM
REM
REM
REM
REM
LXI COMCNT,1 GET NUMBER OF COMMON VARIABLES IN STACK
REM
REM
ASCMN4
CAL COLUMN+666,1 GET COMMON STACK ENTRY
PAX +4 GET EQUIV POINTER
REM
PAX CAL EQUIV,4 GET EQUIV ENTRY FOR THIS COMMON VARIABLE
REM
PAX +2 GET DIMTBL POINTER
CAL LCOMMN
STA EQUIV,4 ASSIGN STORAGE POSITION TO THIS VARIABLE
LFI BARRY IS IT AN ARRAY.
TRA ASCMN3 YES
REM
CAL +1 LENGTH IS 1
REM
REM
ASCMN6
LFT BDOUB IS IT DOUB. PREC. OR COMPLEX.
ALS 1 YES. DOUBLE LENGTH OF STORAGE
REM
ADD LCOMMN BUMP COMMON COUNTER
STO LCOMMN
TIX ASCMN4,1,1 RETURN FOR ANOTHER COMMON TABLE ENTRY
REM
TRA OBJJEN AT END, GO TO OBJJLEN
REM
ASCMN3
CAL DIMTBL-1,2 GET LENGTH OF ARRAY
ANA +077777 SAVE ADDRESS FIELD (LENGTH) ONLY
TRA ASCMN6
* POST PASS 1 - OBJLEN
* THE FOLLOWING ROUTINE CALCULATES THE QUANTITIES,
* (COMMON BREAK)
* CMNBRK = TOPCOM - LCOMMN
* LTVPRL = LTVECT + LPROLG
* (TRANSFER VECTOR + PROLOGUE)
* LTVPRT = LTVPRL + PCOUNT
* (LENGTH THRU TEXT)
* LTPTSC = LTVPRT + LCONST
* (LENGTH THRU CONSTANTS)
* LTHRUS = LTPTSC + LSTRIN
* (LENGTH THRU STRINGS)
* LTHRuw = LTHRUS + LWORKS
* PGMBRK = LTHRuw + LVARBS
* (PROGRAM BREAK)
* IN ADDITION TO THIS, THE OBJECT PROGRAM ENTRY POINT (ENTRY)
* IS CALCULATED. ALSO, IF PGMBRK EXCEEDS CMNBRK, A
* CATASTROPHIC ERROR MESSAGE IS ISSUED.
* FINALLY, THE VARIOUS WORDS FOR THE PROGRAM CARD ARE FORMATTED
* AND READIED FOR PUNCHING (THIS CARD IS NOT ACTUALLY PUNCHED)
* UNTIL JUST BEFORE THE TRANSFER VECTOR IS 'STASHED').
* (COMMON BREAK)

OBJLEN  CAL  TOPCOM
SUB    LCOMMN
SLW    CMNBRK
LXX    LWORKS,4
TXI    **1,4,-1
SGD    FLAG#2,4
CAL    LTVECT
ACL    LPROLG
SLW    LTVPRL
ACL    PCOUNT
SLW    LTVPRT
ACL    LCONST
SLW    LTPTSC
ACL    LSTRIN
SLW    LTHRUS
ACL    LWORKS
SLW    LTHRuw
ACL    LVARBS
SLW    PGMBRK
LAX    CMNBRK
ERROR 32,OUTUND
NODP X
SUB    =1
ALS    18
STD    FLGN2
CAL    PCOUNT
SUB    LARSTS
SLW    LTEXTS

* OBJECT PROGRAM TOO LONG
CAL  LTVECT  CALCULATE ENTRY POINT RELATIVE TO ZERO@FT108050
NZT  SUBR  IF MAIN PROGRAM, FT108060
TRA  OBJL1 ENTRY = LTVECT + LARSTS FT108070
ACL  NOXRS IF SUBPROGRAM, FT108080
ZET  RTNFLG NO XR'S SAVED IF NO RETURN FT108090
ACL  ONE ENTRY = LTVECT + NOXRS + 1 FT108100
NZT  SNAME X FT108110
OBJL1 ACL LARSTS X FT108120
STA  ENTRY X FT108130
REM  THE FOLLOWING INSTRUCTIONS SET-UP THE PROGRAM CARD. FT108140
REM  THE CARD IS ACTUALLY PUNCHED IN THE POST PASS 1 DRIVER. FT108150
REM  THE FOLLOWING INSTRUCTIONS SET-UP THE PROGRAM CARD. FT108160
REM  THE CARD IS ACTUALLY PUNCHED IN THE POST PASS 1 DRIVER. FT108170
ACL  LTVECT FT108180
ALS  18 FT108190
ACL  PGMBRK FT108200
SLW  PCWD3 FT108210
ACL  PCWD1 FT108220
ACL  CMNBRK FT108230
ACL  SNAME FT108240
ACL  ENTRY FT108250
SLW  PCWD2 FT108260
TTL POST PASS 1 - OUTUND

* THE FOLLOWING ROUTINE FORMATS AND OUTPUTS ERRORS DETECTED IN
  SPACE 2

* SYMPASS OF THE FOLLOWING TYPES,
  REM

  1. STATEMENTS WITH NO PATH OF FLOW

  2. UNDEFINED VARIABLES

  3. UNDEFINED STRINGS (FORMATS)

  4. UNDEFINED STATEMENT LABELS

  SPACE 3

OUTUND AXT 4,1 SET TO PICK UP FOUR TYPES OF ERRORS
UND1 CLA TBUND+1,1
       PDX *2
       TXL UND5+1,2,0
       TXI **1,2,-1
       SXD UND3+2
       SXA UND5+1
       STA UND2
       CLA MSUND+1,1
       STQ UND2-4

AXT **3,4 SET UP THE RETURN FROM THE
       SXA EREX,4 PRINT SECTION OF THE ERROR ROUTINE.
       TRA ERR1,4 GO TO ERROR ROUTINE TO SET NOCODE, ETC.
       REM
       TSX LIST,4 RETURN HERE FROM ERROR ROUTINE
       PON **,**
       AXT 0,1
       AXT 0,2
       LDQ BLANKS

UND2 CLA **1
       STQ LIN10+1,2
       STQ LIN10,2
       TXI **1,2,-2
       TXI **1,1,1

UND3 TXH UND4+1,***
       TXH UND2,2,-16
       TSX LIST,4
       PZE LIN10+,16
       TRA UND2-2

REM

UND4 PXA *2
       PAC *2
       SXD **2,2
       TSX LIST,4

UND5 AXT **,1
       TX UND1,1,1
       AXT 16,1
       CAL BLANKS
       SLW LIN10+17,1
       TIX *=1,1
       TRA TSTPP2 GO TEST 'NOCODE'
       SPACE 3

PON UNMS4+,7
PON UNMS3+,5
PON UNMS2+,6
MSUND PON UNMS1+,6
REM
<table>
<thead>
<tr>
<th>UNMS1</th>
<th>BCI</th>
<th>6, ****</th>
<th>UNDEFINED STATEMENT LABELS-</th>
<th>FT108870</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNMS2</td>
<td>BCI</td>
<td>6, ****</td>
<td>UNDEFINED FORMAT NUMBERS-</td>
<td>FT108880</td>
</tr>
<tr>
<td>UNMS3</td>
<td>BCI</td>
<td>5, ****</td>
<td>UNDEFINED VARIABLES-</td>
<td>FT108890</td>
</tr>
<tr>
<td>UNMS4</td>
<td>BCI</td>
<td>7, ****</td>
<td>STATEMENTS WITH NO PATH OF FLOW-</td>
<td>FT108900</td>
</tr>
</tbody>
</table>
**TTL POSTPASS I - TSTPST**

*THE FOLLOWING ROUTINE PUNCHES THE OBJECT PROGRAM SYMBOL TABLE*

*INTO COLUMN BINARY CARDS AS REQUIRED BY THE FMS-II OBJECT TIME*

*DEBUGGING PACKAGE. THE FORMAT OF THESE CARDS CAN BE FOUND IN,*

*IBM 7090/7094 PROGRAMMING SYSTEMS - FORTRAN II OPERATIONS*

*(PUBLICATION NO. C-80-6066-4)*

*REM*

*THE SYMBOLS REQUIRED IN THIS TABLE ARE ASSEMBLED INTO THE*

*LOW-CORE PORTION OF THE POOL REGION (i.e., BEGINNING AT LOCATION*

*COLUMN M AND PROCEEDING UPWARD). THESE ARE THEN SORTED,*

*FORMATTED AND OUTPUT ON SYSCON.*

**SPACE 3**

**TSTPST**

LDI PRGBOX WAS 'SYMBOL TABLE' CC ENCOUNTERED

RNT STC BY SYSTEM MONITOR

TRA OUTPGC-1 NO, GO PUNCH PRGRM CARD

CLS NXTLOC YES, PREPARE TO PUNCH SYMBOL TABLE

ALS 18 GET LENGTH OF AVAILABLE POOL REGION

ADD MAXNL C SAVE THIS IN PST03

STD PST03 POOL POINTER FOR STORING SYMBOLS

AXT LSymb,1 PREPARE TO SCAN COMPLETE SYMBOL TABLE

AXT 0,2 POOL POINTER FOR STORING SYMBOLS

AXT 0,4 2 X (NO. SYMBOLS STORED)

PST01 ZET SYMTAB,1 IS THERE A SYMTAB ENTRY HERE

TRA ++3 YES, SKIP NEXT COUPLE OF INSTRUCTIONS

PST02 TIA PST01,1,1 NO, SCAN ALL OF SYMTAB

TRA PST05 AT END OF SYMTAB GO TO PST05

**SPACE 2**

LDI EQUV,1 GET EQUV ENTRY FOR THIS SYMBOL

LFT 700000 IS THIS AN EXTERNAL FORMULA NUMBER

TRA PST04 NO, GO SEE IF IT IS A VARIABLE

SIL 300000 YES, SET PREFIX = 3

CAL SYMTAB,1 DO NOT ENTER IF A T. V. ELEMENT

ANA =0600000000000 X

TNZ PST02 X IS THERE ROOM FOR THIS ENTRY IN POOL

TXI **1,4,2** IS THERE ROOM FOR THIS ENTRY IN POOL

PST03 TXH PST15,4,... EXIT IF POOL REGION OVERFLOW

RIL 77777 YES, CLEAR OUT EQUV GARBAGE

RIR 700000 X

STI COLUMN+1,2 STORE THIS EQUV WORD IN POOL REGION

CAL SYMTAB,1 NOW GET THE ASSOCIATED SYMBOL

SLW COLUMN+2, AND PUT IT INTO THE POOL REGION ALSO

TXI PST02,2,-2 BUMP POOL PTR AND GO F0E NEXT SYMBOL

PST04 LNT 200000 IS THIS SYMBOL A VARIABLE

TRA PST02 NO, (STRING) IGNORE THIS SYMBOL

LFT BARGT DU NOT ENTER IF SUBPROGRAM ARGUMENT

TRA PST02 X

RIL 700000 YES, TURN OFF PREFIX BITS

LFT BCOUN IF COMMON SET PREFIX = 7

SIL 200000 IF NON-COMMON SET PREFIX = 5

SIL 500000 X

TRA PST03-1 GO STORE THIS SYMBOL

**SPACE 3**

PST05 SXD PST08+2 CALCULATE UPPER LIMIT OF POOL USED

TXL OUTPGC-1,2,0 EXIT IF SYMBOL TABLE NULL

AXT COLUMN+2 X

SXD **1,4** X

PST16 TXI **1,2,1** X

TXI **1,2,-1** SAVE THIS IN SORT CALL

SXA **3,2**
CALL SORT1,COLUMN+1,...
CAL LABEL SAVE OBJECT DECK LABEL
SLW ERASE X
CAL =HST SET-UP FOR 'ST' LABEL ON SYMTAB CARDS
SLW LABEL X
TSX CDLABL,4 SET-UP COLS. 73-80
CAL SNAME GET SUBPROGRAM NAME (OR ZERODES)
SLW 9LEFT+2 THIS IS WORD 3 OF FIRST SYMTAB CARD
LXD PST05+3,1 GET NUMBER OF SYMBOLS IN SYMBOL TABLE
PXD ,1 X
ARS 1 X
ATX -000000000000
SLW 9LEFT+3 THIS IS WORD 4 OF FIRST CARD
ACL 9LEFT+2 GET (PARTIAL) CHECKSUM
SLW SUM X
ATX 20,4 SET TO STASH SYMBOLS AND EQUIV'S
ATX 0,2 INTO BINARY CARD IMAGES
PST07 CALL COLUMN,2 GET NEXT SYMBOL
SLW 9LEFT+24,4 STORE THIS IN CARD IMAGE
ACL SUM ACCUMULATE SUM
SLW SUM X
LDI COLUMN+1,2 SET IND AND LAC TO EQUIV ENTRY
CAL COLUMN+1,2 X
ANA -07777 CLEAR OUT PREFIX
LFT 4000000 IS THIS AN EXTERNAL FORMULA NUMBER
TRA PST12 NO, GO TO PST12
ACL LTVPRL BUMP BY LENGTH OF T.V. AND PROLOGUE
ORA =02000000400000 PUT IN IBM'S REQUIRED BITS
SLW 9LEFT+25,4 PUT THIS INTO CARD IMAGE
ACL SUM ACCUMULATE CHECKSUM
SLW SUM X
PST08 TXL PST09+2,2 ARE THERE MORE SYMBOLS TO GO
TXL PST09+2,2 NO, GO TO PST09
TXI PST07+4,2 IS THIS CARD FULL - NO, GO TO PST07
CAL STW SET WORD 1 FOR IBM
SLW 9LEFT X
IOS20 EQU * WRITE *SYSBIN,BINARY,9LEFT,24,COL73,3 WRITE THIS CARD
STL 10EX
BRA WRITE,,**4
PTW SYSBIN,,BINARY
PZE 9LEFT,,24
MZE COL73,,3
END20 REM
TSX CDLABL,4 BUMP CARD SEQUENCE NUMBER
AXT 22,4 SET FOR 11 SYMBOLS/CARD
STZ SUM RESET CHECKSUM
TRA PST07 GO START ON NEXT CARD
SPACE 2
PST09 AXT 0,1 AT END FILL LAST CARD WITH BLANKS
TXI **1,4,-2 X
TXL PST10+4,0 X
STZ 9LEFT+24,4 X
STZ 9LEFT+25,4 X
TXI **1,1,-2 X
TIX -3,4,2 X
PST10 TXI **1,1,22 GET WORD COUNT
PXD ,1 X
ORA =0600500000000 COMPLETE WORD 1
SLW 9LEFT STORE THIS IN CARD IMAGE
<table>
<thead>
<tr>
<th>Label</th>
<th>Description</th>
<th>Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDLABL</td>
<td><strong>SXA LBLOUT,1</strong> SAVE XR*S</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td><strong>SXA LBLOUT+1,2</strong></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td><strong>SXA LBLOUT+2,4</strong> HAS TITLE BEEN CONVERTED TO COL. IMAGE</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td><strong>ZET TLSWTH</strong> YES, PUT SEQUENCE NO. IN CARD</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td><strong>STZ COL73</strong> NO, ZERO OUT COLUMNS 73 THRU 80</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td><strong>STZ COL73+1</strong></td>
<td>6</td>
</tr>
<tr>
<td></td>
<td><strong>STZ COL73+2</strong></td>
<td>7</td>
</tr>
<tr>
<td></td>
<td><strong>CAL LABEL</strong> PICK UP THE NEW NAME</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td><strong>TNZ ++3</strong> SKIP IF LABEL HAS BEEN FOUND</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td><strong>NZT SNAME</strong> NO LABEL. IS IT A SUBROUTINE.</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td><strong>CAL =H(MAIN)</strong> NO. SET LABELING TO 'MAIN PROGRAM'</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td><strong>SLW TMPL</strong> SAVE NAME</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td><strong>AXT 0,1</strong> COUNT TRAILING BLANKS</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td><strong>AXT 6,2</strong></td>
<td>14</td>
</tr>
<tr>
<td>CNTBLK</td>
<td><strong>ZAC</strong> COUNT TRAILING BLANKS</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td><strong>LGL 6</strong></td>
<td>16</td>
</tr>
<tr>
<td></td>
<td><strong>ERA =060</strong></td>
<td>17</td>
</tr>
<tr>
<td></td>
<td><strong>TXI ++1,1,-6</strong> ASSUME BLANK FOUND, COUNT WITH MINUS ARITH.</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td><strong>TZE ++2</strong></td>
<td>19</td>
</tr>
<tr>
<td></td>
<td><strong>AXT 0,1</strong> NON-BLANK FOUND, RESET COUNTER</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td><strong>TXI CNTBLK,2,1</strong> REPEAT LOOP FOR ALL SIX BCD CHARACTERS</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td><strong>TXL ++3,1,0</strong> SKIP IF NO BLANK CHARACTERS</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td><strong>TXH ++2,1,-24</strong> IF 4 OR MORE TRAILING BLANKS, RESET COUNTER</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td><strong>AXT -24,1</strong></td>
<td>24</td>
</tr>
<tr>
<td></td>
<td><strong>TXI ++1,1,-12</strong> CALCULATE NO. OF DIGITS IN SQNCE NO.</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td><strong>SXA NUMCNT,1</strong> SAVE COUNT OF DIGITS</td>
<td>26</td>
</tr>
<tr>
<td></td>
<td><strong>TXI ++1,1,36</strong> CALCULATE SHIFT NEEDED ON SEQUENCE NO.</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td><strong>SXA STNUM,1</strong> SAVE NO. OF LEADING CHARACTERS (MINUS 2)</td>
<td>28</td>
</tr>
<tr>
<td></td>
<td><strong>TXI ++1,1,12</strong> INITIALIZE CONVERSION ROUT. TO ACTUAL COUNT</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td><strong>AXT 3,4</strong></td>
<td>30</td>
</tr>
<tr>
<td></td>
<td><strong>AXT 24,2</strong></td>
<td>31</td>
</tr>
<tr>
<td></td>
<td><strong>LDQ TMPL</strong> BEGIN CONVERSION OF BCD TO COL. IMAGE.</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td><strong>SXA COLSFT,2</strong> INITIALIZE SHIFT INSTRUCTION</td>
<td>33</td>
</tr>
<tr>
<td></td>
<td><strong>ZAC</strong> LOOK UP COL. IMAGE IN TABLE</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td><strong>LGL 6</strong></td>
<td>35</td>
</tr>
<tr>
<td></td>
<td><strong>SLQ TMPL</strong> SAVE REST OF CHARACTERS TO BE CONVERTED.</td>
<td>36</td>
</tr>
<tr>
<td></td>
<td><strong>XCA</strong> CALCULATE LOCATION IN CONVERSION TABLE.</td>
<td>37</td>
</tr>
<tr>
<td></td>
<td><strong>ZAC</strong></td>
<td>38</td>
</tr>
<tr>
<td></td>
<td><strong>DVP =3</strong></td>
<td>39</td>
</tr>
<tr>
<td></td>
<td><strong>XCA</strong></td>
<td>40</td>
</tr>
<tr>
<td></td>
<td><strong>ADD TBLHD</strong></td>
<td>41</td>
</tr>
<tr>
<td></td>
<td><strong>STA PCKENT</strong></td>
<td>42</td>
</tr>
<tr>
<td>MPY</td>
<td>=12</td>
<td>X</td>
</tr>
<tr>
<td>-------</td>
<td>------</td>
<td>-----</td>
</tr>
<tr>
<td>XCA</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>STA</td>
<td>BTPIN</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCKENT</td>
<td>CAL</td>
<td></td>
</tr>
<tr>
<td>BTPIN</td>
<td>ARS</td>
<td>**</td>
</tr>
<tr>
<td></td>
<td>ANA</td>
<td>=07777</td>
</tr>
<tr>
<td>COLSFT</td>
<td>ALS</td>
<td>**</td>
</tr>
<tr>
<td>ORS</td>
<td>COL7+3,4</td>
<td>STORE IN CARD IMAGE</td>
</tr>
<tr>
<td>TXI</td>
<td>++1,2,-12</td>
<td>MOVE SHIFT POINTER</td>
</tr>
<tr>
<td>TNX</td>
<td>ENDP,2,36</td>
<td>RESET SHIFT POINTER AT END (-12)</td>
</tr>
<tr>
<td>TIX</td>
<td>++1,4,1</td>
<td>RESET COLUMN POINTER.</td>
</tr>
<tr>
<td></td>
<td>CNVT1,1,6</td>
<td>ARE ALL CHARACTERS CONVERTED AND STORED.</td>
</tr>
<tr>
<td>ZET</td>
<td>TLSWTH</td>
<td>YES. IF SEQUENCE NUMBER JUST INSERTED, RETURN</td>
</tr>
<tr>
<td>TRA</td>
<td>LBLOUT</td>
<td>TITLE JUST ADDED. SAVE SHIFT POINTER AND COLUMN POINTER.</td>
</tr>
<tr>
<td>SQA</td>
<td>PNT2,2</td>
<td>TITLE JUST ADDED. SAVE SHIFT POINTER AND COLUMN POINTER.</td>
</tr>
<tr>
<td></td>
<td>PNT4,4</td>
<td>COLUMN POINTER.</td>
</tr>
<tr>
<td>STL</td>
<td>TLSWTH</td>
<td>TURN OFF TITLE SWITCH</td>
</tr>
<tr>
<td>CLA</td>
<td>=-1</td>
<td>INITIALIZE SEQUENCE NUMBERING</td>
</tr>
<tr>
<td></td>
<td>SQSUM</td>
<td>X</td>
</tr>
<tr>
<td>SQNO</td>
<td>Cla</td>
<td>X</td>
</tr>
<tr>
<td>ADD</td>
<td>=1</td>
<td>CONVERT VIA TABLE LOOKUP</td>
</tr>
<tr>
<td>CVR</td>
<td>BCDCNV,,6</td>
<td>CONVERT VIA TABLE LOOKUP</td>
</tr>
<tr>
<td>PNT2</td>
<td>X</td>
<td>ZERO OUT PREVIOUS NUMBER IN CARD IMAGE.</td>
</tr>
<tr>
<td>AXT</td>
<td>++2</td>
<td>ZERO OUT PREVIOUS NUMBER IN CARD IMAGE.</td>
</tr>
<tr>
<td>CAL</td>
<td>=0777777777000</td>
<td>ZERO OUT PREVIOUS NUMBER IN CARD IMAGE.</td>
</tr>
<tr>
<td>SQA</td>
<td>++1,2</td>
<td>SET SHIFT COUNT</td>
</tr>
<tr>
<td>ALS</td>
<td>**</td>
<td>X</td>
</tr>
<tr>
<td>PNT4</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>AXT</td>
<td>++4</td>
<td>PICK UP COLUMN POINTER.</td>
</tr>
<tr>
<td>ANS</td>
<td>COL7+3,4</td>
<td>ZERO OUT PARTIAL WORD</td>
</tr>
<tr>
<td>TNX</td>
<td>STNUM1,4,1</td>
<td>ZERO OUT REMAINDER OF CARD</td>
</tr>
<tr>
<td>STZ</td>
<td>COL7+3,4</td>
<td>INITIALIZE CONVERSION ROUT. FOR SEQUENCE NOFT111550</td>
</tr>
<tr>
<td>TRA</td>
<td>=-2</td>
<td>SHIF’T OFF UNUSABLE LEADING BCD DIGITS.</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>SHIF’T OFF UNUSABLE LEADING BCD DIGITS.</td>
</tr>
<tr>
<td>NUMCN1</td>
<td>AXC</td>
<td>**1</td>
</tr>
<tr>
<td></td>
<td>PZ5</td>
<td>CONVIB</td>
</tr>
<tr>
<td>AXT</td>
<td>++2</td>
<td>RESTORE XR*S</td>
</tr>
<tr>
<td>AXT</td>
<td>**1</td>
<td>X</td>
</tr>
<tr>
<td>TRA</td>
<td>1,4</td>
<td>FT111620</td>
</tr>
<tr>
<td></td>
<td>1,4</td>
<td>FT111630</td>
</tr>
</tbody>
</table>
THE FOLLOWING MACRO WRITES THE PROGRAM CARD ON SYSBIN. THE VARIOUS WORDS IN THE PROGRAM CARD WERE FORMATTED IN ROUTINE

SET-UP FOR OBJECT DECK LABELLING

* OBSJLEN*

3

TSX

CDLABL,4

OUTPGC

WRITE

*SYSBIN,1,PCWD1,3,CMNBRK,1,SNAME,1,ENTRY,1,SUM,18,CO7L3,3

OUTPGC

STL

IOEX

BRA

WRITE,***8

PTW

SYSBIN,1

PZE

PCWD1,3

CMNBRK,1

SNAME,1

ENTRY,1

SUM,18

COL7L3,3

END22

REM
* POST PASS 1 - STVECT

* THE FOLLOWING ROUTINE STASHES THE TRANSFER VECTOR INTO BINARY

* CARD IMAGES. TRANSFER VECTOR ELEMENTS REPRESENTING LIBRARY

* ROUTINES ARE STASHED FIRST AND THEN NON-LIBRARY EXTERNAL

* REFERENCES ARE STASHED, THE NON-LIBRARY REFERENCES WERE CHAINED

* TOGETHER BY THE POST PASS 1 ROUTINE S Y P A S S .

* SPACE 2

* THIS ROUTINE HAS BEEN SET UP AS A SUBROUTINE BECAUSE A MODIFIED

* VERSION OF THIS ROUTINE IS USED BY ROUTINE M A P TO LIST THE

* TRANSFER VECTOR IN THE STORAGE MAP IN POST PASS 2.

* SPACE 3

**TSX** STVECT,4 STASH TRANSFER VECTOR

**TRA** GENPRL GO TO NEXT POST PASS 1 ROUTINE

**REM**

**REM**

**STVECT** NJT LTVECT IMMEDIATE EXIT IFF VECTOR IS EMPTY

**TRA** 1,4 X

**SXH** EXSTV,4 SAVE XR4 FOR RETURN

**AXT** -2,2 XR2 IS USED AS SPECIAL NAME TABLE PTR.

**STVI** CAL SLNTAB-1,2 GET NEXT TABLE ENTRY

**ZET** SLNTAB-2,2 IS THIS NAME USED IN OBJECT PROGRAM

**TXS** STASH,4 YES, GO STASH IT

**TXL** ++2,2,-LSLNTB NO, TRANSFER IFF AT END OF TABLE

**TXI** STVI,2,-2 BUMP POINTER AND GET NEXT NAME

**NJT** TVSW2 ARE THERE ANY OTHER VECTOR ELEMENTS

**TRA** EXSTV NO, GO EXIT THE SCENE, MAN

**LXA** TVSW2,2 GET PTR. TO FIRST TV EL. IN SYMTAB

**STV2** CAL SYMTAB,2 SET ACC TO THIS ELEMENT OF TV

**TSX** STASH,4 STASH THIS ELEMENT

**CAL** EQUIV,2 GET POINTER TO NEXT ELEMENT

**PDX** ,2 PUT THIS INTO XR2

**TXH** STV2,2,0 GO GET NEXT IFF THERE IS A NEXT

**EXSTV** AXT ... ,4 AT END, MAKE LIKE A B-I-G BIRD

**TRA** 1,4 AND FLY AWAY
THE FOLLOWING SUBROUTINE GENERATES THE OBJECT-TIME
CARD IS OUT ON SYSBIN AND THE TRANSFER VECTOR HAS BEEN STASHED.

THE PROLOGUE CODE IS BROKEN INTO THREE SECTIONS,
SECT. 1 = XR SAVING
SECT. 2 = BODY OF PROLOGUE
SECT. 3 = APPENDIX

SECTION 3 PRECEDES SECTIONS 1 AND 2 IN THE OBJECT-TIME

CODE. THE PROLOGUE THUS APPEARS AS,

APPEND AXT ***1 APPENDIX-- CONTAINS AXT*S AS

NECESSARY AND FINAL SUBR.
SUBROUTINE EXIT.
TRA (N+1),4 N = NO. SUBR. ARGUMENTS

ENTRY SXA APPEND,1 XR SAVING AS REQUIRED.
SXA APPEND+1,2 ENTRY IS SUBPROGRAM
SXA APPEND+2,4 ENTRY POINT

CAL 1,4 BODY OF PROLOGUE-- SETS
STA -- ADDRESSES IN TEXT AS
STA -- REQUIRED.
PAX ,1
STZ ++1,1,
STZ --,1
STZ --,1

SPACE 0

IS THIS A SUBROUTINE COMPILED
YES, EXIT FROM THIS SECTION
IS THERE A RETURN STATEMENT.
NO.
ARE ANY XR*S TO BE SAVED IN APPENDIX
YES, GO GENERATE SECTIONS 3 AND 1
OF THE PROLOGUE
SET ACC TO NO. OF SUBR. ARGUMENTS
SET ACC TO TRA (NOARGS),4
STASH THIS INSTRUCTION
IS SECTION 2 EMPTY
YES, GET READY TO EXIT THIS SECTION
ERASE = ARG. AND DELTA
ERASE2 = DELTA
ERASE3 = XR YET THIS ARG.
ERASE4 = PCOUNT
XR1 = PRRTBL + LPROL
SAVE THIS IN FOLLOWING CAL
XR2 = LENGTH OF PROLOG TABLE
SET XR2 TO LENGTH OF PRRTBL
PICK-UP NEXT PRRTBL ENTRY
SAVE PCOUNT FOR THIS ENTRY IN ERASE4
SAVE ARG. AND DELTA
IS IT SAME AS LAST ARG. AND DELTA
NO, GO TO GENP3
**GENP2**
YES, GO TO GENP2

**MẠCER**
PRLBL OUT OF SORT

**REM**
... TO HANDLE SAME ARGUMENT AND DELTA

**REM**

**GENP2**
ZET ERASE2 IS THIS DELTA NON-ZERO

**TRA**
GENP7 YES, GO TO GENP7

**CAL**
STA NO, GET STAREADY FOR STASHING

**ACL**
LTVPRL STA LTVECT+LPROLG

**ACL**
ERASE4 STA LTVECT + LPROLG + PCOUNT

**SSM**
INDICATE ADDRESS RELOCATION

**TSX**
STASH,4 GO STASH THIS INSTRUCTION

**TIX**
GENP1,2,1 GO GET NEXT PRRLBL ENTRY

**TRA**
GENP0 GET READY TO EXIT THIS SECTION

**REM**

**REM**
... TO HANDLE NEW ARG. AND/OR DELTA

**REM**

**GENP3**
SLW ERASE SAVE NEW ARG. AND DELTA

**ANA**
ARGVIT ISOLATE THIS ARG.

**LAS**
ERASE1 IS THIS A NEW ARGUMENT

**TRA**
GENP8 YES, GO TO GENP8

**TRA**
GENP4 NO, GO TO GENP4

**MẠCER**
PRLBL OUT OF SORT

**REM**

**REM**
... SAME ARG. BUT NEW DELTA

**REM**

**GENP4**
ZET ERASE3 ARE XR*S IN USE FOR THIS ARG.

**TRA**
GENP5 YES, GO TO GENP5

**STL**
ERASE3 NO, INDICATE THAT THEY ARE NOW

**CAL**
PAX1 STASH PAX ,1

**TSX**
STASH,4 X

**XEC**
GENP1 GET THIS PRLBL ENTRY AGAIN

**ANA**
DEBIT ISOLATE THIS NEW DELTA

**ALS**
3 PUT NEW DELTA IN ACC (3-17)

**SLW**
ERASE2 SAVE IN ERASE2

**GENP6**
ACL TXII GENERATE TXI PGMCTR+1,1,DELTA

**ACL**
PGMCTR X

**SSM**
INDICATE ADDRESS RELOCATION

**TSX**
STASH,4 STASH THIS TXI

**GENP7**
CAL SXA1 NOW GENERATE SXA (TEXT),1

**ACL**
LTVPRL X

**ACL**
ERASE4 X

**SSM**
INDICATE ADDRESS RELOCATION

**TSX**
STASH,4 STASH THIS SXA (TEXT),1

**TRA**
GENP3-2

**REM**

**REM**
... SAME ARG. BUT NEW DELTA WITH XR*S ALREADY IN USE

**REM**

**GENP5**
LDC ERASE2,1 XRI = -(OLD DELTA)

**XEC**
GENP1 GET PRLBL ENTRY AGAIN

**ANA**
DEBIT ISOLATE NEW DELTA

**ALS**
3 SHIFT NEW DELTA INTO ACC (3-17)

**SLW**
ERASE2 SAVE THIS IN ERASE2

**STO** ++1 X

**TXI** ++1,1,... GET (NEW DELTA - OLD DELTA)

**PXD** ,1 PUT THIS INTO ACC

**TRA**
GENP6 NOW GO STASH THIS TXI AND SXA

**REM**

**REM**
... TO HANDLE NEW ARGUMENT NUMBER

**REM**

**REM**

**REM**

**REM**
GENPB
SLW ERASE1 SAVE NEW ARG. IN ERASE1 FTI13360
XEC GENP1 GET THIS PRLTB ENTRY AGAIN FTI13370
ARS 30 SHIFT NEW DELTA INTO ACC(30-35) FTI13380
ACL CAL4 ACC = CAL (ARG NO.),4 FTI13390

XEC GENP1 GET PRDLB ENTRY AGAIN FTI13410
ANA ARGDEL SAVE NEW ARG. AND DELTA IN ERASE FTI13420
SLW ERASE X FTI13430
ANA DELBIT SAVE NEW DELTA IN ERASE2 FTI13440
TNZ GENP4+2 IF NON-ZERO DELTA, GO TO GENP4+2 FTI13450
SLW ERASE2 X FTI13460
STZ ERASE3 INDICATE NO XR'S YET THIS ARG. FTI13470
TRA GENP2+2 OTHERWISE GO GENERATE STA FTI13480
REM FTI13490
REM FTI13500
REM FTI13510

GENPB
CAL AXTI PUGENERATE AXT*S IN APPENDIX AS REQD. FTI13520
RFT 100000 X FTI13530
TSX STASH4 X FTI13540
CAL AXT2 X FTI13550
RFT 200000 X FTI13560
TSX STASH4 X FTI13570
CAL AXT4 X FTI13580
RFT 400000 X FTI13590
TSX STASH4 X FTI13600
CAL NOARGS GENERATE FINAL SUBR. EXIT (TRA (N),4) FTI13610
ACL TRA4 X FTI13620
TSX STASH4 X FTI13630
CAL ENTRY GET ADDRESS OF FIRST AXI FTI13640
SUB NOXR X FTI13650
SUB ONE X FTI13660
SLW ERASE FTI13670
RNT 100000 GENERATE SXA IFF XR1 TO BE SAVED FTI13680
IRA GENPB OTHERWISE GO TO GENPB FTI13690
ACL SXA1 X FTI13700
SSM INDICATE ADDRESS RELLOCATION FTI13710
TSX STASH4 GO STASH THIS INSTRUCTION FTI13720
CAL ERASE SET UP FOR NEXT XR FTI13730
ACL ONE X FTI13740
STA ERASE X FTI13750

GENPB
RNT 200000 GENERATE SXA --2 IF NECESSARY FTI13760
TRA GENPC OTHERWISE GO TO GENPC FTI13770
ACL SXA2 X FTI13780
SSM X FTI13790
TSX STASH4 X FTI13800
CAL ERASE SET UP FOR NEXT XR FTI13810
ACL ONE X FTI13820
STA ERASE X FTI13830

GENPB
RNT 400000 GENERATE SXA --4 IF NECESSARY FTI13840
TRA GENPA EXIT THIS SECT. GENPRL IF NOT NEC. FTI13850
ACL SXA4 X FTI13860
SSM X FTI13870
TSX STASH4 X FTI13880
TRA GENPA EXIT THIS SECTION FTI13890
REM FTI13900
REM FTI13910
REM FTI13920

GENPB
CAL LARSTS IF A. S. F. ARE INVOLVED IN THIS FTI13930
TZE PASS2 SUBPROGRAM THEN AT END OF PROLOGUE FTI13940
ACL LTVECT GENERATE TRANSFER AROUND THEM TO FTI13950
<table>
<thead>
<tr>
<th></th>
<th></th>
<th>ENTRY POINT OF TEXT PROPER</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACL</td>
<td>LPROLG</td>
<td>FT113960</td>
</tr>
<tr>
<td>ACL</td>
<td>TRA</td>
<td>X</td>
</tr>
<tr>
<td>SSM</td>
<td>X</td>
<td>FT113980</td>
</tr>
<tr>
<td>TSX</td>
<td>STASH,4</td>
<td>FT113990</td>
</tr>
<tr>
<td>TRA</td>
<td>PASS2</td>
<td>FT114000</td>
</tr>
</tbody>
</table>
* TTL POST PASS 1 SUBROUTINES - STDUMP

* THIS SUBROUTINE IS ENTERED ON ENTRY TO POST PASS 1 IF \textit{LIST SYM}.

* IS NON-ZERO. THIS CELL WILL BE NON-ZERO IF THE STATEMENT

* REM

* LIST SYMBOL TABLE

* REM

* APPEARED ANYWHERE IN THE SOURCE PROGRAM.

* SPACE 3

* AVG.

* THIS ROUTINE OUTPUTS ON \textit{MLSTAP} EACH SYMBOL, ITS RELATIVE

* LOCATION IN \textit{SYMTAB} AND ITS CORRESPONDING \textit{ENTRY}.

* REM

* THIS ROUTINE ALSO DUMPS THE USED PORTION OF \textit{POOL} (4 ENTRIES/

* LINE) IN OCTAL ONTO \textit{MLSTAP}.

* SPACE 3

* STDUMP SAVE STDUMX

* I0S23 EQU *

* WRITE *MLSTAP,DEC,STDUMP1,3

* STL IDEX

* BRA WRITE,++,3

* PTW MLSTAP,,DEC

* MZE STDUMP1,,3

* END23 REM

* I0S24 EQU *

* WRITE *MLSTAP,DEC,STDUMP10,10

* STL IDEX

* BRA WRITE,++,3

* PTW MLSTAP,,DEC

* MZE STDUMP10,,10

* END24 REM

* I0S25 EQU *

* WRITE *MLSTAP,DEC,STDUMP11,10

* STL IDEX

* BRA WRITE,++,3

* PTW MLSTAP,,DEC

* MZE STDUMP11,,10

* END25 REM

* I0S26 EQU *

* WRITE *MLSTAP,DEC,STDUMP12,10

* STL IDEX

* BRA WRITE,++,3

* PTW MLSTAP,,DEC

* MZE STDUMP12,,10

* END26 REM

* I0S27 EQU *

* WRITE *MLSTAP,DEC,STDUMP13,10

* STL IDEX

* BRA WRITE,++,3

* PTW MLSTAP,,DEC

* MZE STDUMP13,,10

* END27 REM

* I0S28 EQU *

* WRITE *MLSTAP,DEC,STDUMP14,1

* STL IDEX

* BRA WRITE,++,3

* PTW MLSTAP,,DEC

* MZE STDUMP14,,1

* END28 REM

* AXT 4096,1

* AXT 0,2

* ZET SYMTAB,2
*****

STDMPA TXI  **+1,2,1
      TIX  **-3,1,1
      TRA  PLDMPX

STDUMX RESTOR
      TRA  1#4

STDMP1 BCI  3;1SYMBOL TABLE DUMP
STDMP2 BCI  1;0
STDMP3 PXA  *2
      TSX  OCTBCD,4
      SLW  STDMP+1
      CAL  SYMTAB,2
      SLW  STDMP+2
      CAL  EQUIV,2
      LGR  33
      ALS  12
      ORA  =H 0
      SLW  STDMP5
      AXT  4,#4

STDMP7 ZAC
      LGL  1
      ALS  11
      LGL  1
      ALS  11
      LGL  1
      ORA  =H 0 0 0
      SLW  STDMP+4,#4
      TIX  STDMP7,#4,1
      ZAC
      LGL  6
      STQ  STDMP9
      TSX  OCTBCD,4
      ORA  =H 0 0 0
      SLW  STDMP8
      CAL  STDMP9
      ARS  21
      TSX  OCTBCD,4
      SLW  STDMP8+1

IOS29 EQU *
      WRITE *MLSTAP,DEC,STDMP+10
      STL IOEX
      BRA WRITE,#*,#+3
      PTW MLSTAP,**,DEC
      MZE STDMP4,**,10

END29 REM
      TRA  STDMPA

STDMP4 BCI  1
      PZE
      PZE
      STDMP5 PZE
      STDMP6 BSS 4
      STDMP8 BSS 2
      STDMP9 BSS 1

STM10 BCI  M D V L P A E C A G A E I
      STDM11 BCI  D O A H A S Q O R O X N
      STDM12 BCI  D U R S T S U M G T R T T A R G
      STDM13 BCI  LOCN SYMBOL E B B X H N V N T O Y F F NO ADDR
      REM
      REM
      * ROUTINE TO DUMP POOL REGION
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>FT115810</th>
<th></th>
<th>FT115820</th>
<th></th>
<th>FT115830</th>
<th></th>
<th>FT115840</th>
<th></th>
<th>FT115850</th>
<th></th>
<th>FT115860</th>
</tr>
</thead>
<tbody>
<tr>
<td>END34</td>
<td>REM</td>
<td>TRA STDUMX</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td></td>
<td>FT115810</td>
<td></td>
<td>FT115820</td>
<td></td>
<td>FT115830</td>
<td></td>
<td>FT115840</td>
<td></td>
<td>FT115850</td>
<td></td>
<td>FT115860</td>
</tr>
<tr>
<td>PLOMP1</td>
<td>BCI</td>
<td>3,600MP OF POOL AREA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLOMP4</td>
<td>BCI</td>
<td>1,</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PLOMP3</td>
<td>BES</td>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**THE FOLLOWING 'INSTRUCTIONS' CAUSE PUNCHING OF THE EDITOR**

**LBL 2FSTR**

**PROGRAM CARD FOR THE SECOND PART OF FASTRAN (FASTRAN RESIDES ON**

**THE SYSTEM TAPE AS TWO RECORDS --- NOS. 30 AND 31).**

**SPACE 3**

---

**BEGRC2 EQU ***

**ORG 0**

**9LP 5**

**IORT BEGRC2,,COLUMN-BEGRC2+1**

**TXI BEGINZ,,31 ENTRY PT,,PROGRAM NO.**

**ABS**

**ORG BEGRC2**
TTL PASS 2 - DESCRIPTION

REM SPACE 9

REM FASTRAN COMPILER
REM RESEARCH COMPUTING CENTER
REM INDIANA UNIVERSITY
REM BLOOMINGTON, INDIANA
REM SPACE 3

REM PASS 2

REM
EJECT
* ON ENTRY TO PASS 2 COMPILATION IS TO THE POINT WHERE THE PROGRAM
* CARD IS OUT AND THE TRANSFER VECTOR AND PROLOGUE HAVE BEEN
* 'STASHED' INTO CARD IMAGES.
* THE FOLLOWING IS AN OVERALL PICTURE OF PASS 2. (DETAILED COMMENTS)
* PRECEDE EACH SECTION.
* PASS 2 PERFORMS THE FOLLOWING FUNCTIONS IN THE INDICATED ORDER,
* CITPRO- GENERATE AND STASH OBJECT TIME TEXT CODE. THIS IS
*-done by processing C I T AND C I T F L G.
* SCONST- STASH OBJECT TIME CONSTANTS.
* SSTRIN- STASH OBJECT TIME STRINGS (FORMATS).
* LSTASH- CLOSE PUNCH FILE BY FORCING OUT LAST CARD.
* THIS IS THE END OF PASS 2. AT THIS POINT POST PASS 2 IS ENTERED.
THE FOLLOWING ROUTINE IS RESPONSIBLE FOR PROCESSING CIT AND C I T F L G. THAT IS, THE OBJECT TIME TEXT CODE IS GENERATED USING THE C I T AND C I T F L G FILES GENERATED WITH THE PASS 1 SUBROUTINE C I T B L D.

NOW C I T P R O IS BROKEN INTO THREE SECTIONS,

GETFLG- THIS SECTION OBTAINS THE NEXT C I T WORD AND ITS ASSOCIATED FLAG. IF NECESSARY, GETFLG READS SYSUTI TO PULL IN THE NEXT C I T AND C I T F L G BLOCKS. WHEN GETFLG RECOGNIZES THE "END-OF- C I T" FLAG (= 7T OCTAL), CONTROL IS SENT, BY G E T F L G, TO THE NEXT ROUTINE IN PASS 2 (THAT IS, SC ONST). REM

FLAG PROCESSORS- THE SEVERAL FLAG PROCESSORS SIMPLY PERFORM THE REQUIRED RELOCATION ON A C I T ENTRY ACCORDING TO THE VALUE OF ITS ASSOCIATED FLAG.

2STASH- THIS ROUTINE IS BUILT INTO THE PROCESSOR FOR FLAG = NFT AND IS RESPONSIBLE FOR STASHING THE FINALIZED OBJECT TIME INSTRUCTION INTO A BINARY CARD IMAGE AND INSERTING THE PROPER RELOCATION SPECIFICATION INTO THIS CARD IMAGE. AFTER STASHING, 2STASH SENDS CONTROL TO GETFLG TO PROCESS THE NEXT C I T ENTRY.
EJECT

* THIS IS THE FIRST OF THE THREE CITPBO SECTIONS. THE NEXT
* CIT WORD AND ITS ASSOCIATED FLAG ARE OBTAINED AND PASSED ON TO
* THE NEXT SECTION OF CITPBO.

SPACE 3
NZI CISW1 WAS CIT2 PUT ON TAPE
TRA GETFG1 NO, BYPASS READ OF SYSUT1
IOS35 EQU *
GETFG5 READ SYSUT1,BINARY,CTFLG2,-1 READ CIT2 FLAG BLOCK
GETFG5 STL IOEX
BRA READ,,+++3
PZE SYSUT1,,BINARY
MZE CTFLG2,,1
END35 REM

IOS36 EQU *
READ SYSUT1,BINARY,CIT2,-1 READ CIT2 BLOCK
STL IOEX
BRA READ,,+++3
PZE SYSUT1,,BINARY
MZE CIT2,,1
END36 REM
GETFG1 AXC 1,1 CIT WORD POINTER

REM
PASS2 EQU GETFG1 DEFINE SYMBOL 'PASS2'
GETFLG EQU GETFG1 DEFINE SYMBOL 'GETFLG'
GETFG2 AXC 1,2 FLAG WORD POINTER
AXT 6,4 FLAG WORD BYTE POINTER
GETLCA LDQ ...,2 SET MQ = CURRENT FLAG WORD
ZAC
LGL 6 SHIFT NEW FLAG INTO ACC
GETLCB STQ ...,2 SAVE REMAINDER OF FLAG WORD
STA GETFGA SAVE THIS FLAG TEMPORARILY
LAS =076 IS THIS FLAG = 76 OR 77 (OCTAL)
TRA SCONST FLAG = 77 ... GO TO SCONST
TRA GETFG4 FLAG = 76 ... GO READ NEXT CIT BLOCK
GETLCC CAL ...,1 SET ACC TO THIS CIT ENTRY
LXA GETFG1,1 SET XRI TO CIT WORD POINTER
TIX GETFG3,4,1 IF END OF FLAG WORD SET-UP FOR NEXT
AXT 6,4 RESET FLAG WORD BYTE POINTER
LXA GETFG2,2 BUMP FLAG WORD POINTER
TXI ++1,2,1 X
SXA GETFG2,2 X
GETFG3 TXI ++1,1,1 BUMP CIT WORD POINTER
SXA GETFG1,1 X
SXA GETFG2,1,4 SAVE FLAG WORD BYTE POINTER
LXA GETFGA,1 SET XRI TO FLAG
EJECT

* THIS IS THE SECOND OF THE THREE C I T P R O SECTIONS. THE
* VARIOUS PROCESSORS FOR THE C I T FLAGS RESIDE HERE. ON ENTRY
* TO THIS SECTION THE CURRENT C I T FLAG RESIDES IN XRI.
* SPACE 3
* TXH CITER,1,MAXFLG ERROR IF FLAG TOO HIGH
* TRA FLGTBL,1 TRANSFER TO APPROPRIATE FLAG PROCESSOR

FLGTBL EQU FLGHI
FLAGH
FLAGG
FLAGD
FLAGQ
FLAGW
FLAGX
FLAGS
FLAGC
FLAGF
FLAGT
FLAGV
FLAGP

FLGTBL MAXFLG EQU #-FLGTBL-1 FOR USE IN TEST ABOVE
SPACE 2
CITER SIR MCR,,SCONST ILLEGAL CIT FLAG ENCOUNTERED

REM FLAG = P
REM ACL LTVPRL TEXT RELOCATION
REM SSM SPECIFY ADDRESS RELOCATION
REM TRA FLAGN GO STASH THIS INSTRUCTION
REM REM FLAG = R
REM FLAGR PAX ,1 TEXT RELOCATION ON EQUIV ENTRY
REM SLW ERASE SAVE CIT WORD
REM CAL EQUIV,1 GET EQUIV ENTRY
REM STA ERASE BUILD WORD FOR TEXT RELOCATION
REM CAL ERASE SET ACC FOR TEXT RELOCATION
REM TRA FLGP GO DO TEXT RELOCATION
REM REM FLAG = V
REM REM FLAVG PAX ,1 COMMON OR VARIABLE RELOCATION
REM SLW ERASE SAVE CIT WORD
REM LDI EQUIV,1 GET EQUIV ENTRY
REM PIA SET EQUIV ENTRY INTO ACC
REM STL OVERFG FLAG FOR POSSIBLE ALTERNATE RELOCATION
REM LNT BCQMN IS COMMON FLAG ON
REM TRA FLGV1
REM SSM =077777 YES, DO COMMON RELLOCATION
REM ADD TOPCOM X

PAGE 303
<table>
<thead>
<tr>
<th>TRA</th>
<th>PAX</th>
<th>LTHRU</th>
<th>DO VARIABLE RELOCATION</th>
<th>*F1117770</th>
<th>PAGE 304</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRA</td>
<td>PAX</td>
<td>LTHRU</td>
<td>FLAG+4</td>
<td>F1117780</td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>PAX</td>
<td>LTHRU</td>
<td>FLAG+4</td>
<td>F1117780</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>PAX</td>
<td>LTHRU</td>
<td>REM</td>
<td>F1117790</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>PAX</td>
<td>LTHRU</td>
<td>REM</td>
<td>F1117800</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>PAX</td>
<td>LTHRU</td>
<td>REM</td>
<td>F1117810</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>PAX</td>
<td>LTHRU</td>
<td>REM</td>
<td>F1117820</td>
<td></td>
</tr>
<tr>
<td>FLAG</td>
<td>PAX</td>
<td>,1</td>
<td>EXTERNAL SYMBOL RELOCATION</td>
<td>F1117830</td>
<td></td>
</tr>
<tr>
<td>SLW</td>
<td>ERASE</td>
<td>SAVE WORD TO BE STASHED</td>
<td>F1117840</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAL</td>
<td>EQUIV,1</td>
<td>GET EQUIV ENTRY</td>
<td>F1117850</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUB</td>
<td>ONE</td>
<td>BUILD WORD TO BE RELOCATED</td>
<td>F1117860</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STA</td>
<td>ERASE</td>
<td>SET AGC</td>
<td>F1117870</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAL</td>
<td>ERASE</td>
<td>SET AGC</td>
<td>F1117870</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>FLAGP+1</td>
<td>GO STASH THIS WORD</td>
<td>F1117870</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>PAX</td>
<td>LTHRU</td>
<td>REM</td>
<td>F1117900</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>PAX</td>
<td>LTHRU</td>
<td>REM</td>
<td>F1117900</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>PAX</td>
<td>LTHRU</td>
<td>REM</td>
<td>F1117900</td>
<td></td>
</tr>
<tr>
<td>FLAG</td>
<td>PAX</td>
<td>,1</td>
<td>STRING RELOCATION</td>
<td>F1117930</td>
<td></td>
</tr>
<tr>
<td>SLW</td>
<td>ERASE</td>
<td>SAVE ARGUMENT</td>
<td>F1117940</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAL</td>
<td>EQUIV,1</td>
<td>GET EQUIV ENTRY</td>
<td>F1117950</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACL</td>
<td>LTPTSC</td>
<td>BUMP BY LENGTH TO STRINGS</td>
<td>F1117960</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>FLAGP+1</td>
<td>GO STASH THIS WORD</td>
<td>F1117970</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>PAX</td>
<td>LTHRU</td>
<td>REM</td>
<td>F1117980</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>PAX</td>
<td>LTHRU</td>
<td>REM</td>
<td>F1117990</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>PAX</td>
<td>LTHRU</td>
<td>REM</td>
<td>F1118000</td>
<td></td>
</tr>
<tr>
<td>FLAG</td>
<td>PAX</td>
<td>,1</td>
<td>CONSTANT RELOCATION</td>
<td>F1118010</td>
<td></td>
</tr>
<tr>
<td>ACL</td>
<td>LTVPTS</td>
<td>CONSTANT RELOCATION</td>
<td>F1118020</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>FLAGP+1</td>
<td>GO STASH</td>
<td>F1118030</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>PAX</td>
<td>LTHRU</td>
<td>REM</td>
<td>F1118040</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>PAX</td>
<td>LTHRU</td>
<td>REM</td>
<td>F1118050</td>
<td></td>
</tr>
<tr>
<td>FLAGS</td>
<td>PAC</td>
<td>,1</td>
<td>LIBRARY SUBROUTINE ENTRY RELOCATION</td>
<td>F1118060</td>
<td></td>
</tr>
<tr>
<td>SLW</td>
<td>ERASE</td>
<td>X</td>
<td>F1118070</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAL</td>
<td>0,1</td>
<td>X</td>
<td>F1118080</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SUB</td>
<td>ONE</td>
<td>X</td>
<td>F1118090</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STA</td>
<td>ERASE</td>
<td>X</td>
<td>F1118100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAL</td>
<td>ERASE</td>
<td>X</td>
<td>F1118110</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>FLAGP+1</td>
<td>X</td>
<td>F1118120</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>PAX</td>
<td>LTHRU</td>
<td>REM</td>
<td>F1118130</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>PAX</td>
<td>LTHRU</td>
<td>REM</td>
<td>F1118140</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>PAX</td>
<td>LTHRU</td>
<td>REM</td>
<td>F1118150</td>
<td></td>
</tr>
<tr>
<td>FLAGX</td>
<td>CAL</td>
<td>TRA</td>
<td>SUBPROGRAM EXIT</td>
<td>F1118160</td>
<td></td>
</tr>
<tr>
<td>ACL</td>
<td>LTVECT</td>
<td>GENERATE TRANSFER TO RESTORE SECTION</td>
<td>F1118170</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>FLAGP+1</td>
<td>OF PROLOGUE AND GO STASH</td>
<td>F1118180</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>PAX</td>
<td>LTHRU</td>
<td>REM</td>
<td>F1118190</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>PAX</td>
<td>LTHRU</td>
<td>REM</td>
<td>F1118200</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>PAX</td>
<td>LTHRU</td>
<td>REM</td>
<td>F1118210</td>
<td></td>
</tr>
<tr>
<td>FLAGW</td>
<td>SLW</td>
<td>ERASE</td>
<td>WORKING STORAGE RELOCATION</td>
<td>F1118220</td>
<td></td>
</tr>
<tr>
<td>PAC</td>
<td>,4</td>
<td>X</td>
<td>F1118230</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TXI</td>
<td>,4</td>
<td>X</td>
<td>F1118240</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SXA</td>
<td>ERASE,4</td>
<td>THIS DECREMENT SET BY SYMPASS</td>
<td>F1118250</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAL</td>
<td>ERASE</td>
<td>X</td>
<td>F1118260</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACL</td>
<td>LTHRU</td>
<td>X</td>
<td>F1118270</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>FLAGP+1</td>
<td>X</td>
<td>F1118280</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>PAX</td>
<td>LTHRU</td>
<td>REM</td>
<td>F1118290</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>PAX</td>
<td>LTHRU</td>
<td>REM</td>
<td>F1118300</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>PAX</td>
<td>LTHRU</td>
<td>REM</td>
<td>F1118310</td>
<td></td>
</tr>
<tr>
<td>FLAGE</td>
<td>PAX</td>
<td>,1</td>
<td>NON-LIBRARY EXTERNAL REFERENCE</td>
<td>F1118320</td>
<td></td>
</tr>
<tr>
<td>SLW</td>
<td>ERASE</td>
<td>X</td>
<td>F1118330</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CAL</td>
<td>EQUIV,1</td>
<td>X</td>
<td>F1118340</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>FLAGS+3</td>
<td>X</td>
<td>F1118350</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>PAX</td>
<td>LTHRU</td>
<td>REM</td>
<td>F1118360</td>
<td></td>
</tr>
</tbody>
</table>
REM FLAG = Q
REM
FLAGQ PAX, 4
TZE GETFLG
SXD FLGN1,4
STL ADDIN
TRA GETFLG
REM
REM FLAG = D
REM
FLAGD AXT 1,1
SXD FLGN1,1
STL ADDIN
TRA FLAGV
REM
REM FLAG = G
REM
FLAGG AXT -1,1
TRA FLAGD+1
REM
REM FLAG = H
REM
FLAGH CAL
SSM CAL
TSX STASH,4
CAL SLWB
TSX STASH,4
CAL STZO
SSM
TSX STASH,4
TRA GETFLG
REM
REM FLAG = N
REM
FLAGN ZET ADDIN
TRA FLGN1-2
STZ OUTFDG
REM
REL 118370
REL 118380
REL 118390
REL 118400
REL 118410
REL 118420
REL 118430
REL 118440
REL 118450
REL 118460
REL 118470
REL 118480
REL 118490
REL 118500
REL 118510
REL 118520
REL 118530
REL 118540
REL 118550
REL 118560
REL 118570
REL 118580
REL 118590
REL 118600
REL 118610
REL 118620
REL 118630
REL 118640
REL 118650
REL 118660
REL 118670
REL 118680
REL 118690
REL 118700
REL 118710
REL 118720
REL 118730
STASH

AXT 20,2
STSH1 AXT 0,4
SLW 9LEFT+24,2
ACL SUM
SLW SUM
TXI ++1,4,2
TPL STSH2
TXI ++1,4,1
LDQ ADREL
ZET OVERFG
LDQ =03
STZ OVERFG
ZAC
LGL 72,4
ORS 8LEFT
LGL 36
ORS BRIGHT
STSH2 SXA STSH1,4
CAL PGMCTR
ACL ONE
SLW PGMCTR
TXI EXSTSH-1,2,1
STX STSH3,2,339
STX0 9LEFT,2
CAL SUM
ACL 9LEFT
SLW SUM
TSX CDLABL,4
IOS37 EQU *
WRITE SYSBIN,BINARY,9LEFT,24,COL73,3 WRITE BIN. CARD IMAGE
STL IOEX
BRA WRITE,**,4
PTW SYSBIN,,BINARY
PZE 9LEFT,24
MZE COL73,3
END37 RM
CAL 9LEFT
ACL =20
STA 9LEFT
AXT 20,4
SXA STSH1-1,4
AXT 0,4
SXA STSH1,4
STZ SUM
STZ 8LEFT
STZ BRIGHT
STSH4 TRA ++2
SXA EXSTSH TRA GETFLG
SPACE 3
<table>
<thead>
<tr>
<th>Instruction</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>EJECT</td>
<td></td>
<td>FT119340</td>
</tr>
<tr>
<td>THIS</td>
<td>IS THE TAIL-END OF THE PASS 2 PROCESSOR FOR FLAG = N</td>
<td>FT119350</td>
</tr>
<tr>
<td>SPACE</td>
<td></td>
<td>FT119360</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>FT119370</td>
</tr>
<tr>
<td>SLW</td>
<td></td>
<td>FT119380</td>
</tr>
<tr>
<td>ERASE</td>
<td></td>
<td>FT119390</td>
</tr>
<tr>
<td>PAX</td>
<td></td>
<td>FT119400</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>FT119410</td>
</tr>
<tr>
<td>FLGN1</td>
<td></td>
<td>FT119420</td>
</tr>
<tr>
<td>TXI</td>
<td></td>
<td>FT119430</td>
</tr>
<tr>
<td>++1,4,++</td>
<td></td>
<td>FT119440</td>
</tr>
<tr>
<td>SXA</td>
<td></td>
<td>FT119450</td>
</tr>
<tr>
<td>ERASE,4</td>
<td></td>
<td>FT119460</td>
</tr>
<tr>
<td>LRS</td>
<td></td>
<td>FT119470</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>FT119480</td>
</tr>
<tr>
<td>CAL</td>
<td></td>
<td>FT119490</td>
</tr>
<tr>
<td>ERASE</td>
<td></td>
<td>FT119500</td>
</tr>
<tr>
<td>LLS</td>
<td></td>
<td>FT119510</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>FT119520</td>
</tr>
<tr>
<td>STZ</td>
<td></td>
<td>FT119530</td>
</tr>
<tr>
<td>ADDIN</td>
<td></td>
<td>FT119440</td>
</tr>
<tr>
<td>NZI</td>
<td></td>
<td>FT119450</td>
</tr>
<tr>
<td>OVERRG</td>
<td></td>
<td>FT119460</td>
</tr>
<tr>
<td>TRA</td>
<td></td>
<td>FT119470</td>
</tr>
<tr>
<td>FLGN2</td>
<td></td>
<td>FT119480</td>
</tr>
<tr>
<td>TXH</td>
<td></td>
<td>FT119490</td>
</tr>
<tr>
<td>++4,4,++</td>
<td></td>
<td>FT119500</td>
</tr>
<tr>
<td>LNT</td>
<td></td>
<td>FT119510</td>
</tr>
<tr>
<td>BCOMN</td>
<td></td>
<td>FT119520</td>
</tr>
<tr>
<td>TRA</td>
<td></td>
<td>FT119530</td>
</tr>
<tr>
<td>FLGN+2</td>
<td></td>
<td>FT119540</td>
</tr>
<tr>
<td>TRA</td>
<td></td>
<td>FT119550</td>
</tr>
<tr>
<td>FLGN+2</td>
<td></td>
<td>FT119560</td>
</tr>
<tr>
<td>TRA</td>
<td></td>
<td>FT119570</td>
</tr>
<tr>
<td>FLGN+3</td>
<td></td>
<td>FT119580</td>
</tr>
<tr>
<td>TRA</td>
<td></td>
<td>FT119590</td>
</tr>
<tr>
<td>FLGN+3</td>
<td></td>
<td>FT119600</td>
</tr>
<tr>
<td>TTL</td>
<td>PASS 2 - SCONST</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>----------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>THE FOLLOWING ROUTINE STASHES THE CONSTANTS REQUIRED BY THE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OBJECT PROGRAM INTO BINARY CARD IMAGES. THE CONSTANTS IN</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TABLE A B ARE PUT OUT IN THE ORDER THAT THEY APPEAR IN THAT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TABLE. AFTER ALL OF THESE CONSTANTS HAVE BEEN STASHED CONTROL</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IS RELINQUISHED TO THE NEXT ROUTINE IN PASS 2 (SSTRIN).</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SPACE</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCONST</td>
<td>NOCONT</td>
</tr>
<tr>
<td></td>
<td>ARE ANY NORMAL CONSTANTS IN USE</td>
</tr>
<tr>
<td></td>
<td>NO, GO TO SSTRIN IMMEDIATELY</td>
</tr>
<tr>
<td></td>
<td>NOCONT,2</td>
</tr>
<tr>
<td></td>
<td>EXSCON-1,2</td>
</tr>
<tr>
<td></td>
<td>1,2</td>
</tr>
<tr>
<td></td>
<td>CONTAB+1,2</td>
</tr>
<tr>
<td></td>
<td>STASH,4</td>
</tr>
<tr>
<td></td>
<td>**1,2,1</td>
</tr>
<tr>
<td></td>
<td><strong>-3,2,</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EXSCON BSS</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GO TO SSTRIN AT END OF SCONST</td>
</tr>
</tbody>
</table>
**TTL**  PASS 2 - SSTRIN

* THE FOLLOWING ROUTINE STASHES STRINGS (FORMATS) INTO BINARY CARD

* IMAGES. THESE STRINGS ACTUALLY RESIDED IN POOL BUT WERE

* CHAINED TOGETHER IN S Y P A S S (IN POST PASS 1). AFTER ALL THE

* STRINGS ARE STASHED CONTROL IS GIVEN TO THE NEXT ROUTINE IN PASS

* 2 (L STASH).

<table>
<thead>
<tr>
<th>SPAC</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSTRIN</td>
<td>N2T</td>
</tr>
<tr>
<td>LSTRIN</td>
<td>ARE THERE ANY STRINGS TO STASH</td>
</tr>
<tr>
<td>TRA</td>
<td>LSTASH</td>
</tr>
<tr>
<td>LXA</td>
<td>STSH2,1</td>
</tr>
<tr>
<td>SSTRICAL</td>
<td>POOL,1</td>
</tr>
<tr>
<td>CAL</td>
<td>GET POOL ENTRY FOR THIS STRING</td>
</tr>
<tr>
<td>SLW</td>
<td>ERASE</td>
</tr>
<tr>
<td>PAX</td>
<td>X2</td>
</tr>
<tr>
<td>SXD</td>
<td>**1,2</td>
</tr>
<tr>
<td>TXI</td>
<td>**1,1...</td>
</tr>
<tr>
<td>CAL</td>
<td>POOL,1</td>
</tr>
<tr>
<td>TXS</td>
<td>STASH,4</td>
</tr>
<tr>
<td>TXI</td>
<td>**1,1,-1</td>
</tr>
<tr>
<td>TIX</td>
<td>X-3,2,1</td>
</tr>
<tr>
<td>LXO</td>
<td>ERASE,1</td>
</tr>
<tr>
<td>TXH</td>
<td>SSTR1,1,0</td>
</tr>
<tr>
<td>EXSTR</td>
<td>BSS 0</td>
</tr>
</tbody>
</table>

<p>| FT119710 | FT119720 | FT119730 | FT119740 | FT119750 | FT119760 | FT119770 | FT119780 | FT119790 | FT119800 | FT119810 | FT119820 | FT119830 | FT119840 | FT119850 | FT119860 | FT119870 | FT119880 | FT119890 | FT119900 | FT119910 | FT119920 |</p>
<table>
<thead>
<tr>
<th>Command</th>
<th>Instruction</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTL</td>
<td>PASS 2 - LSTASH</td>
<td>FT119930</td>
</tr>
<tr>
<td></td>
<td>THE FOLLOWING ROUTINE IS USED TO FORCE OUT THE LAST (PERHAPS PARTIALLY FILLED) BINARY CARD IMAGE ONTO SYSBIN. THIS IS DONE USING THE STASH ROUTINE THAT RESIDES IN SECTION 3 OF C I T P R O. UPON COMPLETION OF LSTASH, CONTROL IS PASSED ON TO PASS 2.</td>
<td></td>
</tr>
<tr>
<td>LM</td>
<td>SPACE 3</td>
<td>FT119990</td>
</tr>
<tr>
<td>LSTASH</td>
<td>LXA STSH1-1,2</td>
<td>PICK UP WORD POINTER</td>
</tr>
<tr>
<td></td>
<td>TXH PPASS2,2,19</td>
<td>IF IMAGE FULL GO TO PASS 2</td>
</tr>
<tr>
<td></td>
<td>AXT EXLSTH,4</td>
<td>SET RETURN IN STASH</td>
</tr>
<tr>
<td></td>
<td>SXA EXSTSH,4</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>STZ 9LEFT+24,2</td>
<td>ZERO UNUSED COLUMNS OF LAST CARD</td>
</tr>
<tr>
<td></td>
<td>TIX *-1,2,1</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>LAC STSH1-1,2</td>
<td>GET WORD COUNT FOR LAST CARD</td>
</tr>
<tr>
<td></td>
<td>TXI STSH3,2,340</td>
<td>SET-UP DECR. OF 9 LEFT</td>
</tr>
<tr>
<td>REM</td>
<td>EXLSTH AXT GETFLG,4</td>
<td>RESET STSH4 IN STASH ROUTINE</td>
</tr>
<tr>
<td></td>
<td>SXA EXSTSH,4</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>TRA PPASS2</td>
<td>GO TO POST PASS 2</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>GETFG4</td>
<td>GET FG +1, 1</td>
<td>Reset CIT Word Pointer</td>
</tr>
<tr>
<td>SXA</td>
<td>GETFG2, 1</td>
<td>Also reset flag word pointer</td>
</tr>
<tr>
<td>AXT</td>
<td>6, 1</td>
<td>Reset flag word byte pointer</td>
</tr>
<tr>
<td>SXA</td>
<td>GETFG2 +1, 1</td>
<td>X</td>
</tr>
<tr>
<td>AXT</td>
<td>GETLG -1, 2</td>
<td>End of CIT1 or CIT2</td>
</tr>
<tr>
<td>TRA</td>
<td>GETLG -5</td>
<td>End of CIT2</td>
</tr>
<tr>
<td>AXT</td>
<td>CTFGL2 -1, 1</td>
<td>End of CIT1, re-initialize for CIT2</td>
</tr>
<tr>
<td>SXA</td>
<td>GETLCB, 1</td>
<td></td>
</tr>
<tr>
<td>AXT</td>
<td>CIT2 -1, 1</td>
<td></td>
</tr>
<tr>
<td>SXA</td>
<td>GETLCB, 1</td>
<td></td>
</tr>
<tr>
<td>STL</td>
<td>GTSWCH</td>
<td>Turn on CIT2 switch</td>
</tr>
<tr>
<td>TRA</td>
<td>GETLG -5</td>
<td>Go see if CIT2 is on tape</td>
</tr>
<tr>
<td>SPACE</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

* This is the tail-end of the TVCITL routine

** TVEL2 **

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PXA</td>
<td>1</td>
</tr>
<tr>
<td>STA</td>
<td>SLNTAB -2, 4</td>
</tr>
<tr>
<td>TRA</td>
<td>TVEL1 +2</td>
</tr>
</tbody>
</table>

* User-defined notes and explanations. *
<table>
<thead>
<tr>
<th>TTL</th>
<th>POST PASS 2 - DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>REM</td>
<td>FASTRAN COMPILER</td>
</tr>
<tr>
<td>*</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>INDIANA UNIVERSITY</td>
</tr>
<tr>
<td>*</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>BLOOMINGTON, INDIANA</td>
</tr>
<tr>
<td>*</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>POST PASS 2</td>
</tr>
<tr>
<td>*</td>
<td></td>
</tr>
<tr>
<td>ACTION</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>EJECT</td>
<td>On entry to post pass 2, the object program has been completely output onto relocatable binary cards, the result of an error-free compilation. Post pass 2 is concerned with the following two items:</td>
</tr>
<tr>
<td>SPACE 2</td>
<td>1. Mapping of the object program to indicate the exact core storage layout of the object program (relative to zero).</td>
</tr>
<tr>
<td>REM</td>
<td>2. Determining if the next program on the input tape is another Fortran compilation. If the next program does not involve Fortran then post pass 2 sends control to the initialization section of the compiler to begin this next compilation. If the next program does not involve Fortran then control is relinquished to the system monitor.</td>
</tr>
</tbody>
</table>
THE FOLLOWING ROUTINE IS USED TO OUTPUT ON THE SYSTEM OUTPUT TAPE.

THE STORAGE MAP ASSOCIATED WITH THIS COMPILATION. THIS MAP CONTAINS THE FOLLOWING ITEMS IN THE INDICATED ORDER (ITEMS WHICH DO NOT APPLY TO A PARTICULAR PROGRAM ARE NOT MAPPED).

SPACE 2

STORAGE NOT USED BY PROGRAM
INDEX REGISTERS USED BY THE OBJECT PROGRAM
ORIGIN AND LENGTH OF FOLLOWING,
TRANSFER VECTOR
TEXT
CONSTANTS
STRINGS (FORMATS)
WORKING STORAGE
LOW-CORE VARIABLES
COMMON STORAGE
TRANSFER VECTOR
OCTAL LOCATIONS OF NUMBERED EXECUTABLE STATEMENTS
OCTAL LOCATIONS OF STRINGS (FORMATS)
OCTAL LOCATIONS OF NON-DIMENSIONED NON-COMMON VARIABLES
OCTAL LOCATIONS OF DIMENSIONED NON-COMMON VARIABLES
OCTAL LOCATIONS OF NON-DIMENSIONED COMMON VARIABLES
OCTAL LOCATIONS OF DIMENSIONED COMMON VARIABLES

SPACE 3

PPASS2
CAL LABEL GET LABEL OF PROGRAM BEING COMPILED
IZE MAP1 TRANSFER IF MAIN PROGRAM
SLW LINI+9 STORE SUBPROGRAM NAME IN HEADING
CAL BLANKS X
SLW LINI+10 X
MAP1 TSX LIST,4 GO LIST MAP HEADING
PTW LINI+11 PAGE EJECT
CAL GAR1 RESTORE LIN1 TO 'MAIN PROGRAM'
SLW LINI+9 X
CAL GAR1+1 X
SLW LINI+10 X
CAL PGMBRK GET PROGRAM BREAK AND COMMON BREAK
TSX OCTBCD,4 FOR 'STORAGE NOT USED ...' MESSAGE
ARS 6 RIGHT-ADJUST PROGRAM BREAK
ORA GAR2 LEADING BLANK
SLW LIN2+5 INSERT IN MESSAGE
CAL CMNBRK GET COMMON BREAK
TSX OCTBCD,4 CONVERT TO OCTAL
SLW LIN2+7 INSERT IN MESSAGE
TSX LIST,4 PUT OCTAL PROGRAM LIMITS
PON LIN2+10 DOUBLE SPACED
CAL PGMBRK GET PROGRAM BREAK FOR DECIMAL CONVER.
TSX BINBCD,4 DO DECIMAL CONVERSION
SLW LIN3+5 INSERT IN DECIMAL MESSAGE
CAL CMNBRK DO SAME FOR COMMON BREAK
TSX BINBCD,4 X
ALS 6 LEFT-ADJUST
ORA GAR3 TRAILING BLANK
SLW LIN3+7 INSERT IN DECIMAL MESSAGE
TSX LIST,4 PUT OUT DECIMAL MESSAGE
PZE LIN3+10 SINGLE SPACED
<table>
<thead>
<tr>
<th>EJECT</th>
<th>REM</th>
<th>ANA</th>
<th>ALS</th>
<th>PDX</th>
<th>CAL</th>
<th>SLW</th>
<th>TSX</th>
<th>PON</th>
</tr>
</thead>
<tbody>
<tr>
<td>... TO PUT OUT INDEX REGISTER USAGE MESSAGE</td>
<td>...</td>
<td>TG7BIT</td>
<td>3</td>
<td>1</td>
<td>GARB4,1</td>
<td>LIN4+7</td>
<td>LIST,4</td>
<td>LIN4,,8</td>
</tr>
<tr>
<td>FT121300</td>
<td>FT121310</td>
<td>FT121330</td>
<td>FT121340</td>
<td>FT121350</td>
<td>FT121360</td>
<td>FT121370</td>
<td>FT121380</td>
<td>FT121390</td>
</tr>
</tbody>
</table>

REM  XRFLAG CONTAINS 'OR'-ED XR'S USED

CAL  X R = XR'S USED

ALS  MOVE TO DECREMENT

PDX  XR1 = XR'S USED

CAL  GET APPROPRIATE WORD FOR MESSAGE

SLW  INSERT IN MESSAGE

TSX  PUT OUT THIS MESSAGE

PON  DOUBLE SPACED
EJECT
REM ... TO PUT OUT TABLE OF LIMITS OF VARIOUS SECTIONS OF  
REM THE OBJECT PROGRAM
REM

TSX LIST,4                  PUT OUT 'ORIGIN' LENGTH' HEADING       FT121430
PON LIN5,13                 DOUBLE SPACED                       FT121430
TSX LIST,4                  PUT OUT 'OCTAL -- DECIMAL' MESSAGE    FT121430
PZE LIN6,14                 SINGLE SPACED                       FT121430
AXT 8,1                     PREPARE FIRST 8 TABLE ENTRIES       FT121430
STZ MAPORG                  X                                    FT121430

MAP2
CAL LTVECT+8,1              GET LENGTH OF NEXT ENTRY           FT121430
CAL IF MAP3 IF ZERO GO GET NEXT ENTRY    FT121430
TSX BINBCD,4                CONVERT LENGTH TO DECIMAL          FT121430
ALS 6                       RIGHT-ADJUST                      FT121430
ORA GAR3                    TRAILING BLANK                      FT121430
SLW LIN7+13                 STORE THIS IN MESSAGE               FT121430
CAL LTVECT+8,1              NOW CONVERT LENGTH TO OCTAL           FT121430
TSX OCTBCD,4                X                                    FT121430
SLW LIN7+11                 INSERT IN MESSAGE                    FT121430
CAL MAPORG                  CONVERT ORIGIN OF THIS SECTION TO      FT121430
TSX BINBCD,4                DECIMAL                             FT121430
ALS 6                       LEFT-ADJUST                         FT121430
ORA GAR3                    TRAILING BLANK                      FT121430
SLW LIN7+8                  INSERT IN MESSAGE                    FT121430
CAL MAPORG                  NOW CONVERT ORIGIN TO OCTAL            FT121430
TSX OCTBCD,4                X                                    FT121430
SLW LIN7+6                  INSERT IN MESSAGE                    FT121430
AXT 4,2                     MOVE SECTION NAME INTO PLACE         FT121430
CAL* GAR85,1                USED INDIRECT TABLE ADDRESSING        FT121430
SLW LIN7+4,2                MOVE INTO OUTPUT AREA                  FT121430
TSX *2,1                    MOVE TOTAL OF FOUR WORDS              FT121430
TSX LIST,4                  LIST THIS TABLE ENTRY                   FT121430
PON LIN7,14                 DOUBLE SPACED                       FT121430
CAL MAPORG                  UPDATE MAPORG                       FT121430
ACL LTVECT+8,1              X                                    FT121430
SLW MAPORG                  X                                    FT121430

MAP3
TIX MAP2,1,1                GO BACK FOR NEXT TABLE ENTRY            FT121430
CAL CMNBRK                  LAST TABLE ENTRY IS 'COMMON STORAGE'    FT121430
LAS TOPCOM                  SKIP IF NO COMMON                      FT121430
TRA **2                     SOMETHING'S MESSED UP                  FT121430
TRA MAP28                   SKIP MAP OF COMMON                    FT121430
ADD =1                      X                                    FT121430
SLW ERASE                   X                                    FT121430
TSX BINBCD,4                CONVERT ORIGIN TO DECIMAL              FT121430
ALS 6                       LEFT-ADJUST                         FT121430
ORA GAR3                    TRAILING BLANK                      FT121430
SLW LIN8+8                  INSERT IN MESSAGE                    FT121430
CAL ERASE                   NOW CONVERT ORG COMMON TO OCTAL        FT121430
TSX OCTBCD,4                X                                    FT121430
SLW LIN8+6                  INSERT IN MESSAGE                    FT121430
CAL TOPCOM                  CALCULATE LENGTH OF COMMON            FT121430
SUB CMNBRK                  LENGTH = TOPCOM - CMNBRK               FT121430
SLW TEMP                    SAVE THIS LENGTH FOR CONVER. TO OCTAL    FT121430
TSX BINBCD,4                CONVERT LENGTH TO DECIMAL              FT121430
ALS 6                       LEFT-ADJUST                         FT121430
ORA GAR3                    TRAILING BLANK                      FT121430
SLW LIN8+13                 INSERT IN MESSAGE                    FT121430
CAL TEMP                    CONVERT COMMON LENGTH TO OCTAL          FT121430
TSX OCTBCD,4                X                                    FT121430
SLW LIN8+11                 INSERT IN MESSAGE                    FT121430
EJECT

REM ... TO SET-UP 'POOL' AREA WITH NUMBERED STATEMENTS,

REM FORMATS, AND ALL COMBINATIONS OF DIMENSIONED AND

REM AND NON-DIMENSIONED AND COMMON AND NON-COMMON

REM VARIABLES

REM

MAP7 AXT LSYMB T,1 INITIALIZE FOR SYMTAB SEARCH
AXT 1,2 X

ZET SYMTAB T,1 IS NEXT SYMTAB ENTRY ZERO
TRA MAP8 1 NO, GO SEE WHAT KIND OF SYMBOL IT IS

TRA MAP8 1 NO, GO SEE WHAT KIND OF SYMBOL IT IS

TIX *-2,1,1 YES, GO LOOK AT NEXT SYMBOL
TRA MAP20 +1 GO PUT OUT LAST PORTION OF MAP

MAP8 LDI EQUV T,1 GET EQUV ENTRY FOR THIS SYMBOL

LFT 400000 WAS THIS ENTRY IN ERROR

TRA MAP7+4 YES, GO GET NEXT ENTRY

IIL 700000 NO, INVERT SYMTAB MODE BITS

LNT /MLABL*700000 IS THIS A STATEMENT LABEL

TRA MAP10 NO, GO SEE IF STRING

CAL EQUV T,1 YES, CHAIN TO LAST LABEL

STA POOL T,2 X

PXD 1,1 POOL ENTRY = DEFN.++,SYMTAB POINTER

STD POOL T,2 POOL ENTRY = DEFN.++,SYMTAB POINTER

STZ POOL-1,2 SET NEXT POOL ENTRY TO ZERO

NZT GAR819 IS THIS FIRST LABEL SO FAR

TRA MAP9 4 YES, GO INITIALIZE CHAINING

LXD GAR819 4 NO, CHAIN THIS LABEL TO LAST ONE

PXA ,2 X

STA POOL-1,4 X

SXD GAR819 2 X

SXD GAR819 2 BUMP POOL POINTER FOR NEXT ENTRY

MAP9 SXA GAR819 2 FIRST LABEL ENCOUNTERED

TXI MAP7+4,2,1 BUMP POOL POINTER FOR NEXT ENTRY

MAP9 SXA GAR819 2 FIRST LABEL ENCOUNTERED

SXD GAR819 2 SET-UP FIRST CHAIN LINK

TXI MAP7+4,2,2 BUMP POOL POINTER FOR NEXT ENTRY

MAP10 LNT /MSTRG*700000 IS THIS A STRING

TRA MAP12 NO, GO SEE IF VARIABLE

CAL EQUV T,1 YES, GET EQUV ENTRY

STA POOL T,2 X

PXD 1,1 POOL ENTRY = DEFN.++,SYMTAB POINTER

STD POOL T,2 POOL ENTRY = DEFN.++,SYMTAB POINTER

STZ POOL-1,2 SET NEXT POOL ENTRY TO ZERO

NZT GAR816 IS THIS FIRST STRING SO FAR

TRA MAP11 4 YES, GO INITIALIZE CHAINING

LXD GAR816 4 NO, GO CHAIN THIS STRING TO LAST ONE

PXA ,2 X

STA POOL-1,4 X

SXD GAR816 2 X

SXD GAR816 2 X

MAP11 SXA GAR816 2 FIRST STRING ENCOUNTERED

TXI MAP7+4,2,2 BUMP POOL POINTER FOR NEXT ENTRY

MAP11 SXA GAR816 2 FIRST STRING ENCOUNTERED

SXD GAR816 2 SET-UP FIRST CHAIN LINK

TXI MAP7+4,2,2 BUMP POOL POINTER FOR NEXT ENTRY

MAP12 LFT BARGT IS THIS A SUBROUTINE PARAMETER

TRA MAP7+4 YES, DO NOT PROCESS THIS ENTRY

LNT /MINIT*700000 NO, IS THIS AN INTEGER VARIABLE

TRA MAP14 NO, GO SEE IF REAL VARIABLE

LNT BCOMM NO, IS THIS VARIABLE IN COMMON

TRA MAP15 NO, GO SEE IF IT'S AN ARRAY

LNT BARRY YES, IT IS AN ARRAY

TRA MAP19 NO, MUST BE NON-DIMENSIONED, COMMON

CAL EQUV T,1 CHAIN THIS TO LAST DIMEN.++, COMMON VAR

STA POOL T,2 X
<table>
<thead>
<tr>
<th>PXD</th>
<th>POOL,2</th>
<th>POOL ENTRY = DEFIN.,, SYMTAB POINTER</th>
<th>FT123160</th>
</tr>
</thead>
<tbody>
<tr>
<td>STD</td>
<td>POOL-1,2</td>
<td>SET NEXT POOL ENTRY TO ZERO</td>
<td>FT123170</td>
</tr>
<tr>
<td>STZ</td>
<td>POOL-1,2</td>
<td>SET NEXT POOL ENTRY TO ZERO</td>
<td>FT123180</td>
</tr>
<tr>
<td>NZT</td>
<td>GARB20</td>
<td>IS THIS FIRST DIMEN., COMMON SO FAR</td>
<td>FT123190</td>
</tr>
<tr>
<td>TRA</td>
<td>MAP13</td>
<td>YES, GO INITIALIZE CHAINING</td>
<td>FT123200</td>
</tr>
<tr>
<td>LXD</td>
<td>GARB20,4</td>
<td>NO, CHAIN THIS VARIABLE TO LAST ONE</td>
<td>FT123210</td>
</tr>
<tr>
<td>PXA</td>
<td>,2</td>
<td>X</td>
<td>FT123220</td>
</tr>
<tr>
<td>STA</td>
<td>POOL-1,4</td>
<td>X</td>
<td>FT123230</td>
</tr>
<tr>
<td>SXD</td>
<td>GARB20,2</td>
<td>X</td>
<td>FT123240</td>
</tr>
<tr>
<td>TXI</td>
<td>MAP7+4,2</td>
<td>BUMP POOL POINTER FOR NEXT ENTRY</td>
<td>FT123250</td>
</tr>
<tr>
<td>MAP13</td>
<td>SXA</td>
<td>GARB20,2</td>
<td>FT123260</td>
</tr>
<tr>
<td>SXD</td>
<td>GARB20,2</td>
<td>FIRST DIMEN., COMMON ENCLOSED</td>
<td>FT123270</td>
</tr>
<tr>
<td>TXI</td>
<td>MAP7+4,2</td>
<td>SET-UP FIRST CHAIN LINK</td>
<td>FT123280</td>
</tr>
<tr>
<td>MAP14</td>
<td>LNT</td>
<td>GARB20,2</td>
<td>FT123290</td>
</tr>
<tr>
<td>TRA</td>
<td>MAP7+4</td>
<td>BUMP POOL POINTER FOR NEXT ENTRY</td>
<td>FT123300</td>
</tr>
<tr>
<td>TRA</td>
<td>MAP12+4</td>
<td>NO, DO NOT PROCESS THIS SYMTAB ENTRY</td>
<td>FT123310</td>
</tr>
<tr>
<td>MAP15</td>
<td>LNT</td>
<td>YES, GO TO VARIABLE PROCESSOR</td>
<td>FT123320</td>
</tr>
<tr>
<td>TRA</td>
<td>MAP17</td>
<td>IS THIS AN ARRAY</td>
<td>FT123330</td>
</tr>
<tr>
<td>CAL</td>
<td>EQUIV,1</td>
<td>NO, MUST BE NON-DIMEN., NON-COMMON</td>
<td>FT123340</td>
</tr>
<tr>
<td>STA</td>
<td>POOL,2</td>
<td>CHAIN TO LAST DIMEN., NON-COMMON VAR.</td>
<td>FT123350</td>
</tr>
<tr>
<td>PXD</td>
<td>,1</td>
<td>X</td>
<td>FT123360</td>
</tr>
<tr>
<td>STD</td>
<td>POOL,2</td>
<td>POOL ENTRY = DEFIN.,, SYMTAB POINTER</td>
<td>FT123370</td>
</tr>
<tr>
<td>STZ</td>
<td>POOL-1,2</td>
<td>SET NEXT POOL ENTRY TO ZERO</td>
<td>FT123380</td>
</tr>
<tr>
<td>NZT</td>
<td>GARB18</td>
<td>SET NEXT POOL ENTRY TO ZERO</td>
<td>FT123390</td>
</tr>
<tr>
<td>TRA</td>
<td>MAP16</td>
<td>IS THIS FIRST DIMEN., NON-COMMON VAR.</td>
<td>FT123400</td>
</tr>
<tr>
<td>LXD</td>
<td>GARB18,4</td>
<td>YES, GO INITIALIZE CHAINING</td>
<td>FT123410</td>
</tr>
<tr>
<td>PXA</td>
<td>,2</td>
<td>X</td>
<td>FT123420</td>
</tr>
<tr>
<td>STA</td>
<td>POOL-1,4</td>
<td>X</td>
<td>FT123430</td>
</tr>
<tr>
<td>SXD</td>
<td>GARB18,2</td>
<td>X</td>
<td>FT123440</td>
</tr>
<tr>
<td>TXI</td>
<td>MAP7+4,2</td>
<td>BUMP POOL POINTER FOR NEXT ENTRY</td>
<td>FT123450</td>
</tr>
<tr>
<td>MAP16</td>
<td>SXA</td>
<td>GARB18,2</td>
<td>FT123460</td>
</tr>
<tr>
<td>SXD</td>
<td>GARB18,2</td>
<td>FIRST DIMEN., NON-COMMON ENCLOSED</td>
<td>FT123470</td>
</tr>
<tr>
<td>TXI</td>
<td>MAP7+4,2</td>
<td>SET-UP FIRST CHAIN LINK</td>
<td>FT123480</td>
</tr>
<tr>
<td>MAP17</td>
<td>CAL</td>
<td>GARB18,2</td>
<td>FT123490</td>
</tr>
<tr>
<td>STA</td>
<td>POOL,2</td>
<td>GARB18,2</td>
<td>FT123500</td>
</tr>
<tr>
<td>PXD</td>
<td>,1</td>
<td>X</td>
<td>FT123510</td>
</tr>
<tr>
<td>STD</td>
<td>POOL,2</td>
<td>POOL ENTRY = DEFIN.,, SYMTAB POINTER</td>
<td>FT123520</td>
</tr>
<tr>
<td>STZ</td>
<td>POOL-1,2</td>
<td>SET NEXT POOL ENTRY TO ZERO</td>
<td>FT123530</td>
</tr>
<tr>
<td>NZT</td>
<td>GARB17</td>
<td>IS THIS FIRST NON-DIMEN., NON-COMMON</td>
<td>FT123540</td>
</tr>
<tr>
<td>TRA</td>
<td>MAP18</td>
<td>YES, GO INITIALIZE CHAINING</td>
<td>FT123550</td>
</tr>
<tr>
<td>LXD</td>
<td>GARB17,4</td>
<td>NO, CHAIN THIS VARIABLE TO LAST ONE</td>
<td>FT123560</td>
</tr>
<tr>
<td>PXA</td>
<td>,2</td>
<td>X</td>
<td>FT123570</td>
</tr>
<tr>
<td>STA</td>
<td>POOL-1,4</td>
<td>X</td>
<td>FT123580</td>
</tr>
<tr>
<td>SXD</td>
<td>GARB17,2</td>
<td>X</td>
<td>FT123590</td>
</tr>
<tr>
<td>TXI</td>
<td>MAP7+4,2</td>
<td>BUMP POOL POINTER FOR NEXT ENTRY</td>
<td>FT123600</td>
</tr>
<tr>
<td>MAP18</td>
<td>SXA</td>
<td>GARB17,2</td>
<td>FT123610</td>
</tr>
<tr>
<td>SXD</td>
<td>GARB17,2</td>
<td>FIRST NON-DIMEN., NON-COMMON HIT</td>
<td>FT123620</td>
</tr>
<tr>
<td>TXI</td>
<td>MAP7+4,2</td>
<td>SET-UP FIRST CHAIN LINK</td>
<td>FT123630</td>
</tr>
<tr>
<td>MAP19</td>
<td>CAL</td>
<td>GARB17,2</td>
<td>FT123640</td>
</tr>
<tr>
<td>STA</td>
<td>POOL,2</td>
<td>CHAIN THIS TO LAST NON-DIMEN., COMMON</td>
<td>FT123650</td>
</tr>
<tr>
<td>PXD</td>
<td>,1</td>
<td>X</td>
<td>FT123660</td>
</tr>
<tr>
<td>STD</td>
<td>POOL,2</td>
<td>POOL ENTRY = DEFIN.,, SYMTAB POINTER</td>
<td>FT123670</td>
</tr>
<tr>
<td>STZ</td>
<td>POOL-1,2</td>
<td>SET NEXT POOL ENTRY TO ZERO</td>
<td>FT123680</td>
</tr>
<tr>
<td>NZT</td>
<td>GARB19</td>
<td>IS THIS FIRST NON-DIMEN., COMMON</td>
<td>FT123690</td>
</tr>
<tr>
<td>TRA</td>
<td>MAP20</td>
<td>YES, GO INITIALIZE CHAINING</td>
<td>FT123700</td>
</tr>
<tr>
<td>LXD</td>
<td>GARB19,4</td>
<td>NO, CHAIN THIS VARIABLE TO LAST ONE</td>
<td>FT123710</td>
</tr>
<tr>
<td>PXA</td>
<td>,2</td>
<td>X</td>
<td>FT123720</td>
</tr>
<tr>
<td>STA</td>
<td>POOL-1,4</td>
<td>X</td>
<td>FT123730</td>
</tr>
<tr>
<td>SXD</td>
<td>GARB19,2</td>
<td>X</td>
<td>FT123740</td>
</tr>
<tr>
<td>TXI</td>
<td>MAP7+4,2</td>
<td>BUMP POOL POINTER FOR NEXT ENTRY</td>
<td>FT123750</td>
</tr>
<tr>
<td>MAP20</td>
<td>SXA</td>
<td>GARBI9,2</td>
<td>FIRST NON-DIMEN., COMMON ENCOUNTERED</td>
</tr>
<tr>
<td>-------</td>
<td>-----</td>
<td>----------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>SXD</td>
<td>GARBI9,2</td>
<td>SET-UP FIRST CHAIN LINK</td>
<td>FT123770</td>
</tr>
<tr>
<td>TXI</td>
<td>MAP7+4,2,2</td>
<td>BUMP POOL POINTER FOR NEXT ENTRY</td>
<td>FT123780</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FT123790</td>
</tr>
<tr>
<td>MAP29</td>
<td>PXA</td>
<td>1,2</td>
<td>SET NXTLOC = MAX(NXTLOC, NO. POOL CELLS USED BY MAP)</td>
</tr>
<tr>
<td></td>
<td>LAS</td>
<td>NXTLOC</td>
<td>FT123810</td>
</tr>
<tr>
<td></td>
<td>SLW</td>
<td>NXTLOC</td>
<td>FT123820</td>
</tr>
<tr>
<td></td>
<td>TRA</td>
<td>*+2</td>
<td>FT123830</td>
</tr>
<tr>
<td></td>
<td>NOP</td>
<td></td>
<td>FT123840</td>
</tr>
</tbody>
</table>
EJECT
REM ... TO OUTPUT LOCATIONS OF NUMBERED STATEMENTS, FORMATS, AND VARIABLES

MAP21
STZ ERASE
STZ ERASE
AXT 0+1
SAX MAP27+2,1
NZT GARBI5+1
TRA MAP27+2
CAL GARBI5+1
PAX .2
CAL POOL,2
PDX ,4
CAL SYMTAB,4
TSX RTJUST+4

MAP22
AXC **,4
SLW LIN10-1,4
CAL POOL,2
LXA MAP27+2,4
TXL **3,4,3
ANA =077777

MAP23
SSM
ADD* GARBI3,4
TSX OCTBCD+4
ARS 6
ORA GARBI2
XEC MAP22
SLW LIN10+4
LXA MAP22+4
TXH MAP24-1,4,12
TXI **4,3
CAL POOL-1,2

MAP24
PAX ,2
TXL MAP26+2,0
TRA MAP22-4
TXI **3,4,-2
TXI **2,4+1
LXA MAP22+4

SAX MAP24+5,4
NZT ERASE
TRA MAP25
TSX LIST,4
PZE LIN10,...
ZET ERASE1
TRA MAP27
AXT 2+4
SAX MAP22+4
TRA MAP23

MAP25
SXL ERASE
TSX LIST,4
PON BLANKS,1
LXA MAP27+2,4
CAL GARBI4+1
SLW **2
TSX LIST,4
PON **5,4
LXD MAP24+5,4
SXD **5,4

PAGE 323
<table>
<thead>
<tr>
<th>LXA</th>
<th>MAP27+2,4</th>
<th>GET CURRENT SECTION POINTER</th>
<th>FT124450</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAL</td>
<td>GAR822,4</td>
<td>GET LOCATION OF SECONDARY HEADING</td>
<td>FT124460</td>
</tr>
<tr>
<td>STA</td>
<td>**+2</td>
<td>SAVE MESSAGE LOCATION IN LIST CALL</td>
<td>FT124470</td>
</tr>
<tr>
<td>TXS</td>
<td>LIST,4</td>
<td>OUTPUT SECONDARY HEADING</td>
<td>FT124480</td>
</tr>
<tr>
<td>PON</td>
<td>***</td>
<td>X</td>
<td>FT124490</td>
</tr>
<tr>
<td>TRA</td>
<td>MAP24+4</td>
<td>GO OUTPUT LINE THIS SECTION</td>
<td>FT124500</td>
</tr>
<tr>
<td>MAP26</td>
<td>STL</td>
<td>ERASE1</td>
<td>SET 'EXIT THIS SECTION' SWITCH</td>
</tr>
<tr>
<td>LXA</td>
<td>MAP22,4</td>
<td>GET CURRENT LINE POSITION</td>
<td>FT124520</td>
</tr>
<tr>
<td>MAP27</td>
<td>TXH</td>
<td>MAP24-2,4,2</td>
<td>TRANSFER IF LAST LINE NOT FULL</td>
</tr>
<tr>
<td></td>
<td>STZ</td>
<td>ERASE</td>
<td>RESET 'NO LINE THIS SECTION YET'</td>
</tr>
<tr>
<td></td>
<td>AXT</td>
<td>***</td>
<td>XR1 = CURRENT SECTION NUMBER</td>
</tr>
<tr>
<td>TXH</td>
<td>MAP31,1,4</td>
<td>GO TO M A P 3 1 IF LAST (THE SIXTH) SECTION IS OUT. OTHERWISE GO GET</td>
<td>FT124570</td>
</tr>
<tr>
<td>TXI</td>
<td>**+1,1,1</td>
<td>NEXT SECTION</td>
<td>FT124580</td>
</tr>
<tr>
<td>SXA</td>
<td>**-3,1</td>
<td>RESTORE MAP22</td>
<td>FT124590</td>
</tr>
<tr>
<td>AXI</td>
<td>2,2</td>
<td>X</td>
<td>FT124600</td>
</tr>
<tr>
<td>SXA</td>
<td>MAP22,2</td>
<td>GO GET NEXT SECTION</td>
<td>FT124610</td>
</tr>
<tr>
<td>TRA</td>
<td>MAP21+4</td>
<td>GO GET NEXT SECTION</td>
<td>FT124620</td>
</tr>
<tr>
<td>EJECT</td>
<td>FT124630</td>
<td>PAGE 325</td>
<td></td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>----------</td>
<td>----------</td>
<td></td>
</tr>
<tr>
<td>REM ... TO OUTPUT SYMBOL TABLE MESSAGE IF NECESSARY</td>
<td>FT124640</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>FT124650</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAP31</td>
<td>FT124660</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NZT STOUT</td>
<td>FT124670</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRA FINISH</td>
<td>FT124680</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TSX LIST,4</td>
<td>FT124690</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PON BLANKS,1</td>
<td>FT124700</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PON LIN19,,16</td>
<td>FT124710</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRA FINISH</td>
<td>FT124720</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EJECT</td>
<td>REM</td>
<td>GARB1 BCI</td>
<td>GARB2 BCI</td>
</tr>
<tr>
<td>-------</td>
<td>-----</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2, MAIN PROGRAM</td>
<td>1, 00000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: The image contains a program map or similar listing with various addresses and labels, indicating a structured program layout.
BCI 7, FASTRAN STORAGE MAP FOR MAIN PROGRAM
BCI , STORAGE NOT USED BY PROGRAM- XXXXX TO XXXXX (OCTAL)
BCI , XXXXX TO XXXXX (DECIMAL)
BCI 8, INDEX REGISTERS USED BY THIS PROGRAM- XXXXXX
BCI 6, INDEX REGISTERS LENGTH
BCI 7, ORIGIN LENGTH
BCI 6, OCTAL -- DECIMAL OCTAL -- DECIMAL
BCI 6, OCTAL -- DECIMAL
BCI 8, OCTAL -- DECIMAL
BCI 9, COMMON STORAGE XXXXX XXXXX
BCI 5, XXXXX XXXXX
BCI 3, TRANSFER VECTOR-
BCI 8, TRANSFER VECTOR-
BCI 9, OCTAL LOCATIONS OF NUMBERED STATEMENTS IN TEXT-
BCI 9, ST NO - LOC ST NO - LOC ST NO - LOC
BCI 6, ST NO - LOC ST NO - LOC
BCI 9, ST NO - LOC ST NO - LOC
BCI 9, OCTAL LOCATIONS OF NON-DIMENSIONED NON-COMMON VARIABLES-
BCI 9, OCTAL LOCATIONS OF NON-DIMENSIONED COMMON VARIABLES-
BCI 9, OCTAL LOCATIONS OF DIMENSIONED COMMON VARIABLES-
BCI 9, OCTAL LOCATIONS OF DIMENSIONED NON-COMMON VARIABLES-
BCI 9, SYMBOL TABLE (LABELLED ST000000 IN COLS. 73-80) PREFIX 125530
BCI 6, CEDES BINARY OBJECT DECK **
* POST PASS 2 - FINISH

* THE FOLLOWING ROUTINE IS TEMPORARY AND AUTOMATICALLY SENDS
* CONTROL BACK TO THE DRIVER IF THE SYSTEM INPUT TAPE IS NOT AT
* AN END-OF-FILE.

* REM
* THIS WILL SOON BE REPLACED WITH REGARD TO FASTRAN RESIDING IN
* A MONITOR SYSTEM.

FINISH ZET MONITOR  IF UNDER IU MONITOR, GO TO FIN 3
TRA FIN3

ZET ENFILE ARE WE AT END-OF-FILE
TRA FIN1 YES, GO STOP

IDS38 EQU * BACKR MINTAP,1 BACKSPACE OVER LAST RECORD (THIS WAS
STL IOEX
BRA BACKR,++2
MZE MINTAP,++

END38 REM
TRA START ORIGINALLY READ TO HANDLE CON-
REM

IDS39 EQU * FIN1 WEOF  WRITE END-OF-FILE ON OUTPUT TAPE
FIN1 WEOF
STL IOEX
BRA WEOF,++2
MTW MLSTAP

END39 REM

IDS40 EQU * BACKR MLSTAP,1 BACKSPACE OVER THIS FILE MARK IN CASE
STL IOEX
BRA BACKR,++2
MTW MLSTAP,++

END40 REM

IDS41 EQU **3 SWT 6 PRINT THE FINAL TIME IF SS 6 DOWN
TRA **3

IDS42 EQU * RCLOCK , PRINT THE TIME
STL IOEX
BRA RCLOCK,++1

END41 REM

IDS43 EQU CLOSE MINTAP CLOSE SYSTEM INPUT TAPE
STL IOEX
BRA CLOSE,++2
MZE MINTAP

END42 REM

IDS44 EQU CLOSE MLSTAP CLOSE SYSTEM OUTPUT TAPE
STL IOEX
BRA CLOSE,++2
MTW MLSTAP

END43 REM

IDS45 EQU CLOSE SYSSBIN
STL IOEX
BRA CLOSE,++2
MTW SYSSBIN

END44 REM
IO558 EQU * REWIND SYST1 REWIND FASTRAN'S SCRATCH TAPE
STL IOEX
BRA REWIND,*,**2
MZE SYST1

* THIS INSTRUCTION IS REFERRING TO AN IOS LOCATION******
SIZ BUUT1 CLEAR SYSUT1'S BUFTBL ENTRY
BUUT1 BOOL 233

END58 REM
REM
NZE MONTOR WERE WE UNDER IU MONITOR
TRA FIN2 NO.

IO545 EQU * LOAD ...,1,10 LOAD IOS/FMS-II MONITOR
STL IOEX
BRA LOAD,,*,**3
PZE **,,1
MZE 10

END45 REM
REM
* *** FOR NON-MONITOR OPERATION (SWITCH 5 UP)
REM

IO546 EQU * FIN2 PRINT START4,,4 GIVE OPERATOR DISMOUNT INSTRUCTIONS
FIN2 STL IOEX
BRA PRINT,,*,**2
MZE START4,,4

END46 REM
REM
LQ ALLSVN ARE LAMPS IN WORKING ORDER
CAL ALLSVN
HTR START HALT... WITH RETURN TO INITIALIZATION
REM SECTION IF THERE IS ANOTHER DATA TAPE.

START4 SPACE 3,0 DISMOUNT AND LIST A-5.

FIN3 BCI 4,0 DISMOUNT AND LIST A-5.

SPACE 3

LDI "LASTLX" IS NEXT CARD AN IU MONITOR CC
IIR 410000 RETURN TO MONITOR IFF NEXT CARD IS
LFT 77 NOT BCD OR IF NEXT CARD IS AN IO/FMS
TRA IO547 CONTROL CARD (COL. 1 = 12-6-8)
RNI 770000
TRA IO538 ALLOW FASTRAN TO RETAIN CONTROL
IO547 EQU *

BACKR MINTAP,,2 PREPARE TO RETURN TO MONITOR
SIL IOEX
BRA BACKR,,*,**2
MZE MINTAP,,2

END47 REM
TRA FIN4 X
<table>
<thead>
<tr>
<th>TTL</th>
<th>POST PASS 2 - SUBROUTINES</th>
<th>FT126630</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>POST PASS 2 HAS BUT ONE SIMPLE MINDED SUBROUTINE. IT IS USED</td>
<td>FT126640</td>
</tr>
<tr>
<td></td>
<td>DURING THE MAPPING OF THE OBJECT PROGRAM AND IS USED TO RIGHT</td>
<td>FT126650</td>
</tr>
<tr>
<td></td>
<td>ADJUST A LEFT-ADJUSTED BCD SYMOL IN THE ACCUMULATOR AND INSERT</td>
<td>FT126660</td>
</tr>
<tr>
<td></td>
<td>LEADING BLANKS.</td>
<td>FT126670</td>
</tr>
<tr>
<td></td>
<td>SPACE 3</td>
<td>FT126680</td>
</tr>
<tr>
<td>RTJUST</td>
<td>STI</td>
<td>ERASE2</td>
</tr>
<tr>
<td></td>
<td>PAI</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IIR</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>RNT</td>
<td>77</td>
</tr>
<tr>
<td></td>
<td>TRA</td>
<td>RESET</td>
</tr>
<tr>
<td></td>
<td>ARS</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>ORA</td>
<td>GAR82</td>
</tr>
<tr>
<td></td>
<td>TRA</td>
<td>RTJUST+1</td>
</tr>
<tr>
<td>RESET</td>
<td>LDI</td>
<td>ERASE2</td>
</tr>
<tr>
<td></td>
<td>TRA</td>
<td>1,4</td>
</tr>
</tbody>
</table>
UTILITY ROUTINES - LIST

REM SYSTEM OUTPUT TAPE LISTER -- LIST
REM
REM THIS ROUTINE IS USED TO WRITE LISTING OUTPUT ON THE
REM SYSTEM OUTPUT TAPE MLSTAP. PAGE TITLING AND LINE AND
REM PAGE COUNTING IS HANDLED BY THIS ROUTINE.
REM
REM CALLING SEQUENCE,
REM
REM TSX LIST,4
REM (OP) LOC,COUNT
REM (RETURN)
REM WHERE,
REM LOC = ORIGIN OF MESSAGE TO GO ON MLSTAP
REM COUNT = LENGTH OF MESSAGE (WORDS)
REM (OP) = PZE IFF THIS LINE TO BE SINGLE SPACE
REM = PON IFF THIS LINE TO BE DOUBLE SPACE
REM = PTW IFF PAGE EJECT BEFORE THIS LINE GOES OUT
REM WHEN MLSTAP IS PRINTED, THE FIRST CHARACTER OF EACH
REM MESSAGE IS PRINTED BEGINNING WITH PRINT POSITION SEVEN.
REM IF LOCATION PRIQQ IS NON-ZERO THE MESSAGE GOES OUT
REM ON-LINE AS WELL AS ON MLSTAP. ON-LINE PRINTING HAS
REM FIRST CHARACTER OF THE MESSAGE IN PRINT POSITION ONE AND
REM PRINT CONTROL IS HANDLED BY THE USER.
REM
SPACE 2

LIST SXA EXLIST,4 SAVE XR4 FOR RETURN
CAL 1,4 SET ACC TO ARGUMENT
SLW LIS5 SAVE 1,4 IN TEMPORARY SAVE CELL
ARS 15 BRING (OP) INTO DECREMENT
PDX ,4 PUT (OP) IN XR4
TXH LIS11,4,2 GO GIVE ERROR MESSAGE IF INVALID (OP)
TRAX LISTV,4 GO PROCESS THIS (OP)
REM ...
REM LIS1 LDQ BLANKS SET WORD ONE TO BLANKS
CLA LIS7 GET CURRENT LINE COUNT
ADD LIS6,4 BUMP BY PROPER AMOUNT
CAS LPERPG LPERPG = MAX. LINES PER PAGE
TRA LIS4 GO TO LIS4 IF THIS WON'T FIT
NOP O.K.

STO LIS7 O.K., RESTORE LINE COUNT
LIS2 STQ LIS9 STORE FIRST WORD
CLA LIS5 GET ARGUMENT
PDX ,4 TAKE COUNT MOD 21
TIX *,4,21 X
STA **6 SET-UP WRITE MACRO
SXD **5,4 X

L548 EQU * WRITE *MLSTAP,DEC,LIS9,1,..... WRITE THIS MESSAGE
STL IOEX
BRA WRITE,**4
PTW MLSTAP,**4
PZE LIS9,**1
NZX **,**

END48 REM PRIQQ IS ON-LINE LISTING REQUESTED
TRA EXLIST
LDQ #3
STQ #3
I0549 EQU *
PRINT ...,...
STL IOEX
BRA PRINT,###
MZE ...,...
END49 REM
EXIT ...,...
TRA 2,4
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
REM...
<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>STL</td>
<td>IDEX</td>
<td>FT127990</td>
</tr>
<tr>
<td>2</td>
<td>BRA</td>
<td>PRINT,,++3</td>
<td>FT128000</td>
</tr>
<tr>
<td>3</td>
<td>PZE</td>
<td>TITLE,,20</td>
<td>FT128010</td>
</tr>
<tr>
<td>4</td>
<td>MZE</td>
<td>BLANKS,,1</td>
<td>FT128020</td>
</tr>
<tr>
<td>5</td>
<td>END53</td>
<td>REM</td>
<td>FT128030</td>
</tr>
<tr>
<td>6</td>
<td>LIS1X</td>
<td>AXT,,4 RESTORE XR4</td>
<td>FT128040</td>
</tr>
<tr>
<td>7</td>
<td>LDQ</td>
<td>BLANKS</td>
<td>FT128050</td>
</tr>
<tr>
<td>8</td>
<td>TRA</td>
<td>LIS2 GO PUT OUT THE ORIGINAL MESSAGE</td>
<td>FT128060</td>
</tr>
<tr>
<td>9</td>
<td>SPACE</td>
<td>2</td>
<td>FT128070</td>
</tr>
<tr>
<td>10</td>
<td>LIS5</td>
<td>2</td>
<td>FT128080</td>
</tr>
<tr>
<td>11</td>
<td>LIS6</td>
<td>1</td>
<td>FT128100</td>
</tr>
<tr>
<td>12</td>
<td>LIS7</td>
<td>CURRENT LINE COUNT</td>
<td>FT128110</td>
</tr>
<tr>
<td>13</td>
<td>LIS8</td>
<td>CURRENT PAGE COUNT</td>
<td>FT128120</td>
</tr>
<tr>
<td>14</td>
<td>LIS9</td>
<td></td>
<td>FT128130</td>
</tr>
<tr>
<td>15</td>
<td>LIS10</td>
<td>BCI 1,0</td>
<td>FT128140</td>
</tr>
<tr>
<td>16</td>
<td>LISTV</td>
<td></td>
<td>FT128150</td>
</tr>
<tr>
<td>17</td>
<td>LPERPG</td>
<td>PZE 59 ALLOWABLE LINES PER PAGE OF OUTPUT</td>
<td>FT128160</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>REM</td>
<td>FT128170</td>
</tr>
<tr>
<td>19</td>
<td></td>
<td>REM</td>
<td>FT128180</td>
</tr>
<tr>
<td>20</td>
<td></td>
<td>REM</td>
<td>FT128190</td>
</tr>
<tr>
<td>21</td>
<td></td>
<td>REM</td>
<td>FT128200</td>
</tr>
<tr>
<td>22</td>
<td></td>
<td>REM</td>
<td>FT128210</td>
</tr>
<tr>
<td>23</td>
<td></td>
<td>REM</td>
<td>FT128220</td>
</tr>
<tr>
<td>24</td>
<td></td>
<td>LIS11 MACER MACHINE ERROR</td>
<td>FT128230</td>
</tr>
</tbody>
</table>
UTILITY ROUTINES - SORT

REM HIGH-SPEED TABLE SORT

REM THIS ROUTINE CAN BE USED TO SORT 1- OR 2-ELEMENT
REM TABLES (WITH ELEMENT 1 OR 2 BEING THE KEY),
REM IN ASCENDING OR DESCENDING SEQUENCE, LOGICAL
REM OR ALGEBRAIC.
REM CALLING SEQUENCE:

REM CALL SORT (A, B, I, J, K, L, M)

REM WHERE,

REM \[ A = \text{LOCATION OF FIRST KEY (LOW CORE)} \]
REM \[ B = \text{LOCATION OF LAST KEY (HIGH CORE)} \]
REM \[ I = \text{NO. ELEMENTS PER TABLE ENTRY (1 OR 2)} \]
REM \[ J = \text{ELEMENT WHICH IS KEY (1 OR 2)} \]
REM \[ K = 0 \text{ FOR ASCENDING, 1 FOR DESCENDING} \]
REM \[ L = 0 \text{ FOR LOGICAL, 1 FOR ALGEBRAIC SORT} \]
REM \[ M = \text{LOCATION OF 'MASK' - ZERO IMPLIES NO MASKING} \]

REM \[ \text{THE ARGUMENT 'M' MAY BE OMITTED FROM} \]
REM \[ \text{THE CALLING SEQUENCE.} \]

REM *** NOTE ***
REM THIS SORT ROUTINE HAS BEEN SPECIALIZED (TO CONSERVE
REM SPACE) TO HANDLE THE CASE WHERE,
REM \[ I = 1 \text{ (1 ELEMENT/ENTRY)} \]
REM \[ J = 1 \text{ (KEY IS ELEMENT 1)} \]
REM \[ K = 0 \text{ (ASCENDING SORT)} \]
REM \[ L = 0 \text{ (LOGICAL SORT)} \]
REM \[ M = 0 \text{ (NO MASKING)} \]

REM THE CALLING SEQUENCE TO THIS SPECIALIZED VERSION IS,

REM CALL SORT, A, B

REM *** NOTE ***

- SORT SXA EXSORT-3,1
  SXA EXSORT-2,2
  SXA EXSORT-1,4
  CAL 1,4
  PAX 0,1
  SXD SRT1,1
  SXD SRT10,1
  SXA SRT18,1
  SXA SRT19,1
  SXA SRT21,1
  PAC ,1
  SXD SRT2,1
  AXT 1,1
  REM SRT1 TXI **1,1,**
  SXA SRT22,1
  SXA SRT25,1
  CAL 2,4
  PAX 0,1
  REM SRT2 TXI **1,1,**
  TXL EXSORT-3,1,0
<table>
<thead>
<tr>
<th>SXA</th>
<th>SRT12,1</th>
<th>SRT12,4,0</th>
<th>SRT19,2,0</th>
<th>SRT19,2,4,0</th>
<th>SRT19,2,0</th>
</tr>
</thead>
<tbody>
<tr>
<td>AXT</td>
<td>1,2</td>
<td>1,4</td>
<td>1,2</td>
<td>1,4</td>
<td>1,2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRT36 TXI</td>
<td>++1,1,1,1</td>
<td>++1,1,1,1</td>
<td>++1,1,1,1</td>
<td>++1,1,1,1</td>
<td>++1,1,1,1</td>
</tr>
<tr>
<td>SRT3 SXA</td>
<td>SRT13,1</td>
<td>SRT13,1</td>
<td>SRT13,1</td>
<td>SRT13,1</td>
<td>SRT13,1</td>
</tr>
<tr>
<td>SXD</td>
<td>SRT29,2</td>
<td>SRT29,2</td>
<td>SRT29,2</td>
<td>SRT29,2</td>
<td>SRT29,2</td>
</tr>
<tr>
<td>SRT9 CLA</td>
<td>SRT31</td>
<td>SRT31</td>
<td>SRT31</td>
<td>SRT31</td>
<td>SRT31</td>
</tr>
<tr>
<td>ARS</td>
<td>2,4</td>
<td>2,4</td>
<td>2,4</td>
<td>2,4</td>
<td>2,4</td>
</tr>
<tr>
<td>ALS</td>
<td>1,2</td>
<td>1,2</td>
<td>1,2</td>
<td>1,2</td>
<td>1,2</td>
</tr>
<tr>
<td>STO</td>
<td>SRT31</td>
<td>SRT31</td>
<td>SRT31</td>
<td>SRT31</td>
<td>SRT31</td>
</tr>
<tr>
<td>TZE</td>
<td>EXSORT-3</td>
<td>EXSORT-3</td>
<td>EXSORT-3</td>
<td>EXSORT-3</td>
<td>EXSORT-3</td>
</tr>
<tr>
<td>PAC</td>
<td>0,1</td>
<td>0,1</td>
<td>0,1</td>
<td>0,1</td>
<td>0,1</td>
</tr>
<tr>
<td>SXD</td>
<td>SRT13,1</td>
<td>SRT13,1</td>
<td>SRT13,1</td>
<td>SRT13,1</td>
<td>SRT13,1</td>
</tr>
<tr>
<td>SXD</td>
<td>SRT27,1</td>
<td>SRT27,1</td>
<td>SRT27,1</td>
<td>SRT27,1</td>
<td>SRT27,1</td>
</tr>
<tr>
<td>PAX</td>
<td>0,1</td>
<td>0,1</td>
<td>0,1</td>
<td>0,1</td>
<td>0,1</td>
</tr>
<tr>
<td>SXD</td>
<td>SRT28,1</td>
<td>SRT28,1</td>
<td>SRT28,1</td>
<td>SRT28,1</td>
<td>SRT28,1</td>
</tr>
<tr>
<td>SRT10</td>
<td>TXI</td>
<td>++1,1,1,1</td>
<td>++1,1,1,1</td>
<td>++1,1,1,1</td>
<td>++1,1,1,1</td>
</tr>
<tr>
<td>SRT10</td>
<td>SRT17,1</td>
<td>SRT17,1</td>
<td>SRT17,1</td>
<td>SRT17,1</td>
<td>SRT17,1</td>
</tr>
<tr>
<td>SRT10</td>
<td>SRT20,1</td>
<td>SRT20,1</td>
<td>SRT20,1</td>
<td>SRT20,1</td>
<td>SRT20,1</td>
</tr>
<tr>
<td>SRT12 AXT</td>
<td>++1,1</td>
<td>++1,1</td>
<td>++1,1</td>
<td>++1,1</td>
<td>++1,1</td>
</tr>
<tr>
<td>SRT13 TXI</td>
<td>++1,1,1,1</td>
<td>++1,1,1,1</td>
<td>++1,1,1,1</td>
<td>++1,1,1,1</td>
<td>++1,1,1,1</td>
</tr>
<tr>
<td>SXD</td>
<td>SRT30,1</td>
<td>SRT30,1</td>
<td>SRT30,1</td>
<td>SRT30,1</td>
<td>SRT30,1</td>
</tr>
<tr>
<td>SRT15 AXT</td>
<td>0,1</td>
<td>0,1</td>
<td>0,1</td>
<td>0,1</td>
<td>0,1</td>
</tr>
<tr>
<td>SRT16 PXA</td>
<td>0,1</td>
<td>0,1</td>
<td>0,1</td>
<td>0,1</td>
<td>0,1</td>
</tr>
<tr>
<td>PAC</td>
<td>0,2</td>
<td>0,2</td>
<td>0,2</td>
<td>0,2</td>
<td>0,2</td>
</tr>
<tr>
<td>SRT17 CAL</td>
<td>***2</td>
<td>***2</td>
<td>***2</td>
<td>***2</td>
<td>***2</td>
</tr>
<tr>
<td>SRT18 LAS</td>
<td>***2</td>
<td>***2</td>
<td>***2</td>
<td>***2</td>
<td>***2</td>
</tr>
<tr>
<td>TRA</td>
<td>SRT29</td>
<td>SRT29</td>
<td>SRT29</td>
<td>SRT29</td>
<td>SRT29</td>
</tr>
<tr>
<td>TRA</td>
<td>SRT29</td>
<td>SRT29</td>
<td>SRT29</td>
<td>SRT29</td>
<td>SRT29</td>
</tr>
<tr>
<td>REM</td>
<td>SRT29</td>
<td>SRT29</td>
<td>SRT29</td>
<td>SRT29</td>
<td>SRT29</td>
</tr>
<tr>
<td>SRT19 LDQ</td>
<td>***2</td>
<td>***2</td>
<td>***2</td>
<td>***2</td>
<td>***2</td>
</tr>
<tr>
<td>SRT20 STQ</td>
<td>***2</td>
<td>***2</td>
<td>***2</td>
<td>***2</td>
<td>***2</td>
</tr>
<tr>
<td>SRT21 SLW</td>
<td>***2</td>
<td>***2</td>
<td>***2</td>
<td>***2</td>
<td>***2</td>
</tr>
<tr>
<td>TXH</td>
<td>SRT26,4,0</td>
<td>SRT26,4,0</td>
<td>SRT26,4,0</td>
<td>SRT26,4,0</td>
<td>SRT26,4,0</td>
</tr>
<tr>
<td>REM</td>
<td>SRT22 CAL</td>
<td>***2</td>
<td>***2</td>
<td>***2</td>
<td>***2</td>
</tr>
<tr>
<td>SRT22 CAL</td>
<td>***2</td>
<td>***2</td>
<td>***2</td>
<td>***2</td>
<td>***2</td>
</tr>
<tr>
<td>SRT23 LDQ</td>
<td>***2</td>
<td>***2</td>
<td>***2</td>
<td>***2</td>
<td>***2</td>
</tr>
<tr>
<td>SRT24 SLW</td>
<td>***2</td>
<td>***2</td>
<td>***2</td>
<td>***2</td>
<td>***2</td>
</tr>
<tr>
<td>SRT25 STQ</td>
<td>***2</td>
<td>***2</td>
<td>***2</td>
<td>***2</td>
<td>***2</td>
</tr>
<tr>
<td>SRT26 TXL</td>
<td>SRT29,2,0</td>
<td>SRT29,2,0</td>
<td>SRT29,2,0</td>
<td>SRT29,2,0</td>
<td>SRT29,2,0</td>
</tr>
<tr>
<td>SRT27 TXH</td>
<td>SRT29,2,0</td>
<td>SRT29,2,0</td>
<td>SRT29,2,0</td>
<td>SRT29,2,0</td>
<td>SRT29,2,0</td>
</tr>
<tr>
<td>SRT28 TXI</td>
<td>SRT29,2,0</td>
<td>SRT29,2,0</td>
<td>SRT29,2,0</td>
<td>SRT29,2,0</td>
<td>SRT29,2,0</td>
</tr>
<tr>
<td>REM</td>
<td>SRT29 TXI</td>
<td>++1,1,1,1</td>
<td>++1,1,1,1</td>
<td>++1,1,1,1</td>
<td>++1,1,1,1</td>
</tr>
<tr>
<td>SRT30 TXH</td>
<td>SRT9,1,1</td>
<td>SRT9,1,1</td>
<td>SRT9,1,1</td>
<td>SRT9,1,1</td>
<td>SRT9,1,1</td>
</tr>
<tr>
<td>TRA</td>
<td>SRT16</td>
<td>SRT16</td>
<td>SRT16</td>
<td>SRT16</td>
<td>SRT16</td>
</tr>
<tr>
<td>SRT31</td>
<td>SRT31</td>
<td>SRT31</td>
<td>SRT31</td>
<td>SRT31</td>
<td>SRT31</td>
</tr>
<tr>
<td>AXT</td>
<td>++1</td>
<td>++1</td>
<td>++1</td>
<td>++1</td>
<td>++1</td>
</tr>
<tr>
<td>AXT</td>
<td>++2</td>
<td>++2</td>
<td>++2</td>
<td>++2</td>
<td>++2</td>
</tr>
<tr>
<td>AXT</td>
<td>++4</td>
<td>++4</td>
<td>++4</td>
<td>++4</td>
<td>++4</td>
</tr>
<tr>
<td>EXSORT TRA</td>
<td>3,4</td>
<td>3,4</td>
<td>3,4</td>
<td>3,4</td>
<td>3,4</td>
</tr>
</tbody>
</table>
**UTILITY - SORT1**

*This routine is used by PST (Punch Symbol Table) and SPECIALIZES ROUTINE SORT FOR A DESCENDING 2-ELEMENT SORT WITH SECOND ELEMENT THE KEY. AT END, SORT1 RESTORES SORT TO ITS ORIGINAL STATE. CALLING SEQUENCE,*

`CALL SORT1, A, B`

WHERE,

* A = LOCATION OF FIRST KEY
* B = LOCATION OF LAST KEY

**SPACE 3**

```
SORT1 SXA EXSRT1,4 SAVE XR4 FOR RETURN
AXT 2,1 SET UP FOR 2-ELEMENT TABLE ENTRIES
SXD SRT36,1 X
AXT 0,1 X
SXA SRT36-1,1 X
AXT 2,1 X
SXA SRT36-2,1 X
AXT -1,1 SET UP KEY AS SECOND ELEMENT
SXA SRT1-1,1 X
SXD SRT11,1 X
AXT 3,1 SET UP FOR DESCENDING SORT
CAL 1SRT1+3,1 X
SLW SRT1-3,1 X
CAL 1SRT2+3,1 X
SLW SRT10+4,1 X
TIX *-4,1,1 X
REM
CAL 1,4 PREPARE TO GO TO SORT
SLW **)4 X
CAL 2,4 X
SLW **)3 X
CALL SORT,...
REM
AXT 1,1 AT END OF SORT, RESTORE SORT
SXA SRT1-1,1 X
SXD SRT36,1 X
SXD SRT11,1 X
SXA SRT36-1,1 X
SXA SRT36-2,1 X
AXT 3,1 X
CAL 1SRT2+3,1 X
SLW SRT1-3,1 X
CAL 1SRT1+3,1 X
SLW SRT10+4,1 X
TIX *-4,1,1 X
REM
EXSRT1 AXT ...,4 RESTORE XR4 FOR FINAL RETURN
TRA 3,4 EXIT SORT1
```

**SPACE 2**

```
1SRT1 SXA SRT17,1
SXA SRT20,1
SRT1 SXA SRT18,1
SXA SRT19,1
SXA SRT21,1
```
<table>
<thead>
<tr>
<th>REM</th>
<th>UTILITY ROUTINES -- BCDFIX</th>
<th>FTI30000</th>
</tr>
</thead>
<tbody>
<tr>
<td>REM</td>
<td>BCD-TO-DECREMENT BINARY INTEGER --- BCDFIX</td>
<td>FTI30010</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30020</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30030</td>
</tr>
<tr>
<td>REM</td>
<td>THIS SUBROUTINE CONVERTS THE BCD INTEGER IN ACC (P,1-35) TO A DECREMENT BINARY INTEGER. THE BCD INTEGER IN THE ACCUMULATOR IS LEFT-JUSTIFIED WITH TRAILING BLANKS. THE RESULTING BINARY INTEGER IS PLACED IN ACC (1-17) ON EXIT FROM BCDFIX.</td>
<td>FTI30040</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30050</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30060</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30070</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30080</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30090</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30100</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30110</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30120</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30130</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30140</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30150</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30160</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30170</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30180</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30190</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30200</td>
</tr>
<tr>
<td>REM</td>
<td>CALLING SEQUENCE,</td>
<td>FTI30210</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30220</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30230</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30240</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30250</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30260</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30270</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30280</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30290</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30300</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30310</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30320</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30330</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30340</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30350</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30360</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30370</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30380</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30390</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30400</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30410</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30420</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30430</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30440</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30450</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FTI30460</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>BCDFIX</th>
<th>SXA</th>
<th>EXBDFX</th>
<th>4</th>
<th>SAVE XR4</th>
<th>FTI30210</th>
</tr>
</thead>
<tbody>
<tr>
<td>BCDFIX</td>
<td>SXA</td>
<td>EXBDFX</td>
<td>4</td>
<td>SAVE XR4</td>
<td>FTI30220</td>
</tr>
<tr>
<td>XCL</td>
<td>6,4</td>
<td>PUT ARGUMENT IN MQ</td>
<td>FTI30230</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AXI</td>
<td>SET CHARACTER COUNTER</td>
<td>FTI30240</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STZ</td>
<td>BCDF3</td>
<td>INITIALIZE RESULT TO ZERO</td>
<td>FTI30250</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TQP</td>
<td>BCDF2</td>
<td>CLEAR ACCUMULATOR FOR NEXT DIGIT</td>
<td>FTI30260</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLA</td>
<td>BCDF3</td>
<td>IF NEXT CHAR. NUMERIC GO TO BCDF2</td>
<td>FTI30270</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ALS</td>
<td>18</td>
<td>OTHERWISE GET RESULT</td>
<td>FTI30280</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXBDFX</td>
<td>AXI</td>
<td>AND SHIFT IT INTO ACC (1-17)</td>
<td>FTI30290</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>1,4</td>
<td>AND RESTORE XR4</td>
<td>FTI30300</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>AND EXIT</td>
<td>FTI30310</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FTI30320</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FTI30330</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FTI30340</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FTI30350</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FTI30360</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FTI30370</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FTI30380</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FTI30390</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FTI30400</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FTI30410</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FTI30420</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FTI30430</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FTI30440</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FTI30450</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
<td>FTI30460</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| BCD2   | 6 | SAVE NEXT DIGIT IN BCDF4 | FTI30360 |
| STQ    | BCDF4 | X | FTI30370 |
| CLA    | BCDF3 | GET ACCUMULATED RESULT SO FAR | FTI30380 |
| ALS    | 2 | MULTIPLY ACCUMULATED RESULT BY 10 | FTI30390 |
| ADD    | BCDF3 | X | FTI30400 |
| ALS    | 1 | X | FTI30410 |
| ADD    | BCDF4 | ADD THE NEW DIGIT | FTI30420 |
| STO    | BCDF3 | STORE AS NEW ACCUMULATED RESULT | FTI30430 |
| TIX    | BCDF1,4,1 | GO GET NEXT CHARACTER | FTI30440 |
| IRA    | BCDF1+2 | RETURN TO CALLER | FTI30450 |

<p>| BCDF3  | | | FTI30460 |
| BCDF4  | | | FTI30470 |</p>
<table>
<thead>
<tr>
<th>TTL</th>
<th>Utility Routines - BINBCD</th>
<th>FTI10470</th>
<th>Page 338</th>
</tr>
</thead>
<tbody>
<tr>
<td>REM</td>
<td>BINARY (INTEGER) TO BCD, LESS THAN 1,000,000 (10)</td>
<td>FTI10480</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>ENTRY... TSX BINBCD,4 NUMBER IN 21-35 ACC</td>
<td>FTI10490</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>EXIT... ANSWER IN AC WITH LEADING BLANKS</td>
<td>FTI10510</td>
<td></td>
</tr>
<tr>
<td>BINBCD</td>
<td>LDQ = 0</td>
<td>FTI10530</td>
<td></td>
</tr>
<tr>
<td></td>
<td>TZE 8X1</td>
<td>FTI10540</td>
<td></td>
</tr>
<tr>
<td></td>
<td>LRS 6</td>
<td>FTI10550</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VDP =100000,,6</td>
<td>FTI10560</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VDP =1000829,,6</td>
<td>FTI10570</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VDP =1000823,,6</td>
<td>FTI10580</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VDP =100817,,6</td>
<td>FTI10590</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VDP =10811,,6</td>
<td>FTI10600</td>
<td></td>
</tr>
<tr>
<td></td>
<td>VDP =1B5,,6</td>
<td>FTI10610</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SXA BX,4 FIX TO READ LEADING BLANKS</td>
<td>FTI10620</td>
<td></td>
</tr>
<tr>
<td>AXT</td>
<td>0,4</td>
<td>FTI10630</td>
<td></td>
</tr>
<tr>
<td>TNZ</td>
<td>**+3</td>
<td>FTI10640</td>
<td></td>
</tr>
<tr>
<td>LGL</td>
<td>6</td>
<td>FTI10650</td>
<td></td>
</tr>
<tr>
<td>TXI</td>
<td>*=2,4+6</td>
<td>FTI10660</td>
<td></td>
</tr>
<tr>
<td>ORA</td>
<td>=H 0</td>
<td>FTI10670</td>
<td></td>
</tr>
<tr>
<td>LGL</td>
<td>36,4</td>
<td>FTI10680</td>
<td></td>
</tr>
<tr>
<td>BX</td>
<td>AXT **+4</td>
<td>FTI10690</td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>1,4</td>
<td>FTI10700</td>
<td></td>
</tr>
<tr>
<td>CAL</td>
<td>=H 0 IF ZERO</td>
<td>FTI10710</td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>1,4</td>
<td>FTI10720</td>
<td></td>
</tr>
<tr>
<td>TIL</td>
<td>UTILITY ROUTINES - OCTBCD</td>
<td>FT130730</td>
<td>PAGE 339</td>
</tr>
<tr>
<td>-----</td>
<td>----------------------------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>REM</td>
<td>OCTAL TO BCD CONVERSION</td>
<td>FT130740</td>
<td></td>
</tr>
<tr>
<td>OCTBCD SXA</td>
<td>OCTX,4</td>
<td>FT130750</td>
<td></td>
</tr>
<tr>
<td>AXT</td>
<td>5,6</td>
<td>FT130760</td>
<td></td>
</tr>
<tr>
<td>LGR</td>
<td>15</td>
<td>FT130770</td>
<td></td>
</tr>
<tr>
<td>ZAC</td>
<td>3</td>
<td>FT130780</td>
<td></td>
</tr>
<tr>
<td>ALS</td>
<td>3</td>
<td>FT130790</td>
<td></td>
</tr>
<tr>
<td>LGL</td>
<td>3</td>
<td>FT130800</td>
<td></td>
</tr>
<tr>
<td>TIX</td>
<td>-2,4,1</td>
<td>FT130810</td>
<td></td>
</tr>
<tr>
<td>TZE</td>
<td>OCTY</td>
<td>FT130820</td>
<td></td>
</tr>
<tr>
<td>LGR</td>
<td>36</td>
<td>FT130830</td>
<td></td>
</tr>
<tr>
<td>AXT</td>
<td>0,4</td>
<td>FT130840</td>
<td></td>
</tr>
<tr>
<td>TNZ</td>
<td>+3</td>
<td>FT130850</td>
<td></td>
</tr>
<tr>
<td>LGL</td>
<td>6</td>
<td>FT130860</td>
<td></td>
</tr>
<tr>
<td>TXI</td>
<td>-2,4,6</td>
<td>FT130870</td>
<td></td>
</tr>
<tr>
<td>ORA</td>
<td>=H 0</td>
<td>FT130880</td>
<td></td>
</tr>
<tr>
<td>LGL</td>
<td>36,4</td>
<td>FT130890</td>
<td></td>
</tr>
<tr>
<td>ALS</td>
<td>6</td>
<td>FT130900</td>
<td></td>
</tr>
<tr>
<td>ORA</td>
<td>=H00000</td>
<td>FT130910</td>
<td></td>
</tr>
<tr>
<td>OCTX AXT</td>
<td>**,4</td>
<td>FT130920</td>
<td></td>
</tr>
<tr>
<td>OCTX CAL</td>
<td>=H 0</td>
<td>FT130930</td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>OCTX</td>
<td>FT130940</td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>1,4</td>
<td>FT130950</td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>=H00000</td>
<td>FT130960</td>
<td></td>
</tr>
<tr>
<td>TRA</td>
<td>OCTX</td>
<td>FT130970</td>
<td></td>
</tr>
<tr>
<td>UTILITY ROUTINES - CINSTR</td>
<td>FT130980</td>
<td>PAGE 340</td>
<td></td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td>REM ROUTINE TO CONVERT 12 DIGIT OCTAL NUMBER TO BCD</td>
<td>FT131000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM NO BLANKS ANYWHERE</td>
<td>FT131010</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM ENTRY... LDQ OCTAL.NUMBER</td>
<td>FT131020</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM TSX CINSTR,4</td>
<td>FT131030</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM RETURN</td>
<td>FT131040</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM ANSWER IN AC, MQ IN BCD (LOGICAL)</td>
<td>FT131050</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CINSTR SXA CINX,1</td>
<td>FT131060</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SXA CINX+1,2</td>
<td>FT131070</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AXT 2,1</td>
<td>FT131080</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AXT 6,2</td>
<td>FT131090</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ALS 3</td>
<td>FT131100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LGL 3</td>
<td>FT131110</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIX *-2,2,1</td>
<td>FT131120</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLW CIN1+2,1</td>
<td>FT131130</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIX *-5,1,1</td>
<td>FT131140</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XCL</td>
<td>FT131150</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CINX CAL CIN1</td>
<td>FT131160</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AXT **,1</td>
<td>FT131170</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AXT **,2</td>
<td>FT131180</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TRA 1,4</td>
<td>FT131190</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CIN1 BSS 2</td>
<td>FT131200</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TTL</td>
<td>UTILITY ROUTINES - STASH</td>
<td>FT131210</td>
<td>PAGE 341</td>
</tr>
<tr>
<td>-----</td>
<td>--------------------------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>STASH SXA</td>
<td>XSTASH,1</td>
<td>SAVE XR'S</td>
<td>FT131220</td>
</tr>
<tr>
<td>SXA</td>
<td>XSTASH+1,2</td>
<td>X</td>
<td>FT131230</td>
</tr>
<tr>
<td>SXA</td>
<td>XSTASH+2,4</td>
<td>X</td>
<td>FT131240</td>
</tr>
<tr>
<td>AXT</td>
<td>**+3,1</td>
<td>SET-UP 2STASH FOR RETURN</td>
<td>FT131250</td>
</tr>
<tr>
<td>SXA</td>
<td>EXSTASH,1</td>
<td>X</td>
<td>FT131260</td>
</tr>
<tr>
<td>TRA</td>
<td>2STASH</td>
<td>GO STASH THIS CARD</td>
<td>FT131270</td>
</tr>
<tr>
<td>SPACE 3</td>
<td></td>
<td></td>
<td>FT131280</td>
</tr>
<tr>
<td>AXT</td>
<td>GETFLG,1</td>
<td>RESET 2STASH</td>
<td>FT131290</td>
</tr>
<tr>
<td>SXA</td>
<td>EXSTASH,1</td>
<td>X</td>
<td>FT131300</td>
</tr>
<tr>
<td>XSTASH AXT</td>
<td>**+1</td>
<td>RESTORE XR'S</td>
<td>FT131310</td>
</tr>
<tr>
<td>AXT</td>
<td>**+2</td>
<td>X</td>
<td>FT131320</td>
</tr>
<tr>
<td>AXT</td>
<td>**+4</td>
<td>X</td>
<td>FT131330</td>
</tr>
<tr>
<td>TRA</td>
<td>1,4</td>
<td>RETURN</td>
<td>FT131340</td>
</tr>
</tbody>
</table>
ERROR PROCEDURE - ERROR

Control is given to this routine on execution of any error macro.

Macro. This routine will output the indicated diagnostic along with the location of the error macro in question, on the first entry to this routine an on-line message is printed indicating that a source error was encountered during compilation.

Standard communication is via the error macro...

Error XX,YY which is equivalent to STR FORMXX,YY

Where FORMXX identifies the pertinent error message to be output, and YY gives the point of return after the error message has been issued.

The error messages handled by this routine must not exceed 19 words in length. The word preceding the message (I.E. the word specified by the address field-1 of the str) must have the number of words in the error message (address integer).

The entry into the error routine (cell 2) is set during the pre-initialization section of the compiler.

ERROR TRA ++1 STR ENTRY. SET DURING PRE-INITIALIZE
SXA ERR5,4
CAL 0 GET TRAPPED LOCATION
SUB =1
STA ++1
ERR2 CAL ++ GET INFORMATION WORD
PDX ,4
IXH ++2,4,0 GO TO MACHINE ERROR IF NO RETURN ADDRESS
AXT LOAD4,4
SXA EREX,4
SUB =1
STA ERR7
STA ++1
LXA ***,4 GET COUNT
PX4 ,4
TXI ++1,4,3
SXD ERR4,4
PAC ,4
SXD ERR8,4
AXC 1,4
ERR7 LDQ ++,4
STW HISS+2,4
ERR8 TXL ++2,4,***
TXI ++3,4,1
REM
LXA ERR2,4
PX4 ,4
TSX OCTBCD,4
SLW HISS+1
TXS LIST,4
ERR4 HISS,***
STL ERHERE FLAG FOR ERROR IN CURRENT STATEMENT
ERR1 ZET NOCODE HAS 'NO EXECUTION' BEEN NOTED YET.
TRA ERR5 YES
CAL SBNM GET NAME OF ROUTINE
TNX ++2 SKIP IF IT IS A SUBPROGRAM
CAL =HMAIN GET 'MAIN' LABEL
SLW SCER+3 PUT IN ERROR MESSAGE
LDI PRGBOX TURN ON 'ERROR IN PROCESSOR' BIT
SIR 4
STI PRGBOX
IUS54 EQU *
PRINT SCER+6
STL IOEX
BRA PRINT,++2
MZE SCER,6

END54 REM
IUS55 EQU *
PRINT STDMP2,1
STL IOEX
BRA PRINT,++2
MZE STDMP2,1

END55 REM
REM
STL NOCODE SET 'NO EXECUTION' FLAG ON
ERR5 AXI **,4
ERR6 TRA **
REM
LOAD4 EQU *
IUS56 EQU *
PRINT QQ,5
STL IOEX
BRA PRINT,++2
MZE QQ,5

END56 REM
IUS57 EQU *
PRINT EJECT,1
STL IOEX
BRA PRINT,++2
MZE EJECT,1

END57 REM
HTR *
REM
EJECT BCI 1,1
HISS BCI 3,**** *****
S5 19
REM

PAGE 343
<table>
<thead>
<tr>
<th>EJECT</th>
<th>ERROR</th>
<th>TRANSFER CELLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>FT132360</td>
<td>ERROR1</td>
<td>1,SKEND</td>
</tr>
<tr>
<td>FT132370</td>
<td>ERROR2</td>
<td>3,SKEND</td>
</tr>
<tr>
<td>FT132380</td>
<td>ERROR4</td>
<td>5,SKEND</td>
</tr>
<tr>
<td>FT132390</td>
<td>ERROR6</td>
<td>7,SKEND</td>
</tr>
<tr>
<td>FT132400</td>
<td>ERROR8</td>
<td>9,SKEND</td>
</tr>
<tr>
<td>FT132410</td>
<td>ERROR10</td>
<td>11,SKEND</td>
</tr>
<tr>
<td>FT132420</td>
<td>ERROR12</td>
<td>13,SKEND</td>
</tr>
<tr>
<td>FT132430</td>
<td>ERROR14</td>
<td>15,SKEND</td>
</tr>
<tr>
<td>FT132440</td>
<td>ERROR16</td>
<td>17,SKEND</td>
</tr>
<tr>
<td>FT132450</td>
<td>ERROR18</td>
<td>19,SKEND</td>
</tr>
<tr>
<td>FT132460</td>
<td>ERROR20</td>
<td>21,SKEND</td>
</tr>
<tr>
<td>FT132470</td>
<td>ERROR22</td>
<td>23,SKEND</td>
</tr>
<tr>
<td>FT132480</td>
<td>ERROR24</td>
<td>25,SKEND</td>
</tr>
<tr>
<td>FT132490</td>
<td>ERROR26</td>
<td>27,SKEND</td>
</tr>
</tbody>
</table>
ITL ERROR PROCESSOR - ERROR
* THIS ROUTINE IS ENTERED WHENEVER AN ERRORY MACRO IS EXECUTED, THE ONLY MACROS OF THIS TYPE RESIDE IN SYMPASS.
* THE MACRO ERRORY CAUSES THE FOLLOWING INSTRUCTION TO BE GENERATED,
* TSX ERRORY,4,FLAG
REM
REM
* THIS ROUTINE EXAMINES FLAG TO DETERMINE THE SPECIFIC ERROR ENCONTERED ACCORDING TO THE FOLLOWING,
REM
REM FLAG = 0 UNDEFINED STATEMENT LABEL
REM FLAG = 1 UNDEFINED STATEMENT STRING
REM FLAG = 2 UNDEFINED STATEMENT VARIABLE
REM FLAG = 3 STATEMENT WITH NO PATH OF FLOW
REM FLAG = GREATER THAN 3 INVALID EQUIV ENTRY
REM
REM FOR FLAG 0 THRU 3 THE OFFENDING SYMBOL IS DUMPED INTO A POOL OF OTHER SIMILARLY OFFENDING SYMBOLS (UP TO A MAXIMUM OF 24 OF EACH). THESE SYMBOLS ARE LATER DUMPED ONTO M L S T A P IN ROUTINE OUTUND.
REM
REM FOR FLAGS GREATER THAN 3 THE DIAGNOSTIC,
REM XXXXXXXX HAS AN INVALID EQUIV MODE
REM IS OUTPUT (XXXXXX = OFFENDING SYMBOL).
REM
IN ANY CASE, CONTROL IS RETURNED TO SYMPASS FOR CONTINUED SYMATAB/EQUIV TESTING.
SPACE 3

ERRORY RIL 7000000 SET PREFIX TO 4
SIL 4000000 X
STI EQUIV,1 X
LDQ SYMTAB,1
CAL 0,4
ANA 077000000
PD0 2
TXH ERRY1,2,3
CLA TBUND,2
PDX 4
TXH SPASS1,4,23
STQ* TBUND,2
TXI **1,4,1
PXD 4
STD TBUND,2
TRA SPASS1
REM
REXY STQ FORM39
ERROR 39,SPASS1
REM
NOPHY,4,...
UNVAR,4,...
UNSTR,4,...
TBUND UNLBL,4,...
DIAGNOSTIC MESSAGES

* THESE MESSAGES ARE OUTPUT ON MSLA TAP BY THE EXECUTION OF AN ERROR MACRO. TO OUTPUT THE MESSAGE MXX, THE FOLLOWING MACRO IS EXECUTED.

REM ERROR (XX),RETURN

* SPACE 2
* AFTER THE MESSAGE IS OUT, CONTROL IS GIVEN TO "RETURN" IF IT IS NON-ZERO. OTHERWISE A CATASTROPHIC MESSAGE IS PRINTED ON-LINE AND THE COMPILER STOPS.

FORM1 BCI 4,INVALID STATEMENT NUMBER

FORM2 BCI 4,COMPILER EXPECTS COMMA

FORM3 BCI 6,COMPILER EXPECTS END OF STATEMENT

FORM4 BCI 5,COMPILER EXPECTS OCTAL NUMBER

FORM5 BCI 3,IMPROPER DIGIT

FORM6 BCI 6,STATEMENT MAY TRANSFER TO ITSELF

FORM7 BCI 5,INVALID FORM OF STATEMENT

FORM8 BCI 4,INVALID VARIABLE NAME

FORM9 BCI 4,COMPILER EXPECTS PAREN

FORM10 BCI 5,SUBPROGRAM MAY NOT CALL ITSELF

FORM11 BCI 7,STATEMENT BEGINS WITH INVALID CHARACTER

FORM12 BCI 7,COMPILER EXPECTS FIXED POINT VARIABLE

FORM13 BCI 5,MORE THAN FIVE OCTAL DIGITS

FORM14 BCI 7,VARIABLE FORMAT NAME NOT DECLARED AS ARRAY

FORM15 BCI 3,SYMBOL TOO LONG

FORM16 BCI 6,ARITHMETIC STATEMENT FUNCTION NAME

FORM17 BCI 1,XXXXX ***** THIS CELL FILLED IN WITH NAME

FORM18 BCI 5,APPEARS WITHOUT TERMINAL F

FORM19 BCI 3,VARIABLE SUBSCRIPT

FORM20 BCI 5,MORE THAN THREE DIMENSIONS

FORM21 BCI 6, DIMENSION DECLARATION OF THE ARRAY

FORM22 BCI 1,XXXXX ***** THIS CELL FILLED IN FOR EACH ERROR

FORM23 BCI 8, MUST PRECEDE APPEARANCE IN EXECUTABLE STATEMENT

FORM24 BCI 5,ARRAY ILLEGALLY ENDS IN -F

FORM25 BCI 1,XXXXX ***** THIS CELL FILLED IN FOR EACH ERROR

FORM26 BCI 9, APPEARS MORE THAN ONCE IN DIMENSION DE
<table>
<thead>
<tr>
<th>FORM</th>
<th>BCI</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORM22</td>
<td>3</td>
<td>ARRAY TOO BIG</td>
</tr>
<tr>
<td>FORM23</td>
<td>3</td>
<td>THE FUNCTION NAME</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>XXXXXX ****THIS CELL FILLED IN WITH NAME</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>APPEARS WITHOUT AN ARGUMENT</td>
</tr>
<tr>
<td>FORM24</td>
<td>4</td>
<td>FORMAT STATEMENT NUMBER REFERS TO EXECUTABLE STATEMENT</td>
</tr>
<tr>
<td>FORM25</td>
<td>4</td>
<td>THE SUBPROGRAM ARGUMENT</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>XXXXXX ****THIS CELL FILLED IN FOR EACH ERROR</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>APPEARS IN EQUIVALENCE STATEMENT</td>
</tr>
<tr>
<td>FORM26</td>
<td>6</td>
<td>INCONSISTENCY IN EQUIVALENCE DECLARATIONS INVOLVING SYMBOL</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>XXXXXX ****THIS CELL FILLED IN FOR EACH ERROR</td>
</tr>
<tr>
<td>FORM27</td>
<td>8</td>
<td>APPEARS MORE THAN ONCE IN COMMON DECLARATIONS</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>XXXXXX ****THIS CELL FILLED IN FOR EACH ERROR</td>
</tr>
<tr>
<td>FORM28</td>
<td>2</td>
<td>THE SYMBOL</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>XXXXXX ****THIS CELL FILLED IN FOR EACH ERROR</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>STATEMENT</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>IN ASSIGN OR ASSIGNED GO TO STATEMENT APPEARS IN DIMENSION</td>
</tr>
<tr>
<td>FORM29</td>
<td>3</td>
<td>STATEMENT NUMBERS BELOW ARE NEEDED IN ASSIGNED GO TO LISTS</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>OR THE VARIABLE</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>XXXXXX ****THIS CELL IS FILLED IN FOR EACH ERROR</td>
</tr>
<tr>
<td>FORM30</td>
<td>3</td>
<td>ASSIGN VARIABLE</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>XXXXXX ****THIS CELL IS FILLED IN FOR EACH ERROR</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>NOT IN ASSIGNED GO TO STATEMENT</td>
</tr>
<tr>
<td>FORM31</td>
<td>4</td>
<td>ASSIGNED GO TO VARIABLE</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>XXXXXX ****THIS CELL IS FILLED IN FOR EACH ERROR</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>NOT IN ASSIGN STATEMENT</td>
</tr>
<tr>
<td>FORM32</td>
<td>3</td>
<td>PROGRAM TOO LONG</td>
</tr>
<tr>
<td>FORM33</td>
<td>8</td>
<td>LAST EXECUTABLE STATEMENT MUST BE A TRANSFER</td>
</tr>
<tr>
<td>FORM34</td>
<td>9</td>
<td>FIXED POINT INTERNAL OR LIBRARY FUNCTION INVALID IN DP OR CAF</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>STATEMENT</td>
</tr>
<tr>
<td>FORM35</td>
<td>9</td>
<td>WRONG COLUMN 1 MODE FOR ARITHMETIC STATEMENT FUNCTION</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>STATEMENT</td>
</tr>
<tr>
<td>FORM36</td>
<td>9</td>
<td>WRONG NUMBER OR MODE OF ARGUMENTS IN ARITHMETIC STATEMENT</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>NCITION</td>
</tr>
<tr>
<td>FORM37</td>
<td>3</td>
<td>MISSING END CARD</td>
</tr>
<tr>
<td>FORM38</td>
<td>7</td>
<td>NO SYMBOL TABLE Punched -- POOL OVERFLOW</td>
</tr>
<tr>
<td>FORM39</td>
<td>6</td>
<td>XXXXXX HAS AN INVALID EQUIV MODES</td>
</tr>
<tr>
<td>FORM40</td>
<td>7</td>
<td>INVALID FORM OF SUBSCRIPT FOR ARRAY</td>
</tr>
<tr>
<td>FORM41</td>
<td>1</td>
<td>XXXXXX ****THIS CELL IS FILLED IN WITH NAME</td>
</tr>
<tr>
<td>FORM42</td>
<td>4</td>
<td>INVALID COMPLEX CONSTANT</td>
</tr>
<tr>
<td>FORM74</td>
<td>4, CONSECUTIVE ** ILLEGAL</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------</td>
<td></td>
</tr>
<tr>
<td>FORM75</td>
<td>3, INVALID COMMA</td>
<td></td>
</tr>
<tr>
<td>FORM76</td>
<td>8, SUBPROGRAM NAME IS ARGUMENT OF LIBRARY FUNCTION</td>
<td></td>
</tr>
<tr>
<td>FORM77</td>
<td>3, ILLEGAL CHARACTER</td>
<td></td>
</tr>
<tr>
<td>FORM78</td>
<td>9, ARITHMETIC STATEMENT FUNCTION DEFINED MORE THAN ONCE</td>
<td></td>
</tr>
<tr>
<td>FORM79</td>
<td>3, MISPLACED PERIOD</td>
<td></td>
</tr>
<tr>
<td>FORM80</td>
<td>5, INVALID CHARACTER IN COLUMN 1</td>
<td></td>
</tr>
<tr>
<td>FORM81</td>
<td>10, COMPILED EXPECTS STATEMENT NUMBER, INTEGER OR VARIABLE</td>
<td></td>
</tr>
<tr>
<td>FORM82</td>
<td>8, 0 IS NOT VALID AS STATEMENT NUMBER OR SUBSCRIPT</td>
<td></td>
</tr>
<tr>
<td>FORM83</td>
<td>7, NO STATEMENT ASSOCIATED WITH ABOVE LABEL</td>
<td></td>
</tr>
<tr>
<td>FORM84</td>
<td>5, LEFT SIDE OF STATEMENT INVALID</td>
<td></td>
</tr>
<tr>
<td>FORM85</td>
<td>8, FUNCTION NAME USED AS ARRAY OR SUBPROGRAM NAME</td>
<td></td>
</tr>
<tr>
<td>FORM86</td>
<td>8, ABOVE DO NESTS WITHIN A DO HAVING SAME DO INDEX</td>
<td></td>
</tr>
<tr>
<td>FORM87</td>
<td>9, DO INDEX APPEARS WITHIN THE DO ON LEFT OF EQUAL SIGN</td>
<td></td>
</tr>
<tr>
<td>FORM88</td>
<td>8, BUILT-IN FUNCTION APPEARS ON LEFT OF EQUAL SIGN</td>
<td></td>
</tr>
<tr>
<td>FORM89</td>
<td>11, ARITHMETIC STATEMENT FUNCTIONS MUST PRECEDE EXECUTABLE STATEMENTS</td>
<td></td>
</tr>
<tr>
<td>FORM90</td>
<td>9, SUBSCRIPTS ILLEGAL IN ARITHMETIC STATEMENT FUNCTIONS</td>
<td></td>
</tr>
<tr>
<td>FORM91</td>
<td>3, UNMATCHED PAREN</td>
<td></td>
</tr>
<tr>
<td>FORM92</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FORM93</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FORM94</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FORM95</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FORM96</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FORM97</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FORM98</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FORM99</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSG100</td>
<td>8, COMPILER CAPACITY EXCEEDED. SIMPLIFY PROGRAM.</td>
<td></td>
</tr>
<tr>
<td>MCR</td>
<td>3, MACHINE ERROR ***</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>??</td>
<td>5, TROUBLE. TAKE CORE DUMP.</td>
<td></td>
</tr>
<tr>
<td>SCER</td>
<td>3, SOURCE ERROR IN</td>
<td></td>
</tr>
<tr>
<td>BCI 1</td>
<td>XXXXXX</td>
<td></td>
</tr>
<tr>
<td>BCI 2</td>
<td>PROGRAM.</td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>TABLES</td>
<td></td>
</tr>
<tr>
<td>SPACE 9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>FASTRAN COMPILER</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>RESEARCH COMPUTING CENTER</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>INDIANA UNIVERSITY</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>BLOOMINGTON, INDIANA</td>
<td></td>
</tr>
<tr>
<td>SPACE 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td>TABLES</td>
<td></td>
</tr>
</tbody>
</table>

Remainder of the page is not visible.
The following are the various tables used or generated by FASTRAN: During compilation with the exception of POOL, SYMTAB and EQUIV (these are described later).
TABLES - OPERATOR RANK TABLE

* EACH ARITHMETIC OPERATOR IS ASSIGNED A RANK IN THIS TABLE.
* ALTHOUGH AT PRESENT EACH OPERATOR HAS A UNIQUE RANK, FACILITY IS
* BUILT INTO THE TABLE FOR ASSIGNING ARBITRARY RANKS. THE RANK
* MAY BE INSERTED INTO THE ARITHMETIC STACK (ASTACK), OR IF CODE
* MUST BE GENERATED FIRST. THE OPERATORS ARE RANKED AS FOLLOWS,
* WITH THE LOWEST RANK (LEAST CODE-FORCING) OPERATOR FIRST...

REM
* UNARY MINUS, EXPONENTIATION, DIVIDE, MULTIPLY, BINARY MINUS,
* PLUS, COMMA, EQUALS, LEFT PAREN, RIGHT PAREN, ARITHMETIC
* STATEMENT FUNCTION EQUAL SIGN.

REM
* FOR BOOLEAN STATEMENTS, THE RANKS ARE...

REM
* AND (EQUIVALENT TO MULTIPLY), OR (EQUIVALENT TO PLUS),
* COMPLEMENT (EQUIVALENT TO UNARY MINUS).

REM
* EACH RANK KEY WORD CONTAINS, AMONG OTHER INFORMATION, THE
* RANK AND THE ADDRESS OF THE APPROPRIATE PROCESSOR IN THE
* COMPIL ROUTINE. BIT 16 IS ON IN EACH RANK KEY WORD, INDICATING
* TO THE COMPIL ROUTINE THAT THIS WORD IS AN OPERATOR.

SPACE 2

RANKTB EQU *
RANK  NEG,NEG,UNEQGK,,MINUSP - UNARY MINUS  FT135880
RANK  EXPNL,EXPNL,EXPKEY,EXPONP ** EXPONENTIAL  FT135890
RANK  DIV,DIV,DIVKEY,DIVIDP / DIVIDE  FT135900
RANK  TIMES,TIMES,MPYKEY,OTHROP * MULTIPLY  FT135910
RANK  BNEG,BNEG,NEGKEY - BINARY MINUS  FT135920
RANK  PLUS,PLUS,PLUSKY,,PLUSP + PLUS  FT135930
RANK  COMMA,COMMA,CMAKEY,COMMAP , COMMA  FT135940
RANK  EQUALS,EQUALS,EQLKEY,EQUALP = EQUALS  FT135950
RANK  LPAREN,LPAREN,LPNKEY,,LPARNP ( LEFT PAREN  FT135960
RANK  RPAREN,RPAREN,RPNKEY,,RPARNP ) RIGHT PAREN  FT135970
RANK  ARSTFN,ARSTFN,ASFKKEY = ARITH. STATE. FN. EQUALSFT135980

REM
L RANK EQU *-RANKTB LENGTH OF RANK TABLE  FT136000
REM
REM
REM BOOLEAN OPERATOR EQUIVALENCES

AND  EQU TIMES * BOOLEAN AND  FT136040
OR  EQU PLUS + BOOLEAN OR  FT136050
COMP  EQU NEG - BOOLEAN COMPLEMENT  FT136060
**TABLES - GENERATOR KEY TABLES**

- The generator key tables contain coded descriptions of all allowed arithmetic operations. For an operation of the form A .OPERATOR.B, either of the elements A or B may currently be in the accumulator, in the MQ, or in storage.
- A typical key word in the table might contain the following information (see the key macro for the bit pattern)...
- MODE OF A-OPRAND, LOCATION (AC, MQ, VS) OF A-OPRAND,
- OPERATOR, MODE OF B-OPRAND, LOCATION OF B-OPRAND.

REM

- In addition, the key word contains the address of the appropriate code generator. For unary operations (like -B), the information for the A-oprand is missing.

REM

- There are four generator key tables, one for each pertinent column 1 mode... fixed/floating point, boolean, double precision, and complex arithmetic. Each of these tables contains entries for all allowable operations, including built-in functions and certain other convenient operations internal to the compiler. The four tables are sorted in the pre-initialization phase of the compiler, and remain sorted throughout the compiler's stay in core.

REM

- The compil routine establishes the correct table to use on the basis of the current column 1 mode.
- The gen routine creates a key word (minus the address of the code generator) from the information it finds in astack, and does a masked binary search on the appropriate table.
- If an equality is found with the generated key, then code is produced by the routine designated in the address of the located table word. Most of the syntactical errors in arithmetic statements are discovered by not finding a key in the table.
<table>
<thead>
<tr>
<th>EJECT</th>
<th>KEYTAB</th>
<th>KEYTB0</th>
<th>REM</th>
<th>REAL AND INTEGER KEYS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>KEY1</td>
<td>KEY2</td>
<td></td>
<td>INT, REAL, INTO</td>
</tr>
<tr>
<td></td>
<td>KEY2</td>
<td></td>
<td></td>
<td>REAL, SIGN, REAL, SIGN</td>
</tr>
<tr>
<td></td>
<td>KEY2</td>
<td></td>
<td></td>
<td>REAL, DIM, REAL, DIMO</td>
</tr>
<tr>
<td></td>
<td>KEY2</td>
<td></td>
<td></td>
<td>INTEGR, XDIM, INTEGR, XDIM</td>
</tr>
<tr>
<td></td>
<td>KEY2</td>
<td></td>
<td></td>
<td>INTEGR, MAXO, INTEGR, MXMIN</td>
</tr>
<tr>
<td></td>
<td>KEY2</td>
<td></td>
<td></td>
<td>REAL, MAXI, REAL, MXMIN</td>
</tr>
<tr>
<td></td>
<td>KEY2</td>
<td></td>
<td></td>
<td>INTEGR, XMAXO, INTEGR, MXMIN</td>
</tr>
<tr>
<td></td>
<td>KEY2</td>
<td></td>
<td></td>
<td>REAL, XMAXI, REAL, MXMIN</td>
</tr>
<tr>
<td></td>
<td>KEY2</td>
<td></td>
<td></td>
<td>INTEGR, MINO, INTEGR, MXMIN</td>
</tr>
<tr>
<td></td>
<td>KEY2</td>
<td></td>
<td></td>
<td>REAL, MINI, REAL, MXMIN</td>
</tr>
<tr>
<td></td>
<td>KEY2</td>
<td></td>
<td></td>
<td>INTEGR, XMINTO, INTEGR, XMINTO</td>
</tr>
<tr>
<td></td>
<td>KEY2</td>
<td></td>
<td></td>
<td>REAL, XMINTO, REAL, MXMIN</td>
</tr>
<tr>
<td></td>
<td>KEY2</td>
<td></td>
<td></td>
<td>REAL, MOD, REAL, MODO</td>
</tr>
<tr>
<td></td>
<td>KEY2</td>
<td></td>
<td></td>
<td>INTEGR, XMODO, INTEGR, XMODO</td>
</tr>
<tr>
<td></td>
<td>KEY</td>
<td></td>
<td></td>
<td>INTEGR, VS, EQUALS, INTEGR, VS, IEQI01</td>
</tr>
<tr>
<td></td>
<td>KEY</td>
<td></td>
<td></td>
<td>INTEGR, VS, EQUALS, INTEGR, AC, IEQI02</td>
</tr>
<tr>
<td></td>
<td>KEY</td>
<td></td>
<td></td>
<td>INTEGR, VS, EQUALS, INTEGR, MQ, IEQI03</td>
</tr>
<tr>
<td></td>
<td>KEY</td>
<td></td>
<td></td>
<td>REAL, VS, EQUALS, REAL, VS, IEQRO1</td>
</tr>
<tr>
<td></td>
<td>KEY</td>
<td></td>
<td></td>
<td>REAL, VS, EQUALS, REAL, AC, IEQRO2</td>
</tr>
<tr>
<td></td>
<td>KEY</td>
<td></td>
<td></td>
<td>REAL, VS, EQUALS, REAL, MQ, IEQRO3</td>
</tr>
<tr>
<td></td>
<td>KEY</td>
<td></td>
<td></td>
<td>REAL, VS, EQUALS, REAL, VS, REQRO1</td>
</tr>
<tr>
<td></td>
<td>KEY</td>
<td></td>
<td></td>
<td>REAL, VS, EQUALS, REAL, AC, REQRO2</td>
</tr>
<tr>
<td></td>
<td>KEY</td>
<td></td>
<td></td>
<td>REAL, VS, EQUALS, REAL, MQ, REQRO3</td>
</tr>
<tr>
<td></td>
<td>KEY1</td>
<td></td>
<td></td>
<td>ABS, REAL, ABSO</td>
</tr>
<tr>
<td></td>
<td>KEY1</td>
<td></td>
<td></td>
<td>XABS, INTEGR, XABSO</td>
</tr>
<tr>
<td></td>
<td>KEY1</td>
<td></td>
<td></td>
<td>XINT, REAL, XINTO</td>
</tr>
<tr>
<td></td>
<td>KEY1</td>
<td></td>
<td></td>
<td>FLOAT, INTEGR, FLOAT</td>
</tr>
<tr>
<td></td>
<td>KEY1</td>
<td></td>
<td></td>
<td>XFIX, REAL, XINTO</td>
</tr>
<tr>
<td></td>
<td>KEY1</td>
<td></td>
<td></td>
<td>NEG, INTEGR, INEGO</td>
</tr>
<tr>
<td></td>
<td>KEY1</td>
<td></td>
<td></td>
<td>NEG, REAL, RNEG0</td>
</tr>
<tr>
<td></td>
<td>KEY2</td>
<td></td>
<td></td>
<td>INTEGR, PLUS, INTEGR, IADD0</td>
</tr>
<tr>
<td></td>
<td>KEY2</td>
<td></td>
<td></td>
<td>INTEGR, BNEG, INTEGR, ISUB0</td>
</tr>
<tr>
<td></td>
<td>KEY2</td>
<td></td>
<td></td>
<td>INTEGR, TIMES, INTEGR, IMPOY0</td>
</tr>
<tr>
<td></td>
<td>KEY2</td>
<td></td>
<td></td>
<td>INTEGR, DIV, INTEGR, IDIV0</td>
</tr>
<tr>
<td></td>
<td>KEY2</td>
<td></td>
<td></td>
<td>INTEGR, EXPNL, INTEGR, IEXP0</td>
</tr>
<tr>
<td></td>
<td>KEY2</td>
<td></td>
<td></td>
<td>REAL, PLUS, REAL, RADD0</td>
</tr>
<tr>
<td></td>
<td>KEY2</td>
<td></td>
<td></td>
<td>REAL, BNEG, REAL, RSUB0</td>
</tr>
<tr>
<td></td>
<td>KEY2</td>
<td></td>
<td></td>
<td>REAL, TIMES, REAL, RMPY0</td>
</tr>
<tr>
<td></td>
<td>KEY2</td>
<td></td>
<td></td>
<td>REAL, DIV, REAL, RDIV0</td>
</tr>
<tr>
<td></td>
<td>KEY2</td>
<td></td>
<td></td>
<td>REAL, EXPNL, REAL, REXP0</td>
</tr>
<tr>
<td></td>
<td>KEY2</td>
<td></td>
<td></td>
<td>REAL, EXPNL, INTEGR, REXP0</td>
</tr>
<tr>
<td></td>
<td>KEY</td>
<td></td>
<td></td>
<td><strong>AUXIL, REAL, VS, AUXIL</strong></td>
</tr>
<tr>
<td></td>
<td>KEY</td>
<td></td>
<td></td>
<td><strong>AUXIL, REAL, AC, AUXIL</strong></td>
</tr>
<tr>
<td></td>
<td>KEY</td>
<td></td>
<td></td>
<td><strong>AUXIL, REAL, MQ, AUXIL2</strong></td>
</tr>
<tr>
<td></td>
<td>KEY</td>
<td></td>
<td></td>
<td><strong>AUXIL, INTEGR, VS, AUXIL</strong></td>
</tr>
<tr>
<td></td>
<td>KEY</td>
<td></td>
<td></td>
<td><strong>AUXIL, INTEGR, AC, AUXIL6</strong></td>
</tr>
<tr>
<td></td>
<td>KEY</td>
<td></td>
<td></td>
<td><strong>AUXIL, INTEGR, MQ, AUXIL2</strong></td>
</tr>
<tr>
<td></td>
<td>KEY</td>
<td></td>
<td></td>
<td><strong>XLOC, INTEGR, VS, XLOC01</strong></td>
</tr>
<tr>
<td></td>
<td>KEY</td>
<td></td>
<td></td>
<td><strong>XLOC, REAL, VS, XLOC01</strong></td>
</tr>
<tr>
<td></td>
<td>KEY</td>
<td></td>
<td></td>
<td>REAL, ARSTFN, REAL, VS, ASFRR1</td>
</tr>
<tr>
<td></td>
<td>KEY</td>
<td></td>
<td></td>
<td>REAL, ARSTFN, REAL, AC, ASFRR2</td>
</tr>
<tr>
<td></td>
<td>KEY</td>
<td></td>
<td></td>
<td>REAL, ARSTFN, REAL, MQ, ASFRR3</td>
</tr>
</tbody>
</table>
EJECT

KEYTB1 EQU * BOOLEAN KEYS

REM

KEY4 ABS,REAL,ABSO

KEY4 FLOAT,INTEGR,FLOAT

KEY4 INT,REAL,INTO

KEY3 REAL,MOD,REAL,MODO

KEY3 REAL,SIGN,REAL,SIGN0

KEY3 REAL,DIAM,REAL,DIMO

KEY3 INTEGR,MAXI,REAL,MXMIN

KEY3 REAL,MAXI,REAL,MXMIN

KEY3 INTEGR,MINI,INTEGR,MXMIN

KEY3 REAL,MINI,REAL,MXMIN

KEY3 REAL,AND,REAL,BANDO

KEY3 REAL,OR,REAL,BORO

KEY 0,0,COMP,REAL,VS,BCOM01

KEY 0,0,COMP,REAL,AC,BCOM02

KEY REAL,VS,EQUALS,REAL,VS,BEQ01

KEY REAL,VS,EQUALS,REAL,AC,BEQ02

KEY ,,AUXIL,REAL,VS, AUXIL3

KEY ,,AUXIL,REAL,AC,AUXIL7

KEY ,,AUXIL,INTEGR,AC,AUXIL7

KEY REAL,,ARSTFN,REAL,VS,ASFRR4

KEY REAL,,ARSTFN,REAL,AC,ASFRR5

REM

LKEYTB1 EQU *-KEYTB1 LENGTH OF BOOLEAN KEYS
**TABLES - BUILTIN FUNCTION NAMES**

- The open function table contains two-word entries for each.
- Built-in function in Fastran. The first word of the entry is
  a key word for use by the compil and opngen routines. The
  key consists mainly of a 'pseudo-rank', similar to an
  operator rank, but designed to permit easy recognition as
  a built-in function, and differentiation between
  the different functions. The second word of the entry is the
  BCD name of the function.

REM
- The scan routine, when it encounters a function name
  ending in -F, searches this table (linearly) to detect a
  built-in function. If one is found, the keyword (first word
  in the entry) is used in a stack.

REM
- In order to differentiate between certain single, double,
  and complex functions, the scan routine will append a number
  to the beginning of a double precision or complex function name.
  This number is 4 for double and 1 for complex.
  These functions appear thus in the table.

REM
- The function XLOC has been made a built-in function in Fastran.
- Notice that the IBM double precision and complex *fix* functions
  generate incorrect code, but the names are included in this table.
- For compatibility.

**LIBTAB**

```
LIBTAB EQU 2
```

```
LIBTAB EQU Open routine names
TAB3 MIN1 These eight max and min entries
TAB3 MINO Must be first in LIBTAB, and
TAB3 MAXO in this order. This is necessary
TAB3 MAXI So that the code generators
TAB3 XMNO Will work properly.
TAB3 XMIN1
TAB3 XMAX1
TAB3 XMAXO
TAB3 ABS
TAB3 XABS
TAB3 INT
TAB3 XINT
TAB3 FLOAT
TAB3 XFLOAT
TAB3 MOD
TAB3 XMOD
TAB3 SIGN
TAB3 XSIGN
TAB3 DIM
TAB3 XDIM
TAB3 4ABS DABS 4
TAB3 4FLOAT DFLOAT
TAB3 4FIX DFIX
TAB3 4SIGN DSIGN
TAB3 1FLOAT IFLOAT
TAB3 1FIX IFIX
TAB3 1SIGN ISIGN
TAB3 XLOC
```

**LPIPE**

```
LPIPE EQU 1-LIBTAB Length of open routine name table
```
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTI38850</td>
<td>The reserved library name table contains the names of the two-word entry consists of a flag word and the BCD name of the routine. The flag words are all zeroed out during the initialization of the compiler. When a routine is used in the generated code, the flag word is set non-zero. Later the post pass 1 routine TVCTR and the pass 2 routine SVCTR determine which routines have been used, and insert the BCD names into the transfer vector.</td>
</tr>
<tr>
<td>FTI38970</td>
<td>SPECIAL LIBRARY NAME TABLE. INCLUDES ALL NAMES OF LIBRARY ROUTINES CALLED BY THE PROCESSORS.</td>
</tr>
<tr>
<td>FTI38990</td>
<td>(FPT)</td>
</tr>
<tr>
<td>FTI39000</td>
<td>BCI 1, (FPT)</td>
</tr>
<tr>
<td>FTI39010</td>
<td>EXPL1 BCI 1, EXPL1</td>
</tr>
<tr>
<td>FTI39020</td>
<td>EXPL2 BCI 1, EXPL2</td>
</tr>
<tr>
<td>FTI39030</td>
<td>EXPL3 BCI 1, EXPL3</td>
</tr>
<tr>
<td>FTI39040</td>
<td>D阜AD BCI 1, D阜AD</td>
</tr>
<tr>
<td>FTI39050</td>
<td>D阜SB BCI 1, D阜SB</td>
</tr>
<tr>
<td>FTI39060</td>
<td>(DFMP) BCI 1, (DFMP)</td>
</tr>
<tr>
<td>FTI39070</td>
<td>DFDP BCI 1, (DFDP)</td>
</tr>
<tr>
<td>FTI39080</td>
<td>DEXP2 BCI 1, DEXP2</td>
</tr>
<tr>
<td>FTI39090</td>
<td>DEXP3 BCI 1, DEXP3</td>
</tr>
<tr>
<td>FTI39100</td>
<td>(IFMP) BCI 1, (IFMP)</td>
</tr>
<tr>
<td>FTI39110</td>
<td>IFDP BCI 1, IFDP</td>
</tr>
<tr>
<td>FTI39120</td>
<td>IEXP2 BCI 1, IEXP2</td>
</tr>
<tr>
<td>FTI39130</td>
<td>(ISH) BCI 1, (ISH)</td>
</tr>
<tr>
<td>FTI39140</td>
<td>BCI 1, (ISH)</td>
</tr>
<tr>
<td>FTI39150</td>
<td>(STH) BCI 1, (STH)</td>
</tr>
<tr>
<td>FTI39160</td>
<td>(CSH) BCI 1, (CSH)</td>
</tr>
<tr>
<td>FTI39170</td>
<td>(SCH) BCI 1, (SCH)</td>
</tr>
<tr>
<td>FTI39180</td>
<td>(SPH) BCI 1, (SPH)</td>
</tr>
<tr>
<td>FTI39190</td>
<td>BST BCI 1, (BST)</td>
</tr>
<tr>
<td>FTI39200</td>
<td>(WLR) BCI 1, (WLR)</td>
</tr>
<tr>
<td>FTI39210</td>
<td>(TSB) BCI 1, (TSB)</td>
</tr>
<tr>
<td>FTI39220</td>
<td>(KLR) BCI 1, (KLR)</td>
</tr>
<tr>
<td>FTI39230</td>
<td>(BST) BCI 1, (BST)</td>
</tr>
<tr>
<td>Instruction</td>
<td>Address</td>
</tr>
<tr>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>ADD</td>
<td>FT139720</td>
</tr>
<tr>
<td>ALS17</td>
<td>FT139730</td>
</tr>
<tr>
<td>ALS18</td>
<td>FT139740</td>
</tr>
<tr>
<td>ANA</td>
<td>FT139750</td>
</tr>
<tr>
<td>ANA12</td>
<td>FT139760</td>
</tr>
<tr>
<td>AX11</td>
<td>FT139770</td>
</tr>
<tr>
<td>AX12</td>
<td>FT139780</td>
</tr>
<tr>
<td>AX14</td>
<td>FT139790</td>
</tr>
<tr>
<td>CAL</td>
<td>FT139800</td>
</tr>
<tr>
<td>CAL4</td>
<td>FT139810</td>
</tr>
<tr>
<td>CHS</td>
<td>FT139820</td>
</tr>
<tr>
<td>CLA</td>
<td>FT139830</td>
</tr>
<tr>
<td>CLA774</td>
<td>FT139840</td>
</tr>
<tr>
<td>CLA775</td>
<td>FT139850</td>
</tr>
<tr>
<td>CLA776</td>
<td>FT139860</td>
</tr>
<tr>
<td>CLA777</td>
<td>FT139870</td>
</tr>
<tr>
<td>CLS</td>
<td>FT139880</td>
</tr>
<tr>
<td>CLS774</td>
<td>FT139890</td>
</tr>
<tr>
<td>CLS775</td>
<td>FT139900</td>
</tr>
<tr>
<td>CLS776</td>
<td>FT139910</td>
</tr>
<tr>
<td>CLS777</td>
<td>FT139920</td>
</tr>
<tr>
<td>DCT</td>
<td>FT139930</td>
</tr>
<tr>
<td>DVP</td>
<td>FT139940</td>
</tr>
<tr>
<td>DVP777</td>
<td>FT139950</td>
</tr>
<tr>
<td>FAD</td>
<td>FT139960</td>
</tr>
<tr>
<td>FAD774</td>
<td>FT139970</td>
</tr>
<tr>
<td>FAD775</td>
<td>FT139980</td>
</tr>
<tr>
<td>FAD776</td>
<td>FT139990</td>
</tr>
<tr>
<td>FAD777</td>
<td>FT140000</td>
</tr>
<tr>
<td>FAD778</td>
<td>FT140010</td>
</tr>
<tr>
<td>FDP</td>
<td>FT140020</td>
</tr>
<tr>
<td>FDP777</td>
<td>FT140030</td>
</tr>
<tr>
<td>FMP</td>
<td>FT140040</td>
</tr>
<tr>
<td>FMP777</td>
<td>FT140050</td>
</tr>
<tr>
<td>FRN</td>
<td>FT140060</td>
</tr>
<tr>
<td>FSB</td>
<td>FT140070</td>
</tr>
<tr>
<td>FSB775</td>
<td>FT140080</td>
</tr>
<tr>
<td>FSB777</td>
<td>FT140090</td>
</tr>
<tr>
<td>LDA</td>
<td>FT140100</td>
</tr>
<tr>
<td>LDC4</td>
<td>FT140110</td>
</tr>
<tr>
<td>LDQ</td>
<td>FT140120</td>
</tr>
<tr>
<td>LDQ2</td>
<td>FT140130</td>
</tr>
<tr>
<td>LDQ774</td>
<td>FT140140</td>
</tr>
<tr>
<td>LDQ775</td>
<td>FT140150</td>
</tr>
<tr>
<td>LDQ776</td>
<td>FT140160</td>
</tr>
<tr>
<td>LDQ777</td>
<td>FT140170</td>
</tr>
<tr>
<td>LGR74</td>
<td>FT140180</td>
</tr>
<tr>
<td>LLS</td>
<td>FT140190</td>
</tr>
<tr>
<td>LLS18</td>
<td>FT140200</td>
</tr>
<tr>
<td>LRS</td>
<td>FT140210</td>
</tr>
<tr>
<td>LRS18</td>
<td>FT140220</td>
</tr>
<tr>
<td>LRS35</td>
<td>FT140230</td>
</tr>
<tr>
<td>LXD4</td>
<td>FT140240</td>
</tr>
<tr>
<td>Instruction</td>
<td>Location</td>
</tr>
<tr>
<td>-------------</td>
<td>----------</td>
</tr>
<tr>
<td>MPY</td>
<td>F104250</td>
</tr>
<tr>
<td>NOP</td>
<td>F104260</td>
</tr>
<tr>
<td>ORA9</td>
<td>F104270</td>
</tr>
<tr>
<td>ORA1</td>
<td>F104280</td>
</tr>
<tr>
<td>PAX1</td>
<td>F104290</td>
</tr>
<tr>
<td>PAX4</td>
<td>F104300</td>
</tr>
<tr>
<td>PAXA</td>
<td>F104310</td>
</tr>
<tr>
<td>PAXD4</td>
<td>F104320</td>
</tr>
<tr>
<td>PZE</td>
<td>F104330</td>
</tr>
<tr>
<td>SLF</td>
<td>F104340</td>
</tr>
<tr>
<td>SLF</td>
<td>F104350</td>
</tr>
<tr>
<td>SLT</td>
<td>F104360</td>
</tr>
<tr>
<td>SLW</td>
<td>F104370</td>
</tr>
<tr>
<td>SLW8</td>
<td>F104380</td>
</tr>
<tr>
<td>SPP</td>
<td>F104390</td>
</tr>
<tr>
<td>STA</td>
<td>F104400</td>
</tr>
<tr>
<td>STO</td>
<td>F104410</td>
</tr>
<tr>
<td>STQ</td>
<td>F104420</td>
</tr>
<tr>
<td>STQ2</td>
<td>F104430</td>
</tr>
<tr>
<td>STQ74</td>
<td>F104440</td>
</tr>
<tr>
<td>STQ76</td>
<td>F104450</td>
</tr>
<tr>
<td>STQ77</td>
<td>F104460</td>
</tr>
<tr>
<td>STZ</td>
<td>F104470</td>
</tr>
<tr>
<td>STZ90</td>
<td>F104480</td>
</tr>
<tr>
<td>STX</td>
<td>F104490</td>
</tr>
<tr>
<td>SVA</td>
<td>F104500</td>
</tr>
<tr>
<td>SVA2</td>
<td>F104510</td>
</tr>
<tr>
<td>SVA4</td>
<td>F104520</td>
</tr>
<tr>
<td>SVA5</td>
<td>F104530</td>
</tr>
<tr>
<td>SVA6</td>
<td>F104540</td>
</tr>
<tr>
<td>SVA7</td>
<td>F104550</td>
</tr>
<tr>
<td>SVA8</td>
<td>F104560</td>
</tr>
<tr>
<td>SVD44</td>
<td>F104570</td>
</tr>
<tr>
<td>SVD4</td>
<td>F104580</td>
</tr>
<tr>
<td>TXQ2</td>
<td>F104590</td>
</tr>
<tr>
<td>TXQ4</td>
<td>F104600</td>
</tr>
<tr>
<td>TXQ6</td>
<td>F104610</td>
</tr>
<tr>
<td>TXQ8</td>
<td>F104620</td>
</tr>
<tr>
<td>TXQ20</td>
<td>F104630</td>
</tr>
<tr>
<td>TXQ22</td>
<td>F104640</td>
</tr>
<tr>
<td>TXQ24</td>
<td>F104650</td>
</tr>
<tr>
<td>TXQ26</td>
<td>F104660</td>
</tr>
<tr>
<td>TXQ28</td>
<td>F104670</td>
</tr>
<tr>
<td>TXQ2A</td>
<td>F104680</td>
</tr>
<tr>
<td>TXQ2C</td>
<td>F104690</td>
</tr>
<tr>
<td>TXQ2E</td>
<td>F104700</td>
</tr>
<tr>
<td>TXQ2G</td>
<td>F104710</td>
</tr>
<tr>
<td>TSX</td>
<td>F104720</td>
</tr>
<tr>
<td>TSX4</td>
<td>F104730</td>
</tr>
<tr>
<td>TXH4</td>
<td>F104740</td>
</tr>
<tr>
<td>TXH40</td>
<td>F104750</td>
</tr>
<tr>
<td>TXI1</td>
<td>F104760</td>
</tr>
<tr>
<td>TXI14</td>
<td>F104770</td>
</tr>
<tr>
<td>TXI16</td>
<td>F104780</td>
</tr>
<tr>
<td>TXI18</td>
<td>F104790</td>
</tr>
<tr>
<td>TXI20</td>
<td>F104800</td>
</tr>
<tr>
<td>TXI22</td>
<td>F104810</td>
</tr>
<tr>
<td>TXI24</td>
<td>F104820</td>
</tr>
<tr>
<td>TXI26</td>
<td>F104830</td>
</tr>
</tbody>
</table>

LOPTAB EQU **-OPTAB
<table>
<thead>
<tr>
<th>Table</th>
<th>Values</th>
<th>FT</th>
</tr>
</thead>
<tbody>
<tr>
<td>TTL</td>
<td>TABLES - TRAILING BLANK INSERT</td>
<td>FT140840</td>
</tr>
<tr>
<td>*</td>
<td>TABLE OF CONSTANTS USED BY THE SCANNER TO INSERT TRAILING</td>
<td>FT140850</td>
</tr>
<tr>
<td>*</td>
<td>BLANKS ONTO A SYMBOL</td>
<td>FT140860</td>
</tr>
<tr>
<td>OCT</td>
<td>777777777777</td>
<td>FT140870</td>
</tr>
<tr>
<td>OCT</td>
<td>777777777700</td>
<td>FT140880</td>
</tr>
<tr>
<td>OCT</td>
<td>777777770000</td>
<td>FT140890</td>
</tr>
<tr>
<td>OCT</td>
<td>777777000000</td>
<td>FT140900</td>
</tr>
<tr>
<td>OCT</td>
<td>777700000000</td>
<td>FT140910</td>
</tr>
<tr>
<td>OCT</td>
<td>770000000000</td>
<td>FT140920</td>
</tr>
<tr>
<td>SCANT1</td>
<td>PZE</td>
<td>FT140930</td>
</tr>
<tr>
<td>OCT</td>
<td>60</td>
<td>FT140940</td>
</tr>
<tr>
<td>OCT</td>
<td>6060</td>
<td>FT140950</td>
</tr>
<tr>
<td>OCT</td>
<td>606060</td>
<td>FT140960</td>
</tr>
<tr>
<td>OCT</td>
<td>60606060</td>
<td>FT140970</td>
</tr>
<tr>
<td>OCT</td>
<td>6060606060</td>
<td>FT140980</td>
</tr>
<tr>
<td>OCT</td>
<td>606060606060</td>
<td>FT140990</td>
</tr>
<tr>
<td>SCANT2</td>
<td>OCT</td>
<td>FT141000</td>
</tr>
<tr>
<td>OCT</td>
<td>344413241541</td>
<td>38</td>
</tr>
<tr>
<td>---------</td>
<td>----------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>OCT</td>
<td>340653551066</td>
<td>37</td>
</tr>
<tr>
<td>OCT</td>
<td>335674440705</td>
<td>36</td>
</tr>
<tr>
<td>OCT</td>
<td>332543515403</td>
<td>35</td>
</tr>
<tr>
<td>OCT</td>
<td>326554174006</td>
<td>34</td>
</tr>
<tr>
<td>OCT</td>
<td>323443311470</td>
<td>33</td>
</tr>
<tr>
<td>OCT</td>
<td>320202556055</td>
<td>32</td>
</tr>
<tr>
<td>OCT</td>
<td>314004260106</td>
<td>31</td>
</tr>
<tr>
<td>OCT</td>
<td>311320214722</td>
<td>30</td>
</tr>
<tr>
<td>OCT</td>
<td>306563327102</td>
<td>29</td>
</tr>
<tr>
<td>OCT</td>
<td>303617422401</td>
<td>28</td>
</tr>
<tr>
<td>OCT</td>
<td>277177204002</td>
<td>27</td>
</tr>
<tr>
<td>OCT</td>
<td>274631003151</td>
<td>26</td>
</tr>
<tr>
<td>OCT</td>
<td>271024002440</td>
<td>25</td>
</tr>
<tr>
<td>OCT</td>
<td>265354635550</td>
<td>24</td>
</tr>
<tr>
<td>OCT</td>
<td>262760512755</td>
<td>23</td>
</tr>
<tr>
<td>OCT</td>
<td>257446725444</td>
<td>22</td>
</tr>
<tr>
<td>OCT</td>
<td>253561357240</td>
<td>21</td>
</tr>
<tr>
<td>OCT</td>
<td>250132614200</td>
<td>20</td>
</tr>
<tr>
<td>OCT</td>
<td>245110475000</td>
<td>19</td>
</tr>
<tr>
<td>OCT</td>
<td>241647310000</td>
<td>18</td>
</tr>
<tr>
<td>OCT</td>
<td>236356240000</td>
<td>17</td>
</tr>
<tr>
<td>OCT</td>
<td>233760200000</td>
<td>16</td>
</tr>
<tr>
<td>OCT</td>
<td>227432000000</td>
<td>15</td>
</tr>
<tr>
<td>OCT</td>
<td>224510000000</td>
<td>14</td>
</tr>
<tr>
<td>OCT</td>
<td>221240000000</td>
<td>13</td>
</tr>
<tr>
<td>OCT</td>
<td>215400000000</td>
<td>12</td>
</tr>
<tr>
<td>OCT</td>
<td>212000000000</td>
<td>11</td>
</tr>
<tr>
<td>OCT</td>
<td>207000000000</td>
<td>10</td>
</tr>
<tr>
<td>OCT</td>
<td>203000000000</td>
<td>09</td>
</tr>
<tr>
<td>OCT</td>
<td>200000000000</td>
<td>08</td>
</tr>
<tr>
<td>OCT</td>
<td>175000000000</td>
<td>07</td>
</tr>
<tr>
<td>OCT</td>
<td>171000000000</td>
<td>06</td>
</tr>
<tr>
<td>OCT</td>
<td>166000000000</td>
<td>05</td>
</tr>
<tr>
<td>OCT</td>
<td>163000000000</td>
<td>04</td>
</tr>
<tr>
<td>OCT</td>
<td>157000000000</td>
<td>03</td>
</tr>
<tr>
<td>OCT</td>
<td>154000000000</td>
<td>02</td>
</tr>
<tr>
<td>OCT</td>
<td>151000000000</td>
<td>01</td>
</tr>
<tr>
<td>OCT</td>
<td>146000000000</td>
<td>00</td>
</tr>
<tr>
<td>OCT</td>
<td>142314631463</td>
<td>-1</td>
</tr>
<tr>
<td>OCT</td>
<td>137727024365</td>
<td>-2</td>
</tr>
<tr>
<td>OCT</td>
<td>134570551767</td>
<td>-3</td>
</tr>
<tr>
<td>OCT</td>
<td>130616031310</td>
<td>-4</td>
</tr>
<tr>
<td>OCT</td>
<td>125161550741</td>
<td>-5</td>
</tr>
<tr>
<td>OCT</td>
<td>122132755432</td>
<td>-6</td>
</tr>
<tr>
<td>OCT</td>
<td>116536257220</td>
<td>-7</td>
</tr>
<tr>
<td>OCT</td>
<td>113430214163</td>
<td>-8</td>
</tr>
<tr>
<td>OCT</td>
<td>110023326451</td>
<td>-9</td>
</tr>
<tr>
<td>OCT</td>
<td>104353675565</td>
<td>-10</td>
</tr>
<tr>
<td>OCT</td>
<td>101274544452</td>
<td>-11</td>
</tr>
<tr>
<td>OCT</td>
<td>076226752042</td>
<td>-12</td>
</tr>
<tr>
<td>OCT</td>
<td>072044566404</td>
<td>-13</td>
</tr>
<tr>
<td>OCT</td>
<td>067520453466</td>
<td>-14</td>
</tr>
<tr>
<td>OCT</td>
<td>OCT</td>
<td>OCT</td>
</tr>
<tr>
<td>-----</td>
<td>-----</td>
<td>-----</td>
</tr>
</tbody>
</table>
### Double Precision Floating Point Powers of 10

<table>
<thead>
<tr>
<th>Octal</th>
<th>Decimal</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>377454732312</td>
<td>136044676992</td>
<td>F141850</td>
</tr>
<tr>
<td>373741367020</td>
<td>136044676992</td>
<td>F141860</td>
</tr>
<tr>
<td>370601137163</td>
<td>136044676992</td>
<td>F141870</td>
</tr>
<tr>
<td>365464114134</td>
<td>136044676992</td>
<td>F141880</td>
</tr>
<tr>
<td>361755023372</td>
<td>136044676992</td>
<td>F141890</td>
</tr>
<tr>
<td>356612334310</td>
<td>136044676992</td>
<td>F141900</td>
</tr>
<tr>
<td>353473426555</td>
<td>136044676992</td>
<td>F141910</td>
</tr>
<tr>
<td>347770675742</td>
<td>136044676992</td>
<td>F141920</td>
</tr>
<tr>
<td>344623713116</td>
<td>136044676992</td>
<td>F141930</td>
</tr>
<tr>
<td>341503074076</td>
<td>136044676992</td>
<td>F141940</td>
</tr>
<tr>
<td>336402374713</td>
<td>136044676992</td>
<td>F141950</td>
</tr>
<tr>
<td>332635456171</td>
<td>136044676992</td>
<td>F141960</td>
</tr>
<tr>
<td>327512676455</td>
<td>136044676992</td>
<td>F141970</td>
</tr>
<tr>
<td>324410545213</td>
<td>136044676992</td>
<td>F141980</td>
</tr>
<tr>
<td>320647410336</td>
<td>136044676992</td>
<td>F141990</td>
</tr>
<tr>
<td>315522640261</td>
<td>136044676992</td>
<td>F142000</td>
</tr>
<tr>
<td>312417031701</td>
<td>136044676992</td>
<td>F142010</td>
</tr>
<tr>
<td>306661534465</td>
<td>136044676992</td>
<td>F142020</td>
</tr>
<tr>
<td>303532743536</td>
<td>136044676992</td>
<td>F142030</td>
</tr>
<tr>
<td>300425434430</td>
<td>136044676992</td>
<td>F142040</td>
</tr>
<tr>
<td>274674055531</td>
<td>136044676992</td>
<td>F142050</td>
</tr>
<tr>
<td>271543212741</td>
<td>136044676992</td>
<td>F142060</td>
</tr>
<tr>
<td>266434157115</td>
<td>136044676992</td>
<td>F142070</td>
</tr>
<tr>
<td>262706576511</td>
<td>136044676992</td>
<td>F142080</td>
</tr>
<tr>
<td>257553630407</td>
<td>136044676992</td>
<td>F142090</td>
</tr>
<tr>
<td>254443023471</td>
<td>136044676992</td>
<td>F142100</td>
</tr>
<tr>
<td>250721522450</td>
<td>136044676992</td>
<td>F142110</td>
</tr>
<tr>
<td>24556416672</td>
<td>136044676992</td>
<td>F142120</td>
</tr>
<tr>
<td>242452013710</td>
<td>136044676992</td>
<td>F142130</td>
</tr>
<tr>
<td>23673665450</td>
<td>136044676992</td>
<td>F142140</td>
</tr>
<tr>
<td>233575360400</td>
<td>136044676992</td>
<td>F142150</td>
</tr>
<tr>
<td>230461132000</td>
<td>136044676992</td>
<td>F142160</td>
</tr>
<tr>
<td>22475020000</td>
<td>136044676992</td>
<td>F142170</td>
</tr>
<tr>
<td>221606500000</td>
<td>136044676992</td>
<td>F142180</td>
</tr>
<tr>
<td>216470400000</td>
<td>136044676992</td>
<td>F142190</td>
</tr>
<tr>
<td>212764000000</td>
<td>136044676992</td>
<td>F142200</td>
</tr>
<tr>
<td>207620000000</td>
<td>136044676992</td>
<td>F142210</td>
</tr>
<tr>
<td>204500000000</td>
<td>136044676992</td>
<td>F142220</td>
</tr>
<tr>
<td>201400000000</td>
<td>136044676992</td>
<td>F142230</td>
</tr>
<tr>
<td>197561463146</td>
<td>136044676992</td>
<td>F142240</td>
</tr>
<tr>
<td>193507534121</td>
<td>136044676992</td>
<td>F142250</td>
</tr>
<tr>
<td>19046111564</td>
<td>136044676992</td>
<td>F142260</td>
</tr>
<tr>
<td>186363334272</td>
<td>136044676992</td>
<td>F142270</td>
</tr>
<tr>
<td>182517426524</td>
<td>136044676992</td>
<td>F142280</td>
</tr>
<tr>
<td>179414336750</td>
<td>136044676992</td>
<td>F142290</td>
</tr>
<tr>
<td>175615376246</td>
<td>136044676992</td>
<td>F142300</td>
</tr>
<tr>
<td>1714527461670</td>
<td>136044676992</td>
<td>F142310</td>
</tr>
<tr>
<td>168422701372</td>
<td>136044676992</td>
<td>F142320</td>
</tr>
<tr>
<td>164376633766</td>
<td>136044676992</td>
<td>F142330</td>
</tr>
<tr>
<td>1603437657770</td>
<td>136044676992</td>
<td>F142340</td>
</tr>
<tr>
<td>1561534223542</td>
<td>136044676992</td>
<td>F142350</td>
</tr>
<tr>
<td>1520376461248</td>
<td>136044676992</td>
<td>F142360</td>
</tr>
<tr>
<td>1478873689400</td>
<td>136044676992</td>
<td>F142370</td>
</tr>
<tr>
<td>1437437574760</td>
<td>136044676992</td>
<td>F142380</td>
</tr>
<tr>
<td>1395903513520</td>
<td>136044676992</td>
<td>F142390</td>
</tr>
<tr>
<td>1354379452880</td>
<td>136044676992</td>
<td>F142400</td>
</tr>
<tr>
<td>1312846392240</td>
<td>136044676992</td>
<td>F142410</td>
</tr>
<tr>
<td>1271313331600</td>
<td>136044676992</td>
<td>F142420</td>
</tr>
<tr>
<td>1229770234000</td>
<td>136044676992</td>
<td>F142430</td>
</tr>
<tr>
<td>1188227137600</td>
<td>136044676992</td>
<td>F142440</td>
</tr>
<tr>
<td>1146684041280</td>
<td>136044676992</td>
<td>F142450</td>
</tr>
<tr>
<td>1105141045920</td>
<td>136044676992</td>
<td>F142460</td>
</tr>
<tr>
<td>1063598050560</td>
<td>136044676992</td>
<td>F142470</td>
</tr>
<tr>
<td>1022055056240</td>
<td>136044676992</td>
<td>F142480</td>
</tr>
<tr>
<td>979512061920</td>
<td>136044676992</td>
<td>F142490</td>
</tr>
<tr>
<td>936969067600</td>
<td>136044676992</td>
<td>F142500</td>
</tr>
<tr>
<td>894426073280</td>
<td>136044676992</td>
<td>F142510</td>
</tr>
<tr>
<td>851883078960</td>
<td>136044676992</td>
<td>F142520</td>
</tr>
<tr>
<td>809340084640</td>
<td>136044676992</td>
<td>F142530</td>
</tr>
<tr>
<td>766797090320</td>
<td>136044676992</td>
<td>F142540</td>
</tr>
<tr>
<td>724254095920</td>
<td>136044676992</td>
<td>F142550</td>
</tr>
<tr>
<td>681711101600</td>
<td>136044676992</td>
<td>F142560</td>
</tr>
<tr>
<td>639168097280</td>
<td>136044676992</td>
<td>F142570</td>
</tr>
<tr>
<td>596625092960</td>
<td>136044676992</td>
<td>F142580</td>
</tr>
<tr>
<td>554082098640</td>
<td>136044676992</td>
<td>F142590</td>
</tr>
<tr>
<td>OCT</td>
<td>105447113564</td>
<td>-18</td>
</tr>
<tr>
<td>OCT</td>
<td>101730171123</td>
<td>-19</td>
</tr>
<tr>
<td>OCT</td>
<td>076571624103</td>
<td>-20</td>
</tr>
<tr>
<td>OCT</td>
<td>073456166402</td>
<td>-21</td>
</tr>
<tr>
<td>OCT</td>
<td>067743444004</td>
<td>-22</td>
</tr>
<tr>
<td>OCT</td>
<td>064602666320</td>
<td>-23</td>
</tr>
<tr>
<td>OCT</td>
<td>061465370246</td>
<td>-24</td>
</tr>
<tr>
<td>OCT</td>
<td>055757132075</td>
<td>-25</td>
</tr>
<tr>
<td>OCT</td>
<td>052614110061</td>
<td>-26</td>
</tr>
<tr>
<td>OCT</td>
<td>047474723215</td>
<td>-27</td>
</tr>
<tr>
<td>OCT</td>
<td>043773036657</td>
<td>-28</td>
</tr>
<tr>
<td>OCT</td>
<td>040625513677</td>
<td>-29</td>
</tr>
<tr>
<td>OCT</td>
<td>035504411377</td>
<td>-30</td>
</tr>
<tr>
<td>OCT</td>
<td>032403472631</td>
<td>-31</td>
</tr>
<tr>
<td>OCT</td>
<td>026637304365</td>
<td>-32</td>
</tr>
<tr>
<td>OCT</td>
<td>023514235135</td>
<td>-33</td>
</tr>
<tr>
<td>OCT</td>
<td>020411666074</td>
<td>-34</td>
</tr>
<tr>
<td>OCT</td>
<td>014651264555</td>
<td>-35</td>
</tr>
<tr>
<td>OCT</td>
<td>011524220444</td>
<td>-36</td>
</tr>
<tr>
<td>OCT</td>
<td>006420163520</td>
<td>-37</td>
</tr>
<tr>
<td>OCT</td>
<td>002663437347</td>
<td>-38</td>
</tr>
</tbody>
</table>
**EJECT**

**DOUBLE PRECISION FIXED POINT POWERS OF 10**

<table>
<thead>
<tr>
<th>SPACE</th>
<th>OCT</th>
<th>DEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>314631463146</td>
<td>-1</td>
</tr>
<tr>
<td></td>
<td>256050753412</td>
<td>-2</td>
</tr>
<tr>
<td></td>
<td>136152375747</td>
<td>-3</td>
</tr>
<tr>
<td></td>
<td>326161031312</td>
<td>-4</td>
</tr>
<tr>
<td></td>
<td>142161550741</td>
<td>-5</td>
</tr>
<tr>
<td></td>
<td>275013275543</td>
<td>-6</td>
</tr>
<tr>
<td></td>
<td>371232571275</td>
<td>-7</td>
</tr>
<tr>
<td></td>
<td>230734214106</td>
<td>-8</td>
</tr>
<tr>
<td></td>
<td>134057501155</td>
<td>-9</td>
</tr>
<tr>
<td></td>
<td>157467754727</td>
<td>-10</td>
</tr>
<tr>
<td></td>
<td>127753776057</td>
<td>-11</td>
</tr>
<tr>
<td></td>
<td>010627463004</td>
<td>-12</td>
</tr>
<tr>
<td></td>
<td>000702270232</td>
<td>-13</td>
</tr>
<tr>
<td></td>
<td>000055022334</td>
<td>-14</td>
</tr>
<tr>
<td></td>
<td>00004401657</td>
<td>-15</td>
</tr>
<tr>
<td></td>
<td>0000346453</td>
<td>-16</td>
</tr>
<tr>
<td></td>
<td>0000027035</td>
<td>-17</td>
</tr>
<tr>
<td></td>
<td>0000002234</td>
<td>-18</td>
</tr>
<tr>
<td></td>
<td>000000600166</td>
<td>-19</td>
</tr>
<tr>
<td></td>
<td>0000000013</td>
<td>-20</td>
</tr>
<tr>
<td></td>
<td>000000000001</td>
<td>-21</td>
</tr>
</tbody>
</table>

**F10L EQU**
<table>
<thead>
<tr>
<th>OCT</th>
<th>DEC</th>
<th>OCT</th>
<th>DEC</th>
<th>OCT</th>
<th>DEC</th>
<th>OCT</th>
<th>DEC</th>
<th>OCT</th>
<th>DEC</th>
<th>OCT</th>
<th>DEC</th>
<th>OCT</th>
<th>DEC</th>
<th>OCT</th>
<th>DEC</th>
<th>OCT</th>
<th>DEC</th>
</tr>
</thead>
<tbody>
<tr>
<td>031463146314</td>
<td>-1</td>
<td>002436560507</td>
<td>-2</td>
<td>000203044672</td>
<td>-3</td>
<td>000015066705</td>
<td>-4</td>
<td>000001237055</td>
<td>-5</td>
<td>000000103067</td>
<td>-6</td>
<td>000000006553</td>
<td>-7</td>
<td>000000000527</td>
<td>-8</td>
<td>00000000042</td>
<td>-9</td>
</tr>
<tr>
<td>0</td>
<td>-11</td>
<td>0</td>
<td>-12</td>
<td>0</td>
<td>-13</td>
<td>0</td>
<td>-14</td>
<td>0</td>
<td>-15</td>
<td>0</td>
<td>-16</td>
<td>0</td>
<td>-17</td>
<td>0</td>
<td>-18</td>
<td>0</td>
<td>-19</td>
</tr>
</tbody>
</table>

**FROM EQU**

* DOUBLE PRECISION FIXED POINT POWERS OF 10

**SPACE**

3

* FT142920 FT142930 FT142940 FT142950 FT142960 FT142970 FT142980 FT142990 FT143000 FT143010 FT143020 FT143030 FT143040 FT143050 FT143060 FT143070 FT143080 FT143090 FT143100 FT143110 FT143120 FT143130 FT143140 FT143150 FT143160 FT143170
<table>
<thead>
<tr>
<th>TABLES</th>
<th>BCI</th>
<th>CIT</th>
<th>FLAGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>THIS TABLE IS USED BY CITBLD WHEN THE LIST PASS 1 OPTION IS IN EFFECT.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLES</th>
<th>BCI</th>
<th>CIT</th>
<th>FLAGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1TBL1</td>
<td>1, N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCI</td>
<td>1, P</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCI</td>
<td>1, R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCI</td>
<td>1, V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCI</td>
<td>1, T</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCI</td>
<td>1, F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCI</td>
<td>1, C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCI</td>
<td>1, S</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCI</td>
<td>1, X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCI</td>
<td>1, W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCI</td>
<td>1, E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCI</td>
<td>1, Q</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCI</td>
<td>1, D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCI</td>
<td>1, G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCI</td>
<td>1, H</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCI</td>
<td>1, I</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCI</td>
<td>1, J</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCI</td>
<td>1, K</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CONVTB</td>
<td>OCT</td>
<td>020004001000</td>
<td>02</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
<td>---------------</td>
<td>----</td>
</tr>
<tr>
<td>OCT</td>
<td>020000401000</td>
<td>05</td>
<td>04</td>
</tr>
<tr>
<td>OCT</td>
<td>002000400100</td>
<td>08</td>
<td>07</td>
</tr>
<tr>
<td>OCT</td>
<td>010200000001</td>
<td>11(=)</td>
<td>09</td>
</tr>
<tr>
<td>OCT</td>
<td>000000000042</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OCT</td>
<td>440040000000</td>
<td>17(A)</td>
<td>16(+)</td>
</tr>
<tr>
<td>OCT</td>
<td>404041004200</td>
<td>20(D)</td>
<td>19(C)</td>
</tr>
<tr>
<td>OCT</td>
<td>400440104020</td>
<td>23(G)</td>
<td>22(F)</td>
</tr>
<tr>
<td>OCT</td>
<td>000040014002</td>
<td>-</td>
<td>25(I)</td>
</tr>
<tr>
<td>OCT</td>
<td>000040424102</td>
<td>-</td>
<td>28(=)</td>
</tr>
<tr>
<td>OCT</td>
<td>200000000000</td>
<td>32(=)</td>
<td>-</td>
</tr>
<tr>
<td>OCT</td>
<td>210022002400</td>
<td>35(L)</td>
<td>34(K)</td>
</tr>
<tr>
<td>OCT</td>
<td>201020202040</td>
<td>38(O)</td>
<td>37(N)</td>
</tr>
<tr>
<td>OCT</td>
<td>200120022004</td>
<td>41(P)</td>
<td>40(Q)</td>
</tr>
<tr>
<td>OCT</td>
<td>204221020000</td>
<td>44(=)</td>
<td>43($)</td>
</tr>
<tr>
<td>OCT</td>
<td>000000001042</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>TTL</td>
<td>TABLES - BCD CNV</td>
<td>FTI43650</td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>----------------</td>
<td>----------</td>
<td></td>
</tr>
<tr>
<td></td>
<td>THIS TABLE IS USED BY COLUMN LABEL TO CONVERT BINARY CARD SEQUENCE</td>
<td>FTI43660</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NUMBER TO COLUMN BINARY.</td>
<td>FTI43670</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SPACE 3</td>
<td>FTI43680</td>
<td></td>
</tr>
<tr>
<td>BCD CNV</td>
<td>VFD</td>
<td>06/00, 015/00000, 15/BCD CNV</td>
<td>FTI43690</td>
</tr>
<tr>
<td></td>
<td>VFD</td>
<td>06/01, 015/00000, 15/BCD CNV</td>
<td>FTI43700</td>
</tr>
<tr>
<td></td>
<td>VFD</td>
<td>06/02, 015/00000, 15/BCD CNV</td>
<td>FTI43710</td>
</tr>
<tr>
<td></td>
<td>VFD</td>
<td>06/03, 015/00000, 15/BCD CNV</td>
<td>FTI43720</td>
</tr>
<tr>
<td></td>
<td>VFD</td>
<td>06/04, 015/00000, 15/BCD CNV</td>
<td>FTI43730</td>
</tr>
<tr>
<td></td>
<td>VFD</td>
<td>06/05, 015/00000, 15/BCD CNV</td>
<td>FTI43740</td>
</tr>
<tr>
<td></td>
<td>VFD</td>
<td>06/06, 015/00000, 15/BCD CNV</td>
<td>FTI43750</td>
</tr>
<tr>
<td></td>
<td>VFD</td>
<td>06/07, 015/00000, 15/BCD CNV</td>
<td>FTI43760</td>
</tr>
<tr>
<td></td>
<td>VFD</td>
<td>06/10, 015/00000, 15/BCD CNV</td>
<td>FTI43770</td>
</tr>
<tr>
<td></td>
<td>VFD</td>
<td>06/11, 015/00000, 15/BCD CNV</td>
<td>FTI43780</td>
</tr>
<tr>
<td></td>
<td>VFD</td>
<td>06/00, 015/00000, 15/BCD CNV+1</td>
<td>FTI43790</td>
</tr>
<tr>
<td></td>
<td>VFD</td>
<td>06/01, 015/00000, 15/BCD CNV+1</td>
<td>FTI43800</td>
</tr>
<tr>
<td></td>
<td>VFD</td>
<td>06/02, 015/00000, 15/BCD CNV+1</td>
<td>FTI43810</td>
</tr>
<tr>
<td></td>
<td>VFD</td>
<td>06/03, 015/00000, 15/BCD CNV+1</td>
<td>FTI43820</td>
</tr>
<tr>
<td></td>
<td>VFD</td>
<td>06/04, 015/00000, 15/BCD CNV+1</td>
<td>FTI43830</td>
</tr>
<tr>
<td></td>
<td>VFD</td>
<td>06/05, 015/00000, 15/BCD CNV+1</td>
<td>FTI43840</td>
</tr>
<tr>
<td></td>
<td>VFD</td>
<td>06/06, 015/00000, 15/BCD CNV+1</td>
<td>FTI43850</td>
</tr>
<tr>
<td></td>
<td>VFD</td>
<td>06/07, 015/00000, 15/BCD CNV+1</td>
<td>FTI43860</td>
</tr>
<tr>
<td></td>
<td>VFD</td>
<td>06/10, 015/00000, 15/BCD CNV+1</td>
<td>FTI43870</td>
</tr>
<tr>
<td></td>
<td>VFD</td>
<td>06/11, 015/00000, 15/BCD CNV+1</td>
<td>FTI43880</td>
</tr>
<tr>
<td>TTL</td>
<td>TABLES - MONTH</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>----------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>*</td>
<td>THIS TABLE IS USED BY GETCH TO CONVERT A SIX-CHARACTER DATE TO A READABLE FORM FOR PAGE HEADINGS.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>FT143890</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SPACE</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCI</td>
<td>1, DEC 19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCI</td>
<td>1, NOV 19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCI</td>
<td>1, OCT 19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCI</td>
<td>1, SEP 19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCI</td>
<td>1, AUG 19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCI</td>
<td>1, JUL 19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCI</td>
<td>1, JUN 19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCI</td>
<td>1, MAY 19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCI</td>
<td>1, APR 19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCI</td>
<td>1, MAR 19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCI</td>
<td>1, FEB 19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCI</td>
<td>1, JAN 19</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MONTH</td>
<td>EQU</td>
<td>*</td>
<td></td>
</tr>
</tbody>
</table>

<p>| FT143900 |
| FT143910 |
| FT143920 |
| FT143930 |
| FT143940 |
| FT143950 |
| FT143960 |
| FT143970 |
| FT143980 |
| FT143990 |
| FT144000 |
| FT144010 |
| FT144020 |
| FT144030 |
| FT144040 |
| FT144050 |</p>
<table>
<thead>
<tr>
<th></th>
<th>MISCELLANEOUS CONSTANTS</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>TTL</td>
<td>2</td>
<td>FT144060</td>
</tr>
<tr>
<td>ADREL OCT</td>
<td>760000000000</td>
<td>FT144070</td>
</tr>
<tr>
<td>OCT76 OCT</td>
<td>760000000000</td>
<td>FT144080</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FT144090</td>
</tr>
<tr>
<td>*</td>
<td>THE ORDER OF THE FOLLOWING THREE CARDS IS CRITICAL</td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FT144100</td>
</tr>
<tr>
<td>PCWD1 OCT</td>
<td>400504000000</td>
<td>FT144110</td>
</tr>
<tr>
<td>PCWD2</td>
<td>400504000000</td>
<td>FT144120</td>
</tr>
<tr>
<td>PCWD3</td>
<td>400504000000</td>
<td>FT144130</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FT144140</td>
</tr>
<tr>
<td>ARGDEL OCT</td>
<td>77777777000000</td>
<td>FT144150</td>
</tr>
<tr>
<td>ARGDEL OCT</td>
<td>77777777000000</td>
<td>TO MASK OUT ALL BUT ARG. NO. AND DELTA</td>
</tr>
<tr>
<td>DELBIT OCT</td>
<td>77777777000000</td>
<td>FT144160</td>
</tr>
<tr>
<td>DELBIT OCT</td>
<td>77777777000000</td>
<td>TO MASK OUT ALL BUT DELTA</td>
</tr>
<tr>
<td>TUPCOM OCT</td>
<td>77461</td>
<td>FT144170</td>
</tr>
<tr>
<td>OESBIT</td>
<td>7</td>
<td>FT144180</td>
</tr>
<tr>
<td>OESBIT</td>
<td>7</td>
<td>FT144190</td>
</tr>
<tr>
<td>ONE</td>
<td>1</td>
<td>FT144200</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FT144210</td>
</tr>
<tr>
<td>ALLSVN OCT</td>
<td>777777777777</td>
<td>FT144220</td>
</tr>
<tr>
<td>ALLSVN OCT</td>
<td>777777777777</td>
<td>ALL BITS ON</td>
</tr>
<tr>
<td></td>
<td>FT144230</td>
<td></td>
</tr>
<tr>
<td></td>
<td>FT144240</td>
<td></td>
</tr>
</tbody>
</table>
TTL UNIT DESIGNATIONS AND BUFFERS

REM SPACE 9

REM * * * * * * * * * *

REM FASTRAN COMPILER

REM RESEARCH COMPUTING CENTER

REM INDIANA UNIVERSITY

REM BLOOMINGTON, INDIANA

REM SPACE 3

REM * * * * * * * * * *

REM TAPE BUFFERS

REM * * * * * * * * * *

FT144250
FT144260
FT144270
FT144280
FT144290
FT144300
FT144310
FT144320
FT144330
FT144340
FT144350
FT144360
FT144370
FT144380
FT144390
FT144400
FT144410
FT144420
FT144430
* THE LOGICAL/PHYSICAL TAPE EQUIVALENCES ARE SET-UP HERE.
* BUFFER SPACE FOR THE FOLLOWING FOUR TAPE IS ALLOTED HERE,
* MINTAP = SYSTEM INPUT TAPE
* MLSTAP = SYSTEM OUTPUT TAPE
* SYSBIN = SYSTEM (BINARY) LOAD TAPE
* SYSU1 = SYSTEM SCRATCH TAPE NO. 1
* EACH OF THESE TAPES IS ASSIGNED TWO BUFFERS.

SYSTP1 EQU 1  SYSTEM TAPE
MINTAP EQU 5  SYSTEM INPUT TAPE
MLSTAP PZE 9  SYSTEM OUTPUT TAPE
SYSBIN PZE 9  SYSTEM (BINARY) LOAD TAPE
SYSU1 EQU 2  SYSTEM UTILITY TAPE NO. 1
SPACE 3

BUFMIT BSS 604  MINTAP- 2 BUFFERS OF 302 WORDS EACH
REM (MAX. ALLOWABLE REC. SIZE AT
REM 1401 = 300 WORDS)
REM
BUFMIT BSS 604  MLSTAP- 2 BUFFERS OF 302 WORDS EACH
REM (MAX. ALLOWABLE REC. SIZE AT
REM 1401 = 300 WORDS)
REM
BUFBIN BSS 492  SYSBIN- 2 BUFFERS OF 246 WORDS EACH
REM (8 BINARY CARDS/BLOCK)
REM
BUFU1 BSS 358  SYSU1- 2 BUFFERS OF 179 WORDS EACH
REM (CIT BLOCK (150) + CITFLG
REM BLOCK (25) = 175)
REM
* A COUPLE OF FMS II MONITOR CELLS ...
**TTL** CELLS **REQUIRE**ING **INITIALIZATION**

* REM SPACE 9

* REM FASTRAN COMPILER

* REM RESEARCH COMPUTING CENTER

* REM INDIANA UNIVERSITY

* REM BLOOMINGTON, INDIANA

* SPACE 3

* REM **CELLS** **REQUIRE**ING **INITIALIZATION**

* REM

* FT144910
* FT144920
* FT144930
* FT144940
* FT144950
* FT144960
* FT144970
* FT144980
* FT144990
EJECT

* ALL CELLS LYING BETWEEN TRIGR1 AND TRIGR2 ARE ZEROED OUT

* IN A LOOP IN THE INITIALIZATION ROUTINE.

TRIGR1 EQU * DEMARCATE START OF INITIALIZATION BLOCK
STOUT BSS 1
GTSWCH BSS 1
NOARGS BSS 1
DTRMF BSS 1
DOPNTR BSS 1
DDEFLG BSS 1
FIRSTF BSS 1
MODERR BSS 1
ENFILF BSS 1
LSTSYM BSS 1
LPASS1 BSS 1
PRIQ BSS 1

REM

SBTTL BSS 1
SBNAME BSS 1
SUBR EQU SBNAME
SNAM EQU SBNAME

REM

COMCNT BSS 1
NOCODE BSS 1
TRAFLG BSS 1
ERRERE BSS 1
WKCEL BSS 1
PLCOUNT BSS 1
PGCTR BSS 1
LABEL BSS 1
MODFLG BSS 1

REM

* THE FOLLOWING 'LENGTH' CELLS MUST RETAIN THEIR RELATIVE ORDER.

LIVECT BSS 1
LPOLG BSS 1
LARSTS BSS 1
LTEXB BSS 1
LCONST BSS 1
LCONT EQU LCONST

REM

LSTRIN BSS 1
LWORKS BSS 1
LVARBS BSS 1
LCOMM BSS 1

REM

CITBL6 BSS 1
CITBL7 BSS 1
CITSWH BSS 1
CITSW1 BSS 1
CITSW2 BSS 1
STSW1 BSS 1
TVSW1 BSS 1
TVSW2 BSS 1
TLSW BSS 1
OVERBG BSS 1
XRFLAG BSS 1
NOXRS BSS 1

REM

9LEFT BSS 24

Binary Output Image
SUM EQU 9LEFT+1 FT145600
BLEFT EQU 9LEFT+2 FT145610
BRIGHT EQU 9LEFT+3 FT145620
REM FT145630

IFFLAG BSS 1 ON IF AN ARRAY NAMED 'IF' HAS BEEN FOUND FT145640
FORFLG BSS 1 ON IF ARRAY NAMED 'FORMAT' HAS BEEN FOUND FT145650
GPOINT BSS 1 COMMON POSITION POINTER. DECREMENT INTEGER FT145660
SCAN14 BSS 1 FT145670
ASNFLG BSS 1 POINTS TO FIRST ASSIGN STACK FT145680
ISTKFG BSS 1 TELLS BEGINNING AND END OF ISTACK ENTRIES FT145690
TERMFG BSS 1 CURRENT BOTTOM OF EQUIVALENCE STACKS FT145700
DUBLMOD BSS 1 INDICATES EITHER DOUBLE OR COMPLEX STATE FT145710
FLAG1 BSS 1 ON AFTER CONSTANT HAS BEEN PUT IN STORAGE FT145720
FLAG2 BSS 1 ON IF 0377777 HAS BEEN PUT INTO STORAGE FT145730
REM FT145740
* THE ORDER OF THE FOLLOWING SIX CELLS IS CRITICAL -- USED BY M A P FT145750

GARB20 BSS 1 DIMENSIONED, COMMON FT145760
GARB19 BSS 1 NON-DIMENSIONED, COMMON FT145770
GARB18 BSS 1 DIMENSIONED, NON-COMMON FT145780
GARB17 BSS 1 NON-DIMENSIONED, NON-COMMON FT145790
GARB16 BSS 1 STRINGS FT145800
GARB15 BSS 1 LABELS FT145810
REM FT145820

GETRLF BSS 1 FLAG CELL GOVERNING READING OF NEXT CARD FT145830
TRIGR2 EQU * DEHARCATE END OF INITIALIZATION BLOCK FT145840
<table>
<thead>
<tr>
<th>TTL</th>
<th>MISCELLANEOUS WORKING CELLS</th>
<th>FTI145850</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMBRKB</td>
<td>BSS 1</td>
<td>LOCATION - 1 OF LOWEST COMMON USED</td>
</tr>
<tr>
<td>PDMCSW</td>
<td>BSS 1</td>
<td>VARIABLE PROCESSOR SWITCH</td>
</tr>
<tr>
<td>ENTRY</td>
<td>BSS 1</td>
<td></td>
</tr>
<tr>
<td>FNCFFL</td>
<td>BSS 1</td>
<td>ON FOR FUNCTION SUBPROGRAM</td>
</tr>
<tr>
<td>RNFNLG</td>
<td>BSS 1</td>
<td>NUMBER OF RETURN STATEMENTS IN PROGRAM</td>
</tr>
<tr>
<td>PNFNLG</td>
<td>BSS 1</td>
<td>PARENT CURRENT PAREN TYPE</td>
</tr>
<tr>
<td>GXFLG</td>
<td>BSS 1</td>
<td>INDICATES POSSIBLE COMPLEX CURRENT SITUATION</td>
</tr>
<tr>
<td>PCNT</td>
<td>BSS 1</td>
<td>PARENT LEVEL COUNT CELL</td>
</tr>
<tr>
<td>SCAN3B</td>
<td>BSS 1</td>
<td>SCRATCH FOR THE ACANNER</td>
</tr>
<tr>
<td>I1KAN2</td>
<td>BSS 1</td>
<td>DATA CELL FOR THE TRANSFER-TYPE PROCESSORS</td>
</tr>
<tr>
<td>ADDIN</td>
<td>BSS 1</td>
<td>HOLDS CONSTANT ADDEND AT STASH TIME</td>
</tr>
<tr>
<td>NXTLOC</td>
<td>BSS 1</td>
<td>NEXT AVAILABLE LOCATION IN POOL</td>
</tr>
<tr>
<td>DGPOINT</td>
<td>EQU NXTLOC</td>
<td></td>
</tr>
<tr>
<td>EPPOINT</td>
<td>EQU NXTLOC</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>BSS 2</td>
<td>THESE CELLS HOLD EQUIVALENCES OF OPERANDS AT CODE GENERATION TIME.</td>
</tr>
<tr>
<td>B</td>
<td>BSS 2</td>
<td>HOLDS EQUIV FOR LEFT-HAND FACTOR</td>
</tr>
<tr>
<td>CUL73</td>
<td>BSS 3</td>
<td>SECOND WORD HOLDS ASTACK LOCATION</td>
</tr>
<tr>
<td>SCSUM</td>
<td>BSS 1</td>
<td></td>
</tr>
<tr>
<td>CITB15</td>
<td>BSS 1</td>
<td>TEMPORARY CELL</td>
</tr>
<tr>
<td>SAVIND</td>
<td>BSS 1</td>
<td></td>
</tr>
<tr>
<td>CITB15</td>
<td>BSS 1</td>
<td>BUMP P COUNT ONLY IF THIS CELL IS ZERO</td>
</tr>
<tr>
<td>STSW2</td>
<td>BSS 1</td>
<td></td>
</tr>
<tr>
<td>LIVPL</td>
<td>BSS 1</td>
<td>LIVECT + LPROLG</td>
</tr>
<tr>
<td>LIVPRT</td>
<td>BSS 1</td>
<td></td>
</tr>
<tr>
<td>LIVPTS</td>
<td>EQU LIVPRT</td>
<td></td>
</tr>
<tr>
<td>LIVSCB</td>
<td>BSS 1</td>
<td></td>
</tr>
<tr>
<td>LTHUS</td>
<td>BSS 1</td>
<td></td>
</tr>
<tr>
<td>LHRWU</td>
<td>BSS 1</td>
<td></td>
</tr>
<tr>
<td>PGMBRK</td>
<td>BSS 1</td>
<td></td>
</tr>
<tr>
<td>IVARBS</td>
<td>BSS 1</td>
<td></td>
</tr>
<tr>
<td>ERASE</td>
<td>BSS 1</td>
<td></td>
</tr>
<tr>
<td>GETELA</td>
<td>EQU ERASE</td>
<td></td>
</tr>
<tr>
<td>ERASE1</td>
<td>BSS 1</td>
<td></td>
</tr>
<tr>
<td>ERASE2</td>
<td>BSS 1</td>
<td></td>
</tr>
<tr>
<td>ERASE3</td>
<td>BSS 1</td>
<td></td>
</tr>
<tr>
<td>ERASE4</td>
<td>BSS 1</td>
<td>LENGTH OF BODY OF PROLOGUE</td>
</tr>
<tr>
<td>LPBODY</td>
<td>BSS 1</td>
<td></td>
</tr>
<tr>
<td>LOGEND</td>
<td>BSS 1</td>
<td>THIS CELL SET TO PCOUNT WHEN END STATEMENT</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>IS ENCLOSED</td>
</tr>
<tr>
<td>RISMOD</td>
<td>BSS 1</td>
<td>HOLDS MODE OF FUNCTION AS DEFINED BY COLUMN</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>1 ON RETURN STATEMENTS</td>
</tr>
<tr>
<td>PEND10</td>
<td>BSS 1</td>
<td>ON WHEN LAST PREVIOUS EXECUTABLE STATEMENT WAS A CALL</td>
</tr>
<tr>
<td>RASCAL</td>
<td>BSS 1</td>
<td></td>
</tr>
<tr>
<td>DURG2</td>
<td>BSS 1</td>
<td></td>
</tr>
<tr>
<td>GET18</td>
<td>BSS 1</td>
<td>TITLE CARD LIST FLAG CELL</td>
</tr>
<tr>
<td>IOLT11</td>
<td>BSS 1</td>
<td></td>
</tr>
<tr>
<td>IOLT12</td>
<td>BSS 1</td>
<td></td>
</tr>
<tr>
<td>IOLT13</td>
<td>BSS 1</td>
<td></td>
</tr>
<tr>
<td>IOLT14</td>
<td>BSS 1</td>
<td></td>
</tr>
<tr>
<td>IOLT15</td>
<td>BSS 1</td>
<td></td>
</tr>
<tr>
<td>IOLT16</td>
<td>BSS 1</td>
<td></td>
</tr>
<tr>
<td>IOLT17</td>
<td>BSS 1</td>
<td></td>
</tr>
<tr>
<td>IOLT18</td>
<td>BSS 1</td>
<td></td>
</tr>
<tr>
<td>IOLT19</td>
<td>BSS 1</td>
<td></td>
</tr>
<tr>
<td>IOLT20</td>
<td>BSS 1</td>
<td></td>
</tr>
<tr>
<td>IOLT21</td>
<td>BSS 1</td>
<td></td>
</tr>
<tr>
<td>IOLT22</td>
<td>BSS 1</td>
<td></td>
</tr>
<tr>
<td>IOLT23</td>
<td>BSS 1</td>
<td></td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
<td>Address</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>CURPTR BSS</td>
<td>3</td>
<td>ASTACK POINTERS FOR EACH LEVEL</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DLVL BSS</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>NODDAR BSS</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>APTR BSS</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>PPRTR BSS</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>SAVE BSS</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>MIFLAG BSS</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>M2FLAG BSS</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>M1VAR BSS</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>M2VAR BSS</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>M3VAR BSS</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>DUTrM3 BSS</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>DVSTFG BSS</td>
<td>1</td>
<td>USED BY ROUTINES STZST and STZTSD</td>
</tr>
<tr>
<td>GEN6 BSS</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>REG1 BSS</td>
<td>2</td>
<td>HOLDS REGISTER LOCATION OF RESULT OF A CODE GENERATION.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REGFLG BSS</td>
<td>1</td>
<td>SECOND WORD ADDRESS GETS ASTACK POINTER.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AUXFLG BSS</td>
<td>1</td>
<td>INDICATES IF 'PLACE' OR 'STORE' ROUTINE</td>
</tr>
<tr>
<td>AUXF G1 BSS</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>GUD4 BSS</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>GUD6 BSS</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>GUD31 BSS</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>SPCDFG BSS</td>
<td>1</td>
<td>DP AND CA ROUTING FLAG IN CODE ROUTINE</td>
</tr>
<tr>
<td>LOCAT1 BSS</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>SYMST1 BSS</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>STOLD1 BSS</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>DPTRW7 BSS</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>SCANHT BSS</td>
<td>1</td>
<td>TEMPORARY STORAGE FOR MQ</td>
</tr>
<tr>
<td>HFLAG BSS</td>
<td>1</td>
<td>MUST BE ZERODED AT BEGINNING OF EACH STATE.</td>
</tr>
<tr>
<td>STOSAV BSS</td>
<td>1</td>
<td>TEMPORARY CELL USED BY STORE ROUTINE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*</td>
<td></td>
<td>CELLS USED BY THE SCAN ROUTINE</td>
</tr>
<tr>
<td>SYMBOL BSS</td>
<td>4</td>
<td>SCRATCH LOCATIONS FOR SCANNER</td>
</tr>
<tr>
<td>GETR4 BSS</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>EXPON BSS</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>INTG BSS</td>
<td>1</td>
<td>HOLDS INTEGER PART OF A CONSTANT</td>
</tr>
<tr>
<td>FRAG BSS</td>
<td>2</td>
<td>HOLDS HIGH + LOW PARTS OF FLOATING CONSTANT</td>
</tr>
<tr>
<td>EXPONA BSS</td>
<td>1</td>
<td>ADJUSTIVE EXPONENT IF LEADING ZEROS</td>
</tr>
<tr>
<td>CHAR BSS</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>DECP BSS</td>
<td>1</td>
<td>INDICATES DECIMAL POINT IN CONSTANT</td>
</tr>
<tr>
<td>CX1 BSS</td>
<td>1</td>
<td>USED TO HOLD FIRST PART OF COMPLEX CONSTANT</td>
</tr>
<tr>
<td>CX2 BSS</td>
<td>1</td>
<td>HOLDS SECOND PART OF COMPLEX CONSTANT</td>
</tr>
<tr>
<td>ASNLF BSS</td>
<td>1</td>
<td>START OF LAST BUILT ASSIGN STACK IN POOL</td>
</tr>
<tr>
<td>SETUP2 BSS</td>
<td>1</td>
<td>CONTAINS 'NAME' OF CURRENT ROUTINE</td>
</tr>
<tr>
<td>SAVCL2 BSS</td>
<td>1</td>
<td>HOLDS WORKING CELL POINTER FOR CELL 2 SAVE</td>
</tr>
<tr>
<td>SCPEL BSS</td>
<td>1</td>
<td>HOLDS EQUIV (POOL) POINTER FOR START OF CALLING QUEUE</td>
</tr>
<tr>
<td>ARGcnt BSS</td>
<td>1</td>
<td>NUMBER OF ARGUMENTS IN CALLING SEQUENCE</td>
</tr>
<tr>
<td>STUCG BSS</td>
<td>1</td>
<td>HOLDS CURRENT HI-CORE ARGUMENT LOCATION</td>
</tr>
<tr>
<td>CMPS1 BSS</td>
<td>1</td>
<td>WORKING STORAGE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>*</td>
<td></td>
<td>CELLS USED BY THE EQUIVALENC E PROCESSOR</td>
</tr>
<tr>
<td>INVTFG BSS</td>
<td>1</td>
<td>BEGINNING OF EQUTBL DIRECTOR STACK</td>
</tr>
<tr>
<td>PRIOR BSS</td>
<td>1</td>
<td>ON IF NEW STACK HAS BEEN STARTED</td>
</tr>
<tr>
<td>LINKER BSS</td>
<td>1</td>
<td>ON IF LINK TO OLD STACK HAS BEEN FOUND</td>
</tr>
<tr>
<td>Symbol</td>
<td>Type</td>
<td>Value</td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
<td>-------</td>
</tr>
<tr>
<td>HI</td>
<td>BSS</td>
<td>1</td>
</tr>
<tr>
<td>LO</td>
<td>BSS</td>
<td>1</td>
</tr>
<tr>
<td>TOP</td>
<td>BSS</td>
<td>1</td>
</tr>
<tr>
<td>DEL</td>
<td>BSS</td>
<td>1</td>
</tr>
<tr>
<td>DEL1</td>
<td>BSS</td>
<td>1</td>
</tr>
<tr>
<td>T3</td>
<td>REM</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STT</td>
<td>EQU</td>
<td>*</td>
</tr>
<tr>
<td>BSS</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>CMLT</td>
<td>EQU</td>
<td>*-1</td>
</tr>
<tr>
<td>BSS</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>CADD</td>
<td>EQU</td>
<td>*-1</td>
</tr>
<tr>
<td>BSS</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>VARB</td>
<td>EQU</td>
<td>*-1</td>
</tr>
<tr>
<td>BSS</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>WIND</td>
<td>EQU</td>
<td>*-1</td>
</tr>
<tr>
<td>DIX1</td>
<td>BSS</td>
<td>1</td>
</tr>
<tr>
<td>DIX2</td>
<td>BSS</td>
<td>1</td>
</tr>
<tr>
<td>DIXN</td>
<td>BSS</td>
<td>1</td>
</tr>
<tr>
<td>ASIGN</td>
<td>BSS</td>
<td>1</td>
</tr>
<tr>
<td>SAV1</td>
<td>BSS</td>
<td>1</td>
</tr>
<tr>
<td>SAV2</td>
<td>BSS</td>
<td>1</td>
</tr>
<tr>
<td>SAV3</td>
<td>BSS</td>
<td>1</td>
</tr>
<tr>
<td>TMPX</td>
<td>BSS</td>
<td>1</td>
</tr>
<tr>
<td>CONADD</td>
<td>BSS</td>
<td>1</td>
</tr>
<tr>
<td>EINT</td>
<td>EQU</td>
<td>*</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EEXITG</td>
<td>BSS</td>
<td>1</td>
</tr>
<tr>
<td>CALLFG</td>
<td>BSS</td>
<td>1</td>
</tr>
<tr>
<td>SUBFGLG</td>
<td>BSS</td>
<td>1</td>
</tr>
<tr>
<td>ARITFN</td>
<td>BSS</td>
<td>1</td>
</tr>
<tr>
<td>ARFLAG</td>
<td>BSS</td>
<td>1</td>
</tr>
<tr>
<td>EUFLG</td>
<td>BSS</td>
<td>1</td>
</tr>
<tr>
<td>SYMTS</td>
<td>BSS</td>
<td>1</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDOSTK</td>
<td>EQU</td>
<td>75</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOSTAK</td>
<td>BSS</td>
<td>LDOSTK</td>
</tr>
<tr>
<td>LASTAK</td>
<td>EQU</td>
<td>200</td>
</tr>
<tr>
<td>BSS</td>
<td>LASTAK</td>
<td></td>
</tr>
<tr>
<td>ASTACK</td>
<td>BSS</td>
<td>1</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>WARS IN CORE)</td>
</tr>
<tr>
<td>BSS</td>
<td>1</td>
<td>ASTACK+1 MUST BE ALL ONES FOR PROPER</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>FUNCTIONING OF THE COMPIL ROUTINE.</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>INITIALIZATION OF THE COMPILER SHOULD SET</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>THE CELL.</td>
</tr>
<tr>
<td>BSS</td>
<td>1</td>
<td>PAREN STACK</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PRLNGH</td>
<td>EQU</td>
<td>180</td>
</tr>
<tr>
<td>REM</td>
<td></td>
<td>PROLOGUE TABLE</td>
</tr>
<tr>
<td>PRLTBL</td>
<td>BSS</td>
<td>PRLNGH</td>
</tr>
<tr>
<td>CARD</td>
<td>BSS</td>
<td>115</td>
</tr>
<tr>
<td>--------</td>
<td>-----</td>
<td>-----</td>
</tr>
<tr>
<td>UNLBL</td>
<td>EQU</td>
<td>CAR</td>
</tr>
<tr>
<td>UNSTR</td>
<td>EQU</td>
<td>UNBL</td>
</tr>
<tr>
<td>UNVAR</td>
<td>EQU</td>
<td>UNSTR</td>
</tr>
<tr>
<td>NOPTH</td>
<td>EQU</td>
<td>UNVAR</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td></td>
</tr>
<tr>
<td>CITLNG</td>
<td>EQU</td>
<td>150</td>
</tr>
<tr>
<td>CIT1</td>
<td>BSS</td>
<td>CITLNG</td>
</tr>
<tr>
<td>CIT2</td>
<td>BSS</td>
<td>CITLNG</td>
</tr>
<tr>
<td>CITFLG1</td>
<td>BSS</td>
<td>CITLNG/6</td>
</tr>
<tr>
<td>CITFLG2</td>
<td>BSS</td>
<td>CITLNG/6</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td></td>
</tr>
<tr>
<td>(PATCH</td>
<td>EQU</td>
<td>*</td>
</tr>
<tr>
<td>NOCSIT</td>
<td>EQU</td>
<td>300</td>
</tr>
<tr>
<td>BSS</td>
<td>NOCSIT</td>
<td></td>
</tr>
<tr>
<td>CONTAB</td>
<td>EQU</td>
<td>*-1</td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>REM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>P</td>
<td>10</td>
</tr>
<tr>
<td>COMFLG</td>
<td>EQU</td>
<td>P</td>
</tr>
<tr>
<td>MANGMX</td>
<td>EQU</td>
<td>P+1</td>
</tr>
<tr>
<td>DELMAX</td>
<td>EQU</td>
<td>P+2</td>
</tr>
<tr>
<td>CURRFG</td>
<td>EQU</td>
<td>P+3</td>
</tr>
<tr>
<td>CICELL</td>
<td>EQU</td>
<td>P+4</td>
</tr>
<tr>
<td>IDFLAG</td>
<td>EQU</td>
<td>P+6</td>
</tr>
<tr>
<td>DEFINE</td>
<td>EQU</td>
<td>P+7</td>
</tr>
<tr>
<td>ORIGIN</td>
<td>EQU</td>
<td>P+8</td>
</tr>
<tr>
<td>MONTOR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAXNLC</td>
<td>PZE</td>
<td></td>
</tr>
<tr>
<td>LASTLA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STW1</td>
<td>OCT</td>
<td>600526000000</td>
</tr>
<tr>
<td>Symbol Table Mode Configurations</td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MLABL   BOOL 000000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSTRG   BOOL 100000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MINTG   BOOL 200000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MREAL   BOOL 300000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148170</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148210</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148220</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148230</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flags for Equiv Entry in the Symbol Table Scheme</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BINTF   BOOL 10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148270</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148280</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BEXTF   BOOL 20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148290</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BARRY   BOOL 40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148300</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BASSN   BOOL 100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148310</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148320</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLIBF   BOOL 10C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148330</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148340</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BARGT   BOOL 200</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148350</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148360</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCOMMN  BOOL 400</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148370</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148380</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REGUW   BOOL 1000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148390</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148400</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148410</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148420</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BPATH   BOOL 40000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148430</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148440</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148450</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148460</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flags for Asstack Form of Equiv Word</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BCNST   BOOL 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148470</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148480</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148490</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BOPRX   BOOL 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148500</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148510</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BARIT   BOOL 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148520</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148530</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BOPNF   BOOL 100</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148540</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148550</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148560</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148570</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148580</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INDEXING INFORMATION FLAGS FOR EQUIV</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BINDX   BOOL 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148590</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148600</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BIXXR   BOOL 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148610</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148620</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148630</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148640</td>
<td></td>
<td></td>
</tr>
<tr>
<td>60DIX   BOOL 4000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148650</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148660</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148670</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148680</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TASSN   BOOL 100000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148690</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148700</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TGO TO  BOOL 200000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148710</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148720</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148730</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N      EQU 0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148740</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REM     FTI148750</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P</td>
<td>EQU</td>
<td>1</td>
</tr>
<tr>
<td>---</td>
<td>-----</td>
<td>---</td>
</tr>
<tr>
<td>R</td>
<td>EQU</td>
<td>2</td>
</tr>
<tr>
<td>W</td>
<td>EQU</td>
<td>3</td>
</tr>
<tr>
<td>T</td>
<td>EQU</td>
<td>4</td>
</tr>
<tr>
<td>F</td>
<td>EQU</td>
<td>5</td>
</tr>
<tr>
<td>C</td>
<td>EQU</td>
<td>6</td>
</tr>
<tr>
<td>S</td>
<td>EQU</td>
<td>7</td>
</tr>
<tr>
<td>X</td>
<td>EQU</td>
<td>8</td>
</tr>
<tr>
<td>M</td>
<td>EQU</td>
<td>9</td>
</tr>
<tr>
<td>E</td>
<td>EQU</td>
<td>10</td>
</tr>
<tr>
<td>Q</td>
<td>EQU</td>
<td>11</td>
</tr>
<tr>
<td>D</td>
<td>EQU</td>
<td>12</td>
</tr>
<tr>
<td>G</td>
<td>EQU</td>
<td>13</td>
</tr>
<tr>
<td>H</td>
<td>EQU</td>
<td>14</td>
</tr>
<tr>
<td>I</td>
<td>EQU</td>
<td>15</td>
</tr>
<tr>
<td>J</td>
<td>EQU</td>
<td>16</td>
</tr>
<tr>
<td>END</td>
<td>BOOL</td>
<td>77</td>
</tr>
</tbody>
</table>

---

**BCD Character Equivalents**

<table>
<thead>
<tr>
<th>BOOL</th>
<th>EQUCHAR</th>
<th>IN OCTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>EQUCHAR</td>
<td>13</td>
</tr>
<tr>
<td>26</td>
<td>FCHAR</td>
<td>26</td>
</tr>
<tr>
<td>34</td>
<td>RPCHAR</td>
<td>34</td>
</tr>
<tr>
<td>Label</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>CMCHAR</td>
<td>73</td>
<td>COMMA IN OCTAL</td>
</tr>
<tr>
<td>LPCHAR</td>
<td>74</td>
<td>( IN OCTAL</td>
</tr>
<tr>
<td>EOS</td>
<td>77</td>
<td>END-OF-STATEMENT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REM</td>
</tr>
<tr>
<td>PROCSW</td>
<td></td>
<td>(PROCESSOR FLAG WORD FORMED BY SPRSW MACRO) BITS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BXEQ</td>
<td>400000</td>
<td></td>
</tr>
<tr>
<td>BIRANS</td>
<td>200000</td>
<td></td>
</tr>
<tr>
<td>BDD</td>
<td>100000</td>
<td></td>
</tr>
<tr>
<td>RCALL</td>
<td>40000</td>
<td>PROCSSW FLAG INDICATING CALL STATEMENT</td>
</tr>
<tr>
<td>BEND</td>
<td>20000</td>
<td>PROCSSW FLAG INDICATING END STATEMENT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AEQ</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NOXEQ</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IXANS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NOTRA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>D0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NODD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SPACE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>DEG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BINARY</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>VS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MQ</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PAC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PMQ</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REM</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>INTEGR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REAL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AUXIL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TOPCOCR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>STCC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PROBOX</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EQU</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TOPCOR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OVERFLOW</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TOP OF CORE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OVERFLOW CELL (COMMON+1)</td>
</tr>
</tbody>
</table>
POOL, EQUIV, SYMTAB

REM
SPACE 9
* * * * * * * * * *
REM FASTRAN COMPILER
REM RESEARCH COMPUTING CENTER
REM INDIANA UNIVERSITY
REM BLOOMINGTON, INDIANA
SPACE 3
* * * * * * * * * *
REM POOL, EQUIV, SYMTAB
REM
* * * * * * * * * *
<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EJECT</td>
<td>FT150060</td>
<td>Page 391</td>
</tr>
<tr>
<td>LSYMTH</td>
<td>4096</td>
<td>LENGTH OF THE SYMBOL TABLE</td>
</tr>
<tr>
<td>SYMTAB</td>
<td>FT150070</td>
<td>ORIGIN OF SYMBOL TABLE</td>
</tr>
<tr>
<td>REM</td>
<td>FT150080</td>
<td></td>
</tr>
<tr>
<td>EQUIV</td>
<td>FT150090</td>
<td>ORIGIN OR EQUIVALENT TABLE</td>
</tr>
<tr>
<td>REM</td>
<td>FT150100</td>
<td></td>
</tr>
<tr>
<td>POOL</td>
<td>FT150110</td>
<td>ORIGIN OF STORAGE POOL</td>
</tr>
<tr>
<td>EQUIV</td>
<td>FT150120</td>
<td></td>
</tr>
<tr>
<td>POOL</td>
<td>FT150130</td>
<td></td>
</tr>
<tr>
<td>EQUIV</td>
<td>FT150140</td>
<td></td>
</tr>
<tr>
<td>POOL</td>
<td>FT150140</td>
<td></td>
</tr>
<tr>
<td>POOL</td>
<td>FT150140</td>
<td></td>
</tr>
</tbody>
</table>
PLEADS FOR...

* TO ADVANTAGEOUSLY UTILIZE SPACE, THE BSS ARRAY 'COLUMN'
* (LENGTH LCOLUMN=666) IS PLACED ABOVE THE LITERALS AND BELOW
* THE LAST LITERAL. TO AVOID ClobberING A NECESSARY LITERAL, AND
* TO AVOID HAVING TO REFER TO THE COLUMN REGION BY 'COLUMN+1',
* THE HIGHEST POSSIBLE LITERAL (077777777777) IS NOW GENERATED.
* THIS IS JUST A DUMMY USAGE TO PUT A SCRATCH CELL AT THE LOCATION
* CALLED 'COLUMN'.

** ANY ACTUAL REFERENCES TO THE BIT CONFIGURATION **

** OCTAL 77777777777 MUST BE MADE BY USING THE CELL 'ALLSVN'. **

** DO NOT USE THE LITERAL. ******

REM
* THERE ARE TWO PURPOSES FOR THIS MANEUVER. FIRST, SINCE POOL
* OVERFLOW IS TESTED FOR ONLY AT THE END OF EACH STATEMENT,
* THE POOL MAY OVERFLOW SIGNIFICANTLY BEFORE THE TEST IS MADE.
* THIS ARRAY COLUMN PROVIDES A BUFFER TO KEEP FROM DESTROYING
* THE LITERALS (AND THUS NECESSITATING THE RESTORING OF THE
* COMPILER).
* SECOND, ALL SECTIONS OF THE COMPILER AFTER PASS 1 CAN MAKE USE OF
* AN EXPANDED POOL REGION WHICH INCLUDES THE 666 COLUMN CELLS.

PSE =07777777777 FORCE OUT THE ALL-BIT LITERAL
ORG *-1 RE-USE THE DUMMY CELL

LCOLUMN EQU 666 LENGTH OF THE COLUMN ARRAY
REM
COLUMN END BEGINX END CARD AND DEFINITION OF COLUMN ARRAY
* END TAPE