```
1   *    GENERAL AUTOMATION, INC,  ALL RIGHTS RESERVED
2   ***********************************************
3   *
4   *    PROGRAM NAME  FPH-20
5   *
6   *    MODEL NUMBER  8F020
7   *
8   *    PURPOSE       FORTRAN PHASE-20
9   *
10  *    PROGRAMMER    DICK WALLMANN
11  *
12  ****************   REVISION LIST   **************
13  *
14  *    RV DATE     SCO    BY  REASON FOR CHANGE
15  *    -- -------- ------ --- ----------------------
16  *
17  *    01 11/16/70 NONE   RPH INITIAL RELEASE
18  *
19  ***********************************************
20  ***********************************************
21        HDNG    MPX FORTRAN ** DATA ALLOCATION
22  ***********************************************
23  *STATUS-VERSION 1, MODIFICATION 0
24  *FUNCTION/OPERATION
25  *    * SET UP AND PRINT  VARIABLE ALLOCATIONS
26  *    * ALLOCATE RELATIVE STORAGE FOR ALL COMMON
27  *      AND INSKEL COMMON VARIABLES,
28  *    * ALLOCATES ALL STORAGE RELATIVE TO THE
29  *      BEGINNING OF THE OBJECT PROGRAM AND ALLIGN-
30  *      ED ACCORDING TO ANY EQUIVALENCE STATEMENTS,
31  *      ASSIGNS ALL ALLOCATIONS ACCORDING TO THE
32  *      SPECIFIED PRECISION OF THE PROGRAM.  THE
33  *      SPECIFICATION CAN BE EXPLICIT OR IMPLICIT.
34  *    * PRINTS THE RELATIVE ALLOCATIONS OF THE
35  *      VARIABLES AS THEY ARE ALLOCATED -IF A
36  *      LISTING IS REQUESTED.
37  *    * THE SUBPROLOGUES HEADING EACH SUBROUTINE
38  *      IN THE PHASE RELATE THE SPECIFIC FUNCTION
39  *      OR FUNCTIONS PERFORMED.
40  *
41  *ENTRY POINTS-
42  *    * NEQ IS THE LABEL OF THE INITIAL ENTRY POINT,
43  *      HOWEVER,ENTRY TO THE PHASE OCCURRS AFTER
44  *      EACH BRANCH TO THE PRINTING PHASE.
45  *
46  *INPUT-
47  *    * THE STATEMENT STRING
48  *    * THE SYMBOL TABLE
49  *    * THE FORTRAN COMMUNICATIONS AREA
50  *
51  *OUTPUT-
52  *    * THE STATEMENT STRING
53  *    * THE SYMBOL TABLE
54  *    * THE FORTRAN COMMUNICATIONS AREA
55  *    * ALLOCATION MESSAGES ON THE LIST PRINTER,
56  *      IF REQUESTED.
57  *
58  *EXTERNAL REFERENCES-
59  *    * SUBROUTINES-
```

```
*      ROLRX
*
*EXITS-
*    * NORMAL-
*          PHASE 21 IS LOADED VIA ROLRX AND
*          CONTROL IS PASSED TO IT.
*    * ERRORS-
*      OVERLAP-
*          PROCESSING IS HALTED, PHASE 21 IS
*          CALLED VIA ROLRX.  ONLY OVERLAP ERRORS
*          FROM PREVIOUS PHASES ARE DETECTED.
*      SYNTAX-
*          ERRORS NUMBER 65, 66 OR 67 ARE DETECTED
*          AND PLACED ON STRING.
*
*TABLES/WORK AREAS-
*    * THE STATEMENT STRING
*    * THE SYMBOL TABLE
*    * THE FORTRAN COMMUNICATIONS AREA
*    * BUF+120, A 120 WORD MESSAGE BUILDING AND
*      OUTPUTTING AREA.
*
*ATTRIBUTES-
*    * EACH SUBROUTINE IS DOCUMENTED IN FULL.
*      THE PHASE IS MONOLITHIC IN VERBIAGE
*
*NOTES-NONE
**************************************************
       HDNG    MPX FORTRAN ** DATA ALLOCATION
       ABS     REF CORE
*
*                    SYSTEM EQUATES
MEMRY  EQU     /7FFF CORE      MAXIMUM CORE SIZE
PHSIZ  EQU     4*320           MAXIMUM PHASE SIZE
OVERL  EQU     MEMRY-PHSIZ     PHASES 2-29 START
FCOM   EQU     OVERL-22        FORTRAN COMM. TABLE
PHNTB  EQU     FCOM-56         PHASE TABLE
ROLRX  EQU     PHNTB-50        INTERPHASE CALL
*
LIO    EQU     /85             IOCS ENTRY
       HDNG    MPX FORTRAN ** DATA ALLOCATION
**************************************************
*
*                  FORTRAN COMMUNICATION AREA
*
**************************************************
       ORG     FCOM        FORTRAN COMM AREA
SOFS   BSS     1           START OF STRING
EOFS   BSS     1           END OF STRING
SOFST  BSS     1           START OF SYMBOL TABLE
SOFNS  BSS     1           START OF NON-STMNT NOS.
SOFXT  BSS     1           SIZE OF WORK AREA
SOFGT  BSS     1           SIZE OF CONSTANT AREA
EOFST  BSS     1           END OF SYMBOL TABLE
COMON  BSS     1           HIGH CORE COMMON ADDR
CSIZE  BSS     1           SIZE OF COMMON
ERROR  BSS     1           OVERLAP ERROR
FNAME  BSS     2           PROGRAM NAME
SORF   BSS     1           SUBR (-) OR FUNC (+)
CCWD   BSS     1           CONTROL CARD WD
```

```
120  *                BIT 15 TRANSFER TRACE
121  *                BIT 14 ARITHMETIC TRACE
122  *                BIT 13 EXTENDED PRECISION
123  *                BIT 12 LIST SYMBOL TABLE
124  *                BIT 11 LIST SUBPROGRAM NAMES
125  *                BIT 10 LIST SOURCE PROGRAM
126  *                BIT  9 ONE WORD INTEGERS
127  *                BIT  8 PUNCH
128  *                BIT  7 NONPROCESS PROGRAM
129  IOCS  BSS     1          IOCS CONTROL CARD WORD
130  *
131  *         SEE PHASE ONE FOR BIT PATTERNS
132  *
133  DFCNT BSS     1          DEFINED FILE COUNT
134  *
135  LCOMN BSS     2          INSKEL COMMON SIZE
136  *
137  ICCER BSS     2          IOCS CONTROL CARD ERROR
138  *
139        BSS     2          SYSTEM LOADER USE
140  ****************************************************
141  *
142  *                   END OF FORTRAN COMMUNICATION
143  *                   AREA
144  *
145  ****************************************************
146        HDNG    MPX FORTRAN ** DATA ALLOCATION
147        ORG     OVERL      PHASE ENTRY
148  ****************************************************
149  *
150  *                   PRINT AREA
151  *
152  ****************************************************
153  *
154  *                   INFORMATION FOR SYSTEM LOADER
155  *
156  *                   THIS WORD IS USED TO HOLD THE
157  *                   VALUE FOR INDEX REGISTER 3
158  *                   SO AS TO ALLOW THE PRINT ROUTIN
159  *                   TO RESTORE INDEX REGISTER 3
160  *                   AFTER CALLING THE SYSTEM PRINT
161  *                   ROUTINE
162  NEQ   MDX     IP010      SAVE XR3 HERE
163  *
164  *                   THE FOLLOWING IS THE PRINT
165  *                   AREA PART OF WHICH IS USED FOR
166  *                   INITIAL PROGRAM EXECUTED ONLY
167  *                   ONCE AT THE START OF THE PHASE,
168        DC      ***        USED WITH TYPEWRITER
169  WDCNT DC      ***        PRINT AREA WORD COUNT
170  BUF   EQU     *          BEGINING OF OUTPUT AREA
171  *
172  *                   SET INDEX REGISTER 3 TO POINT
173  *                   AT THE LITERAL POOL AND
174  *                   SUBROUTINES.
175  IP010 LDX  L3 X          LOAD XR3
176        STX   3 NEQ        SAVE FOR PRINT ROUTINE
177  *
178  *                   INITIALIZE THE COMMON LIMITS
179  *
```

```
180         LD      ERROR       CHECK FOR OVERLAP       V1M
181         BSC  L  EXIT,Z      EXIT IF SO              V1M
182         LD   L  COMON       INITIALIZE CSIZE WITH COMO
183         STO  L  CSIZE
184         LD    3 H3FFF-X     INITIALIZE INSKEL LIMITS
185         STO  L  LCOMN
186         STO   3 LCS-X
187  *
188  *                          INITIALIZE THE VARIABLE AREA
189  *                          COUNT TO START ALLOCATION.
190         LD   L  DFCNT       PLACE DEFINE FILE SIZE
191         STO   3 VARCR-X     IN VARCR
192  *
193  *                          SET THE CORRECT REAL VARIABLE
194  *                          SIZE IN RSIZE.
195         LD   L  CCWD        LOAD CONTROL CARD WORD
196         SLA     13          PLACE PRECISION IN SIGN
197         BSC  L  * 2,-       BRANCH IF STANDARD
198         MDX  L  RSIZE,1     INCRAMENT RSIZE TO 3
199  *
200  *                          SET THE CORRECT INTEGER VARIABL
201  *                          SIZE IN ISIZE.
202         LD   L  CCWD        LOAD CONTROL CARD WORD
203         SLA     9           PLACE ONE WORD INT IN SIGN
204         BSC  L  * 2, Z      BRANCH IF ONE WORD INTEGER
205         LD    3 RSIZE-X     SET REAL AND INTEGER SAME
206         STO   3 ISIZE-X
207  *
208  *                          LOAD INDEX REGISTE 1 TO LOOK FO
209  *                          EQUIVALENCE STATEMENTS.
210         LDX  I1 SOFS        LOAD XR1
211  *
212  *                          TEST TO SEE IF STATEMENT IS AN
213  *                          EQUIVALENCE STATEMENT.
214  IP020  LD    1 0           LOAD ID WORD
215         AND   3 HF800-X     SAVE ONLY TYPE
216         EOR   3 HA800-X     TEST FOR EQUIVALENCE
217         BSC  L  IP030, -    BRANCH IF EQUIVALENCE ST,
218  *
219  *                          TEST TO SEE IF STATEMENT IS AN
220  *                          END STATEMENT.
221         EOR   3 HB800-X     TEST FOR END
222         BSC  L  IP040, -    BRANCH IF END STATEMENT
223  *
224  *                          MOVE INDEX REGISTER 1 TO THE
225  *                          NEXT ID WORD.
226         LD    1 0           LOAD ID WORD
227         AND   3 H07FC-X     SAVE STATEMENT NORM
228         SRA     2           POSITION NORM
229         STO     * 1         INITIALIZE MDX INSTRUCTION
230         MDX  L1 *-*         ADJUST INDEX REGISTER 1
231         MDX     IP020       BRANCH
232  *
233  *                          INITIALIZE THE START OF THE
234  *                          EQUIVALENCE STATEMENTS IN INIT.
235  IP 030 MDX   1 2              INCREMENT XR1 BY 1
236         STX  L1 IN01A 1     INITIALIZE INIT ROUTINE
237  *
238  *                          SET ALLOCATION BITS TO REFLECT
239  *                          COMMON FOR EQUIVALENCE STATEMEN
```

```
240  *              EVALUATION.
241         LD    3 H2022-X   PLACE ALLOCATION BITS
242         STO   3 ALOCB-X   IN ALOCB
243  *
244  IP040 LD      ERROR       LOAD ERROR INDICATOR
245         BSC  L  EXIT,Z    BRANCH IF OVERLAP ERROR
246  *
247  *                    BRANCH TO THE MAINLINE PROGRAM.
248  IP999 BSC  L  ML010     BRANCH
249  *
250  *                    THIS IS THE HEADING -VARIABLE
251  *                    ALLOCATION=,
252  TEXT  DC      /E500     V
253         DC      /C100     A
254         DC      /D900     R
255         DC      /C900     I
256         DC      /C100     A
257         DC      /C200     B
258         DC      /D300     L
259         DC      /C500     E
260         DC      /4000     BLANK
261         DC      /C100     A
262         DC      /D300     L
263         DC      /D300     L
264         DC      /D600     O
265         DC      /C300     C
266         DC      /C100     A
267         DC      /E300     T
268         DC      /C900     I
269         DC      /D600     O
270         DC      /D500     N
271         DC      /E200     S
272  *
273  *                    PRINT THE HEADING -VARIABLE
274  *                    ALLOCATIONS=.
275  HD010 LDX  L3 TEXT      PLACE ADDRESS OF PRINT ARE
276         STX  L3 AREA
277         LDX   3 10        PLACE WORD COUNT
278         STX  L3 WDCNT
279         LDX  L3 X         RESTORE XR3 TO POINTER VAL
280         BSI  L  PRINT     CALL PRINT
281  *
282  *                    SET HTEST SWITCH TO INDICATE
283  *                    THAT THE HEADING HAS BEEN
284  *                    PRINTED.
285         STX  L0 HTEST     INDICATE HEADING PRINTED
286  *
287  *                    RESET THE PRINT AREA ADDRESS
288  *                    AND WORD COUNT.
289         LD    3 H003C-X   PLACE FULL WORD COUNT
290         STO  L  WDCNT
291         LD    3 PAP-X     PLACE PRINT AREA ADDRESS
292  *
293  * INPUT       /NONE
294         STO  L  AREA
295         BSC  L  BLKPA 1   BRANCH
296  ********************************************************
297  *
298  *                    END OF PRINT AREA
299  *
```

```
300  ********************************************************
301         HDNG    MPX FORTRAN ** DATA ALLOCATION
302         ORG     BUF+120
303  ********************************************************
304  *
305  *                    MAINLINE
306  *
307  ********************************************************
308  *
309  *                    CALL ALLOC TO ALLOCATE THE
310  *                    BLANK AND INSKEL COMMON
311  *                    VARIABLES.
312  ML010 BSI   L  ALLOC      CALL ALLOC
313  *
314  *                    TEST TO SEE IF ANY EQUIVALENCE
315  *                    STATEMENTS WHERE FOUND AND
316  *                    EVALUATE THEM IF THERE WERE.
317         LD    L  IN01A 1   LOAD EQUIV. NEST ADR.
318         BSI   L  EQUIV,Z   BRANCH IF EQUIV. STS.
319  *
320  *                    CALL ALLOC ROUTINE TO ALLOCATE
321  *                    THE REAL VARIABLES.
322  ML020 BSI   L  ALLOC      CALL ALLOC
323  *
324  *                    CALL ALLOC ROUTINE TO ALLOCATE
325  *                    THE INTEGER VARIABLES.
326         BSI   L  ALLOC      CALL ALLOC
327  *
328  *                    CALCULATE COMMON SIZE AND MAKE
329  *                    IT EVEN.
330         LD    L  COMON      LOAD UPPER COMMON ADR
331         S     L  CSIZE      SUBTRACT LOWER ADR
332         BSC      E          SKIP IF EVEN
333         A      3 H0001-X    MAKE EVEN
334         STO   L  CSIZE      STORE COMMON SIZE
335  *
336  *                    CALCULATE INSKEL COMMON SIZE
337  *                    AND MAKE IT EVEN.
338         LD    L  LCOMN      LOAD UPPER COMMON ADR
339         S      3 LCS-X      SUBTRACT LOWER ADR
340         BSC      E          SKIP IF EVEN
341         A      3 H0001-X    MAKE EVEN
342         STO   L  LCOMN      STORE INSKEL COMMON SIZE
343  *
344  *                    INSERT THE WORK AREA SIZE INTO
345  *                    THE COMMUNICATION AREA
346         LD     3 VARCR-X    LOAD WORK AREA SIZE
347         BSC      E          SKIP IF EVEN
348         A      3 H0001-X    MAKE EVEN
349         STO   L  SOFXT      STORE EVEN WORK AREA SIZE
350  *
351  *                    PLACE CONSTANT AREA SIZE IN
352  *                    SOFGT.
353         LD     3 CAC-X      LOAD CONSTANT AREA SIZE
354         STO   L  SOFGT      SAVE IN SOFGT
355  *
356  *                    TEST TO SEE IF PARTIAL LINE
357  *                    REMAINS TO BE PRINTED.
358         LD     3 PAP-X      LOAD PRINT AREA POINTER
359         S      3 RPAP-X     SUBTRACT START OF PRINT AR
```

```
360         BSI  L  PRINT,Z   BRANCH TO PRINT PART LINE
361   *
362   *                       CALL IN THE NEXT PHASE OF THE
363   *                       COMPILER
364   EXIT  BSI  L  ROLRX     CALL DOWN PHASE 21
365         DC      21            NEXT PHASE NUMBER
366   *
367   *                       LOAD THE NUMBER 65 INTO
368   *                       THE A REGISTER
369   ERR65 LD     3 H0041-X  LOAD 65
370         MDX      ERR      BRANCH
371   *
372   *                       LOAD THE NUMBER 66 INTO
373   *                       THE A REGISTER,
374   ERR66 LD     3 H0042-X  LOAD 66
375         MDX      ERR      BRANCH
376   *
377   *                       LOAD THE NUMBER 67 INTO
378   *                       THE A REGISTER,
379   ERR67 LD     3 H0043-X  LOAD 67
380   *
381   *                       PLACE THE ERROR NUMBER AND
382   *                       INDICATOR.
383   ERR   LDX   I1 IN01A 1  SET XR1 TO POINT AT EQUIV.
384         STO    1 0        PLACE ERROR NUMBER
385         LD     3 H0FFF-X  LOAD ERROR INDICATOR
386         STO    1 -1       PLACE ERROR INDICATOR
387   *
388   *                            REMOVE SPECIAL INDICATORS
389   *
390         BSI  L  RMOVE
391   *
392   *                            CLEAR COMMON INDICATORS
393   *
394         LD     3 H0022-X
395         STO    3 ALOC8-X
396         MDX      ML020    BRANCH
397   ***************************************************
398   *
399   *                       END OF MAINLINE PROGRAM
400   *
401   ***************************************************
402   X     EQU      * 128    INDEX REGISTER 3 VALUE
403   SW1   DC       ***      SWITCH 1
404   SW2   DC       ***      SWITCH 2
405   SW3   DC       ***      SWITCH 3
406   SW4   DC       ***      SWITCH 4
407   ODDSW DC       ***      ODD SWITCH
408   EVSW  DC       ***      EVEN SWITCH
409   TRACK DC       -1       TRACK SWITCH
410   LOW   DC       ***      LOW ADR. OF NEST
411   HIGH  DC       ***      HIGH ADR. OF NEST
412   DEFIN DC       ***      DEFIN IND. FOR NEST
413   RELAD DC       ***      RELATIVE ADR. OF NEST
414   D4    DC       ***      OFFSET FOR SUBSCRIPT
415   NRA   DC       ***     NST RETURN ADR.
416       ***        SAVE AREA FOR NRA
417   BASE  DC       ***      BASE ADR. OF NEST
418   VARCR DC       ***      ADR. OF VARIABLE CORE
419   TAGLP DC       /0312    TAGED LEFT PARENTHESIS
```

```
420  VSIZE  DC      ***      VARIABLE SIZE
421  ISIZE  DC      1        INTEGER SIZE
422  RSIZE  DC      2        REAL SIZE
423  ASIZE  DC      ***      ARRAY SIZE
424  CAC    DC      ***      CONSTANT AREA COUNT
425  PAP    DC      BUF      PRINT AREA POINTER
426  LNEND  DC      BUF+60   PRINT LINE END (CONSTANT)
427  RPAP   DC      BUF      PRINT AREA ADDR (CONSTANT)
428  SAVAD  DC      ***      SAVE HEXIDECIMAL ADDRESS
429  LCS    DC      ***      INSKEL COMMON SIZE
430  SALOC  DC      ***      SAVE ALLOCATION
431  PASSW  DC      ***      PASS SWITCH
432  ALOCB  DC      /0022    ALLOCATION BITS
433         BSS  E  0
434  NAME   DC      ***      PACKED NAME
435         DC      ***      2ND WD OF NAME
436  HTEST  DC      ***      HEADER SWITCH
437  HFFFF  DC      /FFFF    CONSTANT
438  HF800  DC      /F800    CONSTANT
439  HB800  DC      /B800    CONSTANT
440  HA800  DC      /A800    CONSTANT
441  H8000  DC      /8000    CONSTANT
442  H4000  DC      /4000    CONSTANT
443  H3FFF  DC      /3FFF    CONSTANT
444  H2022  DC      /2022    CONSTANT
445  H1800  DC      /1800    CONSTANT
446  H0FFF  DC      /0FFF    CONSTANT
447  H0C00  DC      /0C00    CONSTANT
448  H07FF  DC      /07FF    CONSTANT
449  H07FC  DC      /07FC    CONSTANT
450  H07F2  DC      /07F2    CONSTANT
451  H0300  DC      /0300    CONSTANT
452  H00D9  DC      /00D9    CONSTANT
453  H00C9  DC      /00C9    CONSTANT
454  H00C5  DC      /00C5    CONSTANT
455  H00C3  DC      /00C3    CONSTANT
456  H00C0  DC      /00C0    CONSTANT
457  H007E  DC      /007E    CONSTANT
458  H0060  DC      /0060    CONSTANT
459  H005D  DC      /005D     )
460  H005C  DC      /005C    CONSTANT
461  H004D  DC      /004D     (
462  H0041  DC      /0041    CONSTANT
463  H0042  DC      /0042    CONSTANT
464  H0043  DC      /0043    CONSTANT
465  H0040  DC      /0040    CONSTANT
466  H003F  DC      /003F    CONSTANT
467  H003C  DC      /003C    CONSTANT
468  H0039  DC      /0039    CONSTANT
469  H0028  DC      /0028    CONSTANT
470  H0022  DC      /0022    CONSTANT
471  H0020  DC      /0020    CONSTANT
472  H0012  DC      /0012    CONSTANT
473  H000F  DC      /000F    CONSTANT
474  H000C  DC      /000C    CONSTANT
475  H0009  DC      /0009    CONSTANT
476  H0003  DC      /0003    CONSTANT
477  H0002  DC      /0002    CONSTANT
478  H0001  DC      /0001    CONSTANT
479         HDNG    MPX FORTRAN ** DATA ALLOCATION
```

```
480  ********************************************************
481  *
482  * ROUTINE NAME/ARRL
483  *
484  * FUNCTION     /PLACE IN VSIZE THE SIZE OF EACH
485  *               ELEMENT AN IN ASIZE THE TOTAL SIZE
486  *               OF THE ARRAY.
487  *
488  * ENTRY        /ARRL
489  *
490  * INPUT        /NO SPECIAL INPUT OTHER THAN THE
491  *               SYMBOL TABLE.
492  *
493  * OUTPUT       /VSIZE AND ASIZE WILL BE SET.
494  *
495  * EXTERNAL
496  * REFERANCES   /SUBROUTINES  VARFO  ARE CALLED AND
497  *               REFERANCES TO  VSIZE,ASIZE,H1800
498  *               ARE MADE.
499  *
500  * ERROR        /NONE
501  *
502  * NOTE         /CALLING SEQUENCE    BSI ARRL
503  *
504  ********************************************************
505  *
506  *                    ENTRY-ARRL
507  ARRL  DC       *-*        LINK WORD
508  *
509  *                    CALL VARFO TO DETERMINE THE
510  *                    SIZE OF THE PRESENT SYMBOL.
511         BSI    3 VARFO-X   CALL VARFO
512  *
513  *                    PLACE THE SIZE OF THE SINGLE
514  *                    ELEMENT IN ASIZE.
515         LD     3 VSIZE-X   LOAD THE ELEMENT SIZE
516         STO    3 ASIZE-X   PLACE IN ARRAY SIZE
517  *
518  *                    TEST THE SYMBOL TO SEE IF IT
519  *                    IS DIMENSIONED.
520         LD     2 0         LOAD INDICATOR WORD
521         AND    3 H1800-X   SAVE DIMENSION IND. BITS
522         BSC  I ARRL,-      BR IF NOT DIMENSIONED
523  *
524  *                    CALCULATE THE SIZE OF THE
525  *                    ARRAY.
526         LD     2 -3        LOAD NUMBER OF ELEMENTS
527         M      3 VSIZE-X   MULTIPLY BY ELEMENT SIZE
528         RTE      16
529         STO    3 ASIZE-X   STORE ARRAY SIZE
530  *
531  *                    RETURN
532         BSC  I ARRL        RETURN
533  ********************************************************
534  *
535  *              END OF ARRL ROUTINE
536  *
537  ********************************************************
538         HDNG   MPX FORTRAN ** DATA ALLOCATION
539  ********************************************************
```

```
*
* ROUTINE NAME/VARFO
*
* FUNCTION     /PLACE IN VSIZE THE SIZE OF THE
*               VARIABLE OR CONSTANT THAT IS BEING
*               LOOKED AT.
*
* ENTRY        /VARFO
*
* INPUT        /NO SPECIAL INPUT OTHER THAN THE
*               SYMBOL TABLE.
*
* OUTPUT       /VSIZE WILL BE SET.
*
* EXTERNAL
* REFERANCES   /REFERANCES TO  ISIZE,VSIZE,RSIZE
*               ARE MADE.
*
* ERROR        /NONE
*
* NOTE         /CALLING SEQUENCE    BSI VARFO
*
*****************************************************
*
*                    ENTRY-VARFO
VARFO DC       *-*        LINK WORD
*
*                    SET VARIABLE SIZE TO THAT OF
*                    THE INTEGER VARIABLES.
       LD     3 ISIZE-X   LOAD INTEGER SIZE
       STO    3 VSIZE-X   STORE IN VARIABLE SIZE
*
*                    TEST TO SEE IF THE VARIABLE IS
*                    REAL OR INTEGER.
       LD     2 0         LOAD INDICATOR WORD
       SLA      1         PLACE REAL IND. IN SIGN
       BSC   I  VARFO, Z  BR IF INTEGER
*
*                    SET THE VARIABLE SIZE TO THAT O
*                    THE REAL VARIABLES.
       LD     3 RSIZE-X   LOAD REAL SIZE
       STO    3 VSIZE-X   STORE IN VARIABLE SIZE
*
*                    RETURN
       BSC   I  VARFO     RETURN
*****************************************************
*
*              END OF VARFO ROUTINE
*
*****************************************************
       HDNG    MPX FORTRAN ** DATA ALLOCATION
*****************************************************
*
* ROUTINE NAME/RTN
*
* FUNCTION     /POINTS INDEX REGISTER 1 TO THE FIRS
*               SYMBOL IN THE NEST NOW BEING WORKED
*               ON. INDEX REGISTER 2 IS ALSO SET TO
*               POINT AT SYMBOL TABLE.
*
```

```
600   * ENTRY        /RTN
601   *
602   * INPUT        /NO SPECIAL INPUT.
603   *
604   * OUTPUT       /XR1 AND XR2 ARE SET.
605   *
606   * EXTERNAL
607   * REFERENCES   /SUBROUTINES  XR2  AND REQUIRED AND
608   *               REFERANCES TO  NRA  ARE MADE.
609   *
610   * ERROR        /NONE
611   *
612   * NOTES        /NONE
613   *
614   ************************************************
615   *
616   *                    ENTRY-RTN
617   RTN    DC       *-*        LINK WORD
618   *
619   *                    RESTORE INDEX REGISTER 1 TO
620   *                    POINT AT THE FIRST VARIABLE
621   *                    IN THE PRESENT NEST.
622          LDX   I1 NRA        LOAD XR1 NEST RETURN ADR.
623   *
624   *                    CALL XR2 TO SET INDEX REGISTER
625   *                    TO POINT AT THE INDICATOR WORD
626   *                    OF THE SYMBOL XR1 POINTS AT.
627          BSI    3 XR2-X      CALL XR2
628   *
629   *                    RETURN
630          BSC   I  RTN        RETURN
631   ************************************************
632   *
633   *              END OF RTN ROUTINE
634   *
635   ************************************************
636          HDNG     MPX FORTRAN ** DATA ALLOCATION
637   ************************************************
638   *
639   * ROUTINE NAME/XR2
640   *
641   * FUNCTION     /THIS ROUTINE PICKS UP THE WORD
642   *               SPECIFIED BY INDEX REGISTER 1,
643   *               COMPUTES THE SYMBOL TABLE ADDRESS,
644   *               AND LOAD THAT ADDRESS INTO INDEX
645   *               REGISTER 2.
646   *
647   * ENTRY        /XR2
648   *
649   * INPUT        /NO SPECIAL INPUT OTHER THAN THE
650   *               STRING AND SYMBOL TABLE.
651   *
652   * OUTPUT       /INDEX REGISTER 2 IS SET.
653   *
654   * EXTERNAL
655   * REFERANCES   /REFERANCES TO  SUFST,H07FF  ARE
656   *               MADE.
657   *
658   * ERROR        /NONE
659   *
```

```
660   * NOTES        /CALLING SEQUENCE    BSI XR2
661   *
662   ****************************************************
663   *
664   *                      ENTRY-XR2
665   XR2    DC       *-*          LINK WORD
666   *
667   *                      LOAD INDEX REGISTER 2 WITH THE
668   *                      SYMBOL TABLE ADDRESS OF SYMBOL.
669          LD     1 0            LOAD SYMBOL
670          AND    3 H07FF-X      SEPARATE OFFSET
671          STO      XR2A         SAVE OFFSET
672          LD     L SOFST        CALCULATE
673          A       3 H0003-X
674          S         * 4                SOFST-3*OFSET 3
675          S         * 3
676          S         * 2
677          STO       * 1          INITIALIZE LDX INSTRUCTION
678          LDX    L2 *-*          LOAD XR2
679   XR2A   EQU       *-1          MODIFIABLE
680   *
681   *                      RETURN
682          BSC    I  XR2          RETURN
683   ****************************************************
684   *
685   *               END OF XR2 ROUTINE
686   *
687   ****************************************************
688          HDNG     MPX FORTRAN ** DATA ALLOCATION
689   ****************************************************
690   *
691   * ROUTINE NAME/HILO
692   *
693   * FUNCTION     /THE HIGH AND LOW VALUES AND THE
694   *               DEFINE INDICATORS ARE SAVE FOR
695   *               A SET OF NESTS OF EQUIVALENCE
696   *               STATEMENTS.
697   *
698   * ENTRY        /HILO
699   *
700   * INPUT        /THE VALUE OF THE VARIABLE BEING
701   *               LOOKED AT AS WELL AS THE SYMBOL
702   *               TABLE.
703   *
704   * OUTPUT       /LOW,HIGH AND DEFIN WILL BE SET.
705   *
706   * EXTERNAL
707   * REFERANCES   /SUBROUTINES  ARRL,GETD4  ARE CALLED
708   *               AND REFERANCES  RELAD,VSIZE,BASE,
709   *               ASIZE,LOW,H0001,HIGH,DEFIN  ARE
710   *               MADE.
711   *
712   * ERROR        /NONE
713   *
714   * NOTE         /CALLING SEQUENCE   BSI HILO
715   *
716   ****************************************************
717   *
718   *                      ENTRY-HILO
719   HILO   DC       *-*          LINK WORD
```

```
720  *
721  *                    CALL ARRL TO OBTAIN THE VARIABL
722  *                    SIZE AND THE ARRAY SIZE.
723         BSI    3 ARRL-X     CALL ARRL
724  *
725  *                    CALL GETD4 TO OBTAIN THE OFFSET
726  *                    FOR THE PRESENT VARIABLE.
727         BSI      GETD4      CALL GETD4
728  *
729  *                    CALCULATE THE BASIS FOR
730  *                    DETERMINING IF THIS ARRAY OR
731  *                    ELEMENT WILL DEFINE A NEW
732  *                    LOW OR HIGH FOR THIS NEST.
733         A      3 RELAD-X    CALCULATE
734         A      3 VSIZE-X    BASE D4 RELAD-VSIZE
735         STO    3 BASE-X
736  *
737  *                    TEST SWITCH 3.
738         LD     3 SW3-X      LOAD SWITCH 3
739         BSC  L  HIL03, -     BRANCH IF EQUIV. COMMON
740  *
741  *                  TEST TO SEE IF REAL VARIABLES ARE
742  *                  STANDARD PRECISION.
743  *
744         LD   L  CCWD         LOAD CONTROL CARD WORD
745         SLA     13           PLACE PRECISION IN SIGN
746         BSC  L  HIL05, Z     BRANCH EXTEND. PRECISION
747  *
748  *                  TEST TO SEE IF REAL VARIABLE BEING
749  *                  LOOKED AT.
750  *
751         LD     2 0           LOAD INDICATOR WORD
752         SLA     1            PLACE TYPE IN SIGN
753         BSC  L  HIL05, Z     BRANCH IF INTEGER
754  *
755  *                  TEST TO SEE IF THE LOW VALUE OF
756  *                  THE ARRAY OR VARIABLE IS EVEN.
757  *
758         LD     3 BASE-X      COMPUTE LOW ADDRESS
759         S      3 ASIZE-X
760         BSC  L  HIL06,E      BRANCH IF ODD
761  *
762  *                  TEST TO SEE IF THE ODD SWITCH IS
763  *                  SET ON.
764  *
765         LD     3 ODDSW-X     LOAD ODD SWITCH
766         BSC  L  ERR67,Z      BRANCH IF ALLOCATION ERROR
767  *
768  *                  SET THE EVEN SWITCH ON.
769  *
770         STX  L0 EVSW         SET EVEN SWITCH
771         MDX     HIL05        BRANCH
772  *
773  *                  TEST TO SEE IF THE EVEN SWITCH IS
774  *                  SET ON.
775  *
776  HIL06  LD     3 EVSW-X      LOAD EVEN SWITCH
777         BSC  L  ERR67,Z      BRANCH IF ALLOCATION ERROR
778  *
779  *                  SET ODD SWITCH ON.
```

```
780   *
781           STX  L0 ODDSW
782   *
783   *                 TEST TO SEE IF THIS NEST WILL
784   *                      PRODUCE A NEW LOW.
785   HILO5 LD     3 BASE-X     LOAD PRESENT LOW
786           S      3 ASIZE-X
787           S      3 LOW-X      TEST AGAINST TOTAL LOW
788           BSC  L  HILO1,-     BR IF NOT NEW LOW
789   *
790   *                      SAVE THE NEW LOW FOR ALLOCATION
791   *                      PRUPOSES.
792           A      3 LOW-X      RESTORE NEW LOW
793           STO    3 LOW-X      SAVE NEW LOW
794   *
795   *                      TEST TO SEE IF THIS NEST WILL
796   *                      PRODUCE A NEW HIGH.
797   HILO1 LD     3 BASE-X     LOAD PRESENT HIGH
798           S      3 H0001-X
799           S      3 HIGH-X     TEST AGAINST TOTAL HIGH
800           BSC  L  HILO2,      BR IF NOT NEW HIGH
801   *
802   *                      SAVE THE NEW HIGH FOR ALLOCATIO
803   *                      PURPOSES.
804           A      3 HIGH-X     RESTORE NEW HIGH
805           STO    3 HIGH-X     SAVE NEW HIGH
806   *
807   *                      SAVE THE INDICATOR WORDS FOR
808   *                      THE SYMBOLS IN THE NESTS.
809   HILO2 LD     3 DEFIN-X    LOAD ACCUMULATED IND. WORD
810           OR     2 0          OR IN NEW IND. WORD
811           STO    3 DEFIN-X    SAVE ACCUMULATED IND. WORD
812   *
813   *                                RETURN
814   HILOX BSC  I  HILO         RETURN
815   *
816   *                      TEST TO SEE IF THE VARIABLE
817   *                      NOW BEING LOOKED AT IS IN INSKE
818   *                      COMMON.
819   HILO3 LD     2 0          LOAD INDICATOR WORD
820           SLA    15           PLACE INSKEL IND. IN SIGN
821           BSC  L  HILO4,-     BRANCH IF IN BLANK COMMON
822   *
823   *                      TEST TO SEE IF THE START OF
824   *                      INSKEL COMMON IS BEING EXTENDED
825           LD     3 BASE-X     LOAD PRESENT HIGH
826           S      3 H0001-X
827           S    L  LCOMN       TEST AGAINST INSKEL HIGH
828           BSC  L  ERR66,-Z    BRANCH IF EXTENDING COMMON
829   *
830   *                      TEST TO SEE IF THE END OF INSKE
831   *                      COMMON IS BEING EXTENDED.
832           LD     3 BASE-X     LOAD PRESENT LOW
833           S      3 ASIZE-X
834           S      3 LCS-X      TEST AGAINST INSKEL LOW
835           BSC  L  HILO2,-     BRANCH IF NOT NEW LOW
836   *
837   *                      INDICATE THE NEW LOW FOR INSKEL
838   *                      COMMON.
839           A      3 LCS-X      RESTORE NEW LOW
```

```
840         STO   3 LCS-X      SAVE NEW LOW
841         MDX     HIL02      BRANCH
842   *
843   *                        TEST TO SEE IF THE START OF
844   *                        BLANK COMMON IS BEING EXTENDED.
845   HIL04 LD    3 BASE-X     LOAD PRESENT HIGH
846         S     3 H0001-X
847         S     L COMON      TEST AGAINST COMMON HIGH
848         BSC   L ERR66,-Z   BRANCH IF EXTENDING COMMON
849   *
850   *                        TEST TO SEE IF THE END OF BLANK
851   *                        COMMON IS BEING EXTENDED.
852         LD    3 BASE-X     LOAD PRESENT LOW
853         S     3 ASIZE-X
854         S     L CSIZE      TEST AGAINST COMMON LOW
855         BSC   L HIL02,-    BRANCH IF NOT NEW LOW
856   *
857   *                        INDICATE THE NEW LOW FOR BLANK
858   *                        COMMON.
859         A     L CSIZE      RESTORE NEW LOW
860         STO   L CSIZE      SAVE NEW LOW
861         MDX     HIL02      LOOP
862   *
863   ***************************************************
864   *
865   *              END OF HILO ROUTINE
866   *
867   ***************************************************
868         HDNG    MPX FORTRAN ** DATA ALLOCATION
869   ***************************************************
870   *
871   * ROUTINE NAME/GETD4
872   *
873   * FUNCTION    /PLACE IN D4 THE OFFSET INDICATED
874   *              BY THE SUBSCRIPT IN EQUIVALENCE.
875   *
876   * ENTRY       /GETD4
877   *
878   * INPUT       /INDEX REGISTER 1 POINTING AT THE
879   *              STRING AND INDEX REGISTER 2
880   *              POINTING AT THE SYMBOL TABLE.
881   *
882   * OUTPUT      /D4 WILL BE SET,
883   *
884   * EXTERNAL
885   * REFERANCES  /REFERANCES TO  D4,H1800,H8000 ARE
886   *              MADE.
887   *
888   * ERROR       /NONE
889   *
890   * NOTES       /CALLING SEQUENCE   BSI GETD4
891   *
892   ***************************************************
893   *
894   *
895   GETD4 DC      *-*        ENTRY TO GETD4 RTN
896   *
897   *                        PLACE A ZERO IN D4.
898         SLA     16         ZERO A REGISTER
899         STO   3 D4-X       STORE IN D4
```

```
900   *
901   *                    CHECK TO SEE IF THE SYMBOL IS
902   *                    DIMENSIONED.
903         LD     2 0           LOAD INDICATOR WORD
904         AND    3 H1800-X     SAVE DIMENSION IND. BITS
905         BSC  I GETD4, -      BR IF NOT DIMENSIONED
906   *
907   *                    PLACE THE OFFSET IN D4.
908         LD     1 2           LOAD OFFSET
909         EOR    3 H8000-X     REMOVE LEADING BIT.
910         STO    3 D4-X        STORE D4
911   *
912   *                    RETURN
913         BSC  I GETD4         RETURN
914   ***************************************************
915   *
916   *              END OF GETD4 ROUTINE
917   *
918   ***************************************************
919         HDNG   MPX FORTRAN ** DATA ALLOCATION
920   ***************************************************
921   *
922   * ROUTINE NAME/EVENA
923   *
924   * FUNCTION     /MAKE THE VALUE OF VARCR EVEN.
925   *
926   * ENTRY        /EVENA
927   *
928   * INPUT        /THE VALUE TO BE MADE EVEN IN VARCR.
929   *
930   * OUTPUT       /VARCR IS EVEN.
931   *
932   * EXTERNAL
933   * REFERANCES   /REFERANCES TO  VARCR,H0001,RSIZE
934   *               ARE MADE.
935   *
936   * ERROR        /NONE
937   *
938   * NOTE         /CALLING SEQUENCE   BSI EVENA
939   *
940   ***************************************************
941   *
942   *                    ENTRY-EVENA
943   EVENA DC       ***        LINK WORD
944   *
945   *                    TEST TO SEE IF THE SIZE OF THE
946   *                    VARIABLES ARE STANDARD OR
947   *                    EXTENDED PRECISION.
948         LD     3 RSIZE-X     LOAD REAL SIZE
949         BSC  I EVENA,E       BR IF EXTEND. PRECISION
950   *
951   *                    TEST VARCR FOR BEING EVEN.
952         LD     3 VARCR-X     LOAD VARIABLE CORE ADR.
953         BSC    E             SKIP IF EVEN
954   *
955   *                    INCREMENT THE VARIABLE CORE ADR
956   *                    BY 1 TO MAKE IT EVEN.
957         A      3 H0001-X     INCREMENT VARCR BY 1
958         STO    3 VARCR-X     STORE VARCR
959   *
```

```
960   *                    RETURN
961         BSC  I  EVENA     RETURN
962   ***************************************************
963   *
964   *              END OF EVENA ROUTINE
965   *
966   ***************************************************
967         HDNG    MPX FORTRAN ** DATA ALLOCATION
968   ***************************************************
969   *
970   * ROUTINE NAME/INCR
971   *
972   * FUNCTION     /THIS ROUTINE POSITIONS INDEX
973   *               REGISTERS 1 AND 2 DURING HANDLING
974   *               OF EQUIVALENCE STATEMENTS
975   * ENTRY POINTS/INCR MOVES INDEX REGISTER 1 TO THE
976   *               NEXT SYMBOL IN THE EQUIVALENCE
977   *               STATEMENTS, SKIPS ANY DELETED
978   *               NESTS.
979   *
980   *               INIT PLACES INDEX REGISTER 1 TO THE
981   *               FIRST NON-DELETED NEST.
982   *
983   * INPUT        /NO SPECIAL INPUT OTHER THAN THE
984   *               SYMBOL TABLE.
985   *
986   * OUTPUT       /INDEX REGISTER 1 WILL BE POINTING
987   *               AT THE STRING AND INDEX REGISTER 2
988   *               WILL POINT AT THE ENTRY IN THE
989   *               SYMBOL TABLE ASSOCIATED WITH THE
990   *               STRING SYMBOL.
991   *
992   * EXTERNAL
993   * REFERENCES   /SUBROUTINES  MOVE,XR2  ARE REQUIRED
994   *               AND REFERENCES TO  NRA,H8000  ARE
995   *               MADE.
996   *
997   * ERROR        /NONE
998   *
999   * NOTES        /STORAGE CELL AT IN01A 1 IS FILLED I
1000  *               BY INITIALIZATION WITH THE ADDRESS
1001  *               OF THE FIRST EQUIVALENCE STATEMENT
1002  *               OF A DUMMY IF EQUIVALENCE ARE
1003  *               PRESENT.
1004  *
1005  ***************************************************
1006  *
1007  *                    ENTRY-INCR
1008  INCR  DC       ***        LINK WORD
1009  *
1010  *                    CALL MOVE TO MOVE POINTER TO TH
1011  *                    NEXT SYMBOL IN THE EQUIVALENCE
1012  *                    STATEMENTS.
1013        BSI      MOVE       CALL MOVE
1014        DC       IN01E      BR IF END OF ALL EQUIV
1015        DC       IN01B      BR IF END OF NEST
1016  *
1017  *                    CALL XR2 TO POINT INDEX REGISTE
1018  *                    2 AT THE INDICATOR WORD OF THE
1019  *                    SYMBOL TABLE ENTRY.
```

```
1020          BSI    3 XR2-X      CALL XR2
1021    *
1022    *                         RETURN LINK 2
1023          MDX  L   INCR,2      INCREMENT LINK WORD BY 2
1024          BSC  I   INCR        RETURN
1025    *
1026    *                         ENTRY-INIT
1027    INIT  DC       *-*        LINK WORD
1028    *
1029    *                         MOVE THE LINK WORD TO THE INCR
1030    *                         ENTRY POINT.
1031          LD       INIT       LOAD LINK WORD=INIT
1032          STO      INCR       STORE LINK WORD-INCR
1033    *
1034    *                         SET INDEX REGISTER 1 TO POINT A
1035    *                         THE FIRST NEST.  THIS ADDRESS I
1036    *                         COMPUTED IN THE INITIALIZATION
1037    *                         AN IS STORED HERE.
1038    IN01A LDX  L1 *-*         LOAD XR1 WITH NEST ADDRESS
1039    *
1040    *                         SAVE INDEX REGISTER 1 FOR THE
1041    *                         RTN ROUTINE.
1042    IN01B STX  L1 NRA         SAVE XR1
1043    *
1044    *                         TEST THE LEFT PARENTHESIS TO SE
1045    *                         IF THE NEST HAS BEEN DELETED.
1046          LD     1 -1         LOAD LEFT PARENTHESIS
1047          EOR    3 H8000-X    CHECK FOR DELETE CODE
1048          BSC  L   IN01C, -   BR IF DELETE CODE
1049    *
1050    *                         CALL XR2 TO POINT INDEX REGISTE
1051    *                         2 AT THE INDICATOR WORD OF THE
1052    *                         SYMBOL TABLE ENTRY.
1053          BSI    3 XR2-X      CALL XR2
1054    *
1055    *                         INCREMENT LINK WORD BY 1
1056          MDX  L   INCR,1     INCREMENT LINK WORD BY 1
1057    *
1058    *                         RETURN C  LINK
1059    *
1060    IN01E LD       INCR       INITIALIZE BRANCH INST.
1061          STO      * 1        MODIFY EXIT
1062          BSC  I   *-*        RETURN
1063    *
1064    *                         CALL XR2 TO POINT INDEX REGISTE
1065    *                         2 AT THE INDICATOR WORD OF THE
1066    *                         SYMBOL TABLE ENTRY.
1067    IN01C BSI    3 XR2-X      CALL XR2
1068    *
1069    *                         CALL MOVE TO MOVE POINTER TO TH
1070    *                         NEXT SYMBOL IN THE EQUIVALENCE
1071    *                         STATEMENTS.
1072          BSI      MOVE       CALL MOVE
1073          DC       IN01E      BR IF END OF ALL EQUIV.
1074          DC       IN01B      BR IF END OF NEST
1075          MDX      IN01C      LOOP
1076    ********************************************************
1077    *
1078    *              END OF INCR ROUTINE
1079    *
```

```
1080  ************************************************
1081        HDNG    MPX FORTRAN ** DATA ALLOCATION
1082  ************************************************
1083  *
1084  * ROUTINE NAME/MOVE
1085  *
1086  * FUNCTION     /MOVE INDEX REGISTER 1 TO THE NEXT
1087  *               SYMBOL IN THE EQUIVALENCE STATEMENT
1088  *               PRESENT.
1089  *
1090  * ENTRY        /MOVE
1091  *
1092  * INPUT        /NO SPECIAL INPUT OTHER THAN THE
1093  *               SYMBOL TABLE.
1094  *
1095  * OUTPUT       /INDEX REGISTER 1 WILL POINT AT THE
1096  *               NEXT SYMBOL IN THE EQUIVALENCE
1097  *               STATEMENTS.
1098  *
1099  * EXTERNAL
1100  * REFERENCES   /REFERENCES TO  H1800,H0012,HF800,
1101  *               HA800  ARE MADE.
1102  *
1103  * ERROR        /NONE
1104  *
1105  * NOTES        /NONE
1106  *
1107  ************************************************
1108  *
1109  *                    ENTRY-MOVE
1110  MOVE  DC      *-*        LINK WORD
1111  *
1112  *                    CHECK TO SEE IF PRESENT SYMBOL
1113  *                    IS DIMENSIONED.
1114        LD     2 0           LOAD INDICATOR WORD
1115        AND    3 H1800-X     SAVE DIMENSION BITS
1116        BSC  L MV01A,-       BR IF NOT DIMENSIONED
1117  *
1118  *                    MOVE INDEX REGISTER 1 OVER THE
1119  *                    SUBSCRIPT.
1120        MDX    1 3
1121  *
1122  *                    MOVE INDEX REGISTER 1 OVER THE
1123  *                    SYMBOL.
1124  MV01A MDX    1 2
1125  *
1126  *                    TEST TO SEE IF THERE IS A SYMBO
1127  *                    AT LOCATION SPECIFIED BY XR1.
1128        LD     1 0           LOAD SYMBOL AT XR1
1129        BSC  L MV01B,-       BR IF OPERATOR
1130  *
1131  *                    RETURN TO LINK 2
1132        MDX  L MOVE,2        INCREMENT LINK WORD BY 2
1133        BSC  I MOVE          RETURN
1134  *
1135  *                    TEST THE SEPERATOR TO SEE IF IT
1136  *                    IS A COMMA.
1137  MV01B S      3 H0012-X     TEST FOR COMMA
1138        BSC  L MV01C,Z       BR IF NOT A COMMA
1139  *
```

```
1140  *                     MOVE INDEX REGISTER 1 TO THE
1141  *                     NEXT NEST.
1142        MDX    1 2
1143  *
1144  *                     INCREMENT LINK WORD BY 1
1145  MV01D MDX  L  MOVE,1
1146  *
1147  *                     RETURN C  LINK WORD
1148  MV01E LD       MOVE       LOAD LINK WORD
1149        STO      * 1        INITIALIZE BR INSTRUCTION
1150        BSC  I   *-*        RETURN
1151  *
1152  *                     TEST THE NEXT STATEMENT TO SEE
1153  *                     IF IT IS AN EQUIVALENCE
1154  *                     STATEMENT.
1155  MV01C LD     1 1            LOAD NEXT ID WORD
1156        AND    3 HF800-X      SAVE TYPE BITS
1157        EOR    3 HA800-X      CHECK FOR EQUIVALENCE
1158        BSC  L MV01E,Z        BR IF NOT EQUIV. STATEMENT
1159  *
1160  *                     SKIP TO NEST IN NEXT STATEMENT.
1161        MDX    1 3
1162        MDX      MV01D        LOOP BACK
1163  ****************************************************
1164  *
1165  *               END OF THE MOVE ROUTINE
1166  *
1167  ****************************************************
1168        HDNG     MPX FORTRAN ** DATA ALLOCATION
1169  ****************************************************
1170  *
1171  * ROUTINE NAME/RMOVE
1172  *
1173  * FUNCTION     /REMOVE BEING ALLOCATED BITS FROM
1174  *               THE SYMBOL TABLE.
1175  *
1176  * ENTRY        /RMOVE
1177  *
1178  * INPUT        /THE SYMBOL TABLE IS INPUT.
1179  *
1180  * OUTPUT       /THE SYMBOL TABLE WILL BE FREE
1181  *               OF BEING ALLOCATED BITS.
1182  * EXTERNAL
1183  * REFERANCES   /REFERANCES TO  SOFST,H0300,H1800,
1184  *               D4,EOFST  ARE MADE.
1185  *
1186  * ERROR        /NONE
1187  *
1188  * NOTE         /CALLING SEQUENCE   BSI RMOVE
1189  *
1190  ****************************************************
1191  *
1192  *                     ENTRY-RMOVE
1193  RMOVE DC       *-*        LINK WORD
1194  *
1195  *                     POINT INDEX REGISTER 1 AT THE
1196  *                     START OF THE SYMBOL TABLE.
1197        LDX  I1 SOFST      LOAD XR1 WITH SOFST
1198  *
1199  *                     THE SYMBOL TO SEE IF THE BEING
```

```
1200  *                     ALLOCATED BITS ARE ON.
1201  RM01A LD     1 0          LOAD INDICATOR WORD
1202        AND    3 H0300-X    TEST BEING ALLOCATED BITS
1203        EOR    3 H0300-X
1204        BSC  L RM01B,Z      BR IF NOT BEING ALLOCATED
1205  *
1206  *                     CLEAR BEING ALLOCATED BITS FROM
1207  *                     SYMBOL TABLE.
1208        LD     1 0          LOAD INDICATOR WORD
1209        EOR    3 H0300-X    CLEAR BEING ALLOCATED BITS
1210        STO    1 0          STORE INDICATOR WORD
1211  *
1212  *                     TEST TO SEE IF SYMBOL IS
1213  *                     DIMENSIONED.
1214  RM01B LD     1 0          LOAD INDICATOR WORD
1215        AND    3 H1800-X    SAVE DIMENSIONED BITS
1216        BSC    Z            SKIP IF NOT DIMENSIONED
1217  *
1218  *                     MOVE INDEX REGISTER 1 OVER THE
1219  *                     DIMENSION INFORMATION.
1220        MDX    1 -3
1221  *
1222  *                     MOVE TO NEXT SYMBOL
1223        MDX    1 -3
1224  *
1225  *                     TEST TO SEE THE WHOLE SYMBOL
1226  *                     TABLE HAS BEEN CHECKED.
1227        STX  L1 D4          SAVE PRESENT POSITION
1228        LD     3 D4-X       LOAD PRESENT POSITION
1229        S    L EOFST        TEST FOR END OF TABLE
1230        BSC  I RMOVE,       BR IF END OF TABLE
1231        MDX    RM01A        LOOP BACK
1232  ********************************************************
1233  *
1234  *               END OF RMOVE ROUTINE
1235  *
1236  ********************************************************
1237        HDNG   MPX FORTRAN ** DATA ALLOCATION
1238  ********************************************************
1239  *
1240  * ROUTINE NAME/INSER
1241  *
1242  * FUNCTION    /INSERT THE ALLOCATION IN THE SYMBOL
1243  *              TABLE AND PRINT THE ALLOCATION.
1244  *
1245  * ENTRY       /INSER
1246  *
1247  * INPUT       /ALLOCATION IN A REGISTER. INDEX
1248  *              REGISTER 2 SET TO POINT AT SYMBOL
1249  *              TABLE.
1250  *
1251  * OUTPUT      /ALLOCTION IN SYMBOL TABLE AN SYMBOL
1252  *              READIED FOR PRINTING.
1253  *
1254  * EXTERNAL
1255  * REFERANCES  /SUBROUTINES  TOPAU,TOPA,TOPAD,PRINT
1256  *              BLKPA,NEWPG  ARE CALLED AND
1257  *              REFERANCES TO  SALOC,H0002,ALOCB,
1258  *              NAME,CCWD,H000C,HFFFF,H0040,H0060,
1259  *              H00D9,H00C9,H00C3,H005C,H007E,VSIZE
```

```
1260  *                ASIZE,H1800,PAP,RPAP,LNEND,H003C,
1261  *                WDCNT,PCHSK  ARE MADE.
1262  *
1263  * ERROR          /NONE
1264  *
1265  * NOTE           /CALLING SEQUENCE     LD  ALLOCATION
1266  *                                      BSI INSER
1267  *
1268  ********************************************************
1269  *
1270  *                      ENTRY-INSER
1271  INSER DC       *-*          LINK WORD
1272  *
1273  *                      SAVE THE ALLOCATION IN SALOC.
1274         STO    3 SALOC-X    SAVE ALLOCATION
1275  *
1276  *                      TEST TO SEE IF SYMBOL HAS BEEN
1277  *                      PREVIOUSLY ALLOCATED.
1278         LD     2 0          LOAD INDICATOR WORD
1279         AND    3 H0002-X    SAVE ALLOCATION BIT
1280         BSC  L IS01A, -     BR NOT PREVIOUSLY ALLOCATE
1281  *
1282  *                      TEST TO SEE THAT ALL ALLOCATION
1283  *                      ARE THE SAME.
1284         LD     2 1          LOAD PREVIOUS ALLOCATION
1285         S      3 SALOC-X    CHECK WITH THIS ONE
1286         BSC  L ERR67,Z      BR IF NOT THE SAME
1287         MDX    IS01B        BRANCH
1288  *
1289  *                      PLACE ALLOCATION BITS IN THE
1290  *                      INDICATOR WORD OF THE SYMBOL
1291  *                      TABLE.
1292  IS01A LD     2 0          LOAD INDICATOR WORD
1293         OR     3 ALOCB-X    OR IN ALLOCATION BITS
1294         STO    2 0          STORE INDICATOR WORD
1295  *
1296  *                      SAVE THE VARIABLE NAME
1297  IS01B LD     2 1          LOAD 1ST WORD OF NAME
1298         STO    3 NAME-X     SAVE 1ST WORD OF NAME
1299         LD     2 2          LOAD 2ND WORD OF NAME
1300         STO    3 NAME-X 1   SAVE 2ND WORD OF NAME
1301  *
1302  *                      INSERT THE ALLOCATION FOR THIS
1303  *                      VARIABLE IN THE 2ND WORD OF ITS
1304  *                      SYMBOL TABLE ENTRY.
1305         LD     3 SALOC-X    LOAD ALLOCATION
1306         STO    2 1          PLACE IN SYMBOL TABLE
1307  *
1308  *                      TEST TO SEE LIST SYMBOL TABLE
1309  *                      IS REQUESTED.
1310         LD   L CCWD         LOAD CONTROL CARD WORD
1311         SLA    12           PLACE LIST S.T. BIT IN SIG
1312         BSC  I INSER,-      BRANCH IF NO LISTIN REQUIR
1313  *
1314  *                      TEST TO AVOID LISTING G.T. AND
1315  *                      S.G.T. ENTRY IN SYMBOL TABLE.
1316         LD     2 0          LOAD ID WORD
1317         AND    3 H000C-X    SAVE S.G.T. AN G.T. BITS
1318         BSC  I INSER,Z      BRANCH IF G.T. OT S.G.T.
1319  *
```

```
1320  *                     TEST TO AVOID LISTING PREVIOUSL
1321  *                     LISTED VARIABLE WHEN ALLOCATING
1322  *                     EQUIVALENCE STATEMENTS.
1323          LD    2 2           LOAD 2ND WORD IN SYMBOL TA
1324          EOR   3 HFFFF-X     TEST FOR LISTED
1325          BSC   I  INSER, -   BRANCH IF LISTED
1326  *
1327  *                     INDICATE THAT THIS VARIABLE HAS
1328  *                     BEEN LISTED.
1329          LD    3 HFFFF-X     LOAD PRINTED INDICATOR
1330          STO   2 2           PLACE IN SYMBOL TABLE
1331  *
1332  *                     TEST TO SEE IF THE HEADING HAS
1333  *                     BEEN PRINTED.
1334          LD    3 HTEST-X     LOAD HEADER INDICATOR
1335          BSC   L  HD010, -   BRANCH TO PRINT HEADING
1336  *
1337  *                     PLACE THE NAME IN THE PRINTER
1338  *                     BUFFER.
1339  IS01C LD    3 NAME 1-X
1340          AND   3 H003F-X
1341          BSC   L  GO1,Z      GO TO LOAD A BLANK
1342          LDD   3 NAME-X
1343          RTE     6
1344          STD   3 NAME-X
1345          MDX     IS01C       LOOP
1346  GO1     LD    3 H0040-X     LOAD A BLANK
1347          BSI     TOPAU       CALL TOPAU
1348          LD    3 NAME-X      LOAD NAME
1349          SRA     8           SEPARATE 1ST CHARACTER
1350          BSI     TOPA        CALL TOPA
1351          LD    3 NAME-X      LOAD NAME
1352          SRA     2           SEPARATE 2ND CHARACTER
1353          BSI     TOPA        CALL TOPA
1354          LDD   3 NAME-X      LOAD NAME
1355          SLT     4           SEPARATE 3RD CHARACTER
1356          BSI     TOPA        CALL TOPA
1357          LD    3 NAME 1-X    LOAD NAME
1358          SRA     6           SEPARATE 4TH CHARACTER
1359          BSI     TOPA        CALL TOPA
1360          LD    3 NAME 1-X    LOAD 5TH CHARACTER
1361          BSI     TOPA        CALL TOPA
1362  *                       PLACE LEFT PAREN BETWEEN TH
1363  *                     NAME AND THE VARIABLE TYPES.
1364  *
1365          LD    3 H004D-X
1366          BSI     TOPAU       CALL TOPAU
1367  *
1368  *                     TEST TO SEE IF THE VARIABLE IS
1369  *                     REAL OR INTEGER AND PLACE THE
1370  *                     FIRST LETTER IN THE PRINT AREA.
1371          LD    2 0           LOAD ID WORD
1372          SLA     2           PLACE TYPE IN CARRY
1373          LD    3 H00D9-X     LOAD R
1374          BSC     C           SKIP IF REAL
1375          LD    3 H00C9-X     LOAD I
1376          BSI     TOPAU       CALL TOPAU
1377  *
1378  *                     TEST TO SEE IF THE VARIABLE IS
1379  *                     IN COMMON.
```

```
1380          LD    2 0           LOAD ID WORD
1381          SLA     2           PLACE COMMON IND. IN SIGN
1382          BSC   L IS01D, Z    BRANCH IF IN COMMON
1383   *
1384   *                      PLACE A BLANK IN THE PRINT AREA
1385          LD    3 H0040-X     LOAD BLANK
1386          BSI     TOPAU       CALL TOPAU
1387          MDX     IS01E       BRANCH
1388   *
1389   *                      TEST TO SEE IF THE VARIABLE IS
1390   *                      IN BALNK OR INSKEL COMMON AND
1391   *                      PLACE A C OR AN * IN THE PRINT
1392   *                      AREA.
1393   IS01D  SLA     14          PLACE INSKEL IND. IN CARRY
1394          LD    3 H00C3-X     LOAD C
1395          BSC     C           SKIP IF BLANK COMMON
1396          LD    3 H005C-X     LOAD *
1397          BSI     TOPAU       CALL TOPAU
1398   *
1399   *                          PLACE RIGHT PAREN AFTER TYP
1400   *
1401   IS01E  LD    3 H005D-X
1402          BSI     TOPAU       CALL TOPAU
1403   *
1404   *                      PLACE AN EQUAL SIGN IN THE
1405   *                      PRINT AREA.
1406          LD    3 H007E-X     LOAD EQUAL SIGN
1407          BSI     TOPAU       CALL TOPAU
1408   *
1409   *                      PLACE THE ADDRESS OF THE FIRST
1410   *                      ELEMENT IN THE VARIABLE.
1411          LD    3 SALOC-X     LOAD ADDRESS
1412          BSI     TOPAD       CALL TOPAD
1413   *
1414   *                      TEST TO SEE IF THE VARIABLE IS
1415   *                      AN ARRAY.
1416          LD    2 0           LOAD ID WORD
1417          AND   3 H1800-X     SEPEATE DIMENSION BITS
1418          BSC   L IS01F, -    BRANCH IF NOT DIMENSIONED
1419   *
1420   *                      CALL ARRL AND GET THE DATA
1421   *                      ABOUT THE VARIABLE. COMPUTE
1422   *                      THE SECOND ADDRESS OF THE
1423   *                      ARRAY.
1424          BSI   3 ARRL-X      CALCULATE ARRAY SIZE
1425          LD    3 VSIZE-X     LOAD VARIABLE SIZE
1426          S     3 ASIZE-X     SUBTRACT ARRAY SIZE
1427          A     3 SALOC-X     FIND LOW ADDRESS
1428          STO   3 SALOC-X     SAVE 2ND ADDRESS
1429   *
1430   *                      PLACE THE HYPHEN BETWEEN THE
1431   *                      FIRST AND SECOND ADDRESS.
1432          LD    3 H0060-X     LOAD HYPHEN
1433          BSI     TOPAU       CALL TOPAU
1434   *
1435   *                      PLACE THE SECOND ADDRESS IN THE
1436   *                      PRINT AREA.
1437          LD    3 SALOC-X     LOAD 2ND ADDRESS
1438          BSI     TOPAD       CALL TOPAD
1439          MDX     IS01G       BRANCH
```

```
1440  *
1441  *                    PLACE BLANKS FOR 2ND ADDRESS IN
1442  *                    THE PRINT AREA.
1443  IS01F LD     3 H0040-X   LOAD A BLANK
1444        BSI      TOPAU     CALL TOPAU
1445        LD     3 H0040-X   LOAD A BLANK
1446        BSI      TOPAU     CALL TOPAU
1447        LD     3 H0040-X   LOAD A BLANK
1448        BSI      TOPAU     CALL TOPAU
1449        LD     3 H0040-X   LOAD A BLANK
1450        BSI      TOPAU     CALL TOPAU
1451        LD     3 H0040-X   LOAD A BLANK
1452        BSI      TOPAU     CALL TOPAU
1453  *
1454  *                    TEST TO SEE IF WE ARE AT THE
1455  *                    END OF THE PRINT LINE.
1456  IS01G LD     3 PAP-X     LOAD PRINT AREA POINTER
1457        S      3 LNEND-X   SUBTRACT LINE END
1458        BSC  I   INSER,Z   BRANCH IF LINE NOT FULL
1459  *
1460  *                    RESET THE LINE POINTER AND
1461        LD     3 RPAP-X    LOAD PRINT AREA ADDRESS
1462        STO    3 PAP-X     INITIALIZE PRINT AREA PTR.
1463  *
1464  *                    PRINT THE LINE.
1465        BSI  L   PRINT     CALL PRINT
1466  *
1467  *                    BLANK THE PRINT AREA
1468        BSI      BLKPA
1469  *
1470  *                    TEST IF END OF PAGE AND HANDLE
1471  *                    HEADING.
1472        LD     3 H003C-X
1473        STO  L   WDCNT     RESTORE
1474        LD     3 RPAP-X
1475        STO  L   AREA      RESTORE
1476        BSC  I   INSER     BRANCH
1477  ***************************************************
1478  *
1479  *              END OF INSER ROUTINE
1480  *
1481  ***************************************************
1482        HDNG     MPX FORTRAN ** DATA ALLOCATION
1483  ***************************************************
1484  *
1485  * ROUTINE NAME/ TOPA
1486  *
1487  * FUNCTION     /CONVERT 6 BIT TRUNCATED EBCDIC
1488  *               AND CHECK FOR BLANKS, PLACING THE
1489  *               CHARACTER IN THE PRINT AREA.
1490  *
1491  * ENTRY        /TOPA
1492  *
1493  * INPUT        /THE A REGISTER CONTAINS IN THE RIGH
1494  *               8 BITS THE CHARACTER TO BE PLACED
1495  *               IN THE RIGHT AREA.
1496  *
1497  * OUTPUT       /CHARACTER IS PLACED.
1498  *
1499  * EXTERNAL
```

```
1500  * REFERANCES  /SUBROUTINES  TOPAU  ARE CALLED AND
1501  *              REFERANCES TO  H003F,H0040,H00C0
1502  *              ARE MADE.
1503  *
1504  * ERROR        /NONE
1505  *
1506  * NOTE         /CALLING SEQUENCE     LD  CHARACTER
1507  *                                    BSI TOPA
1508  *
1509  ************************************************
1510  *
1511  *                    ENTRY-TOPA
1512  TOPA  DC       ***        LINK WORD
1513  *
1514  *                    SAVE THE RIGHT 6 BITS IN THE
1515  *                    A REGISTER, TEST TO SEE IF THE
1516  *                    CHARACTER IS BLANK.
1517        AND    3 H003F-X    SAVE 6 BITS
1518        BSC  L   TA01A,Z    BRANCH IF NOT BLANK
1519  *
1520  *                    LOAD A BLANK TO BE PLACED IN
1521  *                    PRINT AREA.
1522        LD     3 H0040-X    LOAD BLANK
1523        MDX      TA01B      BRANCH
1524  *
1525  *                    MAKE THE CHARACTERS EBCDIC.
1526  TA01A OR     3 H00C0-X    OR IN THE HIGH BITS
1527  *
1528  *                    CALL TOPAU TO PLACE CHARACTER
1529  *                    IN PRINT AREA.
1530  TA01B BSI      TOPAU      CALL TOPAU
1531  *
1532  *                    RETURN
1533        BSC  I   TOPA       BRANCH
1534  ************************************************
1535  *
1536  *              END OF TOPA ROUTINE
1537  *
1538  ************************************************
1539        HDNG     MPX FORTRAN ** DATA ALLOCATION
1540  ************************************************
1541  *
1542  * ROUTINE NAME/TOPAU
1543  *
1544  * FUNCTION     /PLACE CHARACTER IN PRINT AREA.
1545  *
1546  * ENTRY        /TOPAU
1547  *
1548  * INPUT        /CHARACTER IN A REGISTER THAT IS TO
1549  *               BE PLACED IN THE PRINT BUFFER.
1550  *
1551  * OUTPUT       /CHARACTER IS PLACED.
1552  *
1553  * EXTERNAL
1554  * REFERANCES   /REFERANCES TO  PAP  AR MADE.
1555  *
1556  * ERROR        /NONE
1557  *
1558  * NOTE         /CALLING SEQUENCE     LD  CHARACTER
1559  *                                    BSI TOPAU
```

```
1560          STO  L3 BUF 120    STORE BLANK
1561    *
1562    ********************************************************
1563    *
1564    *                        ENTRY-TOPAU
1565    TOPAU DC        *-*           LINK WORD
1566    *
1567    *                        MOVE THE CHARACTER TO THE LEFT
1568    *                        SIDE OF THE A REGISTER.
1569          SLA       8             MOVE LEFT CHARACTER
1570    *
1571    *                        PLACE CHARACTER IN PRINT AREA.
1572          STO  I    PAP           PLACE CHARACTER
1573          MDX  L    PAP,1         INCREMENT POINTER
1574    *
1575    *                        RETURN
1576          BSC  I    TOPAU         BRANCH
1577    ********************************************************
1578    *
1579    *               END OF TOPAU ROUTINE
1580    *
1581    ********************************************************
1582          HDNG      MPX FORTRAN ** DATA ALLOCATION
1583    ********************************************************
1584    *
1585    * ROUTINE NAME/TOPAD
1586    *
1587    * FUNCTION      /PLACE HEXIDECIMAL ADDRESS IN THE
1588    *                PRINT AREA.
1589    *
1590    * ENTRY         /TOPAD
1591    *
1592    * INPUT         /HEXIDECIMAL WORD IS IN THE A
1593    *                REGISTER.
1594    *
1595    * OUTPUT        /THE FOUR HEXIDECIMAL CHARACTERS ARE
1596    *                PLACED IN THE PRINT AREA.
1597    *
1598    * EXTERNAL
1599    * REFERANCES    /SUBROUTINES  TOPAH  ARE CALLED AND
1600    *                REFERANCES TO  SAVAD,H4000  ARE
1601    *                MADE.
1602    *
1603    * ERROR         /NONE
1604    *
1605    * NOTE          /CALLING SEQUENCE     LD  ADDRESS
1606    *                                     BSI TOPAD
1607    *
1608    ********************************************************
1609    *
1610    *                        ENTRY-TOPAD
1611    TOPAD DC        *-*           LINK WORD
1612    *
1613    *                        SAVE ADDRESS TO BE PLACED IN
1614    *                        THE PRINT AREA.
1615          STO    3 SAVAD-X      SAVE ADDRESS
1616    *
1617    *                        TEST TO SEE IF ADDRESS IS FOR
1618    *                        INSKEL COMMON.
1619          LD     2 0            LOAD ID WORD
```

```
1620          SLA     15          PLACE INSKEL IND. IN SIGN
1621          BSC  L  TD01A,-     BRANCH IF NOT INSKEL COMMO
1622    *
1623    *                         ADJUST THE SAVED ADDRESS FOR
1624    *                         INSKEL COMMON.
1625          LD      3 SAVAD-X   LOAD SAVED ADDRESS
1626          S       3 H4000-X   ADJUST ADDRESS FOR INSKEL
1627          STO     3 SAVAD-X   STORE SAVED ADDRESS
1628    *
1629    *                         PLACE THE HEX ADDRESS IN THE
1630    *                         PRINT AREA.
1631    TD01A LD      3 SAVAD-X   LOAD SAVED ADDRESS
1632          SRA       12        ISOLATE 1ST HEX CHARACTER
1633          BSI       TOPAH     CALL TOPAH
1634          LD      3 SAVAD-X   LOAD SAVED ADDRESS
1635          SRA       8         ISOLATE 2ND HEX CHARACTER
1636          BSI       TOPAH     CALL TOPAH
1637          LD      3 SAVAD-X   LOAD SAVED ADDRESS
1638          SRA       4         ISOLATE 3RD HEX CHARACTER
1639          BSI       TOPAH     CALL TOPAH
1640          LD      3 SAVAD-X   LOAD 4TH HEX CHARACTER
1641          BSI       TOPAH     CALL TOPAH
1642    *
1643    *                         RETURN
1644          BSC   I   TOPAD     BRANCH
1645    ********************************************************
1646    *
1647    *                   END OF TOPAD ROUTINE
1648    *
1649    ********************************************************
1650          HDNG      MPX FORTRAN ** DATA ALLOCATION
1651    ********************************************************
1652    *
1653    * ROUTINE NAME/ TOPAH
1654    *
1655    * FUNCTION      /THIS ROUTINE CONVERTS A 4 BIT
1656    *                HEXIDECIMAL DIGIT TO AN EBCDIC
1657    *                CHARACTER AND PLACE IN PRINT AREA.
1658    *
1659    * ENTRY         /TOPAH
1660    *
1661    * INPUT         /HEXIDECIMAL DIGIT IN A REGISTER.
1662    *
1663    * OUTPUT        /EBCDIC CHARACTER IS PLACED IN THE
1664    *                PRINT AREA.
1665    *
1666    * EXTERNAL
1667    * REFERANCES    /SUBROUTINES  TOPA  ARE CALLED AND
1668    *                REFERANCES TO  H000F,H0009,H0039
1669    *                ARE MADE.
1670    *
1671    * ERROR         /NONE
1672    *
1673    * NOTE          /CALLING SEQUENCE    LD  DIGIT
1674    *                                    BSI TOPAH
1675    *
1676    ********************************************************
1677    *
1678    *                     ENTRY-TOPAH
1679    TOPAH DC        *-*       LINK WORD
```

```
1680  *
1681  *                    SEPERATE THE LAST HEX CHARACTER
1682  *                    IN THE A REGISTER.
1683        AND    3 H000F-X    HOLD RIGHT HEX CHARACTER
1684  *
1685  *                    TEST AND ADJUST 0-9 TO /30-/39
1686  *                    AND A-F TO /01-/06
1687        S      3 H0009-X    CHECK AGAINST 9
1688        BSC                 SKIP IF A-F
1689        A      3 H0039-X    ADJUST NUMBERS 0-9
1690  *
1691  *                    CALL TOPAU TO PLACE THE
1692  *                    CHARACTER IN THE PRINT AREA.
1693        BSI       TOPA      CALL TOPA
1694  *
1695  *                    RETURN
1696        BSC  I    TOPAH     BRANCH
1697  ************************************************
1698  *
1699  *              END OF TOPAH ROUTINE
1700  *
1701  ************************************************
1702        HDNG      MPX FORTRAN ** DATA ALLOCATION
1703  ************************************************
1704  *
1705  * ROUTINE NAME/BLKPA
1706  *
1707  * FUNCTION     /CLEAR THE PRINT AREA TO BLANKS.
1708  *
1709  * ENTRY        /BLKPA
1710  *
1711  * OUTPUT       /PRINT AREA IS CLEARED.
1712  *
1713  * EXTERNAL
1714  * REFERANCES   /REFERANCES TO  H0040,PAREA  ARE
1715  *               MADE.
1716  *
1717  * ERROR        /NONE
1718  *
1719  * NOTE         /CALLING SEQUENCE    BSI BLKPA
1720  *
1721  ************************************************
1722  *
1723  *                    ENTRY-BLKPA
1724  BLKPA DC           ISO1C     LINK WORD
1725  *
1726  *                    CLEAR THE PRINT AREA USING
1727  *                    INDEX REGISTER 3.
1728        LD     3 H4000-X
1729        LDX    3 -120        LOAD XR3 WITH -120
1730        STO  L3 BUF 120      BLANK PRINT AREA
1731        MDX    3 1           INCREMENT XR3 BY 1
1732        MDX      *-4         BRANCH
1733  *
1734  *                    RESTORE INDEX REGISTER 3 TO
1735  *                    POINT LITERAL POOL.
1736        LDX  L3 X            LOAD XR3
1737  *
1738  *                    RETURN
1739        BSC  I  BLKPA        BRANCH
```

```
1740          HDNG    MPX FORTRAN ** DATA ALLOCATION
1741  ********************************************************
1742  *
1743  * ROUTINE NAME/ALLOC
1744  *
1745  * FUNCTION     /ALLOCATE ALL COMMON AND NON-
1746  *               EQUIVALENCED VARIABLES.
1747  *
1748  * ENTRY        /ALLOC
1749  *
1750  * INPUT        /NO SPECIAL INPUT OTHER THAN THE
1751  *               SYMBOL TABLE,
1752  *
1753  * OUTPUT       /THE PART OF THE SYMBOL TABLE THIS
1754  *               IS CONTROLED BY THE PRESENT SETTING
1755  *               OF TRACK SWITCH IS ALLOCATED.
1756  *
1757  * EXTERNAL
1758  * REFERANCES   /SUBROUTINES  ARRL,INSER,VARFO  ARE
1759  *               CALLED AN REFERANCES TO  TRACK,BASE
1760  *               SOFST,CSIZE,LCOMN,CCWD,VSIZE,RELAD,
1761  *               D4,H1800,EOFST,H0028,H07D2,H0001,
1762  *               VARCR,CAC  ARE MADE.
1763  *
1764  * ERROR        /NONE
1765  *
1766  * NOTES        /CALLING SEQUENCE    BSI ALLOC
1767  *
1768  *               TRACK SWITCH CONTROLS THE ALLOCATIO
1769  *               OF PARTS OF THE SYMBOL TABLE AS
1770  *               FOLLOWS
1771  *                    PLUS -COMMON
1772  *                    ZERO -REAL VARIABLES
1773  *                    MINUS-INTEGER VARIABLES
1774  *
1775  ********************************************************
1776  *
1777  *                   ENTRY-ALLOC
1778  ALLOC DC       *-*       LINK WORD
1779  *
1780  *                   LOAD INDEX REGISTER 2 WITH THE
1781  *                   START OF THE SYMBOL TABLE.
1782        LDX   I2 SOFST     LOAD XR2
1783  *
1784  *                   TEST TRACK SWITCH TO DECIDE
1785  *                   THE ALLOCATION TO BE PREFORMED.
1786  AL01A LD     3 TRACK-X   LOAD TRACK SWITCH
1787        BSC   L  AL01B,-   BR TO ALLOC, REAL SYMBOLS
1788        BSC   L  AL01C,-Z  BR TO ALLOC, INTEGER SYMBO
1789  *
1790  *                   TEST TO SEE IF PRESENT SYMBOL I
1791  *                   A CONSTANT,
1792        LD     2 0         LOAD INDICATOR WORD
1793        BSC   L  AL01D, Z  BR IF CONSTANT
1794  *
1795  *                   REPOSITION SYMBOL TABLE NAME TO
1796  *                   READY FOR LATER PRINTING.
1797        SLT      16        CLEAR Q
1798        LD     2 2         LOAD RIGHT 2 CHARACTERS
1799        SRT      15        PLACE IN Q REGISTER
```

```
1800          LD     2 1           LOAD LEFT 3 CHARACTERS
1801          SLT      1           REMOVE LEADING BIT
1802          RTE      2
1803          STO    2 1           STORE REPOSITIONED NAME
1804          RTE      16
1805          STO    2 2
1806    *
1807    *                          TEST TO SEE IF VARIABLE IS IN
1808    *                          COMMON.
1809          LD     2 0           LOAD INDICATOR WORD
1810          SLA      2           PLACE COMMON IND. IN SIGN
1811          BSC  L   AL01E,-     BR IF NOT IN COMMON
1812    *
1813    *                          TEST TO SEE IF COMMON VARIABLE
1814    *                          IS IN INSKEL COMMON
1815          SLA      13          PLACE INSKEL IND. IN SIGN
1816          BSC  L   AL01F, Z    BR IF IN INSKEL COMMON
1817    *
1818    *                          PLACE BLANK COMMON ADDRESS IN
1819    *                          BASE.
1820          LD   L   CSIZE       LOAD BLANK COMMON ADDRESS
1821          MDX      AL01G       SAVE BASE AND BRANCH
1822    *
1823    *                          PLACE INSKEL COMMON ADDRESS IN
1824    *                          BASE.
1825    AL01F LD     3 LCS-X       LOAD INSKEL COMMON ADDRESS
1826    *
1827    *                          TEST THE ADDRESS IN BASE FOR
1828    *                          BEING ODD.
1829    AL01G STO    3 BASE-X      SAVE BASE
1830          BSC  L   AL01H,E     BR IF BASE IS ODD
1831    *
1832    *                          TEST THIS SYMBOL FOR BEING
1833    *                          REAL.
1834          LD     2 0           LOAD INDICATOR WOR
1835          SLA      1           PLACE REAL IND. IN SIGN
1836          BSC  L   AL01H, Z    BR IF INTEGER
1837    *
1838    *                          TEST TO SEE IF THIS PROGRAM IS
1839    *                          STANDARD PRECISION.
1840          LD   L   CCWD        LOAD CONTROL CARD WORD
1841          SLA      13          PLACE PRECISION IN SIGN
1842          BSC  L   AL01H, Z    BR IF EXTENDED PRECISION
1843    *
1844    *                          CORRECT BASE TO BE ODD FOR THE
1845    *                          RIGHT WORD OF STANDARD PRECISIO
1846    *                          REAL SYMBOLS.
1847          MDX  L   BASE,-1     DECRAMENT BASE BY 1
1848    *
1849    *                          CALL ARRL  TO DETERMINE THE SIZ
1850    *                          OF THE SYMBOL ELEMENTS.
1851    AL01H BSI    3 ARRL-X      CALL ARRL
1852    *
1853    *                          UPDATE THE ADDRESS OF COMMON
1854    *                          BEING HANDLED.
1855          LD     3 BASE-X      LOAD BASE ADDRESS OF SYMBO
1856          S      3 ASIZE-X     SUBTRACT ARRAY SIZE
1857          STO    3 BASE-X      SAVE ADJUSTED BASE
1858    *
1859    *                          TEST TO SEE IF THE COMMON SYMBO
```

```
1860  *                       ID IN INSKEL COMMON.
1861          LD    2 0           LOAD INDICATOR WORD
1862          BSC  L  AL01K,E     BR IF INSKEL COMMON
1863  *
1864  *                       PLACE COMMON BASE ADDRESS BACK
1865  *                       FOR BLANK COMMON.
1866          LD    3 BASE-X      LOAD ADJUSTED BASE ADDRESS
1867          STO  L  CSIZE       SAVE FOR BLANK COMMON
1868          MDX     AL01J       COMP ADDR AND INSERT
1869  *
1870  *                       PLACE COMMON BASE ADDRESS BACK
1871  *                       FOR INSKEL COMMON.
1872  AL01K LD    3 BASE-X      LOAD ADJUSTED BASE ADDRESS
1873          STO   3 LCS-X       SAVE FOR INSKEL COMMON
1874  *
1875  *                       COMPUTE THE ADDRESS OF THE
1876  *                       SYMBOL.
1877  AL01J LD    3 BASE-X      LOAD BASE ADDRESS OF SYMBO
1878          A     3 ASIZE-X     ADD ARRAY SIZE
1879          S     3 VSIZE-X     COMPUTE ADDRESS OF LEFT
1880          A     3 H0001-X           WORD
1881  *
1882  *                       CALL INSER TO PLACE ALLOCATION
1883  *                       IN THE SYMBOL TABLE.
1884  AL01L BSI  L  INSER         CALL INSER
1885  *
1886  *                       TEST TO SEE IF PRESENT SYMBOL
1887  *                       TABLE ENTRY IS DIMENSIONED.
1888  AL01D LD    2 0           LOAD INDICATOR WORD
1889          AND   3 H1800-X     SAVE DIMENSION IND. BITS
1890          BSC     Z           SKIP IF NOT DIMENSIONED
1891  *
1892  *                       MOVE SYMBOL TABLE POINTER OVER
1893  *                       THE DIMENSION INFORMATION.
1894          MDX   2 -3
1895  *
1896  *                       MOVE SYMBOL TABLE POINTER TO
1897  *                       NEXT SYMBOL.
1898          MDX   2 -3
1899  *
1900  *                       TEST TO SEE IF POINTER HAS
1901  *                       PASSED OVER THE WHOLE SYMBOL
1902  *                       TABLE.
1903          STX   L2 D4         SAVE PRESENT POINTER
1904          LD    3 D4-X        TEST FOR END OF TABLE
1905          S    L  EOFST
1906          BSC  L  AL01A,Z-    BR IF NOT END OF TABLE
1907  *
1908  *                       MOVE THE TRACK SWITCH TO ITS
1909  *                       NEXT POSITION.
1910          LD    3 TRACK-X     LOAD TRACK SWITCH
1911          A     3 H0001-X     MOVE TO NEXT POSITION
1912          STO   3 TRACK-X     STORE TRACK SWITCH
1913  *
1914  *                       RETURN
1915          BSC  I  ALLOC       RETURN
1916  *
1917  *                       TEST TO SEE IF THE PRESENT
1918  *                       SYMBOL IS A GENERATED TEMPORARY
1919  AL01E LD    2 0           LOAD INDICATOR WORD
```

```
1920          SLA     12          PLACE G.T. IND IN SIGN
1921          BSC  L  AL01D,-     BR IF NOT G.T.
1922   *
1923   *                          MAKE SURE ALL G.T.S ARE REAL
1924   *                          VARIABLES.
1925          LD      3 H0028-X   LOAD NEW INDICATOR WORD
1926          STO     2 0         SAVE FOR G.T. IND. WORD
1927          MDX       AL01D     CHK IF DIMENSIONED
1928   *
1929   *                          TEST TO SEE IF PRESENT SYMBOL
1930   *                          IS REAL.
1931   AL01B  LD      2 0         LOAD INDICATOR WORD
1932          SLA       1         PLACE REAL IND. IN SIGN
1933          BSC  L  AL01D, Z    BR IF NOT REAL SYMBOL
1934          BSI     3 EVENA-X    MAKE ADDRESS EVEN
1935          MDX       AL01M     CHK IF CONSTANT
1936   *
1937   *                          TEST TO SEE IF PRESENT SYMBOL
1938   *                          IS INTEGER.
1939   AL01C  LD      2 0         LOAD INDICATOR WORD
1940          SLA       1         PLACE INTEGER IND. IN SIGN
1941          BSC  L  AL01D,-     BR IF NOT INTEGER SYMBOL
1942   *
1943   *                          TEST TO SEE IF PRESENT SYMBOL
1944   *                          IS CONSTANT.
1945   AL01M  BSC  L  AL01N,C     BR IF CONSTANT
1946   *
1947   *                          TEST TO SEE IF THE SYMBOL
1948   *                          SHOULD BE ALLOCATED NOW.
1949          LD      2 0         LOAD INDICATOR WORD
1950          AND     3 H07F2-X   SAVE SOME IND. BITS
1951          EOR     3 H0020-X
1952          BSC  L  AL01D,Z     BR IF SHOULD NOT BE ALLOC.
1953   *
1954   *                          TEST TO SEE IF THE PRESENT
1955   *                          SYMBOL IS A SUBSCRIPT GENERATED
1956   *                          TEMPERARY.
1957          LD      2 0         LOAD INDICATOR WORD
1958          SLA       13        PLACE S.G.T. BIT IN SIGN
1959          BSC  L  AL01I,-     BR IF NOT S.G.T.
1960   *
1961   *                          ADJUST THE VARIABLE AREA SIZE
1962   *                          BY 1.
1963          LD      3 VARCR-X   LOAD VARIABLE ADDRESS
1964          A       3 H0001-X   INCRAMENT BY 1
1965          STO     3 VARCR-X   SAVE VARIABLE ADDRESS
1966          S       3 H0001-X   DECREMENT BY 1
1967          MDX       AL01L     GO INSERT
1968   *
1969   *                          CALL ARRL TO FIND THE AMOUNT OF
1970   *                          STORAGE REQUIRED FOR THIS
1971   *                          VARIABLE.
1972   AL01I  BSI     3 ARRL-X    CALL ARRL
1973   *
1974   *                          MODIFY THE SIZE OF VARIABLE
1975   *                          STORAGE AREA.
1976          LD      3 VARCR-X   LOAD VARIABLE ADDRESS
1977          A       3 ASIZE-X   ADJUST FOR SIZE OF SYMBOL
1978          STO     3 VARCR-X   SAVE VARIABLE ADDRESS
1979   *
```

```
1980  *                        COMPUTE ADDRESS OF SYMBOL
1981          S       3 VSIZE-X   REMOVE ELEMENT SIZE
1982          MDX       AL01L     GO INSERT
1983  *
1984  *                        CALL VARFO TO COMPUTE SIZE OF
1985  *                        CONSTANT.
1986  AL01N BSI     3 VARFO-X     CALL VARFO
1987  *
1988  *                        INCRAMENT THE SIZE OF THE
1989  *                        CONSTANT AREA.
1990          LD      3 CAC-X     LOAD CONSTANT AREA COUNT
1991          A       3 VSIZE-X   INCRAMENT BY CONSTANT SIZE
1992          STO     3 CAC-X     SAVE CONSTANT AREA COUNT
1993          MDX       AL01D     CHK IF DIMENSIONED
1994  ********************************************************
1995  *
1996  *                  END OF ALLOC ROUTINE
1997  *
1998  ********************************************************
1999          HDNG      MPX FORTRAN ** DATA ALLOCATION
2000  ********************************************************
2001  *
2002  * ROUTINE NAME/EQUIV
2003  *
2004  * FUNCTION      /THIS ROUTINE HANDLES THE EQUIVALENC
2005  *                STATEMENT FOR BOTH COMMON AND NON-
2006  *                COMMON VARIABLES.
2007  *
2008  * ENTRY         /EQUIV
2009  *
2010  * INPUT         /NO SPECIAL INPUT OTHER THAN THE
2011  *                STRING AND SYMBOL TABLE WITH ALL
2012  *                COMMON VARIABLES ALLOCATED.
2013  *
2014  * OUTPUT        /VARIABLES NOT IN EQUIVALENCE ARE
2015  *                NOT ALLOCATED.
2016  *
2017  * EXTERNAL
2018  * REFERENCES    /SUBROUTINES  INIT,INCR,EVENA,RMOVE,
2019  *                GETD4,RTN,HILO,INSER  ARE REQUIRED
2020  *                AND REFERENCES TO  SW1,SW2,SW3,SW4,
2021  *                LOW,HIGH,DEFIN,TAGLP,H0C00,CARCR,
2022  *                BASE,H0300,NRA,SNRA,D4,RELAD,H0020,
2023  *                H8000  ARE REFERANCED.
2024  *
2025  * ERROR         /ERROR C-65 IS DETECTED AND CONTROL
2026  *                IS TRANSFERED TO ERR65.
2027  *
2028  *
2029  ********************************************************
2030  *
2031  *                     ENTRY - EQUIV
2032  EQUIV DC        ***       LINK WORD
2033  *
2034  *                          CLEAR VARCR TO EQUIVALENCE
2035  *                                COMMON VARIABLES
2036  *
2037          SLA       16
2038          STO     3 VARCR-X
2039  *
```

```
2040  *                     INITIALIZE NEST LIMITS.
2041  EQ01D SLA     16      CLEAR A REGISTER
2042        STO   3 LOW-X   PLACE ZERO IN LOW
2043        STO   3 HIGH-X                HIGH
2044        STO   3 ODDSW-X               ODDSW
2045        STO   3 EVSW-X                EVSW
2046        STO   3 DEFIN-X             AND DEFIN
2047  *
2048  *                     SET SWITCH 4 TO NORMAL.
2049  EQ01H SLA     16      PLACE ZERO IN SW4
2050        STO   3 SW4-X
2051  *
2052  *                     CALL INIT TO FIND FIRST NON-
2053  *                     DELETED NEST IN EQUIVALENCE
2054  *                     STATEMENTS.
2055  EQ01I BSI     3 INIT-X  CALL INIT
2056        DC      EQ03B     BR IF END OF ALL EQUIV.
2057        DC      *         NORMAL RETURN
2058  *
2059  *                     SET SWITCH 2 TO NORMAL
2060        SLA     16      PLACE ZERO IN SW2
2061        STO   3 SW2-X
2062  *
2063  *                     TEST THE LEFT PARENTHESIS IN
2064  *                     NEST TO SEE IF VARIABLES IN
2065  *                     THIS NEST HAVE BEEN LOOKED AT.
2066  EQ01A LD      1 -1      LOAD LEFT PARENTHESIS
2067        S       3 TAGLP-X TEST FOR TAGED PARENTHESIS
2068        BSC   L EQ01G,Z   BR IF LEFT PARENTHESIS OK
2069  *
2070  *                     INCREMENT THE ITEM POINTER TO
2071  *                     THE NEXT SYMBOL.
2072        BSI     3 INCR-X  CALL INCR
2073        DC      EQ01F     BR IF END OF ALL EQUIV.
2074        DC      EQ01A     BR IF END OF NEST
2075        MDX     *-4       LOOP BACK
2076  *
2077  *                     TEST THE SYMBOL BEING LOOKED
2078  *                     AT FOR BEING IN COMMON.
2079  EQ01G LD      2 0       LOAD INDICATOR WORD
2080        SLA     2         PLACE COMMON IND. IN SIGN
2081        BSC   L EQ02B, Z  BR IF SYMBOL IN COMMON
2082  *
2083  *                     TEST THE SYMBOL BEING LOOKED
2084  *                     AT FOR BEING ALLOCATED.
2085        AND     3 H0C00-X
2086        EOR     3 H0C00-X
2087        BSC   L EQ02A, -  BR IF SYMBOL BEING ALOCATE
2088  *
2089  *                     INCREMENT THE ITEM POINTER TO
2090  *                     THE NEXT SYMBOL.
2091  EQ01J BSI     3 INCR-X  CALL INCR
2092        DC      EQ01F     BR IF END OF ALL EQUIV.
2093        DC      EQ01A     BRANCH IF END OF NEST
2094        MDX     EQ01G     LOOP BACK
2095  *
2096  *                     TAG SWITCH 4
2097  EQ01E STO     3 SW4-X   PLACE NONZERO IN SW4
2098        MDX     EQ01I     LOOP BACK
2099  *
```

```
2100  *                        TEST SWITCH 2
2101  EQ01F LD     3 SW2-X     LOAD SWITCH 2
2102        BSC  L  EQ01E,Z     BR IF SWITCH 2 TRANSFER
2103  *
2104  *                        TEST SWITCH 4
2105        LD     3 SW4-X     LOAD SWITCH 4
2106        BSC  L  EQ03B, -    BRANCH IF SWITCH 4 NORMAL
2107  *
2108  *                        CALL EVENA TO MAKE VARCR EVEN
2109        BSI    3 EVENA-X    CALL EVENA
2110  *
2111  *                          TEST TO SEE IF THE LOW ADDRESS
2112  *                          FOR THE NEST IS ODD.
2113  *
2114        LD     3 LOW-X      LOAD LOW ADDRESS
2115        BSC  L  EQ03G,E     BRANCH IF ODD
2116  *
2117  *                          TEST TO SEE IF THE ODD SWITCH IS
2118  *                          SET ON.
2119  *
2120        LD     3 ODDSW-X    LOAD ODD SWITCH
2121        BSC  L  EQ03H,Z     BRANCH ODD SWITCH ON
2122        MDX     EQ03I       BRANCH
2123  *
2124  *                          TEST TO SEE IF THE EVEN SWITCH IS
2125  *                          SET ON.
2126  *
2127  EQ03G LD     3 EVSW-X     LOAD EVEN SWITCH
2128        BSC  L  EQ03I, -    BRANCH IF EVEN SWITCH OFF
2129  *
2130  *                          SUBTRACT 1 FROM LOW
2131  *
2132  EQ03H LD     3 LOW-X      CALCULATE LOW LOW-1
2133        S      3 H0001-X
2134        STO    3 LOW-X
2135  *
2136  *                               ADJUST VARIABLE AREA TO INCLUDE
2137  *                               THE PRESENT NEST, SAVE IN BASE
2138  *                               THE ADDRESS USED FOR ALLOCATION
2139  *                               OF THE NEST.
2140  EQ03I LD     3 VARCR-X    LOAD VARIABLE ADDRESS
2141        S      3 LOW-X      SUBTRACT LOW ADDRESS
2142        STO    3 BASE-X     SAVE BASE FOR NEST
2143        A      3 HIGH-X     ADD HIGH ADDRESS
2144        A      3 H0001-X    INCREMENT
2145        STO    3 VARCR-X    STORE ADJUSTED ADDRESS
2146  *
2147  *                               CALL INIT TO FIND FIRST NON-
2148  *                               DELETED NEST IN EQUIVALENCE
2149  *                               STATEMENTS.
2150        BSI    3 INIT-X     CALL INIT
2151        DC       *          BR IF END OF ALL EQUIV.
2152        DC       *          NORMAL RETURN
2153  *
2154  *                               TEST SYMBOL BEING LOOKED AT
2155  *                               FOR BEING ALLOCATED.
2156  EQ01C LD     2 0          LOAD INDICATOR WORD
2157        AND    3 H0300-X
2158        EOR    3 H0300-X
2159        BSC  L  EQ03A, -    BR IF SYMBOL BEING ALOCATE
```

```
2160  *
2161  *                     INCREMENT THE ITEM POINTER TO
2162  *                     THE NEXT SYMBOL
2163          BSI    3 INCR-X     CALL INCR
2164          DC       EQ01B      BR IF END OF ALL EQUIV.
2165          DC       EQ01C      BR IF END OF NEST
2166          MDX      *-4        LOOP BACK
2167  *
2168  *                     REMOVE BEING ALLOCATED BITS
2169  *                     FROM SYMBOL TABLE.
2170  EQ01B BSI    L  RMOVE       CALL RMOVE
2171          MDX      EQ03B      GO TEST SW3
2172  *
2173  *                     TEST SWITCH 1
2174  EQ02A LD     3 SW1-X        LOAD SW1
2175          BSC    L EQ02C, -   BR IF SWITCH 1 NORMAL
2176  *
2177  *                     SAVE THE PRESENT POSITION AND
2178  *                     READY FOR A SCAN TO FIND THE
2179  *                     ALLOCATION FOR THIS VARIABLE.
2180          STX    1 EQ02G 1    SAVE XR1
2181          STX    2 EQ02G 3    SAVE XR2
2182          LD     3 NRA-X      SAVE NEST RETURN ADDRESS
2183          STO    3 SNRA-X
2184  *
2185  *                     CALL INIT TO FIND FIRST NON-
2186  *                     DELETED NEST IN EQUIVALENCE
2187  *                     STATEMENTS.
2188          BSI    3 INIT-X     CALL INIT
2189          DC       *          BR IF END OF ALL EQUIV.
2190          DC       *          NORMAL RETURN
2191  *
2192  *                     TEST THE LEFT PARENTHESIS IN
2193  *                     NEST TO SEE IF VARIABLES IN
2194  *                     THIS NEST HAVE BEEN LOOKED AT.
2195  EQ02D LD     1 -1           LOAD LEFT PARENTHESIS
2196          S      3 TAGLP-X    TEST FOR TAGED PARENTHESIS
2197          BSC    L EQ02E, -   BR IF PARENTHESIS TAGED
2198  *
2199  *                     INCREMENT THE ITEM POINTER TO
2200  *                     THE NEXT SYMBOL.
2201          BSI    3 INCR-X
2202          DC       EQ02D      BR IF END OF ALL EQUIF.
2203          DC       EQ02D      BR IF END OF NEST
2204          MDX      *-4        LOOP BACK
2205  *
2206  *                     TEST SYMBOL BEING LOOKED AT
2207  *                     TO SEE IF IT IS SAME AS SAVED
2208  *                     ONE.
2209  EQ02E LD     I  EQ02G 1     LOAD SAVE SYMBOL
2210          S      1 0          COMPARE PRESENT SYMBOL
2211          BSC    L EQ02F, -   BR IF SYMBOLS SAME
2212  *
2213  *                     INCREMENT THE ITEM POINTER TO
2214  *                     THE NEXT SYMBOL.
2215          BSI    3 INCR-X     CALL INCR
2216          DC       EQ02D      BR IF END OF ALL EQUIV.
2217          DC       EQ02D      BR IF END OF NEST
2218          MDX      EQ02E      LOOP BACK
2219  *
```

```
2220  *                     GET THE D4 FOR THE VARIABLE
2221  *                     THAT IS BEING LOOKED AT.
2222  EQ02F BSI    3 GETD4-X    CALL GETD4
2223  *
2224  *                     RETURN POINTER TO THE BEGINNING
2225  *                     OF THE PRESENT NEST.
2226        BSI    3 RTN-X      CALL RTN
2227  *
2228  *                     CALCULATE RELATIVE ADDRESS
2229  *                     FOR VARIABLE FOUND IN SAVED
2230  *                     NEST.
2231        LD     1 1          LOAD NEST ADDRESS
2232        A      3 D4-X       ADD D4 FOR VARIABLE IN NES
2233        STO    3 RELAD-X    SAVE RELATIVE ADDRESS
2234  *
2235  *                     RESTORE POINTER TO SAVED
2236  *                     POSITION IN OTHER NEST.
2237  EQ02G LDX  L1 ***       RESTORE XR1
2238        LDX  L2 ***       RESTORE XR2
2239        LD     3 SNRA-X     REPLACE NEST RETURN ADDRES
2240        STO    3 NRA-X
2241        MDX      EQ02H      GO GET D4
2242  *
2243  *                       IS IT THE CORRECT COMMON
2244  *
2245  EQ02B EOR    3 PASSW-X     TEST PASS SWITCH
2246        SLA      13          PLACE INSKEL IND IN SIGN
2247        BSC  L   EQ01J,Z     BR IF NOT CORRECT COMMON
2248  *
2249  *                     GET RELATIVE ADDRESS IN COMMON
2250  *                     FROM SYMBOL TABLE.
2251        LD     2 1          LD VARIABLE BASE ADDRESS
2252        STO    3 RELAD-X    PLACE IN RELAD.
2253        MDX      EQ02I      TAG SW1
2254  *
2255  *                     PLACE ZERO IN RELATIVE ADDRESS
2256  EQ02C SLA      16         STORE ZERO IN RELAD
2257        STO    3 RELAD-X
2258  *
2259  *                     TAG SWITCH 1
2260  EQ02I STX  L0 SW1         PLACE XR0 IN SW1
2261  *
2262  *                     GET THE D4 FOR THE VARIABLE
2263  *                     THAT IS BEING LOOKED AT.
2264  EQ02H BSI    3 GETD4-X    CALL GETD4
2265  *
2266  *                     REMOVE THE D4 FOR THE PRESENT
2267  *                     VARIABLE FROM THE RELATIVE
2268  *                     ADDRESS.
2269        LD     3 RELAD-X    LOAD RELATIVE ADDRESS
2270        S      3 D4-X       REMOVE D4
2271        STO    3 RELAD-X    STORE RELATIVE ADDRESS
2272  *
2273  *                     RETURN POINTER TO THE BEGINNING
2274  *                     OF THE PRESENT NEST.
2275        BSI    3 RTN-X      CALL RTN
2276  *
2277  *                     PLACE THE ADDRESS FOR THE NEST
2278  *                     IN THE NEST.
2279        LD     3 RELAD-X    LOAD RELATIVE ADDRESS
```

```
2280          STO   1 1          PLACE ADDRESS IN NEST
2281   *
2282   *                        TAG THE LEFT PARENTHESIS IN THE
2283   *                        NEST BEING WORKED ON.
2284          LD    3 TAGLP-X    LOAD TAGED LEFT PARENTHESI
2285          STO   1 -1         PLACE OVER OTHER PAREN.
2286   *
2287   *                        TAG SWITCH 2
2288          STO   3 SW2-X      PLACE NON-ZERO IN SW2
2289   *
2290   *                        SET THE BEING ALLOCATED BITS
2291   *                        IN THE SYMBOL TABLE INDICATOR
2292   *                        WORD AS  STATEMENT NUMBER AND
2293   *                        ARITHMETIC STATEMENT FUNCTION .
2294   EQ02J LD     3 ALOCB-X     SAVE INSKEL BIT
2295          AND   3 H0001-X
2296          OR    2 0          PLACE IN IND WD
2297          OR    3 H0300-X    ADD BEING ALLOCATED BITS
2298          STO   2 0          STORE INDICATOR WORD
2299   *
2300   *                        CALL HILO TO ADJUST THE HIGH
2301   *                        AND LOW LIMITS ON THIS NEST
2302   *                        AS WELL AS DECIDE IF THIS NEST
2303   *                        IS DEFINED.
2304          BSI   3 HILO-X     CALL HILO
2305   *
2306   *                        INCREMENT THE ITEM POINTER TO
2307   *                        THE NEXT SYMBOL.
2308          BSI   3 INCR-X     CALL INCR
2309          DC      EQ01E      BR IF END OF ALL EQUIV
2310          DC      EQ01A      BR IF END OF NEST
2311          MDX     EQ02J      LOOP BACK
2312   *
2313   *                        TEST TO SEE THAT THE NEST NOW
2314   *                        BEING ALLOCATED IS DEFINED.
2315   EQ03A LD     3 DEFIN-X    LOAD INDICATOR FOR NEST
2316          AND   3 H0020-X    SAVE ONLY DEFINED BIT
2317          BSC  L ERR65, -    BRANCH IF NOT DEFINED
2318   *
2319   *                        PLACE RELATIVE ADDRESS FOR NEST
2320   *                        IN RELAD.
2321          LD    1 1          LOAD NEST RELATIVE ADDRESS
2322          A     3 BASE-X
2323          STO   3 RELAD-X    STORE IN RELAD
2324   *
2325   *                        CHANGE THE LEFT PARENTHESIS FOR
2326   *                        THE NEST BEING ALLOCATED TO THE
2327   *                        CODE FOR DELETE.
2328          LD    3 H8000-X    LOAD DELETE CODE
2329          STO   1 -1         STORE OVER LEFT PARENTHESI
2330   *
2331   *                        CALL GETD4 TO GET D4 FOR THE
2332   *                        VARIABLE BEING LOOKED AT.
2333   EQ03D BSI    3 GETD4-X       CALL GETD4
2334   *
2335   *                        ALLOCATE VARIABLE AND CALL INSE
2336   *                        TO PLACE ALLOCATION IN SYMBOL
2337   *                        TABLE AND PRINT IF REQUIRED
2338          LD    3 RELAD-X    LOAD NEST ADDRESS
2339          A     3 D4-X       ADD VARIABLE S D4
```

```
2340          BSI  L  INSER      INSERT AND PRINT
2341   *
2342   *                        INCREMENT THE ITEM POINTER TO
2343   *                        THE NEXT SYMBOL.
2344          BSI   3 INCR-X        CALL INCR
2345          DC      EQ01B       BR IF END OF ALL EQUIV
2346          DC      EQ01C       BR IF END OF NEST
2347          MDX     EQ03D       LOOP BACK
2348   *
2349   *                        TEST SWITCH 3
2350   EQ03B LD    3 SW3-X       LOAD SWITCH 3
2351          BSC  L  EQ03E,Z     BR IF SWITCH 3 TRANSFER
2352   *
2353   *                        INITIALIZE THE VARIABLE AREA
2354          LD   L  DFCNT       LOAD DEFINE FILE COUNT
2355          STO   3 VARCR-X     STORE IN VARIABLE CORE
2356   *
2357   *                          TEST FOR INSKEL PASS
2358   *
2359          MDX  L  ALOCB,1      ALTER ALLOCATION BITS
2360          LD    3 PASSW-X      LOAD PASS SWITCH
2361          MDX  L  PASSW,4      ALTER PASS SWITCH
2362          BSC  L  EQ03C, -     BRANCH IF INSKEL PASS
2363   *
2364   *                        ALTER THE ALLOCATION BITS SO AS
2365   *                        NOT TO REFLECT COMMON.
2366          LD    3 H0022-X
2367          STO   3 ALOCB-X
2368   *
2369   *                        TAG SWITCH 3
2370          STO   3 SW3-X       SET SWITCH 3 NON-ZERO
2371   *
2372   *                        CALL INIT TO FIND THE FIRST
2373   *                        NON-DELETED NEST IN EQUIVALENCE
2374   *                        STATEMENTS
2375   EQ03E BSI    3 INIT-X      CALL INIT
2376          DC      EQ03F       BR IF END OF ALL EQUIV.
2377          DC      *           NORMAL RETURN
2378   *
2379   *                        SET THE BEING ALLOCATED BITS IN
2380   *                        THE SYMBOL TABLE FOR THIS
2381   *                        VARIABLE,  SEE EQ02J
2382          LD    2 0           LOAD INDICATOR WORD
2383          OR    3 H0300-X     ADD BEING ALLOCATED BITS
2384          STO   2 0           STORE IND. WORD
2385   *
2386   *                        NORMALIZE SWITCH 1
2387          SLA     16          PUT ZERO IN SW1
2388          STO   3 SW1-X
2389          BSC  L  EQ01D       BRANCH
2390   *
2391   *                          NORMALIZE SW1
2392   *
2393   EQ03C SLA     16          PUT ZERO IN SW1
2394          STO   3 SW1-X
2395          BSC  L  EQUIV 1     GO TO EQUIV RTN
2396   *
2397   *                        RETURN
2398   EQ03F BSC   I  EQUIV      RETURN
2399   ********************************************************
```

```
2400  *
2401  *              END OF EQUIV ROUTINE
2402  *
2403  ********************************************************
2404  *
2405  *
2406  *PRINT ROUTINE FOR PHASES 20,21,22,23,24,25
2407  *     INPUT IS IN EBCIDIC, 1-CH./WORD IN BITS 0-7
2408  *     OUTPUT USES LIO
2409  *
2410        ORG     OVERL+3*320+100
2411  AREA  DC      *-*             PRINT DATA ADDRESS
2412  PRINT DC      *-*             ENTRY
2413        STX   1 SVXR1           SAVE INDEX REGISTERS
2414        STX   2 SVXR2
2415        STX   3 SVXR3
2416        LDX   I2 AREA
2417        STX   2 CNV00+2         MESSAGE ADDRESS
2418        LDX   3 0
2419        LDX   I2 WDCNT          NUMBER OF WORDS
2420  CP01  BSI     CNV00           CONVERT TO ASCII
2421        SLA     8
2422        STO     CP02            SAVE LEFT CHAR.
2423        BSI     CNV00
2424        OR      CP02            PACK TWO CHAR.
2425        STO   L3 BUF            STORE PACKED CHAR.
2426        MDX   3 1
2427        MDX   2 -1
2428        MDX     CP01            CONVERSION LOOP
2429        BSI   L LIO             PRINT LINE
2430        DC      /2005
2431        DC      WDCNT
2432        DC      0
2433        BSI   L LIO             CHECK PRINT STSTUS
2434        DC      /F005
2435        LDX   L1 *-*            RESTORE INDEX REGS.
2436  SVXR1 EQU     *-1
2437        LDX   L2 *-*
2438  SVXR2 EQU     *-1
2439        LDX   L3 *-*
2440  SVXR3 EQU     *-1
2441        BSC   I PRINT           EXIT
2442  CP02  DC      *-*             SAVED CHARACTER
2443  *
2444  *     THIS ROUTINE CONVERYS THE EBCIDIC CHARACTER
2445  *     POINTED AT BY CNV00+2 TO ASCII AND
2446  *     EXITS WITH IT IN RA.
2447  *
2448  CNV00 DC      *-*             ENTRY
2449        LD    L *-*             GET DATA WORD
2450        SRA     8               MAKE CARACTER
2451        MDX   L CNV00+2,1       ADJ. WORD ADDRESS
2452        LDX   1 8
2453  CNV01 CMP   L1 CNVT1-1        COMPARE TO ALPHA
2454        MDX     CNV07           TO SWITCH ROUTINE
2455        MDX     *
2456        MDX   1 -1
2457        MDX     CNV01           COMPARE LOOP
2458        LDX   1 11
2459        STO     CNVT4           SAVE DATA
```

```
2460  CNV02 LD    L1 CNVT3          TEST FOR SPEC. CHAR.
2461        SRA      8
2462        S        CNVT4
2463        BSC  L   CNV04,*-       USE SPECIAL CHAR.
2464        MDX   1  -1
2465        MDX      CNV02          SP. CHAR. LOOP
2466  CNV03 LD       CNVT3          GENERATE SPACE
2467        MDX      CNV05
2468  CNV04 LD    L1 CNVT3          GET SPEC. CHAR.
2469  CNV05 AND      CNVC1          /FF  GET CHARACTER
2470  CNV06 BSC  I   CNV00          EXIT
2471  CNV07 BSC  I1  CNVT2-1        TO ROUTINE
2472  CNV08 S        CNVC2          /7 FOR J-R
2473        MDX      CNV06
2474  CNV09 S        CNVC3          /F FOR S-Z
2475        MDX      CNV06
2476  CNV10 S        CNVC4          /40 FOR 0-9
2477        MDX      CNV06
2478  CNVC1 DC       /FF
2479  CNVC2 DC       /7
2480  CNVC3 DC       /F
2481  CNVC4 DC       /40
2482  CNVT1 DC       /C0            A -1
2483        DC       /C9            I
2484        DC       /D0            J -1
2485        DC       /D9            R
2486        DC       /E1            S -1
2487        DC       /E9            Z
2488        DC       /EF            0 -1
2489        DC       /F9            9
2490  CNVT2 DC       CNV06          A-I
2491        DC       CNV03          SPACE
2492        DC       CNV08          J-R
2493        DC       CNV03          SPACE
2494        DC       CNV09          S-Z
2495        DC       CNV03          SPACE
2496        DC       CNV10          0-9
2497        DC       CNV03          SPACE
2498  CNVT3 DC       /40A0          SPACE
2499        DC       /4BAE          .
2500        DC       /4DA8          (
2501        DC       /4EAB          +
2502        DC       /5BA4          $
2503        DC       /5CAA          *
2504        DC       /5DA9          )
2505        DC       /60AD          -
2506        DC       /61AF          /
2507        DC       /6BAC          ,
2508        DC       /7DA7          =
2509        DC       /7EBD          ¤
2510  CNVT4 DC       *-*            TEMP. AREA
2511        BSS      OVERL-*+320*4  PHASE-20 PATCH AREA
```