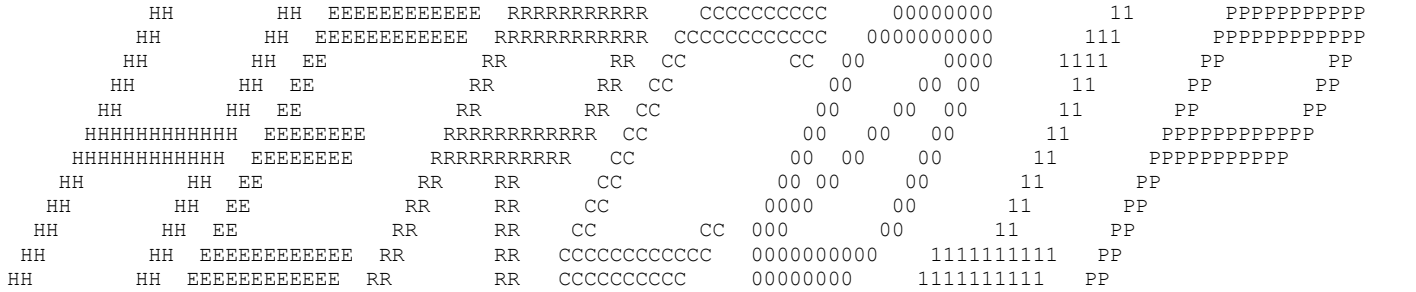


****A START JOB 498 HERC01P ROOM 10.14.44 AM 10 SEP 17 PRINTER1 SYS BSP1 JOB 498 START A****
****A START JOB 498 HERC01P ROOM 10.14.44 AM 10 SEP 17 PRINTER1 SYS BSP1 JOB 498 START A****
****A START JOB 498 HERC01P ROOM 10.14.44 AM 10 SEP 17 PRINTER1 SYS BSP1 JOB 498 START A****
****A START JOB 498 HERC01P ROOM 10.14.44 AM 10 SEP 17 PRINTER1 SYS BSP1 JOB 498 START A****

```
10.14.42 JOB  498  IEF677I WARNING MESSAGE(S) FOR JOB HERC01P  ISSUED
10.14.42 JOB  498  $HASP373 HERC01P  STARTED - INIT  4 - CLASS S - SYS BSP1
10.14.42 JOB  498  IEF403I HERC01P - STARTED - TIME=10.14.42
10.14.42 JOB  498  IEFACTRT - Stepname  Procstep  Program   Retcode
10.14.42 JOB  498  HERC01P    S1                 IDCAMS    RC= 0000
10.14.44 JOB  498  HERC01P    P1        PL1L     IEMAA     RC= 0004
10.14.44 JOB  498  HERC01P    P1        LKED     IEWL      RC= 0000
10.14.44 JOB  498  IEF404I HERC01P - ENDED - TIME=10.14.44
10.14.44 JOB  498  $HASP395 HERC01P  ENDED


------ JES2 JOB STATISTICS ------


 10 SEP 17 JOB EXECUTION DATE


        23 CARDS READ


    12,567 SYSOUT PRINT RECORDS


         0 SYSOUT PUNCH RECORDS


       0.03 MINUTES EXECUTION TIME
```

```
1     //HERC01P JOB MSGCLASS=A,MSGLEVEL=(1,1),CLASS=S                          JOB  498
      ***                                  CLASS A USES MY PROCLIBS        00020000
      ***                                  CLASS S USES DEFAULT PROCLIBS   00030000
      ***                                                                  00040000
      ***   COMPILE AND LINK BASIC360                                      00050000
      ***                                                                  00060000
2     //S1  EXEC  PGM=IDCAMS                                               00070000
3     //SYSPRINT DD SYSOUT=*                                               00080000
4     //SYSIN    DD *                                                      00090000
      ***                                                                  00130000
5     //P1  EXEC PL1LFCL,REGION.PL1L=1024K,                                00140000
      //  PARM.PL1L='LOAD,NODECK,EXTDIC,MACRO,ATR,NEST,XREF,SORMGIN=(2,72,1)' 00150000
6     XXPL1L   EXEC  PGM=IEMAA,PARM='LOAD,NODECK',REGION=52K               05000001
7     XXSYSPRINT DD  SYSOUT=A                                              10000001
8     XXSYSLIN   DD  DSNAME=&&LOADSET,DISP=(MOD,PASS),UNIT=SYSSQ,         *15000001
      XX             SPACE=(80,(250,100))                                  20000001
9     //PL1L.SYSUT3 DD UNIT=SYSDA,SPACE=(80,(3500,1000))                   00160000
      X/SYSUT3  DD  DSNAME=&&SYSUT3,UNIT=SYSDA,SPACE=(80,(250,250)),       *23000019
      XX             DCB=BLKSIZE=80                                        26000019
10    XXSYSUT1   DD  DSNAME=&&SYSUT1,UNIT=SYSDA,SPACE=(1024,(60,60),,CONTIG),*30000001
      XX             SEP=(SYSUT3,SYSLIN),DCB=BLKSIZE=1024                  35000019
11    //PL1L.SYSLIB DD DSN=HERC01.BASIC360.PLI,DISP=SHR                    00170000
12    //PL1L.SYSIN DD DSN=HERC01.BASIC360.PLI(BASIC360),DISP=SHR           00180000
13    XXLKED   EXEC  PGM=IEWL,PARM='XREF,LIST',COND=(9,LT,PL1L),           *40000001
      XX             REGION=96K                                            45000001
14    XXSYSLIB   DD  DSNAME=SYS1.PL1LIB,DISP=SHR                           50000001
15    //LKED.SYSLMOD DD DSN=HERC01.BASIC360.LOADLIB(BASIC360),             00190000
      //     DISP=(NEW,CATLG,CATLG),UNIT=SYSDA,VOL=SER=PUB000,             00200000
      //     SPACE=(TRK,(18,18,18)),                                       00210000
      //     DCB=(DSORG=PO,RECFM=U,BLKSIZE=19069)                          00220000
      ***                                                                  00230000
      X/SYSLMOD  DD  DSNAME=&&GOSET(GO),DISP=(MOD,PASS),                   *55000001
      XX             UNIT=SYSDA,SPACE=(1024,(50,20,1),RLSE)                60000001
16    XXSYSUT1   DD  DSNAME=&&SYSUT1,UNIT=SYSDA,SPACE=(1024,(200,20)),     *65000001
      XX             SEP=(SYSLMOD,SYSLIB),DCB=BLKSIZE=1024                 70000019
17    XXSYSPRINT DD  SYSOUT=A                                              75000001
18    XXSYSLIN   DD  DSNAME=&&LOADSET,DISP=(OLD,DELETE)                    80000001
19    XX         DD  DDNAME=SYSIN                                          85000001
```

```
 STMT NO. MESSAGE
-
   19     IEF686I DDNAME REFERRED TO ON DDNAME KEYWORD IN PRIOR STEP WAS NOT RESOLVED
IEF236I ALLOC. FOR HERC01P S1
IEF237I JES2 ALLOCATED TO SYSPRINT
IEF237I JES2 ALLOCATED TO SYSIN
IEF237I 240  ALLOCATED TO SYS00002
IEF237I 240  ALLOCATED TO SYS00001
IEF285I   HERC01.BASIC360.LOADLIB                      KEPT          *--------0
IEF285I   VOL SER NOS= PUB000.
IEF142I HERC01P S1 - STEP WAS EXECUTED - COND CODE 0000
IEF285I   JES2.JOB00498.SO0102                         SYSOUT
IEF285I   JES2.JOB00498.SI0101                         SYSIN
IEF285I   SYS1.UCAT.TSO                                KEPT          *--------0
IEF285I   VOL SER NOS= PUB000.
IEF373I STEP /S1      / START 17253.1014
IEF374I STEP /S1      / STOP  17253.1014 CPU    0MIN 00.02SEC SRB    0MIN 00.00SEC VIRT   192K SYS   224K
*************************************************************************************************************************
*     1. Jobstep of job: HERC01P     Stepname: S1          Program name: IDCAMS     Executed on 10.09.17 from 10.14.42 to 10.14.42 *
*         elapsed time  24:00:00,04                     CPU-Identifier:  BSP1           Page-in:      0                      *
*             CPU time  00:00:00,02               Virtual Storage used:    192K         Page-out:     0                      *
*           corr. CPU:  00:00:00,02   CPU time has been corrected by  1 / 1,0  multiplier                                   *
*                                                                                                                           *
*     I/O Operation                                                                                                         *
*     Number of records read via DD * or DD DATA:      3                                                                    *
*     DMY.......0 DMY.......0 240.......0                                                                                   *
*                                                                                                                           *
*                                        Charge for step (w/o SYSOUT):          0,03                                    *
*************************************************************************************************************************
IEF236I ALLOC. FOR HERC01P PL1L P1
IEF237I JES2 ALLOCATED TO SYSPRINT
IEF237I 14C  ALLOCATED TO SYSLIN
IEF237I 14B  ALLOCATED TO SYSUT3
IEF237I 149  ALLOCATED TO SYSUT1
IEF237I 240  ALLOCATED TO SYSLIB
IEF237I 240  ALLOCATED TO SYS00025
IEF237I 240  ALLOCATED TO SYSIN
IEF142I HERC01P PL1L P1 - STEP WAS EXECUTED - COND CODE 0004
IEF285I   JES2.JOB00498.SO0103                         SYSOUT
IEF285I   SYS17253.T101442.RA000.HERC01P.LOADSET       PASSED        *----1,750
IEF285I   VOL SER NOS= SMP004.
IEF285I   SYS17253.T101442.RA000.HERC01P.SYSUT3        DELETED       *----4,621
IEF285I   VOL SER NOS= SMP003.
IEF285I   SYS17253.T101442.RA000.HERC01P.SYSUT1        DELETED       *--------0
IEF285I   VOL SER NOS= SMP001.
IEF285I   HERC01.BASIC360.PLI                          KEPT          *-------86
IEF285I   VOL SER NOS= PUB000.
IEF285I   SYS1.UCAT.TSO                                KEPT          *--------0
IEF285I   VOL SER NOS= PUB000.
IEF285I   HERC01.BASIC360.PLI                          KEPT          *------814
IEF285I   VOL SER NOS= PUB000.
IEF373I STEP /PL1L    / START 17253.1014
IEF374I STEP /PL1L    / STOP  17253.1014 CPU    0MIN 01.25SEC SRB    0MIN 00.23SEC VIRT  1036K SYS   200K
*************************************************************************************************************************
*     2. Jobstep of job: HERC01P     Stepname: PL1L        Program name: IEMAA      Executed on 10.09.17 from 10.14.42 to 10.14.44 *
*         elapsed time  24:00:01,64                     CPU-Identifier:  BSP1           Page-in:      0                      *
*             CPU time  00:00:01,48               Virtual Storage used:   1036K         Page-out:     0                      *
*           corr. CPU:  00:00:01,48   CPU time has been corrected by  1 / 1,0  multiplier                                   *
*                                                                                                                           *
*     I/O Operation                                                                                                         *
*     Number of records read via DD * or DD DATA:      0                                                                    *
*     DMY.......0 14C....1750 14B....4621 149.......0 240......86 240.......0 240.....814                                   *
*                                                                                                                           *
*                                        Charge for step (w/o SYSOUT):          2,46                                    *
```

```
*********************************************************************************************************************************************
IEF236I ALLOC. FOR HERC01P LKED P1
IEF237I 148  ALLOCATED TO SYSLIB
IEF237I 240  ALLOCATED TO SYSLMOD
IEF237I 240  ALLOCATED TO SYS00026
IEF237I 170  ALLOCATED TO SYSUT1
IEF237I JES2 ALLOCATED TO SYSPRINT
IEF237I 14C  ALLOCATED TO SYSLIN
IEF237I DMY  ALLOCATED TO
IEF142I HERC01P LKED P1 - STEP WAS EXECUTED - COND CODE 0000
IEF285I   SYS1.PL1LIB                                KEPT          *------329
IEF285I   VOL SER NOS= MVSRES.
IEF285I   HERC01.BASIC360.LOADLIB                    CATALOGED     *------235
IEF285I   VOL SER NOS= PUB000.
IEF285I   SYS1.UCAT.TSO                              KEPT          *--------0
IEF285I   VOL SER NOS= PUB000.
IEF285I   SYS17253.T101442.RA000.HERC01P.SYSUT1      DELETED       *------208
IEF285I   VOL SER NOS= WORK01.
IEF285I   JES2.JOB00498.SO0104                       SYSOUT
IEF285I   SYS17253.T101442.RA000.HERC01P.LOADSET     DELETED       *----1,751
IEF285I   VOL SER NOS= SMP004.
IEF373I STEP /LKED    / START 17253.1014
IEF374I STEP /LKED    / STOP  17253.1014 CPU    0MIN 00.20SEC SRB    0MIN 00.07SEC VIRT    96K SYS   212K
*********************************************************************************************************************************************
*     3. Jobstep of job: HERC01P     Stepname: LKED        Program name: IEWL       Executed on 10.09.17 from 10.14.44 to 10.14.44 *
*         elapsed time  24:00:00,40                     CPU-Identifier:  BSP1           Page-in:      0                            *
*             CPU time  00:00:00,27               Virtual Storage used:     96K         Page-out:      0                            *
*           corr. CPU:  00:00:00,27   CPU time has been corrected by  1 / 1,0  multiplier                                          *
*                                                                                                                                   *
*     I/O Operation                                                                                                                 *
*     Number of records read via DD * or DD DATA:      0                                                                            *
*     148.....329 240.....235 240.......0 170.....208 DMY.......0 14C....1751 DMY.......0                                          *
*                                                                                                                                   *
*                                       Charge for step (w/o SYSOUT):          0,45                                               *
*********************************************************************************************************************************************
IEF375I  JOB /HERC01P / START 17253.1014
IEF376I  JOB /HERC01P / STOP  17253.1014 CPU    0MIN 01.47SEC SRB    0MIN 00.30SEC
```

```
   DELETE HERC01.BASIC360.LOADLIB NONVSAM PURGE                          00100000
IDC0550I ENTRY (A) HERC01.BASIC360.LOADLIB DELETED
IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

   SET LASTCC = 0                                                        00110000
   SET MAXCC = 0                                                         00120000

IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0
```

PL/I F COMPILER OPTIONS SPECIFIED ARE AS FOLLOWS--

```
     LOAD,NODECK,EXTDIC,MACRO,ATR,NEST,XREF,SORMGIN=(2,72,1)
```

THE COMPLETE LIST OF OPTIONS USED DURING THIS COMPILATION IS--

```
  EBCDIC
  CHAR60
  MACRO
  SOURCE2
NOMACDCK
  COMP
  SOURCE
  ATR
  XREF
NOEXTREF
NOLIST
  LOAD
NODECK
  FLAGW
NOSTMT
  SIZE=1049144
  LINECNT=050
  OPT=01
  SORMGIN=(002,072,001)
  EXTDIC
  NEST
  OPLIST
  SYNCHKT
```

```
*OPTIONS IN EFFECT*     EBCDIC,CHAR60,MACRO,SOURCE2,NOMACDCK,COMP,SOURCE,ATR,XREF,NOEXTREF,NOLIST,LOAD,
*OPTIONS IN EFFECT*     NODECK,FLAGW,NOSTMT,SIZE=1049144,LINECNT=050,OPT=01,SORMGIN=(002,072,001),EXTDIC,
*OPTIONS IN EFFECT*     NEST,OPLIST,SYNCHKT
```

COMPILE-TIME MACRO PROCESSOR
MACRO SOURCE2 LISTING

```
               /****** BASIC/360  V2.2  09/10/2017 *******/
               /****** BASIC/360  V2.1  08/22/2017 *******/
               /****** BASIC/360  V2.0  08/08/2016 *******/
   /*********************************************************************
   *                                                                   *
   *   SOUTH HAMMOND INSTITUTE OF TECHNOLOGY  BASIC/360   FALL 1974    *
   *                                                                   *
   *********************************************************************
   *                                                                   *
   *   IMPLEMENT A BASIC COMPILER/INTERPRETER FOR THE IBM/360          *
   *   USING THE ORIGINAL DARTMOUTH SPECS FOR BASIC.  THE PRIMARY      *
   *   INTENT IS TO CREATE A BASIC COMPILER/INTERPRETER FOR BEGINNING  *
   *   STUDENTS TO LEARN THE BASIC LANGUAGE INSTEAD OF GOTRAN ON THE   *
   *   SOON TO BE RETIRED 1620.                                        *
   *                                                                   *
   *   THE TARGET ENVIRONMENT IS A 32K IBM/360 MOD 30 RUNNING          *
   *   DOS/360 AND PL/I(D) COMPILER.                                   *
   *                                                                   *
   *   STUDENTS MAY NOT BE COMPUTER MAJORS AND MOST PROGRAMS WOULD BE  *
   *   SMALL, A SIMPLE MONITOR MONITOR WAS IMPLEMENTED SO THE LAB AID  *
   *   OR INSTRUCTOR COULD ACTUALLY SUBMIT ALL THE BASIC PROGRAMS AS   *
   *   ONE JOB.                                                        *
   *                                                                   *
   *   THIS PACKAGE IS BEING DESIGNED TO HAVE MODULAR SOURCE CODE      *
   *   SINCE IT ENVISIONED THAT THIS PRODUCT WILL BE IMPLEMENTED       *
   *   IN SEVERAL DIFFERENT ENVIRONMENTS                               *
   *      1) SIMPLE BATCH - 1 BASIC PROGRAM AT A TIME                  *
   *      2) MONITOR BATCH - MULTIPLE BASIC PROGRAMS CAN BE EXECUTED   *
   *                         PER RUN.                                  *
   *      3) ONLINE (WISH) - BASIC PROGRAM CAN BE ENTERED, EDITED AND  *
   *                         EXECUTED ON LINE.                         *
   *                                                                   *
   *********************************************************************/
   /*********************************************************************
   *                                                                   *
   *   BASIC/360 V2.2    DRAFT                                         *
   *                                                                   *
   *   V2.2 CHANGE LOG                                                 *
   *  -- FIXES:                                                        *
   *   - FIXED LINE OVERFLOW REPORTED BY MARCUS LOEW                   *
   *   - COSMETIC FIXES TO LISTING                                     *
   *  --ENHANCEMENTS:                                                  *
   *   - ADDED INR - INT WITH ROUNDING                                 *
   *   - ADDED STRING COMPARE TO IF STATEMENT                          *
```

MACRO SOURCE2 LISTING

```
 45    *   - ADDED STOP STATEMENT TO BE USED FOR ABNORMAL ENDING           *
 46    *   - ADDED SUBSCRIPT CHECKING TO PREVENT PROTECTION EXCEPTIONS     *
 47    *                                                                   *
 48    *********************************************************************
 49    /*********************************************************************
 50    *                                                                   *
 51    *   BASIC/360 V2.1                                                  *
 52    *                                                                   *
 53    *********************************************************************
 54    *                                                                   *
 55    *   THIS PROJECT WAS STARTED AS A CLASS PROJECT A WHILE BACK. IN    *
 56    *   TYPICAL IT STYLE, IT WAS SHELVED UNTIL WE HAD TIME TO WORK ON   *
 57    *   IT AGAIN.  IT IS ONLY 42 YEARS LATE.                            *
 58    *                                                                   *
 59    *   I FOUND IT IM MY ARCHIVES AND SCANNED IT.  WITH A LITTLE        *
 60    *   WORK, BASIC/360 LIVES (OR HAS BEEN RESURECTED - DEPENDS ON HOW  *
 61    *   YOU WANT TO LOOK AT IT).                                        *
 62    *                                                                   *
 63    *   V1.0 WORKED BUT I DID SOME TESTING ON IT AND FOUND A FEW BUGS   *
 64    *   IN THE CODE.  THEY WERE FIXED.                                  *
 65    *                                                                   *
 66    *********************************************************************
 67    *                                                                   *
 68    *   V2.1 CHANGE LOG                                                 *
 69    *  -- FIXES:                                                        *
 70    *   - CORRECTED TYPOS.                                              *
 71    *  --ENHANCEMENTS:                                                  *
 72    *   - CLEANED UP CODE FOR IMPLEMENTING BASIC LIBRARY FUNCTIONS      *
 73    *   - ADDED RND FUNCTION TO THE BASIC LIBRARY FUNCTIONS             *
 74    *   - CONSOLDATED THE STRING_STACK INTO SYMBOL_TABLE TO PREPARE     *
 75    *     FOR SUPPORTING STRING VARIABLES.                              *
 76    *   - REVISING CODE TO USE THE SELECT....ENDSELECT MACROS           *
 77    *   - REVISING CREATION OF PC_OPCODE TABLE TO USE MACROS            *
 78    *   - REVISED SYNTAX ERROR MESSAGES WITH MORE DETAIL                *
 79    *   - ADDED SUPPORT TO PCODE INTERPRETER TO ABOUT ILLEGAL MIXED     *
 80    *     MODE (I.E. MIXING NUMERIC AND STRINGS TOGETHER IN A LINE)     *
 81    *   - STRING VARIABLES ADDED.                                       *
 82    *   - STRING CONSTANTS IN LET STATEMENTS ADDED.                     *
 83    *   - SUPPORT FOR STRINGS IN READ AND DATA STATEMENTS.              *
 84    *                                                                   *
 85    *********************************************************************
 86    *                                                                   *
 87    *   V2.0 CHANGE LOG                                                 *
 88    *  -- ID CHANGE.  THERE WAS NO SUCH PLACE AS SOUTH HAMMOND          *
 89    *                 INSTITUTE OF TECHNOLOGY.  IT WAS REALLY PURDUE    *
```

MACRO SOURCE2 LISTING

```
 90    *                UNIVERSITY CALUMET.  THE ACRONYM WAS A JOKE       *
 91    *                ORIGINALLY BUT NOW IS NOT IN GOOD TASTE.          *
 92    *  -- FIXES:                                                       *
 93    *   - IF A PRINT STATEMENT FOLLOWS AN IF STATEMENT, THE COMPILER   *
 94    *     ABORTS WITH A PROTECTION EXCEPTION.                          *
 95    *   - PRINTING VALUES => 1.0E+6 RESULTS IN BAD OUTPUT              *
 96    *   - DEFAULT PRINT COLUMN WIDTHS WERE CHANGED FROM 12 TO 14       *
 97    *   - DIVISION BY ZERO CAUSES JOB TO ABORT                         *
 98    *   - CODE ADDED TO ABORT THE BASIC PROGRAM NOT THE JOB            *
 99    *   - FIXED CODE GENERATION FOR DIM ACCESS.                        *
100    *  --ENHANCEMENTS:                                                 *
101    *   - CHANGED CODE TO UTILIZE PL/I(F) FEATURES.                    *
102    *   - MISC CODE CLEANUP AND COMMENTS ADDED.                        *
103    *   - PC_FORMAT WAS ADDED TO THE VALID OPCODE TABLE TO IDENTIFY     *
104    *     WHAT WAS IN THE PC_OBJECT FIELD.  PRINT_PCODES WAS ALSO      *
105    *     MODIFIED TO USED THE FORMAT CODES INSTEAD OF THE PNEMONICS   *
106    *   - DEF FUNCTIONS HAVE BEEN IMPLEMENTED.                         *
107    *                                                                  *
108    ********************************************************************
109    *                                                                  *
110    *   WISH LIST (OR STUFF PUT OFF UNTIL LATER)                       *
111    *   - SPLIT SINGLE PROGRAM VS BATCH MONITOR.  SINGLE PGM COULD     *
112    *     ADD 'DATAFILE'.  PROCESS DATA STMTS THEN READ DATAFILE       *
113    *     SETS UP FOR IDE MODE.                                        *
114    *   - CHANGE FOR..NEXT TO A DO..WHILE CONSTRUCT                    *
115    *   - TSO ENVIRONMENT IMPLEMENTATION.                              *
116    *   - A 'COPY' OR SOURCE CODE LIB FACILITY.                        *
117    *   - PRINT USING STATEMENT.                                       *
118    *                                                                  *
119    ********************************************************************/
```

MACRO SOURCE2 LISTING

```
120     %INCLUDE FIX2STR;
121     %INCLUDE SELECT;
122     %INCLUDE GENPC;
123     %INCLUDE GENSYM;
124
```

MACRO SOURCE2 LISTING

```
125    BASIC:PROCEDURE OPTIONS(MAIN);
126
127    /*********************************************************************
128    *                                                                    *
129    *   DEFINE MAX SIZES FOR GLOBAL TABLES USED IN THIS PROGRAM          *
130    *                                                                    *
131    *********************************************************************/
132
133        %DECLARE $DATA_STACK        FIXED;
134        %DECLARE $MAX_LINES         FIXED;
135        %DECLARE $MAX_SYM           FIXED;
136        %DECLARE $MAX_PCODE         FIXED;
137        %DECLARE $MAX_EXECS         FIXED;
138        %$DATA_STACK=500;       /* MAX NUMBER OF DATA NUMBERS OR STRING*/
139        %$MAX_LINES=500;        /* MAX NUMBER OF LINE IN BASIC PGM     */
140        %$MAX_SYM=100;          /* MAX NUMBER DATA ELEMENTS IN SYM TBL */
141        %$MAX_PCODE=500;        /* MAX NUMBER OF PCODES                */
142        %$MAX_EXECS=5000;       /* MAX PCODES BEFORE ABORTED AS LOOPED */
143
144    /*********************************************************************
145    *                                                                    *
146    *                         GLOBAL VARIABLES                           *
147    *                                                                    *
148    *********************************************************************/
149
150        DECLARE PAGE_TITLE          CHAR(20) STATIC
151                                    INITIAL('BASIC/360 V2.2.0');
152        DECLARE PAGE_NUM            FIXED DECIMAL(5,0) INITIAL(0);
153        DECLARE PGM_PAGE_NUM        FIXED DECIMAL(5,0) INITIAL(0);
154        DECLARE EOF_SYSIN           BIT(1) ALIGNED INITIAL('0'B);
155        DECLARE EOP_SYSIN           BIT(1) ALIGNED;
156        DECLARE QUOTE_1             CHAR(1) INITIAL('"'),
157                QUOTE_2             CHAR(2) INITIAL('""');
158        DECLARE ERROR_COUNT         FIXED DECIMAL(5,0) INITIAL(0);
159
160        DECLARE 1 STMT_IN,
161                  2 STMT            CHAR(80),
162                 (2 STMT_LEFT,
163                  2 STMT_RIGHT,
164                  2 STMT_CH)        FIXED BINARY ALIGNED;
165        DECLARE STMT_BUFF           CHAR(80);
166        DECLARE LAST_LINE_NUM       FIXED DECIMAL(5,0);
167        DECLARE REF_LINE_NUM        FIXED DECIMAL(5,0);
168        DECLARE WORD                CHAR(8);
169        DECLARE RUN_DATE            CHAR(10);
```

MACRO SOURCE2 LISTING

```
170        DECLARE BASIC_RENUM         BIT(1) ALIGNED INITIAL('0'B);
171        DECLARE MONITOR_STMT        CHAR(80);
172
173     /*****************************************************************
174     *                                                               *
175     *                        GLOBAL DEBUGING                        *
176     *                                                               *
177     *****************************************************************/
178
179        DECLARE STACK_PRINT_DEBUG   BIT(1) ALIGNED INITIAL('0'B);
180        DECLARE EXECUTION_DEBUG     BIT(1) ALIGNED INITIAL('0'B);
181        DECLARE TABLE_PRINT         BIT(1) ALIGNED INITIAL('0'B);
182        DECLARE TABLE_DUMP          BIT(1) ALIGNED INITIAL('0'B);
183        DECLARE ICODE_PRINT         BIT(1) ALIGNED INITIAL('0'B);
184
```

MACRO SOURCE2 LISTING

```
185    /*****************************************************************
186    *                                                               *
187    *                        GLOBAL CONSTANTS                       *
188    *                                                               *
189    *  THESE CONSTANTS ARE USED IN TWO OR MORE OF THE MAJOR MODULES     *
190    *  OF THE COMPILER/INTERPRETER.                                 *
191    *                                                               *
192    *****************************************************************/
193
194        DECLARE TRUE               BIT(1) ALIGNED STATIC INITIAL('1'B);
195        DECLARE FALSE              BIT(1) ALIGNED STATIC INITIAL('0'B);
196        DECLARE ZERO               FIXED BINARY ALIGNED STATIC
197                                                    INITIAL(0);
198
199        DECLARE VALID_VAR_CHARS    CHAR(37) STATIC
200           INITIAL(' ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789');
201
202        DECLARE 1 KEY_WORD_AREA    STATIC,
203                  2 KW_DATA        CHAR(8) INITIAL('DATA'),
204                  2 KW_DEF         CHAR(8) INITIAL('DEF'),
205                  2 KW_DIM         CHAR(8) INITIAL('DIM'),
206                  2 KW_END         CHAR(8) INITIAL('END'),
207                  2 KW_FOR         CHAR(8) INITIAL('FOR'),
208                  2 KW_GOSUB       CHAR(8) INITIAL('GOSUB'),
209                  2 KW_GOTO        CHAR(8) INITIAL('GOTO'),
210                  2 KW_IF          CHAR(8) INITIAL('IF'),
211                  2 KW_LET         CHAR(8) INITIAL('LET'),
212                  2 KW_NEXT        CHAR(8) INITIAL('NEXT'),
213                  2 KW_PRINT       CHAR(8) INITIAL('PRINT'),
214                  2 KW_READ        CHAR(8) INITIAL('READ'),
215                  2 KW_REM         CHAR(8) INITIAL('REM'),
216                  2 KW_RETURN      CHAR(8) INITIAL('RETURN'),
217                  2 KW_RESTORE     CHAR(8) INITIAL('RESTORE'),
218                  2 KW_STOP        CHAR(8) INITIAL('STOP'),
219
220                1 KEY_WORDS(16)    DEFINED KEY_WORD_AREA
221                                   CHAR(8);
222
223        DECLARE 1 SS_CONSTANTS     STATIC ALIGNED,
224                  2  SS_UNKNWN     FIXED BINARY INITIAL(0),
225                  2  SS_UNKNWM_DESC CHAR(8)     INITIAL('UNKNOWN '),
226                  2  SS_CONST      FIXED BINARY INITIAL(1),
227                  2  SS_CONST_DESC  CHAR(8)     INITIAL('CONST   '),
228                  2  SS_FUNC       FIXED BINARY INITIAL(2),
229                  2  SS_FUNC_DESC   CHAR(8)     INITIAL('FUNCTION'),
```

MACRO SOURCE2 LISTING

```
230                 2  SS_VAR         FIXED BINARY INITIAL(3),
231                 2  SS_VAR_DESC    CHAR(8)      INITIAL('VAR     '),
232                 2  SS_DIM_VAR     FIXED BINARY INITIAL(4),
233                 2  SS_DIM_DESC    CHAR(8)      INITIAL('DIM     '),
234                 2  SS_DEF_VAR     FIXED BINARY INITIAL(5),
235                 2  SS_DEF_DESC    CHAR(8)      INITIAL('DEF     '),
236                 2  SS_STRCON      FIXED BINARY INITIAL(6),
237                 2  SS_STRCON_DESC CHAR(8)      INITIAL('STRCON  '),
238                 2  SS_STRVAR      FIXED BINARY INITIAL(7),
239                 2  SS_STRVAR_DESC CHAR(8)      INITIAL('STRVAR  '),
240                 2  SS_STRDIM      FIXED BINARY INITIAL(8),
241                 2  SS_STRDIM_DESC CHAR(8)      INITIAL('STRDIM  ');
242
243       DECLARE 1 SS_CON_TABLE      BASED (SS_CON_TABLE_PTR),
244                 2  SS_TAB(0:8),
245                   3 SS_CODE        FIXED BINARY,
246                   3 SS_DESC        CHAR(8);
```

MACRO SOURCE2 LISTING

```
247    /*****************************************************************
248    *                                                               *
249    *                       PSEUDO OPCODES DEFINITION               *
250    *                                                               *
251    *  THE PC_FORMAT CODES DESCRIBE THE TYPE OF ARGUMENT EACH PSEUDO  *
252    *  EXPECTS.  MOSTLY USED TO CORRECTLY PRINT THE PCODES.           *
253    *      PC_FORMAT_   INDICATES                                   *
254    *      ----------   ------------------------------------------  *
255    *          0        VARIABLE LOCATED IN THE SYMBOL_TABLE        *
256    *          1        OBJECT IS A LINE NUMBER DEFINITION          *
257    *          2        OBJECT IS A LINE NUMBER IN THE LINE_STACK   *
258    *          3        OBJECT IS A STRING                          *
259    *          4        OBJECT IS NOT USED                          *
260    *          5        OBJECT IS AN OFFSET IN THE PC_TABLE         *
261    *                                                               *
262    *   THE GENPC MACRO DEFINES A P-CODE.  GENPC IS GIVEN 4 PARMS:  *
263    *     1)  THE MNEMONIC FOR THE P-CODE                           *
264    *     2)  THE NUMERIC P-CODE                                    *
265    *     3)  THE P-CODE FORMAT AS DEFINED IN PC-FORMAT             *
266    *     4)  TYPE CHECKING ENFORCEMENT - TWO BINARY DIGITS THAT    *
267    *         INDICATE IF NUMBER VS STRING TESTS ARE TO BE MADE.    *
268    *         00=NO CHECKING                                        *
269    *         01=OPERAND MUST BE NUMERIC                            *
270    *         10=OPERAND MUST BE STRING                             *
271    *         11=OPERAND MUST TYPE MUST MATCH ACCUM TYPE            *
272    *                                                               *
273    *****************************************************************/
274
275
276        DECLARE 1 MISC_CODE_DEF     STATIC ALIGNED,
277                  2  PC_FORMAT_0    FIXED BINARY INITIAL(0),
278                  2  PC_FORMAT_1    FIXED BINARY INITIAL(1),
279                  2  PC_FORMAT_2    FIXED BINARY INITIAL(2),
280                  2  PC_FORMAT_3    FIXED BINARY INITIAL(3),
281                  2  PC_FORMAT_4    FIXED BINARY INITIAL(4),
282                  2  PC_FORMAT_5    FIXED BINARY INITIAL(5),
283                  2  PCT_LFEED      FIXED BINARY INITIAL(0),
284                  2  PCT_TAB        FIXED BINARY INITIAL(1),
285                  2  PCT_NOTAB      FIXED BINARY INITIAL(2),
286                  2  EXP_RCVR       FIXED BINARY INITIAL(0),
287                  2  EXP_CALC       FIXED BINARY INITIAL(1),
288                  2  EXP_FN_CALC    FIXED BINARY INITIAL(2);
289
290        DECLARE 1 PC_CONSTANTS      STATIC ALIGNED,
291                  GENPC(SLN,00,1,00)
```

MACRO SOURCE2 LISTING

```
292                  GENPC(LDA,01,0,11)
293                  GENPC(STA,02,0,11)
294                  GENPC(EXP,03,0,01)
295                  GENPC(ADD,04,0,01)
296                  GENPC(SUB,05,0,01)
297                  GENPC(MUL,06,0,01)
298                  GENPC(DIV,07,0,01)
299                  GENPC(RDV,08,0,11)
300                  GENPC(PRV,09,0,11)
301                  GENPC(PCT,10,5,00)
302                  GENPC(FNC,11,0,00)
303                  GENPC(END,12,0,00)
304                  GENPC(B  ,13,2,00)
305                  GENPC(BAL,14,2,00)
306                  GENPC(RET,15,0,00)
307                  GENPC(PRS,16,3,00)
308                  GENPC(LCA,17,0,11)
309                  GENPC(LCB,18,0,11)
310                  GENPC(BEQ,19,2,00)
311                  GENPC(BNE,20,2,00)
312                  GENPC(BGT,21,2,00)
313                  GENPC(BLT,22,2,00)
314                  GENPC(BGE,23,2,00)
315                  GENPC(BLE,24,2,00)
316                  GENPC(FSU,25,0,00)
317                  GENPC(FIX,26,0,00)
318                  GENPC(FUL,27,0,00)
319                  GENPC(FST,28,0,00)
320                  GENPC(FNX,29,0,00)
321                  GENPC(PTB,30,0,00)
322                  GENPC(RST,31,4,00)
323                  GENPC(DSL,32,4,00)
324                  GENPC(LDR,33,0,00)
325                  GENPC(STR,34,0,00)
326                  GENPC(JMP,35,5,00)
327                  GENPC(CFN,36,0,00)
328                  GENPC(RFN,37,0,00)
329                  GENPC(STP,38,0,00)
330
331                1 PC_CON_TABLE      BASED (PC_CON_TABLE_PTR),
332                  2  PC_OPTAB(0:GENPC_CTR),
333                    3 PC_OP_CODE     FIXED BINARY,
334                    3 PC_MNEMONIC    CHAR(4),
335                    3 PC_FORMAT      FIXED BINARY,
336                    3 PC_ALLOW       BIT(2) ALIGNED;
```

MACRO SOURCE2 LISTING

```
337    /*****************************************************************
338    *                                                               *
339    *                     GLOBAL OBJECT STRUCTURES                  *
340    *                                                               *
341    *  THESE ITEMS ARE USED TO EXECUTE THE BASIC PROGRAM.  THE COMPILE  *
342    *  PHASE STORES THE DATA IN THESE OBJECTS AND THE EXECUTION PHASE   *
343    *  EXECUTES THEM.                                               *
344    *                                                               *
345    *  DATA_STACK IS USED TO STORE NUMBERS FROM DATA STATEMENTS.        *
346    *                COMPILE STACKS THEM UP AND EXECUTE UNSTACKS THEM   *
347    *                NUMBERS EACH TIME A READ IS EXECUTED.              *
348    *                                                               *
349    *  LINE_STACK IS USED TO STORE THE BASIC LINE NUMBERS AND THE       *
350    *                OFFSET TO WHERE IN P_CODE THE STATEMENT STARTS.    *
351    *                THESE ARE USED TO FIND WHERE GOTO AND GOSUBS       *
352    *                TRANSFER CONTROL TO IN P_CODE_STACK.               *
353    *                                                               *
354    *  P_CODE_STACK IS USED TO STORE THE EXECUTABLE P_CODES GENERATED   *
355    *                DURING THE COMPILE PROCESS ARE THEN EXECUTED.      *
356    *                                                               *
357    *  SYMBOL_TABLE IS USED TO STORE THE ALL OF THE NUMERIC DATA        *
358    *                VARIABLE, CONSTANTS, DIM VARIABLES AND FUNCTIONS.  *
359    *                THIS TABLE IS POPULATED DURING COMPILE TIME WITH   *
360    *                VARIABLES NAMES, CONSTANTS AND DIMS NAMES.         *
361    *                DURING EXECUTION, THE VALUES FOR ALL VARIABLES     *
362    *                STORED AND RETRIEVED FROM THIS TABLE BY THE        *
363    *                EXECUTION PHASE.  NUMERIC CONSTANTS ARE RETRIEVED  *
364    *                FROM THIS TABLE DURING EXECUTION.                  *
365    *                FUNCTIONS ARE INCLUDED IN THIS TABLE AS WELL.      *
366    *                STRING_TABLE AND SYMBOL TABLE WERE MERGED. IT      *
367    *                IS USED TO STORE THE ALL OF THE STRING DATA.       *
368    *                THIS TABLE IS POPULATED DURING COMPILE TIME WITH   *
369    *                STRING CONSTANTS AND SPACE RESERVED FOR STRING     *
370    *                VARIABLES.                                         *
371    *                DURING EXECUTION, THE VALUES FOR THE CONSTANTS     *
372    *                AND VARIABLES ARE RETRIEVED FROM THIS TABLE.       *
373    *                                                               *
374    *  SOURCE_TABLE  IS USED TO STORE THE SOURCE CODE TO BE COMPILED.   *
375    *                EACH OF THE ENVIRONMENTS LOADS THE BASIC PROGRAM   *
376    *                AND THEM PASSES IT TO THE COMPILER.                *
377    *                                                               *
378    *  DEF_FUNCTIONS IS USED TO STORE THE NAMES OF THE USER DEFINED     *
379    *                FUNCTIONS.                                         *
380    *                                                               *
381    *****************************************************************/
```

MACRO SOURCE2 LISTING

```
382
383        DECLARE 1 DATA_STACK        ALIGNED,
384                  2  (DS_CUR,
385                      DS_MAX)       FIXED BINARY,
386                  2   DS_TABLE($DATA_STACK),
387                     3 DS_STR        FIXED BINARY,
388                     3 DS_ITEM       FLOAT BINARY;
389        DECLARE 1 LINE_STACK        ALIGNED,
390                  2  (LS_CUR,
391                      LS_MAX)       FIXED BINARY,
392                  2   LS_NUM($MAX_LINES),
393                     3  LS_LINE     FIXED DECIMAL(5,0),
394                     3  LS_OFFSET   FIXED BINARY;
395        DECLARE 1 SOURCE_CODE       ALIGNED,
396                  2  (SC_CUR,
397                      SC_MAX)       FIXED BINARY,
398                  2  SOURCE_AREA($MAX_LINES),
399                     3  SOURCE_LINE CHAR(80);
400        DECLARE 1 P_CODE_STACK      ALIGNED,
401                  2  (PC_CUR,
402                      PC_MAX)       FIXED BINARY,
403                  2   PC_NUM($MAX_PCODE),
404                     3  (PC_OPCODE,
405                         PC_OBJECT) FIXED BINARY;
406        DECLARE 1 SYMBOL_STACK      ALIGNED,
407                  2  (SS_CUR,
408                      SS_MAX_FNC,
409                      SS_MAX)       FIXED BINARY,
410                  2  SYMBOL_AREA($MAX_SYM),
411                     3  SYMBOL      CHAR(10),
412                     3  SYM_TYPE    FIXED BINARY,
413                     3  SYM_VALUE   FLOAT DECIMAL,
414                     3  SYM_DIM_MAX FIXED BINARY,
415                     3  STRING_VAL  CHAR(80) VARYING;
416        DECLARE 1 DEF_FUNCTIONS     ALIGNED,
417                  2  (DF_CUR,
418                      DF_MAX)       FIXED BINARY,
419                  2  DEF_FUNC_AREA(10),
420                     3  DF_NAME     CHAR(10),
421                     3 (DF_OFFSET,
422                     3  DF_RETURN)  FIXED BINARY;
423
```

MACRO SOURCE2 LISTING

```
424    /*****************************************************************/
425    /*****************************************************************/
426    /*****************************************************************/
427    /*****************************************************************/
428    /*****************************************************************/
429
430        RUN_DATE=DATE;   /* DATE IS IN YYMMDD FORMAT */
431        RUN_DATE=SUBSTR(RUN_DATE,3,2)|| '/' || SUBSTR(RUN_DATE,5,2) ||
432                 '/20' || SUBSTR(RUN_DATE,1,2);
433
434        ON ENDFILE(SYSIN)
435           EOF_SYSIN,EOP_SYSIN=TRUE;
436
437    /*****************************************************************
438    *                                                                *
439    *  THESE TWO POINTERS MUST BE SET.  THE BASE STRUCTURES ARE MIXED    *
440    *  WITH BINARY AND CHARACTER DATA AND PL/I DOES NOT ALLOW DEFINED    *
441    *  STRUCTURES LIKE THIS.  SO THEY WERE MADE INTO BASED TABLES.       *
442    *                                                                *
443    *****************************************************************/
444        PC_CON_TABLE_PTR = ADDR(PC_CONSTANTS);
445        SS_CON_TABLE_PTR = ADDR(SS_CONSTANTS);
446
447    /*****************************************************************
448    *                                                                *
449    *  THIS IS THE MAIN DRIVING LOOP FOR BASIC.  IF THERE IS A ++        *
450    *  LINE AS FIRST LINE IN SYSIN, IT IS ASSUMED BASIC IS RUNNING IN    *
451    *  MONITOR MODE.  IF NO ++ LINE IS FOUND, IT IS 1 UP MODE.           *
452    *                                                                *
453    *****************************************************************/
454        GET EDIT(STMT_BUFF) (A(80)); /*  THIS PRIMES THE INPUT PROCESS */
455
456        DO WHILE(EOF_SYSIN=FALSE);
457           CALL INITIALIZE;
458           CALL MONITOR;
459           IF SC_MAX>0 THEN
460           DO;
461              CALL COMPILE;
462              IF BASIC_RENUM & (ERROR_COUNT=0) THEN
463              DO;
464                 CALL RENUM;
465                 CALL INITIALIZE;
466                 CALL COMPILE;
467              END;
468              IF ERROR_COUNT=0 THEN
```

MACRO SOURCE2 LISTING

```
469              DO;
470                 CALL EXECUTE;
471                 CALL TERMINATE;
472              END;
473           END;
474        END;
```

MACRO SOURCE2 LISTING

```
475    /*********************************************************************/
476    /*********************************************************************/
477    /*********************************************************************/
478    /*********************************************************************/
479    /*********************************************************************/
480
481    MONITOR:PROC;
482
483    /*********************************************************************
484    *                                                                    *
485    *   THIS PROC READS THE INPUT FILE AND LOADS THE BASIC PROGRAMS     *
486    *   INTO THE SOURCE_CODE STRUCTURE FOR MONITOR MODE.  MONITOR       *
487    *   CONTROL STATEMENTS START WITH ++.  OPTION CARDS START WITH      *
488    *   AN * IS COL 1.  THEY ARE PROCESSED HERE AND PASSED BACK TO      *
489    *   CALLER SO COMPILE CAN PRINT THEM BUT THEN TREATED AS COMMENTS   *
490    *                                                                    *
491    * NESTING:MONITOR                                                    *
492    *********************************************************************/
493
494        ON ENDFILE(SYSIN)
495           EOF_SYSIN,EOP_SYSIN=TRUE;
496        MONITOR_STMT=(80)' ';
497
498        IF SUBSTR(STMT_BUFF,1,2)='++' THEN
499        DO;
500           MONITOR_STMT=STMT_BUFF;
501           EOP_SYSIN=FALSE;
502           TABLE_PRINT=FALSE;
503           TABLE_DUMP=FALSE;
504           STACK_PRINT_DEBUG=FALSE;
505           ICODE_PRINT=FALSE;
506           EXECUTION_DEBUG=FALSE;
507           BASIC_RENUM= (SUBSTR(STMT_BUFF,1,8)='++RENUM ');
508           PGM_PAGE_NUM=0;
509           SC_CUR,SC_MAX=0;
510           GET EDIT(STMT_BUFF) (A(80));
511           DO WHILE((EOF_SYSIN=FALSE)&(EOP_SYSIN=FALSE));
512              SC_MAX=SC_MAX+1;
513              IF SC_MAX>HBOUND(SOURCE_LINE,1) THEN
514              DO;
515                 PUT SKIP LIST
516                   ('**** FATAL ERROR - PROGRAM TO BIG ***');
517              STOP;
518              END;
519              SOURCE_LINE(SC_MAX)=STMT_BUFF;
```

MACRO SOURCE2 LISTING

```
520              GET EDIT(STMT_BUFF) (A(80));
521              IF SUBSTR(STMT_BUFF,1,2)='++' THEN
522              DO;
523                EOP_SYSIN=TRUE;
524              END;
525           END;
526        END;
527        ELSE  /*  NO MONITOR CNTL - TREAT AS STRAIGHT BATCH */
528        DO WHILE(EOF_SYSIN=FALSE);
529           SC_MAX=SC_MAX+1;
530           IF SC_MAX>HBOUND(SOURCE_LINE,1) THEN
531           DO;
532               PUT SKIP LIST('**** FATAL ERROR - PROGRAM TO BIG ***');
533               STOP;
534           END;
535           SOURCE_LINE(SC_MAX)=STMT_BUFF;
536           GET EDIT(STMT_BUFF) (A(80));
537        END;
538
539     END MONITOR;
```

MACRO SOURCE2 LISTING

```
540    /*****************************************************************/
541    /*****************************************************************/
542    /*****************************************************************/
543    /*****************************************************************/
544    /*****************************************************************/
545
546    INITIALIZE:PROC;
547
548    /*****************************************************************
549    *                                                                *
550    *   THIS PROC INITIALIZES ALL OF THE GLOBAL DATA ELEMENTS AND     *
551    *   STRUCTURES FOR THE COMPILATION AND EXECUTION OF THE BASIC     *
552    *   PROGRAM.                                                     *
553    *                                                                *
554    * NESTING:INITIALIZE                                             *
555    *****************************************************************/
556
557        STMT_LEFT=1;
558        STMT_RIGHT=72;
559        LAST_LINE_NUM=-1;
560
561        DS_CUR,DS_MAX=0;
562        LS_CUR,LS_MAX=0;
563        PC_CUR,PC_MAX=0;
564        DF_CUR,DF_MAX=0;
565        ERROR_COUNT=0;
566
567        GENSYM(NULL,SS_VAR,0.0,*)
568
569      /*********************************************************
570      *                                                        *
571      *  THESE ARE THE BUILTIN FUNCTIONS.  IF YOU ADD MORE     *
572      *  BE SURE TO ADD THEM TO THE FUNCTION INTERPRETER IN    *
573      *  EXECUTE   PSUDEO OP CODE FNC                          *
574      *                                                        *
575      *********************************************************/
576
577    /*****************************************************************
578    *                                                                *
579    * IMPORTANT NOTE - IF ANY CHANGES ARE MADE TO LIBRARY FUNCTIONS,  *
580    * THE PCODE FNC SHOULD MATCH THE CHANGES IN THE INITIALIZE PROC   *
581    *                                                                *
582    *****************************************************************/
583
584        GENSYM(SQR,SS_FUNC,0.0,*)
```

MACRO SOURCE2 LISTING

```
585         GENSYM(ABS,SS_FUNC,0.0,*)
586         GENSYM(TAB,SS_FUNC,0.0,*)
587         GENSYM(INT,SS_FUNC,0.0,*)
588         GENSYM(COS,SS_FUNC,0.0,*)
589         GENSYM(SIN,SS_FUNC,0.0,*)
590         GENSYM(TAN,SS_FUNC,0.0,*)
591         GENSYM(RND,SS_FUNC,0.0,*)
592         GENSYM(INR,SS_FUNC,0.0,*)
593
594         SS_CUR,SS_MAX,SS_MAX_FNC = GENSYM_CTR;
595
596     END INITIALIZE;
597
```

MACRO SOURCE2 LISTING

```
598    /*********************************************************************/
599    /*********************************************************************/
600    /*********************************************************************/
601    /*********************************************************************/
602    /*********************************************************************/
603
604    RENUM:PROC;
605
606    /*********************************************************************
607    *                                                                    *
608    *   THIS PROC RENUMBERS THE SOURCE PROGRAM.  IT IS ASSUMED THAT      *
609    *   THE BASIC PROGRAM COMILED WITH NO ERRORS AND THE CONTENTS OF     *
610    *   THE GLOBAL TABLES ARE INTACT.  THE SOURCE_CODE TABLE WILL BE     *
611    *   UPDATED WITH THE RENUMBERED PROGRAM.                             *
612    *                                                                    *
613    *   A "DECK" OF THE RENUMBERED PROGRAM WILL BE WRITTEN TO THE        *
614    *   RENUMFL                                                          *
615    *                                                                    *
616    * NESTING:NONE                                                       *
617    *********************************************************************/
618
619        DECLARE LINE_WORK           CHAR(80);
620        DECLARE LINE_SUB            FIXED BINARY ALIGNED;
621        DECLARE A_BLANK             FIXED BINARY ALIGNED;
622        DECLARE FIRST_CHAR          FIXED BINARY ALIGNED;
623        DECLARE FIRST_DIGIT         FIXED BINARY ALIGNED;
624        DECLARE (I,LAST_CHAR)       FIXED BINARY ALIGNED;
625        DECLARE OLD_LINE_NUM        FIXED DECIMAL(5,0);
626        DECLARE NEW_LINE_NUM(500)   FIXED DECIMAL(5,0);
627        DECLARE CONTINUE_SCAN       BIT(1) ALIGNED;
628        DECLARE EDIT_LINE_NUM       PIC 'ZZZZ9';
629        DECLARE RENUMFL             STREAM OUTPUT FILE;
630
631        DO LINE_SUB = 1 TO LS_MAX;             /* NEW NUM START AT   */
632            NEW_LINE_NUM(LINE_SUB)=LINE_SUB*10; /* 10 BY 10 FOR NOW   */
633        END;                                   /* CORR TO LINE_STACK */
634
635        PUT FILE(RENUMFL) EDIT('++BASIC') (SKIP,A);
636        DO LINE_SUB = 1 TO SC_MAX;
637            LINE_WORK = SOURCE_LINE(LINE_SUB);
638            IF SUBSTR(LINE_WORK,1,1)='*' THEN;
639            ELSE
640            DO;
641               A_BLANK = INDEX(LINE_WORK,' ');  /* FIND FIRST SPACE */
642               IF A_BLANK < 2 THEN
```

MACRO SOURCE2 LISTING

```
643               DO;
644                  PUT SKIP LIST('**** RENUM FATAL ERROR 1 ****');
645                  STOP;
646               END;
647               OLD_LINE_NUM = SUBSTR(LINE_WORK, 1, A_BLANK-1);
648               CONTINUE_SCAN=TRUE;
649               DO I=1 TO LS_MAX WHILE(CONTINUE_SCAN);
650                  IF OLD_LINE_NUM=LS_LINE(I) THEN
651                  DO;
652                      EDIT_LINE_NUM = NEW_LINE_NUM(I);
653                      CALL TRIM_EDIT_NUM;
654                      LINE_WORK=SUBSTR(EDIT_LINE_NUM,FIRST_DIGIT) ||
655                                SUBSTR(LINE_WORK,A_BLANK);
656                      CONTINUE_SCAN=FALSE;
657                  END;
658               END;
659               IF CONTINUE_SCAN THEN
660               DO;
661                  PUT SKIP LIST('**** RENUM FATAL ERROR 2 ****');
662                  STOP;
663               END;
664
665               CONTINUE_SCAN=TRUE;
666               A_BLANK = INDEX(LINE_WORK,' ');  /* FIND FIRST SPACE */
667               IF A_BLANK = 0 THEN
668               DO;
669                  PUT SKIP LIST('**** RENUM FATAL ERROR 3 ****');
670                  STOP;
671               END;
672
673               DO I=A_BLANK+1 TO STMT_RIGHT WHILE(CONTINUE_SCAN);
674                  IF SUBSTR(LINE_WORK,I,1)=' ' THEN;
675                  ELSE CONTINUE_SCAN=FALSE;
676               END;
677
678               I=I-1;
679               IF SUBSTR(LINE_WORK,I,3)='IF ' |
680                  SUBSTR(LINE_WORK,I,2)='GO'   THEN
681               DO;
682                  CONTINUE_SCAN=TRUE;
683                  I=STMT_RIGHT;
684                  DO WHILE(CONTINUE_SCAN);
685                     IF SUBSTR(LINE_WORK,I,1)=' ' THEN
686                        I=I-1;
687                     ELSE
```

MACRO SOURCE2 LISTING

```
688                        CONTINUE_SCAN=FALSE;
689                     END;
690                     CONTINUE_SCAN=TRUE;
691                     LAST_CHAR=I;
692                     DO WHILE(CONTINUE_SCAN);
693                        IF SUBSTR(LINE_WORK,I,1)=' ' THEN
694                           CONTINUE_SCAN=FALSE;
695                        ELSE
696                           I=I-1;
697                     END;
698                     FIRST_CHAR=I;
699                     OLD_LINE_NUM = SUBSTR(LINE_WORK,I+1,LAST_CHAR-I);
700                     CONTINUE_SCAN=TRUE;
701                     DO I=1 TO LS_MAX WHILE(CONTINUE_SCAN);
702                        IF OLD_LINE_NUM=LS_LINE(I) THEN
703                        DO;
704                            EDIT_LINE_NUM = NEW_LINE_NUM(I);
705                            CALL TRIM_EDIT_NUM;
706                            LINE_WORK=SUBSTR(LINE_WORK,1,FIRST_CHAR)
707                                      || SUBSTR(EDIT_LINE_NUM,FIRST_DIGIT);
708                            CONTINUE_SCAN=FALSE;
709                        END;
710                     END;
711                  END;
712                  SOURCE_LINE(LINE_SUB)=LINE_WORK;
713               END;
714               PUT FILE(RENUMFL) EDIT(SOURCE_LINE(LINE_SUB)) (SKIP,A);
715           END;
716           BASIC_RENUM=FALSE;
717
718       TRIM_EDIT_NUM:PROC;
719           SELECT (TRUE)
720              WHEN (NEW_LINE_NUM(I)<10)
721                 FIRST_DIGIT = 5;
722              WHEN (NEW_LINE_NUM(I)<100)
723                 FIRST_DIGIT = 4;
724              WHEN (NEW_LINE_NUM(I)<1000)
725                 FIRST_DIGIT = 3;
726              WHEN (NEW_LINE_NUM(I)<10000)
727                 FIRST_DIGIT = 2;
728              OTHERWISE
729                 FIRST_DIGIT = 1;
730           ENDSELECT;
731       END TRIM_EDIT_NUM;
732
```

MACRO SOURCE2 LISTING

```
733    END RENUM;
```

MACRO SOURCE2 LISTING

```
734    /*****************************************************************/
735    /*****************************************************************/
736    /*****************************************************************/
737    /*****************************************************************/
738    /*****************************************************************/
739
740    COMPILE:PROC;
741
742    /*****************************************************************
743    *                                                                *
744    *   THIS PROC DRIVES THE COMPILE PROCESS FOR THE BASIC PROGRAM    *
745    *                                                                *
746    * NESTING:NONE                                                   *
747    *****************************************************************/
748
749        DECLARE LAST_PCODE_PRINTED        FIXED BINARY ALIGNED INITIAL(0);
750        DECLARE TERMINATE_SCAN            BIT(1) ALIGNED;
751        DECLARE (FUNC_NAME,FUNC_ARG) CHAR(10);
752        DECLARE (TMP_CNT,STR_CNT)              PICTURE '99';
753
754        ON ENDPAGE(SYSPRINT)
755        BEGIN;
756           IF PAGE_NUM > 0 THEN PUT PAGE;
757           PAGE_NUM=PAGE_NUM+1;
758           PGM_PAGE_NUM=PGM_PAGE_NUM+1;
759           PUT EDIT (PAGE_TITLE,'DATE ',RUN_DATE,
760                     'PAGE ',PGM_PAGE_NUM)
761                    (COLUMN(60),A,COLUMN(93),A,A,COLUMN(110),
762                     A,F(5,0));
763           PUT SKIP(2) EDIT(MONITOR_STMT) (A)
764                       EDIT('OFFSET') (SKIP(2),A);
765           PUT SKIP;
766        END;
767
768        SIGNAL ENDPAGE(SYSPRINT);
769
770        STR_CNT = 0;
771        SC_CUR=0;
772        DO WHILE(SC_CUR<SC_MAX);
773            SC_CUR=SC_CUR+1;
774            STMT=SOURCE_LINE(SC_CUR);
775            PUT SKIP EDIT(PC_MAX+1,STMT) (P'999999',COLUMN(25),A);
776            IF SUBSTR(STMT,1,1)='*' THEN      /* DONT COMPILE OPTIONS */
777               CALL PROCESS_OPTS;
778            ELSE
```

MACRO SOURCE2 LISTING

```
779            DO;
780               STMT_CH=STMT_LEFT;
781               TERMINATE_SCAN=FALSE;
782               CALL GET_STMT_NUM(TRUE);
783               LS_MAX=LS_MAX+1;
784               CALL ADD_PCODE(PC_OPCODE_SLN,LS_MAX);
785               LS_LINE(LS_MAX)=LAST_LINE_NUM;
786               LS_OFFSET(LS_MAX)=PC_MAX;
787               CALL GET_KEYWORD;
788               CALL PROCESS_KEYWORD;
789               IF ICODE_PRINT THEN
790                 CALL PRINT_PCODES;
791            END;
792        END;
793
794        ON ENDPAGE(SYSPRINT)
795        BEGIN;
796           IF PAGE_NUM > 0 THEN PUT PAGE;
797           PAGE_NUM=PAGE_NUM+1;
798           PGM_PAGE_NUM=PGM_PAGE_NUM+1;
799           PUT EDIT (PAGE_TITLE,'DATE ',RUN_DATE,
800                     'PAGE ',PGM_PAGE_NUM)
801                    (COLUMN(60),A,COLUMN(93),A,A,COLUMN(110),
802                      A,F(5,0));
803           PUT SKIP(2);
804        END;
805
806        IF TABLE_PRINT THEN;
807        ELSE
808           GO TO END_OF_COMP;
809
810        DECLARE I                  FIXED BINARY ALIGNED;
811
812        CALL PRINT_SYMBOLS;
813
814        PUT SKIP(2) LIST('DEF NAME','OFFSET');
815        DO I=1 TO DF_MAX;
816          PUT SKIP LIST(DF_NAME(I),DF_OFFSET(I));
817        END;
818        PUT SKIP LIST('END OF DEF NAME TABLE');
819
820        PUT SKIP(2) EDIT('OFFSET','LINE  OP   OBJECT') (A,X(7),A);
821        CALL PRINT_PCODES;
822        PUT SKIP LIST('END OF PCODE TABLE');
823
```

MACRO SOURCE2 LISTING

```
824    END_OF_COMP:
825        PUT SKIP(2) EDIT('**** END OF COMPILATION ****') (A);
826        IF ERROR_COUNT=0 THEN
827        DO;
828           PUT EDIT(' NO ERRORS FOUND') (A);
829           IF BASIC_RENUM THEN
830               PUT EDIT(' - RENUMBERING PROGRAM') (A);
831        END;
832        ELSE
833        DO;
834           PUT EDIT(ERROR_COUNT,' ERRORS FOUND') (F(5),A);
835           IF BASIC_RENUM THEN
836               PUT EDIT(' - RENUMBERING BYPASSED') (A);
837        END;
838
839    PROCESS_OPTS:PROC;
840    /*********************************************************************
841    *                                                                    *
842    *   OPTIONS STATEMENTS ARE MIXED IN WITH THE SOURCE PROGRAM.  THEY   *
843    *   HAVE A "*" IN COLUMN 1 AND ARE TREATED AS A COMMENT BY THE       *
844    *   COMPILER.  THESE ARE BASICLY DEBUGGING TOOLS BUILT IN AND WILL   *
845    *   NOT NORMALLY BE USED.  THEY OPTIONS ARE:                         *
846    *   *TABLE    PRINT ALL OBJECT TABLES AT THE END OF COMPILATION      *
847    *   *DUMP     PRINT ALL OBJECT TABLES AT THE END OF EXECUTION        *
848    *   *STACK    PRINT THE PARSING STACK DEBUGGING TRACING.  THIS       *
849    *             OPTION STARTS AS SOON AS THE STATEMENT IS PROCESSED    *
850    *             IT REMAINS IN EFFECT UNTIL THE END OF PROGRAM OR       *
851    *             A *NOSTACK IS PROCESSED.                               *
852    *   *ICODE    PRINTS THE PCODE OBJECT CODE AS IT IS GENERATED.       *
853    *             OPTION STARTS AS SOON AS THE STATEMENT IS PROCESSED    *
854    *             IT REMAINS IN EFFECT UNTIL THE END OF PROGRAM OR       *
855    *             A *NOICODE IS PROCESSED.                               *
856    *   *TRACE    PRINT DEBUGGING INFORMATION WHILE THE BASIC PROGRAM    *
857    *             IS EXECUTING.                                          *
858    *                                                                    *
859    * NESTING:COMPILE                                                    *
860    *********************************************************************/
861
862        IF SUBSTR(STMT,1,7)='*TABLE ' THEN
863           TABLE_PRINT=TRUE;
864        ELSE
865        IF SUBSTR(STMT,1,9)='*NOTABLE ' THEN
866           TABLE_PRINT=FALSE;
867        ELSE
868        IF SUBSTR(STMT,1,6)='*DUMP ' THEN
```

MACRO SOURCE2 LISTING

```
869           TABLE_DUMP=TRUE;
870         ELSE
871         IF SUBSTR(STMT,1,7)='*STACK ' THEN
872           STACK_PRINT_DEBUG=TRUE;
873         ELSE
874         IF SUBSTR(STMT,1,9)='*NOSTACK ' THEN
875           STACK_PRINT_DEBUG=FALSE;
876         ELSE
877         IF SUBSTR(STMT,1,7)='*ICODE ' THEN
878           ICODE_PRINT=TRUE;
879         ELSE
880         IF SUBSTR(STMT,1,9)='*NOICODE ' THEN
881           ICODE_PRINT=FALSE;
882         IF SUBSTR(STMT,1,7)='*TRACE ' THEN
883           EXECUTION_DEBUG=TRUE;
884         ELSE
885         IF SUBSTR(STMT,1,9)='*NOTRACE ' THEN
886           EXECUTION_DEBUG=FALSE;
887         ELSE;  /*  JUST IGNORE INVALID OPTIONS */
888
889     END PROCESS_OPTS;
890
891     PRINT_PCODES:PROC;
892     /*********************************************************************
893     *                                                                   *
894     *                                                                   *
895     * NESTING:COMPILE                                                   *
896     *********************************************************************/
897
898         IF LAST_PCODE_PRINTED < PC_MAX THEN
899         DO;
900           DO I=LAST_PCODE_PRINTED+1 TO PC_MAX;
901             SELECT(PC_FORMAT(PC_OPCODE(I)))
902             WHEN(PC_FORMAT_0)
903                 PUT SKIP EDIT(I,   PC_MNEMONIC(PC_OPCODE(I)),
904                                    SYMBOL(PC_OBJECT(I)))
905                                   (P'999999',X(13),A,X(2),A);
906             WHEN(PC_FORMAT_1)
907                PUT SKIP EDIT(I,    LS_LINE(PC_OBJECT(I)),
908                                    PC_MNEMONIC(PC_OPCODE(I)))
909                                   (P'999999',X(3),A,X(2),A);
910             WHEN(PC_FORMAT_2)
911                 PUT SKIP EDIT(I,   PC_MNEMONIC(PC_OPCODE(I)),
912                                    PC_OBJECT(I))
913                                   (P'999999',X(13),A,X(2),A);
```

MACRO SOURCE2 LISTING

```
914             WHEN(PC_FORMAT_3)
915                 PUT SKIP EDIT(I,   PC_MNEMONIC(PC_OPCODE(I)),
916                                    STRING_VAL(PC_OBJECT(I)))
917                                   (P'999999',X(13),A,X(2),A);
918             WHEN(PC_FORMAT_4)
919                 PUT SKIP EDIT(I,   PC_MNEMONIC(PC_OPCODE(I)),
920                                    ' ')
921                                   (P'999999',X(13),A,X(2),A);
922             WHEN(PC_FORMAT_5)
923                 PUT SKIP EDIT(I,   PC_MNEMONIC(PC_OPCODE(I)),
924                                    PC_OBJECT(I))
925                                   (P'999999',X(13),A,X(2),F(5));
926             OTHERWISE
927                PUT EDIT('**** FATAL ERROR IN COMPILER *****',
928                         '**** INVALID VALUE FOR PC_OPCODE ',
929                         PC_MNEMONIC(PC_OPCODE(I)))
930                         (SKIP(2),A,SKIP,A,A);
931                STOP;
932             ENDSELECT
933           END;
934           LAST_PCODE_PRINTED = PC_MAX;
935        END;
936
937     END PRINT_PCODES;
938
939     SKIP_BLANKS:PROC;
940     /*********************************************************************
941     *                                                                    *
942     *                                                                    *
943     * NESTING:COMPILE                                                    *
944     *********************************************************************/
945        DECLARE CONTINUE_SCAN       BIT(1)    ALIGNED;
946        DECLARE I                   FIXED BIN ALIGNED;
947
948        CONTINUE_SCAN=TRUE;
949        DO I=STMT_CH TO STMT_RIGHT WHILE(CONTINUE_SCAN);
950           IF SUBSTR(STMT,I,1)=' ' THEN ;
951           ELSE
952           DO;
953              STMT_CH=I;
954              CONTINUE_SCAN=FALSE;
955           END;
956        END;
957        IF CONTINUE_SCAN THEN      /* NO NON BLANK FOUND */
958           STMT_CH=STMT_RIGHT+1;
```

MACRO SOURCE2 LISTING

```
 959
 960    END SKIP_BLANKS;
 961
 962    PRINT_ERR:PROC(I,MSG);
 963    /*****************************************************************
 964    *                                                                *
 965    *   PRINTS ALL ERROR MESSAGE FOR THE COMPILE PHASE               *
 966    *                                                                *
 967    *                                                                *
 968    * NESTING:COMPILE                                                *
 969    *****************************************************************/
 970       DECLARE I                   FIXED BINARY ALIGNED;
 971       DECLARE MSG                 CHAR(*);
 972       PUT SKIP EDIT('*****','^',MSG)
 973                    (A,COLUMN(24+I),A,SKIP,COLUMN(11),A);
 974       ERROR_COUNT=ERROR_COUNT+1;
 975       TERMINATE_SCAN=TRUE;
 976    END PRINT_ERR;
 977
 978    LOOKUP_SYMBOL_TABLE:PROC(V) RETURNS(FIXED BINARY);
 979    /*****************************************************************
 980    *                                                                *
 981    *  LOOKS UP SYMBOLS IN THE SYMBOL TABLE.  IF NOT FOUND, IT ADDS  *
 982    *  IT AND DETERMINE WHAT TYPE OF SYMBOL IT IS.                   *
 983    *  IF A ZERO IS RETURNED, THERE WAS AN ERROR.  OTHERWISE THE     *
 984    *  SUBSCRIPT OF THE ITEM "V" IS RETURNED.                        *
 985    *                                                                *
 986    * NESTING:COMPILE                                                *
 987    *****************************************************************/
 988
 989       DECLARE V                   CHAR(10),
 990               OPT                 FIXED BINARY ALIGNED;
 991       DECLARE I                   FIXED BINARY ALIGNED;
 992       DECLARE STR_IND             FIXED BINARY ALIGNED;
 993
 994       ON CONVERSION
 995       BEGIN;
 996          CONTINUE_SCAN=FALSE;
 997          ONCHAR='0';
 998       END;
 999
1000       DO I=1 TO SS_MAX;
1001          IF SYMBOL(I)=V THEN
1002          DO;
1003             IF SYM_TYPE(I)=SS_DIM_VAR &
```

MACRO SOURCE2 LISTING

```
1004                WORD=KW_DIM THEN
1005                   CALL PRINT_ERR(I,'CANNOT REDIM VARIABLE');
1006             RETURN(I);
1007          END;
1008       END; /* OF DO */
1009
1010       IF SS_MAX=HBOUND(SYMBOL,1) THEN
1011       DO;
1012          CALL PRINT_ERR(10,'SYMBOL TABLE OVERFLOW');
1013          RETURN(0);
1014       END;
1015
1016       SS_MAX=SS_MAX+1;
1017       SYMBOL(SS_MAX)=V;
1018       SYM_DIM_MAX(SS_MAX)=0;
1019       STRING_VAL(SS_MAX)='*';
1020       IF SUBSTR(V,1,1) >= 'A' & SUBSTR(V,1,1) <= 'Z' THEN
1021       DO;
1022          STR_IND=VERIFY(V,VALID_VAR_CHARS);
1023          IF STR_IND > 0 THEN
1024          DO;
1025             IF SUBSTR(V,STR_IND,1)='$' THEN
1026             DO;
1027                SYM_VALUE(SS_MAX)=0.0;
1028                IF SUBSTR(V,1,4)='STR$' THEN
1029                   SYM_TYPE(SS_MAX)=SS_STRCON;
1030                ELSE
1031                   IF WORD=KW_DIM THEN
1032                      SYM_TYPE(SS_MAX)=SS_STRDIM;
1033                   ELSE
1034                      SYM_TYPE(SS_MAX)=SS_STRVAR;
1035             END;
1036             ELSE
1037             DO;
1038                CALL PRINT_ERR(I,'INVALID VARIABLE NAME');
1039                RETURN(0);
1040             END;
1041          END;
1042          ELSE
1043          DO;
1044             SYM_VALUE(SS_MAX)=0.0;
1045             IF WORD=KW_DIM THEN
1046                 SYM_TYPE(SS_MAX)=SS_DIM_VAR;
1047             ELSE
1048                 SYM_TYPE(SS_MAX)=SS_VAR;
```

MACRO SOURCE2 LISTING

```
1049          END;
1050       END;
1051       ELSE
1052       DO;
1053          IF VERIFY(V,'0123456789+-.E ') > 0 THEN
1054          DO;
1055             CALL PRINT_ERR(I,'INVALID CONSTANT '||V);
1056             RETURN(0);
1057          END;
1058          ELSE
1059          DO;
1060             SYM_VALUE(SS_MAX)=V;
1061             SYM_TYPE(SS_MAX)=SS_CONST;
1062          END;
1063       END;
1064       RETURN(SS_MAX);
1065
1066    END LOOKUP_SYMBOL_TABLE;
1067
1068    ADD_PCODE:PROC(PCODE,OFFSET);
1069    /*********************************************************************
1070    *                                                                    *
1071    *   ADDS CODES TO THE PSEUDO MACHINE CODE TABLE                      *
1072    *                                                                    *
1073    * NESTING:COMPILE                                                   *
1074    *********************************************************************/
1075
1076       DECLARE PCODE        FIXED BINARY ALIGNED,
1077               OFFSET       FIXED BINARY ALIGNED;
1078
1079       IF PC_MAX+1>HBOUND(PC_OPCODE,1) THEN
1080       DO;
1081          PUT SKIP(2) EDIT('**** FATAL ERROR-PCODE TABLE OVERFLOW *****')
1082                          (A);
1083          STOP;
1084       END;
1085
1086       PC_MAX=PC_MAX+1;
1087       PC_OPCODE(PC_MAX)=PCODE;
1088       PC_OBJECT(PC_MAX)=OFFSET;
1089
1090    END ADD_PCODE;
1091
1092    GET_STMT_NUM:PROC(DEFINITION);
1093    /*********************************************************************
```

MACRO SOURCE2 LISTING

```
1094    *                                                                     *
1095    *   EXTRACT THE STATMENT NUMBER.  MAKE SURE IT IS NUMERIC AND IN      *
1096    *   SEQUENCE.  IF OK, ADD IT TO THE LINE STACK                        *
1097    *                                                                     *
1098    * NESTING:COMPILE                                                     *
1099    **********************************************************************/
1100       DECLARE DEFINITION          BIT(1) ALIGNED;
1101       DECLARE I                   FIXED BINARY ALIGNED;
1102       DECLARE CONTINUE_SCAN       BIT(1) ALIGNED;
1103       DECLARE CH                  CHAR(1);
1104       DECLARE LN                  CHAR(6) VARYING;
1105       DECLARE LINE_NUM            FIXED DECIMAL(5,0);
1106
1107       LN='';
1108       TMP_CNT=0;
1109       CONTINUE_SCAN=TRUE;
1110       DO I=STMT_CH TO STMT_RIGHT WHILE(CONTINUE_SCAN);
1111          CH=SUBSTR(STMT,I,1);
1112          IF CH=' ' THEN CONTINUE_SCAN=FALSE;
1113          ELSE
1114             IF CH < '0' | CH >'9' THEN
1115             DO;
1116                CONTINUE_SCAN=FALSE;
1117                CALL PRINT_ERR(I,'INVALID LINE NUMBER');
1118             END;
1119             ELSE
1120             DO;
1121                LN=LN||CH;
1122                IF LENGTH(LN)>5 THEN
1123                DO;
1124                   CONTINUE_SCAN=FALSE;
1125                   CALL PRINT_ERR(I,'LINE NUMBER TOO LONG');
1126                END;
1127             END;
1128       END;
1129
1130       IF DEFINITION=TRUE THEN
1131       DO;
1132          LINE_NUM=LN;
1133          IF LINE_NUM=LAST_LINE_NUM THEN
1134          DO;
1135             CONTINUE_SCAN=FALSE;
1136             CALL PRINT_ERR(STMT_CH,'DUPLICATE LINE NUMBER');
1137          END;
1138          ELSE
```

MACRO SOURCE2 LISTING

```
1139            IF LINE_NUM<LAST_LINE_NUM THEN
1140            DO;
1141               CONTINUE_SCAN=FALSE;
1142               CALL PRINT_ERR(STMT_CH,'LINE NUMBER OUT OF SEQUENCE');
1143            END;
1144            ELSE
1145            DO;
1146               IF LS_MAX=HBOUND(LS_LINE,1) THEN
1147                  CALL PRINT_ERR(STMT_CH,'TOO MANY LINE NUMBERS');
1148            END;
1149         LAST_LINE_NUM=LINE_NUM;
1150      END;
1151      ELSE
1152         REF_LINE_NUM=LN;
1153
1154      STMT_CH=I;
1155
1156   END GET_STMT_NUM;
1157
1158   GET_KEYWORD:PROC;
1159   /*****************************************************************
1160   *                                                               *
1161   *   EXTRACT THE STATMENT KEYWORD AND VALIDATE IT                *
1162   *                                                               *
1163   * NESTING:COMPILE                                               *
1164   *****************************************************************/
1165      DECLARE (I,J)              FIXED BINARY ALIGNED;
1166      DECLARE CONTINUE_SCAN      BIT(1) ALIGNED;
1167      DECLARE CH                 CHAR(1);
1168      DECLARE KW                 CHAR(9) VARYING;
1169
1170      CALL SKIP_BLANKS;
1171      IF STMT_CH=STMT_RIGHT THEN
1172      DO;
1173         CALL PRINT_ERR(STMT_CH,'BLANK LINE?');
1174         RETURN;
1175      END;
1176
1177      KW='';
1178      CONTINUE_SCAN=TRUE;
1179      DO I=STMT_CH TO STMT_RIGHT WHILE(CONTINUE_SCAN);
1180         CH=SUBSTR(STMT,I,1);
1181         IF CH=' ' THEN
1182         DO;
1183            IF LENGTH(KW)=2 THEN
```

MACRO SOURCE2 LISTING

```
1184             DO;
1185                IF KW='GO' THEN ;
1186                ELSE
1187                   CONTINUE_SCAN=FALSE;
1188             END;
1189             ELSE
1190                CONTINUE_SCAN=FALSE;
1191          END;
1192          ELSE
1193          DO;
1194             KW=KW||CH;
1195             IF LENGTH(KW)>9 THEN
1196             DO;
1197                CONTINUE_SCAN=FALSE;
1198                CALL PRINT_ERR(I,'KEYWORD TOO LONG');
1199             END;
1200          END;
1201       END;
1202
1203       WORD=KW;
1204
1205       CONTINUE_SCAN=TRUE;
1206       DO J=1 TO HBOUND(KEY_WORDS,1) WHILE(CONTINUE_SCAN);
1207          IF WORD=KEY_WORDS(J) THEN
1208              CONTINUE_SCAN=FALSE;
1209       END;
1210       IF CONTINUE_SCAN THEN
1211       DO;
1212          CALL PRINT_ERR(STMT_CH,'INVALID KEYWORD');
1213       END;
1214
1215       STMT_CH=I;
1216
1217    END GET_KEYWORD;
1218
1219    PROCESS_KEYWORD:PROC;
1220    /*********************************************************************
1221    *                                                                    *
1222    *   SYNTAX CHECK AND COMPILE STATEMENTS                              *
1223    *                                                                    *
1224    * NESTING:COMPILE                                                    *
1225    *********************************************************************/
1226       DECLARE I                  FIXED BINARY ALIGNED;
1227       DECLARE ERR_PTR            FIXED BINARY ALIGNED;
1228       DECLARE CONTINUE_SCAN      BIT(1) ALIGNED;
```

MACRO SOURCE2 LISTING

```
1229
1230       SELECT(WORD)
1231       WHEN(KW_REM)             /*  REMARKS - NOTHING TO DO! */
1232       WHEN(KW_END)
1233          CALL SKIP_BLANKS;
1234          IF STMT_CH>STMT_RIGHT THEN
1235              CALL ADD_PCODE(PC_OPCODE_END,ZERO);
1236          ELSE
1237             CALL PRINT_ERR(STMT_CH,
1238                  'INVALID SYNTAX - EXPECTING BLANKS AFTER END');
1239       WHEN(KW_STOP)
1240          CALL SKIP_BLANKS;
1241          IF STMT_CH>STMT_RIGHT THEN
1242              CALL ADD_PCODE(PC_OPCODE_STP,ZERO);
1243          ELSE
1244             CALL PRINT_ERR(STMT_CH,
1245                  'INVALID SYNTAX - EXPECTING BLANKS AFTER STOP');
1246       WHEN(KW_RETURN)
1247          CALL SKIP_BLANKS;
1248          IF STMT_CH>STMT_RIGHT THEN
1249              CALL ADD_PCODE(PC_OPCODE_RET,ZERO);
1250          ELSE
1251              CALL PRINT_ERR(STMT_CH,
1252                   'INVALID SYNTAX - EXPECTING BLANKS AFTER RETURN');
1253       WHEN(KW_GOTO)
1254           CALL SKIP_BLANKS;
1255           IF STMT_CH>STMT_RIGHT THEN
1256              CALL PRINT_ERR(STMT_CH,
1257                   'INVALID SYNTAX - EXPECTING LINE NUMBER AFTER GOTO');
1258           ELSE
1259              CALL PROCESS_GOTO;
1260       WHEN(KW_GOSUB)
1261           CALL SKIP_BLANKS;
1262           IF STMT_CH>STMT_RIGHT THEN
1263              CALL PRINT_ERR(STMT_CH,
1264                   'INVALID SYNTAX - EXPECTING LINE NUMBER AFTER GOSUB');
1265           ELSE
1266              CALL PROCESS_GOSUB;
1267       WHEN(KW_DATA)
1268          CALL SKIP_BLANKS;
1269          IF STMT_CH>STMT_RIGHT THEN
1270             CALL PRINT_ERR(STMT_CH,
1271                  'INVALID SYNTAX - EXPECTING DATA ELEMENTS');
1272          ELSE
1273             CALL EXTRACT_DATA;
```

MACRO SOURCE2 LISTING

```
1274       WHEN(KW_LET)
1275          CALL SKIP_BLANKS;
1276          IF STMT_CH>STMT_RIGHT THEN
1277             CALL PRINT_ERR(STMT_CH,
1278                   'INVALID SYNTAX - EXPECTING LET STATEMENT');
1279          ELSE
1280             CALL PROCESS_LET;
1281       WHEN(KW_DEF)
1282          CALL SKIP_BLANKS;
1283          IF STMT_CH>STMT_RIGHT THEN
1284             CALL PRINT_ERR(STMT_CH,
1285                            'INVALID SYNTAX - EXPECTING FUNCTION');
1286          ELSE
1287             CALL PROCESS_DEF;
1288       WHEN(KW_READ)
1289          CALL SKIP_BLANKS;
1290          IF STMT_CH>STMT_RIGHT THEN
1291             CALL PRINT_ERR(STMT_CH,
1292                  'INVALID SYNTAX - EXPECTING VARIABLE(S) AFTER READ');
1293          ELSE
1294             CALL PROCESS_READ;
1295       WHEN(KW_PRINT)
1296          CALL SKIP_BLANKS;
1297          CALL PROCESS_PRINT;
1298       WHEN(KW_IF)
1299          CALL SKIP_BLANKS;
1300          IF STMT_CH>STMT_RIGHT THEN
1301             CALL PRINT_ERR(STMT_CH,
1302                  'INVALID SYNTAX - EXPECTING COMPARISON FOR IF');
1303          ELSE
1304             CALL PROCESS_IF;
1305       WHEN(KW_FOR)
1306          CALL SKIP_BLANKS;
1307          IF STMT_CH>STMT_RIGHT THEN
1308             CALL PRINT_ERR(STMT_CH,
1309                  'INVALID SYNTAX - INCOMPLETE FOR STATEMENT');
1310          ELSE
1311             CALL PROCESS_FOR;
1312       WHEN(KW_NEXT)
1313          ERR_PTR=STMT_CH;
1314          CALL SKIP_BLANKS;
1315          IF STMT_CH>STMT_RIGHT THEN
1316             CALL PRINT_ERR(ERR_PTR,
1317                  'INVALID SYNTAX - EXPECTING VARIABLE AFTER NEXT');
1318          ELSE
```

MACRO SOURCE2 LISTING

```
1319            CALL PROCESS_NEXT;
1320         WHEN(KW_RESTORE)
1321            CALL SKIP_BLANKS;
1322            IF STMT_CH>STMT_RIGHT THEN
1323               CALL ADD_PCODE(PC_OPCODE_RST,ZERO);
1324            ELSE
1325               CALL PRINT_ERR(STMT_CH,
1326                     'INVALID SYNTAX - EXPECTING BLANKS AFTER RESTORE');
1327         WHEN(KW_DIM)
1328            ERR_PTR=STMT_CH;
1329            CALL SKIP_BLANKS;
1330            IF STMT_CH>STMT_RIGHT THEN
1331               CALL PRINT_ERR(ERR_PTR,
1332                    'INVALID SYNTAX - EXPECTING DIM VARIABLE');
1333            ELSE
1334               CALL PROCESS_DIM;
1335         OTHERWISE
1336            CALL PRINT_ERR(STMT_CH,
1337                              '***INVALID KEYWORD '|| WORD ||'****');
1338         ENDSELECT
1339
1340      END PROCESS_KEYWORD;
1341
1342      PROCESS_GOTO:PROC;
1343      /*****************************************************************
1344      *                                                                *
1345      *   EXTRACT AND VERIFY LINE NUMBER FROM THE GOTO STATEMENT.      *
1346      *   IF THE NUMBER IS CLEAN, ADD A PC B  TO THE CODE              *
1347      *                                                                *
1348      * NESTING:COMPILE                                                *
1349      *****************************************************************/
1350
1351         CALL GET_STMT_NUM(FALSE);
1352         CALL SKIP_BLANKS;
1353         IF STMT_CH>STMT_RIGHT THEN
1354         DO;
1355            I=REF_LINE_NUM;
1356            CALL ADD_PCODE(PC_OPCODE_B,I);
1357         END;
1358         ELSE
1359            CALL PRINT_ERR(STMT_CH,
1360                'INVALID SYNTAX - EXPECTING BLANKS AFTER LINE');
1361
1362      END PROCESS_GOTO;
1363
```

MACRO SOURCE2 LISTING

```
1364    PROCESS_GOSUB:PROC;
1365    /*****************************************************************
1366    *                                                                *
1367    *   EXTRACT AND VERIFY LINE NUMBER FROM THE GOSUB STATEMENT.     *
1368    *   IF THE NUMBER IS CLEAN, ADD A PC BAL TO THE CODE             *
1369    *                                                                *
1370    * NESTING:COMPILE                                                *
1371    *****************************************************************/
1372
1373       CALL GET_STMT_NUM(FALSE);
1374       CALL SKIP_BLANKS;
1375       IF STMT_CH>STMT_RIGHT THEN
1376       DO;
1377          I=REF_LINE_NUM;
1378          CALL ADD_PCODE(PC_OPCODE_BAL,I);
1379       END;
1380       ELSE
1381          CALL PRINT_ERR(STMT_CH,
1382              'INVALID SYNTAX - EXPECTING BLANKS AFTER LINE');
1383
1384    END PROCESS_GOSUB;
1385
1386    EXTRACT_DATA:PROC;
1387    /*****************************************************************
1388    *                                                                *
1389    *   EXTRACT AND VERIFY NUMBERS FROM THE DATA STATEMENTS.         *
1390    *   IF THE NUMBER IS CLEAN, ADD IT TO THE DATA STACK             *
1391    *                                                                *
1392    * NESTING:COMPILE                                                *
1393    *****************************************************************/
1394       DECLARE I                   FIXED BINARY ALIGNED;
1395       DECLARE CH                  CHAR(1);
1396       DECLARE VAL                 CHAR(80) VARYING;
1397       DECLARE HOLD_VAL            CHAR(80) VARYING;
1398       DECLARE NUM_VAL             FLOAT BINARY;
1399       DECLARE CONTINUE_SCAN       BIT(1) ALIGNED;
1400       DECLARE IN_STR              BIT(1) ALIGNED;
1401
1402       ON CONVERSION
1403       BEGIN;
1404          CONTINUE_SCAN=FALSE;
1405          ONCHAR='0';
1406       END;
1407       CONTINUE_SCAN=TRUE;
1408       IN_STR=FALSE;
```

MACRO SOURCE2 LISTING

```
1409       VAL='';
1410       DO I=STMT_CH TO STMT_RIGHT;
1411          CH=SUBSTR(STMT,I,1);
1412          SELECT(TRUE)
1413          WHEN(IN_STR)
1414             VAL=VAL||CH;
1415             IF CH=QUOTE_1 THEN
1416                IN_STR=FALSE;
1417          WHEN(CH=QUOTE_1)
1418             VAL=VAL||CH;
1419             IN_STR=TRUE;
1420          WHEN(CH=' ')
1421          WHEN(CH=',')
1422             CALL EXTRACT_DATA_ITEM;
1423             CONTINUE_SCAN=TRUE;
1424             VAL='';
1425          OTHERWISE
1426             VAL=VAL||CH;
1427          ENDSELECT
1428       END;
1429       CALL EXTRACT_DATA_ITEM;
1430    EXTRACT_DATA_ITEM:PROC;
1431       DECLARE TMP_VAR              CHAR(10);
1432       DECLARE OFFSET               FIXED BINARY ALIGNED;
1433       OFFSET=0;
1434       NUM_VAL=0.0;
1435       HOLD_VAL=VAL;
1436       IF SUBSTR(VAL,1,1)=QUOTE_1 THEN
1437       DO;
1438          IF SUBSTR(VAL,LENGTH(VAL),1)=QUOTE_1 THEN
1439          DO;
1440             TMP_VAR='STR$'||STR_CNT;
1441             STR_CNT=STR_CNT+1;
1442             OFFSET=LOOKUP_SYMBOL_TABLE(TMP_VAR);
1443             IF STACK_PRINT_DEBUG THEN
1444                PUT DATA(VAL,TMP_VAR,OFFSET);
1445             IF LENGTH(VAL)>2 THEN
1446                STRING_VAL(OFFSET)=SUBSTR(VAL,2,LENGTH(VAL)-2);
1447             ELSE
1448                STRING_VAL(OFFSET)='';
1449          END;
1450          ELSE
1451          DO;
1452             CONTINUE_SCAN=FALSE;
1453          END;
```

MACRO SOURCE2 LISTING

```
1454       END;
1455       ELSE
1456       DO;
1457          IF VERIFY(V,'0123456789+-.E ') > 0 THEN
1458             CONTINUE_SCAN=FALSE;
1459          ELSE
1460             NUM_VAL=VAL;
1461       END;
1462       IF CONTINUE_SCAN=FALSE THEN
1463           CALL PRINT_ERR(I,'ILLEGAL CONSTANT IN DATA STATEMENT '
1464                                                 || HOLD_VAL);
1465       ELSE
1466       DO;
1467           IF DS_MAX=HBOUND(DS_ITEM,1) THEN
1468              CALL PRINT_ERR(I,'DATA STACK FULL');
1469           ELSE
1470           DO;
1471              DS_MAX=DS_MAX+1;
1472              DS_STR(DS_MAX)=OFFSET;
1473              DS_ITEM(DS_MAX)=NUM_VAL;
1474           END;
1475       END;
1476    END EXTRACT_DATA_ITEM;
1477    END EXTRACT_DATA;
1478
1479    PROCESS_READ:PROC;
1480    /*****************************************************************
1481    *                                                                *
1482    *   PROCESS READ                                                 *
1483    *                                                                *
1484    * NESTING:COMPILE                                                *
1485    *****************************************************************/
1486       DECLARE I                   FIXED BINARY ALIGNED;
1487       DECLARE CH                  CHAR(1);
1488       DECLARE VAR                 CHAR(10);
1489       DECLARE LEFT_SIDE           CHAR(80) VARYING;
1490       DECLARE NO_COMMA            BIT(1) ALIGNED;
1491
1492     /*  EXTRACT THE RECEIVING FIELDS */
1493
1494       LEFT_SIDE='';
1495       NO_COMMA=TRUE;
1496       DO I=STMT_CH TO STMT_RIGHT;
1497         CH=SUBSTR(STMT,I,1);
1498         IF CH=',' THEN
```

MACRO SOURCE2 LISTING

```
1499         DO;
1500            NO_COMMA=FALSE;
1501            CALL PROCESS_READ_VAR;
1502         END;
1503         ELSE
1504            IF CH=' ' THEN;
1505            ELSE
1506               LEFT_SIDE=LEFT_SIDE||CH;
1507       END;
1508       IF LENGTH(LEFT_SIDE)>0 THEN
1509           CALL PROCESS_READ_VAR;
1510
1511    PROCESS_READ_VAR:PROC;
1512
1513            IF LENGTH(LEFT_SIDE)>10 THEN
1514                CALL PRINT_ERR(STMT_CH,'VARIABLE TOO LONG');
1515            ELSE
1516            DO;
1517                CALL BALANCE_STMT(LEFT_SIDE);
1518                IF TERMINATE_SCAN THEN RETURN;
1519                LEFT_SIDE='('||LEFT_SIDE||')';
1520                CALL PARSE_EXP(LEFT_SIDE,EXP_CALC);
1521                IF PC_MAX > 0 THEN  /*  ERROR IF A TMP IS FOUND */
1522                DO;
1523                IF PC_OPCODE(PC_MAX)=PC_OPCODE_STA THEN
1524                   IF SUBSTR(SYMBOL(PC_OBJECT(PC_MAX)),1,3)='TMP' THEN
1525                      CALL PRINT_ERR(STMT_CH,'EXPRESSION NOT ALLOWED');
1526                IF PC_OPCODE(PC_MAX)=PC_OPCODE_LDA THEN
1527                   IF SUBSTR(SYMBOL(PC_OBJECT(PC_MAX)),1,3)='TMP' THEN
1528                      CALL PRINT_ERR(STMT_CH,'EXPRESSION NOT ALLOWED');
1529                   ELSE
1530                      PC_OPCODE(PC_MAX)=PC_OPCODE_RDV;
1531                END;
1532            END;
1533            LEFT_SIDE='';
1534
1535    END PROCESS_READ_VAR;
1536    END PROCESS_READ;
1537
1538    PROCESS_PRINT:PROC;
1539
1540    /*****************************************************************
1541    *                                                               *
1542    *   PROCESS PRINT                                               *
1543    *                                                               *
```

MACRO SOURCE2 LISTING

```
1544    *   PARSE THE STATEMENT INTO OBJECTS - NUMERIC EXPRESSION OR        *
1545    *   STRING LITTERALS.                                               *
1546    *   ALSO DECIDES IF A LINE FEED SHOULD BE ISSUED OR NOT BASED ON    *
1547    *   ON DANGLING COMMA                                               *
1548    *                                                                   *
1549    * NESTING:COMPILE                                                   *
1550    *********************************************************************/
1551       DECLARE (I,LAST_NB)         FIXED BINARY ALIGNED;
1552       DECLARE CH                  CHAR(1);
1553       DECLARE VAR                 CHAR(10);
1554       DECLARE LEFT_SIDE           CHAR(80) VARYING;
1555       DECLARE NO_COMMA            BIT(1) ALIGNED;
1556       DECLARE IN_STR              BIT(1) ALIGNED;
1557
1558     /*  EXTRACT THE PRINT OBJECT */
1559
1560       LEFT_SIDE='';
1561       LAST_NB=0;
1562       NO_COMMA=TRUE;
1563       DANGLE=FALSE;
1564       IN_STR=FALSE;
1565
1566       IF STMT_CH>STMT_RIGHT THEN     /* PRINT A LINE FEED FOR */
1567       DO;                            /* KEYWORD PRINT ONLY */
1568          I=1;
1569          CALL ADD_PCODE(PC_OPCODE_PCT,PCT_LFEED);
1570          RETURN;
1571       END;
1572
1573       I=STMT_CH;
1574       DO WHILE(I <= STMT_RIGHT);
1575         CH=SUBSTR(STMT,I,1);
1576         IF IN_STR THEN
1577         DO;
1578            IF CH=QUOTE_1 THEN
1579               IN_STR=FALSE;
1580            LEFT_SIDE=LEFT_SIDE||CH;
1581         END;
1582         ELSE
1583         DO;
1584            IF CH=QUOTE_1 THEN
1585            DO;
1586               IN_STR=TRUE;
1587               LEFT_SIDE=LEFT_SIDE||CH;
1588            END;
```

MACRO SOURCE2 LISTING

```
1589         END;
1590
1591         IF IN_STR THEN;
1592         ELSE
1593         DO;
1594            IF CH=' ' THEN
1595            /* PUT SKIP DATA(LAST_NB,I,LEFT_SIDE) PRINT USING ?*/;
1596            ELSE LAST_NB=I;
1597            IF CH=',' | CH=';' THEN
1598            DO;
1599               NO_COMMA=FALSE;
1600               IF SUBSTR(LEFT_SIDE,1,1)=QUOTE_1 THEN
1601                   CALL PROCESS_PRINT_STR;
1602               ELSE
1603                   CALL PROCESS_PRINT_VAR;
1604               IF CH=',' THEN
1605                   CALL ADD_PCODE(PC_OPCODE_PCT,PCT_TAB);
1606               ELSE
1607                   CALL ADD_PCODE(PC_OPCODE_PCT,PCT_NOTAB);
1608            END;
1609            ELSE
1610               IF CH=' ' | CH=QUOTE_1 THEN;
1611               ELSE
1612                  LEFT_SIDE=LEFT_SIDE||CH;
1613         END;
1614       I=I+1;
1615       END;
1616
1617       IF LENGTH(LEFT_SIDE)>0 THEN
1618           IF SUBSTR(LEFT_SIDE,1,1)=QUOTE_1 THEN
1619               CALL PROCESS_PRINT_STR;
1620           ELSE
1621               CALL PROCESS_PRINT_VAR;
1622
1623       IF LAST_NB > 0 THEN
1624       DO;
1625          IF SUBSTR(STMT,LAST_NB,1)=',' |
1626             SUBSTR(STMT,LAST_NB,1)=';' THEN;
1627          ELSE
1628             CALL ADD_PCODE(PC_OPCODE_PCT,PCT_LFEED);
1629       END;
1630       ELSE
1631             CALL PRINT_ERR(STMT_CH,'INVALID PRINT SYNTAX');
1632
1633    PROCESS_PRINT_VAR:PROC;
```

MACRO SOURCE2 LISTING

```
1634
1635       CALL BALANCE_STMT(LEFT_SIDE);
1636       IF TERMINATE_SCAN THEN
1637           RETURN;
1638
1639       LEFT_SIDE='('||LEFT_SIDE||')';
1640       CALL PARSE_EXP(LEFT_SIDE,EXP_CALC);
1641       IF PC_MAX > 0 THEN  /*  ERROR IF A TMP IS FOUND */
1642       DO;
1643          IF PC_OPCODE(PC_MAX)=PC_OPCODE_LDA THEN
1644          DO;
1645             IF PC_FORMAT(PC_OPCODE(PC_MAX-2))=0 THEN
1646             DO;
1647                IF SYMBOL(PC_OBJECT(PC_MAX-2))='TAB       ' THEN
1648                     PC_OPCODE(PC_MAX)=PC_OPCODE_PTB;
1649                ELSE
1650                     PC_OPCODE(PC_MAX)=PC_OPCODE_PRV;
1651             END;
1652             ELSE
1653                  PC_OPCODE(PC_MAX)=PC_OPCODE_PRV;
1654          END;
1655          IF PC_OPCODE(PC_MAX)=PC_OPCODE_STA THEN
1656          DO;
1657             IF PC_FORMAT(PC_OBJECT(PC_MAX-2))=0 THEN
1658             DO;
1659                IF SYMBOL(PC_OBJECT(PC_MAX-2))='TAB       ' THEN;
1660                ELSE
1661                   CALL ADD_PCODE(PC_OPCODE_PRV,PC_OBJECT(PC_MAX));
1662             END;
1663             ELSE
1664                CALL ADD_PCODE(PC_OPCODE_PRV,PC_OBJECT(PC_MAX));
1665          END;
1666       END;
1667
1668       LEFT_SIDE='';
1669
1670    END PROCESS_PRINT_VAR;
1671
1672    PROCESS_PRINT_STR:PROC;
1673
1674       DECLARE  (I,TICS)             FIXED BINARY ALIGNED;
1675       TICS=0;
1676       DO I = 1 TO LENGTH(LEFT_SIDE);
1677           IF SUBSTR(LEFT_SIDE,I,1)=QUOTE_1 THEN
1678               TICS=TICS+1;
```

MACRO SOURCE2 LISTING

```
1679       END;
1680       IF MOD(TICS,2)=1 THEN
1681       DO;
1682          CALL PRINT_ERR(STMT_CH,'UNBALANCED STRING');
1683          RETURN;
1684       END;
1685
1686       IF SS_MAX>=HBOUND(STRING_VAL,1) THEN
1687       DO;
1688          CALL PRINT_ERR(STMT_CH,'STRING CONSTANT TABLE FULL');
1689          RETURN;
1690       END;
1691                      /*  STRIP OFF THE QUOTE MARKS AND REDUCE
1692                          DOUBLE QUOTES TO 1 QUOTE BEFORE SAVING */
1693
1694       LEFT_SIDE=SUBSTR(LEFT_SIDE,2,LENGTH(LEFT_SIDE)-2);
1695       I = 1;
1696       DO WHILE(I<LENGTH(LEFT_SIDE));
1697          IF SUBSTR(LEFT_SIDE,I,2)=QUOTE_2 THEN
1698             LEFT_SIDE=SUBSTR(LEFT_SIDE,1,I)||SUBSTR(LEFT_SIDE,I+2);
1699          I=I+1;
1700       END;
1701       SS_CUR,SS_MAX=SS_MAX+1;
1702       SYMBOL(SS_CUR)='_PRS';
1703       STRING_VAL(SS_CUR)=LEFT_SIDE;
1704       SYM_TYPE(SS_CUR)=SS_STRCON;
1705       CALL ADD_PCODE(PC_OPCODE_PRS,SS_CUR);
1706       LEFT_SIDE='';
1707
1708    END PROCESS_PRINT_STR;
1709
1710    END PROCESS_PRINT;
1711
1712    PROCESS_IF:PROC;
1713    /*********************************************************************
1714    *                                                                    *
1715    *   PROCESS IF                                                      *
1716    *                                                                    *
1717    * NESTING:COMPILE                                                   *
1718    *********************************************************************/
1719       DECLARE I                   FIXED BINARY ALIGNED;
1720       DECLARE CH                  CHAR(1);
1721       DECLARE (LEFT_SIDE,RIGHT_SIDE)
1722                                   CHAR(80) VARYING;
1723       DECLARE NO_OPER             BIT(1) ALIGNED;
```

MACRO SOURCE2 LISTING

```
1724       DECLARE OPER               CHAR(2);
1725       DECLARE THEN_WORD          CHAR(4);
1726
1727     /*  EXTRACT THE LEFT FIELD */
1728
1729       LEFT_SIDE='';
1730       NO_OPER=TRUE;
1731       OPER='..';
1732       DO I=STMT_CH TO STMT_RIGHT WHILE(NO_OPER);
1733         CH=SUBSTR(STMT,I,1);
1734         IF CH='=' |
1735            CH='<' |
1736            CH='>' THEN
1737         DO;
1738            NO_OPER=FALSE;
1739            OPER=CH;
1740            RIGHT_SIDE=SUBSTR(STMT,I+1);
1741         END;
1742         ELSE
1743            IF CH=' ' THEN;
1744            ELSE
1745               LEFT_SIDE=LEFT_SIDE||CH;
1746       END;
1747
1748       STMT_CH=I;
1749       CH=SUBSTR(STMT,I,1);
1750       IF (OPER='= ' & CH='>') | (OPER='> ' & CH='=') THEN
1751       DO;
1752          OPER='>=';
1753          STMT_CH=I+1;
1754       END;
1755       ELSE
1756       IF (OPER='= ' & CH='<') | (OPER='< ' & CH='=') THEN
1757       DO;
1758          OPER='<=';
1759          STMT_CH=I+1;
1760       END;
1761       ELSE
1762       IF (OPER='< ' & CH='>') THEN
1763       DO;
1764          OPER='<>';
1765          STMT_CH=I+1;
1766       END;
1767       ELSE
1768       IF OPER='= ' | OPER='< ' | OPER='> ' THEN;
```

MACRO SOURCE2 LISTING

```
1769       ELSE
1770       DO;
1771          CALL PRINT_ERR(STMT_CH,'NO COMPARISON OPERATOR FOUND');
1772          RETURN;
1773       END;
1774
1775     /*  EXTRACT THE RIGHT FIELD */
1776
1777       RIGHT_SIDE='';
1778       NO_OPER=TRUE;
1779       DO I=STMT_CH TO STMT_RIGHT-3 WHILE(NO_OPER);
1780         CH=SUBSTR(STMT,I,1);
1781         THEN_WORD=SUBSTR(STMT,I,4);
1782         IF THEN_WORD='THEN' THEN
1783         DO;
1784            NO_OPER=FALSE;
1785         END;
1786         ELSE
1787            IF CH=' ' THEN;
1788            ELSE
1789               RIGHT_SIDE=RIGHT_SIDE||CH;
1790       END;
1791
1792       IF NO_OPER=TRUE THEN
1793       DO;
1794          CALL PRINT_ERR(STMT_CH,'THEN NOT FOUND');
1795          RETURN;
1796       END;
1797
1798       STMT_CH=I+4;
1799
1800       CALL SKIP_BLANKS;
1801       IF STMT_CH>STMT_RIGHT THEN
1802         CALL PRINT_ERR(STMT_CH,
1803                        'INVALID SYNTAX - EXPECTING LINE NUMBER');
1804       ELSE
1805       DO;
1806          CALL GET_STMT_NUM(FALSE);
1807
1808          CALL SKIP_BLANKS;
1809
1810          CALL BALANCE_STMT(LEFT_SIDE);
1811          IF TERMINATE_SCAN THEN
1812             RETURN;
1813          LEFT_SIDE='('||LEFT_SIDE||')';
```

MACRO SOURCE2 LISTING

```
1814          CALL PARSE_EXP(LEFT_SIDE,EXP_CALC);
1815          IF PC_MAX > 0 THEN  /*  ERROR IF A TMP IS FOUND */
1816          DO;
1817             IF PC_OPCODE(PC_MAX)=PC_OPCODE_LDA THEN
1818                PC_OPCODE(PC_MAX)=PC_OPCODE_LCA;
1819          END;
1820          CALL BALANCE_STMT(RIGHT_SIDE);
1821          IF TERMINATE_SCAN THEN RETURN;
1822          LEFT_SIDE='('||RIGHT_SIDE||')';
1823          CALL PARSE_EXP(LEFT_SIDE,EXP_CALC);
1824          IF PC_MAX > 0 THEN  /*  ERROR IF A TMP IS FOUND */
1825          DO;
1826             IF PC_OPCODE(PC_MAX)=PC_OPCODE_LDA THEN
1827                PC_OPCODE(PC_MAX)=PC_OPCODE_LCB;
1828          END;
1829          I=REF_LINE_NUM;
1830          SELECT(OPER)
1831             WHEN('= ')
1832                CALL ADD_PCODE(PC_OPCODE_BEQ,I);
1833             WHEN('<>')
1834                CALL ADD_PCODE(PC_OPCODE_BNE,I);
1835             WHEN('< ')
1836                CALL ADD_PCODE(PC_OPCODE_BLT,I);
1837             WHEN('> ')
1838                CALL ADD_PCODE(PC_OPCODE_BGT,I);
1839             WHEN('<=')
1840                CALL ADD_PCODE(PC_OPCODE_BLE,I);
1841             WHEN('>=')
1842                CALL ADD_PCODE(PC_OPCODE_BGE,I);
1843          OTHERWISE
1844             TERMINATE_SCAN=TRUE;
1845             PUT SKIP EDIT('**** INTERNAL COMPILER ERROR IF-01')
1846                           (A);
1847          ENDSELECT
1848
1849       END;
1850
1851    END PROCESS_IF;
1852
1853    PROCESS_FOR:PROC;
1854    /*****************************************************************
1855    *                                                               *
1856    *   PROCESS FOR                                                 *
1857    *                                                               *
1858    * NESTING:COMPILE                                               *
```

MACRO SOURCE2 LISTING

```
1859    ****************************************************************/
1860       DECLARE I                   FIXED BINARY ALIGNED;
1861       DECLARE OFFSET              FIXED BINARY ALIGNED;
1862       DECLARE CH                  CHAR(1);
1863       DECLARE (LEFT_SIDE,START_VAL,TO_VAL,STEP_VAL)
1864                                   CHAR(80) VARYING;
1865       DECLARE CTL_VAR             CHAR(10);
1866       DECLARE NO_OPER             BIT(1) ALIGNED;
1867       DECLARE STEP_WORD           CHAR(4);
1868       DECLARE TO_WORD             CHAR(2);
1869
1870     /*  EXTRACT THE CONTROL VARIABLE */
1871
1872       LEFT_SIDE,START_VAL,TO_VAL,STEP_VAL='';
1873       NO_OPER=TRUE;
1874
1875       DO I=STMT_CH TO STMT_RIGHT WHILE(NO_OPER);
1876         CH=SUBSTR(STMT,I,1);
1877         IF CH='=' THEN
1878         DO;
1879            NO_OPER=FALSE;
1880         END;
1881         ELSE
1882            IF CH=' ' THEN;
1883            ELSE
1884               LEFT_SIDE=LEFT_SIDE||CH;
1885       END;
1886
1887       STMT_CH=I;
1888
1889     /*  EXTRACT THE STARTING VALUE */
1890
1891       NO_OPER=TRUE;
1892       DO I=STMT_CH TO STMT_RIGHT-1 WHILE(NO_OPER);
1893         CH=SUBSTR(STMT,I,1);
1894         TO_WORD=SUBSTR(STMT,I,2);
1895         IF TO_WORD='TO' THEN
1896         DO;
1897            NO_OPER=FALSE;
1898         END;
1899         ELSE
1900            IF CH=' ' THEN;
1901            ELSE
1902               START_VAL=START_VAL||CH;
1903       END;
```

MACRO SOURCE2 LISTING

```
1904
1905       IF NO_OPER=TRUE THEN
1906       DO;
1907          CALL PRINT_ERR(STMT_CH,'TO NOT FOUND');
1908          RETURN;
1909       END;
1910
1911       STMT_CH=I+2;
1912
1913     /*  EXTRACT THE TO VALUE */
1914
1915       NO_OPER=TRUE;
1916       DO I=STMT_CH TO STMT_RIGHT-3 WHILE(NO_OPER);
1917         CH=SUBSTR(STMT,I,1);
1918         STEP_WORD=SUBSTR(STMT,I,4);
1919         IF STEP_WORD='STEP' THEN
1920         DO;
1921            NO_OPER=FALSE;
1922         END;
1923         ELSE
1924            IF CH=' ' THEN;
1925            ELSE
1926               TO_VAL=TO_VAL||CH;
1927       END;
1928
1929       IF NO_OPER=FALSE THEN
1930       DO;
1931          NO_OPER=TRUE;
1932          STMT_CH=I+4;
1933          DO I=STMT_CH TO STMT_RIGHT WHILE(NO_OPER);
1934            CH=SUBSTR(STMT,I,1);
1935            IF CH=' ' THEN;
1936            ELSE
1937               STEP_VAL=STEP_VAL||CH;
1938          END;
1939       END;
1940       ELSE
1941          STEP_VAL='1';
1942
1943       IF STMT_CH>STMT_RIGHT THEN
1944       DO;
1945         CALL PRINT_ERR(STMT_CH,
1946                        'INVALID SYNTAX - EXPECTING BLANKS');
1947         RETURN;
1948       END;
```

MACRO SOURCE2 LISTING

```
1949
1950       CTL_VAR=LEFT_SIDE;
1951       OFFSET=LOOKUP_SYMBOL_TABLE(CTL_VAR);
1952       IF SYM_TYPE(OFFSET)=SS_VAR THEN
1953           CALL ADD_PCODE(PC_OPCODE_FSU,OFFSET);
1954       ELSE
1955       DO;
1956          CALL PRINT_ERR(STMT_CH,'SIMPLE VARIABLE EXPECTED NOT '||
1957                                 CTL_VAR);
1958          RETURN;
1959       END;
1960
1961       CALL BALANCE_STMT(START_VAL);
1962       IF TERMINATE_SCAN THEN RETURN;
1963       START_VAL='('||START_VAL||')';
1964       CALL PARSE_EXP(START_VAL,EXP_CALC);
1965
1966       IF PC_OPCODE(PC_MAX)=PC_OPCODE_LDA THEN
1967          PC_OPCODE(PC_MAX)=PC_OPCODE_FIX;
1968       ELSE
1969          CALL ADD_PCODE(PC_OPCODE_FIX,OFFSET);
1970
1971       CALL BALANCE_STMT(TO_VAL);
1972       IF TERMINATE_SCAN THEN RETURN;
1973       TO_VAL='('||TO_VAL||')';
1974       CALL PARSE_EXP(TO_VAL,EXP_CALC);
1975
1976       IF PC_OPCODE(PC_MAX)=PC_OPCODE_LDA THEN
1977          PC_OPCODE(PC_MAX)=PC_OPCODE_FUL;
1978       ELSE
1979          CALL ADD_PCODE(PC_OPCODE_FUL,OFFSET);
1980
1981       CALL BALANCE_STMT(STEP_VAL);
1982       IF TERMINATE_SCAN THEN RETURN;
1983       STEP_VAL='('||STEP_VAL||')';
1984       CALL PARSE_EXP(STEP_VAL,EXP_CALC);
1985
1986       IF PC_OPCODE(PC_MAX)=PC_OPCODE_LDA THEN
1987          PC_OPCODE(PC_MAX)=PC_OPCODE_FST;
1988       ELSE
1989          CALL ADD_PCODE(PC_OPCODE_FST,OFFSET);
1990
1991    END PROCESS_FOR;
1992
1993    PROCESS_NEXT:PROC;
```

MACRO SOURCE2 LISTING

```
1994    /*****************************************************************
1995    *                                                                *
1996    *   PROCESS NEXT                                                 *
1997    *                                                                *
1998    * NESTING:COMPILE                                                *
1999    *****************************************************************/
2000       DECLARE I                   FIXED BINARY ALIGNED;
2001       DECLARE ERR_PTR             FIXED BINARY ALIGNED;
2002       DECLARE CH                  CHAR(1);
2003       DECLARE (LEFT_SIDE,START_VAL,TO_VAL,STEP_VAL)
2004                                   CHAR(80) VARYING;
2005       DECLARE NO_OPER             BIT(1) ALIGNED;
2006       DECLARE OFFSET              FIXED BINARY ALIGNED;
2007       DECLARE CTL_VAR             CHAR(10);
2008
2009     /*  EXTRACT THE CONTROL VARIABLE */
2010
2011       LEFT_SIDE='';
2012       NO_OPER=TRUE;
2013
2014       DO I=STMT_CH TO STMT_RIGHT WHILE(NO_OPER);
2015         CH=SUBSTR(STMT,I,1);
2016         IF CH=' ' THEN
2017            NO_OPER=FALSE;
2018         ELSE
2019            LEFT_SIDE=LEFT_SIDE||CH;
2020       END;
2021
2022       ERR_PTR=STMT_CH;
2023       STMT_CH=I;
2024
2025       CALL SKIP_BLANKS;
2026       IF STMT_CH>STMT_RIGHT THEN;
2027       ELSE
2028       DO;
2029         CALL PRINT_ERR(ERR_PTR,
2030                        'INVALID SYNTAX - EXPECTING BLANKS');
2031         RETURN;
2032       END;
2033       CTL_VAR=LEFT_SIDE;
2034       OFFSET=LOOKUP_SYMBOL_TABLE(CTL_VAR);
2035       IF SYM_TYPE(OFFSET)=SS_VAR THEN
2036           CALL ADD_PCODE(PC_OPCODE_FNX,OFFSET);
2037       ELSE
2038           CALL PRINT_ERR(ERR_PTR,'SIMPLE VARIABLE EXPECTED');
```

MACRO SOURCE2 LISTING

```
2039
2040    END PROCESS_NEXT;
2041
2042    PROCESS_LET:PROC;
2043    /*********************************************************************
2044    *                                                                    *
2045    *   PROCESS LET                                                      *
2046    *                                                                    *
2047    * NESTING:COMPILE                                                  *
2048    *********************************************************************/
2049       DECLARE I                   FIXED BINARY ALIGNED;
2050       DECLARE CH                  CHAR(1);
2051       DECLARE (LEFT_SIDE,RIGHT_SIDE)
2052                                   CHAR(80) VARYING;
2053       DECLARE NO_EQUAL            BIT(1) ALIGNED;
2054
2055       CH=SUBSTR(STMT,STMT_CH,1);
2056       IF CH<'A' | CH>'Z' THEN
2057          CALL PRINT_ERR(STMT_CH,'EXPECTING VARIABLE');
2058       IF TERMINATE_SCAN THEN RETURN;
2059
2060     /*  EXTRACT THE RECEIVING FIELD */
2061
2062       LEFT_SIDE='';
2063       NO_EQUAL=TRUE;
2064       DO I=STMT_CH TO STMT_RIGHT WHILE(NO_EQUAL);
2065         CH=SUBSTR(STMT,I,1);
2066         IF CH='=' THEN
2067         DO;
2068            NO_EQUAL=FALSE;
2069            RIGHT_SIDE=SUBSTR(STMT,I+1);
2070         END;
2071         ELSE
2072            IF CH=' ' THEN;
2073            ELSE
2074               LEFT_SIDE=LEFT_SIDE||CH;
2075       END;
2076
2077       CALL BALANCE_STMT(LEFT_SIDE);
2078       IF TERMINATE_SCAN THEN RETURN;
2079       CALL BALANCE_STMT(RIGHT_SIDE);
2080       IF TERMINATE_SCAN THEN RETURN;
2081
2082       IF SUBSTR(RIGHT_SIDE,1,1)='(' THEN;
2083       ELSE RIGHT_SIDE='('||RIGHT_SIDE||')';
```

MACRO SOURCE2 LISTING

```
2084
2085       CALL PARSE_EXP(RIGHT_SIDE,EXP_CALC);
2086
2087       IF PC_MAX > 0 THEN        /*  CHANGE A STA TMPXX TO STA RESULT */
2088       DO;
2089          IF PC_OPCODE(PC_MAX)=PC_OPCODE_STA THEN
2090              IF SUBSTR(SYMBOL(PC_OBJECT(PC_MAX)),1,3)='TMP' THEN
2091                 PC_OBJECT(PC_MAX)=1 ;
2092       END;
2093
2094       LEFT_SIDE='('||LEFT_SIDE||')';
2095       CALL PARSE_EXP(LEFT_SIDE,EXP_RCVR);
2096
2097    END PROCESS_LET;
2098
2099    PROCESS_DEF:PROC;
2100    /*********************************************************************
2101    *                                                                    *
2102    *   PROCESS DEF                                                      *
2103    *                                                                    *
2104    * NESTING:COMPILE                                                   *
2105    *********************************************************************/
2106       DECLARE I                   FIXED BINARY ALIGNED;
2107       DECLARE JMP_OFFSET          FIXED BINARY ALIGNED;
2108       DECLARE (OFFSET,OFFSET2)    FIXED BINARY ALIGNED;
2109       DECLARE CH                  CHAR(1);
2110       DECLARE CH2                 CHAR(2);
2111       DECLARE (LEFT_SIDE,RIGHT_SIDE,FUNC_TEMP)
2112                                   CHAR(80) VARYING;
2113       DECLARE TEMP_NAME           CHAR(10);
2114       DECLARE NO_EQUAL            BIT(1) ALIGNED;
2115
2116       TMP_CNT=50;
2117
2118       CH2=SUBSTR(STMT,STMT_CH,2);
2119       IF CH2='FN' THEN;
2120       ELSE
2121       DO;
2122          CALL PRINT_ERR(STMT_CH,'DEF MUST START WITH FN');
2123          RETURN;
2124       END;
2125
2126     /*  EXTRACT THE FUNCTION NAME   */
2127
2128       LEFT_SIDE='';
```

MACRO SOURCE2 LISTING

```
2129       NO_EQUAL=TRUE;
2130       DO I=STMT_CH TO STMT_RIGHT WHILE(NO_EQUAL);
2131         CH=SUBSTR(STMT,I,1);
2132         IF CH='=' THEN
2133         DO;
2134            NO_EQUAL=FALSE;
2135            RIGHT_SIDE=SUBSTR(STMT,I+1);
2136         END;
2137         ELSE
2138            IF CH=' ' THEN;
2139            ELSE
2140               LEFT_SIDE=LEFT_SIDE||CH;
2141       END;
2142
2143       CALL BALANCE_STMT(LEFT_SIDE);
2144       IF TERMINATE_SCAN THEN RETURN;
2145
2146       I=INDEX(LEFT_SIDE,'(');
2147       IF I>0 THEN
2148       DO;
2149          FUNC_TEMP=SUBSTR(LEFT_SIDE,1,I-1);
2150          FUNC_NAME=FUNC_TEMP;
2151          LEFT_SIDE=SUBSTR(LEFT_SIDE,I+1);
2152          IF VERIFY(FUNC_NAME,VALID_VAR_CHARS) > 0 THEN
2153          DO;
2154             CALL PRINT_ERR(I,'INVALID FUNCTION NAME');
2155             RETURN;
2156          END;
2157       END;
2158       ELSE
2159       DO;
2160          CALL PRINT_ERR(STMT_CH,'DEF SYNTAX ERROR');
2161          RETURN;
2162       END;
2163       IF SUBSTR(LEFT_SIDE,LENGTH(LEFT_SIDE),1)=')' THEN
2164       DO;
2165          FUNC_ARG=SUBSTR(LEFT_SIDE,1,LENGTH(LEFT_SIDE)-1);
2166          IF VERIFY(FUNC_ARG,VALID_VAR_CHARS) > 0 THEN
2167          DO;
2168             CALL PRINT_ERR(I,'INVALID FUNCTION ARGUMENT');
2169             RETURN;
2170          END;
2171          FUNC_TEMP=FUNC_TEMP||' '||FUNC_ARG;
2172       END;
2173       ELSE
```

MACRO SOURCE2 LISTING

```
2174       DO;
2175          CALL PRINT_ERR(STMT_CH,'DEF ARGUMENT ERROR');
2176          RETURN;
2177       END;
2178
2179       OFFSET=LOOKUP_SYMBOL_TABLE(FUNC_NAME);
2180       IF OFFSET=SS_MAX THEN
2181          IF SYM_TYPE(OFFSET)=SS_VAR THEN
2182          DO;
2183              SYM_TYPE(OFFSET)=SS_DEF_VAR;
2184              CALL ADD_PCODE(PC_OPCODE_JMP,ZERO);
2185              JMP_OFFSET=PC_MAX;
2186              TEMP_NAME=FUNC_TEMP;
2187              OFFSET2=LOOKUP_SYMBOL_TABLE(FUNC_TEMP);
2188              CALL ADD_PCODE(PC_OPCODE_STA,OFFSET2);
2189              IF OFFSET+1=OFFSET2 THEN;
2190              ELSE
2191              DO;
2192                 CALL PRINT_ERR(ERR_PTR,'FUNC/ARG NOT CONTIG');
2193                 RETURN;
2194              END;
2195              IF DF_MAX>=HBOUND(DF_NAME,1) THEN
2196              DO;
2197                 CALL PRINT_ERR(ERR_PTR,'TOO MANY DEF');
2198                 RETURN;
2199              END;
2200              DF_MAX=DF_MAX+1;
2201              DF_NAME(DF_MAX)=FUNC_NAME;
2202              DF_OFFSET(DF_MAX)=PC_MAX;
2203              DF_RETURN(DF_MAX)=0;
2204          END;
2205          ELSE
2206              CALL PRINT_ERR(ERR_PTR,'DEF SYMBOL NOT FOUND');
2207       ELSE
2208           CALL PRINT_ERR(ERR_PTR,'DEF SYMBOL REDEFINED');
2209
2210       CALL BALANCE_STMT(RIGHT_SIDE);
2211       IF TERMINATE_SCAN THEN RETURN;
2212
2213       IF SUBSTR(RIGHT_SIDE,1,1)='(' THEN;
2214       ELSE RIGHT_SIDE='('||RIGHT_SIDE||')';
2215
2216       CALL PARSE_EXP(RIGHT_SIDE,EXP_FN_CALC);
2217
2218       CALL ADD_PCODE(PC_OPCODE_STA,OFFSET);
```

MACRO SOURCE2 LISTING

```
2219
2220       IF PC_MAX > 0 THEN        /*  CHANGE A STA TMPXX TO STA RESULT */
2221       DO;
2222          IF PC_OPCODE(PC_MAX)=PC_OPCODE_STA THEN
2223              IF SUBSTR(SYMBOL(PC_OBJECT(PC_MAX)),1,3)='TMP' THEN
2224                 PC_OBJECT(PC_MAX)=1 ;
2225       END;
2226       CALL ADD_PCODE(PC_OPCODE_RFN,OFFSET);
2227       PC_OBJECT(JMP_OFFSET)=PC_MAX+1;
2228
2229    END PROCESS_DEF;
2230
2231    BALANCE_STMT:PROC(EXP);
2232    /*****************************************************************
2233    *                                                                *
2234    *   CHECK FOR BALANCE PARENS AND QUOTES                          *
2235    *                                                                *
2236    * NESTING:COMPILE                                                *
2237    *****************************************************************/
2238       DECLARE EXP                CHAR(*) VARYING;
2239       DECLARE I                  FIXED BINARY ALIGNED;
2240       DECLARE (PARENS,QUOTES)    FIXED BINARY ALIGNED;
2241
2242       PARENS=0;
2243       QUOTES=0;
2244       DO I=1 TO LENGTH(EXP);
2245          IF SUBSTR(EXP,I,1)='(' THEN PARENS=PARENS+1;
2246          ELSE
2247             IF SUBSTR(EXP,I,1)=')' THEN
2248             DO;
2249                PARENS=PARENS-1;
2250                IF PARENS < 0 THEN
2251                   CALL PRINT_ERR(STMT_CH,'INVALID USE OF PARENS');
2252                   RETURN;
2253                END;
2254          IF SUBSTR(EXP,I,1)=QUOTE_1 THEN QUOTES=QUOTES+1;
2255       END;
2256
2257       IF PARENS=0 THEN;
2258       ELSE
2259          CALL PRINT_ERR(STMT_CH,'UNBALANCED PARENS');
2260       IF MOD(QUOTES,2)=1 THEN   /* IF QUOTES IS ODD, ERROR */
2261          CALL PRINT_ERR(STMT_CH,'UNBALANCED QUOTES');
2262
2263    END BALANCE_STMT;
```

MACRO SOURCE2 LISTING

```
2264
2265    PARSE_EXP:PROC(EXP,EXP_TYPE);
2266    /*********************************************************************
2267    *                                                                    *
2268    *                                                                    *
2269    *                                                                    *
2270    * NESTING:COMPILE                                                   *
2271    *********************************************************************/
2272       DECLARE EXP                 CHAR(*) VARYING;
2273       DECLARE (I,J,EXP_TYPE)      FIXED BINARY ALIGNED;
2274       DECLARE EXPR                BIT(1) ALIGNED;
2275       DECLARE CH                  CHAR(1);
2276       DECLARE V                   CHAR(10) VARYING;
2277       DECLARE FN_TMP              CHAR(10);
2278       DECLARE (LAST_LP,OFFSET,
2279                RP,BREAKER,
2280                MAX_BREAKER)       FIXED BINARY ALIGNED;
2281       DECLARE NO_PARENS           BIT(1) ALIGNED;
2282       DECLARE CONTINUE_SCAN       BIT(1) ALIGNED;
2283
2284       DECLARE 1 STACK,
2285                 2  STACK_MAX      FIXED BINARY ALIGNED,
2286                 2  STACK_CUR      FIXED BINARY ALIGNED,
2287                 2  ITEMS(50),
2288                    3  WORD        CHAR(10),
2289                    3  OP          CHAR(1);
2290
2291       STACK_MAX,STACK_CUR=0;
2292       CALL POPULATE_STACK;
2293       IF EXP_TYPE=EXP_FN_CALC THEN
2294       DO;
2295          I=INDEX(FUNC_NAME,' ');
2296          J=INDEX(FUNC_ARG,' ');
2297          FN_TMP=SUBSTR(FUNC_NAME,1,I)||SUBSTR(FUNC_ARG,1,J)||(6)' ';
2298          IF STACK_PRINT_DEBUG THEN
2299             PUT SKIP DATA(FN_TMP);
2300          DO I=1 TO STACK_MAX;
2301             IF WORD(I)=FUNC_ARG THEN
2302                WORD(I)=FN_TMP;
2303             IF STACK_PRINT_DEBUG THEN
2304                PUT SKIP DATA(ITEMS(I));
2305          END;
2306       END;
2307
2308       EXPR=(EXP_TYPE=EXP_CALC | EXP_TYPE=EXP_FN_CALC);
```

MACRO SOURCE2 LISTING

```
2309
2310       BREAKER=0;
2311       MAX_BREAKER=STACK_MAX;
2312       CONTINUE_SCAN=TRUE;
2313       DO WHILE(CONTINUE_SCAN & BREAKER <= MAX_BREAKER);
2314          I=0;
2315          NO_PARENS=TRUE;
2316          LAST_LP=0;
2317          DO WHILE(NO_PARENS & I<= STACK_MAX);
2318             I=I+1;
2319             IF OP(I)='(' THEN LAST_LP=I;
2320             ELSE
2321                IF OP(I)=')' THEN
2322                   NO_PARENS=FALSE;
2323          END; /* DO WHILE(NO_PARENS....  */
2324          IF NO_PARENS THEN
2325          DO;
2326             LAST_LP=1;
2327             RP=STACK_MAX;
2328          END;
2329          ELSE
2330             RP=I;
2331          CALL SIMPLIFY_SUB_STACK(LAST_LP,RP);
2332          IF STACK_MAX > 1 THEN
2333             CONTINUE_SCAN = TRUE;
2334          ELSE
2335             CONTINUE_SCAN = FALSE;
2336          BREAKER=BREAKER+1;
2337       END; /* DO WHILE(CONTINUE_SCAN.....  */
2338
2339       IF BREAKER>MAX_BREAKER THEN
2340       DO;
2341           PUT SKIP LIST('BREAKER>MAX');
2342       END;
2343
2344       IF STACK_MAX < 3 THEN  /*  SIMPLE VARIABLE? */
2345       DO;
2346         OFFSET=LOOKUP_SYMBOL_TABLE(WORD(1));
2347         DO I=1 TO SS_MAX;
2348           IF WORD(1)=SYMBOL(I) THEN
2349           DO;
2350             IF EXPR THEN
2351             DO;
2352         /***  IF SYM_TYPE(I)=SS_FUNC THEN
2353         *       RC=PR_FUNC;
```

MACRO SOURCE2 LISTING

```
2354         *     ELSE
2355         *       IF SYM_TYPE(I)=SS_VAR THEN
2356         *          RC=PR_VAR;
2357         *       ELSE
2358         *           IF SYM_TYPE(I)=SS_DIM_VAR THEN
2359         *              RC=PR_SUB_VAR;
2360         *           ELSE
2361         *               IF SYM_TYPE(I)=SS_CONST THEN
2362         *                  RC=PR_VAR;
2363         *               ELSE
2364         *                   CALL PRINT_ERR(STMT_CH,'VARIABLE EXPECTED');
2365         */  END;
2366              ELSE
2367              DO;
2368                IF SYM_TYPE(I)=SS_VAR THEN
2369                DO;
2370                  IF PC_OPCODE(PC_MAX)=PC_OPCODE_STA THEN
2371                    IF SUBSTR(SYMBOL(PC_OBJECT(PC_MAX)),1,3)='TMP' THEN
2372                      PC_OBJECT(PC_MAX)=I;
2373                    ELSE
2374                      /* CALL PRINT_ERR(STMT_CH,
2375                                   'SYNTAX ERROR RESULT EXPECTED') */;
2376                  ELSE
2377                     CALL ADD_PCODE(PC_OPCODE_STA,OFFSET);
2378                END;
2379                ELSE
2380                  IF SYM_TYPE(I)=SS_STRCON  |
2381                     SYM_TYPE(I)=SS_STRVAR  |
2382                     SYM_TYPE(I)=SS_STRDIM  |               /*PAT 02*/
2383                     SYM_TYPE(I)=SS_DIM_VAR THEN;
2384                  ELSE
2385                    CALL PRINT_ERR(STMT_CH,'A VARIABLE IS EXPECTED HERE');
2386              END;
2387              RETURN;
2388            END;
2389           END; /* DO LOOP */
2390           CALL PRINT_ERR(STMT_CH,'VARIABLE ' || WORD(1) || ' UNDEFINED?');
2391       END;
2392       ELSE
2393          CALL PRINT_ERR(STMT_CH,'BIG TIME SYNTAX ERROR IN EXPRESSION');
2394
2395     POPULATE_STACK:PROC;
2396     /*****************************************************************
2397     *                                                               *
2398     *  BREAK EXP INTO WORDS AT THE SPECIAL CHARACTERS               *
```

MACRO SOURCE2 LISTING

```
2399    *  TO SUPPORT STRINGS, ANY STRING CONSTANTS WILL BE EXTRACTED AND   *
2400    *  ADDED TO SYMBOL TABLE USING A GENERATED NAME.  THIS NAME WILL    *
2401    *  BE SUBSTITUTED IN THE STACK FOR THE STRING.                      *
2402    *                                                                   *
2403    *********************************************************************
2404    * NESTING:COMPILE - PARSE_EXP                                       *
2405    *********************************************************************/
2406       DECLARE IN_STR         BIT(1) ALIGNED;
2407       DECLARE STR_WORK       CHAR(80) VARYING;
2408       DECLARE TMP_VAR        CHAR(10);
2409       DECLARE OFFSET         FIXED BINARY ALIGNED;
2410       IN_STR = FALSE;
2411
2412       STR_WORK='';
2413       V='';
2414       DO I=1 TO LENGTH(EXP);
2415          CH=SUBSTR(EXP,I,1);
2416          IF IN_STR THEN
2417          DO;
2418             IF CH=QUOTE_1 THEN
2419             DO;
2420                IN_STR=FALSE;
2421                TMP_VAR='STR$'||STR_CNT;
2422                STR_CNT=STR_CNT+1;
2423                V=TMP_VAR;
2424                OFFSET=LOOKUP_SYMBOL_TABLE(TMP_VAR);
2425                IF STACK_PRINT_DEBUG THEN
2426                   PUT DATA(STR_WORK,V,TMP_VAR,OFFSET);
2427                STRING_VAL(OFFSET)=STR_WORK;
2428             END;
2429             ELSE
2430                STR_WORK=STR_WORK||CH;
2431          END;
2432          ELSE
2433          IF CH=QUOTE_1 THEN
2434          DO;
2435              IN_STR=TRUE;
2436          END;
2437          ELSE
2438          IF CH='(' | CH=')' | CH='+' | CH='-' | CH='*' | CH='/' |
2439             CH='^' THEN
2440          DO;
2441              STACK_MAX=STACK_MAX+1;
2442              WORD(STACK_MAX)=V;
2443              OP(STACK_MAX)=CH;
```

MACRO SOURCE2 LISTING

```
2444              V='';
2445           END;
2446           ELSE
2447               V=V||CH;
2448        END;
2449
2450        IF STACK_MAX=0 THEN   /*  EXP IS A SIMPLE VAR OR CONSTANT */
2451        DO;
2452           STACK_MAX=1;
2453           WORD(1)=V;
2454           OP(1)=' ';
2455        END;
2456        ELSE
2457        DO I=1 TO STACK_MAX;
2458           IF WORD(I)=(10)' ' THEN
2459              IF OP(I)= '(' THEN;
2460              ELSE
2461                 IF I>1 THEN
2462                    IF OP(I-1)=')' | OP(I)='-' THEN;
2463                    ELSE
2464                       CALL PRINT_ERR(STMT_CH,'SYNTAX ERROR');
2465        END;
2466
2467        IF STACK_PRINT_DEBUG THEN
2468        DO;
2469           PUT SKIP LIST('POPULTAT STACK EXIT');
2470           DO I=1 TO STACK_MAX;
2471              PUT SKIP LIST(I,WORD(I),OP(I));
2472           END;
2473        END;
2474
2475     END POPULATE_STACK;
2476
2477     SIMPLIFY_SUB_STACK:PROC(LP,RP);
2478     /*********************************************************************
2479     *                                                                   *
2480     *                                                                   *
2481     *   PROCESS OPERATORE LOCATED BETWEEN THE LEFT PAREN (LP) AND       *
2482     *   RIGHT PAREN (RP).  FIRST CHECK STACK FOR A SIMPLE QUANTITY      *
2483     *   (I.E. (X) ) OR VAIABLE IN THE STACK.  NO NEED TO SIMPLFY THESE  *
2484     *                                                                   *
2485     *                                                                   *
2486     *********************************************************************
2487     * NESTING:COMPILE - PARSE_EXP                                       *
2488     *********************************************************************/
```

MACRO SOURCE2 LISTING

```
2489       DECLARE (LP,RP)                 FIXED BINARY ALIGNED;
2490       DECLARE (I,J,K,HJ,HK,OFFSET)    FIXED BINARY ALIGNED;
2491       DECLARE  TMP_VAR                CHAR(5);
2492
2493       IF STACK_PRINT_DEBUG THEN
2494       DO;
2495          PUT SKIP LIST('SIMPLIFY_SUB_STACK ENTRY FROM',LP,'TO ',RP);
2496          DO I=LP TO RP;
2497             PUT SKIP LIST(I,WORD(I),OP(I));
2498          END;
2499       END;
2500
2501       IF STACK_MAX = 1 THEN
2502           GO TO SIMPLIFY_SUB_STACK_EXIT;
2503
2504            /*  CHECK FOR SIMPLE VARIABLE QUANTITY, DIM VAR OR FUNC */
2505
2506       IF STACK_MAX = 2 THEN
2507       DO;
2508          IF OP(LP)='(' & OP(LP+1)=')' THEN
2509          DO;
2510             IF WORD(LP)=(10)' ' THEN
2511                IF WORD(LP+1)=(10)' ' THEN   /* EMPTY PARENS ERROR */
2512                DO;
2513                    CALL PRINT_ERR(STMT_CH,'EMPTY PARENS ERROR');
2514                    RETURN;
2515                END;
2516                ELSE                    /*  SIMPLE QUANTITY  */
2517           /*** DO;
2518           **      WORD(LP)=WORD(LP+1);
2519           **      OP(LP)=' ';
2520           **      STACK_MAX=STACK_MAX-1;
2521           **      GO TO SIMPLIFY_SUB_STACK_EXIT;
2522           **** END;  *****/;
2523             ELSE
2524             DO;                         /* COULD BE X(Y) OR FNC(Y)  */
2525                IF STACK_PRINT_DEBUG THEN
2526                DO;
2527                   PUT SKIP LIST('FOUND DIM VAR OR FUNC');
2528                END;
2529                OFFSET=LOOKUP_SYMBOL_TABLE(WORD(LP));
2530                IF SYM_TYPE(OFFSET)=SS_DIM_VAR |
2531                   SYM_TYPE(OFFSET)=SS_FUNC       THEN;
2532                ELSE
2533                DO;
```

MACRO SOURCE2 LISTING

```
2534                    CALL PRINT_ERR(STMT_CH,
2535                       'EXPECTING DIM VARIABLE OR FUNCTION CALL');
2536                    RETURN;
2537                 END;
2538              END;
2539           END;
2540        END;
2541
2542        IF OP(LP)= '(' THEN
2543        DO;
2544           HJ=LP+1;
2545           HK=RP-1;
2546        END;
2547        ELSE
2548        DO;
2549           HJ=LP;
2550           HK=RP;
2551        END;
2552
2553        J=HJ;
2554        K=HK;
2555        IF K<J THEN
2556        DO;
2557           IF STACK_PRINT_DEBUG THEN
2558              PUT SKIP(2) LIST('PROCESS_OPERATORS BYPASSED',J,K);
2559        END;
2560        ELSE
2561        DO;
2562           CALL PROCESS_OPERATORS('^','^',PC_OPCODE_EXP,PC_OPCODE_EXP);
2563           J=HJ;
2564           CALL PROCESS_OPERATORS('*','/',PC_OPCODE_MUL,PC_OPCODE_DIV);
2565           J=HJ;
2566           CALL PROCESS_OPERATORS('+','-',PC_OPCODE_ADD,PC_OPCODE_SUB);
2567        END;
2568     /*   IF OP(LP)='(' & OP(LP+1)=')' THEN
2569        DO; */
2570           IF WORD(LP)=(10)' ' THEN          /* QUANTITY - GET RID OF () */
2571           DO;
2572              IF STACK_MAX=1 THEN;
2573              ELSE
2574              IF STACK_MAX=2 THEN
2575              DO;
2576                 WORD(1)=WORD(2);
2577                 OP(1)=' ';
2578                 STACK_MAX=1;
```

MACRO SOURCE2 LISTING

```
2579               OFFSET=LOOKUP_SYMBOL_TABLE(WORD(1));
2580               IF EXPR THEN
2581                   CALL ADD_PCODE(PC_OPCODE_LDA,OFFSET);
2582               ELSE
2583                   CALL ADD_PCODE(PC_OPCODE_STA,OFFSET);
2584            END;
2585            ELSE
2586            DO;
2587               IF STACK_PRINT_DEBUG THEN
2588               DO;
2589                  PUT SKIP LIST('SIMPLIFY_SUB_STACK BEFORE UPDATE',
2590                                LP,RP);
2591                  DO I=1 TO STACK_MAX;
2592                     PUT SKIP LIST(I,WORD(I),OP(I));
2593                  END;
2594               END;
2595               WORD(LP)=WORD(LP+1);
2596               IF LP+2 < STACK_MAX THEN
2597                  OP(LP)=OP(LP+2);
2598               ELSE
2599                  OP(LP)=' ';
2600            /*   WORD(LP+1)=WORD(LP+3); */
2601               DO I=LP+1 TO STACK_MAX;
2602                  ITEMS(I)=ITEMS(I+2);
2603               END;
2604               IF LP+1=STACK_MAX THEN
2605                  STACK_MAX=STACK_MAX-1;
2606               ELSE
2607                  STACK_MAX=STACK_MAX-2;
2608               K=K-1;
2609               J=LP;
2610               IF STACK_PRINT_DEBUG THEN
2611               DO;
2612                  PUT SKIP LIST('SIMPLIFY_SUB_STACK AFTER UPDATE',
2613                                LP,RP);
2614                  DO I=1 TO STACK_MAX;
2615                     PUT SKIP LIST(I,WORD(I),OP(I));
2616                  END;
2617               END;
2618            END;
2619         END;
2620         ELSE    /*  COULD BE A FUNCTION OR DIM VARIABLE */
2621         DO;
2622           OFFSET=LOOKUP_SYMBOL_TABLE(WORD(LP));
2623         /*  PUT SKIP DATA(OFFSET,SYM_TYPE(OFFSET)); */
```

MACRO SOURCE2 LISTING

```
2624            IF SYM_TYPE(OFFSET)=SS_FUNC |
2625               SYM_TYPE(OFFSET)=SS_DEF_VAR THEN
2626               CALL PROCESS_FUNCTION(OFFSET);
2627            ELSE
2628               IF SYM_TYPE(OFFSET)=SS_DIM_VAR |
2629                  SYM_TYPE(OFFSET)=SS_STRDIM  THEN
2630                  CALL PROCESS_SUBSCRIPT(OFFSET);
2631               ELSE
2632                  IF SYM_TYPE(OFFSET)=SS_VAR THEN;
2633                  ELSE
2634                     CALL PRINT_ERR(STMT_CH,'EXPECTING DIM');
2635         END;
2636   /*   END;
2637      ELSE
2638         PUT SKIP LIST('*** NO PARENS ***');*/
2639
2640   SIMPLIFY_SUB_STACK_EXIT:
2641      IF STACK_PRINT_DEBUG THEN
2642      DO;
2643         PUT SKIP LIST('SIMPLIFY_SUB_STACK EXIT');
2644         DO I=1 TO STACK_MAX;
2645            PUT SKIP LIST(I,WORD(I),OP(I));
2646         END;
2647      END;
2648
2649
2650   PROCESS_OPERATORS:PROC(OP1,OP2,PC1,PC2);
2651   /*****************************************************************
2652   *                                                               *
2653   * THIS PROC SCANS FOR OP1 AND OP2 WITHIN ROWS J AND K OF THE STACK  *
2654   * PC1 AND PC2 ARE THE OPCODES FOR OP1 AND OP2 RESPECTIVELY          *
2655   * THE VARIABLES J AND K ARE GLOBAL TO SIMPLYFY_SUB_STACK            *
2656   * THEY CONTAIN THE FIRST AND LAST ROWS FOR THIS PROC TO PROCESS     *
2657   *                                                               *
2658   *****************************************************************
2659   * NESTING:COMPILE - PARSE_EXP - SIMPLYFY_SUB_STACK                  *
2660   *****************************************************************/
2661      DECLARE (OP1,OP2)              CHAR(1),
2662              (PC1,PC2)              FIXED BINARY ALIGNED;
2663      DECLARE OFFSET                 FIXED BINARY ALIGNED;
2664
2665      IF STACK_PRINT_DEBUG THEN
2666      DO;
2667         PUT SKIP(2) LIST('PROCESS_OPERATORS ENTRY',J,K,OP1,OP2);
2668         DO I=1 TO STACK_MAX;
```

MACRO SOURCE2 LISTING

```
2669             PUT SKIP LIST(I,WORD(I),OP(I));
2670          END;
2671       END;
2672
2673       DO WHILE(J<=K);
2674          IF OP(J)=OP1 | OP(J)=OP2 THEN
2675          DO;
2676             TMP_VAR='TMP'||TMP_CNT;
2677             TMP_CNT=TMP_CNT+1;
2678
2679             OFFSET=LOOKUP_SYMBOL_TABLE(WORD(J));
2680             CALL ADD_PCODE(PC_OPCODE_LDA,OFFSET);
2681
2682             OFFSET=LOOKUP_SYMBOL_TABLE(WORD(J+1));
2683             IF OP(J)=OP1 THEN
2684                CALL ADD_PCODE(PC1,OFFSET);
2685             ELSE
2686                CALL ADD_PCODE(PC2,OFFSET);
2687
2688             WORD(J)=TMP_VAR;
2689             OFFSET=LOOKUP_SYMBOL_TABLE(WORD(J));
2690             CALL ADD_PCODE(PC_OPCODE_STA,OFFSET);
2691
2692             OP(J)=OP(J+1);
2693             DO I=J+1 TO STACK_MAX;
2694                ITEMS(I)=ITEMS(I+1);
2695             END;
2696             STACK_MAX=STACK_MAX-1;
2697             K=K-1;
2698             J=LP;
2699             IF STACK_PRINT_DEBUG THEN
2700             DO;
2701                PUT SKIP LIST('PROCESS_OPERATORE UPDATE',J,K);
2702                DO I=1 TO STACK_MAX;
2703                  PUT SKIP LIST(I,WORD(I),OP(I));
2704                END;
2705             END;
2706          END;
2707          J=J+1;
2708       END; /* DO WHILE */
2709
2710       IF STACK_PRINT_DEBUG THEN
2711       DO;
2712          PUT SKIP LIST('PROCESS_OPERATORS EXIT',J,K);
2713          DO I=1 TO STACK_MAX;
```

MACRO SOURCE2 LISTING

```
2714            PUT SKIP LIST(I,WORD(I),OP(I));
2715         END;
2716      END;
2717
2718   END PROCESS_OPERATORS;
2719
2720   PROCESS_FUNCTION:PROC(OFFSET);
2721   /*********************************************************************
2722   *                                                                   *
2723   *                                                                   *
2724   *                                                                   *
2725   * NESTING:COMPILE - PARSE_EXP - SIMPLYFY_SUB_STACK                  *
2726   *********************************************************************/
2727      DECLARE (OFFSET,OFFSET2,OFFSET3,I)       FIXED BINARY ALIGNED;
2728      DECLARE TEMP_SYM                         CHAR(10) INITIAL((10)' ');
2729      OFFSET2=LOOKUP_SYMBOL_TABLE(WORD(LP+1));
2730
2731      IF STACK_PRINT_DEBUG THEN
2732      DO;
2733         PUT SKIP LIST('PROCESS_FUNCTION ENTRY');
2734         DO I=1 TO STACK_MAX;
2735               PUT SKIP LIST(I,WORD(I),OP(I));
2736         END;
2737      END;
2738
2739      IF SYM_TYPE(OFFSET2)=SS_VAR |
2740         SYM_TYPE(OFFSET2)=SS_CONST THEN
2741      DO;
2742         TMP_VAR='TMP'||TMP_CNT;
2743         TMP_CNT=TMP_CNT+1;
2744         TEMP_SYM=TMP_VAR;
2745         OFFSET3=LOOKUP_SYMBOL_TABLE(TEMP_SYM);
2746
2747         IF SYM_TYPE(OFFSET)=SS_FUNC THEN
2748         DO;
2749            CALL ADD_PCODE(PC_OPCODE_LDA,OFFSET2);
2750            CALL ADD_PCODE(PC_OPCODE_FNC,OFFSET);
2751            CALL ADD_PCODE(PC_OPCODE_STA,OFFSET3);
2752         END;
2753         ELSE
2754         IF SYM_TYPE(OFFSET)=SS_DEF_VAR THEN
2755         DO;
2756            CALL ADD_PCODE(PC_OPCODE_LDA,OFFSET2);
2757            CALL ADD_PCODE(PC_OPCODE_CFN,OFFSET);
2758            CALL ADD_PCODE(PC_OPCODE_STA,OFFSET3);
```

MACRO SOURCE2 LISTING

```
2759          END;
2760          ELSE
2761             CALL PRINT_ERR(STMT_CH,'UNKNOWN FUNCTION DETECTED');
2762       /* CALL ADD_PCODE(PC_OPCODE_STA,OFFSET3); */
2763          /*************************************************************
2764          *                                                           *
2765          *  ADJUST THE STACK TO REPLACE THE FUNC REF WITH THE TEMP VAR *
2766          *  IF NO OTHER OPERATORS FOLLOW, PUSH UP 2 ITEMS,  IF OTHER   *
2767          *  OPERATORS FOLLOW, PUSH UP 1 ITEM ONLY.                   *
2768          *                                                           *
2769          *************************************************************/
2770          IF STACK_MAX < 5 THEN
2771          DO;
2772             WORD(LP)=TMP_VAR;
2773             OP(LP)=')';
2774             DO I=LP+2 TO STACK_MAX;
2775               ITEMS(I-1)=ITEMS(I);
2776             END;
2777             STACK_MAX=STACK_MAX-2;
2778          END;
2779          ELSE
2780          DO;
2781             WORD(LP)=TMP_VAR;
2782             OP(LP)=OP(LP+2);
2783             DO I=LP+3 TO STACK_MAX;
2784               ITEMS(I-2)=ITEMS(I);
2785             END;
2786             STACK_MAX=STACK_MAX-2;
2787          END;
2788       END;
2789       ELSE
2790          CALL PRINT_ERR(STMT_CH,'INVALID FUNCTION ARGUMENT');
2791
2792       IF STACK_PRINT_DEBUG THEN
2793       DO;
2794          PUT SKIP LIST('PROCESS_FUNCTION EXIT');
2795          DO I=1 TO STACK_MAX;
2796               PUT SKIP LIST(I,WORD(I),OP(I));
2797          END;
2798       END;
2799
2800    END PROCESS_FUNCTION;
2801
2802    PROCESS_SUBSCRIPT:PROC(OFFSET);
2803    /*****************************************************************
```

MACRO SOURCE2 LISTING

```
2804    *                                                                *
2805    *                                                                *
2806    *                                                                *
2807    * NESTING:COMPILE - PARSE_EXP - SIMPLYFY_SUB_STACK               *
2808    ******************************************************************/
2809       DECLARE (OFFSET,OFFSET2,OFFSET3,I)       FIXED BINARY ALIGNED;
2810       DECLARE TEMP_SYM                         CHAR(10) INITIAL((10)' ');
2811       DECLARE TMP_VAR                          CHAR(10) INITIAL((10)' ');
2812       OFFSET2=LOOKUP_SYMBOL_TABLE(WORD(LP+1));
2813
2814       IF STACK_PRINT_DEBUG THEN
2815       DO;
2816          PUT SKIP LIST('PROCESS_SUBSCRIPT ENTRY');
2817          DO I=1 TO STACK_MAX;
2818                PUT SKIP LIST(I,WORD(I),OP(I));
2819          END;
2820       END;
2821
2822       IF SYM_TYPE(OFFSET2)=SS_VAR |
2823          SYM_TYPE(OFFSET2)=SS_CONST THEN
2824       DO;
2825          IF SYM_TYPE(OFFSET)=SS_DIM_VAR |
2826             SYM_TYPE(OFFSET)=SS_STRDIM  THEN
2827          DO;
2828             IF SYM_TYPE(OFFSET)=SS_DIM_VAR THEN
2829                TMP_VAR='TMP'||TMP_CNT;
2830             ELSE
2831                TMP_VAR='TMP'||TMP_CNT||'$';
2832             TMP_CNT=TMP_CNT+1;
2833             TEMP_SYM=TMP_VAR;
2834             OFFSET3=LOOKUP_SYMBOL_TABLE(TEMP_SYM);
2835
2836             IF EXPR THEN
2837             DO;
2838                CALL ADD_PCODE(PC_OPCODE_LDR,OFFSET2);
2839                CALL ADD_PCODE(PC_OPCODE_DSL,ZERO);
2840                CALL ADD_PCODE(PC_OPCODE_LDA,OFFSET);
2841                CALL ADD_PCODE(PC_OPCODE_STA,OFFSET3);
2842             END;
2843             ELSE
2844             DO;
2845                CALL ADD_PCODE(PC_OPCODE_LDR,OFFSET2);
2846                CALL ADD_PCODE(PC_OPCODE_DSL,ZERO);
2847                CALL ADD_PCODE(PC_OPCODE_STA,OFFSET);
2848                TMP_VAR=SYMBOL(OFFSET);
```

MACRO SOURCE2 LISTING

```
2849            END;
2850          /*************************************************************
2851          *                                                            *
2852          *  ADJUST THE STACK TO REPLACE THE SUB  REF WITH THE TEMP VAR *
2853          *  IF NO OTHER OPERATORS FOLLOW, PUSH UP 2 ITEMS,  IF OTHER   *
2854          *  OPERATORS FOLLOW, PUSH UP 1 ITEM ONLY.                     *
2855          *                                                            *
2856          *************************************************************/
2857             IF STACK_MAX < 5 THEN
2858             DO;
2859                WORD(LP)=TMP_VAR;
2860                OP(LP)=')';
2861                DO I=LP+2 TO STACK_MAX;
2862                  ITEMS(I-1)=ITEMS(I);
2863                END;
2864                STACK_MAX=STACK_MAX-2;
2865             END;
2866             ELSE
2867             DO;
2868                WORD(LP)=TMP_VAR;
2869                OP(LP)=OP(LP+2);
2870                DO I=LP+3 TO STACK_MAX;
2871                  ITEMS(I-2)=ITEMS(I);
2872                END;
2873                STACK_MAX=STACK_MAX-2;
2874             END;
2875          END;
2876          ELSE
2877             CALL PRINT_ERR(STMT_CH,'UNKNOWN SUBSCRIPT DETECTED');
2878       END;
2879       ELSE
2880          CALL PRINT_ERR(STMT_CH,'INVALID SUBSCRIPT');
2881
2882       IF STACK_PRINT_DEBUG THEN
2883       DO;
2884          PUT SKIP LIST('PROCESS_SUBSCRIPT EXIT');
2885          IF STACK_MAX = 0 THEN
2886             PUT SKIP DATA(STACK_MAX);
2887          ELSE
2888          DO I=1 TO STACK_MAX;
2889                PUT SKIP LIST(I,WORD(I),OP(I));
2890          END;
2891       END;
2892
2893    END PROCESS_SUBSCRIPT;
```

MACRO SOURCE2 LISTING

```
2894
2895    END SIMPLIFY_SUB_STACK;
2896
2897    END PARSE_EXP;
2898
2899    PROCESS_DIM:PROC;
2900    /*********************************************************************
2901    *                                                                    *
2902    * PROCESS_DIM                                                        *
2903    *                                                                    *
2904    * NESTING:COMPILE                                                    *
2905    *********************************************************************/
2906       DECLARE (I,NUM_OCCURS)      FIXED BINARY ALIGNED;
2907       DECLARE ERR_PTR             FIXED BINARY ALIGNED;
2908       DECLARE CH                  CHAR(1);
2909       DECLARE (LEFT_SIDE,START_VAL,TO_VAL,STEP_VAL)
2910                                   CHAR(80) VARYING;
2911       DECLARE NO_OPER             BIT(1) ALIGNED;
2912       DECLARE OFFSET              FIXED BINARY ALIGNED;
2913       DECLARE DIM_VAR             CHAR(10);
2914
2915     /*  EXTRACT THE DIM VARIABLE */
2916
2917    NEXT_DIM:
2918       LEFT_SIDE='';
2919       NO_OPER=TRUE;
2920
2921       DO I=STMT_CH TO STMT_RIGHT WHILE(NO_OPER);
2922         CH=SUBSTR(STMT,I,1);
2923         IF CH='(' | CH=' ' THEN
2924            NO_OPER=FALSE;
2925         ELSE
2926            LEFT_SIDE=LEFT_SIDE||CH;
2927       END;
2928
2929       DIM_VAR=LEFT_SIDE;
2930       OFFSET=LOOKUP_SYMBOL_TABLE(DIM_VAR);
2931
2932       IF SYM_TYPE(OFFSET)=SS_DIM_VAR |
2933          SYM_TYPE(OFFSET)=SS_STRDIM  THEN;
2934       ELSE
2935       DO;
2936          CALL PRINT_ERR(STMT_CH,'VARIABLE NOT ALLOWED');
2937          RETURN;
2938       END;
```

MACRO SOURCE2 LISTING

```
2939
2940      /*  EXTRACT THE OCCURANCES - NUMBERS ONLY  */
2941
2942       STMT_CH=I;
2943       CALL SKIP_BLANKS;
2944
2945       LEFT_SIDE='';
2946       NO_OPER=TRUE;
2947
2948       DO I=STMT_CH TO STMT_RIGHT WHILE(NO_OPER);
2949         CH=SUBSTR(STMT,I,1);
2950         IF CH=')'  THEN
2951            NO_OPER=FALSE;
2952         ELSE
2953            IF CH=' ' THEN ;
2954            ELSE
2955               IF CH>='0' & CH<='9' THEN
2956                  LEFT_SIDE=LEFT_SIDE||CH;
2957               ELSE
2958               DO;
2959                  CALL PRINT_ERR(STMT_CH,'EXPECTING NUMBER');
2960                  RETURN;
2961               END;
2962
2963       END;
2964
2965       IF NO_OPER THEN         /* ENSURE THERE IS A ) */
2966       DO;
2967           CALL PRINT_ERR(STMT_CH,'UNEXPECTED END OF STATEMENT');
2968           RETURN;
2969      END;
2970      STMT_CH=I;
2971
2972      NUM_OCCURS=LEFT_SIDE;
2973      IF OFFSET=SS_MAX &
2974         (SYM_TYPE(OFFSET)=SS_DIM_VAR |
2975          SYM_TYPE(OFFSET)=SS_STRDIM) THEN
2976      DO;
2977          IF SS_MAX+NUM_OCCURS > HBOUND(SYMBOL,1) THEN
2978          DO;
2979             CALL PRINT_ERR(STMT_CH,'DIM OCCURS TOO LARGE');
2980             RETURN;
2981          END;
2982          SYM_DIM_MAX(SS_MAX)=NUM_OCCURS;
2983          DO I=1 TO NUM_OCCURS;
```

MACRO SOURCE2 LISTING

```
2984             SS_MAX=SS_MAX+1;
2985             SYMBOL(SS_MAX)=SUBSTR(DIM_VAR,1,9)||'+';
2986             SYM_TYPE(SS_MAX)=SYM_TYPE(OFFSET);
2987             SYM_VALUE(SS_MAX)=0.0;
2988             STRING_VAL(SS_MAX)='*';
2989          END;
2990       END;
2991       ELSE
2992       DO;
2993           CALL PRINT_ERR(STMT_CH,'DUPLICATE DIM VARIABLE');
2994           RETURN;
2995       END;
2996
2997       CALL SKIP_BLANKS;
2998       IF STMT_CH>STMT_RIGHT THEN;
2999       ELSE
3000       DO;
3001          CH=SUBSTR(STMT,STMT_CH,1);
3002          IF CH=',' THEN
3003          DO;
3004             STMT_CH=STMT_CH+1;
3005             GO TO NEXT_DIM;
3006          END;
3007          ELSE
3008          DO;
3009             CALL PRINT_ERR(STMT_CH,'COMMA EXPECTED');
3010             RETURN;
3011          END;
3012
3013    END PROCESS_DIM;
3014
3015    END COMPILE;
```

MACRO SOURCE2 LISTING

```
3016    /*********************************************************************/
3017    /*********************************************************************/
3018    /*********************************************************************/
3019    /*********************************************************************/
3020    /*********************************************************************/
3021
3022    EXECUTE:PROC;
3023
3024    /*********************************************************************
3025    *                                                                    *
3026    *   THIS PROC DRIVES THE EXECUTION OF THE PSEUDO MACHINE CODE.       *
3027    *   ERROR TRAPPING FOR THE BASIC PROGRAM AS WELL AS LIMITATIONS      *
3028    *   ON EXECUTION TO PREVENT RUN AWAY PROGRAMS.                       *
3029    *                                                                    *
3030    *   ACCUM      IS THE PSEUDO COMPUTER ACCUMULATOR                    *
3031    *   CUR_LN     IS CURRENT LINE NUMBER OF THE BASIC PGM EXECUTING     *
3032    *   P_CTR      IS THE COUNTER OF PCODES EXECUTED                     *
3033    *   P_CTR_MAX  IS THE MAXIMUM VALUE P_CTR CAN HAVE.  ONCE THIS       *
3034    *              VALUE IS REACHED, THE BASIC PRORAM IS ABORTED.        *
3035    *   P_PTR      IS THE CURRENT PC_OPCODE TO BE/CURRENTLY EXECUTING    *
3036    *   P_PTR_SUB  IS THE CURRENT PC_OPCODE MODIFIER FOR TYPE CHECKS     *
3037    *                                                                    *
3038    *   GOSUB_STACK IS USED TO IMPLEMENT THE GOSUB AND RETURN STMTS      *
3039    *   FOR_STACK   IS USED TO IMPLEMENT FOR NEXT LOOPS                  *
3040    *                                                                    *
3041    *   THERE IS A SPECIAL REGISTER CALLED DSL_REG (DIM SUBSCRIPT        *
3042    *   LOCATOR) THAT IS USED TO IMPLEMENT SUBSCRIPTS.  THE DSL_REG IS   *
3043    *   SET BY THE DSL PSUEDO INSTRUCTION USING THE CONTENTS OF ACCUM.   *
3044    *   PRIOR TO THE EXECUTION OF THE NEXT PSEDUO INSTRUCTION AFTER THE  *
3045    *   DSL IS EXECUTED, THE VALUE OF THE DSL_REG WILL BE ADDED TO THE   *
3046    *   OFFSET (PC_OFFSET) TO EFFECTIVLY IMPLEMENT THE SUBSCRIPT. THE    *
3047    *   DSL_REG IS THEN SET TO ZERO. THE CONTENST OF THE PC_OFFSET       *
3048    *   FOR THE DSL IS NOT USED NOW.                                     *
3049    *                                                                    *
3050    **********************************************************************
3051    * NESTING:EXECUTE                                                   *
3052    *********************************************************************/
3053
3054        PUT SKIP;
3055        DECLARE PC_INST(0:GENPC_CTR)      LABEL;
3056        DECLARE LIB_FNC(2:GENSYM_CTR)     LABEL;
3057        DECLARE (P_PTR,P_PTR_SUB)         FIXED BINARY ALIGNED;
3058        DECLARE (P_CTR,P_CTR_MAX)         FIXED BINARY ALIGNED;
3059        DECLARE (DSL_REG,OFFSET_VAL)      FIXED BINARY ALIGNED;
3060        DECLARE CUR_LN                    FIXED BINARY ALIGNED;
```

MACRO SOURCE2 LISTING

```
3061        DECLARE CUR_DF                    FIXED BINARY ALIGNED;
3062        DECLARE (I,L,TAB_POS,TAB_AMT,PRINT_TAB_AMT,PRINT_LAST_PCT)
3063                                          FIXED BINARY ALIGNED;
3064        DECLARE (ACCUM,REGISTER,COMP_A,COMP_B)
3065                                          FLOAT DECIMAL;
3066        DECLARE ABNORMAL_STOP             BIT(1) ALIGNED INIT('0'B);
3067        DECLARE (ACCUM_TYPE,ACCUM_STR,
3068                 COMP_RESULT,
3069                 COMP_A_TYPE,COMP_A_STR,
3070                 COMP_B_TYPE,COMP_B_STR)  FIXED BINARY ALIGNED;
3071        DECLARE (COMP_RESULT_LT           INITIAL(1),
3072                 COMP_RESULT_EQ           INITIAL(2),
3073                 COMP_RESULT_GT           INITIAL(3))
3074                                          STATIC FIXED BINARY ALIGNED;
3075        DECLARE 1 GOSUB_STACK       ALIGNED,
3076                  2 (GS_CURM,
3077                     GS_MAX)        FIXED BINARY,
3078                  2  GOSUB_AREA(25),
3079                     3  GS_LINE     FIXED BINARY,
3080                     3  GS_PTR      FIXED BINARY;
3081         DECLARE 1 FOR_STACK        ALIGNED,
3082                   2 (FS_CUR,
3083                      FS_MAX)       FIXED BINARY,
3084                   2  FS_AREA(10),
3085                      3  FS_CTL_VAR FIXED BINARY,
3086                     (3  FS_START,
3087                      3  FS_LIMIT,
3088                      3  FS_STEP)   FLOAT DECIMAL,
3089                      3  FS_INST    FIXED BINARY;
3090
3091         DECLARE PRINT_WORK               PICTURE '(6)-9.V(6)9',
3092                 PRINT_WORK_CHAR(14) DEFINED PRINT_WORK
3093                                     CHAR(1);
3094         DECLARE PRINT_E_FORMAT           CHAR(14);
3095         DECLARE PRINT_ZERO               CHAR(2) VARYING INITIAL(' 0');
3096
3097         DECLARE PRINT_FIELD        CHAR(14) VARYING;
3098         DECLARE 1 PRINT_AREA,
3099                   2  PRINT_LINE    CHAR(120) VARYING;
3100
3101         P_CTR,P_PTR = 0;
3102         P_CTR_MAX = $MAX_EXECS;
3103         ACCUM,REGISTER=0.0;
3104         ACCUM_TYPE=0;
3105         COMP_A,COMP_B=0.0;
```

MACRO SOURCE2 LISTING

```
3106         COMP_A_TYPE,COMP_B_TYPE,COMP_A_STR,COMP_B_STR=0;
3107         GS_CUR,GS_MAX=0;
3108         FS_CUR,FS_MAX=0;
3109         DSL_REG=0;
3110         CUR_DEF=0;
3111
3112         PRINT_LINE='';
3113         PRINT_TAB_AMT=0;
3114         PRINT_LAST_PCT=0;
3115
3116         ON ERROR
3117         BEGIN;
3118             PUT SKIP LIST('FATAL BASIC INTERPRETER ERROR');
3119             PUT SKIP DATA(P_CTR,P_PTR);
3120             PUT SKIP DATA(PC_OPCODE(P_PTR),PC_OBJECT(P_PTR));
3121             PUT SKIP DATA(DSL_REG,OFFSET_VAL);
3122             PUT SKIP DATA(ACCUM,REGISTER);
3123             PUT SKIP DATA(COMP_A,COMP_B,COMP_A_TYPE,COMP_B_TYPE);
3124             PUT SKIP DATA(GS_CUR,GS_MAX);
3125             PUT SKIP DATA(FS_CUR,FS_MAX);
3126             TABLE_DUMP=TRUE;
3127             CALL TERMINATE;
3128             STOP;
3129         END;
3130    P_CODE_NEXT:
3131         P_PTR=P_PTR+1;
3132    P_CODE_JUMP:
3133         P_CTR=P_CTR+1;
3134
3135         IF ABNORMAL_STOP THEN
3136             RETURN;
3137
3138         IF P_CTR>P_CTR_MAX THEN
3139         DO;
3140            CALL PRINT_ERR('**** PROGRAM ABORTED AFTER EXECUTING ' ||
3141                           P_CTR_MAX ||' INSTRUCTIONS ****');
3142            RETURN;
3143         END;
3144
3145         IF P_PTR>PC_MAX THEN
3146         DO;
3147            CALL PRINT_ERR('**** PROGRAM RUN AWAY DETECTED ****');
3148            RETURN;
3149         END;
3150
```

MACRO SOURCE2 LISTING

```
3151    /*********************************************************************
3152    *                                                                    *
3153    *  SUBSCRIPT CHECKING PATCH STARTS HERE                              *
3154    *                                                                    *
3155    *********************************************************************/
3156         IF DSL_REG > 0 THEN
3157            IF DSL_REG > SYM_DIM_MAX(PC_OBJECT(P_PTR)) THEN
3158            DO;
3159                CALL PRINT_ERR('**** SUBSCRIPT TOO LARGE ****');
3160                RETURN;
3161            END;
3162         ELSE
3163            IF DSL_REG < 0  THEN
3164            DO;
3165                CALL PRINT_ERR('**** SUBSCRIPT ENCOUNTERED ****');
3166                RETURN;
3167            END;
3168    /*********************************************************************
3169    *                                                                    *
3170    *  SUBSCRIPT CHECKING PATCH ENDS HERE                                *
3171    *                                                                    *
3172    *********************************************************************/
3173
3174         OFFSET_VAL=PC_OBJECT(P_PTR)+DSL_REG;
3175         IF EXECUTION_DEBUG THEN
3176              PUT SKIP DATA(OFFSET_VAL,P_PTR,DSL_REG,PC_OBJECT(P_PTR));
3177         DSL_REG=0;
3178
3179    /*********************************************************************
3180    *                                                                    *
3181    *  ENFORCE OBJECT TYPING IF OPCODE REQUIRES IT                       *
3182    *                                                                    *
3183    *********************************************************************/
3184         SELECT(PC_ALLOW(PC_OPCODE(P_PTR)))
3185         WHEN('00'B)
3186            P_PTR_SUB = 0;
3187         WHEN('01'B)
3188            IF SYM_TYPE(PC_OBJECT(P_PTR)) < SS_STRCON THEN
3189               P_PTR_SUB = 1;
3190            ELSE
3191            DO;
3192               CALL PRINT_ERR('**** STRING NOT ALLOWED ****');
3193               RETURN;
3194            END;
3195         WHEN('10'B)
```

MACRO SOURCE2 LISTING

```
3196            IF SYM_TYPE(PC_OBJECT(P_PTR)) >= SS_STRCON THEN
3197                P_PTR_SUB = 2;
3198            ELSE
3199            DO;
3200               CALL PRINT_ERR('**** NUMBER NOT ALLOWED ****');
3201               RETURN;
3202            END;
3203         WHEN('11'B)
3204            IF SYM_TYPE(PC_OBJECT(P_PTR)) < SS_STRCON THEN
3205               P_PTR_SUB = 1;
3206            ELSE
3207               P_PTR_SUB = 2;
3208         OTHERWISE
3209            CALL PRINT_ERR
3210                 ('*** FATAL ERROR - TYPE ENFORCEMENT FAILED ***');
3211            SIGNAL ERROR;
3212         ENDSELECT
3213
3214
3215         GO TO PC_INST(PC_OPCODE(P_PTR));
3216
3217    /*     PCODE SLN - SET LINE NUMBER    */
3218
3219    PC_INST(0):
3220         CUR_LN=LS_LINE(PC_OBJECT(P_PTR));
3221         IF EXECUTION_DEBUG THEN
3222            PUT SKIP DATA(CUR_LN);
3223         GO TO P_CODE_NEXT;
3224
3225    /*     PCODE LDA - LOAD ACCUMULATOR    */
3226
3227    PC_INST(1):
3228         ACCUM=SYM_VALUE(OFFSET_VAL);
3229         ACCUM_TYPE=SYM_TYPE(OFFSET_VAL);
3230         ACCUM_STR=OFFSET_VAL;
3231         IF EXECUTION_DEBUG THEN
3232            PUT SKIP DATA(ACCUM,
3233                          SYMBOL(OFFSET_VAL),SYM_TYPE(OFFSET_VAL));
3234         GO TO P_CODE_NEXT;
3235
3236    /*     PCODE LDR - LOAD REGISTER       */
3237
3238    PC_INST(33):
3239         REGISTER=SYM_VALUE(OFFSET_VAL);
3240         IF EXECUTION_DEBUG THEN
```

MACRO SOURCE2 LISTING

```
3241            PUT SKIP DATA(REGISTER,SYMBOL(OFFSET_VAL));
3242         GO TO P_CODE_NEXT;
3243
3244    /*     PCODE STA - STORE ACCUMULATOR    */
3245
3246    PC_INST(2):
3247         IF ACCUM_TYPE >= SS_STRCON THEN
3248            IF SYM_TYPE(OFFSET_VAL) = SS_STRVAR |
3249               SYM_TYPE(OFFSET_VAL) = SS_STRDIM THEN
3250                STRING_VAL(OFFSET_VAL)=STRING_VAL(ACCUM_STR);
3251            ELSE
3252            DO;
3253               CALL PRINT_ERR('**** STRING CANNOT BE STORED ' ||
3254                              'IN A NUMERIC VARIABLE ****');
3255               RETURN;
3256            END;
3257         ELSE
3258            IF SYM_TYPE(OFFSET_VAL) < SS_STRCON THEN
3259              SYM_VALUE(OFFSET_VAL)=ACCUM;
3260            ELSE
3261               DO;
3262                  CALL PRINT_ERR('**** NUMBER CANNOT BE STORED ' ||
3263                                 'IN A STRING VARIABLE ****');
3264                  RETURN;
3265               END;
3266         IF EXECUTION_DEBUG THEN
3267            PUT SKIP DATA(ACCUM,SYMBOL_AREA(OFFSET_VAL),ACCUM_STR);
3268                      /* SYMBOL(OFFSET_VAL),SYM_TYPE(OFFSET_VAL)*/
3269         GO TO P_CODE_NEXT;
3270
3271    /*     PCODE STR - STORE REGISTER       */
3272
3273    PC_INST(34):
3274         SYM_VALUE(OFFSET_VAL)=REGISTER;
3275         IF EXECUTION_DEBUG THEN
3276            PUT SKIP DATA(REGISTER,SYMBOL(OFFSET_VAL));
3277         GO TO P_CODE_NEXT;
3278
3279    /*     PCODE EXP - RAISE ACCUMULATOR    */
3280
3281    PC_INST(3):
3282         ACCUM=ACCUM**SYM_VALUE(OFFSET_VAL);
3283         IF EXECUTION_DEBUG THEN
3284            PUT SKIP DATA(ACCUM,SYMBOL(OFFSET_VAL));
3285         GO TO P_CODE_NEXT;
```

MACRO SOURCE2 LISTING

```
3286
3287    /*     PCODE ADD - ADD TO ACCUMULATOR    */
3288
3289    PC_INST(4):
3290         ACCUM=ACCUM+SYM_VALUE(OFFSET_VAL);
3291         IF EXECUTION_DEBUG THEN
3292            PUT SKIP DATA(ACCUM);
3293         GO TO P_CODE_NEXT;
3294
3295    /*     PCODE SUB - SUBTRACT FROM ACCUMULATOR    */
3296
3297    PC_INST(5):
3298         ACCUM=ACCUM-SYM_VALUE(OFFSET_VAL);
3299         IF EXECUTION_DEBUG THEN
3300            PUT SKIP DATA(ACCUM);
3301         GO TO P_CODE_NEXT;
3302
3303    /*     PCODE MUL - MULTIPLY ACCUMULATOR    */
3304
3305    PC_INST(6):
3306         ACCUM=ACCUM*SYM_VALUE(OFFSET_VAL);
3307         IF EXECUTION_DEBUG THEN
3308            PUT SKIP DATA(ACCUM);
3309         GO TO P_CODE_NEXT;
3310
3311
3312    /*     PCODE DIV - DIVIDE ACCUMULATOR    */
3313
3314    PC_INST(7):
3315         IF SYM_VALUE(OFFSET_VAL)=0.0 THEN
3316         DO;
3317            CALL PRINT_ERR('**** DIVISION BY ZERO DETECTED ****');
3318            RETURN;
3319         END;
3320         ACCUM=ACCUM/SYM_VALUE(OFFSET_VAL);
3321         IF EXECUTION_DEBUG THEN
3322            PUT SKIP DATA(ACCUM);
3323         GO TO P_CODE_NEXT;
3324
3325    /*     PCODE RDV - READ VARIABLE    */
3326
3327    PC_INST(8):
3328         DS_CUR=DS_CUR+1;
3329         IF DS_CUR > DS_MAX THEN
3330         DO;
```

MACRO SOURCE2 LISTING

```
3331            CALL PRINT_ERR('*** NO DATA FOR ' ||
3332                           SYMBOL(OFFSET_VAL));
3333            RETURN;
3334         END;
3335         IF P_PTR_SUB = 1 THEN
3336            SYM_VALUE(OFFSET_VAL) = DS_ITEM(DS_CUR);
3337         ELSE
3338            STRING_VAL(OFFSET_VAL) = STRING_VAL(DS_STR(DS_CUR));
3339         GO TO P_CODE_NEXT;
3340
3341    /*     PCODE PRV - PRINT VARIABLE    */
3342
3343    PC_INST(9):
3344         IF P_PTR_SUB = 2 THEN     /*  IF ARGUMENT IS A STRING, GO TO */
3345             GO TO PC_INST(16);    /*     PRS TO PRINT IT             */
3346
3347         PRINT_FIELD=FORMAT_NUMBER(SYM_VALUE(OFFSET_VAL));
3348         CALL PRINT_BUFFER(PRINT_FIELD);
3349         GO TO P_CODE_NEXT;
3350
3351    /*     PCODE PRS - PRINT STRING   */
3352
3353    PC_INST(16):
3354
3355         CALL PRINT_BUFFER(STRING_VAL(PC_OBJECT(P_PTR)));
3356
3357         GO TO P_CODE_NEXT;
3358
3359    /*     PCODE PCT - PRINT CONTROL      */
3360
3361    PC_INST(10):
3362         SELECT(PC_OBJECT(P_PTR))
3363         WHEN(PCT_LFEED)
3364             CALL FLUSH_BUFFER;
3365             PRINT_TAB_AMT=0;
3366             PRINT_LAST_PCT=0;
3367         WHEN(PCT_TAB)
3368             PRINT_LAST_PCT=PCT_TAB;
3369         WHEN(PCT_NOTAB)
3370             PRINT_LAST_PCT=PCT_NOTAB;
3371         OTHERWISE
3372         DO;
3373            CALL PRINT_ERR
3374               ('**** UNDEFINED PRINT CONTROL VALUE ****');
3375            RETURN;
```

MACRO SOURCE2 LISTING

```
3376         END;
3377         ENDSELECT
3378         GO TO P_CODE_NEXT;
3379
3380    /*     PCODE FNC - FUNCTION CALL  */
3381
3382    /*****************************************************************
3383    *                                                                *
3384    * IMPORTANT NOTE - IF ANY CHANGES ARE MADE TO LIBRARY FUNCTIONS,    *
3385    * THIS PCODE FNC SHOULD MATCH THE CHANGES IN THE INITIALIZE PROC    *
3386    *                                                                *
3387    *****************************************************************/
3388
3389    PC_INST(11):
3390         IF PC_OBJECT(P_PTR) < LBOUND(LIB_FNC,1) |
3391            PC_OBJECT(P_PTR) > HBOUND(LIB_FNC,1) THEN
3392         DO;
3393             PUT SKIP(2) LIST('**** FATAL ERROR - UNDEFINED FUNCTION ',
3394                              SYMBOL(PC_OBJECT(P_PTR)),' ****');
3395             SIGNAL ERROR;
3396         END;
3397         IF EXECUTION_DEBUG THEN
3398            PUT SKIP LIST('FNC',SYMBOL(PC_OBJECT(P_PTR)),
3399                               SYM_VALUE(PC_OBJECT(P_PTR)),
3400                               ACCUM);
3401         GO TO LIB_FNC(PC_OBJECT(P_PTR));
3402    LIB_FNC(2):
3403         IF ACCUM < 0.0 THEN
3404         DO;
3405           CALL PRINT_ERR
3406               ('**** NEGATIVE VALUE IN SQR FUNCTION ****');
3407           RETURN;
3408         END;
3409         ACCUM=SQRT(ACCUM);
3410         SYM_VALUE(PC_OBJECT(P_PTR))=ACCUM;
3411         GO TO END_FNC;
3412    LIB_FNC(3):
3413         ACCUM=ABS(ACCUM);
3414         SYM_VALUE(PC_OBJECT(P_PTR))=ACCUM;
3415         GO TO END_FNC;
3416    LIB_FNC(4):
3417         PRINT_TAB_AMT=ACCUM;
3418         SYM_VALUE(PC_OBJECT(P_PTR))=ACCUM;
3419         IF PRINT_TAB_AMT<1 | PRINT_TAB_AMT>120 THEN
3420         DO;
```

MACRO SOURCE2 LISTING

```
3421           CALL PRINT_ERR
3422               ('**** INVALID VALUE IN TAB FUNCTION ****');
3423           RETURN;
3424         END;
3425         GO TO END_FNC;
3426    LIB_FNC(5):
3427         ACCUM=TRUNC(ACCUM);
3428         SYM_VALUE(PC_OBJECT(P_PTR))=ACCUM;
3429         GO TO END_FNC;
3430    LIB_FNC(6):
3431         ACCUM=COS(ACCUM);
3432         SYM_VALUE(PC_OBJECT(P_PTR))=ACCUM;
3433         GO TO END_FNC;
3434    LIB_FNC(7):
3435         ACCUM=SIN(ACCUM);
3436         SYM_VALUE(PC_OBJECT(P_PTR))=ACCUM;
3437         GO TO END_FNC;
3438    LIB_FNC(8):
3439         ACCUM=TAN(ACCUM);
3440         SYM_VALUE(PC_OBJECT(P_PTR))=ACCUM;
3441         GO TO END_FNC;
3442    LIB_FNC(9):
3443         ACCUM=RND(ACCUM);
3444         SYM_VALUE(PC_OBJECT(P_PTR))=ACCUM;
3445         GO TO END_FNC;
3446    LIB_FNC(10):
3447         IF ACCUM>=0.0 THEN
3448            ACCUM=TRUNC(ACCUM+0.5);
3449         ELSE
3450            ACCUM=TRUNC(ACCUM-0.5);
3451         SYM_VALUE(PC_OBJECT(P_PTR))=ACCUM;
3452       /* GO TO END_FNC; */
3453    END_FNC:
3454         IF EXECUTION_DEBUG THEN
3455            PUT SKIP LIST('FNC',SYMBOL(PC_OBJECT(P_PTR)),
3456                                SYM_VALUE(PC_OBJECT(P_PTR)),
3457                                ACCUM);
3458         GO TO P_CODE_NEXT;
3459
3460
3461    /*     PCODE END - END OF EXECUTION  */
3462
3463    PC_INST(12):
3464         CALL FLUSH_BUFFER;
3465         PUT SKIP(2) EDIT('**** PROGRAM EXECUTION COMPLETE - ',
```

MACRO SOURCE2 LISTING

```
3466                            P_CTR, ' INSTRUCTIONS EXECUTED ****')
3467                            (A,F(8),A);
3468         RETURN;
3469
3470    /*     PCODE B - BRANCH   */
3471
3472    PC_INST(13):
3473    COMMON_BRANCH:
3474
3475         DO LS_CUR=1 TO LS_MAX;
3476             IF LS_LINE(LS_CUR)=PC_OBJECT(P_PTR) THEN
3477             DO;
3478               P_PTR=LS_OFFSET(LS_CUR);
3479               GOTO P_CODE_JUMP;
3480             END;
3481         END;
3482         CALL PRINT_ERR('**** LINE ' || PC_OBJECT(P_PTR) ||
3483                           ' NOT FOUND');
3484         RETURN;
3485
3486    /*     PCODE BAL - BRANCH AND LINK    */
3487
3488    PC_INST(14):
3489
3490         IF EXECUTION_DEBUG THEN
3491            PUT SKIP LIST('BRANCH AND LINK TO',PC_OBJECT(P_PTR));
3492         IF GS_MAX >= HBOUND(GS_LINE,1) THEN
3493         DO;
3494            CALL PRINT_ERR('**** TOO MANY ACTIVE GOSUBS ****');
3495            RETURN;
3496         END;
3497         IF GS_MAX > 0 THEN
3498         DO;
3499            DO GS_CUR=1 TO GS_MAX;
3500               IF GS_LINE(GS_CUR)=CUR_LN THEN
3501               DO;
3502                  CALL PRINT_ERR('**** RECURSIVE GOSUB ****');
3503                  RETURN;
3504               END;
3505            END;
3506         END;
3507         DO LS_CUR=1 TO LS_MAX;
3508             IF LS_LINE(LS_CUR)=PC_OBJECT(P_PTR) THEN
3509             DO;
3510               GS_MAX=GS_MAX+1;
```

MACRO SOURCE2 LISTING

```
3511              GS_LINE(GS_MAX)=CUR_LN;
3512              GS_PTR(GS_MAX)=P_PTR;
3513              P_PTR=LS_OFFSET(LS_CUR);
3514              PUT SKIP DATA(P_PTR);
3515              GOTO P_CODE_JUMP;
3516           END;
3517        END;
3518        CALL PRINT_ERR('**** LINE '||PC_OBJECT(P_PTR)||' NOT FOUND ****');
3519        RETURN;
3520
3521   /*     PCODE RET - RETURN TO LINK    */
3522
3523   PC_INST(15):
3524
3525        IF GS_MAX=0 THEN
3526        DO;
3527           CALL PRINT_ERR('**** RETURN WITHOUT A GOSUB ****');
3528           RETURN;
3529        END;
3530        P_PTR=GS_PTR(GS_MAX);
3531        GS_MAX=GS_MAX-1;
3532
3533        GO TO P_CODE_NEXT;
3534
3535   /*     PCODE PRS - PRINT STRING   */
3536
3537   /*  PC_INST(16):   MOVED TO FOLLOW PRV CODE 9 */
3538
3539   /*     PCODE LCA - LOAD COMPARATOR A   */
3540
3541   PC_INST(17):
3542        COMP_A=SYM_VALUE(OFFSET_VAL);
3543        COMP_A_TYPE=SYM_TYPE(OFFSET_VAL);
3544        COMP_A_STR=OFFSET_VAL;
3545        IF EXECUTION_DEBUG THEN
3546           PUT SKIP DATA(P_PTR_SUB,COMP_A,COMP_A_TYPE,COMP_A_STR);
3547        GO TO P_CODE_NEXT;
3548
3549   /*     PCODE LCB - LOAD COMPARATOR B   */
3550
3551   PC_INST(18):
3552
3553        COMP_B=SYM_VALUE(OFFSET_VAL);
3554        COMP_B_TYPE=SYM_TYPE(OFFSET_VAL);
3555        COMP_B_STR=OFFSET_VAL;
```

MACRO SOURCE2 LISTING

```
3556         IF EXECUTION_DEBUG THEN
3557            PUT SKIP DATA(P_PTR_SUB,COMP_B,COMP_B_TYPE,COMP_B_STR);
3558         GO TO P_CODE_NEXT;
3559
3560    /*     PCODE BEQ - BRANCH IF A=B   */
3561
3562    PC_INST(19):
3563
3564         CALL COMPARE_RTN;
3565         IF COMP_RESULT=COMP_RESULT_EQ THEN
3566            GO TO COMMON_BRANCH;
3567         ELSE
3568            GO TO P_CODE_NEXT;
3569
3570    /*     PCODE BNE - BRANCH IF A<>B   */
3571
3572    PC_INST(20):
3573
3574         CALL COMPARE_RTN;
3575         IF COMP_RESULT=COMP_RESULT_EQ THEN
3576            GO TO P_CODE_NEXT;
3577         ELSE
3578            GO TO COMMON_BRANCH;
3579
3580
3581    /*     PCODE BGT - BRANCH IF A>B   */
3582
3583    PC_INST(21):
3584
3585         CALL COMPARE_RTN;
3586         IF COMP_RESULT=COMP_RESULT_GT THEN
3587            GO TO COMMON_BRANCH;
3588         ELSE
3589            GO TO P_CODE_NEXT;
3590
3591
3592    /*     PCODE BLT - BRANCH IF A<B   */
3593
3594    PC_INST(22):
3595
3596         CALL COMPARE_RTN;
3597         IF COMP_RESULT=COMP_RESULT_LT THEN
3598            GO TO COMMON_BRANCH;
3599         ELSE
3600            GO TO P_CODE_NEXT;
```

MACRO SOURCE2 LISTING

```
3601
3602
3603    /*     PCODE BGE - BRANCH IF A>=B   */
3604
3605    PC_INST(23):
3606
3607         CALL COMPARE_RTN;
3608         IF COMP_RESULT=COMP_RESULT_GT |
3609            COMP_RESULT=COMP_RESULT_EQ THEN
3610            GO TO COMMON_BRANCH;
3611         ELSE
3612            GO TO P_CODE_NEXT;
3613
3614
3615    /*     PCODE BLE - BRANCH IF A<=B   */
3616
3617    PC_INST(24):
3618
3619         CALL COMPARE_RTN;
3620         IF COMP_RESULT=COMP_RESULT_LT |
3621            COMP_RESULT=COMP_RESULT_EQ THEN
3622            GO TO COMMON_BRANCH;
3623         ELSE
3624            GO TO P_CODE_NEXT;
3625
3626         GO TO P_CODE_NEXT;
3627
3628    /*     PCODE FSU - FOR NEXT SETUP   */
3629
3630    PC_INST(25):
3631
3632         IF FS_MAX = 0 THEN   /* SKIP IF NO ACTIVE FORS */
3633            FS_CUR,FS_MAX=1;
3634         ELSE
3635         DO;
3636            FS_CUR=1;
3637            DO WHILE(FS_CUR<=FS_MAX);
3638              IF FS_CTL_VAR(FS_CUR)=PC_OBJECT(P_PTR) THEN /* FOUND IT */
3639                 GO TO RECYCLE_FOR;
3640              ELSE
3641                 FS_CUR=FS_CUR+1;
3642            END;
3643            IF FS_MAX=HBOUND(FS_CTL_VAR,1) THEN
3644            DO;
3645               CALL PRINT_ERR('**** TOO MANY FOR NEXT LOOPS ****');
```

MACRO SOURCE2 LISTING

```
3646              RETURN;
3647           END;
3648           FS_MAX=FS_MAX+1;
3649           FS_CUR=FS_MAX;
3650        END;
3651    RECYCLE_FOR:
3652        FS_CTL_VAR(FS_CUR)=PC_OBJECT(P_PTR);
3653        FS_START(FS_CUR),FS_LIMIT(FS_CUR),FS_STEP(FS_CUR)=0;
3654        FS_INST(FS_CUR)=0;
3655        IF EXECUTION_DEBUG THEN
3656           PUT SKIP DATA(FS_AREA(FS_CUR));
3657        GO TO P_CODE_NEXT;
3658
3659    PC_INST(26):
3660        FS_START(FS_CUR)=SYM_VALUE(OFFSET_VAL);
3661        GO TO P_CODE_NEXT;
3662     PC_INST(27):
3663        FS_LIMIT(FS_CUR)=SYM_VALUE(OFFSET_VAL);
3664        GO TO P_CODE_NEXT;
3665    PC_INST(28):
3666        FS_STEP(FS_CUR)=SYM_VALUE(OFFSET_VAL);
3667        SYM_VALUE(FS_CTL_VAR(FS_CUR))=FS_START(FS_CUR);
3668        FS_INST(FS_CUR)=P_PTR;
3669        IF EXECUTION_DEBUG THEN
3670           PUT SKIP DATA(FS_AREA(FS_CUR));
3671        GO TO P_CODE_NEXT;
3672
3673    PC_INST(29):
3674        IF FS_MAX = 0 THEN   /* ERROR IF NO ACTIVE FORS */
3675        DO;
3676           CALL PRINT_ERR('**** NEXT WITH NO FOR ****');
3677           RETURN;
3678        END;
3679        FS_CUR=1;
3680        DO WHILE(FS_CUR<=FS_MAX);
3681           IF FS_CTL_VAR(FS_CUR)=PC_OBJECT(P_PTR) THEN /* FOUND IT */
3682              GO TO FOUND_FOR;
3683           ELSE
3684              FS_CUR=FS_CUR+1;
3685        END;
3686        CALL PRINT_ERR('**** NEXT WITH NO FOR ****');
3687        RETURN;
3688    FOUND_FOR:
3689        IF FS_STEP(FS_CUR)=0.0 THEN
3690        DO;
```

MACRO SOURCE2 LISTING

```
3691            CALL PRINT_ERR('**** ENDLESS LOOP DETECTED - STEP IS 0 ****');
3692            RETURN;
3693         END;
3694         FS_START(FS_CUR)=FS_START(FS_CUR)+FS_STEP(FS_CUR);
3695         SYM_VALUE(FS_CTL_VAR(FS_CUR))=FS_START(FS_CUR);
3696         IF FS_STEP(FS_CUR)>0.0 THEN
3697         DO;
3698            IF FS_START(FS_CUR)>FS_LIMIT(FS_CUR) THEN   /* LOOP DONE? */
3699               CALL COMPRESS_FS;
3700            ELSE
3701               P_PTR=FS_INST(FS_CUR);
3702         END;
3703         ELSE
3704         DO;
3705            IF FS_START(FS_CUR)<FS_LIMIT(FS_CUR) THEN   /* LOOP DONE? */
3706               CALL COMPRESS_FS;
3707            ELSE
3708               P_PTR=FS_INST(FS_CUR);
3709         END;
3710         GO TO P_CODE_NEXT;
3711
3712    /*     PCODE PTB - PRINT TAB */
3713
3714    PC_INST(30):
3715
3716         GO TO P_CODE_NEXT;
3717
3718    /*     PCODE RST - RESTORE DATA */
3719
3720    PC_INST(31):
3721
3722         DS_CUR=0;
3723
3724         GO TO P_CODE_NEXT;
3725
3726    /*     PCODE DSL - DIM SUBSCRIPT LOCATOR */
3727
3728    PC_INST(32):
3729
3730         DSL_REG=REGISTER;
3731         IF EXECUTION_DEBUG THEN
3732            PUT SKIP DATA(DSL_REG,REGISTER);
3733         GO TO P_CODE_NEXT;
3734
3735    /*     PCODE JMP - JUMP   */
```

MACRO SOURCE2 LISTING

```
3736
3737    PC_INST(35):
3738
3739         P_PTR=PC_OBJECT(P_PTR);
3740         GO TO P_CODE_JUMP;
3741
3742    /*     PCODE CFN - CALL A DEF FUNCTION   */
3743
3744    PC_INST(36):
3745         DO I=1 TO DF_MAX;
3746            IF DF_NAME(I)=SYMBOL(PC_OBJECT(P_PTR)) THEN
3747            DO;
3748               DF_RETURN(I)=P_PTR;
3749               P_PTR=DF_OFFSET(I);
3750               CUR_DEF=I;
3751               GO TO P_CODE_JUMP;
3752            END;
3753         END;
3754         CALL PRINT_ERR('**** USER FUNCTION NOT FOUND ****');
3755         RETURN;
3756
3757    /*     PCODE RFN - RETURN FROM FUNCTION */
3758
3759    PC_INST(37):
3760
3761         P_PTR=DF_RETURN(CUR_DEF);
3762         CUR_DEF=0;
3763         GO TO P_CODE_NEXT;
3764
3765    /*     PCODE STP - STOP EXECUTION  */
3766
3767    PC_INST(38):
3768
3769         CALL PRINT_ERR('**** STOP STATEMENT EXECUTED ****');
3770         RETURN;
3771
3772    COMPARE_RTN:PROC;
3773    /*****************************************************************
3774    *                                                               *
3775    *    THIS ROUTINE DOES ALL THE COMPARES AND SETS A LT, EQ, GT IND.  *
3776    *                                                               *
3777    *    NUMBERIC ITEMS WILL BE COMPARED AND THE LT,EQ, OR GT INDICATOR *
3778    *         SET.                                                  *
3779    *    STRINGS WILL BE COMPARED.  STRINGS OF UNEQUAL LENGTH WILL BE   *
3780    *         PADDED WITH SPACES SO THE LENGTHS WILL BE EQUAL.          *
```

MACRO SOURCE2 LISTING

```
3781    *         A RESULT WILL BE SET TO LT, EQ, OR GT.                  *
3782    *                                                                 *
3783    * NESTING:EXECUTION                                              *
3784    *****************************************************************/
3785
3786         COMP_RESULT=0;
3787
3788         /*  COMPARE NUMERIC ITEMS  */
3789
3790         IF COMP_A_TYPE < SS_STRCON & COMP_B_TYPE < SS_STRCON THEN
3791         DO;
3792            IF COMP_A < COMP_B THEN
3793               COMP_RESULT=COMP_RESULT_LT;
3794            ELSE
3795               IF COMP_A = COMP_B THEN
3796                  COMP_RESULT=COMP_RESULT_EQ;
3797               ELSE
3798                  COMP_RESULT=COMP_RESULT_GT;
3799         END;
3800         ELSE    /*  COMPARE STRING ITEMS  */
3801         DO;
3802            IF EXECUTION_DEBUG THEN
3803               PUT SKIP DATA(SS_STRCON,STRING_VAL(COMP_A_STR),
3804                                 STRING_VAL(COMP_B_STR));
3805            IF COMP_A_TYPE >= SS_STRCON & COMP_B_TYPE >= SS_STRCON THEN
3806            DO;
3807              IF LENGTH(STRING_VAL(COMP_A_STR)) =
3808                 LENGTH(STRING_VAL(COMP_B_STR)) THEN
3809              DO;
3810                 IF STRING_VAL(COMP_A_STR) <
3811                    STRING_VAL(COMP_B_STR) THEN
3812                       COMP_RESULT=COMP_RESULT_LT;
3813                 ELSE
3814                    IF STRING_VAL(COMP_A_STR) =
3815                       STRING_VAL(COMP_B_STR) THEN
3816                          COMP_RESULT=COMP_RESULT_EQ;
3817                    ELSE
3818                          COMP_RESULT=COMP_RESULT_GT;
3819              END;
3820              ELSE
3821                  CALL COMPARE_DIF_LEN;
3822            END;
3823            ELSE  /* OH OH - CANNOT MIX NUMBERS AND STRINGS */
3824            DO;
3825               CALL PRINT_ERR('**** NUMBERS AND STRINGS CANNOT ' ||
```

MACRO SOURCE2 LISTING

```
3826                                'BE COMPARED ****');
3827              RETURN;
3828           END;
3829        END;
3830
3831    COMPARE_DIF_LEN:PROC;
3832        DECLARE (TEMP_A,TEMP_B)       CHAR(80) INITIAL((80)' ');
3833        TEMP_A=STRING_VAL(COMP_A_STR);
3834        TEMP_B=STRING_VAL(COMP_B_STR);
3835        IF TEMP_A < TEMP_B THEN
3836           COMP_RESULT = COMP_RESULT_LT;
3837        ELSE
3838           IF TEMP_A = TEMP_B THEN
3839              COMP_RESULT = COMP_RESULT_EQ;
3840           ELSE
3841              COMP_RESULT = COMP_RESULT_GT;
3842        IF EXECUTION_DEBUG THEN
3843           PUT SKIP DATA(COMP_RESULT,TEMP_A,TEMP_B);
3844    END COMPARE_DIF_LEN;
3845    END COMPARE_RTN;
3846
3847        DECLARE FORMAT_NUMBER ENTRY(FLOAT DECIMAL)
3848                              RETURNS(CHAR(14) VARYING);
3849    FORMAT_NUMBER:PROC(A_NUMBER) RETURNS(CHAR(14) VARYING);
3850    /*****************************************************************
3851    *                                                                *
3852    *    THIS ROUTINE CONVERT THE INTERNAL FLOATING POINT TO CHARACTER  *
3853    *                                                                *
3854    *    THIS ROUTINE IS CALLED BY PRINT_VAR AND THE STR FUNCTION TO    *
3855    *         DO THE CONVERSION.                                     *
3856    *                                                                *
3857    * NESTING:EXECUTION                                             *
3858    *****************************************************************/
3859
3860        DECLARE A_NUMBER        FLOAT DECIMAL,
3861                PRINT_FIELD     CHAR(14) VARYING;
3862
3863        IF ABS(A_NUMBER) >= 1.0E+6 |
3864           ABS(A_NUMBER) <= 1.0E-6 THEN
3865        DO;
3866           IF A_NUMBER = 0.0 THEN
3867               PRINT_FIELD=PRINT_ZERO;
3868           ELSE
3869           DO;
3870               PRINT_E_FORMAT=A_NUMBER;
```

MACRO SOURCE2 LISTING

```
3871                PRINT_FIELD=SUBSTR(PRINT_E_FORMAT,1,12);
3872             END;
3873          END;
3874          ELSE
3875          DO;
3876             PRINT_WORK=A_NUMBER;
3877             I=1;
3878             DO WHILE(PRINT_WORK_CHAR(I)=' ');
3879               I=I+1;
3880             END;
3881             IF PRINT_WORK_CHAR(I)='-' THEN;
3882             ELSE
3883               I=I-1;
3884             J=14;
3885             DO WHILE(PRINT_WORK_CHAR(J)='0');
3886               J=J-1;
3887             END;
3888             IF PRINT_WORK_CHAR(J)='.' THEN
3889               J=J-1;
3890             PRINT_FIELD=SUBSTR(PRINT_WORK,I,J-I+1);
3891          END;
3892          RETURN(PRINT_FIELD);
3893
3894    END FORMAT_NUMBER;
3895
3896    COMPRESS_FS:PROC;
3897    /*********************************************************************
3898    *                                                                    *
3899    *                                                                    *
3900    * NESTING:EXECUTION                                                  *
3901    *********************************************************************/
3902          DECLARE (I,T,B) FIXED BINARY ALIGNED;
3903
3904          IF FS_MAX<=1 THEN
3905          DO;
3906             FS_MAX,FS_CUR=0;
3907             RETURN;
3908          END;
3909          ELSE
3910             IF FS_CUR=FS_MAX THEN
3911             DO;
3912                FS_AREA(FS_MAX)=0;
3913                FS_MAX=FS_MAX-1;
3914                RETURN;
3915             END;
```

MACRO SOURCE2 LISTING

```
3916
3917      /*  DETERMINE TOP AND BOTTOM ROWS IN FS_AREA TO MOVE  */
3918
3919         IF FS_CUR=1 THEN
3920         DO;
3921            T=2;
3922            B=FS_MAX;
3923         END;
3924         ELSE
3925         DO;
3926            T=FS_CUR+1;
3927            B=FS_MAX;
3928         END;
3929
3930         DO I=T TO B;
3931            FS_AREA(I-1)=FS_AREA(I);
3932         END;
3933
3934         FS_MAX=FS_MAX-1;
3935
3936    END COMPRESS_FS;
3937
3938
3939    PRINT_BUFFER:PROC(ITEM);
3940    /*********************************************************************
3941    *                                                                    *
3942    *                                                                    *
3943    * NESTING:EXECUTION                                                  *
3944    *********************************************************************/
3945       DECLARE ITEM                CHAR(*) VARYING;
3946       DECLARE NEXT_TAB            FIXED BINARY ALIGNED;
3947       DECLARE BLANKS              CHAR(120) STATIC INITIAL((120)' ');
3948       DECLARE COL_WIDTH           STATIC FIXED BINARY ALIGNED
3949                                              INITIAL(14);
3950
3951       IF PRINT_TAB_AMT > 0 THEN           /*  THIS IMPLEMENTS TAB() */
3952       DO;
3953          IF LENGTH(PRINT_LINE) < PRINT_TAB_AMT THEN
3954             PRINT_LINE=PRINT_LINE ||
3955                    SUBSTR(BLANKS,1,PRINT_TAB_AMT-LENGTH(PRINT_LINE));
3956          ELSE
3957             IF LENGTH(PRINT_LINE) > PRINT_TAB_AMT THEN
3958             DO;
3959                CALL FLUSH_BUFFER;
3960                PRINT_LINE=SUBSTR(BLANKS,1,PRINT_TAB_AMT);
```

MACRO SOURCE2 LISTING

```
3961             END;
3962             ELSE;  /*  NO ACTION NEEDED ON = */
3963
3964          IF LENGTH(PRINT_LINE)+LENGTH(ITEM) > 120 THEN
3965             CALL FLUSH_BUFFER;
3966
3967          PRINT_LINE=PRINT_LINE || ITEM;
3968
3969          PRINT_TAB_AMT = 0;
3970          PRINT_LAST_PCT = 0;    /*  TAB() OVERRIDES PCT */
3971       END;
3972       ELSE
3973       DO;
3974          IF LENGTH(PRINT_LINE)+LENGTH(ITEM) > 120 THEN
3975             CALL FLUSH_BUFFER;
3976
3977          IF LENGTH(PRINT_LINE) > 0  & PRINT_LAST_PCT = PCT_TAB THEN
3978          DO;
3979             NEXT_TAB=LENGTH(PRINT_LINE)/COL_WIDTH;
3980             NEXT_TAB=(NEXT_TAB+1)*COL_WIDTH;
3981             NEXT_TAB=NEXT_TAB-LENGTH(PRINT_LINE);
3982             IF NEXT_TAB>0 THEN
3983             DO;
3984                IF LENGTH(PRINT_LINE)+NEXT_TAB > 120 THEN
3985                   CALL FLUSH_BUFFER;
3986                ELSE
3987                   PRINT_LINE=PRINT_LINE || SUBSTR(BLANKS,1,NEXT_TAB);
3988             END;
3989          END;
3990
3991          PRINT_LINE=PRINT_LINE || ITEM;
3992       END;
3993    END PRINT_BUFFER;
3994
3995    PRINT_ERR:PROC(MSG);
3996    /*********************************************************************
3997    *                                                                    *
3998    *   PRINTS ALL ERROR MESSAGE FOR THE EXECUTION PHASE                 *
3999    *                                                                    *
4000    * NESTING:EXECUTION                                                  *
4001    *********************************************************************/
4002       DECLARE MSG                 CHAR(*);
4003       CALL FLUSH_BUFFER;
4004       PUT SKIP(2) EDIT('**** PROGRAM EXECUTION TERMINATED IN LINE',
4005                        CUR_LN) (A,F(6));
```

MACRO SOURCE2 LISTING

```
4006       IF TABLE_DUMP | TABLE_PRINT | ICODE_PRINT THEN
4007          PUT EDIT(' @ OFFSET',P_PTR) (A,F(6));
4008       PUT EDIT(' ****',MSG) (A,SKIP,A);
4009       ABNORMAL_STOP=TRUE;    /* FOR END OF PROGRAM EXECUTION */
4010    END PRINT_ERR;
4011
4012    FLUSH_BUFFER:PROC;
4013    /*********************************************************************
4014    *                                                                    *
4015    *                                                                    *
4016    * NESTING:EXECUTION                                                  *
4017    *********************************************************************/
4018
4019       PUT SKIP EDIT(PRINT_LINE) (A);
4020       PRINT_LINE='';
4021
4022    END FLUSH_BUFFER;
4023
4024    /*********************************************************************
4025    *  RND FUNCTION                                                      *
4026    *                                                                    *
4027    * NESTING:EXECUTION                                                  *
4028    *********************************************************************/
4029     %INCLUDE RNDGEN;
4030    END EXECUTE;
```

MACRO SOURCE2 LISTING

```
4031    /*****************************************************************/
4032    /*****************************************************************/
4033    /*****************************************************************/
4034    /*****************************************************************/
4035    /*****************************************************************/
4036
4037    TERMINATE:PROC;
4038
4039        DECLARE I                 FIXED BINARY ALIGNED;
4040
4041        IF TABLE_DUMP THEN
4042           CALL PRINT_SYMBOLS;
4043
4044    END TERMINATE;
4045
4046    PRINT_SYMBOLS:PROC;
4047
4048        PUT SKIP(2) LIST('OFFSET','SYMBOL','TYPE','OCCURS','VALUE');
4049        DO I=1 TO SS_MAX;
4050          PUT SKIP LIST(I,SYMBOL(I),SS_DESC(SYM_TYPE(I)),
4051                        SYM_DIM_MAX(I));
4052          IF SYM_TYPE(I) = SS_STRCON |
4053             SYM_TYPE(I) = SS_STRVAR |
4054             SYM_TYPE(I) = SS_UNKNWN |
4055             SYM_TYPE(I) = SS_STRDIM THEN
4056             PUT LIST(STRING_VAL(I));
4057          ELSE
4058             PUT LIST(SYM_VALUE(I));
4059        END;
4060        PUT SKIP LIST('END OF SYMBOL TABLE');
4061
4062    END PRINT_SYMBOLS;
4063
4064    END BASIC;
```

INCLUDED TEXT FOLLOWS FROM DD.MEMBER = SYSLIB .FIX2STR

```
4065    /******************************************************************** 00000001
4066                                                                         00000002
4067          THE FIX2STR MACRO - CONVERT A FIXED ITEM TO A STRING           00000003
4068                      FOR USE WITH PL/I (F).                             00000004
4069                                                                         00000005
4070    ********************************************************************/00000006
4071    %DECLARE                                                             00000007
```

MACRO SOURCE2 LISTING

```
4072        FIX2STR   ENTRY (FIXED) RETURNS (CHARACTER);                        00000008
4073    %FIX2STR :                                                              00000009
4074        PROCEDURE (N) RETURNS (CHARACTER);                                  00000010
4075                                                                            00000011
4076          DECLARE  (N,ABSN)      FIXED,                                     00000012
4077                   NSTR          CHARACTER;                                 00000013
4078          NSTR = N;                                                         00000014
4079          IF N < 0 THEN                                                     00000015
4080              ABSN = N * -1;                                                00000016
4081          ELSE                                                              00000017
4082              ABSN = N;                                                     00000018
4083          NSTR = ABSN;   /* CONVERSION RESULTS IN 99999999 FORMAT */        00000019
4084          IF ABSN < 10 THEN                                                 00000020
4085                NSTR = SUBSTR(NSTR,8,1);                                    00000021
4086          ELSE                                                              00000022
4087          IF ABSN < 100 THEN                                                00000023
4088                NSTR = SUBSTR(NSTR,7,2);                                    00000024
4089          ELSE                                                              00000025
4090          IF ABSN < 1000 THEN                                               00000026
4091                NSTR = SUBSTR(NSTR,6,3);                                    00000027
4092          ELSE                                                              00000028
4093          IF ABSN < 10000 THEN                                              00000029
4094                NSTR = SUBSTR(NSTR,5,4);                                    00000030
4095          ELSE                                                              00000031
4096                NSTR = '/* FIX2STR - NUMBER TOO LARGE */';                  00000032
4097          IF N<0 THEN                                                       00000033
4098             NSTR = '-' || NSTR;                                            00000034
4099          RETURN(NSTR);                                                     00000035
4100    %END;                                                                   00000036
```

INCLUDED TEXT FOLLOWS FROM DD.MEMBER = SYSLIB .SELECT

```
4101    /*********************************************************************** 00000001
4102                                                                            00000002
4103          THE SELECT, WHEN, BREAK, OTHERWISE AND ENDSELECT MACROS           00000003
4104                      FOR USE WITH PL/I (F).                                00000004
4105                                                                            00000005
4106      THESE MACROS EMULATE THE PL/I SELECT STATEMENT WHICH WAS NOT          00000006
4107      IMPLEMENTED IN THE PL/I (F) COMPILER.                                 00000007
4108                                                                            00000008
4109      TO USE, %INCLUDE THIS MEMBER IN YOUR PROGRAM AND COMPILE WITH         00000009
4110      THE MACRO OPTION. (NOMACRO IS THE DEFAULT).                           00000010
4111                                                                            00000011
4112      CODE THE SELECT:                                                      00000012
```

MACRO SOURCE2 LISTING

```
4113                                                                    00000013
4114          SELECT (A)                SELECT (TRUE)                 00000014
4115          WHEN(1)                   WHEN(A=1)                     00000015
4116            CALL SUB1;                CALL SUB1;                  00000016
4117          WHEN(X)                   WHEN(A=X)                     00000017
4118            CALL XRTN(X);             CALL XTRN(X);               00000018
4119            X=X+A;                    X=X+A;                      00000019
4120          ENDSELECT                 OTHERWISE                     00000020
4121                                      PUT SKIP LIST('ERROR');       00000021
4122                                    ENDSELECT                       00000022
4123                                                                    00000023
4124       NOTE - USING THE MACROS, THE ENDSELECT IS REQUIRED.          00000024
4125                                                                    00000025
4126       THE ALTERNATE SELECT WHEN BREAK IS ALSO SUPPORTED.  THIS ONE IS 00000026
4127       POPULAR WITH C, C++, JAVA, ETC PROGRAMMERS.                  00000027
4128                                                                    00000028
4129       THERE IS A COMPILE TIME VARIABLE DEFINED SELECT_TYPE.  BY DEFAULT 00000029
4130       IT IS SET TO 0. THIS INDICATES UPON COMPLETION OF A WHEN BLOCK, 00000030
4131       THE STATEMENT AFTER THE ENDSELECT WILL BE THE NEXT EXECUTED.  00000031
4132       IF SELECT_TYPE IS SET TO 1, UPON COMPLETION OF A WHEN BLOCK, THE 00000032
4133       NEXT WHEN WILL BE EXECUTED.  A BREAK MACRO IN A WHEN BLOCK    00000033
4134       WILL CAUSE THE STATEMENT AFTER THE ENDSELECT TO BE THE NEXT   00000034
4135       STATEMENT EXECUTED.                                          00000035
4136       TO SWITCH TYPE, CODE A %SELECT_TYPE=1; OR %SELECT_TYPE=0;    00000036
4137     ************************************************************************/00000037
4138                                                                    00000038
4139     %DECLARE                                                       00000039
4140         SELECT    ENTRY (CHARACTER) RETURNS (CHARACTER),           00000040
4141         WHEN      ENTRY (CHARACTER) RETURNS (CHARACTER),           00000041
4142         OTHERWISE ENTRY             RETURNS (CHARACTER),           00000042
4143         BREAK     ENTRY             RETURNS (CHARACTER),           00000043
4144         ENDSELECT ENTRY             RETURNS (CHARACTER);           00000044
4145                                                                    00000045
4146     %DECLARE                                                       00000046
4147          WHEN_CTR            FIXED,                              00000047
4148          SELECT_CTR          FIXED,                              00000048
4149          SELECT_TYPE         FIXED,                              00000049
4150          SELECT_NAME         CHARACTER,                          00000050
4151         (SELEXPR,SPACES)     CHARACTER;                          00000051
4152                                                                    00000052
4153       %WHEN_CTR = 0;                                               00000053
4154       %SELECT_CTR = 0;                                             00000054
4155       %SELECT_TYPE = 0;                                            00000055
4156       %SELECT_NAME = ' ';                                          00000056
4157       %SPACES = '                                                  00000057
```

MACRO SOURCE2 LISTING

```
4158                ';                                                          00000058
4159    /*****************************************************************/00000059
4160    %SELECT :                                                           00000060
4161        PROCEDURE (A) RETURNS (CHARACTER);                              00000061
4162                                                                        00000062
4163          DECLARE  (A,L1,L2) CHARACTER;                                 00000063
4164                                                                        00000064
4165          IF WHEN_CTR > 0 THEN                                          00000065
4166              RETURN('/*** THE SELECT MACRO CANNOT BE NESTED ***/');    00000066
4167                                                                        00000067
4168          IF SELECT_TYPE < 0 | SELECT_TYPE > 1 THEN                     00000068
4169              RETURN('/*** SELECT_TYPE MUST BE 0 OR 1 ***/');           00000069
4170                                                                        00000070
4171          SELECT_CTR = SELECT_CTR+1;                                    00000071
4172          SELECT_NAME = FIX2STR(SELECT_CTR);                            00000072
4173          SELECT_NAME = ' ENDSELECT_MACRO' || SELECT_NAME;              00000073
4174          SELEXPR = A;                                                  00000074
4175          WHEN_CTR = 1;                                                 00000075
4176                                                                        00000076
4177          L1 = '       /*';                                             00000077
4178          L2 = 'SELECT (' || SELEXPR || ')   */ DO;';                   00000078
4179                                                                        00000079
4180          L1 = SUBSTR(L1||SPACES,1,71);                                 00000080
4181                                                                        00000081
4182          RETURN(L1||L2);                                               00000082
4183    %END;                                                               00000083
4184    /*****************************************************************/00000084
4185    %WHEN :                                                             00000085
4186        PROCEDURE (A) RETURNS (CHARACTER);                              00000086
4187                                                                        00000087
4188          DECLARE  (A,L1,L2,L4,L5) CHARACTER;                           00000088
4189                                                                        00000089
4190          IF WHEN_CTR > 1 THEN                                          00000090
4191          DO;                                                           00000091
4192             IF SELECT_TYPE = 0 THEN                                    00000092
4193                L1='             GO TO ' || SELECT_NAME || '; END; /*';  00000093
4194             ELSE                                                       00000094
4195                L1='             END; /*';                             00000095
4196             L2='WHEN (' || A || ')    */';                            00000096
4197          END;                                                          00000097
4198          ELSE                                                          00000098
4199          IF WHEN_CTR = 1 THEN                                          00000099
4200          DO;                                                           00000100
4201             L1='                /*';                                 00000101
4202             L2='WHEN (' || A || ')    */';                            00000102
```

MACRO SOURCE2 LISTING

```
4203          END;                                                                00000103
4204          ELSE                                                                00000104
4205             RETURN('/** WHEN WITHOUT A SELECT **/');                         00000105
4206                                                                              00000106
4207          WHEN_CTR = WHEN_CTR + 1;                                            00000107
4208                                                                              00000108
4209          IF SELEXPR = 'TRUE' THEN                                            00000109
4210             L4='         IF (' || A || ') THEN';                             00000110
4211          ELSE                                                                00000111
4212             L4='         IF (' || SELEXPR || ')=(' || A || ') THEN';         00000112
4213          L5='                      DO;';                                     00000113
4214                                                                              00000114
4215          L1 = SUBSTR(L1||SPACES,1,71);                                       00000115
4216          L2 = SUBSTR(L2||SPACES,1,71);                                       00000116
4217          L4 = SUBSTR(L4||SPACES,1,71);                                       00000117
4218                                                                              00000118
4219          RETURN(L1||L2||L4||L5);                                             00000119
4220     %END ;                                                                   00000120
4221     /*************************************************************************/00000121
4222     %BREAK :                                                                 00000122
4223         PROCEDURE RETURNS (CHARACTER);                                       00000123
4224           DECLARE  (L1,L2) CHARACTER;                                        00000124
4225           IF SELECT_TYPE = 0 THEN                                            00000125
4226              RETURN('/*** BREAK NOT ALLOWED IN THIS SELECT TYPE ***/');      00000126
4227           L1='                  /*';                                        00000127
4228           L2='BREAK         */   GO TO ' || SELECT_NAME ||';';               00000128
4229           L1 = SUBSTR(L1||SPACES,1,71);                                      00000129
4230                                                                              00000130
4231           RETURN(L1||L2);                                                    00000131
4232     %END;                                                                    00000132
4233     /*************************************************************************/00000133
4234     %OTHERWISE :                                                             00000134
4235         PROCEDURE RETURNS (CHARACTER);                                       00000135
4236                                                                              00000136
4237           DECLARE  (L1,L2) CHARACTER;                                        00000137
4238                                                                              00000138
4239           IF WHEN_CTR < 1 THEN                                               00000139
4240              RETURN('/** OTHERWISE OUT OF SEQUENCE **/');                    00000140
4241                                                                              00000141
4242           L1='              END; /*';                                        00000142
4243           L2='OTHERWISE     */  ELSE DO;';                                   00000143
4244                                                                              00000144
4245           L1 = SUBSTR(L1||SPACES,1,71);                                      00000145
4246           L2 = SUBSTR(L2||SPACES,1,71);                                      00000146
4247                                                                              00000147
```

MACRO SOURCE2 LISTING

```
4248          RETURN(L1||L2);                                                   00000148
4249     %END;                                                                  00000149
4250     /*********************************************************************/00000150
4251     %ENDSELECT :                                                           00000151
4252         PROCEDURE RETURNS (CHARACTER);                                     00000152
4253                                                                            00000153
4254           DECLARE  (L1,L2) CHARACTER;                                      00000154
4255                                                                            00000155
4256           IF WHEN_CTR < 1 THEN                                             00000156
4257              RETURN('/** ENDSELECT OUT OF SEQUENCE **/');                  00000157
4258                                                                            00000158
4259           WHEN_CTR = 0;                                                    00000159
4260                                                                            00000160
4261           L1='              END; /*';                                      00000161
4262           L2='ENDSELECT     */  END;';                                     00000162
4263                                                                            00000163
4264           L2 = L2 || SELECT_NAME || ':;';                                  00000164
4265           L1 = SUBSTR(L1||SPACES,1,71);                                    00000165
4266                                                                            00000166
4267           RETURN(L1||L2);                                                  00000167
4268     %END;                                                                  00000168
```

INCLUDED TEXT FOLLOWS FROM DD.MEMBER = SYSLIB .GENPC

```
4269     /********************************************************************* 00000001
4270                                                                            00000002
4271       THIS MACRO GENERATES A SERIES OF ELEMENTS FOR CREATING A TABLE.      00000003
4272                                                                            00000004
4273       THE THIS MACRO IS CUSTOMIZED TO CREATE THE PSEUDO CODE (PC)          00000005
4274       CODE TABLE FOR BASIC/360.                                            00000006
4275                                                                            00000007
4276       TO USE, %INCLUDE THIS MEMBER IN YOUR PROGRAM AND COMPILE WITH        00000008
4277       THE MACRO OPTION. (NOMACRO IS THE DEFAULT).                          00000009
4278                                                                            00000010
4279       CODE:                                                                00000011
4280           GENPC(MMM,PPP,F,A)                                               00000012
4281                                                                            00000013
4282           WHERE                                                            00000014
4283             MMM IS THE OPCODE                                              00000015
4284             PPP IS THE OPCODE                                              00000016
4285             F   IS THE CODE FOR ACCEPTABLE OPERANDS                        00000017
4286             A   IS THE PERMISSIBLE TYPE OF OPERANDS AS BIT STRING.         00000018
4287                                                                            00000019
4288     *********************************************************************/00000020
```

MACRO SOURCE2 LISTING

```
4289                                                                    00000021
4290                                                                    00000022
4291    %DECLARE                                                        00000023
4292        GENPC     ENTRY (CHARACTER,CHARACTER,CHARACTER,CHARACTER)   00000024
4293                  RETURNS (CHARACTER);                              00000025
4294    %DECLARE                                                        00000026
4295        GENPC_CTR            FIXED;                                 00000027
4296        %GENPC_CTR = -1;                                            00000028
4297    %GENPC :                                                        00000029
4298        PROCEDURE (M,P,F,A) RETURNS (CHARACTER);                    00000030
4299                                                                    00000031
4300          DECLARE  (M,P,F,A,L1,L2,L3,L4,L5) CHARACTER;              00000032
4301          DECLARE  GSPACES     CHARACTER;                           00000033
4302          GSPACES = '                                               00000034
4303                    ';                                            00000035
4304                                                                    00000036
4305          L1 = '/* GENPC(' || M || ',' || P || ',' || F || ',' || A ||  00000037
4306               ') */';                                              00000038
4307          L2 = '2  PC_OPCODE_' || M || ' FIXED BINARY ' ||          00000039
4308               'INITIAL(' || P || '),';                             00000040
4309          L3 = '2  PC_MNCODE_' || M || ' CHAR(4)      ' ||          00000041
4310               'INITIAL(''' || M || '''),';                         00000042
4311          L4 = '2  PC_FORMAT_' || M || ' FIXED BINARY ' ||          00000043
4312               'INITIAL(' || F || '),';                             00000044
4313          L5 = '2  PC_ALLOWS_' || M || ' BIT(2)       ' ||          00000045
4314               'INITIAL(''' || A || '''B),';                        00000046
4315                                                                    00000047
4316          L1 = SUBSTR(L1||GSPACES,1,71);                            00000048
4317          L2 = SUBSTR(L2||GSPACES,1,71);                            00000049
4318          L3 = SUBSTR(L3||GSPACES,1,71);                            00000050
4319          L4 = SUBSTR(L4||GSPACES,1,71);                            00000051
4320          GENPC_CTR=GENPC_CTR+1;                                    00000052
4321                                                                    00000053
4322          RETURN(L1||L2||L3||L4||L5);                               00000054
4323    %END;                                                           00000055
4324    /*******************************************************************/00000056
```

INCLUDED TEXT FOLLOWS FROM DD.MEMBER = SYSLIB .GENSYM

```
4325    /******************************************************************* 00000001
4326                                                                        00000002
4327      THIS MACRO GENERATES A SERIES OF ELEMENTS FOR CREATING A TABLE.   00000003
4328                                                                        00000004
4329      THE THIS MACRO IS CUSTOMIZED TO INITIALIZE THE SYMBOL STACK (SS)  00000005
```

MACRO SOURCE2 LISTING

```
4330      CODE TABLE FOR BASIC/360.                                                  00000006
4331                                                                                 00000007
4332      TO USE, %INCLUDE THIS MEMBER IN YOUR PROGRAM AND COMPILE WITH              00000008
4333      THE MACRO OPTION. (NOMACRO IS THE DEFAULT).                                00000009
4334                                                                                 00000010
4335      CODE:                                                                      00000011
4336          GENSYM(S,T,F)                                                          00000012
4337                                                                                 00000013
4338          WHERE                                                                  00000014
4339            S   IS THE SYMBOL TO DEFINE                                         00000015
4340            T   IS THE SYMBOL TYPE                                             00000016
4341            VN  IS THE SYMBOL INITIAL NUMERIC VALUE                            00000017
4342            VS  IS THE SYMBOL INITIAL STRING VALUE                             00000018
4343                                                                               00000019
4344    ************************************************************************/00000020
4345                                                                               00000021
4346                                                                               00000022
4347    %DECLARE                                                                   00000023
4348        GENSYM    ENTRY (CHARACTER,CHARACTER,CHARACTER,CHARACTER)              00000024
4349                  RETURNS (CHARACTER);                                         00000025
4350    %DECLARE                                                                   00000026
4351        GENSYM_CTR            FIXED;                                           00000027
4352        %GENSYM_CTR = 0;                                                       00000028
4353    %GENSYM :                                                                  00000029
4354        PROCEDURE (S,T,VN,VS) RETURNS (CHARACTER);                             00000030
4355                                                                               00000031
4356          DECLARE  (S,T,VN,VS,L1,L2,L3,L4,L5) CHARACTER;                       00000032
4357          DECLARE  GSPACES     CHARACTER;                                      00000033
4358          GSPACES = '                                                          00000034
4359                    ';                                                         00000035
4360                                                                               00000036
4361          GENSYM_CTR=GENSYM_CTR+1;                                             00000037
4362          L1 = '/* GENSYM(' || S || ',' || T || ',' || VN || ','               00000038
4363                           || VS || ') */';                                    00000039
4364          L2 = 'SYMBOL    (' || GENSYM_CTR || ') = ''' || S || ''';';          00000040
4365                                                                               00000041
4366          L3 = 'SYM_TYPE  (' || GENSYM_CTR || ') = ' || T || ';';              00000042
4367                                                                               00000043
4368          L4 = 'SYM_VALUE (' || GENSYM_CTR || ') = ' || VN || ';';             00000044
4369                                                                               00000045
4370          L5 = 'STRING_VAL(' || GENSYM_CTR || ') = ''' || VS || ''';';         00000046
4371                                                                               00000047
4372          L1 = SUBSTR(L1||GSPACES,1,71);                                       00000048
4373          L2 = SUBSTR(L2||GSPACES,1,71);                                       00000049
4374          L3 = SUBSTR(L3||GSPACES,1,71);                                       00000050
```

MACRO SOURCE2 LISTING

```
4375          L4 = SUBSTR(L4||GSPACES,1,71);                                00000051
4376          L5 = SUBSTR(L5||GSPACES,1,71);                                00000052
4377                                                                        00000053
4378          RETURN(L1||L2||L3||L4||L5);                                   00000054
4379     %END;                                                              00000055
4380     /*******************************************************************/00000056
```

INCLUDED TEXT FOLLOWS FROM DD.MEMBER = SYSLIB .RNDGEN

MACRO SOURCE2 LISTING

```
4381    /*********************************************************************** 00000001
4382    /*********************************************************************** 00000002
4383    *                                                                     * 00000003
4384    *   RND - RANDOM NUMBER GENERATOR                                     * 00000004
4385    *                                                                     * 00000005
4386    *   THIS FUNCTION GENERATES 'RANDOM' NUMBERS BETWEEN 0.0 AND 1.0      * 00000006
4387    *   TO USE IT, SIMPLY INCLUDE %RNDGEN IN YOUR PROGRAM.                * 00000007
4388    *                                                                     * 00000008
4389    *   NOTE - THIS IS A SIMPLE RANDOM VALUE THAT IS NOT CONSIDERED       * 00000009
4390    *          STATISTICALY A RANDOM NUMBER.  A STATISTICALY RANDOM       * 00000010
4391    *          REQUIRES MULTIPLE RANDOM VALUES BE GENERATED, A MEAN BE    * 00000011
4392    *          DETERMINED AND STUFF BEYOND THE SCOPE OF THIS MODULE.      * 00000012
4393    *                                                                     * 00000013
4394    *   REVISION HISTORY         COMMENTS                                 * 00000014
4395    *   V1.0.0   01/20/2017      INITIAL VERSION TRANSLATED FROM          * 00000015
4396    *                            FORTRAN IV.                              * 00000016
4397    *                                                                     * 00000017
4398    *********************************************************************** 00000018
4399    ***********************************************************************/ 00000019
4400       DECLARE RND ENTRY(FLOAT DECIMAL) RETURNS(FLOAT DECIMAL);             00000020
4401    RND:PROC(DUMMY) RETURNS(FLOAT DECIMAL);                                 00000021
4402       DECLARE (DUMMY,YFL)     FLOAT DECIMAL;                               00000022
4403       DECLARE ZERO     FIXED BINARY STATIC INITIAL(0);                     00000023
4404       DECLARE ISW      FIXED BINARY STATIC INITIAL(0);                     00000024
4405       DECLARE (IX,IY)  FIXED BINARY(31) STATIC;                            00000025
4406                                                                            00000026
4407       IF ISW = ZERO THEN                                                   00000027
4408       DO;                                                                  00000028
4409       /* IY = 123875; */                                                   00000029
4410          IY = SUBSTR(TIME,4,6);    /* TIME IS HHMMSSTTT FORMAT */          00000030
4411          ISW = 1;                                                          00000031
4412       END;                                                                 00000032
4413       IF MOD(IY,2) = ZERO THEN                                             00000033
4414          IY = IY+1;                                                        00000034
4415       IX = IY;                                                             00000035
4416    (NOFIXEDOVERFLOW):                                                      00000036
4417       IY = IX * 65539;                                                     00000037
4418       IF IY < ZERO THEN                                                    00000038
4419       DO;                                                                  00000039
4420    (NOFIXEDOVERFLOW):                                                      00000040
4421          IY = IY + 2147483647 + 1;                                         00000041
4422       END;                                                                 00000042
4423       YFL = IY;                                                            00000043
4424       YFL = YFL * .4656613E-9;                                             00000044
4425       RETURN(YFL);                                                         00000045
```

MACRO SOURCE2 LISTING

```
4426    END RND;                                                    00000046
```

NO ERROR OR WARNING CONDITION HAS BEEN DETECTED FOR THIS MACRO PASS.

SOURCE LISTING.

```
STMT LEVEL NEST
                  /****** BASIC/360  V2.2  09/10/2017 *******/                  1
                  /****** BASIC/360  V2.1  08/22/2017 *******/                  2
                  /****** BASIC/360  V2.0  08/08/2016 *******/                  3
                 /*********************************************************************      4
                 *                                                                   *      4
                 *   SOUTH HAMMOND INSTITUTE OF TECHNOLOGY  BASIC/360   FALL 1974    *      4
                 *                                                                   *      4
                 *********************************************************************      4
                 *                                                                   *      4
                 *   IMPLEMENT A BASIC COMPILER/INTERPRETER FOR THE IBM/360          *      4
                 *   USING THE ORIGINAL DARTMOUTH SPECS FOR BASIC.  THE PRIMARY      *      4
                 *   INTENT IS TO CREATE A BASIC COMPILER/INTERPRETER FOR BEGINNING  *      4
                 *   STUDENTS TO LEARN THE BASIC LANGUAGE INSTEAD OF GOTRAN ON THE   *      4
                 *   SOON TO BE RETIRED 1620.                                        *      4
                 *                                                                   *      4
                 *   THE TARGET ENVIRONMENT IS A 32K IBM/360 MOD 30 RUNNING          *      4
                 *   DOS/360 AND PL/I(D) COMPILER.                                   *      4
                 *                                                                   *      4
                 *   STUDENTS MAY NOT BE COMPUTER MAJORS AND MOST PROGRAMS WOULD BE  *      4
                 *   SMALL, A SIMPLE MONITOR MONITOR WAS IMPLEMENTED SO THE LAB AID  *      4
                 *   OR INSTRUCTOR COULD ACTUALLY SUBMIT ALL THE BASIC PROGRAMS AS   *      4
                 *   ONE JOB.                                                        *      4
                 *                                                                   *      4
                 *   THIS PACKAGE IS BEING DESIGNED TO HAVE MODULAR SOURCE CODE      *      4
                 *   SINCE IT ENVISIONED THAT THIS PRODUCT WILL BE IMPLEMENTED       *      4
                 *   IN SEVERAL DIFFERENT ENVIRONMENTS                               *      4
                 *      1) SIMPLE BATCH - 1 BASIC PROGRAM AT A TIME                  *      4
                 *      2) MONITOR BATCH - MULTIPLE BASIC PROGRAMS CAN BE EXECUTED   *      4
                 *                         PER RUN.                                  *      4
                 *      3) ONLINE (WISH) - BASIC PROGRAM CAN BE ENTERED, EDITED AND  *      4
                 *                         EXECUTED ON LINE.                         *      4
                 *                                                                   *      4
                 *********************************************************************/     4
                                                                                            33
                 /*********************************************************************     34
                 *                                                                   *     34
                 *   BASIC/360 V2.2    DRAFT                                         *     34
                 *                                                                   *     34
                 *   V2.2 CHANGE LOG                                                 *     34
                 *  -- FIXES:                                                        *     34
                 *   - FIXED LINE OVERFLOW REPORTED BY MARCUS LOEW                   *     34
                 *   - COSMETIC FIXES TO LISTING                                     *     34
```

STMT LEVEL NEST

```
*  --ENHANCEMENTS:                                                 *     34
*   - ADDED INR - INT WITH ROUNDING                                *     34
*   - ADDED STRING COMPARE TO IF STATEMENT                         *     34
*   - ADDED STOP STATEMENT TO BE USED FOR ABNORMAL ENDING          *     34
*   - ADDED SUBSCRIPT CHECKING TO PREVENT PROTECTION EXCEPTIONS    *     34
*                                                                  *     34
********************************************************************     34
/*******************************************************************     34
*                                                                  *     34
*   BASIC/360 V2.1                                                 *     34
*                                                                  *     34
********************************************************************     34
*                                                                  *     34
*   THIS PROJECT WAS STARTED AS A CLASS PROJECT A WHILE BACK. IN   *     34
*   TYPICAL IT STYLE, IT WAS SHELVED UNTIL WE HAD TIME TO WORK ON  *     34
*   IT AGAIN.  IT IS ONLY 42 YEARS LATE.                           *     34
*                                                                  *     34
*   I FOUND IT IM MY ARCHIVES AND SCANNED IT.  WITH A LITTLE       *     34
*   WORK, BASIC/360 LIVES (OR HAS BEEN RESURECTED - DEPENDS ON HOW *     34
*   YOU WANT TO LOOK AT IT).                                       *     34
*                                                                  *     34
*   V1.0 WORKED BUT I DID SOME TESTING ON IT AND FOUND A FEW BUGS  *     34
*   IN THE CODE.  THEY WERE FIXED.                                 *     34
*                                                                  *     34
********************************************************************     34
*                                                                  *     34
*   V2.1 CHANGE LOG                                                *     34
*  -- FIXES:                                                       *     34
*   - CORRECTED TYPOS.                                             *     34
*  --ENHANCEMENTS:                                                 *     34
*   - CLEANED UP CODE FOR IMPLEMENTING BASIC LIBRARY FUNCTIONS     *     34
*   - ADDED RND FUNCTION TO THE BASIC LIBRARY FUNCTIONS            *     34
*   - CONSOLDATED THE STRING_STACK INTO SYMBOL_TABLE TO PREPARE    *     34
*     FOR SUPPORTING STRING VARIABLES.                             *     34
*   - REVISING CODE TO USE THE SELECT....ENDSELECT MACROS          *     34
*   - REVISING CREATION OF PC_OPCODE TABLE TO USE MACROS           *     34
*   - REVISED SYNTAX ERROR MESSAGES WITH MORE DETAIL               *     34
*   - ADDED SUPPORT TO PCODE INTERPRETER TO ABOUT ILLEGAL MIXED    *     34
*     MODE (I.E. MIXING NUMERIC AND STRINGS TOGETHER IN A LINE)    *     34
*   - STRING VARIABLES ADDED.                                      *     34
*   - STRING CONSTANTS IN LET STATEMENTS ADDED.                    *     34
*   - SUPPORT FOR STRINGS IN READ AND DATA STATEMENTS.             *     34
*                                                                  *     34
********************************************************************     34
*                                                                  *     34
```

STMT LEVEL NEST

```
*   V2.0 CHANGE LOG                                                 *     34
*  -- ID CHANGE.  THERE WAS NO SUCH PLACE AS SOUTH HAMMOND          *     34
*                 INSTITUTE OF TECHNOLOGY.  IT WAS REALLY PURDUE    *     34
*                 UNIVERSITY CALUMET.  THE ACRONYM WAS A JOKE       *     34
*                 ORIGINALLY BUT NOW IS NOT IN GOOD TASTE.          *     34
*  -- FIXES:                                                        *     34
*   - IF A PRINT STATEMENT FOLLOWS AN IF STATEMENT, THE COMPILER    *     34
*     ABORTS WITH A PROTECTION EXCEPTION.                           *     34
*   - PRINTING VALUES => 1.0E+6 RESULTS IN BAD OUTPUT               *     34
*   - DEFAULT PRINT COLUMN WIDTHS WERE CHANGED FROM 12 TO 14        *     34
*   - DIVISION BY ZERO CAUSES JOB TO ABORT                          *     34
*   - CODE ADDED TO ABORT THE BASIC PROGRAM NOT THE JOB             *     34
*   - FIXED CODE GENERATION FOR DIM ACCESS.                         *     34
*  --ENHANCEMENTS:                                                  *     34
*   - CHANGED CODE TO UTILIZE PL/I(F) FEATURES.                     *     34
*   - MISC CODE CLEANUP AND COMMENTS ADDED.                         *     34
*   - PC_FORMAT WAS ADDED TO THE VALID OPCODE TABLE TO IDENTIFY     *     34
*     WHAT WAS IN THE PC_OBJECT FIELD.  PRINT_PCODES WAS ALSO       *     34
*     MODIFIED TO USED THE FORMAT CODES INSTEAD OF THE PNEMONICS    *     34
*   - DEF FUNCTIONS HAVE BEEN IMPLEMENTED.                          *     34
*                                                                   *     34
*********************************************************************     34
*                                                                   *     34
*   WISH LIST (OR STUFF PUT OFF UNTIL LATER)                        *     34
*   - SPLIT SINGLE PROGRAM VS BATCH MONITOR.  SINGLE PGM COULD      *     34
*     ADD 'DATAFILE'.  PROCESS DATA STMTS THEN READ DATAFILE        *     34
*     SETS UP FOR IDE MODE.                                         *     34
*   - CHANGE FOR..NEXT TO A DO..WHILE CONSTRUCT                     *     34
*   - TSO ENVIRONMENT IMPLEMENTATION.                               *     34
*   - A 'COPY' OR SOURCE CODE LIB FACILITY.                         *     34
*   - PRINT USING STATEMENT.                                        *     34
*                                                                   *     34
*********************************************************************/    34
                                                                           119
/*********************************************************************  4065
                                                                        4065
      THE FIX2STR MACRO - CONVERT A FIXED ITEM TO A STRING              4065
                  FOR USE WITH PL/I (F).                                4065
                                                                        4065
*********************************************************************/ 4065
                                                                        4070
/*********************************************************************  4101
                                                                        4101
      THE SELECT, WHEN, BREAK, OTHERWISE AND ENDSELECT MACROS           4101
                  FOR USE WITH PL/I (F).                                4101
```

STMT LEVEL NEST

```
                                                                      4101
  THESE MACROS EMULATE THE PL/I SELECT STATEMENT WHICH WAS NOT        4101
  IMPLEMENTED IN THE PL/I (F) COMPILER.                               4101
                                                                      4101
  TO USE, %INCLUDE THIS MEMBER IN YOUR PROGRAM AND COMPILE WITH       4101
  THE MACRO OPTION. (NOMACRO IS THE DEFAULT).                         4101
                                                                      4101
  CODE THE SELECT:                                                    4101
                                                                      4101
      SELECT (A)                SELECT (TRUE)                       4101
      WHEN(1)                   WHEN(A=1)                           4101
        CALL SUB1;                CALL SUB1;                        4101
      WHEN(X)                   WHEN(A=X)                           4101
        CALL XRTN(X);             CALL XTRN(X);                     4101
        X=X+A;                    X=X+A;                            4101
      ENDSELECT                 OTHERWISE                           4101
                                  PUT SKIP LIST('ERROR');           4101
                                ENDSELECT                           4101
                                                                    4101
  NOTE - USING THE MACROS, THE ENDSELECT IS REQUIRED.               4101
                                                                    4101
  THE ALTERNATE SELECT WHEN BREAK IS ALSO SUPPORTED.  THIS ONE IS   4101
  POPULAR WITH C, C++, JAVA, ETC PROGRAMMERS.                       4101
                                                                    4101
  THERE IS A COMPILE TIME VARIABLE DEFINED SELECT_TYPE.  BY DEFAULT 4101
  IT IS SET TO 0. THIS INDICATES UPON COMPLETION OF A WHEN BLOCK,   4101
  THE STATEMENT AFTER THE ENDSELECT WILL BE THE NEXT EXECUTED.      4101
  IF SELECT_TYPE IS SET TO 1, UPON COMPLETION OF A WHEN BLOCK, THE  4101
  NEXT WHEN WILL BE EXECUTED.  A BREAK MACRO IN A WHEN BLOCK        4101
  WILL CAUSE THE STATEMENT AFTER THE ENDSELECT TO BE THE NEXT       4101
  STATEMENT EXECUTED.                                               4101
  TO SWITCH TYPE, CODE A %SELECT_TYPE=1; OR %SELECT_TYPE=0;         4101
*********************************************************************/ 4101
                                                                      4137
                                                                      4145
                                                                      4152
/*********************************************************************/ 4159
/*********************************************************************/ 4184
/*********************************************************************/ 4221
/*********************************************************************/ 4233
/*********************************************************************/ 4250
/*********************************************************************  4269
                                                                      4269
  THIS MACRO GENERATES A SERIES OF ELEMENTS FOR CREATING A TABLE.     4269
                                                                      4269
```

STMT LEVEL NEST

```
  THE THIS MACRO IS CUSTOMIZED TO CREATE THE PSEUDO CODE (PC)          4269
  CODE TABLE FOR BASIC/360.                                            4269
                                                                       4269
  TO USE, %INCLUDE THIS MEMBER IN YOUR PROGRAM AND COMPILE WITH        4269
  THE MACRO OPTION. (NOMACRO IS THE DEFAULT).                          4269
                                                                       4269
  CODE:                                                                4269
      GENPC(MMM,PPP,F,A)                                               4269
                                                                       4269
      WHERE                                                            4269
        MMM IS THE OPCODE                                              4269
        PPP IS THE OPCODE                                              4269
        F   IS THE CODE FOR ACCEPTABLE OPERANDS                        4269
        A   IS THE PERMISSIBLE TYPE OF OPERANDS AS BIT STRING.         4269
                                                                       4269
*************************************************************************/ 4269
                                                                       4288
                                                                       4290
/*************************************************************************/ 4324
/*************************************************************************  4325
                                                                       4325
  THIS MACRO GENERATES A SERIES OF ELEMENTS FOR CREATING A TABLE.      4325
                                                                       4325
  THE THIS MACRO IS CUSTOMIZED TO INITIALIZE THE SYMBOL STACK (SS)     4325
  CODE TABLE FOR BASIC/360.                                            4325
                                                                       4325
  TO USE, %INCLUDE THIS MEMBER IN YOUR PROGRAM AND COMPILE WITH        4325
  THE MACRO OPTION. (NOMACRO IS THE DEFAULT).                          4325
                                                                       4325
  CODE:                                                                4325
      GENSYM(S,T,F)                                                    4325
                                                                       4325
      WHERE                                                            4325
        S   IS THE SYMBOL TO DEFINE                                    4325
        T   IS THE SYMBOL TYPE                                         4325
        VN  IS THE SYMBOL INITIAL NUMERIC VALUE                        4325
        VS  IS THE SYMBOL INITIAL STRING VALUE                         4325
                                                                       4325
*************************************************************************/ 4325
                                                                       4344
                                                                       4346
/*************************************************************************/ 4380
                                                                        124
```

```
STMT LEVEL NEST

  1              BASIC:PROCEDURE OPTIONS(MAIN);                                  125
                                                                                 126
                 /*****************************************************************  127
                 *                                                               *  127
                 *   DEFINE MAX SIZES FOR GLOBAL TABLES USED IN THIS PROGRAM     *  127
                 *                                                               *  127
                 *****************************************************************/ 127
                                                                                 131
                                                                                 132
                            /* MAX NUMBER OF DATA NUMBERS OR STRING*/            138
                             /* MAX NUMBER OF LINE IN BASIC PGM     */           139
                               /* MAX NUMBER DATA ELEMENTS IN SYM TBL */         140
                             /* MAX NUMBER OF PCODES                */           141
                            /* MAX PCODES BEFORE ABORTED AS LOOPED */            142
                                                                                 143
                 /*****************************************************************  144
                 *                                                               *  144
                 *                        GLOBAL VARIABLES                       *  144
                 *                                                               *  144
                 *****************************************************************/ 144
                                                                                 148
                                                                                 149
  2     1            DECLARE PAGE_TITLE          CHAR(20) STATIC                 150
                                                 INITIAL('BASIC/360 V2.2.0');    151
  3     1            DECLARE PAGE_NUM            FIXED DECIMAL(5,0) INITIAL(0);  152
  4     1            DECLARE PGM_PAGE_NUM        FIXED DECIMAL(5,0) INITIAL(0);  153
  5     1            DECLARE EOF_SYSIN           BIT(1) ALIGNED INITIAL('0'B);   154
  6     1            DECLARE EOP_SYSIN           BIT(1) ALIGNED;                 155
  7     1            DECLARE QUOTE_1             CHAR(1) INITIAL('"'),           156
                            QUOTE_2             CHAR(2) INITIAL('""');          157
  8     1            DECLARE ERROR_COUNT         FIXED DECIMAL(5,0) INITIAL(0);  158
                                                                                 159
  9     1            DECLARE 1 STMT_IN,                                          160
                               2 STMT            CHAR(80),                       161
                              (2 STMT_LEFT,                                      162
                               2 STMT_RIGHT,                                     163
                               2 STMT_CH)        FIXED BINARY ALIGNED;           164
 10     1            DECLARE STMT_BUFF           CHAR(80);                       165
 11     1            DECLARE LAST_LINE_NUM       FIXED DECIMAL(5,0);             166
 12     1            DECLARE REF_LINE_NUM        FIXED DECIMAL(5,0);             167
 13     1            DECLARE WORD                CHAR(8);                        168
 14     1            DECLARE RUN_DATE            CHAR(10);                       169
 15     1            DECLARE BASIC_RENUM         BIT(1) ALIGNED INITIAL('0'B);   170
 16     1            DECLARE MONITOR_STMT        CHAR(80);                       171
                                                                                 172
```

```
STMT LEVEL NEST

                /*****************************************************************     173
                *                                                               *     173
                *                         GLOBAL DEBUGING                       *     173
                *                                                               *     173
                *****************************************************************/    173
                                                                                        177
                                                                                        178
 17     1           DECLARE STACK_PRINT_DEBUG   BIT(1) ALIGNED INITIAL('0'B);          179
 18     1           DECLARE EXECUTION_DEBUG     BIT(1) ALIGNED INITIAL('0'B);          180
 19     1           DECLARE TABLE_PRINT         BIT(1) ALIGNED INITIAL('0'B);          181
 20     1           DECLARE TABLE_DUMP          BIT(1) ALIGNED INITIAL('0'B);          182
 21     1           DECLARE ICODE_PRINT         BIT(1) ALIGNED INITIAL('0'B);          183
                                                                                        184
```

```
STMT LEVEL NEST

                /*****************************************************************    185
                *                                                               *    185
                *                        GLOBAL CONSTANTS                       *    185
                *                                                               *    185
                *  THESE CONSTANTS ARE USED IN TWO OR MORE OF THE MAJOR MODULES  *    185
                *  OF THE COMPILER/INTERPRETER.                                 *    185
                *                                                               *    185
                *****************************************************************/    185
                                                                                       192
                                                                                       193
 22     1          DECLARE TRUE               BIT(1) ALIGNED STATIC INITIAL('1'B);    194
 23     1          DECLARE FALSE              BIT(1) ALIGNED STATIC INITIAL('0'B);    195
 24     1          DECLARE ZERO               FIXED BINARY ALIGNED STATIC             196
                                                          INITIAL(0);                  197
                                                                                       198
 25     1          DECLARE VALID_VAR_CHARS    CHAR(37) STATIC                         199
                      INITIAL(' ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789');                200
                                                                                       201
 26     1          DECLARE 1 KEY_WORD_AREA    STATIC,                                 202
                            2 KW_DATA         CHAR(8) INITIAL('DATA'),                203
                            2 KW_DEF          CHAR(8) INITIAL('DEF'),                 204
                            2 KW_DIM          CHAR(8) INITIAL('DIM'),                 205
                            2 KW_END          CHAR(8) INITIAL('END'),                 206
                            2 KW_FOR          CHAR(8) INITIAL('FOR'),                 207
                            2 KW_GOSUB        CHAR(8) INITIAL('GOSUB'),               208
                            2 KW_GOTO         CHAR(8) INITIAL('GOTO'),                209
                            2 KW_IF           CHAR(8) INITIAL('IF'),                  210
                            2 KW_LET          CHAR(8) INITIAL('LET'),                 211
                            2 KW_NEXT         CHAR(8) INITIAL('NEXT'),                212
                            2 KW_PRINT        CHAR(8) INITIAL('PRINT'),               213
                            2 KW_READ         CHAR(8) INITIAL('READ'),                214
                            2 KW_REM          CHAR(8) INITIAL('REM'),                 215
                            2 KW_RETURN       CHAR(8) INITIAL('RETURN'),              216
                            2 KW_RESTORE      CHAR(8) INITIAL('RESTORE'),             217
                            2 KW_STOP         CHAR(8) INITIAL('STOP'),                218
                                                                                       219
                          1 KEY_WORDS(16)     DEFINED KEY_WORD_AREA                   220
                                              CHAR(8);                                221
                                                                                       222
 27     1          DECLARE 1 SS_CONSTANTS     STATIC ALIGNED,                         223
                            2  SS_UNKNWN      FIXED BINARY INITIAL(0),                224
                            2  SS_UNKNWM_DESC CHAR(8)      INITIAL('UNKNOWN '),     225
                            2  SS_CONST       FIXED BINARY INITIAL(1),                226
                            2  SS_CONST_DESC  CHAR(8)      INITIAL('CONST   '),     227
                            2  SS_FUNC        FIXED BINARY INITIAL(2),                228
```

```
STMT LEVEL NEST

                          2  SS_FUNC_DESC   CHAR(8)      INITIAL('FUNCTION'),        229
                          2  SS_VAR         FIXED BINARY INITIAL(3),                 230
                          2  SS_VAR_DESC    CHAR(8)      INITIAL('VAR     '),        231
                          2  SS_DIM_VAR     FIXED BINARY INITIAL(4),                 232
                          2  SS_DIM_DESC    CHAR(8)      INITIAL('DIM     '),        233
                          2  SS_DEF_VAR     FIXED BINARY INITIAL(5),                 234
                          2  SS_DEF_DESC    CHAR(8)      INITIAL('DEF     '),        235
                          2  SS_STRCON      FIXED BINARY INITIAL(6),                 236
                          2  SS_STRCON_DESC CHAR(8)      INITIAL('STRCON  '),        237
                          2  SS_STRVAR      FIXED BINARY INITIAL(7),                 238
                          2  SS_STRVAR_DESC CHAR(8)      INITIAL('STRVAR  '),        239
                          2  SS_STRDIM      FIXED BINARY INITIAL(8),                 240
                          2  SS_STRDIM_DESC CHAR(8)      INITIAL('STRDIM  ');        241
                                                                                       242
 28     1            DECLARE 1 SS_CON_TABLE      BASED (SS_CON_TABLE_PTR),           243
                          2  SS_TAB(0:8),                                            244
                            3 SS_CODE        FIXED BINARY,                          245
                            3 SS_DESC        CHAR(8);                               246
```

```
STMT LEVEL NEST

                /*****************************************************************    247
                *                                                               *    247
                *                       PSEUDO OPCODES DEFINITION               *    247
                *                                                               *    247
                *  THE PC_FORMAT CODES DESCRIBE THE TYPE OF ARGUMENT EACH PSEUDO *    247
                *  EXPECTS.  MOSTLY USED TO CORRECTLY PRINT THE PCODES.         *    247
                *      PC_FORMAT_   INDICATES                                   *    247
                *      ----------   ------------------------------------------  *    247
                *          0        VARIABLE LOCATED IN THE SYMBOL_TABLE       *    247
                *          1        OBJECT IS A LINE NUMBER DEFINITION         *    247
                *          2        OBJECT IS A LINE NUMBER IN THE LINE_STACK  *    247
                *          3        OBJECT IS A STRING                         *    247
                *          4        OBJECT IS NOT USED                         *    247
                *          5        OBJECT IS AN OFFSET IN THE PC_TABLE        *    247
                *                                                               *    247
                *   THE GENPC MACRO DEFINES A P-CODE.  GENPC IS GIVEN 4 PARMS:  *    247
                *     1)  THE MNEMONIC FOR THE P-CODE                           *    247
                *     2)  THE NUMERIC P-CODE                                    *    247
                *     3)  THE P-CODE FORMAT AS DEFINED IN PC-FORMAT             *    247
                *     4)  TYPE CHECKING ENFORCEMENT - TWO BINARY DIGITS THAT    *    247
                *         INDICATE IF NUMBER VS STRING TESTS ARE TO BE MADE.    *    247
                *         00=NO CHECKING                                        *    247
                *         01=OPERAND MUST BE NUMERIC                            *    247
                *         10=OPERAND MUST BE STRING                             *    247
                *         11=OPERAND MUST TYPE MUST MATCH ACCUM TYPE            *    247
                *                                                               *    247
                *****************************************************************/    247
                                                                                     273
                                                                                     274
                                                                                     275
 29     1           DECLARE 1 MISC_CODE_DEF     STATIC ALIGNED,                  276
                              2  PC_FORMAT_0    FIXED BINARY INITIAL(0),         277
                              2  PC_FORMAT_1    FIXED BINARY INITIAL(1),         278
                              2  PC_FORMAT_2    FIXED BINARY INITIAL(2),         279
                              2  PC_FORMAT_3    FIXED BINARY INITIAL(3),         280
                              2  PC_FORMAT_4    FIXED BINARY INITIAL(4),         281
                              2  PC_FORMAT_5    FIXED BINARY INITIAL(5),         282
                              2  PCT_LFEED      FIXED BINARY INITIAL(0),         283
                              2  PCT_TAB        FIXED BINARY INITIAL(1),         284
                              2  PCT_NOTAB      FIXED BINARY INITIAL(2),         285
                              2  EXP_RCVR       FIXED BINARY INITIAL(0),         286
                              2  EXP_CALC       FIXED BINARY INITIAL(1),         287
                              2  EXP_FN_CALC    FIXED BINARY INITIAL(2);         288
                                                                                   289
 30     1           DECLARE 1 PC_CONSTANTS      STATIC ALIGNED,                  290
```

STMT LEVEL NEST

```
/* GENPC(SLN,00,1,00) */                                  291 1
2  PC_OPCODE_SLN FIXED BINARY INITIAL(00),                291 1
2  PC_MNCODE_SLN CHAR(4)      INITIAL('SLN'),             291 1
2  PC_FORMAT_SLN FIXED BINARY INITIAL(1),                 291 1
2  PC_ALLOWS_SLN BIT(2)       INITIAL('00'B),             291 1
/* GENPC(LDA,01,0,11) */                                  292 1
2  PC_OPCODE_LDA FIXED BINARY INITIAL(01),                292 1
2  PC_MNCODE_LDA CHAR(4)      INITIAL('LDA'),             292 1
2  PC_FORMAT_LDA FIXED BINARY INITIAL(0),                 292 1
2  PC_ALLOWS_LDA BIT(2)       INITIAL('11'B),             292 1
/* GENPC(STA,02,0,11) */                                  293 1
2  PC_OPCODE_STA FIXED BINARY INITIAL(02),                293 1
2  PC_MNCODE_STA CHAR(4)      INITIAL('STA'),             293 1
2  PC_FORMAT_STA FIXED BINARY INITIAL(0),                 293 1
2  PC_ALLOWS_STA BIT(2)       INITIAL('11'B),             293 1
/* GENPC(EXP,03,0,01) */                                  294 1
2  PC_OPCODE_EXP FIXED BINARY INITIAL(03),                294 1
2  PC_MNCODE_EXP CHAR(4)      INITIAL('EXP'),             294 1
2  PC_FORMAT_EXP FIXED BINARY INITIAL(0),                 294 1
2  PC_ALLOWS_EXP BIT(2)       INITIAL('01'B),             294 1
/* GENPC(ADD,04,0,01) */                                  295 1
2  PC_OPCODE_ADD FIXED BINARY INITIAL(04),                295 1
2  PC_MNCODE_ADD CHAR(4)      INITIAL('ADD'),             295 1
2  PC_FORMAT_ADD FIXED BINARY INITIAL(0),                 295 1
2  PC_ALLOWS_ADD BIT(2)       INITIAL('01'B),             295 1
/* GENPC(SUB,05,0,01) */                                  296 1
2  PC_OPCODE_SUB FIXED BINARY INITIAL(05),                296 1
2  PC_MNCODE_SUB CHAR(4)      INITIAL('SUB'),             296 1
2  PC_FORMAT_SUB FIXED BINARY INITIAL(0),                 296 1
2  PC_ALLOWS_SUB BIT(2)       INITIAL('01'B),             296 1
/* GENPC(MUL,06,0,01) */                                  297 1
2  PC_OPCODE_MUL FIXED BINARY INITIAL(06),                297 1
2  PC_MNCODE_MUL CHAR(4)      INITIAL('MUL'),             297 1
2  PC_FORMAT_MUL FIXED BINARY INITIAL(0),                 297 1
2  PC_ALLOWS_MUL BIT(2)       INITIAL('01'B),             297 1
/* GENPC(DIV,07,0,01) */                                  298 1
2  PC_OPCODE_DIV FIXED BINARY INITIAL(07),                298 1
2  PC_MNCODE_DIV CHAR(4)      INITIAL('DIV'),             298 1
2  PC_FORMAT_DIV FIXED BINARY INITIAL(0),                 298 1
2  PC_ALLOWS_DIV BIT(2)       INITIAL('01'B),             298 1
/* GENPC(RDV,08,0,11) */                                  299 1
2  PC_OPCODE_RDV FIXED BINARY INITIAL(08),                299 1
2  PC_MNCODE_RDV CHAR(4)      INITIAL('RDV'),             299 1
2  PC_FORMAT_RDV FIXED BINARY INITIAL(0),                 299 1
2  PC_ALLOWS_RDV BIT(2)       INITIAL('11'B),             299 1
```

STMT LEVEL NEST

```
/* GENPC(PRV,09,0,11) */                                300 1
2  PC_OPCODE_PRV FIXED BINARY INITIAL(09),              300 1
2  PC_MNCODE_PRV CHAR(4)      INITIAL('PRV'),           300 1
2  PC_FORMAT_PRV FIXED BINARY INITIAL(0),               300 1
2  PC_ALLOWS_PRV BIT(2)       INITIAL('11'B),           300 1
/* GENPC(PCT,10,5,00) */                                301 1
2  PC_OPCODE_PCT FIXED BINARY INITIAL(10),              301 1
2  PC_MNCODE_PCT CHAR(4)      INITIAL('PCT'),           301 1
2  PC_FORMAT_PCT FIXED BINARY INITIAL(5),               301 1
2  PC_ALLOWS_PCT BIT(2)       INITIAL('00'B),           301 1
/* GENPC(FNC,11,0,00) */                                302 1
2  PC_OPCODE_FNC FIXED BINARY INITIAL(11),              302 1
2  PC_MNCODE_FNC CHAR(4)      INITIAL('FNC'),           302 1
2  PC_FORMAT_FNC FIXED BINARY INITIAL(0),               302 1
2  PC_ALLOWS_FNC BIT(2)       INITIAL('00'B),           302 1
/* GENPC(END,12,0,00) */                                303 1
2  PC_OPCODE_END FIXED BINARY INITIAL(12),              303 1
2  PC_MNCODE_END CHAR(4)      INITIAL('END'),           303 1
2  PC_FORMAT_END FIXED BINARY INITIAL(0),               303 1
2  PC_ALLOWS_END BIT(2)       INITIAL('00'B),           303 1
/* GENPC(B  ,13,2,00) */                                304 1
2  PC_OPCODE_B   FIXED BINARY INITIAL(13),              304 1
2  PC_MNCODE_B   CHAR(4)      INITIAL('B  '),           304 1
2  PC_FORMAT_B   FIXED BINARY INITIAL(2),               304 1
2  PC_ALLOWS_B   BIT(2)       INITIAL('00'B),           304 1
/* GENPC(BAL,14,2,00) */                                305 1
2  PC_OPCODE_BAL FIXED BINARY INITIAL(14),              305 1
2  PC_MNCODE_BAL CHAR(4)      INITIAL('BAL'),           305 1
2  PC_FORMAT_BAL FIXED BINARY INITIAL(2),               305 1
2  PC_ALLOWS_BAL BIT(2)       INITIAL('00'B),           305 1
/* GENPC(RET,15,0,00) */                                306 1
2  PC_OPCODE_RET FIXED BINARY INITIAL(15),              306 1
2  PC_MNCODE_RET CHAR(4)      INITIAL('RET'),           306 1
2  PC_FORMAT_RET FIXED BINARY INITIAL(0),               306 1
2  PC_ALLOWS_RET BIT(2)       INITIAL('00'B),           306 1
/* GENPC(PRS,16,3,00) */                                307 1
2  PC_OPCODE_PRS FIXED BINARY INITIAL(16),              307 1
2  PC_MNCODE_PRS CHAR(4)      INITIAL('PRS'),           307 1
2  PC_FORMAT_PRS FIXED BINARY INITIAL(3),               307 1
2  PC_ALLOWS_PRS BIT(2)       INITIAL('00'B),           307 1
/* GENPC(LCA,17,0,11) */                                308 1
2  PC_OPCODE_LCA FIXED BINARY INITIAL(17),              308 1
2  PC_MNCODE_LCA CHAR(4)      INITIAL('LCA'),           308 1
2  PC_FORMAT_LCA FIXED BINARY INITIAL(0),               308 1
2  PC_ALLOWS_LCA BIT(2)       INITIAL('11'B),           308 1
```

STMT LEVEL NEST

```
/* GENPC(LCB,18,0,11) */                                309 1
2  PC_OPCODE_LCB FIXED BINARY INITIAL(18),              309 1
2  PC_MNCODE_LCB CHAR(4)      INITIAL('LCB'),           309 1
2  PC_FORMAT_LCB FIXED BINARY INITIAL(0),               309 1
2  PC_ALLOWS_LCB BIT(2)       INITIAL('11'B),           309 1
/* GENPC(BEQ,19,2,00) */                                310 1
2  PC_OPCODE_BEQ FIXED BINARY INITIAL(19),              310 1
2  PC_MNCODE_BEQ CHAR(4)      INITIAL('BEQ'),           310 1
2  PC_FORMAT_BEQ FIXED BINARY INITIAL(2),               310 1
2  PC_ALLOWS_BEQ BIT(2)       INITIAL('00'B),           310 1
/* GENPC(BNE,20,2,00) */                                311 1
2  PC_OPCODE_BNE FIXED BINARY INITIAL(20),              311 1
2  PC_MNCODE_BNE CHAR(4)      INITIAL('BNE'),           311 1
2  PC_FORMAT_BNE FIXED BINARY INITIAL(2),               311 1
2  PC_ALLOWS_BNE BIT(2)       INITIAL('00'B),           311 1
/* GENPC(BGT,21,2,00) */                                312 1
2  PC_OPCODE_BGT FIXED BINARY INITIAL(21),              312 1
2  PC_MNCODE_BGT CHAR(4)      INITIAL('BGT'),           312 1
2  PC_FORMAT_BGT FIXED BINARY INITIAL(2),               312 1
2  PC_ALLOWS_BGT BIT(2)       INITIAL('00'B),           312 1
/* GENPC(BLT,22,2,00) */                                313 1
2  PC_OPCODE_BLT FIXED BINARY INITIAL(22),              313 1
2  PC_MNCODE_BLT CHAR(4)      INITIAL('BLT'),           313 1
2  PC_FORMAT_BLT FIXED BINARY INITIAL(2),               313 1
2  PC_ALLOWS_BLT BIT(2)       INITIAL('00'B),           313 1
/* GENPC(BGE,23,2,00) */                                314 1
2  PC_OPCODE_BGE FIXED BINARY INITIAL(23),              314 1
2  PC_MNCODE_BGE CHAR(4)      INITIAL('BGE'),           314 1
2  PC_FORMAT_BGE FIXED BINARY INITIAL(2),               314 1
2  PC_ALLOWS_BGE BIT(2)       INITIAL('00'B),           314 1
/* GENPC(BLE,24,2,00) */                                315 1
2  PC_OPCODE_BLE FIXED BINARY INITIAL(24),              315 1
2  PC_MNCODE_BLE CHAR(4)      INITIAL('BLE'),           315 1
2  PC_FORMAT_BLE FIXED BINARY INITIAL(2),               315 1
2  PC_ALLOWS_BLE BIT(2)       INITIAL('00'B),           315 1
/* GENPC(FSU,25,0,00) */                                316 1
2  PC_OPCODE_FSU FIXED BINARY INITIAL(25),              316 1
2  PC_MNCODE_FSU CHAR(4)      INITIAL('FSU'),           316 1
2  PC_FORMAT_FSU FIXED BINARY INITIAL(0),               316 1
2  PC_ALLOWS_FSU BIT(2)       INITIAL('00'B),           316 1
/* GENPC(FIX,26,0,00) */                                317 1
2  PC_OPCODE_FIX FIXED BINARY INITIAL(26),              317 1
2  PC_MNCODE_FIX CHAR(4)      INITIAL('FIX'),           317 1
2  PC_FORMAT_FIX FIXED BINARY INITIAL(0),               317 1
2  PC_ALLOWS_FIX BIT(2)       INITIAL('00'B),           317 1
```

STMT LEVEL NEST

```
/* GENPC(FUL,27,0,00) */                                318 1
2  PC_OPCODE_FUL FIXED BINARY INITIAL(27),               318 1
2  PC_MNCODE_FUL CHAR(4)      INITIAL('FUL'),            318 1
2  PC_FORMAT_FUL FIXED BINARY INITIAL(0),                318 1
2  PC_ALLOWS_FUL BIT(2)       INITIAL('00'B),            318 1
/* GENPC(FST,28,0,00) */                                319 1
2  PC_OPCODE_FST FIXED BINARY INITIAL(28),               319 1
2  PC_MNCODE_FST CHAR(4)      INITIAL('FST'),            319 1
2  PC_FORMAT_FST FIXED BINARY INITIAL(0),                319 1
2  PC_ALLOWS_FST BIT(2)       INITIAL('00'B),            319 1
/* GENPC(FNX,29,0,00) */                                320 1
2  PC_OPCODE_FNX FIXED BINARY INITIAL(29),               320 1
2  PC_MNCODE_FNX CHAR(4)      INITIAL('FNX'),            320 1
2  PC_FORMAT_FNX FIXED BINARY INITIAL(0),                320 1
2  PC_ALLOWS_FNX BIT(2)       INITIAL('00'B),            320 1
/* GENPC(PTB,30,0,00) */                                321 1
2  PC_OPCODE_PTB FIXED BINARY INITIAL(30),               321 1
2  PC_MNCODE_PTB CHAR(4)      INITIAL('PTB'),            321 1
2  PC_FORMAT_PTB FIXED BINARY INITIAL(0),                321 1
2  PC_ALLOWS_PTB BIT(2)       INITIAL('00'B),            321 1
/* GENPC(RST,31,4,00) */                                322 1
2  PC_OPCODE_RST FIXED BINARY INITIAL(31),               322 1
2  PC_MNCODE_RST CHAR(4)      INITIAL('RST'),            322 1
2  PC_FORMAT_RST FIXED BINARY INITIAL(4),                322 1
2  PC_ALLOWS_RST BIT(2)       INITIAL('00'B),            322 1
/* GENPC(DSL,32,4,00) */                                323 1
2  PC_OPCODE_DSL FIXED BINARY INITIAL(32),               323 1
2  PC_MNCODE_DSL CHAR(4)      INITIAL('DSL'),            323 1
2  PC_FORMAT_DSL FIXED BINARY INITIAL(4),                323 1
2  PC_ALLOWS_DSL BIT(2)       INITIAL('00'B),            323 1
/* GENPC(LDR,33,0,00) */                                324 1
2  PC_OPCODE_LDR FIXED BINARY INITIAL(33),               324 1
2  PC_MNCODE_LDR CHAR(4)      INITIAL('LDR'),            324 1
2  PC_FORMAT_LDR FIXED BINARY INITIAL(0),                324 1
2  PC_ALLOWS_LDR BIT(2)       INITIAL('00'B),            324 1
/* GENPC(STR,34,0,00) */                                325 1
2  PC_OPCODE_STR FIXED BINARY INITIAL(34),               325 1
2  PC_MNCODE_STR CHAR(4)      INITIAL('STR'),            325 1
2  PC_FORMAT_STR FIXED BINARY INITIAL(0),                325 1
2  PC_ALLOWS_STR BIT(2)       INITIAL('00'B),            325 1
/* GENPC(JMP,35,5,00) */                                326 1
2  PC_OPCODE_JMP FIXED BINARY INITIAL(35),               326 1
2  PC_MNCODE_JMP CHAR(4)      INITIAL('JMP'),            326 1
2  PC_FORMAT_JMP FIXED BINARY INITIAL(5),                326 1
2  PC_ALLOWS_JMP BIT(2)       INITIAL('00'B),            326 1
```

```
STMT LEVEL NEST

                        /* GENPC(CFN,36,0,00) */                              327 1
                        2  PC_OPCODE_CFN FIXED BINARY INITIAL(36),            327 1
                        2  PC_MNCODE_CFN CHAR(4)      INITIAL('CFN'),         327 1
                        2  PC_FORMAT_CFN FIXED BINARY INITIAL(0),             327 1
                        2  PC_ALLOWS_CFN BIT(2)       INITIAL('00'B),         327 1
                        /* GENPC(RFN,37,0,00) */                              328 1
                        2  PC_OPCODE_RFN FIXED BINARY INITIAL(37),            328 1
                        2  PC_MNCODE_RFN CHAR(4)      INITIAL('RFN'),         328 1
                        2  PC_FORMAT_RFN FIXED BINARY INITIAL(0),             328 1
                        2  PC_ALLOWS_RFN BIT(2)       INITIAL('00'B),         328 1
                        /* GENPC(STP,38,0,00) */                              329 1
                        2  PC_OPCODE_STP FIXED BINARY INITIAL(38),            329 1
                        2  PC_MNCODE_STP CHAR(4)      INITIAL('STP'),         329 1
                        2  PC_FORMAT_STP FIXED BINARY INITIAL(0),             329 1
                        2  PC_ALLOWS_STP BIT(2)       INITIAL('00'B),         329 1
                      1 PC_CON_TABLE      BASED (PC_CON_TABLE_PTR),           331
                        2  PC_OPTAB(0:       38 ),                           332 1
                          3 PC_OP_CODE     FIXED BINARY,                      333
                          3 PC_MNEMONIC    CHAR(4),                           334
                          3 PC_FORMAT      FIXED BINARY,                      335
                          3 PC_ALLOW       BIT(2) ALIGNED;                    336
```

STMT LEVEL NEST

```
/*****************************************************************    337
*                                                               *    337
*                     GLOBAL OBJECT STRUCTURES                  *    337
*                                                               *    337
*  THESE ITEMS ARE USED TO EXECUTE THE BASIC PROGRAM.  THE COMPILE  *    337
*  PHASE STORES THE DATA IN THESE OBJECTS AND THE EXECUTION PHASE   *    337
*  EXECUTES THEM.                                               *    337
*                                                               *    337
*  DATA_STACK IS USED TO STORE NUMBERS FROM DATA STATEMENTS.    *    337
*                COMPILE STACKS THEM UP AND EXECUTE UNSTACKS THEM   *    337
*                NUMBERS EACH TIME A READ IS EXECUTED.          *    337
*                                                               *    337
*  LINE_STACK IS USED TO STORE THE BASIC LINE NUMBERS AND THE   *    337
*                OFFSET TO WHERE IN P_CODE THE STATEMENT STARTS.    *    337
*                THESE ARE USED TO FIND WHERE GOTO AND GOSUBS   *    337
*                TRANSFER CONTROL TO IN P_CODE_STACK.           *    337
*                                                               *    337
*  P_CODE_STACK IS USED TO STORE THE EXECUTABLE P_CODES GENERATED   *    337
*                DURING THE COMPILE PROCESS ARE THEN EXECUTED.  *    337
*                                                               *    337
*  SYMBOL_TABLE IS USED TO STORE THE ALL OF THE NUMERIC DATA    *    337
*                VARIABLE, CONSTANTS, DIM VARIABLES AND FUNCTIONS.  *    337
*                THIS TABLE IS POPULATED DURING COMPILE TIME WITH   *    337
*                VARIABLES NAMES, CONSTANTS AND DIMS NAMES.     *    337
*                DURING EXECUTION, THE VALUES FOR ALL VARIABLES     *    337
*                STORED AND RETRIEVED FROM THIS TABLE BY THE    *    337
*                EXECUTION PHASE.  NUMERIC CONSTANTS ARE RETRIEVED  *    337
*                FROM THIS TABLE DURING EXECUTION.              *    337
*                FUNCTIONS ARE INCLUDED IN THIS TABLE AS WELL.  *    337
*                STRING_TABLE AND SYMBOL TABLE WERE MERGED. IT  *    337
*                IS USED TO STORE THE ALL OF THE STRING DATA.   *    337
*                THIS TABLE IS POPULATED DURING COMPILE TIME WITH   *    337
*                STRING CONSTANTS AND SPACE RESERVED FOR STRING     *    337
*                VARIABLES.                                     *    337
*                DURING EXECUTION, THE VALUES FOR THE CONSTANTS     *    337
*                AND VARIABLES ARE RETRIEVED FROM THIS TABLE.   *    337
*                                                               *    337
*  SOURCE_TABLE  IS USED TO STORE THE SOURCE CODE TO BE COMPILED.   *    337
*                EACH OF THE ENVIRONMENTS LOADS THE BASIC PROGRAM   *    337
*                AND THEM PASSES IT TO THE COMPILER.            *    337
*                                                               *    337
*  DEF_FUNCTIONS IS USED TO STORE THE NAMES OF THE USER DEFINED     *    337
*                FUNCTIONS.                                     *    337
*                                                               *    337
*****************************************************************/    337
```

```
STMT LEVEL NEST


 31     1            DECLARE 1 DATA_STACK        ALIGNED,
                               2  (DS_CUR,
                                   DS_MAX)       FIXED BINARY,
                               2   DS_TABLE(      500 ),                        1
                                  3 DS_STR        FIXED BINARY,
                                  3 DS_ITEM       FLOAT BINARY;
 32     1            DECLARE 1 LINE_STACK        ALIGNED,
                               2  (LS_CUR,
                                   LS_MAX)       FIXED BINARY,
                               2   LS_NUM(      500 ),                          1
                                  3  LS_LINE     FIXED DECIMAL(5,0),
                                  3  LS_OFFSET   FIXED BINARY;
 33     1            DECLARE 1 SOURCE_CODE       ALIGNED,
                               2  (SC_CUR,
                                   SC_MAX)       FIXED BINARY,
                               2  SOURCE_AREA(      500 ),                      1
                                  3  SOURCE_LINE CHAR(80);
 34     1            DECLARE 1 P_CODE_STACK      ALIGNED,
                               2  (PC_CUR,
                                   PC_MAX)       FIXED BINARY,
                               2   PC_NUM(      500 ),                          1
                                  3  (PC_OPCODE,
                                      PC_OBJECT) FIXED BINARY;
 35     1            DECLARE 1 SYMBOL_STACK      ALIGNED,
                               2  (SS_CUR,
                                   SS_MAX_FNC,
                                   SS_MAX)       FIXED BINARY,
                               2  SYMBOL_AREA(      100 ),                      1
                                  3  SYMBOL      CHAR(10),
                                  3  SYM_TYPE    FIXED BINARY,
                                  3  SYM_VALUE   FLOAT DECIMAL,
                                  3  SYM_DIM_MAX FIXED BINARY,
                                  3  STRING_VAL  CHAR(80) VARYING;
 36     1            DECLARE 1 DEF_FUNCTIONS     ALIGNED,
                               2  (DF_CUR,
                                   DF_MAX)       FIXED BINARY,
                               2  DEF_FUNC_AREA(10),
                                  3  DF_NAME     CHAR(10),
                                  3 (DF_OFFSET,
                                  3  DF_RETURN)  FIXED BINARY;
```

```
STMT LEVEL NEST

                /*****************************************************************/    424
                /*****************************************************************/    425
                /*****************************************************************/    426
                /*****************************************************************/    427
                /*****************************************************************/    428
                                                                                         429
 37     1           RUN_DATE=DATE;   /* DATE IS IN YYMMDD FORMAT */                     430
 38     1           RUN_DATE=SUBSTR(RUN_DATE,3,2)|| '/' || SUBSTR(RUN_DATE,5,2) ||     431
                            '/20' || SUBSTR(RUN_DATE,1,2);                               432
                                                                                         433
 39     1           ON ENDFILE(SYSIN)                                                  434
 40     1              EOF_SYSIN,EOP_SYSIN=TRUE;                                       435
                                                                                         436
                /*********************************************************************  437
                *                                                                   *  437
                *  THESE TWO POINTERS MUST BE SET.  THE BASE STRUCTURES ARE MIXED   *  437
                *  WITH BINARY AND CHARACTER DATA AND PL/I DOES NOT ALLOW DEFINED   *  437
                *  STRUCTURES LIKE THIS.  SO THEY WERE MADE INTO BASED TABLES.      *  437
                *                                                                   *  437
                *********************************************************************/ 437
                                                                                         443
 41     1           PC_CON_TABLE_PTR = ADDR(PC_CONSTANTS);                             444
 42     1           SS_CON_TABLE_PTR = ADDR(SS_CONSTANTS);                             445
                                                                                         446
                /*********************************************************************  447
                *                                                                   *  447
                *  THIS IS THE MAIN DRIVING LOOP FOR BASIC.  IF THERE IS A ++       *  447
                *  LINE AS FIRST LINE IN SYSIN, IT IS ASSUMED BASIC IS RUNNING IN   *  447
                *  MONITOR MODE.  IF NO ++ LINE IS FOUND, IT IS 1 UP MODE.          *  447
                *                                                                   *  447
                *********************************************************************/ 447
                                                                                         453
 43     1           GET EDIT(STMT_BUFF) (A(80)); /*  THIS PRIMES THE INPUT PROCESS */  454
                                                                                         455
 44     1           DO WHILE(EOF_SYSIN=FALSE);                                         456
 45     1    1         CALL INITIALIZE;                                                457
 46     1    1         CALL MONITOR;                                                   458
 47     1    1         IF SC_MAX>0 THEN                                                459
 48     1    1         DO;                                                             460
 49     1    2            CALL COMPILE;                                                461
 50     1    2            IF BASIC_RENUM & (ERROR_COUNT=0) THEN                        462
 51     1    2            DO;                                                          463
 52     1    3               CALL RENUM;                                               464
 53     1    3               CALL INITIALIZE;                                          465
 54     1    3               CALL COMPILE;                                             466
```

```
STMT LEVEL NEST

 55     1    3              END;                                          467
 56     1    2              IF ERROR_COUNT=0 THEN                         468
 57     1    2              DO;                                           469
 58     1    3                 CALL EXECUTE;                              470
 59     1    3                 CALL TERMINATE;                            471
 60     1    3              END;                                          472
 61     1    2           END;                                             473
 62     1    1        END;                                                474
```

```
STMT LEVEL NEST

                /*********************************************************************/    475
                /*********************************************************************/    476
                /*********************************************************************/    477
                /*********************************************************************/    478
                /*********************************************************************/    479
                                                                                             480
 63     1       MONITOR:PROC;                                                                481
                                                                                             482
                /*********************************************************************     483
                *                                                                   *    483
                *   THIS PROC READS THE INPUT FILE AND LOADS THE BASIC PROGRAMS     *    483
                *   INTO THE SOURCE_CODE STRUCTURE FOR MONITOR MODE.  MONITOR       *    483
                *   CONTROL STATEMENTS START WITH ++.  OPTION CARDS START WITH      *    483
                *   AN * IS COL 1.  THEY ARE PROCESSED HERE AND PASSED BACK TO      *    483
                *   CALLER SO COMPILE CAN PRINT THEM BUT THEN TREATED AS COMMENTS   *    483
                *                                                                   *    483
                * NESTING:MONITOR                                                   *    483
                *********************************************************************/    483
                                                                                             492
                                                                                             493
 64     2           ON ENDFILE(SYSIN)                                                        494
 65     2              EOF_SYSIN,EOP_SYSIN=TRUE;                                             495
 66     2           MONITOR_STMT=(80)' ';                                                    496
                                                                                             497
 67     2           IF SUBSTR(STMT_BUFF,1,2)='++' THEN                                       498
 68     2           DO;                                                                      499
 69     2    1         MONITOR_STMT=STMT_BUFF;                                             500
 70     2    1         EOP_SYSIN=FALSE;                                                    501
 71     2    1         TABLE_PRINT=FALSE;                                                  502
 72     2    1         TABLE_DUMP=FALSE;                                                   503
 73     2    1         STACK_PRINT_DEBUG=FALSE;                                            504
 74     2    1         ICODE_PRINT=FALSE;                                                  505
 75     2    1         EXECUTION_DEBUG=FALSE;                                              506
 76     2    1         BASIC_RENUM= (SUBSTR(STMT_BUFF,1,8)='++RENUM ');                    507
 77     2    1         PGM_PAGE_NUM=0;                                                     508
 78     2    1         SC_CUR,SC_MAX=0;                                                    509
 79     2    1         GET EDIT(STMT_BUFF) (A(80));                                        510
 80     2    1         DO WHILE((EOF_SYSIN=FALSE)&(EOP_SYSIN=FALSE));                      511
 81     2    2            SC_MAX=SC_MAX+1;                                                 512
 82     2    2            IF SC_MAX>HBOUND(SOURCE_LINE,1) THEN                             513
 83     2    2            DO;                                                              514
 84     2    3               PUT SKIP LIST                                                 515
                                ('**** FATAL ERROR - PROGRAM TO BIG ***');                 516
 85     2    3               STOP;                                                         517
 86     2    3            END;                                                             518
```

```
STMT LEVEL NEST

 87     2    2              SOURCE_LINE(SC_MAX)=STMT_BUFF;                                519
 88     2    2              GET EDIT(STMT_BUFF) (A(80));                                  520
 89     2    2              IF SUBSTR(STMT_BUFF,1,2)='++' THEN                            521
 90     2    2              DO;                                                           522
 91     2    3                EOP_SYSIN=TRUE;                                             523
 92     2    3              END;                                                          524
 93     2    2           END;                                                             525
 94     2    1        END;                                                                526
 95     2             ELSE  /*  NO MONITOR CNTL - TREAT AS STRAIGHT BATCH */             527
 95     2             DO WHILE(EOF_SYSIN=FALSE);                                        528
 96     2    1           SC_MAX=SC_MAX+1;                                               529
 97     2    1           IF SC_MAX>HBOUND(SOURCE_LINE,1) THEN                           530
 98     2    1           DO;                                                            531
 99     2    2               PUT SKIP LIST('**** FATAL ERROR - PROGRAM TO BIG ***');    532
100     2    2               STOP;                                                      533
101     2    2           END;                                                           534
102     2    1           SOURCE_LINE(SC_MAX)=STMT_BUFF;                                 535
103     2    1           GET EDIT(STMT_BUFF) (A(80));                                   536
104     2    1        END;                                                              537
                                                                                        538
105     2         END MONITOR;                                                         539
```

```
STMT LEVEL NEST

                /*****************************************************************/    540
                /*****************************************************************/    541
                /*****************************************************************/    542
                /*****************************************************************/    543
                /*****************************************************************/    544
                                                                                         545
106     1       INITIALIZE:PROC;                                                         546
                                                                                         547
                /*****************************************************************     548
                *                                                               *    548
                *   THIS PROC INITIALIZES ALL OF THE GLOBAL DATA ELEMENTS AND    *    548
                *   STRUCTURES FOR THE COMPILATION AND EXECUTION OF THE BASIC    *    548
                *   PROGRAM.                                                    *    548
                *                                                               *    548
                * NESTING:INITIALIZE                                            *    548
                *****************************************************************/    548
                                                                                         555
                                                                                         556
107     2           STMT_LEFT=1;                                                         557
108     2           STMT_RIGHT=72;                                                       558
109     2           LAST_LINE_NUM=-1;                                                    559
                                                                                         560
110     2           DS_CUR,DS_MAX=0;                                                     561
111     2           LS_CUR,LS_MAX=0;                                                     562
112     2           PC_CUR,PC_MAX=0;                                                     563
113     2           DF_CUR,DF_MAX=0;                                                     564
114     2           ERROR_COUNT=0;                                                       565
                                                                                         566
                     /* GENSYM(NULL,SS_VAR,0.0,*) */                                 567 1
115     2            SYMBOL    (       1) = 'NULL';                                    567 1
116     2            SYM_TYPE  (       1) = SS_VAR;                                    567 1
117     2            SYM_VALUE (       1) = 0.0;                                       567 1
118     2            STRING_VAL(       1) = '*';                                       567 1
                                                                                         567 1
                  /*********************************************************          569
                  *                                                       *          569
                  *  THESE ARE THE BUILTIN FUNCTIONS.  IF YOU ADD MORE    *          569
                  *  BE SURE TO ADD THEM TO THE FUNCTION INTERPRETER IN   *          569
                  *  EXECUTE   PSUDEO OP CODE FNC                         *          569
                  *                                                       *          569
                  *********************************************************/         569
                                                                                         575
                                                                                         576
                /*****************************************************************     577
                *                                                               *    577
```

```
STMT LEVEL NEST

                * IMPORTANT NOTE - IF ANY CHANGES ARE MADE TO LIBRARY FUNCTIONS,    *    577
                * THE PCODE FNC SHOULD MATCH THE CHANGES IN THE INITIALIZE PROC     *    577
                *                                                                  *    577
                ********************************************************************/   577
                                                                                         582
                                                                                         583
                     /* GENSYM(SQR,SS_FUNC,0.0,*) */                                     584 1
119     2            SYMBOL    (       2) = 'SQR';                                      584 1
120     2            SYM_TYPE  (       2) = SS_FUNC;                                    584 1
121     2            SYM_VALUE (       2) = 0.0;                                        584 1
122     2            STRING_VAL(       2) = '*';                                        584 1
                                                                                         584 1
                     /* GENSYM(ABS,SS_FUNC,0.0,*) */                                     585 1
123     2            SYMBOL    (       3) = 'ABS';                                      585 1
124     2            SYM_TYPE  (       3) = SS_FUNC;                                    585 1
125     2            SYM_VALUE (       3) = 0.0;                                        585 1
126     2            STRING_VAL(       3) = '*';                                        585 1
                                                                                         585 1
                     /* GENSYM(TAB,SS_FUNC,0.0,*) */                                     586 1
127     2            SYMBOL    (       4) = 'TAB';                                      586 1
128     2            SYM_TYPE  (       4) = SS_FUNC;                                    586 1
129     2            SYM_VALUE (       4) = 0.0;                                        586 1
130     2            STRING_VAL(       4) = '*';                                        586 1
                                                                                         586 1
                     /* GENSYM(INT,SS_FUNC,0.0,*) */                                     587 1
131     2            SYMBOL    (       5) = 'INT';                                      587 1
132     2            SYM_TYPE  (       5) = SS_FUNC;                                    587 1
133     2            SYM_VALUE (       5) = 0.0;                                        587 1
134     2            STRING_VAL(       5) = '*';                                        587 1
                                                                                         587 1
                     /* GENSYM(COS,SS_FUNC,0.0,*) */                                     588 1
135     2            SYMBOL    (       6) = 'COS';                                      588 1
136     2            SYM_TYPE  (       6) = SS_FUNC;                                    588 1
137     2            SYM_VALUE (       6) = 0.0;                                        588 1
138     2            STRING_VAL(       6) = '*';                                        588 1
                                                                                         588 1
                     /* GENSYM(SIN,SS_FUNC,0.0,*) */                                     589 1
139     2            SYMBOL    (       7) = 'SIN';                                      589 1
140     2            SYM_TYPE  (       7) = SS_FUNC;                                    589 1
141     2            SYM_VALUE (       7) = 0.0;                                        589 1
142     2            STRING_VAL(       7) = '*';                                        589 1
                                                                                         589 1
                     /* GENSYM(TAN,SS_FUNC,0.0,*) */                                     590 1
143     2            SYMBOL    (       8) = 'TAN';                                      590 1
144     2            SYM_TYPE  (       8) = SS_FUNC;                                    590 1
```

```
STMT LEVEL NEST

145     2              SYM_VALUE (       8) = 0.0;                          590 1
146     2              STRING_VAL(       8) = '*';                          590 1
                                                                            590 1
                       /* GENSYM(RND,SS_FUNC,0.0,*) */                      591 1
147     2              SYMBOL    (       9) = 'RND';                        591 1
148     2              SYM_TYPE  (       9) = SS_FUNC;                      591 1
149     2              SYM_VALUE (       9) = 0.0;                          591 1
150     2              STRING_VAL(       9) = '*';                          591 1
                                                                            591 1
                       /* GENSYM(INR,SS_FUNC,0.0,*) */                      592 1
151     2              SYMBOL    (      10) = 'INR';                        592 1
152     2              SYM_TYPE  (      10) = SS_FUNC;                      592 1
153     2              SYM_VALUE (      10) = 0.0;                          592 1
154     2              STRING_VAL(      10) = '*';                          592 1
                                                                            592 1
155     2             SS_CUR,SS_MAX,SS_MAX_FNC =        10 ;                594 1
                                                                            595
156     2         END INITIALIZE;                                            596
                                                                            597
```

```
STMT LEVEL NEST

                /*********************************************************************/    598
                /*********************************************************************/    599
                /*********************************************************************/    600
                /*********************************************************************/    601
                /*********************************************************************/    602
                                                                                             603
157     1       RENUM:PROC;                                                                  604
                                                                                             605
                /*********************************************************************     606
                *                                                                   *    606
                *   THIS PROC RENUMBERS THE SOURCE PROGRAM.  IT IS ASSUMED THAT     *    606
                *   THE BASIC PROGRAM COMILED WITH NO ERRORS AND THE CONTENTS OF    *    606
                *   THE GLOBAL TABLES ARE INTACT.  THE SOURCE_CODE TABLE WILL BE    *    606
                *   UPDATED WITH THE RENUMBERED PROGRAM.                            *    606
                *                                                                   *    606
                *   A "DECK" OF THE RENUMBERED PROGRAM WILL BE WRITTEN TO THE       *    606
                *   RENUMFL                                                         *    606
                *                                                                   *    606
                * NESTING:NONE                                                      *    606
                *********************************************************************/    606
                                                                                             617
                                                                                             618
158     2           DECLARE LINE_WORK           CHAR(80);                                    619
159     2           DECLARE LINE_SUB            FIXED BINARY ALIGNED;                        620
160     2           DECLARE A_BLANK             FIXED BINARY ALIGNED;                        621
161     2           DECLARE FIRST_CHAR          FIXED BINARY ALIGNED;                        622
162     2           DECLARE FIRST_DIGIT         FIXED BINARY ALIGNED;                        623
163     2           DECLARE (I,LAST_CHAR)       FIXED BINARY ALIGNED;                        624
164     2           DECLARE OLD_LINE_NUM        FIXED DECIMAL(5,0);                          625
165     2           DECLARE NEW_LINE_NUM(500)   FIXED DECIMAL(5,0);                          626
166     2           DECLARE CONTINUE_SCAN       BIT(1) ALIGNED;                              627
167     2           DECLARE EDIT_LINE_NUM       PIC 'ZZZZ9';                                 628
168     2           DECLARE RENUMFL             STREAM OUTPUT FILE;                          629
                                                                                             630
169     2           DO LINE_SUB = 1 TO LS_MAX;              /* NEW NUM START AT   */     631
170     2   1           NEW_LINE_NUM(LINE_SUB)=LINE_SUB*10; /* 10 BY 10 FOR NOW   */     632
171     2   1       END;                                    /* CORR TO LINE_STACK */     633
                                                                                             634
172     2           PUT FILE(RENUMFL) EDIT('++BASIC') (SKIP,A);                              635
173     2           DO LINE_SUB = 1 TO SC_MAX;                                               636
174     2   1           LINE_WORK = SOURCE_LINE(LINE_SUB);                                   637
175     2   1           IF SUBSTR(LINE_WORK,1,1)='*' THEN;                                   638
177     2   1           ELSE                                                                 639
177     2   1           DO;                                                                  640
178     2   2              A_BLANK = INDEX(LINE_WORK,' ');  /* FIND FIRST SPACE */       641
```

```
STMT LEVEL NEST

179     2    2              IF A_BLANK < 2 THEN                                          642
180     2    2              DO;                                                          643
181     2    3                 PUT SKIP LIST('**** RENUM FATAL ERROR 1 ****');           644
182     2    3                 STOP;                                                     645
183     2    3              END;                                                         646
184     2    2              OLD_LINE_NUM = SUBSTR(LINE_WORK, 1, A_BLANK-1);              647
185     2    2              CONTINUE_SCAN=TRUE;                                          648
186     2    2              DO I=1 TO LS_MAX WHILE(CONTINUE_SCAN);                       649
187     2    3                 IF OLD_LINE_NUM=LS_LINE(I) THEN                           650
188     2    3                 DO;                                                       651
189     2    4                     EDIT_LINE_NUM = NEW_LINE_NUM(I);                      652
190     2    4                     CALL TRIM_EDIT_NUM;                                   653
191     2    4                     LINE_WORK=SUBSTR(EDIT_LINE_NUM,FIRST_DIGIT) ||        654
                                           SUBSTR(LINE_WORK,A_BLANK);                   655
192     2    4                     CONTINUE_SCAN=FALSE;                                  656
193     2    4                 END;                                                      657
194     2    3              END;                                                         658
195     2    2              IF CONTINUE_SCAN THEN                                        659
196     2    2              DO;                                                          660
197     2    3                 PUT SKIP LIST('**** RENUM FATAL ERROR 2 ****');           661
198     2    3                 STOP;                                                     662
199     2    3              END;                                                         663
                                                                                        664
200     2    2              CONTINUE_SCAN=TRUE;                                          665
201     2    2              A_BLANK = INDEX(LINE_WORK,' ');  /* FIND FIRST SPACE */      666
202     2    2              IF A_BLANK = 0 THEN                                          667
203     2    2              DO;                                                          668
204     2    3                 PUT SKIP LIST('**** RENUM FATAL ERROR 3 ****');           669
205     2    3                 STOP;                                                     670
206     2    3              END;                                                         671
                                                                                        672
207     2    2              DO I=A_BLANK+1 TO STMT_RIGHT WHILE(CONTINUE_SCAN);           673
208     2    3                 IF SUBSTR(LINE_WORK,I,1)=' ' THEN;                        674
210     2    3                 ELSE CONTINUE_SCAN=FALSE;                                 675
211     2    3              END;                                                         676
                                                                                        677
212     2    2              I=I-1;                                                       678
213     2    2              IF SUBSTR(LINE_WORK,I,3)='IF ' |                             679
                               SUBSTR(LINE_WORK,I,2)='GO'   THEN                        680
214     2    2              DO;                                                          681
215     2    3                 CONTINUE_SCAN=TRUE;                                       682
216     2    3                 I=STMT_RIGHT;                                             683
217     2    3                 DO WHILE(CONTINUE_SCAN);                                  684
218     2    4                    IF SUBSTR(LINE_WORK,I,1)=' ' THEN                      685
219     2    4                       I=I-1;                                              686
```

```
STMT LEVEL NEST

220     2    4                    ELSE                                                687
220     2    4                       CONTINUE_SCAN=FALSE;                             688
221     2    4                    END;                                                689
222     2    3                    CONTINUE_SCAN=TRUE;                                 690
223     2    3                    LAST_CHAR=I;                                        691
224     2    3                    DO WHILE(CONTINUE_SCAN);                            692
225     2    4                       IF SUBSTR(LINE_WORK,I,1)=' ' THEN                693
226     2    4                          CONTINUE_SCAN=FALSE;                          694
227     2    4                       ELSE                                             695
227     2    4                          I=I-1;                                        696
228     2    4                    END;                                                697
229     2    3                    FIRST_CHAR=I;                                       698
230     2    3                    OLD_LINE_NUM = SUBSTR(LINE_WORK,I+1,LAST_CHAR-I);   699
231     2    3                    CONTINUE_SCAN=TRUE;                                 700
232     2    3                    DO I=1 TO LS_MAX WHILE(CONTINUE_SCAN);              701
233     2    4                       IF OLD_LINE_NUM=LS_LINE(I) THEN                  702
234     2    4                       DO;                                              703
235     2    5                           EDIT_LINE_NUM = NEW_LINE_NUM(I);             704
236     2    5                           CALL TRIM_EDIT_NUM;                          705
237     2    5                           LINE_WORK=SUBSTR(LINE_WORK,1,FIRST_CHAR)     706
                                                 || SUBSTR(EDIT_LINE_NUM,FIRST_DIGIT);  707
238     2    5                           CONTINUE_SCAN=FALSE;                         708
239     2    5                       END;                                             709
240     2    4                    END;                                                710
241     2    3                 END;                                                   711
242     2    2                 SOURCE_LINE(LINE_SUB)=LINE_WORK;                       712
243     2    2              END;                                                      713
244     2    1              PUT FILE(RENUMFL) EDIT(SOURCE_LINE(LINE_SUB)) (SKIP,A);   714
245     2    1           END;                                                         715
246     2               BASIC_RENUM=FALSE;                                           716
                                                                                     717
247     2            TRIM_EDIT_NUM:PROC;                                             718
                        /*                                                           719 1
248     3                SELECT (TRUE)   */ DO;                                      719 1
                                            /*                                       720 1
                         WHEN (NEW_LINE_NUM(I)<10)    */                           720 1
249     3    1                  IF (NEW_LINE_NUM(I)<10) THEN                        720 1
250     3    1                                DO;                                   720 1
251     3    2                  FIRST_DIGIT = 5;                                    721
252     3    2                           GO TO  ENDSELECT_MACRO1; END; /*           722 1
                         WHEN (NEW_LINE_NUM(I)<100)    */                         722 1
254     3    1                  IF (NEW_LINE_NUM(I)<100) THEN                       722 1
255     3    1                                DO;                                   722 1
256     3    2                  FIRST_DIGIT = 4;                                    723
257     3    2                           GO TO  ENDSELECT_MACRO1; END; /*           724 1
```

```
STMT LEVEL NEST

                        WHEN (NEW_LINE_NUM(I)<1000)    */                          724 1
259     3    1                  IF (NEW_LINE_NUM(I)<1000) THEN                    724 1
260     3    1                              DO;                                   724 1
261     3    2            FIRST_DIGIT = 3;                                        725
262     3    2                        GO TO  ENDSELECT_MACRO1; END; /*            726 1
                        WHEN (NEW_LINE_NUM(I)<10000)    */                        726 1
264     3    1                  IF (NEW_LINE_NUM(I)<10000) THEN                   726 1
265     3    1                              DO;                                   726 1
266     3    2            FIRST_DIGIT = 2;                                        727
267     3    2                        END; /*                                     728 1
268     3    1           OTHERWISE     */  ELSE DO;                               728 1
                                                                                  728 1
269     3    2            FIRST_DIGIT = 1;                                        729
270     3    2                      END; /*                                       730 1
271     3    1         ENDSELECT     */  END; ENDSELECT_MACRO1:; ;                730 1
274     3         END TRIM_EDIT_NUM;                                               731
                                                                                  732
275     2         END RENUM;                                                       733
```

```
STMT LEVEL NEST

                /*****************************************************************/    734
                /*****************************************************************/    735
                /*****************************************************************/    736
                /*****************************************************************/    737
                /*****************************************************************/    738
                                                                                         739
276     1       COMPILE:PROC;                                                            740
                                                                                         741
                /*****************************************************************     742
                *                                                               *    742
                *   THIS PROC DRIVES THE COMPILE PROCESS FOR THE BASIC PROGRAM   *    742
                *                                                               *    742
                * NESTING:NONE                                                  *    742
                *****************************************************************/    742
                                                                                         747
                                                                                         748
277     2           DECLARE LAST_PCODE_PRINTED        FIXED BINARY ALIGNED INITIAL(0);   749
278     2           DECLARE TERMINATE_SCAN            BIT(1) ALIGNED;                    750
279     2           DECLARE (FUNC_NAME,FUNC_ARG) CHAR(10);                               751
280     2           DECLARE (TMP_CNT,STR_CNT)              PICTURE '99';                 752
                                                                                         753
281     2           ON ENDPAGE(SYSPRINT)                                                 754
282     2           BEGIN;                                                               755
283     3              IF PAGE_NUM > 0 THEN PUT PAGE;                                    756
285     3              PAGE_NUM=PAGE_NUM+1;                                              757
286     3              PGM_PAGE_NUM=PGM_PAGE_NUM+1;                                      758
287     3              PUT EDIT (PAGE_TITLE,'DATE ',RUN_DATE,                            759
                                'PAGE ',PGM_PAGE_NUM)                                    760
                               (COLUMN(60),A,COLUMN(93),A,A,COLUMN(110),                 761
                                A,F(5,0));                                               762
288     3              PUT SKIP(2) EDIT(MONITOR_STMT) (A)                                763
                                   EDIT('OFFSET') (SKIP(2),A);                           764
289     3              PUT SKIP;                                                         765
290     3           END;                                                                 766
                                                                                         767
291     2           SIGNAL ENDPAGE(SYSPRINT);                                            768
                                                                                         769
292     2           STR_CNT = 0;                                                         770
293     2           SC_CUR=0;                                                            771
294     2           DO WHILE(SC_CUR<SC_MAX);                                             772
295     2   1           SC_CUR=SC_CUR+1;                                                 773
296     2   1           STMT=SOURCE_LINE(SC_CUR);                                        774
297     2   1           PUT SKIP EDIT(PC_MAX+1,STMT) (P'999999',COLUMN(25),A);           775
298     2   1           IF SUBSTR(STMT,1,1)='*' THEN      /* DONT COMPILE OPTIONS */     776
299     2   1              CALL PROCESS_OPTS;                                            777
```

```
STMT LEVEL NEST

300     2    1            ELSE                                                  778
300     2    1            DO;                                                   779
301     2    2               STMT_CH=STMT_LEFT;                                 780
302     2    2               TERMINATE_SCAN=FALSE;                              781
303     2    2               CALL GET_STMT_NUM(TRUE);                           782
304     2    2               LS_MAX=LS_MAX+1;                                   783
305     2    2               CALL ADD_PCODE(PC_OPCODE_SLN,LS_MAX);              784
306     2    2               LS_LINE(LS_MAX)=LAST_LINE_NUM;                     785
307     2    2               LS_OFFSET(LS_MAX)=PC_MAX;                          786
308     2    2               CALL GET_KEYWORD;                                  787
309     2    2               CALL PROCESS_KEYWORD;                              788
310     2    2               IF ICODE_PRINT THEN                                789
311     2    2                 CALL PRINT_PCODES;                               790
312     2    2            END;                                                  791
313     2    1         END;                                                     792
                                                                               793
314     2              ON ENDPAGE(SYSPRINT)                                     794
315     2              BEGIN;                                                   795
316     3                 IF PAGE_NUM > 0 THEN PUT PAGE;                        796
318     3                 PAGE_NUM=PAGE_NUM+1;                                  797
319     3                 PGM_PAGE_NUM=PGM_PAGE_NUM+1;                          798
320     3                 PUT EDIT (PAGE_TITLE,'DATE ',RUN_DATE,                799
                                    'PAGE ',PGM_PAGE_NUM)                      800
                                   (COLUMN(60),A,COLUMN(93),A,A,COLUMN(110),   801
                                    A,F(5,0));                                 802
321     3                 PUT SKIP(2);                                          803
322     3              END;                                                     804
                                                                               805
323     2              IF TABLE_PRINT THEN;                                     806
325     2              ELSE                                                     807
325     2                 GO TO END_OF_COMP;                                    808
                                                                               809
326     2              DECLARE I                  FIXED BINARY ALIGNED;         810
                                                                               811
327     2              CALL PRINT_SYMBOLS;                                      812
                                                                               813
328     2              PUT SKIP(2) LIST('DEF NAME','OFFSET');                   814
329     2              DO I=1 TO DF_MAX;                                        815
330     2    1           PUT SKIP LIST(DF_NAME(I),DF_OFFSET(I));               816
331     2    1         END;                                                     817
332     2              PUT SKIP LIST('END OF DEF NAME TABLE');                  818
                                                                               819
333     2              PUT SKIP(2) EDIT('OFFSET','LINE  OP   OBJECT') (A,X(7),A);820
334     2              CALL PRINT_PCODES;                                       821
335     2              PUT SKIP LIST('END OF PCODE TABLE');                     822
```

```
STMT LEVEL NEST

                                                                                823
336     2         END_OF_COMP:                                                  824
                      PUT SKIP(2) EDIT('**** END OF COMPILATION ****') (A);     825
337     2             IF ERROR_COUNT=0 THEN                                     826
338     2             DO;                                                       827
339     2    1           PUT EDIT(' NO ERRORS FOUND') (A);                      828
340     2    1           IF BASIC_RENUM THEN                                    829
341     2    1               PUT EDIT(' - RENUMBERING PROGRAM') (A);            830
342     2    1         END;                                                     831
343     2             ELSE                                                      832
343     2             DO;                                                       833
344     2    1           PUT EDIT(ERROR_COUNT,' ERRORS FOUND') (F(5),A);        834
345     2    1           IF BASIC_RENUM THEN                                    835
346     2    1               PUT EDIT(' - RENUMBERING BYPASSED') (A);           836
347     2    1         END;                                                     837
                                                                                838
348     2         PROCESS_OPTS:PROC;                                            839
                  /*****************************************************************  840
                  *                                                               *  840
                  *   OPTIONS STATEMENTS ARE MIXED IN WITH THE SOURCE PROGRAM.  THEY  *  840
                  *   HAVE A "*" IN COLUMN 1 AND ARE TREATED AS A COMMENT BY THE      *  840
                  *   COMPILER.  THESE ARE BASICLY DEBUGGING TOOLS BUILT IN AND WILL  *  840
                  *   NOT NORMALLY BE USED.  THEY OPTIONS ARE:                        *  840
                  *   *TABLE    PRINT ALL OBJECT TABLES AT THE END OF COMPILATION     *  840
                  *   *DUMP     PRINT ALL OBJECT TABLES AT THE END OF EXECUTION       *  840
                  *   *STACK    PRINT THE PARSING STACK DEBUGGING TRACING.  THIS      *  840
                  *             OPTION STARTS AS SOON AS THE STATEMENT IS PROCESSED   *  840
                  *             IT REMAINS IN EFFECT UNTIL THE END OF PROGRAM OR      *  840
                  *             A *NOSTACK IS PROCESSED.                              *  840
                  *   *ICODE    PRINTS THE PCODE OBJECT CODE AS IT IS GENERATED.      *  840
                  *             OPTION STARTS AS SOON AS THE STATEMENT IS PROCESSED   *  840
                  *             IT REMAINS IN EFFECT UNTIL THE END OF PROGRAM OR      *  840
                  *             A *NOICODE IS PROCESSED.                              *  840
                  *   *TRACE    PRINT DEBUGGING INFORMATION WHILE THE BASIC PROGRAM   *  840
                  *             IS EXECUTING.                                         *  840
                  *                                                               *  840
                  * NESTING:COMPILE                                               *  840
                  *****************************************************************/ 840
                                                                                860
                                                                                861
349     3             IF SUBSTR(STMT,1,7)='*TABLE ' THEN                        862
350     3                TABLE_PRINT=TRUE;                                      863
351     3             ELSE                                                      864
351     3             IF SUBSTR(STMT,1,9)='*NOTABLE ' THEN                      865
352     3                TABLE_PRINT=FALSE;                                     866
```

```
STMT LEVEL NEST

353     3             ELSE                                                          867
353     3             IF SUBSTR(STMT,1,6)='*DUMP ' THEN                             868
354     3                TABLE_DUMP=TRUE;                                           869
355     3             ELSE                                                          870
355     3             IF SUBSTR(STMT,1,7)='*STACK ' THEN                            871
356     3                STACK_PRINT_DEBUG=TRUE;                                    872
357     3             ELSE                                                          873
357     3             IF SUBSTR(STMT,1,9)='*NOSTACK ' THEN                          874
358     3                STACK_PRINT_DEBUG=FALSE;                                   875
359     3             ELSE                                                          876
359     3             IF SUBSTR(STMT,1,7)='*ICODE ' THEN                            877
360     3                ICODE_PRINT=TRUE;                                          878
361     3             ELSE                                                          879
361     3             IF SUBSTR(STMT,1,9)='*NOICODE ' THEN                          880
362     3                ICODE_PRINT=FALSE;                                         881
363     3             IF SUBSTR(STMT,1,7)='*TRACE ' THEN                            882
364     3                EXECUTION_DEBUG=TRUE;                                      883
365     3             ELSE                                                          884
365     3             IF SUBSTR(STMT,1,9)='*NOTRACE ' THEN                          885
366     3                EXECUTION_DEBUG=FALSE;                                     886
367     3             ELSE;  /*  JUST IGNORE INVALID OPTIONS */                     887
                                                                                   888
368     3          END PROCESS_OPTS;                                                889
                                                                                   890
369     2          PRINT_PCODES:PROC;                                               891
                   /*****************************************************************
                   *                                                               *  892
                   *                                                               *  892
                   * NESTING:COMPILE                                               *  892
                   *****************************************************************/ 892
                                                                                   896
                                                                                   897
370     3             IF LAST_PCODE_PRINTED < PC_MAX THEN                           898
371     3             DO;                                                           899
372     3    1          DO I=LAST_PCODE_PRINTED+1 TO PC_MAX;                        900
                                 /*                                                901 1
373     3    2              SELECT (PC_FORMAT(PC_OPCODE(I)))   */ DO;               901 1
                                    /*                                             902 1
                            WHEN (PC_FORMAT_0)    */                              902 1
374     3    3                    IF (PC_FORMAT(PC_OPCODE(I)))=(PC_FORMAT_0) THEN 902 1
375     3    3                                  DO;                               902 1
376     3    4                PUT SKIP EDIT(I,   PC_MNEMONIC(PC_OPCODE(I)),      903
                                                 SYMBOL(PC_OBJECT(I)))             904
                                               (P'999999',X(13),A,X(2),A);       905
377     3    4                        GO TO  ENDSELECT_MACRO2; END; /*            906 1
```

```
STMT LEVEL NEST

                          WHEN (PC_FORMAT_1)    */                                    906 1
379     3    3                    IF (PC_FORMAT(PC_OPCODE(I)))=(PC_FORMAT_1) THEN  906 1
380     3    3                       DO;                                            906 1
381     3    4                 PUT SKIP EDIT(I,    LS_LINE(PC_OBJECT(I)),          907
                                    PC_MNEMONIC(PC_OPCODE(I)))                      908
                                   (P'999999',X(3),A,X(2),A);                       909
382     3    4                       GO TO  ENDSELECT_MACRO2; END; /*               910 1
                          WHEN (PC_FORMAT_2)    */                                    910 1
384     3    3                    IF (PC_FORMAT(PC_OPCODE(I)))=(PC_FORMAT_2) THEN  910 1
385     3    3                       DO;                                            910 1
386     3    4                   PUT SKIP EDIT(I,   PC_MNEMONIC(PC_OPCODE(I)),      911
                                    PC_OBJECT(I))                                   912
                                   (P'999999',X(13),A,X(2),A);                      913
387     3    4                       GO TO  ENDSELECT_MACRO2; END; /*               914 1
                          WHEN (PC_FORMAT_3)    */                                    914 1
389     3    3                    IF (PC_FORMAT(PC_OPCODE(I)))=(PC_FORMAT_3) THEN  914 1
390     3    3                       DO;                                            914 1
391     3    4                   PUT SKIP EDIT(I,   PC_MNEMONIC(PC_OPCODE(I)),      915
                                    STRING_VAL(PC_OBJECT(I)))                       916
                                   (P'999999',X(13),A,X(2),A);                      917
392     3    4                       GO TO  ENDSELECT_MACRO2; END; /*               918 1
                          WHEN (PC_FORMAT_4)    */                                    918 1
394     3    3                    IF (PC_FORMAT(PC_OPCODE(I)))=(PC_FORMAT_4) THEN  918 1
395     3    3                       DO;                                            918 1
396     3    4                   PUT SKIP EDIT(I,   PC_MNEMONIC(PC_OPCODE(I)),      919
                                    ' ')                                            920
                                   (P'999999',X(13),A,X(2),A);                      921
397     3    4                       GO TO  ENDSELECT_MACRO2; END; /*               922 1
                          WHEN (PC_FORMAT_5)    */                                    922 1
399     3    3                    IF (PC_FORMAT(PC_OPCODE(I)))=(PC_FORMAT_5) THEN  922 1
400     3    3                       DO;                                            922 1
401     3    4                   PUT SKIP EDIT(I,   PC_MNEMONIC(PC_OPCODE(I)),      923
                                    PC_OBJECT(I))                                   924
                                   (P'999999',X(13),A,X(2),F(5));                   925
402     3    4                     END; /*                                          926 1
403     3    3                OTHERWISE     */  ELSE DO;                           926 1
                                                                                    926 1
404     3    4                  PUT EDIT('**** FATAL ERROR IN COMPILER *****',      927
                                    '**** INVALID VALUE FOR PC_OPCODE ',            928
                                    PC_MNEMONIC(PC_OPCODE(I)))                      929
                                    (SKIP(2),A,SKIP,A,A);                           930
405     3    4                  STOP;                                               931
406     3    4                     END; /*                                          932 1
407     3    3                ENDSELECT     */  END; ENDSELECT_MACRO2:;             932 1
409     3    2             END;                                                     933
```

```
STMT LEVEL NEST

410     3    1           LAST_PCODE_PRINTED = PC_MAX;                                  934
411     3    1         END;                                                            935
                                                                                       936
412     3         END PRINT_PCODES;                                                    937
                                                                                       938
413     2         SKIP_BLANKS:PROC;                                                    939
                  /*********************************************************************  940
                  *                                                                   *  940
                  *                                                                   *  940
                  * NESTING:COMPILE                                                   *  940
                  *********************************************************************/ 940
                                                                                       944
414     3            DECLARE CONTINUE_SCAN       BIT(1)    ALIGNED;                  945
415     3            DECLARE I                   FIXED BIN ALIGNED;                  946
                                                                                       947
416     3            CONTINUE_SCAN=TRUE;                                              948
417     3            DO I=STMT_CH TO STMT_RIGHT WHILE(CONTINUE_SCAN);                 949
418     3    1          IF SUBSTR(STMT,I,1)=' ' THEN ;                                950
420     3    1          ELSE                                                          951
420     3    1          DO;                                                           952
421     3    2             STMT_CH=I;                                                 953
422     3    2             CONTINUE_SCAN=FALSE;                                       954
423     3    2          END;                                                          955
424     3    1       END;                                                             956
425     3            IF CONTINUE_SCAN THEN      /* NO NON BLANK FOUND */              957
426     3               STMT_CH=STMT_RIGHT+1;                                         958
                                                                                       959
427     3         END SKIP_BLANKS;                                                     960
                                                                                       961
428     2         PRINT_ERR:PROC(I,MSG);                                               962
                  /*********************************************************************  963
                  *                                                                   *  963
                  *   PRINTS ALL ERROR MESSAGE FOR THE COMPILE PHASE                  *  963
                  *                                                                   *  963
                  *                                                                   *  963
                  * NESTING:COMPILE                                                   *  963
                  *********************************************************************/ 963
                                                                                       969
429     3            DECLARE I                   FIXED BINARY ALIGNED;               970
430     3            DECLARE MSG                 CHAR(*);                            971
431     3            PUT SKIP EDIT('*****','^',MSG)                                   972
                              (A,COLUMN(24+I),A,SKIP,COLUMN(11),A);                    973
432     3            ERROR_COUNT=ERROR_COUNT+1;                                       974
433     3            TERMINATE_SCAN=TRUE;                                             975
434     3         END PRINT_ERR;                                                       976
```

```
STMT LEVEL NEST

                                                                                                                977
435     2         LOOKUP_SYMBOL_TABLE:PROC(V) RETURNS(FIXED BINARY);                                            978
                  /*****************************************************************                           979
                  *                                                               *                           979
                  *  LOOKS UP SYMBOLS IN THE SYMBOL TABLE.  IF NOT FOUND, IT ADDS *                           979
                  *  IT AND DETERMINE WHAT TYPE OF SYMBOL IT IS.                  *                           979
                  *  IF A ZERO IS RETURNED, THERE WAS AN ERROR.  OTHERWISE THE    *                           979
                  *  SUBSCRIPT OF THE ITEM "V" IS RETURNED.                       *                           979
                  *                                                               *                           979
                  * NESTING:COMPILE                                               *                           979
                  *****************************************************************/                          979
                                                                                                                987
                                                                                                                988
436     3            DECLARE V                   CHAR(10),                                                    989
                            OPT                  FIXED BINARY ALIGNED;                                        990
437     3            DECLARE I                   FIXED BINARY ALIGNED;                                        991
438     3            DECLARE STR_IND             FIXED BINARY ALIGNED;                                        992
                                                                                                              993
439     3            ON CONVERSION                                                                            994
440     3            BEGIN;                                                                                   995
441     4               CONTINUE_SCAN=FALSE;                                                                  996
442     4               ONCHAR='0';                                                                           997
443     4            END;                                                                                     998
                                                                                                              999
444     3            DO I=1 TO SS_MAX;                                                                       1000
445     3    1          IF SYMBOL(I)=V THEN                                                                  1001
446     3    1          DO;                                                                                  1002
447     3    2             IF SYM_TYPE(I)=SS_DIM_VAR &                                                       1003
                              WORD=KW_DIM THEN                                                              1004
448     3    2                CALL PRINT_ERR(I,'CANNOT REDIM VARIABLE');                                      1005
449     3    2             RETURN(I);                                                                        1006
450     3    2          END;                                                                                 1007
451     3    1       END; /* OF DO */                                                                        1008
                                                                                                            1009
452     3            IF SS_MAX=HBOUND(SYMBOL,1) THEN                                                        1010
453     3            DO;                                                                                    1011
454     3    1          CALL PRINT_ERR(10,'SYMBOL TABLE OVERFLOW');                                          1012
455     3    1          RETURN(0);                                                                           1013
456     3    1       END;                                                                                    1014
                                                                                                            1015
457     3            SS_MAX=SS_MAX+1;                                                                       1016
458     3            SYMBOL(SS_MAX)=V;                                                                      1017
459     3            SYM_DIM_MAX(SS_MAX)=0;                                                                 1018
460     3            STRING_VAL(SS_MAX)='*';                                                                1019
461     3            IF SUBSTR(V,1,1) >= 'A' & SUBSTR(V,1,1) <= 'Z' THEN                                    1020
```

```
STMT LEVEL NEST

462     3           DO;                                                                 1021
463     3    1         STR_IND=VERIFY(V,VALID_VAR_CHARS);                             1022
464     3    1         IF STR_IND > 0 THEN                                            1023
465     3    1         DO;                                                            1024
466     3    2            IF SUBSTR(V,STR_IND,1)='$' THEN                             1025
467     3    2            DO;                                                         1026
468     3    3               SYM_VALUE(SS_MAX)=0.0;                                   1027
469     3    3               IF SUBSTR(V,1,4)='STR$' THEN                             1028
470     3    3                  SYM_TYPE(SS_MAX)=SS_STRCON;                           1029
471     3    3               ELSE                                                     1030
471     3    3                  IF WORD=KW_DIM THEN                                   1031
472     3    3                     SYM_TYPE(SS_MAX)=SS_STRDIM;                        1032
473     3    3                  ELSE                                                  1033
473     3    3                     SYM_TYPE(SS_MAX)=SS_STRVAR;                        1034
474     3    3            END;                                                        1035
475     3    2            ELSE                                                        1036
475     3    2            DO;                                                         1037
476     3    3               CALL PRINT_ERR(I,'INVALID VARIABLE NAME');               1038
477     3    3               RETURN(0);                                               1039
478     3    3            END;                                                        1040
479     3    2         END;                                                           1041
480     3    1         ELSE                                                           1042
480     3    1         DO;                                                            1043
481     3    2            SYM_VALUE(SS_MAX)=0.0;                                      1044
482     3    2            IF WORD=KW_DIM THEN                                         1045
483     3    2                SYM_TYPE(SS_MAX)=SS_DIM_VAR;                            1046
484     3    2            ELSE                                                        1047
484     3    2                SYM_TYPE(SS_MAX)=SS_VAR;                                1048
485     3    2         END;                                                           1049
486     3    1      END;                                                              1050
487     3           ELSE                                                              1051
487     3           DO;                                                               1052
488     3    1         IF VERIFY(V,'0123456789+-.E ') > 0 THEN                        1053
489     3    1         DO;                                                            1054
490     3    2            CALL PRINT_ERR(I,'INVALID CONSTANT '||V);                   1055
491     3    2            RETURN(0);                                                  1056
492     3    2         END;                                                           1057
493     3    1         ELSE                                                           1058
493     3    1         DO;                                                            1059
494     3    2            SYM_VALUE(SS_MAX)=V;                                        1060
495     3    2            SYM_TYPE(SS_MAX)=SS_CONST;                                  1061
496     3    2         END;                                                           1062
497     3    1      END;                                                              1063
498     3           RETURN(SS_MAX);                                                   1064
                                                                                      1065
```

```
STMT LEVEL NEST

499     3         END LOOKUP_SYMBOL_TABLE;                                                1066
                                                                                          1067
500     2         ADD_PCODE:PROC(PCODE,OFFSET);                                           1068
                  /*********************************************************************  1069
                  *                                                                   *  1069
                  *   ADDS CODES TO THE PSEUDO MACHINE CODE TABLE                     *  1069
                  *                                                                   *  1069
                  * NESTING:COMPILE                                                   *  1069
                  *********************************************************************/ 1069
                                                                                          1074
                                                                                          1075
501     3            DECLARE PCODE        FIXED BINARY ALIGNED,                         1076
                            OFFSET       FIXED BINARY ALIGNED;                         1077
                                                                                          1078
502     3            IF PC_MAX+1>HBOUND(PC_OPCODE,1) THEN                                 1079
503     3            DO;                                                                  1080
504     3    1          PUT SKIP(2) EDIT('**** FATAL ERROR-PCODE TABLE OVERFLOW *****')   1081
                                 (A);                                                     1082
505     3    1          STOP;                                                            1083
506     3    1       END;                                                                1084
                                                                                          1085
507     3            PC_MAX=PC_MAX+1;                                                     1086
508     3            PC_OPCODE(PC_MAX)=PCODE;                                             1087
509     3            PC_OBJECT(PC_MAX)=OFFSET;                                            1088
                                                                                          1089
510     3         END ADD_PCODE;                                                          1090
                                                                                          1091
511     2         GET_STMT_NUM:PROC(DEFINITION);                                          1092
                  /*********************************************************************  1093
                  *                                                                   *  1093
                  *   EXTRACT THE STATMENT NUMBER.  MAKE SURE IT IS NUMERIC AND IN    *  1093
                  *   SEQUENCE.  IF OK, ADD IT TO THE LINE STACK                      *  1093
                  *                                                                   *  1093
                  * NESTING:COMPILE                                                   *  1093
                  *********************************************************************/ 1093
                                                                                          1099
512     3            DECLARE DEFINITION          BIT(1) ALIGNED;                       1100
513     3            DECLARE I                   FIXED BINARY ALIGNED;                 1101
514     3            DECLARE CONTINUE_SCAN       BIT(1) ALIGNED;                       1102
515     3            DECLARE CH                  CHAR(1);                              1103
516     3            DECLARE LN                  CHAR(6) VARYING;                      1104
517     3            DECLARE LINE_NUM            FIXED DECIMAL(5,0);                   1105
                                                                                          1106
518     3            LN='';                                                               1107
519     3            TMP_CNT=0;                                                           1108
```

```
STMT LEVEL NEST

520     3           CONTINUE_SCAN=TRUE;                                          1109
521     3           DO I=STMT_CH TO STMT_RIGHT WHILE(CONTINUE_SCAN);             1110
522     3    1         CH=SUBSTR(STMT,I,1);                                      1111
523     3    1         IF CH=' ' THEN CONTINUE_SCAN=FALSE;                       1112
525     3    1         ELSE                                                      1113
525     3    1            IF CH < '0' | CH >'9' THEN                             1114
526     3    1            DO;                                                    1115
527     3    2               CONTINUE_SCAN=FALSE;                                1116
528     3    2               CALL PRINT_ERR(I,'INVALID LINE NUMBER');            1117
529     3    2            END;                                                   1118
530     3    1            ELSE                                                   1119
530     3    1            DO;                                                    1120
531     3    2               LN=LN||CH;                                          1121
532     3    2               IF LENGTH(LN)>5 THEN                                1122
533     3    2               DO;                                                 1123
534     3    3                  CONTINUE_SCAN=FALSE;                             1124
535     3    3                  CALL PRINT_ERR(I,'LINE NUMBER TOO LONG');        1125
536     3    3               END;                                                1126
537     3    2            END;                                                   1127
538     3    1         END;                                                      1128
                                                                                1129
539     3           IF DEFINITION=TRUE THEN                                      1130
540     3           DO;                                                          1131
541     3    1         LINE_NUM=LN;                                              1132
542     3    1         IF LINE_NUM=LAST_LINE_NUM THEN                            1133
543     3    1         DO;                                                       1134
544     3    2            CONTINUE_SCAN=FALSE;                                   1135
545     3    2            CALL PRINT_ERR(STMT_CH,'DUPLICATE LINE NUMBER');       1136
546     3    2         END;                                                      1137
547     3    1         ELSE                                                      1138
547     3    1            IF LINE_NUM<LAST_LINE_NUM THEN                         1139
548     3    1            DO;                                                    1140
549     3    2               CONTINUE_SCAN=FALSE;                                1141
550     3    2               CALL PRINT_ERR(STMT_CH,'LINE NUMBER OUT OF SEQUENCE');1142
551     3    2            END;                                                   1143
552     3    1            ELSE                                                   1144
552     3    1            DO;                                                    1145
553     3    2               IF LS_MAX=HBOUND(LS_LINE,1) THEN                    1146
554     3    2                  CALL PRINT_ERR(STMT_CH,'TOO MANY LINE NUMBERS');1147
555     3    2            END;                                                   1148
556     3    1         LAST_LINE_NUM=LINE_NUM;                                   1149
557     3    1      END;                                                         1150
558     3           ELSE                                                         1151
558     3              REF_LINE_NUM=LN;                                          1152
                                                                                1153
```

```
STMT LEVEL NEST

559     3            STMT_CH=I;                                                    1154
                                                                                   1155
560     3         END GET_STMT_NUM;                                                1156
                                                                                   1157
561     2         GET_KEYWORD:PROC;                                                1158
                  /*****************************************************************   1159
                  *                                                               *   1159
                  *   EXTRACT THE STATMENT KEYWORD AND VALIDATE IT                *   1159
                  *                                                               *   1159
                  * NESTING:COMPILE                                               *   1159
                  *****************************************************************/  1159
                                                                                   1164
562     3            DECLARE (I,J)               FIXED BINARY ALIGNED;             1165
563     3            DECLARE CONTINUE_SCAN       BIT(1) ALIGNED;                   1166
564     3            DECLARE CH                  CHAR(1);                          1167
565     3            DECLARE KW                  CHAR(9) VARYING;                  1168
                                                                                   1169
566     3            CALL SKIP_BLANKS;                                             1170
567     3            IF STMT_CH=STMT_RIGHT THEN                                    1171
568     3            DO;                                                           1172
569     3    1          CALL PRINT_ERR(STMT_CH,'BLANK LINE?');                      1173
570     3    1          RETURN;                                                     1174
571     3    1       END;                                                          1175
                                                                                   1176
572     3            KW='';                                                        1177
573     3            CONTINUE_SCAN=TRUE;                                           1178
574     3            DO I=STMT_CH TO STMT_RIGHT WHILE(CONTINUE_SCAN);              1179
575     3    1          CH=SUBSTR(STMT,I,1);                                        1180
576     3    1          IF CH=' ' THEN                                              1181
577     3    1          DO;                                                         1182
578     3    2             IF LENGTH(KW)=2 THEN                                     1183
579     3    2             DO;                                                      1184
580     3    3                IF KW='GO' THEN ;                                     1185
582     3    3                ELSE                                                  1186
582     3    3                   CONTINUE_SCAN=FALSE;                               1187
583     3    3             END;                                                     1188
584     3    2             ELSE                                                     1189
584     3    2                CONTINUE_SCAN=FALSE;                                  1190
585     3    2          END;                                                        1191
586     3    1          ELSE                                                        1192
586     3    1          DO;                                                         1193
587     3    2             KW=KW||CH;                                               1194
588     3    2             IF LENGTH(KW)>9 THEN                                     1195
589     3    2             DO;                                                      1196
590     3    3                CONTINUE_SCAN=FALSE;                                  1197
```

```
STMT LEVEL NEST

591     3    3               CALL PRINT_ERR(I,'KEYWORD TOO LONG');                    1198
592     3    3             END;                                                       1199
593     3    2          END;                                                          1200
594     3    1       END;                                                          1201
                                                                                   1202
595     3            WORD=KW;                                                      1203
                                                                                   1204
596     3            CONTINUE_SCAN=TRUE;                                           1205
597     3            DO J=1 TO HBOUND(KEY_WORDS,1) WHILE(CONTINUE_SCAN);           1206
598     3    1          IF WORD=KEY_WORDS(J) THEN                                   1207
599     3    1              CONTINUE_SCAN=FALSE;                                    1208
600     3    1       END;                                                          1209
601     3            IF CONTINUE_SCAN THEN                                         1210
602     3            DO;                                                           1211
603     3    1          CALL PRINT_ERR(STMT_CH,'INVALID KEYWORD');                  1212
604     3    1       END;                                                          1213
                                                                                   1214
605     3            STMT_CH=I;                                                    1215
                                                                                   1216
606     3         END GET_KEYWORD;                                                 1217
                                                                                   1218
607     2         PROCESS_KEYWORD:PROC;                                            1219
                  /*******************************************************************   1220
                  *                                                                 *   1220
                  *   SYNTAX CHECK AND COMPILE STATEMENTS                           *   1220
                  *                                                                 *   1220
                  * NESTING:COMPILE                                                 *   1220
                  *******************************************************************/   1220
                                                                                   1225
608     3            DECLARE I                   FIXED BINARY ALIGNED;             1226
609     3            DECLARE ERR_PTR             FIXED BINARY ALIGNED;             1227
610     3            DECLARE CONTINUE_SCAN       BIT(1) ALIGNED;                   1228
                                                                                   1229
                           /*                                                      1230 1
611     3             SELECT (WORD)   */ DO;                                        1230 1
                                      /*                                           1231 1
                     WHEN (KW_REM)    */                                          1231 1
612     3    1                IF (WORD)=(KW_REM) THEN                              1231 1
613     3    1                             DO;             /*  REMARKS - NOTHING TO DO! 1231 1
                   */                                                              1231
614     3    2                    GO TO  ENDSELECT_MACRO3; END; /*                1232 1
                     WHEN (KW_END)    */                                          1232 1
616     3    1                IF (WORD)=(KW_END) THEN                              1232 1
617     3    1                             DO;                                     1232 1
618     3    2           CALL SKIP_BLANKS;                                        1233
```

```
STMT LEVEL NEST

619     3    2          IF STMT_CH>STMT_RIGHT THEN                                    1234
620     3    2              CALL ADD_PCODE(PC_OPCODE_END,ZERO);                       1235
621     3    2          ELSE                                                          1236
621     3    2             CALL PRINT_ERR(STMT_CH,                                    1237
                                'INVALID SYNTAX - EXPECTING BLANKS AFTER END');      1238
622     3    2                GO TO  ENDSELECT_MACRO3; END; /*                        1239 1
                       WHEN (KW_STOP)    */                                          1239 1
624     3    1                IF (WORD)=(KW_STOP) THEN                                1239 1
625     3    1                     DO;                                                1239 1
626     3    2          CALL SKIP_BLANKS;                                             1240
627     3    2          IF STMT_CH>STMT_RIGHT THEN                                    1241
628     3    2              CALL ADD_PCODE(PC_OPCODE_STP,ZERO);                       1242
629     3    2          ELSE                                                          1243
629     3    2             CALL PRINT_ERR(STMT_CH,                                    1244
                                'INVALID SYNTAX - EXPECTING BLANKS AFTER STOP');     1245
630     3    2                GO TO  ENDSELECT_MACRO3; END; /*                        1246 1
                       WHEN (KW_RETURN)    */                                        1246 1
632     3    1                IF (WORD)=(KW_RETURN) THEN                              1246 1
633     3    1                     DO;                                                1246 1
634     3    2          CALL SKIP_BLANKS;                                             1247
635     3    2          IF STMT_CH>STMT_RIGHT THEN                                    1248
636     3    2              CALL ADD_PCODE(PC_OPCODE_RET,ZERO);                       1249
637     3    2          ELSE                                                          1250
637     3    2              CALL PRINT_ERR(STMT_CH,                                   1251
                                'INVALID SYNTAX - EXPECTING BLANKS AFTER RETURN');   1252
638     3    2                GO TO  ENDSELECT_MACRO3; END; /*                        1253 1
                       WHEN (KW_GOTO)    */                                          1253 1
640     3    1                IF (WORD)=(KW_GOTO) THEN                                1253 1
641     3    1                     DO;                                                1253 1
642     3    2             CALL SKIP_BLANKS;                                          1254
643     3    2             IF STMT_CH>STMT_RIGHT THEN                                 1255
644     3    2                CALL PRINT_ERR(STMT_CH,                                 1256
                                'INVALID SYNTAX - EXPECTING LINE NUMBER AFTER GOTO');1257
645     3    2             ELSE                                                       1258
645     3    2                CALL PROCESS_GOTO;                                      1259
646     3    2                GO TO  ENDSELECT_MACRO3; END; /*                        1260 1
                       WHEN (KW_GOSUB)    */                                        1260 1
648     3    1                IF (WORD)=(KW_GOSUB) THEN                               1260 1
649     3    1                     DO;                                                1260 1
650     3    2             CALL SKIP_BLANKS;                                          1261
651     3    2             IF STMT_CH>STMT_RIGHT THEN                                 1262
652     3    2                CALL PRINT_ERR(STMT_CH,                                 1263
                                'INVALID SYNTAX - EXPECTING LINE NUMBER AFTER GOSUB');1264
653     3    2             ELSE                                                       1265
653     3    2                CALL PROCESS_GOSUB;                                     1266
```

```
STMT LEVEL NEST

654     3    2                  GO TO  ENDSELECT_MACRO3; END; /*                    1267 1
                             WHEN (KW_DATA)    */                                    1267 1
656     3    1                    IF (WORD)=(KW_DATA) THEN                          1267 1
657     3    1                         DO;                                          1267 1
658     3    2                 CALL SKIP_BLANKS;                                    1268
659     3    2                 IF STMT_CH>STMT_RIGHT THEN                           1269
660     3    2                    CALL PRINT_ERR(STMT_CH,                           1270
                                       'INVALID SYNTAX - EXPECTING DATA ELEMENTS'); 1271
661     3    2                 ELSE                                                 1272
661     3    2                    CALL EXTRACT_DATA;                                1273
662     3    2                         GO TO  ENDSELECT_MACRO3; END; /*             1274 1
                             WHEN (KW_LET)    */                                    1274 1
664     3    1                    IF (WORD)=(KW_LET) THEN                           1274 1
665     3    1                         DO;                                          1274 1
666     3    2                 CALL SKIP_BLANKS;                                    1275
667     3    2                 IF STMT_CH>STMT_RIGHT THEN                           1276
668     3    2                    CALL PRINT_ERR(STMT_CH,                           1277
                                        'INVALID SYNTAX - EXPECTING LET STATEMENT'); 1278
669     3    2                 ELSE                                                 1279
669     3    2                    CALL PROCESS_LET;                                 1280
670     3    2                         GO TO  ENDSELECT_MACRO3; END; /*             1281 1
                             WHEN (KW_DEF)    */                                    1281 1
672     3    1                    IF (WORD)=(KW_DEF) THEN                           1281 1
673     3    1                         DO;                                          1281 1
674     3    2                 CALL SKIP_BLANKS;                                    1282
675     3    2                 IF STMT_CH>STMT_RIGHT THEN                           1283
676     3    2                    CALL PRINT_ERR(STMT_CH,                           1284
                                          'INVALID SYNTAX - EXPECTING FUNCTION');  1285
677     3    2                 ELSE                                                 1286
677     3    2                    CALL PROCESS_DEF;                                 1287
678     3    2                         GO TO  ENDSELECT_MACRO3; END; /*             1288 1
                             WHEN (KW_READ)    */                                    1288 1
680     3    1                    IF (WORD)=(KW_READ) THEN                          1288 1
681     3    1                         DO;                                          1288 1
682     3    2                 CALL SKIP_BLANKS;                                    1289
683     3    2                 IF STMT_CH>STMT_RIGHT THEN                           1290
684     3    2                    CALL PRINT_ERR(STMT_CH,                           1291
                                       'INVALID SYNTAX - EXPECTING VARIABLE(S) AFTER READ'); 1292
685     3    2                 ELSE                                                 1293
685     3    2                    CALL PROCESS_READ;                                1294
686     3    2                         GO TO  ENDSELECT_MACRO3; END; /*             1295 1
                             WHEN (KW_PRINT)    */                                   1295 1
688     3    1                    IF (WORD)=(KW_PRINT) THEN                         1295 1
689     3    1                         DO;                                          1295 1
690     3    2                 CALL SKIP_BLANKS;                                    1296
```

```
STMT LEVEL NEST

691     3    2          CALL PROCESS_PRINT;                                          1297
692     3    2                     GO TO  ENDSELECT_MACRO3; END; /*                  1298 1
                      WHEN (KW_IF)    */                                            1298 1
694     3    1               IF (WORD)=(KW_IF) THEN                                  1298 1
695     3    1                          DO;                                          1298 1
696     3    2          CALL SKIP_BLANKS;                                            1299
697     3    2          IF STMT_CH>STMT_RIGHT THEN                                   1300
698     3    2             CALL PRINT_ERR(STMT_CH,                                   1301
                               'INVALID SYNTAX - EXPECTING COMPARISON FOR IF');      1302
699     3    2          ELSE                                                         1303
699     3    2             CALL PROCESS_IF;                                          1304
700     3    2                     GO TO  ENDSELECT_MACRO3; END; /*                  1305 1
                      WHEN (KW_FOR)    */                                           1305 1
702     3    1               IF (WORD)=(KW_FOR) THEN                                 1305 1
703     3    1                          DO;                                          1305 1
704     3    2          CALL SKIP_BLANKS;                                            1306
705     3    2          IF STMT_CH>STMT_RIGHT THEN                                   1307
706     3    2             CALL PRINT_ERR(STMT_CH,                                   1308
                               'INVALID SYNTAX - INCOMPLETE FOR STATEMENT');         1309
707     3    2          ELSE                                                         1310
707     3    2             CALL PROCESS_FOR;                                         1311
708     3    2                     GO TO  ENDSELECT_MACRO3; END; /*                  1312 1
                      WHEN (KW_NEXT)    */                                          1312 1
710     3    1               IF (WORD)=(KW_NEXT) THEN                                1312 1
711     3    1                          DO;                                          1312 1
712     3    2          ERR_PTR=STMT_CH;                                             1313
713     3    2          CALL SKIP_BLANKS;                                            1314
714     3    2          IF STMT_CH>STMT_RIGHT THEN                                   1315
715     3    2             CALL PRINT_ERR(ERR_PTR,                                   1316
                               'INVALID SYNTAX - EXPECTING VARIABLE AFTER NEXT');    1317
716     3    2          ELSE                                                         1318
716     3    2             CALL PROCESS_NEXT;                                        1319
717     3    2                     GO TO  ENDSELECT_MACRO3; END; /*                  1320 1
                      WHEN (KW_RESTORE)    */                                       1320 1
719     3    1               IF (WORD)=(KW_RESTORE) THEN                             1320 1
720     3    1                          DO;                                          1320 1
721     3    2          CALL SKIP_BLANKS;                                            1321
722     3    2          IF STMT_CH>STMT_RIGHT THEN                                   1322
723     3    2             CALL ADD_PCODE(PC_OPCODE_RST,ZERO);                       1323
724     3    2          ELSE                                                         1324
724     3    2             CALL PRINT_ERR(STMT_CH,                                   1325
                                'INVALID SYNTAX - EXPECTING BLANKS AFTER RESTORE');  1326
725     3    2                  GO TO  ENDSELECT_MACRO3; END; /*                     1327 1
                      WHEN (KW_DIM)    */                                           1327 1
727     3    1               IF (WORD)=(KW_DIM) THEN                                 1327 1
```

```
STMT LEVEL NEST

728     3    1                            DO;                                          1327 1
729     3    2          ERR_PTR=STMT_CH;                                               1328
730     3    2          CALL SKIP_BLANKS;                                              1329
731     3    2          IF STMT_CH>STMT_RIGHT THEN                                     1330
732     3    2             CALL PRINT_ERR(ERR_PTR,                                     1331
                               'INVALID SYNTAX - EXPECTING DIM VARIABLE');            1332
733     3    2          ELSE                                                           1333
733     3    2             CALL PROCESS_DIM;                                           1334
734     3    2                      END; /*                                            1335 1
735     3    1        OTHERWISE     */  ELSE DO;                                      1335 1
                                                                                      1335 1
736     3    2          CALL PRINT_ERR(STMT_CH,                                       1336
                                  '***INVALID KEYWORD '|| WORD ||'****');             1337
737     3    2                      END; /*                                            1338 1
738     3    1        ENDSELECT     */  END; ENDSELECT_MACRO3:;                       1338 1
740     3         END PROCESS_KEYWORD;                                                 1340
                                                                                      1341
741     2         PROCESS_GOTO:PROC;                                                   1342
                  /*********************************************************************1343
                  *                                                                   *1343
                  *   EXTRACT AND VERIFY LINE NUMBER FROM THE GOTO STATEMENT.         *1343
                  *   IF THE NUMBER IS CLEAN, ADD A PC B  TO THE CODE                 *1343
                  *                                                                   *1343
                  * NESTING:COMPILE                                                   *1343
                  *********************************************************************/1343
                                                                                      1349
                                                                                      1350
742     3            CALL GET_STMT_NUM(FALSE);                                        1351
743     3            CALL SKIP_BLANKS;                                                1352
744     3            IF STMT_CH>STMT_RIGHT THEN                                       1353
745     3            DO;                                                              1354
746     3    1          I=REF_LINE_NUM;                                                1355
747     3    1          CALL ADD_PCODE(PC_OPCODE_B,I);                                 1356
748     3    1       END;                                                              1357
749     3            ELSE                                                             1358
749     3               CALL PRINT_ERR(STMT_CH,                                       1359
                            'INVALID SYNTAX - EXPECTING BLANKS AFTER LINE');          1360
                                                                                      1361
750     3         END PROCESS_GOTO;                                                    1362
                                                                                      1363
751     2         PROCESS_GOSUB:PROC;                                                  1364
                  /*********************************************************************1365
                  *                                                                   *1365
                  *   EXTRACT AND VERIFY LINE NUMBER FROM THE GOSUB STATEMENT.        *1365
                  *   IF THE NUMBER IS CLEAN, ADD A PC BAL TO THE CODE                *1365
```

```
STMT LEVEL NEST
                 *                                                                 *   1365
                 * NESTING:COMPILE                                                 *   1365
                 *******************************************************************/   1365
                                                                                       1371
                                                                                       1372
752     3           CALL GET_STMT_NUM(FALSE);                                          1373
753     3           CALL SKIP_BLANKS;                                                  1374
754     3           IF STMT_CH>STMT_RIGHT THEN                                         1375
755     3           DO;                                                                1376
756     3    1         I=REF_LINE_NUM;                                                 1377
757     3    1         CALL ADD_PCODE(PC_OPCODE_BAL,I);                                1378
758     3    1       END;                                                              1379
759     3           ELSE                                                               1380
759     3              CALL PRINT_ERR(STMT_CH,                                         1381
                           'INVALID SYNTAX - EXPECTING BLANKS AFTER LINE');            1382
                                                                                       1383
760     3         END PROCESS_GOSUB;                                                   1384
                                                                                       1385
761     2         EXTRACT_DATA:PROC;                                                   1386
                  /*******************************************************************   1387
                  *                                                                 *   1387
                  *   EXTRACT AND VERIFY NUMBERS FROM THE DATA STATEMENTS.          *   1387
                  *   IF THE NUMBER IS CLEAN, ADD IT TO THE DATA STACK              *   1387
                  *                                                                 *   1387
                  * NESTING:COMPILE                                                 *   1387
                  *******************************************************************/   1387
                                                                                       1393
762     3           DECLARE I                  FIXED BINARY ALIGNED;                   1394
763     3           DECLARE CH                 CHAR(1);                                1395
764     3           DECLARE VAL                CHAR(80) VARYING;                       1396
765     3           DECLARE HOLD_VAL           CHAR(80) VARYING;                       1397
766     3           DECLARE NUM_VAL            FLOAT BINARY;                           1398
767     3           DECLARE CONTINUE_SCAN      BIT(1) ALIGNED;                         1399
768     3           DECLARE IN_STR             BIT(1) ALIGNED;                         1400
                                                                                       1401
769     3           ON CONVERSION                                                      1402
770     3           BEGIN;                                                             1403
771     4              CONTINUE_SCAN=FALSE;                                            1404
772     4              ONCHAR='0';                                                     1405
773     4           END;                                                               1406
774     3           CONTINUE_SCAN=TRUE;                                                1407
775     3           IN_STR=FALSE;                                                      1408
776     3           VAL='';                                                            1409
777     3           DO I=STMT_CH TO STMT_RIGHT;                                        1410
778     3    1         CH=SUBSTR(STMT,I,1);                                            1411
```

```
STMT LEVEL NEST

                          /*                                                    1412 1
779     3    1           SELECT (TRUE)   */ DO;                                1412 1
                                  /*                                           1413 1
                         WHEN (IN_STR)    */                                   1413 1
780     3    2                    IF (IN_STR) THEN                             1413 1
781     3    2                                DO;                              1413 1
782     3    3             VAL=VAL||CH;                                        1414
783     3    3             IF CH=QUOTE_1 THEN                                  1415
784     3    3                IN_STR=FALSE;                                    1416
785     3    3                       GO TO  ENDSELECT_MACRO4; END; /*          1417 1
                         WHEN (CH=QUOTE_1)    */                              1417 1
787     3    2                    IF (CH=QUOTE_1) THEN                         1417 1
788     3    2                                DO;                              1417 1
789     3    3             VAL=VAL||CH;                                        1418
790     3    3             IN_STR=TRUE;                                        1419
791     3    3                       GO TO  ENDSELECT_MACRO4; END; /*          1420 1
                         WHEN (CH=' ')    */                                  1420 1
793     3    2                    IF (CH=' ') THEN                             1420 1
794     3    2                                DO;                              1420 1
795     3    3                       GO TO  ENDSELECT_MACRO4; END; /*          1421 1
                         WHEN (CH=',')    */                                  1421 1
797     3    2                    IF (CH=',') THEN                             1421 1
798     3    2                                DO;                              1421 1
799     3    3             CALL EXTRACT_DATA_ITEM;                             1422
800     3    3             CONTINUE_SCAN=TRUE;                                 1423
801     3    3             VAL='';                                             1424
802     3    3                          END; /*                                1425 1
803     3    2           OTHERWISE     */  ELSE DO;                            1425 1
                                                                               1425 1
804     3    3             VAL=VAL||CH;                                        1426
805     3    3                          END; /*                                1427 1
806     3    2           ENDSELECT     */  END; ENDSELECT_MACRO4:;             1427 1
808     3    1       END;                                                      1428
809     3            CALL EXTRACT_DATA_ITEM;                                   1429
810     3         EXTRACT_DATA_ITEM:PROC;                                      1430
811     4            DECLARE TMP_VAR              CHAR(10);                    1431
812     4            DECLARE OFFSET               FIXED BINARY ALIGNED;        1432
813     4            OFFSET=0;                                                 1433
814     4            NUM_VAL=0.0;                                              1434
815     4            HOLD_VAL=VAL;                                             1435
816     4            IF SUBSTR(VAL,1,1)=QUOTE_1 THEN                           1436
817     4            DO;                                                       1437
818     4    1          IF SUBSTR(VAL,LENGTH(VAL),1)=QUOTE_1 THEN              1438
819     4    1          DO;                                                    1439
820     4    2             TMP_VAR='STR$'||STR_CNT;                            1440
```

```
STMT LEVEL NEST

821     4    2             STR_CNT=STR_CNT+1;                                          1441
822     4    2             OFFSET=LOOKUP_SYMBOL_TABLE(TMP_VAR);                        1442
823     4    2             IF STACK_PRINT_DEBUG THEN                                   1443
824     4    2                PUT DATA(VAL,TMP_VAR,OFFSET);                            1444
825     4    2             IF LENGTH(VAL)>2 THEN                                       1445
826     4    2                STRING_VAL(OFFSET)=SUBSTR(VAL,2,LENGTH(VAL)-2);          1446
827     4    2             ELSE                                                        1447
827     4    2                STRING_VAL(OFFSET)='';                                   1448
828     4    2          END;                                                          1449
829     4    1          ELSE                                                          1450
829     4    1          DO;                                                           1451
830     4    2             CONTINUE_SCAN=FALSE;                                       1452
831     4    2          END;                                                          1453
832     4    1       END;                                                             1454
833     4            ELSE                                                             1455
833     4            DO;                                                              1456
834     4    1          IF VERIFY(V,'0123456789+-.E ') > 0 THEN                       1457
835     4    1             CONTINUE_SCAN=FALSE;                                       1458
836     4    1          ELSE                                                          1459
836     4    1             NUM_VAL=VAL;                                               1460
837     4    1       END;                                                             1461
838     4            IF CONTINUE_SCAN=FALSE THEN                                      1462
839     4                CALL PRINT_ERR(I,'ILLEGAL CONSTANT IN DATA STATEMENT '       1463
                                                 || HOLD_VAL);                       1464
840     4            ELSE                                                             1465
840     4            DO;                                                              1466
841     4    1           IF DS_MAX=HBOUND(DS_ITEM,1) THEN                             1467
842     4    1              CALL PRINT_ERR(I,'DATA STACK FULL');                      1468
843     4    1           ELSE                                                         1469
843     4    1           DO;                                                          1470
844     4    2              DS_MAX=DS_MAX+1;                                          1471
845     4    2              DS_STR(DS_MAX)=OFFSET;                                    1472
846     4    2              DS_ITEM(DS_MAX)=NUM_VAL;                                  1473
847     4    2           END;                                                         1474
848     4    1       END;                                                             1475
849     4         END EXTRACT_DATA_ITEM;                                              1476
850     3         END EXTRACT_DATA;                                                   1477
                                                                                     1478
851     2         PROCESS_READ:PROC;                                                  1479
                  /*******************************************************************
                  *                                                                  *  1480
                  *   PROCESS READ                                                   *  1480
                  *                                                                  *  1480
                  * NESTING:COMPILE                                                  *  1480
                  *******************************************************************/  1480
```

```
STMT LEVEL NEST

                                                                                                                1485
852     3           DECLARE I                   FIXED BINARY ALIGNED;                                           1486
853     3           DECLARE CH                  CHAR(1);                                                        1487
854     3           DECLARE VAR                 CHAR(10);                                                       1488
855     3           DECLARE LEFT_SIDE           CHAR(80) VARYING;                                               1489
856     3           DECLARE NO_COMMA            BIT(1) ALIGNED;                                                 1490
                                                                                                                1491
                  /*  EXTRACT THE RECEIVING FIELDS */                                                           1492
                                                                                                                1493
857     3           LEFT_SIDE='';                                                                               1494
858     3           NO_COMMA=TRUE;                                                                              1495
859     3           DO I=STMT_CH TO STMT_RIGHT;                                                                 1496
860     3    1        CH=SUBSTR(STMT,I,1);                                                                       1497
861     3    1        IF CH=',' THEN                                                                             1498
862     3    1        DO;                                                                                        1499
863     3    2           NO_COMMA=FALSE;                                                                         1500
864     3    2           CALL PROCESS_READ_VAR;                                                                  1501
865     3    2        END;                                                                                       1502
866     3    1        ELSE                                                                                       1503
866     3    1           IF CH=' ' THEN;                                                                         1504
868     3    1           ELSE                                                                                    1505
868     3    1              LEFT_SIDE=LEFT_SIDE||CH;                                                             1506
869     3    1       END;                                                                                       1507
870     3            IF LENGTH(LEFT_SIDE)>0 THEN                                                                1508
871     3                CALL PROCESS_READ_VAR;                                                                 1509
                                                                                                                1510
872     3         PROCESS_READ_VAR:PROC;                                                                        1511
                                                                                                                1512
873     4                IF LENGTH(LEFT_SIDE)>10 THEN                                                           1513
874     4                    CALL PRINT_ERR(STMT_CH,'VARIABLE TOO LONG');                                       1514
875     4                ELSE                                                                                   1515
875     4                DO;                                                                                    1516
876     4    1               CALL BALANCE_STMT(LEFT_SIDE);                                                       1517
877     4    1               IF TERMINATE_SCAN THEN RETURN;                                                      1518
879     4    1               LEFT_SIDE='('||LEFT_SIDE||')';                                                      1519
880     4    1               CALL PARSE_EXP(LEFT_SIDE,EXP_CALC);                                                 1520
881     4    1               IF PC_MAX > 0 THEN  /*  ERROR IF A TMP IS FOUND */                                  1521
882     4    1               DO;                                                                                 1522
883     4    2               IF PC_OPCODE(PC_MAX)=PC_OPCODE_STA THEN                                             1523
884     4    2                  IF SUBSTR(SYMBOL(PC_OBJECT(PC_MAX)),1,3)='TMP' THEN                              1524
885     4    2                     CALL PRINT_ERR(STMT_CH,'EXPRESSION NOT ALLOWED');                             1525
886     4    2               IF PC_OPCODE(PC_MAX)=PC_OPCODE_LDA THEN                                             1526
887     4    2                  IF SUBSTR(SYMBOL(PC_OBJECT(PC_MAX)),1,3)='TMP' THEN                              1527
888     4    2                     CALL PRINT_ERR(STMT_CH,'EXPRESSION NOT ALLOWED');                             1528
889     4    2                  ELSE                                                                             1529
```

```
STMT LEVEL NEST

889     4    2                    PC_OPCODE(PC_MAX)=PC_OPCODE_RDV;                    1530
890     4    2                END;                                                    1531
891     4    1            END;                                                        1532
892     4                 LEFT_SIDE='';                                               1533
                                                                                      1534
893     4          END PROCESS_READ_VAR;                                              1535
894     3          END PROCESS_READ;                                                  1536
                                                                                      1537
895     2          PROCESS_PRINT:PROC;                                                1538
                                                                                      1539
                   /*********************************************************************  1540
                   *                                                                   *  1540
                   *   PROCESS PRINT                                                   *  1540
                   *                                                                   *  1540
                   *   PARSE THE STATEMENT INTO OBJECTS - NUMERIC EXPRESSION OR        *  1540
                   *   STRING LITTERALS.                                               *  1540
                   *   ALSO DECIDES IF A LINE FEED SHOULD BE ISSUED OR NOT BASED ON    *  1540
                   *   ON DANGLING COMMA                                               *  1540
                   *                                                                   *  1540
                   * NESTING:COMPILE                                                   *  1540
                   *********************************************************************/  1540
                                                                                      1550
896     3             DECLARE (I,LAST_NB)         FIXED BINARY ALIGNED;               1551
897     3             DECLARE CH                  CHAR(1);                            1552
898     3             DECLARE VAR                 CHAR(10);                           1553
899     3             DECLARE LEFT_SIDE           CHAR(80) VARYING;                   1554
900     3             DECLARE NO_COMMA            BIT(1) ALIGNED;                     1555
901     3             DECLARE IN_STR              BIT(1) ALIGNED;                     1556
                                                                                      1557
                    /*  EXTRACT THE PRINT OBJECT */                                   1558
                                                                                      1559
902     3             LEFT_SIDE='';                                                   1560
903     3             LAST_NB=0;                                                      1561
904     3             NO_COMMA=TRUE;                                                  1562
905     3             DANGLE=FALSE;                                                   1563
906     3             IN_STR=FALSE;                                                   1564
                                                                                      1565
907     3             IF STMT_CH>STMT_RIGHT THEN     /* PRINT A LINE FEED FOR */      1566
908     3             DO;                            /* KEYWORD PRINT ONLY */         1567
909     3    1           I=1;                                                         1568
910     3    1           CALL ADD_PCODE(PC_OPCODE_PCT,PCT_LFEED);                     1569
911     3    1           RETURN;                                                      1570
912     3    1         END;                                                           1571
                                                                                      1572
913     3             I=STMT_CH;                                                      1573
```

```
STMT LEVEL NEST

914     3           DO WHILE(I <= STMT_RIGHT);                                  1574
915     3    1        CH=SUBSTR(STMT,I,1);                                      1575
916     3    1        IF IN_STR THEN                                            1576
917     3    1        DO;                                                       1577
918     3    2           IF CH=QUOTE_1 THEN                                     1578
919     3    2              IN_STR=FALSE;                                       1579
920     3    2           LEFT_SIDE=LEFT_SIDE||CH;                               1580
921     3    2        END;                                                      1581
922     3    1        ELSE                                                      1582
922     3    1        DO;                                                       1583
923     3    2           IF CH=QUOTE_1 THEN                                     1584
924     3    2           DO;                                                    1585
925     3    3              IN_STR=TRUE;                                        1586
926     3    3              LEFT_SIDE=LEFT_SIDE||CH;                            1587
927     3    3           END;                                                   1588
928     3    2        END;                                                      1589
                                                                               1590
929     3    1        IF IN_STR THEN;                                           1591
931     3    1        ELSE                                                      1592
931     3    1        DO;                                                       1593
932     3    2           IF CH=' ' THEN                                         1594
933     3    2           /* PUT SKIP DATA(LAST_NB,I,LEFT_SIDE) PRINT USING ?*/; 1595
934     3    2           ELSE LAST_NB=I;                                        1596
935     3    2           IF CH=',' | CH=';' THEN                                1597
936     3    2           DO;                                                    1598
937     3    3              NO_COMMA=FALSE;                                     1599
938     3    3              IF SUBSTR(LEFT_SIDE,1,1)=QUOTE_1 THEN               1600
939     3    3                  CALL PROCESS_PRINT_STR;                         1601
940     3    3              ELSE                                                1602
940     3    3                  CALL PROCESS_PRINT_VAR;                         1603
941     3    3              IF CH=',' THEN                                      1604
942     3    3                  CALL ADD_PCODE(PC_OPCODE_PCT,PCT_TAB);          1605
943     3    3              ELSE                                                1606
943     3    3                  CALL ADD_PCODE(PC_OPCODE_PCT,PCT_NOTAB);        1607
944     3    3           END;                                                   1608
945     3    2           ELSE                                                   1609
945     3    2              IF CH=' ' | CH=QUOTE_1 THEN;                        1610
947     3    2              ELSE                                                1611
947     3    2                 LEFT_SIDE=LEFT_SIDE||CH;                         1612
948     3    2        END;                                                      1613
949     3    1       I=I+1;                                                    1614
950     3    1       END;                                                      1615
                                                                               1616
951     3           IF LENGTH(LEFT_SIDE)>0 THEN                                1617
952     3               IF SUBSTR(LEFT_SIDE,1,1)=QUOTE_1 THEN                  1618
```

```
STMT LEVEL NEST

953     3                   CALL PROCESS_PRINT_STR;                                  1619
954     3                ELSE                                                        1620
954     3                   CALL PROCESS_PRINT_VAR;                                  1621
                                                                                     1622
955     3             IF LAST_NB > 0 THEN                                            1623
956     3             DO;                                                            1624
957     3    1           IF SUBSTR(STMT,LAST_NB,1)=',' |                            1625
958     3    1              SUBSTR(STMT,LAST_NB,1)=';' THEN;                        1626
959     3    1           ELSE                                                       1627
959     3    1              CALL ADD_PCODE(PC_OPCODE_PCT,PCT_LFEED);                1628
960     3    1        END;                                                          1629
961     3             ELSE                                                          1630
961     3                   CALL PRINT_ERR(STMT_CH,'INVALID PRINT SYNTAX');         1631
                                                                                     1632
962     3          PROCESS_PRINT_VAR:PROC;                                          1633
                                                                                     1634
963     4             CALL BALANCE_STMT(LEFT_SIDE);                                 1635
964     4             IF TERMINATE_SCAN THEN                                        1636
965     4                 RETURN;                                                   1637
                                                                                     1638
966     4             LEFT_SIDE='('||LEFT_SIDE||')';                                1639
967     4             CALL PARSE_EXP(LEFT_SIDE,EXP_CALC);                           1640
968     4             IF PC_MAX > 0 THEN  /*  ERROR IF A TMP IS FOUND */            1641
969     4             DO;                                                           1642
970     4    1           IF PC_OPCODE(PC_MAX)=PC_OPCODE_LDA THEN                    1643
971     4    1           DO;                                                        1644
972     4    2              IF PC_FORMAT(PC_OPCODE(PC_MAX-2))=0 THEN                1645
973     4    2              DO;                                                     1646
974     4    3                 IF SYMBOL(PC_OBJECT(PC_MAX-2))='TAB       ' THEN     1647
975     4    3                      PC_OPCODE(PC_MAX)=PC_OPCODE_PTB;               1648
976     4    3                 ELSE                                                1649
976     4    3                      PC_OPCODE(PC_MAX)=PC_OPCODE_PRV;               1650
977     4    3              END;                                                    1651
978     4    2              ELSE                                                    1652
978     4    2                   PC_OPCODE(PC_MAX)=PC_OPCODE_PRV;                   1653
979     4    2           END;                                                       1654
980     4    1           IF PC_OPCODE(PC_MAX)=PC_OPCODE_STA THEN                    1655
981     4    1           DO;                                                        1656
982     4    2              IF PC_FORMAT(PC_OBJECT(PC_MAX-2))=0 THEN                1657
983     4    2              DO;                                                     1658
984     4    3                 IF SYMBOL(PC_OBJECT(PC_MAX-2))='TAB       ' THEN;    1659
986     4    3                 ELSE                                                1660
986     4    3                    CALL ADD_PCODE(PC_OPCODE_PRV,PC_OBJECT(PC_MAX)); 1661
987     4    3              END;                                                    1662
988     4    2              ELSE                                                    1663
```

```
STMT LEVEL NEST

 988     4    2               CALL ADD_PCODE(PC_OPCODE_PRV,PC_OBJECT(PC_MAX));          1664
 989     4    2          END;                                                           1665
 990     4    1       END;                                                           1666
                                                                                     1667
 991     4            LEFT_SIDE='';                                                  1668
                                                                                     1669
 992     4         END PROCESS_PRINT_VAR;                                            1670
                                                                                     1671
 993     3         PROCESS_PRINT_STR:PROC;                                           1672
                                                                                     1673
 994     4            DECLARE  (I,TICS)             FIXED BINARY ALIGNED;           1674
 995     4            TICS=0;                                                        1675
 996     4            DO I = 1 TO LENGTH(LEFT_SIDE);                                 1676
 997     4    1           IF SUBSTR(LEFT_SIDE,I,1)=QUOTE_1 THEN                      1677
 998     4    1               TICS=TICS+1;                                           1678
 999     4    1       END;                                                           1679
1000     4            IF MOD(TICS,2)=1 THEN                                          1680
1001     4            DO;                                                            1681
1002     4    1          CALL PRINT_ERR(STMT_CH,'UNBALANCED STRING');                1682
1003     4    1          RETURN;                                                     1683
1004     4    1       END;                                                           1684
                                                                                     1685
1005     4            IF SS_MAX>=HBOUND(STRING_VAL,1) THEN                           1686
1006     4            DO;                                                            1687
1007     4    1          CALL PRINT_ERR(STMT_CH,'STRING CONSTANT TABLE FULL');       1688
1008     4    1          RETURN;                                                     1689
1009     4    1       END;                                                           1690
                                  /*  STRIP OFF THE QUOTE MARKS AND REDUCE         1691
                                      DOUBLE QUOTES TO 1 QUOTE BEFORE SAVING */    1691
                                                                                     1692
                                                                                     1693
1010     4            LEFT_SIDE=SUBSTR(LEFT_SIDE,2,LENGTH(LEFT_SIDE)-2);             1694
1011     4            I = 1;                                                         1695
1012     4            DO WHILE(I<LENGTH(LEFT_SIDE));                                 1696
1013     4    1          IF SUBSTR(LEFT_SIDE,I,2)=QUOTE_2 THEN                       1697
1014     4    1             LEFT_SIDE=SUBSTR(LEFT_SIDE,1,I)||SUBSTR(LEFT_SIDE,I+2);  1698
1015     4    1          I=I+1;                                                      1699
1016     4    1       END;                                                           1700
1017     4            SS_CUR,SS_MAX=SS_MAX+1;                                        1701
1018     4            SYMBOL(SS_CUR)='_PRS';                                         1702
1019     4            STRING_VAL(SS_CUR)=LEFT_SIDE;                                  1703
1020     4            SYM_TYPE(SS_CUR)=SS_STRCON;                                    1704
1021     4            CALL ADD_PCODE(PC_OPCODE_PRS,SS_CUR);                          1705
1022     4            LEFT_SIDE='';                                                  1706
                                                                                     1707
```

```
 STMT LEVEL NEST

1023     4         END PROCESS_PRINT_STR;                                                  1708
                                                                                           1709
1024     3         END PROCESS_PRINT;                                                      1710
                                                                                           1711
1025     2         PROCESS_IF:PROC;                                                        1712
                   /*********************************************************************  1713
                   *                                                                   *  1713
                   *   PROCESS IF                                                     *  1713
                   *                                                                   *  1713
                   * NESTING:COMPILE                                                   *  1713
                   *********************************************************************/ 1713
                                                                                           1718
1026     3            DECLARE I                   FIXED BINARY ALIGNED;                  1719
1027     3            DECLARE CH                  CHAR(1);                               1720
1028     3            DECLARE (LEFT_SIDE,RIGHT_SIDE)                                     1721
                                                  CHAR(80) VARYING;                      1722
1029     3            DECLARE NO_OPER             BIT(1) ALIGNED;                        1723
1030     3            DECLARE OPER                CHAR(2);                               1724
1031     3            DECLARE THEN_WORD           CHAR(4);                               1725
                                                                                           1726
                     /*  EXTRACT THE LEFT FIELD */                                      1727
                                                                                           1728
1032     3            LEFT_SIDE='';                                                      1729
1033     3            NO_OPER=TRUE;                                                      1730
1034     3            OPER='..';                                                         1731
1035     3            DO I=STMT_CH TO STMT_RIGHT WHILE(NO_OPER);                         1732
1036     3    1         CH=SUBSTR(STMT,I,1);                                            1733
1037     3    1         IF CH='=' |                                                     1734
                          CH='<' |                                                     1735
                          CH='>' THEN                                                  1736
1038     3    1         DO;                                                             1737
1039     3    2            NO_OPER=FALSE;                                               1738
1040     3    2            OPER=CH;                                                     1739
1041     3    2            RIGHT_SIDE=SUBSTR(STMT,I+1);                                 1740
1042     3    2         END;                                                            1741
1043     3    1         ELSE                                                            1742
1043     3    1            IF CH=' ' THEN;                                              1743
1045     3    1            ELSE                                                         1744
1045     3    1               LEFT_SIDE=LEFT_SIDE||CH;                                  1745
1046     3    1       END;                                                              1746
                                                                                           1747
1047     3            STMT_CH=I;                                                         1748
1048     3            CH=SUBSTR(STMT,I,1);                                               1749
1049     3            IF (OPER='= ' & CH='>') | (OPER='> ' & CH='=') THEN                1750
1050     3            DO;                                                                1751
```

```
 STMT LEVEL NEST

1051     3    1          OPER='>=';                                                    1752
1052     3    1          STMT_CH=I+1;                                                  1753
1053     3    1       END;                                                             1754
1054     3            ELSE                                                             1755
1054     3            IF (OPER='= ' & CH='<') | (OPER='< ' & CH='=') THEN              1756
1055     3            DO;                                                              1757
1056     3    1          OPER='<=';                                                    1758
1057     3    1          STMT_CH=I+1;                                                  1759
1058     3    1       END;                                                             1760
1059     3            ELSE                                                             1761
1059     3            IF (OPER='< ' & CH='>') THEN                                     1762
1060     3            DO;                                                              1763
1061     3    1          OPER='<>';                                                    1764
1062     3    1          STMT_CH=I+1;                                                  1765
1063     3    1       END;                                                             1766
1064     3            ELSE                                                             1767
1064     3            IF OPER='= ' | OPER='< ' | OPER='> ' THEN;                       1768
1066     3            ELSE                                                             1769
1066     3            DO;                                                              1770
1067     3    1          CALL PRINT_ERR(STMT_CH,'NO COMPARISON OPERATOR FOUND');       1771
1068     3    1          RETURN;                                                       1772
1069     3    1       END;                                                             1773
                                                                                      1774
                     /*  EXTRACT THE RIGHT FIELD */                                   1775
                                                                                      1776
1070     3            RIGHT_SIDE='';                                                   1777
1071     3            NO_OPER=TRUE;                                                    1778
1072     3            DO I=STMT_CH TO STMT_RIGHT-3 WHILE(NO_OPER);                     1779
1073     3    1         CH=SUBSTR(STMT,I,1);                                          1780
1074     3    1         THEN_WORD=SUBSTR(STMT,I,4);                                   1781
1075     3    1         IF THEN_WORD='THEN' THEN                                      1782
1076     3    1         DO;                                                           1783
1077     3    2            NO_OPER=FALSE;                                             1784
1078     3    2         END;                                                          1785
1079     3    1         ELSE                                                          1786
1079     3    1            IF CH=' ' THEN;                                            1787
1081     3    1            ELSE                                                       1788
1081     3    1               RIGHT_SIDE=RIGHT_SIDE||CH;                              1789
1082     3    1       END;                                                             1790
                                                                                      1791
1083     3            IF NO_OPER=TRUE THEN                                             1792
1084     3            DO;                                                              1793
1085     3    1          CALL PRINT_ERR(STMT_CH,'THEN NOT FOUND');                     1794
1086     3    1          RETURN;                                                       1795
1087     3    1       END;                                                             1796
```

```
 STMT LEVEL NEST

                                                                                                                    1797
1088     3            STMT_CH=I+4;                                                                                  1798
                                                                                                                    1799
1089     3            CALL SKIP_BLANKS;                                                                             1800
1090     3            IF STMT_CH>STMT_RIGHT THEN                                                                    1801
1091     3              CALL PRINT_ERR(STMT_CH,                                                                     1802
                                    'INVALID SYNTAX - EXPECTING LINE NUMBER');                                      1803
1092     3            ELSE                                                                                          1804
1092     3            DO;                                                                                           1805
1093     3    1          CALL GET_STMT_NUM(FALSE);                                                                 1806
                                                                                                                    1807
1094     3    1          CALL SKIP_BLANKS;                                                                         1808
                                                                                                                    1809
1095     3    1          CALL BALANCE_STMT(LEFT_SIDE);                                                             1810
1096     3    1          IF TERMINATE_SCAN THEN                                                                    1811
1097     3    1             RETURN;                                                                                1812
1098     3    1          LEFT_SIDE='('||LEFT_SIDE||')';                                                            1813
1099     3    1          CALL PARSE_EXP(LEFT_SIDE,EXP_CALC);                                                       1814
1100     3    1          IF PC_MAX > 0 THEN  /*  ERROR IF A TMP IS FOUND */                                        1815
1101     3    1          DO;                                                                                       1816
1102     3    2             IF PC_OPCODE(PC_MAX)=PC_OPCODE_LDA THEN                                                1817
1103     3    2                PC_OPCODE(PC_MAX)=PC_OPCODE_LCA;                                                    1818
1104     3    2          END;                                                                                      1819
1105     3    1          CALL BALANCE_STMT(RIGHT_SIDE);                                                            1820
1106     3    1          IF TERMINATE_SCAN THEN RETURN;                                                            1821
1108     3    1          LEFT_SIDE='('||RIGHT_SIDE||')';                                                           1822
1109     3    1          CALL PARSE_EXP(LEFT_SIDE,EXP_CALC);                                                       1823
1110     3    1          IF PC_MAX > 0 THEN  /*  ERROR IF A TMP IS FOUND */                                        1824
1111     3    1          DO;                                                                                       1825
1112     3    2             IF PC_OPCODE(PC_MAX)=PC_OPCODE_LDA THEN                                                1826
1113     3    2                PC_OPCODE(PC_MAX)=PC_OPCODE_LCB;                                                    1827
1114     3    2          END;                                                                                      1828
1115     3    1          I=REF_LINE_NUM;                                                                           1829
                                 /*                                                                             1830 1
1116     3    1           SELECT (OPER)   */ DO;                                                                   1830 1
                                                /*                                                              1831 1
                            WHEN ('= ')    */                                                                   1831 1
1117     3    2                    IF (OPER)=('= ') THEN                                                         1831 1
1118     3    2                                 DO;                                                              1831 1
1119     3    3                 CALL ADD_PCODE(PC_OPCODE_BEQ,I);                                                  1832
1120     3    3                              GO TO  ENDSELECT_MACRO5; END; /*                                     1833 1
                            WHEN ('<>')    */                                                                   1833 1
1122     3    2                    IF (OPER)=('<>') THEN                                                         1833 1
1123     3    2                                 DO;                                                              1833 1
1124     3    3                 CALL ADD_PCODE(PC_OPCODE_BNE,I);                                                  1834
```

```
 STMT LEVEL NEST

1125     3    3                          GO TO  ENDSELECT_MACRO5; END; /*                1835 1
                          WHEN ('< ')    */                                               1835 1
1127     3    2                          IF (OPER)=('< ') THEN                           1835 1
1128     3    2                                DO;                                       1835 1
1129     3    3                 CALL ADD_PCODE(PC_OPCODE_BLT,I);                         1836
1130     3    3                          GO TO  ENDSELECT_MACRO5; END; /*                1837 1
                          WHEN ('> ')    */                                               1837 1
1132     3    2                          IF (OPER)=('> ') THEN                           1837 1
1133     3    2                                DO;                                       1837 1
1134     3    3                 CALL ADD_PCODE(PC_OPCODE_BGT,I);                         1838
1135     3    3                          GO TO  ENDSELECT_MACRO5; END; /*                1839 1
                          WHEN ('<=')    */                                               1839 1
1137     3    2                          IF (OPER)=('<=') THEN                           1839 1
1138     3    2                                DO;                                       1839 1
1139     3    3                 CALL ADD_PCODE(PC_OPCODE_BLE,I);                         1840
1140     3    3                          GO TO  ENDSELECT_MACRO5; END; /*                1841 1
                          WHEN ('>=')    */                                               1841 1
1142     3    2                          IF (OPER)=('>=') THEN                           1841 1
1143     3    2                                DO;                                       1841 1
1144     3    3                 CALL ADD_PCODE(PC_OPCODE_BGE,I);                         1842
1145     3    3                           END; /*                                        1843 1
1146     3    2               OTHERWISE     */  ELSE DO;                                1843 1
                                                                                         1843 1
1147     3    3                 TERMINATE_SCAN=TRUE;                                     1844
1148     3    3                 PUT SKIP EDIT('**** INTERNAL COMPILER ERROR IF-01')      1845
                                    (A);                                                 1846
1149     3    3                           END; /*                                        1847 1
1150     3    2               ENDSELECT     */  END; ENDSELECT_MACRO5:;                 1847 1
1152     3    1            END;                                                         1849
                                                                                         1850
1153     3         END PROCESS_IF;                                                       1851
                                                                                         1852
1154     2         PROCESS_FOR:PROC;                                                     1853
                   /*********************************************************************  1854
                   *                                                                   *  1854
                   *   PROCESS FOR                                                     *  1854
                   *                                                                   *  1854
                   * NESTING:COMPILE                                                   *  1854
                   *********************************************************************/ 1854
                                                                                         1859
1155     3            DECLARE I                   FIXED BINARY ALIGNED;                1860
1156     3            DECLARE OFFSET              FIXED BINARY ALIGNED;                1861
1157     3            DECLARE CH                  CHAR(1);                             1862
1158     3            DECLARE (LEFT_SIDE,START_VAL,TO_VAL,STEP_VAL)                    1863
                                                  CHAR(80) VARYING;                    1864
```

```
 STMT LEVEL NEST

1159     3            DECLARE CTL_VAR             CHAR(10);                          1865
1160     3            DECLARE NO_OPER             BIT(1) ALIGNED;                    1866
1161     3            DECLARE STEP_WORD           CHAR(4);                           1867
1162     3            DECLARE TO_WORD             CHAR(2);                           1868
                                                                                     1869
                    /*  EXTRACT THE CONTROL VARIABLE */                              1870
                                                                                     1871
1163     3            LEFT_SIDE,START_VAL,TO_VAL,STEP_VAL='';                        1872
1164     3            NO_OPER=TRUE;                                                  1873
                                                                                     1874
1165     3            DO I=STMT_CH TO STMT_RIGHT WHILE(NO_OPER);                     1875
1166     3    1         CH=SUBSTR(STMT,I,1);                                         1876
1167     3    1         IF CH='=' THEN                                               1877
1168     3    1         DO;                                                          1878
1169     3    2            NO_OPER=FALSE;                                            1879
1170     3    2         END;                                                         1880
1171     3    1         ELSE                                                         1881
1171     3    1            IF CH=' ' THEN;                                           1882
1173     3    1            ELSE                                                      1883
1173     3    1               LEFT_SIDE=LEFT_SIDE||CH;                               1884
1174     3    1       END;                                                          1885
                                                                                     1886
1175     3            STMT_CH=I;                                                     1887
                                                                                     1888
                    /*  EXTRACT THE STARTING VALUE */                               1889
                                                                                     1890
1176     3            NO_OPER=TRUE;                                                  1891
1177     3            DO I=STMT_CH TO STMT_RIGHT-1 WHILE(NO_OPER);                   1892
1178     3    1         CH=SUBSTR(STMT,I,1);                                         1893
1179     3    1         TO_WORD=SUBSTR(STMT,I,2);                                    1894
1180     3    1         IF TO_WORD='TO' THEN                                         1895
1181     3    1         DO;                                                          1896
1182     3    2            NO_OPER=FALSE;                                            1897
1183     3    2         END;                                                         1898
1184     3    1         ELSE                                                         1899
1184     3    1            IF CH=' ' THEN;                                           1900
1186     3    1            ELSE                                                      1901
1186     3    1               START_VAL=START_VAL||CH;                               1902
1187     3    1       END;                                                          1903
                                                                                     1904
1188     3            IF NO_OPER=TRUE THEN                                           1905
1189     3            DO;                                                            1906
1190     3    1          CALL PRINT_ERR(STMT_CH,'TO NOT FOUND');                     1907
1191     3    1          RETURN;                                                     1908
1192     3    1       END;                                                          1909
```

```
 STMT LEVEL NEST

                                                                                                                    1910
1193     3            STMT_CH=I+2;                                                                                  1911
                                                                                                                    1912
                     /*  EXTRACT THE TO VALUE */                                                                    1913
                                                                                                                    1914
1194     3            NO_OPER=TRUE;                                                                                 1915
1195     3            DO I=STMT_CH TO STMT_RIGHT-3 WHILE(NO_OPER);                                                  1916
1196     3    1         CH=SUBSTR(STMT,I,1);                                                                        1917
1197     3    1         STEP_WORD=SUBSTR(STMT,I,4);                                                                 1918
1198     3    1         IF STEP_WORD='STEP' THEN                                                                    1919
1199     3    1         DO;                                                                                         1920
1200     3    2           NO_OPER=FALSE;                                                                            1921
1201     3    2         END;                                                                                        1922
1202     3    1         ELSE                                                                                        1923
1202     3    1           IF CH=' ' THEN;                                                                           1924
1204     3    1           ELSE                                                                                      1925
1204     3    1             TO_VAL=TO_VAL||CH;                                                                      1926
1205     3    1       END;                                                                                          1927
                                                                                                                    1928
1206     3            IF NO_OPER=FALSE THEN                                                                         1929
1207     3            DO;                                                                                           1930
1208     3    1         NO_OPER=TRUE;                                                                               1931
1209     3    1         STMT_CH=I+4;                                                                                1932
1210     3    1         DO I=STMT_CH TO STMT_RIGHT WHILE(NO_OPER);                                                  1933
1211     3    2           CH=SUBSTR(STMT,I,1);                                                                      1934
1212     3    2           IF CH=' ' THEN;                                                                           1935
1214     3    2           ELSE                                                                                      1936
1214     3    2             STEP_VAL=STEP_VAL||CH;                                                                  1937
1215     3    2         END;                                                                                        1938
1216     3    1       END;                                                                                          1939
1217     3            ELSE                                                                                          1940
1217     3              STEP_VAL='1';                                                                               1941
                                                                                                                    1942
1218     3            IF STMT_CH>STMT_RIGHT THEN                                                                    1943
1219     3            DO;                                                                                           1944
1220     3    1         CALL PRINT_ERR(STMT_CH,                                                                     1945
                                      'INVALID SYNTAX - EXPECTING BLANKS');                                         1946
1221     3    1         RETURN;                                                                                     1947
1222     3    1       END;                                                                                          1948
                                                                                                                    1949
1223     3            CTL_VAR=LEFT_SIDE;                                                                            1950
1224     3            OFFSET=LOOKUP_SYMBOL_TABLE(CTL_VAR);                                                          1951
1225     3            IF SYM_TYPE(OFFSET)=SS_VAR THEN                                                               1952
1226     3              CALL ADD_PCODE(PC_OPCODE_FSU,OFFSET);                                                       1953
1227     3            ELSE                                                                                          1954
```

```
 STMT LEVEL NEST

1227     3            DO;                                                          1955
1228     3    1          CALL PRINT_ERR(STMT_CH,'SIMPLE VARIABLE EXPECTED NOT '||  1956
                                    CTL_VAR);                                     1957
1229     3    1          RETURN;                                                  1958
1230     3    1       END;                                                        1959
                                                                                  1960
1231     3            CALL BALANCE_STMT(START_VAL);                              1961
1232     3            IF TERMINATE_SCAN THEN RETURN;                             1962
1234     3            START_VAL='('||START_VAL||')';                             1963
1235     3            CALL PARSE_EXP(START_VAL,EXP_CALC);                        1964
                                                                                  1965
1236     3            IF PC_OPCODE(PC_MAX)=PC_OPCODE_LDA THEN                    1966
1237     3               PC_OPCODE(PC_MAX)=PC_OPCODE_FIX;                        1967
1238     3            ELSE                                                       1968
1238     3               CALL ADD_PCODE(PC_OPCODE_FIX,OFFSET);                   1969
                                                                                  1970
1239     3            CALL BALANCE_STMT(TO_VAL);                                 1971
1240     3            IF TERMINATE_SCAN THEN RETURN;                             1972
1242     3            TO_VAL='('||TO_VAL||')';                                   1973
1243     3            CALL PARSE_EXP(TO_VAL,EXP_CALC);                           1974
                                                                                  1975
1244     3            IF PC_OPCODE(PC_MAX)=PC_OPCODE_LDA THEN                    1976
1245     3               PC_OPCODE(PC_MAX)=PC_OPCODE_FUL;                        1977
1246     3            ELSE                                                       1978
1246     3               CALL ADD_PCODE(PC_OPCODE_FUL,OFFSET);                   1979
                                                                                  1980
1247     3            CALL BALANCE_STMT(STEP_VAL);                               1981
1248     3            IF TERMINATE_SCAN THEN RETURN;                             1982
1250     3            STEP_VAL='('||STEP_VAL||')';                               1983
1251     3            CALL PARSE_EXP(STEP_VAL,EXP_CALC);                         1984
                                                                                  1985
1252     3            IF PC_OPCODE(PC_MAX)=PC_OPCODE_LDA THEN                    1986
1253     3               PC_OPCODE(PC_MAX)=PC_OPCODE_FST;                        1987
1254     3            ELSE                                                       1988
1254     3               CALL ADD_PCODE(PC_OPCODE_FST,OFFSET);                   1989
                                                                                  1990
1255     3         END PROCESS_FOR;                                              1991
                                                                                  1992
1256     2         PROCESS_NEXT:PROC;                                            1993
                   /*********************************************************** 1994
                   *                                                         * 1994
                   *   PROCESS NEXT                                          * 1994
                   *                                                         * 1994
                   * NESTING:COMPILE                                         * 1994
                   ***********************************************************/ 1994
```

```
 STMT LEVEL NEST

                                                                                                                1999
1257     3            DECLARE I                    FIXED BINARY ALIGNED;                                      2000
1258     3            DECLARE ERR_PTR              FIXED BINARY ALIGNED;                                      2001
1259     3            DECLARE CH                   CHAR(1);                                                   2002
1260     3            DECLARE (LEFT_SIDE,START_VAL,TO_VAL,STEP_VAL)                                           2003
                                                   CHAR(80) VARYING;                                          2004
1261     3            DECLARE NO_OPER              BIT(1) ALIGNED;                                            2005
1262     3            DECLARE OFFSET               FIXED BINARY ALIGNED;                                      2006
1263     3            DECLARE CTL_VAR              CHAR(10);                                                  2007
                                                                                                               2008
                     /*  EXTRACT THE CONTROL VARIABLE */                                                       2009
                                                                                                               2010
1264     3            LEFT_SIDE='';                                                                           2011
1265     3            NO_OPER=TRUE;                                                                           2012
                                                                                                               2013
1266     3            DO I=STMT_CH TO STMT_RIGHT WHILE(NO_OPER);                                              2014
1267     3    1         CH=SUBSTR(STMT,I,1);                                                                   2015
1268     3    1         IF CH=' ' THEN                                                                         2016
1269     3    1            NO_OPER=FALSE;                                                                      2017
1270     3    1         ELSE                                                                                   2018
1270     3    1            LEFT_SIDE=LEFT_SIDE||CH;                                                            2019
1271     3    1       END;                                                                                     2020
                                                                                                               2021
1272     3            ERR_PTR=STMT_CH;                                                                        2022
1273     3            STMT_CH=I;                                                                              2023
                                                                                                               2024
1274     3            CALL SKIP_BLANKS;                                                                       2025
1275     3            IF STMT_CH>STMT_RIGHT THEN;                                                             2026
1277     3            ELSE                                                                                    2027
1277     3            DO;                                                                                     2028
1278     3    1         CALL PRINT_ERR(ERR_PTR,                                                                2029
                                  'INVALID SYNTAX - EXPECTING BLANKS');                                        2030
1279     3    1         RETURN;                                                                                2031
1280     3    1       END;                                                                                     2032
1281     3            CTL_VAR=LEFT_SIDE;                                                                      2033
1282     3            OFFSET=LOOKUP_SYMBOL_TABLE(CTL_VAR);                                                    2034
1283     3            IF SYM_TYPE(OFFSET)=SS_VAR THEN                                                         2035
1284     3                CALL ADD_PCODE(PC_OPCODE_FNX,OFFSET);                                               2036
1285     3            ELSE                                                                                    2037
1285     3                CALL PRINT_ERR(ERR_PTR,'SIMPLE VARIABLE EXPECTED');                                 2038
                                                                                                               2039
1286     3          END PROCESS_NEXT;                                                                         2040
                                                                                                               2041
1287     2          PROCESS_LET:PROC;                                                                         2042
                   /*****************************************************************                       2043
```

```
STMT LEVEL NEST

                *                                                               *   2043
                *   PROCESS LET                                                 *   2043
                *                                                               *   2043
                * NESTING:COMPILE                                               *   2043
                *****************************************************************/   2043
                                                                                      2048
1288     3        DECLARE I                  FIXED BINARY ALIGNED;                  2049
1289     3        DECLARE CH                 CHAR(1);                               2050
1290     3        DECLARE (LEFT_SIDE,RIGHT_SIDE)                                    2051
                                            CHAR(80) VARYING;                      2052
1291     3        DECLARE NO_EQUAL           BIT(1) ALIGNED;                        2053
                                                                                      2054
1292     3        CH=SUBSTR(STMT,STMT_CH,1);                                        2055
1293     3        IF CH<'A' | CH>'Z' THEN                                           2056
1294     3           CALL PRINT_ERR(STMT_CH,'EXPECTING VARIABLE');                  2057
1295     3        IF TERMINATE_SCAN THEN RETURN;                                    2058
                                                                                      2059
                 /*  EXTRACT THE RECEIVING FIELD */                                2060
                                                                                      2061
1297     3        LEFT_SIDE='';                                                     2062
1298     3        NO_EQUAL=TRUE;                                                    2063
1299     3        DO I=STMT_CH TO STMT_RIGHT WHILE(NO_EQUAL);                       2064
1300     3    1     CH=SUBSTR(STMT,I,1);                                           2065
1301     3    1     IF CH='=' THEN                                                 2066
1302     3    1     DO;                                                            2067
1303     3    2        NO_EQUAL=FALSE;                                             2068
1304     3    2        RIGHT_SIDE=SUBSTR(STMT,I+1);                                2069
1305     3    2     END;                                                           2070
1306     3    1     ELSE                                                           2071
1306     3    1        IF CH=' ' THEN;                                             2072
1308     3    1        ELSE                                                        2073
1308     3    1           LEFT_SIDE=LEFT_SIDE||CH;                                 2074
1309     3    1     END;                                                           2075
                                                                                      2076
1310     3        CALL BALANCE_STMT(LEFT_SIDE);                                     2077
1311     3        IF TERMINATE_SCAN THEN RETURN;                                    2078
1313     3        CALL BALANCE_STMT(RIGHT_SIDE);                                    2079
1314     3        IF TERMINATE_SCAN THEN RETURN;                                    2080
                                                                                      2081
1316     3        IF SUBSTR(RIGHT_SIDE,1,1)='(' THEN;                               2082
1318     3        ELSE RIGHT_SIDE='('||RIGHT_SIDE||')';                             2083
                                                                                      2084
1319     3        CALL PARSE_EXP(RIGHT_SIDE,EXP_CALC);                              2085
                                                                                      2086
1320     3        IF PC_MAX > 0 THEN       /*  CHANGE A STA TMPXX TO STA RESULT */  2087
```

```
 STMT LEVEL NEST

1321     3           DO;                                                                2088
1322     3    1         IF PC_OPCODE(PC_MAX)=PC_OPCODE_STA THEN                         2089
1323     3    1             IF SUBSTR(SYMBOL(PC_OBJECT(PC_MAX)),1,3)='TMP' THEN         2090
1324     3    1                PC_OBJECT(PC_MAX)=1 ;                                    2091
1325     3    1       END;                                                               2092
                                                                                        2093
1326     3           LEFT_SIDE='('||LEFT_SIDE||')';                                     2094
1327     3           CALL PARSE_EXP(LEFT_SIDE,EXP_RCVR);                                2095
                                                                                        2096
1328     3         END PROCESS_LET;                                                     2097
                                                                                        2098
1329     2         PROCESS_DEF:PROC;                                                    2099
                  /*********************************************************************2100
                  *                                                                   *2100
                  *   PROCESS DEF                                                     *2100
                  *                                                                   *2100
                  * NESTING:COMPILE                                                   *2100
                  *********************************************************************/2100
                                                                                        2105
1330     3           DECLARE I                   FIXED BINARY ALIGNED;                2106
1331     3           DECLARE JMP_OFFSET          FIXED BINARY ALIGNED;                2107
1332     3           DECLARE (OFFSET,OFFSET2)    FIXED BINARY ALIGNED;                2108
1333     3           DECLARE CH                  CHAR(1);                             2109
1334     3           DECLARE CH2                 CHAR(2);                             2110
1335     3           DECLARE (LEFT_SIDE,RIGHT_SIDE,FUNC_TEMP)                         2111
                                                 CHAR(80) VARYING;                    2112
1336     3           DECLARE TEMP_NAME           CHAR(10);                            2113
1337     3           DECLARE NO_EQUAL            BIT(1) ALIGNED;                      2114
                                                                                        2115
1338     3           TMP_CNT=50;                                                      2116
                                                                                        2117
1339     3           CH2=SUBSTR(STMT,STMT_CH,2);                                      2118
1340     3           IF CH2='FN' THEN;                                                2119
1342     3           ELSE                                                             2120
1342     3           DO;                                                              2121
1343     3    1         CALL PRINT_ERR(STMT_CH,'DEF MUST START WITH FN');              2122
1344     3    1         RETURN;                                                        2123
1345     3    1       END;                                                             2124
                                                                                        2125
                   /*  EXTRACT THE FUNCTION NAME   */                                 2126
                                                                                        2127
1346     3           LEFT_SIDE='';                                                    2128
1347     3           NO_EQUAL=TRUE;                                                   2129
1348     3           DO I=STMT_CH TO STMT_RIGHT WHILE(NO_EQUAL);                      2130
1349     3    1        CH=SUBSTR(STMT,I,1);                                          2131
```

```
 STMT LEVEL NEST

1350     3    1         IF CH='=' THEN                                                2132
1351     3    1         DO;                                                           2133
1352     3    2            NO_EQUAL=FALSE;                                            2134
1353     3    2            RIGHT_SIDE=SUBSTR(STMT,I+1);                               2135
1354     3    2         END;                                                          2136
1355     3    1         ELSE                                                          2137
1355     3    1            IF CH=' ' THEN;                                            2138
1357     3    1            ELSE                                                       2139
1357     3    1               LEFT_SIDE=LEFT_SIDE||CH;                                2140
1358     3    1       END;                                                           2141
                                                                                     2142
1359     3            CALL BALANCE_STMT(LEFT_SIDE);                                  2143
1360     3            IF TERMINATE_SCAN THEN RETURN;                                 2144
                                                                                     2145
1362     3            I=INDEX(LEFT_SIDE,'(');                                        2146
1363     3            IF I>0 THEN                                                    2147
1364     3            DO;                                                            2148
1365     3    1          FUNC_TEMP=SUBSTR(LEFT_SIDE,1,I-1);                           2149
1366     3    1          FUNC_NAME=FUNC_TEMP;                                         2150
1367     3    1          LEFT_SIDE=SUBSTR(LEFT_SIDE,I+1);                             2151
1368     3    1          IF VERIFY(FUNC_NAME,VALID_VAR_CHARS) > 0 THEN                2152
1369     3    1          DO;                                                          2153
1370     3    2             CALL PRINT_ERR(I,'INVALID FUNCTION NAME');                2154
1371     3    2             RETURN;                                                   2155
1372     3    2          END;                                                         2156
1373     3    1       END;                                                           2157
1374     3            ELSE                                                           2158
1374     3            DO;                                                            2159
1375     3    1          CALL PRINT_ERR(STMT_CH,'DEF SYNTAX ERROR');                  2160
1376     3    1          RETURN;                                                      2161
1377     3    1       END;                                                           2162
1378     3            IF SUBSTR(LEFT_SIDE,LENGTH(LEFT_SIDE),1)=')' THEN              2163
1379     3            DO;                                                            2164
1380     3    1          FUNC_ARG=SUBSTR(LEFT_SIDE,1,LENGTH(LEFT_SIDE)-1);            2165
1381     3    1          IF VERIFY(FUNC_ARG,VALID_VAR_CHARS) > 0 THEN                 2166
1382     3    1          DO;                                                          2167
1383     3    2             CALL PRINT_ERR(I,'INVALID FUNCTION ARGUMENT');            2168
1384     3    2             RETURN;                                                   2169
1385     3    2          END;                                                         2170
1386     3    1          FUNC_TEMP=FUNC_TEMP||' '||FUNC_ARG;                          2171
1387     3    1       END;                                                           2172
1388     3            ELSE                                                           2173
1388     3            DO;                                                            2174
1389     3    1          CALL PRINT_ERR(STMT_CH,'DEF ARGUMENT ERROR');                2175
1390     3    1          RETURN;                                                      2176
```

```
 STMT LEVEL NEST

1391     3    1       END;                                                                2177
                                                                                          2178
1392     3            OFFSET=LOOKUP_SYMBOL_TABLE(FUNC_NAME);                              2179
1393     3            IF OFFSET=SS_MAX THEN                                               2180
1394     3               IF SYM_TYPE(OFFSET)=SS_VAR THEN                                  2181
1395     3               DO;                                                              2182
1396     3    1              SYM_TYPE(OFFSET)=SS_DEF_VAR;                                 2183
1397     3    1              CALL ADD_PCODE(PC_OPCODE_JMP,ZERO);                          2184
1398     3    1              JMP_OFFSET=PC_MAX;                                           2185
1399     3    1              TEMP_NAME=FUNC_TEMP;                                         2186
1400     3    1              OFFSET2=LOOKUP_SYMBOL_TABLE(FUNC_TEMP);                      2187
1401     3    1              CALL ADD_PCODE(PC_OPCODE_STA,OFFSET2);                       2188
1402     3    1              IF OFFSET+1=OFFSET2 THEN;                                    2189
1404     3    1              ELSE                                                         2190
1404     3    1              DO;                                                          2191
1405     3    2                 CALL PRINT_ERR(ERR_PTR,'FUNC/ARG NOT CONTIG');            2192
1406     3    2                 RETURN;                                                   2193
1407     3    2              END;                                                         2194
1408     3    1              IF DF_MAX>=HBOUND(DF_NAME,1) THEN                            2195
1409     3    1              DO;                                                          2196
1410     3    2                 CALL PRINT_ERR(ERR_PTR,'TOO MANY DEF');                   2197
1411     3    2                 RETURN;                                                   2198
1412     3    2              END;                                                         2199
1413     3    1              DF_MAX=DF_MAX+1;                                             2200
1414     3    1              DF_NAME(DF_MAX)=FUNC_NAME;                                   2201
1415     3    1              DF_OFFSET(DF_MAX)=PC_MAX;                                    2202
1416     3    1              DF_RETURN(DF_MAX)=0;                                         2203
1417     3    1           END;                                                            2204
1418     3               ELSE                                                             2205
1418     3                  CALL PRINT_ERR(ERR_PTR,'DEF SYMBOL NOT FOUND');               2206
1419     3            ELSE                                                               2207
1419     3                CALL PRINT_ERR(ERR_PTR,'DEF SYMBOL REDEFINED');                2208
                                                                                          2209
1420     3            CALL BALANCE_STMT(RIGHT_SIDE);                                     2210
1421     3            IF TERMINATE_SCAN THEN RETURN;                                     2211
                                                                                          2212
1423     3            IF SUBSTR(RIGHT_SIDE,1,1)='(' THEN;                                2213
1425     3            ELSE RIGHT_SIDE='('||RIGHT_SIDE||')';                              2214
                                                                                          2215
1426     3            CALL PARSE_EXP(RIGHT_SIDE,EXP_FN_CALC);                            2216
                                                                                          2217
1427     3            CALL ADD_PCODE(PC_OPCODE_STA,OFFSET);                              2218
                                                                                          2219
1428     3            IF PC_MAX > 0 THEN        /*  CHANGE A STA TMPXX TO STA RESULT */  2220
1429     3            DO;                                                                2221
```

```
 STMT LEVEL NEST

1430     3    1          IF PC_OPCODE(PC_MAX)=PC_OPCODE_STA THEN                          2222
1431     3    1              IF SUBSTR(SYMBOL(PC_OBJECT(PC_MAX)),1,3)='TMP' THEN          2223
1432     3    1                 PC_OBJECT(PC_MAX)=1 ;                                     2224
1433     3    1       END;                                                                2225
1434     3            CALL ADD_PCODE(PC_OPCODE_RFN,OFFSET);                               2226
1435     3            PC_OBJECT(JMP_OFFSET)=PC_MAX+1;                                     2227
                                                                                         2228
1436     3         END PROCESS_DEF;                                                       2229
                                                                                         2230
1437     2         BALANCE_STMT:PROC(EXP);                                                2231
                  /*********************************************************************  2232
                  *                                                                   *  2232
                  *   CHECK FOR BALANCE PARENS AND QUOTES                             *  2232
                  *                                                                   *  2232
                  * NESTING:COMPILE                                                   *  2232
                  *********************************************************************/ 2232
                                                                                         2237
1438     3            DECLARE EXP                 CHAR(*) VARYING;                    2238
1439     3            DECLARE I                   FIXED BINARY ALIGNED;               2239
1440     3            DECLARE (PARENS,QUOTES)     FIXED BINARY ALIGNED;               2240
                                                                                         2241
1441     3            PARENS=0;                                                           2242
1442     3            QUOTES=0;                                                           2243
1443     3            DO I=1 TO LENGTH(EXP);                                              2244
1444     3    1          IF SUBSTR(EXP,I,1)='(' THEN PARENS=PARENS+1;                     2245
1446     3    1          ELSE                                                             2246
1446     3    1             IF SUBSTR(EXP,I,1)=')' THEN                                   2247
1447     3    1             DO;                                                           2248
1448     3    2                PARENS=PARENS-1;                                           2249
1449     3    2                IF PARENS < 0 THEN                                         2250
1450     3    2                   CALL PRINT_ERR(STMT_CH,'INVALID USE OF PARENS');        2251
1451     3    2                   RETURN;                                                 2252
1452     3    2                END;                                                       2253
1453     3    1          IF SUBSTR(EXP,I,1)=QUOTE_1 THEN QUOTES=QUOTES+1;                 2254
1455     3    1       END;                                                                2255
                                                                                         2256
1456     3            IF PARENS=0 THEN;                                                   2257
1458     3            ELSE                                                                2258
1458     3               CALL PRINT_ERR(STMT_CH,'UNBALANCED PARENS');                     2259
1459     3            IF MOD(QUOTES,2)=1 THEN   /* IF QUOTES IS ODD, ERROR */          2260
1460     3               CALL PRINT_ERR(STMT_CH,'UNBALANCED QUOTES');                     2261
                                                                                         2262
1461     3         END BALANCE_STMT;                                                      2263
                                                                                         2264
1462     2         PARSE_EXP:PROC(EXP,EXP_TYPE);                                          2265
```

```
STMT LEVEL NEST

                /*****************************************************************   2266
                *                                                               *   2266
                *                                                               *   2266
                *                                                               *   2266
                * NESTING:COMPILE                                               *   2266
                *****************************************************************/  2266
                                                                                      2271
1463     3          DECLARE EXP                CHAR(*) VARYING;                      2272
1464     3          DECLARE (I,J,EXP_TYPE)     FIXED BINARY ALIGNED;                 2273
1465     3          DECLARE EXPR               BIT(1) ALIGNED;                       2274
1466     3          DECLARE CH                 CHAR(1);                              2275
1467     3          DECLARE V                  CHAR(10) VARYING;                     2276
1468     3          DECLARE FN_TMP             CHAR(10);                             2277
1469     3          DECLARE (LAST_LP,OFFSET,                                         2278
                            RP,BREAKER,                                           2279
                            MAX_BREAKER)       FIXED BINARY ALIGNED;                 2280
1470     3          DECLARE NO_PARENS          BIT(1) ALIGNED;                       2281
1471     3          DECLARE CONTINUE_SCAN      BIT(1) ALIGNED;                       2282
                                                                                      2283
1472     3          DECLARE 1 STACK,                                                 2284
                             2  STACK_MAX      FIXED BINARY ALIGNED,                 2285
                             2  STACK_CUR      FIXED BINARY ALIGNED,                 2286
                             2  ITEMS(50),                                          2287
                                3  WORD        CHAR(10),                             2288
                                3  OP          CHAR(1);                              2289
                                                                                      2290
1473     3          STACK_MAX,STACK_CUR=0;                                           2291
1474     3          CALL POPULATE_STACK;                                             2292
1475     3          IF EXP_TYPE=EXP_FN_CALC THEN                                     2293
1476     3          DO;                                                              2294
1477     3    1        I=INDEX(FUNC_NAME,' ');                                       2295
1478     3    1        J=INDEX(FUNC_ARG,' ');                                        2296
1479     3    1        FN_TMP=SUBSTR(FUNC_NAME,1,I)||SUBSTR(FUNC_ARG,1,J)||(6)' ';   2297
1480     3    1        IF STACK_PRINT_DEBUG THEN                                     2298
1481     3    1           PUT SKIP DATA(FN_TMP);                                     2299
1482     3    1        DO I=1 TO STACK_MAX;                                          2300
1483     3    2           IF WORD(I)=FUNC_ARG THEN                                   2301
1484     3    2              WORD(I)=FN_TMP;                                         2302
1485     3    2           IF STACK_PRINT_DEBUG THEN                                  2303
1486     3    2              PUT SKIP DATA(ITEMS(I));                                2304
1487     3    2        END;                                                          2305
1488     3    1     END;                                                             2306
                                                                                      2307
1489     3          EXPR=(EXP_TYPE=EXP_CALC | EXP_TYPE=EXP_FN_CALC);                 2308
                                                                                      2309
```

```
 STMT LEVEL NEST

1490     3           BREAKER=0;                                                  2310
1491     3           MAX_BREAKER=STACK_MAX;                                      2311
1492     3           CONTINUE_SCAN=TRUE;                                         2312
1493     3           DO WHILE(CONTINUE_SCAN & BREAKER <= MAX_BREAKER);           2313
1494     3    1         I=0;                                                     2314
1495     3    1         NO_PARENS=TRUE;                                          2315
1496     3    1         LAST_LP=0;                                               2316
1497     3    1         DO WHILE(NO_PARENS & I<= STACK_MAX);                     2317
1498     3    2            I=I+1;                                                2318
1499     3    2            IF OP(I)='(' THEN LAST_LP=I;                          2319
1501     3    2            ELSE                                                  2320
1501     3    2               IF OP(I)=')' THEN                                  2321
1502     3    2                  NO_PARENS=FALSE;                                2322
1503     3    2         END; /* DO WHILE(NO_PARENS....  */                       2323
1504     3    1         IF NO_PARENS THEN                                        2324
1505     3    1         DO;                                                      2325
1506     3    2            LAST_LP=1;                                            2326
1507     3    2            RP=STACK_MAX;                                         2327
1508     3    2         END;                                                     2328
1509     3    1         ELSE                                                     2329
1509     3    1            RP=I;                                                 2330
1510     3    1         CALL SIMPLIFY_SUB_STACK(LAST_LP,RP);                     2331
1511     3    1         IF STACK_MAX > 1 THEN                                    2332
1512     3    1            CONTINUE_SCAN = TRUE;                                 2333
1513     3    1         ELSE                                                     2334
1513     3    1            CONTINUE_SCAN = FALSE;                                2335
1514     3    1         BREAKER=BREAKER+1;                                       2336
1515     3    1       END; /* DO WHILE(CONTINUE_SCAN.....  */                    2337
                                                                                 2338
1516     3           IF BREAKER>MAX_BREAKER THEN                                 2339
1517     3           DO;                                                         2340
1518     3    1          PUT SKIP LIST('BREAKER>MAX');                          2341
1519     3    1       END;                                                       2342
                                                                                 2343
1520     3           IF STACK_MAX < 3 THEN  /*  SIMPLE VARIABLE? */             2344
1521     3           DO;                                                         2345
1522     3    1        OFFSET=LOOKUP_SYMBOL_TABLE(WORD(1));                     2346
1523     3    1        DO I=1 TO SS_MAX;                                        2347
1524     3    2          IF WORD(1)=SYMBOL(I) THEN                              2348
1525     3    2          DO;                                                    2349
1526     3    3            IF EXPR THEN                                         2350
1527     3    3            DO;                                                  2351
                        /***  IF SYM_TYPE(I)=SS_FUNC THEN                      2352
                        *       RC=PR_FUNC;                                     2352
                        *     ELSE                                              2352
```

```
STMT LEVEL NEST

                     *       IF SYM_TYPE(I)=SS_VAR THEN                                  2352
                     *          RC=PR_VAR;                                               2352
                     *       ELSE                                                        2352
                     *           IF SYM_TYPE(I)=SS_DIM_VAR THEN                          2352
                     *              RC=PR_SUB_VAR;                                       2352
                     *           ELSE                                                    2352
                     *               IF SYM_TYPE(I)=SS_CONST THEN                        2352
                     *                  RC=PR_VAR;                                       2352
                     *               ELSE                                                2352
                     *                   CALL PRINT_ERR(STMT_CH,'VARIABLE EXPECTED');    2352
                     */                                                                  2352
1528     3    4      END;                                                                2365
1529     3    3            ELSE                                                         2366
1529     3    3            DO;                                                          2367
1530     3    4              IF SYM_TYPE(I)=SS_VAR THEN                                 2368
1531     3    4              DO;                                                        2369
1532     3    5                IF PC_OPCODE(PC_MAX)=PC_OPCODE_STA THEN                  2370
1533     3    5                  IF SUBSTR(SYMBOL(PC_OBJECT(PC_MAX)),1,3)='TMP' THEN   2371
1534     3    5                    PC_OBJECT(PC_MAX)=I;                                2372
1535     3    5                  ELSE                                                  2373
                                  /* CALL PRINT_ERR(STMT_CH,                          2374
                                                'SYNTAX ERROR RESULT EXPECTED') */    2374
1535     3    5    ;                                                                   2375
1536     3    5                ELSE                                                    2376
1536     3    5                   CALL ADD_PCODE(PC_OPCODE_STA,OFFSET);                2377
1537     3    5              END;                                                       2378
1538     3    4              ELSE                                                       2379
1538     3    4                IF SYM_TYPE(I)=SS_STRCON  |                              2380
                                  SYM_TYPE(I)=SS_STRVAR  |                            2381
                                  SYM_TYPE(I)=SS_STRDIM  |               /*PAT 02*/   2382
1539     3    4                   SYM_TYPE(I)=SS_DIM_VAR THEN;                        2383
1540     3    4                ELSE                                                   2384
1540     3    4                  CALL PRINT_ERR(STMT_CH,'A VARIABLE IS EXPECTED HERE');2385
1541     3    4            END;                                                       2386
1542     3    3            RETURN;                                                    2387
1543     3    3          END;                                                         2388
1544     3    2         END; /* DO LOOP */                                           2389
1545     3    1         CALL PRINT_ERR(STMT_CH,'VARIABLE ' || WORD(1) || ' UNDEFINED?');2390
1546     3    1       END;                                                           2391
1547     3            ELSE                                                           2392
1547     3               CALL PRINT_ERR(STMT_CH,'BIG TIME SYNTAX ERROR IN EXPRESSION');2393
                                                                                    2394
1548     3         POPULATE_STACK:PROC;                                              2395
                  /*****************************************************************2396
                  *                                                               *2396
```

```
STMT LEVEL NEST

                  *  BREAK EXP INTO WORDS AT THE SPECIAL CHARACTERS                  *   2396
                  *  TO SUPPORT STRINGS, ANY STRING CONSTANTS WILL BE EXTRACTED AND  *   2396
                  *  ADDED TO SYMBOL TABLE USING A GENERATED NAME.  THIS NAME WILL   *   2396
                  *  BE SUBSTITUTED IN THE STACK FOR THE STRING.                    *   2396
                  *                                                                 *   2396
                  *******************************************************************   2396
                  * NESTING:COMPILE - PARSE_EXP                                     *   2396
                  *******************************************************************/  2396
                                                                                        2405
1549     4           DECLARE IN_STR         BIT(1) ALIGNED;                           2406
1550     4           DECLARE STR_WORK       CHAR(80) VARYING;                         2407
1551     4           DECLARE TMP_VAR        CHAR(10);                                 2408
1552     4           DECLARE OFFSET         FIXED BINARY ALIGNED;                     2409
1553     4           IN_STR = FALSE;                                                  2410
                                                                                      2411
1554     4           STR_WORK='';                                                     2412
1555     4           V='';                                                            2413
1556     4           DO I=1 TO LENGTH(EXP);                                           2414
1557     4    1         CH=SUBSTR(EXP,I,1);                                           2415
1558     4    1         IF IN_STR THEN                                                2416
1559     4    1         DO;                                                           2417
1560     4    2            IF CH=QUOTE_1 THEN                                         2418
1561     4    2            DO;                                                        2419
1562     4    3               IN_STR=FALSE;                                           2420
1563     4    3               TMP_VAR='STR$'||STR_CNT;                                2421
1564     4    3               STR_CNT=STR_CNT+1;                                      2422
1565     4    3               V=TMP_VAR;                                              2423
1566     4    3               OFFSET=LOOKUP_SYMBOL_TABLE(TMP_VAR);                    2424
1567     4    3               IF STACK_PRINT_DEBUG THEN                               2425
1568     4    3                  PUT DATA(STR_WORK,V,TMP_VAR,OFFSET);                 2426
1569     4    3               STRING_VAL(OFFSET)=STR_WORK;                            2427
1570     4    3            END;                                                       2428
1571     4    2            ELSE                                                       2429
1571     4    2               STR_WORK=STR_WORK||CH;                                  2430
1572     4    2         END;                                                          2431
1573     4    1         ELSE                                                          2432
1573     4    1         IF CH=QUOTE_1 THEN                                            2433
1574     4    1         DO;                                                           2434
1575     4    2             IN_STR=TRUE;                                              2435
1576     4    2         END;                                                          2436
1577     4    1         ELSE                                                          2437
1577     4    1         IF CH='(' | CH=')' | CH='+' | CH='-' | CH='*' | CH='/' |      2438
                        CH='^' THEN                                                   2439
1578     4    1         DO;                                                           2440
1579     4    2             STACK_MAX=STACK_MAX+1;                                    2441
```

```
 STMT LEVEL NEST

1580     4    2              WORD(STACK_MAX)=V;                                          2442
1581     4    2              OP(STACK_MAX)=CH;                                           2443
1582     4    2              V='';                                                       2444
1583     4    2           END;                                                           2445
1584     4    1           ELSE                                                           2446
1584     4    1               V=V||CH;                                                   2447
1585     4    1        END;                                                              2448
                                                                                        2449
1586     4             IF STACK_MAX=0 THEN   /*  EXP IS A SIMPLE VAR OR CONSTANT */    2450
1587     4             DO;                                                               2451
1588     4    1           STACK_MAX=1;                                                   2452
1589     4    1           WORD(1)=V;                                                     2453
1590     4    1           OP(1)=' ';                                                     2454
1591     4    1        END;                                                              2455
1592     4             ELSE                                                              2456
1592     4             DO I=1 TO STACK_MAX;                                              2457
1593     4    1           IF WORD(I)=(10)' ' THEN                                        2458
1594     4    1              IF OP(I)= '(' THEN;                                         2459
1596     4    1              ELSE                                                        2460
1596     4    1                 IF I>1 THEN                                            2461
1597     4    1                    IF OP(I-1)=')' | OP(I)='-' THEN;                    2462
1599     4    1                    ELSE                                                2463
1599     4    1                       CALL PRINT_ERR(STMT_CH,'SYNTAX ERROR');          2464
1600     4    1        END;                                                              2465
                                                                                        2466
1601     4             IF STACK_PRINT_DEBUG THEN                                         2467
1602     4             DO;                                                               2468
1603     4    1           PUT SKIP LIST('POPULTAT STACK EXIT');                          2469
1604     4    1           DO I=1 TO STACK_MAX;                                           2470
1605     4    2              PUT SKIP LIST(I,WORD(I),OP(I));                             2471
1606     4    2           END;                                                           2472
1607     4    1        END;                                                              2473
                                                                                        2474
1608     4          END POPULATE_STACK;                                                  2475
                                                                                        2476
1609     3          SIMPLIFY_SUB_STACK:PROC(LP,RP);                                      2477
                    /*********************************************************************2478
                    *                                                                   *2478
                    *                                                                   *2478
                    *   PROCESS OPERATORE LOCATED BETWEEN THE LEFT PAREN (LP) AND       *2478
                    *   RIGHT PAREN (RP).  FIRST CHECK STACK FOR A SIMPLE QUANTITY      *2478
                    *   (I.E. (X) ) OR VAIABLE IN THE STACK.  NO NEED TO SIMPLFY THESE  *2478
                    *                                                                   *2478
                    *                                                                   *2478
                    *********************************************************************2478
```

```
STMT LEVEL NEST

                 * NESTING:COMPILE - PARSE_EXP                                    *   2478
                 ********************************************************************/   2478
                                                                                        2488
1610     4          DECLARE (LP,RP)                FIXED BINARY ALIGNED;               2489
1611     4          DECLARE (I,J,K,HJ,HK,OFFSET)   FIXED BINARY ALIGNED;               2490
1612     4          DECLARE  TMP_VAR               CHAR(5);                            2491
                                                                                        2492
1613     4          IF STACK_PRINT_DEBUG THEN                                          2493
1614     4          DO;                                                                2494
1615     4    1        PUT SKIP LIST('SIMPLIFY_SUB_STACK ENTRY FROM',LP,'TO ',RP);      2495
1616     4    1        DO I=LP TO RP;                                                   2496
1617     4    2          PUT SKIP LIST(I,WORD(I),OP(I));                                2497
1618     4    2        END;                                                             2498
1619     4    1      END;                                                               2499
                                                                                        2500
1620     4          IF STACK_MAX = 1 THEN                                              2501
1621     4              GO TO SIMPLIFY_SUB_STACK_EXIT;                                 2502
                                                                                        2503
                        /*  CHECK FOR SIMPLE VARIABLE QUANTITY, DIM VAR OR FUNC */     2504
                                                                                        2505
1622     4          IF STACK_MAX = 2 THEN                                              2506
1623     4          DO;                                                                2507
1624     4    1        IF OP(LP)='(' & OP(LP+1)=')' THEN                                2508
1625     4    1        DO;                                                              2509
1626     4    2          IF WORD(LP)=(10)' ' THEN                                       2510
1627     4    2            IF WORD(LP+1)=(10)' ' THEN   /* EMPTY PARENS ERROR */     2511
1628     4    2            DO;                                                          2512
1629     4    3              CALL PRINT_ERR(STMT_CH,'EMPTY PARENS ERROR');              2513
1630     4    3              RETURN;                                                    2514
1631     4    3            END;                                                         2515
1632     4    2            ELSE                    /*  SIMPLE QUANTITY  */           2516
                         /*** DO;                                                     2517
                         **      WORD(LP)=WORD(LP+1);                                 2517
                         **      OP(LP)=' ';                                          2517
                         **      STACK_MAX=STACK_MAX-1;                               2517
                         **      GO TO SIMPLIFY_SUB_STACK_EXIT;                       2517
                         **** END;  *****/                                           2517
1632     4    2    ;                                                                 2522
1633     4    2          ELSE                                                         2523
1633     4    2          DO;                          /* COULD BE X(Y) OR FNC(Y)  */  2524
1634     4    3            IF STACK_PRINT_DEBUG THEN                                  2525
1635     4    3            DO;                                                        2526
1636     4    4              PUT SKIP LIST('FOUND DIM VAR OR FUNC');                  2527
1637     4    4            END;                                                       2528
1638     4    3            OFFSET=LOOKUP_SYMBOL_TABLE(WORD(LP));                      2529
```

```
 STMT LEVEL NEST

1639     4    3               IF SYM_TYPE(OFFSET)=SS_DIM_VAR |                          2530
1640     4    3                  SYM_TYPE(OFFSET)=SS_FUNC       THEN;                  2531
1641     4    3               ELSE                                                      2532
1641     4    3               DO;                                                       2533
1642     4    4                   CALL PRINT_ERR(STMT_CH,                               2534
                                    'EXPECTING DIM VARIABLE OR FUNCTION CALL');        2535
1643     4    4                   RETURN;                                               2536
1644     4    4               END;                                                      2537
1645     4    3            END;                                                         2538
1646     4    2         END;                                                            2539
1647     4    1      END;                                                              2540
                                                                                       2541
1648     4           IF OP(LP)= '(' THEN                                               2542
1649     4           DO;                                                               2543
1650     4    1         HJ=LP+1;                                                       2544
1651     4    1         HK=RP-1;                                                       2545
1652     4    1      END;                                                              2546
1653     4           ELSE                                                              2547
1653     4           DO;                                                               2548
1654     4    1         HJ=LP;                                                         2549
1655     4    1         HK=RP;                                                         2550
1656     4    1      END;                                                              2551
                                                                                       2552
1657     4           J=HJ;                                                             2553
1658     4           K=HK;                                                             2554
1659     4           IF K<J THEN                                                       2555
1660     4           DO;                                                               2556
1661     4    1         IF STACK_PRINT_DEBUG THEN                                      2557
1662     4    1            PUT SKIP(2) LIST('PROCESS_OPERATORS BYPASSED',J,K);         2558
1663     4    1      END;                                                              2559
1664     4           ELSE                                                              2560
1664     4           DO;                                                               2561
1665     4    1         CALL PROCESS_OPERATORS('^','^',PC_OPCODE_EXP,PC_OPCODE_EXP);   2562
1666     4    1         J=HJ;                                                          2563
1667     4    1         CALL PROCESS_OPERATORS('*','/',PC_OPCODE_MUL,PC_OPCODE_DIV);   2564
1668     4    1         J=HJ;                                                          2565
1669     4    1         CALL PROCESS_OPERATORS('+','-',PC_OPCODE_ADD,PC_OPCODE_SUB);   2566
1670     4    1      END;                                                              2567
                  /*   IF OP(LP)='(' & OP(LP+1)=')' THEN                              2568
                     DO; */                                                            2568
                                                                                       2569
1671     4              IF WORD(LP)=(10)' ' THEN          /* QUANTITY - GET RID OF () */  2570
1672     4              DO;                                                            2571
1673     4    1            IF STACK_MAX=1 THEN;                                       2572
1675     4    1            ELSE                                                       2573
```

```
 STMT LEVEL NEST

1675     4    1            IF STACK_MAX=2 THEN                                    2574
1676     4    1            DO;                                                    2575
1677     4    2               WORD(1)=WORD(2);                                    2576
1678     4    2               OP(1)=' ';                                          2577
1679     4    2               STACK_MAX=1;                                        2578
1680     4    2               OFFSET=LOOKUP_SYMBOL_TABLE(WORD(1));                2579
1681     4    2               IF EXPR THEN                                        2580
1682     4    2                   CALL ADD_PCODE(PC_OPCODE_LDA,OFFSET);           2581
1683     4    2               ELSE                                                2582
1683     4    2                   CALL ADD_PCODE(PC_OPCODE_STA,OFFSET);           2583
1684     4    2            END;                                                   2584
1685     4    1            ELSE                                                   2585
1685     4    1            DO;                                                    2586
1686     4    2               IF STACK_PRINT_DEBUG THEN                           2587
1687     4    2               DO;                                                 2588
1688     4    3                  PUT SKIP LIST('SIMPLIFY_SUB_STACK BEFORE UPDATE',2589
                                        LP,RP);                                  2590
1689     4    3                  DO I=1 TO STACK_MAX;                            2591
1690     4    4                     PUT SKIP LIST(I,WORD(I),OP(I));              2592
1691     4    4                  END;                                            2593
1692     4    3               END;                                                2594
1693     4    2               WORD(LP)=WORD(LP+1);                                2595
1694     4    2               IF LP+2 < STACK_MAX THEN                            2596
1695     4    2                  OP(LP)=OP(LP+2);                                 2597
1696     4    2               ELSE                                                2598
1696     4    2                  OP(LP)=' ';                                      2599
                            /*   WORD(LP+1)=WORD(LP+3); */                       2600
1697     4    2               DO I=LP+1 TO STACK_MAX;                             2601
1698     4    3                  ITEMS(I)=ITEMS(I+2);                             2602
1699     4    3               END;                                                2603
1700     4    2               IF LP+1=STACK_MAX THEN                              2604
1701     4    2                  STACK_MAX=STACK_MAX-1;                           2605
1702     4    2               ELSE                                                2606
1702     4    2                  STACK_MAX=STACK_MAX-2;                           2607
1703     4    2               K=K-1;                                              2608
1704     4    2               J=LP;                                               2609
1705     4    2               IF STACK_PRINT_DEBUG THEN                           2610
1706     4    2               DO;                                                 2611
1707     4    3                  PUT SKIP LIST('SIMPLIFY_SUB_STACK AFTER UPDATE',2612
                                        LP,RP);                                  2613
1708     4    3                  DO I=1 TO STACK_MAX;                            2614
1709     4    4                     PUT SKIP LIST(I,WORD(I),OP(I));              2615
1710     4    4                  END;                                            2616
1711     4    3               END;                                                2617
1712     4    2            END;                                                   2618
```

```
 STMT LEVEL NEST

1713     4    1          END;                                                    2619
1714     4               ELSE    /*  COULD BE A FUNCTION OR DIM VARIABLE */     2620
1714     4               DO;                                                     2621
1715     4    1            OFFSET=LOOKUP_SYMBOL_TABLE(WORD(LP));                2622
                          /*  PUT SKIP DATA(OFFSET,SYM_TYPE(OFFSET)); */       2623
1716     4    1            IF SYM_TYPE(OFFSET)=SS_FUNC |                        2624
                             SYM_TYPE(OFFSET)=SS_DEF_VAR THEN                  2625
1717     4    1               CALL PROCESS_FUNCTION(OFFSET);                    2626
1718     4    1            ELSE                                                 2627
1718     4    1               IF SYM_TYPE(OFFSET)=SS_DIM_VAR |                  2628
                                SYM_TYPE(OFFSET)=SS_STRDIM  THEN               2629
1719     4    1                  CALL PROCESS_SUBSCRIPT(OFFSET);                2630
1720     4    1               ELSE                                              2631
1720     4    1                  IF SYM_TYPE(OFFSET)=SS_VAR THEN;               2632
1722     4    1                  ELSE                                           2633
1722     4    1                     CALL PRINT_ERR(STMT_CH,'EXPECTING DIM');    2634
1723     4    1          END;                                                    2635
                   /*   END;                                                   2636
                      ELSE                                                     2636
                         PUT SKIP LIST('*** NO PARENS ***');*/                 2636
                                                                               2638
                                                                               2639
1724     4         SIMPLIFY_SUB_STACK_EXIT:                                    2640
                      IF STACK_PRINT_DEBUG THEN                                2641
1725     4            DO;                                                      2642
1726     4    1          PUT SKIP LIST('SIMPLIFY_SUB_STACK EXIT');             2643
1727     4    1          DO I=1 TO STACK_MAX;                                  2644
1728     4    2             PUT SKIP LIST(I,WORD(I),OP(I));                    2645
1729     4    2          END;                                                  2646
1730     4    1       END;                                                     2647
                                                                               2648
                                                                               2649
1731     4         PROCESS_OPERATORS:PROC(OP1,OP2,PC1,PC2);                    2650
                   /*************************************************************2651
                   *                                                           *2651
                   * THIS PROC SCANS FOR OP1 AND OP2 WITHIN ROWS J AND K OF THE STACK  *2651
                   * PC1 AND PC2 ARE THE OPCODES FOR OP1 AND OP2 RESPECTIVELY          *2651
                   * THE VARIABLES J AND K ARE GLOBAL TO SIMPLYFY_SUB_STACK            *2651
                   * THEY CONTAIN THE FIRST AND LAST ROWS FOR THIS PROC TO PROCESS     *2651
                   *                                                           *2651
                   *************************************************************2651
                   * NESTING:COMPILE - PARSE_EXP - SIMPLYFY_SUB_STACK          *2651
                   *************************************************************/2651
                                                                               2660
1732     5            DECLARE (OP1,OP2)              CHAR(1),                 2661
```

```
STMT LEVEL NEST

                         (PC1,PC2)              FIXED BINARY ALIGNED;              2662
1733     5           DECLARE OFFSET              FIXED BINARY ALIGNED;              2663
                                                                                    2664
1734     5           IF STACK_PRINT_DEBUG THEN                                      2665
1735     5           DO;                                                            2666
1736     5    1         PUT SKIP(2) LIST('PROCESS_OPERATORS ENTRY',J,K,OP1,OP2);    2667
1737     5    1         DO I=1 TO STACK_MAX;                                        2668
1738     5    2            PUT SKIP LIST(I,WORD(I),OP(I));                          2669
1739     5    2         END;                                                        2670
1740     5    1       END;                                                          2671
                                                                                    2672
1741     5           DO WHILE(J<=K);                                                2673
1742     5    1         IF OP(J)=OP1 | OP(J)=OP2 THEN                               2674
1743     5    1         DO;                                                         2675
1744     5    2            TMP_VAR='TMP'||TMP_CNT;                                  2676
1745     5    2            TMP_CNT=TMP_CNT+1;                                       2677
                                                                                    2678
1746     5    2            OFFSET=LOOKUP_SYMBOL_TABLE(WORD(J));                     2679
1747     5    2            CALL ADD_PCODE(PC_OPCODE_LDA,OFFSET);                    2680
                                                                                    2681
1748     5    2            OFFSET=LOOKUP_SYMBOL_TABLE(WORD(J+1));                   2682
1749     5    2            IF OP(J)=OP1 THEN                                        2683
1750     5    2               CALL ADD_PCODE(PC1,OFFSET);                           2684
1751     5    2            ELSE                                                     2685
1751     5    2               CALL ADD_PCODE(PC2,OFFSET);                           2686
                                                                                    2687
1752     5    2            WORD(J)=TMP_VAR;                                         2688
1753     5    2            OFFSET=LOOKUP_SYMBOL_TABLE(WORD(J));                     2689
1754     5    2            CALL ADD_PCODE(PC_OPCODE_STA,OFFSET);                    2690
                                                                                    2691
1755     5    2            OP(J)=OP(J+1);                                           2692
1756     5    2            DO I=J+1 TO STACK_MAX;                                   2693
1757     5    3               ITEMS(I)=ITEMS(I+1);                                  2694
1758     5    3            END;                                                     2695
1759     5    2            STACK_MAX=STACK_MAX-1;                                   2696
1760     5    2            K=K-1;                                                   2697
1761     5    2            J=LP;                                                    2698
1762     5    2            IF STACK_PRINT_DEBUG THEN                                2699
1763     5    2            DO;                                                      2700
1764     5    3               PUT SKIP LIST('PROCESS_OPERATORE UPDATE',J,K);        2701
1765     5    3               DO I=1 TO STACK_MAX;                                  2702
1766     5    4                 PUT SKIP LIST(I,WORD(I),OP(I));                     2703
1767     5    4               END;                                                  2704
1768     5    3            END;                                                     2705
1769     5    2         END;                                                        2706
```

```
 STMT LEVEL NEST

1770     5    1          J=J+1;                                                    2707
1771     5    1       END; /* DO WHILE */                                        2708
                                                                                  2709
1772     5            IF STACK_PRINT_DEBUG THEN                                 2710
1773     5            DO;                                                       2711
1774     5    1          PUT SKIP LIST('PROCESS_OPERATORS EXIT',J,K);           2712
1775     5    1          DO I=1 TO STACK_MAX;                                   2713
1776     5    2             PUT SKIP LIST(I,WORD(I),OP(I));                     2714
1777     5    2          END;                                                   2715
1778     5    1       END;                                                      2716
                                                                                  2717
1779     5         END PROCESS_OPERATORS;                                       2718
                                                                                  2719
1780     4         PROCESS_FUNCTION:PROC(OFFSET);                               2720
                  /*****************************************************************
                  *                                                               *  2721
                  *                                                               *  2721
                  *                                                               *  2721
                  * NESTING:COMPILE - PARSE_EXP - SIMPLYFY_SUB_STACK              *  2721
                  *****************************************************************/ 2721
                                                                                  2726
1781     5            DECLARE (OFFSET,OFFSET2,OFFSET3,I)       FIXED BINARY ALIGNED;  2727
1782     5            DECLARE TEMP_SYM                         CHAR(10) INITIAL((10)' '); 2728
1783     5            OFFSET2=LOOKUP_SYMBOL_TABLE(WORD(LP+1));                  2729
                                                                                  2730
1784     5            IF STACK_PRINT_DEBUG THEN                                 2731
1785     5            DO;                                                       2732
1786     5    1          PUT SKIP LIST('PROCESS_FUNCTION ENTRY');               2733
1787     5    1          DO I=1 TO STACK_MAX;                                   2734
1788     5    2               PUT SKIP LIST(I,WORD(I),OP(I));                   2735
1789     5    2          END;                                                   2736
1790     5    1       END;                                                      2737
                                                                                  2738
1791     5            IF SYM_TYPE(OFFSET2)=SS_VAR |                             2739
                        SYM_TYPE(OFFSET2)=SS_CONST THEN                           2740
1792     5            DO;                                                       2741
1793     5    1          TMP_VAR='TMP'||TMP_CNT;                                2742
1794     5    1          TMP_CNT=TMP_CNT+1;                                     2743
1795     5    1          TEMP_SYM=TMP_VAR;                                      2744
1796     5    1          OFFSET3=LOOKUP_SYMBOL_TABLE(TEMP_SYM);                 2745
                                                                                  2746
1797     5    1          IF SYM_TYPE(OFFSET)=SS_FUNC THEN                       2747
1798     5    1          DO;                                                    2748
1799     5    2             CALL ADD_PCODE(PC_OPCODE_LDA,OFFSET2);              2749
1800     5    2             CALL ADD_PCODE(PC_OPCODE_FNC,OFFSET);               2750
```

```
 STMT LEVEL NEST

1801     5    2            CALL ADD_PCODE(PC_OPCODE_STA,OFFSET3);                    2751
1802     5    2         END;                                                         2752
1803     5    1         ELSE                                                         2753
1803     5    1         IF SYM_TYPE(OFFSET)=SS_DEF_VAR THEN                          2754
1804     5    1         DO;                                                          2755
1805     5    2            CALL ADD_PCODE(PC_OPCODE_LDA,OFFSET2);                    2756
1806     5    2            CALL ADD_PCODE(PC_OPCODE_CFN,OFFSET);                     2757
1807     5    2            CALL ADD_PCODE(PC_OPCODE_STA,OFFSET3);                    2758
1808     5    2         END;                                                         2759
1809     5    1         ELSE                                                         2760
1809     5    1            CALL PRINT_ERR(STMT_CH,'UNKNOWN FUNCTION DETECTED');      2761
                      /* CALL ADD_PCODE(PC_OPCODE_STA,OFFSET3); */                  2762
                         /*************************************************************
                         *                                                           *
                         *  ADJUST THE STACK TO REPLACE THE FUNC REF WITH THE TEMP VAR *
                         *  IF NO OTHER OPERATORS FOLLOW, PUSH UP 2 ITEMS,  IF OTHER  *
                         *  OPERATORS FOLLOW, PUSH UP 1 ITEM ONLY.                   *
                         *                                                           *
                         *************************************************************/
                                                                                   2769
1810     5    1         IF STACK_MAX < 5 THEN                                        2770
1811     5    1         DO;                                                          2771
1812     5    2            WORD(LP)=TMP_VAR;                                         2772
1813     5    2            OP(LP)=')';                                               2773
1814     5    2            DO I=LP+2 TO STACK_MAX;                                   2774
1815     5    3              ITEMS(I-1)=ITEMS(I);                                    2775
1816     5    3            END;                                                      2776
1817     5    2            STACK_MAX=STACK_MAX-2;                                    2777
1818     5    2         END;                                                         2778
1819     5    1         ELSE                                                         2779
1819     5    1         DO;                                                          2780
1820     5    2            WORD(LP)=TMP_VAR;                                         2781
1821     5    2            OP(LP)=OP(LP+2);                                          2782
1822     5    2            DO I=LP+3 TO STACK_MAX;                                   2783
1823     5    3              ITEMS(I-2)=ITEMS(I);                                    2784
1824     5    3            END;                                                      2785
1825     5    2            STACK_MAX=STACK_MAX-2;                                    2786
1826     5    2         END;                                                         2787
1827     5    1       END;                                                           2788
1828     5            ELSE                                                         2789
1828     5               CALL PRINT_ERR(STMT_CH,'INVALID FUNCTION ARGUMENT');      2790
                                                                                   2791
1829     5            IF STACK_PRINT_DEBUG THEN                                    2792
1830     5            DO;                                                          2793
1831     5    1          PUT SKIP LIST('PROCESS_FUNCTION EXIT');                   2794
```

```
 STMT LEVEL NEST

1832     5    1          DO I=1 TO STACK_MAX;                                          2795
1833     5    2               PUT SKIP LIST(I,WORD(I),OP(I));                          2796
1834     5    2          END;                                                          2797
1835     5    1       END;                                                             2798
                                                                                      2799
1836     5         END PROCESS_FUNCTION;                                              2800
                                                                                      2801
1837     4         PROCESS_SUBSCRIPT:PROC(OFFSET);                                    2802
                   /*******************************************************************
                   *                                                                 *  2803
                   *                                                                 *  2803
                   *                                                                 *  2803
                   * NESTING:COMPILE - PARSE_EXP - SIMPLYFY_SUB_STACK                *  2803
                   *******************************************************************/ 2803
                                                                                      2808
1838     5            DECLARE (OFFSET,OFFSET2,OFFSET3,I)       FIXED BINARY ALIGNED;   2809
1839     5            DECLARE TEMP_SYM                         CHAR(10) INITIAL((10)' '); 2810
1840     5            DECLARE TMP_VAR                          CHAR(10) INITIAL((10)' '); 2811
1841     5            OFFSET2=LOOKUP_SYMBOL_TABLE(WORD(LP+1));                       2812
                                                                                      2813
1842     5            IF STACK_PRINT_DEBUG THEN                                      2814
1843     5            DO;                                                            2815
1844     5    1          PUT SKIP LIST('PROCESS_SUBSCRIPT ENTRY');                    2816
1845     5    1          DO I=1 TO STACK_MAX;                                         2817
1846     5    2               PUT SKIP LIST(I,WORD(I),OP(I));                         2818
1847     5    2          END;                                                         2819
1848     5    1       END;                                                            2820
                                                                                      2821
1849     5            IF SYM_TYPE(OFFSET2)=SS_VAR |                                  2822
                        SYM_TYPE(OFFSET2)=SS_CONST THEN                               2823
1850     5            DO;                                                            2824
1851     5    1          IF SYM_TYPE(OFFSET)=SS_DIM_VAR |                            2825
                           SYM_TYPE(OFFSET)=SS_STRDIM  THEN                           2826
1852     5    1          DO;                                                         2827
1853     5    2             IF SYM_TYPE(OFFSET)=SS_DIM_VAR THEN                      2828
1854     5    2                TMP_VAR='TMP'||TMP_CNT;                               2829
1855     5    2             ELSE                                                     2830
1855     5    2                TMP_VAR='TMP'||TMP_CNT||'$';                          2831
1856     5    2             TMP_CNT=TMP_CNT+1;                                       2832
1857     5    2             TEMP_SYM=TMP_VAR;                                        2833
1858     5    2             OFFSET3=LOOKUP_SYMBOL_TABLE(TEMP_SYM);                   2834
                                                                                      2835
1859     5    2             IF EXPR THEN                                             2836
1860     5    2             DO;                                                      2837
1861     5    3                CALL ADD_PCODE(PC_OPCODE_LDR,OFFSET2);                2838
```

```
 STMT LEVEL NEST

1862     5    3               CALL ADD_PCODE(PC_OPCODE_DSL,ZERO);                    2839
1863     5    3               CALL ADD_PCODE(PC_OPCODE_LDA,OFFSET);                  2840
1864     5    3               CALL ADD_PCODE(PC_OPCODE_STA,OFFSET3);                 2841
1865     5    3            END;                                                      2842
1866     5    2            ELSE                                                      2843
1866     5    2            DO;                                                       2844
1867     5    3               CALL ADD_PCODE(PC_OPCODE_LDR,OFFSET2);                 2845
1868     5    3               CALL ADD_PCODE(PC_OPCODE_DSL,ZERO);                    2846
1869     5    3               CALL ADD_PCODE(PC_OPCODE_STA,OFFSET);                  2847
1870     5    3               TMP_VAR=SYMBOL(OFFSET);                                2848
1871     5    3            END;                                                      2849
                       /*************************************************************   2850
                       *                                                           *   2850
                       *  ADJUST THE STACK TO REPLACE THE SUB  REF WITH THE TEMP VAR *   2850
                       *  IF NO OTHER OPERATORS FOLLOW, PUSH UP 2 ITEMS,  IF OTHER   *   2850
                       *  OPERATORS FOLLOW, PUSH UP 1 ITEM ONLY.                   *   2850
                       *                                                           *   2850
                       *************************************************************/  2850
                                                                                        2856
1872     5    2            IF STACK_MAX < 5 THEN                                     2857
1873     5    2            DO;                                                       2858
1874     5    3               WORD(LP)=TMP_VAR;                                      2859
1875     5    3               OP(LP)=')';                                            2860
1876     5    3               DO I=LP+2 TO STACK_MAX;                                2861
1877     5    4                 ITEMS(I-1)=ITEMS(I);                                 2862
1878     5    4               END;                                                   2863
1879     5    3               STACK_MAX=STACK_MAX-2;                                 2864
1880     5    3            END;                                                      2865
1881     5    2            ELSE                                                      2866
1881     5    2            DO;                                                       2867
1882     5    3               WORD(LP)=TMP_VAR;                                      2868
1883     5    3               OP(LP)=OP(LP+2);                                       2869
1884     5    3               DO I=LP+3 TO STACK_MAX;                                2870
1885     5    4                 ITEMS(I-2)=ITEMS(I);                                 2871
1886     5    4               END;                                                   2872
1887     5    3               STACK_MAX=STACK_MAX-2;                                 2873
1888     5    3            END;                                                      2874
1889     5    2         END;                                                         2875
1890     5    1         ELSE                                                         2876
1890     5    1            CALL PRINT_ERR(STMT_CH,'UNKNOWN SUBSCRIPT DETECTED');     2877
1891     5    1      END;                                                           2878
1892     5           ELSE                                                          2879
1892     5              CALL PRINT_ERR(STMT_CH,'INVALID SUBSCRIPT');               2880
                                                                                   2881
1893     5           IF STACK_PRINT_DEBUG THEN                                     2882
```

```
 STMT LEVEL NEST

1894     5            DO;                                                          2883
1895     5    1          PUT SKIP LIST('PROCESS_SUBSCRIPT EXIT');                  2884
1896     5    1          IF STACK_MAX = 0 THEN                                     2885
1897     5    1             PUT SKIP DATA(STACK_MAX);                              2886
1898     5    1          ELSE                                                      2887
1898     5    1          DO I=1 TO STACK_MAX;                                      2888
1899     5    2               PUT SKIP LIST(I,WORD(I),OP(I));                      2889
1900     5    2          END;                                                      2890
1901     5    1       END;                                                         2891
                                                                                   2892
1902     5         END PROCESS_SUBSCRIPT;                                        2893
                                                                                   2894
1903     4         END SIMPLIFY_SUB_STACK;                                       2895
                                                                                   2896
1904     3         END PARSE_EXP;                                                2897
                                                                                   2898
1905     2         PROCESS_DIM:PROC;                                             2899
                   /*********************************************************** 2900
                   *                                                         * 2900
                   * PROCESS_DIM                                             * 2900
                   *                                                         * 2900
                   * NESTING:COMPILE                                         * 2900
                   ***********************************************************/ 2900
                                                                                   2905
1906     3            DECLARE (I,NUM_OCCURS)      FIXED BINARY ALIGNED;         2906
1907     3            DECLARE ERR_PTR             FIXED BINARY ALIGNED;         2907
1908     3            DECLARE CH                  CHAR(1);                      2908
1909     3            DECLARE (LEFT_SIDE,START_VAL,TO_VAL,STEP_VAL)             2909
                                                  CHAR(80) VARYING;             2910
1910     3            DECLARE NO_OPER             BIT(1) ALIGNED;               2911
1911     3            DECLARE OFFSET              FIXED BINARY ALIGNED;         2912
1912     3            DECLARE DIM_VAR             CHAR(10);                     2913
                                                                                   2914
                    /*  EXTRACT THE DIM VARIABLE */                              2915
                                                                                   2916
1913     3         NEXT_DIM:                                                     2917
                      LEFT_SIDE='';                                              2918
1914     3            NO_OPER=TRUE;                                              2919
                                                                                   2920
1915     3            DO I=STMT_CH TO STMT_RIGHT WHILE(NO_OPER);                 2921
1916     3    1         CH=SUBSTR(STMT,I,1);                                      2922
1917     3    1         IF CH='(' | CH=' ' THEN                                   2923
1918     3    1            NO_OPER=FALSE;                                         2924
1919     3    1         ELSE                                                      2925
1919     3    1            LEFT_SIDE=LEFT_SIDE||CH;                               2926
```

```
 STMT LEVEL NEST

1920     3    1       END;                                                                    2927
                                                                                              2928
1921     3            DIM_VAR=LEFT_SIDE;                                                      2929
1922     3            OFFSET=LOOKUP_SYMBOL_TABLE(DIM_VAR);                                    2930
                                                                                              2931
1923     3            IF SYM_TYPE(OFFSET)=SS_DIM_VAR |                                        2932
1924     3               SYM_TYPE(OFFSET)=SS_STRDIM  THEN;                                    2933
1925     3            ELSE                                                                    2934
1925     3            DO;                                                                     2935
1926     3    1          CALL PRINT_ERR(STMT_CH,'VARIABLE NOT ALLOWED');                      2936
1927     3    1          RETURN;                                                              2937
1928     3    1       END;                                                                    2938
                                                                                              2939
                     /*  EXTRACT THE OCCURANCES - NUMBERS ONLY  */                           2940
                                                                                              2941
1929     3            STMT_CH=I;                                                              2942
1930     3            CALL SKIP_BLANKS;                                                       2943
                                                                                              2944
1931     3            LEFT_SIDE='';                                                           2945
1932     3            NO_OPER=TRUE;                                                           2946
                                                                                              2947
1933     3            DO I=STMT_CH TO STMT_RIGHT WHILE(NO_OPER);                              2948
1934     3    1         CH=SUBSTR(STMT,I,1);                                                 2949
1935     3    1         IF CH=')'  THEN                                                      2950
1936     3    1            NO_OPER=FALSE;                                                    2951
1937     3    1         ELSE                                                                 2952
1937     3    1            IF CH=' ' THEN ;                                                  2953
1939     3    1            ELSE                                                              2954
1939     3    1               IF CH>='0' & CH<='9' THEN                                      2955
1940     3    1                  LEFT_SIDE=LEFT_SIDE||CH;                                    2956
1941     3    1               ELSE                                                           2957
1941     3    1               DO;                                                            2958
1942     3    2                  CALL PRINT_ERR(STMT_CH,'EXPECTING NUMBER');                 2959
1943     3    2                  RETURN;                                                     2960
1944     3    2               END;                                                           2961
                                                                                              2962
1945     3    1       END;                                                                    2963
                                                                                              2964
1946     3            IF NO_OPER THEN         /* ENSURE THERE IS A ) */                       2965
1947     3            DO;                                                                     2966
1948     3    1           CALL PRINT_ERR(STMT_CH,'UNEXPECTED END OF STATEMENT');              2967
1949     3    1           RETURN;                                                             2968
1950     3    1      END;                                                                     2969
1951     3           STMT_CH=I;                                                              2970
                                                                                              2971
```

```
 STMT LEVEL NEST

1952     3           NUM_OCCURS=LEFT_SIDE;                                          2972
1953     3           IF OFFSET=SS_MAX &                                             2973
                        (SYM_TYPE(OFFSET)=SS_DIM_VAR |                              2974
                         SYM_TYPE(OFFSET)=SS_STRDIM) THEN                           2975
1954     3           DO;                                                            2976
1955     3    1          IF SS_MAX+NUM_OCCURS > HBOUND(SYMBOL,1) THEN               2977
1956     3    1          DO;                                                        2978
1957     3    2             CALL PRINT_ERR(STMT_CH,'DIM OCCURS TOO LARGE');         2979
1958     3    2             RETURN;                                                 2980
1959     3    2          END;                                                       2981
1960     3    1          SYM_DIM_MAX(SS_MAX)=NUM_OCCURS;                            2982
1961     3    1          DO I=1 TO NUM_OCCURS;                                      2983
1962     3    2             SS_MAX=SS_MAX+1;                                        2984
1963     3    2             SYMBOL(SS_MAX)=SUBSTR(DIM_VAR,1,9)||'+';                2985
1964     3    2             SYM_TYPE(SS_MAX)=SYM_TYPE(OFFSET);                      2986
1965     3    2             SYM_VALUE(SS_MAX)=0.0;                                  2987
1966     3    2             STRING_VAL(SS_MAX)='*';                                 2988
1967     3    2          END;                                                       2989
1968     3    1       END;                                                          2990
1969     3            ELSE                                                          2991
1969     3            DO;                                                           2992
1970     3    1          CALL PRINT_ERR(STMT_CH,'DUPLICATE DIM VARIABLE');          2993
1971     3    1          RETURN;                                                    2994
1972     3    1       END;                                                          2995
                                                                                    2996
1973     3            CALL SKIP_BLANKS;                                             2997
1974     3            IF STMT_CH>STMT_RIGHT THEN;                                   2998
1976     3            ELSE                                                          2999
1976     3            DO;                                                           3000
1977     3    1          CH=SUBSTR(STMT,STMT_CH,1);                                 3001
1978     3    1          IF CH=',' THEN                                             3002
1979     3    1          DO;                                                        3003
1980     3    2             STMT_CH=STMT_CH+1;                                      3004
1981     3    2             GO TO NEXT_DIM;                                         3005
1982     3    2          END;                                                       3006
1983     3    1          ELSE                                                       3007
1983     3    1          DO;                                                        3008
1984     3    2             CALL PRINT_ERR(STMT_CH,'COMMA EXPECTED');               3009
1985     3    2             RETURN;                                                 3010
1986     3    2          END;                                                       3011
                                                                                    3012
1987     3    1    END PROCESS_DIM;                                                3013
                                                                                    3014
1989     2         END COMPILE;                                                     3015
```

```
STMT LEVEL NEST

                /*********************************************************************/   3016
                /*********************************************************************/   3017
                /*********************************************************************/   3018
                /*********************************************************************/   3019
                /*********************************************************************/   3020
                                                                                          3021
1990     1      EXECUTE:PROC;                                                             3022
                                                                                          3023
                /*********************************************************************   3024
                *                                                                   *   3024
                *   THIS PROC DRIVES THE EXECUTION OF THE PSEUDO MACHINE CODE.      *   3024
                *   ERROR TRAPPING FOR THE BASIC PROGRAM AS WELL AS LIMITATIONS     *   3024
                *   ON EXECUTION TO PREVENT RUN AWAY PROGRAMS.                      *   3024
                *                                                                   *   3024
                *   ACCUM      IS THE PSEUDO COMPUTER ACCUMULATOR                   *   3024
                *   CUR_LN     IS CURRENT LINE NUMBER OF THE BASIC PGM EXECUTING    *   3024
                *   P_CTR      IS THE COUNTER OF PCODES EXECUTED                    *   3024
                *   P_CTR_MAX  IS THE MAXIMUM VALUE P_CTR CAN HAVE.  ONCE THIS      *   3024
                *              VALUE IS REACHED, THE BASIC PRORAM IS ABORTED.       *   3024
                *   P_PTR      IS THE CURRENT PC_OPCODE TO BE/CURRENTLY EXECUTING   *   3024
                *   P_PTR_SUB  IS THE CURRENT PC_OPCODE MODIFIER FOR TYPE CHECKS    *   3024
                *                                                                   *   3024
                *   GOSUB_STACK IS USED TO IMPLEMENT THE GOSUB AND RETURN STMTS     *   3024
                *   FOR_STACK   IS USED TO IMPLEMENT FOR NEXT LOOPS                 *   3024
                *                                                                   *   3024
                *   THERE IS A SPECIAL REGISTER CALLED DSL_REG (DIM SUBSCRIPT       *   3024
                *   LOCATOR) THAT IS USED TO IMPLEMENT SUBSCRIPTS.  THE DSL_REG IS  *   3024
                *   SET BY THE DSL PSUEDO INSTRUCTION USING THE CONTENTS OF ACCUM.  *   3024
                *   PRIOR TO THE EXECUTION OF THE NEXT PSEDUO INSTRUCTION AFTER THE *   3024
                *   DSL IS EXECUTED, THE VALUE OF THE DSL_REG WILL BE ADDED TO THE  *   3024
                *   OFFSET (PC_OFFSET) TO EFFECTIVLY IMPLEMENT THE SUBSCRIPT. THE   *   3024
                *   DSL_REG IS THEN SET TO ZERO. THE CONTENST OF THE PC_OFFSET      *   3024
                *   FOR THE DSL IS NOT USED NOW.                                    *   3024
                *                                                                   *   3024
                *********************************************************************   3024
                * NESTING:EXECUTE                                                   *   3024
                *********************************************************************/   3024
                                                                                          3052
                                                                                          3053
1991     2          PUT SKIP;                                                             3054
1992     2          DECLARE PC_INST(0:       38 )      LABEL;                           3055 1
1993     2          DECLARE LIB_FNC(2:       10 )     LABEL;                            3056 1
1994     2          DECLARE (P_PTR,P_PTR_SUB)         FIXED BINARY ALIGNED;            3057
1995     2          DECLARE (P_CTR,P_CTR_MAX)         FIXED BINARY ALIGNED;            3058
1996     2          DECLARE (DSL_REG,OFFSET_VAL)      FIXED BINARY ALIGNED;            3059
```

```
 STMT LEVEL NEST

1997     2             DECLARE CUR_LN                    FIXED BINARY ALIGNED;            3060
1998     2             DECLARE CUR_DF                    FIXED BINARY ALIGNED;            3061
1999     2             DECLARE (I,L,TAB_POS,TAB_AMT,PRINT_TAB_AMT,PRINT_LAST_PCT)         3062
                                                         FIXED BINARY ALIGNED;            3063
2000     2             DECLARE (ACCUM,REGISTER,COMP_A,COMP_B)                             3064
                                                         FLOAT DECIMAL;                   3065
2001     2             DECLARE ABNORMAL_STOP             BIT(1) ALIGNED INIT('0'B);       3066
2002     2             DECLARE (ACCUM_TYPE,ACCUM_STR,                                     3067
                               COMP_RESULT,                                              3068
                               COMP_A_TYPE,COMP_A_STR,                                   3069
                               COMP_B_TYPE,COMP_B_STR)  FIXED BINARY ALIGNED;            3070
2003     2             DECLARE (COMP_RESULT_LT           INITIAL(1),                      3071
                               COMP_RESULT_EQ           INITIAL(2),                      3072
                               COMP_RESULT_GT           INITIAL(3))                      3073
                                                         STATIC FIXED BINARY ALIGNED;     3074
2004     2             DECLARE 1 GOSUB_STACK       ALIGNED,                               3075
                                 2 (GS_CURM,                                             3076
                                    GS_MAX)        FIXED BINARY,                         3077
                                 2  GOSUB_AREA(25),                                      3078
                                    3  GS_LINE     FIXED BINARY,                         3079
                                    3  GS_PTR      FIXED BINARY;                         3080
2005     2              DECLARE 1 FOR_STACK        ALIGNED,                               3081
                                  2 (FS_CUR,                                             3082
                                     FS_MAX)       FIXED BINARY,                         3083
                                  2  FS_AREA(10),                                        3084
                                     3  FS_CTL_VAR FIXED BINARY,                         3085
                                    (3  FS_START,                                        3086
                                     3  FS_LIMIT,                                        3087
                                     3  FS_STEP)   FLOAT DECIMAL,                        3088
                                     3  FS_INST    FIXED BINARY;                         3089
                                                                                         3090
2006     2              DECLARE PRINT_WORK               PICTURE '(6)-9.V(6)9',          3091
                               PRINT_WORK_CHAR(14) DEFINED PRINT_WORK                    3092
                                                   CHAR(1);                              3093
2007     2              DECLARE PRINT_E_FORMAT           CHAR(14);                       3094
2008     2              DECLARE PRINT_ZERO               CHAR(2) VARYING INITIAL(' 0');  3095
                                                                                         3096
2009     2              DECLARE PRINT_FIELD        CHAR(14) VARYING;                     3097
2010     2              DECLARE 1 PRINT_AREA,                                            3098
                                  2  PRINT_LINE    CHAR(120) VARYING;                    3099
                                                                                         3100
2011     2              P_CTR,P_PTR = 0;                                                 3101
2012     2              P_CTR_MAX =      5000 ;                                          3102 1
2013     2              ACCUM,REGISTER=0.0;                                              3103
2014     2              ACCUM_TYPE=0;                                                    3104
```

```
 STMT LEVEL NEST

2015     2              COMP_A,COMP_B=0.0;                                          3105
2016     2              COMP_A_TYPE,COMP_B_TYPE,COMP_A_STR,COMP_B_STR=0;            3106
2017     2              GS_CUR,GS_MAX=0;                                            3107
2018     2              FS_CUR,FS_MAX=0;                                            3108
2019     2              DSL_REG=0;                                                  3109
2020     2              CUR_DEF=0;                                                  3110
                                                                                   3111
2021     2              PRINT_LINE='';                                              3112
2022     2              PRINT_TAB_AMT=0;                                            3113
2023     2              PRINT_LAST_PCT=0;                                           3114
                                                                                   3115
2024     2              ON ERROR                                                    3116
2025     2              BEGIN;                                                      3117
2026     3                  PUT SKIP LIST('FATAL BASIC INTERPRETER ERROR');         3118
2027     3                  PUT SKIP DATA(P_CTR,P_PTR);                             3119
2028     3                  PUT SKIP DATA(PC_OPCODE(P_PTR),PC_OBJECT(P_PTR));       3120
2029     3                  PUT SKIP DATA(DSL_REG,OFFSET_VAL);                      3121
2030     3                  PUT SKIP DATA(ACCUM,REGISTER);                          3122
2031     3                  PUT SKIP DATA(COMP_A,COMP_B,COMP_A_TYPE,COMP_B_TYPE);   3123
2032     3                  PUT SKIP DATA(GS_CUR,GS_MAX);                           3124
2033     3                  PUT SKIP DATA(FS_CUR,FS_MAX);                           3125
2034     3                  TABLE_DUMP=TRUE;                                        3126
2035     3                  CALL TERMINATE;                                         3127
2036     3                  STOP;                                                   3128
2037     3              END;                                                        3129
2038     2         P_CODE_NEXT:                                                     3130
                        P_PTR=P_PTR+1;                                             3131
2039     2         P_CODE_JUMP:                                                     3132
                        P_CTR=P_CTR+1;                                             3133
                                                                                   3134
2040     2              IF ABNORMAL_STOP THEN                                       3135
2041     2                  RETURN;                                                 3136
                                                                                   3137
2042     2              IF P_CTR>P_CTR_MAX THEN                                     3138
2043     2              DO;                                                         3139
2044     2    1            CALL PRINT_ERR('**** PROGRAM ABORTED AFTER EXECUTING ' ||  3140
                                    P_CTR_MAX ||' INSTRUCTIONS ****');             3141
2045     2    1            RETURN;                                                 3142
2046     2    1         END;                                                       3143
                                                                                   3144
2047     2              IF P_PTR>PC_MAX THEN                                        3145
2048     2              DO;                                                         3146
2049     2    1            CALL PRINT_ERR('**** PROGRAM RUN AWAY DETECTED ****');  3147
2050     2    1            RETURN;                                                 3148
2051     2    1         END;                                                       3149
```

```
STMT LEVEL NEST

                                                                                3150
                  /*********************************************************************  3151
                  *                                                                   *  3151
                  *  SUBSCRIPT CHECKING PATCH STARTS HERE                             *  3151
                  *                                                                   *  3151
                  *********************************************************************/ 3151
                                                                                3155
2052     2             IF DSL_REG > 0 THEN                                      3156
2053     2                IF DSL_REG > SYM_DIM_MAX(PC_OBJECT(P_PTR)) THEN       3157
2054     2                DO;                                                   3158
2055     2    1               CALL PRINT_ERR('**** SUBSCRIPT TOO LARGE ****');  3159
2056     2    1               RETURN;                                           3160
2057     2    1           END;                                                  3161
2058     2             ELSE                                                     3162
2058     2                IF DSL_REG < 0  THEN                                  3163
2059     2                DO;                                                   3164
2060     2    1               CALL PRINT_ERR('**** SUBSCRIPT ENCOUNTERED ****'); 3165
2061     2    1               RETURN;                                           3166
2062     2    1           END;                                                  3167
                  /*********************************************************************  3168
                  *                                                                   *  3168
                  *  SUBSCRIPT CHECKING PATCH ENDS HERE                               *  3168
                  *                                                                   *  3168
                  *********************************************************************/ 3168
                                                                                3172
                                                                                3173
2063     2             OFFSET_VAL=PC_OBJECT(P_PTR)+DSL_REG;                     3174
2064     2             IF EXECUTION_DEBUG THEN                                  3175
2065     2                  PUT SKIP DATA(OFFSET_VAL,P_PTR,DSL_REG,PC_OBJECT(P_PTR)); 3176
2066     2             DSL_REG=0;                                               3177
                                                                                3178
                  /*********************************************************************  3179
                  *                                                                   *  3179
                  *  ENFORCE OBJECT TYPING IF OPCODE REQUIRES IT                      *  3179
                  *                                                                   *  3179
                  *********************************************************************/ 3179
                                                                                3183
                            /*                                                  3184 1
2067     2              SELECT (PC_ALLOW(PC_OPCODE(P_PTR)))   */ DO;             3184 1
                                    /*                                          3185 1
                       WHEN ('00'B)    */                                       3185 1
2068     2    1                IF (PC_ALLOW(PC_OPCODE(P_PTR)))=('00'B) THEN     3185 1
2069     2    1                          DO;                                    3185 1
2070     2    2            P_PTR_SUB = 0;                                       3186
2071     2    2                     GO TO  ENDSELECT_MACRO6; END; /*            3187 1
```

```
 STMT LEVEL NEST

                        WHEN ('01'B)    */                                                3187 1
2073     2    1                  IF (PC_ALLOW(PC_OPCODE(P_PTR)))=('01'B) THEN            3187 1
2074     2    1                             DO;                                          3187 1
2075     2    2            IF SYM_TYPE(PC_OBJECT(P_PTR)) < SS_STRCON THEN                3188
2076     2    2               P_PTR_SUB = 1;                                             3189
2077     2    2            ELSE                                                          3190
2077     2    2            DO;                                                           3191
2078     2    3               CALL PRINT_ERR('**** STRING NOT ALLOWED ****');            3192
2079     2    3               RETURN;                                                    3193
2080     2    3            END;                                                          3194
2081     2    2                        GO TO  ENDSELECT_MACRO6; END; /*                  3195 1
                        WHEN ('10'B)    */                                              3195 1
2083     2    1                  IF (PC_ALLOW(PC_OPCODE(P_PTR)))=('10'B) THEN            3195 1
2084     2    1                             DO;                                          3195 1
2085     2    2            IF SYM_TYPE(PC_OBJECT(P_PTR)) >= SS_STRCON THEN               3196
2086     2    2                P_PTR_SUB = 2;                                           3197
2087     2    2            ELSE                                                          3198
2087     2    2            DO;                                                           3199
2088     2    3               CALL PRINT_ERR('**** NUMBER NOT ALLOWED ****');            3200
2089     2    3               RETURN;                                                    3201
2090     2    3            END;                                                          3202
2091     2    2                        GO TO  ENDSELECT_MACRO6; END; /*                  3203 1
                        WHEN ('11'B)    */                                              3203 1
2093     2    1                  IF (PC_ALLOW(PC_OPCODE(P_PTR)))=('11'B) THEN            3203 1
2094     2    1                             DO;                                          3203 1
2095     2    2            IF SYM_TYPE(PC_OBJECT(P_PTR)) < SS_STRCON THEN                3204
2096     2    2               P_PTR_SUB = 1;                                             3205
2097     2    2            ELSE                                                          3206
2097     2    2               P_PTR_SUB = 2;                                             3207
2098     2    2                           END; /*                                       3208 1
2099     2    1          OTHERWISE     */  ELSE DO;                                     3208 1
                                                                                        3208 1
2100     2    2            CALL PRINT_ERR                                              3209
                               ('*** FATAL ERROR - TYPE ENFORCEMENT FAILED ***');       3210
2101     2    2            SIGNAL ERROR;                                               3211
2102     2    2                        END; /*                                         3212 1
2103     2    1          ENDSELECT     */  END; ENDSELECT_MACRO6:;                      3212 1
                                                                                        3214
2105     2              GO TO PC_INST(PC_OPCODE(P_PTR));                               3215
                                                                                        3216
                     /*     PCODE SLN - SET LINE NUMBER    */                          3217
                                                                                        3218
2106     2          PC_INST(0):                                                        3219
                        CUR_LN=LS_LINE(PC_OBJECT(P_PTR));                               3220
2107     2              IF EXECUTION_DEBUG THEN                                        3221
```

```
 STMT LEVEL NEST

2108     2                PUT SKIP DATA(CUR_LN);                                    3222
2109     2             GO TO P_CODE_NEXT;                                           3223
                                                                                    3224
                    /*     PCODE LDA - LOAD ACCUMULATOR    */                       3225
                                                                                    3226
2110     2          PC_INST(1):                                                     3227
                       ACCUM=SYM_VALUE(OFFSET_VAL);                                 3228
2111     2             ACCUM_TYPE=SYM_TYPE(OFFSET_VAL);                            3229
2112     2             ACCUM_STR=OFFSET_VAL;                                       3230
2113     2             IF EXECUTION_DEBUG THEN                                     3231
2114     2                PUT SKIP DATA(ACCUM,                                     3232
                                  SYMBOL(OFFSET_VAL),SYM_TYPE(OFFSET_VAL));       3233
2115     2             GO TO P_CODE_NEXT;                                          3234
                                                                                    3235
                    /*     PCODE LDR - LOAD REGISTER       */                       3236
                                                                                    3237
2116     2          PC_INST(33):                                                    3238
                       REGISTER=SYM_VALUE(OFFSET_VAL);                              3239
2117     2             IF EXECUTION_DEBUG THEN                                     3240
2118     2                PUT SKIP DATA(REGISTER,SYMBOL(OFFSET_VAL));              3241
2119     2             GO TO P_CODE_NEXT;                                          3242
                                                                                    3243
                    /*     PCODE STA - STORE ACCUMULATOR    */                      3244
                                                                                    3245
2120     2          PC_INST(2):                                                     3246
                       IF ACCUM_TYPE >= SS_STRCON THEN                              3247
2121     2                IF SYM_TYPE(OFFSET_VAL) = SS_STRVAR |                     3248
                             SYM_TYPE(OFFSET_VAL) = SS_STRDIM THEN                  3249
2122     2                    STRING_VAL(OFFSET_VAL)=STRING_VAL(ACCUM_STR);          3250
2123     2                ELSE                                                      3251
2123     2                DO;                                                       3252
2124     2    1              CALL PRINT_ERR('**** STRING CANNOT BE STORED ' ||     3253
                                        'IN A NUMERIC VARIABLE ****');              3254
2125     2    1              RETURN;                                                3255
2126     2    1           END;                                                      3256
2127     2             ELSE                                                       3257
2127     2                IF SYM_TYPE(OFFSET_VAL) < SS_STRCON THEN                  3258
2128     2                  SYM_VALUE(OFFSET_VAL)=ACCUM;                            3259
2129     2                ELSE                                                      3260
2129     2                   DO;                                                    3261
2130     2    1                 CALL PRINT_ERR('**** NUMBER CANNOT BE STORED ' || 3262
                                          'IN A STRING VARIABLE ****');             3263
2131     2    1                 RETURN;                                            3264
2132     2    1              END;                                                  3265
2133     2             IF EXECUTION_DEBUG THEN                                    3266
```

```
 STMT LEVEL NEST

2134     2                PUT SKIP DATA(ACCUM,SYMBOL_AREA(OFFSET_VAL),ACCUM_STR);          3267
                                 /* SYMBOL(OFFSET_VAL),SYM_TYPE(OFFSET_VAL)*/          3268
2135     2             GO TO P_CODE_NEXT;                                                3269
                                                                                         3270
                    /*     PCODE STR - STORE REGISTER       */                         3271
                                                                                         3272
2136     2          PC_INST(34):                                                         3273
                         SYM_VALUE(OFFSET_VAL)=REGISTER;                                 3274
2137     2             IF EXECUTION_DEBUG THEN                                           3275
2138     2                PUT SKIP DATA(REGISTER,SYMBOL(OFFSET_VAL));                    3276
2139     2             GO TO P_CODE_NEXT;                                                3277
                                                                                         3278
                    /*     PCODE EXP - RAISE ACCUMULATOR    */                         3279
                                                                                         3280
2140     2          PC_INST(3):                                                          3281
                         ACCUM=ACCUM**SYM_VALUE(OFFSET_VAL);                             3282
2141     2             IF EXECUTION_DEBUG THEN                                           3283
2142     2                PUT SKIP DATA(ACCUM,SYMBOL(OFFSET_VAL));                       3284
2143     2             GO TO P_CODE_NEXT;                                                3285
                                                                                         3286
                    /*     PCODE ADD - ADD TO ACCUMULATOR    */                         3287
                                                                                         3288
2144     2          PC_INST(4):                                                          3289
                         ACCUM=ACCUM+SYM_VALUE(OFFSET_VAL);                              3290
2145     2             IF EXECUTION_DEBUG THEN                                           3291
2146     2                PUT SKIP DATA(ACCUM);                                          3292
2147     2             GO TO P_CODE_NEXT;                                                3293
                                                                                         3294
                    /*     PCODE SUB - SUBTRACT FROM ACCUMULATOR    */                 3295
                                                                                         3296
2148     2          PC_INST(5):                                                          3297
                         ACCUM=ACCUM-SYM_VALUE(OFFSET_VAL);                              3298
2149     2             IF EXECUTION_DEBUG THEN                                           3299
2150     2                PUT SKIP DATA(ACCUM);                                          3300
2151     2             GO TO P_CODE_NEXT;                                                3301
                                                                                         3302
                    /*     PCODE MUL - MULTIPLY ACCUMULATOR    */                     3303
                                                                                         3304
2152     2          PC_INST(6):                                                          3305
                         ACCUM=ACCUM*SYM_VALUE(OFFSET_VAL);                              3306
2153     2             IF EXECUTION_DEBUG THEN                                           3307
2154     2                PUT SKIP DATA(ACCUM);                                          3308
2155     2             GO TO P_CODE_NEXT;                                                3309
                                                                                         3310
                                                                                         3311
```

```
STMT LEVEL NEST
                    /*     PCODE DIV - DIVIDE ACCUMULATOR    */                                3312
                                                                                                3313
2156     2          PC_INST(7):                                                                 3314
                         IF SYM_VALUE(OFFSET_VAL)=0.0 THEN                                      3315
2157     2                   DO;                                                                3316
2158     2    1                 CALL PRINT_ERR('**** DIVISION BY ZERO DETECTED ****');          3317
2159     2    1                 RETURN;                                                         3318
2160     2    1              END;                                                               3319
2161     2                   ACCUM=ACCUM/SYM_VALUE(OFFSET_VAL);                                 3320
2162     2                   IF EXECUTION_DEBUG THEN                                            3321
2163     2                      PUT SKIP DATA(ACCUM);                                           3322
2164     2                   GO TO P_CODE_NEXT;                                                 3323
                                                                                                3324
                    /*     PCODE RDV - READ VARIABLE    */                                     3325
                                                                                                3326
2165     2          PC_INST(8):                                                                 3327
                         DS_CUR=DS_CUR+1;                                                       3328
2166     2                   IF DS_CUR > DS_MAX THEN                                            3329
2167     2                   DO;                                                                3330
2168     2    1                 CALL PRINT_ERR('*** NO DATA FOR ' ||                            3331
                                          SYMBOL(OFFSET_VAL));                                  3332
2169     2    1                 RETURN;                                                         3333
2170     2    1              END;                                                               3334
2171     2                   IF P_PTR_SUB = 1 THEN                                              3335
2172     2                      SYM_VALUE(OFFSET_VAL) = DS_ITEM(DS_CUR);                        3336
2173     2                   ELSE                                                               3337
2173     2                      STRING_VAL(OFFSET_VAL) = STRING_VAL(DS_STR(DS_CUR));            3338
2174     2                   GO TO P_CODE_NEXT;                                                 3339
                                                                                                3340
                    /*     PCODE PRV - PRINT VARIABLE    */                                    3341
                                                                                                3342
2175     2          PC_INST(9):                                                                 3343
                         IF P_PTR_SUB = 2 THEN     /*  IF ARGUMENT IS A STRING, GO TO */    3344
2176     2                    GO TO PC_INST(16);    /*     PRS TO PRINT IT             */    3345
                                                                                                3346
2177     2                   PRINT_FIELD=FORMAT_NUMBER(SYM_VALUE(OFFSET_VAL));                  3347
2178     2                   CALL PRINT_BUFFER(PRINT_FIELD);                                    3348
2179     2                   GO TO P_CODE_NEXT;                                                 3349
                                                                                                3350
                    /*     PCODE PRS - PRINT STRING   */                                       3351
                                                                                                3352
2180     2          PC_INST(16):                                                                3353
                                                                                                3354
                         CALL PRINT_BUFFER(STRING_VAL(PC_OBJECT(P_PTR)));                       3355
                                                                                                3356
```

```
 STMT LEVEL NEST

2181     2              GO TO P_CODE_NEXT;                                                    3357
                                                                                              3358
                     /*     PCODE PCT - PRINT CONTROL      */                                3359
                                                                                              3360
2182     2           PC_INST(10):                                                             3361
                                 /*                                                           3362 1
                           SELECT (PC_OBJECT(P_PTR))   */ DO;                                 3362 1
                                                  /*                                         3363 1
                           WHEN (PCT_LFEED)    */                                            3363 1
2183     2    1                    IF (PC_OBJECT(P_PTR))=(PCT_LFEED) THEN                    3363 1
2184     2    1                                  DO;                                        3363 1
2185     2    2              CALL FLUSH_BUFFER;                                              3364
2186     2    2              PRINT_TAB_AMT=0;                                                3365
2187     2    2              PRINT_LAST_PCT=0;                                               3366
2188     2    2                           GO TO  ENDSELECT_MACRO7; END; /*                  3367 1
                           WHEN (PCT_TAB)    */                                            3367 1
2190     2    1                    IF (PC_OBJECT(P_PTR))=(PCT_TAB) THEN                      3367 1
2191     2    1                                  DO;                                        3367 1
2192     2    2              PRINT_LAST_PCT=PCT_TAB;                                         3368
2193     2    2                           GO TO  ENDSELECT_MACRO7; END; /*                  3369 1
                           WHEN (PCT_NOTAB)    */                                            3369 1
2195     2    1                    IF (PC_OBJECT(P_PTR))=(PCT_NOTAB) THEN                    3369 1
2196     2    1                                  DO;                                        3369 1
2197     2    2              PRINT_LAST_PCT=PCT_NOTAB;                                       3370
2198     2    2                            END; /*                                          3371 1
2199     2    1           OTHERWISE     */  ELSE DO;                                        3371 1
                                                                                              3371 1
2200     2    2           DO;                                                                3372
2201     2    3              CALL PRINT_ERR                                                  3373
                               ('**** UNDEFINED PRINT CONTROL VALUE ****');                   3374
2202     2    3              RETURN;                                                         3375
2203     2    3           END;                                                               3376
2204     2    2                           END; /*                                           3377 1
2205     2    1            ENDSELECT     */  END; ENDSELECT_MACRO7:;                        3377 1
2207     2              GO TO P_CODE_NEXT;                                                    3378
                                                                                              3379
                     /*     PCODE FNC - FUNCTION CALL  */                                    3380
                                                                                              3381
                     /*****************************************************************     3382
                     *                                                               *      3382
                     * IMPORTANT NOTE - IF ANY CHANGES ARE MADE TO LIBRARY FUNCTIONS,    *   3382
                     * THIS PCODE FNC SHOULD MATCH THE CHANGES IN THE INITIALIZE PROC    *   3382
                     *                                                               *      3382
                     *****************************************************************/     3382
                                                                                              3387
```

```
STMT LEVEL NEST
                                                                                                    3388
2208     2         PC_INST(11):                                                                     3389
                       IF PC_OBJECT(P_PTR) < LBOUND(LIB_FNC,1) |                                    3390
                          PC_OBJECT(P_PTR) > HBOUND(LIB_FNC,1) THEN                                 3391
2209     2              DO;                                                                         3392
2210     2    1             PUT SKIP(2) LIST('**** FATAL ERROR - UNDEFINED FUNCTION ',             3393
                                    SYMBOL(PC_OBJECT(P_PTR)),' ****');                              3394
2211     2    1             SIGNAL ERROR;                                                          3395
2212     2    1         END;                                                                       3396
2213     2              IF EXECUTION_DEBUG THEN                                                   3397
2214     2                 PUT SKIP LIST('FNC',SYMBOL(PC_OBJECT(P_PTR)),                          3398
                                        SYM_VALUE(PC_OBJECT(P_PTR)),                            3399
                                        ACCUM);                                                 3400
2215     2              GO TO LIB_FNC(PC_OBJECT(P_PTR));                                          3401
2216     2         LIB_FNC(2):                                                                    3402
                       IF ACCUM < 0.0 THEN                                                      3403
2217     2              DO;                                                                       3404
2218     2    1           CALL PRINT_ERR                                                        3405
                             ('**** NEGATIVE VALUE IN SQR FUNCTION ****');                      3406
2219     2    1           RETURN;                                                               3407
2220     2    1         END;                                                                    3408
2221     2              ACCUM=SQRT(ACCUM);                                                      3409
2222     2              SYM_VALUE(PC_OBJECT(P_PTR))=ACCUM;                                      3410
2223     2              GO TO END_FNC;                                                          3411
2224     2         LIB_FNC(3):                                                                  3412
                       ACCUM=ABS(ACCUM);                                                        3413
2225     2              SYM_VALUE(PC_OBJECT(P_PTR))=ACCUM;                                      3414
2226     2              GO TO END_FNC;                                                          3415
2227     2         LIB_FNC(4):                                                                  3416
                       PRINT_TAB_AMT=ACCUM;                                                     3417
2228     2              SYM_VALUE(PC_OBJECT(P_PTR))=ACCUM;                                      3418
2229     2              IF PRINT_TAB_AMT<1 | PRINT_TAB_AMT>120 THEN                             3419
2230     2              DO;                                                                     3420
2231     2    1           CALL PRINT_ERR                                                        3421
                             ('**** INVALID VALUE IN TAB FUNCTION ****');                       3422
2232     2    1           RETURN;                                                               3423
2233     2    1         END;                                                                    3424
2234     2              GO TO END_FNC;                                                          3425
2235     2         LIB_FNC(5):                                                                  3426
                       ACCUM=TRUNC(ACCUM);                                                      3427
2236     2              SYM_VALUE(PC_OBJECT(P_PTR))=ACCUM;                                      3428
2237     2              GO TO END_FNC;                                                          3429
2238     2         LIB_FNC(6):                                                                  3430
                       ACCUM=COS(ACCUM);                                                        3431
2239     2              SYM_VALUE(PC_OBJECT(P_PTR))=ACCUM;                                      3432
```

```
 STMT LEVEL NEST

2240     2              GO TO END_FNC;                                          3433
2241     2          LIB_FNC(7):                                                 3434
                        ACCUM=SIN(ACCUM);                                      3435
2242     2              SYM_VALUE(PC_OBJECT(P_PTR))=ACCUM;                      3436
2243     2              GO TO END_FNC;                                          3437
2244     2          LIB_FNC(8):                                                 3438
                        ACCUM=TAN(ACCUM);                                      3439
2245     2              SYM_VALUE(PC_OBJECT(P_PTR))=ACCUM;                      3440
2246     2              GO TO END_FNC;                                          3441
2247     2          LIB_FNC(9):                                                 3442
                        ACCUM=RND(ACCUM);                                      3443
2248     2              SYM_VALUE(PC_OBJECT(P_PTR))=ACCUM;                      3444
2249     2              GO TO END_FNC;                                          3445
2250     2          LIB_FNC(10):                                                3446
                        IF ACCUM>=0.0 THEN                                     3447
2251     2                 ACCUM=TRUNC(ACCUM+0.5);                             3448
2252     2              ELSE                                                    3449
2252     2                 ACCUM=TRUNC(ACCUM-0.5);                             3450
2253     2              SYM_VALUE(PC_OBJECT(P_PTR))=ACCUM;                      3451
                      /* GO TO END_FNC; */                                     3452
2254     2          END_FNC:                                                    3453
                        IF EXECUTION_DEBUG THEN                                3454
2255     2                 PUT SKIP LIST('FNC',SYMBOL(PC_OBJECT(P_PTR)),       3455
                                        SYM_VALUE(PC_OBJECT(P_PTR)),           3456
                                        ACCUM);                                3457
2256     2              GO TO P_CODE_NEXT;                                      3458
                                                                               3459
                                                                               3460
                   /*     PCODE END - END OF EXECUTION  */                     3461
                                                                               3462
2257     2          PC_INST(12):                                                3463
                        CALL FLUSH_BUFFER;                                     3464
2258     2              PUT SKIP(2) EDIT('**** PROGRAM EXECUTION COMPLETE - ',  3465
                                        P_CTR, ' INSTRUCTIONS EXECUTED ****')  3466
                                        (A,F(8),A);                            3467
2259     2              RETURN;                                                 3468
                                                                               3469
                   /*     PCODE B - BRANCH   */                                3470
                                                                               3471
2260     2          PC_INST(13):                                                3472
                   COMMON_BRANCH:                                              3473
                                                                               3474
                        DO LS_CUR=1 TO LS_MAX;                                 3475
2261     2    1             IF LS_LINE(LS_CUR)=PC_OBJECT(P_PTR) THEN            3476
2262     2    1             DO;                                                 3477
```

```
 STMT LEVEL NEST

2263     2    2              P_PTR=LS_OFFSET(LS_CUR);                                    3478
2264     2    2              GOTO P_CODE_JUMP;                                           3479
2265     2    2            END;                                                          3480
2266     2    1         END;                                                             3481
2267     2              CALL PRINT_ERR('**** LINE ' || PC_OBJECT(P_PTR) ||              3482
                                    ' NOT FOUND');                                     3483
2268     2              RETURN;                                                         3484
                                                                                       3485
                     /*     PCODE BAL - BRANCH AND LINK    */                         3486
                                                                                       3487
2269     2           PC_INST(14):                                                       3488
                                                                                       3489
                       IF EXECUTION_DEBUG THEN                                         3490
2270     2                PUT SKIP LIST('BRANCH AND LINK TO',PC_OBJECT(P_PTR));        3491
2271     2              IF GS_MAX >= HBOUND(GS_LINE,1) THEN                             3492
2272     2              DO;                                                             3493
2273     2    1            CALL PRINT_ERR('**** TOO MANY ACTIVE GOSUBS ****');         3494
2274     2    1            RETURN;                                                     3495
2275     2    1         END;                                                           3496
2276     2              IF GS_MAX > 0 THEN                                              3497
2277     2              DO;                                                             3498
2278     2    1            DO GS_CUR=1 TO GS_MAX;                                      3499
2279     2    2               IF GS_LINE(GS_CUR)=CUR_LN THEN                           3500
2280     2    2               DO;                                                      3501
2281     2    3                  CALL PRINT_ERR('**** RECURSIVE GOSUB ****');          3502
2282     2    3                  RETURN;                                               3503
2283     2    3               END;                                                     3504
2284     2    2            END;                                                        3505
2285     2    1         END;                                                           3506
2286     2              DO LS_CUR=1 TO LS_MAX;                                          3507
2287     2    1             IF LS_LINE(LS_CUR)=PC_OBJECT(P_PTR) THEN                  3508
2288     2    1             DO;                                                       3509
2289     2    2               GS_MAX=GS_MAX+1;                                        3510
2290     2    2               GS_LINE(GS_MAX)=CUR_LN;                                 3511
2291     2    2               GS_PTR(GS_MAX)=P_PTR;                                   3512
2292     2    2               P_PTR=LS_OFFSET(LS_CUR);                                3513
2293     2    2               PUT SKIP DATA(P_PTR);                                   3514
2294     2    2               GOTO P_CODE_JUMP;                                       3515
2295     2    2             END;                                                      3516
2296     2    1         END;                                                           3517
2297     2              CALL PRINT_ERR('**** LINE '||PC_OBJECT(P_PTR)||' NOT FOUND ****'); 3518
2298     2              RETURN;                                                         3519
                                                                                       3520
                     /*     PCODE RET - RETURN TO LINK    */                         3521
                                                                                       3522
```

```
 STMT LEVEL NEST

2299     2         PC_INST(15):                                                    3523
                                                                                   3524
                       IF GS_MAX=0 THEN                                            3525
2300     2             DO;                                                         3526
2301     2    1           CALL PRINT_ERR('**** RETURN WITHOUT A GOSUB ****');      3527
2302     2    1           RETURN;                                                  3528
2303     2    1         END;                                                       3529
2304     2              P_PTR=GS_PTR(GS_MAX);                                       3530
2305     2              GS_MAX=GS_MAX-1;                                            3531
                                                                                   3532
2306     2              GO TO P_CODE_NEXT;                                          3533
                                                                                   3534
                    /*     PCODE PRS - PRINT STRING   */                           3535
                                                                                   3536
                    /*  PC_INST(16):   MOVED TO FOLLOW PRV CODE 9 */               3537
                                                                                   3538
                    /*     PCODE LCA - LOAD COMPARATOR A   */                      3539
                                                                                   3540
2307     2         PC_INST(17):                                                    3541
                       COMP_A=SYM_VALUE(OFFSET_VAL);                               3542
2308     2              COMP_A_TYPE=SYM_TYPE(OFFSET_VAL);                           3543
2309     2              COMP_A_STR=OFFSET_VAL;                                      3544
2310     2              IF EXECUTION_DEBUG THEN                                     3545
2311     2                 PUT SKIP DATA(P_PTR_SUB,COMP_A,COMP_A_TYPE,COMP_A_STR);  3546
2312     2              GO TO P_CODE_NEXT;                                          3547
                                                                                   3548
                    /*     PCODE LCB - LOAD COMPARATOR B   */                      3549
                                                                                   3550
2313     2         PC_INST(18):                                                    3551
                                                                                   3552
                       COMP_B=SYM_VALUE(OFFSET_VAL);                               3553
2314     2              COMP_B_TYPE=SYM_TYPE(OFFSET_VAL);                           3554
2315     2              COMP_B_STR=OFFSET_VAL;                                      3555
2316     2              IF EXECUTION_DEBUG THEN                                     3556
2317     2                 PUT SKIP DATA(P_PTR_SUB,COMP_B,COMP_B_TYPE,COMP_B_STR);  3557
2318     2              GO TO P_CODE_NEXT;                                          3558
                                                                                   3559
                    /*     PCODE BEQ - BRANCH IF A=B   */                          3560
                                                                                   3561
2319     2         PC_INST(19):                                                    3562
                                                                                   3563
                       CALL COMPARE_RTN;                                           3564
2320     2              IF COMP_RESULT=COMP_RESULT_EQ THEN                          3565
2321     2                 GO TO COMMON_BRANCH;                                     3566
2322     2              ELSE                                                        3567
```

```
 STMT LEVEL NEST

2322     2                  GO TO P_CODE_NEXT;                                  3568
                                                                                3569
                    /*     PCODE BNE - BRANCH IF A<>B   */                      3570
                                                                                3571
2323     2          PC_INST(20):                                                3572
                                                                                3573
                         CALL COMPARE_RTN;                                      3574
2324     2               IF COMP_RESULT=COMP_RESULT_EQ THEN                     3575
2325     2                  GO TO P_CODE_NEXT;                                  3576
2326     2               ELSE                                                   3577
2326     2                  GO TO COMMON_BRANCH;                                3578
                                                                                3579
                                                                                3580
                    /*     PCODE BGT - BRANCH IF A>B   */                       3581
                                                                                3582
2327     2          PC_INST(21):                                                3583
                                                                                3584
                         CALL COMPARE_RTN;                                      3585
2328     2               IF COMP_RESULT=COMP_RESULT_GT THEN                     3586
2329     2                  GO TO COMMON_BRANCH;                                3587
2330     2               ELSE                                                   3588
2330     2                  GO TO P_CODE_NEXT;                                  3589
                                                                                3590
                                                                                3591
                    /*     PCODE BLT - BRANCH IF A<B   */                       3592
                                                                                3593
2331     2          PC_INST(22):                                                3594
                                                                                3595
                         CALL COMPARE_RTN;                                      3596
2332     2               IF COMP_RESULT=COMP_RESULT_LT THEN                     3597
2333     2                  GO TO COMMON_BRANCH;                                3598
2334     2               ELSE                                                   3599
2334     2                  GO TO P_CODE_NEXT;                                  3600
                                                                                3601
                                                                                3602
                    /*     PCODE BGE - BRANCH IF A>=B   */                      3603
                                                                                3604
2335     2          PC_INST(23):                                                3605
                                                                                3606
                         CALL COMPARE_RTN;                                      3607
2336     2               IF COMP_RESULT=COMP_RESULT_GT |                        3608
                            COMP_RESULT=COMP_RESULT_EQ THEN                     3609
2337     2                  GO TO COMMON_BRANCH;                                3610
2338     2               ELSE                                                   3611
2338     2                  GO TO P_CODE_NEXT;                                  3612
```

```
STMT LEVEL NEST
                                                                                                                    3613
                                                                                                                    3614
                    /*     PCODE BLE - BRANCH IF A<=B   */                                                          3615
                                                                                                                    3616
2339     2          PC_INST(24):                                                                                    3617
                                                                                                                    3618
                         CALL COMPARE_RTN;                                                                          3619
2340     2           IF COMP_RESULT=COMP_RESULT_LT |                                                               3620
                        COMP_RESULT=COMP_RESULT_EQ THEN                                                             3621
2341     2             GO TO COMMON_BRANCH;                                                                       3622
2342     2           ELSE                                                                                          3623
2342     2             GO TO P_CODE_NEXT;                                                                         3624
                                                                                                                    3625
2343     2           GO TO P_CODE_NEXT;                                                                            3626
                                                                                                                    3627
                    /*     PCODE FSU - FOR NEXT SETUP   */                                                         3628
                                                                                                                    3629
2344     2          PC_INST(25):                                                                                    3630
                                                                                                                    3631
                         IF FS_MAX = 0 THEN   /* SKIP IF NO ACTIVE FORS */                                       3632
2345     2             FS_CUR,FS_MAX=1;                                                                           3633
2346     2           ELSE                                                                                          3634
2346     2           DO;                                                                                           3635
2347     2    1        FS_CUR=1;                                                                                  3636
2348     2    1        DO WHILE(FS_CUR<=FS_MAX);                                                                  3637
2349     2    2          IF FS_CTL_VAR(FS_CUR)=PC_OBJECT(P_PTR) THEN /* FOUND IT */                              3638
2350     2    2             GO TO RECYCLE_FOR;                                                                    3639
2351     2    2          ELSE                                                                                     3640
2351     2    2             FS_CUR=FS_CUR+1;                                                                      3641
2352     2    2        END;                                                                                       3642
2353     2    1        IF FS_MAX=HBOUND(FS_CTL_VAR,1) THEN                                                        3643
2354     2    1        DO;                                                                                        3644
2355     2    2           CALL PRINT_ERR('**** TOO MANY FOR NEXT LOOPS ****');                                    3645
2356     2    2           RETURN;                                                                                 3646
2357     2    2        END;                                                                                       3647
2358     2    1        FS_MAX=FS_MAX+1;                                                                           3648
2359     2    1        FS_CUR=FS_MAX;                                                                             3649
2360     2    1      END;                                                                                         3650
2361     2          RECYCLE_FOR:                                                                                  3651
                      FS_CTL_VAR(FS_CUR)=PC_OBJECT(P_PTR);                                                      3652
2362     2           FS_START(FS_CUR),FS_LIMIT(FS_CUR),FS_STEP(FS_CUR)=0;                                         3653
2363     2           FS_INST(FS_CUR)=0;                                                                           3654
2364     2           IF EXECUTION_DEBUG THEN                                                                      3655
2365     2             PUT SKIP DATA(FS_AREA(FS_CUR));                                                            3656
2366     2           GO TO P_CODE_NEXT;                                                                           3657
```

```
STMT LEVEL NEST

                                                                                                    3658
2367     2         PC_INST(26):                                                                     3659
                       FS_START(FS_CUR)=SYM_VALUE(OFFSET_VAL);                                      3660
2368     2             GO TO P_CODE_NEXT;                                                           3661
2369     2          PC_INST(27):                                                                    3662
                       FS_LIMIT(FS_CUR)=SYM_VALUE(OFFSET_VAL);                                      3663
2370     2             GO TO P_CODE_NEXT;                                                           3664
2371     2         PC_INST(28):                                                                     3665
                       FS_STEP(FS_CUR)=SYM_VALUE(OFFSET_VAL);                                       3666
2372     2             SYM_VALUE(FS_CTL_VAR(FS_CUR))=FS_START(FS_CUR);                              3667
2373     2             FS_INST(FS_CUR)=P_PTR;                                                       3668
2374     2             IF EXECUTION_DEBUG THEN                                                      3669
2375     2                PUT SKIP DATA(FS_AREA(FS_CUR));                                           3670
2376     2             GO TO P_CODE_NEXT;                                                           3671
                                                                                                    3672
2377     2         PC_INST(29):                                                                     3673
                       IF FS_MAX = 0 THEN   /* ERROR IF NO ACTIVE FORS */                         3674
2378     2             DO;                                                                          3675
2379     2    1           CALL PRINT_ERR('**** NEXT WITH NO FOR ****');                             3676
2380     2    1           RETURN;                                                                   3677
2381     2    1        END;                                                                         3678
2382     2             FS_CUR=1;                                                                    3679
2383     2             DO WHILE(FS_CUR<=FS_MAX);                                                    3680
2384     2    1           IF FS_CTL_VAR(FS_CUR)=PC_OBJECT(P_PTR) THEN /* FOUND IT */               3681
2385     2    1              GO TO FOUND_FOR;                                                       3682
2386     2    1           ELSE                                                                     3683
2386     2    1              FS_CUR=FS_CUR+1;                                                      3684
2387     2    1        END;                                                                         3685
2388     2             CALL PRINT_ERR('**** NEXT WITH NO FOR ****');                                3686
2389     2             RETURN;                                                                      3687
2390     2         FOUND_FOR:                                                                       3688
                       IF FS_STEP(FS_CUR)=0.0 THEN                                                  3689
2391     2             DO;                                                                          3690
2392     2    1           CALL PRINT_ERR('**** ENDLESS LOOP DETECTED - STEP IS 0 ****');            3691
2393     2    1           RETURN;                                                                   3692
2394     2    1        END;                                                                         3693
2395     2             FS_START(FS_CUR)=FS_START(FS_CUR)+FS_STEP(FS_CUR);                           3694
2396     2             SYM_VALUE(FS_CTL_VAR(FS_CUR))=FS_START(FS_CUR);                              3695
2397     2             IF FS_STEP(FS_CUR)>0.0 THEN                                                  3696
2398     2             DO;                                                                          3697
2399     2    1           IF FS_START(FS_CUR)>FS_LIMIT(FS_CUR) THEN   /* LOOP DONE? */            3698
2400     2    1              CALL COMPRESS_FS;                                                      3699
2401     2    1           ELSE                                                                     3700
2401     2    1              P_PTR=FS_INST(FS_CUR);                                                3701
2402     2    1        END;                                                                         3702
```

```
 STMT LEVEL NEST

2403     2              ELSE                                                          3703
2403     2              DO;                                                           3704
2404     2    1            IF FS_START(FS_CUR)<FS_LIMIT(FS_CUR) THEN   /* LOOP DONE? */ 3705
2405     2    1               CALL COMPRESS_FS;                                       3706
2406     2    1            ELSE                                                       3707
2406     2    1               P_PTR=FS_INST(FS_CUR);                                  3708
2407     2    1            END;                                                       3709
2408     2              GO TO P_CODE_NEXT;                                            3710
                                                                                      3711
                    /*     PCODE PTB - PRINT TAB */                                   3712
                                                                                      3713
2409     2          PC_INST(30):                                                      3714
                                                                                      3715
                         GO TO P_CODE_NEXT;                                           3716
                                                                                      3717
                    /*     PCODE RST - RESTORE DATA */                                3718
                                                                                      3719
2410     2          PC_INST(31):                                                      3720
                                                                                      3721
                         DS_CUR=0;                                                    3722
                                                                                      3723
2411     2              GO TO P_CODE_NEXT;                                            3724
                                                                                      3725
                    /*     PCODE DSL - DIM SUBSCRIPT LOCATOR */                       3726
                                                                                      3727
2412     2          PC_INST(32):                                                      3728
                                                                                      3729
                         DSL_REG=REGISTER;                                            3730
2413     2              IF EXECUTION_DEBUG THEN                                       3731
2414     2                 PUT SKIP DATA(DSL_REG,REGISTER);                           3732
2415     2              GO TO P_CODE_NEXT;                                            3733
                                                                                      3734
                    /*     PCODE JMP - JUMP   */                                      3735
                                                                                      3736
2416     2          PC_INST(35):                                                      3737
                                                                                      3738
                         P_PTR=PC_OBJECT(P_PTR);                                      3739
2417     2              GO TO P_CODE_JUMP;                                            3740
                                                                                      3741
                    /*     PCODE CFN - CALL A DEF FUNCTION   */                       3742
                                                                                      3743
2418     2          PC_INST(36):                                                      3744
                         DO I=1 TO DF_MAX;                                            3745
2419     2    1               IF DF_NAME(I)=SYMBOL(PC_OBJECT(P_PTR)) THEN            3746
2420     2    1               DO;                                                    3747
```

```
 STMT LEVEL NEST

2421     2    2               DF_RETURN(I)=P_PTR;                                    3748
2422     2    2               P_PTR=DF_OFFSET(I);                                    3749
2423     2    2               CUR_DEF=I;                                             3750
2424     2    2               GO TO P_CODE_JUMP;                                     3751
2425     2    2            END;                                                      3752
2426     2    1          END;                                                        3753
2427     2               CALL PRINT_ERR('**** USER FUNCTION NOT FOUND ****');       3754
2428     2               RETURN;                                                    3755
                                                                                    3756
                    /*     PCODE RFN - RETURN FROM FUNCTION */                     3757
                                                                                    3758
2429     2          PC_INST(37):                                                    3759
                                                                                    3760
                         P_PTR=DF_RETURN(CUR_DEF);                                  3761
2430     2               CUR_DEF=0;                                                 3762
2431     2               GO TO P_CODE_NEXT;                                         3763
                                                                                    3764
                    /*     PCODE STP - STOP EXECUTION  */                          3765
                                                                                    3766
2432     2          PC_INST(38):                                                    3767
                                                                                    3768
                         CALL PRINT_ERR('**** STOP STATEMENT EXECUTED ****');       3769
2433     2               RETURN;                                                    3770
                                                                                    3771
2434     2          COMPARE_RTN:PROC;                                               3772
                    /*****************************************************************3773
                    *                                                               *3773
                    *    THIS ROUTINE DOES ALL THE COMPARES AND SETS A LT, EQ, GT IND.  *3773
                    *                                                               *3773
                    *    NUMBERIC ITEMS WILL BE COMPARED AND THE LT,EQ, OR GT INDICATOR *3773
                    *         SET.                                                  *3773
                    *    STRINGS WILL BE COMPARED.  STRINGS OF UNEQUAL LENGTH WILL BE   *3773
                    *         PADDED WITH SPACES SO THE LENGTHS WILL BE EQUAL.      *3773
                    *         A RESULT WILL BE SET TO LT, EQ, OR GT.                *3773
                    *                                                               *3773
                    * NESTING:EXECUTION                                             *3773
                    *****************************************************************/3773
                                                                                    3784
                                                                                    3785
2435     3               COMP_RESULT=0;                                             3786
                                                                                    3787
                         /*  COMPARE NUMERIC ITEMS  */                              3788
                                                                                    3789
2436     3               IF COMP_A_TYPE < SS_STRCON & COMP_B_TYPE < SS_STRCON THEN  3790
2437     3               DO;                                                        3791
```

```
 STMT LEVEL NEST

2438     3    1            IF COMP_A < COMP_B THEN                                    3792
2439     3    1               COMP_RESULT=COMP_RESULT_LT;                             3793
2440     3    1            ELSE                                                       3794
2440     3    1               IF COMP_A = COMP_B THEN                                 3795
2441     3    1                  COMP_RESULT=COMP_RESULT_EQ;                          3796
2442     3    1               ELSE                                                    3797
2442     3    1                  COMP_RESULT=COMP_RESULT_GT;                          3798
2443     3    1         END;                                                          3799
2444     3              ELSE    /*  COMPARE STRING ITEMS  */                         3800
2444     3              DO;                                                           3801
2445     3    1            IF EXECUTION_DEBUG THEN                                  3802
2446     3    1               PUT SKIP DATA(SS_STRCON,STRING_VAL(COMP_A_STR),       3803
                                          STRING_VAL(COMP_B_STR));                  3804
2447     3    1            IF COMP_A_TYPE >= SS_STRCON & COMP_B_TYPE >= SS_STRCON THEN 3805
2448     3    1            DO;                                                      3806
2449     3    2              IF LENGTH(STRING_VAL(COMP_A_STR)) =                    3807
                               LENGTH(STRING_VAL(COMP_B_STR)) THEN                  3808
2450     3    2              DO;                                                    3809
2451     3    3                 IF STRING_VAL(COMP_A_STR) <                         3810
                                  STRING_VAL(COMP_B_STR) THEN                       3811
2452     3    3                    COMP_RESULT=COMP_RESULT_LT;                      3812
2453     3    3                 ELSE                                                3813
2453     3    3                    IF STRING_VAL(COMP_A_STR) =                      3814
                                     STRING_VAL(COMP_B_STR) THEN                    3815
2454     3    3                       COMP_RESULT=COMP_RESULT_EQ;                   3816
2455     3    3                    ELSE                                            3817
2455     3    3                         COMP_RESULT=COMP_RESULT_GT;                3818
2456     3    3              END;                                                   3819
2457     3    2              ELSE                                                   3820
2457     3    2                  CALL COMPARE_DIF_LEN;                              3821
2458     3    2            END;                                                     3822
2459     3    1            ELSE  /* OH OH - CANNOT MIX NUMBERS AND STRINGS */     3823
2459     3    1            DO;                                                      3824
2460     3    2               CALL PRINT_ERR('**** NUMBERS AND STRINGS CANNOT ' ||  3825
                                    'BE COMPARED ****');                            3826
2461     3    2               RETURN;                                               3827
2462     3    2            END;                                                     3828
2463     3    1         END;                                                        3829
                                                                                    3830
2464     3         COMPARE_DIF_LEN:PROC;                                          3831
2465     4              DECLARE (TEMP_A,TEMP_B)       CHAR(80) INITIAL((80)' ');  3832
2466     4              TEMP_A=STRING_VAL(COMP_A_STR);                            3833
2467     4              TEMP_B=STRING_VAL(COMP_B_STR);                            3834
2468     4              IF TEMP_A < TEMP_B THEN                                   3835
2469     4                 COMP_RESULT = COMP_RESULT_LT;                          3836
```

```
 STMT LEVEL NEST

2470     4              ELSE                                                    3837
2470     4                 IF TEMP_A = TEMP_B THEN                              3838
2471     4                    COMP_RESULT = COMP_RESULT_EQ;                     3839
2472     4                 ELSE                                                 3840
2472     4                    COMP_RESULT = COMP_RESULT_GT;                     3841
2473     4              IF EXECUTION_DEBUG THEN                                 3842
2474     4                 PUT SKIP DATA(COMP_RESULT,TEMP_A,TEMP_B);            3843
2475     4           END COMPARE_DIF_LEN;                                       3844
2476     3           END COMPARE_RTN;                                           3845
                                                                               3846
2477     2              DECLARE FORMAT_NUMBER ENTRY(FLOAT DECIMAL)              3847
                                      RETURNS(CHAR(14) VARYING);               3848
2478     2           FORMAT_NUMBER:PROC(A_NUMBER) RETURNS(CHAR(14) VARYING);    3849
                    /*****************************************************************
                    *                                                               *  3850
                    *    THIS ROUTINE CONVERT THE INTERNAL FLOATING POINT TO CHARACTER  *  3850
                    *                                                               *  3850
                    *    THIS ROUTINE IS CALLED BY PRINT_VAR AND THE STR FUNCTION TO    *  3850
                    *         DO THE CONVERSION.                                    *  3850
                    *                                                               *  3850
                    * NESTING:EXECUTION                                             *  3850
                    *****************************************************************/  3850
                                                                               3858
                                                                               3859
2479     3              DECLARE A_NUMBER        FLOAT DECIMAL,                  3860
                               PRINT_FIELD     CHAR(14) VARYING;               3861
                                                                               3862
2480     3              IF ABS(A_NUMBER) >= 1.0E+6 |                            3863
                           ABS(A_NUMBER) <= 1.0E-6 THEN                        3864
2481     3              DO;                                                     3865
2482     3     1           IF A_NUMBER = 0.0 THEN                               3866
2483     3     1               PRINT_FIELD=PRINT_ZERO;                          3867
2484     3     1           ELSE                                                 3868
2484     3     1           DO;                                                  3869
2485     3     2               PRINT_E_FORMAT=A_NUMBER;                         3870
2486     3     2               PRINT_FIELD=SUBSTR(PRINT_E_FORMAT,1,12);         3871
2487     3     2           END;                                                 3872
2488     3     1        END;                                                    3873
2489     3              ELSE                                                    3874
2489     3              DO;                                                     3875
2490     3     1           PRINT_WORK=A_NUMBER;                                 3876
2491     3     1           I=1;                                                 3877
2492     3     1           DO WHILE(PRINT_WORK_CHAR(I)=' ');                    3878
2493     3     2             I=I+1;                                             3879
2494     3     2           END;                                                 3880
```

```
 STMT LEVEL NEST

2495     3    1            IF PRINT_WORK_CHAR(I)='-' THEN;                                3881
2497     3    1            ELSE                                                           3882
2497     3    1              I=I-1;                                                       3883
2498     3    1            J=14;                                                          3884
2499     3    1            DO WHILE(PRINT_WORK_CHAR(J)='0');                              3885
2500     3    2              J=J-1;                                                       3886
2501     3    2            END;                                                           3887
2502     3    1            IF PRINT_WORK_CHAR(J)='.' THEN                                 3888
2503     3    1              J=J-1;                                                       3889
2504     3    1            PRINT_FIELD=SUBSTR(PRINT_WORK,I,J-I+1);                        3890
2505     3    1         END;                                                              3891
2506     3              RETURN(PRINT_FIELD);                                             3892
                                                                                         3893
2507     3         END FORMAT_NUMBER;                                                    3894
                                                                                         3895
2508     2         COMPRESS_FS:PROC;                                                     3896
                   /*********************************************************************  3897
                   *                                                                   *  3897
                   *                                                                   *  3897
                   * NESTING:EXECUTION                                                 *  3897
                   *********************************************************************/ 3897
                                                                                         3901
2509     3              DECLARE (I,T,B) FIXED BINARY ALIGNED;                           3902
                                                                                         3903
2510     3              IF FS_MAX<=1 THEN                                               3904
2511     3              DO;                                                             3905
2512     3    1            FS_MAX,FS_CUR=0;                                            3906
2513     3    1            RETURN;                                                     3907
2514     3    1         END;                                                           3908
2515     3              ELSE                                                            3909
2515     3                 IF FS_CUR=FS_MAX THEN                                        3910
2516     3                 DO;                                                          3911
2517     3    1               FS_AREA(FS_MAX)=0;                                        3912
2518     3    1               FS_MAX=FS_MAX-1;                                          3913
2519     3    1               RETURN;                                                   3914
2520     3    1            END;                                                        3915
                                                                                         3916
                     /*  DETERMINE TOP AND BOTTOM ROWS IN FS_AREA TO MOVE  */           3917
                                                                                         3918
2521     3              IF FS_CUR=1 THEN                                                3919
2522     3              DO;                                                             3920
2523     3    1            T=2;                                                        3921
2524     3    1            B=FS_MAX;                                                   3922
2525     3    1         END;                                                           3923
2526     3              ELSE                                                            3924
```

```
 STMT LEVEL NEST

2526     3              DO;                                                          3925
2527     3    1            T=FS_CUR+1;                                            3926
2528     3    1            B=FS_MAX;                                              3927
2529     3    1          END;                                                       3928
                                                                                   3929
2530     3              DO I=T TO B;                                                 3930
2531     3    1            FS_AREA(I-1)=FS_AREA(I);                               3931
2532     3    1          END;                                                       3932
                                                                                   3933
2533     3              FS_MAX=FS_MAX-1;                                             3934
                                                                                   3935
2534     3          END COMPRESS_FS;                                                 3936
                                                                                   3937
                                                                                   3938
2535     2          PRINT_BUFFER:PROC(ITEM);                                         3939
                    /*****************************************************************
                    *                                                               *  3940
                    *                                                               *  3940
                    * NESTING:EXECUTION                                             *  3940
                    *****************************************************************/ 3940
                                                                                   3944
2536     3             DECLARE ITEM                 CHAR(*) VARYING;                3945
2537     3             DECLARE NEXT_TAB             FIXED BINARY ALIGNED;           3946
2538     3             DECLARE BLANKS               CHAR(120) STATIC INITIAL((120)' ');  3947
2539     3             DECLARE COL_WIDTH            STATIC FIXED BINARY ALIGNED     3948
                                                                 INITIAL(14);      3949
                                                                                   3950
2540     3             IF PRINT_TAB_AMT > 0 THEN            /*  THIS IMPLEMENTS TAB() */  3951
2541     3             DO;                                                          3952
2542     3    1           IF LENGTH(PRINT_LINE) < PRINT_TAB_AMT THEN                3953
2543     3    1              PRINT_LINE=PRINT_LINE ||                               3954
                                    SUBSTR(BLANKS,1,PRINT_TAB_AMT-LENGTH(PRINT_LINE));  3955
2544     3    1           ELSE                                                      3956
2544     3    1              IF LENGTH(PRINT_LINE) > PRINT_TAB_AMT THEN             3957
2545     3    1              DO;                                                    3958
2546     3    2                 CALL FLUSH_BUFFER;                                  3959
2547     3    2                 PRINT_LINE=SUBSTR(BLANKS,1,PRINT_TAB_AMT);          3960
2548     3    2              END;                                                   3961
2549     3    1              ELSE;  /*  NO ACTION NEEDED ON = */                    3962
                                                                                   3963
2550     3    1           IF LENGTH(PRINT_LINE)+LENGTH(ITEM) > 120 THEN             3964
2551     3    1              CALL FLUSH_BUFFER;                                     3965
                                                                                   3966
2552     3    1           PRINT_LINE=PRINT_LINE || ITEM;                           3967
                                                                                   3968
```

```
 STMT LEVEL NEST

2553     3    1          PRINT_TAB_AMT = 0;                                          3969
2554     3    1          PRINT_LAST_PCT = 0;    /*  TAB() OVERRIDES PCT */        3970
2555     3    1       END;                                                           3971
2556     3            ELSE                                                        3972
2556     3            DO;                                                         3973
2557     3    1          IF LENGTH(PRINT_LINE)+LENGTH(ITEM) > 120 THEN            3974
2558     3    1             CALL FLUSH_BUFFER;                                    3975
                                                                                  3976
2559     3    1          IF LENGTH(PRINT_LINE) > 0  & PRINT_LAST_PCT = PCT_TAB THEN 3977
2560     3    1          DO;                                                       3978
2561     3    2             NEXT_TAB=LENGTH(PRINT_LINE)/COL_WIDTH;                 3979
2562     3    2             NEXT_TAB=(NEXT_TAB+1)*COL_WIDTH;                       3980
2563     3    2             NEXT_TAB=NEXT_TAB-LENGTH(PRINT_LINE);                  3981
2564     3    2             IF NEXT_TAB>0 THEN                                     3982
2565     3    2             DO;                                                    3983
2566     3    3                IF LENGTH(PRINT_LINE)+NEXT_TAB > 120 THEN           3984
2567     3    3                   CALL FLUSH_BUFFER;                               3985
2568     3    3                ELSE                                                3986
2568     3    3                   PRINT_LINE=PRINT_LINE || SUBSTR(BLANKS,1,NEXT_TAB); 3987
2569     3    3             END;                                                   3988
2570     3    2          END;                                                      3989
                                                                                  3990
2571     3    1          PRINT_LINE=PRINT_LINE || ITEM;                           3991
2572     3    1       END;                                                        3992
2573     3         END PRINT_BUFFER;                                              3993
                                                                                  3994
2574     2         PRINT_ERR:PROC(MSG);                                           3995
                   /*****************************************************************
                   *                                                              *  3996
                   *   PRINTS ALL ERROR MESSAGE FOR THE EXECUTION PHASE           *  3996
                   *                                                              *  3996
                   * NESTING:EXECUTION                                            *  3996
                   *****************************************************************/ 3996
                                                                                  4001
2575     3            DECLARE MSG                CHAR(*);                       4002
2576     3            CALL FLUSH_BUFFER;                                        4003
2577     3            PUT SKIP(2) EDIT('**** PROGRAM EXECUTION TERMINATED IN LINE',  4004
                                      CUR_LN) (A,F(6));                           4005
2578     3            IF TABLE_DUMP | TABLE_PRINT | ICODE_PRINT THEN            4006
2579     3               PUT EDIT(' @ OFFSET',P_PTR) (A,F(6));                  4007
2580     3            PUT EDIT(' ****',MSG) (A,SKIP,A);                         4008
2581     3            ABNORMAL_STOP=TRUE;    /* FOR END OF PROGRAM EXECUTION */  4009
2582     3         END PRINT_ERR;                                                 4010
                                                                                  4011
2583     2         FLUSH_BUFFER:PROC;                                             4012
```

```
 STMT LEVEL NEST

                  /*********************************************************************   4013
                  *                                                                   *   4013
                  *                                                                   *   4013
                  * NESTING:EXECUTION                                                 *   4013
                  *********************************************************************/  4013
                                                                                           4017
                                                                                           4018
2584     3           PUT SKIP EDIT(PRINT_LINE) (A);                                      4019
2585     3           PRINT_LINE='';                                                      4020
                                                                                           4021
2586     3        END FLUSH_BUFFER;                                                      4022
                                                                                           4023
                  /*********************************************************************   4024
                  *  RND FUNCTION                                                     *   4024
                  *                                                                   *   4024
                  * NESTING:EXECUTION                                                 *   4024
                  *********************************************************************/  4024
                                                                                           4028
```

```
STMT LEVEL NEST

                /*********************************************************************  4381
                /*********************************************************************  4381
                *                                                                   *  4381
                *   RND - RANDOM NUMBER GENERATOR                                   *  4381
                *                                                                   *  4381
                *   THIS FUNCTION GENERATES 'RANDOM' NUMBERS BETWEEN 0.0 AND 1.0    *  4381
                *   TO USE IT, SIMPLY INCLUDE %RNDGEN IN YOUR PROGRAM.              *  4381
                *                                                                   *  4381
                *   NOTE - THIS IS A SIMPLE RANDOM VALUE THAT IS NOT CONSIDERED     *  4381
                *          STATISTICALY A RANDOM NUMBER.  A STATISTICALY RANDOM     *  4381
                *          REQUIRES MULTIPLE RANDOM VALUES BE GENERATED, A MEAN BE  *  4381
                *          DETERMINED AND STUFF BEYOND THE SCOPE OF THIS MODULE.    *  4381
                *                                                                   *  4381
                *   REVISION HISTORY         COMMENTS                               *  4381
                *   V1.0.0   01/20/2017      INITIAL VERSION TRANSLATED FROM        *  4381
                *                            FORTRAN IV.                            *  4381
                *                                                                   *  4381
                *********************************************************************  4381
                *********************************************************************/ 4381
                                                                                        4399
2587     2          DECLARE RND ENTRY(FLOAT DECIMAL) RETURNS(FLOAT DECIMAL);           4400
2588     2        RND:PROC(DUMMY) RETURNS(FLOAT DECIMAL);                              4401
2589     3          DECLARE (DUMMY,YFL)     FLOAT DECIMAL;                           4402
2590     3          DECLARE ZERO     FIXED BINARY STATIC INITIAL(0);                   4403
2591     3          DECLARE ISW      FIXED BINARY STATIC INITIAL(0);                   4404
2592     3          DECLARE (IX,IY)  FIXED BINARY(31) STATIC;                          4405
                                                                                        4406
2593     3          IF ISW = ZERO THEN                                                 4407
2594     3          DO;                                                                4408
                    /* IY = 123875; */                                                 4409
2595     3    1        IY = SUBSTR(TIME,4,6);    /* TIME IS HHMMSSTTT FORMAT */      4410
2596     3    1        ISW = 1;                                                        4411
2597     3    1     END;                                                               4412
2598     3          IF MOD(IY,2) = ZERO THEN                                           4413
2599     3             IY = IY+1;                                                      4414
2600     3          IX = IY;                                                           4415
                  (NOFIXEDOVERFLOW):                                                    4416
2601     3          IY = IX * 65539;                                                   4417
2602     3          IF IY < ZERO THEN                                                  4418
2603     3          DO;                                                                4419
                  (NOFIXEDOVERFLOW):                                                    4420
2604     3    1        IY = IY + 2147483647 + 1;                                       4421
2605     3    1     END;                                                               4422
2606     3          YFL = IY;                                                          4423
2607     3          YFL = YFL * .4656613E-9;                                           4424
```

```
 STMT LEVEL NEST

2608     3            RETURN(YFL);                                          4425
2609     3          END RND;                                                4426
2610     2          END EXECUTE;                                            4030
```

```
STMT LEVEL NEST

                /*****************************************************************/   4031
                /*****************************************************************/   4032
                /*****************************************************************/   4033
                /*****************************************************************/   4034
                /*****************************************************************/   4035
                                                                                      4036
2611     1      TERMINATE:PROC;                                                       4037
                                                                                      4038
2612     2          DECLARE I                 FIXED BINARY ALIGNED;                  4039
                                                                                      4040
2613     2          IF TABLE_DUMP THEN                                               4041
2614     2             CALL PRINT_SYMBOLS;                                           4042
                                                                                      4043
2615     2      END TERMINATE;                                                        4044
                                                                                      4045
2616     1      PRINT_SYMBOLS:PROC;                                                   4046
                                                                                      4047
2617     2          PUT SKIP(2) LIST('OFFSET','SYMBOL','TYPE','OCCURS','VALUE');     4048
2618     2          DO I=1 TO SS_MAX;                                                4049
2619     2    1       PUT SKIP LIST(I,SYMBOL(I),SS_DESC(SYM_TYPE(I)),                4050
                                   SYM_DIM_MAX(I));                                   4051
2620     2    1       IF SYM_TYPE(I) = SS_STRCON |                                   4052
                        SYM_TYPE(I) = SS_STRVAR |                                     4053
                        SYM_TYPE(I) = SS_UNKNWN |                                     4054
                        SYM_TYPE(I) = SS_STRDIM THEN                                  4055
2621     2    1          PUT LIST(STRING_VAL(I));                                    4056
2622     2    1       ELSE                                                           4057
2622     2    1          PUT LIST(SYM_VALUE(I));                                     4058
2623     2    1       END;                                                           4059
2624     2            PUT SKIP LIST('END OF SYMBOL TABLE');                          4060
                                                                                      4061
2625     2      END PRINT_SYMBOLS;                                                    4062
                                                                                      4063
2626     1      END BASIC;                                                            4064
```

ATTRIBUTE AND CROSS-REFERENCE TABLE

DCL NO.		IDENTIFIER	ATTRIBUTES AND REFERENCES
160	*********	A_BLANK	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 178,179,184,191,201,202,207
2479		A_NUMBER	PARAMETER,ALIGNED,DECIMAL,FLOAT(SINGLE) 2478,2480,2480,2482,2485,2490
2001		ABNORMAL_STOP	AUTOMATIC,ALIGNED,INITIAL,STRING(1),BIT 2040,2581
		ABS	GENERIC,BUILT-IN FUNCTION 2224,2480,2480
2000		ACCUM	AUTOMATIC,ALIGNED,DECIMAL,FLOAT(SINGLE) 2013,2030,2110,2114,2128,2134,2140,2140,2142,2144,2144,2146,2148 2148,2150,2152,2152,2154,2161,2161,2163,2214,2216,2221,2221,2222 2224,2224,2225,2227,2228,2235,2235,2236,2238,2238,2239,2241,2241 2242,2244,2244,2245,2247,2247,2248,2250,2251,2251,2252,2252,2253 2255
2002	*********	ACCUM_STR	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 2112,2122,2134
2002	*********	ACCUM_TYPE	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 2014,2111,2120
500		ADD_PCODE	ENTRY,DECIMAL,FLOAT(SINGLE) 305,620,628,636,723,747,757,910,942,943,959,986,988,1021,1119,1124 1129,1134,1139,1144,1226,1238,1246,1254,1284,1397,1401,1427,1434 1536,1682,1683,1747,1750,1751,1754,1799,1800,1801,1805,1806,1807 1861,1862,1863,1864,1867,1868,1869
		ADDR	GENERIC,BUILT-IN FUNCTION 41,42
2509	*********	B	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 2524,2528,2530
1437		BALANCE_STMT	ENTRY,DECIMAL,FLOAT(SINGLE) 876,963,1095,1105,1231,1239,1247,1310,1313,1359,1420
1		BASIC	ENTRY,DECIMAL,FLOAT(SINGLE)

DCL NO.	IDENTIFIER	ATTRIBUTES AND REFERENCES
15	BASIC_RENUM	AUTOMATIC,ALIGNED,INITIAL,STRING(1),BIT 50,76,246,340,345
2538	BLANKS	STATIC,UNALIGNED,INITIAL,STRING(120),CHARACTER 2543,2547,2568
1469 *********	BREAKER	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1490,1493,1514,1514,1516
1908	CH	AUTOMATIC,UNALIGNED,STRING(1),CHARACTER 1916,1917,1917,1919,1934,1935,1937,1939,1939,1940,1977,1978
1466	CH	AUTOMATIC,UNALIGNED,STRING(1),CHARACTER 1557,1560,1571,1573,1577,1577,1577,1577,1577,1577,1577,1581,1584
1333	CH	AUTOMATIC,UNALIGNED,STRING(1),CHARACTER 1349,1350,1355,1357
1289	CH	AUTOMATIC,UNALIGNED,STRING(1),CHARACTER 1292,1293,1293,1300,1301,1306,1308
1259	CH	AUTOMATIC,UNALIGNED,STRING(1),CHARACTER 1267,1268,1270
1157	CH	AUTOMATIC,UNALIGNED,STRING(1),CHARACTER 1166,1167,1171,1173,1178,1184,1186,1196,1202,1204,1211,1212,1214
1027	CH	AUTOMATIC,UNALIGNED,STRING(1),CHARACTER 1036,1037,1037,1037,1040,1043,1045,1048,1049,1049,1054,1054,1059 1073,1079,1081
897	CH	AUTOMATIC,UNALIGNED,STRING(1),CHARACTER 915,918,920,923,926,932,935,935,941,945,945,947
853	CH	AUTOMATIC,UNALIGNED,STRING(1),CHARACTER 860,861,866,868
763	CH	AUTOMATIC,UNALIGNED,STRING(1),CHARACTER 778,782,783,787,789,793,797,804
564	CH	AUTOMATIC,UNALIGNED,STRING(1),CHARACTER 575,576,587
515	CH	AUTOMATIC,UNALIGNED,STRING(1),CHARACTER

DCL NO.	IDENTIFIER	ATTRIBUTES AND REFERENCES
		522,523,525,525,531
1334	CH2	AUTOMATIC,UNALIGNED,STRING(2),CHARACTER 1339,1340
2539 *********	COL_WIDTH	STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 2561,2562
2260	COMMON_BRANCH	STATEMENT LABEL CONSTANT 2321,2326,2329,2333,2337,2341
2000	COMP_A	AUTOMATIC,ALIGNED,DECIMAL,FLOAT(SINGLE) 2015,2031,2307,2311,2438,2440
2002 *********	COMP_A_STR	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 2016,2309,2311,2446,2449,2451,2453,2466
2002 *********	COMP_A_TYPE	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 2016,2031,2308,2311,2436,2447
2000	COMP_B	AUTOMATIC,ALIGNED,DECIMAL,FLOAT(SINGLE) 2015,2031,2313,2317,2438,2440
2002 *********	COMP_B_STR	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 2016,2315,2317,2446,2449,2451,2453,2467
2002 *********	COMP_B_TYPE	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 2016,2031,2314,2317,2436,2447
2002 *********	COMP_RESULT	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 2320,2324,2328,2332,2336,2336,2340,2340,2435,2439,2441,2442,2452 2454,2455,2469,2471,2472,2474
2003 *********	COMP_RESULT_EQ	STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 2320,2324,2336,2340,2441,2454,2471
2003 *********	COMP_RESULT_GT	STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 2328,2336,2442,2455,2472
2003 *********	COMP_RESULT_LT	STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 2332,2340,2439,2452,2469
2464	COMPARE_DIF_LEN	ENTRY,DECIMAL,FLOAT(SINGLE) 2457

DCL NO.	IDENTIFIER	ATTRIBUTES AND REFERENCES
2434	COMPARE_RTN	ENTRY,DECIMAL,FLOAT(SINGLE) 2319,2323,2327,2331,2335,2339
276	COMPILE	ENTRY,DECIMAL,FLOAT(SINGLE) 49,54
2508	COMPRESS_FS	ENTRY,DECIMAL,FLOAT(SINGLE) 2400,2405
1471	CONTINUE_SCAN	AUTOMATIC,ALIGNED,STRING(1),BIT 1492,1493,1512,1513
767	CONTINUE_SCAN	AUTOMATIC,ALIGNED,STRING(1),BIT 771,774,800,830,835,838
610	CONTINUE_SCAN	AUTOMATIC,ALIGNED,STRING(1),BIT
563	CONTINUE_SCAN	AUTOMATIC,ALIGNED,STRING(1),BIT 573,574,582,584,590,596,597,599,601
514	CONTINUE_SCAN	AUTOMATIC,ALIGNED,STRING(1),BIT 520,521,524,527,534,544,549
414	CONTINUE_SCAN	AUTOMATIC,ALIGNED,STRING(1),BIT 416,417,422,425
166	CONTINUE_SCAN	AUTOMATIC,ALIGNED,STRING(1),BIT 185,186,192,195,200,207,210,215,217,220,222,224,226,231,232,238
	CONTINUE_SCAN	AUTOMATIC,ALIGNED,DECIMAL,FLOAT(SINGLE) 441
	COS	GENERIC,BUILT-IN FUNCTION 2238
1263	CTL_VAR	AUTOMATIC,UNALIGNED,STRING(10),CHARACTER 1281,1282
1159	CTL_VAR	AUTOMATIC,UNALIGNED,STRING(10),CHARACTER 1223,1224,1228
	CUR_DEF	AUTOMATIC,ALIGNED,DECIMAL,FLOAT(SINGLE) 2020,2423,2429,2430
1998 *********	CUR_DF	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0)

DCL NO.		IDENTIFIER	ATTRIBUTES AND REFERENCES
1997	*********	CUR_LN	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 2106,2108,2279,2290,2577
		DANGLE	AUTOMATIC,ALIGNED,DECIMAL,FLOAT(SINGLE) 905
31		DATA_STACK	AUTOMATIC,STRUCTURE
		DATE	BUILT-IN FUNCTION 37
36		DEF_FUNC_AREA	(10)IN DEF_FUNCTIONS,AUTOMATIC,STRUCTURE
36		DEF_FUNCTIONS	AUTOMATIC,STRUCTURE
512		DEFINITION	PARAMETER,ALIGNED,STRING(1),BIT 511,539
36	*********	DF_CUR	IN DEF_FUNCTIONS,AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 113
36	*********	DF_MAX	IN DEF_FUNCTIONS,AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 113,329,1408,1413,1413,1414,1415,1416,2418
36		DF_NAME	IN DEF_FUNC_AREA(10) IN DEF_FUNCTIONS,AUTOMATIC,ALIGNED,STRING (10),CHARACTER 330,1408,1414,2419
36	*********	DF_OFFSET	IN DEF_FUNC_AREA(10) IN DEF_FUNCTIONS,AUTOMATIC,ALIGNED,BINARY,FIXED (15,0) 330,1415,2422
36	*********	DF_RETURN	IN DEF_FUNC_AREA(10) IN DEF_FUNCTIONS,AUTOMATIC,ALIGNED,BINARY,FIXED (15,0) 1416,2421,2429
1912		DIM_VAR	AUTOMATIC,UNALIGNED,STRING(10),CHARACTER 1921,1922,1963
31	*********	DS_CUR	IN DATA_STACK,AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 110,2165,2165,2166,2172,2173,2410
31		DS_ITEM	IN DS_TABLE(500) IN DATA_STACK,AUTOMATIC,ALIGNED,BINARY, FLOAT(SINGLE)

DCL NO.		IDENTIFIER	ATTRIBUTES AND REFERENCES
			841,846,2172
31	*********	DS_MAX	IN DATA_STACK,AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 110,841,844,844,845,846,2166
31	*********	DS_STR	IN DS_TABLE(500) IN DATA_STACK,AUTOMATIC,ALIGNED,BINARY,FIXED (15,0) 845,2173
31		DS_TABLE	(500)IN DATA_STACK,AUTOMATIC,STRUCTURE
1996	*********	DSL_REG	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 2019,2029,2052,2053,2058,2063,2065,2066,2412,2414
2589		DUMMY	PARAMETER,ALIGNED,DECIMAL,FLOAT(SINGLE) 2588
167		EDIT_LINE_NUM	AUTOMATIC,UNALIGNED,DECIMAL,PICTURE(ZZZZ9) 189,191,235,237
2254		END_FNC	STATEMENT LABEL CONSTANT 2223,2226,2234,2237,2240,2243,2246,2249
336		END_OF_COMP	STATEMENT LABEL CONSTANT 325
272		ENDSELECT_MACRO1	STATEMENT LABEL CONSTANT 252,257,262
408		ENDSELECT_MACRO2	STATEMENT LABEL CONSTANT 377,382,387,392,397
739		ENDSELECT_MACRO3	STATEMENT LABEL CONSTANT 614,622,630,638,646,654,662,670,678,686,692,700,708,717,725
807		ENDSELECT_MACRO4	STATEMENT LABEL CONSTANT 785,791,795
1151		ENDSELECT_MACRO5	STATEMENT LABEL CONSTANT 1120,1125,1130,1135,1140
2104		ENDSELECT_MACRO6	STATEMENT LABEL CONSTANT 2071,2081,2091
2206		ENDSELECT_MACRO7	STATEMENT LABEL CONSTANT

DCL NO.		IDENTIFIER	ATTRIBUTES AND REFERENCES
			2188,2193
5		EOF_SYSIN	AUTOMATIC,ALIGNED,INITIAL,STRING(1),BIT 40,44,65,80,95
6		EOP_SYSIN	AUTOMATIC,ALIGNED,STRING(1),BIT 40,65,70,80,91
1907	*********	ERR_PTR	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0)
1258	*********	ERR_PTR	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1272,1278,1285
609	*********	ERR_PTR	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 712,715,729,732
		ERR_PTR	AUTOMATIC,ALIGNED,DECIMAL,FLOAT(SINGLE) 1405,1410,1418,1419
8		ERROR_COUNT	AUTOMATIC,ALIGNED,INITIAL,DECIMAL,FIXED(5,0) 50,56,114,337,344,432,432
1990		EXECUTE	ENTRY,DECIMAL,FLOAT(SINGLE) 58
18		EXECUTION_DEBUG	AUTOMATIC,ALIGNED,INITIAL,STRING(1),BIT 75,364,366,2064,2107,2113,2117,2133,2137,2141,2145,2149,2153,2162 2213,2254,2269,2310,2316,2364,2374,2413,2445,2473
1463		EXP	PARAMETER,UNALIGNED,STRING(*),CHARACTER,VARYING 1462,1556,1557
1438		EXP	PARAMETER,UNALIGNED,STRING(*),CHARACTER,VARYING 1437,1443,1444,1446,1453
29	*********	EXP_CALC	IN MISC_CODE_DEF,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 880,967,1099,1109,1235,1243,1251,1319,1489
29	*********	EXP_FN_CALC	IN MISC_CODE_DEF,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 1426,1475,1489
29	*********	EXP_RCVR	IN MISC_CODE_DEF,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 1327
1464	*********	EXP_TYPE	PARAMETER,ALIGNED,BINARY,FIXED(15,0)

DCL NO.		IDENTIFIER	ATTRIBUTES AND REFERENCES
			1462,1475,1489,1489
1465		EXPR	AUTOMATIC,ALIGNED,STRING(1),BIT 1489,1526,1681,1859
761		EXTRACT_DATA	ENTRY,DECIMAL,FLOAT(SINGLE) 661
810		EXTRACT_DATA_ITEM	ENTRY,DECIMAL,FLOAT(SINGLE) 799,809
23		FALSE	STATIC,ALIGNED,INITIAL,STRING(1),BIT 44,70,71,72,73,74,75,80,80,95,192,210,220,226,238,246,302,352,358 362,366,422,441,524,527,534,544,549,582,584,590,599,742,752,771,775 784,830,835,838,863,905,906,919,937,1039,1077,1093,1169,1182,1200 1206,1269,1303,1352,1502,1513,1553,1562,1918,1936
161	*********	FIRST_CHAR	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 229,237
162	*********	FIRST_DIGIT	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 191,237,251,256,261,266,269
2583		FLUSH_BUFFER	ENTRY,DECIMAL,FLOAT(SINGLE) 2185,2257,2546,2551,2558,2567,2576
1468		FN_TMP	AUTOMATIC,UNALIGNED,STRING(10),CHARACTER 1479,1481,1484
2005		FOR_STACK	AUTOMATIC,STRUCTURE
2478		FORMAT_NUMBER	ENTRY,STRING(14),CHARACTER,VARYING 2177
2390		FOUND_FOR	STATEMENT LABEL CONSTANT 2385
2005		FS_AREA	(10)IN FOR_STACK,AUTOMATIC,STRUCTURE 2365,2375,2517,2531,2531
2005	*********	FS_CTL_VAR	IN FS_AREA(10) IN FOR_STACK,AUTOMATIC,ALIGNED,BINARY,FIXED (15,0) 2349,2353,2361,2372,2384,2396
2005	*********	FS_CUR	IN FOR_STACK,AUTOMATIC,ALIGNED,BINARY,FIXED(15,0)

DCL NO.		IDENTIFIER	ATTRIBUTES AND REFERENCES
			2018,2033,2345,2347,2348,2349,2351,2351,2359,2361,2362,2362,2362 2363,2365,2367,2369,2371,2372,2372,2373,2375,2382,2383,2384,2386 2386,2390,2395,2395,2395,2396,2396,2397,2399,2399,2401,2404,2404 2406,2512,2515,2521,2527
2005	*********	FS_INST	IN FS_AREA(10) IN FOR_STACK,AUTOMATIC,ALIGNED,BINARY,FIXED (15,0) 2363,2373,2401,2406
2005		FS_LIMIT	IN FS_AREA(10) IN FOR_STACK,AUTOMATIC,ALIGNED,DECIMAL,FLOAT(SINGLE) 2362,2369,2399,2404
2005	*********	FS_MAX	IN FOR_STACK,AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 2018,2033,2344,2345,2348,2353,2358,2358,2359,2377,2383,2510,2512 2515,2517,2518,2518,2524,2528,2533,2533
2005		FS_START	IN FS_AREA(10) IN FOR_STACK,AUTOMATIC,ALIGNED,DECIMAL,FLOAT(SINGLE) 2362,2367,2372,2395,2395,2396,2399,2404
2005		FS_STEP	IN FS_AREA(10) IN FOR_STACK,AUTOMATIC,ALIGNED,DECIMAL,FLOAT(SINGLE) 2362,2371,2390,2395,2397
279		FUNC_ARG	AUTOMATIC,UNALIGNED,STRING(10),CHARACTER 1380,1381,1386,1478,1479,1483
279		FUNC_NAME	AUTOMATIC,UNALIGNED,STRING(10),CHARACTER 1366,1368,1392,1414,1477,1479
1335		FUNC_TEMP	AUTOMATIC,UNALIGNED,STRING(80),CHARACTER,VARYING 1365,1366,1386,1386,1399,1400
561		GET_KEYWORD	ENTRY,DECIMAL,FLOAT(SINGLE) 308
511		GET_STMT_NUM	ENTRY,DECIMAL,FLOAT(SINGLE) 303,742,752,1093
2004		GOSUB_AREA	(25)IN GOSUB_STACK,AUTOMATIC,STRUCTURE
2004		GOSUB_STACK	AUTOMATIC,STRUCTURE
		GS_CUR	AUTOMATIC,ALIGNED,DECIMAL,FLOAT(SINGLE) 2017,2032,2278,2279
2004	*********	GS_CURM	IN GOSUB_STACK,AUTOMATIC,ALIGNED,BINARY,FIXED(15,0)

DCL NO.	IDENTIFIER	ATTRIBUTES AND REFERENCES
2004 *********	GS_LINE	IN GOSUB_AREA(25) IN GOSUB_STACK,AUTOMATIC,ALIGNED,BINARY,FIXED (15,0) 2271,2279,2290
2004 *********	GS_MAX	IN GOSUB_STACK,AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 2017,2032,2271,2276,2278,2289,2289,2290,2291,2299,2304,2305,2305
2004 *********	GS_PTR	IN GOSUB_AREA(25) IN GOSUB_STACK,AUTOMATIC,ALIGNED,BINARY,FIXED (15,0) 2291,2304
	HBOUND	GENERIC,BUILT-IN FUNCTION 82,97,452,502,553,597,841,1005,1408,1955,2208,2271,2353
1611 *********	HJ	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1650,1654,1657,1666,1668
1611 *********	HK	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1651,1655,1658
765	HOLD_VAL	AUTOMATIC,UNALIGNED,STRING(80),CHARACTER,VARYING 815,839
2612 *********	I	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0)
2509 *********	I	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 2530,2531,2531
1999 *********	I	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 2418,2419,2421,2422,2423,2491,2492,2493,2493,2495,2497,2497,2504 2504
1906 *********	I	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1915,1916,1929,1933,1934,1951,1961
1838 *********	I	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1845,1846,1846,1846,1876,1877,1877,1884,1885,1885,1898,1899,1899 1899
1781 *********	I	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1787,1788,1788,1788,1814,1815,1815,1822,1823,1823,1832,1833,1833 1833
1611 *********	I	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0)

DCL NO.	IDENTIFIER	ATTRIBUTES AND REFERENCES
		1616,1617,1617,1617,1689,1690,1690,1690,1697,1698,1698,1708,1709 1709,1709,1727,1728,1728,1728,1737,1738,1738,1738,1756,1757,1757 1765,1766,1766,1766,1775,1776,1776,1776
1464	********* I	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1477,1479,1482,1483,1484,1486,1494,1497,1498,1498,1499,1500,1501 1509,1523,1524,1530,1534,1538,1538,1538,1538,1556,1557,1592,1593 1594,1596,1597,1597,1604,1605,1605,1605
1439	********* I	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1443,1444,1446,1453
1330	********* I	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1348,1349,1353,1362,1363,1365,1367,1370,1383
1288	********* I	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1299,1300,1304
1257	********* I	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1266,1267,1273
1155	********* I	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1165,1166,1175,1177,1178,1179,1193,1195,1196,1197,1209,1210,1211
1026	********* I	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1035,1036,1041,1047,1048,1052,1057,1062,1072,1073,1074,1088,1115 1119,1124,1129,1134,1139,1144
994	********* I	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 996,997,1011,1012,1013,1014,1014,1015,1015
896	********* I	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 909,913,914,915,934,949,949
852	********* I	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 859,860
762	********* I	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 777,778,839,842
608	********* I	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0)
562	********* I	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 574,575,591,605

DCL NO.		IDENTIFIER	ATTRIBUTES AND REFERENCES
513	*********	I	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 521,522,528,535,559
437	*********	I	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 444,445,447,448,449,476,490
429	*********	I	PARAMETER,ALIGNED,BINARY,FIXED(15,0) 428,431
415	*********	I	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 417,418,421
326	*********	I	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 329,330,330,372,374,376,376,376,379,381,381,381,384,386,386,386,389 391,391,391,394,396,396,399,401,401,401,404,746,747,756,757
163	*********	I	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 186,187,189,207,208,212,212,213,213,216,218,219,219,223,225,227,227 229,230,230,232,233,235,249,254,259,264
	*********	I	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 2618,2619,2619,2619,2619,2620,2620,2620,2620,2621,2622
21		ICODE_PRINT	AUTOMATIC,ALIGNED,INITIAL,STRING(1),BIT 74,310,360,362,2578
1549		IN_STR	AUTOMATIC,ALIGNED,STRING(1),BIT 1553,1558,1562,1575
901		IN_STR	AUTOMATIC,ALIGNED,STRING(1),BIT 906,916,919,925,929
768		IN_STR	AUTOMATIC,ALIGNED,STRING(1),BIT 775,780,784,790
		INDEX	GENERIC,BUILT-IN FUNCTION 178,201,1362,1477,1478
106	*********	INITIALIZE	ENTRY,BINARY,FIXED(15,0) 45,53
2591	*********	ISW	STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 2593,2596
2536		ITEM	PARAMETER,UNALIGNED,STRING(*),CHARACTER,VARYING

DCL NO.		IDENTIFIER	ATTRIBUTES AND REFERENCES
			2535,2550,2552,2557,2571
1472		ITEMS	(50)IN STACK,AUTOMATIC,STRUCTURE 1486,1698,1698,1757,1757,1815,1815,1823,1823,1877,1877,1885,1885
2592		IX	STATIC,ALIGNED,BINARY,FIXED(31,0) 2600,2601
2592		IY	STATIC,ALIGNED,BINARY,FIXED(31,0) 2595,2598,2599,2599,2600,2601,2602,2604,2604,2606
1611	*********	J	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1657,1659,1662,1666,1668,1704,1736,1741,1742,1742,1746,1748,1749 1752,1753,1755,1755,1756,1761,1764,1770,1770,1774
1464	*********	J	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1478,1479
562	*********	J	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 597,598
	*********	J	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 2498,2499,2500,2500,2502,2503,2503,2504
1331	*********	JMP_OFFSET	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1398,1435
1611	*********	K	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1658,1659,1662,1703,1703,1736,1741,1760,1760,1764,1774
26		KEY_WORD_AREA	STATIC,STRUCTURE
26		KEY_WORDS	(16)AUTOMATIC,DEFINED,UNALIGNED,STRING(8),CHARACTER 597,598
565		KW	AUTOMATIC,UNALIGNED,STRING(9),CHARACTER,VARYING 572,578,580,587,587,588,595
26		KW_DATA	IN KEY_WORD_AREA,STATIC,UNALIGNED,INITIAL,STRING(8),CHARACTER 656
26		KW_DEF	IN KEY_WORD_AREA,STATIC,UNALIGNED,INITIAL,STRING(8),CHARACTER 672
26		KW_DIM	IN KEY_WORD_AREA,STATIC,UNALIGNED,INITIAL,STRING(8),CHARACTER

DCL NO.	IDENTIFIER	ATTRIBUTES AND REFERENCES
		447,471,482,727
26	KW_END	IN KEY_WORD_AREA,STATIC,UNALIGNED,INITIAL,STRING(8),CHARACTER 616
26	KW_FOR	IN KEY_WORD_AREA,STATIC,UNALIGNED,INITIAL,STRING(8),CHARACTER 702
26	KW_GOSUB	IN KEY_WORD_AREA,STATIC,UNALIGNED,INITIAL,STRING(8),CHARACTER 648
26	KW_GOTO	IN KEY_WORD_AREA,STATIC,UNALIGNED,INITIAL,STRING(8),CHARACTER 640
26	KW_IF	IN KEY_WORD_AREA,STATIC,UNALIGNED,INITIAL,STRING(8),CHARACTER 694
26	KW_LET	IN KEY_WORD_AREA,STATIC,UNALIGNED,INITIAL,STRING(8),CHARACTER 664
26	KW_NEXT	IN KEY_WORD_AREA,STATIC,UNALIGNED,INITIAL,STRING(8),CHARACTER 710
26	KW_PRINT	IN KEY_WORD_AREA,STATIC,UNALIGNED,INITIAL,STRING(8),CHARACTER 688
26	KW_READ	IN KEY_WORD_AREA,STATIC,UNALIGNED,INITIAL,STRING(8),CHARACTER 680
26	KW_REM	IN KEY_WORD_AREA,STATIC,UNALIGNED,INITIAL,STRING(8),CHARACTER 612
26	KW_RESTORE	IN KEY_WORD_AREA,STATIC,UNALIGNED,INITIAL,STRING(8),CHARACTER 719
26	KW_RETURN	IN KEY_WORD_AREA,STATIC,UNALIGNED,INITIAL,STRING(8),CHARACTER 632
26	KW_STOP	IN KEY_WORD_AREA,STATIC,UNALIGNED,INITIAL,STRING(8),CHARACTER 624
1999 *********	L	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0)
163 *********	LAST_CHAR	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 223,230

DCL NO.		IDENTIFIER	ATTRIBUTES AND REFERENCES
11		LAST_LINE_NUM	AUTOMATIC,ALIGNED,DECIMAL,FIXED(5,0) 109,306,542,547,556
1469	*********	LAST_LP	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1496,1500,1506,1510
896	*********	LAST_NB	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 903,934,955,957,957
277	*********	LAST_PCODE_PRINTED	AUTOMATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 370,372,410
		LBOUND	GENERIC,BUILT-IN FUNCTION 2208
1909		LEFT_SIDE	AUTOMATIC,UNALIGNED,STRING(80),CHARACTER,VARYING 1913,1919,1919,1921,1931,1940,1940,1952
1335		LEFT_SIDE	AUTOMATIC,UNALIGNED,STRING(80),CHARACTER,VARYING 1346,1357,1357,1359,1362,1365,1367,1367,1378,1378,1380,1380
1290		LEFT_SIDE	AUTOMATIC,UNALIGNED,STRING(80),CHARACTER,VARYING 1297,1308,1308,1310,1326,1326,1327
1260		LEFT_SIDE	AUTOMATIC,UNALIGNED,STRING(80),CHARACTER,VARYING 1264,1270,1270,1281
1158		LEFT_SIDE	AUTOMATIC,UNALIGNED,STRING(80),CHARACTER,VARYING 1163,1173,1173,1223
1028		LEFT_SIDE	AUTOMATIC,UNALIGNED,STRING(80),CHARACTER,VARYING 1032,1045,1045,1095,1098,1098,1099,1108,1109
899		LEFT_SIDE	AUTOMATIC,UNALIGNED,STRING(80),CHARACTER,VARYING 902,920,920,926,926,938,947,947,951,952,963,966,966,967,991,996,997 1010,1010,1010,1012,1013,1014,1014,1014,1019,1022
855		LEFT_SIDE	AUTOMATIC,UNALIGNED,STRING(80),CHARACTER,VARYING 857,868,868,870,873,876,879,879,880,892
		LENGTH	GENERIC,BUILT-IN FUNCTION 532,578,588,818,825,826,870,873,951,996,1010,1012,1378,1380,1443 1556,2449,2449,2542,2543,2544,2550,2550,2557,2557,2559,2561,2563 2566

DCL NO.		IDENTIFIER	ATTRIBUTES AND REFERENCES
1993		LIB_FNC	(2:10)AUTOMATIC,INITIAL,LABEL 2216,2224,2227,2235,2238,2241,2244,2247,2250,2208,2208,2215
517		LINE_NUM	AUTOMATIC,ALIGNED,DECIMAL,FIXED(5,0) 541,542,547,556
32		LINE_STACK	AUTOMATIC,STRUCTURE
159	*********	LINE_SUB	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 169,170,170,173,174,242,244
158		LINE_WORK	AUTOMATIC,UNALIGNED,STRING(80),CHARACTER 174,175,178,184,191,191,201,208,213,213,218,225,230,237,237,242
516		LN	AUTOMATIC,UNALIGNED,STRING(6),CHARACTER,VARYING 518,531,531,532,541,558
435	*********	LOOKUP_SYMBOL_TABLE	ENTRY,BINARY,FIXED(15,0) 822,1224,1282,1392,1400,1522,1566,1638,1680,1715,1746,1748,1753,1783 1796,1841,1858,1922
1610	*********	LP	PARAMETER,ALIGNED,BINARY,FIXED(15,0) 1609,1615,1616,1624,1624,1626,1627,1638,1648,1650,1654,1671,1688 1693,1693,1694,1695,1695,1696,1697,1700,1704,1707,1715,1761,1783 1812,1813,1814,1820,1821,1821,1822,1841,1874,1875,1876,1882,1883 1883,1884
32	*********	LS_CUR	IN LINE_STACK,AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 111,2260,2261,2263,2286,2287,2292
32		LS_LINE	IN LS_NUM(500) IN LINE_STACK,AUTOMATIC,ALIGNED,DECIMAL,FIXED (5,0) 187,233,306,381,553,2106,2261,2287
32	*********	LS_MAX	IN LINE_STACK,AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 111,169,186,232,304,304,305,306,307,553,2260,2286
32		LS_NUM	(500)IN LINE_STACK,AUTOMATIC,STRUCTURE
32	*********	LS_OFFSET	IN LS_NUM(500) IN LINE_STACK,AUTOMATIC,ALIGNED,BINARY,FIXED (15,0) 307,2263,2292
1469	*********	MAX_BREAKER	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0)

DCL NO.		IDENTIFIER	ATTRIBUTES AND REFERENCES
			1491,1493,1516
29		MISC_CODE_DEF	STATIC,STRUCTURE
		MOD	GENERIC,BUILT-IN FUNCTION 1000,1459,2598
63	*********	MONITOR	ENTRY,BINARY,FIXED(15,0) 46
16		MONITOR_STMT	AUTOMATIC,UNALIGNED,STRING(80),CHARACTER 66,69,288
2575		MSG	PARAMETER,UNALIGNED,STRING(*),CHARACTER 2574,2580
430		MSG	PARAMETER,UNALIGNED,STRING(*),CHARACTER 428,431
165		NEW_LINE_NUM	(500)AUTOMATIC,ALIGNED,DECIMAL,FIXED(5,0) 170,189,235,249,254,259,264
1913		NEXT_DIM	STATEMENT LABEL CONSTANT 1981
2537	*********	NEXT_TAB	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 2561,2562,2562,2563,2563,2564,2566,2568
900		NO_COMMA	AUTOMATIC,ALIGNED,STRING(1),BIT 904,937
856		NO_COMMA	AUTOMATIC,ALIGNED,STRING(1),BIT 858,863
1337		NO_EQUAL	AUTOMATIC,ALIGNED,STRING(1),BIT 1347,1348,1352
1291		NO_EQUAL	AUTOMATIC,ALIGNED,STRING(1),BIT 1298,1299,1303
1910		NO_OPER	AUTOMATIC,ALIGNED,STRING(1),BIT 1914,1915,1918,1932,1933,1936,1946
1261		NO_OPER	AUTOMATIC,ALIGNED,STRING(1),BIT 1265,1266,1269

DCL NO.	IDENTIFIER	ATTRIBUTES AND REFERENCES
1160	NO_OPER	AUTOMATIC,ALIGNED,STRING(1),BIT 1164,1165,1169,1176,1177,1182,1188,1194,1195,1200,1206,1208,1210
1029	NO_OPER	AUTOMATIC,ALIGNED,STRING(1),BIT 1033,1035,1039,1071,1072,1077,1083
1470	NO_PARENS	AUTOMATIC,ALIGNED,STRING(1),BIT 1495,1497,1502,1504
1906 *********	NUM_OCCURS	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1952,1955,1960,1961
766	NUM_VAL	AUTOMATIC,ALIGNED,BINARY,FLOAT(SINGLE) 814,836,846
1911 *********	OFFSET	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1922,1923,1923,1953,1953,1953,1964
1838 *********	OFFSET	PARAMETER,ALIGNED,BINARY,FIXED(15,0) 1837,1851,1851,1853,1863,1869,1870
1781 *********	OFFSET	PARAMETER,ALIGNED,BINARY,FIXED(15,0) 1780,1797,1800,1803,1806
1733 *********	OFFSET	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1746,1747,1748,1750,1751,1753,1754
1611 *********	OFFSET	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1638,1639,1639,1680,1682,1683,1715,1716,1716,1717,1718,1718,1719 1720
1552 *********	OFFSET	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1566,1568,1569
1469 *********	OFFSET	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1522,1536
1332 *********	OFFSET	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1392,1393,1394,1396,1402,1427,1434
1262 *********	OFFSET	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1282,1283,1284
1156 *********	OFFSET	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0)

DCL NO.		IDENTIFIER	ATTRIBUTES AND REFERENCES
			1224,1225,1226,1238,1246,1254
812	*********	OFFSET	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 813,822,824,826,827,845
501	*********	OFFSET	PARAMETER,ALIGNED,BINARY,FIXED(15,0) 500,509
1838	*********	OFFSET2	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1841,1849,1849,1861,1867
1781	*********	OFFSET2	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1783,1791,1791,1799,1805
1332	*********	OFFSET2	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1400,1401,1402
1838	*********	OFFSET3	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1858,1864
1781	*********	OFFSET3	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1796,1801,1807
1996	*********	OFFSET_VAL	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 2029,2063,2065,2110,2111,2112,2114,2114,2116,2118,2121,2121,2122 2127,2128,2134,2136,2138,2140,2142,2144,2148,2152,2156,2161,2168 2172,2173,2177,2307,2308,2309,2313,2314,2315,2367,2369,2371
164		OLD_LINE_NUM	AUTOMATIC,ALIGNED,DECIMAL,FIXED(5,0) 184,187,230,233
		ONCHAR	BUILT-IN FUNCTION 442,772
1472		OP	IN ITEMS(50) IN STACK,AUTOMATIC,UNALIGNED,STRING(1),CHARACTER 1499,1501,1581,1590,1594,1597,1597,1605,1617,1624,1624,1648,1678 1690,1695,1695,1696,1709,1728,1738,1742,1742,1749,1755,1755,1766 1776,1788,1813,1821,1821,1833,1846,1875,1883,1883,1899
1732		OP1	PARAMETER,UNALIGNED,STRING(1),CHARACTER 1731,1736,1742,1749
1732		OP2	PARAMETER,UNALIGNED,STRING(1),CHARACTER 1731,1736,1742

DCL NO.	IDENTIFIER	ATTRIBUTES AND REFERENCES
1030	OPER	AUTOMATIC,UNALIGNED,STRING(2),CHARACTER 1034,1040,1049,1049,1051,1054,1054,1056,1059,1061,1064,1064,1064 1117,1122,1127,1132,1137,1142
436 *********	OPT	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0)
2039	P_CODE_JUMP	STATEMENT LABEL CONSTANT 2264,2294,2417,2424
2038	P_CODE_NEXT	STATEMENT LABEL CONSTANT 2109,2115,2119,2135,2139,2143,2147,2151,2155,2164,2174,2179,2181 2207,2256,2306,2312,2318,2322,2325,2330,2334,2338,2342,2343,2366 2368,2370,2376,2408,2409,2411,2415,2431
34	P_CODE_STACK	AUTOMATIC,STRUCTURE
1995 *********	P_CTR	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 2011,2027,2039,2039,2042,2258
1995 *********	P_CTR_MAX	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 2012,2042,2044
1994 *********	P_PTR	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 2011,2027,2028,2028,2038,2038,2047,2053,2063,2065,2065,2068,2073 2075,2083,2085,2093,2095,2105,2106,2180,2183,2190,2195,2208,2208 2210,2214,2214,2215,2222,2225,2228,2236,2239,2242,2245,2248,2253 2255,2255,2261,2263,2267,2270,2287,2291,2292,2293,2297,2304,2349 2361,2373,2384,2401,2406,2416,2416,2419,2421,2422,2429,2579
1994 *********	P_PTR_SUB	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 2070,2076,2086,2096,2097,2171,2175,2311,2317
3	PAGE_NUM	AUTOMATIC,ALIGNED,INITIAL,DECIMAL,FIXED(5,0) 283,285,285,316,318,318
2	PAGE_TITLE	STATIC,UNALIGNED,INITIAL,STRING(20),CHARACTER 287,320
1440 *********	PARENS	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1441,1445,1445,1448,1448,1449,1456
1462	PARSE_EXP	ENTRY,DECIMAL,FLOAT(SINGLE) 880,967,1099,1109,1235,1243,1251,1319,1327,1426
1732 *********	PC1	PARAMETER,ALIGNED,BINARY,FIXED(15,0)

DCL NO.	IDENTIFIER	ATTRIBUTES AND REFERENCES
		1731,1750
1732 *********	PC2	PARAMETER,ALIGNED,BINARY,FIXED(15,0) 1731,1751
30	PC_ALLOW	IN PC_OPTAB(0:38) IN PC_CON_TABLE,BASED(PC_CON_TABLE_PTR),ALIGNED, STRING(2),BIT 2068,2073,2083,2093
30	PC_ALLOWS_ADD	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_B	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_BAL	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_BEQ	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_BGE	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_BGT	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_BLE	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_BLT	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_BNE	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_CFN	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_DIV	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_DSL	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_END	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_EXP	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_FIX	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_FNC	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_FNX	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_FST	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT

DCL NO.	IDENTIFIER	ATTRIBUTES AND REFERENCES
30	PC_ALLOWS_FSU	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_FUL	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_JMP	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_LCA	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_LCB	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_LDA	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_LDR	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_MUL	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_PCT	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_PRS	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_PRV	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_PTB	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_RDV	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_RET	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_RFN	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_RST	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_SLN	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_STA	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_STP	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_STR	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_ALLOWS_SUB	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(2),BIT
30	PC_CON_TABLE	BASED(PC_CON_TABLE_PTR),STRUCTURE
	PC_CON_TABLE_PTR	AUTOMATIC,ALIGNED,POINTER

DCL NO.	IDENTIFIER	ATTRIBUTES AND REFERENCES
		41,374,376,379,381,384,386,389,391,394,396,399,401,404,972,982,2068 2073,2083,2093
30	PC_CONSTANTS	STATIC,STRUCTURE 41
34 *********	PC_CUR	IN P_CODE_STACK,AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 112
30 *********	PC_FORMAT	IN PC_OPTAB(0:38) IN PC_CON_TABLE,BASED(PC_CON_TABLE_PTR),ALIGNED, BINARY,FIXED(15,0) 374,379,384,389,394,399,972,982
29 *********	PC_FORMAT_0	IN MISC_CODE_DEF,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 374
29 *********	PC_FORMAT_1	IN MISC_CODE_DEF,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 379
29 *********	PC_FORMAT_2	IN MISC_CODE_DEF,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 384
29 *********	PC_FORMAT_3	IN MISC_CODE_DEF,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 389
29 *********	PC_FORMAT_4	IN MISC_CODE_DEF,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 394
29 *********	PC_FORMAT_5	IN MISC_CODE_DEF,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 399
30 *********	PC_FORMAT_ADD	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30 *********	PC_FORMAT_B	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30 *********	PC_FORMAT_BAL	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30 *********	PC_FORMAT_BEQ	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30 *********	PC_FORMAT_BGE	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30 *********	PC_FORMAT_BGT	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30 *********	PC_FORMAT_BLE	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)

DCL NO.	IDENTIFIER	ATTRIBUTES AND REFERENCES
30	********* PC_FORMAT_BLT	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30	********* PC_FORMAT_BNE	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30	********* PC_FORMAT_CFN	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30	********* PC_FORMAT_DIV	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30	********* PC_FORMAT_DSL	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30	********* PC_FORMAT_END	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30	********* PC_FORMAT_EXP	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30	********* PC_FORMAT_FIX	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30	********* PC_FORMAT_FNC	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30	********* PC_FORMAT_FNX	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30	********* PC_FORMAT_FST	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30	********* PC_FORMAT_FSU	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30	********* PC_FORMAT_FUL	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30	********* PC_FORMAT_JMP	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30	********* PC_FORMAT_LCA	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30	********* PC_FORMAT_LCB	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30	********* PC_FORMAT_LDA	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30	********* PC_FORMAT_LDR	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30	********* PC_FORMAT_MUL	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30	********* PC_FORMAT_PCT	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30	********* PC_FORMAT_PRS	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30	********* PC_FORMAT_PRV	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30	********* PC_FORMAT_PTB	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)

DCL NO.	IDENTIFIER	ATTRIBUTES AND REFERENCES
30 *********	PC_FORMAT_RDV	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30 *********	PC_FORMAT_RET	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30 *********	PC_FORMAT_RFN	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30 *********	PC_FORMAT_RST	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30 *********	PC_FORMAT_SLN	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30 *********	PC_FORMAT_STA	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30 *********	PC_FORMAT_STP	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30 *********	PC_FORMAT_STR	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30 *********	PC_FORMAT_SUB	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
1992	PC_INST	(0:38)AUTOMATIC,INITIAL,LABEL 2106,2110,2116,2120,2136,2140,2144,2148,2152,2156,2165,2175,2180 2182,2208,2257,2260,2269,2299,2307,2313,2319,2323,2327,2331,2335 2339,2344,2367,2369,2371,2377,2409,2410,2412,2416,2418,2429,2432 2105,2176
34 *********	PC_MAX	IN P_CODE_STACK,AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 112,297,307,370,372,410,502,507,507,508,509,881,883,884,886,887,889 968,970,972,974,975,976,978,980,982,984,986,988,1100,1102,1103,1110 1112,1113,1236,1237,1244,1245,1252,1253,1320,1322,1323,1324,1398 1415,1428,1430,1431,1432,1435,1532,1533,1534,2047
30	PC_MNCODE_ADD	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNCODE_B	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNCODE_BAL	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNCODE_BEQ	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNCODE_BGE	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNCODE_BGT	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNCODE_BLE	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER

DCL NO.	IDENTIFIER	ATTRIBUTES AND REFERENCES
30	PC_MNCODE_BLT	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNCODE_BNE	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNCODE_CFN	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNCODE_DIV	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNCODE_DSL	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNCODE_END	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNCODE_EXP	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNCODE_FIX	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNCODE_FNC	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNCODE_FNX	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNCODE_FST	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNCODE_FSU	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNCODE_FUL	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNCODE_JMP	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNCODE_LCA	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNCODE_LCB	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNCODE_LDA	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNCODE_LDR	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNCODE_MUL	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNCODE_PCT	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNCODE_PRS	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNCODE_PRV	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNCODE_PTB	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER

DCL NO.	IDENTIFIER	ATTRIBUTES AND REFERENCES
30	PC_MNCODE_RDV	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNCODE_RET	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNCODE_RFN	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNCODE_RST	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNCODE_SLN	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNCODE_STA	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNCODE_STP	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNCODE_STR	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNCODE_SUB	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(4),CHARACTER
30	PC_MNEMONIC	IN PC_OPTAB(0:38) IN PC_CON_TABLE,BASED(PC_CON_TABLE_PTR),UNALIGNED, STRING(4),CHARACTER 376,381,386,391,396,401,404
34	PC_NUM	(500)IN P_CODE_STACK,AUTOMATIC,STRUCTURE
34 *********	PC_OBJECT	IN PC_NUM(500) IN P_CODE_STACK,AUTOMATIC,ALIGNED,BINARY,FIXED (15,0) 376,381,386,391,401,509,884,887,974,982,984,986,988,1323,1324,1431 1432,1435,1533,1534,2028,2053,2063,2065,2075,2085,2095,2106,2180 2183,2190,2195,2208,2208,2210,2214,2214,2215,2222,2225,2228,2236 2239,2242,2245,2248,2253,2255,2255,2261,2267,2270,2287,2297,2349 2361,2384,2416,2419
30 *********	PC_OP_CODE	IN PC_OPTAB(0:38) IN PC_CON_TABLE,BASED(PC_CON_TABLE_PTR),ALIGNED, BINARY,FIXED(15,0)
34 *********	PC_OPCODE	IN PC_NUM(500) IN P_CODE_STACK,AUTOMATIC,ALIGNED,BINARY,FIXED (15,0) 374,376,379,381,384,386,389,391,394,396,399,401,404,502,508,883,886 889,970,972,975,976,978,980,1102,1103,1112,1113,1236,1237,1244,1245 1252,1253,1322,1430,1532,2028,2068,2073,2083,2093,2105
30 *********	PC_OPCODE_ADD	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 1669

DCL NO.	IDENTIFIER	ATTRIBUTES AND REFERENCES
30	********* PC_OPCODE_B	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 747
30	********* PC_OPCODE_BAL	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 757
30	********* PC_OPCODE_BEQ	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 1119
30	********* PC_OPCODE_BGE	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 1144
30	********* PC_OPCODE_BGT	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 1134
30	********* PC_OPCODE_BLE	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 1139
30	********* PC_OPCODE_BLT	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 1129
30	********* PC_OPCODE_BNE	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 1124
30	********* PC_OPCODE_CFN	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 1806
30	********* PC_OPCODE_DIV	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 1667
30	********* PC_OPCODE_DSL	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 1862,1868
30	********* PC_OPCODE_END	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 620
30	********* PC_OPCODE_EXP	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 1665,1665
30	********* PC_OPCODE_FIX	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 1237,1238
30	********* PC_OPCODE_FNC	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 1800

DCL NO.	IDENTIFIER	ATTRIBUTES AND REFERENCES
30	********* PC_OPCODE_FNX	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 1284
30	********* PC_OPCODE_FST	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 1253,1254
30	********* PC_OPCODE_FSU	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 1226
30	********* PC_OPCODE_FUL	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 1245,1246
30	********* PC_OPCODE_JMP	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 1397
30	********* PC_OPCODE_LCA	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 1103
30	********* PC_OPCODE_LCB	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 1113
30	********* PC_OPCODE_LDA	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 886,970,1102,1112,1236,1244,1252,1682,1747,1799,1805,1863
30	********* PC_OPCODE_LDR	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 1861,1867
30	********* PC_OPCODE_MUL	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 1667
30	********* PC_OPCODE_PCT	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 910,942,943,959
30	********* PC_OPCODE_PRS	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 1021
30	********* PC_OPCODE_PRV	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 976,978,986,988
30	********* PC_OPCODE_PTB	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 975
30	********* PC_OPCODE_RDV	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 889

DCL NO.		IDENTIFIER	ATTRIBUTES AND REFERENCES
30	*********	PC_OPCODE_RET	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 636
30	*********	PC_OPCODE_RFN	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 1434
30	*********	PC_OPCODE_RST	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 723
30	*********	PC_OPCODE_SLN	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 305
30	*********	PC_OPCODE_STA	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 883,980,1322,1401,1427,1430,1532,1536,1683,1754,1801,1807,1864,1869
30	*********	PC_OPCODE_STP	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 628
30	*********	PC_OPCODE_STR	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)
30	*********	PC_OPCODE_SUB	IN PC_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 1669
30		PC_OPTAB	(0:38)IN PC_CON_TABLE,BASED(PC_CON_TABLE_PTR),STRUCTURE
501	*********	PCODE	PARAMETER,ALIGNED,BINARY,FIXED(15,0) 500,508
29	*********	PCT_LFEED	IN MISC_CODE_DEF,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 910,959,2183
29	*********	PCT_NOTAB	IN MISC_CODE_DEF,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 943,2195,2197
29	*********	PCT_TAB	IN MISC_CODE_DEF,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 942,2190,2192,2559
4		PGM_PAGE_NUM	AUTOMATIC,ALIGNED,INITIAL,DECIMAL,FIXED(5,0) 77,286,286,287,319,319,320
1548		POPULATE_STACK	ENTRY,DECIMAL,FLOAT(SINGLE) 1474
2010		PRINT_AREA	AUTOMATIC,STRUCTURE

DCL NO.	IDENTIFIER	ATTRIBUTES AND REFERENCES
2535	PRINT_BUFFER	ENTRY,DECIMAL,FLOAT(SINGLE) 2178,2180
2007	PRINT_E_FORMAT	AUTOMATIC,UNALIGNED,STRING(14),CHARACTER 2485,2486
428	PRINT_ERR	ENTRY,DECIMAL,FLOAT(SINGLE) 448,454,476,490,528,535,545,550,554,569,591,603,621,629,637,644,652 660,668,676,684,698,706,715,724,732,736,749,759,839,842,874,885,888 961,1002,1007,1067,1085,1091,1190,1220,1228,1278,1285,1294,1343,1370 1375,1383,1389,1405,1410,1418,1419,1450,1458,1460,1540,1545,1547 1599,1629,1642,1722,1809,1828,1890,1892,1926,1942,1948,1957,1970 1984
2574	PRINT_ERR	ENTRY,DECIMAL,FLOAT(SINGLE) 2044,2049,2055,2060,2078,2088,2100,2124,2130,2158,2168,2201,2218 2231,2267,2273,2281,2297,2301,2355,2379,2388,2392,2427,2432,2460
2479	PRINT_FIELD	AUTOMATIC,UNALIGNED,STRING(14),CHARACTER,VARYING 2483,2486,2504,2506
2009	PRINT_FIELD	AUTOMATIC,UNALIGNED,STRING(14),CHARACTER,VARYING 2177,2178
1999 *********	PRINT_LAST_PCT	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 2023,2187,2192,2197,2554,2559
2010	PRINT_LINE	IN PRINT_AREA,AUTOMATIC,UNALIGNED,STRING(120),CHARACTER,VARYING 2021,2542,2543,2543,2543,2544,2547,2550,2552,2552,2557,2559,2561 2563,2566,2568,2568,2571,2571,2584,2585
369	PRINT_PCODES	ENTRY,DECIMAL,FLOAT(SINGLE) 311,334
2616	PRINT_SYMBOLS	ENTRY,DECIMAL,FLOAT(SINGLE) 327,2614
1999 *********	PRINT_TAB_AMT	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 2022,2186,2227,2229,2229,2540,2542,2543,2544,2547,2553
2006	PRINT_WORK	AUTOMATIC,UNALIGNED,DECIMAL,PICTURE(------9.V999999) 2490,2504
2006	PRINT_WORK_CHAR	(14)AUTOMATIC,DEFINED,UNALIGNED,STRING(1),CHARACTER 2492,2495,2499,2502

DCL NO.	IDENTIFIER	ATTRIBUTES AND REFERENCES
2008	PRINT_ZERO	AUTOMATIC,UNALIGNED,INITIAL,STRING(2),CHARACTER,VARYING 2483
1329	PROCESS_DEF	ENTRY,DECIMAL,FLOAT(SINGLE) 677
1905	PROCESS_DIM	ENTRY,DECIMAL,FLOAT(SINGLE) 733
1154	PROCESS_FOR	ENTRY,DECIMAL,FLOAT(SINGLE) 707
1780	PROCESS_FUNCTION	ENTRY,DECIMAL,FLOAT(SINGLE) 1717
751	PROCESS_GOSUB	ENTRY,DECIMAL,FLOAT(SINGLE) 653
741	PROCESS_GOTO	ENTRY,DECIMAL,FLOAT(SINGLE) 645
1025	PROCESS_IF	ENTRY,DECIMAL,FLOAT(SINGLE) 699
607	PROCESS_KEYWORD	ENTRY,DECIMAL,FLOAT(SINGLE) 309
1287	PROCESS_LET	ENTRY,DECIMAL,FLOAT(SINGLE) 669
1256	PROCESS_NEXT	ENTRY,DECIMAL,FLOAT(SINGLE) 716
1731	PROCESS_OPERATORS	ENTRY,DECIMAL,FLOAT(SINGLE) 1665,1667,1669
348	PROCESS_OPTS	ENTRY,DECIMAL,FLOAT(SINGLE) 299
895	PROCESS_PRINT	ENTRY,DECIMAL,FLOAT(SINGLE) 691
993	PROCESS_PRINT_STR	ENTRY,DECIMAL,FLOAT(SINGLE) 939,953

DCL NO.		IDENTIFIER	ATTRIBUTES AND REFERENCES
962		PROCESS_PRINT_VAR	ENTRY,DECIMAL,FLOAT(SINGLE) 940,954
851		PROCESS_READ	ENTRY,DECIMAL,FLOAT(SINGLE) 685
872		PROCESS_READ_VAR	ENTRY,DECIMAL,FLOAT(SINGLE) 864,871
1837		PROCESS_SUBSCRIPT	ENTRY,DECIMAL,FLOAT(SINGLE) 1719
7		QUOTE_1	AUTOMATIC,UNALIGNED,INITIAL,STRING(1),CHARACTER 783,787,816,818,918,923,938,945,952,997,1453,1560,1573
7		QUOTE_2	AUTOMATIC,UNALIGNED,INITIAL,STRING(2),CHARACTER 1013
1440	*********	QUOTES	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1442,1454,1454,1459
2361		RECYCLE_FOR	STATEMENT LABEL CONSTANT 2350
12		REF_LINE_NUM	AUTOMATIC,ALIGNED,DECIMAL,FIXED(5,0) 558,746,756,1115
2000		REGISTER	AUTOMATIC,ALIGNED,DECIMAL,FLOAT(SINGLE) 2013,2030,2116,2118,2136,2138,2412,2414
157		RENUM	ENTRY,DECIMAL,FLOAT(SINGLE) 52
168		RENUMFL	FILE,EXTERNAL,STREAM,OUTPUT 172,244
1335		RIGHT_SIDE	AUTOMATIC,UNALIGNED,STRING(80),CHARACTER,VARYING 1353,1420,1423,1425,1425,1426
1290		RIGHT_SIDE	AUTOMATIC,UNALIGNED,STRING(80),CHARACTER,VARYING 1304,1313,1316,1318,1318,1319
1028		RIGHT_SIDE	AUTOMATIC,UNALIGNED,STRING(80),CHARACTER,VARYING 1041,1070,1081,1081,1105,1108

DCL NO.		IDENTIFIER	ATTRIBUTES AND REFERENCES
2588		RND	ENTRY,DECIMAL,FLOAT(SINGLE) 2247
1610	*********	RP	PARAMETER,ALIGNED,BINARY,FIXED(15,0) 1609,1615,1616,1651,1655,1688,1707
1469	*********	RP	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1507,1509,1510
14		RUN_DATE	AUTOMATIC,UNALIGNED,STRING(10),CHARACTER 37,38,38,38,38,287,320
33	*********	SC_CUR	IN SOURCE_CODE,AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 78,293,294,295,295,296
33	*********	SC_MAX	IN SOURCE_CODE,AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 47,78,81,81,82,87,96,96,97,102,173,294
1609		SIMPLIFY_SUB_STACK	ENTRY,DECIMAL,FLOAT(SINGLE) 1510
1724		SIMPLIFY_SUB_STACK_EXIT	STATEMENT LABEL CONSTANT 1621
		SIN	GENERIC,BUILT-IN FUNCTION 2241
413		SKIP_BLANKS	ENTRY,DECIMAL,FLOAT(SINGLE) 566,618,626,634,642,650,658,666,674,682,690,696,704,713,721,730,743 753,1089,1094,1274,1930,1973
33		SOURCE_AREA	(500)IN SOURCE_CODE,AUTOMATIC,STRUCTURE
33		SOURCE_CODE	AUTOMATIC,STRUCTURE
33		SOURCE_LINE	IN SOURCE_AREA(500) IN SOURCE_CODE,AUTOMATIC,ALIGNED,STRING(80), CHARACTER 82,87,97,102,174,242,244,296
		SQRT	GENERIC,BUILT-IN FUNCTION 2221
28	*********	SS_CODE	IN SS_TAB(0:8) IN SS_CON_TABLE,BASED(SS_CON_TABLE_PTR),ALIGNED, BINARY,FIXED(15,0)

DCL NO.		IDENTIFIER	ATTRIBUTES AND REFERENCES
28		SS_CON_TABLE	BASED(SS_CON_TABLE_PTR),STRUCTURE
		SS_CON_TABLE_PTR	AUTOMATIC,ALIGNED,POINTER 42,2619
27	*********	SS_CONST	IN SS_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 495,1791,1849
27		SS_CONST_DESC	IN SS_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(8),CHARACTER
27		SS_CONSTANTS	STATIC,STRUCTURE 42
35	*********	SS_CUR	IN SYMBOL_STACK,AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 155,1017,1018,1019,1020,1021
27		SS_DEF_DESC	IN SS_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(8),CHARACTER
27	*********	SS_DEF_VAR	IN SS_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 1396,1716,1803
28		SS_DESC	IN SS_TAB(0:8) IN SS_CON_TABLE,BASED(SS_CON_TABLE_PTR),UNALIGNED, STRING(8),CHARACTER 2619
27		SS_DIM_DESC	IN SS_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(8),CHARACTER
27	*********	SS_DIM_VAR	IN SS_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 447,483,1538,1639,1718,1851,1853,1923,1953
27	*********	SS_FUNC	IN SS_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 120,124,128,132,136,140,144,148,152,1639,1716,1797
27		SS_FUNC_DESC	IN SS_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(8),CHARACTER
35	*********	SS_MAX	IN SYMBOL_STACK,AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 155,444,452,457,457,458,459,460,468,470,472,473,481,483,484,494,495 498,1005,1017,1017,1393,1523,1953,1955,1960,1962,1962,1963,1964,1965 1966,2618
35	*********	SS_MAX_FNC	IN SYMBOL_STACK,AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 155
27	*********	SS_STRCON	IN SS_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0)

DCL NO.		IDENTIFIER	ATTRIBUTES AND REFERENCES
			470,1020,1538,2075,2085,2095,2120,2127,2436,2436,2446,2447,2447,2620
27		SS_STRCON_DESC	IN SS_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(8),CHARACTER
27	*********	SS_STRDIM	IN SS_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 472,1538,1718,1851,1923,1953,2121,2620
27		SS_STRDIM_DESC	IN SS_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(8),CHARACTER
27	*********	SS_STRVAR	IN SS_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 473,1538,2121,2620
27		SS_STRVAR_DESC	IN SS_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(8),CHARACTER
28		SS_TAB	(0:8)IN SS_CON_TABLE,BASED(SS_CON_TABLE_PTR),STRUCTURE
27		SS_UNKNWM_DESC	IN SS_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(8),CHARACTER
27	*********	SS_UNKNWN	IN SS_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 2620
27	*********	SS_VAR	IN SS_CONSTANTS,STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 116,484,1225,1283,1394,1530,1720,1791,1849
27		SS_VAR_DESC	IN SS_CONSTANTS,STATIC,ALIGNED,INITIAL,STRING(8),CHARACTER
1472		STACK	AUTOMATIC,STRUCTURE
1472	*********	STACK_CUR	IN STACK,AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1473
1472	*********	STACK_MAX	IN STACK,AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 1473,1482,1491,1497,1507,1511,1520,1579,1579,1580,1581,1586,1588 1592,1604,1620,1622,1673,1675,1679,1689,1694,1697,1700,1701,1701 1702,1702,1708,1727,1737,1756,1759,1759,1765,1775,1787,1810,1814 1817,1817,1822,1825,1825,1832,1845,1872,1876,1879,1879,1884,1887 1887,1896,1897,1898
17		STACK_PRINT_DEBUG	AUTOMATIC,ALIGNED,INITIAL,STRING(1),BIT 73,356,358,823,1480,1485,1567,1601,1613,1634,1661,1686,1705,1724 1734,1762,1772,1784,1829,1842,1893
1909		START_VAL	AUTOMATIC,UNALIGNED,STRING(80),CHARACTER,VARYING
1260		START_VAL	AUTOMATIC,UNALIGNED,STRING(80),CHARACTER,VARYING

DCL NO.	IDENTIFIER	ATTRIBUTES AND REFERENCES
1158	START_VAL	AUTOMATIC,UNALIGNED,STRING(80),CHARACTER,VARYING 1163,1186,1186,1231,1234,1234,1235
1909	STEP_VAL	AUTOMATIC,UNALIGNED,STRING(80),CHARACTER,VARYING
1260	STEP_VAL	AUTOMATIC,UNALIGNED,STRING(80),CHARACTER,VARYING
1158	STEP_VAL	AUTOMATIC,UNALIGNED,STRING(80),CHARACTER,VARYING 1163,1214,1214,1217,1247,1250,1250,1251
1161	STEP_WORD	AUTOMATIC,UNALIGNED,STRING(4),CHARACTER 1197,1198
9	STMT	IN STMT_IN,AUTOMATIC,UNALIGNED,STRING(80),CHARACTER 296,297,298,349,351,353,355,357,359,361,363,365,418,522,575,778,860 915,957,957,1036,1041,1048,1073,1074,1166,1178,1179,1196,1197,1211 1267,1292,1300,1304,1339,1349,1353,1916,1934,1977
10	STMT_BUFF	AUTOMATIC,UNALIGNED,STRING(80),CHARACTER 43,67,69,76,79,87,88,89,102,103
9 *********	STMT_CH	IN STMT_IN,AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 301,417,421,426,521,545,550,554,559,567,569,574,603,605,619,621,627 629,635,637,643,644,651,652,659,660,667,668,675,676,683,684,697,698 705,706,712,714,722,724,729,731,736,744,749,754,759,777,859,874,885 888,907,913,961,1002,1007,1035,1047,1052,1057,1062,1067,1072,1085 1088,1090,1091,1165,1175,1177,1190,1193,1195,1209,1210,1218,1220 1228,1266,1272,1273,1275,1292,1294,1299,1339,1343,1348,1375,1389 1450,1458,1460,1540,1545,1547,1599,1629,1642,1722,1809,1828,1890 1892,1915,1926,1929,1933,1942,1948,1951,1957,1970,1974,1977,1980 1980,1984
9	STMT_IN	AUTOMATIC,STRUCTURE
9 *********	STMT_LEFT	IN STMT_IN,AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 107,301
9 *********	STMT_RIGHT	IN STMT_IN,AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 108,207,216,417,426,521,567,574,619,627,635,643,651,659,667,675,683 697,705,714,722,731,744,754,777,859,907,914,1035,1072,1090,1165,1177 1195,1210,1218,1266,1275,1299,1348,1915,1933,1974
280	STR_CNT	AUTOMATIC,UNALIGNED,DECIMAL,PICTURE(99) 292,820,821,821,1563,1564,1564

DCL NO.		IDENTIFIER	ATTRIBUTES AND REFERENCES
438	*********	STR_IND	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 463,464,466
1550		STR_WORK	AUTOMATIC,UNALIGNED,STRING(80),CHARACTER,VARYING 1554,1568,1569,1571,1571
35		STRING_VAL	IN SYMBOL_AREA(100) IN SYMBOL_STACK,AUTOMATIC,ALIGNED,STRING(80), CHARACTER,VARYING 118,122,126,130,134,138,142,146,150,154,391,460,826,827,1005,1019 1569,1966,2122,2122,2173,2173,2180,2446,2446,2449,2449,2451,2451 2453,2453,2466,2467,2621
		SUBSTR	GENERIC,BUILT-IN FUNCTION 38,38,38,67,76,89,175,184,191,191,208,213,213,218,225,230,237,237 298,349,351,353,355,357,359,361,363,365,418,461,461,466,469,522,575 778,816,818,826,860,884,887,915,938,952,957,957,997,1010,1013,1014 1014,1036,1041,1048,1073,1074,1166,1178,1179,1196,1197,1211,1267 1292,1300,1304,1316,1323,1339,1349,1353,1365,1367,1378,1380,1423 1431,1444,1446,1453,1479,1479,1533,1557,1916,1934,1963,1977,2486 2504,2543,2547,2568,2595
35	*********	SYM_DIM_MAX	IN SYMBOL_AREA(100) IN SYMBOL_STACK,AUTOMATIC,ALIGNED,BINARY,FIXED (15,0) 459,1960,2053,2619
35	*********	SYM_TYPE	IN SYMBOL_AREA(100) IN SYMBOL_STACK,AUTOMATIC,ALIGNED,BINARY,FIXED (15,0) 116,120,124,128,132,136,140,144,148,152,447,470,472,473,483,484,495 1020,1225,1283,1394,1396,1530,1538,1538,1538,1538,1639,1639,1716 1716,1718,1718,1720,1791,1791,1797,1803,1849,1849,1851,1851,1853 1923,1923,1953,1953,1964,1964,2075,2085,2095,2111,2114,2121,2121 2127,2308,2314,2619,2620,2620,2620,2620
35		SYM_VALUE	IN SYMBOL_AREA(100) IN SYMBOL_STACK,AUTOMATIC,ALIGNED,DECIMAL, FLOAT(SINGLE) 117,121,125,129,133,137,141,145,149,153,468,481,494,1965,2110,2116 2128,2136,2140,2144,2148,2152,2156,2161,2172,2177,2214,2222,2225 2228,2236,2239,2242,2245,2248,2253,2255,2307,2313,2367,2369,2371 2372,2396,2622
35		SYMBOL	IN SYMBOL_AREA(100) IN SYMBOL_STACK,AUTOMATIC,ALIGNED,STRING(10), CHARACTER 115,119,123,127,131,135,139,143,147,151,376,445,452,458,884,887,974 984,1018,1323,1431,1524,1533,1870,1955,1963,2114,2118,2138,2142,2168

DCL NO.		IDENTIFIER	ATTRIBUTES AND REFERENCES
			2210,2214,2255,2419,2619
35		SYMBOL_AREA	(100)IN SYMBOL_STACK,AUTOMATIC,STRUCTURE 2134
35		SYMBOL_STACK	AUTOMATIC,STRUCTURE
		SYSIN	FILE,EXTERNAL 39,43,64,79,88,103
		SYSPRINT	FILE,EXTERNAL 84,99,181,197,204,281,284,287,288,289,291,297,314,317,320,321,328 330,332,333,335,336,339,341,344,346,376,381,386,391,396,401,404,431 504,824,1148,1481,1486,1518,1568,1603,1605,1615,1617,1636,1662,1688 1690,1707,1709,1726,1728,1736,1738,1764,1766,1774,1776,1786,1788 1831,1833,1844,1846,1895,1897,1899,1991,2026,2027,2028,2029,2030 2031,2032,2033,2065,2108,2114,2118,2134,2138,2142,2146,2150,2154 2163,2210,2214,2255,2258,2270,2293,2311,2317,2365,2375,2414,2446 2474,2577,2579,2580,2584,2617,2619,2621,2622,2624
2509	*********	T	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 2523,2527,2530
1999	*********	TAB_AMT	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0)
1999	*********	TAB_POS	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0)
20		TABLE_DUMP	AUTOMATIC,ALIGNED,INITIAL,STRING(1),BIT 72,354,2034,2578,2613
19		TABLE_PRINT	AUTOMATIC,ALIGNED,INITIAL,STRING(1),BIT 71,323,350,352,2578
		TAN	GENERIC,BUILT-IN FUNCTION 2244
2465		TEMP_A	AUTOMATIC,UNALIGNED,INITIAL,STRING(80),CHARACTER 2466,2468,2470,2474
2465		TEMP_B	AUTOMATIC,UNALIGNED,INITIAL,STRING(80),CHARACTER 2467,2468,2470,2474
1336		TEMP_NAME	AUTOMATIC,UNALIGNED,STRING(10),CHARACTER 1399

DCL NO.		IDENTIFIER	ATTRIBUTES AND REFERENCES
1839		TEMP_SYM	AUTOMATIC,UNALIGNED,INITIAL,STRING(10),CHARACTER 1857,1858
1782		TEMP_SYM	AUTOMATIC,UNALIGNED,INITIAL,STRING(10),CHARACTER 1795,1796
2611		TERMINATE	ENTRY,DECIMAL,FLOAT(SINGLE) 59,2035
278		TERMINATE_SCAN	AUTOMATIC,ALIGNED,STRING(1),BIT 302,433,877,964,1096,1106,1147,1232,1240,1248,1295,1311,1314,1360 1421
1031		THEN_WORD	AUTOMATIC,UNALIGNED,STRING(4),CHARACTER 1074,1075
994	*********	TICS	AUTOMATIC,ALIGNED,BINARY,FIXED(15,0) 995,998,998,1000
		TIME	BUILT-IN FUNCTION 2595
280		TMP_CNT	AUTOMATIC,UNALIGNED,DECIMAL,PICTURE(99) 519,1338,1744,1745,1745,1793,1794,1794,1854,1855,1856,1856
1840		TMP_VAR	AUTOMATIC,UNALIGNED,INITIAL,STRING(10),CHARACTER 1854,1855,1857,1870,1874,1882
1612		TMP_VAR	AUTOMATIC,UNALIGNED,STRING(5),CHARACTER 1744,1752,1793,1795,1812,1820
1551		TMP_VAR	AUTOMATIC,UNALIGNED,STRING(10),CHARACTER 1563,1565,1566,1568
811		TMP_VAR	AUTOMATIC,UNALIGNED,STRING(10),CHARACTER 820,822,824
1909		TO_VAL	AUTOMATIC,UNALIGNED,STRING(80),CHARACTER,VARYING
1260		TO_VAL	AUTOMATIC,UNALIGNED,STRING(80),CHARACTER,VARYING
1158		TO_VAL	AUTOMATIC,UNALIGNED,STRING(80),CHARACTER,VARYING 1163,1204,1204,1239,1242,1242,1243
1162		TO_WORD	AUTOMATIC,UNALIGNED,STRING(2),CHARACTER

DCL NO.	IDENTIFIER	ATTRIBUTES AND REFERENCES
		1179,1180
247	TRIM_EDIT_NUM	ENTRY,DECIMAL,FLOAT(SINGLE) 190,236
22	TRUE	STATIC,ALIGNED,INITIAL,STRING(1),BIT 40,65,91,185,200,215,222,231,303,350,354,356,360,364,416,433,520,539 573,596,774,790,800,858,904,925,1033,1071,1083,1147,1164,1176,1188 1194,1208,1265,1298,1347,1492,1495,1512,1575,1914,1932,2034,2581
	TRUNC	GENERIC,BUILT-IN FUNCTION 2235,2251,2252
1467	V	AUTOMATIC,UNALIGNED,STRING(10),CHARACTER,VARYING 1555,1565,1568,1580,1582,1584,1584,1589
436	V	PARAMETER,UNALIGNED,STRING(10),CHARACTER 435,445,458,461,461,463,466,469,488,490,494
	V	AUTOMATIC,ALIGNED,DECIMAL,FLOAT(SINGLE) 834
764	VAL	AUTOMATIC,UNALIGNED,STRING(80),CHARACTER,VARYING 776,782,782,789,789,801,804,804,815,816,818,818,824,825,826,826,836
25	VALID_VAR_CHARS	STATIC,UNALIGNED,INITIAL,STRING(37),CHARACTER 463,1368,1381
898	VAR	AUTOMATIC,UNALIGNED,STRING(10),CHARACTER
854	VAR	AUTOMATIC,UNALIGNED,STRING(10),CHARACTER
	VERIFY	GENERIC,BUILT-IN FUNCTION 463,488,834,1368,1381
1472	WORD	IN ITEMS(50) IN STACK,AUTOMATIC,UNALIGNED,STRING(10),CHARACTER 1483,1484,1522,1524,1545,1580,1589,1593,1605,1617,1626,1627,1638 1671,1677,1677,1680,1690,1693,1693,1709,1715,1728,1738,1746,1748 1752,1753,1766,1776,1783,1788,1812,1820,1833,1841,1846,1874,1882 1899
13	WORD	AUTOMATIC,UNALIGNED,STRING(8),CHARACTER 447,471,482,595,598,612,616,624,632,640,648,656,664,672,680,688,694 702,710,719,727,736

DCL NO.		IDENTIFIER	ATTRIBUTES AND REFERENCES
2589		YFL	AUTOMATIC,ALIGNED,DECIMAL,FLOAT(SINGLE) 2606,2607,2607,2608
24	*********	ZERO	STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 620,628,636,723,1397,1862,1868
2590	*********	ZERO	STATIC,ALIGNED,INITIAL,BINARY,FIXED(15,0) 2593,2598,2602

AGGREGATE LENGTH TABLE

STATEMENT NO.	IDENTIFIER	LENGTH IN BYTES
31	DATA_STACK	4002
36	DEF_FUNCTIONS	144
2005	FOR_STACK	164
2004	GOSUB_STACK	104
26	KEY_WORD_AREA	128
26	KEY_WORDS	128
1993	LIB_FNC	72
32	LINE_STACK	3004
29	MISC_CODE_DEF	24
165	NEW_LINE_NUM	1500
34	P_CODE_STACK	2004
30	PC_CON_TABLE	389
30	PC_CONSTANTS	389
1992	PC_INST	312
2010	PRINT_AREA	120
2006	PRINT_WORK_CHAR	14
33	SOURCE_CODE	40004
28	SS_CON_TABLE	90
27	SS_CONSTANTS	90
1472	STACK	554
9	STMT_IN	86

STATEMENT NO.	IDENTIFIER	LENGTH IN BYTES
35	SYMBOL_STACK	10004

STORAGE REQUIREMENTS.

THE STORAGE AREA FOR THE PROCEDURE LABELLED BASIC IS 61132 BYTES LONG.

THE STORAGE AREA FOR THE ON UNIT AT STATEMENT NO. 39 IS 232 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED MONITOR IS 236 BYTES LONG.

THE STORAGE AREA FOR THE ON UNIT AT STATEMENT NO. 64 IS 232 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED INITIALIZE IS 280 BYTES LONG.

THE STORAGE AREA FOR THE PROCEDURE LABELLED RENUM IS 2016 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED TRIM_EDIT_NUM IS 224 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED COMPILE IS 316 BYTES LONG.

THE STORAGE AREA FOR THE ON UNIT AT STATEMENT NO. 282 IS 240 BYTES LONG.

THE STORAGE AREA FOR THE ON UNIT AT STATEMENT NO. 315 IS 236 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED PROCESS_OPTS IS 224 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED PRINT_PCODES IS 292 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED SKIP_BLANKS IS 244 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED PRINT_ERR IS 236 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED LOOKUP_SYMBOL_TABLE IS 368 BYTES LONG.

THE STORAGE AREA FOR THE ON UNIT AT STATEMENT NO. 440 IS 240 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED ADD_PCODE IS 244 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED GET_STMT_NUM IS 356 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED GET_KEYWORD IS 332 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED PROCESS_KEYWORD IS 384 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED PROCESS_GOTO IS 280 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED PROCESS_GOSUB IS 280 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED EXTRACT_DATA IS 532 BYTES LONG.

THE STORAGE AREA FOR THE ON UNIT AT STATEMENT NO. 770 IS 240 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED EXTRACT_DATA_ITEM IS 524 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED PROCESS_READ IS 436 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED PROCESS_READ_VAR IS 436 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED PROCESS_PRINT IS 444 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED PROCESS_PRINT_VAR IS 420 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED PROCESS_PRINT_STR IS 468 BYTES LONG.

THE STORAGE AREA FOR THE PROCEDURE LABELLED PROCESS_IF IS 656 BYTES LONG.

THE STORAGE AREA FOR THE PROCEDURE LABELLED PROCESS_FOR IS 868 BYTES LONG.

THE STORAGE AREA FOR THE PROCEDURE LABELLED PROCESS_NEXT IS 740 BYTES LONG.

THE STORAGE AREA FOR THE PROCEDURE LABELLED PROCESS_LET IS 632 BYTES LONG.

THE STORAGE AREA FOR THE PROCEDURE LABELLED PROCESS_DEF IS 848 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED BALANCE_STMT IS 288 BYTES LONG.

THE STORAGE AREA FOR THE PROCEDURE LABELLED PARSE_EXP IS 1024 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED POPULATE_STACK IS 468 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED SIMPLIFY_SUB_STACK IS 364 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED PROCESS_OPERATORS IS 280 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED PROCESS_FUNCTION IS 324 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED PROCESS_SUBSCRIPT IS 352 BYTES LONG.

THE STORAGE AREA FOR THE PROCEDURE LABELLED PROCESS_DIM IS 792 BYTES LONG.

THE STORAGE AREA FOR THE PROCEDURE LABELLED EXECUTE IS 1704 BYTES LONG.

THE STORAGE AREA FOR THE ON UNIT AT STATEMENT NO. 2025 IS 248 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED COMPARE_RTN IS 284 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED COMPARE_DIF_LEN IS 404 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED FORMAT_NUMBER IS 288 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED COMPRESS_FS IS 240 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED PRINT_BUFFER IS 516 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED PRINT_ERR IS 236 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED FLUSH_BUFFER IS 224 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED RND IS 280 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED TERMINATE IS 232 BYTES LONG.

THE STORAGE AREA (IN STATIC) FOR THE PROCEDURE LABELLED PRINT_SYMBOLS IS 264 BYTES LONG.

THE PROGRAM CSECT IS NAMED BASIC   AND IS 59602 BYTES LONG.

THE STATIC CSECT IS NAMED **BASICA AND IS 25344 BYTES LONG.

STATISTICS   MACRO RECORDS =   4426,SOURCE RECORDS =   4620,PROG TEXT STMNTS =   2626,OBJECT BYTES =   59602

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN ON UNIT

OFFSET (HEX)	0000	0048	0058
STATEMENT NO	40	40	

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN ON UNIT

OFFSET (HEX)	0000	0048	0058
STATEMENT NO	65	65	

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE MONITOR

OFFSET (HEX)	0000	0054	0062	0070	007E	007E	0084	008A	0090	0096	009C	00A2	00A8	00C8	00D2	00E2	011C	0174	0184	01A2	01A2
STATEMENT NO	63	64	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84

OFFSET (HEX)	01C8	01CE	01CE	01E8	021E	0234	0234	023A	023A	023E	0242	025C	026C	0286	0286	02AC	02B2	02B2	02CC	0302	0306
STATEMENT NO	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102	103	104	105

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE INITIALIZE

OFFSET (HEX)	0000	00E4	00EE	00F4	00FA	0106	0116	012A	013E	0148	0162	017E	0196	01D0	01DE	01FA	0212	024C	025A	0276	028E
STATEMENT NO	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126

OFFSET (HEX)	02C8	02D6	02F2	030A	0344	0352	036E	0386	03C0	03CE	03EA	0402	043C	044A	0466	047E	04B8	04C6	04E2	04FA	0534
STATEMENT NO	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147

OFFSET (HEX)	0542	055E	0576	05B0	05C6	05E6	0602	0630	064E
STATEMENT NO	148	149	150	151	152	153	154	155	156

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE TRIM_EDIT_NUM

OFFSET (HEX)	0000	003C	003C	005A	005A	0060	0064	0064	0082	0082	0088	008C	008C	00AA	00AA	00B0	00B4	00B4	00D2	00D2	00D8
STATEMENT NO	247	248	249	250	251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267

OFFSET (HEX)	00DC	00DC	00E6	00E6	00E6	00E6	00E6
STATEMENT NO	268	269	270	271	272	273	274

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE RENUM

OFFSET (HEX)	0000	0074	0088	00AA	00BE	0106	011A	0134	0146	014A	014A	0170	017C	017C	01A2	01A8	01A8	01E2	01E8	0226	0244
STATEMENT NO	157	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188

OFFSET (HEX) 0244 0262 026C 02DA 02E0 02E0 02E6 02EE 02EE 0314 031A 031A 0320 0346 0352 0352 0378 037E 037E 03C8 03E8
STATEMENT NO 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209

OFFSET (HEX) 03EC 03F2 03F8 0404 0468 0468 046E 0478 0480 04A0 04B0 04B6 04BA 04C0 04C6 04CE 04EE 04F8 0504 0508 050E
STATEMENT NO 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230

OFFSET (HEX) 055A 0560 059E 05BC 05BC 05DA 05E4 064A 0650 0650 0656 0656 0670 0670 06D8 06EC 06F6
STATEMENT NO 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 275

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN ON UNIT

OFFSET (HEX) 0000 0048 005A 006E 007C 0082 014A 01B4 01C8
STATEMENT NO 282 283 284 285 286 287 288 289 290

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN ON UNIT

OFFSET (HEX) 0000 0048 005A 006E 007C 0082 014A 015E
STATEMENT NO 315 316 317 318 319 320 321 322

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE PROCESS_OPTS

OFFSET (HEX) 0000 003C 0052 005C 0072 007C 0092 009C 00B2 00BC 00D2 00DC 00F2 00FC 0112 0118 012E 0138 014E 0158 0158
STATEMENT NO 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE PRINT_PCODES

OFFSET (HEX) 0000 0088 009C 009C 00B8 00B8 00EC 00EC 01D6 01DA 01DA 020E 020E 02E8 02EC 02EC 0320 0320 03EA 03EE 03EE
STATEMENT NO 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389

OFFSET (HEX) 0422 0422 04F8 04FC 04FC 0530 0530 05EA 05EE 05EE 0622 0622 06F0 06F4 06F4 07AE 07B4 07B4 07B4 07B4 07D0
STATEMENT NO 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410

OFFSET (HEX) 07DA 07DA
STATEMENT NO 411 412

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE SKIP_BLANKS

OFFSET (HEX) 0000 0038 003E 0084 00A8 00AC 00AC 00B6 00BC 00BC 00C2 00CA 00DA
STATEMENT NO 413 416 417 418 419 420 421 422 423 424 425 426 427

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE PRINT_ERR

OFFSET (HEX) 0000 0044 00E8 00F6 0100
STATEMENT NO 428 431 432 433 434

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN ON UNIT

OFFSET (HEX) 0000 0048 0074 0098
STATEMENT NO 440 441 442 443

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE LOOKUP_SYMBOL_TABLE

OFFSET (HEX) 0000 00A8 00B6 00CA 00F0 00F0 0142 016A 017C 017C 0190 01B2 01B2 01E0 01F0 01F0 0200 0228 023C 0262 02AE
STATEMENT NO 435 439 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462

OFFSET (HEX) 02AE 0318 0324 0324 034C 034C 0364 0372 038E 03A0 03BC 03D8 03DC 03DC 0404 0414 0414 0418 0418 0430 0442
STATEMENT NO 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483

OFFSET (HEX) 045A 0476 0476 047A 047A 04B6 04B6 04F2 0502 0506 0506 0534 0548 0548 0548 055E
STATEMENT NO 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE ADD_PCODE

OFFSET (HEX) 0000 006C 0092 0092 00C8 00CE 00CE 00DE 00F6 010E
STATEMENT NO 500 502 503 504 505 506 507 508 509 510

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE GET_STMT_NUM

OFFSET (HEX) 0000 0054 005A 0068 006E 00B4 00D0 00DE 00E8 0128 0128 012E 0156 015A 015A 01A0 01AC 01AC 01B2 01DA 01DA
STATEMENT NO 511 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537

OFFSET (HEX) 01DA 01E0 01FE 01FE 0218 0226 0226 022C 0254 0258 0266 0266 026C 0294 0298 0298 02BA 02E2 02E2 02EC 02F0
STATEMENT NO 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558

OFFSET (HEX) 030E 0318
STATEMENT NO 559 560

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE GET_KEYWORD

OFFSET (HEX) 0000 0044 004E 005E 005E 0086 008C 008C 0092 0098 00DE 00FA 0108 0108 0114 0114 0130 0134 013A 013E 0144
STATEMENT NO 561 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585

OFFSET (HEX) 0148 0148 018E 019A 019A 01A0 01C8 01C8 01C8 01CE 01E0 01E6 022C 024A 0250 0256 025E 025E 028A 028A 0294
STATEMENT NO 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE PROCESS_KEYWORD

OFFSET (HEX)	0000	003C	003C	004E	004E	0052	0052	0064	0064	006E	007E	0090	00BC	00C0	00C0	00D2	00D2	00DC	00EC	00FE	012A
STATEMENT NO	607	611	612	613	614	615	616	617	618	619	620	621	622	623	624	625	626	627	628	629	630
OFFSET (HEX)	012E	012E	0140	0140	014A	015A	016C	0198	019C	019C	01AE	01AE	01B8	01C8	01F4	01FE	0202	0202	0214	0214	021E
STATEMENT NO	631	632	633	634	635	636	637	638	639	640	641	642	643	644	645	646	647	648	649	650	651
OFFSET (HEX)	022E	025A	0264	0268	0268	027A	027A	0284	0294	02C0	02CA	02CE	02CE	02E0	02E0	02EA	02FA	0326	0330	0334	0334
STATEMENT NO	652	653	654	655	656	657	658	659	660	661	662	663	664	665	666	667	668	669	670	671	672
OFFSET (HEX)	0346	0346	0350	0360	038C	0396	039A	039A	03AC	03AC	03B6	03C6	03F2	03FC	0400	0400	0412	0412	041C	0426	042A
STATEMENT NO	673	674	675	676	677	678	679	680	681	682	683	684	685	686	687	688	689	690	691	692	693
OFFSET (HEX)	042A	043C	043C	0446	0456	0482	048C	0490	0490	04A2	04A2	04AC	04BC	04E8	04F2	04F6	04F6	0508	0508	050E	0518
STATEMENT NO	694	695	696	697	698	699	700	701	702	703	704	705	706	707	708	709	710	711	712	713	714
OFFSET (HEX)	0528	0554	055E	0562	0562	0574	0574	057E	058E	05A0	05CC	05D0	05D0	05E2	05E2	05E8	05F2	0602	062E	0638	063C
STATEMENT NO	715	716	717	718	719	720	721	722	723	724	725	726	727	728	729	730	731	732	733	734	735
OFFSET (HEX)	063C	067A	067A	067A	067A																
STATEMENT NO	736	737	738	739	740																

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE PROCESS_GOTO

OFFSET (HEX)	0000	003C	004A	0054	0064	0064	0076	0096	009A	00C6
STATEMENT NO	741	742	743	744	745	746	747	748	749	750

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE PROCESS_GOSUB

OFFSET (HEX)	0000	003C	004A	0054	0064	0064	0076	0096	009A	00C6
STATEMENT NO	751	752	753	754	755	756	757	758	759	760

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN ON UNIT

OFFSET (HEX)	0000	0048	0052	0076
STATEMENT NO	770	771	772	773

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE EXTRACT_DATA_ITEM

OFFSET (HEX)	0000	004C	0052	005C	0080	0096	0096	00B2	00B2	00DC	00FE	0122	012A	014E	015E	01BA	01D0	01D4	01D4	01DE	01DE
STATEMENT NO	810	813	814	815	816	817	818	819	820	821	822	823	824	825	826	827	828	829	830	831	832

```
OFFSET (HEX)  01E2 01E2 0234 0242 0260 0260 027A 02D6 02D6 02F4 0324 0324 0334 0344 0358 0358 0358
STATEMENT NO   833  834  835  836  837  838  839  840  841  842  843  844  845  846  847  848  849


TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE          EXTRACT_DATA

OFFSET (HEX)  0000 0044 0052 0058 0058 005E 007A 0096 0096 009E 009E 00E4 00F2 00F8 00FC 00FC 010A 010A 0150 0156 015A
STATEMENT NO   761  769  774  775  776  777  778  779  780  781  782  783  784  785  786  787  788  789  790  791  792

OFFSET (HEX)  015A 0168 0168 016C 016C 017A 017A 0184 018A 0190 0194 0194 01DA 01DA 01DA 01DA 01EE 01F8
STATEMENT NO   793  794  795  796  797  798  799  800  801  802  803  804  805  806  807  808  809  850


TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE          PROCESS_READ_VAR

OFFSET (HEX)  0000 0074 0084 00B4 00B4 00CE 00DA 00E0 0144 0164 0174 0174 0190 01C2 01E6 0206 0238 0260 0280 0280 0280
STATEMENT NO   872  873  874  875  876  877  878  879  880  881  882  883  884  885  886  887  888  889  890  891  892

OFFSET (HEX)  028A
STATEMENT NO   893


TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE          PROCESS_READ

OFFSET (HEX)  0000 003C 0042 0048 0064 0080 008E 008E 0094 009E 00A2 00B0 00B4 00FA 010E 011A 0124
STATEMENT NO   851  857  858  859  860  861  862  863  864  865  866  867  868  869  870  871  894


TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE          PROCESS_PRINT_VAR

OFFSET (HEX)  0000 0074 008E 009A 00A0 0104 0124 0134 0134 0150 0150 017E 017E 01B2 01C6 01E2 01E6 0202 0202 0222 0222
STATEMENT NO   962  963  964  965  966  967  968  969  970  971  972  973  974  975  976  977  978  979  980  981  982

OFFSET (HEX)  0254 0254 0284 0288 02B8 02BC 02EC 02EC 02EC 02F6
STATEMENT NO   983  984  985  986  987  988  989  990  991  992


TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE          PROCESS_PRINT_STR

OFFSET (HEX)  0000 0084 008A 009E 00C2 00CE 00E2 0102 0102 012E 0134 0134 0156 0156 017E 0184 0184 01F4 01FA 020C 022C
STATEMENT NO   993  995  996  997  998  999 1000 1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 1011 1012 1013 1014

OFFSET (HEX)  02BC 02C8 02CC 02E4 0302 0342 035E 037A 0384
STATEMENT NO  1015 1016 1017 1018 1019 1020 1021 1022 1023


TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE          PROCESS_PRINT
```

OFFSET (HEX)	0000	003C	0042	0048	004E	007A	0080	008C	008C	0092	00A0	00A6	00A6	00B0	00C0	00D8	00E0	00E0	00EA	00F0	0136
STATEMENT NO	895	902	903	904	905	906	907	908	909	910	911	912	913	914	915	916	917	918	919	920	921
OFFSET (HEX)	013A	013A	0148	0148	014E	0194	0194	0194	019C	01A0	01A0	01AE	01B2	01B8	01F8	01F8	01FE	0210	021E	0228	0236
STATEMENT NO	922	923	924	925	926	927	928	929	930	931	932	933	934	935	936	937	938	939	940	941	942
OFFSET (HEX)	0248	0256	025A	029A	029E	02E4	02E4	02F0	02F4	0300	0312	0320	032E	033A	033A	03A6	03AA	03B8	03BC	03E8	
STATEMENT NO	943	944	945	946	947	948	949	950	951	952	953	954	955	956	957	958	959	960	961	1024	

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE PROCESS_IF

OFFSET (HEX)	0000	00B4	00BA	00C0	00CA	0110	012C	018E	018E	0194	019E	01EE	01F2	0200	0204	024A	0250	025A	0272	02F6	02F6
STATEMENT NO	1025	1032	1033	1034	1035	1036	1037	1038	1039	1040	1041	1042	1043	1044	1045	1046	1047	1048	1049	1050	1051
OFFSET (HEX)	0300	0310	0314	0398	0398	03A2	03B2	03B6	03F6	03F6	0400	0410	0414	0476	047A	047A	04A6	04AC	04AC	04B2	04B8
STATEMENT NO	1052	1053	1054	1055	1056	1057	1058	1059	1060	1061	1062	1063	1064	1065	1066	1067	1068	1069	1070	1071	1072
OFFSET (HEX)	0502	051E	0536	0544	0544	054A	054E	055C	0560	05A6	05AC	05C2	05C2	05EE	05F4	05F4	0604	060E	061E	064A	064A
STATEMENT NO	1073	1074	1075	1076	1077	1078	1079	1080	1081	1082	1083	1084	1085	1086	1087	1088	1089	1090	1091	1092	1093
OFFSET (HEX)	0658	0662	0678	0684	068A	06E6	0702	0712	0712	072E	073E	073E	0754	0760	0766	07C2	07DE	07EE	07EE	080A	081A
STATEMENT NO	1094	1095	1096	1097	1098	1099	1100	1101	1102	1103	1104	1105	1106	1107	1108	1109	1110	1111	1112	1113	1114
OFFSET (HEX)	081A	082C	082C	083A	083A	085A	085E	085E	086C	086C	088C	0890	0890	089E	089E	08BE	08C2	08C2	08D0	08D0	08F0
STATEMENT NO	1115	1116	1117	1118	1119	1120	1121	1122	1123	1124	1125	1126	1127	1128	1129	1130	1131	1132	1133	1134	1135
OFFSET (HEX)	08F4	08F4	0902	0902	0922	0926	0926	0934	0934	0954	0958	0958	0962	0998	0998	0998	0998	0998			
STATEMENT NO	1136	1137	1138	1139	1140	1141	1142	1143	1144	1145	1146	1147	1148	1149	1150	1151	1152	1153			

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE PROCESS_FOR

OFFSET (HEX)	0000	00E8	0100	0106	014C	0168	0176	0176	017C	0180	018E	0192	01D8	01DE	01E8	01EE	0234	0250	0268	0276	0276
STATEMENT NO	1154	1163	1164	1165	1166	1167	1168	1169	1170	1171	1172	1173	1174	1175	1176	1177	1178	1179	1180	1181	1182
OFFSET (HEX)	027C	0280	028E	0292	02D8	02DE	02F4	02F4	0320	0326	0326	0336	033C	0382	039E	03B6	03C4	03C4	03CA	03CE	03DC
STATEMENT NO	1183	1184	1185	1186	1187	1188	1189	1190	1191	1192	1193	1194	1195	1196	1197	1198	1199	1200	1201	1202	1203
OFFSET (HEX)	03E0	0426	042C	0442	0442	0448	0458	049A	04B6	04C4	04C8	050E	0514	0518	052E	053E	053E	0566	056C	056C	057A
STATEMENT NO	1204	1205	1206	1207	1208	1209	1210	1211	1212	1213	1214	1215	1216	1217	1218	1219	1220	1221	1222	1223	1224
OFFSET (HEX)	059E	05BA	05DA	05DA	0612	0618	0618	062E	063A	0640	069C	06B8	06D8	06EC	070C	0722	072E	0734	0790	07AC	07CC
STATEMENT NO	1225	1226	1227	1228	1229	1230	1231	1232	1233	1234	1235	1236	1237	1238	1239	1240	1241	1242	1243	1244	1245
OFFSET (HEX)	07E0	0800	0816	0822	0828	0884	08A0	08C0	08D4	08F4											
STATEMENT NO	1246	1247	1248	1249	1250	1251	1252	1253	1254	1255											

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE PROCESS_NEXT

```
OFFSET (HEX)  0000 00BC 00C2 00C8 010E 012A 0138 0142 0188 018E 0198 019E 01A8 01B8 01BC 01BC 01E4 01EA 01EA 01F8 021C
STATEMENT NO  1256 1264 1265 1266 1267 1268 1269 1270 1271 1272 1273 1274 1275 1276 1277 1278 1279 1280 1281 1282 1283

OFFSET (HEX)  0238 0258 0280
STATEMENT NO  1284 1285 1286
```

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE PROCESS_LET

```
OFFSET (HEX)  0000 00BC 00D8 0118 0144 0150 0156 015C 0162 01A8 01C4 01D2 01D2 01D8 0224 0228 0236 023A 0280 0286 029C
STATEMENT NO  1287 1292 1293 1294 1295 1296 1297 1298 1299 1300 1301 1302 1303 1304 1305 1306 1307 1308 1309 1310 1311

OFFSET (HEX)  02A8 02AE 02C4 02D0 02D6 02E8 02EC 0348 0364 0374 0374 0390 03C2 03D2 03D2 042E 044A
STATEMENT NO  1312 1313 1314 1315 1316 1317 1318 1319 1320 1321 1322 1323 1324 1325 1326 1327 1328
```

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE PROCESS_DEF

```
OFFSET (HEX)  0000 0170 017E 019A 01A4 01A8 01A8 01D4 01DA 01DA 01E0 01E6 022C 0248 0256 0256 025C 02A8 02AC 02BA 02BE
STATEMENT NO  1329 1338 1339 1340 1341 1342 1343 1344 1345 1346 1347 1348 1349 1350 1351 1352 1353 1354 1355 1356 1357

OFFSET (HEX)  0304 030A 0320 032C 0332 035C 0368 0368 03AC 03BE 0430 049C 049C 04C4 04CA 04CA 04CE 04CE 04FA 0500 0500
STATEMENT NO  1358 1359 1360 1361 1362 1363 1364 1365 1366 1367 1368 1369 1370 1371 1372 1373 1374 1375 1376 1377 1378

OFFSET (HEX)  0520 0520 0552 05BE 05BE 05E6 05EC 05EC 064C 0650 0650 067C 0682 0682 06AA 06BA 06D6 06D6 06E6 06F0 06FA
STATEMENT NO  1379 1380 1381 1382 1383 1384 1385 1386 1387 1388 1389 1390 1391 1392 1393 1394 1395 1396 1397 1398 1399

OFFSET (HEX)  0708 072C 074C 075C 0760 0760 078C 0792 0792 07B4 07B4 07DC 07E2 07E2 07F2 0810 0828 083C 0840 0870 089C
STATEMENT NO  1400 1401 1402 1403 1404 1405 1406 1407 1408 1409 1410 1411 1412 1413 1414 1415 1416 1417 1418 1419 1420

OFFSET (HEX)  08B2 08BE 08C4 08D6 08DA 0936 0952 0972 0982 0982 099E 09D0 09E0 09E0 0A00 0A1C
STATEMENT NO  1421 1422 1423 1424 1425 1426 1427 1428 1429 1430 1431 1432 1433 1434 1435 1436
```

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE BALANCE_STMT

```
OFFSET (HEX)  0000 0048 004E 0054 0068 0090 00A0 00C8 00C8 00D4 00E0 0108 010E 010E 0136 0142 0156 0162 0166 0192 01B2
STATEMENT NO  1437 1441 1442 1443 1444 1445 1446 1447 1448 1449 1450 1451 1452 1453 1454 1455 1456 1457 1458 1459 1460

OFFSET (HEX)  01DE
STATEMENT NO  1461
```

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE POPULATE_STACK

OFFSET (HEX) 0000 0048 004E 0054 005E 0072 0096 009E 009E 00AC 00AC 00B2 00DC 00FE 0110 0134 013C 0160 0198 019C 01E6
STATEMENT NO 1548 1553 1554 1555 1556 1557 1558 1559 1560 1561 1562 1563 1564 1565 1566 1567 1568 1569 1570 1571 1572

OFFSET (HEX) 01EA 01FC 01FC 0202 0206 0304 0304 0314 033E 0350 0356 035A 03A8 03C4 03D0 03D0 03D6 03FC 0408 040C 0420
STATEMENT NO 1573 1574 1575 1576 1577 1578 1579 1580 1581 1582 1583 1584 1585 1586 1587 1588 1589 1590 1591 1592 1593

OFFSET (HEX) 043E 0454 0458 0468 04C6 04CA 04FA 0516 0522 0522 0544 0558 05D2 05EE 05EE
STATEMENT NO 1594 1595 1596 1597 1598 1599 1600 1601 1602 1603 1604 1605 1606 1607 1608

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE PROCESS_OPERATORS

OFFSET (HEX) 0000 004C 0058 0058 00BE 00D2 0150 016C 016C 017C 01E8 01E8 020C 022E 0274 0294 02E4 0302 0324 0342 0366
STATEMENT NO 1731 1734 1735 1736 1737 1738 1739 1740 1741 1742 1743 1744 1745 1746 1747 1748 1749 1750 1751 1752 1753

OFFSET (HEX) 03A8 03C8 03F4 0410 0466 047E 048E 049A 04A4 04B0 04B0 04EE 0502 0580 059C 059C 059C 05AC 05B0 05BC 05BC
STATEMENT NO 1754 1755 1756 1757 1758 1759 1760 1761 1762 1763 1764 1765 1766 1767 1768 1769 1770 1771 1772 1773 1774

OFFSET (HEX) 05FE 0612 0690 06AC 06AC
STATEMENT NO 1775 1776 1777 1778 1779

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE PROCESS_FUNCTION

OFFSET (HEX) 0000 0064 00B8 00C4 00C4 00E6 00FA 0174 0188 0188 01E4 01E4 0208 022A 023A 025E 027E 027E 029A 02BA 02DA
STATEMENT NO 1780 1783 1784 1785 1786 1787 1788 1789 1790 1791 1792 1793 1794 1795 1796 1797 1798 1799 1800 1801 1802

OFFSET (HEX) 02DE 02FE 02FE 031A 033A 035A 035E 0386 0396 0396 03BE 03CE 03EA 043C 0450 0460 0464 0464 0490 04B4 04D0
STATEMENT NO 1803 1804 1805 1806 1807 1808 1809 1810 1811 1812 1813 1814 1815 1816 1817 1818 1819 1820 1821 1822 1823

OFFSET (HEX) 0522 0536 0546 0546 054A 0576 0582 0582 05A4 05B8 0632 0646 0646
STATEMENT NO 1824 1825 1826 1827 1828 1829 1830 1831 1832 1833 1834 1835 1836

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE PROCESS_SUBSCRIPT

OFFSET (HEX) 0000 0074 00C8 00D4 00D4 00F6 010A 0184 0198 0198 01F4 01F4 0258 0258 0278 02A6 02D6 0300 0306 032A 0336
STATEMENT NO 1837 1841 1842 1843 1844 1845 1846 1847 1848 1849 1850 1851 1852 1853 1854 1855 1856 1857 1858 1859 1860

OFFSET (HEX) 0336 0356 0364 0384 03A4 03A8 03A8 03C8 03D6 03F6 0414 0414 0424 0424 043E 044E 046A 04BC 04D0 04E0 04E4
STATEMENT NO 1861 1862 1863 1864 1865 1866 1867 1868 1869 1870 1871 1872 1873 1874 1875 1876 1877 1878 1879 1880 1881

OFFSET (HEX) 04E4 0502 0526 0542 0594 05A8 05B8 05B8 05BC 05E4 05E8 0610 061C 061C 063E 064E 0676 068A 0704 0718 0718
STATEMENT NO 1882 1883 1884 1885 1886 1887 1888 1889 1890 1891 1892 1893 1894 1895 1896 1897 1898 1899 1900 1901 1902

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE SIMPLIFY_SUB_STACK

OFFSET (HEX) 0000 0058 0064 0064 00B8 00D8 0152 0166 0166 0172 0176 0186 0186 01F0 01F0 0212 022E 022E 0256 025C 0260
STATEMENT NO 1609 1613 1614 1615 1616 1617 1618 1619 1620 1621 1622 1623 1624 1625 1626 1627 1628 1629 1630 1631 1632

OFFSET (HEX) 0264 0264 0270 0270 0292 0292 02DC 0338 033C 033C 0364 036A 036A 036A 036A 036A 038C 038C 0398 03A8 03AC
STATEMENT NO 1633 1634 1635 1636 1637 1638 1639 1640 1641 1642 1643 1644 1645 1646 1647 1648 1649 1650 1651 1652 1653

OFFSET (HEX) 03AC 03B6 03C0 03C0 03C6 03C6 03D2 03D2 03DE 041C 0420 0420 044C 0452 047E 0484 04B0 04B0 04D2 04D2 04DE
STATEMENT NO 1654 1655 1656 1657 1658 1659 1660 1661 1662 1663 1664 1665 1666 1667 1668 1669 1670 1671 1672 1673 1674

OFFSET (HEX) 04E2 04F2 04F2 050E 051A 0520 055E 0566 058A 05AA 05AE 05AE 05BA 05BA 0600 0614 068E 06A2 06A2 06D4 06E4
STATEMENT NO 1675 1676 1677 1678 1679 1680 1681 1682 1683 1684 1685 1686 1687 1688 1689 1690 1691 1692 1693 1694 1695

OFFSET (HEX) 070C 0724 0748 079A 07AE 07C6 07D6 07E6 07F2 07FC 0808 0808 084A 085E 08D8 08EC 08EC 08EC 08F0 08F0 093A
STATEMENT NO 1696 1697 1698 1699 1700 1701 1702 1703 1704 1705 1706 1707 1708 1709 1710 1711 1712 1713 1714 1715 1716

OFFSET (HEX) 0996 09B0 0A0C 0A26 0A42 0A46 0A6E 0A6E 0A7A 0A7A 0A9C 0AB0 0B2A 0B3E 0B3E
STATEMENT NO 1717 1718 1719 1720 1721 1722 1723 1724 1725 1726 1727 1728 1729 1730 1903

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE PARSE_EXP

OFFSET (HEX) 0000 011C 0128 0132 0146 0146 0170 019A 020C 0218 023C 024C 0266 0278 0284 0312 0326 0326 0370 0376 037C
STATEMENT NO 1462 1473 1474 1475 1476 1477 1478 1479 1480 1481 1482 1483 1484 1485 1486 1487 1488 1489 1490 1491 1492

OFFSET (HEX) 0382 03AA 03B0 03B6 03BC 03E4 03F0 040A 0414 042E 0434 0438 0440 0440 0446 044C 0450 0456 0474 0480 048A
STATEMENT NO 1493 1494 1495 1496 1497 1498 1499 1500 1501 1502 1503 1504 1505 1506 1507 1508 1509 1510 1511 1512 1513

OFFSET (HEX) 0490 049C 04A0 04AC 04AC 04CE 04CE 04DA 04DA 0518 052C 0552 0552 055A 055A 055E 055E 057A 057A 0596 05C8
STATEMENT NO 1514 1515 1516 1517 1518 1519 1520 1521 1522 1523 1524 1525 1526 1527 1528 1529 1530 1531 1532 1533 1534

OFFSET (HEX) 05E0 05E4 0604 0608 06C4 06C8 06F0 06F0 06F6 06F6 070A 074C 0750 0778
STATEMENT NO 1535 1536 1537 1538 1539 1540 1541 1542 1543 1544 1545 1546 1547 1904

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE PROCESS_DIM

OFFSET (HEX) 0000 0114 011A 0120 0166 0182 01C2 01CC 0212 0218 0226 024A 02A6 02AA 02AA 02D2 02D8 02D8 02E2 02EC 02F2
STATEMENT NO 1905 1913 1914 1915 1916 1917 1918 1919 1920 1921 1922 1923 1924 1925 1926 1927 1928 1929 1930 1931 1932

OFFSET (HEX) 02F8 033E 035A 0368 0372 0380 0384 03C4 040E 040E 0436 043C 043C 0442 044A 044A 0472 0478 0478 0482 049C
STATEMENT NO 1933 1934 1935 1936 1937 1938 1939 1940 1941 1942 1943 1944 1945 1946 1947 1948 1949 1950 1951 1952 1953

OFFSET (HEX) 051C 051C 0542 0542 0566 056C 056C 0584 0594 05A4 05CE 05EA 05FE 0624 0638 063C 063C 0664 066A 066A 0674
STATEMENT NO 1954 1955 1956 1957 1958 1959 1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973 1974

OFFSET (HEX) 0684 0688 0688 06A4 06B2 06B2 06BE 06C2 06C6 06C6 06EE 06F4 06F4 06F4
STATEMENT NO 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE COMPILE

OFFSET (HEX)	0000	0094	00A2	00B8	00C2	00CC	00DC	00E8	00FE	0170	0186	0194	0194	019E	01A4	01B2	01C2	01E2	01FC	0210	021A
STATEMENT NO	276	281	291	292	293	294	295	296	297	298	299	300	301	302	303	304	305	306	307	308	309
OFFSET (HEX)	0224	0230	023A	023A	023E	024C	0258	025C	0260	026A	0288	029C	02FC	0310	0336	0394	039E	03C4	03FA	040C	040C
STATEMENT NO	310	311	312	313	314	323	324	325	327	328	329	330	331	332	333	334	335	336	337	338	339
OFFSET (HEX)	043E	044A	0480	0484	0484	04D8	04E4	051A	051A												
STATEMENT NO	340	341	342	343	344	345	346	347	1989												

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN ON UNIT

OFFSET (HEX)	0000	0060	0082	00A6	0118	013C	0160	0184	01A8	01CC	01D6	01E0	01E6
STATEMENT NO	2025	2026	2027	2028	2029	2030	2031	2032	2033	2034	2035	2036	2037

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE COMPARE_DIF_LEN

OFFSET (HEX)	0000	0058	007A	0090	009A	00A4	00AE	00BC	00C6	00D2	00F6
STATEMENT NO	2464	2466	2467	2468	2469	2470	2471	2472	2473	2474	2475

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE COMPARE_RTN

OFFSET (HEX)	0000	0044	004E	0096	0096	00A6	00B0	00C0	00CA	00D4	00D8	00D8	00E4	015C	01A8	01A8	01D2	01D2	01FE	0208	0240
STATEMENT NO	2434	2435	2436	2437	2438	2439	2440	2441	2442	2443	2444	2445	2446	2447	2448	2449	2450	2451	2452	2453	2454
OFFSET (HEX)	024A	0254	0258	0262	0266	0266	0282	0288	0288	0288											
STATEMENT NO	2455	2456	2457	2458	2459	2460	2461	2462	2463	2476											

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE FORMAT_NUMBER

OFFSET (HEX)	0000	0050	0098	0098	00A8	00D4	00D4	00F6	010C	010C	0110	0110	0132	0138	0152	015E	0162	0178	017C	018C	0196
STATEMENT NO	2478	2480	2481	2482	2483	2484	2485	2486	2487	2488	2489	2490	2491	2492	2493	2494	2495	2496	2497	2498	2499
OFFSET (HEX)	01B4	01C0	01C4	01DE	01EA	024E	024E	0280													
STATEMENT NO	2500	2501	2502	2503	2504	2505	2506	2507													

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE COMPRESS_FS

OFFSET (HEX)	0000	0038	0048	0048	0054	005A	005E	006E	006E	00BE	00CA	00D0	00D0	00E0	00E0	00E6	00EC	00F0	00F0	0100	0106
STATEMENT NO	2508	2510	2511	2512	2513	2514	2515	2516	2517	2518	2519	2520	2521	2522	2523	2524	2525	2526	2527	2528	2529

OFFSET (HEX) 0106 011E 01B8 01CC 01DC
STATEMENT NO 2530 2531 2532 2533 2534

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE PRINT_BUFFER

OFFSET (HEX) 0000 0048 0058 0058 0064 00D4 00E4 00E4 00EE 0132 0136 0136 0150 015A 01F4 01FE 0204 0208 0208 0222 022C
STATEMENT NO 2535 2540 2541 2542 2543 2544 2545 2546 2547 2548 2549 2550 2551 2552 2553 2554 2555 2556 2557 2558 2559

OFFSET (HEX) 0274 0274 029A 02B2 02C4 02D0 02D0 02E0 02EE 0358 0358 0358 03F2 03F2
STATEMENT NO 2560 2561 2562 2563 2564 2565 2566 2567 2568 2569 2570 2571 2572 2573

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE PRINT_ERR

OFFSET (HEX) 0000 0044 004E 009E 00C2 0112 0170 017A
STATEMENT NO 2574 2576 2577 2578 2579 2580 2581 2582

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE FLUSH_BUFFER

OFFSET (HEX) 0000 003C 0076 0080
STATEMENT NO 2583 2584 2585 2586

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE RND

OFFSET (HEX) 0000 004C 0058 0058 009E 00A4 00A4 00C8 00D4 00DE 00F2 0102 0102 0116 011A 013A 0148 0158
STATEMENT NO 2588 2593 2594 2595 2596 2597 2598 2599 2600 2601 2602 2603 2604 2605 2606 2607 2608 2609

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE EXECUTE

OFFSET (HEX) 0000 0790 07A4 07B0 07B6 07C2 07C8 07D4 07EC 07FC 0808 080E 0814 081A 0820 0826 0834 0840 084C 0854 085A
STATEMENT NO 1990 1991 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021 2022 2023 2024 2038 2039 2040 2041 2042

OFFSET (HEX) 0866 0866 08C0 08C6 08C6 08D6 08D6 08F2 08F8 08F8 0904 0928 0928 0944 094A 094E 095A 095A 0976 097C 0980
STATEMENT NO 2043 2044 2045 2046 2047 2048 2049 2050 2051 2052 2053 2054 2055 2056 2057 2058 2059 2060 2061 2062 2063

OFFSET (HEX) 0998 09A4 09EE 09F4 09F4 0A36 0A36 0A3C 0A40 0A40 0A82 0A82 0AAA 0AB4 0AB4 0AD0 0AD6 0AD6 0ADA 0ADA 0B18
STATEMENT NO 2064 2065 2066 2067 2068 2069 2070 2071 2072 2073 2074 2075 2076 2077 2078 2079 2080 2081 2082 2083 2084

OFFSET (HEX) 0B18 0B40 0B4A 0B4A 0B66 0B6C 0B6C 0B70 0B70 0BB2 0BB2 0BDA 0BE4 0BEA 0BEE 0BEE 0C0A 0C18 0C18 0C18 0C18
STATEMENT NO 2085 2086 2087 2088 2089 2090 2091 2092 2093 2094 2095 2096 2097 2098 2099 2100 2101 2102 2103 2104 2105

OFFSET (HEX) 0C36 0C64 0C6C 0C90 0C94 0CA8 0CBC 0CC2 0CCE 0D56 0D5A 0D6E 0D7A 0DDC 0DE0 0DF0 0E48 0EBC 0EBC 0ED8 0EDE
STATEMENT NO 2106 2107 2108 2109 2110 2111 2112 2113 2114 2115 2116 2117 2118 2119 2120 2121 2122 2123 2124 2125 2126

OFFSET (HEX)	0EE2	0EFE	0F16	0F16	0F32	0F38	0F38	0F44	1060	1064	1078	1084	10E6	10EA	110E	111A	117C	1180	1198	11A4	11C8
STATEMENT NO	2127	2128	2129	2130	2131	2132	2133	2134	2135	2136	2137	2138	2139	2140	2141	2142	2143	2144	2145	2146	2147
OFFSET (HEX)	11CC	11E4	11F0	1214	1218	1230	123C	1260	1264	127C	127C	1298	129E	129E	12B6	12C2	12E6	12EA	12FA	1306	1306
STATEMENT NO	2148	2149	2150	2151	2152	2153	2154	2155	2156	2157	2158	2159	2160	2161	2162	2163	2164	2165	2166	2167	2168
OFFSET (HEX)	133E	1344	1344	1350	1374	13EC	13F0	13FC	140E	145C	1472	1476	14A8	14AC	14AC	14C8	14C8	14D2	14D8	14DE	14E2
STATEMENT NO	2169	2170	2171	2172	2173	2174	2175	2176	2177	2178	2179	2180	2181	2182	2183	2184	2185	2186	2187	2188	2189
OFFSET (HEX)	14E2	14FE	14FE	1504	1508	1508	1524	1524	152A	152E	152E	152E	154A	1550	1550	1550	1550	1550	1554	15C4	15C4
STATEMENT NO	2190	2191	2192	2193	2194	2195	2196	2197	2198	2199	2200	2201	2202	2203	2204	2205	2206	2207	2208	2209	2210
OFFSET (HEX)	1632	1640	1640	164C	16DC	16FA	1706	1706	1722	1728	1728	1748	1768	176E	1778	1798	179E	17BC	17DC	181E	181E
STATEMENT NO	2211	2212	2213	2214	2215	2216	2217	2218	2219	2220	2221	2222	2223	2224	2225	2226	2227	2228	2229	2230	2231
OFFSET (HEX)	183A	1840	1840	1846	1856	1876	187C	189C	18BC	18C2	18E2	1902	1908	1928	1948	194E	1972	1992	1998	19A6	19C8
STATEMENT NO	2232	2233	2234	2235	2236	2237	2238	2239	2240	2241	2242	2243	2244	2245	2246	2247	2248	2249	2250	2251	2252
OFFSET (HEX)	19E4	1A04	1A12	1AA2	1AA6	1AB0	1B16	1B1C	1B32	1B6E	1B6E	1B7E	1B82	1B82	1BA2	1C0A	1C10	1C1E	1C5A	1C76	1C76
STATEMENT NO	2253	2254	2255	2256	2257	2258	2259	2260	2261	2262	2263	2264	2265	2266	2267	2268	2269	2270	2271	2272	2273
OFFSET (HEX)	1C92	1C98	1C98	1CA6	1CA6	1D06	1D3E	1D3E	1D5A	1D60	1D60	1D66	1D66	1D7C	1DB8	1DB8	1DC4	1DD4	1DE4	1DF4	1E18
STATEMENT NO	2274	2275	2276	2277	2278	2279	2280	2281	2282	2283	2284	2285	2286	2287	2288	2289	2290	2291	2292	2293	2294
OFFSET (HEX)	1E1C	1E1C	1E3C	1EA4	1EAA	1EB8	1EB8	1ED4	1EDA	1EDA	1EEA	1EF6	1EFA	1F0E	1F22	1F28	1F36	1F5A	1F5E	1F72	1F86
STATEMENT NO	2295	2296	2297	2298	2299	2300	2301	2302	2303	2304	2305	2306	2307	2308	2309	2310	2311	2312	2313	2314	2315
OFFSET (HEX)	1F8C	1F9A	1FBE	1FC2	1FCC	1FD4	1FDA	1FDE	1FE8	1FF0	1FF4	1FFA	2004	200C	2012	2016	2020	2028	202E	2032	203C
STATEMENT NO	2316	2317	2318	2319	2320	2321	2322	2323	2324	2325	2326	2327	2328	2329	2330	2331	2332	2333	2334	2335	2336
OFFSET (HEX)	2078	207E	2082	208C	20C8	20CE	20D2	20D6	20E4	20F6	20F6	20FC	210A	2126	212C	2138	2140	215C	215C	2178	217E
STATEMENT NO	2337	2338	2339	2340	2341	2342	2343	2344	2345	2346	2347	2348	2349	2350	2351	2352	2353	2354	2355	2356	2357
OFFSET (HEX)	217E	218A	2190	2190	21AC	21DC	21EC	21FA	22D6	22DA	22F6	22FA	2316	231A	2336	2356	2366	2374	2450	2454	2462
STATEMENT NO	2358	2359	2360	2361	2362	2363	2364	2365	2366	2367	2368	2369	2370	2371	2372	2373	2374	2375	2376	2377	2378
OFFSET (HEX)	2462	247E	2484	2484	248A	2498	24B4	24BA	24C6	24CE	24EA	24F0	2506	2506	2522	2528	2528	254C	2570	2586	2586
STATEMENT NO	2379	2380	2381	2382	2383	2384	2385	2386	2387	2388	2389	2390	2391	2392	2393	2394	2395	2396	2397	2398	2399
OFFSET (HEX)	25A4	25B4	25C4	25CA	25CA	25E8	25F8	2608	2608	260C	2610	261A	261E	263C	264A	266E	2672	2686	268A	26A0	26DC
STATEMENT NO	2400	2401	2402	2403	2404	2405	2406	2407	2408	2409	2410	2411	2412	2413	2414	2415	2416	2417	2418	2419	2420
OFFSET (HEX)	26DC	26F0	2704	2720	2724	2724	273C	2758	275E	2794	279E	27A2	27BE	27C4							
STATEMENT NO	2421	2422	2423	2424	2425	2426	2427	2428	2429	2430	2431	2432	2433	2610							

TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE TERMINATE

```
OFFSET (HEX)  0000 0038 0044 004E
STATEMENT NO  2611 2613 2614 2615


TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE          PRINT_SYMBOLS

OFFSET (HEX)  0000 0088 00A6 00BA 016A 0236 026C 029E 02BA 02DC
STATEMENT NO  2616 2617 2618 2619 2620 2621 2622 2623 2624 2625


TABLE OF OFFSETS AND STATEMENT NUMBERS WITHIN PROCEDURE          BASIC

OFFSET (HEX)  0000 02D8 02FA 032E 033C 0348 0350 0382 0398 03A2 03AC 03BC 03BC 03C6 03F0 03F0 03FA 0404 040E 040E 041C
STATEMENT NO     1   37   38   39   41   42   43   44   45   46   47   48   49   50   51   52   53   54   55   56   57

OFFSET (HEX)  041C 0426 0430 0430 0430 0434
STATEMENT NO    58   59   60   61   62 2626
```

COMPILER DIAGNOSTICS.

WARNINGS.

IEM0227I NO FILE/STRING OPTION SPECIFIED IN ONE OR MORE GET/PUT STATEMENTS. SYSIN/SYSPRINT HAS BEEN ASSUMED IN EACH CASE.

IEM0764I ONE OR MORE FIXED BINARY ITEMS OF PRECISION 15 OR LESS HAVE BEEN GIVEN HALFWORD STORAGE. THEY ARE FLAGGED '**********' IN THE XREF/ATR LIST.

IEM1790I DATA CONVERSIONS WILL BE DONE BY SUBROUTINE CALL IN THE FOLLOWING STATEMENTS 184, 230, 494, 541, 558, 834, 836, 1952, 2485, 2490, 2595.

END OF DIAGNOSTICS.

AUXILIARY STORAGE WILL NOT BE USED FOR DICTIONARY WHEN SIZE = 216K

COMPILE TIME .02 MINS

ELAPSED TIME .02 MINS

```
 F64-LEVEL LINKAGE EDITOR OPTIONS SPECIFIED XREF,LIST
          DEFAULT OPTION(S) USED -  SIZE=(65536,38912)
****BASIC360  DOES NOT EXIST BUT HAS BEEN ADDED TO DATA SET
AUTHORIZATION CODE IS         0.
```

CROSS REFERENCE TABLE

CONTROL SECTION NAME		ORIGIN	LENGTH	ENTRY NAME	LOCATION	NAME	LOCATION	NAME	LOCATION	NAME	LOCATION
BASIC		00	E8D2								
**BASICA		E8D8	6300								
IHEMAIN		14BD8	4								
IHENTRY		14BE0	C								
RENUMFL		14BF0	38								
SYSIN		14C28	38								
IHESPRT		14C60	38								
IHECSM	*	14C98	114								
				IHECSMH	14C98	IHECSML	14CA4	IHECSMF	14CC2	IHECSMV	14CD6
				IHECSMB	14D36						
IHEDCN	*	14DB0	256								
				IHEDCNA	14DB0	IHEDCNB	14DB2				
IHEDMA	*	15008	F8								
				IHEDMAA	15008						
IHEERR	*	15100	745								
				IHEERRD	15100	IHEERRC	1510A	IHEERRB	15114	IHEERRA	1511E
				IHEERRE	157AE						
IHEUPA	*	15848	E8								
				IHEUPAA	15848	IHEUPAB	158B2				
IHEUPB	*	15930	E4								
				IHEUPBA	15930	IHEUPBB	15998				
IHEVQB	*	15A18	4DC								
				IHEVQBA	15A18						
IHEVFA	*	15EF8	16C								
				IHEVFAA	15EF8						
IHEVFB	*	16068	F0								
				IHEVFBA	16068						
IHEVFC	*	16158	26								
				IHEVFCA	16158						
IHEVFD	*	16180	66								
				IHEVFDA	16180						
IHEVFE	*	161E8	20								
				IHEVFEA	161E8						
IHEVKB	*	16208	30A								
				IHEVKBA	16208						
IHEVKC	*	16518	2D0								
				IHEVKCA	16518						
IHEVKF	*	167E8	67C								
				IHEVKFA	167E8						
IHEVKG	*	16E68	4DE								
				IHEVKGA	16E68						
IHEVPA	*	17348	1E0								

NAME		ORIGIN	LENGTH
IHEVPB	*	17528	1A2
IHEVPC	*	176D0	1F0
IHEVPD	*	178C0	105
IHEVPE	*	179C8	278
IHEVPF	*	17C40	50
IHEVPG	*	17C90	229
IHEVPH	*	17EC0	B4
IHEABN	*	17F78	C
IHEM91	*	17F88	178
IHETER	*	18100	110
IHEVQA	*	18210	FC
IHEDDO	*	18310	288
IHEDDP	*	18598	28C
IHEIOF	*	18828	2DC
IHELDO	*	18B08	418
IHEPRT	*	18F20	308
IHEDNC	*	19228	2B2
IHEVSB	*	194E0	CE
IHEVSC	*	195B0	AC
IHEOCL	*	19660	580
IHEVQC	*	19BE0	268
IHEVSE	*	19E48	15D
IHEVSF	*	19FA8	EC
IHESAP	*	1A098	B1C

NAME	LOCATION	NAME	LOCATION	NAME	LOCATION	NAME	LOCATION
IHEVPAA	17348						
IHEVPBA	17528						
IHEVPCA	176D0						
IHEVPDA	178C0						
IHEVPEA	179C8						
IHEVPFA	17C40						
IHEVPGA	17C90						
IHEVPHA	17EC0						
IHEABND	17F78						
IHEM91B	17F88	IHEM91A	17F90	IHEM91C	180A2		
IHETERA	18100						
IHEVQAA	18210						
IHEDDOA	18310	IHEDDOB	18312	IHEDDOC	18314	IHEDDOD	18316
IHEDDOE	18318						
IHEDDPA	18598	IHEDDPB	1859A	IHEDDPC	1859C	IHEDDPD	1859E
IHEIOFB	18828	IHEIOFA	1882A	IHEITAZ	18AC6	IHEITAX	18AD2
IHEITAA	18AE6						
IHELDOA	18B08	IHELDOB	18B0A	IHELDOC	18B0E		
IHEPRTA	18F20	IHEPRTB	18F22				
IHEDNCA	19228						
IHEVSBA	194E0						
IHEVSCA	195B0						
IHEOCLA	19660	IHEOCLB	19662	IHEOCLC	19664	IHEOCLD	19666
IHEVQCA	19BE0						
IHEVSEA	19E48	IHEVSEB	19E4A				
IHEVSFA	19FA8						
IHESADA	1A098	IHESAPC	1A0B2	IHESAPD	1A0BA	IHESAPA	1A0C2
IHESAPB	1A0CA	IHESADF	1A0D2	IHESADB	1A0DA	IHESADE	1A0E2

NAME		ORIGIN	LENGTH
IHEBEG	*	1ABB8	80
IHESIZ	*	1AC38	C
IHETAB	*	1AC48	C
IHEDIB	*	1AC58	114
IHEIOD	*	1AD70	29A
IHEKCD	*	1B010	110
IHEVSD	*	1B120	1A0
IHEDOA	*	1B2C0	23A
IHEDBN	*	1B500	167
IHEDOB	*	1B668	144
IHEDOE	*	1B7B0	DA
IHEIOB	*	1B890	1E0
IHEIOP	*	1BA70	1F7
IHEIOX	*	1BC68	14C
IHECSC	*	1BDB8	C6
IHECSK	*	1BE80	13A
IHEIOA	*	1BFC0	16A
IHEOSD	*	1C130	D8
IHEOSS	*	1C208	34
IHEOST	*	1C240	52
IHESRC	*	1C298	1A2
IHEXXS	*	1C440	90
IHEEXS	*	1C4D0	F8

NAME	LOCATION	NAME	LOCATION	NAME	LOCATION	NAME	LOCATION
IHESAFC	1A0EA	IHESAFA	1A0F2	IHESAFB	1A0FA	IHESAFD	1A102
IHESARA	1A10A	IHESAFQ	1A112	IHESARC	1A904	IHESADD	1AA18
IHESAFF	1AA52						
IHEBEGN	1ABB8	IHEBEGA	1ABF8				
IHESIZE	1AC38						
IHETABS	1AC48						
IHEDIBA	1AC58	IHEDIBB	1AC5A				
IHEIODG	1AD70	IHEIODP	1AD72	IHEIODT	1AE6A		
IHEKCDA	1B010	IHEKCDB	1B012				
IHEVSDA	1B120	IHEVSDB	1B122				
IHEDOAA	1B2C0	IHEDOAB	1B2C2				
IHEDBNA	1B500						
IHEDOBA	1B668	IHEDOBB	1B66A	IHEDOBC	1B66C		
IHEDOEA	1B7B0						
IHEIOBA	1B890	IHEIOBB	1B898	IHEIOBC	1B8A0	IHEIOBD	1B8A8
IHEIOBE	1B8B0	IHEIOBT	1B998				
IHEIOPA	1BA70	IHEIOPB	1BA72	IHEIOPC	1BA76		
IHEIOXA	1BC68	IHEIOXB	1BC6A	IHEIOXC	1BC6E		
IHECSC0	1BDB8						
IHECSKR	1BE80	IHECSKK	1BEAC				
IHEIOAA	1BFC0	IHEIOAB	1BFC2	IHEIOAC	1BFC4	IHEIOAD	1BFC6
IHEIOAT	1C0B0						
IHEOSDA	1C130						
IHEOSSA	1C208						
IHEOSTA	1C240						
IHESRCA	1C298	IHESRCB	1C29A	IHESRCC	1C29C	IHESRCD	1C29E
IHESRCE	1C2A0	IHESRCF	1C2A2				
IHEXXS0	1C440						
IHEEXS0	1C4D0						

NAME		ORIGIN	LENGTH
IHELNS	*	1C5C8	108
IHESNS	*	1C6D0	148
IHESQS	*	1C818	A4
IHETNS	*	1C8C0	100

NAME	LOCATION	NAME	LOCATION	NAME	LOCATION	NAME	LOCATION
IHELNS2	1C5C8	IHELNSD	1C5CA	IHELNSE	1C5CE		
IHESNSK	1C6D0	IHESNSC	1C6DC	IHESNSZ	1C6FE	IHESNSS	1C70A
IHESQS0	1C818						
IHETNSD	1C8C0	IHETNSR	1C8CC				

LOCATION	REFERS TO SYMBOL	IN CONTROL SECTION
2F40	**BASICA	**BASICA
10	**BASICA	**BASICA
7C4	**BASICA	**BASICA
1D50	**BASICA	**BASICA
E8F8	IHESADA	IHESAP
E900	BASIC	BASIC
EB3C	BASIC	BASIC
EB44	BASIC	BASIC
EC85	BASIC	BASIC
EC95	BASIC	BASIC
ECA5	BASIC	BASIC
ECB5	BASIC	BASIC
ECC5	BASIC	BASIC
ECD5	BASIC	BASIC
ECE5	BASIC	BASIC
ECF5	BASIC	BASIC
ED05	BASIC	BASIC
ED15	BASIC	BASIC
ED25	BASIC	BASIC
ED35	BASIC	BASIC
ED45	BASIC	BASIC
ED55	BASIC	BASIC
ED65	BASIC	BASIC
ED6C	BASIC	BASIC
ED74	BASIC	BASIC
ED7C	BASIC	BASIC
ED84	BASIC	BASIC
ED8C	BASIC	BASIC
ED94	BASIC	BASIC
ED9C	IHEDNCA	IHEDNC
EDA4	IHESAFD	IHESAP
EDAC	IHECSMF	IHECSM
EDB4	IHEDDOC	IHEDDO
EDBC	IHELDOA	IHELDO
EDC4	IHEIOXC	IHEIOX
EDCC	IHEIOXB	IHEIOX
EDD4	IHEIOPB	IHEIOP
EDDC	IHEDOEA	IHEDOE
EDE4	IHEDOBB	IHEDOB
EDEC	IHEDIBA	IHEDIB

LOCATION	REFERS TO SYMBOL	IN CONTROL SECTION
40C8	**BASICA	**BASICA
458	**BASICA	**BASICA
1B80	**BASICA	**BASICA
D3F4	**BASICA	**BASICA
E8FC	IHESADB	IHESAP
E904	BASIC	BASIC
EB40	BASIC	BASIC
EB48	BASIC	BASIC
EC8D	BASIC	BASIC
EC9D	BASIC	BASIC
ECAD	BASIC	BASIC
ECBD	BASIC	BASIC
ECCD	BASIC	BASIC
ECDD	BASIC	BASIC
ECED	BASIC	BASIC
ECFD	BASIC	BASIC
ED0D	BASIC	BASIC
ED1D	BASIC	BASIC
ED2D	BASIC	BASIC
ED3D	BASIC	BASIC
ED4D	BASIC	BASIC
ED5D	BASIC	BASIC
ED68	BASIC	BASIC
ED70	BASIC	BASIC
ED78	BASIC	BASIC
ED80	BASIC	BASIC
ED88	BASIC	BASIC
ED90	BASIC	BASIC
ED98	IHEDMAA	IHEDMA
EDA0	IHEDCNA	IHEDCN
EDA8	IHESADB	IHESAP
EDB0	IHEDDOB	IHEDDO
EDB8	IHEDDOA	IHEDDO
EDC0	IHELDOB	IHELDO
EDC8	IHEIOXC	IHEIOX
EDD0	IHEIOXB	IHEIOX
EDD8	IHEIOPB	IHEIOP
EDE0	IHEDOEA	IHEDOE
EDE8	IHEDOBB	IHEDOB
EDF0	IHEDIBA	IHEDIB

LOCATION	REFERS TO SYMBOL	IN CONTROL SECTION	LOCATION	REFERS TO SYMBOL	IN CONTROL SECTION
EDF4	IHEDOAA	IHEDOA	EDF8	IHEDOAA	IHEDOA
EDFC	IHEIOBB	IHEIOB	EE00	IHEIOBA	IHEIOB
EE04	IHEIOBT	IHEIOB	EE08	IHEIOBC	IHEIOB
EE0C	IHEIOAT	IHEIOA	EE10	IHEIOAA	IHEIOA
EE14	IHESAFC	IHESAP	EE18	IHESAFB	IHESAP
EE1C	IHEERRB	IHEERR	EE20	IHEOSSA	IHEOSS
E908	IHESAFA	IHESAP	EE24	IHESRCD	IHESRC
EE28	IHECSC0	IHECSC	EE2C	IHECSKK	IHECSK
EE30	IHEXXS0	IHEXXS	EE34	IHEOSDA	IHEOSD
EE38	IHESRCB	IHESRC	EE3C	IHEOSTA	IHEOST
EE40	IHESQS0	IHESQS	EE44	IHESNSC	IHESNS
EE48	IHESNSS	IHESNS	EE4C	IHETNSR	IHETNS
EE50	RENUMFL	RENUMFL	EE54	SYSIN	SYSIN
EE58	IHESPRT	IHESPRT	10AB4	IHESPRT	IHESPRT
10AB8	BASIC	BASIC	10AC0	IHESPRT	IHESPRT
10AC5	BASIC	BASIC	10AC8	IHESPRT	IHESPRT
10ACD	BASIC	BASIC	10AD0	IHESPRT	IHESPRT
10AD4	BASIC	BASIC	10ADC	IHESPRT	IHESPRT
10AE0	BASIC	BASIC	10AE8	IHESPRT	IHESPRT
10AEC	BASIC	BASIC	10AF4	IHESPRT	IHESPRT
10AF9	BASIC	BASIC	10AFC	IHESPRT	IHESPRT
10B01	BASIC	BASIC	10B04	IHESPRT	IHESPRT
10B08	BASIC	BASIC	10B10	IHESPRT	IHESPRT
10B14	BASIC	BASIC	10B1C	IHESPRT	IHESPRT
10B20	BASIC	BASIC	10B28	IHESPRT	IHESPRT
10B2C	BASIC	BASIC	10B34	IHESPRT	IHESPRT
10B38	BASIC	BASIC	10B40	IHESPRT	IHESPRT
10B44	BASIC	BASIC	10B4C	IHESPRT	IHESPRT
10B50	BASIC	BASIC	10B58	IHESPRT	IHESPRT
10B5C	BASIC	BASIC	10B64	IHESPRT	IHESPRT
10B68	BASIC	BASIC	10B70	IHESPRT	IHESPRT
10B74	BASIC	BASIC	10B7C	IHESPRT	IHESPRT
10B80	BASIC	BASIC	10B88	IHESPRT	IHESPRT
10B8C	BASIC	BASIC	10B94	IHESPRT	IHESPRT
10B98	BASIC	BASIC	10BA0	IHESPRT	IHESPRT
10BA4	BASIC	BASIC	10BAC	IHESPRT	IHESPRT
10BB0	BASIC	BASIC	10BB8	IHESPRT	IHESPRT
10BBC	BASIC	BASIC	10BC4	IHESPRT	IHESPRT
10BC8	BASIC	BASIC	10BD0	IHESPRT	IHESPRT
10BD4	BASIC	BASIC	10BDC	IHESPRT	IHESPRT
10BE0	BASIC	BASIC	10BE8	IHESPRT	IHESPRT
10BEC	BASIC	BASIC	10BF4	IHESPRT	IHESPRT
10BF8	BASIC	BASIC	10C00	IHESPRT	IHESPRT
10C04	BASIC	BASIC	10C0C	IHESPRT	IHESPRT
10C10	BASIC	BASIC	10C18	IHESPRT	IHESPRT
10C1C	BASIC	BASIC	10C24	IHESPRT	IHESPRT
10C28	BASIC	BASIC	10C30	IHESPRT	IHESPRT
10C34	BASIC	BASIC	10C3C	IHESPRT	IHESPRT
10C40	BASIC	BASIC	10C48	IHESPRT	IHESPRT
10C4C	BASIC	BASIC	10C54	IHESPRT	IHESPRT
10C58	BASIC	BASIC	10C60	IHESPRT	IHESPRT
10C64	BASIC	BASIC	10C6C	IHESPRT	IHESPRT

LOCATION	REFERS TO SYMBOL	IN CONTROL SECTION	LOCATION	REFERS TO SYMBOL	IN CONTROL SECTION
10C70	BASIC	BASIC	10C78	IHESPRT	IHESPRT
10C7C	BASIC	BASIC	10C84	IHESPRT	IHESPRT
10C88	BASIC	BASIC	10C90	IHESPRT	IHESPRT
10C94	BASIC	BASIC	10C9C	IHESPRT	IHESPRT
10CA0	BASIC	BASIC	10CA8	IHESPRT	IHESPRT
10CAC	BASIC	BASIC	10CB4	IHESPRT	IHESPRT
10CB8	BASIC	BASIC	10CC0	IHESPRT	IHESPRT
10CC4	BASIC	BASIC	10CCC	IHESPRT	IHESPRT
10CD0	BASIC	BASIC	10CD8	IHESPRT	IHESPRT
10CDC	BASIC	BASIC	10CE4	IHESPRT	IHESPRT
10CE8	BASIC	BASIC	10CF0	IHESPRT	IHESPRT
10CF4	BASIC	BASIC	10CFC	IHESPRT	IHESPRT
10D00	BASIC	BASIC	10D08	IHESPRT	IHESPRT
10D0C	BASIC	BASIC	10D14	IHESPRT	IHESPRT
10D18	BASIC	BASIC	10D20	IHESPRT	IHESPRT
10D24	BASIC	BASIC	10D2C	IHESPRT	IHESPRT
10D30	BASIC	BASIC	10D38	IHESPRT	IHESPRT
10D3C	BASIC	BASIC	10D44	IHESPRT	IHESPRT
10D48	BASIC	BASIC	10D50	IHESPRT	IHESPRT
10D54	BASIC	BASIC	10D5C	IHESPRT	IHESPRT
10D60	BASIC	BASIC	10D68	IHESPRT	IHESPRT
10D6C	BASIC	BASIC	10D74	IHESPRT	IHESPRT
10D78	BASIC	BASIC	10D80	IHESPRT	IHESPRT
10D84	BASIC	BASIC	10D8C	IHESPRT	IHESPRT
10D90	BASIC	BASIC	10D98	IHESPRT	IHESPRT
10D9C	BASIC	BASIC	10DA4	IHESPRT	IHESPRT
10DA8	BASIC	BASIC	10DB0	IHESPRT	IHESPRT
10DB4	BASIC	BASIC	10DBC	IHESPRT	IHESPRT
10DC0	BASIC	BASIC	10DC8	IHESPRT	IHESPRT
10DCC	BASIC	BASIC	10DD4	IHESPRT	IHESPRT
10DD8	BASIC	BASIC	10DE0	IHESPRT	IHESPRT
10DE5	BASIC	BASIC	10DE8	IHESPRT	IHESPRT
10DEC	BASIC	BASIC	10DF4	IHESPRT	IHESPRT
10DF8	BASIC	BASIC	10E00	IHESPRT	IHESPRT
10E04	BASIC	BASIC	10E0C	IHESPRT	IHESPRT
10E10	BASIC	BASIC	10E18	IHESPRT	IHESPRT
10E1D	BASIC	BASIC	10E20	IHESPRT	IHESPRT
10E24	BASIC	BASIC	10E2C	IHESPRT	IHESPRT
10E30	BASIC	BASIC	10E38	IHESPRT	IHESPRT
10E3D	BASIC	BASIC	10E40	IHESPRT	IHESPRT
10E44	BASIC	BASIC	10E4C	IHESPRT	IHESPRT
10E50	BASIC	BASIC	10E58	IHESPRT	IHESPRT
10E5C	BASIC	BASIC	10E64	IHESPRT	IHESPRT
10E68	BASIC	BASIC	10E70	IHESPRT	IHESPRT
10E74	BASIC	BASIC	10E7C	IHESPRT	IHESPRT
10E80	BASIC	BASIC	10E88	IHESPRT	IHESPRT
10E8D	BASIC	BASIC	10E90	IHESPRT	IHESPRT
10E95	BASIC	BASIC	10E98	IHESPRT	IHESPRT
10E9D	BASIC	BASIC	10EA0	IHESPRT	IHESPRT
10EA5	BASIC	BASIC	10EA8	IHESPRT	IHESPRT
10EAC	BASIC	BASIC	10EB4	IHESPRT	IHESPRT
10EB8	BASIC	BASIC	10EC0	IHESPRT	IHESPRT

LOCATION	REFERS TO SYMBOL	IN CONTROL SECTION	LOCATION	REFERS TO SYMBOL	IN CONTROL SECTION
10EC4	BASIC	BASIC	10ECC	IHESPRT	IHESPRT
10ED0	BASIC	BASIC	10ED8	IHESPRT	IHESPRT
10EDC	BASIC	BASIC	10EE4	IHESPRT	IHESPRT
10EE8	BASIC	BASIC	10EF0	IHESPRT	IHESPRT
10EF4	BASIC	BASIC	10EFC	IHESPRT	IHESPRT
10F01	BASIC	BASIC	10F04	IHESPRT	IHESPRT
10F09	BASIC	BASIC	10F0C	IHESPRT	IHESPRT
10F10	BASIC	BASIC	10F18	IHESPRT	IHESPRT
10F1C	BASIC	BASIC	10F24	IHESPRT	IHESPRT
10F28	BASIC	BASIC	10F30	IHESPRT	IHESPRT
10F35	BASIC	BASIC	10F38	IHESPRT	IHESPRT
10F3D	BASIC	BASIC	10F40	RENUMFL	RENUMFL
10F45	BASIC	BASIC	10F48	IHESPRT	IHESPRT
10F4C	BASIC	BASIC	10F54	IHESPRT	IHESPRT
10F58	BASIC	BASIC	10F60	IHESPRT	IHESPRT
10F64	BASIC	BASIC	10F6C	RENUMFL	RENUMFL
10F71	BASIC	BASIC	10F74	SYSIN	SYSIN
10F79	BASIC	BASIC	10F7C	IHESPRT	IHESPRT
10F80	BASIC	BASIC	10F88	SYSIN	SYSIN
10F8D	BASIC	BASIC	10F90	IHESPRT	IHESPRT
10F94	BASIC	BASIC	10F9C	SYSIN	SYSIN
10FA1	BASIC	BASIC	10FA4	SYSIN	SYSIN
10FA9	BASIC	BASIC	EB4C	BASIC	BASIC
EB50	BASIC	BASIC	EB54	BASIC	BASIC
EB58	BASIC	BASIC	EB5C	BASIC	BASIC
EB60	BASIC	BASIC	EB64	BASIC	BASIC
EB68	BASIC	BASIC	EB6C	BASIC	BASIC
EB70	BASIC	BASIC	EB74	BASIC	BASIC
EB78	BASIC	BASIC	EB7C	BASIC	BASIC
EB80	BASIC	BASIC	EB84	BASIC	BASIC
EB88	BASIC	BASIC	EB8C	BASIC	BASIC
EB90	BASIC	BASIC	EB94	BASIC	BASIC
EB98	BASIC	BASIC	EB9C	BASIC	BASIC
EBA0	BASIC	BASIC	EBA4	BASIC	BASIC
EBA8	BASIC	BASIC	EBAC	BASIC	BASIC
EBB0	BASIC	BASIC	EBB4	BASIC	BASIC
EBB8	BASIC	BASIC	EBBC	BASIC	BASIC
EBC0	BASIC	BASIC	EBC4	BASIC	BASIC
EBC8	BASIC	BASIC	EBCC	BASIC	BASIC
EBD0	BASIC	BASIC	EBD4	BASIC	BASIC
EBD8	BASIC	BASIC	EBDC	BASIC	BASIC
EBE0	BASIC	BASIC	EBE4	BASIC	BASIC
EBED	BASIC	BASIC	EBF5	BASIC	BASIC
EBFD	BASIC	BASIC	EC05	BASIC	BASIC
EC0D	BASIC	BASIC	EC15	BASIC	BASIC
EC1D	BASIC	BASIC	EC25	BASIC	BASIC
EC2D	BASIC	BASIC	EC35	BASIC	BASIC
EC3D	BASIC	BASIC	EC45	BASIC	BASIC
EC4D	BASIC	BASIC	EC55	BASIC	BASIC
EC5D	BASIC	BASIC	EC65	BASIC	BASIC
EC6D	BASIC	BASIC	EC75	BASIC	BASIC
EC7D	BASIC	BASIC	14BD8	BASIC	BASIC

LOCATION	REFERS TO SYMBOL	IN CONTROL SECTION	LOCATION	REFERS TO SYMBOL	IN CONTROL SECTION
14BE8	IHESAPC	IHESAP	14FCC	IHEDMAA	IHEDMA
14FD0	IHEVQBA	IHEVQB	14FD4	IHEERRB	IHEERR
14FD8	IHEUPAB	IHEUPA	14FDC	IHEUPBB	IHEUPB
14FE0	IHEUPAA	IHEUPA	14FE4	IHEUPBA	IHEUPB
15074	IHEVFBA	IHEVFB	15078	IHEVFCA	IHEVFC
15080	IHEVFAA	IHEVFA	15084	IHEVPAA	IHEVPA
15088	IHEVKGA	IHEVKG	1508C	IHEVPDA	IHEVPD
15090	IHEVKFA	IHEVKF	15094	IHEVPBA	IHEVPB
15098	IHEVPCA	IHEVPC	150C4	IHEVFDA	IHEVFD
150C8	IHEVFEA	IHEVFE	150D0	IHEVPHA	IHEVPH
150D4	IHEVPGA	IHEVPG	150D8	IHEVKCA	IHEVKC
150DC	IHEVPFA	IHEVPF	150E0	IHEVKBA	IHEVKB
150E4	IHEVPEA	IHEVPE	157F4	IHEM91A	IHEM91
157F8	IHEM91B	IHEM91	157FC	IHEM91C	IHEM91
15800	IHEABND	IHEABN	15824	IHETERA	IHETER
15A0C	IHEDMAA	IHEDMA	15E28	IHEERRB	IHEERR
15E2C	IHEVQAA	IHEVQA	16140	IHEERRB	IHEERR
16E5C	IHEERRB	IHEERR	17EAC	IHEERRB	IHEERR
17F70	IHEERRB	IHEERR	180E8	IHEERRC	IHEERR
182F4	IHEERRB	IHEERR	18578	IHEIOFA	IHEIOF
1857C	IHELDOC	IHELDO	18580	IHEPRTB	IHEPRT
18584	IHEDDPA	IHEDDP	18588	IHEDDPB	IHEDDP
17334	IHEERRB	IHEERR	17524	IHEERRB	IHEERR
176A8	IHEERRC	IHEERR	176B8	IHEERRB	IHEERR
178A8	IHEERRC	IHEERR	178AC	IHEERRB	IHEERR
179B8	IHEERRB	IHEERR	17C28	IHEERRB	IHEERR
1858C	IHEDDPC	IHEDDP	18590	IHEDDPD	IHEDDP
18AF4	IHEERRB	IHEERR	18AF8	IHEERRC	IHEERR
18F0C	IHEERRC	IHEERR	18F10	IHEIOFA	IHEIOF
18F14	IHEVSBA	IHEVSB	18F18	IHEVSCA	IHEVSC
18F1C	IHEDNCA	IHEDNC	191F4	IHEOCLA	IHEOCL
191F8	IHEIOFA	IHEIOF	191FC	IHESPRT	IHESPRT
194BC	IHEDMAA	IHEDMA	194C0	IHEUPAB	IHEUPA
194C4	IHEVSCA	IHEVSC	194C8	IHEVSEB	IHEVSE
194CC	IHEVQCA	IHEVQC	19BC0	IHEIOFA	IHEIOF
19BC8	IHEERRB	IHEERR	19BCC	IHEERRC	IHEERR
19D30	IHEVSEB	IHEVSE	19DB4	IHEERRC	IHEERR
19DD0	IHEERRB	IHEERR	19E24	IHEVSCA	IHEVSC
19F98	IHEERRB	IHEERR	19F9C	IHEERRC	IHEERR
1A08C	IHEERRC	IHEERR	1AAB4	IHEMAIN	IHEMAIN
1AAD4	IHEOCLD	IHEOCL	1AAD8	IHESIZE	IHESIZ
1AADC	IHEBEGA	IHEBEG	1AB60	IHEITAX	IHEIOF
1AB64	IHEERRB	IHEERR	1AB68	IHEERRC	IHEERR
1AB6C	IHETABS	IHETAB	1AB70	IHEITAZ	IHEIOF
1A305	IHEERRA	IHEERR	1ABA4	IHEPRTA	IHEPRT
1ABA8	IHEPRTB	IHEPRT	1ABAC	IHEDDOD	IHEDDO
1ABB0	IHEOCLC	IHEOCL	1AD48	IHEVSDA	IHEVSD
1AD4C	IHEVSEA	IHEVSE	1AD50	IHEKCDA	IHEKCD
1AD54	IHEDCNA	IHEDCN	1AD58	IHEIODG	IHEIOD
1AD5C	IHEIODT	IHEIOD	1AD60	IHEVSCA	IHEVSC
1AFF8	IHEIOFA	IHEIOF	1AFFC	IHEERRB	IHEERR
1B000	IHEERRC	IHEERR	1B118	IHEERRB	IHEERR

LOCATION	REFERS TO SYMBOL	IN CONTROL SECTION	LOCATION	REFERS TO SYMBOL	IN CONTROL SECTION
1B2B4	IHEERRB	IHEERR	1B4E0	IHEIODP	IHEIOD
1B4E4	IHEIODT	IHEIOD	1B4E8	IHEDMAA	IHEDMA
1B4EC	IHEDCNA	IHEDCN	1B4F0	IHEDBNA	IHEDBN
1B4F4	IHEVQCA	IHEVQC	1B650	IHEDMAA	IHEDMA
1B654	IHEUPBB	IHEUPB	1B658	IHEUPAB	IHEUPA
1B65C	IHEERRB	IHEERR	1B78C	IHEERRC	IHEERR
1B790	IHEIODP	IHEIOD	1B794	IHEIODT	IHEIOD
1B798	IHEDNCA	IHEDNC	1B79C	IHEVSCA	IHEVSC
1B7A0	IHEVSEA	IHEVSE	1B7A4	IHEVSBA	IHEVSB
1B7A8	IHEVSFA	IHEVSF	1B870	IHEVSBA	IHEVSB
1B874	IHEDMAA	IHEDMA	1B878	IHEIODP	IHEIOD
1B87C	IHEIODT	IHEIOD	1B880	IHEDCNA	IHEDCN
1B884	IHEDBNA	IHEDBN	1BA44	IHEIOPA	IHEIOP
1BA48	IHEIOPB	IHEIOP	1BA4C	IHEIOPC	IHEIOP
1BA50	IHEIOPA	IHEIOP	1BA54	IHEOCLC	IHEOCL
1BA68	IHEERRB	IHEERR	1BA6C	IHEERRC	IHEERR
1BC4C	IHEIOFA	IHEIOF	1BC50	IHEERRB	IHEERR
1BC54	IHEERRC	IHEERR	1BD9C	IHEERRB	IHEERR
1BDA0	IHEERRC	IHEERR	1BDA4	IHEIODG	IHEIOD
1BDA8	IHEIODP	IHEIOD	1BDAC	IHEIODT	IHEIOD
1BDB0	IHEIOFA	IHEIOF	1C11C	IHEERRC	IHEERR
1C120	IHEERRB	IHEERR	1C124	IHEIOPB	IHEIOP
1C234	IHEERRB	IHEERR	1C238	IHESAFQ	IHESAP
1C434	IHEERRC	IHEERR	1C4C4	IHEERRC	IHEERR
1C4C8	IHEEXS0	IHEEXS	1C4CC	IHELNSE	IHELNS
1C698	IHEERRC	IHEERR	1C7C4	IHEERRC	IHEERR
1C8B8	IHEERRC	IHEERR	1C984	IHEERRC	IHEERR

LOCATION 1820C REQUESTS CUMULATIVE PSEUDO REGISTER LENGTH

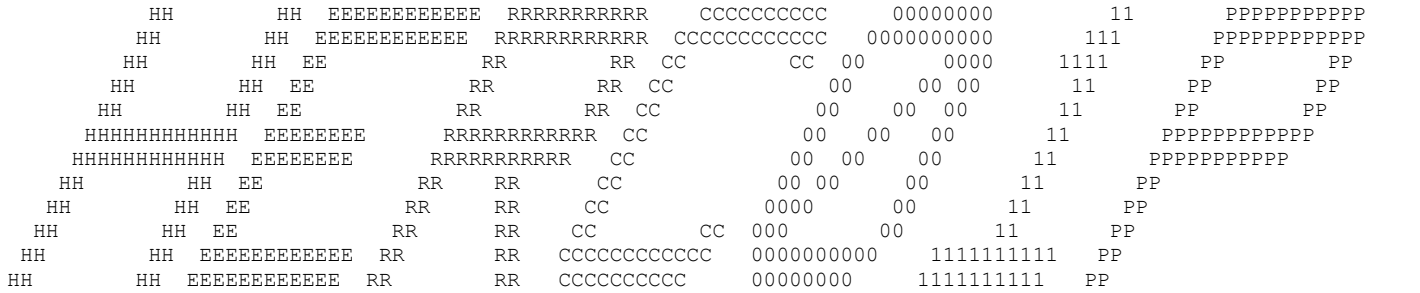
PSEUDO REGISTERS

NAME	ORIGIN	LENGTH	NAME	ORIGIN	LENGTH	NAME	ORIGIN	LENGTH	NAME	ORIGIN	LENGTH
IHEQINV	00	4	IHEQERR	4	4	IHEQTIC	8	4	IHEQLWF	C	4
IHEQSLA	10	4	IHEQLW0	14	4	**BASICB	18	4	**BASICC	1C	4
**BASICD	20	4	**BASICE	24	4	**BASICF	28	4	**BASICG	2C	4
**BASICH	30	4	**BASICI	34	4	**BASICJ	38	4	**BASICK	3C	4
**BASICL	40	4	**BASICM	44	4	**BASICN	48	4	**BASICO	4C	4
**BASICP	50	4	**BASICQ	54	4	**BASICR	58	4	**BASICS	5C	4
**BASICT	60	4	**BASICU	64	4	**BASICV	68	4	**BASICW	6C	4
**BASICX	70	4	**BASICY	74	4	**BASICZ	78	4	**BASIC$	7C	4
**BASIC0	80	4	**BASIC1	84	4	**BASIC2	88	4	**BASIC3	8C	4
**BASIC4	90	4	**BASIC5	94	4	**BASIC6	98	4	**BASIC7	9C	4
**BASIC8	A0	4	**BASIC9	A4	4	**BASIC*	A8	4	**BASIC/	AC	4
**BASIC+	B0	4	**BASIC-	B4	4	**BASIC.	B8	4	**BASIC=	BC	4
**BASIC'	C0	4	/*BASICA	C4	4	/*BASICB	C8	4	/*BASICC	CC	4
/*BASICD	D0	4	/*BASICE	D4	4	/*BASICF	D8	4	/*BASICG	DC	4
/*BASICH	E0	4	/*BASICI	E4	4	/*BASICJ	E8	4	/*BASICK	EC	4
/*BASICL	F0	4	RENUMFL	F4	4	SYSIN	F8	4	IHEQSPR	FC	4
IHEQLSA	100	4	IHEQLW1	104	4	IHEQLW2	108	4	IHEQLW3	10C	4
IHEQLW4	110	4	IHEQLWE	114	4	IHEQLCA	118	4	IHEQVDA	11C	4

NAME	ORIGIN	LENGTH	NAME	ORIGIN	LENGTH	NAME	ORIGIN	LENGTH	NAME	ORIGIN	LENGTH
IHEQFVD	120	4	IHEQLPR	124	4	IHEQFOP	128	4	IHEQCFL	12C	8
IHEQADC	134	4	IHEQXLV	138	8	IHEQEVT	140	8	IHEQSAR	148	4
IHEQRTC	14C	4	IHEQSFC	150	4						

TOTAL LENGTH OF PSEUDO REGISTERS 154
ENTRY ADDRESS 14BE0

TOTAL LENGTH 1C9C0



****A END JOB 498 HERC01P ROOM 10.14.44 AM 10 SEP 17 PRINTER1 SYS BSP1 JOB 498 END A****
****A END JOB 498 HERC01P ROOM 10.14.44 AM 10 SEP 17 PRINTER1 SYS BSP1 JOB 498 END A****
****A END JOB 498 HERC01P ROOM 10.14.44 AM 10 SEP 17 PRINTER1 SYS BSP1 JOB 498 END A****