HIGH SPEED TEXT SEARCH SYSTEM

HSTS SOFTWARE
LISTINGS

VOL. 3 OF 5

Query
Translator

STAT

HSTS MASTER COMPUTER SOFTWARE LISTINGS

SL120100

VOLUME 3 of 5

Prepared for:

Central Intelligence Agency
Washington, DC 20505

[redacted] R80-016 STAT

March 1980

STAT

QUERY
TRANSLATOR

Approved For Release 2005/08/22 : CIA-RDP85-00514R000200010001-4

```
                        ;
                        ; MACRO TO PRINT BUFFER
                        ; CREATES SPOOL FILE RECORDS
                        ;
                        .MACRO  PRT,START,END
                                MOV     R4,-(SP)
                                MOV     START,R4
                                MOV     R4,LOCAT
                                MOV     END,ENDLOC
                                JSR     PC,PRINT
                                MOV     (SP)+,R4
                        .ENDM
                        .MACRO  PRTH,START,END
                                MOV     R4,-(SP)
                                MOV     START,R4
                                MOV     R4,LOCAT
                                MOV     END,ENDLOC
                                JSR     PC,PRNTH
                                MOV     (SP)+,R4
                        .ENDM
                        ;
                        ; MACRO TO TEST CHARACTER SIGNIFICANCE
                        ; PARAMETERS ARE CONDITION,TRUE,AND FALSE
                        ;       CONDITION=THE TEST CONDITION
                        ;       TRUE      =PATH TO TAKE IF CONDITION TRUE
                        ;       FALSE     =PATH TO TAKE IF CONDITION FALSE
                        ; ANY PARAMETER CAN BE PROCEEDED BY "!" WHICH CAUSES THAT
                        ;  PARAMETER TO BE A SUBROUTINE CALL
                        ;
```

```
                                        .PSECT  NUMRNG
                                        .SBTTL  NUMRNG NUMBER RANGE CODE
                                ;
                                ;  THE FOLLOWING CODE IS USED TO PARSE AND CREATE THE EMATRIX
                                ;  ENTRIES FOR NUMERICAL RANGES.
                                ;
                                ;  DATA
                                ;
000000  000000                  LBEG:   .WORD   0       ;BEGINNING OF LOW RANGE NUMBER
000002  000000                  HBEG:   .WORD   0       ;BEGINNING OF HIGH RANGE NUMBER
000004  000000                  NMAX:   .WORD   0       ;MAXIMUM # TERMS
000006  000000                  EXPVPT: .WORD   0       ;EXPANSION VECTOR ENTRY POINTER
000010  000000                  EXPSPT: .WORD   0       ;POINTER TO EXPANSION CONTROL WORD
000012  000000                  LSZ::   .WORD   0       ;SIZE OF LOW RANGE NUMBER
000014  000000                  HSZ::   .WORD   0       ;SIZE OF HIGH RANGE NUMBER
000016  000000                  LSZD:   .WORD   0       ;SIZE OF LOW RANGE (PAST DECIMAL)
000020  000000                  HSZD:   .WORD   0       ;SIZE OF HIGH RANGE (PAST DECIMAL)
000022                          EXPSMR:                 ;PTR TO MERGE ENTRY FOR SINGLE ENTRY OF MR
000022  000000                  DECFLG: .WORD   0       ;DECIMAL FLAG
000024                          INCLUD::                        ;FLAG TO INCLUDE VECTORS IN INIT. VECTOR
000024  000000                  SGNIND: .WORD   0       ;SIGN INDICATOR
000026                          NUMVLU:                 ;NUMBER VALUE FOR COMPARISON "CHKRNG"
000026  000000                  LHCSZ:  .WORD   0       ;# COMMON CHAR'S TO LOW AND HIGH RANGE
000030  000000                  RNGCDE::.WORD   0       ;RANGE CODE BYTE (SEE FLAG DEF'S BELOW)
000032  000000                  NFPTR:  .WORD   0       ;POINTER TO NUMERICAL FLDC
000034  000000                  NF1PTR: .WORD   0       ;SAME AS ABOVE, USED WHEN MULTI-RANGE
000036  000000                  DPTR:   .WORD   0       ;POINTER TO DECIMAL POINT EMX ENTRY
000040  000000                  NVPTR:  .WORD   0       ;POINTER TO NUMERICAL VLDC
000042  000000                  ZPTR:   .WORD   0       ;POINTER TO ZVLDC
000044  000000                  MMM:    .WORD   0       ;CHAR POSITION NUMBER
000046  000000                  NNN:    .WORD   0       ;TERM NUMBER
000050  000000                  NOHTRM: .WORD   0       ;NO HIGH TERM FLAG
                                ;
                                ;
                                ;  RNGCDE FLAG VALUES
                                ;
          000200                MR2     =BIT7   ;MULTI-RANGE 2ND PASS (RESERVED FOR USE IN "NUMRNG")
          000100                MR      =BIT6   ;MULTIPLE RANGES (TWO RANGES)
          000040                CM      =BIT5   ;COMMON CHAR'S EXCLUDED
          000020                PD      =BIT4   ;PAST DECIMAL POINT (BECAUSE OF CM EXCLUSION)
          000010                DNUM    =BIT3   ;DECIMAL NUMBER
          000004                INCLD   =BIT2   ;FLAG TO INDICATE VECTORS SHOULD BE INCLUDED IN INIT VECTOR
          000002                USER3   =BIT1   ;FINAL PASS, R3 POINTS TO DESIRED MERGE VECTOR
          000001                MR3     =BIT0   ;SINGLE ENTRY RANGE OF A MULTI-RANGE (1ST PASS)
          000400                MR1     =BIT8   ;1ST PASS OF MULTI-RANGE
                                ;
000052      060     072         ZERO:   .ASCII  /0:/
```

```
                                        ;  SUBROUTINE TO PARSE NUMERICAL RANGE SPECIFICATION
                                        ;
                                        ;       INPUT                           OUTPUT
                                        ;  R0=  ..........SCRATCH.............
                                        ;  R1=  CHAR AFTER SEG-OP               CHAR AFTER RANGE PTR
                                        ;  R2=  #FLUTBL                          (SAME)
                                        ;  R3=  ..........SCRATCH.....................
                                        ;  R4=  FLU NODE ENTRY PTR              NEW PTR (AFTER RANGE)
                                        ;  R5=  ..........SCRATCH.....................
                                        ;
000054  005003                          PNBRNG::CLR     R3                      ;R3=CHAR COUNTER
000056  012705  177777                          MOV     #-1,R5                  ;R5=LOW RANGE CHAR COUNT
000062  005304                                  DEC     R4
000064                                          SAVE    R4                      ;R4=RANGE BEGIN PTR
000066  005067  177736                          CLR     RNGCDE                  ;INIT VARIABLES
000072  005067  177730                          CLR     LHCSZ
000076  005067  177714                          CLR     LSZD
000102  005067  177712                          CLR     HSZD
000106  005067  177710                          CLR     DECFLG
000112  005067  177706                          CLR     SGNIND
000116  012702  000000G                         MOV     #LEXTB3,R2              ;SWITCH TO NEW PARSING TABLE
000122  112724  000000G                         MOVB    #SEGOP,(R4)+            ;REPLACE WITH SEGMENT OPERATOR CODE
000126                                          SAVE    R4                      ;R4=LOW BEGIN PTR
000130  022767  000000G 000000G                 CMP     #NMRNGF,FLUTYP          ;1ST ENTRY FOR FLU?
000136  001415                                  BEQ     4$                      ;YES
000140  105764  177776                          TSTB    -2(R4)                  ;NO: IS PRIOR CHAR A DON'T CARE OR RANGE?
000144  100566                                  BMI     701$                    ;YES: ERROR
000146                                          SAVE    R1                      ;CHECK PRIOR ENTRY FOR NUMERIC
000150  010401                                  MOV     R4,R1
000152  162701  000002                          SUB     #2,R1
000156  004767  000000G                         JSR     PC,STEP
000162                                          CHECK   CSTNUM,701$             ;ILLEGAL IF NUMBER
000170                                          RESTOR  R1
000172  111124                          4$:     MOVB    (R1),(R4)+              ;XFER CHAR TO FLU NODE
000174  005203                                  INC     R3                      ;COUNT CHAR'S XFER'ED
000176  004767  000000G                         JSR     PC,STEP                 ;IDENTIFY CHAR
000202                                          CHECK   CSTMIN,,10$             ;CHECK IF MINUS
000210  005267  177610                          INC     SGNIND                  ;NOTE SIGN OCCURENCE
000214  005303                                  DEC     R3                      ;ADVANCE PAST SIGN
000216  005304                                  DEC     R4
000220  005705                                  TST     R5                      ;WHICH PASS?
000222  100415                                  BMI     5$                      ;1ST, OK
000224  022767  000001  177572                  CMP     #1,SGNIND               ;2ND, IF 1ST SIGN, SET TO -1
000232  001011                                  BNE     5$
000234  012767  177777  177562                  MOV     #-1,SGNIND
000242  000405                                  BR      5$
000244                                  10$:    CHECK   CSTZER,,11$             ;CHECK IF "0"
000252  005304                                  DEC     R4                      ;SUPPRESS LEADING ZEROS
000254  005303                                  DEC     R3
000256  111124                          5$:     MOVB    (R1),(R4)+              ;XFER CHAR TO FLU NODE
000260  005203                                  INC     R3                      ;COUNT CHAR'S XFER'ED
000262  004767  000000G                         JSR     PC,STEP                 ;IDENTIFY CHAR
000266  000766                                  BR      10$
000270                                  11$:    CHECK   CSTNUM,12$              ;IF NON-ZERO NUMBER CONTINUE
000276                                          CHECK   CSTDP,50$               ;IF DECIMAL POINT
000304                                          CHECK   CSTER,13$,700$          ;IF END, XFER AT LEAST 1 ZERO: ELSE, ERROR
000314  111124                          12$:    MOVB    (R1),(R4)+              ;XFER, COUNT, AND IDENTIFY
```

```
182 000166                      FSCFLU:
183 000166                              TFORM   TS              ;T(S0,CONSTANT)=TS
184 000170                              TFORM   TS              ;T(S1,CONSTANT)=TS
185 000172                              TFORM   TS              ;T(S2,CONSTANT)=TS
186 000174                              TFORM   TS              ;T(S3,CONSTANT)=TS
187 000176                              TFORM   TS              ;T(S4,CONSTANT)=TS
188 000200                              TFORM   TS              ;T(S5,CONSTANT)=TS
189 000202                              TFORM   TS              ;T(S6,CONSTANT)=TS
```

```
                                .SBTTL  MISCELLANEOUS STORAGE
                        ;
                                .EVEN
        000204          STATE:  .BLKW   1               ;CURRENT STATE
        000206          ARGTMP: .BLKW   2               ;TEMPS FOR TRANSFORMATION AND SIMPLIFICATION SUBROUTINES
                        ;
                                .IF     GT,OUTPUT
                        NMSGT0: .ASCII  /QTFORM NORMAL EXIT, QUERY ROOT IS:  /
                        NMSGP0: .ASCII  /      /
                        NMSGL0  =       .-NMSGT0
                        EMSGT:  .ASCII  /QTFORM QUERY ERROR, CODE IS:  /
                        EMSGP:  .ASCII  /      /
                        EMSGL   =       .-EMSGT
                        NMSGT1: .ASCII  /DUMP OF QUERY LOGIC NODES/
                        NMSGL1  =       .-NMSGT1
                                .ENDC
                        ;
                        ;       OPERATOR SIMPLIFICATION ATTRIBUTE TABLE
                        ;
        000212          TRTAB:
        000212    000           .BYTE   0
        000213    014           .BYTE   IDENT!NILPT                     ;SUBDOC
        000214    000           .BYTE   0
        000215    014           .BYTE   IDENT!NILPT                     ;NOT
        000216    147           .BYTE   IDEMP!TAUTL!IDENT!ABSAND!ABSPX0 ;OR
        000217    147           .BYTE   IDEMP!TAUTL!IDENT!ABSAND!ABSPX0 ;HIGH-OR
        000220    000           .BYTE   0
        000221    031           .BYTE   IDEMP!NILPT!ABSOR               ;AND
        000222    000           .BYTE   0
        000223    231           .BYTE   IDEMP!NILPT!ABSOR!ABSPX1        ;W-PROX
        000224    000           .BYTE   0
        000225    231           .BYTE   IDEMP!NILPT!ABSOR!ABSPX1        ;S-PROX
        000226    000           .BYTE   0
        000227    231           .BYTE   IDEMP!NILPT!ABSOR!ABSPX1        ;P-PROX
                        ;
```

```
                                        .SBTTL  QUERY TRANSFORMATION MAIN ROUTINE
                                        ;
                                        .PSECT  QTFORM
                                QTFORM::
                                        SAVE    R1,R3,R4,R5             ;SAVE CALLER'S REGISTERS
000010  005067  000204'                  CLR     STATE                   ;INITIALIZE CURRENT STATE
000014  012705  000000G                 MOV     #ARGSTK,TOP             ;INITIALIZE NODE STACK
000020  012701  000000G                 MOV     #SNODE,R1               ;INITIALIZE SNODE
000024  005021                          CLR     (R1)+
000026  005021                          CLR     (R1)+
000030  005021                          CLR     (R1)+
000032  005021                          CLR     (R1)+
                                        ;
                                        ;       PROCESS CURRENT NODE AND TRAVERSE LEFT SUBTREE
000034                          GROUP1:
000034  032702  100000                  BIT     #QNFLU,CUR              ;IS CURRENT NODE A FLU?
000040                                  BON     GROUP2                  ;TO RIGHT SUBTREE TRAVERSAL AND TREE BACKUP
000042  032762  100000  000000G         BIT     #QNTREE,QNATTR(CUR)     ;HAS SUBTREE ALREADY BEEN TRAVERSED?
000050                                  BON     GROUP2                  ;TO RIGHT SUBTREE TRAVERSAL AND TREE BACKUP
000052                                  CALL    SIMPFY                  ;SIMPLIFICATION SUBROUTINE
000056  103766                          BCS     GROUP1                  ;REPROCESS IF SIMPLIFICATION OCCURRED
                                        ;
000060  116201  000000G                 MOVB    QNOPCD(CUR),R1          ;GET CURRENT OPERATOR CODE
000064  116101  000022'                  MOVB    FSCOFV(R1),R1           ;GET OPERATOR'S FSC OFFSET
000070  062701  000040'                  ADD     #FSCTBL,R1              ;GET OPERATOR'S FSC BASE
000074  066701  000204'                  ADD     STATE,R1                ;GET FSC INDEX FOR CURRENT STATE AND OPERATOR
000100  066701  000204'                  ADD     STATE,R1                ;SECOND ADD GETS WORD INDEX
000104  011100                          MOV     @R1,R0                  ;GET FSC ENTRY
000106  100412                          BMI     10$                     ;BRANCH IF TRANSITION
000110  116762  000204'  000000G         MOVB    STATE,QNSTE(CUR)        ;SAVE CURRENT STATE
000116  010067  000204'                  MOV     R0,STATE                ;SET NEW STATE
000122                                  CALL    NODPUT                  ;PUSH CURRENT NODE ONTO STACK
000126  016202  000000G                 MOV     QNARG1(CUR),CUR         ;GET NODE'S LEFT SUBTREE
000132  000740                          BR      GROUP1                  ;PROCESS LEFT SUBTREE
000134                          10$:
000134  110000                          MOVB    R0,R0                   ;CLEAR HIGH ORDER BIT TO INDEX INTO
000136                                  CALL    @FSCTV(R0)              ;TRANSFORMATION TRANSFER VECTOR
000142  000734                          BR      GROUP1                  ;REPROCESS TRANSFORMED TREE
                                        ;
                                        ;       TRAVERSE RIGHT SUBTREE/BACKUP TREE
000144                          GROUP2:
000144  005715                          TST     @TOP                    ;TEST IF NODE STACK EMPTY
000146  001431                          BEQ     XTFORM                  ;IF YES, TERMINATE TRANSFORMATION PROCESSING
000150  032764  004000  000000G         BIT     #QNSTAK,QNATTR(STK)     ;HAS RIGHT SUBTREE JUST BEEN PROCESSED
000156                                  BON     10$                     ;IF YES, BRANCH AND BACK UP TREE
000160  052764  004000  000000G         BIS     #QNSTAK,QNATTR(STK)     ;NO, MARK RIGHT SUBTREE TRAVERSAL
000166  016402  000000G                 MOV     QNARG2(STK),CUR         ;GET NODE'S RIGHT SUBTREE
000172  001764                          BEQ     GROUP2                  ;UNARY OP - AUTOMATIC BACKUP
000174  000717                          BR      GROUP1                  ;PROCESS RIGHT SUBTREE
000176                          10$:
000176  052764  100000  000000G         BIS     #QNTREE,QNATTR(STK)     ;MARK NODE AS COMPLETED
000204  116467  000000G 000204'         MOVB    QNSTE(STK),STATE        ;RESTORE CURRENT STATE
000212                                  NODPOP  CUR                     ;POP OFF TOP OF NODE STACK
000216  000752                          BR      GROUP2                  ;CHECK PARENT NODE
```

```
                                    ;       UNDEFINED TRANSITION EXIT.
000220                              QTX:
000220  012767  000005  000000G      MOV     #0E,IHS,ERRCDE
                                    ;
                                    ;       ERROR EXIT FROM QUERY TRANSFORMATION ROUTINE.
000226                              XERROR:
                                            .IF     GT,OUTPUT
                                            CVTASC  ERRCDE,EMSGP
                                            PUT$S   #LIST,#EMSGT,#EMSGL
                                            PUT$S   #LIST,#NMSGT1,#NMSGL1
                                            PRT     #QNPOOL,QNDNXT
                                            .ENDC
000226  000167  000000G              JMP     ERROR
                                    ;
                                    ;       NORMAL EXIT FOR QUERY TRANSFORMATION ROUTINE.
000232                              XTFORM:
                                            .IF     GT,OUTPUT
                                            MOV     R2,ARGTMP
                                            CVTASC  ARGTMP,NMSGP0
                                            PUT$S   #LIST,#NMSGT0,#NMSGL0
                                            PUT$S   #LIST,#NMSGT1,#NMSGL1
                                            PRT     #QNPOOL,QNDNXT
                                            .ENDC
000232                                      RESTORE R1,R3,R4,R5
000242  000207                             RETURN
```

```
                                        .SBTTL  GET NEW QUERY LOGIC NODE SUBROUTINE
                                ;
                                ;       (R3) = BASE ADDRESS OF OBTAINED LOGIC NODE ON EXIT
                                ;       QNDNXT IS UPDATED TO POINT TO NEXT AVAILABLE NODE
                                ;       SUBROUTINE ZEROES OUT NEWLY ACQUIRED NODE
                                ;
                                GETQND:
000244  026727  000000G 000000G        CMP     QNDNXT,#QNTLST          ;ANY NODES LEFT IN POOL?
000252  103405                        BLO     1$                      ;BRANCH IF YES
                                ;
                                ;       ERROR - QUERY LOGIC NODE SPACE EXHAUSTED
                                ;
000254  012767  000146  000000G        MOV     #LNPOVF,ERRCDE          ;SET UP ERROR CODE
000262  000167  177740                 JMP     XERROR
                                ;
                                1$:
000266  062767  000000G 000000G        ADD     #QNDSIZ,QNDNXT          ;UPDATE NEXT AVAILABLE NODE POINTER
000274  016703  000000G                MOV     QNDNXT,R3               ;SET R3 PAST NODE FOR ZEROING SEQUENCE
                                ;
000300  005043                         CLR     -(R3)
000302  005043                         CLR     -(R3)
000304  005043                         CLR     -(R3)
000306  005043                         CLR     -(R3)
                                ;
                                ;       R3 NOW HAS BASE ADDRESS OF NEW LOGIC NODE
                                ;
000310  000207                         RETURN
```

```
                        .SBTTL  DISTRIBUTIVE TRANSFORMATION SCHEMATA
                        ;
                        ;                       LEFT DISTRIBUTIVE FORM
                        ;
                        ;                 ********************************************
                        ;                 *             *             *              *
                        ;       STK       *    OP-1     *    #CUR     *      X       *
                        ;                 *             *             *              *
                        ;                 ********************************************
                        ;                 *             *             *              *
                        ;       CUR       *    OP-2     *      Y      *      Z       *
                        ;                 *             *             *              *
                        ;                 ********************************************
                        ;
                        ;                         BECOMES
                        ;
                        ;                 ********************************************
                        ;                 *             *             *              *
                        ;       NC        *    OP-2     *    #NL      *    #NR       *
                        ;                 *             *             *              *
                        ;                 ********************************************
                        ;                 *             *             *              *
                        ;       NL        *    OP-1     *      Y      *      X       *
                        ;                 *             *             *              *
                        ;                 ********************************************
                        ;                 *             *             *              *
                        ;       NR        *    OP-2     *      Z      *      X       *
                        ;                 *             *             *              *
                        ;                 ********************************************
                        ;
                        ;                       RIGHT DISTRIBUTIVE FORM
                        ;
                        ;                 ********************************************
                        ;                 *             *             *              *
                        ;       STK       *    OP-1     *      X      *    #CUR      *
                        ;                 *             *             *              *
                        ;                 ********************************************
                        ;                 *             *             *              *
                        ;       CUR       *    OP-2     *      Y      *      Z       *
                        ;                 *             *             *              *
                        ;                 ********************************************
                        ;
                        ;                         BECOMES
                        ;
                        ;                 ********************************************
                        ;                 *             *             *              *
                        ;       NC        *    OP-2     *    #NL      *    #NR       *
                        ;                 *             *             *              *
                        ;                 ********************************************
                        ;                 *             *             *              *
                        ;       NL        *    OP-1     *      X      *      Y       *
                        ;                 *             *             *              *
                        ;                 ********************************************
                        ;                 *             *             *              *
                        ;       NR        *    OP-2     *      X      *      Z       *
                        ;                 *             *             *              *
                        ;                 ********************************************
```

```
                                QTD:
000312                                  BIT     #QNSTAK,QNATTR(STK)     ;TEST FOR LEFT OR RIGHT FORM
000312  032764  004000  000000G       BON     10$                     ;BRANCH IF RIGHT SUBTREE
000320
                                ;
                                ;       PUSH ARGUMENTS FOR LEFT FORM ONTO STACK
                                ;
000322                                  PUSH    <QNARG1(CUR),QNARG2(STK),QNARG2(CUR),QNARG2(STK)>
000342  000410                          BR      20$
                                ;
                                ;       PUSH ARGUMENTS FOR RIGHT FORM ONTO STACK
                                10$:
000344                                  PUSH    <QNARG1(STK),QNARG1(CUR),QNARG1(STK),QNARG2(CUR)>
                                ;
                                20$:
000364                                  MOV     #ARGTMP+4,R0            ;SET ARGUMENT FIELD LOOP LIMIT
                                21$:
000370                                  CALL    GETQND                  ;GET SUBTREE NODE
000374                                  POP     QNARG2(NEW)             ;SET RIGHT OPERAND
000400                                  POP     QNARG1(NEW)             ;SET LEFT OPERAND
000404                                  OPGEN   STK                     ;INTERCHANGE OPERATOR
000412                                  CALL    SETPOL                  ;SET POLARITY ATTRIBUTES
000416  010340                          MOV     NEW,-(R0)               ;SAVE NODE ADDRESS
000420  020027  000206'                  CMP     R0,#ARGTMP              ;BOTH OPERAND NODES?
000424  101361                          BHI     21$
                                ;
                                ;       GENERATE TRANSFORMATION ROOT NODE
                                ;
000426                                  CALL    GETQND                  ;GET NODE FOR ROOT
000432  012063  000000G               MOV     (R0)+,QNARG1(NEW)       ;SET LEFT OPERAND
000436  012063  000000G               MOV     (R0)+,QNARG2(NEW)       ;SET RIGHT OPERAND
000442                                  OPGEN   CUR                     ;INTERCHANGE OPERATOR
000450                                  CALL    SETPOL                  ;SET POLARITY ATTRIBUTES
000454  010302                          MOV     NEW,CUR                 ;ROOT NODE BECOMES CURRENT
000456  000167  002146                  JMP     RETRCT                  ;ADJUST STACK AND EXIT TRANSFORM
```

```
                        .SBTTL  NEGATION TRANSFORMATION SCHEMATA

                                LEFT NEGATION

                        ;*************************************************
                        ;*              *              *                *
                        ;*      OP      *      #CUR    *        X       *
                        ;*              *              *                *
                        ;*************************************************
                        ;*              *              *                *
                        ;*      -       *      Y       *                *
                        ;*              *              *                *
                        ;*************************************************

                                BECOMES

                        ;*************************************************
                        ;*              *              *                *
                        ;*      -       *      #NL     *                *
                        ;*              *              *                *
                        ;*************************************************
                        ;*              *              *                *
                        ;*      OP'     *      Y       *      #NR       *
                        ;*              *              *                *
                        ;*************************************************
                        ;*              *              *                *
                        ;*      -       *      X       *                *
                        ;*              *              *                *
                        ;*************************************************

                                RIGHT NEGATION

                        ;*************************************************
                        ;*              *              *                *
                        ;*      OP      *      X       *      #CUR      *
                        ;*              *              *                *
                        ;*************************************************
                        ;*              *              *                *
                        ;*      -       *      Y       *                *
                        ;*              *              *                *
                        ;*************************************************

                                BECOMES

                        ;*************************************************
                        ;*              *              *                *
                        ;*      -       *      #NL     *                *
                        ;*              *              *                *
                        ;*************************************************
                        ;*              *              *                *
                        ;*      OP'     *      #NR     *      Y         *
                        ;*              *              *                *
                        ;*************************************************
                        ;*              *              *                *
                        ;*      -       *      X       *                *
                        ;*              *              *                *
                        ;*************************************************
```

```
 487 000462                                 QTN:
 488 000462  032764  004000  000000G        BIT     #QNSTAK,QNATTR(STK)     ;TEST FOR LEFT OR RIGHT SUBTREE
 489 000470                                 BON     10$                     ;BRANCH IF RIGHT SUBTREE
 490                                        ;
 491 000472                                 PUSH    QNARG2(STK)             ;ARGUMENT FOR LEFT FORM
 492 000476  000402                         BR      20$
 493 000500                                 10$:
 494 000500                                 PUSH    QNARG1(STK)             ;ARGUMENT FOR RIGHT FORM
 495 000504                                 20$:
 496 000504                                 CALL    GETQND                  ;NEW RIGHT SUBTREE
 497 000510                                 POP     QNARG1(NEW)             ;OPERAND
 498 000514                                 OPGEN   CUR                     ;NOT OPERATOR
 499 000522                                 CALL    SETPOL                  ;SET POLARITY ATTRIBUTES
 500 000526                                 PUSH    NEW                     ;SAVE AS ARGUMENT FOR LEFT SUBTREE
 501                                        ;
 502 000530                                 CALL    GETQND                  ;NEW LEFT SUBTREE
 503 000534                                 POP     QNARG1(NEW)             ;LEFT ARGUMENT IS #NR
 504 000540  016263  000000G 000000G        MOV     QNARG1(CUR),QNARG2(NEW) ;RIGHT ARGUMENT
 505 000546                                 OPCMP   STK                     ;GET COMPLEMENTARY OPERATOR
 506 000564                                 CALL    SETPOL                  ;SET POLARITY ATTRIBUTES
 507 000570                                 PUSH    NEW                     ;SAVE AS ARGUMENT FOR ROOT
 508                                        ;
 509 000572                                 CALL    GETQND                  ;GET NEW ROOT NODE (NC)
 510 000576                                 POP     QNARG1(NEW)             ;GET OPERAND #NL
 511 000602                                 OPGEN   CUR                     ;NOT OPERATOR
 512 000610                                 CALL    SETPOL                  ;SET POLARITY ATTRIBUTES
 513                                        ;
 514 000614  010302                         MOV     NEW,CUR                 ;ROOT NODE BECOMES CURRENT
 515 000616  000167  002006                 JMP     RETRCT                  ;ADJUST STACK AND EXIT TRANSFORM
```

```
                                        .SBTTL  COMPLEMENTATION TRANSFORMATION SCHEMA
                                ;
                                ;               ***************************************************
                                ;               *               *               *                 *
                                ;       STK     *       -       *       #CUR    *                 *
                                ;               *               *               *                 *
                                ;               ***************************************************
                                ;               *               *               *                 *
                                ;       CUR     *       OP      *       X       *       Y         *
                                ;               *               *               *                 *
                                ;               ***************************************************
                                ;
                                ;                       BECOMES
                                ;
                                ;               ***************************************************
                                ;               *               *               *                 *
                                ;       NC      *       OP'     *       #NL     *       #NR       *
                                ;               *               *               *                 *
                                ;               ***************************************************
                                ;               *               *               *                 *
                                ;       NL      *       -       *       X       *                 *
                                ;               *               *               *                 *
                                ;               ***************************************************
                                ;               *               *               *                 *
                                ;       NR      *       -       *       Y       *                 *
                                ;               *               *               *                 *
                                ;               ***************************************************
                                ;
000622                          QTC:
000622                                  PUSH    <QNARG1(CUR),QNARG2(CUR)> ;SET UP ARGUMENTS
000632  012700  000212'                  MOV     #ARGTMP+4,R0              ;SET UP SUBTREE GENERATION LOOP
000636                          10$:
000636                                  CALL    GETQND                    ;GET NODE FOR SUBTREE
000642                                  POP     QNARG1(NEW)               ;SET OPERAND
000646                                  OPGEN   STK                       ;SET NOT OPERATOR
000654                                  CALL    SETPOL                    ;SET POLARITY ATTRIBUTES
000660  010340                          MOV     NEW,-(R0)                 ;SAVE NODE ADDRESS - ROOT NODE ARGUMENT
000662  020027  000206'                 CMP     R0,#ARGTMP                ;LOOP END TEST
000666  101363                          BHI     10$                       ;FIRST RIGHT SUBTREE - THEN LEFT
                                ;
000670                                  CALL    GETQND                    ;GET NEW ROOT NODE
000674  012063  000000G                 MOV     (R0)+,QNARG1(NEW)         ;LINK LEFT ARGUMENT: @NL
000700  012063  000000G                 MOV     (R0)+,QNARG2(NEW)         ;LINK RIGHT ARGUMENT: @NR
000704                                  OPCMP   CUR                       ;GENERATE COMPLEMENTARY OPERATOR
000722  052763  070000  000000G         BIS     #POLRTY,QNATTR(NEW)       ;SET POLARITY ATTRIBUTES
000730  010302                          MOV     NEW,CUR                   ;ROOT NODE BECOMES CURRENT
000732  000167  001672                  JMP     RETRCT                    ;ADJUST STACK AND EXIT TRANSFORM
```

```
565                     .SBTTL  PROXIMITY NEGATION SCHEMATA
566                     ;
567                     ;               ***************************************************
568                     ;               *               *               *                 *
569                     ;       STK     *       PROXOP  *       #CUR    *       Y       *
570                     ;               *               *               *                 *
571                     ;               ***************************************************
572                     ;               *               *               *                 *
573                     ;       CUR     *       -       *       X       *                 *
574                     ;               *               *               *                 *
575                     ;               ***************************************************
576                     ;                       OR
577                     ;               ***************************************************
578                     ;               *               *               *                 *
579                     ;       STK     *       PROXOP  *       Y       *       #CUR      *
580                     ;               *               *               *                 *
581                     ;               ***************************************************
582                     ;               *               *               *                 *
583                     ;       CUR     *       -       *       X       *                 *
584                     ;               *               *               *                 *
585                     ;               ***************************************************
586                     ;
587                     ;                       BECOMES
588                     ;
589                     ;               ***************************************************
590                     ;               *               *               *                 *
591                     ;       NC      *       AND     *       Y       *       #NR       *
592                     ;               *               *               *                 *
593                     ;               ***************************************************
594                     ;               *               *               *                 *
595                     ;       NR      *       -       *       X       *                 *
596                     ;               *               *               *                 *
597                     ;               ***************************************************
598                     ;               *               *               *                 *
599                     ;       NR      *       PROXOP  *       Y       *       X         *
600                     ;               *               *               *                 *
601                     ;               ***************************************************
```

```
000736                            QTPN:
000736  032764  004000  000000G           BIT     #QNSTAK,QNATTR(STK)      ;TEST FOR LEFT OR RIGHT SUBTREE
000744                                    BON     10$                      ;BRANCH IF RIGHT SUBTREE
                                  ;
                                  ;       ARGUMENT LIST FOR LEFT FORM
                                  ;
000746                                    PUSH    <QNARG2(STK),QNARG1(CUR),QNARG2(STK)>
000762  000406                            BR      20$
                                  ;
                                  ;       ARGUMENT LIST FOR RIGHT FORM
000764                            10$:
000764                                    PUSH    <QNARG1(STK),QNARG1(STK),QNARG1(CUR)>
001000                            20$:
001000                                    CALL    GETQND                   ;NEW LEFT SUBTREE
001004                                    POP     QNARG2(NEW)              ;RIGHT ARGUMENT
001010                                    POP     QNARG1(NEW)              ;LEFT ARGUMENT
001014                                    OPGEN   STK                      ;PROXIMITY OPERATOR
001022                                    CALL    SETPOL                   ;SET POLARITY ATTRIBUTES
001026                                    PUSH    NEW                      ;SAVE AS ARGUMENT FOR LEFT SUBTREE
                                  ;
001030                                    CALL    GETQND                   ;NEW RIGHT SUBTREE
001034                                    POP     QNARG1(NEW)              ;ARGUMENT IS #NL
001040                                    OPGEN   CUR                      ;NOT OPERATOR
001046                                    CALL    SETPOL                   ;SET POLARITY ATTRIBUTES
001052                                    PUSH    NEW                      ;SAVE AS ARGUMENT FOR ROOT
                                  ;
001054                                    CALL    GETQND                   ;GET NEW ROOT NODE (NC)
001060                                    POP     QNARG2(NEW)              ;RIGHT ARGUMENT IS #NR
001064                                    POP     QNARG1(NEW)              ;LEFT ARGUMENT
001070  012763  000000G 000000G           MOV     #TOKAND,QNOPCD(NEW)      ;AND OPERATOR
001076                                    CALL    SETPOL                   ;SET POLARITY ATTRIBUTES
                                  ;
001102  010302                            MOV     NEW,CUR                  ;ROOT NODE BECOMES CURRENT
001104  000167  001520                    JMP     RETRCT                   ;ADJUST STACK AND EXIT TRANSFORM
```

```
                        .SBTTL  ASSOCIATIVE TRANSFORMATION SCHEMATA
                        ;
                        ;                       LEFT ASSOCIATIVE FORM
                        ;
                        ;                  ***************************************************
                        ;                  *            *            *            *            *
                        ;        STK       *     OR     *     #CUR   *     X      *            *
                        ;                  *            *            *            *            *
                        ;                  ***************************************************
                        ;                  *            *            *            *            *
                        ;        CUR       *  HIGH-OR   *     #NAND  *     Y      *            *
                        ;                  *            *            *            *            *
                        ;                  ***************************************************
                        ;                               OR
                        ;                  ***************************************************
                        ;                  *            *            *            *            *
                        ;        STK       *     OR     *     #CUR   *     X      *            *
                        ;                  *            *            *            *            *
                        ;                  ***************************************************
                        ;                  *            *            *            *            *
                        ;        CUR       *  HIGH-OR   *     Y      *     #NAND  *            *
                        ;                  *            *            *            *            *
                        ;                  ***************************************************
                        ;
                        ;                       RIGHT DISTRIBUTIVE FORM
                        ;
                        ;                  ***************************************************
                        ;                  *            *            *            *            *
                        ;        STK       *     OR     *     X      *     #CUR   *            *
                        ;                  *            *            *            *            *
                        ;                  ***************************************************
                        ;                  *            *            *            *            *
                        ;        CUR       *  HIGH-OR   *     #NAND  *     Y      *            *
                        ;                  *            *            *            *            *
                        ;                  ***************************************************
                        ;                               OR
                        ;                  ***************************************************
                        ;                  *            *            *            *            *
                        ;        STK       *     OR     *     X      *     #CUR   *            *
                        ;                  *            *            *            *            *
                        ;                  ***************************************************
                        ;                  *            *            *            *            *
                        ;        CUR       *  HIGH-OR   *     Y      *     #NAND  *            *
                        ;                  *            *            *            *            *
                        ;                  ***************************************************
                        ;
                        ;                            BECOME
                        ;
                        ;                  ***************************************************
                        ;                  *            *            *            *            *
                        ;        NC        *  HIGH-OR   *     #NAND  *     #NR    *            *
                        ;                  *            *            *            *            *
                        ;                  ***************************************************
                        ;                  *            *            *            *            *
                        ;        NR        *     OR     *     X      *     Y      *            *
                        ;                  *            *            *            *            *
                        ;                  ***************************************************
```

```
696 001110                              QTA:
697 001110  032762  040000  000000G      BIT     #QNLPOL,QNATTR(CUR)     ;TEST WHICH ARGUMENT IS SUBORDINATE AND
698 001116                               BOFF    10$                     ;BRANCH IF RIGHT
699                                     ;
700 001120                               PUSH    <QNARG1(CUR),QNARG2(CUR)>
701 001130  000404                       BR      20$
702 001132                              10$:
703 001132                               PUSH    <QNARG2(CUR),QNARG1(CUR)>
704 001142                              20$:
705 001142  032764  004000  000000G      BIT     #QNSTAK,QNATTR(STK)     ;TEST FOR LEFT OR RIGHT SUBTREE
706 001150                               BON     30$                     ;BRANCH IF RIGHT SUBTREE
707                                     ;
708 001152                               PUSH    QNARG2(STK)
709 001156  000402                       BR      40$
710 001160                              30$:
711 001160                               PUSH    QNARG1(STK)
712 001164                              40$:
713 001164                               CALL    GETQND                  ;GET NEW SUBORDINATE NODE
714 001170                               POP     QNARG2(NEW)             ;RIGHT ARGUMENT
715 001174                               POP     QNARG1(NEW)             ;LEFT ARGUMENT
716 001200                               OPGEN   STK                     ;LOW-LEVEL OR OPERATOR
717 001206                               CALL    SETPOL                  ;POSSIBLE SUBELEM ATTRIBUTE INHERITED
718 001212                               PUSH    NEW                     ;SAVE AS RIGHT ARGUMENT FOR ROOT NODE
719                                     ;
720 001214                               CALL    GETQND                  ;NEW ROOT NODE
721 001220                               POP     QNARG2(NEW)             ;RIGHT ARGUMENT IS NEW SUBORDINATE NODE
722 001224                               POP     QNARG1(NEW)             ;LEFT ARGUMENT
723 001230                               OPGEN   CUR                     ;OPERATOR IS HIGH-OR
724 001236                               CALL    SETPOL                  ;SET POLARITY ATTRIBUTES
725                                     ;
726 001242  010302                       MOV     NEW,CUR                 ;ROOT NODE BECOMES CURRENT
727 001244  000167  001360               JMP     RETRCT                  ;ADJUST STACK AND EXIT TRANSFORM
```

```
                                .SBTTL  HIGH-OPERATOR TRANSFORMATION
                                ;
                                ;               ***************************************************
                                ;               *               *               *               *
                                ;       STK     *       OP      *       #CUR    *       X       *
                                ;               *               *               *               *
                                ;               ***************************************************
                                ;               *               *               *               *
                                ;       CUR     *       AND     *       Y       *       Z       *
                                ;               *               *               *               *
                                ;               ***************************************************
                                ;
                                ;                       BECOMES
                                ;
                                ;               ***************************************************
                                ;               *               *               *               *
                                ;       NC      *       HIGH-OP *       #CUR    *       X       *
                                ;               *               *               *               *
                                ;               ***************************************************
                                ;
                                QTH:
001250  010403                         MOV     STK,NEW                 ;REUSE STACK NODE
001252  052763  000001  000000G         BIS     #QNHLOP,QNOPCD(NEW)     ;MAKE OPERATOR HIGH-LEVEL
001260  010302                         MOV     NEW,CUR                 ;ROOT NODE BECOMES CURRENT
001262  000167  001342                 JMP     RETRCT                  ;ADJUST STACK AND EXIT TRANSFORM
```

```
                                              .SBTTL  DOUBLE NEGATION TRANSFORMATION SCHEMA
                                      ;
                                      ;                 ************************************************
                                      ;                 *              *              *                *
                                      ;       STK       *      -       *     #CUR     *                *
                                      ;                 *              *              *                *
                                      ;                 ************************************************
                                      ;                 *              *              *                *
                                      ;       CUR       *      -       *      X       *                *
                                      ;                 *              *              *                *
                                      ;                 ************************************************
                                      ;
                                      ;                        BECOMES
                                      ;
                                      ;                 ************************************************
                                      ;                 *                                              *
                                      ;                 *      CUR     <---     X                      *
                                      ;                 *                                              *
                                      ;                 ************************************************
                                      ;
    001266                            QTDN:
    001266  012703  000000G                   MOV     #SNODE,NEW          ;SET CONSTANT ARGUMENT NODE
    001272  016263  000000G 000000G           MOV     QNARG1(CUR),QNARG1(NEW) ;SET ITS ARGUMENT
    001300                                    CALL    SETPOL              ;SET ITS ATTRIBUTES
    001304  010302                            MOV     NEW,CUR             ;MAKE CONSTANT ARGUMENT NODE CURRENT
    001306  000167  001316                    JMP     RETRCT              ;ADJUST STACK AND EXIT TRANSFORM
                                      ;
```

```
                                        .SBTTL  SIMPLIFICATION  ROUTINE
                                ;
                                ;       REGISTER USAGE
                                ;
                                ;       (R0) - ARGUMENT BEING TESTED
                                ;       (R1) - SCRATCH
                                ;
001312                          SIMPFY:
001312  016200  000000G                 MOV     QNARG1(CUR),R0          ;SET LEFT ARGUMENT AS THE ONE TO BE CHECKED
001316  111201                          MOVB    @CUR,R1                 ;GET CURRENT OPERATOR
001320  132761  000020  000000G         BITB    #SYNOP2,SYNTAB(R1)      ;IS CURRENT OPERATOR BINARY
001326                                  BOFF    30$                     ;IF NOT, CHECK LEFT ARGUMENT ONLY
                                ;
                                ;       IDEMPOTENT TEST
                                ;
001330  026200  000000G                 CMP     QNARG2(CUR),R0          ;DOES ARG1 = ARG2?
001334  001012                          BNE     10$                     ;BRANCH IF NOT
001336  132761  000001  000212'        BITB    #IDEMP,TRTAB(R1)        ;DOES IDEMPOTENT LAW APPLY?
001344                                  BOFF    10$                     ;BRANCH IF NOT
001346  012702  000000G                 MOV     #SNODE,CUR              ;GET ADDRESS OF CONSTANT NODE
001352  010062  000000G                 MOV     R0,QNARG1(CUR)          ;DEFINE ARGUMENT AS CONSTANT
001356  000167  001266                  JMP     RELINK                  ;ADJUST STACK AND EXIT ROUTINE
                                ;
                                ;       TEST FOR TAUTOLOGY, CONTRADICTION, OR ABSORPTION ON LEFT ARGUMENT
001362                          10$:
001362  016267  000000G 000206'        MOV     QNARG2(CUR),ARGTMP      ;SET RIGHT ARGUMENT AS IDENTITY RESULT
001370                                  CHECK   QNFLU,15$               ;SKIP IF ARGUMENT IS NOT A NODE
001376  022710  000000G                 CMP     #TOKNOT,@R0             ;IS ARGUMENT'S OPERATOR A NOT?
001402  001004                          BNE     15$                     ;SKIP IF OPERATOR OTHER THAN NOT
001404                                  CALL    STAUTL                  ;TAUTOLOGY/CONTRADICTION ROUTINE
001410                                  CALL    DEMORG                  ;CHECK FOR DEMORGAN LAW SIMPLIFICATION
001414                          15$:
001414                                  CALL    ABSORB                  ;ABSORPTION ROUTINE
                                ;
                                ;       RETURN HERE IFF TAUTOLOGY, CONTRADICTION, OR ABSORPTION INAPPLICABLE
                                ;       TEST FOR TAUTOLOGY, CONTRADICTION, OR ABSORPTION ON RIGHT ARGUMENT
001420                          20$:
001420  016200  000000G                 MOV     QNARG2(CUR),R0          ;SET RIGHT ARGUMENT AS THE ONE TO BE CHECKED
001424  016267  000000G 000206'        MOV     QNARG1(CUR),ARGTMP      ;SET LEFT ARGUMENT AS IDENTITY RESULT
001432                                  CHECK   QNFLU,25$               ;SKIP IF ARGUMENT IS NOT A NODE
001440  022710  000000G                 CMP     #TOKNOT,@R0             ;IS ARGUMENT'S OPERATOR A NOT?
001444  001002                          BNE     25$                     ;SKIP IF OPERATOR OTHER THAN NOT
001446                                  CALL    STAUTL                  ;TAUTOLOGY/CONTRADICTION ROUTINE
001452                          25$:
001452                                  CALL    ABSORB                  ;ABSORPTION ROUTINE
```

```
                                ;       RETURN HERE IFF TAUTOLOGY, CONTRADICTION, OR ABSORPTION INAPPLICABLE.
                                ;
001456                                  CALL    POLSIM                  ;POLARITY SIMPLIFICATION ROUTINE.
001462  000405                          BR      XSIMP                   ;EXIT SIMPLIFICATION
                                ;
                                ;       SIMPLIFICATION CHECKS FOR UNARY OPERATOR.
001464                          30$:
001464  022712  000000G                 CMP     #TOKNOT,@CUR            ;CHECK IF CURRENT OPERATOR IS NOT.
001470  001002                          BNE     XSIMP                   ;EXIT IF OTHER THAN NOT.
001472                                  CALL    STAUTU                  ;RECHECK TAUTOLOGY/CONTRADICTION.
                                ;
                                ;       EXIT POINT FOR NO SIMPLIFICATION
001476                          XSIMP:
001476  000241                          CLC                             ;SWITCH TO AVOID NODE REPROCESSING.
001500  000207                          RETURN
```

```
                                        ;       TAUTOLOGY/CONTRADICTION SUBROUTINE
                                        ;       (R0) = THE ARGUMENT OF THE CURRENT NODE BEING CHECKED
                                        ;       (ARGTMP) = THE OTHER ARGUMENT OF THE CURRENT NODE
001502                                  STAUTL:
001502  026067  000000G 000206'                CMP     QNARG1(R0),ARGTMP       ;COMPARE ARGUMENTS OF NOT AND CURRENT NODE
001510  001420                                BEQ     STAUTR                  ;SIMPLIFY IF ARGUMENTS ARE EQUAL
001512  001041                                BNE     XTAUT                   ;EXIT TEST IF ARGUMENTS NOT EQUAL
                                        ;
                                        ;       ENTRY POINT FOR TAUTOLOGY/CONTRADICTION CHECK WHEN CURRENT OPERATOR IS NOT
                                        ;       THIS TEST SUCCEEDS ONLY WHEN A CHAIN OF SIMPLIFICATIONS OCCURS
                                        ;       DURING A BACKUP OF THE TREE
                                        ;
001514                                  STAUTU:
001514  005715                                TST     @TOP                    ;IF STACK IS EMPTY
001516  001437                                BEQ     XTAUT                   ;EXIT TEST
001520  032764  004000  000000G                BIT     #QNSTAK,QNATTR(STK)     ;STACK LINKAGE?
001526                                          BON     10$                     ;BRANCH IF RIGHT
001530  026400  000000G                        CMP     QNARG2(STK),R0          ;COMPARE - LEFT LINKAGE
001534  000402                                BR      15$
001536                                  10$:
001536  026400  000000G                        CMP     QNARG1(STK),R0          ;COMPARE - RIGHT LINKAGE
001542                                  15$:
001542  001025                                BNE     XTAUT                   ;EXIT IF ARGUMENTS ARE UNEQUAL
001544  010402                                MOV     STK,CUR                 ;STACK MUST BE RETRACTED
001546                                          CALL    RETRCT                  ;CALL RETRACT SUBROUTINE
001552                                  STAUTR:
001552  005726                                TST     (SP)+                   ;NO RETURN TO SIMPFY - RESTORE SYSTEM STACK
001554  111201                                MOVB    @CUR,R1                 ;GET CURRENT OPERATOR TO TEST WHICH LAW
001556  012702  000000G                        MOV     #SNODE,CUR              ;REPLACE CURRENT NODE WITH CONSTANT NODE
001562  132761  000002  000212'                BITB    #TAUTL,TRTAB(R1)        ;LAW OF TAUTOLOGY?
001570                                          BOFF    30$                     ;NO - APPLY LAW OF CONTRADICTION
                                        ;
001572  012762  100000  000000G                MOV     #QNFLU,QNARG1(CUR)      ;RESULT IS LOGICAL CONSTANT TRUE
001600  000167  001044                        JMP     RELINK                  ;EXIT SIMPLIFICATION ROUTINE
001604                                  30$:
001604  012762  140000  000000G                MOV     #QNFLS!QNFLU,QNARG1(CUR) ;RESULT IS LOGICAL CONSTANT FALSE
001612  000167  001032                        JMP     RELINK                  ;EXIT SIMPLIFICATION ROUTINE
                                        ;
                                        ;       EXIT POINT FOR NO SIMPLIFICATION
001616                                  XTAUT:
001616  000207                                RETURN
```

```
                                        ;       DEMORGAN LAW SIMPLIFICATION TEST
                                        DEMORG:
001620  016701  000206'                        MOV     ARGTMP,R1               ;FETCH OTHER ARGUMENT
001624  003461                                BLE     XDMORG                  ;EXIT IF FLU OR NULL
001626  022711  000000G                       CMP     #TOKNOT,@R1             ;IS ITS OPERATOR ALSO NOT?
001632  001056                              BNE     XDMORG                  ;EXIT IF OTHER THAN NOT
001634  022712  000000G                       CMP     #TOKAND,@CUR            ;CHECK IF OPERATOR IS AND
001640  001010                              BNE     10$                     ;SIMPLIFY UNCONDITIONALLY IF NOT AND
                                        ;
                                        ;       FOR AND OPERATOR TO BE SIMPLIFIED, ALL POLARITY BITS OF ITS ARGUMENTS MUST BE OFF
                                        ;
001642  032761  070000  000000G               BIT     #POLRTY,QNATTR(R1)      ;TEST POLARITY OF LEFT ARGUMENT
001650                                      BON     XDMORG                  ;NO SIMPLIFICATION IF ANY ARE ON
001652  032760  070000  000000G               BIT     #POLRTY,QNATTR(R0)      ;TEST POLARITY OF RIGHT ARGUMENT
001660                                      BON     XDMORG                  ;NO SIMPLIFICATION IF ANY ARE ON
001662                                  10$:
001662  022712  000000G                       CMP     #TOKHOR,@CUR            ;IF CURRENT OPERATOR IS OR
001666  001403                              BEQ     20$                     ;BRANCH TO SELECT OPERATOR
001670  012712  000000G                       MOV     #TOKOR,@CUR             ;FOR AND/PROXOP, SIMPLIFIED OPERATOR IS OR
001674  000402                              BR      30$
001676                                  20$:
001676  012712  000000G                       MOV     #TOKAND,@CUR            ;FOR OR, SIMPLIFIED OPERATOR IS AND
001702                                  30$:
001702  016162  000000G 000000G               MOV     QNARG1(R1),QNARG1(CUR)  ;SET LEFT ARGUMENT
001710  016062  000000G 000000G               MOV     QNARG1(R0),QNARG2(CUR)  ;SET RIGHT ARGUMENT
001716  042762  070000  000000G               BIC     #POLRTY,QNATTR(CUR)     ;CLEAR POLARITY BITS
                                        ;
001724                                      CALL    GETQND                  ;GET NEW ROOT NODE
001730  012713  000000G                       MOV     #TOKNOT,@NEW            ;SET NOT AS OPERATOR
001734  010263  000000G                       MOV     CUR,QNARG1(NEW)         ;SIMPLIFIED NODE IS ITS LEFT ARGUMENT
001740                                      CALL    SETPOL                  ;INHERIT ITS ATTRIBUTES
001744  012702  000000G                       MOV     #SNODE,CUR              ;CONSTANT NODE BECOMES CURRENT
001750  010362  000000G                       MOV     NEW,QNARG1(CUR)         ;SET ITS ARGUMENT
001754  010203                              MOV     CUR,NEW                 ;SET UP SETPOL CALL
001756                                      CALL    SETPOL                  ;SET POLARITY ATTRIBUTES
001762  005726                              TST     (SP)+                   ;NOT RETURNING TO SIMPFY
001764  000167  000660                      JMP     RELINK                  ;EXIT SIMPLIFICATION ROUTINE
                                        ;
                                        ;       EXIT POINT FOR NO SIMPLIFICATION
001770                                  XDMORG:
001770  000207                              RETURN
```

```
                                ;       ABSORPTION LAW SIMPLIFICATION SUBROUTINE
                                ;       AT ENTRY, STK AND CUR MUST BE SET
                                ;       (R0) = THE ARGUMENT BEING COMPARED TO THE STACK OPERAND (I.E. THE NON-LINK OPERAND)
                                ;       (ARGTMP) = THE OTHER ARGUMENT OF THE CURRENT NODE
001772'                         ABSORB:
001772' 005715                          TST     @TOP                    ;CHECK IF STACK EMPTY
001774  001455                         BEQ     XABSRB                  ;AUTOMATIC EXIT IF SO
001776  032764  004000  000000G'       BIT     #QNSTAK,QNATTR(STK)     ;STACK LINKAGE?
002004                                  BON     10$                     ;BRANCH IF RIGHT SUBTREE
                                ;
002006  026467  000000G 000206'         CMP     QNARG2(STK),ARGTMP      ;COMPARE STACK'S RIGHT ARGUMENT
002014  000403                         BR      20$
002016                          10$:
002016  026467  000000G 000206'         CMP     QNARG1(STK),ARGTMP      ;COMPARE STACK'S LEFT ARGUMENT
002024                          20$:
002024  001041                         BNE     XABSRB                  ;EXIT UNLESS ARGUMENTS ARE EQUAL
002026  111201                         MOVB    @CUR,R1                 ;FETCH CURRENT OPERATOR
002030  116101  000212'                 MOVB    TRTAB(R1),R1            ;AND GET ITS SIMPLIFICATION ATTRIBUTE
002034  111400                         MOVB    @STK,R0                 ;FETCH STACK OPERATOR
002036                                  CHECK   BIT3,50$                ;CHECK FOR PROXOP - BRANCH IF SO
002044                                  CHECK   BIT1,40$                ;CHECK FOR AND - BRANCH IF SO
                                ;
                                ;       OR (EITHER LOW-LEVEL OR HIGH-LEVEL) OPERATOR
002052  012700  000020                 MOV     #ABSOR,R0               ;SELECT ABSORPTION ATTRIBUTE FOR OR
002056  000410                         BR      60$                     ;GO ABSORB
                                ;
                                ;       AND OPERATOR
002060                          40$:
002060  012700  000040                 MOV     #ABSAND,R0              ;SELECT ABSORPTION ATTRIBUTE FOR AND
002064  000405                         BR      60$                     ;GO ABSORB
                                ;
                                ;       PROXIMITY OPERATOR
002066                          50$:
002066  012700  000100                 MOV     #ABSPX0,R0              ;SET UP ABSORPTION ATTRIBUTE FOR PROXIMITY
002072  032701  000300                 BIT     #ABSPX0!ABSPX1,R1       ;DOES OPERATOR ABSORB INTO PROXIMITY
002076                                  BOFF    XABSRB                  ;EXIT IF NOT
                                ;
                                ;       ABSORPTION TO BE PERFORMED - R0 CONTAINS THE BIT WHICH SELECTS THE TYPE
                                ;       IF THE CORRESPONDING BIT IN R1 IS ON, RESULT IS ARGUMENT (FROM ARGTMP)
                                ;       IF THE CORRESPONDING BIT IN R1 IS OFF, RESULT IS JUST THE CURRENT NODE
002100                          60$:
002100  030001                         BIT     R0,R1                   ;TEST APPROPRIATE ABSORPTION ATTRIBUTE
002102'                                 BOFF    70$                     ;EVERYTHING ALREADY SET
002104  016702  000206'                 MOV     ARGTMP,CUR              ;GET RESULT OF ABSORPTION
002110                          70$:
002110  010203                         MOV     CUR,NEW                 ;SAVE RESULT
002112  012702  000000G'               MOV     #SNODE,CUR              ;MAKE CONSTANT NODE CURRENT
002116  010362  000000G'               MOV     NEW,QNARG1(CUR)         ;SET RESULT IN CONSTANT NODE
002122  005726                         TST     (SP)+                   ;NOT RETURNING TO SIMPFY - FIX HARDWARE STACK
002124  000167  000500                 JMP     RETRCT                  ;EXIT WITH STACK RETRACTION
                                ;
                                ;       EXIT POINT FOR NO SIMPLIFICATION
002130                          XABSRB:
002130  000207                         RETURN
```

```
                                          ;       POLARITY SIMPLIFICATION SUBROUTINE
                                          ;       AT ENTRY, STK AND CUR MUST BE SET
                                          ;
002132                                    POLSIM:
002132  005715                                    TST     @TOP                    ;IF STACK IS EMPTY
002134  001534                                    BEQ     XPOLSM                  ;AUTOMATIC EXIT
002136  021412                                    CMP     @STK,@CUR               ;DOES CURRENT OPERATOR = STACK OPERATOR
002140  001132                                    BNE     XPOLSM                  ;EXIT IF NOT EQUAL
002142  032712  000010                            BIT     #BIT3,@CUR              ;IS CURRENT OPERATOR PROXIMITY
002146                                            BON     XPOLSM                  ;IF SO, POLARITY INAPPLICABLE
002150  022712  000000G                           CMP     #TOKAND,@CUR            ;IF OPERATOR IS AND
002154  001004                                    BNE     10$
002156  032762  010000  000000G                   BIT     #QNPOL,QNATTR(CUR)      ;SKIP POLARITY SIMPLIFICATION UNLESS
002164                                            BOFF    XPOLSM                  ;POLARITY SETTINGS ARE DUE TO LOW LEVEL NOTS
002166                                    10$:
002166  032762  040000  000000G                   BIT     #QNLPOL,QNATTR(CUR)     ;DOES CURRENT NODE HAVE LEFT POLARITY
002174                                            BOFF    15$                     ;NO - WILL CHECK FOR RIGHT POLARITY
002176  032762  020000  000000G                   BIT     #QNRPOL,QNATTR(CUR)     ;IS POLARITY ALSO RIGHT?
002204                                            BON     XPOLSM                  ;EXIT BECAUSE POLARITY NOT MIXED
002206  000404                                    BR      20$                     ;DO SIMPLIFICATION
002210                                    15$:
002210  032762  020000  000000G                   BIT     #QNRPOL,QNATTR(CUR)     ;IS POLARITY RIGHT ALONE?
002216                                            BOFF    XPOLSM                  ;EXIT BECAUSE CURRENT OPERATOR HAS NO POLARITY
002220                                    20$:
002220                                            CALL    GETQND                  ;GET NEW NODE
002224  011567  000206'                           MOV     @TOP,ARGTMP             ;GET ADDRESS OF NODE AT TOP OF STACK
002230  032764  004000  000000G                   BIT     #QNSTAK,QNATTR(STK)     ;STACK LINKAGE?
002236                                            BON     25$                     ;BRANCH IF RIGHT SUBTREE
                                          ;
002240  062767  000000G 000206'                   ADD     #QNARG2,ARGTMP          ;SET TO RIGHT ARGUMENT OF STACK NODE
002246  010364  000000G                           MOV     NEW,QNARG1(STK)         ;LINK STACK TO NEW NODE
002252  012767  020000  000210'                   MOV     #QNRPOL,ARGTMP+2        ;STACK'S RIGHT POLARITY TO SELECT OTHER STACK ARGUME
002260  000410                                    BR      30$
002262                                    25$:
002262  062767  000000G 000206'                   ADD     #QNARG1,ARGTMP          ;SET TO LEFT ARGUMENT OF STACK NODE
002270  010364  000000G                           MOV     NEW,QNARG2(STK)         ;LINK STACK TO NEW NODE
002274  012767  040000  000210'                   MOV     #QNLPOL,ARGTMP+2        ;STACK'S LEFT POLARITY TO SELECT OTHER STACK ARGUMEN
002302                                    30$:
002302  032762  040000  000000G                   BIT     #QNLPOL,QNATTR(CUR)     ;CHECK CURRENT POLARITY
002310                                            BOFF    40$                     ;BRANCH IF RIGHT POLARITY
002312  036764  000210' 000000G                   BIT     ARGTMP+2,QNATTR(STK)    ;CHECK APPROPRIATE STACK POLARITY
002320                                            BON     50$                     ;IF ON, USE CURRENT RIGHT ARGUMENT IN STACK
002322                                            BOFF    60$                     ;IF OFF, USE CURRENT LEFT ARGUMENT IN STACK
002324                                    40$:
002324  036764  000210' 000000G                   BIT     ARGTMP+2,QNATTR(STK)    ;CHECK APPROPRIATE STACK POLARITY
002332                                            BON     60$                     ;IF ON, USE CURRENT LEFT ARGUMENT AS NEW STACK ARGUM
```

```
1031 002334                            50$:
1032 002334  016263  000000G 000000G          MOV     QNARG1(CUR),QNARG1(NEW) ;CURRENT LEFT ARGUMENT IS UNCHANGED
1033 002342  017763  000206' 000000G          MOV     @ARGTMP,QNARG2(NEW)     ;OLD STACK ARGUMENT IS INTERCHANGED
1034 002350  016277  000000G 000206'          MOV     QNARG2(CUR),@ARGTMP     ;WITH CURRENT RIGHT ARGUMENT
1035 002356  000411                          BR      70$
1036 002360                            60$:
1037 002360  016263  000000G 000000G          MOV     QNARG2(CUR),QNARG2(NEW) ;CURRENT RIGHT ARGUMENT IS UNCHANGED
1038 002366  017763  000206' 000000G          MOV     @ARGTMP,QNARG1(NEW)     ;OLD STACK ARGUMENT IS INTERCHANGED
1039 002374  016277  000000G 000206'          MOV     QNARG1(CUR),@ARGTMP     ;WITH CURRENT LEFT ARGUMENT
1040 002402                            70$:
1041 002402  011213                          MOV     @CUR,@NEW               ;COPY CURRENT OPERATOR
1042 002404                                  CALL    SETPOL                  ;SET NEW POLARITY ATTRIBUTES
1043 002410  010402                          MOV     STK,CUR                 ;STACK NODE TO BE RETRACTED
1044 002412  010203                          MOV     CUR,NEW                 ;SETUP FOR SETPOL ROUTINE
1045 002414                                  CALL    SETPOL                  ;SET POLARITY ATTRIBUTES
1046 002420  005726                          TST     (SP)+                   ;NOT RETURNING TO SIMPFY
1047 002422  000167  000202                  JMP     RETRCT                  ;EXIT SIMPLIFICATION
1048                                    ;
1049                                    ;       EXIT POINT FOR NO POLARITY SIMPLIFICATION
1050 002426                            XPOLSM:
1051 002426  000207                          RETURN
```

```
                                        ;       CONSTANT NODE SIMPLIFICATION
                                        ;       (CUR)=SNODE, THE CONSTANT ARGUMENT NODE
                                        ;
002430                                  QTS:
002430  010203                          MOV     CUR,NEW                 ;SAVE CURRENT NODE
002432  016202  000000G                 MOV     QNARG1(CUR),CUR         ;SET CONSTANT TO CURRENT NODE
002436  005715                          TST     @TOP                    ;CHECK IF STACK EMPTY
002440  001472                          BEQ     QTSX                    ;YES - ALL DONE - EXIT
002442  005702                          TST     CUR                     ;FLU/LOGICAL CONSTANT OR NODE
002444  100054                          BPL     QTSN                    ;BRANCH IF NODE
002446  032702  037777                  BIT     #LOGCON,CUR             ;FLU OR LOGICAL CONSTANT
002452                                  BON     QTSN                    ;BRANCH IF FLU
                                        ;
                                        ;       LOGICAL CONSTANT
                                        ;
002454  022764  000000G 000000G         CMP     #TOKNOT,QNOPCD(STK)     ;IS OPERATOR NOT?
002462  001005                          BNE     10$                     ;BRANCH IF NOT
002464  012701  040000                  MOV     #QNFLS,R1               ;SET CORRECT SIMPLIFICATION VALUE
002470  074102                          XOR     R1,CUR                  ;BY INVERTING THE CONSTANT
002472  010263  000000G                 MOV     CUR,QNARG1(NEW)         ;SET VALUE IN LEFT ARGUMENT OF NODE
002476                                  10$:
002476  032764  004000  000000G         BIT     #QNSTAK,QNATTR(STK)     ;STACK LINKAGE?
002504                                  BON     15$                     ;BRANCH IF RIGHT SUBTREE
                                        ;
002506  016467  000000G 000206'         MOV     QNARG2(STK),ARGTMP      ;FETCH IDENTITY VALUE - LEFT SUBTREE
002514  000403                          BR      20$
002516                                  15$:
002516  016467  000000G 000206'         MOV     QNARG1(STK),ARGTMP      ;FETCH IDENTITY VALUE - RIGHT SUBTREE
002524                                  20$:
002524  032702  040000                  BIT     #QNFLS,CUR              ;IS LOGICAL CONSTANT TRUE OR FALSE?
002530                                  BON     30$                     ;BRANCH IF FALSE
                                        ;
                                        ;       LOGICAL CONSTANT OF TRUE - OPERATOR'S IDENT BIT DETERMINES RESULT
                                        ;       IF SET, RESULT IS CONSTANT (WHICH IS POINTED TO BY NODE IN NEW)
                                        ;       IF NOT SET, RESULT IS OTHER ARGUMENT (WHICH IS IN ARGTMP)
                                        ;
002532  012767  000004  000210'         MOV     #IDENT,ARGTMP+2
002540  000403                          BR      40$
                                        ;
                                        ;       LOGICAL CONSTANT OF FALSE - OPERATOR'S NILPT BIT DETERMINES RESULT
                                        ;       IF SET, RESULT IS CONSTANT (WHICH IS IN CUR)
                                        ;       IF NOT SET, RESULT IS OTHER ARGUMENT (WHICH IS IN ARGTMP)
002542                                  30$:
002542  012767  000010  000210'         MOV     #NILPT,ARGTMP+2
002550                                  40$:
002550  016401  000000G                 MOV     QNOPCD(STK),R1          ;GET OPERATOR CODE FOR TABLE INDEX
002554  136761  000210' 000212'         BITB    ARGTMP+2,TRTAB(R1)      ;TEST WHICH VALUE TO BE RETURNED
002562                                  BON     50$                     ;BRANCH IF CONSTANT TO BE RETURNED
                                        ;
002564  016703  000206'                 MOV     ARGTMP,NEW              ;RETURNED VALUE IS OTHER ARGUMENT
002570                                  50$:
002570  010302                          MOV     NEW,CUR                 ;RESTORE CURRENT NODE
002572  000167  000032                  JMP     RETRCT                  ;EXIT WITH SIMPLIFICATION DONE
```

```
1107                                    ;       CONSTANT ARGUMENT IS FLU OR NODE ID
1108 002576                             QTSN:
1109 002576  032764  004000  000000G     BIT     #QNSTAK,QNATTR(STK)     ;STACK LINKAGE?
1110 002604                             BON     10$
1111                                    ;
1112 002606  010264  000000G     MOV     CUR,QNARG1(STK)         ;REPLACE ARGUMENT - LEFT SUBTREE
1113 002612  000402              BR      20$
1114 002614                             10$:
1115 002614  010264  000000G     MOV     CUR,QNARG2(STK)         ;REPLACE ARGUMENT - RIGHT SUBTREE
1116 002620                             20$:
1117 002620  010402              MOV     STK,CUR                 ;RETRACT NODE FROM STACK
1118 002622  000167  000002      JMP     RETRCT                  ;EXIT TRANSFORMATION
1119                                    ;
1120                                    ;       EXIT FROM SUBROUTINE WITHOUT SIMPLIFICATION - RETURNS TO CALLER
1121 002626                             QTSX:
1122 002626  000207                     RETURN
```

```
                                        .SBTTL  TRANSFORMATION EXIT ROUTINE
                                        ;       COMMON EXIT FROM ALL TRANSFORMATIONS
                                        ;       EXIT ALSO FOR SIMPLIFICATION ROUTINES WHEN PERFORMED
                                        ;
002630                                  RETRCT:
002630  116467  000000G 000204'                MOVB    QNSTE(STK),STATE        ;EXIT WITH RETRACTION
002636  042764  104000  000000G                BIC     #TRVRSE,QNATTR(STK)     ;RESTORE CURRENT STATE
002644                                          NODPOP                          ;RESET ANY SUBTREE TRAVERSAL FLAGS
002650                                                                          ;POP NODE STACK & DISCARD
002650                                  RELINK:
002650  005715                                  TST     @TOP                    ;TEST IF STACK EMPTY
002652  001414                                  BEQ     30$                     ;EXIT IF IT IS
002654  032764  004000  000000G                BIT     #QNSTAK,QNATTR(STK)     ;TEST SUBTREE TRAVERSAL
002662                                          BON     10$                     ;BRANCH IF RIGHT
                                        ;
002664  010264  000000G                        MOV     CUR,QNARG1(STK)         ;RELINK STACK'S LEFT ARGUMENT
002670  000402                                  BR      20$
002672                                  10$:
002672  010264  000000G                        MOV     CUR,QNARG2(STK)         ;RELINK STACK'S RIGHT ARGUMENT
002676                                  20$:
002676  010403                                  MOV     STK,NEW
002700                                          CALL    SETPOL                  ;SETUP FOR POLARITY ADJUSTING
002704                                                                          ;RECHECK POLARITY OF STACK NODE
002704  000261                          30$:
002706  000207                                  SEC                             ;INDICATES TRANSFORMATION OR SIMPLIFICATION DONE
                                                RETURN
```

```
1149                                            .SBTTL  NODE STACK PUSH ROUTINE
1150 002710                             NODPUT:
1151 002710  022705  000003G                    CMP     #TOKSTK+3,TOP           ;CHECK IF ENOUGH ROOM IN STACK
1152 002714  103405                             BLO     10$                     ;BRANCH IF OK
1153                                    ;
1154 002716  012767  000145  000000G            MOV     #ASKOVF,ERRCDE          ;SET NODE STACK OVERFLOW ERROR CODE
1155 002724  000167  175276                     JMP     XERROR
1156 002730                             10$:
1157 002730  010204                             MOV     CUR,STK                 ;CURRENT NODE BECOMES TOP OF STACK NODE
1158 002732  010445                             MOV     STK,-(TOP)              ;PUSH NODE ADDRESS INTO STACK
1159 002734  000207                             RETURN
```

```
1161                                            .SBTTL  SET·POLARITY·ATTRIBUTE·SUBROUTINE·
1162·                                   ;
1163                                    ;       (NEW)  =·ADDRESS·OF·NODE·WHICH·IS·TO·HAVE·POLARITY·ATTRIBUTES·SET·
1164                                    ;       (R1)   =·SUCCESSIVE·ARGUMENTS·OF·THE·NODE·- SAVED·AND RESTORED·
1165                                    ;
1166 002736                             SETPOL:
1167 002736                                     SAVE·   R1
1168 002740  042763  070000  000000G·           BIC·    #POLRTY,QNATTR(NEW)     ;RESET·POLARITY·ATTRIBUTES·
1169                                    ;
1170 002746  016301  000000G·                   MOV·    QNARG1(NEW),R1          ;SET·UP·LEFT·ARGUMENT·OF·NODE·
1171 002752· 003406                             BLE·    SRPOL·                  ;BRANCH·IF FLU·
1172·002754                                     CALL·   SPOL                    ;SET·POLARITY·SUBROUTINE·
1173 002760  103003                             BCC·    SRPOL·                  ;BRANCH·IF NO·POLARITY·
1174 002762· 052763  040000  000000G·           BIS·    #QNLPOL,QNATTR(NEW)     ;SET·LEFT·POLARITY·
1175 002770                             SRPOL:
1176 002770  016301  000000G·                   MOV·    QNARG2(NEW),R1          ;SET·UP·RIGHT·ARGUMENT·OF·NODE
1177 002774  003406                             BLE·    XSETOP·                 ;BRANCH·IF FLU·
1178 002776                                     CALL·   SPOL                    ;SET·POLARITY·SUBROUTINE·
1179 003002· 103003                             BCC·    XSETOP·                 ;BRANCH·IF NO·POLARITY·
1180 003004  052763  020000  000000G·           BIS·    #QNRPOL,QNATTR(NEW)     ;SET·RIGHT POLARITY·
1181 003012·                            XSETOP:
1182·003012· 032763  010000  000000G·           BIT·    #QNPOL,QNATTR(NEW)      ;IF·POLARITY·HAS·NOT BEEN·SET·
1183 003020                                     BOFF·   10$                     ;OPERATOR·IS·LOW·LEVEL·
1184 003022· 005713                             TST·    @NEW                    ;IF·CONSTANT·ARGUMENT·OPERATOR
1185 003024  001414                             BEQ·    XSETPL·                 ;AUTOMATIC EXIT·
1186 003026  052713  000001                     BIS·    #QNHLOP,@NEW·           ;OTHERWISE, MAKE·OPERATOR·HIGH LEVEL·
1187 003032· 000411                             BR      XSETPL·                 ;EXIT·SUBROUTINE·
1188 003034                             10$:
1189 003034  032763  070000  000000G·           BIT·    #POLRTY,QNATTR(NEW)     ;IF·ANY·POLARITY·BITS·ARE·SET·
1190 003042·                                    BON·    XSETPL·                 ;NOTHING·MORE·TO·DO·
1191 003044  022713  000000G·                   CMP·    #TOKHOR,@NEW·           ;IS·OPERATOR·HIGH-LEVEL·OR?·
1192·003050  001002·                            BNE·    XSETPL·                 ;EXIT·IF·NOT·
1193 003052· 012713  000000G·                   MOV·    #TOKOR,@NEW·            ;MAKE·OPERATOR·LOW-LEVEL·OR·
1194 003056                             XSETPL:
1195 003056                                     RESTORE·R1
1196 003060  000207                             RETURN·
```

```
1198                                    ;
1199                                    ;       INHERIT ARGUMENT ATTRIBUTE SUBROUTINE.
1200                                    ;       NOTE - THIS SUBROUTINE IS ALSO USED BY THE PARSER MODULE.
1201 003062                            SPOL::
1202 003062                                    SAVE    R0
1203 003064  122711  000000G                   CMPB    #TOKSBD,@R1             ;IS ARGUMENT'S OPERATOR SUBDOC
1204 003070  001425                            BEQ     10$                     ;NODE DOES NOT INHERIT POLARITY.
1205 003072  016100  000000G                   MOV     QNATTR(R1),R0           ;FETCH ATTRIBUTES OF ARGUMENT.
1206 003076                                    CHECK   QNSBEL,,5$              ;TEST IF ARGUMENT HAS POLARITY
1207 003104  052763  002000  000000G           BIS     #QNSBEL,QNATTR(NEW)     ;SET NODE'S POLARITY IF SO.
1208 003112                            5$:
1209 003112                                    CHECK   QNPOL,20$               ;POLARITY INHERITED.
1210 003120  122711  000000G                   CMPB    #TOKNOT,@R1             ;IF ARGUMENT'S OPERATOR IS NOT
1211 003124  001414                            BEQ     20$                     ;POLARITY ALSO INHERITED.
1212 003126                                    CHECK   LRPOL,30$               ;LEFT OR RIGHT POLARITY TO BE INHERITED.
1213 003134  122711  000000G                   CMPB    #TOKAND,@R1             ;IF ARGUMENT'S OPERATOR IS AND
1214 003140  001411                            BEQ     30$                     ;SET SWITCH TO INHERIT DIRECTIONAL POLARITY.
1215 003142  000403                            BR      15$                     ;OTHERWISE, CLEAR SWITCH.
1216 003144                            10$:
1217 003144  052763  002000  000000G           BIS     #QNSBEL,QNATTR(NEW)     ;SET SUBELEM ATTRIBUTE IN NEW NODE.
1218 003152                            15$:
1219 003152  000241                            CLC                             ;NO POLARITY EXIT.
1220 003154  000404                            BR      40$                     ;SUBROUTINE EXIT.
1221 003156                            20$:
1222 003156  052763  010000  000000G           BIS     #QNPOL,QNATTR(NEW)      ;SET NEW NODE'S POLARITY.
1223 003164                            30$:
1224 003164  000261                            SEC                             ;SET SWITCH TO ASSIGN LEFT OR RIGHT POLARITY ON RETU
1225 003166                            40$:
1226 003166                                    RESTORE R0
1227 003170  000207                            RETURN
1228         000001                            .END
```

QUERY TRANSFORMATION PROGRAM MACRO M1110 27-MAR-80 12:57 PAGE 47-1
SYMBOL TABLE

```
ABSAND=  000040          B.HRLP  000126      010 FN.LGU  000036      011 QE.ISO=  000016          SIMPFY  001312R     017
ABSOR =  000020          B.HRLR  000122      010 FN.MFO  000024      011 QE.MFL=  000013          SLBOVF=  000164
ABSORB  001772R     017  B.HRLW  000124      010 FN.MHR  000010      011 QE.MFM=  000010          SNODE =  ****** GX
ABSPX0=  000100          B.NMBR  000052      010 FN.NMB  000044      011 QE.MNT=  000012          SPOL    003062RG    017
ABSPX1=  000200          B.NORY  000232      010 FN.OLS  000006      011 QE.MOP=  000007          SRPOL   002770R     017
ARGSTK=  ****** GX       B.QLSZ  000106      010 FN.QRY  000020      011 QE.MSD=  000011          SR.ARE  000114      002
ARGTMP  000206R     016  B.QMAP  000234      010 FN.SF0  000030      011 QE.MTS=  000002          SR.ARS  000106      002
ASKOVF=  000145          B.QSPL  000316      010 FN.SFI  000032      011 QE.NLQ=  000006          SR.DAY  000010      002
BITVAL=  000000          B.QTTM  000076      010 FN.SHD  000042      011 QE.NOS=  000025          SR.DLT  000014      002
BIT0  =  000001          B.QUQP  000056      010 FSCAND  000132R     016 QE.NRB=  000024          SR.ECB  000047      002
BIT1  =  000002          B.SFDB  000010      010 FSCFLU  000166R     016 QE.NRI=  000023          SR.ECH  000046      002
BIT10 =  002000          B.SIZE  000772      010 FSCHOR  000150R     016 QE.NTK=  000027          SR.ECL  000050      002
BIT11 =  004000          B.SNDP  000012      010 FSCNOT  000060R     016 QE.PXI=  000017          SR.FIB  000012      002
BIT12 =  010000          B.SSQ   000004      010 FSCOFV  000022R     016 QE.PX2=  000020          SR.GRE  000100      002
BIT13 =  020000          B.SSQF  000050      010 FSCOR   000114R     016 QE.PX3=  000021          SR.GRS  000072      002
BIT14 =  040000          B.STAT  000044      010 FSCPXP  000076R     016 QE.PX4=  000022          SR.LEN  000122      002
BIT15 =  100000          B.STTE  000053      010 FSCPXS  000076R     016 QE.ROI=  000144          SR.LIN  000066      002
BIT2  =  000004          B.UDOC  000110      010 FSCPXW  000076R     016 QE.UBP=  000014          SR.LIP  000062      002
BIT3  =  000010          CF.B0 =  000070          FSCSBD  000042R     016 QLBOVF=  000163          SR.MON  000006      002
BIT4  =  000020          CF.B2 =  000067          FSCTBL  000040R     016 QNARGI=  ****** GX       SR.NDC  000042      002
BIT5  =  000040          CF.B4 =  000066          FSCTV   000000R     016 QNARG2=  ****** GX       SR.NDS  000036      002
BIT6  =  000100          CF.B6 =  000065          GETQND  000244R     017 QNATTR=  ****** GX       SR.NIN  000030      002
BIT7  =  000200          CF.DR0=  000064          GROUP1  000034R     017 QNDNXT=  ****** GX       SR.NIP  000022      002
BIT8  =  000400          CF.DRI=  000063          GROUP2  000144R     017 QNDSIZ=  ****** GX       SR.SDB  000032      002
BIT9  =  001000          CUR     =%000002         IDEMP =  000001          QNFLS =  040000          SR.SRC  000002      002
BS.CLS=  000002          DBSLEN=  000116          IDENT =  000004          QNFLU =  100000          SR.SUN  000000      002
BS.DBU=  000004          DEMORG  001620R     017  LNPOVF=  000146          QNHLOP=  000001          SR.TWS  000056      002
BS.INA=  000000          DH.BF0  000002      005  LOGCON=  037777          QNLPOL=  040000          SR.WSL  000052      002
BS.OPN=  000001          DH.BF1  000004      005  LRPOL =  060000          QNOPCD=  ****** GX       SR.YR   000004      002
BS.SRC=  000003          DH.CTL  000000      005  M     =  000062          QNPOL =  010000          SR.1IN  000024      002
BYTE0 =  000000          DH.DMC  000010      005  N     =  000002          QNRPOL=  020000          SR.1IP  000016      002
BYTE1 =  000001          DH.FLG  000006      005  NDBOVF=  000175          QNSBEL=  002000          SS.FID  000002      004
BYTE2 =  000002          DN.DCK  000000      013  NEW   =%000003          QNSTAK=  004000          SS.FNB  000010      004
BYTE3 =  000003          DN.NTP  000004      013  NILPT =  000010          QNSTE =  ****** GX       SS.FVR  000006      004
BYTE4 =  000004          DN.NXT  000006      013  NODPUT  002710R     017  QNTLST=  ****** GX       SS.LEN  000012      004
BYTE5 =  000005          DN.ROT  000002      013  N.BFAC=  000004          QNTREE=  100000          SS.STT  000000      004
BYTE6 =  000006          DN.SIZ  000010      013  N.BHGH=  000006          QNXMPS=  165777          STATE   000204R     016
BYTE7 =  000007          EMAOVF=  000166          N.BTCH=  000004          QRYOVF=  000156          STAUTL  001502R     017
BYTE8 =  000010          EMBOVF=  000167          N.BUFB=  004000          QTA     001110R     017  STAUTR  001552R     017
BYTE9 =  000011          EMCOVF=  000170          N.BUFW=  002000          QTC     000622R     017  STAUTU  001514R     017
BYTVAL=  000012          ERRCDE=  ****** GX       N.FOS =  000764          QTD     000312R     017  STK     =%000004
B.BSTA   000054     010  ERROR =  ****** GX       N.PKSZ=  000020          QTDN    001266R     017  ST.ASZ  000020      006
B.CNTX   000046     010  FALOVF=  000160          N.PKTS=  000043          QTFORM  000000RG    017  ST.BSZ  000024      006
B.CQUQ   000060     010  FD.FID  000000      003  N.QURY=  000031          QTH     001250R     017  ST.BTC  000000      006
B.FEMA   000132     010  FD.FNB  000006      003  N.SUNT=  000002          QTN     000462R     017  ST.CSZ  000030      006
B.FEMB   000142     010  FD.FVR  000004      003  OUTPUT=  000000          QTPN    000736R     017  ST.HRL  000010      006
B.FEMC   000152     010  FD.LEN  000010      003  POLOVF=  000157          QTS     002430R     017  ST.LEN  000044      006
B.FFSA   000202     010  FN.DBR  000026      011  POLRTY=  070000          QTSN    002576R     017  ST.QRY  000002      006
B.FFSB   000212     010  FN.DBS  000022      011  POLSIM   002132R     017  QTSX    002626R     017  ST.QSZ  000034      006
B.FFSC   000222     010  FN.DHR  000040      011  QBFOVF=  000150          QTX     000220R     017  ST.SCH  000040      006
B.FMHR   000172     010  FN.EMA  000012      011  QEXOVF=  000165          Q.FDSC  000004      007  ST.UHL  000004      006
B.FQLS   000162     010  FN.EMB  000014      011  QE.DWI=  000001          Q.NQBK  000000      007  ST.XLT  000014      006
B.FSAZ   000100     010  FN.EMC  000016      011  QE.DW2=  000003          Q.NUHL  000002      007  SU.DBU=  000004
B.FSBZ   000102     010  FN.FSA  000000      011  QE.FTB=  000026          Q.SIZE  000014      007  SU.DON=  000006
B.FSCZ   000104     010  FN.FSB  000002      011  QE.IHS=  000005          RELINK  002650R     017  SU.IDL=  000000
B.HBLK   000120     010  FN.FSC  000004      011  QE.ILC=  000004          RETRCT  002630R     017  SU.LOD=  000001
B.HDOC   000114     010  FN.LGQ  000034      011  QE.INO=  000015          SETPOL  002736R     017  SU.SRC=  000002
```

```
QUERY TRANSFORMATION PROGRAM.    MACRO M1110 27-MAR-80 10:57 PAGE 16
SYMBOL TABLE.

SU.SRR= 000005          TA.      000012.     014 TRVRSE= 104000          WORD2. = 000004          XFOSMR= 000007
SU.XPD= 000003          TAUTL. = 000002.         TS       000014      014 WORD3 = 000006          XGTSRE= 000014
SYNEST= 000002.         TC.      000006      014 TSKOVF= 000147          WORD4 = 000010          XHITSK= 000011
SYNNUL= 000001          TD.      000002.     014 TTBOVF= 000161          WORD5 = 000012.         XHLMER= 000002
SYNOP1= 000004          TDBOVF= 000174           UNDEF.   000000      014 WORD6 = 000014          XHOTSK= 000010
SYNOP2= 000020          TDN.     000020      014 VI1OVF= 000173          WORD7 = 000016          XMSCHE= 000000
SYNSBD= 000010          TH.      000010      014 VI2OVF= 000172.         WORD8 = 000020          XPOLSM. 002426R     017
SYNTAB= ****** GX.      TN.      000004      014 VI3OVF= 000171          WORD9 = 000022.         XQTS  = 000003
SYNVAL= 000040          TOKAND= ****** GX.       WN.NTP. 000004      012.WRDVAL= 000024          XQT0  = 000001
S.HRL. = 000240         TOKHOR= ****** GX.       WN.NXT. 000006      012.XABSRB. 002130R.    017 XSETOP. 003012R     017
S0      000000      015 TOKNOT= ****** GX.       WN.ROT. 000002.     012.XBATCH= 000013          XSETPL. 003056R.    017
S1      000001      015 TOKOR. = ****** GX.      WN.SIZ. 000010      012.XDBLOA= 000004          XSIMP.  001476R     017
S2.     000002.     015 TOKSBD= ****** GX.       WN.SRC. 000000      012.XDBPRO= 000012.         XSULOA= 000005
S3      000003      015 TOKSTK= ****** GX.       WN.TYP. 000001      012.XDMCIN= 000006          XTAUT.  001616R.    017
S4      000004      015 TOP.   =%000005          WORD0 = 000000          XDMORG. 001770R.    017 XTBOVF= 000162
S5      000005      015 TPN.     000016      014 WORD1 = 000002.         XERROR. 000226R     017 XTFORM. 000232R     017
S6      000006      015 TRTAB.   000212R     016

. ABS.  000000      000
        000000      001
SRCOFF. 000122.     002.
FDSCOF. 000010      003
SUSOFF. 000012.     004
DHROFF. 000012.     005
STTOFF. 000044      006
QSPLOF. 000014      007
BSTOFF. 000772.     010
FNOFFS. 000044      011
WNODOF. 000010      012.
DNODOF. 000010      013
TRCODE. 000022.     014
STCODE. 000007      015
QTDATA. 000230      016
QTFORM. 003172.     017
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  6364 WORDS. (.25 PAGES)
DYNAMIC MEMORY:  7028 WORDS. ( 27 PAGES)
ELAPSED TIME:  00:01:32.
QTFORM,QTFORM/NL:ME:BEX/-SP=[20,1]P,M,E,MDQT0,JBN,QTFORM.
```

```
000316  005203                          INC     R3
000320  004767  000000G                 JSR     PC,STEP
000324                                  CHECK   CSTNUM,12$      ;IF NUMBER CONTINUE
000332                                  CHECK   CSTDP,50$       ;IF DECIMAL POINT
000340                                  CHECK   CSTER,14$,700$  ;IF END, PROCESS NEXT NUMBER; ELSE, ERROR
000350  005304                  13$:     DEC     R4              ;XFER ZERO
000352  112724  000060                  MOVB    #'0,(R4)+
000356  000470                          BR      15$
000360  005303                  50$:     DEC     R3              ;REMOVE DECIMAL FROM COUNT
000362  005304                          DEC     R4              ;AND REPLACE WITH DATA BASE DEC POINT
000364  112724  000000G                 MOVB    #DECPT,(R4)+
000370                                  SAVE    R3
000372  005705                          TST     R5              ;WHICH PASS?
000374  100003                          BPL     51$             ;SECOND
000376  005367  177420                  DEC     DECFLG          ;1ST, DECFLG=-1
000402  000406                          BR      52$
000404  005267  177412          51$:    INC     DECFLG          ;2RD
000410  001003                          BNE     52$             ;DECFLG=1
000412  012767  000002  177402          MOV     #2,DECFLG       ;DECFLG=2, IF PRIOR ALSO DECIMAL
000420  005003                  52$:    CLR     R3              ;R3=CHAR COUNT
000422  111124                  53$:    MOVB    (R1),(R4)+      ;XFER CHAR
000424  005203                          INC     R3
000426  004767  000000G                 JSR     PC,STEP
000432                                  CHECK   CSTNUM,53$      ;IF NUMBER
000440                                  CHECK   CSTER,,700$     ;IF NOT NUMBER OR END OF RANGE
000446  005303                          DEC     R3              ;IF END OF RANGE
000450  005304                          DEC     R4
000452  122744  000060          54$:    CMPB    #'0,-(R4)       ;ELIMINATE TRAILING 0'S
000456  001002                          BNE     55$
000460  005303                          DEC     R3
000462  000773                          BR      54$
000464  005204                  55$:    INC     R4              ;NOT A ZERO
000466  005705                          TST     R5              ;WHICH PASS?
000470  100403                          BMI     56$             ;1ST
000472  010367  177322                  MOV     R3,HSZD         ;2RD, STORE SIZE OF HIGH DECIMAL STRING
000476  000402                          BR      57$
000500  010367  177312          56$:    MOV     R3,LSZD         ;STORE LOW
000504                          57$:    RESTOR  R3
000506  000414                          BR      15$
000510  012767  000023  000000G 700$:    MOV     #0E,NRI,ERRCDE          ;ILLEGAL, REPORT ERROR
000516  000167  000000G                 JMP     ERROR
000522  012767  000024  000000G 701$:    MOV     #0E,NRB,ERRCDE
000530  000167  000000G                 JMP     ERROR
000534  005304                  14$:    DEC     R4              ;BACK UP FOR CSTER
000536  005303                          DEC     R3
000540  112724  000000G          15$:    MOVB    #SEGOP,(R4)+    ;XFER SEGMENT OPERATOR
000544  005705                          TST     R5              ;1ST OR 2ND NUMBER
000546  100004                          BPL     16$             ;2ND
000550                                  SAVE    R4              ;1ST: SET UP TO PROCESS 2ND NUMBER
000552  010305                          MOV     R3,R5
000554  005003                          CLR     R3
000556  000605                          BR      4$
000560  026727  177236  000001  16$:    CMP     DECFLG,#1       ;PROCEED ACCORDING TO DECIMAL FLAG
000566  003052                          BGT     162$            ;IF BOTH LOW AND HIGH HAD DECIMAL POINT
000570  101033                          BHI     163$            ;IF ONLY LOW HAD DP
000572  002450                          BLT     162$            ;IF NEITHER HAD DP
000574                                  SAVE    R4              ;IF ONLY HIGH HAD DP
```

```
163 000576  122705  000001                  CMPB    #1,R5           ;IF ONLY "0" REPLACE IT WITH DEC. PT.
164 000602  001012                          BNE     165$
165 000604  016600  000004                  MOV     4(SP),R0        ;  GET LBEG PTR
166 000610  122710  000060                  CMPB    #'0,(R0)
167 000614  001005                          BNE     165$
168 000616  112710  000000G                 MOVB    #DECPT,(R0)
169 000622  005305                          DEC     R5
170 000624                                  RESTOR  R4
171 000626  000432                          BR      162$
172 000630  010400              165$:       MOV     R4,R0           ;SHIFT NODE CHAR'S BACK UNTIL SEGOP FOUND
173 000632  005200                          INC     R0
174 000634  005266  000002                  INC     2(SP)           ;       SHIFT HBEG ALSO
175 000640  114440                          MOVB    -(R4),-(R0)     ;       TRAILING SEGOP
176 000642  114440              164$:       MOVB    -(R4),-(R0)     ;       INTERMEDIATE CHAR'S
177 000644  100376                          BPL     164$            ;       IF MINUS, SEGOP FOUND
178 000646  112714  000000G                 MOVB    #DECPT,(R4)     ;INSERT DECIMAL POINT
179 000652                                  RESTOR  R4
180 000654  005204                          INC     R4              ;R4=PAST NEW LAST CHAR
181 000656  000416                          BR      162$
182 000660  005304              163$:       DEC     R4              ;APPEND DECIMAL POINT TO END OF NODE
183 000662  022703  000001                  CMP     #1,R3           ;IF HIGH=0, ELIMINATE 0
184 000666  001006                          BNE     166$
185 000670  122776  000060  000000          CMPB    #'0,@(SP)
186 000676  001002                          BNE     166$
187 000700  005303                          DEC     R3
188 000702  005304                          DEC     R4
189 000704  112724  000000G     166$:       MOVB    #DECPT,(R4)+
190 000710  112724  000000G                 MOVB    #SEGOP,(R4)+
191 000714  026727  177104  000001  162$:   CMP     SGNIND,#1       ;TEST FOR SIGN PRESENCE
192 000722  003006                          BGT     160$            ;IF BOTH NEGATIVE
193 000724  101271                          BHI     700$            ;IF ONLY SECOND NEGATIVE
194 000726  002423                          BLT     161$            ;IF BOTH POSITIVE
195 000730  052767  000100  177072          BIS     #MR,RNGCDE      ;IF ONLY FIRST NEGATIVE
196 000736  000470                          BR      20$
197 000740  016700  177052          160$:   MOV     LSZD,R0         ;SWITCH LOW AND HIGH
198 000744  016767  177050  177044          MOV     HSZD,LSZD
199 000752  010067  177042                  MOV     R0,HSZD
200 000756  011600                          MOV     (SP),R0
201 000760  016616  000002                  MOV     2(SP),(SP)
202 000764  010066  000002                  MOV     R0,2(SP)
203 000770  010300                          MOV     R3,R0
204 000772  010503                          MOV     R5,R3
205 000774  010005                          MOV     R0,R5
206 000776  020503              161$:       CMP     R5,R3           ;COMPARE LOW AND HIGH RANGE SIZES
207 001000  103447                          BLO     20$             ;LOW<HIGH
208 001002  101242                          BHI     700$            ;LOW>HIGH
209 001004  010500                          MOV     R5,R0           ;LOW=HIGH: COMPARE NUMBERS
210 001006  011605                          MOV     (SP),R5         ;R5=HIGH
211 001010  016603  000002                  MOV     2(SP),R3        ;R3=LOW
212 001014  005700                          TST     R0              ;IF SIZE=0, THEN PAST DECIMAL
213 001016  001406                          BEQ     60$
214 001020  122325              17$:        CMPB    (R3)+,(R5)+
215 001022  103436                          BLO     20$             ;LOW<HIGH
216 001024  101231                          BHI     700$            ;LOW>HIGH
217 001026  005267  176774                  INC     LHCSZ           ;COUNT COMMON CHAR'S
218 001032  077006                          SOB     R0,17$
219 001034  005267  176766          60$:    INC     LHCSZ           ;1 MORE COMMON
```

```
    220 001040  052767  000020  176762          BIS     #PD,RNGCDE      ;NOTE PAST DECIMAL
    221 001046  122325                          CMPB    (R3)+,(R5)+     ;DEC POINT
    222 001050  001217                          BNE     700$            ;MUST EQUAL
    223 001052                                  SAVE    R2,R4
    224 001056  016704  176736                  MOV     HSZD,R4         ;R4=SIZE OF HIGH DECIMAL STRING
    225 001062  001612                          BEQ     700$            ;CAN'T BE 0
    226 001064  016702  176726                  MOV     LSZD,R2         ;R2=SIZE OF LOW DECIMAL STRING
    227 001070  001411                          BEQ     67$             ;IF 0, FINISHED
    228 001072  122325                  63$:    CMPB    (R3)+,(R5)+
    229 001074  103407                          BLO     67$             ;LOW<HIGH
    230 001076  101117                          BHI     1700$           ;LOW>HIGH
    231 001100  005267  176722                  INC     LHCSZ           ;ANOTHER COMMON
    232 001104  005304                          DEC     R4
    233 001106  001513                          BEQ     1700$           ;HIGH STRING CAN'T END NOW
    234 001110  005302                          DEC     R2
    235 001112  001367                          BNE     63$             ;IF MORE CHAR'S IN LOW STRING
    236 001114                          67$:    RESTOR  R2,R4
    237 001120                          20$:    RESTOR  R3,R5
    238 001124                                  SAVE    R4
    239 001126  005767  176670                  TST     DECFLG
    240 001132  001403                          BEQ     200$
    241 001134  052767  000010  176666          BIS     #DNUM,RNGCDE    ;IF DECIMAL NUMBER, NOTE IN RANGE CODE
    242 001142  026727  176656  000001  200$:   CMP     SGNIND,#1       ;TEST SIGN INDICATOR
    243 001150  002403                          BLT     201$            ;IF NO SIGN
    244 001152  001414                          BEQ     203$            ;IF MULTI-RANGES
    245 001154  112724  000055                  MOVB    #'-,(R4)+       ;IF BOTH NEGATIVE, XFER SIGN
    246 001160  112724  000000G         201$:   MOVB    #ZVLDC,(R4)+    ;XFER ZVLDC
    247 001164  016700  176636                  MOV     LHCSZ,R0
    248 001170  001405                          BEQ     203$            ;NO COMMON
    249 001172  052767  000040  176630          BIS     #CM,RNGCDE      ;COMMON, FLAG IT
    250 001200  112324                  202$:   MOVB    (R3)+,(R4)+     ;XFER COMMON
    251 001202  077002                          SOB     R0,202$
    252 001204  112724  000000G         203$:   MOVB    #SEGOP,(R4)+    ;XFER SEGMENT OPERATOR
    253 001210  116724  176614                  MOVB    RNGCDE,(R4)+    ;XFER RANGE CODE
    254 001214  112324                  204$:   MOVB    (R3)+,(R4)+     ;XFER LOW
    255 001216  100376                          BPL     204$
    256 001220  010503                          MOV     R5,R3           ;XFER HIGH
    257 001222  066703  176600                  ADD     LHCSZ,R3
    258 001226  112324                  205$:   MOVB    (R3)+,(R4)+
    259 001230  100376                          BPL     205$
    260 001232  012603                          MOV     (SP)+,R3        ;SET UP TO OVERWRITE NODE WITH REFORMATED
    261 001234  160304                          SUB     R3,R4           ;       NUMBER RANGE VALUES
    262 001236  010400                          MOV     R4,R0
    263 001240  012604                          MOV     (SP)+,R4
    264 001242  112324                  206$:   MOVB    (R3)+,(R4)+
    265 001244  077002                          SOB     R0,206$
    266 001246                                  SAVE    R1              ;CHECK IF NEXT CHAR A NUMBER
    267 001250  012702  000000G                 MOV     #FLUTBL,R2      ;SWITCH BACK TO FLU TABLE
    268 001254  004767  000000G                 JSR     PC,STEP
    269 001260                                  CHECK   CSTTS!CSTNF,30$                 ;LEGAL
    270 001266                                  CHECK   CSTVDC!CSTFDC!CSTSO!CSTDW,1701$ ;ILLEGAL
    271 001274                                  CHECK   CSTESC,31$              ;IF ELSE, ASSUME SEARCHABLE
    272 001302  005301                          DEC     R1                      ;IF SEARCHABLE, RETEST FOR NUMBER
    273 001304  012702  000000G         31$:    MOV     #LEXTB3,R2              ;IF ESCAPE, CHECK NEXT FOR NUMBER
    274 001310  004767  000000G                 JSR     PC,STEP
    275 001314                                  CHECK   CSTNUM,1701$            ;IF NUMBER ILLEGAL
    276 001322  012702  000000G                 MOV     #FLUTBL,R2
```

```
NUMRNG NUMBER RANGE CODE

    277 001326                              30$:    RESTOR  R1
    278 001330  000207                             RTS     PC
    279 001332  000167  177164          1701$:  JMP     701$
    280 001336  000167  177146          1700$:  JMP     700$
```

```
 282·                                   ;
 283                                    ;  SUBROUTINE·TO·CHECK·A·RANGE FOR·CHAR·AND·TERM·COUNT·
 284                                    ;
 285 001342·  005003                    CHKRNG: CLR·    R3              ;R3=CHAR COUNT·
 286 001344   005305                    1$:     DEC·    R5              ;R5=INPUT·CHAR·COUNT·
 287 001346   112401                            MOVB·   (R4)+,R1        ;GET·CHAR·FROM·NODE·
 288 001350   100440                            BMI·    4$              ;IF·SEGOP·
 289 001352·  122701  000000G·                   CMPB·   #DECPT,R1               ;IS·CHAR·A DEC·POINT?·
 290 001356   001026                            BNE·    3$              ;NO·
 291 001360                                     SAVE·   R3              ;YES·
 292·001362·  026627  000004  000012'           CMP·    4(SP),#LSZ·     ;IS·RANGE·LOW·OR·HIGH?·
 293 001370   001003                            BNE·    2$              ;HIGH·
 294 001372·  012746  000016'                   MOV·    #LSZD,-(SP)     ;LOW: CHKRNG·FOR·LOW·DECIMAL
 295 001376   000402·                           BR      20$
 296 001400   012746  000020'           2$:     MOV·    #HSZD,-(SP)     ;CHKRNG·FOR·HIGH·DECIMAL·
 297 001404   004767  177732            20$:    JSR·    PC,CHKRNG·
 298 001410                                     RESTOR· R3
 299 001412·  126427  177776  000000G·          CMPB·   -2(R4),#DECPT·  ;LAST·CHAR·A·DEC·POINT?·
 300 001420   001021                            BNE·    41$             ;NO·
 301 001422·  126467  177775  176376           CMPB·   -3(R4),NUMVLU·  ;YES: WAS·PRIOR·THE·SPECIAL·NUMBER?·
 302·001430   001015                            BNE·    41$             ;NO·
 303 001432·  000413                            BR      42$             ;YES·
 304 001434'  005203                    3$:     INC·    R3              ;COUNT·CHAR·
 305 001436   126701  176364                   CMPB·   NUMVLU,R1       ;SPECIAL CHAR?·
 306 001442·  001401                            BEQ·    31$             ;YES: COUNT·1
 307 001444   005202·                           INC·    R2·             ;NO: COUNT·2·
 308 001446   005202·                   31$:    INC·    R2·
 309 001450   000735                            BR      1$
 310 001452·  126467  177776  176346   4$:     CMPB·   -2(R4),NUMVLU·  ;WAS·LAST·CHAR·SPECIAL·NUMBER?·
 311 001460   001001                            BNE·    41$             ;NO·
 312·001462·  005202·                   42$:    INC·    R2·             ;YES: COUNT·2·
 313 001464   160302·                   41$:    SUB·    R3,R2·          ;TERM·COUNT·=·COUNT·-·CHAR·SIZE·
 314 001466   010376  000002                   MOV·    R3,@2(SP)       ;STORE·COUNT·
 315 001472·  012616                            MOV·    (SP)+,(SP)      ;POP·PARAMETER·FROM·STACK·
 316 001474   000207                            RTS·    PC·
```

```
                                        ;
                                        ;  SUBROUTINE TO CREATE EMATRIX ENTRIES FOR NUMERICAL RANGE.
                                        ;
                                        ;        INPUT.                     OUTPUT.
                                        ;  R0=.  EMX PTR.                   EMX PRT (AFTER RANGE)
                                        ;  R1=.  ..........SCRATCH.....................
                                        ;  R2=.  (SAVED)                   (RESTORED)
                                        ;  R3=.  (SAVED)                   (RESTORED)
                                        ;  R4=.  NODE CHAR PTR.            NODE CHAR PTR AFTER RANGE.
                                        ;  R5=.  #.CHAR'S LEFT IN NODE.    #.CHAR'S LEFT IN NODE (AFTER RANGE)
                                        ;
001476                          NUMRNG::SAVE    R2,R3
001502  005067  176316                  CLR     INCLUD
001506  016701  000000G                 MOV     VINXT,R1            ;CHECK IF VECTORS SHOULD BE INCLUDED
001512  016102  177776G                 MOV     VI-2(R1),R2         ;   IN INITIAL VECTOR.
001516  062702  000000G                 ADD     #EMX,R2             ;R2=LAST EMX REF IN INIT VECTOR
001522  020002                         CMP     R0,R2               ;R0=CURRENT EMX LOC
001524  001002                         BNE     6$                  ;DON'T INCLUDE
001526  005267  176272                  INC     INCLUD              ;INCLUDE
001532  005002                  6$:     CLR     R2                  ;R2=MAX # TERMS
001534  112467  176270                  MOVB    (R4)+,RNGCDE        ;GET RANGE CODE
001540  005305                         DEC     R5
001542  005067  176270                  CLR     DPTR                ;INITIALIZE MERGE POINTERS
001546  005067  176266                  CLR     NVPTR
001552  005067  176254                  CLR     NFPTR
001556  005067  176252                  CLR     NFIPTR
001562  032767  000100  176240          BIT     #MR,RNGCDE          ;IS THIS A MULTI-RANGE?
001570                                  BOFF    1000$               ;NO
001572  052767  000400  176230          BIS     #MR1,RNGCDE         ;YES; FLAG AS 1ST PASS
001600  032767  000010  176222          BIT     #DNUM,RNGCDE        ;IF NOT DECIMAL, FAKE LOW ENTRY
001606                                  BON     9$
001610  012767  000001  176174          MOV     #1,LSZ
001616  012767  000052' 176154          MOV     #ZERO,LBEG
001624  005202                         INC     R2
001626  000404                         BR      19$
001630  000167  000356          1000$:  JMP     1$
001634  005067  176152          9$:     CLR     LSZ                 ;INIT LOW OF 1ST RANGE
001640  005067  176152          19$:    CLR     LSZD
001644  005767  176154                  TST     INCLUD              ;INCLUDE IN INIT.?
001650  001011                         BNE     92$                 ;YES
001652  012720  000000C                 MOV     #EMXNSQ!EMXEXF!4,(R0)+  ;CREATE EXPANSION VECTOR
001656  010010                         MOV     R0,(R0)
001660  062720  000000C                 ADD     #4-EMX,(R0)+        ;ENTER 1ST ENTRY
001664  010046                         MOV     R0,-(SP)            ;SAVE LOC IN EXP VEC FOR 2RD ENTRY
001666  062700  000002                 ADD     #2,R0
001672  000405                         BR      7$
001674  052767  000004  176126  92$:    BIS     #INCLD,RNGCDE       ;NOTE 2RD RANGE SHOULD ALSO BE INCLUDED
001702  005067  176116                  CLR     INCLUD
001706  012701  000000G          7$:     MOV     #MINEMX,R1          ;ENTER MINUS IN EMX
001712  012120                         MOV     (R1)+,(R0)+
001714  012120                         MOV     (R1)+,(R0)+
001716  012120                         MOV     (R1)+,(R0)+
001720  012701  000000G          17$:    MOV     #ZVDEMX,R1          ;ENTER ZVLDC
001724  012120                         MOV     (R1)+,(R0)+
001726  012120                         MOV     (R1)+,(R0)+
001730  012120                         MOV     (R1)+,(R0)+
001732  122714  000000G                 CMPB    #DECPT,(R4)
```

```
                        ;THIS FILE (E.MAC) CONTAINS ALL THE QUERY ERROR CODES.
                        ;
000001                  QE.DWJ=1        ;RESERVED.
000002                  QE.MTS=2        ;MULTIPLE TERM SEP'S OR MISSING TERM.
000003                  QE.DWJ=3        ;RESERVED.
000004                  QE.ILC=4        ;ILLEGAL CHARACTER.
000005                  QE.IHS=5        ;INTERNAL HSTS ERROR, LOGIC PROCESSING.
000006                  QE.NLQ=6        ;NULL QUERY.
000007                  QE.MOP=7        ;MISSING OPERATOR.
000010                  QE.MFM=8.       ;MISPLACED FLU-MOD.
000011                  QE.MSD=9.       ;MISPLACED SUBDOC OPERATOR.
000012                  QE.MNT=10.      ;MISPLACED NOT OPERATOR.
000013                  QE.MFL=11.      ;MISSING FLU / TWO CONSECUTIVE OPERATORS.
000014                  QE.UBP=12.      ;UNBALANCED PARENTHESES.
000015                  QE.INO=13.      ;INSUFFICIENT NUMBER OF OPERANDS.
000016                  QE.ISO=14.      ;ILLEGAL SUBELEMENT OPERAND.
000017                  QE.PX1=15.      ;MISSING PROX, WINDOW, OR DOC TYPE,ZONE, OR SUBZONE.
000020                  QE.PX2=16.      ;PROX, WINDOW OR FLU-MOD ID TOO LARGE.
000021                  QE.PX3=17.      ;MISSING PROX, UNIT.
000022                  QE.PX4=18.      ;MISSING PROX, DELIMITER
000023                  QE.NRI=19.      ;ILLEGAL NUMERICAL RANGE SPECIFICATION.
000024                  QE.NRB=20.      ;NUMERICAL RANGE BORDERED BY NUMERIC.
000025                  QE.NOS=21.      ;NO STX FOUND IN QUERY.
000026                  QE.FTB=22.      ;FLU TOO BIG.
000027                  QE.NTK=23.      ;UNDEFINED TOKEN.
                        ;
                        ;QE.RO1=100.    ;GENERAL RESOURCE OVERFLOW (LABEL DEFINED IN M.MAC)
000145                  ASKOVF=101.     ;ARGUMENT STACK OVERFLOW
000146                  LNPOVF=102.     ;LOGIC NODE POOL OVERFLOW.
000147                  TSKOVF=103.     ;TOKEN STACK OVERFLOW.
000150                  QBFOVF=104.     ;QUERY TOO BIG
000156                  QRYOVF=110.     ;# QUERIES OVERFLOWED.
000157                  POLOVF=111.     ;FLU POOL OVERFLOW.
000160                  FALOVF=112.     ;FAL
000161                  TTBOVF=113.     ;TTABLE.
000162                  XTBOVF=114.     ;XTABLE.
000163                  QLBOVF=115.     ;QLB
000164                  SLBOVF=116.     ;SDLB.
000165                  QEXOVF=117.     ;QEX
000166                  EMAOVF=118.     ;EMA
000167                  EMBOVF=119.     ;EMB
000170                  EMCOVF=120.     ;EMC
000171                  VI3OVF=121.     ;VI QT3
000172                  VI2OVF=122.     ;VI QT2.
000173                  VI1OVF=123.     ;VI QT1
000174                  TDBOVF=124.     ;TDCTB.
000175                  NDBOVF=125.     ;NODE POOL QT2
                        ;
```

```
                        ;
                        ; MACRO TO PRINT BUFFER
                        ; CREATES SPOOL FILE RECORDS
                        ;
                        .MACRO  PRT,START,END
                                MOV     R4,-(SP)
                                MOV     START,R4
                                MOV     R4,LOCAT
                                MOV     END,ENDLOC
                                JSR     PC,PRINT
                                MOV     (SP)+,R4
                        .ENDM
                        .MACRO  PRTH,START,END
                                MOV     R4,-(SP)
                                MOV     START,R4
                                MOV     R4,LOCAT
                                MOV     END,ENDLOC
                                JSR     PC,PRNTH
                                MOV     (SP)+,R4
                        .ENDM
                        ;
                        ; MACRO TO TEST CHARACTER SIGNIFICANCE
                        ; PARAMETERS ARE CONDITION,TRUE,AND FALSE
                        ;       CONDITION=THE TEST CONDITION
                        ;       TRUE    =PATH TO TAKE IF CONDITION TRUE
                        ;       FALSE   =PATH TO TAKE IF CONDITION FALSE
                        ; ANY PARAMETER CAN BE PROCEEDED BY "!" WHICH CAUSES THAT
                        ;  PARAMETER TO BE A SUBROUTINE CALL
                        ;
```

```
                                        .PSECT  FLUCDE
                                        .SBTTL  FLU CODE
                                ;
                                ; MISCELLANEOUS
                                ;
000000  000000                  CMPOFF: .WORD   0       ;CMP OFFSET IN NODE FOR SCAN SUBROUTINE
                                ;
000002  000000                  FLUTYP::.WORD   0       ;FLU TYPE FLAGS WORD FOR FSA DISCRIMINATION
        000001                  ONYVDC=1                ;VLDC ONLY
        000002                  TRLVDC=2                ;TRAILING VLDC
        000004                  MBDVDC=4                ;IMBEDED VLDC
        000010                  INIVDC==10              ;INITIAL VLDC
        000020                  FDC=20                  ;FLDC
        000040                  SRCHBL=40               ;SEARCHABLE
        000100                  NMRNGF==100             ;NUMERICAL RANGE
                                ;
        040000                  TERMF=BIT14             ;FLAG BIT TO SIGNIFY FLU IS NOT A CWP TERM
        100000                  CWPF=BIT15              ;FLAG BIT TO SIGNIFY FLU IS A CWP TERM
000004  000000                  TYPFLU: .WORD   0       ;TEMP STORAGE FOR ABOVE FLAGS
                                ;
000006  000000                  SLBEG:: .WORD   0       ;FLU MOD ADDRESS OF SUB-LEVEL FOR SORTING
000010  000000                  FM1E:   .WORD   0       ;END OF UNSORTED FLU MOD
000012  000000                  FM2S:   .WORD   0       ;START OF SORTED FLU MOD
000014  000000                  FM2C:   .WORD   0       ;NEXT ADDRESS IN SORTED FLU MOD
                                ;
        000300                  DTFLG   =300    ;DOC TYPE MOD ID
        000200                  ZFLG    =200    ;ZONE MOD ID
        000100                  SZFLG   =100    ;SUB-ZONE MOD ID
```

```
                                        ;
                                        ; SUBROUTINE TO PARSE AND PROCESS FLU'S
                                        ;
                                        ;                       INPUT                           OUTPUT
                                        ;       R0=     SCRATCH                         SCRATCH
                                        ;       R1=     2ND CHAR ADR OF FLU             CHAR ADR AFTER FLU
                                        ;       R2=     (SAVED)                         (RESTORED)
                                        ;       R3=     SCRATCH                         FLU ID
                                        ;       R4=     (SAVED)                         (RESTORED)
                                        ;       R5=     (SAVED)                         (RESTORED)
                                        ;
000016  000167  002744          ERR1J:  JMP     ERR1
000022  000167  002752          ERR2J:  JMP     ERR2
000026                          PFLU::  SAVE    R2,R4,R5
000034  012767  040000  177742          MOV     #TERMF,TYPFLU   ;FLAG TERM AS NON-CWP
000042  012702  000000G                 MOV     #FLUTBL,R2      ;R2=CST TABLE FOR FLU PARSING
000046  005301                          DEC     R1              ;RESET TO 1ST CHAR ADR OF FLU
000050                                  CALL    STEP            ;SET SIGNIFICANCE OF 1ST CHAR
000054                          1$:     CHECK   CSTDW,!STEP,10$ ;IF DUMMY WORD STEP TO NEXT CHAR
000066                                  CHECK   CSTTS,!STEP,ERR1J       ;IF TERM SEPARATOR,STEP; IF NOT,ERROR
000100  000765                          BR      1$              ;CONTINUE FLUSHING LEADING DUMMY WORDS
000102                          10$:    CHECK   CSTTS!CSTNF,ERR2J,!PTERM        ;IF NOT TERM SEP OR NON-FLU
                                                                ;THEN MUST BE TERM, SO PROCESS
000114                                  CHECK   CSTTS,,500$     ;IF NOT TERM SEP, MUST BE END OF FLU
000122  016704  000000G                 MOV     NXTNDE,R4       ;CWP TERM, SO ALLOCATE CWP FLU NODE
000126  010467  000000G                 MOV     R4,CWPNDE
000132  005267  000000G                 INC     FLUID
000136  062704  000004                  ADD     #4,R4
000142  012724  002000                  MOV     #2000,(R4)+             ;TYP/SIZ
000146  016724  000000G                 MOV     FLUID,(R4)+             ;FLUID
000152  010324                          MOV     R3,(R4)+                ;1ST TERM ID
000154                                  CALL    STEP            ;SET UP NEXT CHAR
000160                                  SAVE    R0,R1
000164  012703  000002                  MOV     #2,R3           ;CURRENT NUMBER OF TERMS IN CWP
000170                          11$:    CHECK   CSTNF,13$       ;NON-FLU FOUND, FINISHED ALLOCATION PRE-SCAN
000176                                  CHECK   CSTTS,12$       ;IF TERM-SEP, COUNT
000204                                  CHECK   CSTESC,100$     ;IF ESCAPE, SKIP NEXT CHAR
000212                                  CHECK   CSTSO,101$,!STEP        ;IF SEG OPER,SKIP SIGNS,ELSE CONTINUE
000224  000761                          BR      11$
000226  005201                  100$:   INC     R1              ;SKIP CHAR AFTER ESCAPE
000230                                  CALL    STEP
000234  000755                          BR      11$
000236  000167  000334          500$:   JMP     50$
000242  012705  000002          101$:   MOV     #2,R5           ;# SEG OPER'S TO CONSIDER
000246  012702  000000G                 MOV     #LEXTB3,R2      ;SWITCH TO NUMBER RANGE LEX TABLE
000252                          102$:   CALL    STEP            ;CHECK CHAR AFTER SEG OP
000256                                  CHECK   CSTNUM!CSTDP!CSTMIN,102$        ;IF NUM RANGE, CONTINUE
000264  005305                          DEC     R5
000266  001371                          BNE     102$            ;KEEP LOOKING FOR END OF NUM, RNG.
000270  012702  000000G                 MOV     #FLUTBL,R2      ;SWITCH BACK TO FLU LEX TABLE
000274                                  CALL    STEP            ;IF 0, OUT OF NUMERICAL RANGE
000300  000733                          BR      11$
000302  005203                  12$:    INC     R3              ;ADD ONE MORE TERM TO CWP TERM COUNT
000304                                  CALL    STEP            ;STEP & CONTINUE
000310  000727                          BR      11$
000312  010300                  13$:    MOV     R3,R0           ;R0=CWP NODE BYTE COUNT
000314  006300                          ASL     R0
```

```
FLU.CODE.

 87 000316  032700  177400                    BIT.    #177400,R0
 88 000322. 001402.                          BEQ.    130$
 89 000324  000167  002506                    JMP.    ERR21           ;IF.CWP.> 256. , ERROR.TOO.BIG!!!
 90 000330  016703  000000G.          130$:   MOV.    CWPNDE,R3       ;STORE.COUNT.
 91 000334  110063  000000G.                  MOVB.   R0,NDSIZ(R3)
 92.000340  062700  000000G.                  ADD.    #NDTRM,R0       ;PRE-ALLOCATE.NODE
 93 000344  060067  000000G.                  ADD.    R0,NXTNDE.
 94 000350                                   RESTORE.R0,R1
 95 000354                          20$:     CHECK.  CSTNC!CSTVDC!CSTFDC!CSTESC!CSTSO,21$    ;CHECK.FOR.TERM.
 96 000362.                                  CHECK.  CSTNF,ERR2J.    ;CHECK.FOR.NON-FLU
 97 000370                                   CHECK.  CSTDW,22$,ERR2J.;CHECK.FOR.DUMMY.WORD, IF.NOT.THEN.ILLEGAL.
 98 000400                          21$:     CALL.   PTERM.          ;PROCESS TERM.
 99 000404  010324                          MOV.    R3,(R4)+        ;RECORD.TERM.ID.IN CWP.FLU.NODE.
100 000406                                   CHECK.  CSTTS,211$      ;MORE.CWP?.
101 000414                                   CHECK.  CSTNF,30$,ERR3J.;CWP.END OR.ILLEGAL.
102.000424                          211$:    CALL.   STEP            ;SET.UP.NEXT.CHAR.SIGNIFICANCE.
103 000430  000751                          BR      20$
104 000432. 005024                  22$:     CLR.    (R4)+           ;ENTER.DUMMY.WORD.TERM.ID.IN CWP.NODE.
105 000434                                   CALL.   STEP
106 000440                                   CHECK.  CSTTS,211$      ;MORE.CWP?.
107 000446                                   CHECK.  CSTNF,221$,ERR3J.       ;CWP.END.OR.ILLEGAL.
108 000456  005744                  221$:    TST.    -(R4)           ;REMOVE.TRAILING.DUMMY.WORDS
109 000460  001776                          BEQ.    221$
110 000462. 005724                          TST.    (R4)+           ;ALL.REMOVED.
111 000464  016703  000000G.          30$:    MOV.    CWPNDE,R3       ;GET.NODE.ADR.
112.000470  062703  000000G.                  ADD.    #NDTRM,R3       ;CALCULATE.INDEX.
113 000474  011303                          MOV.    (R3),R3
114 000476  042703  000000G.                  BIC.    #CPIXMK,R3
115 000502. 006303                          ASL.    R3
116 000504  062703  000000G.                  ADD.    #CWPIDX,R3      ;R3=POOL HEADER(A.CWP.INDEX.WORD)
117 000510  012767  000000G. 177262.         MOV.    #NDTRM,CMPOFF.  ;STORE.OFFSET.IN.NODE.OF.TERM.ID'S.
118 000516  016702. 000000G.                  MOV.    CWPNDE,R2.      ;ADR.OF.NODE.TO.SCAN.POOL.FOR.
119 000522.                                  CALL.   SCAN            ;SCAN.FOR.NODE.
120 000526  103410                          BCS.    40$             ;IF.NODE NOT.FOUND AND.INSERTED.IN.POOL?.
121 000530  016767  000000G. 000000G.        MOV.    CWPNDE,NXTNDE.  ;NODE.FOUND, SO.DEALLOCATE.THIS.DUPLICATE.
122.000536  005367  000000G.                  DEC.    FLUID.
123 000542. 016403  000000G.                  MOV.    NDFLID(R4),R3   ;RETURN.ORIGINAL.FLU.ID.OF.FLU.
124 000546  000413                          BR      50$
125 000550  016403  000000G.          40$:    MOV.    NDFLID(R4),R3   ;RETURN.FLU.ID.OF.NEW.FLU.
126 000554  005267  000000G.                  INC.    NODEC.          ;KEEP.COUNT.OF.NODES.
127 000560  116404  000000G.                  MOVB.   NDSIZ(R4),R4    ;GET.NODE.SIZE.
128.000564  042704  177400                    BIC.    #177400,R4
129 000570  006204                          ASR.    R4
130 000572. 060467  000000G.                  ADD.    R4,EMCQSZ.      ;ACCUMULATE.SIZE.OF.EMC.(AND TDCTC)
131 000576  005301                  50$:     DEC.    R1              ;RESET.TO.CHAR.AFTER.FLU.
132.000600  016704  000000G.                  MOV.    NFALE,R4        ;ENTER.THIS.FLU.IN FAL.
133 000604  010302.                          MOV.    R3,R2.
134 000606  042702. 140000                    BIC.    #140000,R2.
135 000612. 020427  000000C.                  CMP.    R4,#FAL+FALMSZ  ;FAL.OVERFLOW?.
136 000616  103011                          BHIS.   51$             ;YES.
137 000620  010224                          MOV.    R2,(R4)+
138 000622. 005277  000000G.                  INC.    @CFALS.
139 000626  010467  000000G.                  MOV.    R4,NFALE.
140 000632.                                  RESTORE.R2,R4,R5
141 000640  000207                          RTS.    PC.
142.000642. 012767  000160  000000G. 51$:     MOV.    #FALOVF,ERRCDE
143 000650  000167  000000G.                  JMP.    ERRORV.         ;NON-FATAL.OVERFLOW.
```

```
144 000654  000167  002132          ERR3J:  JMP     ERR3
145 000660  000167  002140          ERR4J:  JMP     ERR4
146                                 ;
147                                 ; SUBROUTINE TO PARSE AND PROCESS TERMS.
148                                 ;
149                                 ;               INPUT.                          OUTPUT.
150                                 ;       R0=.    CHAR SIGNIFICANCE(1ST CHAR)     CHAR SIGNIF(CHAR AFTER TERM)
151                                 ;       R1=.    2ND CHAR ADR OF TERM.           2ND CHAR ADR AFTER TERM.
152                                 ;       R2=.    CST TABLE ADR FO FLU PROCESSING. (SAME)
153                                 ;       R3=.    SCRATCH.                        TERM ID OF TERM.
154                                 ;       R4=.    (SAVED)                         (RESTORED)
155                                 ;
156 000664                          PTERM:  SAVE    R2,R4
157 000670  005067  000000G.               CLR     VDCCNT.         ;INIT # VLDC'S IN TERM.
158 000674  005067  177102                 CLR     FLUTYP.         ;FLUTYP WILL BE USE TO DETERMINE FLU TYPE.
159 000700  016704  000000G.               MOV     NXTNDE,R4       ;ALLOCATE TERM NODE.
160 000704  010467  000000G.               MOV     R4,FLUNDE.
161 000710  005267  000000G.               INC     FLUID.
162 000714  062704  000004                 ADD     #4,R4
163 000720  005024                        CLR     (R4)+                   ;TYP/SIZ
164 000722  016724  000000G.               MOV     FLUID,(R4)+             ;FLUID
165 000726  005024                        CLR     (R4)+                   ;FLU MOD NODE ADR
166 000730  005024                        CLR     (R4)+                   ;FLU MOD EXTENSION NODE ADR
167 000732  116124  177777                MOVB    -1(R1),(R4)+            ;1ST CHAR
168 000736                          10$:    CHECK   CSTNC,17$       ;NON-CONTROL (SEARCHABLE)?
169 000744                                  CHECK   CSTNF!CSTDW!CSTTS,20$   ;NON-TERM?
170 000752                                  CHECK   CSTFDC,11$      ;FLDC?
171 000760                                  CHECK   CSTVDC,12$      ;VLDC?
172 000766                                  CHECK   CSTSO,13$       ;SEGMENT OPERATOR (NUMERICAL RANGE)?
173 000774                                  CHECK   CSTESC,14$,ERR4J        ;ESCAPE OR ILLEGAL?
174 001004  112764  000000G 177777  11$:    MOVB    #FLDC,-1(R4)    ;REPLACE CHAR IN NODE WITH FLDC INDICATOR
175 001012  052767  000020  176762.        BIS     #FDC,FLUTYP.    ;NOTE OCCURENCE
176 001020  000526                        BR      17i$
177 001022  005304                 12$:    DEC     R4              ;REMOVE LAST CHAR FROM NODE
178 001024                          120$:   CALL    STEP            ;SET UP NEXT CHAR
179 001030                                  CHECK   CSTFDC,121$     ;FLDC?
180 001036                                  CHECK   CSTVDC,120$,125$;VLDC? OR NON-DON'T-CARE?
181 001046  112724  000000G.        121$:   MOVB    #FLDC,(R4)+     ;TRANSFER VLDC INDICATOR
182 001052  052767  000020  176722.        BIS     #FDC.FLUTYP.    ;NOTE OCCURENCE
183 001060  000761                        BR      120$
184 001062                          125$:   CHECK   CSTTS!CSTNF!CSTDW,.126$ ;IF NON-TERM,NEXT; IF TERM,126$
185 001070  052767  000002  176704        BIS     #TRLVDC,FLUTYP  ;NOTE TRAILLING VLDC
186 001076  032767  000040  176676        BIT     #SRCHBL,FLUTYP  ;HAS ANYTHING PRIOR BEEN NOTED?
187 001104                                BOFF    1251$           ;NO
188 001106  112724  000000G.            MOVB    #VLDC,(R4)+     ;YES; SO JUST TRANSFER VLDC INDICATOR
189 001112  005204                        INC     R4              ;ADJUST R4 (NORMALLY AT 20$ EXTRA CHAR WAS XFER)
190 001114  005267  000000G.            INC     VDCCNT.         ;COUNT VLDC'S
191 001120  000474                        BR      20$
192 001122  032767  000020  176652  1251$: BIT    #FDC,FLUTYP.    ;HAS A FLDC OCCURRED?
193 001130                                BON     1252$           ;YES; THEN DON'T ENTER EXTRA
194 001132  112724  000000G.            MOVB    #FLDC,(R4)+     ;NO; THEN DON'T LET VLDC STAND ALONE
195 001136  042767  000016  176636  1252$: BIC   #TRLVDC!MBDVDC!INIVDC,FLUTYP.    ;NOTE AS FLDC-VLDC ONLY
196 001144  052767  000021  176630        BIS     #FDC!ONYVDC,FLUTYP.
197 001152  112724  000000G.            MOVB    #VLDC,(R4)+     ;TRANSFER VLDC
198 001156  005204                        INC     R4              ;ADJUST R4 FOR 20$
199 001160  005267  000000G.            INC     VDCCNT.         ;COUNT VLDC'S
200 001164  000452.                     BR      20$
```

```
  201 001166  005301                          126$:   DEC     R1                      ;RESET TO REPROCESS CHAR
  202 001170  112724  000000G                         MOVB    #VLDC,(R4)+             ;TRANSFER VLDC INDICATOR
  203 001174  005267  000000G                         INC     VDCCNT                  ;COUNT VLDC'S
  204 001200  032767  000060  176574                  BIT     #SRCHBL!FDC,FLUTYP      ;ANYTHING PRIOR NOTED?
  205 001206                                          BEQ     1261$                   ;YES
  206 001210  052767  000010  176564                  BIS     #INIVDC,FLUTYP  ;NO: SO NOTE AS INITIAL VLDC
  207 001216  000424                                  BR      17$
  208 001220  052767  000004  176554  1261$:          BIS     #MBDVDC,FLUTYP  ;NOTE AS IMBEDED VLDC
  209 001226  000420                                  BR      17$
  210 001230  052767  000100  176544  13$:           BIS     #NMRNGF,FLUTYP  ;NOTE AS NUMERICAL RANGE
  211 001236                                          CALL    PNBRNG                  ;PARSE NUMBER RANGE
  212 001242  000412                                  BR      17$
  213 001244  000167  001554          172$:           JMP     ERR4
  214 001250  111164  177777          14$:            MOVB    (R1),-1(R4)     ;REPLACE CHAR WITH NEXT
  215 001254                                          CALL    STEP
  216 001260                                          CHECK   CSTCET,17$,172$ ;CHAR IS EITHER SEARCHABLE(CET ENTRY) OR ILLEGAL
  217 001270  052767  000040  176504  17$:           BIS     #SRCHBL,FLUTYP  ;NOTE AS SEARCHABLE
  218 001276  111124                  171$:           MOVB    (R1),(R4)+      ;TRANSFER AND SET UP NEXT CHAR
  219 001300                                          CALL    STEP
  220 001304                                          CHECK   CSTNF!CSTTS!CSTDW,,10$  ;IF TERM,NEXT; IF NON-TERM,10$
  221 001312  010467  000000G         20$:            MOV     R4,NXTNDE       ;STORE NEXT FREE NODE ADR
  222 001316  042767  000001  000000G                 BIC     #1,NXTNDE       ;LAST CHAR WAS JUNK, SO JUST TRUNCATE
  223 001324  005304                                  DEC     R4
  224 001326  016703  000000G                         MOV     FLUNDE,R3       ;CALCULATE TERM SIZE
  225 001332  160304                                  SUB     R3,R4
  226 001334  162704  000000G                         SUB     #NDCHR,R4
  227 001340  062703  000000G                         ADD     #NDSIZ,R3
  228 001344  032704  177400                          BIT     #177400,R4      ;BE SURE IT HASN'T OVERFLOWED BYTE
  229 001350                                          BNE     71$             ;IT HAS
  230 001352  110423                                  MOVB    R4,(R3)+        ;TRANSFER TERM SIZ TO NODE
  231 001354  032767  000115  176420                  BIT     #ONYVDC!MBDVDC!INIVDC!NMRNGF,FLUTYP     ;WHICH FSA?
  232 001362                                          BNE     201$            ;FSA-B
  233 001364  112713  000000G                         MOVB    #NDFSAA,(R3)    ;FSA-A
  234 001370  005267  000000G                         INC     NODEA           ;KEEP COUNT OF NODES
  235 001374  010467  000000G                         MOV     R4,CHRCNT       ;AND CHAR COUNT
  236 001400  012767  177777  000000G                 MOV     #-1,VDCCNT      ;FLAG AS FSA-A TERM
  237 001406  000406                                  BR      202$
  238 001410  112713  000000G         201$:           MOVB    #NDFSAB,(R3)
  239 001414  005267  000000G                         INC     NODEB           ;KEEP COUNT OF NODES
  240 001420  010467  000000G                         MOV     R4,CHRCNT       ;AND CHAR COUNT
  241 001424  062703  000000C         202$:           ADD     #NDCHR-NDSIZ-1,R3
  242 001430  111303                                  MOVB    (R3),R3         ;CALCULATE INDEX
  243 001432  042703  000000G                         BIC     #FLIXMK,R3
  244 001436  006303                                  ASL     R3
  245 001440  062703  000000G                         ADD     #FLUIDX,R3      ;R3=POOL HEADER(A TERM INDEX WORD)
  246 001444  012767  000000G 176326                  MOV     #NDCHR,CMPOFF   ;STORE OFFSET IN NODE OF CHAR
  247 001452  016702  000000G                         MOV     FLUNDE,R2       ;R2=NODE TO SCAN FOR
  248 001456  016762  000000G 000000G                 MOV     ACTFMN,NDFMN(R2);LINK TO ACTIVE FLU MODIFIER NODE
  249 001464                                          CALL    SCAN            ;SCAN FOR NODE
  250 001470  103415                                  BCS     52$             ;IF NOT FOUND AND NODE MERGED INTO POOL
  251 001472                                          SAVE    R4
  252 001474  016767  000000G 000000G                 MOV     FLUNDE,NXTNDE   ;NODE FOUND, DEALLOCATE THIS DUPLICATE
  253 001502  005367  000000G                         DEC     FLUID
  254 001506  122764  000000G 000000G                 CMPB    #NDFSAB,NDTYP(R4)       ;FSA-B NODE?
  255 001514  001011                                  BNE     2021$                   ;NO
  256 001516  005367  000000G                         DEC     NODEB                   ;YES: ADJUST NODE COUNT
  257 001522  000410                                  BR      2022$
```

```
 258 001524  005746                    52$:    TST     -(SP)
 259 001526  004767  000000G                   JSR     PC,SZCAL1       ;CALCULATE RESOUCE UTILIZATION
 260 001532  000454                            BR      50$
 261 001534  000167  001276            71$:    JMP     ERR21
 262 001540  005367  000000G           2021$:  DEC     NODEA                   ;ADJUST NODE COUNT
 263 001544  026764  000000G 000000G   2022$:  CMP     ACTFMN,NDFMN(R4)        ;FLU FOUND, BUT DO MODIFIERS MATCH?
 264 001552  001444                            BEQ     50$             ;YES
 265 001554  005067  000000G                   CLR     CHRCNT
 266 001560  016403  000000G           30$:    MOV     NDFMEN(R4),R3   ;DON'T KNOW YET, CHECK REST OF MODIFIERS
 267 001564  001411                            BEQ     31$             ;NO (I.E. LAST MOD AND STILL NO MATCH)
 268 001566  012767  177777  000000G           MOV     #-1,CHRCNT      ;FLAG THAT EXP VEC ALREADY CREATED
 269 001574  010304                            MOV     R3,R4           ;SWITCH TO LINK
 270 001576  026764  000000G 000000G           CMP     ACTFMN,NDFMN(R4);STILL DON'T KNOW, CHECK THIS MODIFIER
 271 001604  001427                            BEQ     50$             ;YES
 272 001606  000764                            BR      30$
 273 001610  016703  000000G           31$:    MOV     NXTNDE,R3       ;CREATE LINK TO ANOTHER MODIFIER
 274 001614  062767  000006  000000G           ADD     #6,NXTNDE       ;ALLOCATE LINK NODE
 275 001622  162703  000000G                   SUB     #NDFLID,R3      ;NORMALIZE OFFSETS
 276 001626  010364  000000G                   MOV     R3,NDFMEN(R4)   ;LINK PRIOR NODE TO THIS LINK NODE
 277 001632  005267  000000G                   INC     FLUID           ;ASSIGN NEXT FLUID
 278 001636  016763  000000G 000000G           MOV     FLUID,NDFLID(R3);FILL IN FLU ID, MOD NODE, AND NEXT NODE LINK
 279 001644  016763  000000G 000000G           MOV     ACTFMN,NDFMN(R3)
 280 001652  005063  000000G                   CLR     NDFMEN(R3)
 281 001656  004767  000000G                   JSR     PC,SZCAL1       ;CALCULATE RESOURCE UTILIZATION
 282 001662  010304                            MOV     R3,R4           ;USE THIS LINK NODE TO REPORT FLUID
 283 001664                            50$:    CHECK   CSTTS,,51$      ;CHECK TO SEE IF TERM IS PART OF A CURRENT CWP
 284 001672  012767  100000  176104            MOV     #CWPF,TYPFLU    ;IT WAS, SO FLAG IT AS SO
 285 001700  056764  176100  000000G 51$:    BIS     TYPFLU,NDFLID(R4)       ;MASK TYPE FLU WITH FLU ID
 286 001706  016403  000000G                   MOV     NDFLID(R4),R3   ;RETURN TERM ID
 287 001712  042703  140000                    BIC     #140000,R3      ; BUT WITHOUT TYPE FLU BITS
 288 001716                                    RESTORE R2,R4,R4
 289 001724  000207                            RTS     PC
 290                                    ;
 291                                    ; SUBROUTINE TO SCAN NODE POOL FOR NODE AND MERGE INTO POOL IF NOT FOUND
 292                                    ;
 293                                    ;                       INPUT                           OUTPUT
 294                                    ;       R0=     ...............UNUSED:..................
 295                                    ;       R1=     ...............UNUSED:..................
 296                                    ;       R2=     NODE TO SCAN FOR                        (SAME)
 297                                    ;       R3=     POOL HEADER                             SCRATCH
 298                                    ;       R4=     SCRATCH                                 ADR NODE IN POOL(OLD OR NEW)
 299                                    ;       R5=     (SAVED)                                 (RESTORED)
 300                                    ;
 301 001726  011304                    SCAN:   MOV     (R3),R4         ;GET 1ST NODE ADR
 302 001730  001460                            BEQ     MERGE           ;NONE: MERGE WITH POOL HEADER
 303 001732                                    SAVE    R3,R5
 304 001736  116203  000000G                   MOVB    NDSIZ(R2),R3    ;R3=NODE SIZE
 305 001742  042703  177400                    BIC     #177400,R3
 306 001746  126403  000000G           1$:     CMPB    NDSIZ(R4),R3    ;CMP CURRENT NODE SIZE WITH TEST NODE
 307 001752  101043                            BHI     410$            ;PAST WITHOUT FINDING
 308 001754  001404                            BEQ     11$             ;EQUAL: CHECK FURTHER
 309 001756  011405                            MOV     (R4),R5         ;LESS THAN: STEP TO NEXT NODE
 310 001760  001442                            BEQ     401$            ;IF NO MORE NODES...
 311 001762  010504                            MOV     R5,R4           ;IF MORE NODES
 312 001764  000770                            BR      1$
 313 001766                            11$:    SAVE R2,R3,R4
 314 001774  066702  176000                    ADD     CMPOFF,R2       ;STEP TO CONTENT PORTION OF NODE
```

```
    315 002000  066704  175774                  ADD     CMPOFF,R4
    316 002004  005703                          TST     R3                      ;ZERO LENGTH COMPARES ALWAYS WORK
    317 002006  001404                          BEQ     1100$
    318 002010  122422                  110$:   CMPB    (R4)+,(R2)+             ;CMP NODE CONTENTS
    319 002012  103411                          BLO     111$                    ;LESS THAN
    320 002014  101017                          BHI     112$                    ;PAST WITHOUT FINDING
    321 002016  077304                          SOB     R3,110$                 ;EQUAL: CONTINUE UNTIL ALL CHECKED
    322 002020                          1100$:  RESTORE R2,R3,R4                ;ALL
    323 002026                                  RESTORE R3,R5                   ;THE NODE MATCHES!!!!!!!
    324 002032  000241                          CLC
    325 002034  000207                          RTS     PC
    326 002036                          111$:   RESTORE R2,R3,R4
    327 002044  011405                          MOV     (R4),R5                 ;STEP TO NEXT NODE
    328 002046  001407                          BEQ     401$                    ;IF NO MORE NODES...
    329 002050  010504                          MOV     R5,R4                   ;IF MORE NODES
    330 002052  000735                          BR      1$
    331 002054                          112$:   RESTORE R2,R3,R4
    332 002062  016404  000000G         410$:   MOV     NDBKW(R4),R4            ;BACK UP TO PREVIOUS NODE
    333 002066                          401$:   RESTORE R3,R5
    334 002072  005704                  MERGE:  TST     R4                      ;R4=POOL HEADER?
    335 002074  001410                          BEQ     10$                     ;YES
    336 002076  011403                          MOV     (R4),R3                 ;NO:ADJUST LINKAGES
    337 002100  010312                          MOV     R3,(R2)
    338 002102  010462  000000G                 MOV     R4,NDBKW(R2)
    339 002106  010214                          MOV     R2,(R4)
    340 002110  010263  000000G                 MOV     R2,NDBKW(R3)
    341 002114  000411                          BR      11$
    342 002116  010462  000000G         10$:    MOV     R4,NDBKW(R2)            ;ADJUST LINKAGES
    343 002122  011304                          MOV     (R3),R4
    344 002124  010412                          MOV     R4,(R2)
    345 002126  010213                          MOV     R2,(R3)
    346 002130  005704                          TST     R4                      ;HEADER?
    347 002132  001402                          BEQ     11$                     ;YES: NO BACK LINK
    348 002134  010264  000000G                 MOV     R2,NDBKW(R4)            ;NO: SET BACK LINK
    349 002140  010204                  11$:    MOV     R2,R4                   ;R4=MERGED NODE
    350 002142  000261                          SEC                             ;SET MERGE FLAG
    351 002144  000207                          RTS     PC
    352                                 ;
    353                                 ;
    354 002146  012404                  GETNDE::MOV     (R4)+,R4                ;GET NEXT NODE IN CHAIN
    355 002150  001004                          BNE     10$                     ;IF PRESENT
    356 002152  005305                  1$:     DEC     R5                      ;MORE POINTERS TO NODES?
    357 002154  002407                          BLT     2$                      ;NO
    358 002156  012304                          MOV     (R3)+,R4                ;YES: GET NODE FROM POINTER
    359 002160  001774                          BEQ     1$                      ;IF NULL POINTER
    360 002162  130264  000000G         10$:    BITB    R2,NDTYP(R4)            ;CORRECT TYPE?
    361 002166                                  BOFF    GETNDE                  ;NO
    362 002170  000241                          CLC                             ;YES:---THIS IS THE NEXT NODE----
    363 002172  000207                          RTS     PC
    364 002174  000261                  2$:     SEC                             ;--------NO MORE NODES------------
    365 002176  000207                          RTS     PC
    366                                 ;
    367                                 ; SUBROUTINE TO CREATE FLU MOD NODE
    368                                 ;
    369 002200                          PFLUMD::SAVE    R2,R3,R4,R5
    370 002210  012702  000000G                 MOV     #LEXTB3,R2              ;SWITCH TO NUMERICAL SIGNIFICANCE TABLE
    371 002214  016704  000000G                 MOV     NXTNDE,R4               ;CREATE FLU NODE AT NEXT POSITION IN POOL
```

```
372 002220  004767  000000G                     JSR     PC,STEP         ;GET SIGNIFICANCE
373 002224                              10$:    CHECK   CSTDT,100$      ;IF DOC TYPE
374 002232                                      CHECK   CSTZ,200$       ;IF ZONE
375 002240                                      CHECK   CSTSZ,300$      ;IF SUB-ZONE
376 002246                                      CHECK   CSTEFM,500$,700$        ;IF FLU MOD END, OR ELSE(ERROR)
377 002256  112714  000300              100$:   MOVB    #DTFLG,(R4)     ;FLAG AS DOC TYPE
378 002262  000412                              BR      400$
379 002264  012767  000004  000000G 700$:       MOV     #QE,ILC,ERRCDE  ;REPORT ILLEGAL CHARACTER
380 002272  000167  000000G                     JMP     ERROR
381 002276  112714  000200              200$:   MOVB    #ZFLG,(R4)      ;FLAG AS ZONE
382 002302  000402                              BR      400$
383 002304  112714  000100              300$:   MOVB    #SZFLG,(R4)     ;FLAG AS SUB-ZONE
384 002310  005005                      400$:   CLR     R5              ;R5 <-- BINARY ACCUMALATOR
385 002312  111103                      401$:   MOVB    (R1),R3         ;GET NUMBER
386 002314  004767  000000G                     JSR     PC,STEP         ;SET SIGNIFICANCE
387 002320                                      CHECK   CSTNUM,,410$    ;IF NOT NUMBER
388 002326  042703  177760                      BIC     #177760,R3
389 002332  070527  000012                      MUL     #10.,R5         ;SHIFT CURRENT RESULT
390 002336  060305                              ADD     R3,R5           ;AND APPEND CURRENT DIGIT
391 002340  000764                              BR      401$
392 002342  032705  177700              410$:   BIT     #177700,R5      ;FLU-MOD ID IN RANGE?
393 002346  001002                              BNE     411$
394 002350  150524                              BISB    R5,(R4)+        ;APPEND VALUE TO MOD ID AND TRANSFER
395 002352  000724                              BR      10$
396 002354  012767  000020  000000G 411$:       MOV     #QE,PX2,ERRCDE  ;REPORT RANGE ERROR
397 002362  000167  000000G                     JMP     ERROR
398 002366  012767  000000G 000000G 606$:       MOV     #NULFMN,ACTFMN  ;SET TO NULL FLU MODIFIER
399 002374  000167  000200                      JMP     605$
400 002400                              500$:   SAVE    R1
401 002402  010467  175402                      MOV     R4,FM1E         ;STORE END OF FLU MOD (UNSORTED)
402 002406  016705  000000G                     MOV     NXTNDE,R5       ;RESET TO BEGINNING OF UNSORTED FLU MOD
403 002412  010403                              MOV     R4,R3           ;CALCULATE SIZE
404 002414  160503                              SUB     R5,R3
405 002416  001763                              BEQ     606$            ;NULL LENGTH
406 002420  162703  000005                      SUB     #5,R3           ;ALLOW FOR HEADER
407 002424  100001                              BPL     501$            ;END WILL BE PASSED NODE HEADER
408 002426  160304                              SUB     R3,R4           ;START SORTED VALUES PAST HEADER(NO MOVE REQ'ED)
409 002430  010467  175356              501$:   MOV     R4,FM2S         ;SAVE START OF SORTED MODIFIERS
410 002434  010467  175354                      MOV     R4,FM2C         ;CURRENT NODE POSITION
411 002440  016703  175344                      MOV     FM1E,R3         ;R5=INPUT START, R3=INPUT END
412 002444  012700  000300                      MOV     #300,R0         ;R0=LOW, R1=HIGH, R2=IT, R4=TEST VALUE
413 002450  012701  000377                      MOV     #377,R1
414 002454  004767  000134                      JSR     PC,FMSORT       ;SORT THE FLU MODIFIERS
415 002460  016700  175326                      MOV     FM2S,R0         ;R0=START, R1=SIZE
416 002464  016701  175324                      MOV     FM2C,R1
417 002470  160001                              SUB     R0,R1
418 002472  016703  000000G                     MOV     NXTNDE,R3       ;START OF FLU MOD NODE
419 002476  062703  000000G                     ADD     #NDSIZ,R3       ;STEP TO SIZE POSITION IN HEADER
420 002502  110123                              MOVB    R1,(R3)+        ;TRANSFER SIZE
421 002504  020300                              CMP     R3,R0           ;IS SHIFTING NECESSARY
422 002506  001003                              BNE     601$            ;YES
423 002510  016703  175300                      MOV     FM2C,R3         ;NO: SET END OF NODE
424 002514  000402                              BR      602$
425 002516  112023                      601$:   MOVB    (R0)+,(R3)+     ;SHIFT NODE
426 002520  077102                              SOB     R1,601$
427 002522  016702  000000G             602$:   MOV     NXTNDE,R2       ;RESET TO BEGINNING OF NODE
428 002526  010267  175256                      MOV     R2,FM1E         ;STORE POSITION IN CASE NODE IS TO BE DELETED
```

```
    429 002532  032703  000001                  BIT     #1,R3           ;LENGTH ODD
    430 002536  001401                          BEQ     603$            ;NO
    431 002540  005203                          INC     R3              ;YES: MAKE EVEN
    432 002542  010367  000000G         603$:   MOV     R3,NXTNDE       ;ALLOCATED NODE
    433 002546  012703  000000G                 MOV     #FMDIDX,R3      ;SCAN FOR DUPLICATE
    434 002552  012767  000000G 175220         MOV     #NDFMS,CMPOFF
    435 002560  004767  177142                  JSR     PC,SCAN
    436 002564  103403                          BCS     604$            ;NO DUPLICATED,NODE MERGED
    437 002566  016767  175216  000000G         MOV     FM1E,NXTNDE     ;DUPLICATE: ELIMINATE
    438 002574  010467  000000G         604$:   MOV     R4,ACTFMN       ;SET AS ACTIVE FLU MOD
    439 002600                          605$:   RESTORE R2,R3,R4,R5,R1
    440 002612  000207                          RTS     PC
    441                                 ;
    442                                 ; SUBROUTINE TO SORT THE FLU MODIFIERS
    443                                 ;
    444 002614  010546                  FMSORT: MOV     R5,-(SP)
    445 002616  121500                          CMPB    (R5),R0
    446 002620  103002                          BHIS    2$              ;TEST MOD IS >= TO LOW CUT OFF
    447 002622  000436                          BR      12$             ;TEST MOD < LOW CUT OFF (SUB-LEVEL)
    448 002624  011605                  1$:     MOV     (SP),R5         ;RESET START RANGE
    449 002626  010102                  2$:     MOV     R1,R2           ;RESET CURRENT LOW TO HIGH RANGE
    450 002630  112504                  3$:     MOVB    (R5)+,R4        ;R4 = TEST MOD
    451 002632  042704  177400                  BIC     #177400,R4
    452 002636  020503                          CMP     R5,R3           ;AT END OF RANGE?
    453 002640  101015                          BHI     10$             ;YES
    454 002642  020400                          CMP     R4,R0           ;NO: IS IT < LOW CUT OFF
    455 002644  103771                          BLO     3$              ;YES
    456 002646  001410                          BEQ     4$              ;IT IS = TO CUT OFF, THEREFORE IT MUST BE LOWEST
    457 002650  020401                          CMP     R4,R1           ;IT IS > LOW CUT OFF, WHAT ABOUT HIGH?
    458 002652  101010                          BHI     10$             ;IT IS > HIGH CUT OFF (IT IS HIGHER LEVEL)
    459 002654  020402                          CMP     R4,R2           ;IT IS IN RANGE, IS IT < CURRENT LOW?
    460 002656  103364                          BHIS    3$              ;NO
    461 002660  010402                          MOV     R4,R2           ;YES: SET IT AS CURRENT LOW
    462 002662  010567  175120                  MOV     R5,SLBEG        ;STORE LOCATION FOR SUB-LEVEL PROCESSING
    463 002666  000760                          BR      3$              ;KEEP CHECKING
    464 002670  010402                  4$:     MOV     R4,R2           ;CURRENT LOW = TEST VALUE
    465 002672  000404                          BR      11$             ;AND IT IS THE LOWEST
    466 002674  020201                  10$:    CMP     R2,R1           ;WAS A VALUE FOUND?
    467 002676  001431                          BEQ     20$             ;NO: FINISHED SORTING THIS LEVEL
    468 002700  016705  175102                  MOV     SLBEG,R5        ;RESET TO SORT SUB-LEVEL
    469 002704  110277  175104          11$:    MOVB    R2,@FM2C        ;TRANSFER MOD TO SORTED AREA
    470 002710  005267  175100                  INC     FM2C
    471 002714  010200                          MOV     R2,R0           ;SET LOW TO ONE ABOVE LAST FOUND VALUE
    472 002716  005200                          INC     R0
    473 002720  022701  000177          12$:    CMP     #177,R1         ;ARE WE AT THE LOWEST LEVEL?
    474 002724  001737                          BEQ     1$              ;YES
    475 002726  020503                          CMP     R5,R3           ;NO: ARE WE AT THE END?
    476 002730  001414                          BEQ     20$             ;YES
    477 002732                                  SAVE    R0              ;NO: STEP DOWN A LEVEL
    478 002734  162701  000100                  SUB     #100,R1
    479 002740  010100                          MOV     R1,R0
    480 002742  162700  000077                  SUB     #77,R0
    481 002746  004767  177642                  JSR     PC,FMSORT       ;SORT THE SUB-LEVEL
    482 002752  062701  000100                  ADD     #100,R1         ;STEP UP A LEVEL
    483 002756                                  RESTORE R0
    484 002760  000721                          BR      1$
    485 002762  012605                  20$:    MOV     (SP)+,R5        ;FINISHED
```

```
    486 002764  000207                            RTS     PC
    487                                 ;
    488                                 ; ERROR ROUTINES
    489                                 ;
    490 002766  012767  000001  000000G ERR1:    MOV     #QE,DW1,ERRCDE
    491 002774  000167  000000G                   JMP     ERROR
    492 003000  012767  000002  000000G ERR2:    MOV     #QE,MTS,ERRCDE
    493 003006  000167  000000G                   JMP     ERROR
    494 003012  012767  000003  000000G ERR3:    MOV     #QE,DW2,ERRCDE
    495 003020  000167  000000G                   JMP     ERROR
    496 003024  012767  000004  000000G ERR4:    MOV     #QE,ILC,ERRCDE
    497 003032  000167  000000G                   JMP     ERROR
    498 003036  012767  000026  000000G ERR21:   MOV     #QE,FTB,ERRCDE
    499 003044  000167  000000G                   JMP     ERROR
    500                                 ;
    501         000001                          .END
```

SYMBOL TABLE

```
ACTFMN= ****** GX
ASKOVF= 000145
BITVAL= 000000
BIT0  = 000001
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BS.CLS= 000002
BS.DBU= 000004
BS.INA= 000000
BS.OPN= 000001
BS.SRC= 000003
BYTE0 = 000000
BYTE1 = 000001
BYTE2 = 000002
BYTE3 = 000003
BYTE4 = 000004
BYTE5 = 000005
BYTE6 = 000006
BYTE7 = 000007
BYTE8 = 000010
BYTE9 = 000011
BYTVAL= 000012
B.BSTA  000054
B.CNTX  000046
B.CQUQ  000060
B.FEMA  000132
B.FEMB  000142
B.FEMC  000152
B.FFSA  000202
B.FFSB  000212
B.FFSC  000222
B.FMHR  000172
B.FQLS  000162
B.FSAZ  000100
B.FSBZ  000102
B.FSCZ  000104
B.HBLK  000120
B.HDOC  000114
B.HRLP  000126
B.HRLR  000122
B.HRLW  000124
B.NMBR  000052
B.NQRY  000232
B.QLSZ  000106
B.QMAP  000234
B.QSPL  000316
B.QTTM  000076
B.QUQP  000056
B.SFDB  000010
B.SIZE  000772
B.SNDP  000012
B.SSQ   000004
B.SSQF  000050
B.STAT  000044
B.STTE  000053
B.UDOC  000110
CFALS = ****** GX
CF.B0 = 000070
CF.B2 = 000067
CF.B4 = 000066
CF.B6 = 000065
CF.DR0= 000064
CF.DR1= 000063
CHRCNT= ****** GX
CMPOFF  000000R
CPIXMK= ****** GX
CSTCET= ****** GX
CSTDP = ****** GX
CSTDT = ****** GX
CSTDW = ****** GX
CSTEFM= ****** GX
CSTESC= ****** GX
CSTFDC= ****** GX
CSTMIN= ****** GX
CSTNC = ****** GX
CSTNF = ****** GX
CSTNUM= ****** GX
CSTSO = ****** GX
CSTSZ = ****** GX
010 CSTTS = ****** GX
010 CSTVDC= ****** GX
010 CSTZ  = ****** GX
010 CWPF  = 100000
010 CWPIDX= ****** GX
010 CWPNDE= ****** GX
010 DBSLEN= 000116
010 DH.BF0  000002
010 DH.BF1  000004
010 DH.CTL  000000
010 DH.DMC  000010
010 DH.FLG  000006
010 DN.DCK  000000
010 DN.NTP  000004
010 DN.NXT  000006
010 DN.ROT  000002
010 DN.SIZ  000010
010 DTFLG = 000300
010 EMAOVF= 000166
010 EMBOVF= 000167
010 EMCOVF= 000170
010 EMCQSZ= ****** GX
010 ERRCDE= ****** GX
010 ERROR = ****** GX
010 ERRORV= ****** GX
010 ERR1    002766R
010 ERR1J   000016R
010 ERR2    003000R
010 ERR2J   000022R
010 ERR21   003036R
010 ERR3    003012R
010 ERR3J   000654R
010 ERR4    003024R
010 ERR4J   000660R
    FAL   = ****** GX
    FALMSZ= ****** GX
    FALOVF= 000150
    FDC   = 000020
    FD.FID  000000
    FD.FNB  000006
    FD.FVR  000004
    FD.LEN  000010
014 FLDC  = ****** GX
    FLIXMK= ****** GX
    FLUID = ****** GX
    FLUIDX= ****** GX
    FLUNDE= ****** GX
    FLUTBL= ****** GX
    FLUTYP  000002RG
    FMDIDX= ****** GX
    FMSORT  002614R
    FMIE    000010R
    FM2C    000014R
    FM2S    000012R
    FN.DBR  000026
    FN.DBS  000022
    FN.DHR  000040
    FN.EMA  000012
    FN.EMB  000014
    FN.EMC  000016
    FN.FSA  000000
    FN.FSB  000002
    FN.FSC  000004
    FN.LGQ  000034
005 FN.LGU  000036
005 FN.MFO  000024
005 FN.MHR  000010
005 FN.NMB  000044
005 FN.QLS  000006
013 FN.QRY  000020
013 FN.SF0  000030
013 FN.SF1  000032
013 FN.SHD  000042
013 GETNDE  002146RG
    INIVDC= 000010 G
    LEXTB3= ****** GX
    LNPOVF= 000146
    M     = 000062
    MBDVDC= 000004
    MERGE   002072R
    N     = 000002
    NDBKW = ****** GX
014 NDBOVF= 000175
014 NDCHR = ****** GX
014 NDFLID= ****** GX
014 NDFMEN= ****** GX
014 NDFMN = ****** GX
014 NDFMS = ****** GX
014 NDFSAA= ****** GX
014 NDFSAB= ****** GX
014 NDSIZ = ****** GX
    NDTRM = ****** GX
    NDTYP = ****** GX
    NFALE = ****** GX
    NMRNGF= 000100 G
003 NODEA = ****** GX
003 NODEB = ****** GX
003 NODEC = ****** GX
003 NULFMN= ****** GX
    NXTNDE= ****** GX
    N.BFAC= 000004
    N.BHGH= 000006
    N.BTCH= 000004
    N.BUFB= 004000
    N.BUFW= 002000
014 N.FOS = 000764
    N.PKSZ= 000020
014 N.PKTS= 000043
014 N.QURY= 000031
014 N.SUNT= 000002
014 ONYVDC= 000001
011 PFLU    000026RG
011 PFLUMD  002200RG
011 PNBRNG= ****** GX
011 POLOVF= 000157
011 PTERM   000664R
011 QBFOVF= 000150
011 QEXOVF= 000165
011 QE.DW1= 000001
011 QE.DW2= 000003
011 QE.FTB= 000026
011 QE.IHS= 000005
011 QE.ILC= 000004
011 QE.INO= 000015
011 QE.ISO= 000016
011 QE.MFL= 000013
011 QE.MFM= 000010
011 QE.MNT= 000012
011 QE.MOP= 000007
011 QE.MSD= 000011
014 QE.MTS= 000002
    QE.NLQ= 000006
    QE.NOS= 000025
    QE.NRB= 000024
    QE.NRI= 000023
    QE.NTK= 000027
014 QE.PX1= 000017
    QE.PX2= 000020
    QE.PX3= 000021
    QE.PX4= 000022
    QE.RO1= 000144
    QE.UBP= 000014
    QLBOVF= 000163
    QRYOVF= 000156
    Q.FDSC  000004  007
    Q.NOBK  000000  007
    Q.NUHL  000002  007
    Q.SIZE  000014  007
    SCAN    001726R 014
    SLBEG   000006RG 014
    SLBOVF= 000164
    SRCHBL= 000040
    SR.ARE  000114  002
    SR.ARS  000106  002
    SR.DAY  000010  002
    SR.DLT  000014  002
    SR.ECB  000047  002
    SR.ECH  000046  002
    SR.ECL  000050  002
    SR.FIB  000012  002
    SR.GRE  000100  002
    SR.GRS  000072  002
    SR.LEN  000122  002
    SR.LIN  000066  002
    SR.LIP  000062  002
    SR.MON  000006  002
    SR.NDC  000042  002
    SR.NDS  000036  002
014 SR.NIN  000030  002
014 SR.NIP  000022  002
    SR.SDB  000032  002
    SR.SRC  000002  002
014 SR.SUN  000000  002
    SR.TWS  000056  002
    SR.WSL  000052  002
    SR.YR   000004  002
    SR.IIN  000024  002
    SR.IIP  000016  002
    SS.FID  000002  004
    SS.FNB  000010  004
    SS.FVR  000006  004
    SS.LEN  000012  004
    SS.STT  000000  004
    STEP  = ****** GX
    ST.ASZ  000020  006
    ST.BSZ  000024  006
    ST.BTC  000000  006
    ST.CSZ  000030  006
    ST.HRL  000010  006
    ST.LEN  000044  006
    ST.QRY  000002  006
    ST.QSZ  000034  006
    ST.SCH  000040  006
```

```
.MAIN.  MACRO M1110  27-MAR-80 12:50  PAGE 17-10
SYMBOL TABLE

ST.UHL  000004      006 S.HRL = 000240          VLDC  = ****** GX      WORD4 = 000010          XFOSMR= 000007
ST.XLT  000014      006 TDBOVF= 000174          WN.NTP  000004    012 WORD5 = 000012          XGTSRE= 000014
SU.DBU= 000004          TERMF = 040000          WN.NXT  000006    012 WORD6 = 000014          XHITSK= 000011
SU.DON= 000006          TRLVDC= 000002          WN.ROT  000002    012 WORD7 = 000016          XHLMER= 000002
SU.IDL= 000000          TSKOVF= 000147          WN.SIZ  000010    012 WORD8 = 000020          XHOTSK= 000010
SU.LOD= 000001          TTBOVF= 000161          WN.SRC  000000    012 WORD9 = 000022          XMSCHE= 000000
SU.SRC= 000002          TYPFLU  000004R     014 WN.TYP  000001    012 WRDVAL= 000024          XQTS  = 000003
SU.SRR= 000005          VDCCNT= ****** GX       WORD0 = 000000        XBATCH= 000013          XQT0  = 000001
SU.XPD= 000003          VI1OVF= 000173          WORD1 = 000002        XDBLOA= 000004          XSULOA= 000005
SZCALI= ****** GX       VI2OVF= 000172          WORD2 = 000004        XDBPRO= 000012          XTBOVF= 000162
SZFLG = 000100          VI3OVF= 000171          WORD3 = 000006        XDMCIN= 000006          ZFLG  = 000200

. ABS.  000000     000
        000000     001
SRCOFF  000122     002
FDSCOF  000010     003
SUSOFF  000012     004
DHROFF  000012     005
STTOFF  000044     006
QSPLOF  000014     007
BSTOFF  000772     010
FNOFFS  000044     011
WNODOF  000010     012
DNODOF  000010     013
FLUCDE  003050     014
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  6485 WORDS  (26 PAGES)
DYNAMIC MEMORY:  8084 WORDS  ( 31 PAGES)
ELAPSED TIME:  00:04:26
FLUCDE,FLUCDE/-SP/NL:ME:BEX=[20,1]P,M,E,MDQT0,FLUCDE
```

```
 432 002252  004767  177064          3$:     JSR     PC,CHKRNG       ;CHECK LOW RANGE
 433 002256  010467  175520          5$:     MOV     R4,HBEG         ;SET BEGINNING OF HIGH NUMBER
 434 002262  112767  000060  175536          MOVB    #'0,NUMVLU      ;SET SPECIAL NUMBER FOR HIGH RANGE
 435 002270  032767  000020  175532          BIT     #PD,RNGCDE
 436 002276                                  BOFF    4$
 437 002300  005067  175510                  CLR     HSZ             ;PROCESS DECIMAL NUMBER
 438 002304  012746  000020'                 MOV     #HSZD,-(SP)
 439 002310  005202                          INC     R2
 440 002312  000417                          BR      10$
 441 002314  012746  000014'         4$:     MOV     #HSZ,-(SP)      ;PROCESS BOTH BEFORE AND AFTER DEC POINT
 442 002320  032767  000010  175502          BIT     #DNUM,RNGCDE    ;DECIMAL?
 443 002326                                  BOFF    8$              ;NO
 444 002330  005202                          INC     R2              ;YES: DEC # ALWAYS REQUIRES EXTRA TERM
 445 002332  032767  000040  175470  8$:     BIT     #CM,RNGCDE
 446 002340                                  BON     10$             ;IF COMMON CHAR ELIMINATED, DON'T ...
 447 002342  122714  000061                  CMPB    #'1,(R4)        ;  CHECK FOR LEADING 1
 448 002346  001001                          BNE     10$             ;     NOT LEADING 1
 449 002350  005302                          DEC     R2              ;     LEADING 1 COUNTS 1
 450 002352  004767  176764          10$:    JSR     PC,CHKRNG       ;CHECK HIGH RANGE
 451 002356  032767  000020  175444          BIT     #PD,RNGCDE
 452 002364                                  BOFF    11$
 453 002366  005202                          INC     R2
 454 002370  005767  175422                  TST     LSZD            ;IF PAST DEC POINT AND
 455 002374  001011                          BNE     12$             ;   NO TRAILING NUMBER FOR LOW
 456 002376  000444                          BR      30$
 457 002400  026767  175406  175406  11$:    CMP     LSZ,HSZ         ;COMPARE PLACE OF LOW AND HIGH
 458 002406  001013                          BNE     13$             ;NOT EQUAL
 459 002410  026727  175400  000001          CMP     HSZ,#1          ;EQUAL: LAST CHAR?
 460 002416  001424                          BEQ     16$             ;YES
 461 002420  117701  175356          12$:    MOVB    @HBEG,R1        ;EQUAL: COMPARE CHARS
 462 002424  005301                          DEC     R1
 463 002426  127701  175346                  CMPB    @LBEG,R1
 464 002432  001415                          BEQ     15$             ;LOW=HIGH-1
 465 002434  001015                          BNE     16$             ;LOW<HIGH-1
 466 002436  005767  175350          13$:    TST     LSZ             ;DOES LOW START WITH DEC POINT?
 467 002442  001014                          BNE     20$             ;NO
 468 002444  016701  175344                  MOV     HSZ,R1          ;YES
 469 002450  005301                          DEC     R1
 470 002452  001015                          BNE     21$             ;HIGH RANGE > 1 PLACE
 471 002454  127727  175322  000061  14$:    CMPB    @HBEG,#'1       ;HIGH=1?
 472 002462  003012                          BGT     30$             ;HIGH>1
 473 002464  000401                          BR      16$             ;HIGH=1
 474 002466  005302                  15$:    DEC     R2              ;HIGH=1
 475 002470  005302                  16$:    DEC     R2
 476 002472  000406                          BR      30$
 477 002474  016701  175314          20$:    MOV     HSZ,R1          ;IF SIZES DON'T MATCH, THEN ADD
 478 002500  005301                          DEC     R1              ;DIFFERENCE - 1
 479 002502  166701  175304                  SUB     LSZ,R1
 480 002506  060102                  21$:    ADD     R1,R2
 481 002510  010267  175270          30$:    MOV     R2,NMAX         ;STORE # TERMS FOR RANGE
 482 002514  006302                          ASL     R2
 483 002516  010067  175266                  MOV     R0,EXPSPT       ;SAVE CURRENT EMX LOC
 484 002522  022702  000002                  CMP     #2,R2           ;ONLY 1 TERM?
 485 002526  001511                          BEQ     31$             ;YES
 486 002530  012710  000000C                 MOV     #EMXNSQ!EMXEXF,(R0)     ;NO: CREATE EXP VECTOR
 487 002534  050220                          BIS     R2,(R0)+
 488 002536  010067  175244                  MOV     R0,EXPVPT
```

```
 375 001736  001431                        BEQ··   91$             ;IF·LEADING·DECIMAL·POINT·
 376 001740  005767  176060                 TST·    INCLUD·
 377 001744  001403                         BEQ·    51$
 378 001746  062767  000002  000000G·       ADD·    #2,VINXT·       ;IF·EXP·MUST·BE·INCLUDED, ADJUST·FOR·OVERWRITE·
 379 001754  122714  000060          51$:   CMPB·   #'0,(R4)
 380 001760  001136                         BNE·    5$              ;IF·ELSE
 381 001762· 010067  176022                 MOV·    R0,EXPSPT·      ;IF·ONLY A·ZERO·
 382·001766  112703  000060                 MOVB·   #'0,R3          ;ENTER·ZERO·
 383 001772· 004767  004072                 JSR·    PC,EMXSNG·
 384 001776  062704  000002                 ADD·    #2,R4
 385 002002· 162705  000002                 SUB·    #2,R5
 386 002006  005067  176030                 CLR·    ZPTR
 387 002012·                                SAVE·   R4,R5
 388 002016  000167  001324                 JMP·    44$             ;COMPLETE·ENTRY·
 389 002022· 052767  000060  176000  91$:    BIS·    #PD!CM,RNGCDE·  ;LEADING DEC, ABSORB·DEC·POINT·
 390 002030  005204                         INC·    R4
 391 002032· 005305                         DEC·    R5
 392·002034  005767  175764                 TST·    INCLUD·
 393 002040  001414                         BEQ·    18$
 394 002042· 016703  000000G·               MOV·    VINXT,R3        ;INCLUDE IN·INIT, VECTOR·
 395 002046  062767  000002  000000G·       ADD·    #2,VINXT·
 396 002054  010063  000000G·               MOV·    R0,VI(R3)
 397 002060  162763  000000G· 000000G·      SUB·    #EMX,VI(R3)
 398 002066  005067  175732                 CLR·    INCLUD·         ;BUT·DON'T·INCLUDE ANYTHING·ELSE·
 399 002072· 012701  000000G·          18$: MOV·    #DPEMX,R1       ;ENTER·DEC·POINT·
 400 002076  012120                         MOV·    (R1)+,(R0)+
 401 002100  012120                         MOV·    (R1)+,(R0)+
 402·002102· 012120                         MOV·    (R1)+,(R0)+
 403 002104  122714  000000G·               CMPB·   #SEGOP,(R4)
 404 002110  001062·                        BNE·    5$              ;IF·MORE
 405 002112· 005305                         DEC·    R5              ;IF·DECIMAL·POINT·ONLY, CREATE·ZVLDC·
 406 002114  005204                         INC·    R4
 407 002116  032767  000200  175704         BIT·    #MR2,RNGCDE·
 408 002124                                 BOFF·   100$
 409 002126  016720  175710                 MOV·    ZPTR,(R0)+      ;OR·IF·IT·ALREADY·EXIST·FOR·THIS·FLU,
 410 002132·                                SAVE·   R4,R5           ;JUST·LINK·TO·IT·
 411 002136  000167  001316                 JMP·    45$
 412·002142· 032767  000100  175660  100$:  BIT·    #MR,RNGCDE·
 413 002150                                 BOFF·   101$
 414 002152· 010001                         MOV·    R0,R1           ;IF·MULTI-RANGE, ENABLE·LINK TO·THIS·ENTRY·
 415 002154  162701  000000G·               SUB·    #EMX,R1
 416 002160  052701  000000G·               BIS·    #EMXNSQ,R1
 417 002164  010167  175652                 MOV·    R1,ZPTR·
 418 002170  012701  000000G·          101$: MOV·   #ZVDEMX,R1      ;ENTER·ZVLDC·
 419 002174  012120                         MOV·    (R1)+,(R0)+
 420 002176  012120                         MOV·    (R1)+,(R0)+
 421 002200  012120                         MOV·    (R1)+,(R0)+
 422·002202·                                SAVE·   R4,R5
 423 002206  000167  001134                 JMP·    44$
 424 002212· 010467  175562          1$:    MOV·    R4,LBEG·        ;SET·BEGINNING·OF·LOW·NUMBER
 425 002216  112767  000071  175602·       MOVB·   #'9,NUMVLU·     ;SET·SPECIAL·NUMBER·FOR·LOW·RANGE·
 426 002224  032767  000020  175576         BIT·    #PD,RNGCDE·
 427 002232·                                BOFF·   2$
 428·002234  005067  175552                 CLR·    LSZ·            ;IF·PAST DEC·POINT, PROCESS·DEC·NUMBER
 429 002240  012746  000016'                MOV·    #LSZD,-(SP)
 430 002244  000402·                        BR      3$
 431 002246  012746  000012'          2$:   MOV·    #LSZ,-(SP)      ;IF·NOT·PAST·DEC, PROCESS·BOTH·
```

```
489 002542  060200                          ADD     R2,R0           ;ADJUST EMX START BY EXP VECTOR SIZE
490 002544  032767  000010  175256          BIT     #DNUM,RNGCDE    ;DECIMAL NUMBER?
491 002552                                  BOFF    32$             ;NO
492 002554  032767  000020  175246          BIT     #PD,RNGCDE      ;YES: PAST DECIMAL?
493 002562                                  BON     303$            ;YES
494 002564  005767  175246                  TST     DPTR            ;NO:
495 002570  001045                          BNE     32$             ;IF THE ENTRIES ALREADY EXIST, DON'T RECREATE
496 002572  004767  003640                  JSR     PC,NFGEN        ;GENERATE NFLDC'S
497 002576  010001                          MOV     R0,R1           ;DPTR=MERGE VECTOR TO DEC POINT
498 002600  162701  000000G                 SUB     #EMX,R1
499 002604  052701  000000G                 BIS     #EMXNSQ,R1
500 002610  010167  175222                  MOV     R1,DPTR
501 002614  012701  000000G                 MOV     #DPEMX,R1       ;ENTER DEC POINT
502 002620  012120                          MOV     (R1)+,(R0)+
503 002622  012120                          MOV     (R1)+,(R0)+
504 002624  012120                          MOV     (R1)+,(R0)+
505 002626  005767  175206                  TST     NVPTR
506 002632  001406                          BEQ     304$            ;IF IT DOESN'T EXIST
507 002634  016720  175200                  MOV     NVPTR,(R0)+     ;IF IT DOES EXIST, LINK DPTR TO IT
508 002640  000434                          BR      320$
509 002642  005767  175172          303$:   TST     NVPTR
510 002646  001031                          BNE     320$            ;IF IT EXIST, DON'T ALTER
511 002650  010001                  304$:   MOV     R0,R1           ;NVPTR=MERGE VECTOR TO NVLDC
512 002652  162701  000000G                 SUB     #EMX,R1
513 002656  052701  000000G                 BIS     #EMXNSQ,R1
514 002662  010167  175152                  MOV     R1,NVPTR
515 002666  012701  000000G                 MOV     #NVDEMX,R1      ;ENTER NUMERICAL VLDC
516 002672  012120                          MOV     (R1)+,(R0)+
517 002674  012120                          MOV     (R1)+,(R0)+
518 002676  012120                          MOV     (R1)+,(R0)+
519 002700  005020                          CLR     (R0)+           ;ALLOCATE MERGE VECTOR
520 002702  000413                          BR      320$
521 002704  004767  003526          32$:    JSR     PC,NFGEN        ;GENERATE NFLDC'S
522 002710  103410                          BCS     320$            ;IF NONE
523 002712  032767  000010  175110          BIT     #DNUM,RNGCDE    ;DECIMAL NUMBER?
524 002720                                  BOFF    321$            ;NO
525 002722  016720  175110                  MOV     DPTR,(R0)+      ;YES: LINK TO DECIMAL POINT
526 002726  000401                          BR      320$
527 002730  005020                  321$:   CLR     (R0)+           ;ALLOCATE MERGE VECTOR
528 002732  010001                  320$:   MOV     R0,R1           ;ENTER 1ST EMX PTR IN EXPANSION VECTOR
529 002734  162701  000000G                 SUB     #EMX,R1
530 002740  010177  175042                  MOV     R1,@EXPVPT
531 002744  062767  000002  175034          ADD     #2,EXPVPT       ;STEP TO NEXT ENTRY IN EXP VECTOR
532 002752                          31$:    SAVE    R4,R5
533 002756  005067  175066                  CLR     NOHTRM
534 002762  032767  000201  175040          BIT     #MR2!MR3,RNGCDE
535 002770                                  BON     36$             ;IF ZPTR VALID
536 002772  005067  175044                  CLR     ZPTR            ;IF INVALID
537 002776  032767  000020  175024  36$:    BIT     #PD,RNGCDE      ;PAST DEC POINT?
538 003004                                  BOFF    33$             ;NO
539 003006  004767  002050                  JSR     PC,RLOWD        ;YES: CREATE LOW DEC TERM ENTRIES
540 003012  004767  002460                  JSR     PC,RHIGHD       ;CREATE HIGH DEC TERM ENTRIES
541 003016  000417                          BR      35$
542 003020  004767  000734          33$:    JSR     PC,RLOW         ;CREATE LOW TERM EMX ENTRIES
543 003024  016701  174764                  MOV     HSZ,R1          ;R1=# MID TERM EMX ENTRIES
544 003030  i66701  174756                  SUB     LSZ,R1
545 003034  005301                          DEC     R1
```

```
546 003036  003402
547 003040  004767  001342          331$:   BLE     34$             ;THERE ARE NO MID TERM ENTRIES
548 003044  005767  175000          34$:    JSR     PC,RMID         ;CREATE MID TERM EMX ENTRIES
549 003050  001002                          TST     NOHTRM          ;IF NO HIGH TERM, SHIP
550 003052  004767  001404                  BNE     35$
551 003056  022767  000001  174720  35$:    JSR     PC,RHIGH        ;CREATE HIGH TERM EMX ENTRIES
552 003064  001530                          CMP     #1,NMAX         ;ONLY ONE TERM?
553 003066  005767  174732                  BEQ     44$             ;YES
554 003072  001423                          TST     INCLUD          ;NO: SHOULD VECTORS BE INCLUDED?
555 003074  016702  174710                  BEQ     42$             ;NO
556 003100  062702  000002                  MOV     EXPSPT,R2
557 003104  016705  174674                  ADD     #2,R2           ;YES: COPY POINTERS IN INIT VECTOR
558 003110  006305                          MOV     NMAX,R5
559 003112  162767  000002  000000G         ASL     R5
560 003120  016701  000000G                 SUB     #2,VINXT
561 003124  060567  000000G                 MOV     VINXT,R1
562 003130  006205                          ADD     R5,VINXT        ;AND ADJUST VECTOR COUNT
563 003132  062701  000000G                 ASR     R5
564 003136  012221                          ADD     #VI,R1
565 003140  077502                  41$:    MOV     (R2)+,(R1)+
566 003142  032767  000400  174660  42$:    SOB     R5,41$
567 003150                                  BIT     #MR1,RNGCDE     ;MULTIPLE RANGE?
568 003152  016701  174652                  BOFF    421$            ;NO
569 003156  052701  000200                  MOV     RNGCDE,R1       ;YES
570 003162  000561                          BIS     #MR2,R1         ;SET UP FOR SECOND PASS
571 003164  016702  174620          421$:   BR      452$
572 003170  010001                          MOV     EXPSPT,R2       ;FILL IN ALL MERGE ENTRIES
573 003172  162701  000000G                 MOV     R0,R1
574 003176  052701  000000G                 SUB     #EMX,R1
575 003202  032767  000002  174620          BIS     #EMXNSO,R1      ;R1=MERGE ENTRY VALUE
576 003210                                  BIT     #USER3,RNGCDE   ;FINAL PASS?
577 003212  010163  177776                  BOFF    457$            ;NO
578 003216  000405                          MOV     R1,-2(R3)       ;YES: USE OLD LAST MERGE ENTRY
579 003220  005760  177776          457$:   BR      456$
580 003224  001002                          TST     -2(R0)
581 003226  010160  177776                  BNE     456$            ;IF MERGE SET ALREADY
582 003232  032767  000010  174570  456$:   MOV     R1,-2(R0)       ;LAST MERGE ENTRY
583 003240                                  BIT     #DNUM,RNGCDE    ;DECIMAL NUMBER?
584 003242  016705  174536                  BON     46$             ;YES
585 003246  005305                          MOV     NMAX,R5         ;R5=# TERMS
586 003250  062702  000004                  DEC     R5
587 003254  016703  174552                  ADD     #4,R2           ;FILL IN OTHERS BY PTR PAST ENTRY-2
588 003260  001413                          MOV     NFPTR,R3        ;FILL IN MERGE VECTOR AFTER NFLDC'S
589 003262  042703  000000G                 BEQ     43$             ;NOT ONE
590 003266  010163  000000G                 BIC     #EMXNSO,R3
591 003272  016703  174536                  MOV     R1,EMX(R3)
592 003276  001404                          MOV     NF1PTR,R3       ;AND SAVED ONE, IF MULTI-RANGE
593 003300  042703  000000G                 BEQ     43$             ;NOT ONE
594 003304  010163  000000G                 BIC     #EMXNSO,R3
595 003310  012203                  43$:    MOV     R1,EMX(R3)
596 003312  005763  177776G                 MOV     (R2)+,R3
597 003316  001002                          TST     EMX-2(R3)       ;IF ALREADY FILLED IN
598 003320  010163  177776G                 BNE     430$            ;   DON'T ALTER
599 003324  077507                          MOV     R1,EMX-2(R3)
600 003326  000454                  430$:   SOB     R5,43$
601 003330  016703  174504          46$:    BR      45$
602 003334  042703  000000G                 MOV     NVPTR,R3        ;ENTER MERGE VECTOR
                                            BIC     #EMXNSO,R3
```

```
    603 003340  010163  000006G                  MOV     R1,EMX+6(R3)
    604 003344  000445                            BR      45$
    605 003346  032767  000400  174454  44$:      BIT     #MR1,RNGCDE
    606 003354                                    BOFF    441$
    607 003356  010067  174440                    MOV     R0,EXPSMR       ;IF MULTI-RANGE, ALLOCATE MERGE VECTOR
    608 003362  005020                            CLR     (R0)+
    609 003364  052767  000001  174436            BIS     #MR3,RNGCDE     ;FLAG PATH OCCURRENCE
    610 003372  000411                            BR      440$
    611 003374  032767  000100  174426  441$:     BIT     #MR,RNGCDE
    612 003402                                    BOFF    440$
    613 003404  010010                            MOV     R0,(R0)         ;IF SECOND PASS, JUST ENTER MERGE
    614 003406  162710  177776G                   SUB     #EMX-2,(R0)
    615 003412  052720  000000G                   BIS     #EMXNSQ,(R0)+
    616 003416  005767  174402            440$:   TST     INCLUD          ;SHOULD INIT VECTOR INCLUDE PTR?
    617 003422  001416                            BEQ     45$             ;NO
    618 003424  016703  174360                    MOV     EXPSPT,R3       ;YES; SO INCLUDE IT
    619 003430  162767  000002  000000G           SUB     #2,VINXT
    620 003436  016701  000000G                   MOV     VINXT,R1
    621 003442  162703  000000G                   SUB     #EMX,R3         ;YES; CONVERT TO LOC
    622 003446  010361  000000G                   MOV     R3,VI(R1)       ;ENTER IN INIT VECTOR
    623 003452  062767  000002  000000G           ADD     #2,VINXT
    624 003460  016701  174344            45$:    MOV     RNGCDE,R1       ;MULTI-RANGE CHECK
    625 003464  032701  000200                    BIT     #MR2,R1
    626 003470                                    BON     455$            ;IF 2ND PASS OF MULTI-RANGE
    627 003472  032701  000400                    BIT     #MR1,R1
    628 003476                                    BON     452$            ;IF 1ST PASS
    629 003500  032701  000001                    BIT     #MR3,R1
    630 003504                                    BOFF    450$            ;IF SINGLE RANGE
    631 003506  016703  174310                    MOV     EXPSMR,R3       ;IF 2ND PASS, COND 3
    632 003512  010013                            MOV     R0,(R3)         ;FILL IN MERGE VECTOR
    633 003514  162713  000000G                   SUB     #EMX,(R3)
    634 003520  052713  000000G                   BIS     #EMXNSQ,(R3)
    635 003524  000510                            BR      450$
    636 003526  042701  000460            452$:   BIC     #MR1!PD!CM,R1   ;IF MULTI-RANGE 1ST PASS, RESET FLAGS
    637 003532                                    RESTOR  R4,R5
    638 003536  032701  000004                    BIT     #INCLD,R1
    639 003542                                    BOFF    451$
    640 003544  012767  000001  174252           MOV     #1,INCLUD       ;IF INCLUDE WAS ACTIVE BEFORE, REACTIVATE
    641 003552  000405                            BR      4517$
    642 003554                            451$:   RESTOR  R3
    643 003556  010002                            MOV     R0,R2           ;POINT 2RD ENTRY OF EXP VECTOR HERE
    644 003560  162702  000000G                   SUB     #EMX,R2
    645 003564  010213                            MOV     R2,(R3)
    646 003566  005002                    4517$:  CLR     R2              ;RESET NMAX COUNTER
    647 003570  032767  000010  174232           BIT     #DNUM,RNGCDE
    648 003576                                    BON     4518$           ;IF DEC NUMBER
    649 003600  012767  000001  174204            MOV     #1,LSZ          ;IF NON-DEC, FAKE LOW
    650 003606  005202                            INC     R2
    651 003610  000402                            BR      4519$
    652 003612  005067  174174            4518$:  CLR     LSZ             ;RESET VARIABLES
    653 003616  005067  174174            4519$:  CLR     LSZD
    654 003622  005767  174176                    TST     INCLUD
    655 003626  001412                            BEQ     453$
    656 003630  016703  000000G                   MOV     VINXT,R3        ;INCLUDE PTR TO ZVLDC IN INIT. VECTOR
    657 003634  010063  000000G                   MOV     R0,VI(R3)
    658 003640  162763  000000G 000000G           SUB     #EMX,VI(R3)
    659 003646  062767  000002  000000G           ADD     #2,VINXT
```

```
    660 003654  032701  000200          453$:   BIT     #MR2,R1          ;IF SPECIAL 2RD PASS NEEDED, SAVE STATE
    661 003660                                  BOFF    454$
    662 003662                                  SAVE    EXPSPT,R0,NMAX
    663 003674  010167  174130          454$:   MOV     R1,RNGCDE        ;RESET RANGE CODE
    664 003700  000167  176014                  JMP     17$              ;START SECOND PASS
    665 003704                          455$:   RESTOR  R4,R5            ;SET UP FOR FINAL PASS
    666 003710                                  RESTOR  EXPSPT,R3,NMAX
    667 003722                                  SAVE    R4,R5
    668 003726  042767  000600  174074          BIC     #MR1!MR2,RNGCDE ;NO MORE PASSES
    669 003734  052767  000002  174066          BIS     #USER3,RNGCDE   ;USE R3 FOR LAST MERGE VECTOR
    670 003742  000167  177216                  JMP     421$
    671 003746                          450$:   RESTOR  R2,R3,R4,R5
    672 003756  000207                          RTS     PC
```

.MAIN. MACRO M1110 27-MAR-80 13:04
TABLE OF CONTENTS

```
    12-     2     NUMRNG NUMBER RANGE CODE
```

```
                                ;THIS FILE (E.MAC) CONTAINS ALL THE QUERY ERROR CODES
                                ;
000001                          QE.DW1=1        ;RESERVED
000002                          QE.MTS=2        ;MULTIPLE TERM SEP'S OR MISSING TERM
000003                          QE.DW2=3        ;RESERVED
000004                          QE.ILC=4        ;ILLEGAL CHARACTER
000005                          QE.IHS=5        ;INTERNAL HSTS ERROR, LOGIC PROCESSING
000006                          QE.NLQ=6        ;NULL QUERY
000007                          QE.MOP=7        ;MISSING OPERATOR
000010                          QE.MFM=8.       ;MISPLACED FLU-MOD
000011                          QE.MSD=9.       ;MISPLACED SUBDOC OPERATOR
000012                          QE.MNT=10.      ;MISPLACED NOT OPERATOR
000013                          QE.MFL=11.      ;MISSING FLU / TWO CONSECUTIVE OPERATORS
000014                          QE.UBP=12.      ;UNBALANCED PARENTHESES
000015                          QE.INO=13.      ;INSUFFICIENT NUMBER OF OPERANDS
000016                          QE.ISO=14.      ;ILLEGAL SUBELEMENT OPERAND
000017                          QE.PX1=15.      ;MISSING PROX, WINDOW, OR DOC TYPE,ZONE, OR SUBZONE
000020                          QE.PX2=16.      ;PROX, WINDOW OR FLU-MOD ID TOO LARGE
000021                          QE.PX3=17.      ;MISSING PROX, UNIT
000022                          QE.PX4=18.      ;MISSING PROX, DELIMITER
000023                          QE.NRI=19.      ;ILLEGAL NUMERICAL RANGE SPECIFICATION
000024                          QE.NRB=20.      ;NUMERICAL RANGE BORDERED BY NUMERIC
000025                          QE.NOS=21.      ;NO STX FOUND IN QUERY
000026                          QE.FTB=22.      ;FLU TOO BIG
000027                          QE.NTK=23.      ;UNDEFINED TOKEN
                                ;
                                ;QE.RO1=100.    ;GENERAL RESOURCE OVERFLOW (LABEL DEFINED IN M.MAC)
000145                          ASKOVF=101.     ;ARGUMENT STACK OVERFLOW
000146                          LNPOVF=102.     ;LOGIC NODE POOL OVERFLOW
000147                          TSKOVF=103.     ;TOKEN STACK OVERFLOW
000150                          QBFOVF=104.     ;QUERY TOO BIG
000156                          QRYOVF=110.     ;# QUERIES OVERFLOWED
000157                          POLOVF=111.     ;FLU POOL OVERFLOW
000160                          FALOVF=112.     ;FAL
000161                          TTBOVF=113.     ;TTABLE
000162                          XTBOVF=114.     ;XTABLE
000163                          QLBOVF=115.     ;QLB
000164                          SLBOVF=116.     ;SDLB
000165                          QEXOVF=117.     ;QEX
000166                          EMAOVF=118.     ;EMA
000167                          EMBOVF=119.     ;EMB
000170                          EMCOVF=120.     ;EMC
000171                          VI3OVF=121.     ;VI QT3
000172                          VI2OVF=122.     ;VI QT2
000173                          VI1OVF=123.     ;VI QT1
000174                          TDBOVF=124.     ;TDCTB
000175                          NDBOVF=125.     ;NODE POOL QT2
                                ;
```

Approved For Release 2005/08/22 : CIA-RDP85-00514R000200010001-4

```
                        ;
000240                  FDOF$L
000240                  FCSBT$
```

```
                        .TITLE  QUERY TRANSLATOR SCHEDULER
                        .SBTTL  QTS
                ;
                ;   PURPOSE OF THE QUERY TRANSLATOR SCHEDULER IS TO COMMAND THE
                ;   QUERY TRANSLATOR TASKS (QT1, QT2 AND QT3) TO RUN.
                ;
                ;   QTS IS ACTIVATED BY MSCHED VIA TH RUN$ DIRECTIVE, AFTER MSCHED QUEUES,
                ;   VIA THE RCVQ, TO QTS A 13-WORD PACKET CONTAINING THE NUMBER OF THE
                ;   BATCH TO BE TRANSLATED.
                ;
                ;   QTS PASSES TO THE TRANSLATOR TASKS, IN A 13-WORD BUFFER VIA THE SDAT$
                ;   DIRECTIVE, THE 4-WORD FDSC FOR THE APPROPRIATE E-MATRIX, THE 3-WORD
                ;   DIRECTORY FID, AND THE DEVICE NAME AND UNIT FOR THE TDCT FILE TO BE
                ;   CREATED BY THE TRANSLATOR TASK.
                ;   QTS RECEIVES, IN A 13-WORD BUFFER, THE 4-WORD FDSC OF THE NEW TDCT
                ;   ALONG WITH THE NUMBER OF FSA STATES IN THE TDCT, AND THE ELAPSED
                ;   SECONDS AND TICS FOR THE TRANSLATION PROCESS.
                ;
                ;   UPON COMPLETION OF THE TRANSLATION PROCESS QTS PASSES TO MSCHED,
                ;   VIA THE SSQ, AN ACK RECORD.  IF ONE OF THE TRANSLATORS RETURNED
                ;   AN ERROR CODE TO QTS INDICATING BATCH OVERFLOW, THE ACK RECORD
                ;   SENT TO MSCHED REFLECTS THIS ERROR CONDITION.
                ;
                ;   ASSEMBLER PARAMETERS:
                ;       MAC>QTS,LP=P,M,T,QTS
                ;
                ;   TASK BUILD PARAMETERS:
                ;       TKB>QTS,LP=QTS,[50,1]MSGOUT
                ;       TKB>/
                ;       TKB>TASK=QTS
                ;       TKB>COMMON=MCOM:RW
                ;       TKB>STACK=191
                ;       TKB>/
                ;
                        .NLIST  MEB
                        .NLIST  CND
                        .MCALL  EXIT$S,SDAT$C,SPND$S,RCVX$C,RSUM$C,RQST$C
                        .MCALL  QIOW$S,RCVD$C,MOUT$S
                        .MCALL  FDOF$L,FCSBT$,NMBLK$,GTIM$S
                ;
                .MACRO  TEXT
                        MOVB    (R4)+,(R3)+
                        BNE     .-2
                        DEC     R3
                .ENDM
                .MACRO  DECNV   VAL
                        MOV     VAL,R1
                        JSR     PC,DECASC
                .ENDM
                .MACRO  TIMCNV  SECS,TICKS
                        MOV     TICKS,R1
                        MOV     SECS,R0
                        BEQ     .+10
                        JSR     PC,TIMASC
                        BR      .+6
                        JSR     PC,TICASC
                .ENDM
```

```
                                        ;
                                        ;   CONSTANTS AND BUFFER DEFINITIONS
                                        ;
000000                                  STARSW  =       0
000015                                  CR      =       15              ;ASCII CARRIAGE RETURN
000012                                  LF      =       12              ;ASCII LINE FEED
000000                                  $DSW    =       0               ;DIRECTIVE STATUS WORD
                                        ;
                                        ;
                                        ;
000000                                          .PSECT
                                        ;
                                        ;
                                        ;
000000  066577  000000                  QT1:    .RAD50  /QT1    /
000004  066600  000000                  QT2:    .RAD50  /QT2    /
000010  066601  000000                  QT3:    .RAD50  /QT3    /
000014  114373  031314                  XSCHED: .RAD50  /XSCHED/
000020  000000                          BAT.NO: .WORD   0
000022                                  RECB1:  .BLKW   2               ;SENDING TASK'S NAME
000026                                  SNDB1:  .BLKW   13.
000060                                  RECB2:  .BLKW   2
000064                                  SNDB2:  .BLKW   13.
000116                                  RECB3:  .BLKW   2
000122                                  SNDB3:  .BLKW   13.
```

```
     88                             ;
     89                             ;   ALLOCATE DUMMY FILENAME BLOCK.
     90                             ;
     91 000154                      NMBLK:  NMBLK$
```

```
                                        .SBTTL  MAIN LOOP
                                ;
                                ;
                                ;
000212                          START:
000212                                  RCVX$C  ,RECB1
000220  016767  177604  177572          MOV     SNDB1+2,BAT.NO          ;SAVE BATCH NUMBER
000226                                  GTIM$S  #GTIM1
                                ;
                                ;   RUN QT1
                                ;
000240  012701  000132          1$:     MOV     #B.FEMA,R1      ;R1 = EMA'S FDSC OFFSET
000244  012705  000000                  MOV     #FN.FSA,R5      ;R5 = FILE NUMBER FOR TDCTA
000250  012704  000026'                 MOV     #SNDB1,R4       ;R4-> SEND BUFFER
000254                                  CALL    BLDFNB          ;BUILD 13-WORD PACKET TO BE SENT
                                                                ;TO QT1
                                ;
                                ;   TRANSMIT 13-WORD PACKET TO QT1
                                ;
000260                                  SDAT$C  QT1,SNDB1
000266  103004                          BCC     2$
000270                                  CALL    DIRERR
000274  000167  001224                  JMP     EXIT
000300                          2$:     RQST$C  QT1,SNDB1               ;REQUEST QT1 TO BE RUN
000306  103004                          BCC     3$
000310                                  CALL    DIRERR
000314  000167  001204                  JMP     EXIT
                                ;
000320                          3$:     SPND$S                          ;WAIT FOR REPLY FROM QT1
000326                                  RCVD$C  QT1,RECB1               ;RECEIVE DATA FROM QT1
                                ;
                                ;   13-WORD PACKET RECEIVED FROM QT1 HAS FOLLOWING CONFIGURATION:
                                ;
                                ;       FD.FID          - TDCTA'S FID
                                ;       FD.FVR          -   "     VERSION NUMBER
                                ;       FD.FNB          -   "     FILE NUMBER
                                ;       SD.FSA          - NUMBER OF FSA STATES GENERATED BY QT1(-1 IF ERROR)
                                ;       SD.SEC          - ELAPSED SECONDS
                                ;       SD.TIC          -    "    TICS
                                ;
                                ;   LOAD TDCTA'S FDSC INTO BATCH STATUS TABLE
                                ;
000334  012700  000026'                 MOV     #SNDB1,R0               ;R0->RECEIVED DATA
000340  005760  000010                  TST     SD.FSA(R0)              ;ERROR FROM QT1 ?
000344  002004                          BGE     10$                     ;BRANCH IF NO
000346  016000  000010                  MOV     SD.FSA(R0),R0           ;R0 = ERROR CODE
000352  000167  000346                  JMP     NACK                    ;SEND NACK TO MSCHED
000356  016701  177436          10$:    MOV     BAT.NO,R1               ;R1 = BATCH NUMBER
000362  016101  000000G                 MOV     BSTPTR(R1),R1           ;R1->BATCH STATUS TABLE
000366  016061  000000  000202          MOV     FD.FID(R0),B.FFSA(R1)   ;FID
000374  016061  000002  000204          MOV     FD.FID+2(R0),B.FFSA+2(R1)
000402  016061  000004  000206          MOV     FD.FVR(R0),B.FFSA+4(R1) ;VERSION NUMBER
000410  016061  000010  000100          MOV     SD.FSA(R0),B.FSAZ(R1)   ;SIZE OF FSA-A TDCT
                                ;
                                ;   QT1 IS DONE, RUN QT2
                                ;
000416  012701  000142                  MOV     #B.FEMB,R1      ;R1 = EMB'S FDSC OFFSET
```

```
                        MOV     #FN.FSB,R5      ;R5 = FILE NUMBER FOR TDCTB
                        MOV     #SNDB2,R4       ;R4-> SEND BUFFER
                        CALL    BLDFNB          ;BUILD 13-WORD PACKET TO BE SENT
                                                ;TO QT2

                ;
                ;   TRANSMIT 13-WORD PACKET TO QT2
                ;
                        SDAT$C  QT2,SNDB2
                        BCC     4$
                        CALL    DIRERR
                        JMP     EXIT
                4$:     RQST$C  QT2,SNDB2               ;REQUEST QT2 TO BE RUN
                        BCC     5$
                        CALL    DIRERR
                        JMP     EXIT
                ;
                5$:     SPND$S                          ;WAIT FOR REPLY FROM QT2
                        RCVD$C  QT2,RECB2               ;RECEIVE DATA FROM QT2
                ;
                ;   13-WORD PACKET RECEIVED FROM QT2 HAS FOLLOWING CONFIGURATION:
                ;
                ;       FD.FID          - TDCTA'S FID
                ;       FD.FVR          -   "     VERSION NUMBER
                ;       FD.FNB          -   "     FILE NUMBER (-VE IF ERROR FROM QT2)
                ;       SD.FSA          - NUMBER OF FSA STATES GENERATED BY QT2
                ;       SD.SEC          - ELAPSED SECONDS
                ;       SD.TIC          -    "    TICS
                ;
                ;   LOAD TDCTB'S FDSC INTO BATCH STATUS TABLE
                ;
                        MOV     #SNDB2,R0               ;R0->RECEIVED DATA
                        TST     SD.FSA(R0)              ;ERROR FROM QT2?
                        BGE     11$                     ;BRANCH IF NO
                        MOV     SD.FSA(R0),R0           ;R0 = ERROR CODE
                        JMP     NACK                    ;SEND NACK TO MSCHED
                11$:    MOV     BAT.NO,R1               ;R1 = BATCH NUMBER
                        MOV     BSTPTR(R1),R1           ;R1->BATCH STATUS TABLE
                        MOV     FD.FID(R0),B.FFSB(R1)   ;FID
                        MOV     FD.FID+2(R0),B.FFSB+2(R1)
                        MOV     FD.FVR(R0),B.FFSB+4(R1) ;VERSION NUMBER
                        MOV     SD.FSA(R0),B.FSBZ(R1)   ;SIZE OF FSA-B TDCT
                ;
                ;   QT2 IS DONE, RUN QT3
                ;
                        MOV     BAT.NO,SNDB3    ;TRANSMIT TO QT3 THE BATCH NUMBER
                ;
                ;   TRANSMIT 1-WORD PACKET TO QT3
                ;
                        SDAT$C  QT3,SNDB3
                        BCC     6$
                        CALL    DIRERR
                        JMP     EXIT
                6$:     RQST$C  QT3,SNDB3               ;REQUEST QT3 TO BE RUN
                        BCC     7$
                        CALL    DIRERR
                        JMP     EXIT
                ;
```

```
000642
000650
000656  016761  000000C 000104          7$:     SPND$S                          ;WAIT FOR REPLY FROM QT3
000664  005767  000000C                          RCVD$C  QT3,RECB3               ;RECEIVE DATA FROM QT3
000670  002004                                   MOV     SD.FSA+SNDB3,B.FSCZ(R1) ;SIZE OF FSA-C TDCT
000672  016700  000000C                          TST     FD.FID+SNDB3            ;ERROR FROM QT3 ?
000676  000167  000022                          BGE     ACK                     ;BRANCH IF NO
                                                 MOV     FD.FID+SNDB3,R0         ;R0 = ERROR CODE
                                                 JMP     NACK                    ;SEND NACK TO MSCHED
                                         ;
                                         ;   13-WORD PACKET RECEIVED FROM QT3 HAS FOLLOWING CONFIGURATION:
                                         ;
                                         ;       FD.FID          - BATCH NUMBER (-VE IF ERROR RETURN FROM QT3)
                                         ;       SD.FSA          - NUMBER OF FSA STATES GENERATED BY QT1
                                         ;       SD.SEC          - ELAPSED SECONDS
                                         ;       SD.TIC          -    "     TICS
                                         ;
                                         ;   ACK TO XSCHED
                                         ;
                                         ;   GET AN EMPTY PACKET FROM POOL
                                         ;
000702                                   ACK:    CALL    GETFRE                  ;ON RETURN, R2 -> PACKET
                                         ;
                                         ;   BUILD PACKET (ACK)
                                         ;
000706  012762  000003  000002                   MOV     #XQTS,2(R2)             ;COMMAND SOURCE
000714  016762  177100  000004                   MOV     BAT.NO,4(R2)            ;BATCH NUMBER
000722  000415                                   BR      ACK1
                                         ;
                                         ;   NACK TO MSCHED
                                         ;
000724                                   NACK:   CALL    GETFRE
000730  012762  000003  000002                   MOV     #XQTS,2(R2)
000736  112762  000001  000003                   MOVB    #1,3(R2)
000744  016762  177050  000004                   MOV     BAT.NO,4(R2)
000752  010062  000006                          MOV     R0,6(R2)                ;R0 = ERROR CODE
                                         ;
                                         ;   QUEUE PACKET TO BOTTOM OF SSQ
                                         ;
000756                                   ACK1:   GTIM$S  #SECBUF
000770                                           CALL    DELTIM
000774  016705  000206'                          MOV     SECBUF,R5
001000  070527  000074                          MUL     #60.,R5
001004  066705  000210'                          ADD     SECBUF+2,R5
001010  010561  000076                          MOV     R5,B.QTTM(R1)
001014                                           CALL    PUTSSQ
                                         ;
                                         ;   PRINT STATISTICS.
                                         ;
001020  016701  176774                   PRNT:   MOV     BAT.NO,R1       ;GET # QUERIES
001024  016101  000000G                          MOV     BSTPTR(R1),R1
001030  016167  000232  000716                   MOV     B.NQRY(R1),QRYS
001036  012703  001756'                          MOV     #PRTBUF,R3      ;INIT FOR PRINT OUT
001042  012704  001664'                          MOV     #MSGTXT,R4
001046                                           TEXT
001054                                           DECNV   BAT.NO                  ;PRINT BATCH NUMBER
001064                                           TEXT
001072                                           DECNV   QRYS                    ;PRINT # QUERIES
001102                                           TEXT
```

```
001110                            DECNV·   SNDB1+SD.FSA·             ;PRINT·#·QT1 STATES·
001120                            TEXT·
001126                            TIMCNV·  SNDB1+SD.SEC,SNDB1+SD.TIC·        ;PRINT·XLATE TIME·
001152·                           TEXT·
001160                            DECNV·   SNDB2+SD.FSA·             ;SAME·FOR·QT2·
001170                            TEXT·
001176                            TIMCNV·  SNDB2+SD.SEC,SNDB2+SD.TIC·
001222·                           TEXT·
001230                            DECNV·   SNDB3+SD.FSA·             ;AND·QT3
001240                            TEXT·
001246                            TIMCNV·  SNDB3+SD.SEC,SNDB3+SD.TIC·
001272·                           TEXT·
001300                            DECNV·   QTSTAT·           ;PRINT·CLOSE·CODE·
001310                            TEXT·
001316                            DECNV·   QTSTAT+2·         ;PRINT·EMA·ACTUAL·SIZE·
001326                            TEXT·
001334                            DECNV·   QTSTAT+4          ;PRINT·EMA·EST·SIZE·
001344                            TEXT·
001352·                           DECNV·   QTSTAT+6          ;PRINT·EMB·ACTUAL·SIZE·
001362·                           TEXT·
001370                            DECNV·   QTSTAT+10         ;PRINT·EMB·EST·SIZE·
001400                            TEXT·
001406                            DECNV·   SNDB2+SD.NPS·     ;PRINT·FSA-B·NODE·POOL·SIZE·
001416                            TEXT·
001424  005067  000000G·          CLR·     QTSTAT·           ;INIT·QT STATUS·WORDS·
001430  005067  000002G·          CLR·     QTSTAT+2·
001434  005067  000004G·          CLR·     QTSTAT+4
001440  005067  000006G·          CLR·     QTSTAT+6
001444  005067  000010G·          CLR·     QTSTAT+10
001450  162703  001756'           SUB·     #PRTBUF,R3        ;R3=BUFFER·SIZE·
001454                            QIOW$S·  #IO.WLB,#6,#2,,,,<#PRTBUF,R3,#40>        ;PRINT·LINE·
                          ;
                          ;
001524  000167  176462    EXIT:   JMP·     START·
```

```
                                        ; CONVERTION·SUBROUTINES·FOR·QTS·CONSOLE PRINT·OUT
                                        ;
001530  012705  000005        DECASC: MOV·    #5,R5           ;CONVERT DECIMAL·TO·ASCII·
001534  012702· 001604'               MOV·    #100$,R2·
001540  005000                        CLR·    R0
001542· 005305                1$:     DEC·    R5
001544  001413                        BEQ·    10$             ;IF·LAST DIGIT·
001546  071022·                        DIV·    (R2)+,R0        ;R0=ISOLATED·DIGIT
001550  005700                        TST·    R0
001552· 001773                        BEQ·    1$              ;IF·LEADING·ZERO, SUPPRESS·
001554  052700  000060        2$:     BIS·    #60,R0          ;CONVERT AND·XFER·DIGIT·
001560  110023                        MOVB·   R0,(R3)+
001562· 005000                        CLR·    R0
001564  005305                        DEC·    R5
001566  001402·                       BEQ·    10$             ;IF·LAST DIGIT·
001570  071022·                       DIV·    (R2)+,R0        ;R0=ISOLATED·DIGIT
001572· 000770                        BR      2$
001574  052701  000060        10$:    BIS·    #60,R1          ;CONVERT AND·XFER·LAST·DIGIT
001600  110123                        MOVB·   R1,(R3)+
001602· 000207                        RTS·    PC·
001604  023420  001750  000144  100$:   .WORD·  10000.,1000.,100.,10.
001612· 000012·
                                        ;
001614  020027  000011        TIMASC: CMP·    R0,#9.          ;IF·>9 , XFER·"9+"
001620  101014                        BHI·    TIM9
001622· 052700  000060                BIS·    #60,R0          ;CONVERT AND·XFER·
001626  110023                        MOVB·   R0,(R3)+
001630  005000                        CLR·    R0
001632· 112723  000056        TICASC: MOVB·   #'.,(R3)+       ;XFER·"."
001636  071027  000006                DIV·    #6,R0           ;R0=·TENTH·OF·A·SECOND·
001642· 052700  000060                BIS·    #60,R0          ;CONVERT AND·XFER·
001646  110023                        MOVB·   R0,(R3)+
001650  000207                        RTS·    PC·
001652· 112723  000071        TIM9:   MOVB·   #'9,(R3)+
001656  112723  000053                MOVB·   #'+,(R3)+
001662· 000207                        RTS·    PC·
                                        ;
                                        ; BUFFERS·AND·TABLES FOR·PRINT OUT·
                                        ;
001664      102·     101      124  MSGTXT: .ASCIZ· /BAT#/·
001667      043      000
001671      040      121      122·        .ASCIZ· /·QRYS=/·
001674      131      123      075
001677      000
001700      040      121      124         .ASCIZ· /·QT1=/·
001703      061      075      000
001706      050      000                  .ASCIZ· /(/·
001710      051      040      121         .ASCIZ· /) QT2=/·
001713      124      062      075
001716      000
001717      050      000                  .ASCIZ· /(/·
001721      051      040      121         .ASCIZ· /) QT3=/·
001724      124      063      075
001727      000
001730      050      000                  .ASCIZ· /(/·
001732·     051      040      133         .ASCIZ· /) [/·
001735      000
```

```
346 001736      054     000                     .ASCIZ  /,/
347 001740      054     000                     .ASCIZ  /,/
348 001742      054     000                     .ASCIZ  /,/
349 001744      054     000                     .ASCIZ  /,/
350 001746      054     000                     .ASCIZ  /,/
351 001750      135     000                     .ASCIZ  /]/
352 001752      000     000                     .BYTE   0,0     ;SAFETY
353                                             .EVEN
354 001754  000000                      QRYS:   .WORD   0
355 001756                              PRTBUF: .BLKB   80.
```

```
                                        .SBTTL  BLDFNB
                                ;
                                ;   SUBROUTINE TO BUILD 13-WORD DATA PACKET TO BE SENT TO TRANSLATOR TASKS.
                                ;
                                ;   ON ENTRY,
                                ;       R1 = FDSC OFFSET OF EMATRIX
                                ;       R5 = FILE NUMBER OF TDCT
                                ;       R4-> SEND BUFFER
                                ;
                                ;   PACKET LAYOUT:
                                ;
                                ;       N.FID           - EMATRIX FID (2-WORD)
                                ;       N.FID+4         -    "    DEVICE NAME
                                ;       N.FNAM          - TDCT FILENAME
                                ;       N.FTYP          -  "   TYPE
                                ;       N.FVER          - EMATRIX UNIT NUMBER
                                ;       N.STAT          - TDCT DEVICE NAME
                                ;       N.NEXT          - TDCT UNIT NUMBER
                                ;       N.DID           - TDCT UFD'S FID (3-WORD)
                                ;
002076                          BLDFNB: SAVE    R2,R3
002102  016703  175712                  MOV     BAT.NO,R3               ;R3 = BATCH NUMBER
002106  016303  000000G                 MOV     BSTPTR(R3),R3           ;R3-> BATCH STATUS TABLE
002112  060301                          ADD     R3,R1                   ;R1->FDSC OF EMATRIX
002114  012700  000052'                 MOV     #NMBLK-F.FNB,R0         ;R0->FDB
002120                                  CALL    BLDEFL                  ;BUILD FILENAME BLOCK FOR EMATRIX
002124  012702  000154'                 MOV     #NMBLK,R2               ;R2->FNB
002130  016264  000000  000000          MOV     N.FID(R2),N.FID(R4)     ;FID (EMATRIX)
002136  016264  000002  000002          MOV     N.FID+2(R2),N.FID+2(R4)
002144  016264  000032  000004          MOV     N.DVNM(R2),N.FID+4(R4)  ;DEVICE NAME (EMATRIX)
002152  016264  000034  000016          MOV     N.UNIT(R2),N.FVER(R4)   ;UNIT NUMBER (EMATRIX)
                                ;
                                ;   BUILD FNB FOR TDCT
                                ;       R0->FDB
                                ;
002160  010501                          MOV     R5,R1                   ;R1 = FILE NUMBER
002162                                  CALL    BLDNFL                  ;BUILD FNB FOR TDCT
                                ;
                                ;       R4-> SEND BUFFER
                                ;       R2-> NMBLK
                                ;
002166  016264  000006  000006          MOV     N.FNAM(R2),N.FNAM(R4)   ;FILENAME (TDCT)
002174  016264  000010  000010          MOV     N.FNAM+2(R2),N.FNAM+2(R4)
002202  016264  000012  000012          MOV     N.FNAM+4(R2),N.FNAM+4(R4)
002210  016264  000014  000014          MOV     N.FTYP(R2),N.FTYP(R4)   ;FILE TYPE (TDCT)
002216  016264  000032  000020          MOV     N.DVNM(R2),N.STAT(R4)   ;DEVICE NAME (TDCT)
002224  016264  000034  000022          MOV     N.UNIT(R2),N.NEXT(R4)   ;DEVICE UNIT NUMBER (TDCT)
002232  016264  000024  000024          MOV     N.DID(R2),N.DID(R4)     ;UFD'S FID (TDCT)
002240  016264  000026  000026          MOV     N.DID+2(R2),N.DID+2(R4)
002246  016264  000030  000030          MOV     N.DID+4(R2),N.DID+4(R4)
002254                                  RESTOR  R2,R3
002260                                  EXIT    BLDFNB
```

```
                                                .SBTTL  ERROR HANDLER CODE
                                        ;
                                        ;
                                        ;
002262  011667  000160          DIRERR: MOV     (SP),PAR2
002266  016767  000000  000150          MOV     $DSW,PAR1               ;$DSW
002274                                  MOUT$S  #MSG1,#PAR1
002314  000207                          RTS     PC
                                        ;
                                        ;
                                        ;
002316  011667  000122          FCSERR: MOV     (SP),PAR1
002322                                  MOUT$S  #MSG2,#PAR1
002342  000167  177156                  JMP     EXIT
                                        ;
002346  000025                  MSG1:   .WORD   LN1E-LN1
002350  002356'                         .WORD   LN1
002352  000041                  MSG2:   .WORD   LN2E-LN2
002354  002403'                         .WORD   LN2
                                        ;
002356      105     122     122 LN1:    .ASCIZ  /ERROR NO.=%1D,PC=%1O/
002361      117     122     040
002364      116     117     056
002367      075     045     061
002372      104     054     120
002375      103     075     045
002400      061     117     000
002403                          LN1E:
002403      106     103     123 LN2:    .ASCIZ  /FCS ERROR, PC=%1O, ERROR NO.=%1D/
002406      040     105     122
002411      122     117     122
002414      054     040     120
002417      103     075     045
002422      061     117     054
002425      040     105     122
002430      122     117     122
002433      040     116     117
002436      056     075     045
002441      061     104     000
002444                          LN2E:
                                        .EVEN
002444  000000                  PAR1:   .WORD   0
002446  000000                  PAR2:   .WORD   0
        000212'                         .END    START
```

```
ACK     000702R
ACKi    000756R
BAT.NO  000020R
BITVAL= 000000
BIT0  = 000001
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BLDEFL= ****** GX
BLDFNB  002076R
BLDNFL= ****** GX
BSTPTR= ****** GX
BS.CLS= 000002
BS.DBU= 000004
BS.INA= 000000
BS.OPN= 000001
BS.SRC= 000003
BYTE0 = 000000
BYTE1 = 000001
BYTE2 = 000002
BYTE3 = 000003
BYTE4 = 000004
BYTE5 = 000005
BYTE6 = 000006
BYTE7 = 000007
BYTE8 = 000010
BYTE9 = 000011
BYTVAL= 000012
B.BSTA  000054
B.CNTX  000046
B.CQUO  000060
B.FEMA  000132
B.FEMB  000142
B.FEMC  000152
B.FFSA  000202
B.FFSB  000212
B.FFSC  000222
B.FMHR  000172
B.FQLS  000162
B.FSAZ  000100
B.FSBZ  000102
B.FSCZ  000104
B.HBLK  000120
B.HDOC  000114
B.HRLP  000126
B.HRLR  000122
B.HRLW  000124
B.NMBR  000052
B.NQRY  000232
B.QLSZ  000106
B.QMAP  000234
B.QSPL  000316
B.QTTM  000076
B.QUQP  000056
B.SFDB  000010
B.SIZE  000772
B.SNDP  000012
B.SSQ   000004
B.SSQF  000050
B.STAT  000044
B.STTE  000053
B.UDOC  000110
CBIT  = 010000 G
CF.B0 = 000070
CF.B2 = 000067
CF.B4 = 000066
CF.B6 = 000065
CF.DR0= 000064
CF.DR1= 000063
CH.AND= 000001
CR    = 000015
DBSLEN= 000116
DECASC  001530R
DELTIM  000000RG
DH.BF0  000002
DH.BF1  000004
DH.CTL  000000
DH.DMC  000010
DH.FLG  000006
DIRERR  002262R
DN.DCK  000000
DN.NTP  000004
DN.NXT  000006
DN.ROT  000002
DN.SIZ  000010
010 ELSBIT= 010000 G
010 EMXCHF= 020000 G
010 EMXDTF= 010000 G
010 EMXEXF= 000001 G
010 EMXFDC= 002000 G
010 EMXMCD= 000002 G
010 EMXMCF= 001000 G
010 EMXMTV= 040000 G
010 EMXNB1= 000002 G
010 EMXNB2= 000004 G
010 EMXNFD= 000400 G
010 EMXNSQ= 100000 G
010 EMXNVD= 001000 G
010 EMXSZF= 002000 G
010 EMXTRL= 010000 G
010 EMXVDC= 004000 G
010 EMXVVV= 000001 G
010 EMXZNF= 004000 G
010 EMX0VD= 000200 G
010 EOBIT = 010000 G
010 EXIT    001524R
010 FA.APD= 000100
010 FA.CRE= 000010
010 FA.DLK= 001000
010 FA.ENB= 100000
010 FA.EXC= 002000
010 FA.EXT= 000004
010 FA.NSP= 000100
010 FA.POS= 010000
010 FA.RD = 000001
010 FA.RWD= 004000
010 FA.SEQ= 040000
010 FA.SHR= 000040
010 FA.TMP= 000020
    FA.WCK= 020000
    FA.WRT= 000002
    FCSERR  002316R
    FD.BLK= 000010
    FD.CCL= 000002
    FD.COM= 020000
    FD.CR = 000002
    FD.DIR= 000010
    FD.FID  000000
    FD.FNB  000006
    FD.FTN= 000001
014 FD.FVR  000004
005 FD.Fii= 040000
005 FD.INS= 000010
005 FD.ISP= 002000
005 FD.LEN  000010
005 FD.MNT= 100000
    FD.OSP= 004000
013 FD.PLC= 000004
013 FD.PRN= 000004
013 FD.PSE= 010000
013 FD.RAH= 000001
013 FD.RAN= 000002
    FD.REC= 000001
    FD.RWM= 000001
    FD.SDI= 000020
    FD.SQD= 000040
    FD.TTY= 000004
    FD.WBH= 000002
    FF.CHR= 000005
    FF.NV = 000003
    FF.POE= 000002
    FF.RWD= 000001
    FF.RWF= 000006
    FF.SPC= 000004
    FN.DBR  000026
    FN.DBS  000022
    FN.DHR  000040
    FN.EMA  000012
    FN.EMB  000014
    FN.EMC  000016
    FN.FSA  000000
    FN.FSB  000002
    FN.FSC  000004
    FN.LGO  000034
    FN.LGU  000036
    FN.MFO  000024
    FN.MHR  000010
    FN.NMB  000044
    FN.QLS  000006
    FN.QRY  000020
    FN.SF0  000030
    FN.SF1  000032
    FN.SHD  000042
    FO.APD= 000106
    FO.MFY= 000002
    FO.RD = 000001
    FO.UPD= 000006
    FO.WRT= 000016
    F.ACTL= 000076
    F.ALOC= 000040
    F.BBFS= 000062
    F.BDB = 000070
    F.BGBC= 000057
    F.BKDN= 000026
003 F.BKDS= 000020
003 F.BKEF= 000050
    F.BKP1= 000051
003 F.BKST= 000024
    F.BKVB= 000064
    F.CHR = 000075
    F.CNTG= 000034
003 F.DFNB= 000046
    F.DSPT= 000044
    F.DVNM= 000134
    F.EFBK= 000010
    F.EFN = 000050
    F.EOBB= 000032
    F.ERR = 000052
    F.FACC= 000043
    F.FFBY= 000014
    F.FNAM= 000110
    F.FNB = 000102
    F.FTYP= 000116
    F.FVER= 000120
    F.HIBK= 000004
    F.LUN = 000042
    F.MBCT= 000054
    F.MBC1= 000055
    F.MBFG= 000056
    F.NRBD= 000024
    F.NREC= 000030
011 F.OVBS= 000030
011 F.RACC= 000016
011 F.RATT= 000001
011 F.RCNM= 000034
011 F.RCTL= 000017
011 F.RSIZ= 000002
011 F.RTYP= 000000
011 F.SEQN= 000100
011 F.SPDV= 000072
011 F.SPUN= 000074
011 F.STBK= 000036
011 F.UNIT= 000136
011 F.URBD= 000020
011 F.VBN = 000064
011 F.VBSZ= 000060
011 GETFRE= ****** GX
011 GTIM1   000220RG    014
011 G.TICP= 000016
011 G.TICT= 000014
    G.TIDA= 000004
    G.TIHR= 000006
    G.TIMI= 000010
    G.TIMO= 000002
    G.TISC= 000012
    G.TIYR= 000000
    HOUR.1= 000020
    HOUR.2= 000006
    IO.WLB= ****** GX
    IWBIT = 004000 G
    JMPBIT= 004000 G
    LF    = 000012
    LNi     002356R
    LNiE    002403R
    LN2     002403R
    LN2E    002444R
    L$STAT= 000006 G
    M     = 000062
    MIN.1 = 000022
    MIN.2 = 000010
    MSGOUT= ****** GX
    MSGTXT  001664R
    MSG1    002346R
    MSG2    002352R
    N     = 000002
    NACK    000724R
    NB.DEV= 000200
    NB.DIR= 000100
    NB.NAM= 000004
    NB.SD1= 000400
    NB.SD2= 001000
    NB.SNM= 000040
    NB.STP= 000020
    NB.SVR= 000010
    NB.TYP= 000002
    NB.VER= 000001
    NMBLK   000154R
    N.BFAC= 000004
    N.BHGH= 000006
    N.BTCH= 000004
    N.BUFB= 004000
    N.BUFW= 002000
    N.DID = 000024
```

QUERY TRANSACTOR SCHEDULER MACRO M1110 27-MAR-80 17:19 PAGE 1
SYMBOL TABLE

```
N.DVNM= 000032          R.VAR = 000002          SR.SUN  000000      002 S.DABA= 000006      WN.SRC  000000      012
N.FID = 000000          R.VDBA= 000006          SR.TWS  000056      002 S.DAEF= 000010      WN.TYP  000001      012
N.FNAM= 000006          R.VDTN= 000002          SR.WSL  000052      002 S.DATN= 000002      WORD0 = 000000
N.FOS = 000764          R.VXBA= 000006          SR.YR   000004      002 S.FATT= 000016      WORD1 = 000002
N.FTYP= 000014          R.VXTN= 000002          SR.1IN  000024      002 S.FDB = 000140      WORD2 = 000004
N.FVER= 000016          SD.FSA= 000010 G    003 SR.1IP  000016      002 S.FNAM= 000006      WORD3 = 000006
N.NEXT= 000022          SD.NPS= 000016 G    003 SSBIT = 004000 G        S.FNB = 000036      WORD4 = 000010
N.PKSZ= 000020          SD.SEC= 000012 G    003 SS.FID  000002      004 S.FNBW= 000017      WORD5 = 000012
N.PKTS= 000043          SD.TIC= 000014 G    003 SS.FNB  000010      004 S.FNTY= 000004      WORD6 = 000014
N.QURY= 000031          SECBUF  000206RG    014 SS.FVR  000006      004 S.FTYP= 000002      WORD7 = 000016
N.STAT= 000020          SEC.1 = 000024          SS.LEN  000012      004 S.HRL = 000240      WORD8 = 000020
N.SUNT= 000002          SEC.2 = 000012          SS.STT  000000      004 S.NFEN= 000020      WORD9 = 000022
N.UNIT= 000034          SEG1  = 000000 G        STARSW= 000000          S1BIT = 001000 G    WRDVAL= 000024
PAR1    002444R         SEG2  = 000002 G        START   000212R         S2BIT = 002000 G    XBATCH= 000013
PAR2    002446R         SEG3  = 000004 G        ST$CNG= 120000 G        S3BIT = 004000 G    XDBLOA= 000004
PRNT    001020R         SNDB1   000026R         ST$INR= 060000 G        TAEBIT= 020000 G    XDBPRO= 000012
PRTBUF  001756R         SNDB2   000064R         ST$INX= 040000 G        TBIT  = 004000 G    XDMCIN= 000006
PUTSSQ= ****** GX       SNDB3   000122R         ST$JSQ= 100000 G        TICASC  001632R     XFOSMR= 000007
QE.ROI= 000144          SR.ARE  000114      002 ST$MAT= 160000 G        TIC.1 = 000026      XGTSRE= 000014
QRYS    001754R         SR.ARS  000106      002 ST$SEQ= 140000 G        TIC.2 = 000014      XHITSK= 000011
QTSTAT= ****** GX       SR.DAY  000010      002 ST.ASZ  000020      006 TIMASC  001614R     XHLMER= 000002
QT1     000000R         SR.DLT  000014      002 ST.BSZ  000024      006 TIM9    001652R     XHOTSK= 000010
QT2     000004R         SR.ECB  000047      002 ST.BTC  000000      006 TXTBIT= 010000 G    XMSCHE= 000000
QT3     000010R         SR.ECH  000046      002 ST.CSZ  000030      006 T.JSBY= 000000 G    XQTS  = 000003
Q.FDSC  000004      007 SR.ECL  000050      002 ST.HRL  000010      006 T.MATC= 000000 G    XQT0  = 000001
Q.NOBK  000000      007 SR.FIB  000012      002 ST.LEN  000044      006 T.NBAS= 000002 G    XSCHED  000014R
Q.NUHL  000002      007 SR.GRE  000100      002 ST.QRY  000002      006 T.NDEF= 000000 G    XSULOA= 000005
Q.SIZE  000014      007 SR.GRS  000072      002 ST.QSZ  000034      006 T.SBY1= 000000 G    $DSW  = 000000
RECB1   000022R         SR.LEN  000122      002 ST.SCH  000040      006 T.SBY2= 000001 G    $$$   = 000140R    015
RECB2   000060R         SR.LIN  000066      002 ST.UHL  000004      006 T.SBY3= 000002 G    $$$ARG= 000002
RECB3   000116R         SR.LIP  000062      002 ST.XLT  000014      006 T.STAD= 000002 G    $$$OST= 000010
R.FIX = 000001          SR.MON  000006      002 SU.DBU= 000004          T.TRAN= 000000 G    $$$T1 = 000003
R.QSGC= 000015          SR.NDC  000042      002 SU.DON= 000006          T.TYPW= 000004 G    ...GBL= 000000
R.QSPC= 000014          SR.NDS  000036      002 SU.IDL= 000000      WN.NTP  000004      012 ...PC1= 000000
R.QSPN= 000006          SR.NIN  000030      002 SU.LOD= 000001      WN.NXT  000006      012 ...PC2= 000170R
R.QSPR= 000012          SR.NIP  000022      002 SU.SRC= 000002      WN.ROT  000002      012 ...PC3= 000000
R.QSTN= 000002          SR.SDB  000032      002 SU.SRR= 000005      WN.SIZ  000010      012 ...TPC= 000140
R.SEQ = 000003          SR.SRC  000002      002 SU.XPD= 000003
```

```
. ABS.  000000      000
        002450      001
SRCOFF  000122      002
FDSCOF  000010      003
SUSOFF  000012      004
DHROFF  000012      005
STTOFF  000044      006
QSPLOF  000014      007
BSTOFF  000772      010
FNOFFS  000044      011
WNODOF  000010      012
DNODOF  000010      013
TRNOFF  000240      014
$DPB$$  000150      015
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  6397 WORDS  ( 25 PAGES)
DYNAMIC MEMORY:  7028 WORDS  ( 27 PAGES)
ELAPSED TIME:  00:00:54
```

QUERY·TRANSLATOR·SCHEDULER·
SYMBOL·TABLE·

MACRO M1110 27-MAR-80 17:10 PAGE B0-5

QTS,@TS/-SP=[20,1]P,M,T,QTS·

```
QTS.TSK;30   MEMORY ALLOCATION MAP  TKB                    PAGE  1
                    27-MAR-80   16:33

TASK    NAME     : QTS
PARTITION NAME   : HSTSPR
IDENTIFICATION   : 0331
TASK  UIC        : [20,3]
STACK     LIMITS: 000212 001211 001000 00512.
PRG XFR ADDRESS: 001424
TOTAL ADDRESS WINDOWS: 2.
TASK  IMAGE  SIZE    : 1728. WORDS
TASK ADDRESS LIMITS: 000000 006507
R-W DISK BLK LIMITS: 000002 000010 000007 00007.

*** ROOT SEGMENT: QTS

R/W MEM  LIMITS: 000000 006507 006510 03400.
DISK BLK LIMITS: 000002 000010 000007 00007.

MEMORY ALLOCATION SYNOPSIS:

SECTION                                          TITLE   IDENT   FILE
-------
. BLK.:(RW,I,LCL,REL,CON) 001212 003340 01760.
                          001212 002450 01320.  QUERY           QTS.OBJ;1
BSTOFF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000.  QUERY           QTS.OBJ;1
DHROFF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000.  QUERY           QTS.OBJ;1
DNODOF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000.  QUERY           QTS.OBJ;1
FDSCOF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000.  QUERY           QTS.OBJ;1
FNOFFS:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000.  QUERY           QTS.OBJ;1
MSGOUT:(RW,I,LCL,REL,CON) 004552 001214 00652.
                          004552 001214 00652.  MESSAG          MSGOUT.OBJ;1
QSPLOF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000.  QUERY           QTS.OBJ;1
SRCOFF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000.  QUERY           QTS.OBJ;1
STTOFF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000.  QUERY           QTS.OBJ;1
SUSOFF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000.  QUERY           QTS.OBJ;1
TRNOFF:(RW,I,LCL,REL,CON) 005766 000240 00160.
                          005766 000240 00160.  QUERY           QTS.OBJ;1
UNODOF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000.  QUERY           QTS.OBJ;1
$DPB$$:(RW,I,LCL,REL,CON) 006226 000150 00104.
                          006226 000150 00104.  QUERY           QTS.OBJ;1
$$RESL:(RW,I,LCL,REL,CON) 006376 000112 00074.
```

GLOBAL SYMBOLS:

CBIT 010000	EMXMTV 040000	EMXVVV 000001	SD.FSA 000010	ST$CNG 120000	TAEBIT 020000	T.SBY3 000002	
DELTIM 005766-R	EMXNB1 000002	EMXZNF 004000	SD.NPS 000016	ST$INR 060000	TBIT 004000	T.STAD 000002	
ELSBIT 010000	EMXNB2 000004	EMX0VD 000200	SD.SEC 000012	ST$INX 040000	TXTBIT 010000	T.TRAN 000000	
EMXCHF 020000	EMXNFD 000400	EOBIT 010000	SD.TIC 000014	ST$JSQ 100000	T.JSBY 000000	T.TYPW 000004	
EMXDTF 010000	EMXNSQ 100000	GTIMI 006206-R	SECBUF 006174-R	ST$MAT 160000	T.MATC 000000	$EDMSG 005202-R	
EMXEXF 000001	EMXNVD 001000	IWBIT 004000	SEG1 000000	ST$SEQ 140000	T.NBAS 000002		
EMXFDC 002000	EMXSZF 002000	JMPBIT 004000	SEG2 000002	S1BIT 001000	T.NDEF 000000		
EMXMCD 000002	EMXTRL 010000	L$STAT 000006	SEG3 000004	S2BIT 002000	T.SBY1 000000		
EMXMCF 001000	EMXVDC 004000	MSGOUT 004754-R	SSBIT 004000	S3BIT 004000	T.SBY2 000001		

*** TASK BUILDER STATISTICS:

TOTAL WORK FILE REFERENCES: 7144.
WORK FILE READS: 0.
WORK FILE WRITES: 0.
SIZE OF CORE POOL: 6634. WORDS (25. PAGES)
SIZE OF WORK FILE: 1536. WORDS (6. PAGES)

ELAPSED TIME:00:00:08

TABLE·OF·CONTENTS·

```
                        ;THIS FILE (E.MAC) CONTAINS ALL THE QUERY ERROR CODES
                        ;
000001                  QE.DW1=1        ;RESERVED
000002                  QE.MTS=2        ;MULTIPLE TERM SEP'S OR MISSING TERM
000003                  QE.DW2=3        ;RESERVED
000004                  QE.ILC=4        ;ILLEGAL CHARACTER
000005                  QE.IHS=5        ;INTERNAL HSTS ERROR, LOGIC PROCESSING
000006                  QE.NLQ=6        ;NULL QUERY
000007                  QE.MOP=7        ;MISSING OPERATOR
000010                  QE.MFM=8.       ;MISPLACED FLU-MOD
000011                  QE.MSD=9.       ;MISPLACED SUBDOC OPERATOR
000012                  QE.MNT=10.      ;MISPLACED NOT OPERATOR
000013                  QE.MFL=11.      ;MISSING FLU / TWO CONSECUTIVE OPERATORS
000014                  QE.UBP=12.      ;UNBALANCED PARENTHESES
000015                  QE.INO=13.      ;INSUFFICIENT NUMBER OF OPERANDS
000016                  QE.ISO=14.      ;ILLEGAL SUBELEMENT OPERAND
000017                  QE.PX1=15.      ;MISSING PROX, WINDOW, OR DOC TYPE,ZONE, OR SUBZONE
000020                  QE.PX2=16.      ;PROX, WINDOW OR FLU-MOD ID TOO LARGE
000021                  QE.PX3=17.      ;MISSING PROX, UNIT
000022                  QE.PX4=18.      ;MISSING PROX, DELIMITER
000023                  QE.NRI=19.      ;ILLEGAL NUMERICAL RANGE SPECIFICATION
000024                  QE.NRB=20.      ;NUMERICAL RANGE BORDERED BY NUMERIC
000025                  QE.NOS=21.      ;NO STX FOUND IN QUERY
000026                  QE.FTB=22.      ;FLU TOO BIG
000027                  QE.NTK=23.      ;UNDEFINED TOKEN
                        ;
                        ;QE.RO1=100.    ;GENERAL RESOURCE OVERFLOW (LABEL DEFINED IN M.MAC)
000145                  ASKOVF=101.     ;ARGUMENT STACK OVERFLOW
000146                  LNPOVF=102.     ;LOGIC NODE POOL OVERFLOW
000147                  TSKOVF=103.     ;TOKEN STACK OVERFLOW
000150                  QBFOVF=104.     ;QUERY TOO BIG
000156                  QRYOVF=110.     ;# QUERIES OVERFLOWED
000157                  POLOVF=111.     ;FLU POOL OVERFLOW
000160                  FALOVF=112.     ;FAL
000161                  TTBOVF=113.     ;TTABLE
000162                  XTBOVF=114.     ;XTABLE
000163                  QLBOVF=115.     ;QLB
000164                  SLBOVF=116.     ;SDLB
000165                  QEXOVF=117.     ;QEX
000166                  EMAOVF=118.     ;EMA
000167                  EMBOVF=119.     ;EMB
000170                  EMCOVF=120.     ;EMC
000171                  VI3OVF=121.     ;VI QT3
000172                  VI2OVF=122.     ;VI QT2
000173                  VI1OVF=123.     ;VI QT1
000174                  TDBOVF=124.     ;TDCTB
000175                  NDBOVF=125.     ;NODE POOL QT2
                        ;
```

```
                        ;
                        ; MACRO TO PRINT BUFFER
                        ; CREATES SPOOL FILE RECORDS
                        ;
                        .MACRO  PRT,START,END
                                MOV     R4,-(SP)
                                MOV     START,R4
                                MOV     R4,LOCAT
                                MOV     END,ENDLOC
                                JSR     PC,PRINT
                                MOV     (SP)+,R4
                        .ENDM
                        .MACRO  PRTH,START,END
                                MOV     R4,-(SP)
                                MOV     START,R4
                                MOV     R4,LOCAT
                                MOV     END,ENDLOC
                                JSR     PC,PRNTH
                                MOV     (SP)+,R4
                        .ENDM
                        ;
                        ; MACRO TO TEST CHARACTER SIGNIFICANCE
                        ; PARAMETERS ARE CONDITION,TRUE,AND FALSE
                        ;       CONDITION=THE TEST CONDITION
                        ;       TRUE    =PATH TO TAKE IF CONDITION TRUE
                        ;       FALSE   =PATH TO TAKE IF CONDITION FALSE
                        ; ANY PARAMETER CAN BE PROCEEDED BY "!" WHICH CAUSES THAT
                        ;  PARAMETER TO BE A SUBROUTINE CALL
                        ;
```

```
                                        .TITLE QT0
000000                                  .PSECT  QT0
                                        .MCALL OFNB$R,FDBDF$,FDRC$A,FDRC$A,FDOP$A,FDBF$A,NMBLK$,FDAT$A
                                        .MCALL FSRSZ$,READ$,GET$S,CLOSE$,FDBK$A,WAIT$,PRINT$,PUT$S,OFNB$W
                                        .MCALL SDAT$S,RSUM$S,FINIT$,RCVD$S,SPND$S,OPEN$W,WAIT$,WRITE$
                                        .MCALL FDAT$R,DELET$,EXIT$S
000003                          QRYLUN=3
000003                          QRYEFN=3
000000     117     120     105  OPFAIL: .ASCII /OPEN FAILURE ON INPUT FILE/
000032                          OPFSIZ=.-OPFAIL
                                        .EVEN
                                        FSRSZ$ 0,,QT0
000032                          FDBQRY: FDBDF$
000172                                  FDRC$A  FD,RWM
000172                                  FDBK$A  QRYBUF,N,BUFB,,QRYEFN,IOSB
000172                                  FDOP$A  QRYLUN
000172                          QLSFDB: FDBDF$
000332                                  FDRC$A  FD,RWM
000332                                  FDBK$A  QLS,QLSIZ,,QRYEFN,IOSB
000332                                  FDOP$A  QRYLUN
000332                          TSKNAM::.BLKW   2
000336                          CMDCDE::.BLKW   1
000340                          BATNO:: .BLKW   1
000342                          QRYNO:: .BLKW   1
000344                          ERRCDE::.BLKW   1
000346                                  .BLKW   9.
000370  000000                  IOSB::  .WORD   0
000372  000000                  .MOLUN::.WORD   0
000374  000000                  FLUID:: .WORD   0
000376  000000                  ZERO::  .WORD   0       ;CONTENTS ALWAYS =0
000400  000000                  AUTCLS: .WORD   0       ;AUTO CLOSE FLAG
000402  000000                  QRYSIZ: .WORD   0
000404  000000                  SAVSTK: .WORD   0       ;STACK POINTER FOR ERROR RECOVERY
000406  000000                  QRYFS:  .WORD   0               ;FLU ID AT START OF QRY
                                ;
                                ;
                                ;  QUERY RETURN CODES
                                ;
000001                          QRYOK   ==1     ;1.0 QUERY PARSED AND MERGED OK
002001                          QRYBAD  ==2001  ;1.4 QUERY IS BAD, NOT MERGED
001001                          CLSOLY  ==1001  ;1.2 BATCH CLOSED ONLY
001401                          CLSIN   ==1401  ;1.3 QUERY MERGED AND BATCH CLOSED
000401                          BATOVF  ==401   ;1.1 BATCH OPEN, LAST QUERY WOULD NOT FIT
002401                          CORUPT  ==2401  ;1.5 BATCH IS CORRUPTED BY LAST QUERY
003001                          CORUNK  ==3001  ;1.6 BATCH IS CORRUPTED, UNKNOWN SOURCE
003401                          BATABD  ==3401  ;1.7 BATCH ABORTED
                                ;
000400                          CLBREQ  ==400   ;0.1 CLOSE BATCH REQUEST
                                ;
```

DISPATCH·ROUTINE·

```
                                        .SBTTL  DISPATCH ROUTINE
                                ;
                                ;   THIS ROUTINE IS USED TO DEQUEUE NODES SENT TO QT0 VIA SEND DIRECTIVES
                                ;   AND, BASED ON THE SENDER TASK'S NAME, SELECT THE APPROPRIATE
                                ;   PROCESSING ROUTINE.
                                ;
000410                          START:  FINIT$
000414  010667  177764                  MOV     SP,SAVSTK       ;SAVE STACK POINTER
000420                          START3::CALL    BATINI          ;INITIALIZE DATA STRUCTURES
                                        .IF     DF,QT0SUM
                                        OPEN$W  #LIST,#LTLUN,,,#PRTBUF,#BSIZ            ;OPEN PRINT FILE
                                        PUT$S   #LIST,#HEADER,#HDRSIZ
                                        BCC     START1
                                        MOVB    F.ERR(R0),R1
                                        MOV     R1,PAR2
                                        CALL    FCSERR
                                        JMP     EXIT
                                        .ENDC
000424                          START1:
000424                                  RCVD$S  ,#TSKNAM
000442  103024                          BCC     2$
000444  026727  000000G 000000G         CMP     $DSW,#IE.ITS    ;NO DATA CURRENTLY QUEUED?
000452  001404                          BEQ     3$              ;BRANCH IF TRUE
000454                                  CALL    DIRERR
000460  000167  003704                  JMP     EXIT
000464                          3$:
000464                                  SPND$S                  ;SUSPEND
000472  000167  177726                  JMP     START1
000476  012767  001001  177632  1$:     MOV     #CLSOLY,CMDCDE          ;CLOSE ONLY
000504  005067  177670                  CLR     AUTCLS
000510  000167  000652                  JMP     CLSBAT
000514                          2$:
000514  005267  177660                  INC     AUTCLS
000520  026727  177612  000400        CMP     CMDCDE,#CLBREQ  ;WHICH COMMAND?
000526  002417                          BLT     START2          ;PROCESS QUERY
000530  001762                          BEQ     1$              ;CLOSE BATCH
000532  012767  003401  177576          MOV     #BATABD,CMDCDE  ;ABORT BATCH
000540  005067  177600                  CLR     ERRCDE
000544  016706  177634        BATREJ::MOV     SAVSTK,SP       ;RESTOR STACK
                                        .IF     DF,QT0SUM
                                        DELET$  #LIST
                                        .ENDC
000550  105067  177565                  CLRB    BATNO+1
000554                                  CALL    SSO
000560  000717                          BR      START3
```

```
                                        .SBTTL  READ QUERY
                                ;
                                ;   NODE RECEIVED FORM MSCHED,  LOAD FILE DESCRIPTOR
                                ;   INFORMATION CONTAINED IN RECEIVE NODE INTO FILENAME BLOCK OF UQR
                                ;
000562  000167  003550          TOOBIG: JMP     ERROR5
000566                          START2:
000566  016701  177550                  MOV     QRYNO,R1        ;STORE QUERY NAME
000572  016767  000000G 000000G         MOV     QRYMAX,QRYMS    ;STORE CURRENT QUERY MAXIMUM
000600  020167  000000G                 CMP     R1,QRYMAX       ;IS CURRENT QRYNO > CURRENT MAX?
000604  003412                          BLE     11$             ;NO
000606  010167  000000G                 MOV     R1,QRYMAX       ;YES: UPDATE MAX
000612  020127  000000G                 CMP     R1,#QRYMSZ      ;QCL OVERFLOW?
000616  101405                          BLOS    11$             ;NO
000620  012767  000156  177516         MOV     #QRYOVF,ERRCDE  ;YES: NON-FATAL
000626  000167  002040                 JMP     ERRORV
000632  010100                  11$:    MOV     R1,R0           ;POINT THIS QUERY TO CURRENT FAL ADDRESS
000634  006300                          ASL     R0
000636  062700  000000G                 ADD     #QCL,R0
000642  016710  000000G                 MOV     FALAD,(R0)
000646  010067  177534                  MOV     R0,QRYFS        ;TEMP STORAGE FOR QCL POINTER
000652  016700  177462                  MOV     BATNO,R0        ;FIND FDSC ENTRY IN TABLE
000656  016000  000000G                 MOV     BSTPTR(R0),R0
000662  062700  000316                  ADD     #B,QSPL,R0
000666  016701  177450                  MOV     QRYNO,R1
000672  070127  000014                  MUL     #Q.SIZE,R1
000676  060001                          ADD     R0,R1
000700  016167  000000  177474         MOV     Q.NQBK(R1),QRYSIZ       ;STORE QUERY SIZE
000706  005761  000002                  TST     Q.NUHL(R1)      ;ANY UHL BLOCKS?
000712  001403                          BEQ     2$              ;NO
000714  052777  100000  177464         BIS     #BIT15,@QRYFS   ;SET UHL FLAG IN QCL POINTER
000722  062701  000004          2$:     ADD     #Q.FDSC,R1      ;STEP TO FDSC ENTRY
000726  012700  000032'                 MOV     #FDBQRY,R0
000732  004767  000000G                 JSR     PC,BLDEFL       ;BUILD EXISTING FILE FNB
000736                                  OFNB$R                  ;OPEN QUERY SPOOL FILE
000750  103545                          BCS     4$
000752                                  READ$                   ;READ QUERY FILE
000756  103542                          BCS     4$
000760                                  WAIT$
000764  005367  177412                  DEC     QRYSIZ          ;ANY MORE BLOCKS LEFT OF QUERY?
000770  001274                          BNE     TOOBIG          ;YES
000772  012701  000010G                 MOV     #QRYBUF+10,R1   ;R1<-QUERY BUFFER
000776  122721  000000G          110$:  CMPB    #STX,(R1)+
001002  001375                          BNE     110$
001004  020127  000001G                 CMP     R1,#QRYEND+1    ;WITHIN QUERY BUFFER?
001010  001510                          BEQ     7$              ;NO: NO STX FOUND
001012  012767  000000G 000000G         MOV     #NULFMN,ACTFMN  ;RESET ACTIVE FLU-MOD TO NULL
001020  016767  000000G 000000G         MOV     NXTNDE,QRYPS    ;STORE CURRENT POOL ADDRESS IN CASE OF ERROR
001026  016767  177342  177352         MOV     FLUID,QRYFS     ;STORE CURRENT FLU ID
001034  016767  000000G 000000G         MOV     NFALE,QRYFAS    ;STORE CURRENT FAL LOCATION
001042  016767  000000G 000000G         MOV     NFALE,CFALS     ;SET CURRENT FAL ELEMENT START
001050  062767  000002  000000G         ADD     #2,NFALE        ;STEP PAST COUNTER
001056  005077  000000G                 CLR     @CFALS          ;CLEAR COUNTER
001062  005067  002742                  CLR     EMCOSZ          ;INIT QUERY STATISTICS STORAGE
001066  005067  002740                  CLR     EMAQSZ
001072  004767  000000G                 JSR     PC,PARSER       ;STX FOUND: PARSE QUERY
001076                                  CLOSE$  #FDBQRY         ;CLOSE QUERY SPOOL FILE
```

```
154 001106  026727  000000G 000000C          CMP     NXTNDE,#POLEND+FNPOSZ+FNPOSZ
155 001114  103053                          BHIS    9$                  ;IF NODE POOL OVERFLOW (FATAL)
156
157 001116  004767  000000G                 JSR     PC,QTFORM           ;TRANSFORM LOGIC
158 001122  022702  000000C                 CMP     #QNFLU!QNFLS,R2     ;IS LOGIC FALSE
159 001126  001416                          BEQ     10$                 ;       THEN IGNORE
160 001130  022702  000000G                 CMP     #QNFLU,R2           ;IS LOGIC TRUE
161 001134  001474                          BEQ     6$                  ;       THEN SET UP DUMMY QLS
162 001136  026727  177232  000000C         CMP     FLUID,#TTBMSZ/4
163 001144  103405                          BLO     8$
164 001146  012767  000161  177170          MOV     #TTBOVF,ERRCDE      ;IF TTABLE OVERFLOW (NON-FATAL)
165 001154  000167  001512                  JMP     ERRORV
166 001160  004767  000000G         8$:     JSR     PC,GENQLS           ;GENERATE QLS FOR CURRENT QUERY
167 001164  016767  000000G 000000G 10$:     MOV     NFALE,FALAD         ;UPDATE FAL ADDRESS
168 001172  162767  000000G 000000G          SUB     #FAL,FALAD
169 001200  004767  000470                  JSR     PC,RESCHK           ;CHECK FOR RESOURCE OVERFLOW
170 001204  022767  001401  177124          CMP     #CLSIN,CMDCDE       ;CLOSE BATCH ALSO?
171 001212  001465                          BEQ     CLSBAT              ;YES
172 001214  012767  000001  177114          MOV     #QRYOK,CMDCDE       ;RESULT= QUERY MERGED SUCCESFULLY
173 001222                                  CALL    SSQ                 ;SEND SSQ PACKET TO MSCHED
174 001226  000167  177172                  JMP     START1
175 001232  012767  000025  177104  7$:     MOV     #QE.NOS,ERRCDE      ;REPORT ERROR
176 001240  000167  002742                  JMP     ERROR
177 001244  012767  041462  000000G 9$:     MOV     #"2C,QLSBUF         ;RE-INIT QLSBUF
178 001252  012767  000157  177064          MOV     #POLOVF,ERRCDE
179 001260  000167  002672                  JMP     ERRORF
180 001264                          4$:
181 001264  116767  176614  003412          MOVB    FDBQRY+F.ERR,PAR2
182 001272  052767  177400  003404          BIS     #177400,PAR2
183 001300  004767  002620                  JSR     PC,FCSERR
184 001304  000167  003060                  JMP     EXIT
185 001310                          5$:
186 001310  012767  000000G 003366          MOV     #$DSW,PAR2
187 001316  004767  002542                  JSR     PC,DIRERR
188 001322  000167  003042                  JMP     EXIT
189 001326  016705  000000G         6$:     MOV     NQLBE,R5            ;CREATE DUMMY QLB (NOT WITH NO POINTER)
190 001332  062705  000002                  ADD     #2,R5               ;STEP TO QRY ELEMENT COUNT
191 001336  112725  000001                  MOVB    #1,(R5)+            ;# ELEMENTS=1
192 001342  010567  000000G                 MOV     R5,NQLBE            ;UPDATE NEXT QLB ELEMENT POINTER
193 001346  112725  000000G                 MOVB    #B$NOT,(R5)+        ;FLAG ELEMENT AS NOT
194 001352  116715  176764                  MOVB    QRYNO,(R5)          ;STORE QRY ID
195 001356  062767  000003  000000G         ADD     #3,QLBAD            ;UPDATE QLB INTERNAL OFFSET
196 001364  000677                          BR      10$
197 001366                          CLSBAT:
198 001366  056767  002460  000000G         BIS     OVFCLS,QTSTAT               ;NOTE ANY OVERFLOW ERROR
199 001374  005067  002452                  CLR     OVFCLS
200 001400  016767  002430  000004G         MOV     EMATSZ,QTSTAT+4             ;STORE EMA AND EMB EST
201 001406  016767  002426  000010G         MOV     EMBTSZ,QTSTAT+10
202 001414  005767  000000G                 TST     NODEA               ;IF NO QUERY, DON'T CLOSE
203 001420  001020                          BNE     100$
204 001422  005767  000000G                 TST     NODEB
205 001426  001015                          BNE     100$
206 001430  005767  176744                  TST     AUTCLS              ;IF AUTO CLOSE, RETURN (1,5), ELSE (1,6)
207 001434                                  BOFF    101$
208 001436  012767  002401  176672          MOV     #CORUPT,CMDCDE
209 001444  000167  177074                  JMP     BATREJ
210 001450  012767  003001  176660  101$:   MOV     #CORUNK,CMDCDE      ;ABORT BATCH
```

```
211 001456  000167  177062                      JMP     BATREJ
212 001462  004767  000000G         100$:       JSR     PC,GENTX        ;GENERATE TTABLE AND XTABLE
213                                             .IF     DF,QT0SUM
214                                             JSR     PC,BATSUM       ;PRINT BATCH SUMMARY
215                                             JSR     PC,QLSSUM       ;PRINT QLS SUMMARY
216                                             .ENDC
217 001466  062767  000002  000000G             ADD     #2,QLBAD        ;CONVERT QLB,SDLB,QEX ADDRESSES TO BYTE SIZES
218 001474  062767  140004  000000G             ADD     #4-40000.SDLBAD
219 001502  016767  176632  000000G             MOV     BATNO,QLSBAT    ;XFER BATCH NUMBER
220 001510  042767  000001  000000G             BIC     #1,SDLBAD
221 001516  006367  000000G                     ASL     QEXAD
222 001522  022767  000006  000000G             CMP     #6,QEXAD        ;ANY ENTER OTHER THAN DUMMIES?
223 001530  001002                              BNE     10$             ;       YES
224 001532  005067  000000G                     CLR     QEXAD           ;       NO--THEN RESET
225 001536  012700  000172'         10$:        MOV     #QLSFDB,R0      ;CREATE QLS FILE
226 001542  012701  000006                      MOV     #FN.QLS,R1
227 001546                                      FDAT$R  ,,,,#8.
228 001554  004767  000000G                     JSR     PC,BLDNFL
229 001560                                      OFNB$W
230 001572  103426                              BCS     1$
231 001574  016702  176540                      MOV     BATNO,R2                ;R2 = BATCH NUMBER
232 001600  016202  000000G                     MOV     BSTPTR(R2),R2           ;R2->BATCH STATUS TABLE
233 001604  016762  176464  000162              MOV     QLSFDB+F.FNB+N.FID,B.FQLS(R2)
234 001612  016762  176460  000164              MOV     QLSFDB+F.FNB+N.FID+2,B.FQLS+2(R2)
235 001620  016762  176466  000166              MOV     QLSFDB+F.FNB+N.FVER,B.FQLS+4(R2)
236 001626                                      WRITE$
237 001632  103406                              BCS     1$
238 001634                                      WAIT$
239 001640                                      CLOSE$
240 001644  000167  000000G                     JMP     CLSEMX
241 001650  116767  000052G 003026  1$:         MOVB    EMXFDB+F.ERR,PAR2
242 001656  052767  177400  003020              BIS     #177400,PAR2
243 001664  004767  002234                      JSR     PC,FCSERR
244 001670  000167  002474                      JMP     EXIT
245 001674                          RESCHK:
246 001674  016705  176474                      MOV     FLUID,R5        ;R5=# TTABLE ENTRIES
247 001700  005205                              INC     R5
248 001702  005002                              CLR     R2
249 001704  012703  000002G                     MOV     #TTABLE+2,R3    ;FIRST TTABLE COUNTER
250 001710  062302                  1$:         ADD     (R3)+,R2        ;SUM COUNTERS
251 001712  062703  000002                      ADD     #2,R3
252 001716  077504                              SOB     R5,1$
253 001720  020227  000000C                     CMP     R2,#XTBMSZ/2-TRMCUT     ;XTABLE PAST CUT-OFF?
254 001724  103416                              BLO     3$                      ;NO
255 001726  020227  000000C                     CMP     R2,#XTBMSZ/2            ;YES: OVERFLOWED?
256 001732  103405                              BLO     2$                      ;NO
257 001734  012767  000162  176402              MOV     #XTBOVF,ERRCDE          ;YES
258 001742  000167  002210                      JMP     ERRORF
259 001746  012767  001401  176362  2$:         MOV     #CLSIN,CMDCDE   ;FLAG TO CLOSE AND WHY
260 001754  012767  000000G 000000G             MOV     #FLUCLS,QTSTAT
261 001762  026727  176406  000000C 3$:         CMP     FLUID,#TTBMSZ/4-TRMCUT  ;TTABLE PAST CUT-OFF?
262 001770  103406                              BLO     4$
263 001772  012767  001401  176336              MOV     #CLSIN,CMDCDE
264 002000  012767  000000G 000000G             MOV     #FLUCLS,QTSTAT
265 002006  026727  000000G 000000C 4$:         CMP     FALAD,#FALMSZ-FLUCUT    ;FAL PAST CUT-OFF?
266 002014  103406                              BLO     5$
267 002016  012767  001401  176312              MOV     #CLSIN,CMDCDE
```

```
268 002024  012767  000000G 000000G
269 002032  026727  000000G 000000G 5$:
270 002040  103406
271 002042  012767  001401  176266
272 002050  012767  000000G 000000G
273 002056  026727  000000G 000000C 6$:
274 002064  103417
275 002066  026727  000000G 000000G
276 002074  103405
277 002076  012767  000163  176240
278 002104  000167  002046
279 002110  012767  001401  176220  7$:
280 002116  012767  000000G 000000G
281 002124  026727  000000G 000000C 10$:
282 002132  103417
283 002134  026727  000000G 000000C
284 002142  103405
285 002144  012767  000164  176172
286 002152  000167  002000
287 002156  012767  001401  176152  11$:
288 002164  012767  000000G 000000G
289 002172  026727  000000G 000000C 12$:
290 002200  103417
291 002202  026727  000000G 000000C
292 002210  103405
293 002212  012767  000165  176124
294 002220  000167  001732
295 002224  012767  001401  176104  13$:
296 002232  012767  000000G 000000G
297 002240  026767  000000G 177771G 14$:
298 002246  103417
299 002250  026767  000000G 000000G
300 002256  103405
301 002260  012767  000171  176056
302 002266  000167  001664
303 002272  012767  001401  176036  15$:
304 002300  012767  000000G 000000G
305 002306  016705  001516          16$:
306 002312  005067  001512
307 002316  070527  000003
308 002322  066705  001500
309 002326  010567  001474
310 002332  020527  000000C
311 002336  103416
312 002340  020527  000000G
313 002344  103405
314 002346  012727  000170  000344'
315 002354  000167  001576
316 002360  012767  001401  175750  17$:
317 002366  012767  000000G 000000G
318 002374  016705  001432          20$:
319 002400  005067  001426
320 002404  066705  001424
321 002410  010567  001420
322 002414  020527  000000C
323 002420  103416
324 002422  020527  000000C

MOV     #FLUCLS,QTSTAT
CMP     NXTNDE,#POLEND          ;NODE POOL PAST CUT-OFF?
BLO     6$
MOV     #CLSIN,CMDCDE
MOV     #ELSCLS,QTSTAT
CMP     QLBAD,#QLBMSZ-<FLUCUT*2>        ;QLB PAST CUT-OFF?
BLO     10$                             ;NO
CMP     QLBAD,#QLBMSZ                   ;YES: QLB OVERFLOW?
BLO     7$                              ;NO
MOV     #QLBOVF,ERRCDE                  ;YES
JMP     ERRORF
MOV     #CLSIN,CMDCDE
MOV     #LOGCLS,QTSTAT
CMP     SDLBAD,#SLBMSZ!40000-FLUCUT     ;SDLB PAST CUTOFF?
BLO     12$                             ;NO
CMP     SDLBAD,#SLBMSZ!40000            ;YES: OVERFLOW?
BLO     11$                             ;NO
MOV     #SLBOVF,ERRCDE                  ;YES
JMP     ERRORF
MOV     #CLSIN,CMDCDE
MOV     #LOGCLS,QTSTAT
CMP     QEXAD,#QEXMSZ!100000/2-FLUCUT   ;QEX PAST CUTOFF?
BLO     14$                             ;NO
CMP     QEXAD,#QEXMSZ!100000/2          ;YES: OVERFLOW?
BLO     13$                             ;NO
MOV     #QEXOVF,ERRCDE                  ;YES
JMP     ERRORF
MOV     #CLSIN,CMDCDE
MOV     #LOGCLS,QTSTAT
CMP     NODEC,VEC3MX-7          ;QT3 VECTORS PAST CUTOFF?
BLO     16$                     ;NO
CMP     NODEC,VI3MSZ            ;YES: OVERFLOW?
BLO     15$                     ;NO
MOV     #VI3OVF,ERRCDE          ;YES
JMP     ERRORF
MOV     #CLSIN,CMDCDE
MOV     #ELSCLS,QTSTAT
MOV     EMCQSZ,R5               ;R5=EMC SIZE
CLR     EMCQSZ
MUL     #3,R5
ADD     EMCTSZ,R5
MOV     R5,EMCTSZ
CMP     R5,#VEC3MX-7*21./2      ;EMC PAST CUT-OFF?
BLO     20$
CMP     R5,#EMCMSZ              ;OVERFLOW?
BLO     17$
MOV     #EMCOVF,#ERRCDE
JMP     ERRORF
MOV     #CLSIN,CMDCDE
MOV     #ELSCLS,QTSTAT
MOV     EMAQSZ,R5               ;R5=EMA SIZE
CLR     EMAQSZ
ADD     EMATSZ,R5
MOV     R5,EMATSZ
CMP     R5,#EMAMSZ/2-EMACUT     ;EMA PAST CUTOFF?
BLO     22$
CMP     R5,#EMAMSZ/2            ;OVERFLOW?
```

```
    325 002426  103405                          BLO·    21$
    326 002430  012767  000166  175706          MOV·    #EMAOVF,ERRCDE
    327 002436  000167  001514                  JMP·    ERRORF·
    328·002442· 012767  001401  175666  21$:    MOV·    #CLSIN,CMDCDE·
    329 002450  012767  000000G·000000G·        MOV·    #EMACLS,QTSTAT
    330 002456  026727  000000G·000000C·22$:    CMP·    NODEA,#VEC1MX-TRMCUT·   ;VI·IN·QT1 PAST·LIMIT?·
    331 002464  103417                          BLO·    24$
    332·002466  026727  000000G·000000G·        CMP·    NODEA,#VEC1MX·          ;OVERFLOW?
    333 002474  103405                          BLO·    23$
    334 002476  012767  000173  175640          MOV·    #VI1OVF,ERRCDE
    335 002504  000167  001446                  JMP·    ERRORF·
    336 002510  012767  001401  175620  23$:    MOV·    #CLSIN,CMDCDE·
    337 002516  012767  000000G·000000G·        MOV·    #ELSCLS,QTSTAT
    338 002524  016705  001306          24$:    MOV·    EMBQSZ,R5               ;R5=EMB·SIZE·
    339 002530  005067  001302                  CLR·    EMBQSZ·
    340 002534  066705  001300                  ADD·    EMBTSZ,R5
    341 002540  010567  001274                  MOV·    R5,EMBTSZ·
    342·002544  020527  000000C·                CMP·    R5,#EMBMSZ-EMBCUT·      ;PAST·CUTOFF?·
    343 002550  103416                          BLO·    26$
    344 002552· 020527  000000G·                CMP·    R5,#EMBMSZ·             ;OVERFLOW?
    345 002556  103405                          BLO·    25$
    346 002560  012767  000167  175556          MOV·    #EMBOVF,ERRCDE
    347 002566  000167  001364                  JMP·    ERRORF·
    348 002572· 012767  001401  175536  25$:    MOV·    #CLSIN,CMDCDE·
    349 002600  012767  000000G·000000G·        MOV·    #EMBCLS,QTSTAT
    350 002606  016705  001230          26$:    MOV·    VIBQSZ,R5               ;R5=VI·FSAB·SIZE·
    351 002612· 005067  001224                  CLR·    VIBQSZ·
    352·002616  066705  001222                  ADD·    VIBTSZ,R5
    353 002622· 010567  001216                  MOV·    R5,VIBTSZ·
    354 002626  020527  000000C·                CMP·    R5,#VEC2MX-VIBCUT·      ;PAST·CUTOFF?·
    355 002632· 103416                          BLO·    30$
    356 002634  020527  000000G·                CMP·    R5,#VI2MSZ·             ;OVERFLOW?
    357 002640  103405                          BLO·    27$
    358 002642· 012767  000172  175474          MOV·    #VI2OVF,ERRCDE
    359 002650  000167  001302                  JMP·    ERRORF·
    360 002654  012767  001401  175454  27$:    MOV·    #CLSIN,CMDCDE·
    361 002662· 012767  000000G·000000G·        MOV·    #ELSCLS,QTSTAT
    362·002670                          30$:
    363 002670  000207                          RTS·    PC·
    364 002672· 012767  000401  175436  ERRORV::MOV·    #BATOVF,CMDCDE
    365 002700  016767  175440  001144          MOV·    ERRCDE,OVFCLS·
    366 002706  016706  175472                  MOV·    SAVSTK,SP·      ;RESTORE STACK·POINTER·
    367 002712· 004767  002246                  JSR·    PC,ORYDEL·
    368 002716                                  CALL·   SSO·
    369 002722· 000167  175476                  JMP·    START1
    370                                 ;
    371                                 ;
    372·                                ; SUBROUTINE·TO·CALCULATE·SIZE OF·TERM'S RESOURCE·UTILIZATION·
    373                                 ;
    374 002726                          SZCAL1::
    375 002726  016702· 000000G·                MOV·    ACTFMN,R2·      ;GET·FLU-MOD·COUNT
    376 002732· 116202· 000000G·                MOVB·   NDSIZ(R2),R2·
    377 002736  042702· 177400                  BIC·    #177400,R2·
    378 002742· 062702· 000002                  ADD·    #2,R2·          ;BIAS·FOR·INTERWORD·AND·MATCH·CODE·
    379 002746  005767  001074                  TST·    CHRCNT·         ;NEW·FLU?·
    380 002752· 003025                          BGT·    2$              ;YES·
    381 002754  001412·                         BEQ·    1$              ;NO: NEW FLU-MOD·(FIRST)
```

```
 382·002756  005302·
 383 002760  010267  001062
 384 002764  006302·
 385 002766  066702·  001054
 386 002772·  005767  001052
 387 002776  100055
 388 003000  100425
 389 003002·  010267  001040          1$:
 390 003006  006302·
 391 003010  066702·  001032
 392·003014  005302·
 393 003016  005767  001026
 394 003022·  100043
 395 003024  100413
 396 003026  005767  001016          2$:
 397 003032·  100013
 398 003034  066702·  001006
 399 003040  010267  001002
 400 003044  006302·
 401 003046  066702·  000774
 402·003052·  005302·
 403 003054  060267  000752          3$:
 404 003060  000207
 405 003062·  032767  000000G·000000G·  4$:
 406 003070
 407 003072·  066702·  000750          7$:
 408 003076  010267  000744
 409 003102·  006302·
 410 003104  066702·  000736
 411 003110  005302·
 412·003112·  032767  000000G·000000G·
 413 003120
 414 003122·  005267  000714
 415 003126  005267  000710          6$:
 416 003132·  060267  000700          5$:
 417 003136  000207
 418 003140                          10$:
 419 003140
 420 003150  116405  000000G·
 421 003154  042705  177400
 422·003160  062704  000000G··
 423 003164  005267  000652
 424 003170  005067  000662
 425 003174  005067  000660
 426 003200  005067  000656
 427 003204  005067  000000G·
 428·003210  005067  000640
 429 003214  112401
 430 003216  100016
 431 003220  022701  000000G·
 432·003224  001411
 433 003226  022701  000000G·
 434 003232·  001424
 435 003234  022701  000000G·
 436 003240  001005
 437 003242·  005267  000574
 438 003246  000402·

DEC·    R2·             ;NO: NEW FLU-MOD·
MOV·    R2,CHRCNT·      ;MUL·BY·3 (WORDS/EMA·ENTRY)
ASL·    R2·
ADD·    CHRCNT,R2·
TST·    VDCCNT·         ;FSA-A·OR·-B?·
BPL·    5$              ;FSA-B·
BMI·    3$              ;FSA-A·
MOV·    R2,CHRCNT·      ;MUL·BY·3
ASL·    R2·
ADD·    CHRCNT,R2·
DEC·    R2·
TST·    VDCCNT·         ;FSA-A·OR·FSA-B?·
BPL·    5$              ;FSA-B·
BMI·    3$              ;FSA-A·
TST·    VDCCNT·         ;A·OR·B?
BPL·    4$              ;B·
ADD·    CHRCNT,R2·      ;FSA-A: R2=ENTRIES IN·EMA·
MOV·    R2,CHRCNT·      ;MUL·BY·3
ASL·    R2·
ADD·    CHRCNT,R2·
DEC·    R2·             ;REMOVE·MATCH·CODE EXTRA·WORD·
ADD·    R2,EMAQSZ·      ;ACCUMALATE·EMA·SIZE·FOR·QRY
RTS·    PC·
BIT·    #NMRNGF,FLUTYP  ;NUMBER·RANGE?·
BEQ·    10$             ;YES·
ADD·    CHRCNT,R2·      ;NO: R2=#EMB·ENTRIES·
MOV·    R2,CHRCNT·      ;MUL·BY·3
ASL·    R2·
ADD·    CHRCNT,R2·
DEC·    R2·             ;ADJUST·FOR·MATCH·CODE·
BIT·    #INIVDC,FLUTYP
BEQ·    6$              ;IF·INIT VLDC, STEP·VI·COUNT BY·2·
INC·    VIBQSZ·
INC·    VIBQSZ·
ADD·    R2,EMBQSZ·      ;ACCUMALATE·EMB·SIZE·FOR·QRY
RTS·    PC·

SAVE·   R0,R1,R2,R4
MOVB·   NDSIZ(R4),R5    ;GET·#·CHAR'S·
BIC·    #177400,R5
ADD·    #NDCHR,R4       ;POSITION·TO·CHAR'S·
INC·    VIBQSZ·         ;ALWAYS·1 VECTOR·
CLR·    EXPCNT·         ;#·VECTORS·IN·RANGE·
CLR·    EMBCNT·         ;COUNT·1ST·CHAR·
CLR·    FLG9
CLR·    INCLUD·
CLR·    EXPSAV·
MOVB·   (R4)+,R1        ;GET·1ST CHAR·
BPL·    11$             ;IF·LEADING·CHAR·
CMP·    #ZVLDC,R1
BEQ·    12$             ;IF·LEADING·ZVLDC·
CMP·    #SEGOP,R1
BEQ·    13$             ;IF·LEADING·SEG-OP
CMP·    #VLDC,R1
BNE·    11$             ;IF·ELSE
INC·    VIBQSZ·         ;IF·LEADING·VLDC·
BR      11$
```

```
 439 003250  005267  000000G·          12$:    INC·    INCLUD·          ;INCLUDE RANGE·VECTORS·IN·INIT·VECTOR·
 440 003254  005305                    11$:    DEC·    R5
 441 003256  001563                            BEQ·    30$              ;IF·NO·MORE·CHARS·
 442·003260  005267  000574                    INC·    EMBCNT·          ;ACC·EMB SIZE·
 443 003264  112401                            MOVB·   (R4)+,R1         ;GET·NEXT·CHAR·
 444 003266  100003                            BPL·    14$
 445 003270  022701  000000G·                  CMP·    #SEGOP,R1
 446 003274  001413                            BEQ·    20$              ;IF·START·OF·NUMBER·RANGE·
 447 003276  005067  000000G·          14$:    CLR·    INCLUD·          ;RANGE·VECTORS·ISOLATED·FOR·INIT·
 448 003302·  000764                           BR      11$
 449 003304  005267  000000G·          13$:    INC·    INCLUD·          ;ADJUST·FOR·MULTI-RANGE·
 450 003310  062767  000002  000524            ADD·    #2,VIBOSZ·
 451 003316  062767  000004  000534            ADD·    #4,EMBCNT·
 452·003324  112467  000000G·          20$:    MOVB·   (R4)+,RNGCDE·    ;GET·RANGE·CODE·
 453 003330  005002·                           CLR·    R2·
 454 003332·  012700  000002                   MOV·    #2,R0
 455 003336  032767  000000G·000000G·         BIT·    #PD,RNGCDE·      ;ADJUST·FOR·ANY·DECIMAL·BIAS
 456 003344                                    BON·    21$
 457 003346  032767  000000G·000000G·         BIT·    #DNUM,RNGCDE·
 458 003354                                    BOFF·   22$
 459 003356  005267  000476                    INC·    EMBCNT·
 460 003362·  062767  000002  000470  21$:    ADD·    #2,EMBCNT·
 461 003370  012703  000071            22$:    MOV·    #'9,R3           ;SET·UP·SPECIAL·CHAR·
 462·003374  032767  000000G·000000G·         BIT·    #MR,RNGCDE·
 463 003402·                                   BOFF·   23$
 464 003404  012703  000060                    MOV·    #'0,R3           ;IF·MULTI-RANGE, CHANGE·TO·HIGH·
 465 003410  004767  000244            23$:    JSR·    PC,SZNR·         ;CHECK·LOW·RANGE·
 466 003414  010267  000000G·                  MOV·    R2,LSZ·
 467 003420  005002·                           CLR·    R2·
 468 003422·  012700  000002                   MOV·    #2,R0
 469 003426  004767  000226                    JSR·    PC,SZNR·         ;CHECK·HIGH·RANGE·
 470 003432·  010267  000000G·                  MOV·    R2,HSZ·
 471 003436  032767  000000G·000000G·         BIT·    #MR,RNGCDE·
 472·003444                                    BON·    24$              ;IF·MULTI-RANGE·
 473 003446  010201                            MOV·    R2,R1
 474 003450  005301                            DEC·    R1
 475 003452·  003402·                          BLE·    29$
 476 003454  060167  000400                    ADD·    R1,EMBCNT·       ;ADJUST·FOR·NFLDC'S·
 477 003460  166702·  000000G·          29$:    SUB·    LSZ,R2·
 478 003464  005302·                           DEC·    R2·
 479 003466  003440                            BLE·    27$
 480 003470  060267  000362                    ADD·    R2,EXPCNT·
 481 003474  006302·                           ASL·    R2·              ;ADJUST·FOR·MID·RANGE·TERMS·
 482·003476  060267  000356                    ADD·    R2,EMBCNT·
 483 003502·  000432·                          BR      27$
 484 003504  026767  000000G·000000G·24$:     CMP·    LSZ,HSZ·
 485 003512·  002402·                          BLT·    25$
 486 003514  016702·  000000G·                 MOV·    LSZ,R2·
 487 003520  060267  000334            25$:    ADD·    R2,EMBCNT·       ;ADJUST·FOR·NFLDC'S·
 488 003524  016702·  000000G·                 MOV·    LSZ,R2·          ;ADJUST·FOR·MID·RANGE·TERMS·
 489 003530  005302·                           DEC·    R2·
 490 003532·  003405                           BLE·    26$
 491 003534  060267  000316                    ADD·    R2,EXPCNT·
 492·003540  006302·                           ASL·    R2·
 493 003542·  060267  000312                   ADD·    R2,EMBCNT·
 494 003546  016702·  000000G·          26$:    MOV·    HSZ,R2·
 495 003552·  005302·                          DEC·    R2·
```

READ·QUERY

```
 496 003554  003405                        BLE··    27$
 497 003556  060267  000274                ADD·     R2,EXPCNT·
 498 003562· 006302·                       ASL·     R2·
 499 003564  060267  000270                ADD·     R2,EMBCNT·
 500 003570  016702· 000262        27$:    MOV·     EXPCNT,R2·       ;SAVE·EXPCNT·
 501 003574  005767  000000G·              TST·     INCLUD·
 502·003600  001402·                       BEQ·     28$
 503 003602· 060267  000234                ADD·     R2,VIBOSZ·       ;INCLUDE·RANGE·VECTORS·IN·INIT·
 504 003606  005202·               28$:    INC·     R2·
 505 003610  060267  000240                ADD·     R2,EXPSAV·
 506 003614  005067  000236                CLR·     EXPCNT·
 507 003620  005067  000000G·              CLR·     INCLUD·
 508 003624  000613                        BR       11$
 509 003626                        30$:    RESTOR·  R0,R1,R2,R4
 510 003636  060267  000216                ADD·     R2,EMBCNT·       ;MUL·BY·3
 511 003642· 016702· 000212                MOV·     EMBCNT,R2·
 512·003646  006302·                       ASL·     R2·
 513 003650  066702· 000204                ADD·     EMBCNT,R2·
 514 003654  000167  177252                JMP·     5$
 515                                ;
 516                                ;   SUBROUTINE·TO·PREDICT·EMB·AND·VIB·SIZES·FOR·NUMERICAL·RANGE·TERMS·
 517                                ;
 518                                ;REGISTER·USEAGE·IS·AS·FOLLOWS:
 519                                ;       R0=BIAS·(#·EMB·ENTRIES FOR·CHAR)
 520                                ;       R1=CHAR·
 521                                ;       R2=CHAR·COUNT·PRIOR·TO DEC-PT·
 522·                               ;       R3=SPECIAL·CHAR·THAT·DOESN'T·COUNT·AS·MUCH
 523                                ;       R4=INPUT·CHAR·POINTER·
 524                                ;       R5=INPUT·CHAR·COUNT·REMAINING·
 525                                ;
 526 003660  005305                SZNR:   DEC·     R5               ;GET·NEXT·CHAR·(R5=CHAR·COUNT)
 527 003662· 112401                        MOVB·    (R4)+,R1
 528·003664  100417                        BMI·     10$              ;IF·SEG-OP·OR·DEC-PT·
 529 003666  005202·                       INC·     R2·              ;COUNT·CHARS·
 530 003670  020301                        CMP·     R3,R1            ;SPECIAL CHAR?·
 531 003672· 001410                        BEQ·     1$               ;YES·
 532·003674  005267  000156                INC·     EXPCNT·          ;NO·
 533 003700  060067  000154                ADD·     R0,EMBCNT·       ;ACC·EMB SIZE·
 534 003704  005200                        INC·     R0
 535 003706  005067  000150                CLR·     FLG9
 536 003712· 000762·                       BR       SZNR
 537 003714  005200                1$:     INC·     R0
 538 003716  005267  000140                INC·     FLG9
 539 003722· 000756                        BR       SZNR
 540 003724  005767  000132        10$:    TST·     FLG9             ;WAS·PRIOR·CHAR·SPECIAL?·
 541 003730                                BOFF·    11$              ;NO·
 542·003732·  060067  000122               ADD·     R0,EMBCNT·       ;YES: ADJUST·
 543 003736  005367  000116                DEC·     EMBCNT·
 544 003742· 005267  000110                INC·     EXPCNT·
 545 003746  005067  000110                CLR·     FLG9
 546 003752· 022701  000000G·      11$:    CMP·     #DECPT,R1        ;IS·CHAR A·DEC-PT?
 547 003756  001022·                       BNE·     20$              ;NO·
 548 003760  005200                        INC·     R0
 549 003762· 005305                12$:    DEC·     R5               ;GET·NEXT·CHAR·
 550 003764  111401                        MOVB·    (R4),R1
 551 003766  100756                        BMI·     10$              ;IF·SEG-OP·
 552·003770  020301                        CMP·     R3,R1            ;SPECIAL CHAR?·
```

READ·QUERY

```
553 003772· 001410                      BEQ·    13$             ;YES·
554 003774  005267  000056              INC·    EXPCNT·         ;NO: COUNT·VECTORS
555 004000  060067  000054              ADD·    R0,EMBCNT·      ;ACC·EMB SIZE·
556 004004  005200                      INC·    R0
557 004006  005067  000050              CLR·    FLG9
558 004012· 000763                      BR      12$
559 004014  005200              13$:    INC·    R0
560 004016  005267  000040              INC·    FLG9
561 004022· 000757                      BR      12$
562·004024  000207              20$:    RTS·    PC·
563 004026  000000              EMCTSZ::.WORD·  0
564 004030  000000              EMCQSZ::.WORD·  0
565 004032· 000000              EMAQSZ::.WORD·  0
566 004034  000000              EMATSZ::.WORD·  0
567 004036  000000              EMBQSZ::.WORD·  0
568 004040  000000              EMBTSZ::.WORD·  0
569 004042· 000000              VIBQSZ::.WORD·  0
570 004044  000000              VIBTSZ::.WORD·  0
571 004046  000000              CHRCNT::.WORD·  0
572·004050  000000              VDCCNT::.WORD·  0
573 004052· 000000              OVFCLS::.WORD·  0
574 004054  000000              EXPSAV: .WORD·  0
575 004056  000000              EXPCNT: .WORD·  0
576 004060  000000              EMBCNT: .WORD·  0
577 004062· 000000              FLG9:   .WORD·  0
```

```
                                        .SBTTL  ERROR HANDLING ROUTINE
                                        .NLIST  MEB
                                ;
                                ;       DIRECTIVE ERROR
                                ;
004064  011667  000612          DIRERR::MOV     (SP),PAR1               ;PC AT DIRECTIVE ERROR
004070  016767  000000G 000606          MOV     $DSW,PAR2               ;$DSW
004076                                  MOUT$S  #MSG1,#PAR1
004116  005726                          TST     (SP)+
004120  000167  000244                  JMP     EXIT                    ;EXIT
                                ;
                                ;       FCS ERROR
                                ;
004124  011667  000552          FCSERR::MOV     (SP),PAR1               ;PC
004130                                  MOUT$S  #MSG2,#PAR1
004150  005726                          TST     (SP)+                   ;RESTORE STACK
004152  000167  000212                  JMP     EXIT
                                ;
                                ;       OTHER ERRORS
                                ;
004156  016767  174162  000516  ERRORF::MOV     ERRCDE,PAR1
004164                                  MOUT$S  #MSG6,#PAR1
004204  000471                          BR      EXIT
                                ;
                                ; BAD QUERY
                                ;
004206  016767  174132  000466  ERROR:: MOV     ERRCDE,PAR1
004214  016706  174164                  MOV     SAVSTK,SP
004220  016700  174114                  MOV     BATNO,R0                ;GET EQID
004224  016000  000000G                 MOV     BSTPTR(R0),R0
004230  062700  000234                  ADD     #B,QMAP,R0
004234  066700  174102                  ADD     QRYNO,R0
004240  066700  174076                  ADD     QRYNO,R0
004244  011067  000434                  MOV     (R0),PAR2
004250                                  MOUT$S  #MSG3,#PAR1
004270  004767  000670          REMQRY: JSR     PC,QRYDEL               ;BACK OUT FLU'S FROM POOL
004274  012767  002001  174034        MOV     #QRYBAD,CMDCDE          ;SEND BAD QRY MSG BACK TO CALLER
004302                                  CALL    SSQ                     ;SEND PACKET TO MSCHED
004306  005067  174032                  CLR     ERRCDE                  ;ZERO OUT ERROR CODE
004312  005067  177514                  CLR     EMAQSZ                  ;RE-INIT STAT
004316  005067  177514                  CLR     EMBQSZ
004322  005067  177514                  CLR     VIBQSZ
004326  005067  177476                  CLR     EMCQSZ
004332  000167  174066                  JMP     START1
                                ;
                                ;       QUERY TOO BIG
                                ;
004336                          ERROR5:
004336  012767  000150  174000        MOV     #QBFOVF,ERRCDE
004344  016706  174034                MOV     SAVSTK,SP               ;RESTORE STACK POINTER
004350                                  MOUT$S  #MSG5
004366  000740                          BR      REMQRY
                                ;
                                ;       CLEAN UP FOR ERROR RESTART
                                ;
004370                          EXIT::  CLOSE$  #FDBQRY
                                        .IF     DF,QT0SUM
```

```
                                        PRINT$  #LIST
                                        .ENDC
004400  016706  174000                  MOV     SAVSTK,SP
004404  005767  173770                  TST     AUTCLS          ;IF AUTO CLOSE, RETURN (1,5), ELSE (1,6)
004410                                  BOFF    1$
004412  012767  002401  173716          MOV     #CORUPT,CMDCDE
004420  000403                          BR      2$
004422  012767  003001  173706  1$:     MOV     #CORUNK,CMDCDE  ;RESULT= CORRUPT BATCH
004430                          2$:     CALL    SSQ             ;SEND SSQ PACKET TO MSCHED
004434                                  EXIT$S
                                ;
                                ;   ERROR MESSAGES ARE PRINTED ON TI BY MO:...
                                ;   STRING DESCRIPTORS
                                ;
004442  000041                  MSG1:   .WORD   LN1E-LN1                ;LENGTH OF FORMAT STRING
004444  004466'                         .WORD   LN1                     ;ADDS OF FORMAT STRING
004446  000040                  MSG2:   .WORD   LN2E-LN2
004450  004527'                         .WORD   LN2
004452  000037                  MSG3:   .WORD   LN3E-LN3
004454  004567'                         .WORD   LN3
004456  000017                  MSG5:   .WORD   LN5E-LN5
004460  004626'                         .WORD   LN5
004462  000035                  MSG6:   .WORD   LN6E-LN6
004464  004645'                         .WORD   LN6
                                ;
                                ;   FORMAT STRINGS
                                ;
004466      104     111     122 LN1:    .ASCIZ  /DIR. ERROR, PC = %10, $DSW = %1D/
004527                          LN1E:
004527      106     103     123 LN2:    .ASCIZ  /FCS  ERROR, PC = %10, ERR = %1D/
004567                          LN2E:
004567      121     125     105 LN3:    .ASCIZ  /QUERY BAD, ERROR=%1D  EQID=%1D/
004626                          LN3E:
004626      121     125     105 LN5:    .ASCIZ  /QUERY TOO BIG!/
004645                          LN5E:
004645      102     101     124 LN6:    .ASCIZ  /BATCH CORRUPTED, ERROR = %1D/
004702                          LN6E:
                                        .EVEN
                                ;
                                ;   PARAMETERS
                                ;
004702  000000                  PAR1::  .WORD   0                       ;PC
004704  000000                  PAR2::  .WORD   0                       ;$DSW OR F.ERR
                                ;
004706  005067  173462          BATINI: CLR     FLUID
004712  005067  177110                  CLR     EMCTSZ
004716  005067  177112                  CLR     EMATSZ
004722  005067  177112                  CLR     EMBTSZ
004726  005067  177112                  CLR     VIBTSZ
004732  005067  000000G                 CLR     NODEA
004736  005067  000000G                 CLR     NODEB
004742  005067  000000G                 CLR     NODEC
004746  012767  000000G 000000G         MOV     #POOL,NXTNDE
004754  012705  000000G                 MOV     #FLIXSZ,R5
004760  012704  000000G                 MOV     #FLUIDX,R4
004764  005024                  1$:     CLR     (R4)+
004766  077502                          SOB     R5,1$
```

```
 693 004770  012705  000000G·                MOV·    #CPIXSZ,R5
 694 004774  012704  000000G·                MOV·    #CWPIDX,R4
 695 005000  005024                  2$:     CLR·    (R4)+
 696 005002· 077502·                         SOB·    R5,2$
 697 005004  012704  000000G·                MOV·    #NULFMN,R4      ;RESET·FLU·MOD·
 698 005010  010467  000000G·                MOV·    R4,ACTFMN·
 699 005014  010467  000000G·                MOV·    R4,FMDIDX·
 700 005020  005024                          CLR·    (R4)+
 701 005022· 005024                          CLR·    (R4)+
 702·005024  012704  000000G·                MOV·    #TTABLE,R4      ;CLEAR·TTABLE·
 703 005030  012705  000000G·                MOV·    #QLB,R5
 704 005034  160405                          SUB·    R4,R5
 705 005036  006205                          ASR·    R5
 706 005040  005024                  10$:    CLR·    (R4)+
 707 005042· 077502·                         SOB·    R5,10$
 708 005044  005067  000000G·                CLR·    PRXFLG·
 709 005050  005067  000000G·                CLR·    NOTFLG·
 710 005054  005067  000000G·                CLR·    SDFLG·
 711 005060  012767  177776G·000000G·        MOV·    #QLB-2,NQLBE·
 712·005066  012767  177776  000000G·        MOV·    #-2,QLBAD·
 713 005074  012767  177775G·000000G·        MOV·    #SDLB-3,SDLBE·
 714 005102· 012767  037775  0000000G·       MOV·    #40000-3,SDLBAD·
 715 005110  012767  000006G·000000G·        MOV·    #QEX+6,NAQEX·
 716 005116  012704  000000G·                MOV·    #QEX,R4         ;TRANSFER·DUMMY·QEX·ENTRIES·
 717 005122· 012724  000000G·                MOV·    #QEXFWV,(R4)+
 718 005126  012724  000000G·                MOV·    #QEXFSV,(R4)+
 719 005132· 012724  000000G·                MOV·    #QEXFPV,(R4)+
 720 005136  012767  100003  000000G·        MOV·    #100003,QEXAD·
 721 005144  012767  000000G·000000G·        MOV·    #FAL,NFALE·
 722·005152· 005067  000000G·                CLR·    FALAD·
 723 005156  005067  000000G·                CLR·    QRYMAX·
 724 005162· 000207                          RTS·    PC·
 725                                 ;
 726                                 ; SUBROUTINE·TO·BACK OUT·FLU·NODES·FROM·POOL·
 727                                 ;
 728·005164  016700  173152          QRYDEL::MOV·    QRYNO,R0        ;CLEAR·QCL·FAL·POINTER·
 729 005170  006300                          ASL·    R0
 730 005172· 005060  000000G·                CLR·    QCL(R0)
 731 005176  016767  000000G·000000G·        MOV·    QRYMS,QRYMAX·   ;RESET·QUERY·MAXIMUM·
 732·005204  016767  000000G·000000G·        MOV·    QRYFAS,NFALE·   ;RESET·FAL·LOCATION·
 733 005212· 016700  000000G·                MOV·    QRYPS,R0        ;R0 =·NODE·ADDRESS AT·WHICH·TO·START·DELETION·
 734 005216  010001                          MOV·    R0,R1           ;R1 =·SAME, BUT·ADJUSTED·FOR FMEN'S·
 735 005220  162701  000000G·                SUB·    #NDFLID,R1
 736 005224  012702· 177777                  MOV·    #-1,R2·         ;ALL·NODE·TYPES·
 737 005230  012703  000000G·                MOV·    #FLUIDX,R3      ;SET·UP·TO·SCAN·FOR·FLU·NODES·
 738 005234  012704  000376'                 MOV·    #ZERO,R4
 739 005240  012705  000000G·                MOV·    #FLIXSZ,R5
 740 005244  004767  000000G·        10$:    JSR·    PC,GETNDE·      ;GET·NODE·
 741 005250  103434                          BCS·    100$            ;NO·MORE NODES·
 742·005252· 020400                          CMP·    R4,R0           ;IS·NODE PAST·CUT·OFF?·
 743 005254  103016                          BHIS·   40$             ;YES: DELETE·
 744 005256                                  SAVE·   R4,R3           ;NO: CHECK·FMEN·LINKS·
 745 005262· 016403  000000G·        20$:    MOV·    NDFMEN(R4),R3   ;GET·LINK·
 746 005266  001406                          BEQ·    31$             ;NONE·
 747 005270  020301                          CMP·    R3,R1           ;PAST·CUT·OFF?·
 748 005272· 103002·                         BHIS·   30$             ;YES·
 749 005274  010304                          MOV·    R3,R4           ;NO: TRY NEXT·
```

ERROR HANDLING ROUTINE

```
750 005276  000771                          BR      20$
751 005300  005064  000000G         30$:    CLR     NDFMEN(R4)          ;DELETE REST OF FMEN CHAIN
752 005304                         31$:    RESTORE R4,R3               ;FINISHED FMEN'S
753 005310  000755                         BR      10$
754 005312  126427  000000G 000000G 40$:    CMPB    NDTYP(R4),#NDFSAB       ;WHICH NODE TYPE
755 005320  001003                         BNE     41$                 ;FSA-A
756 005322  005367  000000G                DEC     NODEB               ;FSA-B: ADJUST NODE COUNT
757 005326  000402                         BR      42$
758 005330  005367  000000G         41$:    DEC     NODEA
759 005334  004767  000116          42$:    JSR     PC,DELNDE           ;DELETE NODE
760 005340  000741                         BR      10$
761 005342  012703  000000G         100$:   MOV     #CWPIDX,R3          ;SET UP FOR CWP NODE SCAN
762 005346  012704  000376'                MOV     #ZERO,R4
763 005352  012705  000000G                MOV     #CPIXSZ,R5
764 005356  004767  000000G         110$:   JSR     PC,GETNDE           ;GET NODE
765 005362  103407                         BCS     200$                ;NO MORE NODES
766 005364  020400                         CMP     R4,R0               ;PAST CUT OFF?
767 005366  103773                         BLO     110$                ;NO
768 005370  005367  000000G                DEC     NODEC               ;YES: DELETE NODE AND ADJUST NODE COUNT
769 005374  004767  000056                 JSR     PC,DELNDE
770 005400  000766                         BR      110$
771 005402  016704  000000G         200$:   MOV     FMDIDX,R4           ;SET UP FOR FLU MOD NODE SCAN
772 005406  005005                         CLR     R5
773 005410  004767  000000G         210$:   JSR     PC,GETNDE           ;GET NODE
774 005414  103405                         BCS     300$                ;NO MORE NODES
775 005416  020400                         CMP     R4,R0               ;PAST CUT OFF?
776 005420  103773                         BLO     210$                ;NO
777 005422  004767  000030                 JSR     PC,DELNDE           ;YES: DELETE NODE
778 005426  000770                         BR      210$
779 005430                         300$:   CLOSE$  #FDBQRY             ;CLOSE QRY FILE
780 005440  016767  172742  172726       MOV     QRYFS,FLUID         ;RESET FLU ID
781 005446  016767  000000G 000000G      MOV     QRYPS,NXTNDE        ;RESET NEXT AVAILABLE NODE ADDRESS
782 005454  000207                         RTS     PC
783                                ;
784                                ; SUBROUTINE TO DE-LINK A NODE
785                                ;
786 005456                         DELNDE: SAVE    R0,R1
787 005462  011401                         MOV     (R4),R1             ;R1=NEXT NODE
788 005464  016400  000002                 MOV     2(R4),R0            ;R0=PRIOR NODE
789 005470  001410                         BEQ     2$                  ;NO PRIOR NODE
790 005472  010110                         MOV     R1,(R0)             ;SET FORWARD LINK OF PRIOR NODE
791 005474  005701                 10$:    TST     R1                  ;ANY NEXT NODE?
792 005476  001402                         BEQ     1$                  ;NO
793 005500  010061  000002                 MOV     R0,2(R1)            ;SET BACKWARD LINK OF NEXT NODE
794 005504                         1$:     RESTORE R0,R1               ;FINISHED
795 005510  000207                         RTS     PC
796 005512  010163  177776         2$:     MOV     R1,-2(R3)           ;SET INDEX POINTER TO NEXT NODE
797 005516  000766                         BR      10$
798                                ;
799                                ; SUBROUTINE TO SET UP SIGNIFICANCE OF NEXT CHARACTER
800                                ;
801                                ;                   INPUT                   OUTPUT
802                                ;       R0=         SCRATCH                 SIGNIFICANCE
803                                ;       R1=         ADR OF CHAR             ADR OF NEXT CHAR
804                                ;       R2=         CHAR SIGNIF TABLE       (SAME)
805                                ;       R3=         (SAVED)                 (RESTORED)
806                                ;       R4=         ..... UNUSED .....
```

ERROR·HANDLING·ROUTINE·

```
                                ;       R5=·    ............UNUSED.............
                                ;
005520                          STEP::  SAVE·   R3
005522· 005003                          CLR·    R3
005524  112103                         MOVB·   (R1)+,R3        ;GET·NEXT·CHAR·
005526  100432·                         BMI·    ILLCHR·         ;>128. IS·ILLEGAL·
005530  006303                         ASL·    R3              ;CONVERT CHAR·TO·WORD·INDEX·
005532· 060203                         ADD·    R2,R3           ;ADD·TABLE·START·TO·INDEX·
005534  011300                         MOV·    (R3),R0         ;GET·SIGNIFICANCE·FROM·TABLE FOR·CHAR·
005536  001426                         BEQ·    ILLCHR·         ;IF·NO·SIGNIFICANCE, IT'S·ILLEGAL·
005540                                 RESTORE·R3
005542· 000207                         RTS·    PC·
                                ;
                                ;   SUBROUTINE·TO·SEND·TO·MSCHED, VIA·ENTRY·IN·SSQ, AN·ACKNOWLEDGEMENT
                                ;   RECORD.
                                ;
                                ;   ON·ENTRY,
                                ;       CMDCDE·CONTAINS·COMMAND·CODE·
                                ;       ERRCDE·CONTAINS·ERROR·CODE·(EQUAL·TO·ZERO·IF·NO·ERROR)
                                ;
                                ;   FIRST, GET·AN·EMPTY·PACKET FROM·POOL.
                                ;
005544                          SSQ::   CALL·   GETFRE·         ;ON·RETURN, R2-> PACKET·
                                ;
                                ;   BUILD·PACKET·
                                ;
005550                                  SAVE·   R2·
005552· 062702· 000002                  ADD·    #2,R2·          ;BYPASS·LINK·WORD·
005556  016722· 172554                  MOV·    CMDCDE,(R2)+    ;COMMAND CODE·
005562· 016722· 172552                  MOV·    BATNO,(R2)+     ;BATCH·NUMBER·
005566  016722· 172550                  MOV·    QRYNO,(R2)+     ;QID·
005572· 005767  172546                  TST·    ERRCDE·         ;ERROR?·
005576  001402·                         BEQ·    1$              ;BRANCH·IF·NO·
005600  016722· 172540                  MOV·    ERRCDE,(R2)+    ;ERROR·
                                ;
                                ;   QUEUE·PACKET·TO·BOTTOM·OF·SSQ·
                                ;
005604                          1$:     RESTOR· R2·             ;R2-> PACKET·
005606                                  CALL·   PUTSSQ·
005612·                                 EXIT·   SSQ·            ;RETURN·
005614  012767  000004  172522· ILLCHR::MOV·    #QE.ILC,ERRCDE  ;NOTE·ERROR·CONDITION·
005622·                                 RESTORE·R3
005624  000167  176356                  JMP·    ERROR·
                000410'                 .END·START·
```

```
ACTFMN=·****** GX·      B.FSBZ  000102·     010 EMACUT=·****** GX·      FN.MHR· 000010      011 GENQLS=·****** GX·
ASKOVF=·000145          B.FSCZ  000104      010 EMAMSZ=·****** GX·      FN.NMB· 000044      011 GENTX·=·****** GX·
AUTCLS· 000400R·    014 B.HBLK  000120      010 EMAOVF=·000166          FN.OLS· 000006      011 GETFRE=·****** GX·
BATABD=·003401 G·       B.HDOC  000114      010 EMAOSZ· 004032RG·   014 FN.ORY· 000020      011 GETNDE=·****** GX·
BATINI· 004706R·    014 B.HRLP  000126      010 EMATSZ· 004034RG·   014 FN.SF0  000030      011 HSZ·   =·****** GX·
BATNO·  000340RG·   014 B.HRLR  000122·     010 EMBCLS=·****** GX·      FN.SF1  000032·     011 IE.ITS=·****** GX·
BATOVF=·000401 G·       B.HRLW  000124      010 EMBCNT· 004060R·    014 FN.SHD· 000042·     011 ILLCHR· 005614RG·   014
BATREJ· 000544RG·   014 B.NMBR  000052·     010 EMBCUT=·****** GX·      FO.RD·=·****** GX·      INCLUD=·****** GX·
BITVAL=·000000          B.NQRY  000232·     010 EMBMSZ=·****** GX·      FO.WRT=·****** GX·      INIVDC=·****** GX·
BIT0  =·000001          B.QLSZ  000106      010 EMBOVF=·000167          F.ACTL=·000076          IOSB    000370RG·   014
BIT1  =·000002·         B.QMAP  000234      010 EMBQSZ· 004036RG·   014 F.ALOC=·000040          LNPOVF=·000146
BIT10 =·002000          B.QSPL  000316      010 EMBTSZ· 004040RG·   014 F.BBFS=·000062·         LN1     004466R·    014
BIT11 =·004000          B.QTTM  000076      010 EMCMSZ=·****** GX·      F.BDB·=·000070          LN1E    004527R·    014
BIT12 =·010000          B.QUQP  000056      010 EMCOVF=·000170          F.BGBC=·000057          LN2·    004527R·    014
BIT13 =·020000          B.SFDB  000010      010 EMCQSZ· 004030RG·   014 F.BKDN=·000026          LN2E    004567R·    014
BIT14 =·040000          B.SIZE  000772·     010 EMCTSZ· 004026RG·   014 F.BKDS=·000020          LN3     004567R·    014
BIT15 =·100000          B.SNDP  000012·     010 EMXFDB=·****** GX·      F.BKEF=·000050          LN3E    004626R·    014
BIT2· =·000004          B.SSQ·  000004      010 ERRCDE· 000344RG·   014 F.BKP1=·000051          LN5     004626R·    014
BIT3  =·000010          B.SSQF  000050      010 ERROR·  004206RG·   014 F.BKST=·000024          LN5E    004645R·    014
BIT4  =·000020          B.STAT  000044      010 ERRORF· 004156RG·   014 F.BKVB=·000064          LN6     004645R·    014
BIT5  =·000040          B.STTE  000053      010 ERRORV· 002672RG·   014 F.CHR·=·000075          LN6E    004702R·    014
BIT6  =·000100          B.UDOC  000110      010 ERROR5  004336R·    014 F.CNTG=·000034          LOGCLS=·****** GX·
BIT7  =·000200          CFALS·=·****** GX·      EXIT·   004370RG·   014 F.DFNB=·000046          LSZ·   =·****** GX·
BIT8·=·000400           CF.B0·=·000070          EXPCNT· 004056R·    014 F.DSPT=·000044          M·     =·000062
BIT9  =·001000          CF.B2·=·000067          EXPSAV· 004054R·    014 F.DVNM=·000134          MR·    =·****** GX·
BLDEFL=·****** GX·      CF.B4 =·000066          FAL·   =·****** GX·      F.EFBK=·000010         MSGOUT=·****** GX·
BLDNFL=·****** GX·      CF.B6 =·000065          FALAD·=·****** GX·       F.EFN·=·000050         MSG1    004442R·    014
BSTPTR=·****** GX·      CF.DR0=·000064          FALMSZ=·****** GX·       F.EOBB=·000032·        MSG2    004446R·    014
BS.CLS=·000002·         CF.DR1=·000063          FALOVF=·000160           F.ERR·=·000052·        MSG3    004452R·    014
BS.DBU=·000004          CHRCNT  004046RG    014 FCSERR· 004124RG·   014 F.FACC=·000043          MSG5    004456R·    014
BS.INA=·000000          CLBRED=·000400 G        FDBQRY· 000032R·    014 F.FFBY=·000014          MSG6    004462R·    014
BS.OPN=·000001          CLSBAT  001366R·    014 FD.FID· 000000      003 F.FNAM=·000110          N·     =·000002
BS.SRC=·000003          CLSEMX=·****** GX·      FD.FNB· 000006      003 F.FNB·=·000102·         NAQEX·=·****** GX·
BYTE0 =·000000          CLSIN·=·001401 G        FD.FVR· 000004      003 F.FTYP=·000116          NDBOVF=·000175
BYTE1 =·000001          CLSOLY=·001001 G        FD.LEN· 000010      003 F.FVER=·000120          NDCHR·=·****** GX·
BYTE2·=·000002·         CMDCDE  000336RG    014 FD.RWM=·****** GX·      F.HIBK=·000004          NDFLID=·****** GX·
BYTE3 =·000003          CORUNK=·003001 G        FLG9    004062R·    014 F.LUN·=·000042·         NDFMEN=·****** GX·
BYTE4 =·000004          CORUPT=·002401 G        FLIXSZ=·****** GX·       F.MBCT=·000054         NDFSAB=·****** GX·
BYTE5 =·000005          CPIXSZ=·****** GX·      FLUCLS=·****** GX·       F.MBCi=·000055         NDSIZ·=·****** GX·
BYTE6 =·000006          CWPIDX=·****** GX·      FLUCUT=·****** GX·       F.MBFG=·000056         NDTYP·=·****** GX·
BYTE7 =·000007          DBSLEN=·000116          FLUID·  000374RG·   014 F.NRBD=·000024          NFALE·=·****** GX·
BYTE8·=·000010          DECPT·=·****** GX·      FLUIDX=·****** GX·       F.NREC=·000030         NMRNGF=·****** GX·
BYTE9 =·000011          DELNDE  005456R·    014 FLUTYP=·****** GX·       F.OVBS=·000030         NODEA·=·****** GX·
BYTVAL=·000012·         DH.BF0  000002·     005 FMDIDX=·****** GX·       F.RACC=·000016         NODEB·=·****** GX·
B$NOT·=·****** GX·      DH.BF1  000004      005 FNPOSZ=·****** GX·       F.RATT=·000001         NODEC·=·****** GX·
B.BSTA· 000054      010 DH.CTL  000000      005 FN.DBR· 000026      011 F.RCNM=·000034          NOTFLG=·****** GX·
B.CNTX· 000046      010 DH.DMC  000010      005 FN.DBS· 000022·     011 F.RCTL=·000017          NQLBE·=·****** GX·
B.CQUQ· 000060      010 DH.FLG  000006      005 FN.DHR· 000040      011 F.RSIZ=·000002·         NULFMN=·****** GX·
B.FEMA· 000132·     010 DIRERR  004064RG    014 FN.EMA· 000012·     011 F.RTYP=·000000          NXTNDE=·****** GX·
B.FEMB· 000142·     010 DNUM·  =·****** GX·      FN.EMB· 000014     011 F.SEQN=·000100          N.BFAC=·000004
B.FEMC· 000152·     010 DN.DCK  000000      013 FN.EMC· 000016      011 F.SPDV=·000072·         N.BHGH=·000006
B.FFSA· 000202·     010 DN.NTP  000004      013 FN.FSA· 000000      011 F.SPUN=·000074          N.BTCH=·000004
B.FFSB· 000212·     010 DN.NXT  000006      013 FN.FSB· 000002·     011 F.STBK=·000036          N.BUFB=·004000
B.FFSC· 000222·     010 DN.ROT  000002·     013 FN.FSC· 000004      011 F.UNIT=·000136          N.BUFW=·002000
B.FMHR· 000172·     010 DN.SIZ  000010      013 FN.LGQ· 000034      011 F.URBD=·000020          N.DID·=·000024
B.FQLS· 000162·     010 ELSCLS=·****** GX·      FN.LGU· 000036      011 F.VBN·=·000064          N.DVNM=·000032
B.FSAZ· 000100      010 EMACLS=·****** GX·      FN.MFQ· 000034      011 F.VBSZ=·000060          N.FID·=·000000
```

```
N.FNAM=·000006          QE.NRB=·000024          SDLB··=·****** GX·      ST.CSZ··000030      006 VLDC···=·****** GX·
N.FOS·=·000764          QE.NRI=·000023          SDLBAD=·****** GX·      ST.HRL··000010      006 WN.NTP··000004      012·
N.FTYP=·000014          QE.NTK=·000027          SDLBE·=·****** GX·      ST.LEN··000044      006 WN.NXT··000006      012·
N.FVER=·000016          QE.PX1=·000017          SEGOP·=·****** GX·      ST.QRY··000002·     006 WN.ROT··000002      012·
N.NEXT=·000022·         QE.PX2=·000020          SLBMSZ=·****** GX·      ST.QSZ··000034      006 WN.SIZ··000010      012·
N.PKSZ=·000020          QE.PX3=·000021          SLBOVF=·000164          ST.SCH··000040      006 WN.SRC··000000      012·
N.PKTS=·000043          QE.PX4=·000022·         SR.ARE··000114      002·ST.UHL··000004      006 WN.TYP··000001      012·
N.QURY=·000031          QE.RDI=·000144          SR.ARS··000106      002·ST.XLT··000014      006 WORD0···=·000000
N.STAT=·000020          QE.UBP=·000014          SR.DAY··000010      002·SU.DBU=·000004          WORD1···=·000002
N.SUNT=·000002·         QLB···=·****** GX·      SR.DLT··000014      002·SU.DON=·000006          WORD2···=·000004
N.UNIT=·000034          QLBAD·=·****** GX·      SR.ECB··000047      002·SU.IDL=·000000          WORD3···=·000006
OPFAIL··000000R·    014 QLBMSZ=·****** GX·      SR.ECH··000046      002·SU.LOD=·000001          WORD4···=·000010
OPFSIZ=·000032·         QLBOVF=·000163          SR.ECL··000050      002·SU.SRC=·000002·         WORD5···=·000012
OVFCLS··004052RG·   014 QLS···=·****** GX·      SR.FIB··000012·     002·SU.SRR=·000005          WORD6···=·000014
PARSER=·****** GX·      QLSBAT=·****** GX·      SR.GRE··000100      002·SU.XPD=·000003          WORD7···=·000016
PAR$$$=·000027          QLSBUF=·****** GX·      SR.GRS··000072·     002·SZCALI··002726RG·   014 WORD8···=·000020
PAR1····004702RG·   014 QLSFDB··000172R·   014 SR.LEN··000122·     002·SZNR····003660R·   014 WORD9···=·000022
PAR2····004704RG·   014 QLSIZ·=·****** GX·      SR.LIN··000066          WRDVAL=·000024
PD····=·****** GX·      QNFLS·=·****** GX·      SR.LIP··000062·         XBATCH=·000013
POLEND=·****** GX·      QNFLU·=·****** GX·      SR.MDN··000006          XDBLOA=·000004
POLOVF=·000157          QRYBAD=·002001 G        SR.NDC··000042·         XDBPRO=·000012
POOL··=·****** GX·      QRYBUF=·****** GX·      SR.NDS··000036          XDMCIN=·000006
PRXFLG=·****** GX·      QRYDEL··005164RG    014 SR.NIN··000030          XFDSMR=·000007
PUTSSQ=·****** GX·      QRYEFN=·000003          SR.NIP··000022·         XGTSRE=·000014
QBFOVF=·000150          QRYEND=·****** GX·      SR.SDB··000032·         XHITSK=·000011
QCL···=·****** GX·      QRYFAS=·****** GX·      SR.SRC··000002·         XHLMER=·000002
QEX···=·****** GX·      QRYFS···000406R·    014 SR.SUN··000000          XHOTSK=·000010
QEXAD·=·****** GX·      QRYLUN=·000003          SR.TWS··000056          XMSCHE=·000000
QEXFPV=·****** GX·      QRYMAX=·****** GX·      SR.WSL··000052·     002·TOOBIG··000562R·    014 XQTS···=·000003
QEXFSV=·****** GX·      QRYMS·=·****** GX·      SR.YR···000004      002·TRMCUT=·****** GX·      XQT0···=·000001
QEXFWV=·****** GX·      QRYMSZ=·****** GX·      SR.1IN··000024      002·TSKNAM··000332RG·   014 XSULOA=·000005
QEXMSZ=·****** GX·      QRYNO···000342RG    014 SR.1IP··000016          TSKOVF=·000147          XTBMSZ=·****** GX·
QEXOVF=·000165          QRYOK·=·000001 G        SSQ·····005544RG·   014 TTABLE=·****** GX·      XTBOVF=·000162
QE.DW1=·000001          QRYOVF=·000156          SS.FID··000002·     004 TTBMSZ=·****** GX·      ZERO····000376RG·   014
QE.DW2=·000003          QRYPS·=·****** GX·      SS.FNB··000010          TTBOVF=·000161          ZVLDC·=·****** GX·
QE.FTB=·000026          QRYSIZ··000402R·    014 SS.FVR··000006      004 VDCCNT··004050RG·   014 $DSW···=·****** GX·
QE.IHS=·000005          QTFORM=·****** GX·      SS.LEN··000012·         VEC1MX=·****** GX·      .CLOSE=·****** G·
QE.ILC=·000004          QTSTAT=·****** GX·      SS.STT··000000      004 VEC2MX=·****** GX·      .FINIT=·****** G·
QE.IND=·000015          Q.FDSC··000004      007 START···000410R·    014 VEC3MX=·****** GX·      .FSRCB=·****** G·
QE.ISO=·000016          Q.NQBK··000000      007 START1··000424R·    014 VIBCUT=·****** GX·      .MOLUN··000372RG·   014
QE.MFL=·000013          Q.NUHL··000002·     007 START2··000566R·    014 VIBQSZ··004042RG·   014 .OPFNB=·****** G·
QE.MFM=·000010          Q.SIZE··000014      007 START3··000420RG·   014 VIBTSZ··004044RG·   014 .READ··=·****** G·
QE.MNT=·000012·         REMQRY··004270R·    014 STEP····005520RG·   014 VI1OVF=·000173          .WAIT··=·****** G·
QE.MOP=·000007          RESCHK··001674R·    014 STX···=·****** GX·      VI2MSZ=·****** GX·      .WRITE=·****** G·
QE.MSD=·000011          RNGCDE=·****** GX·      ST.ASZ··000020      006 VI2OVF=·000172·         ...PC1=·000172R·    014
QE.MTS=·000002·         SAVSTK··000404R·    014 ST.BSZ··000024      006 VI3MSZ=·****** GX·      ...PC2=·000332R·    014
QE.NLQ=·000006          SDFLG·=·****** GX·      ST.BTC··000000      006 VI3OVF=·000171          ...TPC=·000140
QE.NOS=·000025

. ABS:  000000    000
        000000    001
SRCOFF· 000122·   002·
FDSCOF· 000010    003
SUSOFF· 000012·   004
DHROFF· 000012·   005
STTOFF· 000044    006
QSPLOF· 000014    007
```

```
QT0.....MACRO.M1110  27-MAR-80 12:54  PAGE 15-7
SYMBOL.TABLE.

BSTOFF.  000772.     010
FNOFFS.  000044      011
WNODOF.  000010      012.
DNODOF.  000010      013
QT0      005630      014
$$FSR1   000000      015
ERRORS.DETECTED:  0

VIRTUAL.MEMORY.USED:  9793 WORDS.  (.39 PAGES)
DYNAMIC.MEMORY:  11252.WORDS.  (.43 PAGES)
ELAPSED.TIME:  00:01:15
QT0,QT0/-SP/NL:ME:BEX=[20,1]P,M,E,MDQT0,QT0
```

```
 1                              ; QTSIZE.MAC    "QUERY TRANSLATORS BUFFER SIZE CONFIGUATION FILE"
 2                              ;       THIS FILE CONTROLS THE SIZE OF ALL BUFFERS THAT CAN VARY
 3                              ;       IN SIZE DUE TO THE AMOUNT OF QUERIES OR OTHER SUCH PARAMETERS
 4                              ;       THAT WOULD BE CHARACTERISTIC OF A SITE.  THESE BUFFERS ARE
 5                              ;       IN QT0, QT1, QT2, OR QT3.
 6                              ;
 7                              ;---QT3---BUFFERS-------------------------------------------------------------
 8                              ;
 9       000125                 VEC3MX==85.                                     ;MAXIMUM # CWP'S IN BATCH
10       000025                 AVELGT==3*2*7/2                                 ;AVERAGE CWP SIZE IN TDCT (BYTES)
11       000400                 VI3MSZ==VEC3MX+2/256.+1*256.                    ;SIZE OF VI (NEAREST BLOCK IN BYTES)
12       001000                 EMCMSZ==VEC3MX/51.+1*256.                       ;SIZE OF EMC (NEAREST BLOCK)
13       004000                 TDCMSZ==VEC3MX*AVELGT+8./N.BUFW+1*N.BUFW        ;SIZE OF TDCT
14       000010                 TDCBLK==TDCMSZ/256.                             ;VIRTUAL BLOCK SIZE OF TDCT
15       000524                 NP3MSZ==VEC3MX*4                                ;SIZE OF NODE POOL
16       000125                 NP3OSZ==VEC3MX                                  ;SIZE OF NODE POOL OVERFLOW AREA
17                              ;
18                              ;-----QT2----BUFFERS---------------------------------------------------------
19                              ;
20       000375                 VEC2MX==253.                                    ;MAXIMUM # VECTORS
21       000400                 VI2MSZ==VEC2MX+2/256.+1*256.                    ;SIZE OF VI
22       004400                 EMBMSZ==9.*256.                                 ;SIZE OF EMB
23       000003                 TDBBLK==3                                       ;#BLOCKS (N.BUFW) IN TDCT BUFFER
24       000074                 TDBOSZ==3*20.                                   ;SIZE OF TDCT BUFFER OVERFLOW
25       007775                 TDBMAD==4093.                                   ;MAXIMUM ADDRESS (48 BITS) IN TDCT
26       000020                 NIXCNT==16.                                     ;# INDEX POINTERS FOR NORMAL NODES
27       177760                 NNMASK==177760                                  ;MASK FOR NORMAL INDEX
28       000010                 EIXCNT==8.                                      ;# INDEX PTRS FOR ELSE NODES
29       177761                 ELSMSK==177761                                  ;MASK FOR ELSE INDEX
30       000004                 ENCNT==4                                        ;# ENTRIES TO USE FOR HASH
31       036000                 NP2MSZ==256.*60.                                ;SIZE OF NODE POOL
32       000153                 NP2OSZ==7+100.                                  ;SIZE OF NODE POOL OVERFLOW AREAR
33                              ;
34                              ;------QT1-----BUFFERS-------------------------------------------------------
35                              ;
36       000375                 VEC1MX==253.                                    ;MAX. # TERMS IN FSA-A
37       000400                 VI1MSZ==VEC2MX+2/256.+1*256.                    ;SIZE OF VI
38       000004                 TDABLK==4                                       ;#BLOCKS (N.BUFW) IN TDCT BUFFER
39       007764                 TDABMX==<TDABLK*<N.BUFW-4>>+4
40       016000                 EMAMSZ==7.*4*256.                               ;SIZE OF EMA
41       007760                 TDAMAD==340.*12.                                ;MAX. ADDRESS IN TDCT
42       010000                 NP1MSZ==4096.                                   ;SIZE OF NORMAL NODE POOL
43       000144                 NPECNT==100.                                    ;#NODES IN ELSE POOL
44       000043                 NPEVSZ==35.                                     ;#VECTORS IN ELSE NODES
45                              ;
46                              ;-----FLU---MODIFIER----BUFFERS-----------------------------------------------
47                              ;
48       002260                 FMNPSZ==1200.                                   ;SIZE OF NORMAL FLU MOD NODE POOL
49       000400                 FMEPSZ==256.                                    ;SIZE OF ELSE FLU MOD NODE POOL
50                              ;
51                              ;------QT0------BUFFERS-------------------------------------------------------
52                              ;
53       002000                 QRYSZ==N.BUFW                                   ;SIZE OF QUERY BUFFER
54       000040                 FLIXSZ==32.                                     ;# INDEX PTRS FOR TERMS
55       177740                 FLIXMK==177740                                  ;MASK FOR TERM INDEX
56       000010                 CPIXSZ==8.                                      ;# INDEX PTRS FOR CWP'S
57       177770                 CPIXMK==177770                                  ;MASK FOR CWP INDEX
```

```
000001                  FMIXSZ==1                               ;# INDEX PTRS FOR FLU-MOD'S
005734                  FNPSZ==3036.                            ;FLU NODE POOL SIZE
000400                  FNPOSZ==256.                            ;SIZE OF POOL OVERFLOW AREA
000400                  TSTKMX==256.                            ;MAX. # TOKENS TO BE PUSHED
000400                  ASTKMX==256.                            ;MAX. # ARGUMENTS TO BE PUSHED
000024                  PSTKMX==20.                             ;MAX. # PROX NODES IN A CHAIN
000100                  HSTKMX==64.                             ;MAX. NESTING OF QUERY
001000                  QNDCNT==512.                            ;# NODES IN LOGIC POOL
000400                  VI0MSZ==VI1MSZ                          ;ASSUME MAX VI IS IN QT1
                        .IIF LT,VI0MSZ-VI2MSZ,VI0MSZ==VI2MSZ    ; IF NOT, RESET QT0'S TO QT2'S VI SIZE
016000                  EMXMSZ==EMAMSZ                          ;ASSUME MAX EMX IS IT QT1
                        .IIF LT,EMXMSZ-EMBMSZ,EMXMSZ==EMBMSZ    ; IF NOT, RESET TO QT2'S
                        ;
                        ;-----ERROR AND CLOSE CODES---------------------------------------
                        ;
000040                  TRMCUT==32.     ;TERM CUT-OFF
000016                  FLUCUT==14.     ;FLU CUT-OFF
001700                  EMACUT==TRMCUT*30.      ;EMA CUT-OFF
000036                  VIBCUT==30.             ;VIB CUT-OFF
001130                  EMBCUT==VIBCUT*20.      ;EMB CUT-OFF
                        ;
000001                  FLUCLS==1       ;CLOSED DUE TO FLU COUNT
000002                  LOGCLS==2       ;CLOSED DUE TO LOGIC COUNT (QLB,SDLB,QEX)
000003                  EMACLS==3       ;CLOSED DUE TO EMA SIZE
000004                  EMBCLS==4       ;CLOSED DUE TO EMB SIZE
000005                  TDBCLS==5       ;CLOSED DUE TO TDCTB SIZE
000006                  NDBCLS==6       ;CLOSED DUE TO QT2 NODE POOL SIZE
000007                  ELSCLS==7       ;CLOSED DUE TO OTHER CONDITIONS (FLU POOL,ETC)
                        ;
```

```
                                ;
                                ; MAXIMUM QLS SUB-BUFFER SIZES
                                ;
000031                          QRYMSZ==N.QURY
000500                          FALMSZ==160.*2
003270                          TTBMSZ==430.*4
002010                          XTBMSZ==516.*2
000516                          QLBMSZ==334.
000354                          SLBMSZ==236.
000600                          QEXMSZ==192.*2
                                ;
                                ; QUERY RESOLVER BIT DEFINITIONS
                                ;
000200                          B$SUC==BIT7     ;SUCCESS BIT
000100                          B$NOT==BIT6     ;QLB NOT ENTRY
000040                          B$MUL==BIT5     ;QLB MULTI-NOT ENTRY
000020                          B$FMN==BIT4     ;QLB FIRST MULTI-NOT ENTRY
                                ;
020000                          B$FST==BIT13    ;QEX FIRST PROX ENTRY
100000                          B$PUFH==BIT15   ;QEX FORWARD PROX UNIT - HIGH ORDER BIT
040000                          B$PUFL==BIT14   ;QEX FORWARD PROX UNIT - LOW ORDER BIT
010000                          B$PUBH==BIT12   ;QEX BACKWARD PROX UNIT - HIGH ORDER BIT
004000                          B$PUBL==BIT11   ;QEX BACKWARD PROX UNIT - LOW ORDER BIT
                                ;
                                ; QLS BUFFER OFFSETS (CONVERTED TO ADDRESSES)
                                ;
000000                          QLSXX1=0                ;OFFSET OF COMM HEADER
000004                          QLSXI7=QLSXX1+4         ;BATCH NUMBER
000006                          QLSXX2=QLSXI7+2         ;BYTE SIZE OF QLB (USED)
000010                          QLSXX3=QLSXX2+2         ;BYTE SIZE OF SDLB (USED)
000012                          QLSXX4=QLSXX3+2         ;BYTE SIZE OF QEX (USED)
000014                          QLSXX5=QLSXX4+2         ;MAXIMUM QUERY ID
000016                          QLSXX6=QLSXX5+2         ;BYTE SIZE OF FAL (USED)
000020                          QLSXX7=QLSXX6+2         ;BYTE SIZE OF TTABLE (USED)
000022                          QLSXX8=QLSXX7+2         ;BYTE SIZE OF XTABLE (USED)
000024                          QLSXI0=QLSXX8+2         ;START OF QCL SUB-BUFFER
000106                          QLSXI1=QLSXI0+<QRYMSZ*2>;START OF FAL SUB-BUFFER
000606                          QLSXI2=QLSXI1+FALMSZ    ;START OF TTABLE SUB-BUFFER
004076                          QLSXI3=QLSXI2+TTBMSZ    ;START OF XTABLE SUB-BUFFER
006106                          QLSXI4=QLSXI3+XTBMSZ    ;START OF QLB SUB-BUFFER
006624                          QLSXI5=QLSXI4+QLBMSZ    ;START OF SDLB SUB-BUFFER
007200                          QLSXI6=QLSXI5+SLBMSZ    ;START OF QEX SUB-BUFFER
010000                          QLSIZ==QLSXI6+QEXMSZ    ;SIZE OF QLS IN BYTES
```

```
                                        ;
                                        ; SYMBOLIC DEFINITIONS
                                        ;
          000002                        STX==2
          000003                        ETX==3
          000001                        NDFSAA==1       ;TYPE FSA-A
          000002                        NDFSAB==2       ;TYPE FSA-B
                                        ;
                                        ; FLU NODE OFFSET DEFINITIONS
                                        ;
 000000                                         .PSECT  FLUNDE,ABS
 000000                                 NDFWD:: .BLKW   1       ;FORWARD POINTER
 000002                                 NDBKW:: .BLKW   1       ;BAKWARD POINTER
 000004                                 NDSIZ:: .BLKB   1       ;NODE CONTENT SIZE
 000005                                 NDFMS::                 ;START OF FLU MODIFIERS
 000005                                 NDTYP:: .BLKB   1       ;NODE TYPE
 000006                                 NDFLID::.BLKW   1       ;FLU ID
 000010                                 NDTRM::                 ;START OF CWP TERMS
 000010                                 NDFMN:: .BLKW   1       ;FLU MOD NODE POINTER
 000012                                 NDFMEN::.BLKW   1       ;FLU MOD EXTENSION NODE POINTER
 000014                                 NDCHR::                 ;START OF CHAR'S
                                        ;
                                        ;
                                        ; LOGIC NODE OFFSET DEFINITIONS
                                        ;
 000000                                         .PSECT  QNBASE,ABS
 000000                                 QNOPCD::.BLKB   1       ;OPERATOR CODE FIELD
 000001                                 QNWIN:: .BLKB   1       ;PROXIMITY WINDOW FIELD
 000002                                 QNARG1::.BLKW   1       ;FIRST (LEFT) ARGUMENT OF NODE
 000004                                 QNARG2::.BLKW   1       ;SECORD (RIGHT) ARGUMENT OF NODE
 000006                                 QNSTE::                 ;USE COUNT FIELD (QTFORM - LOW BYTE)
 000006                                 QNATTR::.BLKW   1       ;ATTRIBUTE FIELD OF NODE
          000010                        QNDSIZ  ==      .       ;BYTE SIZE OF QUERY LOGIC NODE
                                        ;
                                        ; FLU NODE VARIBLES AND BUFFERS
                                        ;
                                        ; .
 000000                                         .PSECT  AABUFS
 000000                                 FLUIDX::.BLKW   FLIXSZ  ;FLU NODE INDEXES
 000100                                 CWPIDX::.BLKW   CPIXSZ  ;CWP NODE INDEXES
 000120                                 FMDIDX::.BLKW   FMIXSZ  ;FLU MOD NODE INDEXES
                                        ;
 000122  000154'                        NXTNDE::.WORD   POOL    ;NEXT NODE ADR
 000124  000000                         FLUNDE::.WORD   0       ;CURRENT FLU NODE
 000126  000000                         CWPNDE::.WORD   0       ;CURRENT CWP NODE
 000130  000146'                        ACTFMN::.WORD   NULFMN  ;ACTIVE FLU MODIFIER NODE
 000132  000000                         QRYPS:: .WORD   0       ;NODE ADDRESS AT START OF QUERY
 000134  000000                         QRYMS:: .WORD   0       ;MAX QUERY ID AT START OF QUERY
 000136  000000                         QRYFAS::.WORD   0       ;FAL LOCATION AT START OF QUERY
                                        ;
 000140  000000                         NODEA:: .WORD   0       ;FSA-A NODE COUNT
 000142  000000                         NODEB:: .WORD   0       ;FSA-B NODE COUNT
 000144  000000                         NODEC:: .WORD   0       ;FSA-C NODE COUNT
                                        ;
 000146  000000  000000  000000         NULFMN::.WORD   0,0,0   ;NULL FLU MODIFIER NODE
 000154                                 POOL::  .BLKW   FNPSZ   ;NODE POOL
 014044                                 POLEND::.BLKW   FNPOSZ  ;END OF POOL AND OVERFLOW
```

```
                                        ;
                                        ; QLS BUFFER AND EMX BUFFER.
                                        ;       NOTE: EMX BUFFER OVERLAYS OTHER BUFFERS.
                                        ;
015044      062     103                 QLSBUF::.ASCII  /2C/    ;COMM HEADER FOR QLS FILE.
015046  000000                          EMXLGT::.WORD   0       ;BLOCK LENGTH OF EMX.
015050  000000                          VILGT:: .WORD   0       ;NO. OF ENTRIES IN VI.
015052                                  VI::    .BLKW   VI0MSZ-2.               ;VI BLOCK(S)
016046                                  EMX::   .BLKW   <QLSIZ/2>-VI0MSZ-1      ;EMX CONTINUES,QLS ENDS.
                                        ;
                                        ; STACKS.
                                        ;
                                        ;------------FOLLOWING STACKS ARE OVERLAYED, PICK LARGER SIZE.
          000200                        STK1==TSTKMX/2.                         ;ASSUME TOKEN STACK IS LARGER.
                                        .IIF LT,STK1-PSTKMX,STK1==PSTKMX; IF NOT, RESET TO PROX STACK SIZE.
          000400                        STK2==ASTKMX.                           ;ASSUME ARG. STACK IS LARGER
                                        .IIF LT,STK2-HSTKMX,STK2==HSTKMX; IF NOT, RESET.
                                        ;
025044      000                                 .BYTE   0       ;FORCE ODD START
025045      000G                        TOKSTK::.BYTE   TOKETX  ;TOKEN STACK INITIALIZED TO ETX TOKEN
025046                                  SPXSTK::.BLKW   STK1    ;QLS DOUBLE END PROX STACK & TOKEN STACK.
025446                                  FPXSTK::.BLKW   1
025450                                  BOTSTK::.BLKW   STK2    ;QLS HIEARCHY STACK & PARSER ARG STACK.
026450                                  TOPSTK::
026450                                  ARGSTK::.BLKW   1       ;BASE OF ARGUMENT STACK.
                                        ;
                                        ;
                                        ; QUERY LOGIC NODE POOL BUFFER
                                        ;
026452  000000  000000  000000          SNODE:: .WORD   0,0,0,0 ;SPECIAL CONSTANT ARGUMENT NODE.
026462                                  QNPOOL::.BLKB   QNDCNT*QNDSIZ   ;QUERY LOGIC NODE POOL.
036462                                  QNDLST::                ;LAST NODE FOR PARSER.
                                        ;
                                        ; QUERY BUFFER.
                                        ;
036462                                  QRYBUF::.BLKW   QRYSZ   ;QUERY BUFFER.
042462      002     003                 QRYEND::.BYTE   STX,ETX ;SAFETY (IF QUERY DOESN'T HAVE THEM)
                                        ;
042464                                  QNTLST::                        ;QTFORM OVER WRITES QRYBUF WITH LOGIC NODES.
          003571                        EMXRND=EMXMSZ-<<.-EMX>/2>
                                        .IIF LT,EMXRND,EMXRND=0
042464                                          .BLKW   EMXRND.
```

```
                        ;
        015044'         QLSHD==QLSXX1+QLSBUF            ;CONVERT TO ADDRESS
        015050'         QLSBAT==QLSX17+QLSBUF
        015052'         QLBAD==QLSXX2+QLSBUF
        015054'         SDLBAD==QLSXX3+QLSBUF
        015056'         QEXAD==QLSXX4+QLSBUF
        015060'         QRYMAX==QLSXX5+QLSBUF
        015062'         FALAD==QLSXX6+QLSBUF
        015064'         TTSIZ==QLSXX7+QLSBUF
        015066'         XTSIZ==QLSXX8+QLSBUF
        015070'         QCL==QLSXi0+QLSBUF
        015152'         FAL==QLSXi1+QLSBUF
        015652'         TTABLE==QLSXi2+QLSBUF
        02i142'         XTABLE==QLSXi3+QLSBUF
        023152'         QLB==QLSXi4+QLSBUF
        023670'         SDLB==QLSXi5+QLSBUF
        024244'         QEX==QLSXi6+QLSBUF
                        ;
        015044'         QLS==QLSBUF
```

```
.MAIN.  MACRO M1110  27-MAR-80 12:53  PAGE 1

       1        000001                          .END
```

```
ACTFMN  000130RG    016 B.FQLS  000162      010 EMXMSZ= 016000 G        NDSIZ   000004 G    014 QLSX16= 007200
ARGSTK  026450RG    016 B.FSAZ  000100      010 EMXRND= 003571          NDTRM   000010 G    014 QLSX17= 000004
ASTKMX= 000400 G        B.FSBZ  000102      010 ENCNT = 000004 G        NDTYP   000005 G    014 QNARG1  000002 G    015
AVELGT= 000025 G        B.FSCZ  000104      010 ETX   = 000003 G        NIXCNT= 000020 G        QNARG2  000004 G    015
BITVAL= 000000          B.HBLK  000120      010 FAL   = 015152RG    016 NNMASK= 177760 G    016 QNDCNT= 001000 G
BIT0  = 000001          B.HDOC  000114      010 FALAD = 015062RG    016 NODEA   000140RG        NODEB   000142RG    016 QNDLST  036462RG    016
BIT1  = 000002          B.HRLP  000126      010 FALMSZ= 000500 G    003 NODEC   000144RG    016 QNDSIZ= 000010 G    015
BIT10 = 002000          B.HRLR  000122      010 FD.FID  000000      003 NPECNT= 000144 G        QNOPCD  000000 G    015
BIT11 = 004000          B.HRLW  000124      010 FD.FNB  000006      003 NPEVSZ= 000043 G        QNPOOL  026462RG    016
BIT12 = 010000          B.NMBR  000052      010 FD.FVR  000004      003 NP1MSZ= 010000 G        QNSTE   000006 G    015
BIT13 = 020000          B.NQRY  000232      010 FD.LEN  000010          NP2MSZ= 036000 G        QNTLST  042464RG    016
BIT14 = 040000          B.QLSZ  000106      010 FLIXMK= 177740 G        NP2OSZ= 000153 G        QNWIN   000001 G    015
BIT15 = 100000          B.QMAP  000234      010 FLIXSZ= 000040 G        NP3MSZ= 000524 G        QRYBUF  036462RG    016
BIT2  = 000004          B.QSPL  000316      010 FLUCLS= 000001 G        NP3OSZ= 000125 G        QRYEND  042462RG    016
BIT3  = 000010          B.QTTM  000076      010 FLUCUT= 000016 G    016 NULFMN  000146RG    016 QRYFAS  000136RG    016
BIT4  = 000020          B.QUQP  000056      010 FLUIDX  000000RG    016 NXTNDE  000122RG    016 QRYMAX= 015860RG    016
BIT5  = 000040          B.SFDB  000010      010 FLUNDE  000124RG    016 N.BFAC= 000004          QRYMS   000134RG    016
BIT6  = 000100          B.SIZE  000772      010 FMDIDX  000120RG        N.BHGH= 000006          QRYMSZ= 000031 G
BIT7  = 000200          B.SNDP  000012      010 FMEPSZ= 000400 G        N.BTCH= 000004          QRYPS   000132RG    016
BIT8  = 000400          B.SSQ   000004      010 FMIXSZ= 000001 G        N.BUFB= 004000          QRYSZ = 002000 G
BIT9  = 001000          B.SSQF  000050      010 FMNPSZ= 002260 G        N.BUFW= 002000          Q.FDSC  000004      007
BOTSTK  025450RG    016 B.STAT  000044      010 FNPOSZ= 000400 G        N.FOS = 000764          Q.NQBK  000000      007
BS.CLS= 000002          B.STTE  000053      010 FNPSZ = 005734 G    011 N.PKSZ= 000020          Q.NUHL  000002      007
BS.DBU= 000004          B.UDOC  000110      010 FN.DBR  000026      011 N.PKTS= 000043          Q.SIZE  000014      007
BS.INA= 000000          CF.B0 = 000070          FN.DBS  000022          N.QURY= 000031          SDLB   = 023670RG    016
BS.OPN= 000001          CF.B2 = 000067          FN.DHR  000040      011 N.SUNT= 000002          SDLBAD= 015054RG    016
BS.SRC= 000003          CF.B4 = 000066          FN.EMA  000012          NULL    000000
BYTE0 = 000000          CF.B6 = 000065          FN.EMB  000014      011 POLEND  014044RG    016 SLBMSZ= 000354 G
BYTE1 = 000001          CF.DR0= 000064          FN.EMC  000016      011 POOL    000154RG    016 SNODE   026452RG    016
BYTE2 = 000002          CF.DR1= 000063          FN.FSA  000000      011 PSTKMX= 000024 G        SPXSTK  025046RG    016
BYTE3 = 000003          CPIXMK= 177770 G        FN.FSB  000002      011 QCL   = 015070RG    016 SR.ARE  000114      002
BYTE4 = 000004          CPIXSZ= 000010 G        FN.FSC  000004      011 QEX   = 024244RG    016 SR.ARS  000106      002
BYTE5 = 000005          CWPIDX  000100RG    016 FN.LGQ  000034      011 QEXAD = 015056RG    016 SR.DAY  000010      002
BYTE6 = 000006          CWPNDE  000126RG    016 FN.LGU  000036      011 QEXMSZ= 000600 G        SR.DLT  000014      002
BYTE7 = 000007          DBSLEN= 000116          FN.MFO  000024      011 QE.RO1= 000144          SR.ECB  000047      002
BYTE8 = 000010          DH.BF0  000002      005 FN.MHR  000010      011 QLB   = 023152RG    016 SR.ECH  000046      002
BYTE9 = 000011          DH.BF1  000004      005 FN.NMB  000044      011 QLBAD = 015052RG    016 SR.ECL  000050      002
BYTVAL= 000012          DH.CTL  000000      005 FN.QLS  000006      011 QLBMSZ= 000516 G        SR.FIB  000012      002
B$FMN = 000020 G        DH.DMC  000010      005 FN.QRY  000020      011 QLS   = 015044RG    016 SR.GRE  000100      002
B$FST = 020000 G        DH.FLG  000006      005 FN.SF0  000030      011 QLSBAT= 015050RG    016 SR.GRS  000072      002
B$MUL = 000040 G        DN.DCK  000000      013 FN.SF1  000032      011 QLSBUF  015044RG    016 SR.LEN  000122      002
B$NOT = 000100 G        DN.NTP  000004      013 FN.SHD  000042      011 QLSHD = 015044RG    016 SR.LIN  000066      002
B$PUBH= 010000 G        DN.NXT  000006      013 FPXSTK  025446RG    016 QLSIZ = 010000 G        SR.LIP  000062      002
B$PUBL= 004000 G        DN.ROT  000002      013 HSTKMX= 000100 G        QLSXX1= 000000          SR.MON  000006      002
B$PUFH= 100000 G        DN.SIZ  000010      013 LOGCLS= 000002 G        QLSXX2= 000006          SR.NDC  000042      002
B$PUFL= 040000 G        EIXCNT= 000010 G        M     = 000062          QLSXX3= 000010          SR.NDS  000036      002
B$SUC = 000200 G        ELSCLS= 000007 G        N     = 000002          QLSXX4= 000012          SR.NIN  000030      002
B.BSTA  000054      010 ELSMSK= 177761 G        NDBCLS= 000006 G        QLSXX5= 000014          SR.NIP  000022      002
B.CNTX  000046      010 EMACLS= 000003 G        NDBKW   000002 G    014 QLSXX6= 000016          SR.SDB  000032      002
B.COUQ  000060      010 EMACUT= 001700 G        NDCHR   000014 G    014 QLSXX7= 000020          SR.SRC  000002      002
B.FEMA  000132      010 EMAMSZ= 016000 G        NDFLID  000006 G    014 QLSXX8= 000022          SR.SUN  000000      002
B.FEMB  000142      010 EMBCLS= 000004 G        NDFMEN  000012 G    014 QLSXi0= 000024          SR.TWS  000056      002
B.FEMC  000152      010 EMBCUT= 001130 G        NDFMN   000010 G    014 QLSXi1= 000106          SR.WSL  000052      002
B.FFSA  000202      010 EMBMSZ= 004400 G        NDFMS   000005 G    014 QLSXi2= 000606          SR.YR   000004      002
B.FFSB  000212      010 EMCMSZ= 001000 G        NDFSAA= 000001 G        QLSXi3= 004076          SR.1IN  000024      002
B.FFSC  000222      010 EMX     016046RG        NDFSAB= 000002 G        QLSXi4= 006106          SR.1IP  000016      002
B.FMHR  000172      010 EMXLGT  015046RG    016 NDFWD   000000 G    014 QLSXi5= 006624          SS.FID  000002      004
```

SYMBOL TABLE

```
SS.FNB  000010      004 SU.DBU=  000004          TOKETX=  ****** GX      VI3MSZ=  000400 G       WRDVAL=  000024
SS.FVR  000006      004 SU.DON=  000006          TOKSTK   025045RG   016 WN.NTP   000004      012 XBATCH=  000013
SS.LEN  000012      004 SU.IDL=  000000          TOPSTK   026450RG   016 WN.NXT   000006      012 XDBLOA=  000004
SS.STT  000000      004 SU.LOD=  000001          TRMCUT=  000040 G       WN.ROT   000002      012 XDBPRO=  000012
STK1   =  000200 G      SU.SRC=  000002          TSTKMX=  000400 G       WN.SIZ   000010      012 XDMCIN=  000006
STK2   =  000400 G      SU.SRR=  000005          TTABLE=  015652RG   016 WN.SRC   000000      012 XFOSMR=  000007
STX    =  000002 G      SU.XPD=  000003          TTBMSZ=  003270 G       WN.TYP   000001      012 XGTSRE=  000014
ST.ASZ  000020      006 S.HRL =  000240          TTSIZ =  015064RG   016 WORD0 =  000000          XHITSK=  000011
ST.BSZ  000024      006 TDABLK=  000004 G        VEC1MX=  000375 G       WORD1 =  000002          XHLMER=  000002
ST.BTC  000000      006 TDABMX=  007764 G        VEC2MX=  000375 G       WORD2 =  000004          XHOTSK=  000010
ST.CSZ  000030      006 TDAMAD=  007760 G        VEC3MX=  000125 G       WORD3 =  000006          XMSCHE=  000000
ST.HRL  000010      006 TDBBLK=  000003 G        VI       015052RG   016 WORD4 =  000010          XQTS   =  000003
ST.LEN  000044      006 TDBCLS=  000005 G        VIBCUT=  000036 G       WORD5 =  000012          XQT0   =  000001
ST.QRY  000002      006 TDBMAD=  007775 G        VILGT    015050RG   016 WORD6 =  000014          XSULOA=  000005
ST.QSZ  000034      006 TDBOSZ=  000074 G        VI0MSZ=  000400 G       WORD7 =  000016          XTABLE=  021142RG   016
ST.SCH  000040      006 TDCBLK=  000010 G        VI1MSZ=  000400 G       WORD8 =  000020          XTBMSZ=  002010 G
ST.UHL  000004      006 TDCMSZ=  004000 G        VI2MSZ=  000400 G       WORD9 =  000022          XTSIZ =  015066RG   016
ST.XLT  000014      006

. ABS.  000000     000
        000000     001
SRCOFF  000122     002
FDSCOF  000010     003
SUSOFF  000012     004
DHROFF  000012     005
STTOFF  000044     006
QSPLOF  000014     007
BSTOFF  000772     010
FNOFFS  000044     011
WNODOF  000010     012
DNODOF  000010     013
FLUNDE  000014     014
QNBASE  000010     015
AABUFS  052046     016
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  2901 WORDS  ( 12 PAGES)
DYNAMIC MEMORY:  3860 WORDS  ( 14 PAGES)
ELAPSED TIME:  00:00:24
BUFFER,BUFFER/-SP/NL:BEX=[20,1]P,M,QTSIZE,Q,BUFFER,QG,END
```

```
     1 000000                                .PSECT GSTAB
     2                                  ;
     3                                  ; TOKEN TYPE DEFINITIONS
     4                                  ;
     5          000000                  TOKFLU  ==      0               ;FLU
     6          000001                  TOKSBD  ==      1               ;SUBDOC
     7          000002                  TOKFMD  ==      2               ;FLUMOD
     8          000003                  TOKNOT  ==      3               ;NOT
     9          000004                  TOKOR   ==      4               ;OR
    10          000005                  TOKHOR  ==      5               ;HIGH-OR
    11          000007                  TOKAND  ==      7               ;AND
    12          000011                  TOKPXW  ==      9.              ;WORD PROXIMITY
    13          000013                  TOKPXS  ==      11.             ;SENTENCE PROXIMITY
    14          000014                  TOKLP   ==      12.             ;LEFT-NEST
    15          000015                  TOKPXP  ==      13.             ;PARAGRAPH PROXIMITY
    16          000016                  TOKRP   ==      14.             ;RIGHT NEST
    17          000017                  TOKETX  ==      15.             ;END OF TEXT
    18                                  ;
    19                                  ; LEXICAL TABLE 1 CODE DEFINITIONS
    20                                  ;
    21          000400                  LTFLU   ==      BIT8            ;CALL PFLU SUBROUTINE
    22          001000                  LTFMD   ==      BIT9            ;CALL PFLUMD SUBROUTINE
    23          002000                  LTPRX   ==      BIT10           ;CALL PPROX SUBROUTINE
    24          004000                  LTSKP   ==      BIT11           ;STREAM PAST CHARACTER
    25          010000                  LTTOK   ==      BIT12           ;SINGLE CHARACTER TOKEN
    26                                  ;
    27 000000                           LEXTB1::
    28 000000  000000  000000  000000           0,0,0                  ;NUL,SOH,STX
    29 000006  010017                           TOKETX!LTTOK           ;ETX
    30 000010  000000  000000  000000           0,0,0,0,0,0,0          ;EOT,ENQ,ACK,BEL,BS,HT,LF
       000016  000000  000000  000000
       000024  000000
    31 000026  000000  000000  000000           0,0,0,0,0,0            ;VT,FF,CR,SO,SI,DLE
       000034  000000  000000  000000
    32 000042  000000  000000  000000           0,0,0,0,0,0,0          ;DC1,DC2,DC3,DC4,NAK,SYN,ETB
       000050  000000  000000  000000
       000056  000000
    33 000060  000000  000000  000000           0,0,0,0,0,0,0,0        ;CAN,EM,SUB,ESC,FS,GS,RS,US
       000066  000000  000000  000000
       000074  000000  000000
    34 000100  004000                           LTSKP                  ;SPACE
    35 000102  000400                           LTFLU                  ;!
    36 000104  000400                           LTFLU                  ;"
    37 000106  000400                           LTFLU                  ;#
    38 000110  010001                           LTTOK!TOKSBD           ;$      (LTTOK!TOKSBD)
    39 000112  000400                           LTFLU                  ;%
    40 000114  010007                           TOKAND!LTTOK           ;&
    41 000116  000400                           LTFLU                  ;'
    42 000120  010014                           TOKLP!LTTOK            ;(
    43 000122  010016                           TOKRP!LTTOK            ;)
    44 000124  000400                           LTFLU                  ;*
    45 000126  010004                           TOKOR!LTTOK            ;+
    46 000130  000400                           LTFLU                  ;,
    47 000132  010003                           TOKNOT!LTTOK           ;-
    48 000134  000400                           LTFLU                  ;.
    49 000136  000400                           LTFLU                  ;/
    50 000140  000400                           LTFLU                  ;0
```

```
 51 000142  000400          LTFLU           ;1
 52 000144  000400          LTFLU           ;2
 53 000146  000400          LTFLU           ;3
 54 000150  000400          LTFLU           ;4
 55 000152  000400          LTFLU           ;5
 56 000154  000400          LTFLU           ;6
 57 000156  000400          LTFLU           ;7
 58 000160  000400          LTFLU           ;8
 59 000162  000400          LTFLU           ;9
 60 000164  000400          LTFLU           ;:
 61 000166  000400          LTFLU           ;;
 62 000170  002000          LTPRX           ;<
 63 000172  000400          LTFLU           ;=
 64 000174  000400          LTFLU           ;>
 65 000176  000400          LTFLU           ;?
 66 000200  000400          LTFLU           ;@
 67 000202  000400          LTFLU           ;A
 68 000204  000400          LTFLU           ;B
 69 000206  000400          LTFLU           ;C
 70 000210  000400          LTFLU           ;D
 71 000212  000400          LTFLU           ;E
 72 000214  000400          LTFLU           ;F
 73 000216  000400          LTFLU           ;G
 74 000220  000400          LTFLU           ;H
 75 000222  000400          LTFLU           ;I
 76 000224  000400          LTFLU           ;J
 77 000226  000400          LTFLU           ;K
 78 000230  000400          LTFLU           ;L
 79 000232  000400          LTFLU           ;M
 80 000234  000400          LTFLU           ;N
 81 000236  000400          LTFLU           ;O
 82 000240  000400          LTFLU           ;P
 83 000242  000400          LTFLU           ;Q
 84 000244  000400          LTFLU           ;R
 85 000246  000400          LTFLU           ;S
 86 000250  000400          LTFLU           ;T
 87 000252  000400          LTFLU           ;U
 88 000254  000400          LTFLU           ;V
 89 000256  000400          LTFLU           ;W
 90 000260  000400          LTFLU           ;X
 91 000262  000400          LTFLU           ;Y
 92 000264  000400          LTFLU           ;Z
 93 000266  001000          LTFMD           ;[
 94 000270  000400          LTFLU           ;\
 95 000272  000400          LTFLU           ;]
 96 000274  000400          LTFLU           ;↑
 97 000276  000400          LTFLU           ;←
 98 000300  000400          LTFLU           ;ACCENT GRAVE
 99 000302  000400          LTFLU           ;A
100 000304  000400          LTFLU           ;B
101 000306  000400          LTFLU           ;C
102 000310  000400          LTFLU           ;D
103 000312  000400          LTFLU           ;E
104 000314  000400          LTFLU           ;F
105 000316  000400          LTFLU           ;G
106 000320  000400          LTFLU           ;H
107 000322  000400          LTFLU           ;I
```

```
108 000324  000400                          LTFLU·                  ;J·
109 000326  000400                          LTFLU·                  ;K·
110 000330  000400                          LTFLU·                  ;L·
111 000332· 000400                          LTFLU·                  ;M·
112·000334  000400                          LTFLU·                  ;N·
113 000336  000400                          LTFLU·                  ;O·
114 000340  000400                          LTFLU·                  ;P·
115 000342· 000400                          LTFLU·                  ;Q·
116 000344  000400                          LTFLU·                  ;R·
117 000346  000400                          LTFLU·                  ;S·
118 000350  000400                          LTFLU·                  ;T·
119 000352· 000400                          LTFLU·                  ;U·
120 000354  000400                          LTFLU·                  ;V·
121 000356  000400                          LTFLU·                  ;W·
122·000360  000400                          LTFLU·                  ;X·
123 000362· 000400                          LTFLU·                  ;Y·
124 000364  000400                          LTFLU·                  ;Z·
125 000366  000400                          LTFLU·                  ;{
126 000370  000400                          LTFLU·                  ;|
127 000372· 000400                          LTFLU·                  ;RIGHT·BRACE·
128·000374  000400                          LTFLU·                  ;TILDE·
129 000376  000400                          LTFLU·                  ;DEL·
130                                 ;
```

```
                                ;
                                ; CST TABLE FOR FLU PARSING
                                ;
          000001                CSTNC==1              ;NON-CONTROL (SEARCHABLE)
          000002                CSTTS==2              ;TERM SEPARATOR
          000004                CSTSO==4              ;SEGMENT OPERATOR (NUMERICAL RANGE)
          000010                CSTESC==10      ;ESCAPE
          000020                CSTNF==20       ;NON-FLU
          000040                CSTVDC==40      ;VLDC
          000100                CSTFDC==100     ;FLDC
          000200                CSTDW==200      ;DUMMY WORD
          100000                CSTCET==100000  ;SEARCHABLE (CCET ENTRY FOR CHAR)
                                ;
000400  000000  000000  000000  FLUTBL::0,0,0                                 ;NULL,SOH,STX
000406  000020                        CSTNF                                   ;ETX
000410  000000  000000  000000        0,0,0,0,0                               ;EOT,ENQ,ACK,BEL,BS
000416  000000  000000
000422  000000  000000  000000        0,0,0,0,0,0,0,0                         ;HT,LF,VT,FF,CR,SO,SI,DLE
000430  000000  000000  000000
000436  000000  000000
000442  000000  000000  000000        0,0,0,0,0,0,0,0                         ;DC1,DC2,DC3,DC4,NAK,SYN,ETB,CAN
000450  000000  000000  000000
000456  000000  000000
000462  000000  000000  000000        0,0,0,0,0,0,0                           ;EM,SUB,ESC,FS,GS,RS,US
000470  000000  000000  000000
000476  000000
000500  000002                        CSTTS                                   ;SPACE
000502  000000                        0                                       ;!
000504  000000                        0                                       ;" (CSTDW)
000506  100001                        CSTNC!CSTCET                            ;#
000510  000020                        CSTNF                                   ;$
000512  100001                        CSTNC!CSTCET                            ;%
000514  100020                        CSTNF!CSTCET                            ;&
000516  100001                        CSTNC!CSTCET                            ;'
000520  000020                        CSTNF                                   ;(
000522  000020                        CSTNF                                   ;)
000524  100100                        CSTFDC!CSTCET                           ;*
000526  100020                        CSTNF!CSTCET                            ;+
000530  000000                        0                                       ;,
000532  100020                        CSTNF!CSTCET                            ;-
000534  100001                        CSTNC!CSTCET                            ;.
000536  100001                        CSTNC!CSTCET                            ;/
000540  100001                        CSTNC!CSTCET                            ;0
000542  100001                        CSTNC!CSTCET                            ;1
000544  100001                        CSTNC!CSTCET                            ;2
000546  100001                        CSTNC!CSTCET                            ;3
000550  100001                        CSTNC!CSTCET                            ;4
000552  100001                        CSTNC!CSTCET                            ;5
000554  100001                        CSTNC!CSTCET                            ;6
000556  100001                        CSTNC!CSTCET                            ;7
000560  100001                        CSTNC!CSTCET                            ;8
000562  100001                        CSTNC!CSTCET                            ;9
000564  000004                        CSTSO                                   ;:
000566  000000                        0                                       ;;
000570  000020                        CSTNF                                   ;<
000572  100001                        CSTNC!CSTCET                            ;= SIG. SPACE
000574  000020                        CSTNF                                   ;>
```

```
.MAIN.  MACRO M1110  27-MAR-80 12:53  PAGE 1-1

  182 000576  000040                CSTVDC                          ;?
  183 000600  000000                0                               ;@
  184 000602  100001                CSTNC!CSTCET                    ;A
  185 000604  100001                CSTNC!CSTCET                    ;B
  186 000606  100001                CSTNC!CSTCET                    ;C
  187 000610  100001                CSTNC!CSTCET                    ;D
  188 000612  100001                CSTNC!CSTCET                    ;E
  189 000614  100001                CSTNC!CSTCET                    ;F
  190 000616  100001                CSTNC!CSTCET                    ;G
  191 000620  100001                CSTNC!CSTCET                    ;H
  192 000622  100001                CSTNC!CSTCET                    ;I
  193 000624  100001                CSTNC!CSTCET                    ;J
  194 000626  100001                CSTNC!CSTCET                    ;K
  195 000630  100001                CSTNC!CSTCET                    ;L
  196 000632  100001                CSTNC!CSTCET                    ;M
  197 000634  100001                CSTNC!CSTCET                    ;N
  198 000636  100001                CSTNC!CSTCET                    ;O
  199 000640  100001                CSTNC!CSTCET                    ;P
  200 000642  100001                CSTNC!CSTCET                    ;Q
  201 000644  100001                CSTNC!CSTCET                    ;R
  202 000646  100001                CSTNC!CSTCET                    ;S
  203 000650  100001                CSTNC!CSTCET                    ;T
  204 000652  100001                CSTNC!CSTCET                    ;U
  205 000654  100001                CSTNC!CSTCET                    ;V
  206 000656  100001                CSTNC!CSTCET                    ;W
  207 000660  100001                CSTNC!CSTCET                    ;X
  208 000662  100001                CSTNC!CSTCET                    ;Y
  209 000664  100001                CSTNC!CSTCET                    ;Z
  210 000666  100020                CSTNF!CSTCET                    ;[
  211 000670  100001                CSTNC!CSTCET                    ;\ DECIMAL POINT
  212 000672  100020                CSTNF!CSTCET                    ;]
  213 000674  100010                CSTESC!CSTCET                   ;↑
  214 000676  000000                0                               ;←
  215 000700  000000                0                               ;'
  216 000702  100001                CSTNC!CSTCET                    ;A
  217 000704  100001                CSTNC!CSTCET                    ;B
  218 000706  100001                CSTNC!CSTCET                    ;C
  219 000710  100001                CSTNC!CSTCET                    ;D
  220 000712  100001                CSTNC!CSTCET                    ;E
  221 000714  100001                CSTNC!CSTCET                    ;F
  222 000716  100001                CSTNC!CSTCET                    ;G
  223 000720  100001                CSTNC!CSTCET                    ;H
  224 000722  100001                CSTNC!CSTCET                    ;I
  225 000724  100001                CSTNC!CSTCET                    ;J
  226 000726  100001                CSTNC!CSTCET                    ;K
  227 000730  100001                CSTNC!CSTCET                    ;L
  228 000732  100001                CSTNC!CSTCET                    ;M
  229 000734  100001                CSTNC!CSTCET                    ;N
  230 000736  100001                CSTNC!CSTCET                    ;O
  231 000740  100001                CSTNC!CSTCET                    ;P
  232 000742  100001                CSTNC!CSTCET                    ;Q
  233 000744  100001                CSTNC!CSTCET                    ;R
  234 000746  100001                CSTNC!CSTCET                    ;S
  235 000750  100001                CSTNC!CSTCET                    ;T
  236 000752  100001                CSTNC!CSTCET                    ;U
  237 000754  100001                CSTNC!CSTCET                    ;V
  238 000756  100001                CSTNC!CSTCET                    ;W
```

```
                                ;
                                ; LEXICAL TABLE 3 CODE DEFINITIONS (PROXIMITY)
000001                          CSTDT   ==      BIT0            ;DOCUMENT TYPE
000002                          CSTZ    ==      BIT1            ;ZONE
000004                          CSTSZ   ==      BIT2            ;SUBZONE
000010                          CSTPGR  ==      BIT3            ;PARAGRAPH UNIT
000020                          CSTSEN  ==      BIT4            ;SENTENCE UNIT
000040                          CSTWD   ==      BIT5            ;WORD UNIT
000100                          CSTNUM  ==      BIT6            ;NUMERIC
000200                          CSTDP   ==      BIT7            ;DECIMAL POINT
000400                          CSTMIN  ==      BIT8            ;MINUS
001000                          CSTP    ==      BIT9            ;PLUS
002000                          CSTER   ==      BIT10           ;RANGE SEPARATOR
004000                          CSTEP   ==      BIT11           ;END PROXIMITY
010000                          CSTEFM  ==      BIT12           ;END FLU MODIFIER
020000                          CSTZER  ==      BIT13           ;ZERO DIGIT INDICATOR
                                ;
000012                                  .RADIX  10
001000                          LEXTB3::
000200                                  .REPT   128.
                                        .WORD   BIT15
                                        .ENDR
                                ;
001126'                         .       =       LEXTB3+<43*2>
001126  001000                          .WORD   CSTP            ;+
001130                                  .BLKW   1
001132  000400                          .WORD   CSTMIN          ;-
001134  000200                          .WORD   CSTDP           ;.
001136  100000                          .WORD   BIT15           ;/
                                ;
001140  020100                          .WORD   CSTZER!CSTNUM   ;0
000011                                  .REPT   9
                                        .WORD   CSTNUM          ;1-9
                                        .ENDR
                                ;
001164  002000                          .WORD   CSTER           ;:
001166                                  .BLKW   3
001174  004000                          .WORD   CSTEP           ;>
                                ;
001240'                         .       =       LEXTB3+<80*2>
001240  000010                          .WORD   CSTPGR          ;P
001242                                  .BLKW   2
001246  000024                          .WORD   CSTSZ!CSTSEN    ;S
001250  000001                          .WORD   CSTDT           ;T
001252                                  .BLKW   2
001256  000040                          .WORD   CSTWD           ;W
001260                                  .BLKW   2
001264  000002                          .WORD   CSTZ            ;Z
                                ;
001266                                  .BLKW   2
001272  010000                          .WORD   CSTEFM          ;J
                                ;
001340'                         .       =       LEXTB3+<112*2>
001340  000010                          .WORD   CSTPGR          ;P
001342                                  .BLKW   2
001346  000024                          .WORD   CSTSZ!CSTSEN    ;S
001356  000001                          .WORD   CSTDT           ;T
```

```
 239 000760  100001          CSTNC!CSTCET            ;X
 240 000762  100001          CSTNC!CSTCET            ;Y
 241 000764  100001          CSTNC!CSTCET            ;Z
 242 000766  000000          0                       ;[
 243 000770  000000          0                       ;\
 244 000772  000000          0                       ;]
 245 000774  000000          0                       ;↑
 246 000776  000000          0                       ;DEL
```

```
305 001352                                      .BLKW   2
306 001356  000040                              .WORD   CSTWD           ;W
307 001360                                      .BLKW   2
308 001364  000002                              .WORD   CSTZ            ;Z
309                                 ;
310         001400'                 .           =       LEXTB3+<128*2>
311         000001                              .END
```

SYMBOL TABLE

```
BITVAL= 000000
BIT0  = 000001
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BS.CLS= 000002
BS.DBU= 000004
BS.INA= 000000
BS.OPN= 000001
BS.SRC= 000003
BYTE0 = 000000
BYTE1 = 000001
BYTE2 = 000002
BYTE3 = 000003
BYTE4 = 000004
BYTE5 = 000005
BYTE6 = 000006
BYTE7 = 000007
BYTE8 = 000010
BYTE9 = 000011
BYTVAL= 000012
B.BSTA  000054
B.CNTX  000046
B.CQUQ  000060
B.FEMA  000132
B.FEMB  000142
B.FEMC  000152
B.FFSA  000202
B.FFSB  000212
B.FFSC  000222
B.FMHR  000172
B.FQLS  000162
B.FSAZ  000100
B.FSBZ  000102
B.FSCZ  000104
B.HBLK  000120
B.HDOC  000114
B.HRLP  000126
B.HRLR  000122
B.HRLW  000124
B.NMBR  000052
B.NQRY  000232
B.QLSZ  000106
B.QMAP  000234
B.QSPL  000316
B.QTTM  000076
B.QUQP  000056
B.SFDB  000010
B.SIZE  000772
B.SNDP  000012
B.SSQ   000004
B.SSQF  000050
B.STAT  000044
B.STTE  000053
B.UDOC  000110
CF.B0 = 000070
CF.B2 = 000067
CF.B4 = 000066
CF.B6 = 000065
CF.DR0= 000064
CF.DR1= 000063
CSTCET= 100000 G
CSTDP = 000200 G
CSTDT = 000001 G
CSTDW = 000200 G
CSTEFM= 010000 G
CSTEP = 004000 G
CSTER = 002000 G
CSTESC= 000010 G
CSTFDC= 000100 G
CSTMIN= 000400 G
010 CSTNC = 000001 G
010 CSTNF = 000020 G
010 CSTNUM= 000100 G
010 CSTP  = 001000 G
010 CSTPGR= 000010 G
010 CSTSEN= 000020 G
010 CSTSO = 000004 G
010 CSTSZ = 000004 G
010 CSTTS = 000002 G
010 CSTVDC= 000040 G
010 CSTWD = 000040 G
010 CSTZ  = 000002 G
010 CSTZER= 020000 G
010 DBSLEN= 000116
010 DH.BF0  000002
010 DH.BFI  000004
010 DH.CTL  000000
010 DH.DMC  000010
010 DH.FLG  000006
010 DN.DCK  000000
010 DN.NTP  000004
010 DN.NXT  000006
010 DN.ROT  000002
010 DN.SIZ  000010
010 FD.FID  000000
010 FD.FNB  000006
010 FD.FVR  000004
010 FD.LEN  000010
010 FLUTBL  000400RG
010 FN.DBR  000026
010 FN.DBS  000022
010 FN.DHR  000040
010 FN.EMA  000012
010 FN.EMB  000014
    FN.EMC  000016
    FN.FSA  000000
    FN.FSB  000002
    FN.FSC  000004
    FN.LGD  000034
    FN.LGU  000036
    FN.MFD  000024
    FN.MHR  000010
    FN.NMB  000044
    FN.OLS  000006
    FN.ORY  000020
    FN.SF0  000030
    FN.SF1  000032
    FN.SHD  000042
    LEXTB1  000000RG
    LEXTB3  001000RG
    LTFLU = 000400 G
    LTFMD = 001000 G
    LTPRX = 002000 G
    LTSKP = 004000 G
    LTTOK = 010000 G
    M     = 000062
    N     = 000002
    N.BFAC= 000004
    N.BHGH= 000006
    N.BTCH= 000004
    N.BUFB= 004000
    N.BUFW= 002000
    N.FOS = 000764
    N.PKSZ= 000020
005 N.PKTS= 000043
005 N.QURY= 000031
005 N.SUNT= 000002
005 QE.RQ1= 000144
005 Q.FDSC  000004
013 Q.NQBK  000000
013 Q.NUHL  000002
013 Q.SIZE  000014
013 SR.ARE  000114
013 SR.ARS  000106
003 SR.DAY  000010
003 SR.DLT  000014
003 SR.ECB  000047
003 SR.ECH  000046
014 SR.ECL  000050
011 SR.FIB  000012
011 SR.GRE  000100
011 SR.GRS  000072
011 SR.LEN  000122
011 SR.LIN  000066
011 SR.LIP  000062
011 SR.MDN  000006
011 SR.NDC  000042
011 SR.NDS  000036
011 SR.NIN  000030
011 SR.NIP  000022
011 SR.SDB  000032
011 SR.SRC  000002
011 SR.SUN  000000
011 SR.TWS  000056
011 SR.WSL  000052
011 SR.YR   000004
011 SR.1IN  000024
011 SR.1IP  000016
014 SS.FID  000002
014 SS.FNB  000010
    SS.FVR  000006
    SS.LEN  000012
    SS.STT  000000
    ST.ASZ  000020
    ST.BSZ  000024
    ST.BTC  000000
    ST.CSZ  000030
    ST.HRL  000010
    ST.LEN  000044
    ST.QRY  000002
    ST.QSZ  000034
    ST.SCH  000040
    ST.UHL  000004
    ST.XLT  000014
    SU.DBU= 000004
    SU.DON= 000006
    SU.IDL= 000000
007 SU.LOD= 000001
007 SU.SRC= 000002
007 SU.SRR= 000005
007 SU.XPD= 000003
002 S.HRL = 000240
002 TOKAND= 000007 G
002 TOKETX= 000017 G
002 TOKFLU= 000000 G
002 TOKFMD= 000002 G
002 TOKHOR= 000005 G
002 TOKLP = 000014 G
002 TOKNOT= 000003 G
002 TOKOR = 000004 G
002 TOKPXP= 000015 G
002 TOKPXS= 000013 G
002 TOKPXW= 000011 G
002 TOKRP = 000016 G
002 TOKSBD= 000001 G
002 WN.NTP  000004  012
002 WN.NXT  000006  012
002 WN.ROT  000002  012
002 WN.SIZ  000010  012
002 WN.SRC  000000  012
002 WN.TYP  000001  012
002 WORD0 = 000000
002 WORD1 = 000002
002 WORD2 = 000004
002 WORD3 = 000006
002 WORD4 = 000010
002 WORD5 = 000012
004 WORD6 = 000014
004 WORD7 = 000016
004 WORD8 = 000020
004 WORD9 = 000022
004 WRDVAL= 000024
006 XBATCH= 000013
006 XDBLOA= 000004
006 XDBPRO= 000012
006 XDMCIN= 000006
006 XFOSMR= 000007
006 XGTSRE= 000014
006 XHITSK= 000011
006 XHLMER= 000002
006 XHOTSK= 000010
006 XMSCHE= 000000
006 XQTS  = 000003
    XQT0  = 000001
    XSULOA= 000005

. ABS.  000000  000
        000000  001
SRCOFF  000122  002
FDSCOF  000010  003
SUSOFF  000012  004
DHROFF  000012  005
```

SYMBOL TABLE

```
STTOFF  000044     006
QSPLOF  000014     007
BSTOFF  000772     010
FNOFFS  000044     011
WNODOF  000010     012
DNODOF  000010     013
CSTAB   001400     014
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  2258 WORDS  (9 PAGES)
DYNAMIC MEMORY:  2804 WORDS  ( 10 PAGES)
ELAPSED TIME:  00:00:25
CSTAB,CSTAB/-SP=[20,1]P,M,CSTAB
```

```
                                ;THIS FILE (E.MAC) CONTAINS ALL THE QUERY ERROR CODES.
                                ;
000001                          QE.DW1=1        ;RESERVED.
000002                          QE.MTS=2        ;MULTIPLE TERM SEP'S OR MISSING TERM.
000003                          QE.DW2=3        ;RESERVED.
000004                          QE.ILC=4        ;ILLEGAL CHARACTER.
000005                          QE.IHS=5        ;INTERNAL HSTS ERROR, LOGIC PROCESSING.
000006                          QE.NLQ=6        ;NULL QUERY.
000007                          QE.MOP=7        ;MISSING OPERATOR.
000010                          QE.MFM=8.       ;MISPLACED FLU-MOD.
000011                          QE.MSD=9.       ;MISPLACED SUBDOC OPERATOR.
000012                          QE.MNT=10.      ;MISPLACED NOT OPERATOR.
000013                          QE.MFL=11.      ;MISSING FLU / TWO CONSECUTIVE OPERATORS.
000014                          QE.UBP=12.      ;UNBALANCED PARENTHESES.
000015                          QE.INO=13.      ;INSUFFICIENT NUMBER OF OPERANDS.
000016                          QE.ISO=14.      ;ILLEGAL SUBELEMENT OPERAND.
000017                          QE.PX1=15.      ;MISSING PROX, WINDOW, OR DOC TYPE,ZONE, OR SUBZONE.
000020                          QE.PX2=16.      ;PROX, WINDOW OR FLU-MOD ID TOO LARGE.
000021                          QE.PX3=17.      ;MISSING PROX, UNIT.
000022                          QE.PX4=18.      ;MISSING PROX, DELIMITER
000023                          QE.NRI=19.      ;ILLEGAL NUMERICAL RANGE SPECIFICATION.
000024                          QE.NRB=20.      ;NUMERICAL RANGE BORDERED BY NUMERIC.
000025                          QE.NOS=21.      ;NO STX FOUND IN QUERY.
000026                          QE.FTB=22.      ;FLU TOO BIG.
000027                          QE.NTK=23.      ;UNDEFINED TOKEN.
                                ;
                                ;QE.RO1=100.    ;GENERAL RESOURCE OVERFLOW (LABEL DEFINED IN M.MAC)
000145                          ASKOVF=101.     ;ARGUMENT STACK OVERFLOW
000146                          LNPOVF=102.     ;LOGIC NODE POOL OVERFLOW.
000147                          TSKOVF=103.     ;TOKEN STACK OVERFLOW.
000150                          QBFOVF=104.     ;QUERY TOO BIG
000156                          QRYOVF=110.     ;# QUERIES OVERFLOWED.
000157                          POLOVF=111.     ;FLU POOL OVERFLOW.
000160                          FALOVF=112.     ;FAL
000161                          TTBOVF=113.     ;TTABLE.
000162                          XTBOVF=114.     ;XTABLE.
000163                          QLBOVF=115.     ;QLB
000164                          SLBOVF=116.     ;SDLB.
000165                          QEXOVF=117.     ;QEX
000166                          EMAOVF=118.     ;EMA
000167                          EMBOVF=119.     ;EMB
000170                          EMCOVF=120.     ;EMC
000171                          VI3OVF=121.     ;VI QT3
000172                          VI2OVF=122.     ;VI QT2.
000173                          VI1OVF=123.     ;VI QT1
000174                          TDBOVF=124.     ;TDCTB.
000175                          NDBOVF=125.     ;NODE POOL QT2
                                ;
```

```
                        ;
                        ; MACRO TO PRINT BUFFER
                        ; CREATES SPOOL FILE RECORDS
                        ;
                        .MACRO  PRT,START,END
                                MOV     R4,-(SP)
                                MOV     START,R4
                                MOV     R4,LOCAT
                                MOV     END,ENDLOC
                                JSR     PC,PRINT
                                MOV     (SP)+,R4
                        .ENDM
                        .MACRO  PRTH,START,END
                                MOV     R4,-(SP)
                                MOV     START,R4
                                MOV     R4,LOCAT
                                MOV     END,ENDLOC
                                JSR     PC,PRNTH
                                MOV     (SP)+,R4
                        .ENDM
                        ;
                        ; MACRO TO TEST CHARACTER SIGNIFICANCE
                        ; PARAMETERS ARE CONDITION,TRUE,AND FALSE
                        ;       CONDITION=THE TEST CONDITION
                        ;       TRUE    =PATH TO TAKE IF CONDITION TRUE
                        ;       FALSE   =PATH TO TAKE IF CONDITION FALSE
                        ; ANY PARAMETER CAN BE PROCEEDED BY "!" WHICH CAUSES THAT
                        ;  PARAMETER TO BE A SUBROUTINE CALL
                        ;
```

```
                                        .TITLE  PARSER ROUTINE - QUERY TRANSLATION MODULE
                                        .SBTTL  SYMBOLIC VARIABLES
000000                                  .PSECT  PSDATA
                                ;
000012                                  .RADIX  10
                                        .MCALL  PUT$S,MOUT$S
                                ;
                                .IIF    NDF     DEBUG,DEBUG=0   ;DEBUG SWITCH
                                .IIF    NDF     OUTPUT,OUTPUT=0 ;OUTPUT SWITCH
000000                          ASTKTP: .BLKW   1               ;CURRENT TOP OF ARGUMENT STACK
                                ;
                                ; SYNTHESIS TABLE CODE DEFINITIONS
                                ;
000001                          SYNNUL  =       BIT0            ;NO SYNTHETIC ACTION
000002                          SYNEST  =       BIT1            ;END NESTING TOKEN
000004                          SYNOP1  =       BIT2            ;UNARY OPERATOR
000010                          SYNSBD  =       BIT3            ;SUBELEMENT OPERAND ILLEGAL
000020                          SYNOP2  =       BIT4            ;BINARY OPERATOR
000040                          SYNVAL  =       BIT5            ;TOKEN HAS VALUE ATTRIBUTE
                                ;
```

```
                        ;       PRECEDENCE TABLE CODE DEFINITIONS
                        ;
000001                  LTPCD   =       BIT0            ;YIELDS TO
000004                  GTPCD   =       BIT2            ;PRECEDES
000002                  EQPCD   =       BIT1            ;HAS SAME PRECEDENCE AS
177751                  UNDEF   =       -QE.NTK         ;UNDEFINED TOKEN
177771                  ERCD1   =       -QE.MOP         ;MISSING OPERATOR
177770                  ERCD2   =       -QE.MFM         ;MISPLACED FLUMOD
177767                  ERCD3   =       -QE.MSD         ;MISPLACED SUBDOC
177762                  ERCD4   =       -QE.ISO         ;ILLEGAL SUBLELEM OPERAND
177766                  ERCD5   =       -QE.MNT         ;MISPLACED NOT OPERAND
177765                  ERCD6   =       -QE.MFL         ;MISSING FLU
177764                  ERCD7   =       -QE.UBP         ;UNBALANCED PARENTHESES
                        ;
```

```
                                .SBTTL  QUERY LOGIC NODE VARIABLE DEFINITIONS AND STORAGE
                        ;
                        ;       LOGIC NODE DEFINITIONS
                        ;
100000                  QNFLU   ==      BIT15           ;OPERAND WORD - SET IF FLU OR LOGICAL CONSTANT
040000                  QNFLS   ==      BIT14           ;OPERAND WORD - LOGICALLY FALSE CONSTANT
100000                  QNTREE  =       BIT15           ;ATTRIBUTE WORD - SUBTREES TRAVERSED
040000                  QNLPOL  =       BIT14           ;ATTRIBUTE WORD - NODE HAS LEFT POLARITY
020000                  QNRPOL  =       BIT13           ;ATTRIBUTE WORD - NODE HAS RIGHT POLARITY
010000                  QNPOL   =       BIT12           ;ATTRIBUTE WORD - NODE HAS POLARITY
004000                  QNSTAK  =       BIT11           ;ATTRIBUTE WORD - LEFT SUBTREE TRAVERSED
002000                  QNSBEL  =       BIT10           ;ATTRIBUTE WORD - SUBELEMENT ILLEGAL
000001                  QNHLOP  =       BIT0            ;OPERATOR CODE BYTE - HIGH LEVEL OPERATOR
                        ;
```

```
                                        .SBTTL  PRECEDENCE TABLE AND TOKEN OFFSET VECTOR
                                ;
                                ;       TOKEN OFFSET VECTOR
                                ;
000002                          PTVECT:
000002  001                             .BYTE   PCDFLU - PCDTBL         ;FLU OFFSET
000003  041                             .BYTE   PCDSBD - PCDTBL         ;SUBDOC OFFSET
000004  021                             .BYTE   PCDFMD - PCDTBL         ;FLUMOD OFFSET
000005  061                             .BYTE   PCDNOT - PCDTBL         ;NOT OFFSET
000006  101                             .BYTE   PCDOR  - PCDTBL         ;OR OFFSET
000007  101                             .BYTE   PCDHOR - PCDTBL         ;HIGH-OR OFFSET
000010  000                             .BYTE   0
000011  121                             .BYTE   PCDAND - PCDTBL         ;AND OFFSET
000012  000                             .BYTE   0
000013  141                             .BYTE   PCDPXW - PCDTBL         ;W-PROX OFFSET
000014  000                             .BYTE   0
000015  141                             .BYTE   PCDPXS - PCDTBL         ;S-PROX OFFSET
000016  161                             .BYTE   PCDLP  - PCDTBL         ;L-NEST OFFSET
000017  141                             .BYTE   PCDPXP - PCDTBL         ;P-PROX OFFSET
000020  201                             .BYTE   PCDRP  - PCDTBL         ;R-NEST OFFSET
000021  221                             .BYTE   PCDETX - PCDTBL         ;ETX OFFSET
                                ;
                                ; PRECEDENCE TABLE DEFINITION - IN COLUMN VECTOR ORDER
                                ;
000022                          PCDTBL:
000022  351                             .BYTE   UNDEF                   ;NON-TOKENS
000023                          PCDFLU:
000023  371                             .BYTE   ERCD1                   ;FLU INC, FLU
000024  001                             .BYTE   LTPCD                   ;SUBDOC < FLU
000025  004                             .BYTE   GTPCD                   ;FLUMOD > FLU
000026  001                             .BYTE   LTPCD                   ;NOT < FLU
000027  001                             .BYTE   LTPCD                   ;OR < FLU
000030  001                             .BYTE   LTPCD                   ;HIGH-OR < FLU
000031  351                             .BYTE   UNDEF
000032  001                             .BYTE   LTPCD                   ;AND < FLU
000033  351                             .BYTE   UNDEF
000034  001                             .BYTE   LTPCD                   ;W-PROX < FLU
000035  351                             .BYTE   UNDEF
000036  001                             .BYTE   LTPCD                   ;S-PROX < FLU
000037  001                             .BYTE   LTPCD                   ;LP < FLU
000040  001                             .BYTE   LTPCD                   ;P-PROX < FLU
000041  371                             .BYTE   ERCD1                   ;RP INC, FLU
000042  001                             .BYTE   LTPCD                   ;ETX < FLU
000043                          PCDFMD:
000043  370                             .BYTE   ERCD2                   ;FLU INC, FLUMOD
000044  001                             .BYTE   LTPCD                   ;SUBDOC < FLUMOD
000045  370                             .BYTE   ERCD2                   ;FLUMOD INC, FLUMOD
000046  001                             .BYTE   LTPCD                   ;NOT < FLUMOD
000047  001                             .BYTE   LTPCD                   ;OR < FLUMOD
000050  001                             .BYTE   LTPCD                   ;HIGH-OR < FLUMOD
000051  351                             .BYTE   UNDEF
000052  001                             .BYTE   LTPCD                   ;AND < FLUMOD
000053  351                             .BYTE   UNDEF
000054  001                             .BYTE   LTPCD                   ;W-PROX < FLUMOD
000055  351                             .BYTE   UNDEF
000056  001                             .BYTE   LTPCD                   ;S-PROX < FLUMOD
000057  001                             .BYTE   LTPCD                   ;LP < FLUMOD
```

```
    109 000060     001                     .BYTE   LTPCD           ;P-PROX < FLUMOD
    110 000061     001                     .BYTE   LTPCD           ;RP < FLUMOD
    111 000062     001                     .BYTE   LTPCD           ;ETX < FLUMOD
    112 000063                     PCDSBD:
    113 000063     367                     .BYTE   ERCD3           ;FLU INC, SUBDOC
    114 000064     367                     .BYTE   ERCD3           ;SUBDOC INC, SUBDOC
    115 000065     004                     .BYTE   GTPCD           ;FLUMOD > SUBDOC
    116 000066     001                     .BYTE   LTPCD           ;NOT < SUBDOC
    117 000067     001                     .BYTE   LTPCD           ;OR < SUBDOC
    118 000070     001                     .BYTE   LTPCD           ;HIGH-OR < SUBDOC
    119 000071     351                     .BYTE   UNDEF
    120 000072     001                     .BYTE   LTPCD           ;AND < SUBDOC
    121 000073     351                     .BYTE   UNDEF
    122 000074     362                     .BYTE   ERCD4           ;W-PROX INC, SUBDOC
    123 000075     351                     .BYTE   UNDEF
    124 000076     362                     .BYTE   ERCD4           ;S-PROX INC, SUBDOC
    125 000077     001                     .BYTE   LTPCD           ;LP < SUBDOC
    126 000100     362                     .BYTE   ERCD4           ;P-PROX INC, SUBDOC
    127 000101     367                     .BYTE   ERCD3           ;RP INC, SUBDOC
    128 000102     001                     .BYTE   LTPCD           ;ETX < SUBDOC
    129 000103                     PCDNOT:
    130 000103     366                     .BYTE   ERCD5           ;FLU INC, NOT
    131 000104     366                     .BYTE   ERCD5           ;SUBDOC INC, NOT
    132 000105     366                     .BYTE   ERCD5           ;FLUMOD INC, NOT
    133 000106     001                     .BYTE   LTPCD           ;NOT < NOT
    134 000107     001                     .BYTE   LTPCD           ;OR < NOT
    135 000110     001                     .BYTE   LTPCD           ;HIGH-OR < NOT
    136 000111     351                     .BYTE   UNDEF
    137 000112     001                     .BYTE   LTPCD           ;AND < NOT
    138 000113     351                     .BYTE   UNDEF
    139 000114     001                     .BYTE   LTPCD           ;W-PROX < NOT
    140 000115     351                     .BYTE   UNDEF
    141 000116     001                     .BYTE   LTPCD           ;S-PROX < NOT
    142 000117     001                     .BYTE   LTPCD           ;LP < NOT
    143 000120     001                     .BYTE   LTPCD           ;P-PROX < NOT
    144 000121     366                     .BYTE   ERCD5           ;RP INC, NOT
    145 000122     001                     .BYTE   LTPCD           ;ETX < NOT
    146 000123                     PCDOR:
    147 000123                     PCDHOR:
    148 000123     004                     .BYTE   GTPCD           ;FLU > OR
    149 000124     004                     .BYTE   GTPCD           ;SUBDOC > OR
    150 000125     365                     .BYTE   ERCD6           ;FLUMOD INC, OR
    151 000126     004                     .BYTE   GTPCD           ;NOT > OR
    152 000127     004                     .BYTE   GTPCD           ;OR > OR
    153 000130     004                     .BYTE   GTPCD           ;HIGH-OR > OR
    154 000131     351                     .BYTE   UNDEF
    155 000132     001                     .BYTE   LTPCD           ;AND < OR
    156 000133     351                     .BYTE   UNDEF
    157 000134     004                     .BYTE   GTPCD           ;W-PROX > OR
    158 000135     351                     .BYTE   UNDEF
    159 000136     004                     .BYTE   GTPCD           ;S-PROX > OR
    160 000137     001                     .BYTE   LTPCD           ;LP < OR
    161 000140     004                     .BYTE   GTPCD           ;P-PROX > OR
    162 000141     004                     .BYTE   GTPCD           ;RP > OR
    163 000142     001                     .BYTE   LTPCD           ;ETX < OR
    164 000143                     PCDAND:
    165 000143     004                     BYTE    GTPCD           ;FLU > AND
```

```
PRECEDENCE TABLE AND TOKEN OFFSET VECTOR

  166 000144     004                     .BYTE   GTPCD             ;SUBDOC > AND
  167 000145     365                     .BYTE   ERCD6             ;FLUMOD INC, AND
  168 000146     004                     .BYTE   GTPCD             ;NOT > AND
  169 000147     004                     .BYTE   GTPCD             ;OR > AND
  170 000150     004                     .BYTE   GTPCD             ;HIGH-OR > AND
  171 000151     351                     .BYTE   UNDEF
  172 000152     004                     .BYTE   GTPCD             ;AND > AND
  173 000153     351                     .BYTE   UNDEF
  174 000154     004                     .BYTE   GTPCD             ;W-PROX > AND
  175 000155     351                     .BYTE   UNDEF
  176 000156     004                     .BYTE   GTPCD             ;S-PROX > AND
  177 000157     001                     .BYTE   LTPCD             ;LP < AND
  178 000160     004                     .BYTE   GTPCD             ;P-PROX > AND
  179 000161     004                     .BYTE   GTPCD             ;RP > AND
  180 000162     001                     .BYTE   LTPCD             ;ETX < AND
  181 000163                     PCDPXW:
  182 000163                     PCDPXS:
  183 000163                     PCDPXP:
  184 000163     004                     .BYTE   GTPCD             ;FLU > PROXOP
  185 000164     004                     .BYTE   GTPCD             ;SUBDOC > PROXOP
  186 000165     365                     .BYTE   ERCD6             ;FLUMOD INC, PROXOP
  187 000166     004                     .BYTE   GTPCD             ;NOT > PROXOP
  188 000167     001                     .BYTE   LTPCD             ;OR < PROXOP
  189 000170     001                     .BYTE   LTPCD             ;HIGH-OR < PROXOP
  190 000171     351                     .BYTE   UNDEF
  191 000172     001                     .BYTE   LTPCD             ;AND < PROXOP
  192 000173     351                     .BYTE   UNDEF
  193 000174     004                     .BYTE   GTPCD             ;W-PROX > PROXOP
  194 000175     351                     .BYTE   UNDEF
  195 000176     004                     .BYTE   GTPCD             ;S-PROX > PROXOP
  196 000177     001                     .BYTE   LTPCD             ;LP < PROXOP
  197 000200     004                     .BYTE   GTPCD             ;P-PROX > PROXOP
  198 000201     004                     .BYTE   GTPCD             ;RP > PROXOP
  199 000202     001                     .BYTE   LTPCD             ;ETX < PROXOP
  200 000203                     PCDLP:
  201 000203     371                     .BYTE   ERCD1             ;FLU INC, LP
  202 000204     001                     .BYTE   LTPCD             ;SUBDOC < LP
  203 000205     004                     .BYTE   GTPCD             ;FLUMOD > LP
  204 000206     001                     .BYTE   LTPCD             ;NOT < LP
  205 000207     001                     .BYTE   LTPCD             ;OR < LP
  206 000210     001                     .BYTE   LTPCD             ;HIGH-OR < LP
  207 000211     351                     .BYTE   UNDEF
  208 000212     001                     .BYTE   LTPCD             ;AND < LP
  209 000213     351                     .BYTE   UNDEF
  210 000214     001                     .BYTE   LTPCD             ;W-PROX < LP
  211 000215     351                     .BYTE   UNDEF
  212 000216     001                     .BYTE   LTPCD             ;S-PROX < LP
  213 000217     001                     .BYTE   LTPCD             ;LP < LP
  214 000220     001                     .BYTE   LTPCD             ;P-PROX < LP
  215 000221     371                     .BYTE   ERCD1             ;RP INC, LP
  216 000222     001                     .BYTE   LTPCD             ;ETX < LP
  217 000223                     PCDRP:
  218 000223     004                     .BYTE   GTPCD             ;FLU > RP
  219 000224     004                     .BYTE   GTPCD             ;SUBDOC > RP
  220 000225     365                     .BYTE   ERCD6             ;FLUMOD INC, RP
  221 000226     004                     .BYTE   GTPCD             ;NOT > RP
  222 000227     004                     .BYTE   GTPCD             ;OR > RP
```

PARSER ROUTINE - QUERY TRANSLAT MACRO M1[illegible] [illegible] PAGE 58
PRECEDENCE TABLE AND TOKEN OFFSET VECTOR

```
 223 000230      004                     .BYTE   GTPCD                   ;HIGH-OR > RP
 224 000231      351                     .BYTE   UNDEF
 225 000232      004                     .BYTE   GTPCD                   ;AND > RP
 226 000233      351                     .BYTE   UNDEF
 227 000234      004                     .BYTE   GTPCD                   ;W-PROX > RP
 228 000235      351                     .BYTE   UNDEF
 229 000236      004                     .BYTE   GTPCD                   ;S-PROX > RP
 230 000237      002                     .BYTE   EQPCD                   ;LP = RP
 231 000240      004                     .BYTE   GTPCD                   ;P-PROX > RP
 232 000241      004                     .BYTE   GTPCD                   ;RP > RP
 233 000242      364                     .BYTE   ERCD7                   ;ETX INC, RP
 234 000243                      PCDETX:
 235 000243      004                     .BYTE   GTPCD                   ;FLU > ETX
 236 000244      004                     .BYTE   GTPCD                   ;SUBDOC > ETX
 237 000245      365                     .BYTE   ERCD6                   ;FLUMOD INC, ETX
 238 000246      004                     .BYTE   GTPCD                   ;NOT > ETX
 239 000247      004                     .BYTE   GTPCD                   ;OR > ETX
 240 000250      004                     .BYTE   GTPCD                   ;HIGH-OR > ETX
 241 000251      351                     .BYTE   UNDEF
 242 000252      004                     .BYTE   GTPCD                   ;AND > ETX
 243 000253      351                     .BYTE   UNDEF
 244 000254      004                     .BYTE   GTPCD                   ;W-PROX > ETX
 245 000255      351                     .BYTE   UNDEF
 246 000256      004                     .BYTE   GTPCD                   ;S-PROX > ETX
 247 000257      364                     .BYTE   ERCD7                   ;LP INC, ETX
 248 000260      004                     .BYTE   GTPCD                   ;P-PROX > ETX
 249 000261      004                     .BYTE   GTPCD                   ;RP > ETX
 250 000262      002                     .BYTE   EQPCD                   ;ETX = ETX
 251                             ;
 252                             ; SYNTHESIS TABLE
 253                             ;
 254 000263                      SYNTAB::
 255 000263      041                     .BYTE   SYNNUL!SYNVAL           ;FLU
 256 000264      014                     .BYTE   SYNOP1!SYNSBD           ;SUBDOC
 257 000265      001                     .BYTE   SYNNUL                  ;FLUMOD
 258 000266      004                     .BYTE   SYNOP1                  ;NOT
 259 000267      020                     .BYTE   SYNOP2                  ;OR
 260 000270      020                     .BYTE   SYNOP2                  ;HIGH-OR
 261 000271      000                     .BYTE   0                       ;NON-TOKEN
 262 000272      020                     .BYTE   SYNOP2                  ;AND
 263 000273      000                     .BYTE   0                       ;NON-TOKEN
 264 000274      070                     .BYTE   SYNOP2!SYNVAL!SYNSBD    ;W-PROXOP
 265 000275      000                     .BYTE   0
 266 000276      070                     .BYTE   SYNOP2!SYNVAL!SYNSBD    ;S-PROXOP
 267 000277      001                     .BYTE   SYNNUL                  ;L-NEST
 268 000300      070                     .BYTE   SYNOP2!SYNVAL!SYNSBD    ;P-PROXOP
 269 000301      002                     .BYTE   SYNEST                  ;R-NEST
 270 000302      001                     .BYTE   SYNNUL                  ;ETX
 271                             ;
```

```
                                        .SBTTL  MISCELLANEOUS VARIABLES AND DATA STORAGE
                                ;
000377          PWINMX  =       255                     ;MAXIMUM PROXIMITY WINDOW SIZE
177400          HIBYTE  =       ^B1111111100000000      ;HIGH ODER BYTE
000360          DIGZON  =       ^B11110000              ;ASCII DIGIT ZONE MASK
165777          QNXMPS  =       ^C<QNSBEL!QNPOL>        ;NODE INHERITED ATTRIBUTE MASK
                                ;
                                        .PSECT  PSDATA
                                        .EVEN
000303  QNDNXT::.BLKW   1                               ;POINTER TO NEXT AVAILABLE NODE
000304  TVSAV:  .BLKW   1                               ;TOKEN VALUE SAVE FIELD
000306  QBSAV:  .BLKW   1                               ;QUERY TEXT POINTER SAVE FIELD
000310  QRANK:  .BLKW   1                               ;QUERY RANK
000312  ;
                                        .IF     GT,OUTPUT
                                NMSGT0: .ASCII  /PARSER NORMAL EXIT, QUERY ROOT IS:  /
                                NMSGP0: .ASCII  /       /
                                NMSGL0  =       .-NMSGT0
                                EMSGT:  .ASCII  /PARSER QUERY ERROR, CODE IS:  /
                                EMSGP:  .ASCII  /       /
                                EMSGL   =       .-EMSGT
                                NMSGTi: .ASCII  /DUMP OF QUERY LOGIC NODES/
                                NMSGLi  =       .-NMSGTi
                                        .ENDC
                                ;
                                        .IF     GT,DEBUG
                                MSG1:   .ASCII  /PUSH TOKEN - DUMP OF TOKEN STACK/
                                MSG1L   =       .-MSG1
                                MSG2:   .ASCII  /POP TOKEN - DUMP OF TOKEN STACK/
                                MSG2L   =       .-MSG2
                                MSG3:   .ASCII  /PUSH LOGIC NODE - DUMP OF ARGUMENT STACK/
                                MSG3L   =       .-MSG3
                                MSG4:   .ASCII  /PUSH TOKEN VALUE - DUMP OF ARGUMENT STACK/
                                MSG4L   =       .-MSG4
                                MSG5:   .ASCII  /PUSH PROXIMITY WINDOW - DUMP OF ARGUMENT STACK/
                                MSG5L   =       .-MSG5
                                        .ENDC
```

```
                                        .SBTTL  PARSER - MAIN ROUTINE
                                ;
                                ;       REGISTER USAGE
                                ;
                                ;       R0 - SCRATCH USAGE;
                                ;       R1 - HOLDS TOKEN VALUES AND NODE ADDRESSES POPPED FROM ARGUMENT STACK
                                ;       R2 - HOLDS THE TOKEN TO BE SYNTHESIZED (POPPED FROM TOKEN STACK)
                                ;       R3 - PRECEDENCE COLUMN VECTOR BASE; ADDRESS OF NEW NODE
                                ;       R4 - CURRENT INPUT TOKEN
                                ;       R5 - TOP OF TOKEN STACK
                                ;
                                ;       AT EXIT FROM ROUTINE, (R2) = ADDRESS OF ROOT NODE OF QUERY OR FLU ID
                                ;
000000                                  .PSECT  PARSER
000000                          PARSER::
000000                                  SAVE    R1,R3,R4,R5             ;SAVE CALLER'S REGISTERS
000010  012767  000000G 000000'         MOV     #ARGSTK,ASTKTP          ;SET ARGUMENT STACK TOP TO STACK BASE
000016  005067  000000G                 CLR     ARGSTK                  ;INITIALIZE ARGUMENT STACK TO NULL QUERY
000022  012705  000000G                 MOV     #TOKSTK,R5              ;SET TOKEN STACK TOP TO STACK BASE
000026  112715  000000G                 MOVB    #TOKETX,(R5)            ;PUT ETX ON TOP OF STACK
000032  012767  000000G 000304'         MOV     #QNPOOL,QNDNXT          ;SET NEXT LOGIC NODE TO START OF POOL
000040  005067  000312'                 CLR     QRANK                   ;INITIALIZE QUERY RANK
000044  010167  000310'                 MOV     R1,QBSAV                ;INITIALIZE QUERY TEXT POINTER
                                ;
                                ;       SCAN FOR NEXT INPUT TOKEN, ON RETURN, R4 CONTAINS THE TOKEN TYPE CODE;
                                ;       IF THE TOKEN HAS A VALUE ATTRIBUTE, R3 CONTAINS THE VALUE
000050                          GETTOK:
000050                                  CALL    TOKSCN                  ;GET NEXT INPUT TOKEN
000054  010367  000306'                 MOV     R3,TVSAV                ;SAVE INPUT TOKEN VALUE
                                ;
                                ;       EXTRACT PRECEDENCE RELATION FROM TABLE FOR TOP OF STACK TOKEN
                                ;       AND CURRENT INPUT TOKEN
000060                          PRTEST:
000060  116403  000002'                 MOVB    PTVECT(R4),R3           ;LOAD INPUT TOKEN'S PCD VECTOR OFFSET
000064  042703  177400                  BIC     #HIBYTE,R3              ;CLEAR HIGH ORDER BYTE
000070  111500                          MOVB    @R5,R0                  ;FETCH TOP OF STACK TOKEN
000072  060003                          ADD     R0,R3                   ;AND USE AS INDEX INTO PCD VECTOR
000074  116300  000022'                 MOVB    PCDTBL(R3),R0           ;EXTRACT PRECEDENCE RELATION CODE
000100  100004                          BPL     1$                      ;TOKENS INCOMPARABLE IF SIGN BIT SET
000102  005400                          NEG     R0                      ;MAKE ERROR CODE POSITIVE
000104  010067  000000G                 MOV     R0,ERRCDE               ;PLACE ERROR CODE IN GLOBAL FIELD
000110  000455                          BR      XERROR
000112                          1$:
000112  120027  000002                  CMPB    R0,#EQPCD               ;CHECK THE PRECEDENCE RELATION
000116  103406                          BLO     TKPUSH                  ;STACK TOKEN YIELDS, PUSH INPUT TOKEN
000120  001402                          BEQ     2$                      ;EQUAL PRECEDENCE
000122  000167  000142                  JMP     PSYNTH                  ;STACK TOKEN PRECEDES, SYNTHESIZE IT
                                ;
                                ;       EQUAL STACK AND INPUT PRECEDENCE - THUS IF STACK TOKEN IS ETX,
                                ;       THE PARSE NOW TERMINATES; OTHERWISE, THE INPUT TOKEN IS PUSHED.
000126                          2$:
000126  122715  000000G                 CMPB    #TOKETX,@R5             ;IS STACK TOKEN ETX?
000132  001446                          BEQ     XPARSE
```

```
                                        ;PUSH INPUT TOKEN ONTO THE TOKEN STACK
000134                          TKPUSH:
000134  116400  000263'                MOVB    SYNTAB(R4),R0           ;GET SYNTHESIS BYTE FOR INPUT TOKEN
000140                          CHECK:  SYNVAL,,2$              ;BRANCH IF TOKEN HAS NO ASSOCIATED VALUE
000146  016703  000306'                MOV     TVSAV,R3                ;SET UP TOKEN VALUE FOR PUSHING
000152                          CALL    ARGPUT                  ;PUSH TOKEN VALUE INTO ARGUMENT STACK
000156                          2$:
000156  132764  000020  000263'        BITB    #SYNOP2,SYNTAB(R4)      ;IF BINARY OPERATOR, PERFORM QUERY RANK TEST
000164                                  BOFF    4$
000166  005767  000312'                TST     QRANK                   ;QUERY RANK MUST BE POSITIVE
000172  003004                          BGT     3$
000174  012767  000015  000000G       MOV     #QE.INO,ERRCDE          ;LOAD ERROR CODE
000202  000420                          BR      XERROR
000204                          3$:
000204  005367  000312'                DEC     QRANK                   ;REDUCE RANK FOR BINARY OPERATOR
000210                          4$:
000210  062705  000002                ADD     #2,R5                   ;ADJUST TOP OF TOKEN STACK FOR OVERFLOW TEST
000214  020567  000000'                CMP     R5,ASTKTP               ;COMPARE UPPER BOUNDARY OF TOKEN STACK
                                                                        ;WITH LOWER BOUNDARY OF ARGUMENT STACK
000220  103404                          BLO     5$                      ;BRANCH IF ENOUGH ROOM
                                        ;
000222  012767  000147  000000G       MOV     #TSKOVF,ERRCDE          ;SET TOKEN STACK OVERFLOW ERROR CODE
000230  000405                          BR      XERROR
000232                          5$:
000232  110445                          MOVB    R4,-(R5)                ;CORRECT STACK POINTER AND PUSH TOKEN
000234  000705                          BR      GETTOK
```

```
                                ;       NORMAL AND ERROR EXITS FROM PARSER
                                ;
000236                          XNOQRY:                                         ;NULL QUERY
000236  012767  000006  000000G         MOV     #QE,NLQ,ERRCDE                ;SET UP ERROR CODE
000244                          XERROR:
                                        .IF     GT,OUTPUT
                                        CVTASC  ERRCDE,EMSGP
                                        PUT$S   #LIST,#EMSGT,#EMSGL
                                        PUT$S   #LIST,#NMSGT1,#NMSGL1
                                        PRT     #QNPOOL,QNDNXT
                                        .ENDC
000244  000167  000000G               JMP     ERROR
                                ;
                                ;       PARSER NORMAL EXIT
                                ;
000250                          XPARSE:
000250  017702  000000'               MOV     @ASTKTP,R2                      ;ROOT NODE ADDRESS OR SINGLE TERM FLU ID
000254  001770                        BEQ     XNOQRY                          ;NULL QUERY
                                        .IF     GT,OUTPUT
                                        MOV     R2,TVSAV
                                        CVTASC  TVSAV,NMSGP0
                                        PUT$S   #LIST,#NMSGT0,#NMSGL0
                                        PUT$S   #LIST,#NMSGT1,#NMSGL1
                                        PRT     #QNPOOL,QNDNXT
                                        .ENDC
000256                                  RESTORE R1,R3,R4,R5                     ;RESTORE CALLER'S REGISTERS
000266  000207                          RETURN
```

```
                                        ;       SYNTHESIS ROUTINES FOR PARSER. THE TOKEN AT THE TOP OF THE TOKEN STACK
                                        ;       IS PROCESSED, AND THE NEW TOP OF STACK TOKEN IS TESTED FOR PRECEDENCE
                                        ;       WITH THE INPUT TOKEN
                                        ;
000270                                  PSYNTH:
000270  111502                          MOVB    @R5,R2                  ;POP TOKEN TO BE SYNTHESIZED
000272  005305                          DEC     R5
000274  116200  000263'                 MOVB    SYNTAB(R2),R0           ;GET SYNTACTIC SIGNIFICANCE
000300                                  CHECK   SYNNUL,PSYNX            ;IF NO SYNTHESIS REQUIRED, EXIT
000306                                  CHECK   SYNEST,,PSYNOP          ;IF R-NEST, POP L-NEST AND EXIT
000314  005305                          DEC     R5
000316                                  PSYNX:
000316  000167  177536                  JMP     PRTEST
                                        ;
                                        ;       SYNTHESIZE OPERATOR
000322                                  PSYNOP:
000322  005767  000312'                 TST     QRANK                   ;QUERY RANK MUST BE POSITIVE
000326  003004                          BGT     1$
000330  012767  000015  000000G        MOV     #QE.INO,ERRCDE          ;LOAD ERROR CODE
000336  000742                          BR      XERROR
000340                                  1$:
000340                                  CALL    GETQND                  ;GET LOGIC NODE - -> R3
000344  110213                          MOVB    R2,@R3                  ;STORE OPERATOR CODE IN RELATIVE BYTE 0
000346                                  CHECK   SYNOP1,PLARG            ;IF UNARY OPERATOR, NO RIGHT ARG
                                        ;
                                        ;       PROCESS RIGHT ARGUMENT WHICH IS NOW AT TOP OF ARGUMENT STACK
                                        ;
000354                                  CALL    ARGPOP                  ;POP RIGHT ARGUMENT FROM STACK INTO R1
000360                                  CALL    NODCHK                  ;SPECIAL NODE ATTRIBUTE PROCESSING
000364  103003                          BCC     2$                      ;BRANCH IF NO POLARITY
000366  052763  020000  000000G        BIS     #QNRPOL,QNATTR(R3)      ;SET RIGHT POLARITY OF NEW NODE
000374                                  2$:
000374  010163  000000G                MOV     R1,QNARG2(R3)           ;LINK RIGHT ARGUMENT TO NEW NODE
                                        ;
                                        ;       CHECK IF OPERATOR HAS AN ASSOCIATED TOKEN VALUE (I.E., PROXIMITY WINDOW)
                                        ;       IF SO, IT IS NOW AT THE TOP OF THE ARGUMENT STACK
                                        ;
000400                                  CHECK   SYNVAL,,PLARG           ;CHECK IF OPERATOR HAS TOKEN VALUE
000406                                  CALL    ARGPOP                  ;POP OPERATOR TOKEN VALUE - PUT IT IN
000412  110163  000000G                MOVB    R1,QNWIN(R3)            ;WINDOW FIELD OF LOGIC NODE
                                        ;
                                        ;       PROCESS LEFT ARGUMENT OF NODE
000416                                  PLARG:
000416                                  CALL    ARGPOP                  ;POP LEFT ARGUMENT FROM ARGUMENT STACK
000422                                  CALL    NODCHK                  ;SPECIAL NODE ATTRIBUTE PROCESSING
000426  103003                          BCC     1$                      ;BRANCH IF NO POLARITY
000430  052763  040000  000000G        BIS     #QNLPOL,QNATTR(R3)      ;SET LEFT POLARITY OF NEW NODE
000436                                  1$:
000436  032763  010000  000000G        BIT     #QNPOL,QNATTR(R3)       ;IF POLARITY BIT HAS NOT BEEN SET
000444                                  BOFF    2$                      ;OPERATOR IS OK AS IS
000446  052713  000001                  BIS     #QNHLOP,@R3             ;OTHERWISE, MARK OPERATOR HIGH-LEVEL
000452                                  2$:
000452  010163  000000G                MOV     R1,QNARG1(R3)           ;LINK LEFT ARGUMENT TO NEW NODE
000456                                  CALL    ARGPUT                  ;PUSH NEW NODE ADDRESS INTO ARGUMENT STACK
000462  000167  177372                  JMP     PRTEST                  ;EXIT SYNTHESIS ROUTINE
```

```
                                        .SBTTL  NODCHK SUBROUTINE
                                ;
                                ;       THIS SUBROUTINE SETS ATTRIBUTES INHERITED BY HIGH LEVEL NODES FROM THEIR ARGUMENTS
                                ;       IT FURTHER VERIFIES THAT A SUBDOC ELEMENT IS VALID FOR THE CURRENT OPERATOR
                                ;       (R1) - AN ARGUMENT OF THE NODE BEING CONSTRUCTED
                                ;       (R2) - THE OPERATOR BEING SYNTHESIZED
                                ;       (R3) - THE ADDRESS OF THE NODE BEING CONSTRUCTED
                                ;
                        NODCHK:
                                        TST     R1                      ;TEST TYPE OF ARGUMENT
                                        BMI     50$                     ;EXIT IF ARGUMENT FLU
                                        CALL    SPOL                    ;SET POLARITY SUBROUTINE
                                        BIT     #QNSBEL,QNATTR(R3)      ;IF ARGUMENT DOES NOT HAVE SUBELEM ATTRIBUTE
                                        BOFF    XNDCHK                  ;EXIT
                                        CHECK   SYNSBD,,XNDCHK          ;EXIT IF SUBELEM ARGUMENT IS OK
                                        MOV     #QE.ISD,ERRCDE          ;SET UP SUBELEMENT ERROR CODE
                                        JMP     XERROR
                        50$:
                                        CLC                             ;EXIT FOR FLU - NO POLARITY
                        XNDCHK:
                                        RETURN                          ;EXIT SUBROUTINE NODCHK
```

```
                                        .SBTTL  GET NEW QUERY LOGIC NODE SUBROUTINE
                                ;
                                ;       (R3) = BASE ADDRESS OF OBTAINED LOGIC NODE ON EXIT
                                ;       QNDNXT IS UPDATED TO POINT TO NEXT AVAILABLE NODE
                                ;       SUBROUTINE ZEROES OUT NEWLY ACQUIRED NODE
                                ;
000532                          GETQND:
000532  026727  000304' 000000G          CMP     QNDNXT,#QNDLST          ;ANY NODES LEFT IN POOL?
000540  103405                          BLO     1$                      ;BRANCH IF YES
                                ;
                                ;       ERROR - QUERY LOGIC NODE SPACE EXHAUSTED
                                ;
000542  012767  000146  000000G          MOV     #LNPOVF,ERRCDE          ;SET UP ERROR CODE
000550  000167  177470                  JMP     XERROR
                                ;
000554                          1$:
000554  062767  000000G 000304'         ADD     #QNDSIZ,QNDNXT          ;UPDATE NEXT AVAILABLE NODE POINTER
000562  016703  000304'                 MOV     QNDNXT,R3               ;SET R3 PAST NODE FOR ZEROING SEQUENCE
                                ;
000566  005043                          CLR     -(R3)
000570  005043                          CLR     -(R3)
000572  005043                          CLR     -(R3)
000574  005043                          CLR     -(R3)
                                ;
                                ;       R3 NOW HAS BASE ADDRESS OF NEW LOGIC NODE
                                ;
000576  000207                          RETURN
```

```
                                        .SBTTL  PUSH/POP ARGUMENT STACK SUBROUTINES

                                ;
                                ;       (R1) - VALUE POPPED FROM STACK ON EXIT FROM ARGPOP
                                ;       (R3) - INPUT VALUE TO BE PUSHED INTO STACK BY ARGPUT
                                ;       (R5) - CURRENT TOP OF TOKEN STACK (OVERFLOW BOUNDARY FOR ARGUMENT STACK)
                                ;
000600                          ARGPUT::
000600  162767  000002  000000' SUB     #2,ASTKTP               ;SET UP OVERFLOW TEST
000606  005205                  INC     R5                      ;TEMPORARILY ADJUST STACK BOUNDARY
000610  020567  000000'          CMP     R5,ASTKTP               ;CHECK LOW BOUNDARY
000614  103405                  BLO     1$                      ;BRANCH IF ROOM
                                ;
                                ;       ERROR -- ARGUMENT STACK OVERFLOW
                                ;
000616  012767  000145  000000G MOV     #ASKOVF,ERRCDE          ;SET UP ERROR CODE
000624  000167  177414          JMP     XERROR
                                ;
000630                          1$:
000630  010377  000000'          MOV     R3,@ASTKTP              ;MOVE VALUE INTO STACK
000634  005305                  DEC     R5                      ;READJUST LOW BOUNDARY
000636  000207                  RETURN
                                ;
000640                          ARGPOP:
000640  017701  000000'          MOV     @ASTKTP,R1              ;MOVE TOP OF ARGUMENT STACK VALUE
000644  062767  000002  000000' ADD     #2,ASTKTP               ;POP STACK - ADJUST TOP OF STACK
000652  000207                  RETURN
```

```
                                        .SBTTL  TOKEN SCANNER SUBROUTINE
                                ;
                                ;       REGISTER USAGE
                                ;
                                ;       (R1) - ADDRESS OF NEXT QUERY TEXT CHARACTER TO BE SCANNED, UPDATED
                                ;       (R2) - ADDRESS OF MAIN PARSING TABLE,
                                ;       (R3) - FLU ID (WHEN NEXT TOKEN IS FLU)
                                ;       (R4) - TOKEN TYPE CODE
                                ;       (R5) - TOP OF TOKEN STACK
                                ;       (R6) - TOP OF ARGUMENT STACK
                                ;
                                TOKSCN:
                                        MOV     QBSAV,R1                ;RESTORE QUERY TEXT POINTER
                                        MOV     #LEXTB1,R2              ;MAIN LEXICAL TABLE
                                1$:
                                        CALL    STEP                    ;SET CHARACTER SIGNIFICANCE IN R0
                                                                        ;AND STEP TO NEXT CHARACTER
                                ;
                                        CHECK   LTTOK,,2$               ;CHECK IF SINGLE CHARACTER TOKEN
                                        MOVB    R0,R4                   ;EXTRACT TOKEN TYPE CODE
                                        BR      XTSCAN                  ;EXIT FROM SCAN
                                ;
                                2$:
                                        CHECK   LTFLU,,3$               ;CHECK IF START OF FLU
                                        CALL    PFLU                    ;CALL FLU PROCESSING ROUTINE
                                        BIS     #QNFLU,R3               ;SET ARGUMENT SIGN BIT TO DISTINGUISH
                                                                        ;FLU ID FROM LOGIC NODE ADDRESS
                                        CLR     R4                      ;SET FLU TOKEN CODE (TOKFLU = 0)
                                        INC     QRANK                   ;UPDATE QUERY RANK
                                        BR      XTSCAN                  ;EXIT FROM SCAN
                                ;
                                3$:
                                        CHECK   LTFMD,,4$               ;CHECK IF START OF FLUMOD
                                        CALL    PFLUMD                  ;CALL FLUMOD PROCESSING ROUTINE
                                        MOV     #TOKFMD,R4              ;SET FLUMOD TOKEN TYPE CODE
                                        BR      XTSCAN                  ;EXIT FROM SCAN
                                ;
                                4$:
                                        CHECK   LTPRX,,5$               ;CHECK IF START OF PROXOP
                                        CALL    PPROX                   ;CALL PROXIMITY PROCESSING ROUTINE
                                ;
                                ;       PPROX SETS R4 TO APPROPRIATE TOKEN CODE BASED ON PROXIMITY UNIT
                                ;       IT ALSO COMPUTES AND STORES THE VALUE OF THE PROXIMITY WINDOW IN R3
                                ;
                                        BR      XTSCAN                  ;EXIT FROM SCAN
                                ;
                                5$:
                                        CHECK   LTSKP,1$                ;CHECK FOR STREAM PAST CHARACTER
                                ;
                                XTSCAN:
                                        MOV     R1,QBSAV                ;SAVE UPDATED QUERY TEXT POINTER
                                        RETURN
```

```
                                        .SBTTL  PROXIMITY SYNTAX ROUTINE
                                ;
                                ;       REGISTER USAGE
                                ;
                                ;       (R1) - ON ENTRY, POINTS TO THE FIRST CHARACTER PAST THE L-PROX
                                ;              ON EXIT, POINTS TO THE CHARACTER AFTER THE R-PROX
                                ;       (R2) - ADDRESS OF LEXICAL TABLE 3 (PROXIMITY AND FLUMOD CHARACTER SIGNIFICANCE)
                                ;       (R3) - ACCUMULATES BINARY VALUE OF PROXIMITY WINDOW
                                ;       (R4) - TOKEN TYPE CODE ACCORDING TO PROXIMITY UNIT
                                ;
001000                          PPROX:
001000  005003                         CLR     R3                      ;INITIALIZE PROXIMITY WINDOW ACCUMULATOR
001002  012702  000000G                MOV     #LEXTB3,R2              ;SET PROXIMITY LEXICAL TABLE
                                ;
001006  111104                         MOVB    @R1,R4                  ;SAVE POTENTIAL ASCII DIGIT
001010                                  CALL    STEP                    ;GET SIGNIFICANCE; STEP TO NEXT CHARACTER
001014                                  CHECK   CSTNUM,1$               ;VERIFY THAT WE HAVE FIRST DIGIT
                                ;
                                ;       ERROR - NO PROXIMITY WINDOW
                                ;
001022  012767  000017  000000G         MOV     #QE.PX1,ERRCDE          ;SET UP ERROR CODE
001030  000167  177210                  JMP     XERROR
                                ;
001034                          1$:
001034  142704  000360                  BICB    #DIGZON,R4              ;CLEAR HIGH ORDER BITS FROM ASCII DIGIT
001040  060403                         ADD     R4,R3                   ;ACCUMULATE VALUE OF DIGIT IN WINDOW
                                ;
                                ;       PROCESS NEXT CHARACTER (TRAILING DIGIT OR PROXIMITY UNIT)
                                ;
001042  111104                         MOVB    @R1,R4                  ;SAVE POTENTIAL DIGIT
001044                                  CALL    STEP                    ;GET SIGNIFICANCE; STEP TO NEXT CHARACTER
001050                                  CHECK   CSTNUM,,2$              ;BRANCH IF NON-NUMERIC TO VERIFY PROXIMITY UNIT
                                ;
001056  070327  000012                  MUL     #10,R3                  ;ACCUMULATE WINDOW REGISTER BASE 10
001062  020327  000377                  CMP     R3,#PWINMX              ;CHECK ACCUMULATED WINDOW SIZE
001066  003762                         BLE     1$                      ;BRANCH IF SIZE OK TO ACCUMULATE
                                ;
                                ;       ERROR - PROXIMITY VALUE TOO BIG
                                ;
001070  012767  000020  000000G         MOV     #QE.PX2,ERRCDE          ;SET UP ERROR CODE
001076  000167  177142                  JMP     XERROR
                                ;
001102                          2$:
001102                                  CHECK   CSTWD,,3$               ;CHECK IF WORD UNIT
001110  112704  000000G                 MOVB    #TOKPXW,R4              ;SET TOKEN TYPE CODE
001114  000421                         BR      XPREND
001116                          3$:
001116                                  CHECK   CSTSEN,,4$              ;CHECK IF SENTENCE UNIT
001124  112704  000000G                 MOVB    #TOKPXS,R4              ;SET TOKEN TYPE CODE
001130  000413                         BR      XPREND
001132                          4$:
001132                                  CHECK   CSTPGR,,5$              ;CHECK IF PARAGRAPH UNIT - IF NOT
                                                                        ;MISSING PROXIMITY UNIT ERROR
001140  112704  000000G                 MOVB    #TOKPXP,R4              ;SET TOKEN TYPE CODE
001144  000405                         BR      XPREND
                                ;
                                ;       ERROR - MISSING PROXIMITY UNIT
```

```
                                        ;
                                        5$:
001146                                  
001146  012767  000021  000000G          MOV     #QE.PX3,ERRCDE          ;SET UP ERROR CODE
001154  000167  177064                    JMP     XERROR
                                        ;
001160                                  XPREND:
001160                                  
001164                                  
                                        ;
                                        ;       ERROR - MISSING R-PROX DELIMITER
                                        ;
001172  012767  000022  000000G          MOV     #QE.PX4,ERRCDE          ;SET UP ERROR CODE
001200  000167  177040                    JMP     XERROR
                                        ;
001204                                  XPROXX:
001204  000207                          RETURN                          ;EXIT FROM PPROX
        000001                          .END
```

```
ARGPOP  000640R    015
ARGPUT  000600RG   015
ARGSTK= ****** GX
ASKOVF= 000145
ASTKTP  000000R    014
BITVAL= 000000
BIT0  = 000001
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BS.CLS= 000002
BS.DBU= 000004
BS.INA= 000000
BS.OPN= 000001
BS.SRC= 000003
BYTE0 = 000000
BYTE1 = 000001
BYTE2 = 000002
BYTE3 = 000003
BYTE4 = 000004
BYTE5 = 000005
BYTE6 = 000006
BYTE7 = 000007
BYTE8 = 000010
BYTE9 = 000011
BYTVAL= 000012
B.BSTA  000054     010
B.CNTX  000046     010
B.CQUQ  000060     010
B.FEMA  000132     010
B.FEMB  000142     010
B.FEMC  000152     010
B.FFSA  000202     010
B.FFSB  000212     010
B.FFSC  000222     010
B.FMHR  000172     010
B.FQLS  000162     010
B.FSAZ  000100     010
B.FSBZ  000102     010
B.FSCZ  000104     010
B.HBLK  000120     010
B.HDOC  000114     010
B.HRLP  000126     010
B.HRLR  000122     010
B.HRLW  000124     010
B.NMBR  000052     010
B.NQRY  000232     010
B.QLSZ  000106     010
B.QMAP  000234     010
B.QSPL  000316     010
B.QTTM  000076     010
B.QUQP  000056     010
B.SFDB  000010     010
B.SIZE  000772     010
B.SNDP  000012     010
B.SSQ   000004     010
B.SSQF  000050     010
B.STAT  000044     010
B.STTE  000053     010
B.UDOC  000110     010
CF.B0 = 000070
CF.B2 = 000067
CF.B4 = 000066
CF.B6 = 000065
CF.DR0= 000064
CF.DR1= 000063
CSTEP = ****** GX
CSTNUM= ****** GX
CSTPGR= ****** GX
CSTSEN= ****** GX
CSTWD = ****** GX
DBSLEN= 000116
DEBUG = 000000
DH.BF0  000002     005
DH.BF1  000004     005
DH.CTL  000000     005
DH.DMC  000010     005
DH.FLG  000006     005
DIGZON= 000360
DN.DCK  000000     013
DN.NTP  000004     013
DN.NXT  000006     013
DN.ROT  000002     013
DN.SIZ  000010     013
EMAOVF= 000166
EMBOVF= 000167
EMCOVF= 000170
EQPCD = 000002
ERCD1 = 177771
ERCD2 = 177770
ERCD3 = 177767
ERCD4 = 177762
ERCD5 = 177766
ERCD6 = 177765
ERCD7 = 177764
ERRCDE= ****** GX
ERROR = ****** GX
FALOVF= 000160
FD.FID  000000     003
FD.FNB  000006     003
FD.FVR  000004     003
FD.LEN  000010     003
FN.DBR  000026     011
FN.DBS  000022     011
FN.DHR  000040     011
FN.EMA  000012     011
FN.EMB  000014     011
FN.EMC  000016     011
FN.FSA  000000     011
FN.FSB  000002     011
FN.FSC  000004     011
FN.LGQ  000034     011
FN.LGU  000036     011
FN.MFO  000024     011
FN.MHR  000010     011
FN.NMB  000044     011
FN.QLS  000006     011
FN.QRY  000020     011
FN.SF0  000030     011
FN.SF1  000032     011
FN.SHD  000042     011
GETQND  000532R    015
GETTOK  000050R    015
GTPCD = 000004
HIBYTE= 177400
LEXTB1= ****** GX
LEXTB3= ****** GX
LNPOVF= 000146
LTFLU = ****** GX
LTFMD = ****** GX
LTPCD = 000001
LTPRX = ****** GX
LTSKP = ****** GX
LTTOK = ****** GX
M     = 000062
N     = 000002
NDBOVF= 000175
NODCHK  000466R    015
N.BFAC= 000004
N.BHGH= 000006
N.BTCH= 000004
N.BUFB= 004000
N.BUFW= 002000
N.FOS = 000764
N.PKSZ= 000020
N.PKTS= 000043
N.QURY= 000031
N.SUNT= 000002
OUTPUT= 000000
PARSER  000000RG   015
PCDAND  000143R    014
PCDETX  000243R    014
PCDFLU  000023R    014
PCDFMD  000043R    014
PCDHOR  000123R    014
PCDLP   000203R    014
PCDNOT  000103R    014
PCDOR   000123R    014
PCDPXP  000163R    014
PCDPXS  000163R    014
PCDPXW  000163R    014
PCDRP   000223R    014
PCDSBD  000063R    014
PCDTBL  000022R    014
PFLU  = ****** GX
PFLUMD= ****** GX
PLARG   000416R    015
POLOVF= 000157
PPROX   001000R    015
PRTEST  000060R    015
PSYNOP  000322R    015
PSYNTH  000270R    015
PSYNX   000316R    015
PTVECT  000002R    014
PWINMX= 000377
QBFOVF= 000150
QBSAV   000310R    014
QEXOVF= 000165
QE.DW1= 000001
QE.DW2= 000003
QE.FTB= 000026
QE.IHS= 000005
QE.ILC= 000004
QE.INO= 000015
QE.ISO= 000016
QE.MFL= 000013
QE.MFM= 000010
QE.MNT= 000012
QE.MOP= 000007
QE.MSD= 000011
QE.MTS= 000002
QE.NLQ= 000006
QE.NOS= 000025
QE.NRB= 000024
QE.NRI= 000023
QE.NTK= 000027
QE.PX1= 000017
QE.PX2= 000020
QE.PX3= 000021
QE.PX4= 000022
QE.RO1= 000144
QE.UBP= 000014
QLBOVF= 000163
QNARG1= ****** GX
QNARG2= ****** GX
QNATTR= ****** GX
QNDLST= ****** GX
QNDNXT  000304RG   014
QNDSIZ= ****** GX
QNFLS = 040000 G
QNFLU = 100000 G
QNHLOP= 000001
QNLPOL= 040000
QNPOL = 010000
QNPOOL= ****** GX
QNRPOL= 020000
QNSBEL= 002000
QNSTAK= 004000
QNTREE= 100000
QNWIN = ****** GX
QNXMPS= 165777
QRANK   000312R    014
QRYOVF= 000156
Q.FDSC  000004     007
Q.NQBK  000000     007
Q.NUHL  000002     007
Q.SIZE  000014     007
SLBOVF= 000164
SPOL  = ****** GX
SR.ARE  000114     002
SR.ARS  000106     002
SR.DAY  000010     002
SR.DLT  000014     002
SR.ECB  000047     002
SR.ECH  000046     002
SR.ECL  000050     002
SR.FIB  000012     002
SR.GRE  000100     002
SR.GRS  000072     002
SR.LEN  000122     002
SR.LIN  000066     002
SR.LIP  000062     002
SR.MON  000006     002
SR.NDC  000042     002
SR.NDS  000036     002
SR.NIN  000030     002
SR.NIP  000022     002
SR.SDB  000032     002
SR.SRC  000002     002
SR.SUN  000000     002
SR.TWS  000056     002
SR.WSL  000052     002
SR.YR   000004     002
SR.1IN  000024     002
SR.1IP  000016     002
SS.FID  000002     004
SS.FNB  000010     004
SS.FVR  000006     004
SS.LEN  000012     004
SS.STT  000000     004
STEP  = ****** GX
ST.ASZ  000020     006
ST.BSZ  000024     006
ST.BTC  000000     006
ST.CSZ  000030     006
ST.HRL  000010     006
ST.LEN  000044     006
ST.QRY  000002     006
ST.QSZ  000034     006
ST.SCH  000040     006
ST.UHL  000004     006
ST.XLT  000014     006
SU.DBU= 000004
```

```
PARSER ROUTINE - QUERY TRANSLAT MACRO M1148 [illegible]
SYMBOL TABLE

SU.DON= 000006          TDBOVF= 000174          VI2OVF= 000172          WORD6 = 000014          XHLMER= 000002
SU.IDL= 000000          TKPUSH  000134R     015 VI3OVF= 000171          WORD7 = 000016          XHOTSK= 000010
SU.LOD= 000001          TOKETX= ****** GX.      WN.NTP  000004      012 WORD8 = 000020          XMSCHE= 000000
SU.SRC= 000002          TOKFMD= ****** GX.      WN.NXT  000006      012 WORD9 = 000022          XNDCHK  000530R     015
SU.SRR= 000005          TOKPXP= ****** GX.      WN.ROT  000002      012 WRDVAL= 000024          XNOQRY  000236R     015
SU.XPD= 000003          TOKPXS= ****** GX.      WN.SIZ  000010      012 XBATCH= 000013          XPARSE  000250R     015
SYNEST= 000002          TOKPXW= ****** GX.      WN.SRC  000000      012 XDBLOA= 000004          XPREND  001160R     015
SYNNUL= 000001          TOKSCN  000654R     015 WN.TYP  000001      012 XDBPRO= 000012          XPROXX  001204R     015
SYNOP1= 000004          TOKSTK= ****** GX.      WORD0 = 000000          XDMCIN= 000006          XQTS  = 000003
SYNOP2= 000020          TSKOVF= 000147          WORD1 = 000002          XERROR  000244R     015 XQT0  = 000001
SYNSBD= 000010          TTBOVF= 000161          WORD2 = 000004          XFOSMR= 000007          XSULOA= 000005
SYNTAB  000263RG    014 TVSAV   000306R     014 WORD3 = 000006          XGTSRE= 000014          XTBOVF= 000162
SYNVAL= 000040          UNDEF = 177751          WORD4 = 000010          XHITSK= 000011          XTSCAN  000772R     015
S.HRL = 000240          VI1OVF= 000173          WORD5 = 000012

. ABS.  000000  000
        000000  001
SRCOFF  000122  002
FDSCOF  000010  003
SUSOFF  000012  004
DHROFF  000012  005
STTOFF  000044  006
QSPLOF  000014  007
BSTOFF  000772  010
FNOFFS  000044  011
WNODOF  000010  012
DNODOF  000010  013
PSDATA  000314  014
PARSER  001206  015
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  5880 WORDS  (23 PAGES)
DYNAMIC MEMORY:  7028 WORDS  ( 27 PAGES)
ELAPSED TIME:  00:02:00
PARSER,PARSER/NL:ME:BEX/-SP=[20,1]P,M,E,MDQT0,PARSER
```

.MAIN. MAC(10)M1110 27-MAR-80 12:57

TABLE OF CONTENTS

```
                                ;THIS FILE (E.MAC) CONTAINS ALL THE QUERY ERROR CODES
                                ;
000001                          QE.DW1=1        ;RESERVED
000002                          QE.MTS=2        ;MULTIPLE TERM SEP'S OR MISSING TERM
000003                          QE.DW2=3        ;RESERVED
000004                          QE.ILC=4        ;ILLEGAL CHARACTER
000005                          QE.IHS=5        ;INTERNAL HSTS ERROR, LOGIC PROCESSING
000006                          QE.NLQ=6        ;NULL QUERY
000007                          QE.MOP=7        ;MISSING OPERATOR
000010                          QE.MFM=8.       ;MISPLACED FLU-MOD
000011                          QE.MSD=9.       ;MISPLACED SUBDOC OPERATOR
000012                          QE.MNT=10.      ;MISPLACED NOT OPERATOR
000013                          QE.MFL=11.      ;MISSING FLU / TWO CONSECUTIVE OPERATORS
000014                          QE.UBP=12.      ;UNBALANCED PARENTHESES
000015                          QE.INO=13.      ;INSUFFICIENT NUMBER OF OPERANDS
000016                          QE.ISO=14.      ;ILLEGAL SUBELEMENT OPERAND
000017                          QE.PX1=15.      ;MISSING PROX, WINDOW, OR DOC TYPE,ZONE, OR SUBZONE
000020                          QE.PX2=16.      ;PROX, WINDOW OR FLU-MOD ID TOO LARGE
000021                          QE.PX3=17.      ;MISSING PROX, UNIT
000022                          QE.PX4=18.      ;MISSING PROX, DELIMITER
000023                          QE.NRI=19.      ;ILLEGAL NUMERICAL RANGE SPECIFICATION
000024                          QE.NRB=20.      ;NUMERICAL RANGE BORDERED BY NUMERIC
000025                          QE.NOS=21.      ;NO STX FOUND IN QUERY
000026                          QE.FTB=22.      ;FLU TOO BIG
000027                          QE.NTK=23.      ;UNDEFINED TOKEN
                                ;
                                ;QE.RO1=100.    ;GENERAL RESOURCE OVERFLOW (LABEL DEFINED IN M.MAC)
000145                          ASKOVF=101.     ;ARGUMENT STACK OVERFLOW
000146                          LNPOVF=102.     ;LOGIC NODE POOL OVERFLOW
000147                          TSKOVF=103.     ;TOKEN STACK OVERFLOW
000150                          QBFOVF=104.     ;QUERY TOO BIG
000156                          QRYOVF=110.     ;# QUERIES OVERFLOWED
000157                          POLOVF=111.     ;FLU POOL OVERFLOW
000160                          FALOVF=112.     ;FAL
000161                          TTBOVF=113.     ;TTABLE
000162                          XTBOVF=114.     ;XTABLE
000163                          QLBOVF=115.     ;QLB
000164                          SLBOVF=116.     ;SDLB
000165                          QEXOVF=117.     ;QEX
000166                          EMAOVF=118.     ;EMA
000167                          EMBOVF=119.     ;EMB
000170                          EMCOVF=120.     ;EMC
000171                          VI3OVF=121.     ;VI QT3
000172                          VI2OVF=122.     ;VI QT2
000173                          VI1OVF=123.     ;VI QT1
000174                          TDBOVF=124.     ;TDCTB
000175                          NDBOVF=125.     ;NODE POOL QT2
                                ;
```

```
                        ;
                        ; MACRO TO PRINT BUFFER.
                        ; CREATES SPOOL FILE RECORDS.
                        ;
                        .MACRO  PRT,START,END
                                MOV     R4,-(SP)
                                MOV     START,R4
                                MOV     R4,LOCAT
                                MOV     END,ENDLOC
                                JSR     PC,PRINT
                                MOV     (SP)+,R4
                        .ENDM
                        .MACRO  PRTH,START,END
                                MOV     R4,-(SP)
                                MOV     START,R4
                                MOV     R4,LOCAT
                                MOV     END,ENDLOC
                                JSR     PC,PRNTH
                                MOV     (SP)+,R4
                        .ENDM
                        ;
                        ; MACRO TO TEST CHARACTER SIGNIFICANCE.
                        ; PARAMETERS ARE CONDITION,TRUE,AND FALSE.
                        ;       CONDITION=THE TEST CONDITION.
                        ;       TRUE      =PATH TO TAKE IF CONDITION TRUE.
                        ;       FALSE     =PATH TO TAKE IF CONDITION FALSE.
                        ; ANY PARAMETER CAN BE PROCEEDED BY "!" WHICH CAUSES THAT
                        ;  PARAMETER TO BE A SUBROUTINE CALL.
                        ;
```

```
                                        .SBTTL  PRIVATE MACROS
000012                                  .RADIX  10
                                ;
                                .MACRO  PUSH    ARGLIST
                                        .IRP    ARG,<ARGLIST>
                                        .IF     NB,ARG
                                        MOV     ARG,-(SP)
                                        .ENDC
                                        .ENDM
                                        .ENDM
                                ;
                                ;       POP MACRO
                                ;
                                .MACRO  POP     ARG
                                        MOV     (SP)+,ARG
                                        .ENDM
                                ;
                                ;       OPGEN MACRO
                                ;
                                .MACRO  OPGEN   REG
                                        MOV     QNOPCD(REG),QNOPCD(NEW)
                                        .ENDM
                                ;
                                ;       OPCMP MACRO
                                ;
                                .MACRO  OPCMP   REG
                                        MOV     QNOPCD(REG),QNOPCD(NEW)
                                        MOV     #BIT1,R0
                                        XOR     R0,QNOPCD(NEW)
                                        .ENDM
                                ;
                                ;       NODE POP MACRO
                                ;
                                .MACRO  NODPOP  SL
                                        .IF     NB,SL
                                        MOV     (TOP)+,SL
                                        .IFF
                                        TST     (TOP)+
                                        .ENDC
                                        MOV     @TOP,STK
                                        .ENDM
                                ;
                                ;       CONVERT TO ASCII SETUP MACRO
                                ;
                                .MACRO  CVTASC  SRCE,TARG
                                        MOV     #SRCE,R4
                                        MOV     #TARG,R3
                                        MOV     #TYPE2B!BLKZER,R1
                                        CALL    OCTASC
                                        .ENDM
```

```
                        .TITLE  QUERY TRANSFORMATION PROGRAM
000012                  .RADIX  10
                        .NLIST  BEX
                        .MCALL  PUT$S

                ;
                ;       SPECIAL TABLE GENERATION MACROS STATE AND TFORM
                ;
                .MACRO  STATE   S
                        .WORD   S
                        .ENDM
                ;
                .MACRO  TFORM   R
                        .WORD   BIT15!R
                        .ENDM
```

```
                                .SBTTL  SYMBOLIC VARIABLES
                                ;
                                ;       CONDITIONAL ASSEMBLY VARIABLES
                                ;
                                .IIF    NDF     OUTPUT,OUTPUT=0
                                ;
                                ;       REGISTER DEFINITIONS
                                ;
000002                          CUR     =       %2                      ;POINTS TO CURRENT NODE
000003                          NEW     =       %3                      ;POINTS TO NEWLY GOTTEN NODE
000004                          STK     =       %4                      ;POINTS TO NODE AT TOP OF STACK
000005                          TOP     =       %5                      ;POINTS TO TOP OF STACK
                                ;
                                ;       LOGIC NODE DEFINITIONS
                                ;
100000                          QNFLU   =       BIT15           ;OPERAND WORD - SET IF FLU OR LOGICAL CONSTANT
040000                          QNFLS   =       BIT14           ;OPERAND WORD - LOGICALLY FALSE CONSTANT
100000                          QNTREE  =       BIT15           ;ATTRIBUTE WORD - SUBTREES TRAVERSED
040000                          QNLPOL  =       BIT14           ;ATTRIBUTE WORD - NODE HAS LEFT POLARITY
020000                          QNRPOL  =       BIT13           ;ATTRIBUTE WORD - NODE HAS RIGHT POLARITY
010000                          QNPOL   =       BIT12           ;ATTRIBUTE WORD - NODE HAS POLARITY
004000                          QNSTAK  =       BIT11           ;ATTRIBUTE WORD - LEFT SUBTREE TRAVERSED
002000                          QNSBEL  =       BIT10           ;ATTRIBUTE WORD - SUBELEMENT ILLEGAL
000001                          QNHLOP  =       BIT0            ;OPERATOR CODE BYTE - HIGH LEVEL OPERATOR
```

```
                        ;       SYNTHESIS TABLE CODE DEFINITIONS
                        ;
000001          SYNNUL  =       BIT0            ;NO SYNTHETIC ACTION
000002          SYNEST  =       BIT1            ;END NESTING TOKEN
000004          SYNOP1  =       BIT2            ;UNARY OPERATOR
000010          SYNSBD  =       BIT3            ;SUBELEMENT OPERAND ILLEGAL
000020          SYNOP2  =       BIT4            ;BINARY OPERATOR
000040          SYNVAL  =       BIT5            ;TOKEN HAS VALUE ATTRIBUTE
                        ;
                        ;       OPERATOR SIMPLIFICATION ATTRIBUTE DEFINITIONS
                        ;
000001          IDEMP   =       BIT0            ;OP(X,X)=X
000002          TAUTL   =       BIT1            ;OP(X,-X)=OP(-X,X)=TRUE (FALSE IF NOT SET)
000004          IDENT   =       BIT2            ;OP(X,TRUE)=OP(TRUE,X)=TRUE (X IF NOT SET)
000010          NILPT   =       BIT3            ;OP(X,FALSE)=OP(FALSE,X)=FALSE (X IF NOT SET)
000020          ABSOR   =       BIT4            ;OR(OP(A,B),A)=A (OP(A,B) IF NOT SET)
000040          ABSAND  =       BIT5            ;AND(OP(A,B),A)=A (OP(A,B) IF NOT SET)
000100          ABSPX0  =       BIT6            ;PROXOP(OP(A,B),A)=A  (INAPPLICABLE IF NOT SET)
000200          ABSPX1  =       BIT7            ;PROXOP(OP(A,B),A)=OP(A,B)  (INAPPLICABLE IF NOT SET
                        ;
                        ;       MISCELLANEOUS DEFINITIONS
                        ;
037777          LOGCON  =       ^O37777         ;MASK TO DISTINGUISH FLU FROM CONSTANT
104000          TRVRSE  =       ONTREE!ONSTAK   ;MASK TO CLEAR TRANSFORMED NODES
060000          LRPOL   =       ONLPOL!ONRPOL   ;MASK DEFINING LEFT OR RIGHT POLARITY
070000          POLRTY  =       ONPOL!LRPOL     ;MASK DEFINING ALL POLARITY BITS
165777          QNXMPS  =       ^C<ONSBEL!ONPOL>        ;;MASK FOR ALL EXCEPT POLARITY AND SUBELEM
```

```
                                .SBTTL  FINITE STATE CONTROL STRUCTURES
                                ;
                                ;       TRANSFORMATION CODES
                                ;
                                        .PSECT  TRCODE,ABS
000000                  UNDEF:  .BLKW   1
000000                  TD:     .BLKW   1
000002                  TN:     .BLKW   1
000004                  TC:     .BLKW   1
000006                  TH:     .BLKW   1
000010                  TA:     .BLKW   1
000012                  TS:     .BLKW   1
000014                  TPN:    .BLKW   1
000016                  TDN:    .BLKW   1
000020                  ;
                        ;       STATE CODES
                        ;
                                .PSECT  STCODE,ABS
000000                  S0:     .BLKB   1
000001                  S1:     .BLKB   1
000002                  S2:     .BLKB   1
000003                  S3:     .BLKB   1
000004                  S4:     .BLKB   1
000005                  S5:     .BLKB   1
000006                  S6:     .BLKB   1
                        ;
                        ;       TRANSFER VECTOR - TRANSFORM ROUTINES
                        ;
                                .PSECT  QTDATA
000000                  FSCTV:  .EVEN
000000  000220'                .WORD   QTX             ;UNDEFINED TRANSITION ROUTINE
000002  000312'                .WORD   QTD             ;TD  ROUTINE ADDRESS
000004  000462'                .WORD   QTN             ;TN  ROUTINE ADDRESS
000006  000622'                .WORD   QTC             ;TC  ROUTINE ADDRESS
000010  001250'                .WORD   QTH             ;TH  ROUTINE ADDRESS
000012  001110'                .WORD   QTA             ;TA  ROUTINE ADDRESS
000014  002430'                .WORD   QTS             ;TS  ROUTINE ADDRESS
000016  000736'                .WORD   QTPN            ;TPN ROUTINE ADDRESS
000020  001266'                .WORD   QTDN            ;TDN ROUTINE ADDRESS
                        ;
                        ;       FSC TABLE COLUMN VECTOR OFFSETS
                        FSCOFV:
000022                          .BYTE   FSCFLU-FSCTBL   ;CONSTANT
000022  126                     .BYTE   FSCSBD-FSCTBL   ;SUBDOC
000023  002                     .BYTE   0
000024  000                     .BYTE   FSCNOT-FSCTBL   ;NOT
000025  020                     .BYTE   FSCOR-FSCTBL    ;OR
000026  054                     .BYTE   FSCHOR-FSCTBL   ;HIGH-OR
000027  110                     .BYTE   0
000030  000                     .BYTE   FSCAND-FSCTBL   ;AND
000031  072                     .BYTE   0
000032  000                     .BYTE   FSCPXW-FSCTBL   ;W-PROX
000033  036                     .BYTE   0
000034  000                     .BYTE   FSCPXS-FSCTBL   ;S-PROX
000035  036                     .BYTE   0
000036  000                     .BYTE   FSCPXP-FSCTBL   ;P-PROX
000037  036                     ;
```

```
                        .SBTTL  FINITE STATE CONTROL TABLE
                FSCTBL: .EVEN
                        TFORM   0               ;NON-OPERATORS
                FSCSBD:
                        STATE   S0              ;T(S0,SUBDOC)=S0
                        STATE   S0              ;T(S1,SUBDOC)=S0
                        STATE   S0              ;T(S2,SUBDOC)=S0
                        STATE   S0              ;T(S3,SUBDOC)=S0
                        STATE   S0              ;T(S4,SUBDOC)=S0
                        STATE   S0              ;T(S5,SUBDOC)=S0
                        TFORM   UNDEF           ;T(S6,SUBDOC)=UNDEF
                FSCNOT:
                        STATE   S3              ;T(S0,NOT)=S3
                        TFORM   UNDEF           ;T(S1,NOT)=UNDEF
                        STATE   S3              ;T(S2,NOT)=S3
                        TFORM   TDN             ;T(S3,NOT)=TDN
                        TFORM   UNDEF           ;T(S4,NOT)=UNDEF
                        TFORM   TN              ;T(S5,NOT)=TN
                        TFORM   TPN             ;T(S6,NOT)=TPN
                FSCPXW:
                FSCPXS:
                FSCPXP:
                        STATE   S6              ;T(S0,PROXOP)=S6
                        STATE   S6              ;T(S1,PROXOP)=S6
                        STATE   S6              ;T(S2,PROXOP)=S6
                        STATE   S6              ;T(S3,PROXOP)=S6
                        STATE   S6              ;T(S4,PROXOP)=S6
                        STATE   S6              ;T(S5,PROXOP)=S6
                        STATE   S6              ;T(S6,PROXOP)=S6
                FSCOR:
                        STATE   S1              ;T(S0,OR)=S1
                        STATE   S1              ;T(S1,OR)=S1
                        STATE   S4              ;T(S2,OR)=S4
                        STATE   S1              ;T(S3,OR)=S1    /* MODIFICATION */
                        STATE   S4              ;T(S4,OR)=S4
                        STATE   S4              ;T(S5,OR)=S4
                        STATE   S6              ;T(S6,OR)=S6
                FSCAND:
                        STATE   S2              ;T(S0,AND)=S2
                        TFORM   TH              ;T(S1,AND)=TH
                        STATE   S2              ;T(S2,AND)=S2
                        STATE   S5              ;T(S3,AND)=S5
                        TFORM   TD              ;T(S4,AND)=TD
                        STATE   S5              ;T(S5,AND)=S5
                        TFORM   TD              ;T(S6,AND)=S6
                FSCHOR:
                        STATE   S0              ;T(S0,HIGH)=S0
                        TFORM   TA              ;T(S1,HIGH)=TA
                        TFORM   TD              ;T(S2,HIGH)=TD
                        TFORM   TC              ;T(S3,HIGH)=TC
                        TFORM   UNDEF           ;T(S4,HIGH)=UNDEF
                        TFORM   TD              ;T(S5,HIGH)=TD
                        TFORM   TD              ;T(S6,HIGH)=TD
```

```
                                        ;
                                        ;  RLOW --- SUBROUTINE TO CREATE LOW RANGE EMX TERMS.
                                        ;
003760  005767  174026          RLOW:   TST     LSZ             ;ANY CHARS BEFORE DEC POINT?
003764  001014                          BNE     14$             ;YES.
003766  005767  174024                  TST     LSZD            ;DEC NUMBER?.
003772  001405                          BEQ     15$             ;NO.
003774  016701  174000                  MOV     LBEG,R1
004000  004767  001056                  JSR     PC,RLOWD        ;YES: CREATE DEC LOW TERM ENTRIES.
004004  000207                          RTS     PC
004006  162767  000002  173772  15$:    SUB     #2,EXPVPT       ;OVERLAY INIT PTR.
004014  000207                          RTS     PC
004016  012767  000001  174020  14$:    MOV     #1,MMM          ;INITIALIZE.
004024  012767  000001  174014          MOV     #1,NNN.
004032  016701  173742                  MOV     LBEG,R1
004036  016702  173740                  MOV     HBEG,R2.
004042  026767  173744  173744          CMP     LSZ,HSZ         ;DO LOW AND HIGH CHAR COUNT MATCH?.
004050  001032                          BNE     20$             ;NO.
004052  112103                  10$:    MOVB    (R1)+,R3        ;R3=LOW.
004054  112204                          MOVB    (R2)+,R4        ;R4=HIGH
004056  026767  173762  173726  11$:    CMP     MMM,LSZ         ;LAST CHAR POSITION?.
004064  002410                          BLT     12$             ;NO.
004066  005767  173724                  TST     LSZD            ;YES: ANY DEC CHARS?.
004072  001005                          BNE     12$             ;YES.
004074  004767  001756                  JSR     PC,EMXNG        ;NO: ENTRY EMX ENTRY.
004100  005267  173744                  INC     NOHTRM          ;THERE WILL BE NO HIGH TERM!
004104  000207                          RTS     PC              ;FINISHED.
004106  005203                  12$:    INC     R3              ;ENTER ENTRY FOR LOW+1 TO HIGH-1
004110  005304                          DEC     R4
004112  004767  001740                  JSR     PC,EMXNG
004116  103005                          BCC     13$
004120  005303                          DEC     R3              ;NO SUCH ENTRY, ENTER LOW.
004122  004767  001742                  JSR     PC,EMXSNG
004126  005267  173714                  INC     NNN
004132  005267  173706          13$:    INC     MMM             ;SET TO PROCESS NEXT POSITION.
004136  112103                  20$:    MOVB    (R1)+,R3        ;R3=LOW.
004140  112704  000071                  MOVB    #'9,R4          ;R4=HIGH (=9)
004144  026767  173674  173674          CMP     MMM,NNN
004152  002422                          BLT     21$             ;IF M<N.
004154  003032                          BGT     24$             ;IF M>N.
004156  005767  173634                  TST     LSZD            ;ANY CHARS AFTER DEC POINT?.
004162  001004                          BNE     22$             ;YES.
004164  026767  173654  173620          CMP     MMM,LSZ         ;IF M=N, LAST POSITION?.
004172  001401                          BEQ     27$             ;YES: ENTER LOW TO 9
004174  005203                  22$:    INC     R3              ;NO: ENTER LOW+1 TO 9
004176  004767  001654          27$:    JSR     PC,EMXNG
004202  103010                          BCC     23$
004204  005303                          DEC     R3
004206  004767  001656                  JSR     PC,EMXSNG       ;NO SUCH ENTRY: ENTER 9
004212  005267  173630                  INC     NNN
004216  000402                          BR      23$
004220  004767  001644          21$:    JSR     PC,EMXSNG       ;ENTER LOW.
004224  005267  173614          23$:    INC     MMM
004230  026767  173610  173554          CMP     MMM,LSZ         ;FINISHED ALL CHAR POSITIONS?.
004236  003737                          BLE     20$             ;NO.
004240  000412                          BR      25$             ;YES.
004242  016705  173544          24$:    MOV     LSZ,R5          ;R5=REMAINING CHAR POSITIONS
```

```
731 004246  166705  173572                  SUB     MMM,R5
732 004252  005205                          INC     R5
733 004254  004767  002140                  JSR     PC,EMXNDG       ;ENTER TRAILING NFLDC ENTRIES
734 004260  005267  173562                  INC     NNN
735 004264  000424                          BR      252$
736 004266  032767  000010  173534  25$:    BIT     #DNUM,RNGCDE    ;DEC NUMBER?
737 004274                                  BOFF    251$            ;NO
738 004276  026767  173544  173506          CMP     NNN,LSZ         ;FINISHED TERMS FOR LOW?
739 004304  003032                          BGT     26$             ;YES
740 004306  016720  173524                  MOV     DPTR,(R0)+      ;YES: ENTER MERGE VECTOR
741 004312  005267  173530                  INC     NNN             ;NO
742 004316  000407                          BR      252$
743 004320  005020                  251$:   CLR     (R0)+           ;ALLOCATE MERGE ENTRY
744 004322  005267  173520                  INC     NNN             ;STEP TO NEXT TERM
745 004326  026767  173514  173456          CMP     NNN,LSZ         ;FINISHED TERMS FOR LOW?
746 004334  003016                          BGT     26$             ;YES
747 004336  012767  000001  173500  252$:   MOV     #1,MMM          ;RESET FOR NEXT TERM
748 004344  010003                          MOV     R0,R3           ;NO: ENTER THIS EMX LOC IN EXP VECTOR
749 004346  162703  000000G                 SUB     #EMX,R3
750 004352  010377  173430                  MOV     R3,@EXPVPT
751 004356  062767  000002  173422          ADD     #2,EXPVPT
752 004364  016701  173410                  MOV     LBEG,R1
753 004370  000662                          BR      20$
754 004372  005767  173420          26$:    TST     LSZD            ;ANY DEC CHARS?
755 004376  001402                          BEQ     261$            ;NO
756 004400  004767  000456                  JSR     PC,RLOWD        ;YES: ENTER DEC LOW TERMS
757 004404  000207                  261$:   RTS     PC
```

```
                                        ;
                                        ;  RMID --- SUBROUTINE TO CREATE MID TERM ENTRIES.
                                        ;
004406  016702  173400          RMID:   MOV     LSZ,R2          ;R2=# NFLDC'S TO ENTER FOR TERM
004412  010205                  1$:     MOV     R2,R5
004414  005202                          INC     R2
004416  010003                          MOV     R0,R3           ;ENTER THIS ENTRY IN EXP VECTOR
004420  162703  000000G                 SUB     #EMX,R3
004424  010377  173356                  MOV     R3,@EXPVPT
004430  062767  000002  173350          ADD     #2,EXPVPT
004436  012720  060000                  MOV     #60000,(R0)+    ;ENTER LEADING (1-9) CLUSTER
004442  012720  100000                  MOV     #BIT15,(R0)+
004446  012720  005774                  MOV     #5774,(R0)+
004452  004767  001742                  JSR     PC,EMXNDG       ;CREATE NFLDC ENTRIES
004456  077123                  3$:     SOB     R1,1$
004460  000207                          RTS     PC
```

```
                                        ;
                                        ;  RHIGH --- SUBROUTINE TO CREATE HIGH TERM ENTRIES.
                                        ;
004462  012767  000001  173354  RHIGH:  MOV     #1,MMM          ;INITIALIZE
004470  012767  000001  173350          MOV     #1,NNN
004476  016701  173276                  MOV     LBEG,R1
004502  016702  173274                  MOV     HBEG,R2
004506  010003                          MOV     R0,R3           ;ENTER THIS POS IN EXP VECTOR
004510  162703  000000G                 SUB     #EMX,R3
004514  010377  173266                  MOV     R3,@EXPVPT
004520  062767  000002  173260          ADD     #2,EXPVPT
004526  005767  173260                  TST     LSZ             ;IF LOW=0 AND DECIMAL
004532  001411                          BEQ     20$
004534  026767  173252  173252  1$:     CMP     LSZ,HSZ         ;# CHAR LOW = HIGH?
004542  001005                          BNE     20$             ;NO
004544  112103                  10$:    MOVB    (R1)+,R3        ;YES: R3=LOW
004546  112204                          MOVB    (R2)+,R4        ;R4=HIGH
004550  005267  173272          12$:    INC     NNN             ;SET TO PROCESS NEXT POSITION
004554  000416                          BR      30$
004556  112703  000061          20$:    MOVB    #'1,R3          ;R3=LOW (=1)
004562  112204                          MOVB    (R2)+,R4        ;R4=HIGH
004564  005267  173254                  INC     MMM
004570  005304                          DEC     R4              ;ENTER ENTRY FOR 1 TO HIGH-1
004572  004767  001260                  JSR     PC,EMXNG
004576  103004                          BCC     30$
004600  004767  001264                  JSR     PC,EMXSNG       ;NO SUCH ENTRY, ENTER 1 ,
004604  005267  173236                  INC     NNN
004610  112204                  30$:    MOVB    (R2)+,R4        ;R4=HIGH
004612  112703  000060          300$:   MOVB    #'0,R3          ;R3=LOW (=0)
004616  026767  173222  173222          CMP     MMM,NNN
004624  002414                          BLT     31$             ;IF M<N
004626  003036                          BGT     32$             ;IF M>N
004630  026767  173212  173156          CMP     NNN,HSZ         ;LAST POSITION
004636  001013                          BNE     33$             ;NO
004640  032767  000010  173162          BIT     #DNUM,RNGCDE    ;DEC NUMBER?
004646                                  BON     33$             ;YES
004650  004767  001202                  JSR     PC,EMXNG        ;NO; ENTER 0 TO HIGH
004654  000414                          BR      34$
004656  010403                  31$:    MOV     R4,R3           ;ENTER HIGH
004660  004767  001204                  JSR     PC,EMXSNG
004664  000410                          BR      34$
004666  005304                  33$:    DEC     R4              ;ENTER 0 TO HIGH-1
004670  004767  001162                  JSR     PC,EMXNG
004674  103004                          BCC     34$
004676  004767  001166                  JSR     PC,EMXSNG       ;NO SUCH ENTRY, ENTER 0
004702  005267  173140                  INC     NNN
004706  005267  173132          34$:    INC     MMM
004712  026767  173126  173074          CMP     MMM,HSZ         ;FINISHED ALL CHAR POSITIONS IN TERM?
004720  003733                          BLE     30$             ;NO
004722  000412                          BR      35$             ;YES
004724  016705  173064          32$:    MOV     HSZ,R5          ;R5=REMAINING CHAR POSITIONS
004730  166705  173110                  SUB     MMM,R5
004734  005205                          INC     R5
004736  004767  001456                  JSR     PC,EMXNDG       ;CREATE TRAILING NFLDC ENTRIES
004742  005267  173100                  INC     NNN
004746  000424                          BR      352$
004750  032767  000010  173052  35$:    BIT     #DNUM,RNGCDE    ;DEC NUMBER?
```

NUMRNG NUMBER RANGE CODE

```
  833 004756                                          BOFF:   351$            ;NO
  834 004760  026767  173062  173026                 CMP     NNN,HSZ         ;FINISHED ALL TERMS?
  835 004766  003032                                  BGT     36$             ;YES
  836 004770  016720  173042                          MOV     DPTR,(R0)+      ;YES: ENTER MERGE VECTOR
  837 004774  005267  173046                          INC     NNN             ;NO
  838 005000  000407                                  BR      352$
  839 005002  005020                          351$:   CLR     (R0)+           ;ALLOCATE MERGE ENTRY
  840 005004  005267  173036                          INC     NNN             ;STEP TO NEXT TERM
  841 005010  026767  173032  172776                 CMP     NNN,HSZ         ;FINISHED ALL TERMS?
  842 005016  003020                                  BGT     361$            ;YES
  843 005020  012767  000001  173016  352$:          MOV     #1,MMM          ;SET UP FOR NEXT TERM
  844 005026  010003                                  MOV     R0,R3           ;NO: ENTER EXP VECTOR AND RESET
  845 005030  162703  000000G                         SUB     #EMX,R3
  846 005034  010377  172746                          MOV     R3,@EXPVPT
  847 005040  062767  000002  172740                 ADD     #2,EXPVPT
  848 005046  016702  172730                          MOV     HBEG,R2
  849 005052  000656                                  BR      30$
  850 005054  004767  000416          36$:            JSR     PC,RHIGHD       ;YES: ENTER DEC TERMS
  851 005060  000207                  361$:           RTS     PC              ;YES
```

```
                                        ;
                                        ;  SUBROUTINE TO CREATE LOW TERMS FOR DEC ENTRIES.
                                        ;
005062  032767  000020  172740  RLOWD:  BIT     #PD,RNGCDE      ;PAST DECIMAL?
005070                                  BOFF    10$             ;NO
005072  005767  172720                  TST     LSZD            ;ANY CHARS?
005076  001041                          BNE     2$              ;YES
005100  032767  000201  172722          BIT     #MR2!MR3,RNGCDE
005106                                  BOFF    100$            ;IF NO PREVIOUS ZVLDC
005110  016720  172726                  MOV     ZPTR,(R0)+      ;IF PREVIOUS, LINK TO IT
000403                                  BR      101$
005116  010067  172720          100$:   MOV     R0,ZPTR         ;MERGE VECTOR TO ZVLDC
005122  005020                          CLR     (R0)+
005124  122777  000060  172650  101$:   CMPB    #'0,@HBEG       ;HIGH=0...?
005132  001001                          BNE     1$              ;NO
005134  000207                          RTS     PC              ;YES
005136  010003                  1$:     MOV     R0,R3           ;ENTER ENTRY IN EXP VECTOR
005140  162703  000000G                 SUB     #EMX,R3
005144  010377  172636                  MOV     R3,@EXPVPT
005150  062767  000002  172630          ADD     #2,EXPVPT
005156  112703  000060                  MOVB    #'0,R3          ;ENTER 0 TO HIGH-1
005162  117704  172614                  MOVB    @HBEG,R4
005166  005304                          DEC     R4
005170  004767  000662                  JSR     PC,EMXNG
005174  016720  172640                  MOV     NVPTR,(R0)+     ;ENTER MERGE VECTOR
005200  000207                          RTS     PC
005202  016701  172572          2$:     MOV     LBEG,R1         ;INITIALIZE
005206  016702  172570                  MOV     HBEG,R2
005212  012767  000001  172624          MOV     #1,MMM
005220  012767  000001  172620          MOV     #1,NNN
005226  112103                  3$:     MOVB    (R1)+,R3        ;R3=LOW
005230  112204                          MOVB    (R2)+,R4        ;R4=HIGH
005232  005304                          DEC     R4
005234  026767  172604  172554  6$:     CMP     MMM,LSZD        ;LAST TERM?
005242  001401                          BEQ     3$              ;YES: ENTER LOW TO HIGH-1
005244  005203                          INC     R3              ;NO: ENTER LOW+1 TO HIGH-1
005246  004767  000604                  JSR     PC,EMXNG
005252  103005                          BCC     4$
005254  005303                          DEC     R3              ;ENTER LOW, IF LOW=9
005256  004767  000606                  JSR     PC,EMXSNG
005262  005267  172560                  INC     NNN
005266  005267  172552          4$:     INC     MMM             ;STEP TO NEXT CHAR
005272  016720  172542                  MOV     NVPTR,(R0)+     ;ENTER MERGE VECTOR
005276  005267  172544                  INC     NNN             ;STEP TO NEXT TERM
005302  026767  172540  172506          CMP     NNN,LSZD        ;FINISHED ALL TERMS?
005310  003051                          BGT     7$              ;YES
005312  010003                          MOV     R0,R3           ;ENTER THIS TERM IN EXP VECTOR
005314  162703  000000G                 SUB     #EMX,R3
005320  010377  172462                  MOV     R3,@EXPVPT
005324  062767  000002  172454          ADD     #2,EXPVPT
005332  032767  000020  172470          BIT     #PD,RNGCDE      ;PAST DEC POINT
005340                                  BON     11$             ;YES
005342  016701  172432                  MOV     LBEG,R1         ;NO: REPEAT NUMBER BEFORE DEC
005346  016702  172440                  MOV     LSZ,R2
005352  001404                          BEQ     52$             ;IF NO NUMBER
005354  112103                  51$:    MOVB    (R1)+,R3        ;GET LOW
005356  004767  000506                  JSR     PC,EMXSNG       ;ENTER LOW
```

```
    910 005362  077204                          SOB     R2,51$
    911 005364  012702  000000G         52$:    MOV     #DPEMX,R2       ;ENTER DEC POINT
    912 005370  012220                          MOV     (R2)+,(R0)+
    913 005372  012220                          MOV     (R2)+,(R0)+
    914 005374  012220                          MOV     (R2)+,(R0)+
    915 005376  005201                          INC     R1
    916 005400  000402                          BR      12$
    917 005402  016701  172372          11$:    MOV     LBEG,R1         ;START OF NUMBER AFTER DEC POINT
    918 005406  016702  172434          12$:    MOV     NNN,R2
    919 005412  005302                          DEC     R2              ;R2=# LEADING CHARS
    920 005414  112103                  5$:     MOVB    (R1)+,R3        ;GET CHAR
    921 005416  004767  000446                  JSR     PC,EMXSNG       ;ENTER CHAR
    922 005422  077204                          SOB     R2,5$
    923 005424  112103                          MOVB    (R1)+,R3        ;R3=LOW
    924 005426  112704  000071                  MOVB    #'9,R4          ;R4=HIGH (=9)
    925 005432  000700                          BR      6$
    926 005434  000207                  7$:     RTS     PC
    927 005436  005201                  10$:    INC     R1              ;STEP PAST DEC POINT
    928 005440  012767  000001  172376          MOV     #1,MMM          ;INIT
    929 005446  012767  000001  172372          MOV     #1,NNN
    930 005454  112103                          MOVB    (R1)+,R3        ;R3=LOW
    931 005456  112704  000071                  MOVB    #'9,R4          ;R4=HIGH (=9)
    932 005462  012702  000000G                 MOV     #DPEMX,R2       ;ENTER DEC POINT
    933 005466  012220                          MOV     (R2)+,(R0)+
    934 005470  012220                          MOV     (R2)+,(R0)+
    935 005472  012220                          MOV     (R2)+,(R0)+
    936 005474  000657                          BR      6$
```

```
                                        ;
                                        ;  SUBROUTINE TO CREATE HIGH TERM DECIMAL ENTRIES.
                                        ;
005476  032767  000020  172324  RHIGHD: BIT     #PD,RNGCDE      ;PAST DECIMAL?
005504                                BOFF    10$             ;NO
005506  012767  000002  172330          MOV     #2,MMM          ;INIT
005514  012767  000002  172324          MOV     #2,NNN
005522  000462                        BR      4$
005524  032767  000400  172276  101$:  BIT     #MR1,RNGCDE
005532                                BOFF    1$              ;IF NOT MULTI-RANGE, JUST ENTRY ZVLDC
005534  000412                        BR      100$            ;IF MR, CHECK FURTHER
005536  005767  172300          7$:   TST     ZPTR            ;WAS A MERGE TO ZVLDC CREATED?
005542  001770                        BEQ     101$            ;NO
005544  032767  000201  172256          BIT     #MR2!MR3,RNGCDE
005552                                BOFF    100$            ;YES
005554  016720  172262                MOV     ZPTR,(R0)+      ;YES, IN FACT THE ZVLDC WAS ALSO
005560  000207                        RTS     PC
005562  010001                  100$: MOV     R0,R1           ;POINT MERGE TO HERE
005564  162701  177776G                SUB     #EMX-2,R1
005570  052701  000000G                BIS     #EMXNSQ,R1
005574  010177  172242                MOV     R1,@ZPTR
005600  010167  172236                MOV     R1,ZPTR
005604  010120                        MOV     R1,(R0)+        ;PUT MERGE IN THIS PATH ALSO
005606  012701  000000G          1$:   MOV     #ZVDEMX,R1      ;ENTER ZVLDC
005612  012120                        MOV     (R1)+,(R0)+
005614  012120                        MOV     (R1)+,(R0)+
005616  012120                        MOV     (R1)+,(R0)+
005620  005020                        CLR     (R0)+           ;ALLOCATE MERGE VECTOR
005622  000207                        RTS     PC
005624  112703  000060          2$:   MOVB    #'0,R3          ;R3=LOW (=0)
005630  112204                        MOVB    (R2)+,R4        ;R4=HIGH
005632  005304                        DEC     R4              ;ENTER 0 TO HIGH-1
005634  004767  000216                JSR     PC,EMXNG
005640  103007                        BCC     3$
005642  004767  000222                JSR     PC,EMXSNG       ;ENTER LOW IF HIGH=0
005646  005267  172174                INC     NNN
005652  005267  172166                INC     MMM
005656  000762                        BR      2$
005660  016720  172154          3$:   MOV     NVPTR,(R0)+     ;ENTER MERGE VECTOR
005664  005267  172156                INC     NNN             ;STEP TO NEXT TERM
005670  010003                  4$:   MOV     R0,R3           ;ENTER THIS TERM IN EXP VECTOR
005672  162703  000000G                SUB     #EMX,R3
005676  010377  172104                MOV     R3,@EXPVPT
005702  062767  000002  172076        ADD     #2,EXPVPT
005710  032767  000020  172112        BIT     #PD,RNGCDE      ;PAST DECIMAL?
005716                                BON     11$             ;YES
005720  016702  172056                MOV     HBEG,R2         ;NO
005724  016701  172064                MOV     HSZ,R1
005730  001404                        BEQ     52$             ;IF NO HIGH
005732  112203                  51$:  MOVB    (R2)+,R3        ;ENTER HIGH BEFORE DEC
005734  004767  000130                JSR     PC,EMXSNG
005740  077104                        SOB     R1,51$
005742  012701  000000G          52$:  MOV     #DPEMX,R1       ;ENTER DEC POINT
005746  012120                        MOV     (R1)+,(R0)+
005750  012120                        MOV     (R1)+,(R0)+
005752  012120                        MOV     (R1)+,(R0)+
005754  005202                        INC     R2              ;STEP PAST DEC
```

NUMRNG NUMBER RANGE CODE

```
  995 005756  000402                          BR      12$
  996 005760  016702  172016          11$:    MOV     HBEG,R2         ;R2=START OF NUMBER PAST DEC POINT
  997 005764  016701  172056          12$:    MOV     NNN,R1
  998 005770  005301                          DEC     R1              ;R1=# LEADING CHARS
  999 005772  112203                  5$:     MOVB    (R2)+,R3        ;GET HIGH
 1000 005774  004767  000070                  JSR     PC,EMXSNG       ;ENTER CHAR
 1001 006000  077104                          SOB     R1,5$
 1002 006002  026767  172040  172010          CMP     NNN,HSZD        ;LAST CHAR
 1003 006010  003252                          BGT     7$              ;YES
 1004 006012  000704                          BR      2$              ;NO
 1005 006014  012767  000001  172022  10$:    MOV     #1,MMM          ;INIT
 1006 006022  012767  000001  172016          MOV     #1,NNN
 1007 006030  005202                          INC     R2              ;STEP PAST DEC
 1008 006032  012701  000000G                 MOV     #DPEMX,R1       ;ENTER DEC POINT
 1009 006036  012120                          MOV     (R1)+,(R0)+
 1010 006040  012120                          MOV     (R1)+,(R0)+
 1011 006042  012120                          MOV     (R1)+,(R0)+
 1012 006044  005767  171750                  TST     HSZD
 1013 006050  001265                          BNE     2$
 1014 006052  000167  177460                  JMP     7$
```

```
1016                                    ;
1017                                    ;  SUPPORT SUBROUTINES: THESE CREATE EMX ENTRIES
1018                                    ;
1019 006056  020304                     EMXNG:  CMP     R3,R4           ;CREATE NUMERICAL ENTRY
1020 006060  002442                             BLT     EMXMNG          ;LOW<HIGH: CREATE MULTIVALUE ENTRY
1021 006062  001402                             BEQ     EMXSNG          ;LOW=HIGH: CREATE SINGLE VALUE ENTRY
1022 006064  000261                             SEC                     ;LOW>HIGH: ILLEGAL
1023 006066  000207                             RTS     PC
1024                                    ;
1025 006070  042703  000060             EMXSNG: BIC     #60,R3          ;CREATE SINGLE VALUE ENTRY
1026 006074  006303                             ASL     R3              ;R3=INPUT NUMBER
1027 006076  016320  006116'                    MOV     20$(R3),(R0)+   ;ENTER CONTOL WORD FOR NUMBER
1028 006102  012720  100000                     MOV     #BIT15,(R0)+    ;ENTER NIB1
1029 006106  016320  006142'                    MOV     30$(R3),(R0)+   ;ENTER NIB2 FOR NUMBER
1030 006112  000241                             CLC
1031 006114  000207                             RTS     PC
1032 006116  020257  020277  020057  20$:      .WORD   20257,20277,20057,20077,20117,20137,20157,20177
1033 006136  020217  020237                     .WORD   20217,20237
1034 006142  002000  004000  000004  30$:      .WORD   BIT10,BIT11,BIT2,BIT3,BIT4,BIT5,BIT6,BIT7,BIT8,BIT9
1035                                    ;
1036 006166  012720  000000C            EMXMNG: MOV     #EMXCHF!EMXMTV,(R0)+    ;ENTER CONTROL WORD
1037 006172  012720  100000                     MOV     #BIT15,(R0)+            ;ENTER NIB1
1038 006176  042703  000060                     BIC     #60,R3          ;ENTER NIB2 FROM TABLE
1039 006202  042704  000060                     BIC     #60,R4          ;R3=LOW, R4=HIGH
1040 006206  006303                             ASL     R3
1041 006210  006304                             ASL     R4
1042 006212  016404  006242'                    MOV     10$(R4),R4      ;GET INDEX INTO SECOND TABLE
1043 006216  060304                             ADD     R3,R4           ;GET VALUE FROM SECOND TABLE
1044 006220  011420                             MOV     (R4),(R0)+
1045 006222  020427  006266'                    CMP     R4,#9$          ;DID ENTRY COVER ALL VALUES?
1046 006226  001003                             BNE     20$             ;NO
1047 006230  052760  000000G 177772-            BIS     #EMXNFD,-6(R0)  ;YES: CONVERT TO NFLDC
1048 006236  000241                     20$:    CLC
1049 006240  000207                             RTS     PC
1050 006242  177777  006416'  006412'  10$:    .WORD   -1,1$,2$,3$,4$,5$,6$,7$,8$,9$
1051 006266  007774  005774  001774  9$:     .WORD   7774,5774,1774,1770,1760,1740,1700,1600,1400
1052 006310  006774  004774  000774  8$:     .WORD   6774,4774,774,770,760,740,700,600
1053 006330  006374  004374  000374  7$:     .WORD   6374,4374,374,370,360,340,300
1054 006346  006174  004174  000174  6$:     .WORD   6174,4174,174,170,160,140
1055 006362  006074  004074  000074  5$:     .WORD   6074,4074,74,70,60
1056 006374  006034  004034  000034  4$:     .WORD   6034,4034,34,30
1057 006404  006014  004014  000014  3$:     .WORD   6014,4014,14
1058 006412  006004  004004          2$:     .WORD   6004,4004
1059 006416  006000                  1$:     .WORD   6000
1060                                    ;
1061 006420  070527  000006             EMXNDG: MUL     #6,R5           ;CREATE NFLDC MERGE VECTOR
1062 006424  005405                             NEG     R5
1063 006426  066705  171400                     ADD     NFPTR,R5
1064 006432  010520                             MOV     R5,(R0)+
1065 006434  000207                             RTS     PC
1066                                    ;
1067 006436  016702  171352             NFGEN:  MOV     HSZ,R2          ;GET MAX #-NFLDC'S
1068 006442  005302                             DEC     R2
1069 006444  003002                             BGT     1$              ;IF #-NFLDC'S > 1
1070 006446  000261                             SEC                     ;IF NONE
1071 006450  000207                             RTS     PC
1072 006452  005767  171354             1$:     TST     NFPTR           ;IF PREVIOUSLY SET
```

```
1073 006456  001403                          BEQ     2$              ;   NOT SET
1074 006460  016767  171346  171346          MOV     NFPTR,NF1PTR    ;   SET: SAVE
1075 006466  012701  000000G         2$:     MOV     #NFEMX,R1       ;GENERATE NFLDC'S
1076 006472  012120                          MOV     (R1)+,(R0)+
1077 006474  012120                          MOV     (R1)+,(R0)+
1078 006476  012120                          MOV     (R1)+,(R0)+
1079 006500  077206                          SOB     R2,2$
1080 006502  010001                          MOV     R0,R1           ;SET NFLDC POINTER
1081 006504  162701  000000G                 SUB     #EMX,R1
1082 006510  052701  000000G                 BIS     #EMXNSO,R1
1083 006514  010167  171312                  MOV     R1,NFPTR
1084 006520  000241                          CLC
1085 006522  000207                          RTS     PC
1086                                 ;
1087         000001                          .END
```

```
ASKOVF= 000145          B.QSPL  000316      010 EMXNG   006056R     014 NF1PTR  000034R     014 RLOWD   005062R     014
BITVAL= 000000          B.QTTM  000076      010 EMXNSQ= ****** GX       NMAX    000004R     014 RMID    004406R     014
BIT0  = 000001          B.QUQP  000056      010 EMXSNG  006070R     014 NMRNGF= ****** GX       RNGCDE  000030RG    014
BIT1  = 000002          B.SFDB  000010      010 ERRCDE= ****** GX       NNN     000046R     014 SEGOP = ****** GX
BIT10 = 002000          B.SIZE  000772      010 ERROR = ****** GX       NOHTRM  000050R     014 SGNIND  000024R     014
BIT11 = 004000          B.SNDP  000012      010 EXPSMR  000022R     014 NUMRNG  001476RG    014 SLBOVF= 000164
BIT12 = 010000          B.SSQ   000004      010 EXPSPT  000010R     014 NUMVLU  000026R     014 SR.ARE  000114      002
BIT13 = 020000          B.SSQF  000050      010 EXPVPT  000006R     014 NVDEMX= ****** GX       SR.ARS  000106      002
BIT14 = 040000          B.STAT  000044      010 FALOVF= 000160          NVPTR   000040R     014 SR.DAY  000010      002
BIT15 = 100000          B.STTE  000053      010 FD.FID  000000      003 N.BFAC= 000004          SR.DLT  000014      002
BIT2  = 000004          B.UDOC  000110      010 FD.FNB  000006      003 N.BHGH= 000006          SR.ECB  000047      002
BIT3  = 000010          CF.B0 = 000070          FD.FVR  000004      003 N.BTCH= 000004          SR.ECH  000046      002
BIT4  = 000020          CF.B2 = 000067          FD.LEN  000010      003 N.BUFB= 004000          SR.ECL  000050      002
BIT5  = 000040          CF.B4 = 000066          FLUTBL= ****** GX       N.BUFW= 002000          SR.FIB  000012      002
BIT6  = 000100          CF.B6 = 000065          FLUTYP= ****** GX       N.FOS = 000764          SR.GRE  000100      002
BIT7  = 000200          CF.DR0= 000064          FN.DBR  000026      011 N.PKSZ= 000020          SR.GRS  000072      002
BIT8  = 000400          CF.DR1= 000063          FN.DBS  000022      011 N.PKTS= 000043          SR.LEN  000122      002
BIT9  = 001000          CHKRNG  001342R     014 FN.DHR  000040      011 N.QURY= 000031          SR.LIN  000066      002
BS.CLS= 000002          CM    = 000040          FN.EMA  000012      011 N.SUNT= 000002          SR.LIP  000062      002
BS.DBU= 000004          CSTDP = ****** GX       FN.EMB  000014      011 PD    = 000020 G        SR.MDN  000006      002
BS.INA= 000000          CSTDW = ****** GX       FN.EMC  000016      011 PNBRNG  000054RG    014 SR.NDC  000042      002
BS.OPN= 000001          CSTER = ****** GX       FN.FSA  000000      011 POLOVF= 000157          SR.NDS  000036      002
BS.SRC= 000003          CSTESC= ****** GX       FN.FSB  000002      011 QBFOVF= 000150          SR.NIN  000030      002
BYTE0 = 000000          CSTFDC= ****** GX       FN.FSC  000004      011 QEXOVF= 000165          SR.NIP  000022      002
BYTE1 = 000001          CSTMIN= ****** GX       FN.LGO  000034      011 QE.DW1= 000001          SR.SDB  000032      002
BYTE2 = 000002          CSTNF = ****** GX       FN.LGU  000036      011 QE.DW2= 000003          SR.SRC  000002      002
BYTE3 = 000003          CSTNUM= ****** GX       FN.MFO  000024      011 QE.FTB= 000026          SR.SUN  000000      002
BYTE4 = 000004          CSTSO = ****** GX       FN.MHR  000010      011 QE.IHS= 000005          SR.TWS  000056      002
BYTE5 = 000005          CSTTS = ****** GX       FN.NMB  000044      011 QE.ILC= 000004          SR.WSL  000052      002
BYTE6 = 000006          CSTVDC= ****** GX       FN.QLS  000006      011 QE.INO= 000015          SR.YR   000004      002
BYTE7 = 000007          CSTZER= ****** GX       FN.QRY  000020      011 QE.ISO= 000016          SR.1IN  000024      002
BYTE8 = 000010          DBSLEN= 000116          FN.SF0  000030      011 QE.MFL= 000013          SR.1IP  000016      002
BYTE9 = 000011          DECFLG  000022R     014 FN.SF1  000032      011 QE.MFM= 000010          SS.FID  000002      004
BYTVAL= 000012          DECPT = ****** GX       FN.SHD  000042      011 QE.MNT= 000012          SS.FNB  000010      004
B.BSTA  000054      010 DH.BF0  000002      005 HBEG    000002R     014 QE.MOP= 000007          SS.FVR  000006      004
B.CNTX  000046      010 DH.BFI  000004      005 HSZ     000014RG    014 QE.MSD= 000011          SS.LEN  000012      004
B.CQUQ  000060      010 DH.CTL  000000      005 HSZD    000020R     014 QE.MTS= 000002          SS.STT  000000      004
B.FEMA  000132      010 DH.DMC  000010      005 INCLD = 000004          QE.NLQ= 000006          STEP  = ****** GX
B.FEMB  000142      010 DH.FLG  000006      005 INCLUD  000024RG    014 QE.NOS= 000025          ST.ASZ  000020      006
B.FEMC  000152      010 DNUM  = 000010 G        LBEG    000000R     014 QE.NRB= 000024          ST.BSZ  000024      006
B.FFSA  000202      010 DN.DCK  000000      013 LEXTB3= ****** GX       QE.NRI= 000023          ST.BTC  000000      006
B.FFSB  000212      010 DN.NTP  000004      013 LHCSZ   000026R     014 QE.NTK= 000027          ST.CSZ  000030      006
B.FFSC  000222      010 DN.NXT  000006      013 LNPOVF= 000146          QE.PX1= 000017          ST.HRL  000010      006
B.FMHR  000172      010 DN.ROT  000002      013 LSZ     000012RG    014 QE.PX2= 000020          ST.LEN  000044      006
B.FQLS  000162      010 DN.SIZ  000010      013 LSZD    000016R     014 QE.PX3= 000021          ST.QRY  000002      006
B.FSAZ  000100      010 DPEMX = ****** GX       M     = 000062          QE.PX4= 000022          ST.QSZ  000034      006
B.FSBZ  000102      010 DPTR    000036R     014 MINEMX= ****** GX       QE.RO1= 000144          ST.SCH  000040      006
B.FSCZ  000104      010 EMAOVF= 000166          MMM     000044R     014 QE.UBP= 000014          ST.UHL  000004      006
B.HBLK  000120      010 EMBOVF= 000167          MR    = 000100 G        QLBOVF= 000163          ST.XLT  000014      006
B.HDOC  000114      010 EMCOVF= 000170          MR1   = 000400          QRYOVF= 000156          SU.DBU= 000004
B.HRLP  000126      010 EMX   = ****** GX       MR2   = 000200          Q.FDSC  000004      007 SU.DON= 000006
B.HRLR  000122      010 EMXCHF= ****** GX       MR3   = 000001          Q.NOBK  000000      007 SU.IDL= 000000
B.HRLW  000124      010 EMXEXF= ****** GX       N     = 000002          Q.NUHL  000002      007 SU.LOD= 000001
B.NMBR  000052      010 EMXMNG  006166R     014 NDBOVF= 000175          Q.SIZE  000014      007 SU.SRC= 000002
B.NQRY  000232      010 EMXMTV= ****** GX       NFEMX = ****** GX       RHIGH   004462R     014 SU.SRR= 000005
B.QLSZ  000106      010 EMXNDG  006420R     014 NFGEN   006436R     014 RHIGHD  005476R     014 SU.XPD= 000003
B.QMAP  000234      010 EMXNFD= ****** GX       NFPTR   000032R     014 RLOW    003760R     014 S.HRL = 000240
```

```
.MAIN.  MAC[illegible]M1110  27-MAR-80 13:04  PAGE 04-7
SYMBOL TABLE

TDBOVF=·000174          WN.NTP  000004      012·WORD3 =·000006      XDBLOA=·000004      XQTS  =·000003
TSKOVF=·000147          WN.NXT  000006      012·WORD4 =·000010      XDBPRO=·000012·     XQT0  =·000001
TTBOVF=·000161          WN.ROT  000002·     012·WORD5 =·000012·     XDMCIN=·000006      XSULOA=·000005
USER3 =·000002·         WN.SIZ  000010      012·WORD6 =·000014      XFOSMR=·000007      XTBOVF=·000162
VI·   =·****** GX·      WN.SRC  000000      012·WORD7 =·000016      XGTSRE=·000014      ZERO    000052R·    014
VINXT·=·****** GX·      WN.TYP  000001      012·WORD8 =·000020      XHITSK=·000011      ZPTR    000042R·    014
VI1OVF=·000173          WORD0 =·000000          WORD9 =·000022·     XHLMER=·000002·     ZVDEMX=·****** GX·
VI2OVF=·000172·         WORD1 =·000002·         WRDVAL=·000024      XHOTSK=·000010      ZVLDC·=·****** GX·
VI3OVF=·000171          WORD2·=·000004          XBATCH=·000013      XMSCHE=·000000

. ABS.  000000    000
        000000    001
SRCOFF· 000122·   002·
FDSCOF· 000010    003
SUSOFF· 000012·   004
DHROFF· 000012·   005
STTOFF· 000044    006
QSPLOF· 000014    007
BSTOFF· 000772·   010
FNOFFS· 000044    011
WNODOF· 000010    012·
DNODOF· 000010    013
NUMRNG· 006524    014
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  6856 WORDS· (·27 PAGES)
DYNAMIC MEMORY:  8084 WORDS· ( 31 PAGES)
ELAPSED TIME:  00:02:32·
NUMRNG,NUMRNG/-SP/NL:ME:8EX=[20,1]P,M,E,MDQT0.NUMRNG·
```

TABLE OF CONTENTS

```
                                ;THIS FILE (E.MAC) CONTAINS ALL THE QUERY ERROR CODES
                                ;
000001                          QE.DW1=1        ;RESERVED
000002                          QE.MTS=2        ;MULTIPLE TERM SEP'S OR MISSING TERM
000003                          QE.DW2=3        ;RESERVED
000004                          QE.ILC=4        ;ILLEGAL CHARACTER
000005                          QE.IHS=5        ;INTERNAL HSTS ERROR, LOGIC PROCESSING
000006                          QE.NLQ=6        ;NULL QUERY
000007                          QE.MOP=7        ;MISSING OPERATOR
000010                          QE.MFM=8.       ;MISPLACED FLU-MOD
000011                          QE.MSD=9.       ;MISPLACED SUBDOC OPERATOR
000012                          QE.MNT=10.      ;MISPLACED NOT OPERATOR
000013                          QE.MFL=11.      ;MISSING FLU / TWO CONSECUTIVE OPERATORS
000014                          QE.UBP=12.      ;UNBALANCED PARENTHESES
000015                          QE.INO=13.      ;INSUFFICIENT NUMBER OF OPERANDS
000016                          QE.ISO=14.      ;ILLEGAL SUBELEMENT OPERAND
000017                          QE.PX1=15.      ;MISSING PROX, WINDOW, OR DOC TYPE,ZONE, OR SUBZONE
000020                          QE.PX2=16.      ;PROX, WINDOW OR FLU-MOD ID TOO LARGE
000021                          QE.PX3=17.      ;MISSING PROX, UNIT
000022                          QE.PX4=18.      ;MISSING PROX, DELIMITER
000023                          QE.NRI=19.      ;ILLEGAL NUMERICAL RANGE SPECIFICATION
000024                          QE.NRB=20.      ;NUMERICAL RANGE BORDERED BY NUMERIC
000025                          QE.NOS=21.      ;NO STX FOUND IN QUERY
000026                          QE.FTB=22.      ;FLU TOO BIG
000027                          QE.NTK=23.      ;UNDEFINED TOKEN
                                ;
                                ;QE.RO1=100.    ;GENERAL RESOURCE OVERFLOW (LABEL DEFINED IN M.MAC)
000145                          ASKOVF=101.     ;ARGUMENT STACK OVERFLOW
000146                          LNPOVF=102.     ;LOGIC NODE POOL OVERFLOW
000147                          TSKOVF=103.     ;TOKEN STACK OVERFLOW
000150                          QBFOVF=104.     ;QUERY TOO BIG
000156                          QRYOVF=110.     ;# QUERIES OVERFLOWED
000157                          POLOVF=111.     ;FLU POOL OVERFLOW
000160                          FALOVF=112.     ;FAL
000161                          TTBOVF=113.     ;TTABLE
000162                          XTBOVF=114.     ;XTABLE
000163                          QLBOVF=115.     ;QLB
000164                          SLBOVF=116.     ;SDLB
000165                          QEXOVF=117.     ;QEX
000166                          EMAOVF=118.     ;EMA
000167                          EMBOVF=119.     ;EMB
000170                          EMCOVF=120.     ;EMC
000171                          VI3OVF=121.     ;VI QT3
000172                          VI2OVF=122.     ;VI QT2
000173                          VI1OVF=123.     ;VI QT1
000174                          TDBOVF=124.     ;TDCTB
000175                          NDBOVF=125.     ;NODE POOL QT2
                                ;
```

```
                                ;
                                ; MAXIMUM QLS SUB-BUFFER SIZES
                                ;
000031                          QRYMSZ==N.QURY
000500                          FALMSZ==160.*2
003270                          TTBMSZ==430.*4
002010                          XTBMSZ==516.*2
000516                          QLBMSZ==334.
000354                          SLBMSZ==236.
000600                          QEXMSZ==192.*2
                                ;
                                ; QUERY RESOLVER BIT DEFINITIONS
                                ;
000200                          B$SUC==BIT7     ;SUCCESS BIT
000100                          B$NOT==BIT6     ;QLB NOT ENTRY
000040                          B$MUL==BIT5     ;QLB MULTI-NOT ENTRY
000020                          B$FMN==BIT4     ;QLB FIRST MULTI-NOT ENTRY
                                ;
020000                          B$FST==BIT13    ;QEX FIRST PROX ENTRY
100000                          B$PUFH==BIT15   ;QEX FORWARD PROX UNIT - HIGH ORDER BIT
040000                          B$PUFL==BIT14   ;QEX FORWARD PROX UNIT - LOW ORDER BIT
010000                          B$PUBH==BIT12   ;QEX BACKWARD PROX UNIT - HIGH ORDER BIT
004000                          B$PUBL==BIT11   ;QEX BACKWARD PROX UNIT - LOW ORDER BIT
                                ;
                                ; QLS BUFFER OFFSETS (CONVERTED TO ADDRESSES)
                                ;
000000                          QLSXX1=0                ;OFFSET OF COMM HEADER
000004                          QLSX17=QLSXX1+4         ;BATCH NUMBER
000006                          QLSXX2=QLSX17+2         ;BYTE SIZE OF QLB (USED)
000010                          QLSXX3=QLSXX2+2         ;BYTE SIZE OF SDLB (USED)
000012                          QLSXX4=QLSXX3+2         ;BYTE SIZE OF QEX (USED)
000014                          QLSXX5=QLSXX4+2         ;MAXIMUM QUERY ID
000016                          QLSXX6=QLSXX5+2         ;BYTE SIZE OF FAL (USED)
000020                          QLSXX7=QLSXX6+2         ;BYTE SIZE OF TTABLE (USED)
000022                          QLSXX8=QLSXX7+2         ;BYTE SIZE OF XTABLE (USED)
000024                          QLSX10=QLSXX8+2         ;START OF QCL SUB-BUFFER
000106                          QLSX11=QLSX10+<QRYMSZ*2>;START OF FAL SUB-BUFFER
000606                          QLSX12=QLSX11+FALMSZ    ;START OF TTABLE SUB-BUFFER
004076                          QLSX13=QLSX12+TTBMSZ    ;START OF XTABLE SUB-BUFFER
006106                          QLSX14=QLSX13+XTBMSZ    ;START OF QLB SUB-BUFFER
006624                          QLSX15=QLSX14+QLBMSZ    ;START OF SDLB SUB-BUFFER
007200                          QLSX16=QLSX15+SLBMSZ    ;START OF QEX SUB-BUFFER
010000                          QLSIZ==QLSX16+QEXMSZ    ;SIZE OF QLS IN BYTES
```

```
                                        .SBTTL  QLSDAT
000000                                  .PSECT  AAAQLS
                                ;IMMEDIATE DEFINITIONS
                                ;
        177710                  QEXWAL=<B$PUFH!B$PUFL!B$PUBL>/400
        177720                  QEXSAL=<B$PUFH!B$PUFL!B$PUBH>/400
        177730                  QEXPAL=<B$PUFH!B$PUFL!B$PUBH!B$PUBL>/400
                                ;
        100000                  QEXFW=100000
        100001                  QEXFS=100001
        100002                  QEXFP=100002
                                ;
        074000                  QEXFWV=-B$PUFL!B$FST!B$PUBH!B$PUBL
        134000                  QEXFSV=-B$PUFH!B$FST!B$PUBH!B$PUBL
        174000                  QEXFPV=-B$PUFH!B$PUFL!B$FST!B$PUBH!B$PUBL
                                ;
                                ; FLAGS AND COUNTERS
                                ;
000000  000000                  PASSI:: .WORD   0       ;PASS INDICATOR (-1,0,1=PASS 1,2,3 RESPECTFULLY)
000002  000000                  NOTFLG::.WORD   0       ;NOT FLAG (NOT IN ABOVE HIEARCHY)
000004  000000                  FMNFLG::.WORD   0       ;FIRST NOT OF A MULTIPLE NOT INDICATOR
000006  000000                  SDFLG:: .WORD   0       ;SUB-DOCUMENT LOGIC INDICATOR
000010  000000G                 FPXPT:: .WORD   FPXSTK  ;1ST PROX LINK STACK POINTER
000012  000000G                 SPXPT:: .WORD   SPXSTK  ;2ND PROX LINK STACK POINTER
000014                          PRXFLG::                ;PROX INDICATOR
000014      000                 FPXCT:: .BYTE   0       ;COUNT OF ENTRIES ON 1ST STACK
000015      000                 SPXCT:: .BYTE   0       ;COUNT OF ENTRIES ON 2ND STACK
                                ;
                                ; POINTERS FOR QLS BUILDING AND INTERNAL LINKING
                                ;
000016  000000                  COLBS:: .WORD   0       ;START OF CURRENT QLB
000020  177776G                 NQLBE:: .WORD   QLB-2   ;NEXT ENTRY TO BE GENERATED IN QLB
000022  000000                  SDLBS:: .WORD   0       ;START OF CURRENT SDLB
000024  177775G                 SDLBE:: .WORD   SDLB-3  ;NEXT ENTRY TO BE GENERATED IN SDLB
000026  000000G                 NAQEX:: .WORD   QEX     ;NEXT ALLOCATION ADDRESS FOR QEX
000030  000000G                 CFALS:: .WORD   FAL     ;CURRENT FAL START
000032  000002G                 NFALE:: .WORD   FAL+2   ;NEXT ENTRY TO BE GENERATED IN FAL
                                ;
                                ; QLS SYNTHESIS SUBROUTINES TABLE
                                ;
000034  000466' 000466' 000466' CDETBL: PERROR,PERROR,PERROR      ;TOKFLU
000042  000614' 000702' 000466'         P1SUB,P2SUB,PERROR      ;TOKSBD
000050  000466' 000466' 000466'         PERROR,PERROR,PERROR    ;TOKFMD
000056  000562' 000602' 000466'         P1NOT,P2NOT,PERROR      ;TOKNOT
000064  000762' 000772' 000756'         P1OR,P2OR,P3OR          ;TOKOR
000072  000762' 000550' 000542'         P1OR,P2HOR,P3AND        ;TOKHOR
000100  000466' 000466' 000466'         PERROR,PERROR,PERROR    ;?
000106  000500' 000534' 000542'         P1AND,P2AND,P3AND       ;TOKAND
000114  000466' 000466' 000466'         PERROR,PERROR,PERROR    ;?
000122  001014' 001120' 001154'         P1PRX,P2PRX,P3PRX       ;TOKPXW
000130  000466' 000466' 000466'         PERROR,PERROR,PERROR    ;?
000136  001014' 001120' 001154'         P1PRX,P2PRX,P3PRX       ;TOKPXS
000144  000466' 000466' 000466'         PERROR,PERROR,PERROR    ;TOKLP
000152  001014' 001120' 001154'         P1PRX,P2PRX,P3PRX       ;TOKPXP
```

```
                                        .SBTTL  QLSCDE
000000                                  .PSECT  QLSCDE
                                ;
                                ; QLS SYNTHESIS LOGIC
                                ;
000000                          GENQLS::                        ;SUBROUTINE TO GENERATE QLS
000000  012767  177777  000000'          MOV     #-1,PASSI       ;PASS INDICATOR=-PASS 1
000006  012704  000000G                 MOV     #TOPSTK,R4      ;RESET TO TOP OF HIEARCHY STACK
000012  005044                          CLR     -(R4)           ;PUT NULL ON TOP OF STACK
000014  005067  000014'                 CLR     PRXFLG          ;RESET PROX COUNTERS
000020  012767  000000G 000010'         MOV     #FPXSTK,FPXPT   ;INITIALIZE PROX STACKS
000026  012767  000000G 000012'         MOV     #SPXSTK,SPXPT
000034  016705  000000G                 MOV     QLBAD,R5        ;R5 <-- CURRENT LOGIC OFFSET
000040  004767  001140                  JSR     PC,NEWQLB       ;CREATE NEW QLB
000044  005702                  BQLS1:  TST     R2              ;R2 = FLU OR NODE ADDRESS
000046  100151                          BPL     BQLS2           ; IT WAS A LOGIC NODE
000050  042702  100000                  BIC     #BIT15,R2       ; IT WAS A FLU: REMOVE FLU INDICATOR
000054  006302                          ASL     R2              ;FIND TTABLE ENTRY
000056  006302                          ASL     R2
000060  062702  000000G                 ADD     #TTABLE,R2
000064  016703  000000G                 MOV     NXTNDE,R3       ;GET NEXT AVAILABLE NODE FOR LINK
000070  062767  000004  000000G         ADD     #4,NXTNDE       ;ALLOCATE THE LINK
000076  005262  000002                  INC     2(R2)           ;STEP FLU LINK COUNTER
000102  005767  000014'                 TST     PRXFLG          ;FLU BELOW PROX?
000106  001402                          BEQ     1$              ;NO
000110  005262  000002                  INC     2(R2)           ;ALLOCATE ANOTHER LINK
000114  005712                  1$:     TST     (R2)            ;FIND LAST LINK
000116  001402                          BEQ     2$              ;FOUND
000120  011202                          MOV     (R2),R2         ;NOT FOUND, TRY NEXT
000122  000774                          BR      1$
000124  010312                  2$:     MOV     R3,(R2)         ;APPEND NEW LINK
000126  005767  000014'                 TST     PRXFLG          ;IS FLU BELOW A PROX?
000132  001513                          BEQ     4$              ;NO
000134  016723  000000G                 MOV     NXTNDE,(R3)+    ;CREATE TWO LINKS
000140  062767  000004  000000G         ADD     #4,NXTNDE
000146  017701  000010'                 MOV     @FPXPT,R1       ;GET FIRST PROX ADDRESS OFF STACK
000152  016702  000012'                 MOV     SPXPT,R2        ;GET SECOND
000156  016202  177776                  MOV     -2(R2),R2
000162  105767  000014'                 TSTB    FPXCT           ;IST THERE A FIRST PROX ELEMENT?
000166  001452                          BEQ     20$             ;NO--POINT TO DUMMY
000170  010123                          MOV     R1,(R3)+        ;YES-SET 1ST LINK TO IT
000172  005023                  3$:     CLR     (R3)+           ;2ND LINK IS CURRENTLY THE LAST, END CHAIN
000174  105767  000015'                 TSTB    SPXCT           ;IS THERE A 2ND PROX ELEMENT?
000200  001471                          BEQ     5$              ;NO--POINT TO QLB OR SDLB
000202  010223                          MOV     R2,(R3)+        ;YES-SET 2ND LINK TO IT
000204  105767  000014'                 TSTB    FPXCT           ;WAS THERE A 1ST LINK?
000210  001437                          BEQ     30$             ;NO--SKIP PREDICTIVE ROUTINE
000212  006302                          ASL     R2              ;YES-R2<--UNIT OF SECOND LINK
000214  016202  000000G                 MOV     QEX(R2),R2
000220  072227  000003                  ASH     #3,R2
000224  042702  037777                  BIC     #-<B$PUFH+B$PUFL+1>,R2
000230  006301                          ASL     R1              ;R1<--ADDRESS OF 1ST LINK QEX ELEMENT
000232  062701  000000G                 ADD     #QEX,R1
000236  032711  020000                  BIT     #B$FST,(R1)     ;IGNORE FLAG SET?
000242                                  BON     30$             ;YES
000244  032711  002000                  BIT     #BIT10,(R1)     ;HAS PREDICTIVE UNIT BEEN SET?
000250                                  BON     31$             ;YES
```

```
     58 000252  042711  140000                    BIC     #B$PUFH!B$PUFL,(R1)     ;NO-THEN SET UNIT
     59 000256  050211                            BIS     R2,(R1)
     60 000260  052711  002000                    BIS     #BIT10,(R1)     ;FLAG AS SET
     61 000264  000411                            BR      30$
     62 000266  011100                    31$:    MOV     (R1),R0         ;R0<--PREDICTIVE UNIT OF 1ST LINK
     63 000270  042700  037777                    BIC     #-<B$PUFH+B$PUFL+1>,R0
     64 000274  020200                            CMP     R2,R0           ;UNITS MATCH?
     65 000276  001404                            BEQ     30$             ;YES
     66 000300  042711  140000                    BIC     #B$PUFH!B$PUFL,(R1)     ;NO--SET UNIT AS UNPREDICTABLE
     67 000304  052711  020000                    BIS     #B$FST,(R1)     ;DON'T DO FURTHER UNIT CHECKING
     68 000310  012402                    30$:    MOV     (R4)+,R2        ;START UPWARD PASS
     69 000312  000444                            BR      BQLS3
     70 000314  010201                    20$:    MOV     R2,R1           ;R1<--ADDRESS OF 2ND LINK QEX ELEMENT
     71 000316  006301                            ASL     R1
     72 000320  062701  000000G                   ADD     #QEX,R1
     73 000324  032711  010000                    BIT     #B$PUBH,(R1)    ;IS UNIT=WORDS?
     74 000330                                    BOFF    21$             ;YES
     75 000332  032711  004000                    BIT     #B$PUBL,(R1)    ;NO--IS UNIT=SENTENCES?
     76 000336                                    BOFF    22$             ;YES
     77 000340  012723  100002                    MOV     #QEXFP,(R3)+    ;NO--SET TO PARAGRAPH DUMMY ENTRY
     78 000344  000712                            BR      3$
     79 000346  012723  100000            21$:    MOV     #QEXFW,(R3)+    ;SET TO WORD DUMMY ENTRY
     80 000352  000707                            BR      3$
     81 000354  012723  100001            22$:    MOV     #QEXFS,(R3)+    ;SET TO SENTENCE DUMMY ENTRY
     82 000360  000704                            BR      3$
     83 000362  005023                    4$:     CLR     (R3)+
     84 000364  010513                    5$:     MOV     R5,(R3)
     85 000366  012402                            MOV     (R4)+,R2        ;START UPWARD PASS
     86 000370  000415                            BR      BQLS3
     87 000372  116201  000000G           BQLS2:  MOVB    QNOPCD(R2),R1   ; IT WAS A LOGIC NODE
     88 000376  070127  000006                    MUL     #6,R1           ;CALCULATE SYNTHESIS SUBROUTINE ADDRESS
     89 000402  016700  000000'                   MOV     PASSI,R0
     90 000406  006300                            ASL     R0
     91 000410  060100                            ADD     R1,R0
     92 000412  004770  000036'                   JSR     PC,@CDETBL+2(R0)        ;EXECUTE SYNTHESIS SUBROUTINE
     93 000416  005767  000000'                   TST     PASSI           ;WHICH PASS?
     94 000422  002610                            BLT     BQLS1           ;PASS 1
     95 000424  005702                    BQLS3:  TST     R2              ;PASS 2 OR 3: WHAT WAS PREVIOUS PASS
     96 000426  001412                            BEQ     ITSOVR          ;NUL: ITS OVER, QLS HAS BEEN GENERATED
     97 000430  100403                            BMI     BQLS4           ;PASS 2
     98 000432  005067  000000'                   CLR     PASSI           ;PASS 1: SO SET AS PASS 2
     99 000436  000755                            BR      BQLS2
    100 000440  012767  000001  000000' BQLS4:    MOV     #1,PASSI        ;SET AS PASS 3
    101 000446  042702  100000                    BIC     #BIT15,R2       ;REMOVE PASS INDICATED
    102 000452  000747                            BR      BQLS2
    103 000454  004767  000652           ITSOVR:  JSR     PC,ENDQLB       ;END QLB IF NOT ALREADY ENDED
    104 000460  010567  000000G                   MOV     R5,QLBAD        ;SAVE LAST QLB OFFSET ALLOCATED
    105 000464  000207                            RTS     PC
    106 000466  012767  000005  000000G PERROR:   MOV     #QE.IHS,ERRCDE  ;NOTE ERROR
    107 000474  000167  000000G                   JMP     ERRORF
    108                                  ;
    109                                  ; SYNTHESIS SUBROUTINES
    110                                  ;
    111 000500  005767  000002'          P1AND:  TST     NOTFLG          ;NOT PRESENT?
    112 000504  001526                           BEQ     P1OR            ;NO
    113 000506  005767  000004'                  TST     FMWFLG          ;FIRST MULTI-NOT?
    114 000512  001404                           BEQ     1$              ;NO
```

```
115 000514  112777  000060  000020'          MOVB    #B$MUL!B$FMN,@NQLBE     ;YES: BUILD QLB ENTRY
116 000522  000517                            BR      P1OR
117 000524  112777  000040  000020' 1$:      MOVB    #B$MUL,@NQLBE           ;BUILD QLB ENTRY
118 000532  000513                            BR      P1OR
119 000534  004767  000534          P2AND:    JSR     PC,NXTQLB       ;ALLOCATE ANOTHER QLB ENTRY
120 000540  000514                            BR      P2OR
121 000542  004767  000564          P3AND:    JSR     PC,ENDQLB       ;END QLB
122 000546  000503                            BR      P3OR
123 000550  004767  000556          P2HOR:    JSR     PC,ENDQLB       ;END CURRENT QLB
124 000554  004767  000424                    JSR     PC,NEWQLB       ;ALLOCATE ANOTHER QLB ENTRY
125 000560  000504                            BR      P2OR
126 000562  005267  000002'          P1NOT:    INC     NOTFLG          ;FLAG NOT OCCURRENCE
127 000566  112777  000100  000020'          MOVB    #B$NOT,@NQLBE   ;BUILD QLB ENTRY
128 000574  005267  000004'                    INC     FMNFLG          ;1ST MULTI-NOT INDICATION
129 000600  000470                            BR      P1OR
130 000602  005067  000002'          P2NOT:    CLR     NOTFLG          ;END NOT'S ENFLUENCE
131 000606  005267  000004'                    INC     FMNFLG          ;RESET 1ST MULTI-NOT INDICATOR
132 000612  000461                            BR      P3OR
133 000614  005267  000006'          P1SUB:    INC     SDFLG           ;FLAG SUB-DOC LOGIC OCCURENCE
134 000620  016744  000016'                    MOV     CQLBS,-(R4)     ;SAVE QLB POINTERS
135 000624  016744  000020'                    MOV     NQLBE,-(R4)
136 000630  016744  000002'                    MOV     NOTFLG,-(R4)
137 000634  016744  000004'                    MOV     FMNFLG,-(R4)
138 000640  005067  000002'                    CLR     NOTFLG
139 000644  005067  000004'                    CLR     FMNFLG          ;SAVE AND RESET NOT STATUS
140 000650  010567  000000G                   MOV     R5,QLBAD
141 000654  016767  000022' 000016'          MOV     SDLBS,CQLBS     ;RESET TO SUB-DOC POINTERS
142 000662  016767  000024' 000020'          MOV     SDLBE,NQLBE
143 000670  016705  000000G                   MOV     SDLBAD,R5
144 000674  004767  000304                    JSR     PC,NEWQLB       ;ALLOCATE NEW QLB
145 000700  000430                            BR      P1OR
146 000702  004767  000424          P2SUB:    JSR     PC,ENDQLB       ;END SDLB
147 000706  010567  000000G                   MOV     R5,SDLBAD       ;SAVE CURRENT SDLB POINTERS
148 000712  016767  000020' 000024'          MOV     NQLBE,SDLBE
149 000720  016767  000016' 000022'          MOV     CQLBS,SDLBS
150 000726  012467  000004'                    MOV     (R4)+,FMNFLG    ;RESTORE NOT STATUS
151 000732  012467  000002'                    MOV     (R4)+,NOTFLG
152 000736  012467  000020'                    MOV     (R4)+,NQLBE     ;RESTORE OLD QLB POINTERS
153 000742  012467  000016'                    MOV     (R4)+,CQLBS
154 000746  016705  000000G                   MOV     QLBAD,R5
155 000752  005067  000006'                    CLR     SDFLG           ;END SUB-DOC ENFLUENCE
156                                 ;         BR      P3OR
157 000756  012402                  P3OR:     MOV     (R4)+,R2        ;START UPWARD PASS
158 000760  000207                            RTS     PC
159 000762  010244                  P1OR:     MOV     R2,-(R4)        ;STACK CURRENT NODE
160 000764  016202  000000G                   MOV     QNARG1(R2),R2   ;EXAMINE LEFT ARGUMENT
161 000770  000207                            RTS     PC
162 000772  010244                  P2OR:     MOV     R2,-(R4)        ;STACK CURRENT NODE
163 000774  052714  100000                    BIS     #BIT15,(R4)     ;FLAG AS PASS 2
164 001000  016202  000000G                   MOV     QNARG2(R2),R2   ;EXAMINE RIGHT ARGUMENT
165 001004  012767  177777  000000'          MOV     #-1,PASSI       ;SET PASS AS PASS 1
166 001012  000207                            RTS     PC
167 001014  016701  000012'          P1PRX:    MOV     SPXPT,R1        ;POP NEXT QEX ADDRESS ON 2ND PROX STACK
168 001020  016721  000000G                   MOV     QEXAD,(R1)+
169 001024  010167  000012'                    MOV     R1,SPXPT
170 001030  016703  000026'                    MOV     NAQEX,R3        ;R3<--NEXT FREE QEX LOCATION
171 001034  062767  000002  000026'          ADD     #2,NAQEX        ;ALLOCATE QEX LOCATION
```

```
QLSCDE
                        ;
    172 001042  005267  000000G         INC     QEXAD
    173 001046  105267  000015'         INCB    SPXCT           ;INDICATE WORD ADDED TO 2ND STACK
    174 001052  011201                   MOV     (R2),R1         ;R1=OPERATOR FIELD OF LOGIC NODE
    175 001054  000301                   SWAB    R1              ;TRANSFER RANGE AS IS
    176 001056  110123                   MOVB    R1,(R3)+
    177 001060  000301                   SWAB    R1
    178 001062  122701  000000G         CMPB    #TOKPXW,R1      ;IS UNIT=WORDS?
    179 001066  001406                   BEQ     1$              ;YES
    180 001070  122701  000000G         CMPB    #TOKPXS,R1      ;NO--IS UNIT=SENTENCES?
    181 001074  001406                   BEQ     2$              ;YES
    182 001076  112723  177730         MOVB    #QEXPAL,(R3)+   ;NO--SET TO PARAGRAPH
    183 001102  000727                   BR      P1OR
    184 001104  112723  177710   1$:     MOVB    #QEXWAL,(R3)+   ;SET TO WORDS
    185 001110  000724                   BR      P1OR
    186 001112  112723  177720   2$:     MOVB    #QEXSAL,(R3)+   ;SET TO SENTENCES
    187 001116  000721                   BR      P1OR
    188 001120  162767  000002  000010' P2PRX:  SUB     #2,FPXPT        ;POP 2ND STACK QEX ADDRESS AND PUSH IT
    189 001126  162767  000002  000012'         SUB     #2,SPXPT        ; ON TO 1ST STACK
    190 001134  017777  000012' 000010'         MOV     @SPXPT,@FPXPT
    191 001142  105267  000014'                 INCB    FPXCT
    192 001146  105367  000015'                 DECB    SPXCT
    193 001152  000707                          BR      P2OR
    194 001154  017701  000010'   P3PRX:  MOV     @FPXPT,R1       ;CLEAR TEMPORARY FLAGS
    195 001160  006301                          ASL     R1
    196 001162  042761  022000  000000G         BIC     #BIT10!B$FST,QEX(R1)
    197 001170  062767  000002  000010'         ADD     #2,FPXPT        ;POP QEX ADDRESS OFF 1ST STACK
    198 001176  105367  000014'                 DECB    FPXCT
    199 001202  000665                          BR      P3OR
    200                         ;
    201 001204  062767  000002  000020' NEWQLB: ADD     #2,NQLBE        ;ALLOCATE NEW QLB
    202 001212  062705  000003                  ADD     #3,R5
    203 001216  005267  000004'                 INC     FMNFLG          ;RESET 1ST MULTI-NOT FLAG
    204 001222  005767  000006'                 TST     SDFLG           ;SUB-DOC LOGIC?
    205 001226  001412                          BEQ     1$              ;NO
    206 001230  005267  000020'                 INC     NQLBE           ;COMPENSATE FOR DOUBLE BYTE POINTER
    207 001234  005205                          INC     R5
    208 001236  032767  000001  000020'         BIT     #1,NQLBE        ;EVEN WORD?
    209 001244  001403                          BEQ     1$              ;YES
    210 001246  005267  000020'                 INC     NQLBE           ;NO: MAKE EVEN
    211 001252  005205                          INC     R5
    212 001254  016767  000020' 000016' 1$:     MOV     NQLBE,CQLBS     ;1ST BYTE OF QLB
    213 001262  005267  000020'                 INC     NQLBE           ;2ND BYTE OF QLB
    214 001266  105077  000020'                 CLRB    @NQLBE          ;FLAG AS AND
    215 001272  000207                          RTS     PC
    216 001274  005267  000020'   NXTQLB: INC     NQLBE           ;ALLOCATE ANOTHER QLB ENTRY
    217 001300  005205                          INC     R5
    218 001302  005067  000004'                 CLR     FMNFLG          ;CAN'T BE 1ST MULTI-NOT
    219 001306  005767  000002'                 TST     NOTFLG          ;NOT OCCURRED?
    220 001312  001404                          BEQ     1$              ;NO
    221 001314  112777  000040  000020'         MOVB    #B$MUL,@NQLBE   ;BUILD QLB ENTRY
    222 001322  000207                          RTS     PC
    223 001324  105077  000020'   1$:     CLRB    @NQLBE          ;BUILD QLB ENTRY
    224 001330  000207                          RTS     PC
    225 001332  016703  000016'   ENDQLB: MOV     CQLBS,R3        ;1ST BYTE OF QLB
    226 001336  016701  000020'             MOV     NQLBE,R1
    227 001342  160301                      SUB     R3,R1           ;CALCULATE SIZE OF QLB
    228 001344  110113                      MOVB    R1,(R3)         ;STORE SIZE
```

```
    229 001346  060103                        ADD     R1,R3           ;GET BACK TO END
    230 001350  005203                        INC     R3
    231 001352  005767  000006'               TST     SDFLG           ;SUB-DOC LOGIC?
    232 001356  001003                        BNE     1$              ;YES
    233 001360  116723  000000G               MOVB    QRYNO,(R3)+     ;LAST BYTE = QRY ID
    234 001364  000207                        RTS     PC
    235 001366  016701  000000G       1$:     MOV     QLBAD,R1        ;LAST BYTE = QLB POINTER
    236 001372  032703  000001                BIT     #1,R3           ;BUT ALIGN ON EVEN BYTE
    237 001376  001401                        BEQ     2$
    238 001400  005203                        INC     R3
    239 001402  010123                2$:     MOV     R1,(R3)+        ;TRANSFER QLB POINTER
    240 001404  000207                        RTS     PC
    241                                       ;
    242                                       ; SUBROUTINE TO GENERATE TTABLE AND XTABLE
    243                                       ;
    244 001406  016705  000000G       GENTX:: MOV     FLUID,R5        ;R5 <-- # ENTRIES IN TTABLE
    245 001412  005205                        INC     R5
    246 001414  012703  000000G               MOV     #TTABLE,R3      ;LOAD TABLE ADDRESSES
    247 001420  012704  000000G               MOV     #XTABLE,R4
    248 001424  005713                1$:     TST     (R3)            ;ANY LINKS FOR THIS FLU ID?
    249 001426  001412                        BEQ     3$              ;NO
    250 001430  011302                        MOV     (R3),R2         ;YES: GET LINK
    251 001432  010413                        MOV     R4,(R3)         ;NOW POINT TO XTABLE
    252 001434  162713  000000G               SUB     #XTABLE,(R3)
    253 001440  016224  000002        2$:     MOV     2(R2),(R4)+     ;TRANSFER LINK TO XTABLE
    254 001444  005712                        TST     (R2)            ;MORE LINKS?
    255 001446  001402                        BEQ     3$              ;NO
    256 001450  011202                        MOV     (R2),R2         ;GET NEXT LINK
    257 001452  000772                        BR      2$
    258 001454  062703  000004        3$:     ADD     #4,R3           ;STEP TO NEXT TTABLE ENTRY
    259 001460  077517                        SOB     R5,1$           ;FINISHED?
    260 001462  162703  000000G               SUB     #TTABLE,R3      ;SAVE TABLE END OFFSETS
    261 001466  162704  000000G               SUB     #XTABLE,R4
    262 001472  010367  000000G               MOV     R3,TTSIZ
    263 001476  010467  000000G               MOV     R4,XTSIZ
    264 001502  000207                        RTS     PC              ;YES
    265         000001                        .END
```

```
ASKOVF= 000145          B.FQLS  000162          010 FLUID= ****** GX          P2AND   000534R          015 QLSXX6= 000016
BITVAL= 000000          B.FSAZ  000100          010 FMNFLG 000004RG      014 P2HOR   000550R          015 QLSXX7= 000020
BIT0  = 000001          B.FSBZ  000102          010 FN.DBR 000026        011 P2NOT   000602R          015 QLSXX8= 000022
BIT1  = 000002          B.FSCZ  000104          010 FN.DBS 000022        011 P2OR    000772R          015 QLSX10= 000024
BIT10 = 002000          B.HBLK  000120          010 FN.DHR 000040        011 P2PRX   001120R          015 QLSX11= 000106
BIT11 = 004000          B.HDOC  000114          010 FN.EMA 000012        011 P2SUB   000702R          015 QLSX12= 000606
BIT12 = 010000          B.HRLP  000126          010 FN.EMB 000014        011 P3AND   000542R          015 QLSX13= 004076
BIT13 = 020000          B.HRLR  000122          010 FN.EMC 000016        011 P3OR    000756R          015 QLSX14= 006106
BIT14 = 040000          B.HRLW  000124          010 FN.FSA 000000        011 P3PRX   001154R          015 QLSX15= 006624
BIT15 = 100000          B.NMBR  000052          010 FN.FSB 000002        011 QBFOVF= 000150               QLSX16= 007200
BIT2  = 000004          B.NQRY  000232          010 FN.FSC 000004        011 QEX   = ****** GX            QLSX17= 000004
BIT3  = 000010          B.QLSZ  000106          010 FN.LGQ 000034        011 QEXAD = ****** GX            QNARG1= ****** GX
BIT4  = 000020          B.QMAP  000234          010 FN.LGU 000036        011 QEXFP = 100002               QNARG2= ****** GX
BIT5  = 000040          B.QSPL  000316          010 FN.MFO 000024        011 QEXFPV= 174000 G             QNOPCD= ****** GX
BIT6  = 000100          B.QTTM  000076          010 FN.MHR 000010        011 QEXFS = 100001               QRYMSZ= 000031 G
BIT7  = 000200          B.QUQP  000056          010 FN.NMB 000044        011 QEXFSV= 134000 G             QRYNO = ****** GX
BIT8  = 000400          B.SFDB  000010          010 FN.QLS 000006        011 QEXFW = 100000               QRYOVF= 000156
BIT9  = 001000          B.SIZE  000772          010 FN.QRY 000020        011 QEXFWV= 074000 G             Q.FDSC  000004      007
BQLS1   000044R     015 B.SNDP  000012          010 FN.SF0 000030        011 QEXMSZ= 000600 G             Q.NQBK  000000      007
BQLS2   000372R     015 B.SSQ   000004          010 FN.SFI 000032        011 QEXOVF= 000165               Q.NUHL  000002      007
BQLS3   000424R     015 B.SSQF  000050          010 FN.SHD 000042        011 QEXPAL= 177730               Q.SIZE  000014      007
BQLS4   000440R     015 B.STAT  000044          010 FPXCT  000014RG      014 QEXSAL= 177720               SDFLG   000006RG    014
BS.CLS= 000002          B.STTE  000053          010 FPXPT  000010RG      014 QEXWAL= 177710               SDLB  = ****** GX
BS.DBU= 000004          B.UDOC  000110          010 FPXSTK= ****** GX        QE.DWI= 000001               SDLBAD= ****** GX
BS.INA= 000000          CDETBL  000034R         014 GENQLS 000000RG      015 QE.DW2= 000003               SDLBE   000024RG    014
BS.OPN= 000001          CFALS   000030RG        014 GENTX  001406RG      015 QE.FTB= 000026               SDLBS   000022RG    014
BS.SRC= 000003          CF.B0 = 000070              ITSOVR 000454R       015 QE.IHS= 000005               SLBMSZ= 000354 G
BYTE0 = 000000          CF.B2 = 000067              LNPOVF= 000146           QE.ILC= 000004               SLBOVF= 000164
BYTE1 = 000001          CF.B4 = 000066              M     = 000062           QE.INO= 000015               SPXCT   000015RG    014
BYTE2 = 000002          CF.B6 = 000065              N     = 000002           QE.ISO= 000016               SPXPT   000012RG    014
BYTE3 = 000003          CF.DR0= 000064              NAQEX  000026RG      014 QE.MFL= 000013               SPXSTK= ****** GX
BYTE4 = 000004          CF.DR1= 000063              NDBOVF= 000175           QE.MFM= 000010               SR.ARE  000114      002
BYTE5 = 000005          COLBS   000016RG    014 NEWQLB 001204R       015 QE.MNT= 000012               SR.ARS  000106      002
BYTE6 = 000006          DBSLEN= 000116              NFALE  000032RG      014 QE.MOP= 000007               SR.DAY  000010      002
BYTE7 = 000007          DH.BF0  000002          005 NOTFLG 000002RG      014 QE.MSD= 000011               SR.DLT  000014      002
BYTE8 = 000010          DH.BF1  000004          005 NQLBE  000020RG      014 QE.MTS= 000002               SR.ECB  000047      002
BYTE9 = 000011          DH.CTL  000000          005 NXTNDE= ****** GX        QE.NLQ= 000006               SR.ECH  000046      002
BYTVAL= 000012          DH.DMC  000010          005 NXTQLB 001274R       015 QE.NOS= 000025               SR.ECL  000050      002
B$FMN = 000020 G        DH.FLG  000006          005 N.BFAC= 000004           QE.NRB= 000024               SR.FIB  000012      002
B$FST = 020000 G        DN.DCK  000000          013 N.BHGH= 000006           QE.NRI= 000023               SR.GRE  000100      002
B$MUL = 000040 G        DN.NTP  000004          013 N.BTCH= 000004           QE.NTK= 000027               SR.GRS  000072      002
B$NOT = 000100 G        DN.NXT  000006          013 N.BUFB= 004000           QE.PX1= 000017               SR.LEN  000122      002
B$PUBH= 010000 G        DN.ROT  000002          013 N.BUFW= 002000           QE.PX2= 000020               SR.LIN  000066      002
B$PUBL= 004000 G        DN.SIZ  000010          013 N.FOS = 000764           QE.PX3= 000021               SR.LIP  000062      002
B$PUFH= 100000 G        EMAOVF= 000166              N.PKSZ= 000020           QE.PX4= 000022               SR.MON  000006      002
B$PUFL= 040000 G        EMBOVF= 000167              N.PKTS= 000043           QE.RO1= 000144               SR.NDC  000042      002
B$SUC = 000200 G        EMCOVF= 000170              N.QURY= 000031           QE.UBP= 000014               SR.NDS  000036      002
B.BSTA  000054      010 ENDQLB  001332R     015 N.SUNT= 000002           QLB   = ****** GX              SR.NIN  000030      002
B.CNTX  000046      010 ERRCDE= ****** GX        PASSI  000000RG      014 QLBAD = ****** GX              SR.NIP  000022      002
B.COUQ  000060      010 ERRORF= ****** GX        PERROR 000466R       015 QLBMSZ= 000516 G               SR.SDB  000032      002
B.FEMA  000132      010 FAL   = ****** GX        POLOVF= 000157           QLBOVF= 000163               SR.SRC  000002      002
B.FEMB  000142      010 FALMSZ= 000500 G         PRXFLG 000014RG      014 QLSIZ = 010000 G             SR.SUN  000000      002
B.FEMC  000152      010 FALOVF= 000160           P1AND  000500R       015 QLSXX1= 000000               SR.TWS  000056      002
B.FFSA  000202      010 FD.FID  000000      003 P1NOT  000562R       015 QLSXX2= 000006               SR.WSL  000052      002
B.FFSB  000212      010 FD.FNB  000006      003 P1OR   000762R       015 QLSXX3= 000010               SR.YR   000004      002
B.FFSC  000222      010 FD.FVR  000004      003 P1PRX  001014R       015 QLSXX4= 000012               SR.1IN  000024      002
B.FMHR  000172      010 FD.LEN  000010      003 P1SUB  000514R       015 QLSXX5= 000014               SR.1IP  000016      002
```

```
.MAIN.  MAC(10)M1110  27-MAR-80 12:58  PAGE 17-6
SYMBOL TABLE

SS.FID  000002      004 ST.UHL  000004      006 TSKOVF=  000147          WORD0 =  000000          XDMCIN=  000006
SS.FNB  000010      004 ST.XLT  000014      006 TTABLE=  ****** GX       WORD1 =  000002          XFOSMR=  000007
SS.FVR  000006      004 SU.DBU=  000004         TTBMSZ=  003270 G        WORD2 =  000004          XGTSRE=  000014
SS.LEN  000012      004 SU.DON=  000006         TTBOVF=  000161          WORD3 =  000006          XHITSK=  000011
SS.STT  000000      004 SU.IDL=  000000         TTSIZ =  ****** GX       WORD4 =  000010          XHLMER=  000002
ST.ASZ  000020      006 SU.LOD=  000001         VI1OVF=  000173          WORD5 =  000012          XHOTSK=  000010
ST.BSZ  000024      006 SU.SRC=  000002         VI2OVF=  000172          WORD6 =  000014          XMSCHE=  000000
ST.BTC  000000      006 SU.SRR=  000005         VI3OVF=  000171          WORD7 =  000016          XQTS   =  000003
ST.CSZ  000030      006 SU.XPD=  000003         WN.NTP  000004      012 WORD8 =  000020          XQT0   =  000001
ST.HRL  000010      006 S.HRL =  000240         WN.NXT  000006      012 WORD9 =  000022          XSULOA=  000005
ST.LEN  000044      006 TDBOVF=  000174         WN.ROT  000002      012 WRDVAL=  000024          XTABLE=  ****** GX
ST.QRY  000002      006 TOKPXS=  ****** GX      WN.SIZ  000010      012 XBATCH=  000013          XTBMSZ=  002010 G
ST.QSZ  000034      006 TOKPXW=  ****** GX      WN.SRC  000000      012 XDBLOA=  000004          XTBOVF=  000162
ST.SCH  000040      006 TOPSTK=  ****** GX      WN.TYP  000001      012 XDBPRO=  000012          XTSIZ =  ****** GX

. ABS.  000000     000
        000000     001
SRCOFF  000122     002
FDSCOF  000010     003
SUSOFF  000012     004
DHROFF  000012     005
STTOFF  000044     006
QSPLOF  000014     007
BSTOFF  000772     010
FNOFFS  000044     011
WNODOF  000010     012
DNODOF  000010     013
AAAQLS  000160     014
QLSCDE  001504     015
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  2901 WORDS  ( 12 PAGES)
DYNAMIC MEMORY:  3860 WORDS  ( 14 PAGES)
ELAPSED TIME:  00:00:28
QLSCDE,QLSCDE/-SP/NL:ME:BEX=[20,1]P,M,E,Q,QLSDAT,QLSCDE
```

TABLE OF CONTENTS

```
                        ;THIS FILE (E.MAC) CONTAINS ALL THE QUERY ERROR CODES
                        ;
000001                  QE.DW1=1          ;RESERVED
000002                  QE.MTS=2          ;MULTIPLE TERM SEP'S OR MISSING TERM
000003                  QE.DW2=3          ;RESERVED
000004                  QE.ILC=4          ;ILLEGAL CHARACTER
000005                  QE.IHS=5          ;INTERNAL HSTS ERROR, LOGIC PROCESSING
000006                  QE.NLQ=6          ;NULL QUERY
000007                  QE.MOP=7          ;MISSING OPERATOR
000010                  QE.MFM=8.         ;MISPLACED FLU-MOD
000011                  QE.MSD=9.         ;MISPLACED SUBDOC OPERATOR
000012                  QE.MNT=10.        ;MISPLACED NOT OPERATOR
000013                  QE.MFL=11.        ;MISSING FLU / TWO CONSECUTIVE OPERATORS
000014                  QE.UBP=12.        ;UNBALANCED PARENTHESES
000015                  QE.INO=13.        ;INSUFFICIENT NUMBER OF OPERANDS
000016                  QE.ISO=14.        ;ILLEGAL SUBELEMENT OPERAND
000017                  QE.PX1=15.        ;MISSING PROX, WINDOW, OR DOC TYPE,ZONE, OR SUBZONE
000020                  QE.PX2=16.        ;PROX, WINDOW OR FLU-MOD ID TOO LARGE
000021                  QE.PX3=17.        ;MISSING PROX, UNIT
000022                  QE.PX4=18.        ;MISSING PROX, DELIMITER
000023                  QE.NRI=19.        ;ILLEGAL NUMERICAL RANGE SPECIFICATION
000024                  QE.NRB=20.        ;NUMERICAL RANGE BORDERED BY NUMERIC
000025                  QE.NOS=21.        ;NO STX FOUND IN QUERY
000026                  QE.FTB=22.        ;FLU TOO BIG
000027                  QE.NTK=23.        ;UNDEFINED TOKEN
                        ;
                        ;QE.ROI=100.      ;GENERAL RESOURCE OVERFLOW (LABEL DEFINED IN M.MAC)
000145                  ASKOVF=101.       ;ARGUMENT STACK OVERFLOW
000146                  LNPOVF=102.       ;LOGIC NODE POOL OVERFLOW
000147                  TSKOVF=103.       ;TOKEN STACK OVERFLOW
000150                  QBFOVF=104.       ;QUERY TOO BIG
000156                  QRYOVF=110.       ;# QUERIES OVERFLOWED
000157                  POLOVF=111.       ;FLU POOL OVERFLOW
000160                  FALOVF=112.       ;FAL
000161                  TTBOVF=113.       ;TTABLE
000162                  XTBOVF=114.       ;XTABLE
000163                  QLBOVF=115.       ;QLB
000164                  SLBOVF=116.       ;SDLB
000165                  QEXOVF=117.       ;QEX
000166                  EMAOVF=118.       ;EMA
000167                  EMBOVF=119.       ;EMB
000170                  EMCOVF=120.       ;EMC
000171                  VI3OVF=121.       ;VI QT3
000172                  VI2OVF=122.       ;VI QT2
000173                  VI1OVF=123.       ;VI QT1
000174                  TDBOVF=124.       ;TDCTB
000175                  NDBOVF=125.       ;NODE POOL QT2
                        ;
```

```
                        ;
                        ; MACRO TO PRINT BUFFER
                        ; CREATES SPOOL FILE RECORDS
                        ;
                        .MACRO  PRT,START,END
                                MOV     R4,-(SP)
                                MOV     START,R4
                                MOV     R4,LOCAT
                                MOV     END,ENDLOC
                                JSR     PC,PRINT
                                MOV     (SP)+,R4
                        .ENDM
                        .MACRO  PRTH,START,END
                                MOV     R4,-(SP)
                                MOV     START,R4
                                MOV     R4,LOCAT
                                MOV     END,ENDLOC
                                JSR     PC,PRNTH
                                MOV     (SP)+,R4
                        .ENDM
                        ;
                        ; MACRO TO TEST CHARACTER SIGNIFICANCE
                        ; PARAMETERS ARE CONDITION,TRUE,AND FALSE
                        ;       CONDITION=THE TEST CONDITION
                        ;       TRUE    =PATH TO TAKE IF CONDITION TRUE
                        ;       FALSE   =PATH TO TAKE IF CONDITION FALSE
                        ; ANY PARAMETER CAN BE PROCEEDED BY "!" WHICH CAUSES THAT
                        ;  PARAMETER TO BE A SUBROUTINE CALL
                        ;
```

```
                                        .SBTTL  @CCET
000000                                  .PSECT  CCET
                                ;   CHARACTER CODE EQUATE TABLE (CCET)
                                ;
                                ;  ASCII TO 6-BIT CONVERSION TABLE
                                ;
000000  200                     CET6B:: .BYTE   200     ;NULL-----0
000001  074                             .BYTE   74      ;SOH
000002  074                             .BYTE   74      ;STX
000003  074                             .BYTE   74      ;ETX
000004  074                             .BYTE   74      ;EOT
000005  074                             .BYTE   74      ;ENQ
000006  074                             .BYTE   74      ;ACK
000007  074                             .BYTE   74      ;BEL
000010  074                             .BYTE   74      ;BS-------10
000011  074                             .BYTE   74      ;HT
000012  005                             .BYTE   5       ;LF
000013  074                             .BYTE   74      ;VT
000014  074                             .BYTE   74      ;FF
000015  005                             .BYTE   5       ;CR
000016  074                             .BYTE   74      ;SO
000017  074                             .BYTE   74      ;SI
000020  074                             .BYTE   74      ;DLE------20
000021  201                             .BYTE   201     ;DC1 - PARAGRAPH
000022  202                             .BYTE   202     ;DC2 - ZONE
000023  203                             .BYTE   203     ;DC3 - SUBZONE
000024  204                             .BYTE   204     ;DC4 - SENTENCE
000025  074                             .BYTE   74      ;NAK
000026  074                             .BYTE   74      ;SYN
000027  074                             .BYTE   74      ;ETB
000030  074                             .BYTE   74      ;CAN------30
000031  074                             .BYTE   74      ;EM
000032  074                             .BYTE   74      ;SUB
000033  074                             .BYTE   74      ;ESC
000034  074                             .BYTE   74      ;FS
000035  074                             .BYTE   74      ;GS
000036  074                             .BYTE   74      ;RS
000037  074                             .BYTE   74      ;US
000040  005                             .BYTE   5       ;SPACE----40
000041  006                             .BYTE   6       ;!
000042  007                             .BYTE   7       ;"
000043  010                             .BYTE   10      ;#
000044  074                             .BYTE   74      ;$
000045  011                             .BYTE   11      ;%
000046  012                             .BYTE   12      ;&
000047  013                             .BYTE   13      ;'
000050  014                             .BYTE   14      ;(--------50
000051  015                             .BYTE   15      ;)
000052  016                             .BYTE   16      ;*
000053  017                             .BYTE   17      ;+
000054  020                             .BYTE   20      ;,
000055  021                             .BYTE   21      ;-
000056                          DECPT==.-CET6B
000056  022                             .BYTE   22      ;.
000057  023                             .BYTE   23      ;/
000060  024                             .BYTE   24      ;0--------60
000061  025                             .BYTE   25      ;1
```

```
     58 000062      026                 .BYTE   26      ;2
     59 000063      027                 .BYTE   27      ;3
     60 000064      030                 .BYTE   30      ;4
     61 000065      031                 .BYTE   31      ;5
     62 000066      032                 .BYTE   32      ;6
     63 000067      033                 .BYTE   33      ;7
     64 000070      034                 .BYTE   34      ;8--------70
     65 000071      035                 .BYTE   35      ;9
     66 000072      075                 .BYTE   75      ;:
     67 000073      076                 .BYTE   76      ;;
     68 000074      074                 .BYTE   74      ;<
     69 000075      036                 .BYTE   36      ;=
     70 000076      074                 .BYTE   74      ;>
     71 000077      077                 .BYTE   77      ;?
     72 000100      074                 .BYTE   74      ;@--------100
     73 000101      037                 .BYTE   37      ;A
     74 000102      040                 .BYTE   40      ;B
     75 000103      041                 .BYTE   41      ;C
     76 000104      042                 .BYTE   42      ;D
     77 000105      043                 .BYTE   43      ;E
     78 000106      044                 .BYTE   44      ;F
     79 000107      045                 .BYTE   45      ;G
     80 000110      046                 .BYTE   46      ;H--------110
     81 000111      047                 .BYTE   47      ;I
     82 000112      050                 .BYTE   50      ;J
     83 000113      051                 .BYTE   51      ;K
     84 000114      052                 .BYTE   52      ;L
     85 000115      053                 .BYTE   53      ;M
     86 000116      054                 .BYTE   54      ;N
     87 000117      055                 .BYTE   55      ;O
     88 000120      056                 .BYTE   56      ;P--------120
     89 000121      057                 .BYTE   57      ;Q
     90 000122      060                 .BYTE   60      ;R
     91 000123      061                 .BYTE   61      ;S
     92 000124      062                 .BYTE   62      ;T
     93 000125      063                 .BYTE   63      ;U
     94 000126      064                 .BYTE   64      ;V
     95 000127      065                 .BYTE   65      ;W
     96 000130      066                 .BYTE   66      ;X---------130
     97 000131      067                 .BYTE   67      ;Y
     98 000132      070                 .BYTE   70      ;Z
     99 000133      071                 .BYTE   71      ;[
    100 000134      072                 .BYTE   72      ;\ (DEC POINT)
    101 000135      073                 .BYTE   73      ;]
    102 000136      074                 .BYTE   74      ;^
    103 000137      074                 .BYTE   74      ;←
    104 000140      074                 .BYTE   74      ;@---------140
    105 000141      037                 .BYTE   37      ;A
    106 000142      040                 .BYTE   40      ;B
    107 000143      041                 .BYTE   41      ;C
    108 000144      042                 .BYTE   42      ;D
    109 000145      043                 .BYTE   43      ;E
    110 000146      044                 .BYTE   44      ;F
    111 000147      045                 .BYTE   45      ;G
    112 000150      046                 .BYTE   46      ;H--------150
    113 000151      047                 .BYTE   47      ;I
    114 000152      050                 .BYTE   50      ;J
```

```
    115 000153     051                          .BYTE   51      ;K
    116 000154     052                          .BYTE   52      ;L
    117 000155     053                          .BYTE   53      ;M
    118 000156     054                          .BYTE   54      ;N
    119 000157     055                          .BYTE   55      ;O
    120 000160     056                          .BYTE   56      ;P--------160
    121 000161     057                          .BYTE   57      ;Q
    122 000162     060                          .BYTE   60      ;R
    123 000163     061                          .BYTE   61      ;S
    124 000164     062                          .BYTE   62      ;T
    125 000165     063                          .BYTE   63      ;U
    126 000166     064                          .BYTE   64      ;V
    127 000167     065                          .BYTE   65      ;W
    128 000170     066                          .BYTE   66      ;X---------170
    129 000171     067                          .BYTE   67      ;Y
    130 000172     070                          .BYTE   70      ;Z
    131 000173     074                          .BYTE   74      ;[
    132 000174     074                          .BYTE   74      ;\
    133 000175     074                          .BYTE   74      ;]
    134 000176     074                          .BYTE   74      ;↑
    135 000177     074                          .BYTE   74      ;DEL------177
    136                                 ;
    137                                 ;   END OF ASCII TO 6-BIT CONVERSION TABLE
    138                                 ;
```

```
 1                                              .SBTTL  EMX
 2  000000                                      .PSECT  EMX
 3                                      ;  E-MATRIX (FSA-A &FSA-B) OFFSETS AND BIT DEFINITIONS
 4                                      ;
 5                                      ;=======BYTE0   ;CHARACTER CODE (BINARY VALUE)
 6                                      ;=======BYTE0   ;DOCUMENT TYPE  (BINARY VALUE)
 7                                      ;=======BYTE0   ;ZONE           (BINARY VALUE)
 8                                      ;=======BYTE0   ;SUBZONE        (BINARY VALUE)
 9                                      ;=======WORD0   ;# VECTORS THAT EMX EXPANDS TO (BYTES)
10                                      ;=======WORD0   ;VECTOR EMX MERGES TO (OFFSET FROM EMX START)
11          000002                      EMXMCD==WORD1   ;MATCH CODE     (BINARY VALUE)
12          000002                      EMXNB1==WORD1   ;NIBBLE 1       (BIT MATRIX)
13          000004                      EMXNB2==WORD2   ;NIBBLE 2       (BIT MATRIX)
14                                      ;
15                                      ;  EMX FLAG DEFINITIONS IN CONTROL WORD
16                                      ;
17          100000                      EMXNSQ==BIT15   ;NON-SEQUENCIAL EMX FLOW
18          000001                      EMXEXF==BIT0    ;EMX FLOW EXPANDS
19                                              ;FOLLOWING VALID ONLY IF EMXNSQ=0
20                                              ;NOTE: ALL IN NEXT GROUP ARE MUTUALLY EXCLUSIVE EXCEPT
21                                              ;       EMXMTV WHICH CAN OCCUR WITH ALL EXCEPT EMXMCF
22          040000                      EMXMTV==BIT14   ;THIS EMX ENTRY REPRESENTS MULTIPLE VALUES
23          020000                      EMXCHF==BIT13   ;THIS EMX ENTRY IS A CHARACTER
24          010000                      EMXDTF==BIT12   ;THIS EMX ENTRY IS A DOCUMENT TYPE
25          004000                      EMXZNF==BIT11   ;THIS EMX ENTRY IS A ZONE
26          002000                      EMXSZF==BIT10   ;THIS EMX ENTRY IS A SUBZONE
27          001000                      EMXMCF==BIT9    ;THIS EMX ENTRY IS A MATCH CODE
28                                              ;FOLLOWING VALID ONLY IF EMXMTV=1
29          004000                      EMXVDC==BIT11   ;VLDC  (VARIABLE LENGTH DON'T CARE)
30          002000                      EMXFDC==BIT10   ;FLDC  (FIXED LENGTH DON'T CARE)
31          001000                      EMXNVD==BIT9    ;NVLDC (NUMERICAL VLDC)
32          000400                      EMXNFD==BIT8    ;NFLDC (NUMERICAL FLDC)
33          000200                      EMX0VD==BIT7    ;ZVLDC (ZERO VLDC)
34                                              ;FOLLOWING VALID ONLY IF EMXVDC=1
35          010000                      EMXTRL==BIT12   ;THIS ENTRY IS FOLLOWED BY AN INTERWORD
36          000001                      EMXVVV==BIT0    ;SUCCESSIVE VLDC'S IN SAME TERM
37                                      ;
38                                      ;  DON'T CARE DEFINITIONS
39                                      ;
40          177777                      INTRWD  ==-1    ;INTER WORD
41          177776                      VLDC    ==-2    ;VLDC
42          177775                      FLDC    ==-3    ;FLDC
43          177774                      NVLDC   ==-4    ;NUMERICAL VLDC
44          177773                      NFLDC   ==-5    ;NUMERICAL FLDC
45          177772                      ZVLDC   ==-6    ;ZERO VLDC
46          177770                      SEGOP   ==-10   ;SEGMENT OPERATOR (NUMERICAL RANGE INDICATOR)
47                                      ;
48                                      ;  6-BIT CODE TO EMX ENTRY CONVERSION TABLE
49                                      ;
50                                      ;       FIRST PART IS DON'T CARE ENTRIES (NEGATIVE INDEX FROM CETEMX)
51                                      ;
52  000000  060200  100000  002000  ZVDEMX::.WORD   EMXMTV!EMXCHF!EMX0VD,100000,2000          ;ZVLDC
53  000006  060400  100000  007774  NFEMX:: .WORD   EMXMTV!EMXCHF!EMXNFD,100000,7774          ;NFLDC
54  000014  061400  100000  007774  NVDEMX::.WORD   EMXMTV!EMXCHF!EMXNVD!EMXNFD,100000,7774  ;NVLDC
55  000022  062000  177776  037777           .WORD   EMXMTV!EMXCHF!EMXFDC,177776,37777         ;FLDC
56  000030  064000  177776  037777           .WORD   EMXMTV!EMXCHF!EMXVDC,177776,37777         ;VLDC
57  000036  020000  000001  000001  INTERW::.WORD   EMXCHF,1,1                                 ;INTER WORD
```

```
   58 000044  000000  000000  000000  CETEMX::.WORD   0,0,0                   ;0---SUBDOC/DOC ID MARK(RES.
   59 000052  000000  000000  000000          .WORD   0,0,0                   ;1---ZONE MARK (RESERVED)
   60 000060  000000  000000  000000          .WORD   0,0,0                   ;2---SUBZONE MARK (RESERVED)
   61 000066  000000  000000  000000          .WORD   0,0,0                   ;3---PARAGRAPH MARK (RESERVED)
   62 000074  000000  000000  000000          .WORD   0,0,0                   ;4---SENTENCE MARK (RESERVED)
   63 000102  020000  000001  000001          .WORD   EMXCHF,BIT0,BIT0        ;5---SPACE      (IW)
   64 000110  020000  000001  000001          .WORD   EMXCHF,BIT0,BIT0        ;6---!          (IW)
   65 000116  020000  000001  000001          .WORD   EMXCHF,BIT0,BIT0        ;7---"          (IW)
   66 000124  020301  000002  010000          .WORD   EMXCHF+301,BIT1,BIT12   ;10--#
   67 000132  020302  000004  010000          .WORD   EMXCHF+302,BIT2,BIT12   ;11--%
   68 000140  020313  004000  010000          .WORD   EMXCHF+313,BIT11,BIT12  ;12--&
   69 000146  020307  000200  010000          .WORD   EMXCHF+307,BIT7,BIT12   ;13--'
   70 000154  020000  000001  000001          .WORD   EMXCHF,BIT0,BIT0        ;14--(          (IW)
   71 000162  020000  000001  000001          .WORD   EMXCHF,BIT0,BIT0        ;15--)          (IW)
   72 000170  020305  000040  010000          .WORD   EMXCHF+305,BIT5,BIT12   ;16--*
   73 000176  020306  000100  010000          .WORD   EMXCHF+306,BIT6,BIT12   ;17--+
   74 000204  020000  000001  000001          .WORD   EMXCHF,BIT0,BIT0        ;20--,          (IW)
   75 000212  020304  000020  010000  MINEMX::.WORD   EMXCHF+304,BIT4,BIT12   ;21--MINUS SIGN
   76 000220  020324  000020  020000          .WORD   EMXCHF+324,BIT4,BIT13   ;22--.          (IW)
   77 000226  020310  000400  010000          .WORD   EMXCHF+310,BIT8,BIT12   ;23--/
   78 000234  020257  100000  002000          .WORD   EMXCHF+257,BIT15,BIT10  ;24--0
   79 000242  020277  100000  004000          .WORD   EMXCHF+277,BIT15,BIT11  ;25--1
   80 000250  020057  100000  000004          .WORD   EMXCHF+57,BIT15,BIT2    ;26--2
   81 000256  020077  100000  000010          .WORD   EMXCHF+77,BIT15,BIT3    ;27--3
   82 000264  020117  100000  000020          .WORD   EMXCHF+117,BIT15,BIT4   ;30--4
   83 000272  020137  100000  000040          .WORD   EMXCHF+137,BIT15,BIT5   ;31--5
   84 000300  020157  100000  000100          .WORD   EMXCHF+157,BIT15,BIT6   ;32--6
   85 000306  020177  100000  000200          .WORD   EMXCHF+177,BIT15,BIT7   ;33--7
   86 000314  020217  100000  000400          .WORD   EMXCHF+217,BIT15,BIT8   ;34--8
   87 000322  020237  100000  001000          .WORD   EMXCHF+237,BIT15,BIT9   ;35--9
   88 000330  020311  001000  010000          .WORD   EMXCHF+311,BIT9,BIT12   ;36--SIG. SPACE
   89 000336  020001  000002  000001          .WORD   EMXCHF+1,BIT1,BIT0      ;37--A  :
   90 000344  020002  000004  000001          .WORD   EMXCHF+2,BIT2,BIT0      ;40--B  A
   91 000352  020003  000010  000001          .WORD   EMXCHF+3,BIT3,BIT0      ;41--C  L
   92 000360  020004  000020  000001          .WORD   EMXCHF+4,BIT4,BIT0      ;42--D  P
   93 000366  020005  000040  000001          .WORD   EMXCHF+5,BIT5,BIT0      ;43--E  H
   94 000374  020006  000100  000001          .WORD   EMXCHF+6,BIT6,BIT0      ;44--F  A
   95 000402  020007  000200  000001          .WORD   EMXCHF+7,BIT7,BIT0      ;45--G  B
   96 000410  020010  000400  000001          .WORD   EMXCHF+10,BIT0,BIT0     ;46--H  E
   97 000416  020011  001000  000001          .WORD   EMXCHF+11,BIT9,BIT0     ;47--I  T
   98 000424  020012  002000  000001          .WORD   EMXCHF+12,BIT10,BIT0    ;50--J  I
   99 000432  020013  004000  000001          .WORD   EMXCHF+13,BIT11,BIT0    ;51--K  C
  100 000440  020014  010000  000001          .WORD   EMXCHF+14,BIT12,BIT0    ;52--L  S
  101 000446  020015  020000  000001          .WORD   EMXCHF+15,BIT13,BIT0    ;53--M  :
  102 000454  020016  040000  000001          .WORD   EMXCHF+16,BIT14,BIT0    ;54--N  :
  103 000462  020033  004000  000002          .WORD   EMXCHF+33,BIT11,BIT1    ;55--O  :
  104 000470  020034  010000  000002          .WORD   EMXCHF+34,BIT12,BIT1    ;56--P  A
  105 000476  020021  000002  000002          .WORD   EMXCHF+21,BIT1,BIT1     ;57--Q  L
  106 000504  020022  000004  000002          .WORD   EMXCHF+22,BIT2,BIT1     ;60--R  P
  107 000512  020023  000010  000002          .WORD   EMXCHF+23,BIT3,BIT1     ;61--S  H
  108 000520  020024  000020  000002          .WORD   EMXCHF+24,BIT4,BIT1     ;62--T  A
  109 000526  020025  000040  000002          .WORD   EMXCHF+25,BIT5,BIT1     ;63--U  B
  110 000534  020026  000100  000002          .WORD   EMXCHF+26,BIT6,BIT1     ;64--V  E
  111 000542  020027  000200  000002          .WORD   EMXCHF+27,BIT7,BIT1     ;65--W  T
  112 000550  020030  000400  000002          .WORD   EMXCHF+30,BIT8,BIT1     ;66--X  I
  113 000556  020031  001000  000002          .WORD   EMXCHF+31,BIT9,BIT1     ;67--Y  C
  114 000564  020032  002000  000002          .WORD   EMXCHF+32,BIT10,BIT1    ;70--Z  :
```

```
    115 000572  020312  002000  010000           .WORD   EMXCHF+312,BIT10,BIT12          ;71--[
    116 000600  020303  000010  010000  DPEMX::  .WORD   EMXCHF+303,BIT3,BIT12           ;72--DECIMAL POINT
    117 000606  020314  010000  010000           .WORD   EMXCHF+314,BIT12,BIT12          ;73--]
    118 000614  020315  020000  010000           .WORD   EMXCHF+315,BIT13,BIT12          ;74--ESCAPE
    119 000622  020000  000001  000001           .WORD   EMXCHF,BIT0,BIT0                ;75--:          (IW)
    120 000630  020000  000001  000001           .WORD   EMXCHF,BIT0,BIT0                ;76--;          (IW)
    121 000636  020000  000001  000001           .WORD   EMXCHF,BIT0,BIT0                ;77--?          (IW)
    122                                  ;
    123                                  ;  END OF 6-BIT TO EMX ENTRY CONVERSION TABLE
    124                                  ;
```

```
                                        .MCALL  FDBDF$,FDAT$A,FDRC$A,FDBK$A,FDOP$A,FDBF$A,NMBLK$
                                        .MCALL  SDAT$S,RSUM$S,FDAT$R,OFNB$W,CLOSE$,PRINT$
                                        .SBTTL  BUFFER DEFINITIONS
000000                                  .PSECT  EMXDAT
                                ;
                                ;   .XQIO ROUTINE PARAMETER BUFFERS
                                ;
000000  000000                 PARBUF::.WORD   0                       ;PAR 1 - BUFFER ADDRESS
000002  000000                         .WORD   0                       ;PAR 2 - BUFFER LENGTH (BYTES)
000004  000000                         .WORD   0                       ;PAR 3
000006  000000                         .WORD   0                       ;PAR 4 - VIRTUAL DISK ADDRESS (HI)
000010  000001                         .WORD   1                       ;PAR 5 -    "       "       "    (LO)
                                ;
                                ;  EMATRIX FDB
                                ;
000012                          EMXFDB::FDBDF$
                                        FDRC$A  FD.RWM!FD.RAN
                                        FDBK$A  EMXLGT,512.,,,IOST
                                        FDOP$A  LUNFIL
                                        FDBF$A  ,,1
000152                          IOST:   .BLKW   2
        000001                  LUNFIL=1
000156  000000                  VINXT:: .WORD   0       ;NEXT AVAILABLE OFFSET IN VI
000160  000000                  EMASIZ: .WORD   0
000162  000000                  EMBSIZ: .WORD   0
000164  000000                  EMCSIZ: .WORD   0
000166  000000                  SAVBST: .WORD   0       ;BATCH STATUS TABLE ADDRESS
000170  000000                  EXPVPT: .WORD   0       ;POINTER TO ENTRY IN EXPANSION VECTOR
000172  000000                  FLUIDS: .WORD   0       ;FLU ID FOR CURRENT FLU-MOD NODE
000174  000000                  N1:     .WORD   0       ;STORAGE FOR FLU-MOD TERM NIBBLES (CURRENT)
000176  000000                  N2:     .WORD   0
000200  002000                  N0:     .WORD   EMXSZF
000202  000000                  ZN1:    .WORD   0       ;ACTIVE ZONE NIBBLE STORAGE
000204  000000                  ZN2:    .WORD   0
000206  004000                  ZN0:    .WORD   EMXZNF
000210  000000                  TN1:    .WORD   0       ;ACTIVE DOC-TYPE NIBBLE STORAGE
000212  000000                  TN2:    .WORD   0
000214  010000                  TN0:    .WORD   EMXDTF
000216  000000                  PRIORS: .WORD   0       ;PRIOR FLU-MOD TERM TYPE
000220  000000                  PRIORV: .WORD   0       ;PRIOR FLU-MOD TERM VALUE (NIBBLE 2)
000222  000000                  COUNT:  .WORD   0       ;MISCELLANEOUS COUNTER
000224  000000                  VDCFLG: .WORD   0       ;FLAG FOR VLDC OCCURRENCE
                                ;
                                ; STATE TABLE FOR FLU-MOD PROCESSING
                                ;
000226                          STABLE:
000226     000     000     370          .BYTE   0,0,-S2,JSB1,-S3,JSB3,-S5,JSB6          ;STATE 1 ENTRIES
        000010                  S2=.-STABLE
000236     010     002     370          .BYTE   S2,JSB2,-S2,JSB12,-S3,JSB13,-S5,JSB9    ;STATE 2
        000020                  S3=.-STABLE
000246     030     010     010          .BYTE   S4,JSB5,S2,JSB1,-S3,JSB13,-S5,JSB9      ;STATE 3
        000030                  S4=.-STABLE
000256     030     006     370          .BYTE   S4,JSB4,-S2,JSB12,-S3,JSB13,-S5,JSB10   ;STATE 4
        000040                  S5=.-STABLE
000266     050     016     010          .BYTE   S6,JSB8,S2,JSB1,S3,JSB3,-S5,JSB9        ;STATE 5
        000050                  S6=.-STABLE
000276     050     014     370          .BYTE   S6,JSB7,-S2,JSB12,-S3,JSB13,-S5,JSB11   ;STATE 6
```

```
 59 000000                              .PSECT  EMXCDE
 60                                     ;
 61                                     ;
 62                                     ;  ROUTINE TO COMPLETE BATCH CLOSURE BY CREATING EMX FILES FROM FLU NODES.
 63                                     ;  (ALL REGISTERS SCRATCH)
 64                                     ;
 65 000000                              CLSEMX::
 66 000000  016702  000000G             MOV     BATNO,R2        ;GET BATCH STATUS TABLE ADDRESS
 67 000004  016202  000000G             MOV     BSTPTR(R2),R2
 68 000010  010267  000166'             MOV     R2,SAVBST       ;SAVE BATCH STATUS TABLE ADDRESS
 69 000014  005767  000000G             TST     NODEA           ;ANY FLU'S FOR FSA-A
 70 000020  001406                      BEQ     100$            ;NO
 71 000022  000167  000072              JMP     10$             ;YES
 72 000026  000167  000142      120$:   JMP     20$
 73 000032  000167  000212      130$:   JMP     30$
 74 000036  012701  000012      100$:   MOV     #FN.EMA,R1      ;CREATE DUMMY EMA
 75 000042  005000                      CLR     R0
 76 000044  004767  002172              JSR     PC,WRTEMX
 77 000050  005767  000000G     1$:     TST     NODEB           ;ANY FLU'S FOR FSA-B?
 78 000054  001364                      BNE     120$            ;YES
 79 000056  012701  000014              MOV     #FN.EMB,R1      ;CREATE DUMMY EMB
 80 000062  005000                      CLR     R0
 81 000064  004767  002152              JSR     PC,WRTEMX
 82 000070  005767  000000G     2$:     TST     NODEC           ;ANY FLU'S FOR FSA-C?
 83 000074  001356                      BNE     130$            ;YES
 84 000076  012701  000016              MOV     #FN.EMC,R1      ;CREATE DUMMY EMC
 85 000102  005000                      CLR     R0
 86 000104  004767  002132              JSR     PC,WRTEMX
 87 000110                      3$:
 88                                     .IF     DF,QT0SUM
 89                                     PRINT$  #LIST
 90                                     .ENDC
 91 000110                              CALL    SSQ             ;SEND ASK TO MSCHED VIA SSQ
 92 000114  000167  000000G             JMP     START3          ;CONTINUE PROCESSING QUERIES
 93                                     ;
 94                                     ;  ROUTINE TO CREATE EMA
 95                                     ;
 96 000120  012703  000000G     10$:    MOV     #FLUIDX,R3      ;R3->FLU INDEX POINTERS
 97 000124  012705  000000G             MOV     #FLIXSZ,R5      ;R5->#.FLU INDEX POINTERS
 98 000130  012702  000000G             MOV     #NDFSAA,R2      ;R2->TYPE FLU IN POOL DESIRED
 99 000134  004767  000154              JSR     PC,GENEMA       ;GENERATE EMATRIX FOR FSA-A
100 000140  162700  000000G             SUB     #EMX,R0         ;STORE EMX SIZE
101 000144  010067  000160'             MOV     R0,EMASIZ
102 000150  006200                      ASR     R0
103 000152  010067  000002G             MOV     R0,QTSTAT+2     ;STORE EMA SIZE IN WORDS IN STAT AREA
104 000156  006300                      ASL     R0
105 000160  012701  000012              MOV     #FN.EMA,R1      ;FILE TYPE
106 000164  004767  002052              JSR     PC,WRTEMX       ;WRITE THE EMX DATA TO DISK
107 000170  000167  177654              JMP     1$
108                                    ;
109                                    ;  ROUTINE TO CREATE EMB
110                                    ;
111 000174  012703  000000G    20$:    MOV     #FLUIDX,R3      ;R3->FLU INDEX POINTERS
112 000200  012705  000000G            MOV     #FLIXSZ,R5      ;R5->#.FLU INDEX POINTERS
113 000204  012702  000000G            MOV     #NDFSAB,R2      ;R2->TYPE FLU IN POOL DESIRED
114 000210  004767  000100             JSR     PC,GENEMA       ;GENERATE EMATRIX FOR FSA-B
115 000214  162700  000000G            SUB     #EMX,R0         ;STORE EMX SIZE
```

```
 116 000220  010067  000162'                MOV     R0,EMBSIZ
 117 000224  006200                         ASR     R0
 118 000226  010067  000006G                MOV     R0,QTSTAT+6     ;STORE EMB SIZE IN WORDS IN STAT AREA.
 119 000232  006300                         ASL     R0
 120 000234  012701  000014                 MOV     #FN.EMB,R1      ;FILE TYPE.
 121 000240  004767  001776                 JSR     PC,WRTEMX       ;WRITE THE EMX DATA TO DISK.
 122 000244  000167  177620                 JMP     2$
 123                                ;
 124                                ;  ROUTINE TO CREATE EMC.
 125                                ;
 126 000250  012703  000000G        30$:    MOV     #CWPIDX,R3      ;R3->CWP INDEX POINTERS.
 127 000254  012705  000000G                MOV     #CPIXSZ,R5
                                                                    ;R5->#.CWP INDEX POINTERS.
 128 000260  012702  177777                 MOV     #-1,R2          ;R2->TYPE NODE DESIRED (ALL)
 129 000264  004767  001624                 JSR     PC,GENEMC       ;GENERATE EMATRIX FOR FSA-C
 130 000270  162700  000000G                SUB     #EMX,R0         ;STORE EMX SIZE.
 131 000274  010067  000164'                MOV     R0,EMCSIZ
 132 000300  012701  000016                 MOV     #FN.EMC,R1      ;FILE TYPE.
 133 000304  004767  001732                 JSR     PC,WRTEMX       ;WRITE THE EMX DATA TO DISK.
 134 000310  000167  177574                 JMP     3$
 135                                ;
 136                                ;  SUBROUTINE TO GENERATE EMA OR EMB.
 137                                ;
 138                                ;               INPUT.                          OUTPUT.
 139                                ;       R0=.    SCRATCH.                        END OF EMX ADDRESS.
 140                                ;       R1=.    SCRATCH.                        SCRATCH.
 141                                ;       R2=.    TYPE OF NODE DESIRED.           (SAME)
 142                                ;       R3=.    INDEX POINTER ADDRESS.          END OF POINTERS ADDRESS.
 143                                ;       R4=.    SCRATCH.                        0
 144                                ;       R5=.    #. INDEX POINTERS.              0
 145                                ;
 146 000314  012704  000000G        GENEMA: MOV     #ZERO,R4        ;INITIALIZE (TRIGGER INDEX POINTER REFERENCE)
 147 000320  005067  000156'                CLR     VINXT.          ;       RESET NEXT VI ADDRESS.
 148 000324  012700  000000G                MOV     #EMX,R0         ;       SET START OF EMX BUFFER.
 149 000330  004767  000000G        100$:   JSR     PC,GETNDE       ;GET NEXT NODE OF TYPE DESIRED.
 150 000334  103530                         BCS     3$              ;NO MORE NODES.
 151 000336  005067  000224'                CLR     VDCFLG          ;INIT VLDC OCCURENCE FLAG.
 152 000342                                 SAVE    R4,R5
 153 000346  016701  000156'                MOV     VINXT,R1        ;MAKE VI ENTRY.
 154 000352  010061  000000G                MOV     R0,VI(R1)
 155 000356  162761  000000G 000000G        SUB     #EMX,VI(R1)
 156 000364  062767  000002  000156'        ADD     #2,VINXT
 157 000372  116405  000000G                MOVB    NDSIZ(R4),R5    ;R5->#.CHARS IN NODE.
 158 000376  042705  177400                 BIC     #177400,R5
 159 000402  062704  000000G                ADD     #NDCHR,R4       ;R4->@.1ST.CHAR.
 160 000406  122714  177772                 CMPB    #ZVLDC,(R4)     ;IS FIRST CHAR A ZERO VLDC?.
 161 000412  001001                         BNE     101$            ;NO.
 162 000414  000403                         BR      102$            ;YES: THEN INCLUDE NEXT ENTRY.
 163 000416  122714  177776         101$:   CMPB    #VLDC,(R4)      ;IS FIRST CHAR A VLDC?.
 164 000422  001010                         BNE     1$              ;NO.
 165 000424  062701  000002G        102$:   ADD     #VI+2,R1        ;YES: SET NEXT VI TO NEXT EMX ENTRY.
 166 000430  010011                         MOV     R0,(R1)
 167 000432  162711  177772G                SUB     #EMX-6,(R1)
 168 000436  062767  000002  000156'        ADD     #2,VINXT
 169 000444  112401                 1$:     MOVB    (R4)+,R1        ;GET CHAR FROM NODE.
 170 000446  100425                         BMI     2i$             ;DON'T CARE.
 171 000450  116101  000000'                MOVB    CET6B(R1),R1    ;CONVERT ASCII TO 6-BIT CODE
 172 000454  001466                         BEQ     10$             ;ILLEGAL
```

```
 173 000456  070127  000006          2$:     MUL     #6,R1           ;CONVERT 6-BIT CODE TO EMX TABLE INDEX
 174 000462  062701  000044'                 ADD     #CETEMX,R1      ;R1->ADDRESS OF EMX ENTRY IN TABLE
 175 000466  012120                          MOV     (R1)+,(R0)+     ;TRANSFER EMX ENTRY FOR CHAR
 176 000470  012120                  4$:     MOV     (R1)+,(R0)+
 177 000472  012120                          MOV     (R1)+,(R0)+
 178 000474  077515                  6$:     SOB     R5,1$           ;DO FOR ALL CHARS IN NODE
 179 000476                          5$:     RESTORE R4,R5
 180 000502  012701  000036'                 MOV     #INTERW,R1      ;TRANSFER INTER WORD EMX ENTRY
 181 000506  012120                          MOV     (R1)+,(R0)+
 182 000510  012120                          MOV     (R1)+,(R0)+
 183 000512  012120                          MOV     (R1)+,(R0)+
 184 000514  004767  000124              JSR     PC,GENFMD       ;GENERATE FLU MOD EMX ENTRIES
 185 000520  000703                      BR      100$
 186 000522  122701  177776          21$:    CMPB    #VLDC,R1        ;IS IT A VLDC?
 187 000526  001025                      BNE     23$             ;NO
 188 000530  070127  000006              MUL     #6,R1           ;YES: CONVERT 6-BIT CODE TO EMX TABLE INDEX
 189 000534  062701  000044'             ADD     #CETEMX,R1      ;R1->ADDRESS OF EMX ENTRY IN TABLE
 190 000540  012120                      MOV     (R1)+,(R0)+     ;XFER 1ST EMX ENTRY WORD
 191 000542  020527  000001              CMP     R5,#1           ;LAST ENTRY?
 192 000546  001003                      BNE     24$             ;NO
 193 000550  052760  010000  177776      BIS     #EMXTRL,-2(R0)  ;YES: FLAG AS TRAILING
 194 000556  005767  000224'      24$:    TST     VDCFLG          ;HAS A VLDC OCCURED BEFORE IN THIS TERM?
 195 000562  001404                      BEQ     22$             ;NO
 196 000564  052760  000001  177776      BIS     #EMXVVV,-2(R0)  ;YES: FLAG IT
 197 000572  000736                      BR      4$              ;RESUME XFER
 198 000574  005267  000224'      22$:    INC     VDCFLG          ;FLAG OCCURRENCE
 199 000600  000733                      BR      4$              ;RESUME XFER
 200 000602  122701  177770      23$:    CMPB    #SEGOP,R1       ;IS IT A NUMBER RANGE?
 201 000606  001323                      BNE     2$              ;NO
 202 000610  004767  000000G             JSR     PC,NUMRNG       ;YES: CONVERT AND XFER
 203 000614  000727                      BR      6$              ;COMPLETE REST OF NODE
 204 000616  016701  000156'      3$:     MOV     VINXT,R1        ;STORE # VECTORS
 205 000622  006201                      ASR     R1
 206 000624  010167  000000G             MOV     R1,VILGT
 207 000630  000207                      RTS     PC
 208 000632  012767  000004  000000G 10$:    MOV     #QE.ILC,ERRCDE  ;REPORT ERROR
 209 000640  000167  000000G             JMP     ERRORF
 210                                ;
 211                                ; SUBROUTINE TO GENERATE FLU MOD EMX ENTRIES
 212                                ;
 213                                ;
 214 000644                         GENFMD: SAVE    R2,R3,R5,R4
 215 000654  005005                      CLR     R5              ;R5 = COUNT OF FLU MOD NODES LINKED TO FLU
 216 000656  016404  000000G      1$:     MOV     NDFMEN(R4),R4   ;GET FLU MOD LINK NODE
 217 000662  001402                      BEQ     2$              ;NO MORE LINK NODES
 218 000664  005205                      INC     R5              ;COUNT OF LINK NODES
 219 000666  000773                      BR      1$
 220 000670  011604                  2$:     MOV     (SP),R4         ;GET FLU NODE ADDRESS
 221 000672  005705                      TST     R5              ;ANY LINK NODES?
 222 000674  001420                      BEQ     4$              ;NO
 223 000676  005205                      INC     R5              ;YES: CONVERT TO FLU MOD COUNT
 224 000700  006305                      ASL     R5              ;CONSTRUCT AND TRANSFER EXPANSION VECTOR
 225 000702  010510                      MOV     R5,(R0)
 226 000704  052720  140001              BIS     #EMXNSQ!EMXEXF!BIT14,(R0)+
 227 000710  010067  000170'             MOV     R0,EXPVPT       ;ADDRESS OF VECTORS IN EXPANSION
 228 000714  060500                      ADD     R5,R0           ;STEP PAST EXPANSION VECTOR
 229 000716  010001                  3$:     MOV     R0,R1           ;TRANSFER EMX OFFSET TO EXPANSION VECTOR
```

```
    230 000720  162701  000000G                 SUB     #EMX,R1
    231 000724  010177  000170'                 MOV     R1,@EXPVPT
    232 000730  062767  000002  000170'         ADD     #2,EXPVPT
    233 000736  016467  000000G 000172' 4$:     MOV     NDFLID(R4),FLUIDS       ;STORE FLU ID IN FIXED LOCATION
    234 000744  004767  000030                  JSR     PC,FMNDCD               ;PROCESS THE ACTUAL FLU-MOD NODE
    235 000750  012720  001000                  MOV     #EMXMCF,(R0)+   ;CREATE MATCH CODE EMX ENTRY
    236 000754  016420  000000G                 MOV     NDFLID(R4),(R0)+
    237 000760  016404  000000G                 MOV     NDFMEN(R4),R4   ;GET NEXT LINK
    238 000764  001354                          BNE     3$              ;CONTINUE
    239 000766                                  RESTORE R2,R3,R5,R4     ;NO MORE LINKS, FINISHED
    240 000776  000207                          RTS     PC
    241                             ;
    242                             ;  SUBROUTINE TO GENERATE EMX ENTRIES FOR FLU-MOD NODE
    243                             ;
    244                             ;                       INPUT                           OUTPUT
    245                             ;       R0=     EMX ADDRESS (BEGIN)             EMX ADDRESS (END)
    246                             ;       R1=     SCRATCH                         SCRATCH
    247                             ;       R2=     SCRATCH                         SCRATCH
    248                             ;       R3=     SCRATCH                         SCRATCH
    249                             ;       R4=     FLU OR LINK NODE ADR (SAVED)    (RESTORED)
    250                             ;       R5=     SCRATCH                         SCRATCH
    251                             ;
    252 001000  010446              FMNDCD: MOV     R4,-(SP)        ;SAVE FLU OR LINK NODE ADDRESS
    253 001002  005067  000222'             CLR     COUNT           ;INITIALIZE # EMX SUB-VECTORS NECESSARY
    254 001006  016404  000000G             MOV     NDFMN(R4),R4    ;R4 <- FLU-MOD NODE
    255 001012  116401  000000G             MOVB    NDSIZ(R4),R1    ;R1 <- # FLU-MOD TERMS
    256 001016  001515                      BEQ     77$             ;IF NO FLU-MOD TERMS, SKIP ALL PROCESSING
    257 001020  042701  177400              BIC     #177400,R1
    258 001024  062704  000000G             ADD     #NDFMS,R4       ;R4 <- 1ST FLU-MOD TERM ADDRESS
    259 001030  005005                      CLR     R5              ;R5 <- STATE1 IN STABLE
    260 001032  005067  000216'             CLR     PRIORS          ;PRIOR FLU-MOD TERM TYPE = 0 (CAN'T MATCH)
    261 001036  112403              1$:     MOVB    (R4)+,R3        ;GET FLU-MOD TERM
    262 001040  042703  177400              BIC     #177400,R3
    263 001044  005002                      CLR     R2              ;R2=TERM TYPE (1,2,3=S,Z,T)
    264 001046  071227  000100              DIV     #100,R2         ;R3=TERM VALUE
    265 001052  010246                      MOV     R2,-(SP)        ;SAVE TERM TYPE
    266 001054  005002                      CLR     R2              ;R3=NIBBLE 1 VALUE
    267 001056  071227  000020              DIV     #20,R2          ;R2=NIBBLE 2 VALUE
    268 001062  020267  000220'             CMP     R2,PRIORV       ;ARE NIBBLE 1 VALUES OF CURRENT=PRIOR?
    269 001066  001406                      BEQ     2$              ;YES
    270 001070  010267  000220'             MOV     R2,PRIORV       ;SET CURRENT AS PRIOR NIBBLE 1 VALUE
    271 001074  012602                      MOV     (SP)+,R2        ;RESTORE CURRENT TERM TYPE
    272 001076  010267  000216'             MOV     R2,PRIORS       ;SET AS PRIOR
    273 001102  000410                      BR      3$
    274 001104  012602              2$:     MOV     (SP)+,R2        ;RESTORE CURRENT TERM TYPE
    275 001106  020267  000216'             CMP     R2,PRIORS       ;IS CURRENT=PRIOR TERM TYPE?
    276 001112  001403                      BEQ     21$             ;YES
    277 001114  010267  000216'             MOV     R2,PRIORS       ;NO: SET AS PRIOR
    278 001120  000401                      BR      3$
    279 001122  005002              21$:    CLR     R2              ;SET TERM TYPE AS SPECIAL (SINGLE EXIT STATE?)
    280 001124  006302              3$:     ASL     R2
    281 001126  060502                      ADD     R5,R2           ;GET NEXT STATE
    282 001130  116205  000226'             MOVB    STABLE(R2),R5
    283 001134  100003                      BPL     4$              ;IF INCREMENT FLAG NOT SET
    284 001136  005267  000222'             INC     COUNT           ;IF INCREMENT FLAG SET, COUNT EMX SUB-VECTORS
    285 001142  005405                      NEG     R5              ;REMOVE INCREMENT FLAG EFFECT ON STATE OFFSET
    286 001144  077144              4$:     SOB     R1,1$           ;DO FOR ALL FLU-MOD TERMS
```

```
 287 001146  011604                        MOV     (SP),R4          ;RESTORE FLU OR LINK NODE ADDRESS
 288 001150  016746  000170'                MOV     EXPVPT,-(SP)     ;SAVE EXPANSION VECTOR POINTER
 289 001154  016701  000222'                MOV     COUNT,R1         ;R1 <- # EMX SUB-VECTORS NECESSARY
 290 001160  016404  000000G                MOV     NDFMN(R4),R4     ;GET FLU=MOD NODE
 291 001164  116467  000000G 000222'        MOVB    NDSIZ(R4),COUNT  ;COUNT <- #FLU-MOD TERMS
 292 001172  062704  000000G                ADD     #NDFMS,R4        ;R4 <- 1ST FLU-MOD TERM ADDRESS
 293 001176  005005                         CLR     R5               ;R5 <- STATE 1 IN STABLE
 294 001200  005067  000216'                CLR     PRIORS           ;NOW PRIOR STATE CAN'T MATCH
 295 001204  005067  000210'                CLR     TN1              ;A DOCUMENT TYPE HAS NOT YET BEEN SPECIFIED
 296 001210  005067  000202'                CLR     ZN1              ;A ZONE HAS NOT YET BEEN SPECIFIED
 297 001214  022701  000001                 CMP     #1,R1            ;ONLY ONE EMX SUB-VECTOR NECESSARY?
 298 001220  001416                         BEQ     10$              ;YES: THEN DON'T CREATE EXPANSION VECTOR
 299 001222  006301                         ASL     R1               ;NO: THEN CREATE EXPANSION VECTOR
 300 001224  010110                         MOV     R1,(R0)          ;TRANSFER # VECTORS
 301 001226  052720  100001                 BIS     #EMXNSQ!EMXEXF,(R0)+     ;FLAG AS EXPANSION VECTOR
 302 001232  010067  000170'                MOV     R0,EXPVPT        ;EXPVPT <- @ 1ST EMX OFFSET IN EXPANSION
 303 001236  060100                         ADD     R1,R0            ;R0 <- @ NEXT EMX ENTRY AFTER EXPANSION VECTOR
 304 001240  004767  000626                 JSR     PC,EMXLK2        ;LINK TO NEXT CREATED EMX ENTRY
 305 001244  000404                         BR      10$
 306 001246  012667  000170'        76$:    MOV     (SP)+,EXPVPT     ;RESTORE AND RETURN
 307 001252  012604                 77$:    MOV     (SP)+,R4
 308 001254  000207                         RTS     PC
 309 001256  112403                 10$:    MOVB    (R4)+,R3         ;GET FLU-MOD TERM
 310 001260  042703  177400                 BIC     #177400,R3
 311 001264  005002                         CLR     R2               ;R2=TERM TYPE
 312 001266  071227  000100                 DIV     #100,R2          ;R3=TERM VALUE
 313 001272  010246                         MOV     R2,-(SP)         ;SAVE TERM TYPE
 314 001274  110367  000200'                MOVB    R3,N0            ;STORE TERM VALUE
 315 001300  005002                         CLR     R2               ;R3=NIBBLE 1 VALUE
 316 001302  071227  000020                 DIV     #20,R2           ;R2=NIBBLE 2 VALUE
 317 001306  012701  000001                 MOV     #1,R1            ;CONVERT AND STORE 1ST NIBBLE
 318 001312  072103                         ASH     R3,R1
 319 001314  010167  000174'                MOV     R1,N1
 320 001320  012701  000001                 MOV     #1,R1            ;CONVERT AND STORE 2RD NIBBLE
 321 001324  072102                         ASH     R2,R1
 322 001326  010167  000176'                MOV     R1,N2
 323 001332  020267  000220'                CMP     R2,PRIORV        ;DOES CURRENT=PRIOR NIBBLE 2
 324 001336  001406                         BEQ     12$              ;YES
 325 001340  010267  000220'                MOV     R2,PRIORV        ;SET PRIOR=CURRENT NIBBLE 2 VALUE
 326 001344  012602                         MOV     (SP)+,R2         ;RESTORE CURRENT TERM TYPE
 327 001346  010267  000216'                MOV     R2,PRIORS        ;SET PRIOR=CURRENT TERM TYPE
 328 001352  000410                         BR      13$
 329 001354  012602                 12$:    MOV     (SP)+,R2         ;RESTORE CURRENT TERM TYPE
 330 001356  020267  000216'                CMP     R2,PRIORS        ;DO CURRENT = PRIOR TERM TYPE
 331 001362  001403                         BEQ     121$             ;YES
 332 001364  010267  000216'                MOV     R2,PRIORS        ;SET PRIOR=CURRENT TERM TYPE
 333 001370  000401                         BR      13$
 334 001372  005002                 121$:   CLR     R2               ;SET AS SPECIAL
 335 001374  006302                 13$:    ASL     R2
 336 001376  060502                         ADD     R5,R2            ;R5 WAS CURRENT STATE, R2 WAS CURRENT INPUT
 337 001400  116201  000227'                MOVB    STABLE+1(R2),R1  ;R1=SUBROUTINE OFFSET
 338 001404  116205  000226'                MOVB    STABLE(R2),R5    ;R5=NEXT STATE
 339 001410  100001                         BPL     14$
 340 001412  005405                         NEG     R5               ;REMOVE INCREMENT FLAG EFFECT IF PRESENT
 341 001414  004771  001460'        14$:    JSR     PC,@100$(R1)     ;PROCESS ACCORDING TO STATE TRANSFER
 342 001420  005367  000222'                DEC     COUNT            ;DO FOR ALL FLU-MOD TERMS
 343 001424  003314                         BGT     10$              ;NOT YET FINISHED
```

```
344 001426  020527  000030                  CMP     R5,#S4          ;IF STATE 4 OR 5, THEN ENTER LAST TERM
345 001432  001403                          BEQ     30$
346 001434  020527  000040                  CMP     R5,#S5
347 001440  001302                          BNE     76$
348 001442  052760  040000  177772  30$:    BIS     #EMXMTV,-6(R0)  ;FLAG AS MULTI-VALUE EMX ENTRY
349 001450  056760  000174' 177774          BIS     N1,-4(R0)       ;SET ADDITIONAL BIT
350 001456  000673                          BR      76$
351 001460                      100$:   ;SPECIAL SUBROUTINE DISPATCH TABLE
352         000000              JSB1=.-100$
353 001460  001512'                         .WORD   SB1
354         000002              JSB2=.-100$
355 001462  001530'                         .WORD   SB2
356         000004              JSB3=.-100$
357 001464  001546'                         .WORD   SB3
358         000006              JSB4=.-100$
359 001466  001606'                         .WORD   SB4
360         000010              JSB5=.-100$
361 001470  001622'                         .WORD   SB5
362         000012              JSB6=.-100$
363 001472  001704'                         .WORD   SB6
364         000014              JSB7=.-100$
365 001474  001750'                         .WORD   SB7
366         000016              JSB8=.-100$
367 001476  001764'                         .WORD   SB8
368         000020              JSB9=.-100$
369 001500  002100'                         .WORD   SB9
370         000022              JSB10=.-100$
371 001502  002164'                         .WORD   SB10
372         000024              JSB11=.-100$
373 001504  002146'                         .WORD   SB11
374         000026              JSB12=.-100$
375 001506  002010'                         .WORD   SB12
376         000030              JSB13=.-100$
377 001510  002042'                         .WORD   SB13
378                             ;  SPECIAL SUBROUTINES
379 001512  016720  000200'    SB1:    MOV     N0,(R0)+        ;CREATE SUB-ZONE EMX ENTRY
380 001516  016720  000174'            MOV     N1,(R0)+
381 001522  016720  000176'            MOV     N2,(R0)+
382 001526  000207                    RTS     PC
383 001530  052760  040000  177772  SB2:    BIS     #EMXMTV,-6(R0)  ;CREATE MULTI-SUB-ZONE EMX ENTRY
384 001536  056760  000174' 177774          BIS     N1,-4(R0)
385 001544  000207                    RTS     PC
386 001546  016767  000174' 000202' SB3:    MOV     N1,ZN1          ;STORE CURRENT ZONE NIBBLES
387 001554  016767  000176' 000204'         MOV     N2,ZN2
388 001562  116767  000200' 000206'         MOVB    N0,ZN0
389 001570  016720  000206'    ZZZ:    MOV     ZN0,(R0)+       ;CREATE ZONE EMX ENTRY
390 001574  016720  000202'            MOV     ZN1,(R0)+
391 001600  016720  000204'            MOV     ZN2,(R0)+
392 001604  000207                    RTS     PC
393 001606  052760  040000  177772  SB4:    BIS     #EMXMTV,-6(R0)  ;CREATE MULTI-ZONE EMX ENTRY (PRIOR)
394 001614  056760  000202' 177774          BIS     ZN1,-4(R0)
395 001622  016767  000174' 000202' SB5:    MOV     N1,ZN1          ;STORE CURRENT ZONE NIBBLES
396 001630  016767  000176' 000204'         MOV     N2,ZN2
397 001636  116767  000200' 000206'         MOVB    N0,ZN0
398 001644  000207                    RTS     PC
399 001646  052760  040000  177772  SB11:   BIS     #EMXMTV,-6(R0)  ;CREATE MUTI-DOC-TYPE EMX ENTRY (PRIOR)
400 001654  056760  000210' 177774          BIS     TN1,-4(R0)
```

```
 401 001662  000406                        BR      SB9
 402 001664  052760  040000  177772  SB10: BIS     #EMXMTV,-6(R0)  ;CREATE MULTI-ZONE EMX ENTRY (PRIOR)
 403 001672  056760  000202' 177774        BIS     ZN1,-4(R0)
 404 001700  004767  000156          SB9:  JSR     PC,EMXLNK       ;CREATE LINK IN EXPANSION VECTOR TO NEXT
 405 001704  016767  000174' 000210' SB6:  MOV     N1,TN1          ;STORE CURRENT DOC-TYPE NIBBLES
 406 001712  016767  000176' 000212'       MOV     N2,TN2
 407 001720  116767  000200' 000214'       MOVB    N0,TN0
 408 001726  005067  000202'               CLR     ZN1             ;       AND INDICATE "NO ZONE ACTIVE"
 409 001732  016720  000214'          TTT: MOV     TN0,(R0)+       ;CREATE DOC-TYPE EMX ENTRY
 410 001736  016720  000210'               MOV     TN1,(R0)+
 411 001742  016720  000212'               MOV     TN2,(R0)+
 412 001746  000207                        RTS     PC
 413 001750  052760  040000  177772  SB7:  BIS     #EMXMTV,-6(R0)  ;CREATE MULTI-DOC-TYPE EMX ENTRY (PRIOR)
 414 001756  056760  000210' 177774        BIS     TN1,-4(R0)
 415 001764  016767  000174' 000210' SB8:  MOV     N1,TN1          ;STORE CURRENT DOC-TYPE NIBBLES
 416 001772  016767  000176' 000212'       MOV     N2,TN2
 417 002000  116767  000200' 000214'       MOVB    N0,TN0
 418 002006  000207                        RTS     PC
 419 002010  004767  000046          SB12: JSR     PC,EMXLNK       ;CREATE LINK TO NEXT EMX ENTRY
 420 002014  005767  000210'               TST     TN1             ;IF A DOC-TYPE IS ACTIVE
 421 002020  001402                        BEQ     1$
 422 002022  004767  177704                JSR     PC,TTT          ;       THEN CREATE DOC-TYPE EMX ENTRY
 423 002026  005767  000202'          1$:  TST     ZN1             ;IF A ZONE IS ACTIVE
 424 002032  001627                        BEQ     SB1
 425 002034  004767  177530                JSR     PC,ZZZ          ;       THEN CREATE ZONE EMX ENTRY
 426 002040  000624                        BR      SB1             ;CREATE SUB-ZONE EMX ENTRY
 427 002042  004767  000014          SB13: JSR     PC,EMXLNK       ;CREATE LINK TO NEXT EMX ENTRY
 428 002046  005767  000210'               TST     TN1             ;IF A DOC-TYPE IS ACTIVE
 429 002052  001635                        BEQ     SB3
 430 002054  004767  177652                JSR     PC,TTT          ;       THEN CREATE DOC-TYPE EMX ENTRY
 431 002060  000632                        BR      SB3             ;CREATE ZONE EMX ENTRY
 432                                 ;
 433 002062  012720  001000          EMXLNK: MOV   #EMXMCF,(R0)+   ;CREATE MATCH CODE EMX ENTRY
 434 002066  016720  000172'               MOV     FLUIDS,(R0)+
 435 002072  010001                  EMXLK2: MOV   R0,R1           ;R1=EMX OFFSET
 436 002074  162701  000000G               SUB     #EMX,R1
 437 002100  010177  000170'               MOV     R1,@EXPVPT      ;TRANSFER OFFSET TO EXPANSION VECTOR
 438 002104  062767  000002  000170'       ADD     #2,EXPVPT       ;POINT TO NEXT ENTRY IN EXPANSION VECTOR
 439 002112  000207                        RTS     PC
 440                                 ;
 441                                 ;  SUBROUTINE TO GENERATE EMC
 442                                 ;
 443                                 ;               INPUT                           OUTPUT
 444                                 ;       R0=.    SCRATCH                         END OF EMX ADDRESS
 445                                 ;       R1=.    SCRATCH                         SCRATCH
 446                                 ;       R2=.    TYPE OF NODE DESIRED            (SAME)
 447                                 ;       R3=.    INDEX POINTER ADDRESS           END OF POINTERS ADDRESS
 448                                 ;       R4=.    SCRATCH                         0
 449                                 ;       R5=.    #. INDEX POINTERS               0
 450                                 ;
 451 002114  012704  000000G         GENEMC: MOV   #ZERO,R4        ;INITIALIZE (TRIGGER INDEX POINTER REFERENCE)
 452 002120  005067  000156'               CLR     VINXT           ;       RESET NEXT VI ADDRESS
 453 002124  012700  000000G               MOV     #EMX,R0         ;       SET START OF EMX BUFFER
 454 002130  004767  000000G         100$: JSR     PC,GETNDE       ;GET NEXT NODE OF TYPE DESIRED
 455 002134  103434                        BCS     3$              ;NO MORE NODES
 456 002136                                SAVE    R4,R5
 457 002142  016705  000156'               MOV     VINXT,R5        ;MAKE VI ENTRY
```

```
    458 002146  010065  000000G·                   MOV·    R0,VI(R5)
    459 002152· 162765  000000G·000000G·           SUB·    #EMX,VI(R5)
    460 002160  062767  000002  000156'            ADD·    #2,VINXT·
    461 002166  116405  000000G·                   MOVB·   NDSIZ(R4),R5    ;R5->#·TERMS·IN·NODE·
    462·002172· 042705  177400                     BIC·    #177400,R5
    463 002176  006205                             ASR·    R5
    464 002200  062704  000000G·                   ADD·    #NDTRM,R4       ;R4->0·1ST·TERM·
    465 002204  012420                     1$:     MOV·    (R4)+,(R0)+     ;TRANSFER·NODE·ENTRY·TO·EMX·
    466 002206  077502·                            SOB·    R5,1$           ;DO·FOR·ALL·ENTRIES·
    467 002210                                     RESTORE·R4,R5
    468 002214  016410  000000G·                   MOV·    NDFLID(R4),(R0) ;ENTER·MATCH·CODE·
    469 002220  052720  100000                     BIS·    #BIT15,(R0)+
    470 002224  000741                             BR      100$
    471 002226  016701  000156'            3$:     MOV·    VINXT,R1        ;STORE·#·VECTORS·
    472·002232· 006201                             ASR·    R1
    473 002234  010167  000000G·                   MOV·    R1,VILGT·
    474 002240  000207                             RTS·    PC·
    475                                    ;
    476                                            .SBTTL· WRITE·EMX·TO·DISK·
    477                                    ;
    478                                    ;   THE·PURPOSE·OF·THIS·SUBROUTINE·IS·TO·OPEN·THE·EMATRIX·FILE, WRITE·
    479                                    ;   THE·VI·VECTOR·TO·VIRTUAL·BLOCK·1, WRITE·THE·VT·VECTOR·TO·VIRTUAL·
    480                                    ;   BLOCKS·2, 3·AND·4, CALCULATE·THE·NUMBER·OF·BLOCKS·EMX·REQUIRES, WRITE·
    481                                    ;   EMX, CLOSE·THE·FILE·AND·REINITIALIZE THE·EMX,VI·AND·VT·BUFFERS.
    482·                                   ;
    483                                    ;       ON ENTRY,
    484                                    ;       R1 =·FILE·TYPE·
    485                                    ;       R2 =·SCRATCH
    486                                    ;
    487 002242·                            WRTEMX::
    488 002242·                                    SAVE·   R0,R1,R2,R3
    489 002252· 010003                             MOV·    R0,R3                   ;R3->EMX·SIZE·
    490 002254  012700  000012'                    MOV·    #EMXFDB,R0              ;R0->FDB·
    491 002260  004767  000000G·                   JSR·    PC,BLDNFL·              ;BUILD·NEW FILE·FNB·
    492·002264                                     OFNB$W· R0
    493 002276  103010                             BCC·    2$
    494 002300  116001  000052                     MOVB·   F.ERR(R0),R1            ;R1 =·FCS·ERROR·
    495 002304  010167  000000G·                   MOV·    R1,PAR2·
    496 002310                                     CALL·   FCSERR·
    497 002314  000167  000000G·                   JMP·    EXIT
    498                                    ;
    499                                    ;   DETERMINE·BLOCK·LENGTH·OF·EMX·AND· INSERT·IN·EMXLGT·
    500                                    ;
    501 002320                             2$:
    502·002320  004767  000146                     JSR·    PC,ETFDSC·
    503 002324  005002·                            CLR·    R2·
    504 002326  071227  001000                     DIV·    #512.,R2·               ;R2·=·QUOTIENT·
    505                                                                            ;R3 =·REMAINDER·
    506 002332· 005703                             TST·    R3                      ;TEST·REMAINDER·
    507 002334  003401                             BLE·    4$                      ;BRANCH·IF ZERO·
    508 002336  005202·                            INC·    R2·
    509 002340  010267  000000G·           4$:     MOV·    R2,EMXLGT·              ;SAVE·LENGTH·
    510                                    ;
    511                                    ;   WRITE·VI·AND·EMX·TO·OPENED DISK·FILE
    512·                                   ;
    513 002344  012767  000000G·000000'            MOV·    #EMXLGT,PARBUF          ;PAR·1 =·BUFFER·ADDS'
    514 002352· 005202·                            INC·    R2                      ;ADD·BLOCK·LENGTH·OF VI·
```

```
    515 002354  010201                         MOV     R2,R1
    516                                 ;
    517                                 ;   SET END-OF-FILE POINTER IN FDB
    518                                 ;
    519 002356  005060  000010          11$:    CLR     F,EFBK(R0)
    520 002362  010160  000012                  MOV     R1,F.EFBK+2(R0)
    521 002366  005260  000012                  INC     F,EFBK+2(R0)
    522 002372  012702  000200                  MOV     #200,R2         ;ALLOCATED NECCESSARY BLOCKS
    523 002376                                  CALL    .EXTND
    524                                 ;
    525                                 ;
    526                                 ;
    527 002402  070127  001000                  MUL     #512.,R1                ;R1 = BYTE LENGTH OF FILE TO BE WRITTEN
    528 002406  010167  000002'                 MOV     R1,PARBUF+2             ;PAR 2 = FILE LENGTH
    529 002412  012701  000000G                 MOV     #IO.WVB,R1              ;R1 = IO FUNCTION CODE
    530 002416  012702  000005                  MOV     #5,R2                   ;R2 = NO. OF QIO PARAMETERS
    531 002422  012703  000000'                 MOV     #PARBUF,R3              ;R3 -> QIO PARAMETERS
    532 002426                                  CALL    .XQIO
    533 002432  103010                          BCC     5$
    534 002434  116001  000052                  MOVB    F,ERR(R0),R1            ;R1 = IO ERROR
    535 002440  010167  000000G                 MOV     R1,PAR2
    536 002444                                  CALL    FCSERR
    537 002450  000167  000000G                 JMP     EXIT
    538                                 ;
    539                                 ;   CLOSE FILE
    540                                 ;
    541 002454                          5$:     CLOSE$  R0
    542 002460                                  RESTOR  R0,R1,R2,R3
    543 002470  000207                          RTS     PC
    544                                 ;
    545 002472                          ETFDSC: SAVE    R1
    546 002474  162701  000012                  SUB     #FN.EMA,R1      ;GET FDSC AREA ADDRESS
    547 002500  006301                          ASL     R1
    548 002502  006301                          ASL     R1
    549 002504  016702  000166'                 MOV     SAVBST,R2
    550 002510  060102                          ADD     R1,R2
    551 002512  062702  000132                  ADD     #B,FEMA,R2
    552 002516  016722  000114'                 MOV     EMXFDB+F.FNB+N.FID,(R2)+        ;TRANSFER FDSC
    553 002522  016722  000116'                 MOV     EMXFDB+F.FNB+N.FID+2,(R2)+
    554 002526  016722  000132'                 MOV     EMXFDB+F.FNB+N.FVER,(R2)+
    555 002532                                  RESTORE R1
    556 002534  010122                          MOV     R1,(R2)+
    557 002536  000207                          RTS     PC
    558         000001                          .END
```

SYMBOL TABLE

```
ASKOVF= 000145
BATNO=  ****** GX
BITVAL= 000000
BIT0  = 000001
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BLDNFL= ****** GX
BSTPTR= ****** GX
BS.CLS= 000002
BS.DBU= 000004
BS.INA= 000000
BS.OPN= 000001
BS.SRC= 000003
BYTE0 = 000000
BYTE1 = 000001
BYTE2 = 000002
BYTE3 = 000003
BYTE4 = 000004
BYTE5 = 000005
BYTE6 = 000006
BYTE7 = 000007
BYTE8 = 000010
BYTE9 = 000011
BYTVAL= 000012
B.BSTA  000054
B.CNTX  000046
B.CQUQ  000060
B.FEMA  000132
B.FEMB  000142
B.FEMC  000152
B.FFSA  000202
B.FFSB  000212
B.FFSC  000222
B.FMHR  000172
B.FQLS  000162
B.FSAZ  000100
B.FSBZ  000102
B.FSCZ  000104
B.HBLK  000120
B.HDOC  000114
B.HRLP  000126
B.HRLR  000122
B.HRLW  000124
B.NMBR  000052
B.NQRY  000232
B.QLSZ  000106
B.QMAP  000234
B.QSPL  000316
B.QTTM  000076
B.QUQP  000056
B.SFDB  000010
B.SIZE  000772
B.SNDP  000012
B.SSQ   000004
B.SSQF  000050
B.STAT  000044
B.STTE  000053
B.UDOC  000110
CETEMX  000044RG
CET6B   000000RG
CF.B0 = 000070
CF.B2 = 000067
CF.B4 = 000066
CF.B6 = 000065
CF.DR0= 000064
CF.DR1= 000063
CLSEMX  000000RG
COUNT   000222R
CPIXSZ= ****** GX
CWPIDX= ****** GX
DBSLEN= 000116
DECPT = 000056 G
DH.BF0  000002
DH.BF1  000004
DH.CTL  000000
DH.DMC  000010
DH.FLG  000006
DN.DCK  000000
DN.NTP  000004
DN.NXT  000006
DN.ROT  000002
010 DN.SIZ  000010
010 DPEMX   000600RG
010 EMAOVF= 000166
010 EMASIZ  000160R
010 EMBOVF= 000167
010 EMBSIZ  000162R
010 EMCOVF= 000170
010 EMCSIZ  000164R
010 EMX   = ****** GX
010 EMXCHF= 020000 G
010 EMXDTF= 010000 G
010 EMXEXF= 000001 G
010 EMXFDB  000012RG
010 EMXFDC= 002000 G
010 EMXLGT= ****** GX
010 EMXLK2  002072R
010 EMXLNK  002062R
010 EMXMCD= 000002 G
010 EMXMCF= 001000 G
010 EMXMTV= 040000 G
010 EMXNB1= 000002 G
010 EMXNB2= 000004 G
010 EMXNFD= 000400 G
010 EMXNSQ= 100000 G
010 EMXNVD= 001000 G
010 EMXSZF= 002000 G
010 EMXTRL= 010000 G
010 EMXVDC= 004000 G
010 EMXVVV= 000001 G
010 EMXZNF= 004000 G
010 EMX0VD= 000200 G
010 ERRCDE= ****** GX
010 ERRORF= ****** GX
010 ETFDSC  002472R
015 EXIT  = ****** GX
014 EXPVPT  000170R
    FALOVF= 000160
    FCSERR= ****** GX
    FD.FID  000000
    FD.FNB  000006
    FD.FVR  000004
    FD.LEN  000010
017 FD.RAN= ****** GX
016 FD.RWM= ****** GX
    FLDC  = 177775 G
    FLIXSZ= ****** GX
    FLUIDS  000172R
    FLUIDX= ****** GX
005 FMNDCD  001000R
005 FN.DBR  000026
005 FN.DBS  000022
005 FN.DHR  000040
005 FN.EMA  000012
013 FN.EMB  000014
013 FN.EMC  000016
013 FN.FSA  000000
013 FN.FSB  000002
013 FN.FSC  000004
015 FN.LGQ  000034
    FN.LGU  000036
016 FN.MFO  000024
    FN.MHR  000010
016 FN.NMB  000044
    FN.QLS  000006
016 FN.QRY  000020
    FN.SF0  000030
    FN.SF1  000032
    FN.SHD  000042
    FO.WRT= ****** GX
016 F.ACTL= 000076
    F.ALOC= 000040
    F.BBFS= 000062
017 F.BDB = 000070
017 F.BGBC= 000057
    F.BKDN= 000026
    F.BKDS= 000020
    F.BKEF= 000050
    F.BKP1= 000051
    F.BKST= 000024
    F.BKVB= 000064
    F.CHR = 000075
    F.CNTG= 000034
    F.DFNB= 000046
    F.DSPT= 000044
    F.DVNM= 000134
    F.EFBK= 000010
    F.EFN = 000050
    F.EOBB= 000032
    F.ERR = 000052
    F.FACC= 000043
017 F.FFBY= 000014
    F.FNAM= 000110
016 F.FNB = 000102
    F.FTYP= 000116
    F.FVER= 000120
003 F.HIBK= 000004
003 F.LUN = 000042
003 F.MBCT= 000054
003 F.MBC1= 000055
    F.MBFG= 000056
    F.NRBD= 000024
    F.NREC= 000030
    F.OVBS= 000030
016 F.RACC= 000016
    F.RATT= 000001
017 F.RCNM= 000034
011 F.RCTL= 000017
011 F.RSIZ= 000002
011 F.RTYP= 000000
011 F.SEQN= 000100
011 F.SPDV= 000072
011 F.SPUN= 000074
011 F.STBK= 000036
011 F.UNIT= 000136
011 F.URBD= 000020
011 F.VBN = 000064
011 F.VBSZ= 000060
011 GENEMA  000314R
011 GENEMC  002114R
011 GENFMD  000644R
011 GETNDE= ****** GX
011 INTERW  000036RG
011 INTRWD= 177777 G
011 IOST    000152R
011 IO.WVB= ****** GX
    JSB1  = 000000
    JSB10 = 000022
    JSB11 = 000024
    JSB12 = 000026
    JSB13 = 000030
    JSB2  = 000002
    JSB3  = 000004
    JSB4  = 000006
    JSB5  = 000010
    JSB6  = 000012
    JSB7  = 000014
    JSB8  = 000016
    JSB9  = 000020
    LNPOVF= 000146
    LUNFIL= 000001
    M     = 000062
    MINEMX  000212RG    015
    N     = 000002
    NDBOVF= 000175
    NDCHR = ****** GX
    NDFLID= ****** GX
    NDFMEN= ****** GX
    NDFMN = ****** GX
    NDFMS = ****** GX
    NDFSAA= ****** GX
    NDFSAB= ****** GX
    NDSIZ = ****** GX
    NDTRM = ****** GX
    NFEMX   000006RG    015
    NFLDC = 177773 G
    NODEA = ****** GX
    NODEB = ****** GX
    NODEC = ****** GX
    NUMRNG= ****** GX
    NVDEMX  000014RG    015
    NVLDC = 177774 G
    N.BFAC= 000004
    N.BHGH= 000006
    N.BTCH= 000004
    N.BUFB= 004000
    N.BUFW= 002000
    N.DID = 000024
    N.DVNM= 000032
    N.FID = 000000
    N.FNAM= 000006
    N.FOS = 000764
    N.FTYP= 000014
    N.FVER= 000016
    N.NEXT= 000022
017 N.PKSZ= 000020
017 N.PKTS= 000043
017 N.QURY= 000031
    N.STAT= 000020
015 N.SUNT= 000002
    N.UNIT= 000034
016 N0      000200R     016
    N1      000174R     016
    N2      000176R     016
    PARBUF  000000RG    016
    PAR$$$= 000000
    PAR2  = ****** GX
    POLOVF= 000157
    PRIORS  000216R     016
    PRIORV  000220R     016
    QBFOVF= 000150
    QEXOVF= 000165
```

SYMBOL TABLE

```
QE.DWI=  000001          SB12    002010R  017 SR.YR    000004  002 S.FTYP=  000002          WORD6 =  000014
QE.DW2=  000003          SB13    002042R  017 SR.IIN   000024  002 S.HRL =  000240          WORD7 =  000016
QE.FTB=  000026          SB2     001530R  017 SR.IIP   000016  002 S.NFEN=  000020          WORD8 =  000020
QE.IHS=  000005          SB3     001546R  017 SSQ     =******GX      S2    =  000010          WORD9 =  000022
QE.ILC=  000004          SB4     001606R  017 SS.FID   000002  004 S3    =  000020          WRDVAL=  000024
QE.INQ=  000015          SB5     001622R  017 SS.FNB   000010  004 S4    =  000030          WRTEMX   002242RG    017
QE.ISQ=  000016          SB6     001704R  017 SS.FVR   000006  004 S5    =  000040          XBATCH=  000013
QE.MFL=  000013          SB7     001750R  017 SS.LEN   000012  004 S6    =  000050          XDBLOA=  000004
QE.MFM=  000010          SB8     001764R  017 SS.STT   000000  004 TDBOVF=  000174          XDBPRO=  000012
QE.MNT=  000012          SB9     001700R  017 STABLE    000226R 016 TN0       000214R    016 XDMCIN=  000006
QE.MOP=  000007          SEGOP = 177770 G         START3=******GX      TN1       000210R    016 XFOSMR=  000007
QE.MSD=  000011          SLBOVF=  000164          ST.ASZ   000020  006 TN2       000212R    016 XGTSRE=  000014
QE.MTS=  000002          SR.ARE   000114  002 ST.BSZ   000024  006 TSKOVF=  000147              XHITSK=  000011
QE.NLQ=  000006          SR.ARS   000106  002 ST.BTC   000000  006 TTBOVF=  000161              XHLMER=  000002
QE.NOS=  000025          SR.DAY   000010  002 ST.CS2   000030  006 TTT       001732R    017 XHOTSK=  000010
QE.NRB=  000024          SR.DLT   000014  002 ST.HRL   000010  006 VDCFLG    000224R    016 XMSCHE=  000000
QE.NRI=  000023          SR.ECB   000047  002 ST.LEN   000044  006 VI      =******GX        XQTS  =  000003
QE.NTK=  000027          SR.ECH   000046  002 ST.QRY   000002  006 VILGT   =******GX        XQT0  =  000001
QE.PX1=  000017          SR.ECL   000050  002 ST.QSZ   000034  006 VINXT     000156RG   016 XSULOA=  000005
QE.PX2=  000020          SR.FIB   000012  002 ST.SCH   000040  006 VI10VF=  000173              XTBOVF=  000162
QE.PX3=  000021          SR.GRE   000100  002 ST.UHL   000004  006 VI20VF=  000172              ZERO  =******GX
QE.PX4=  000022          SR.GRS   000072  002 ST.XLT   000014  006 VI30VF=  000171              ZN0       000206R    016
QE.RO1=  000144          SR.LEN   000122  002 SU.DBU=  000004      VLDC  =  177776 G            ZN1       000202R    016
QE.UBP=  000014          SR.LIN   000066  002 SU.DON=  000006      WN.NTP    000004    012 ZN2       000204R    016
QLBOVF=  000163          SR.LIP   000062  002 SU.IDL=  000000      WN.NXT    000006    012 ZVDEMX    000000RG   015
QRYOVF=  000156          SR.MON   000006  002 SU.LOD=  000001      WN.ROT    000002    012 ZVLDC =  177772 G
QTSTAT=******GX          SR.NDC   000042  002 SU.SRC=  000002      WN.SIZ    000010    012 ZZZ       001570R    017
Q.FDSC   000004    007   SR.NDS   000036  002 SU.SRR=  000005      WN.SRC    000000    012 .CLOSE=  ******G
Q.NQBK   000000    007   SR.NIN   000030  002 SU.XPD=  000003      WN.TYP    000001    012 .EXTND=  ******GX
Q.NUHL   000002    007   SR.NIP   000022  002 S.FATT=  000016      WORD0 =  000000              .OPFNB=  ******G
Q.SIZE   000014    007   SR.SDB   000032  002 S.FDB =  000140      WORD1 =  000002              .XQIO =  ******GX
SAVBST   000166R   016   SR.SRC   000002  002 S.FNAM=  000006      WORD2 =  000004              ...PC1=  000012R    016
SB1      001512R   017   SR.SUN   000000  002 S.FNB =  000036      WORD3 =  000006              ...PC2=  000152R    016
SB10     001664R   017   SR.TWS   000056  002 S.FNBW=  000017      WORD4 =  000010              ...TPC=  000140
SB11     001646R   017   SR.WSL   000052  002 S.FNTY=  000004      WORD5 =  000012

. ABS.   000000    000
         000000    001
SRCOFF   000122    002
FDSCOF   000010    003
SUSOFF   000012    004
DHROFF   000012    005
STTOFF   000044    006
QSPLOF   000014    007
BSTOFF   000772    010
FNOFFS   000044    011
WNODOF   000010    012
DNODOF   000010    013
CCET     000200    014
EMX      000644    015
EMXDAT   000306    016
EMXCDE   002540    017
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  7788 WORDS (31 PAGES)
DYNAMIC MEMORY:  9140 WORDS ( 35 PAGES)
ELAPSED TIME:  00:01:01
EMXCDE,EMXCDE/-SP/NL:ME:BEX=[20,1]JP[illegible]
```

.MAIN. MAC M1110 27-MAR-80 13:17
TABLE OF CONTENTS

12- 1 PRTCDE

```
                        ;
                        ; MACRO TO PRINT BUFFER
                        ; CREATES SPOOL FILE RECORDS
                        ;
                        .MACRO  PRT,START,END
                                MOV     R4,-(SP)
                                MOV     START,R4
                                MOV     R4,LOCAT
                                MOV     END,ENDLOC
                                JSR     PC,PRINT
                                MOV     (SP)+,R4

                        .ENDM
                        .MACRO  PRTH,START,END
                                MOV     R4,-(SP)
                                MOV     START,R4
                                MOV     R4,LOCAT
                                MOV     END,ENDLOC
                                JSR     PC,PRNTH
                                MOV     (SP)+,R4
                        .ENDM
                        ;
                        ; MACRO TO TEST CHARACTER SIGNIFICANCE
                        ; PARAMETERS ARE CONDITION,TRUE,AND FALSE
                        ;       CONDITION=THE TEST CONDITION
                        ;       TRUE      =PATH TO TAKE IF CONDITION TRUE
                        ;       FALSE     =PATH TO TAKE IF CONDITION FALSE
                        ; ANY PARAMETER CAN BE PROCEEDED BY "!" WHICH CAUSES THAT
                        ;  PARAMETER TO BE A SUBROUTINE CALL
                        ;
```

```
                                ;
                                TYPE1B==0       ;1 BYTE------SIZE OF NUMBER
                                TYPE2B==2       ;2 BYTES
                                TYPE3B==4       ;3 BYTES
                                TYPE4B==6       ;4 BYTES
                                SKPZER==1000    ;SKIP LEADING ZERO'S
                                BLKZER==400     ;BLANK LEADING ZERO'S
                                ;
                                ;
000000  010046                  DEC2:   MOV R0,-(SP)    ;SAVE REGISTERS
000002  010146                          MOV R1,-(SP)
000004  112401                          MOVB (R4)+,R1   ;GET BYTE CONTAINING 2 DIGITS
000006  042701  177600                  BIC #177600,R1  ;IF MORE THAN 2 DIGITS, CLEAR GARBAGE OFF
000012  005000                          CLR R0
000014  071027  000012                  DIV #10.,R0     ;1ST DIGIT IN R0, 2RD DIGIT IN R1
000020  052700  000060                  BIS #60,R0      ;MAKE ASCII AND TRANSFER
000024  052701  000060                  BIS #60,R1
000030  110023                          MOVB R0,(R3)+
000032  110123                          MOVB R1,(R3)+
000034  012601                          MOV (SP)+,R1    ;RESTORE REGISTERS
000036  012600                          MOV (SP)+,R0
000040  000207                          RTS PC
                                ;
000042  010046                  DECASC: MOV R0,-(SP)    ;SAVE REGISTERS
000044  010146                          MOV R1,-(SP)
000046  010246                          MOV R2,-(SP)
000050  010546                          MOV R5,-(SP)
000052  005046                          CLR -(SP)       ;NOT A LEADING ZERO INDICATOR
000054  012702  000144'                 MOV #100$,R2    ;SET UP DIVISOR TABLE
000060  012401                          MOV (R4)+,R1    ;GET WORD TO BE CONVERTED
000062  012705  000004                  MOV #4,R5       ;SET LOOP COUNT
000066  005000                  1$:     CLR R0
000070  071022                          DIV (R2)+,R0    ;R0=DIGIT, R1=REMAINDER
000072  005700                          TST R0          ;IF 0, SEE IF LEADING 0
000074  001005                          BNE 11$         ; NOT A ZERO
000076  005716                          TST (SP)
000100  001003                          BNE 11$         ; NOT A LEADING ZERO
000102  112723  000040                  MOVB #40,(R3)+  ;XFER A SPACE
000106  000404                          BR 12$
000110  052700  000060          11$:    BIS #60,R0      ;CONVERT TO ASCII
000114  110023                          MOVB R0,(R3)+   ;XFER
000116  005216                          INC (SP)        ;SET "NOT-A-LEADING-0" INDICATOR
000120  077516                  12$:    SOB R5,1$
000122  052701  000060                  BIS #60,R1      ;XFER LAST DIGIT
000126  110123                          MOVB R1,(R3)+
000130  005726                          TST (SP)+       ;CLEAN UP STACK AND RETURN
000132  012605                          MOV (SP)+,R5
000134  012602                          MOV (SP)+,R2
000136  012601                          MOV (SP)+,R1
000140  012600                          MOV (SP)+,R0
000142  000207                          RTS PC
000144  023420                  100$:   10000.  ;DIVISORS
000146  001750                          1000.
000150  000144                          100.
000152  000012                          10.
000154  010546                  RAD50:  MOV R5,-(SP)    ;INITIALIZE
000156  010146                          MOV R1,-(SP)
```

```
 58 000160  010046                     MOV R0,-(SP)
 59 000162  005000                     CLR R0
 60 000164  012401                     MOV (R4)+,R1    ;GET RAD50 WORD
 61 000166  071027  003100             DIV #3100,R0    ;BREAK IT UP
 62 000172  010005                     MOV R0,R5       ;  1ST CHAR IN R5
 63 000174  005000                     CLR R0
 64 000176  071027  000050             DIV #50,R0      ;  2RD CHAR IN R0 , 3RD CHAR IN R1
 65 000202  004767  000024             JSR PC,RADASC   ;CONVERT 1ST CHAR
 66 000206  010005                     MOV R0,R5
 67 000210  004767  000016             JSR PC,RADASC   ;CONVERT 2RD CHAR
 68 000214  010105                     MOV R1,R5
 69 000216  004767  000010             JSR PC,RADASC   ;CONVERT 3RD CHAR
 70 000222  012600                     MOV (SP)+,R0    ;CLEAN UP
 71 000224  012601                     MOV (SP)+,R1
 72 000226  012605                     MOV (SP)+,R5
 73 000230  000207                     RTS PC
 74 000232  120527  000033     RADASC: CMPB R5,#33     ;?WHAT IS THE CHAR?
 75 000236  001416                     BEQ 2$          ;IT IS A "$"
 76 000240  002410                     BLT 1$
 77 000242  120527  000035             CMPB R5,#35
 78 000246  001415                     BEQ 3$          ;IT IS "ILLEGAL"
 79 000250  002417                     BLT 4$          ;IT IS A "."
 80 000252  062705  000022             ADD #22,R5      ;IT IS A "NUMBER" ;TRANSFER "NUMBER"
 81 000256  110523                     MOVB R5,(R3)+
 82 000260  000207                     RTS PC
 83 000262  005705             1$:     TST R5
 84 000264  003014                     BGT 5$          ;IT IS A "LETTER"
 85 000266  112723  000040             MOVB #' ,(R3)+  ;IT IS A " "       ;TRANSFER " "
 86 000272  000207                     RTS PC
 87 000274  112723  000044     2$:     MOVB #'$,(R3)+                     ;TRANSFER "$"
 88 000300  000207                     RTS PC
 89 000302  112723  000077     3$:     MOVB #'?,(R3)+                     ;TRANSFER "?"
 90 000306  000207                     RTS PC
 91 000310  112723  000056     4$:     MOVB #'.,(R3)+                     ;TRANSFER "."
 92 000314  000207                     RTS PC
 93 000316  062705  000100     5$:     ADD #100,R5                        ;TRANSFER "LETTER"
 94 000322  110523                     MOVB R5,(R3)+
 95 000324  000207                     RTS PC
 96 000326  010046             OCTASC::MOV R0,-(SP)    ;INITIALIZE
 97 000330  010246                     MOV R2,-(SP)
 98 000332  010546                     MOV R5,-(SP)
 99 000334  010146                     MOV R1,-(SP)
100 000336  010305                     MOV R3,R5
101 000340  010102                     MOV R1,R2
102 000342  072227  177771             ASH #-7.,R2
103 000346  010246                     MOV R2,-(SP)
104 000350  042701  001400             BIC #1400,R1
105 000354  005002                     CLR R2
106 000356  020127  000006             CMP R1,#TYPE4B  ;?TYPE IN RANGE?
107 000362  003407                     BLE OCTGO       ;YES
108 000364  005726                     TST (SP)+       ;NO:CLEAN UP
109 000366  012601                     MOV (SP)+,R1
110 000370  012605                     MOV (SP)+,R5
111 000372  012602                     MOV (SP)+,R2
112 000374  012600                     MOV (SP)+,R0
113 000376  000261                     SEC             ;ERROR RETURN
114 000400  000207                     RTS PC
```

```
 115 000402  000171  000406'          OCTGO:  JMP @1$(R1)      ;DISPATCH TO APPROPRIATE SET UP.
 116 000406  000416'                  1$:     10$              ;TYPE1B.
 117 000410  000476'                          12$              ;TYPE2B.
 118 000412  000436'                          13$              ;TYPE3B.
 119 000414  000460'                          11$              ;TYPE4B.
 120 000416  112403                   10$:    MOVB (R4)+,R3    ;GET # *BYTE*
 121 000420  042703  177400                   BIC #177400,R3   ;A -BYTE WILL PUT GARBAGE IN UPPER BYTE.
 122 000424  073227  000012                   ASHC #10.,R2     ;INITIAL POSITIONING.
 123 000430  012700  000003           100$:   MOV #3,R0        ;SET LOOP COUNT.
 124 000434  000425                          BR 121$
 125 000436  116403  177776           13$:    MOVB -2(R4),R3   ;GET UPPER BYTE.
 126 000442  042703  177400                   BIC #177400,R3   ;A -BYTE WILL PUT GARBAGE IN UPPER BYTE.
 127 000446  073227  000013                   ASHC #11.,R2     ;INITIAL POSITIONING.
 128 000452  062701  000004                   ADD #4,R1        ;INDICATE SECOND PASS NEEDED.
 129 000456  000764                          BR 100$
 130 000460  016403  177776           11$:    MOV -2(R4),R3    ;GET UPPER WORD OF # *DOUBLE WORD*
 131 000464  073227  000002                   ASHC #2,R2       ;INITIAL POSITIONING.
 132 000470  062701  000004                   ADD #4,R1        ;INDICATE SECOND PASS NEEDED.
 133 000474  000403                          BR 120$
 134 000476  012403                   12$:    MOV (R4)+,R3     ;GET # *WORD*
 135 000500  073227  000001                   ASHC #1,R2       ;INITIAL POSITIONING.
 136 000504  012700  000006           120$:   MOV #6,R0        ;SET LOOP COUNT.
 137 000510  052702  000060           121$:   BIS #60,R2       ;MAKE ASCII.
 138 000514  005716                   20$:    TST (SP)         ;FORMAT CHECK NEEDED?.
 139 000516  001034                          BNE 3$           ;YES.
 140 000520  110225                   21$:    MOVB R2,(R5)+    ;TRANSFER.
 141 000522  012702  000006           22$:    MOV #6,R2        ;SET UP FOR NEXT TRANSFER.
 142 000526  073227  000003                   ASHC #3,R2.
 143 000532  077010                          SOB R0,20$       ;?FINISHED?--NO.
 144 000534  022701  000006                   CMP #TYPE4B,R1   ;YES.
 145 000540  002436                          BLT 33$
 146 000542  012601                          MOV (SP)+,R1
 147 000544  000171  000550'                 JMP @23$(R1)     ;DISPATCH TO HANDLE "ALL ZERO" CONTENTION.
 148 000550  000572'                  23$:    232$
 149 000552  000556'                          230$
 150 000554  000566'                          231$
 151 000556  112765  000060  177777  230$:    MOVB #60,-1(R5)  ;REPLACE LAST BLANK WITH "0"
 152 000564  000402                          BR 232$
 153 000566  112725  000060           231$:   MOVB #60,(R5)+   ;TRANSFER LONE "0"
 154 000572  010503                   232$:   MOV R5,R3        ;CLEAN UP $ RETURN.
 155 000574  012601                          MOV (SP)+,R1
 156 000576  012605                          MOV (SP)+,R5
 157 000600  012602                          MOV (SP)+,R2
 158 000602  012600                          MOV (SP)+,R0
 159 000604  000241                          CLC
 160 000606  000207                          RTS PC
 161 000610  120227  000060           3$:     CMPB R2,#'0      ;?CHARACTER A "0"?.
 162 000614  001006                          BNE 30$          ;NO.
 163 000616  022716  000004                   CMP #4,(SP)      ;YES:?WHAT'S THE FORMAT?.
 164 000622  001737                          BEQ 22$          ;SKIP LEADING ZERO'S.
 165 000624  112725  000040                   MOVB #40,(R5)+   ;BLANK LEADING ZERO'S.
 166 000630  000734                          BR 22$
 167 000632  005016                   30$:    CLR (SP)         ;FORMAT CHECK NO LONGER NEEDED.
 168 000634  000731                          BR 21$
 169 000636  005002                   33$:    CLR R2           ;CLEAR CHARACTER REGISTER.
 170 000640  162701  000004                   SUB #4,R1        ;SECOND PASS UNDERWAY,DON'T DO IT AGAIN.
 171 000644  012403                          MOV (R4)+,R3     ;GET LOWER WORD OF #.
```

```
000646  100023                          BPL 331$          ;IF BIT 15=0 NO CORRECTION NEEDED
000650  042703  100000                   BIC #100000,R3    ;CORRECT NEXT CHARACTER
000654  011600                          MOV (SP),R0       ;CORRECT LAST CHARACTER VIA DISPATCH
000656  005016                          CLR (SP)          ;FORMAT CHECK NO LONGER NECESSARY
000660  000170  000664'                 JMP @330$(R0)
000664  000672'             330$:      3301$
000666  000702'                        3302$
000670  000712'                        3303$
000672  152765  000001  177777  3301$: BISB #1,-1(R5)    ;IF PASS ALL MODE,SET LEAST SIGNIFICANT BIT
000700  000406                         BR 331$
000702  112765  000061  177777  3302$: MOVB #61,-1(R5)   ;IF BLKZER, CONVERT TO "1"
000710  000402                         BR 331$
000712  112725  000061          3303$: MOVB #61,(R5)+     ;IF SKPZER,TRANSFER A "1"
000716  005724                  331$:  TST (R4)+          ;CORRECT POSITION OF NEXT #
000720  073227  000004                 ASHC #4,R2        ;INITIAL POSITIONING
000724  012700  000005                 MOV #5,R0         ;SET LOOP COUNT
000730  000667                         BR 121$
```

```
                                        .SBTTL  PRTCDE
                                        .PSECT  PRTCDE
                                        .MCALL  FDBDF$,FDAT$A,FDRC$A,FDOP$A,FDBF$A,NMBLK$,PUT$,FSRSZ$
                    000004              LTLUN==4
000000      104     125     115     HEADER::.ASCII /DUMP OF DATA IN QT0/
                    000023              HDRSIZ==.-HEADER
000023      106     114     125     FLUHDR: .ASCII  /FLU NODE SUMMARY/
                    000020              FHSIZ=.-FLUHDR
000043      106     114     125     FLUHD2: .ASCII  /FLU ID=FLU/
                    000012              FH2SIZ=.-FLUHD2
000055      102     101     124     BSMSG:  .ASCII  /BATCH SUMMARY/
                    000015              BSSIZ=.-BSMSG
000072      121     122     131     BS2MSG: .ASCII  /QRY ID=EQID/
                    000013              BS2SIZ=.-BS2MSG
000105      121     114     123     QLSHD1: .ASCII  /QLS SUMMARY/
                    000013              QLSIZ1=.-QLSHD1
000120      124     124     101     QLSHD2: .ASCII  /TTABLE AND XTABLE (FLU ID = LOGIC ADDRESSES)/
                    000054              QLSIZ2=.-QLSHD2
000174      121     114     102     QLSHD3: .ASCII  /QLB (ADDRESS:ENTRIES->QUERY ID)/
                    000037              QLSIZ3=.-QLSHD3
000233      123     104     114     QLSHD4: .ASCII  /SDLB (ADDRESS:ENTRIES->POINTER)/
                    000037              QLSIZ4=.-QLSHD4
000272      121     105     130     QLSHD5: .ASCII  /QEX (ADDRESS: RANGE-UNIT(FORWARD UNIT)FIRST FLAG)/
                    000061              QLSIZ5=.-QLSHD5
000353      121     103     114     QLSHD6: .ASCII  /QCL AND FAL (QUERY ID = FLU'S)/
                    000036              QLSIZ6=.-QLSHD6
000411      040                 NEWLIN: .BYTE   40
                                        .EVEN
                                        FSRSZ$  1,,PRTCDE
                                LIST::  FDBDF$
                                        FDAT$A R,VAR,FD,CR,BSIZ
                                        FDRC$A ,PRTBUF,BSIZ
                                        FDOP$A LTLUN,,LTNB
                                        FDBF$A ,,,1
                                LTNB:   NMBLK$ QT0SUM,PRT,0,SY,0
000610                          PRTBUF::.BLKB   120.
                    000170              BSIZ==.-PRTBUF
                                        .EVEN
001000      101     102     103 AT::    .ASCII /ABCDEFGHIJKLMNOPQRSTUVWXYZ/
001032  000000                  COUNT:  .WORD   0       ;QRY ID PRINT STORAGE
                                ;
                                ;  QLS SUMMARY FORMAT SUBROUTINE
                                ;
001034                          QLSSUM::
001034                                  PUT$    #LIST,#NEWLIN,#1        ;NEW LINE
001060                                  PUT$    ,#QLSHD1,#QLSIZ1        ;"BATCH SUMMARY"
001100                                  PUT$    ,#QLSHD6,#QLSIZ6        ;QCL/FAL HEADER
001120  012702  000000G                 MOV     #QCL,R2         ;SET UP TO FORMAT QCL/FAL SUMMARY
001124  012701  001002                  MOV     #TYPE2B!SKPZER,R1
001130  005067  177676                  CLR     COUNT           ;COUNT<-QUERY ID
001134  026767  177672  000000G 1$:     CMP     COUNT,QRYMAX    ;FINISHED SUMMARY?
001142  101045                          BHI     100$            ;YES
001144  012204                          MOV     (R2)+,R4        ;R4<-FAL ENTRY ADDRESS
001146  042704  100000                  BIC     #100000,R4      ;  REMOVE UHL FLAG
001152  062704  000000G                 ADD     #FAL,R4
001156  012405                          MOV     (R4)+,R5        ;R5<-# FAL ENTRIES
001160  001433                          BEQ     3$              ;IF NO ENTRIES
```

```
    58 001162  010400                          MOV     R4,R0           ;SAVE R4
    59 001164  012704  001032'                 MOV     #COUNT,R4       ;CONVERT AND TRANSFER QUERY ID
    60 001170  012703  000610'                 MOV     #PRTBUF,R3
    61 001174  004767  000326'                 JSR     PC,OCTASC
    62 001200  010004                          MOV     R0,R4           ;RESTORE R4
    63 001202  112723  000075                  MOVB    #'=,(R3)+       ;TRANSFER "="
    64 001206  004767  000326'          2$:    JSR     PC,OCTASC       ;CONVERT AND TRANSFER FLU ID (BIT VECTOR)
    65 001212  112723  000054                  MOVB    #',,(R3)+       ;TRANSFER ","
    66 001216  077505                          SOB     R5,2$           ;DO ALL ENTRIES
    67 001220  005303                          DEC     R3              ;REMOVE LAST ","
    68 001222  162703  000610'                 SUB     #PRTBUF,R3      ;PRINT LINE
    69 001226                                  PUT$    #LIST,#PRTBUF,R3
    70 001250  005267  177556          3$:    INC     COUNT           ;CONTINUE WITH NEXT QUERY
    71 001254  000727                          BR      1$
    72 001256  012767  177777  177546  100$:  MOV     #-1,COUNT       ;COUNT = CURRENT FLU ID
    73 001264                                  PUT$    #LIST,#NEWLIN,#1        ;NEW LINE
    74 001310                                  PUT$    ,#QLSHD2,#QLSIZ2        ;PRINT HEADER
    75 001330  012702  000000G                 MOV     #TTABLE,R2      ;SET UP TO FORMAT TTABLE/XTABLE SUMMARY
    76 001334  012701  001002                  MOV     #TYPE2B!SKPZER,R1
    77 001340  005267  177466          101$:  INC     COUNT           ;STEP TO NEXT FLU ID
    78 001344  012703  000610'                 MOV     #PRTBUF,R3      ;RESET TO BEGINNING OF PRINT BUFFER
    79 001350  026767  177456  000000G         CMP     COUNT,FLUID     ;FINISHED SUMMARY?
    80 001356  101063                          BHI     200$            ;YES
    81 001360  012704  001032'                 MOV     #COUNT,R4       ;NO: CONVERT AND TRANSFER 'FLUID='
    82 001364  004767  000326'                 JSR     PC,OCTASC
    83 001370  112723  000075                  MOVB    #'=,(R3)+
    84 001374  012204                          MOV     (R2)+,R4        ;R4<-XTABLE ADDRESS
    85 001376  062704  000000G                 ADD     #XTABLE,R4
    86 001402  012205                          MOV     (R2)+,R5        ;R5<-# XTABLE ENTRIES
    87 001404  001755                          BEQ     101$            ;IF NONE, THEN STEP TO NEXT FLU
    88 001406  020327  000771'          102$:  CMP     R3,#PRTBUF+113. ;NEAR END OF BUFFER?
    89 001412  103022                          BHIS    103$            ;YES
    90 001414  004767  000326'          121$:  JSR     PC,OCTASC       ;CONVERT AND TRANSFER XTABLE ENTRY
    91 001420  112723  000054                  MOVB    #',,(R3)+       ;TRANSFER ","
    92 001424  077510                          SOB     R5,102$         ;DO FOR ALL XTABLE ENTRIES
    93 001426  005303                          DEC     R3              ;REMOVE LAST ","
    94 001430  162703  000610'                 SUB     #PRTBUF,R3      ;PRINT LINE
    95 001434                                  PUT$    #LIST,#PRTBUF,R3
    96 001456  000730                          BR      101$
    97 001460  162703  000610'          103$:  SUB     #PRTBUF,R3      ;PRINT LINE
    98 001464                                  PUT$    #LIST,#PRTBUF,R3
    99 001506  012703  000610'                 MOV     #PRTBUF,R3      ;RESET TO BEGINNING OF PRINT BUFFER
   100 001512  012700  000006                  MOV     #6,R0           ;BLANK OUT FLU ID FIELD
   101 001516  112723  000040          131$:  MOVB    #40,(R3)+
   102 001522  077003                          SOB     R0,131$
   103 001524  000733                          BR      121$
   104 001526                          200$:  PUT$    #LIST,#NEWLIN,#1        ;NEW LINE
   105 001552                                  PUT$    ,#QLSHD3,#QLSIZ3        ;START QLS SUMMARY
   106 001572  012704  000000G                 MOV     #QLB,R4         ;R4<-START OF QLB
   107 001576  005067  177230                  CLR     COUNT           ;COUNT<-QLB ADDRESS
   108 001602  012701  001000                  MOV     #TYPE1B!SKPZER,R1
   109 001606  012703  000610'          201$:  MOV     #PRTBUF,R3      ;RESET TO BEGINNING OF PRINT BUFFER
   110 001612  010400                          MOV     R4,R0           ;SAVE R4
   111 001614  012704  001032'                 MOV     #COUNT,R4       ;PRINT ADDRESS
   112 001620  004767  000326'                 JSR     PC,OCTASC
   113 001624  010004                          MOV     R0,R4           ;RESTORE R4
   114 001626  112723  000072                  MOVB    #':,(R3)+       ;TRANSFER ": "
```

```
115 001632  112723  000040                      MOVB    #40,(R3)+
116 001636  112405                              MOVB    (R4)+,R5        ;R5<-#ENTRIES IN THIS QEL
117 001640  004767  000326'          202$:      JSR     PC,OCTASC       ;CONVERT AND TRANSFER ENTRY
118 001644  112723  000054                      MOVB    #',,(R3)+       ;TRANSFER ","
119 001650  077505                              SOB     R5,202$         ;DO FOR ALL ENTIES
120 001652  005303                              DEC     R3              ;REMOVE LAST ","
121 001654  112723  000055                      MOVB    #'-,(R3)+       ;TRANSFER "->"
122 001660  112723  000076                      MOVB    #'>,(R3)+
123 001664  004767  000326'                     JSR     PC,OCTASC       ;CONVERT AND TRANSFER QUERY ID
124 001670  162703  000610'                     SUB     #PRTBUF,R3      ;PRINT LINE
125 001674                                      PUT$    #LIST,#PRTBUF,R3
126 001716  010402                             MOV     R4,R2           ;UPDATE QLB ADDRESS
127 001720  162702  000000G                     SUB     #QLB,R2
128 001724  010267  177102                      MOV     R2,COUNT
129 001730  020267  000000G                     CMP     R2,QLBAD        ;AT END OF QLB?
130 001734  103724                            BLO     201$            ;NO
131 001736  022767  037775  000000G            CMP     #40000-3.SDLBAD ;ANY SDLB ENTRIES?
132 001744  103114                            BHIS    400$            ;NO
133 001746                                      PUT$    #LIST,#NEWLIN,#1        ;NEW LINE
134 001772                                      PUT$    ,#QLSHD4,#QLSIZ4        ;START SDLB SUMMARY
135 002012  012704  000000G                     MOV     #SDLB,R4        ;R4<-START OF SDLB
136 002016  012767  040000  177006             MOV     #40000,COUNT    ;COUNT<-SDLB ADDRESS
137 002024  012701  001002                      MOV     #TYPE2B!SKPZER,R1
138 002030  012703  000610'          301$:      MOV     #PRTBUF,R3      ;RESET TO BEGINNING OF PRINT BUFFER
139 002034  010400                              MOV     R4,R0           ;SAVE R4
140 002036  012704  001032'                     MOV     #COUNT,R4       ;PRINT ADDRESS
141 002042  004767  000326'                     JSR     PC,OCTASC
142 002046  010004                              MOV     R0,R4           ;RESTORE R4
143 002050  112723  000072                      MOVB    #':,(R3)+       ;TRANSFER ": "
144 002054  112723  000040                      MOVB    #40,(R3)+
145 002060  112405                              MOVB    (R4)+,R5        ;R5<-#ENTRIES IN THIS SDEL
146 002062  012701  001000                      MOV     #TYPE1B!SKPZER,R1
147 002066  004767  000326'          302$:      JSR     PC,OCTASC       ;CONVERT AND TRANSFER ENTRY
148 002072  112723  000054                      MOVB    #',,(R3)+       ;TRANSFER ","
149 002076  077505                              SOB     R5,302$         ;DO FOR ALL ENTRIES
150 002100  005303                              DEC     R3              ;REMOVE LAST ","
151 002102  112723  000055                      MOVB    #'-,(R3)+       ;TRANSFER "->"
152 002106  112723  000076                      MOVB    #'>,(R3)+
153 002112  005204                              INC     R4              ;ADJUST FOR POSSIBLE PAD BYTE
154 002114  042704  000001                      BIC     #1,R4
155 002120  012701  001002                      MOV     #TYPE2B!SKPZER,R1
156 002124  004767  000326'                     JSR     PC,OCTASC       ;CONVERT AND TRANSFER POINTER
157 002130  162703  000610'                     SUB     #PRTBUF,R3      ;PRINT LINE
158 002134                                      PUT$    #LIST,#PRTBUF,R3
159 002156  010402                             MOV     R4,R2           ;UPDATE SDLB ADDRESS
160 002160  162702  140000G                     SUB     #SDLB-40000,R2
161 002164  010267  176642                      MOV     R2,COUNT
162 002170  020267  000000G                     CMP     R2,SDLBAD       ;AT END OF SDLB?
163 002174  103715                            BLO     301$            ;NO
164 002176  022767  100003  000000G  400$:     CMP     #100003,QEXAD   ;ANY QEX ENTRIES?
165 002204  103402                            BLO     401$            ;YES
166 002206  000167  000376                     JMP     500$            ;NO
167 002212                           401$:      PUT$    #LIST,#NEWLIN,#1        ;NEW LINE
168 002236                                      PUT$    #LIST,#QLSHD5,#QLSIZ5   ;START QEX SUMMARY
169 002262  012702  000000G                     MOV     #QEX,R2         ;R2<-START OF QEX
170 002266  012767  100000  176536             MOV     #100000,COUNT   ;COUNT<-QEX ADDRESS
171 002274  012703  000610'          444$:      MOV     #PRTBUF,R3      ;RESET TO BEGINNING OF PRINT BUFFER
```

```
172 002300  012701  000002                  MOV     #TYPE2B,R1
173 002304  012704  001032'                 MOV     #COUNT,R4       ;CONVERT AND TRANSFER QEX ADDRESS
174 002310  004767  000326'                 JSR     PC,OCTASC
175 002314  112723  000072                  MOVB    #':,(R3)+       ;TRANSFER ": "
176 002320  112723  000040                  MOVB    #40,(R3)+
177 002324  010300                  410$:   MOV     R3,R0           ;R0<--END OF THIS NEW FIELD
178 002326  062700  000010                  ADD     #8.,R0
179 002332  112205                          MOVB    (R2)+,R5        ;GET RANGE
180 002334  042705  177400                  BIC     #177400,R5
181 002340  005004                          CLR     R4              ;CONVERT RANGE TO DECIMAL ASCII
182 002342  071427  000144                  DIV     #100.,R4
183 002346  005704                          TST     R4
184 002350  001005                          BNE     402$
185 002352  071427  000012                  DIV     #10.,R4         ;1ST DIGIT WAS LEADING ZERO
186 002356  005704                          TST     R4
187 002360  001412                          BEQ     403$            ;2ND DIGIT WAS ALSO LEADING ZERO
188 002362  000406                          BR      4021$
189 002364  052704  000060          402$:   BIS     #60,R4
190 002370  110423                          MOVB    R4,(R3)+        ;TRANSFER 1ST DIGIT
191 002372  005004                          CLR     R4
192 002374  071427  000012                  DIV     #10.,R4
193 002400  052704  000060          4021$:  BIS     #60,R4
194 002404  110423                          MOVB    R4,(R3)+        ;TRANSFER 2ND DIGIT
195 002406  052705  000060          403$:   BIS     #60,R5
196 002412  110523                          MOVB    R5,(R3)+        ;TRANSFER 3RD DIGIT
197 002414  112205                          MOVB    (R2)+,R5        ;GET UNIT
198 002416  042705  177400                  BIC     #177400,R5
199 002422  005004                          CLR     R4
200 002424  071427  000100                  DIV     #100,R4         ;R4=FORWARD (1ST) UNIT
201 002430  116463  002636' 000002         MOVB    UCT(R4),2(R3)   ;TRANSFER TO PRINT BUFFER
202 002436  005004                          CLR     R4
203 002440  071427  000010                  DIV     #10,R4
204 002444  022704  000004                  CMP     #4,R4           ;"FIRST" FLAG SET?
205 002450  103006                          BHIS    404$            ;NO--TRANSFER BLANK
206 002452  112763  000106  000004         MOVB    #'F,4(R3)       ;YES-TRANSFER "F"
207 002460  162704  000004                  SUB     #4,R4           ;REMOVE FLAG EFFECT
208 002464  000403                          BR      4041$
209 002466  112763  000040  000004  404$:   MOVB    #40,4(R3)
210 002474  116423  002636'         4041$:  MOVB    UCT(R4),(R3)+   ;TRANSFER BACKWARD (2ND) UNIT
211 002500  112723  000050                  MOVB    #'(,(R3)+       ;FILL IN SEPARATORS IN PRINT BUFFER
212 002504  005203                          INC     R3
213 002506  112723  000051                  MOVB    #'),(R3)+
214 002512  005203                          INC     R3
215 002514  000402                          BR      4051$
216 002516  112723  000040          405$:   MOVB    #40,(R3)+       ;BLANK OUT REST OF FIELD
217 002522  020300                  4051$:  CMP     R3,R0
218 002524  103774                          BLO     405$
219 002526  005267  176300                  INC     COUNT
220 002532  026767  176274  000000G         CMP     COUNT,QEXAD     ;FINISHED QEX?
221 002540  103004                          BHIS    407$            ;YES
222 002542  032767  000007  176262          BIT     #7,COUNT        ;NO--FINISHED LINE?
223 002550  001265                          BNE     410$            ;NO
224 002552  162703  000610'         407$:   SUB     #PRTBUF,R3      ;PRINT LINE
225 002556                                  PUT$    #LIST,#PRTBUF,R3
226 002600  026767  176226  000000G         CMP     COUNT,QEXAD     ;AT END OF QEX?
227 002606  103632                          BLO     444$            ;NO
228 002610                          500$:   PUT$    #LIST,#NEWLIN,#1        ;YES -AND NEW SPACE LINE
```

```
    229 002634  000207                            RTS     PC
    230                                 ;
    231 002636      125      127      123  UCT:    .ASCII  /UWSP/          ;UNIT CHARACTER REPRESENTATIONS
    232
    233 002642                               FLUSUM: PUT$    #LIST,#FLUHDR,#FHSIZ    ;PRINT HEADER
    234 002666                                       PUT$    ,#FLUHD2,#FH2SIZ
    235 002706  012703  000000G                      MOV     #FLUIDX,R3      ;PRINT FSA-A NODES
    236 002712  012705  000000G                      MOV     #FLIXSZ,R5
    237 002716  012704  000000G                      MOV     #ZERO,R4
    238 002722  012702  000000G                      MOV     #NDFSAA,R2
    239 002726  004767  000350                       JSR     PC,NPA
    240 002732  012703  000000G                      MOV     #FLUIDX,R3      ;PRINT FSA-B NODES
    241 002736  012705  000000G                      MOV     #FLIXSZ,R5
    242 002742  012704  000000G                      MOV     #ZERO,R4
    243 002746  012702  000000G                      MOV     #NDFSAB,R2
    244 002752  004767  000324                       JSR     PC,NPA
    245 002756  012703  000000G                      MOV     #CWPIDX,R3      ;PRINT FSA-C NODES
    246 002762  012705  000000G                      MOV     #CPIXSZ,R5
    247 002766  012704  000000G                      MOV     #ZERO,R4
    248 002772  012702  177777                       MOV     #-1,R2
    249 002776  004767  000660                       JSR     PC,NPO
    250 003002  000207                               RTS     PC
    251                                 ;
    252 003004  004767  000000G              NSP:    JSR     PC,GETNDE       ;GET NEXT NODE
    253 003010  103416                               BCS     1$              ;NO MORE NODES
    254 003012  116401  000000G                      MOVB    NDSIZ(R4),R1    ;SET UP END OF NODE
    255 003016  062701  000000G                      ADD     #NDCHR,R1
    256 003022                                       PRT     R4,R1
    257 003044  000757                               BR      NSP
    258 003046  000207                       1$:     RTS     PC
    259                                 ;
    260                                 ;
    261 003050  000000                       LOCAT:: .WORD                   ;START PRINT LOCATION
    262 003052  000000                       ENDLOC::.WORD                   ;END PRINT LOCATION
    263                                 ;
    264 003054                               BATSUM::PUT$    #LIST,#BSMSG,#BSSIZ     ;PRINT HEADER
    265 003100                                       PUT$    #LIST,#BS2MSG,#BS2SIZ
    266 003124  016700  000000G                      MOV     BATNO,R0        ;GET PRINT PARAMETERS
    267 003130  016000  000000G                      MOV     BSTPTR(R0),R0
    268 003134  016005  000232                       MOV     B;NQRY(R0),R5   ;    R5->NUMBER OF QUERIES
    269 003140  062700  000234                       ADD     #B,QMAP,R0      ;    R0->EQID'S
    270 003144  005067  175662                       CLR     COUNT           ;    COUNT->QRY ID
    271 003150  012703  000610'              1$:     MOV     #PRTBUF,R3
    272 003154  012704  001032'                      MOV     #COUNT,R4
    273 003160  004767  000042'                      JSR     PC,DECASC       ;PRINT QRY ID
    274 003164  112723  000075                       MOVB    #'=,(R3)+       ;TRANSFER "="
    275 003170  016704  175636                       MOV     COUNT,R4
    276 003174  006304                               ASL     R4
    277 003176  060004                               ADD     R0,R4
    278 003200  004767  000042'                      JSR     PC,DECASC       ;PRINT EQID
    279 003204                                       SAVE    R0
    280 003206  162703  000610'                      SUB     #PRTBUF,R3
    281 003212                                       PUT$    #LIST,#PRTBUF,R3        ;ENTER LINE IN FILE
    282 003234                                       RESTORE R0
    283 003236  005267  175570                       INC     COUNT
    284 003242  026705  175564                       CMP     COUNT,R5        ;FINISHED ALL QUERIES?
    285 003246  002740                               BLT     1$              ;NO
```

```
                                        PUT$    #LIST,#NEWLIN,#1        ;SPACE LINE
003250
003274  004767  177342                JSR     PC,FLUSUM       ;PRINT FLU SUMMARY
003300  000207                        RTS     PC
                                ;
003302                          NPA:                            ;PRINT NODES IN ASCII
003302  004767  000000G                JSR     PC,GETNDE       ;GET A NODE
003306  103414                        BCS     100$            ;NO MORE NODES
003310                                  SAVE    R3,R4
003314  004767  000034                JSR     PC,FLIDPT       ;PRINT FLU ID
003320  011604                        MOV     (SP),R4
003322  004767  000054                JSR     PC,FLUPRT       ;PRINT FLU
003326  004767  000120                JSR     PC,FMPRT        ;PRINT FLU MOD
003332                                  RESTORE R3,R4
003336  000761                        BR      NPA
003340  000207                  100$:   RTS     PC
003342      041     041     077 DC:     .ASCII  /!!?*)#(!:!/
                                ;
003354  012703  000610'          FLIDPT: MOV     #PRTBUF,R3      ;SET UP
003360  012701  000402                  MOV     #TYPE2B!BLKZER,R1
003364  062704  000000G                 ADD     #NDFLID,R4
003370  004767  000326'                 JSR     PC,OCTASC       ;PRINT FLUID
003374  112723  000075                  MOVB    #'=,(R3)+       ;TRANSFER "="
003400  000207                          RTS     PC
                                ;
003402  116401  000000G          FLUPRT: MOVB    NDSIZ(R4),R1    ;    R1->NUMBER CHAR'S
003406  042701  177400                  BIC     #177400,R1
003412  062704  000000G                 ADD     #NDCHR,R4       ;    R4->1ST CHAR
003416  020127  000074                  CMP     R1,#60.         ;ONLY PRINT 1ST 60. CHARS
003422  101402                          BLOS    10$
003424  012701  000074                  MOV     #60.,R1
003430  112423                  10$:    MOVB    (R4)+,(R3)+     ;TRANSFER CHAR'S
003432  100402                          BMI     12$             ;ADJUST FOR DON'T CARES
003434  077103                  11$:    SOB     R1,10$
003436  000207                          RTS     PC
003440  114300                  12$:    MOVB    -(R3),R0        ;BACK UP OUTPUT AND GET CODE
003442  005400                          NEG     R0              ;CONVERT DON'T CARE CODE TO CHAR
003444  116023  003342'                 MOVB    DC(R0),(R3)+
003450  000771                          BR      11$
                                ;
003452  016604  000002          FMPRT:  MOV     2(SP),R4        ;GET FLU NODE ADDRESS
003456                                  SAVE    R5,R2,R4
003464  112723  000040                  MOVB    #40.,(R3)+      ;TRANSFER " ["
003470  112723  000133                  MOVB    #'[,(R3)+
003474  016404  000000G                 MOV     NDFMN(R4),R4    ;GET FLU MOD NODE ADDRESS
003500  116405  000000G                 MOVB    NDSIZ(R4),R5    ;GET NODE SIZE
003504  001426                          BEQ     22$             ;IF NULL NODE (I.E. [])
003506  062704  000000G                 ADD     #NDFMS,R4       ;STEP TO START OF MODIFIERS
003512  112401                  20$:    MOVB    (R4)+,R1        ;GET MODIFIER
003514  042701  177400                  BIC     #177400,R1
003520  005000                          CLR     R0
003522  071027  000100                  DIV     #100.,R0 ;R0=MOD ID, R1=MOD VALUE
003526  116023  003656'                 MOVB    100$(R0),(R3)+  ;TRANSFER MOD ID (ASCII)
003532  005000                          CLR     R0
003534  071027  000012                  DIV     #10.,R0         ;R0=UPPER DIGIT, R1=LOWER DIGIT
003540  005700                          TST     R0              ;SKIP IF ZERO
003542  001403                          BEQ     21$
003544  052700  000060                  BIS     #60,R0          ;CONVERT AND TRANSFER
```

```
  343 003550  110023                          MOVB    R0,(R3)+
  344 003552  052701  000060          21$:    BIS     #60,R1          ;CONVERT AND TRANSFER
  345 003556  110123                          MOVB    R1,(R3)+
  346 003560  077524                          SOB     R5,20$          ;REPEAT FOR ALL MODIFIERS
  347 003562  112723  000135          22$:    MOVB    #']',(R3)+      ;CLOSE OUT MODIFIERS
  348 003566  162703  000610'                 SUB     #PRTBUF,R3      ;PUT LINE IN SPOOL FILE
  349 003572                                  PUT$    #LIST,#PRTBUF,R3
  350 003614  012604                          MOV     (SP)+,R4        ;GET FLU NODE OR LINK NODE
  351 003616  016404  000000G                 MOV     NDFMEN(R4),R4   ;GET NEXT LINK NODE
  352 003622  001412                          BEQ     30$             ;NO MORE LEFT
  353 003624  010446                          MOV     R4,-(SP)        ;SAVE LINK NODE ADDRESS
  354 003626  004767  177522                  JSR     PC,FLIDPT       ;PRINT FLU ID FROM LINK
  355 003632  016604  000010                  MOV     10(SP),R4       ;PRINT FLU NODE
  356 003636  004767  177540                  JSR     PC,FLUPRT
  357 003642  011604                          MOV     (SP),R4         ;PRINT FLU MOD NODE FROM LINK
  358 003644  000167  177614                  JMP     10$
  359 003650                          30$:    RESTORE R5,R2
  360 003654  000207                          RTS     PC
  361 003656      055     123     132 100$:   .ASCII  /-SZT/
  362                                         ;
  363 003662  012701  000402          NPO:    MOV     #TYPE2B!BLKZER,R1       ;PRINT NODE IN OCTAL
  364 003666  004767  000000G                 JSR     PC,GETNDE       ;GET A NODE
  365 003672  103446                          BCS     100$            ;NO MORE NODES
  366 003674                                  SAVE    R3,R4
  367 003700  012703  000610'                 MOV     #PRTBUF,R3      ;SET UP
  368 003704  062704  000000G                 ADD     #NDFLID,R4
  369 003710  004767  000326'                 JSR     PC,OCTASC       ;PRINT FLUID
  370 003714  112723  000075                  MOVB    #'=,(R3)+       ;TRANSFER "="
  371 003720  011604                          MOV     (SP),R4
  372 003722  116400  000000G                 MOVB    NDSIZ(R4),R0    ;R0->TERM'S (NUMBER OF)
  373 003726  006200                          ASR     R0
  374 003730  062704  000000G                 ADD     #NDTRM,R4       ;R4->1ST TERM
  375 003734  012701  001002                  MOV     #TYPE2B!SKPZER,R1
  376 003740  000402                          BR      11$
  377 003742  112723  000054          10$:    MOVB    #',,(R3)+       ;TRANSFER ","
  378 003746  004767  000326'         11$:    JSR     PC,OCTASC       ;PRINT TERMS
  379 003752  077005                          SOB     R0,10$
  380 003754  162703  000610'                 SUB     #PRTBUF,R3
  381 003760                                  PUT$    #LIST,#PRTBUF,R3        ;ENTER LINE IN FILE
  382 004002                                  RESTORE R3,R4
  383 004006  000725                          BR      NPO
  384 004010  000207                  100$:   RTS     PC
  385                                         ;
  386                                         ;
  387 004012                          PRINT:: SAVE    R0,R1,R3,R5
  388 004022  012701  000402                  MOV     #TYPE2B!BLKZER,R1
  389 004026  012703  000610'         1$:     MOV     #PRTBUF,R3      ;RESET TO BEGINNING OF PRINT BUFFER
  390 004032  012705  000020                  MOV     #16.,R5         ;PRINT 16. WORDS
  391 004036                                  SAVE    R4
  392 004040  012704  003050'                 MOV     #LOCAT,R4
  393 004044  004767  000326'                 JSR     PC,OCTASC       ;START WITH LOCATION
  394 004050                                  RESTORE R4
  395 004052  112723  000072                  MOVB    #':,(R3)+
  396 004056  062767  000040  176764          ADD     #32.,LOCAT      ;ADVANCE LOCATION ADDRESS
  397 004064  112723  000040          10$:    MOVB    #40,(R3)+       ;SPACE THEN WORD (16 TIMES)
  398 004070  004767  000326'                 JSR     PC,OCTASC
  399 004074  077505                          SOB     R5,10$
```

```
                                        PUT$    #LIST,#PRTBUF,#119.     ;ENTER IN SPOOL FILE
004122  026767  176722  176722          CMP     LOCAT,ENDLOC    ;CONTINUE UNTIL ENTIRE RANGE PRINTED
004130  103736                          BLO     1$
004132                                  RESTORE R0,R1,R3,R5
004142  000207                          RTS     PC
                                ;
004144                          PRNTH:: SAVE    R0,R3,R5
004152  012703  000610'          1$:     MOV     #PRTBUF,R3      ;RESET TO BEGINNING OF PRINT BUFFER
004156  012705  000020                  MOV     #16.,R5         ;PRINT 16. WORDS
004162                                  SAVE    R4
004164  012704  003050'                  MOV     #LOCAT,R4
004170  004767  000072                  JSR     PC,HEX          ;START WITH LOCATION
004174                                  RESTORE R4
004176  112723  000072                  MOVB    #':,(R3)+
004202  062767  000040  176640          ADD     #32.,LOCAT      ;ADVANCE LOCATION ADDRESS
004210  112723  000040          10$:    MOVB    #40,(R3)+       ;SPACE THEN WORD (16 TIMES)
004214  004767  000046                  JSR     PC,HEX
004220  077505                          SOB     R5,10$
004222                                  PUT$    #LIST,#PRTBUF,#86.      ;ENTER IN SPOOL FILE
004246  026767  176576  176576          CMP     LOCAT,ENDLOC    ;CONTINUE UNTIL ENTIRE RANGE PRINTED
004254  103736                          BLO     1$
004256                                  RESTORE R0,R3,R5
004264  000207                          RTS     PC
                                ;
004266                          HEX:    SAVE    R0,R1,R5
004274  112405                          MOVB    (R4)+,R5        ;GET LOW BYTE
004276  112401                          MOVB    (R4)+,R1        ;GET HIGH BYTE
004300  004767  000016                  JSR     PC,HEXB         ;CONVERT AND TRANSFER HIGH BYTE
004304  010501                          MOV     R5,R1           ;CONVERT AND TRANSFER LOW BYTE
004306  004767  000010                  JSR     PC,HEXB
004312                                  RESTORE R0,R1,R5
004320  000207                          RTS     PC
                                ;
004322  042701  177400          HEXB:   BIC     #177400,R1      ;PUT BITS 7,6,5,4 IN R0
004326  005000                          CLR     R0              ;    BITS 3,2,1,0 IN R1
004330  071027  000020                  DIV     #16.,R0
004334  116023  004346'                  MOVB    HEXV(R0),(R3)+  ;TRANSFER HEX
004340  116123  004346'                  MOVB    HEXV(R1),(R3)+
004344  000207                          RTS     PC
                                ;
004346     060     061     062  HEXV:   .ASCII  /0123456789ABCDEF/
        000001                          .END
```

SYMBOL TABLE

```
AT       001000RG      014 B.HBLK   000120      010 FLUIDX= ****** GX        F.RACC= 000016          N.UNIT= 000034
BATNO  = ****** GX         B.HDOC   000114      010 FLUPRT  003402R      014 F.RATT= 000001          OCTASC  000326RG
BATSUM   003054RG      014 B.HRLP   000126      010 FLUSUM  002642R      014 F.RCNM= 000034          OCTGO   000402R
BITVAL = 000000            B.HRLR   000122      010 FMPRT   003452R      014 F.RCTL= 000017          PAR$$$= 000027
BIT0   = 000001            B.HRLW   000124      010 FN.DBR  000026      011 F.RSIZ= 000002          PRINT   004012RG      014
BIT1   = 000002            B.NMBR   000052      010 FN.DBS  000022      011 F.RTYP= 000000          PRNTH   004144RG      014
BIT10  = 002000            B.NQRY   000232      010 FN.DHR  000040      011 F.SEQN= 000100          PRTBUF  000610RG      014
BIT11  = 004000            B.QLSZ   000106      010 FN.EMA  000012      011 F.SPDV= 000072          QCL     = ****** GX
BIT12  = 010000            B.QMAP   000234      010 FN.EMB  000014      011 F.SPUN= 000074          QEX     = ****** GX
BIT13  = 020000            B.QSPL   000316      010 FN.EMC  000016      011 F.STBK= 000036          QEXAD   = ****** GX
BIT14  = 040000            B.QTTM   000076      010 FN.FSA  000000      011 F.UNIT= 000136          QE.RO1= 000144
BIT15  = 100000            B.QUQP   000056      010 FN.FSB  000002      011 F.URBD= 000020          QLB     = ****** GX
BIT2   = 000004            B.SFDB   000010      010 FN.FSC  000004      011 F.VBN = 000064          QLBAD   = ****** GX
BIT3   = 000010            B.SIZE   000772      010 FN.LGQ  000034      011 F.VBSZ= 000060          QLSHD1  000105R       014
BIT4   = 000020            B.SNDP   000012      010 FN.LGU  000036      011 GETNDE= ****** GX       QLSHD2  000120R       014
BIT5   = 000040            B.SSQ    000004      010 FN.MFO  000024      011 HDRSIZ= 000023 G        QLSHD3  000174R       014
BIT6   = 000100            B.SSQF   000050      010 FN.MHR  000010      011 HEADER  000000RG    014 QLSHD4  000233R       014
BIT7   = 000200            B.STAT   000044      010 FN.NMB  000044      011 HEX     004266R     014 QLSHD5  000272R       014
BIT8   = 000400            B.STTE   000053      010 FN.QLS  000006      011 HEXB    004322R     014 QLSHD6  000353R       014
BIT9   = 001000            B.UDOC   000110      010 FN.QRY  000020      011 HEXV    004346R     014 QLSIZ1= 000013
BLKZER= 000400 G           CF.B0  = 000070          FN.SF0  000030      011 LIST    000412RG    014 QLSIZ2= 000054
BSIZ   = 000170 G          CF.B2  = 000067          FN.SF1  000032      011 LOCAT   003050RG    014 QLSIZ3= 000037
BSMSG    000055R       014 CF.B4  = 000066          FN.SHD  000042          LTLUN = 000004 G        QLSIZ4= 000037
BSSIZ  = 000015            CF.B6  = 000065          F.ACTL= 000076          LTNB    000552R     014 QLSIZ5= 000061
BSTPTR= ****** GX          CF.DR0= 000064          F.ALOC= 000040          M       = 000062        QLSIZ6= 000036
BS.CLS= 000002             CF.DR1= 000063          F.BBFS= 000062          N       = 000002        QLSSUM  001034RG      014
BS.DBU= 000004             COUNT   001032R     014 F.BDB = 000070          NDCHR = ****** GX       QRYMAX= ****** GX
BS.INA= 000000             CPIXSZ= ****** GX       F.BGBC= 000057          NDFLID= ****** GX       Q.FDSC  000004        007
BS.OPN= 000001             CWPIDX= ****** GX       F.BKDN= 000026          NDFMEN= ****** GX       Q.NQBK  000000        007
BS.SRC= 000003             DBSLEN= 000116          F.BKDS= 000020          NDFMN = ****** GX       Q.NUHL  000002        007
BS2MSG   000072R       014 DC      003342R     014 F.BKEF= 000050          NDFMS = ****** GX       Q.SIZE  000014        007
BS2SIZ= 000013             DECASC  000042R         F.BKP1= 000051          NDFSAA= ****** GX       RADASC  000232R
BYTE0  = 000000            DEC2    000000R         F.BKST= 000024          NDFSAB= ****** GX       RAD50   000154R
BYTE1  = 000001            DH.BF0  000002      005 F.BKVB= 000064          NDSIZ = ****** GX       R.VAR = ****** GX
BYTE2  = 000002            DH.BF1  000004      005 F.CHR = 000075          NDTRM = ****** GX       SDLB    = ****** GX
BYTE3  = 000003            DH.CTL  000000      005 F.CNTG= 000034          NEWLIN  000411R     014 SDLBAD= ****** GX
BYTE4  = 000004            DH.DMC  000010      005 F.DFNB= 000046          NPA     003302R     014 SKPZER= 001000 G
BYTE5  = 000005            DH.FLG  000006      005 F.DSPT= 000044          NPO     003662R     014 SR.ARE  000114        002
BYTE6  = 000006            DH.DCK  000000      013 F.DVNM= 000134          NSP     003004R     014 SR.ARS  000106        002
BYTE7  = 000007            DN.NTP  000004      013 F.EFBK= 000010          N.BFAC= 000004          SR.DAY  000010        002
BYTE8  = 000010            DN.NXT  000006      013 F.EFN = 000050          N.BHGH= 000006          SR.DLT  000014        002
BYTE9  = 000011            DN.ROT  000002      013 F.EOBB= 000032          N.BTCH= 000004          SR.ECB  000047        002
BYTVAL= 000012             DN.SIZ  000010      013 F.ERR = 000052          N.BUFB= 004000          SR.ECH  000046        002
B.BSTA   000054        010 ENDLOC  003852RG    014 F.FACC= 000043          N.BUFW= 002000          SR.ECL  000050        002
B.CNTX   000046        010 FAL     = ****** GX     F.FFBY= 000014          N.DID = 000024          SR.FIB  000012        002
B.COUQ   000060        010 FD.CR  = ****** GX      F.FNAM= 000110          N.DVNM= 000032          SR.GRE  000100        002
B.FEMA   000132        010 FD.FID  000000      003 F.FNB = 000102          N.FID = 000000          SR.GRS  000072        002
B.FEMB   000142        010 FD.FNB  000006      003 F.FTYP= 000116          N.FNAM= 000006          SR.LEN  000122        002
B.FEMC   000152        010 FD.FVR  000004      003 F.FVER= 000120          N.FOS = 000764          SR.LIN  000066        002
B.FFSA   000202        010 FD.LEN  000010      003 F.HIBK= 000004          N.FTYP= 000014          SR.LIP  000062        002
B.FFSB   000212        010 FHSIZ  = 000020         F.LUN = 000042          N.FVER= 000016          SR.MON  000006        002
B.FFSC   000222        010 FH2SIZ= 000012          F.MBCT= 000054          N.NEXT= 000022          SR.NDC  000042        002
B.FMHR   000172        010 FLIDPT  003354R     014 F.MBC1= 000055          N.PKSZ= 000020          SR.NDS  000036        002
B.FQLS   000162        010 FLIXSZ= ****** GX       F.MBFG= 000056          N.PKTS= 000043          SR.NIN  000030        002
B.FSAZ   000100        010 FLUHDR  000023R     014 F.NRBD= 000024          N.QURY= 000031          SR.NIP  000022        002
B.FSBZ   000102        010 FLUHD2  000043R     014 F.NREC= 000030          N.STAT= 000020          SR.SDB  000032        002
B.FSCZ   000104        010 FLUID  = ****** GX      F.OVBS= 000030          N.SUNT= 000002          SR.SRC  000002        002
```

```
.MAIN.  MACRO M1110  27-MAR-80 13:17  PAGE 12-9
SYMBOL TABLE

SR.SUN  000000      002 ST.QRY  000002      006 S.FNBW= 000017          WORD0 = 000000          XGTSRE= 000014
SR.TWS  000056      002 ST.QSZ  000034      006 S.FNTY= 000004          WORD1 = 000002          XHITSK= 000011
SR.WSL  000052      002 ST.SCH  000040      006 S.FTYP= 000002          WORD2 = 000004          XHLMER= 000002
SR.YR   000004      002 ST.UHL  000004      006 S.HRL = 000240          WORD3 = 000006          XHOTSK= 000010
SR.1IN  000024      002 ST.XLT  000014      006 S.NFEN= 000020          WORD4 = 000010          XMSCHE= 000000
SR.1IP  000016      002 SU.DBU= 000004          TTABLE= ****** GX       WORD5 = 000012          XQTS  = 000003
SS.FID  000002      004 SU.DON= 000006          TYPE1B= 000000 G        WORD6 = 000014          XQT0  = 000001
SS.FNB  000010      004 SU.IDL= 000000          TYPE2B= 000002 G        WORD7 = 000016          XSULOA= 000005
SS.FVR  000006      004 SU.LOD= 000001          TYPE3B= 000004 G        WORD8 = 000020          XTABLE= ****** GX
SS.LEN  000012      004 SU.SRC= 000002          TYPE4B= 000006 G        WORD9 = 000022          ZERO  = ****** GX
SS.STT  000000      004 SU.SRR= 000005          UCT     002636R     014 WRDVAL= 000024          .FSRCB= ****** G
ST.ASZ  000020      006 SU.XPD= 000003          WN.NTP  000004      012 XBATCH= 000013          .PUT  = ****** G
ST.BSZ  000024      006 S.BFHD= 000020          WN.NXT  000006      012 XDBLOA= 000004          ...PC1= 000412R     014
ST.BTC  000000      006 S.FATT= 000016          WN.ROT  000002      012 XDBPRO= 000012          ...PC2= 000606R     014
ST.CSZ  000030      006 S.FDB = 000140          WN.SIZ  000010      012 XDMCIN= 000006          ...PC3= 000412R     014
ST.HRL  000010      006 S.FNAM= 000006          WN.SRC  000000      012 XFOSMR= 000007          ...TPC= 000140
ST.LEN  000044      006 S.FNB = 000036          WN.TYP  000001      012

. ABS.  000000    000
        000732    001
SRCOFF  000122    002
FDSCOF  000010    003
SUSOFF  000012    004
DHROFF  000012    005
STTOFF  000044    006
QSPLOF  000014    007
BSTOFF  000772    010
FNOFFS  000044    011
WNODOF  000010    012
DNODOF  000010    013
PRTCDE  004366    014
$$FSR1  001020    015
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  6701 WORDS  (27 PAGES)
DYNAMIC MEMORY:  8084 WORDS  ( 31 PAGES)
ELAPSED TIME:  00:01:06
PRTCDE,PRTCDE/-SP/NL:ME:BEX=[20,1]JP,M,MDQT0,PRTSUBS,PRTCDE
```

```
TASK    NAME    : QT0
PARTITION NAME  : HSTSPR
IDENTIFICATION  : 0736
TASK  UIC       : [20,3]
TASK  PRIORITY  : 45.
STACK    LIMITS: 000236 001235 001000 00512.
PRG XFR ADDRESS: 104356
TASK ATTRIBUTES: CP
TOTAL ADDRESS WINDOWS: 3.
TASK  IMAGE  SIZE  : 18944. WORDS
TASK ADDRESS LIMITS: 000000 111767
R-W DISK BLK LIMITS: 000002 000113 000112 00074.

*** ROOT SEGMENT: BUFFER


R/W MEM  LIMITS: 000000 111767 111770 37880.
DISK BLK LIMITS: 000002 000113 000112 00074.


MEMORY ALLOCATION SYNOPSIS:

SECTION                                          TITLE   IDENT   FILE
-------                                          -----   -----   ----
. BLK.:(RW,I,LCL,REL,CON) 001236 000670 00440.
AAAQLS:(RW,I,LCL,REL,CON) 002126 000160 00112.
                          002126 000160 00112.  .MAIN.          QLSCDE.OBJ;1
AABUFS:(RW,I,LCL,REL,CON) 002306 052046 21542.
                          002306 052046 21542.  .MAIN.          BUFFER.OBJ;1
BSTOFF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000.  .MAIN.          BUFFER.OBJ;1
                          000000 000000 00000.  QT0             QT0.OBJ;1
                          000000 000000 00000.  .MAIN.          QLSCDE.OBJ;1
                          000000 000000 00000.  .MAIN.          FLUCDE.OBJ;1
                          000000 000000 00000.  .MAIN.          EMXCDE.OBJ;1
                          000000 000000 00000.  .MAIN.          NUMRNG.OBJ;1
                          000000 000000 00000.  PARSER          PARSER.OBJ;1
                          000000 000000 00000.  QUERY           QTFORM.OBJ;1
                          000000 000000 00000.  .MAIN.          CSTAB.OBJ;1
CCET  :(RW,I,LCL,REL,CON) 054354 000200 00128.
                          054354 000200 00128.  .MAIN.          EMXCDE.OBJ;1
CSTAB :(RW,I,LCL,REL,CON) 054554 001400 00768.
                          054554 001400 00768.  .MAIN.          CSTAB.OBJ;1
DHROFF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000.  .MAIN.          BUFFER.OBJ;1
                          000000 000000 00000.  QT0             QT0.OBJ;1
                          000000 000000 00000.  .MAIN.          QLSCDE.OBJ;1
                          000000 000000 00000.  .MAIN.          FLUCDE.OBJ;1
                          000000 000000 00000.  .MAIN.          EMXCDE.OBJ;1
                          000000 000000 00000.  .MAIN.          NUMRNG.OBJ;1
                          000000 000000 00000.  PARSER          PARSER.OBJ;1
                          000000 000000 00000.  QUERY           QTFORM.OBJ;1
                          000000 000000 00000.  .MAIN.          CSTAB.OBJ;1
DNODOF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000.  .MAIN.          BUFFER.OBJ;1
```

```
                        000000 000000 00000. QT0            QT0.OBJ;1
                        000000 000000 00000. .MAIN.         QLSCDE.OBJ;1
                        000000 000000 00000. .MAIN.         FLUCDE.OBJ;1
                        000000 000000 00000. .MAIN.         EMXCDE.OBJ;1
                        000000 000000 00000. .MAIN.         NUMRNG.OBJ;1
                        000000 000000 00000. PARSER         PARSER.OBJ;1
                        000000 000000 00000. QUERY          QTFORM.OBJ;1
                        000000 000000 00000. .MAIN.         CSTAB.OBJ;1
EMX   :(RW,I,LCL,REL,CON) 056154 000644 00420.
                        056154 000644 00420. .MAIN.         EMXCDE.OBJ;1
EMXCDE:(RW,I,LCL,REL,CON) 057020 002540 01376.
                        057020 002540 01376. .MAIN.         EMXCDE.OBJ;1
EMXDAT:(RW,I,LCL,REL,CON) 061560 000306 00198.
                        061560 000306 00198. .MAIN.         EMXCDE.OBJ;1
FDSCOF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                        000000 000000 00000. .MAIN.         BUFFER.OBJ;1
                        000000 000000 00000. QT0            QT0.OBJ;1
                        000000 000000 00000. .MAIN.         QLSCDE.OBJ;1
                        000000 000000 00000. .MAIN.         FLUCDE.OBJ;1
                        000000 000000 00000. .MAIN.         EMXCDE.OBJ;1
                        000000 000000 00000. .MAIN.         NUMRNG.OBJ;1
                        000000 000000 00000. PARSER         PARSER.OBJ;1
                        000000 000000 00000. QUERY          QTFORM.OBJ;1
                        000000 000000 00000. .MAIN.         CSTAB.OBJ;1
FLUCDE:(RW,I,LCL,REL,CON) 062066 003050 01576.
                        062066 003050 01576. .MAIN.         FLUCDE.OBJ;1
FLUNDE:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                        000000 000000 00000. .MAIN.         BUFFER.OBJ;1
FNOFFS:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                        000000 000000 00000. .MAIN.         BUFFER.OBJ;1
                        000000 000000 00000. QT0            QT0.OBJ;1
                        000000 000000 00000. .MAIN.         QLSCDE.OBJ;1
                        000000 000000 00000. .MAIN.         FLUCDE.OBJ;1
                        000000 000000 00000. .MAIN.         EMXCDE.OBJ;1
                        000000 000000 00000. .MAIN.         NUMRNG.OBJ;1
                        000000 000000 00000. PARSER         PARSER.OBJ;1
                        000000 000000 00000. QUERY          QTFORM.OBJ;1
                        000000 000000 00000. .MAIN.         CSTAB.OBJ;1
MSGOUT:(RW,I,LCL,REL,CON) 065136 001214 00652.
                        065136 001214 00652. MESSAG         MSGOUT.OBJ;1
NUMRNG:(RW,I,LCL,REL,CON) 066352 006524 03412.
                        066352 006524 03412. .MAIN.         NUMRNG.OBJ;1
PARSER:(RW,I,LCL,REL,CON) 075076 001206 00646.
                        075076 001206 00646. PARSER         PARSER.OBJ;1
PSDATA:(RW,I,LCL,REL,CON) 076304 000314 00204.
                        076304 000314 00204. PARSER         PARSER.OBJ;1
QLSCDE:(RW,I,LCL,REL,CON) 076620 001504 00836.
                        076620 001504 00836. .MAIN.         QLSCDE.OBJ;1
QNBASE:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                        000000 000000 00000. .MAIN.         BUFFER.OBJ;1
QSPLOF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                        000000 000000 00000. .MAIN.         BUFFER.OBJ;1
                        000000 000000 00000. QT0            QT0.OBJ;1
                        000000 000000 00000. .MAIN.         QLSCDE.OBJ;1
                        000000 000000 00000. .MAIN.         FLUCDE.OBJ;1
```

```
                        000000 000000 00000. .MAIN.      EMXCDE.OBJ;1
                        000000 000000 00000. .MAIN.      NUMRNG.OBJ;1
                        000000 000000 00000. PARSER      PARSER.OBJ;1
                        000000 000000 00000. QUERY       QTFORM.OBJ;1
                        000000 000000 00000. .MAIN.      CSTAB.OBJ;1
QTDATA:(RW,I,LCL,REL,CON) 100324 000230 00152.
                        100324 000230 00152. QUERY       QTFORM.OBJ;1
QTFORM:(RW,I,LCL,REL,CON) 100554 003172 01658.
                        100554 003172 01658. QUERY       QTFORM.OBJ;1
QT0   :(RW,I,LCL,REL,CON) 103746 005630 02968.
                        103746 005630 02968. QT0         QT0.OBJ;1
SRCOFF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                        000000 000000 00000. .MAIN.      BUFFER.OBJ;1
                        000000 000000 00000. QT0         QT0.OBJ;1
                        000000 000000 00000. .MAIN.      QLSCDE.OBJ;1
                        000000 000000 00000. .MAIN.      FLUCDE.OBJ;1
                        000000 000000 00000. .MAIN.      EMXCDE.OBJ;1
                        000000 000000 00000. .MAIN.      NUMRNG.OBJ;1
                        000000 000000 00000. PARSER      PARSER.OBJ;1
                        000000 000000 00000. QUERY       QTFORM.OBJ;1
                        000000 000000 00000. .MAIN.      CSTAB.OBJ;1
STCODE:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                        000000 000000 00000. QUERY       QTFORM.OBJ;1
STTOFF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                        000000 000000 00000. .MAIN.      BUFFER.OBJ;1
                        000000 000000 00000. QT0         QT0.OBJ;1
                        000000 000000 00000. .MAIN.      QLSCDE.OBJ;1
                        000000 000000 00000. .MAIN.      FLUCDE.OBJ;1
                        000000 000000 00000. .MAIN.      EMXCDE.OBJ;1
                        000000 000000 00000. .MAIN.      NUMRNG.OBJ;1
                        000000 000000 00000. PARSER      PARSER.OBJ;1
                        000000 000000 00000. QUERY       QTFORM.OBJ;1
                        000000 000000 00000. .MAIN.      CSTAB.OBJ;1
SUSOFF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                        000000 000000 00000. .MAIN.      BUFFER.OBJ;1
                        000000 000000 00000. QT0         QT0.OBJ;1
                        000000 000000 00000. .MAIN.      QLSCDE.OBJ;1
                        000000 000000 00000. .MAIN.      FLUCDE.OBJ;1
                        000000 000000 00000. .MAIN.      EMXCDE.OBJ;1
                        000000 000000 00000. .MAIN.      NUMRNG.OBJ;1
                        000000 000000 00000. PARSER      PARSER.OBJ;1
                        000000 000000 00000. QUERY       QTFORM.OBJ;1
                        000000 000000 00000. .MAIN.      CSTAB.OBJ;1
TRCODE:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                        000000 000000 00000. QUERY       QTFORM.OBJ;1
UNODOF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                        000000 000000 00000. .MAIN.      BUFFER.OBJ;1
                        000000 000000 00000. QT0         QT0.OBJ;1
                        000000 000000 00000. .MAIN.      QLSCDE.OBJ;1
                        000000 000000 00000. .MAIN.      FLUCDE.OBJ;1
                        000000 000000 00000. .MAIN.      EMXCDE.OBJ;1
                        000000 000000 00000. .MAIN.      NUMRNG.OBJ;1
                        000000 000000 00000. PARSER      PARSER.OBJ;1
                        000000 000000 00000. QUERY       QTFORM.OBJ;1
                        000000 000000 00000. .MAIN.      CSTAB.OBJ;1
```

```
$$FSR1:(RW,D,GBL,REL,OVR)  111576 000000 00000.
                           111576 000000 00000. QT0          QT0.OBJ;1
$$FSR2:(RW,D,GBL,REL,CON)  111576 000104 00068.
$$RESL:(RW,I,LCL,REL,CON)  111702 000066 00054.
$$RESM:(RW,I,LCL,REL,CON)  140000 015600 07040.


GLOBAL SYMBOLS:

ACTFMN 002436-R  CSTNUM 000100    EMXNB2 000004    HSZ    066366-R  NP1MSZ 010000    QNDNXT 076610-R  TDABMX 007764
ARGPUT 075676-R  CSTP   001000    EMXNFD 000400    ILLCHR 111562-R  NP2MSZ 036000    QNDSIZ 000010    TDAMAD 007760
ARGSTK 030756-R  CSTPGR 000010    EMXNSQ 100000    INCLUD 066376-R  NP2OSZ 000153    QNFLS  040000    TDBBLK 000003
ASTKMX 000400    CSTSEN 000020    EMXNVD 001000    INIVDC 000010    NP3MSZ 000524    QNFLU  100000    TDBCLS 000005
AVELGT 000025    CSTSO  000004    EMXSZF 002000    INTERW 056212-R  NP3OSZ 000125    QNOPCD 000000    TDBMAD 007775
BATABD 003401    CSTSZ  000004    EMXTRL 010000    INTRWD 177777    NOLBE  002146-R  QNPOOL 030770-R  TDBOSZ 000074
BATNO  104306-R  CSTTS  000002    EMXVDC 004000    IOSB   104336-R  NULFMN 002454-R  QNSTE  000006    TDCBLK 000010
BATOVF 000401    CSTVDC 000040    EMXVVV 000001    LEXTBI 054554-R  NUMRNG 070050-R  QNTLST 044772-R  TDCMSZ 004000
BATREJ 104512-R  CSTWD  000040    EMXZNF 004000    LEXTB3 055554-R  NVDEMX 056170-R  QNWIN  000001    TOKAND 000007
BOTSTK 027756-R  CSTZ   000002    EMX0VD 000200    LOGCLS 000002    NVLDC  177774    QRYBAD 002001    TOKETX 000017
B$FMN  000020    CSTZER 020000    ENCNT  000004    LSZ    066364-R  NXTNDE 002430-R  QRYBUF 040770-R  TOKFLU 000000
B$FST  020000    CWPIDX 002406-R  ERRCDE 104312-R  LTFLU  000400    OVFCLS 110020-R  QRYDEL 111132-R  TOKFMD 000002
B$MUL  000040    CWPNDE 002434-R  ERROR  110154-R  LTFMD  001000    PARBUF 061560-R  QRYEND 044770-R  TOKHOR 000005
B$NOT  000100    DECPT  000056    ERRORF 110124-R  LTPRX  002000    PARSER 075076-R  QRYFAS 002444-R  TOKLP  000014
B$PUBH 010000    DIRERR 110032-R  ERRORV 106640-R  LTSKP  004000    PAR1   110650-R  QRYMAX 017366-R  TOKNOT 000003
B$PUBL 004000    DNUM   000010    ETX    000003    LTTOK  010000    PAR2   110652-R  QRYMS  002442-R  TOKOR  000004
B$PUFH 100000    DPEMX  056754-R  EXIT   110336-R  MINEMX 056366-R  PASSI  002126-R  QRYMSZ 000031    TOKPXP 000015
B$PUFL 040000    EIXCNT 000010    FAL    017460-R  MR     000100    PD     000020    QRYNO  104310-R  TOKPXS 000013
B$SUC  000200    ELSCLS 000007    FALAD  017370-R  MSGOUT 065340-R  PFLU   062114-R  QRYOK  000001    TOKPXW 000011
CETEMX 056220-R  ELSMSK 177761    FALMSZ 000500    NAQEX  002154-R  PFLUMD 064266-R  QRYPS  002440-R  TOKRP  000016
CET6B  054354-R  EMACLS 000003    FCSERR 110072-R  NDBCLS 000006    PNBRNG 066426-R  QRYSZ  002000    TOKSBD 000001
CFALS  002156-R  EMACUT 001700    FLDC   177775    NDBKW  000002    POLEND 016352-R  QTFORM 100554-R  TOKSTK 027353-R
CHRCNT 110014-R  EMAMSZ 016000    FLIXMK 177740    NDCHR  000014    POOL   002462-R  RNGCDE 066402-R  TOPSTK 030756-R
CLBREQ 000400    EMAQSZ 110000-R  FLIXSZ 000040    NDFLID 000006    PRXFLG 002142-R  SDFLG  002134-R  TRMCUT 000040
CLSEMX 057020-R  EMATSZ 110002-R  FLUCLS 000001    NDFMEN 000012    PSTKMX 000024    SDLB   026176-R  TSKNAM 104300-R
CLSIN  001401    EMBCLS 000004    FLUCUT 000016    NDFMN  000010    QCL    017376-R  SDLBAD 017362-R  TSTKMX 000400
CLSOLY 001001    EMBCUT 001130    FLUID  104342-R  NDFMS  000005    QEX    026552-R  SDLBE  002152-R  TTABLE 020160-R
CMDCDE 104304-R  EMBMSZ 004400    FLUIDX 002306-R  NDFSAA 000001    QEXAD  017364-R  SDLBS  002150-R  TTBMSZ 003270
CORUNK 003001    EMBQSZ 110004-R  FLUNDE 002432-R  NDFSAB 000002    QEXFPV 174000    SEGOP  177770    TTSIZ  017372-R
CORUPT 002401    EMBTSZ 110006-R  FLUTBL 055154-R  NDFWD  000000    QEXFSV 134000    SLBEG  062074-R  VDCCNT 110016-R
CPIXMK 177770    EMCMSZ 001000    FLUTYP 062070-R  NDSIZ  000004    QEXFWV 074000    SLBMSZ 000354    VEC1MX 000375
CPIXSZ 000010    EMCQSZ 107776-R  FMDIDX 002426-R  NDTRM  000010    QEXMSZ 000600    SNODE  030760-R  VEC2MX 000375
CQLBS  002144-R  EMCTSZ 107774-R  FMEPSZ 000400    NDTYP  000005    QLB    025460-R  SPOL   103636-R  VEC3MX 000125
CSTCET 100000    EMX    020354-R  FMIXSZ 000001    NFALE  002160-R  QLBAD  017360-R  SPXCT  002143-R  VI     017360-R
CSTDP  000200    EMXCHF 020000    FMNFLG 002132-R  NFEMX  056162-R  QLBMSZ 000516    SPXPT  002140-R  VIBCUT 000036
CSTDT  000001    EMXDTF 010000    FMNPSZ 002260    NFLDC  177773    QLS    017352-R  SPXSTK 027354-R  VIBQSZ 110010-R
CSTDW  000200    EMXEXF 000001    FNPOSZ 000400    NIXCNT 000020    QLSBAT 017356-R  SSQ    111512-R  VIBTSZ 110012-R
CSTEFM 010000    EMXFDB 061572-R  FNPSZ  005734    NMRNGF 000100    QLSBUF 017352-R  START3 104366-R  VILGT  017356-R
CSTEP  004000    EMXFDC 002000    FPXCT  002142-R  NNMASK 177760    QLSHD  017352-R  STEP   111466-R  VINXT  061736-R
CSTER  002000    EMXLGT 017354-R  FPXPT  002136-R  NODEA  002446-R  QLSIZ  010000    STK1   000200    VI0MSZ 000400
CSTESC 000010    EMXMCD 000002    FPXSTK 027754-R  NODEB  002450-R  QNARG1 000002    STK2   000400    VI1MSZ 000400
CSTFDC 000100    EMXMCF 001000    GENQLS 076620-R  NODEC  002452-R  QNARG2 000004    STX    000002    VI2MSZ 000400
CSTMIN 000400    EMXMSZ 016000    GENTX  100226-R  NOTFLG 002130-R  QNATTR 000006    SYNTAB 076567-R  VI3MSZ 000400
CSTNC  000001    EMXMTV 040000    GETNDE 064234-R  NPECNT 000144    QNDCNT 001000    SZCAL1 106674-R  VLDC   177776
CSTNF  000020    EMXNB1 000002    HSTKMX 000100    NPEVSZ 000043    QNDLST 040770-R  TDABLK 000004    WRTEMX 061262-R
```

```
XTABLE 023450-R  XTSIZ  017374-R  ZVDEMX 056154-R  $EDMSG 065566-R
XTBMSZ 002010    ZERO   104344-R  ZVLDC  177772    .MOLUN 104340-R
```

*** TASK BUILDER STATISTICS:

TOTAL WORK FILE REFERENCES: 72140.
WORK FILE READS: 0.
WORK FILE WRITES: 0.
SIZE OF CORE POOL: 6634. WORDS (25. PAGES)
SIZE OF WORK FILE: 4864. WORDS (19. PAGES)

ELAPSED TIME:00:00:30

```
TASK    NAME    : QT0
PARTITION NAME  : HSTSPR
IDENTIFICATION  : 0736
TASK  UIC       : [20,3]
TASK  PRIORITY  : 45.
STACK    LIMITS: 000236 001235 001000 00512.
PRG XFR ADDRESS: 112200
TASK ATTRIBUTES: CP
TOTAL ADDRESS WINDOWS: 3.
TASK  IMAGE  SIZE   : 21760. WORDS
TASK ADDRESS LIMITS: 000000 124737
R-W DISK BLK LIMITS: 000002 000126 000125 00085.

*** ROOT SEGMENT: BUFFER


R/W MEM  LIMITS: 000000 124737 124740 43488.
DISK BLK LIMITS: 000002 000126 000125 00085.


MEMORY ALLOCATION SYNOPSIS:

SECTION                                          TITLE   IDENT   FILE
-------                                          -----   -----   ----
. BLK.:(RW,I,LCL,REL,CON) 001236 001730 00984.
                          001236 000732 00474.  .MAIN.          PRTCDE.OBJ;1
AAAQLS:(RW,I,LCL,REL,CON) 003166 000160 00112.
                          003166 000160 00112.  .MAIN.          QLSCDE.OBJ;1
AABUFS:(RW,I,LCL,REL,CON) 003346 052046 21542.
                          003346 052046 21542.  .MAIN.          BUFFER.OBJ;1
BSTOFF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000.  .MAIN.          BUFFER.OBJ;1
                          000000 000000 00000.  QT0             QT0S.OBJ;1
                          000000 000000 00000.  .MAIN.          QLSCDE.OBJ;1
                          000000 000000 00000.  .MAIN.          FLUCDE.OBJ;1
                          000000 000000 00000.  .MAIN.          EMXCDES.OBJ;1
                          000000 000000 00000.  .MAIN.          NUMRNG.OBJ;1
                          000000 000000 00000.  PARSER          PARSER.OBJ;1
                          000000 000000 00000.  QUERY           QTFORM.OBJ;1
                          000000 000000 00000.  .MAIN.          CSTAB.OBJ;1
                          000000 000000 00000.  .MAIN.          PRTCDE.OBJ;1
CCET  :(RW,I,LCL,REL,CON) 055414 000200 00128.
                          055414 000200 00128.  .MAIN.          EMXCDES.OBJ;1
CSTAB :(RW,I,LCL,REL,CON) 055614 001400 00768.
                          055614 001400 00768.  .MAIN.          CSTAB.OBJ;1
DHROFF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000.  .MAIN.          BUFFER.OBJ;1
                          000000 000000 00000.  QT0             QT0S.OBJ;1
                          000000 000000 00000.  .MAIN.          QLSCDE.OBJ;1
                          000000 000000 00000.  .MAIN.          FLUCDE.OBJ;1
                          000000 000000 00000.  .MAIN.          EMXCDES.OBJ;1
                          000000 000000 00000.  .MAIN.          NUMRNG.OBJ;1
                          000000 000000 00000.  PARSER          PARSER.OBJ;1
                          000000 000000 00000.  QUERY           QTFORM.OBJ;1
                          000000 000000 00000.  .MAIN.          CSTAB.OBJ;1
```

```
                        000000 000000 00000. .MAIN.      PRTCDE.OBJ;1
DNODOF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                        000000 000000 00000. .MAIN.      BUFFER.OBJ;1
                        000000 000000 00000. QT0         QT0S.OBJ;1
                        000000 000000 00000. .MAIN.      QLSCDE.OBJ;1
                        000000 000000 00000. .MAIN.      FLUCDE.OBJ;1
                        000000 000000 00000. .MAIN.      EMXCDES.OBJ;1
                        000000 000000 00000. .MAIN.      NUMRNG.OBJ;1
                        000000 000000 00000. PARSER      PARSER.OBJ;1
                        000000 000000 00000. QUERY       QTFORM.OBJ;1
                        000000 000000 00000. .MAIN.      CSTAB.OBJ;1
                        000000 000000 00000. .MAIN.      PRTCDE.OBJ;1
EMX   :(RW,I,LCL,REL,CON) 057214 000644 00420.
                        057214 000644 00420. .MAIN.      EMXCDES.OBJ;1
EMXCDE:(RW,I,LCL,REL,CON) 060060 002734 01500.
                        060060 002734 01500. .MAIN.      EMXCDES.OBJ;1
EMXDAT:(RW,I,LCL,REL,CON) 063014 000306 00198.
                        063014 000306 00198. .MAIN.      EMXCDES.OBJ;1
FDSCOF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                        000000 000000 00000. .MAIN.      BUFFER.OBJ;1
                        000000 000000 00000. QT0         QT0S.OBJ;1
                        000000 000000 00000. .MAIN.      QLSCDE.OBJ;1
                        000000 000000 00000. .MAIN.      FLUCDE.OBJ;1
                        000000 000000 00000. .MAIN.      EMXCDES.OBJ;1
                        000000 000000 00000. .MAIN.      NUMRNG.OBJ;1
                        000000 000000 00000. PARSER      PARSER.OBJ;1
                        000000 000000 00000. QUERY       QTFORM.OBJ;1
                        000000 000000 00000. .MAIN.      CSTAB.OBJ;1
                        000000 000000 00000. .MAIN.      PRTCDE.OBJ;1
FLUCDE:(RW,I,LCL,REL,CON) 063322 003050 01576.
                        063322 003050 01576. .MAIN.      FLUCDE.OBJ;1
FLUNDE:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                        000000 000000 00000. .MAIN.      BUFFER.OBJ;1
FNOFFS:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                        000000 000000 00000. .MAIN.      BUFFER.OBJ;1
                        000000 000000 00000. QT0         QT0S.OBJ;1
                        000000 000000 00000. .MAIN.      QLSCDE.OBJ;1
                        000000 000000 00000. .MAIN.      FLUCDE.OBJ;1
                        000000 000000 00000. .MAIN.      EMXCDES.OBJ;1
                        000000 000000 00000. .MAIN.      NUMRNG.OBJ;1
                        000000 000000 00000. PARSER      PARSER.OBJ;1
                        000000 000000 00000. QUERY       QTFORM.OBJ;1
                        000000 000000 00000. .MAIN.      CSTAB.OBJ;1
                        000000 000000 00000. .MAIN.      PRTCDE.OBJ;1
MSGOUT:(RW,I,LCL,REL,CON) 066372 001214 00652.
                        066372 001214 00652. MESSAG      MSGOUT.OBJ;1
NUMRNG:(RW,I,LCL,REL,CON) 067606 006524 03412.
                        067606 006524 03412. .MAIN.      NUMRNG.OBJ;1
PARSER:(RW,I,LCL,REL,CON) 076332 001206 00646.
                        076332 001206 00646. PARSER      PARSER.OBJ;1
PRTCDE:(RW,I,LCL,REL,CON) 077540 004366 02294.
                        077540 004366 02294. .MAIN.      PRTCDE.OBJ;1
PSDATA:(RW,I,LCL,REL,CON) 104126 000314 00204.
                        104126 000314 00204. PARSER      PARSER.OBJ;1
QLSCDE:(RW,I,LCL,REL,CON) 104442 001504 00836.
```

```
                        104442 001504 00836. .MAIN.          QLSCDE.OBJ;1
QNBASE:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                        000000 000000 00000. .MAIN.          BUFFER.OBJ;1
QSPLOF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                        000000 000000 00000. .MAIN.          BUFFER.OBJ;1
                        000000 000000 00000. QT0             QT0S.OBJ;1
                        000000 000000 00000. .MAIN.          QLSCDE.OBJ;1
                        000000 000000 00000. .MAIN.          FLUCDE.OBJ;1
                        000000 000000 00000. .MAIN.          EMXCDES.OBJ;1
                        000000 000000 00000. .MAIN.          NUMRNG.OBJ;1
                        000000 000000 00000. PARSER          PARSER.OBJ;1
                        000000 000000 00000. QUERY           QTFORM.OBJ;1
                        000000 000000 00000. .MAIN.          CSTAB.OBJ;1
                        000000 000000 00000. .MAIN.          PRTCDE.OBJ;1
QTDATA:(RW,I,LCL,REL,CON) 106146 000230 00152.
                        106146 000230 00152. QUERY           QTFORM.OBJ;1
QTFORM:(RW,I,LCL,REL,CON) 106376 003172 01658.
                        106376 003172 01658. QUERY           QTFORM.OBJ;1
QT0    :(RW,I,LCL,REL,CON) 111570 006152 03178.
                        111570 006152 03178. QT0             QT0S.OBJ;1
SRCOFF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                        000000 000000 00000. .MAIN.          BUFFER.OBJ;1
                        000000 000000 00000. QT0             QT0S.OBJ;1
                        000000 000000 00000. .MAIN.          QLSCDE.OBJ;1
                        000000 000000 00000. .MAIN.          FLUCDE.OBJ;1
                        000000 000000 00000. .MAIN.          EMXCDES.OBJ;1
                        000000 000000 00000. .MAIN.          NUMRNG.OBJ;1
                        000000 000000 00000. PARSER          PARSER.OBJ;1
                        000000 000000 00000. QUERY           QTFORM.OBJ;1
                        000000 000000 00000. .MAIN.          CSTAB.OBJ;1
                        000000 000000 00000. .MAIN.          PRTCDE.OBJ;1
STCODE:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                        000000 000000 00000. QUERY           QTFORM.OBJ;1
STTOFF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                        000000 000000 00000. .MAIN.          BUFFER.OBJ;1
                        000000 000000 00000. QT0             QT0S.OBJ;1
                        000000 000000 00000. .MAIN.          QLSCDE.OBJ;1
                        000000 000000 00000. .MAIN.          FLUCDE.OBJ;1
                        000000 000000 00000. .MAIN.          EMXCDES.OBJ;1
                        000000 000000 00000. .MAIN.          NUMRNG.OBJ;1
                        000000 000000 00000. PARSER          PARSER.OBJ;1
                        000000 000000 00000. QUERY           QTFORM.OBJ;1
                        000000 000000 00000. .MAIN.          CSTAB.OBJ;1
                        000000 000000 00000. .MAIN.          PRTCDE.OBJ;1
SUSOFF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                        000000 000000 00000. .MAIN.          BUFFER.OBJ;1
                        000000 000000 00000. QT0             QT0S.OBJ;1
                        000000 000000 00000. .MAIN.          QLSCDE.OBJ;1
                        000000 000000 00000. .MAIN.          FLUCDE.OBJ;1
                        000000 000000 00000. .MAIN.          EMXCDES.OBJ;1
                        000000 000000 00000. .MAIN.          NUMRNG.OBJ;1
                        000000 000000 00000. PARSER          PARSER.OBJ;1
                        000000 000000 00000. QUERY           QTFORM.OBJ;1
                        000000 000000 00000. .MAIN.          CSTAB.OBJ;1
                        000000 000000 00000. .MAIN.          PRTCDE.OBJ;1
```

```
TRCODE:(RW,I,LCL,ABS,CON)  000000 000000 00000.
                           000000 000000 00000. QUERY          QTFORM.OBJ;1
UNODOF:(RW,I,LCL,ABS,CON)  000000 000000 00000.
                           000000 000000 00000. .MAIN.         BUFFER.OBJ;1
                           000000 000000 00000. QT0            QT0S.OBJ;1
                           000000 000000 00000. .MAIN.         QLSCDE.OBJ;1
                           000000 000000 00000. .MAIN.         FLUCDE.OBJ;1
                           000000 000000 00000. .MAIN.         EMXCDES.OBJ;1
                           000000 000000 00000. .MAIN.         NUMRNG.OBJ;1
                           000000 000000 00000. PARSER         PARSER.OBJ;1
                           000000 000000 00000. QUERY          QTFORM.OBJ;1
                           000000 000000 00000. .MAIN.         CSTAB.OBJ;1
                           000000 000000 00000. .MAIN.         PRTCDE.OBJ;1
$$FSR1:(RW,D,GBL,REL,OVR)  117742 001020 00528.
                           117742 000000 00000. QT0            QT0S.OBJ;1
                           117742 001020 00528. .MAIN.         PRTCDE.OBJ;1
$$FSR2:(RW,D,GBL,REL,CON)  120762 000104 00068.
$$RESL:(RW,I,LCL,REL,CON)  121066 003650 01960.
$$RESM:(RW,I,LCL,REL,CON)  140000 015600 07040.


GLOBAL SYMBOLS:

ACTFMN 003476-R  CMDCDE 112126-R  DECPT  000056    EMXNB1 000002    FLUTBL 056214-R  LTFLU  000400    NODEC  003512-R
ARGPUT 077132-R  CORUNK 003001    DIRERR 116002-R  EMXNB2 000004    FLUTYP 063324-R  LTFMD  001000    NOTFLG 003170-R
ARGSTK 032016-R  CORUPT 002401    DNUM   000010    EMXNFD 000400    FMDIDX 003466-R  LTLUN  000004    NPECNT 000144
ASTKMX 000400    CPIXMK 177770    DPEMX  060014-R  EMXNSQ 100000    FMEPSZ 000400    LTPRX  002000    NPEVSZ 000043
AT     100540-R  CPIXSZ 000010    EIXCNT 000010    EMXNVD 001000    FMIXSZ 000001    LTSKP  004000    NP1MSZ 010000
AVELGT 000025    CQLBS  003204-R  ELSCLS 000007    EMXSZF 002000    FMNFLG 003172-R  LTTOK  010000    NP2MSZ 036000
BATABD 003401    CSTCET 100000    ELSMSK 177761    EMXTRL 010000    FMNPSZ 002260    MINEMX 057426-R  NP2OSZ 000153
BATNO  112130-R  CSTDP  000200    EMACLS 000003    EMXVDC 004000    FNPOSZ 000400    MR     000100    NP3MSZ 000524
BATOVF 000401    CSTDT  000001    EMACUT 001700    EMXVVV 000001    FNPSZ  005734    MSGOUT 066574-R  NP3OSZ 000125
BATREJ 112442-R  CSTDW  000200    EMAMSZ 016000    EMXZNF 004000    FPXCT  003202-R  NAQEX  003214-R  NQLBE  003206-R
BATSUM 102614-R  CSTEFM 010000    EMAQSZ 115750-R  EMX0VD 000200    FPXPT  003176-R  NDBCLS 000006    NULFMN 003514-R
BLKZER 000400    CSTEP  004000    EMATSZ 115752-R  ENCNT  000004    FPXSTK 031014-R  NDBKW  000002    NUMRNG 071304-R
BOTSTK 031016-R  CSTER  002000    EMBCLS 000004    ENDLOC 102612-R  GENQLS 104442-R  NDCHR  000014    NVDEMX 057230-R
BSIZ   000170    CSTESC 000010    EMBCUT 001130    ERRCDE 112134-R  GENTX  106050-R  NDFLID 000006    NVLDC  177774
B$FMN  000020    CSTFDC 000100    EMBMSZ 004400    ERROR  116124-R  GETNDE 065470-R  NDFMEN 000012    NXTNDE 003470-R
B$FST  020000    CSTMIN 000400    EMBOSZ 115754-R  ERRORF 116074-R  HDRSIZ 000023    NDFMN  000010    OCTASC 001564-R
B$MUL  000040    CSTNC  000001    EMBTSZ 115756-R  ERRORV 114610-R  HEADER 077540-R  NDFMS  000005    OVFCLS 115770-R
B$NOT  000100    CSTNF  000020    EMCMSZ 001000    ETX    000003    HSTKMX 000100    NDFSAA 000001    PARBUF 063014-R
B$PUBH 010000    CSTNUM 000100    EMCQSZ 115746-R  EXIT   116306-R  HSZ    067622-R  NDFSAB 000002    PARSER 076332-R
B$PUBL 004000    CSTP   001000    EMCTSZ 115744-R  FAL    020520-R  ILLCHR 117726-R  NDFWD  000000    PAR1   117014-R
B$PUFH 100000    CSTPGR 000010    EMX    021414-R  FALAD  020430-R  INCLUD 067632-R  NDSIZ  000004    PAR2   117016-R
B$PUFL 040000    CSTSEN 000020    EMXCHF 020000    FALMSZ 000500    INIVDC 000010    NDTRM  000010    PASS1  003166-R
B$SUC  000200    CSTSO  000004    EMXDTF 010000    FCSERR 116042-R  INTERW 057252-R  NDTYP  000005    PD     000020
CETEMX 057260-R  CSTSZ  000004    EMXEXF 000001    FLDC   177775    INTRWD 177777    NFALE  003220-R  PFLU   063350-R
CET6B  055414-R  CSTTS  000002    EMXFDB 063026-R  FLIXMK 177740    IOSB   112160-R  NFEMX  057222-R  PFLUMD 065522-R
CFALS  003216-R  CSTVDC 000040    EMXFDC 002000    FLIXSZ 000040    LEXTB1 055614-R  NFLDC  177773    PNBRNG 067662-R
CHRCNT 115764-R  CSTWD  000040    EMXLGT 020414-R  FLUCLS 000001    LEXTB3 056614-R  NIXCNT 000020    POLEND 017412-R
CLBREQ 000400    CSTZ   000002    EMXMCD 000002    FLUCUT 000016    LIST   100152-R  NMRNGF 000100    POOL   003522-R
CLSEMX 060060-R  CSTZER 020000    EMXMCF 001000    FLUID  112164-R  LOCAT  102610-R  NNMASK 177760    PRINT  103552-R
CLSIN  001401    CWPIDX 003446-R  EMXMSZ 016000    FLUIDX 003346-R  LOGCLS 000002    NODEA  003506-R  PRNTH  103704-R
CLSOLY 001001    CWPNDE 003474-R  EMXMTV 040000    FLUNDE 003472-R  LSZ    067620-R  NODEB  003510-R  PRTBUF 100350-R
```

```
PRXFLG 003202-R  QNARG2 000004    QRYMS  003502-R  SPOL   111460-R  TDCBLK 000010    TSTKMX 000400    VI1MSZ 000400
PSTKMX 000024    QNATTR 000006    QRYMSZ 000031    SPXCT  003203-R  TDCMSZ 004000    TTABLE 021220-R  VI2MSZ 000400
QCL    020436-R  QNDCNT 001000    QRYNO  112132-R  SPXPT  003200-R  TOKAND 000007    TTBMSZ 003270    VI3MSZ 000400
QEX    027612-R  QNDLST 042030-R  QRYOK  000001    SPXSTK 030414-R  TOKETX 000017    TTSIZ  020432-R  VLDC   177776
QEXAD  020424-R  QNDNXT 104432-R  QRYPS  003500-R  SSQ    117656-R  TOKFLU 000000    TYPE1B 000000    WRTEMX 062516-R
QEXFPV 174000    QNDSIZ 000010    QRYSZ  002000    START3 112210-R  TOKFMD 000002    TYPE2B 000002    XTABLE 024510-R
QEXFSV 134000    QNFLS  040000    QTFORM 106376-R  STEP   117632-R  TOKHOR 000005    TYPE3B 000004    XTBMSZ 002010
QEXFWV 074000    QNFLU  100000    QT0SUM 000001    STK1   000200    TOKLP  000014    TYPE4B 000006    XTSIZ  020434-R
QEXMSZ 000600    QNOPCD 000000    RNGCDE 067636-R  STK2   000400    TOKNOT 000003    VDCCNT 115766-R  ZERO   112166-R
QLB    026520-R  QNPOOL 032030-R  SDFLG  003174-R  STX    000002    TOKOR  000004    VEC1MX 000375    ZVDEMX 057214-R
QLBAD  020420-R  QNSTE  000006    SDLB   027236-R  SYNTAB 104411-R  TOKPXP 000015    VEC2MX 000375    ZVLDC  177772
QLBMSZ 000516    QNTLST 046032-R  SDLBAD 020422-R  SZCAL1 114644-R  TOKPXS 000013    VEC3MX 000125    $EDMSG 067022-R
QLS    020412-R  QNWIN  000001    SDLBE  003212-R  TDABLK 000004    TOKPXW 000011    VI     020420-R  .MOLUN 112162-R
QLSBAT 020416-R  QRYBAD 002001    SDLBS  003210-R  TDABMX 007764    TOKRP  000016    VIBCUT 000036
QLSBUF 020412-R  QRYBUF 042030-R  SEGOP  177770    TDAMAD 007760    TOKSBD 000001    VIBOSZ 115760-R
QLSHD  020412-R  QRYDEL 117276-R  SKPZER 001000    TDBBLK 000003    TOKSTK 030413-R  VIBTSZ 115762-R
QLSIZ  010000    QRYEND 046030-R  SLBEG  063330-R  TDBCLS 000005    TOPSTK 032016-R  VILGT  020416-R
QLSSUM 100574-R  QRYFAS 003504-R  SLBMSZ 000354    TDBMAD 007775    TRMCUT 000040    VINXT  063172-R
QNARG1 000002    QRYMAX 020426-R  SNODE  032020-R  TDBOSZ 000074    TSKNAM 112122-R  VI0MSZ 000400


*** TASK BUILDER STATISTICS:

    TOTAL WORK FILE REFERENCES: 94001.
    WORK FILE READS: 0.
    WORK FILE WRITES: 0.
    SIZE OF CORE POOL: 6634. WORDS (25. PAGES)
    SIZE OF WORK FILE: 5632. WORDS (22. PAGES)

    ELAPSED TIME:00:00:39
```

FSA-A TRANSLATOR (QT1) MACRO M1110 27-MAR-80 13:07
TABLE OF CONTENTS

```
                        ; QTSIZE.MAC    "QUERY TRANSLATORS BUFFER SIZE CONFIGUATION FILE"
                        ;       THIS FILE CONTROLS THE SIZE OF ALL BUFFERS THAT CAN VARY
                        ;       IN SIZE DUE TO THE AMOUNT OF QUERIES OR OTHER SUCH PARAMETERS
                        ;       THAT WOULD BE CHARACTERISTIC OF A SITE.  THESE BUFFERS ARE
                        ;       IN QT0, QT1, QT2, OR QT3.
                        ;
                        ;---QT3---BUFFERS-----------------------------------------------------------
                        ;
000125                  VEC3MX==85.                             ;MAXIMUM # CWP'S IN BATCH
000025                  AVELGT==3*2*7/2.                        ;AVERAGE CWP SIZE IN TDCT (BYTES)
000400                  VI3MSZ==VEC3MX+2/256.+1*256.            ;SIZE OF VI (NEAREST BLOCK IN BYTES)
001000                  EMCMSZ==VEC3MX/51.+1*256.               ;SIZE OF EMC (NEAREST BLOCK)
004000                  TDCMSZ==VEC3MX*AVELGT+8./N.BUFW+1*N.BUFW;SIZE OF TDCT
000010                  TDCBLK==TDCMSZ/256.                     ;VIRTUAL BLOCK SIZE OF TDCT
000524                  NP3MSZ==VEC3MX*4                        ;SIZE OF NODE POOL
000125                  NP3OSZ==VEC3MX.                         ;SIZE OF NODE POOL OVERFLOW AREA
                        ;
                        ;-----QT2----BUFFERS-------------------------------------------------------
                        ;
000375                  VEC2MX==253.                            ;MAXIMUM # VECTORS
000400                  VI2MSZ==VEC2MX+2/256.+1*256.            ;SIZE OF VI
004400                  EMBMSZ==9.*256.                         ;SIZE OF EMB
000003                  TDBBLK==3                               ;# BLOCKS (N.BUFW) IN TDCT BUFFER
000074                  TDBOSZ==3*20.                           ;SIZE OF TDCT BUFFER OVERFLOW
007775                  TDBMAD==4093.                           ;MAXIMUM ADDRESS (48 BITS) IN TDCT
000020                  NIXCNT==16.                             ;# INDEX POINTERS FOR NORMAL NODES
177760                  NNMASK==177760                          ;MASK FOR NORMAL INDEX
000010                  EIXCNT==8.                              ;# INDEX PTRS FOR ELSE NODES
177761                  ELSMSK==177761                          ;MASK FOR ELSE INDEX
000004                  ENCNT==4                                ;# ENTRIES TO USE FOR HASH
036000                  NP2MSZ==256.*60.                        ;SIZE OF NODE POOL
000153                  NP2OSZ==7+100.                          ;SIZE OF NODE POOL OVERFLOW AREAR
                        ;
                        ;------QT1-----BUFFERS-----------------------------------------------------
                        ;
000375                  VEC1MX==253.                            ;MAX. # TERMS IN FSA-A
000400                  VI1MSZ==VEC2MX+2/256.+1*256.            ;SIZE OF VI
000004                  TDABLK==4                               ;# BLOCKS (N.BUFW) IN TDCT BUFFER
007764                  TDABMX==<TDABLK*<N.BUFW-4)>+4
016000                  EMAMSZ==7.*4*256.                       ;SIZE OF EMA
007760                  TDAMAD==340.*12.                        ;MAX. ADDRESS IN TDCT
010000                  NP1MSZ==4096.                           ;SIZE OF NORMAL NODE POOL
000144                  NPECNT==100.                            ;# NODES IN ELSE POOL
000043                  NPEVSZ==35.                             ;# VECTORS IN ELSE NODES
                        ;
                        ;-----FLU---MODIFIER----BUFFERS--------------------------------------------
                        ;
002260                  FMNPSZ==1200.                           ;SIZE OF NORMAL FLU MOD NODE POOL
000400                  FMEPSZ==256.                            ;SIZE OF ELSE FLU MOD NODE POOL
                        ;
                        ;------QT0------BUFFERS-----------------------------------------------------
                        ;
002000                  QRYSZ==N.BUFW.                          ;SIZE OF QUERY BUFFER
000040                  FLIXSZ==32.                             ;# INDEX PTRS FOR TERMS
177740                  FLIXMK==177740                          ;MASK FOR TERM INDEX
000010                  CPIXSZ==8.                              ;# INDEX PTRS FOR CWP'S
177770                  CPIXMK==177770                          ;MASK FOR CWP INDEX
```

```
                                FMIXSZ==1                               ;# INDEX PTRS FOR FLU-MOD'S
                                FNPSZ==3036.                            ;FLU NODE POOL SIZE
                                FNPOSZ==256.                            ;SIZE OF POOL OVERFLOW AREA
                                TSTKMX==256.                            ;MAX. # TOKENS TO BE PUSHED
                                ASTKMX==256.                            ;MAX. # ARGUMENTS TO BE PUSHED
                                PSTKMX==20.                             ;MAX. # PROX NODES IN A CHAIN
                                HSTKMX==64.                             ;MAX. NESTING OF QUERY
                                QNDCNT==512.                            ;# NODES IN LOGIC POOL
                                VI0MSZ==VI1MSZ                          ;ASSUME MAX VI IS IN QT1
                                .IIF LT,VI0MSZ-VI2MSZ,VI0MSZ==VI2MSZ    ; IF NOT, RESET QT0'S TO QT2'S VI SIZE
                                EMXMSZ==EMAMSZ                          ;ASSUME MAX EMX IS IT QT1
                                .IIF LT,EMXMSZ-EMBMSZ,EMXMSZ==EMBMSZ    ; IF NOT, RESET TO QT2'S
                                ;
                                ;-----ERROR AND CLOSE CODES----------------------------------------
                                ;
                                TRMCUT==32.     ;TERM CUT-OFF
                                FLUCUT==14.     ;FLU CUT-OFF
                                EMACUT==TRMCUT*30.      ;EMA CUT-OFF
                                VIBCUT==30.             ;VIB CUT-OFF
                                EMBCUT==VIBCUT*20.      ;EMB CUT-OFF
                                ;
                                FLUCLS==1       ;CLOSED DUE TO FLU COUNT
                                LOGCLS==2       ;CLOSED DUE TO LOGIC COUNT (QLB,SDLB,QEX)
                                EMACLS==3       ;CLOSED DUE TO EMA SIZE
                                EMBCLS==4       ;CLOSED DUE TO EMB SIZE
                                TDBCLS==5       ;CLOSED DUE TO TDCTB SIZE
                                NDBCLS==6       ;CLOSED DUE TO QT2 NODE POOL SIZE
                                ELSCLS==7       ;CLOSED DUE TO OTHER CONDITIONS (FLU POOL,ETC)
                                ;
```

```
                        .TITLE  FSA-A TRANSLATOR (QT1)
                        .SBTTL
                        .SBTTL  MODULE NAME:     QUERY TRANSLATOR (FSA-A)
                        .SBTTL  AUTHOR:
                        .SBTTL
                        .SBTTL
                        .SBTTL
                        .SBTTL
                        .SBTTL  RELEASE DATE:    18 OCT 1978
                        .SBTTL  RELEASE NUMBER:  SL120024
                        .SBTTL
                        .SBTTL  **********
                        .SBTTL  FSA-A TRANSLATOR (QT1)
                        .SBTTL  **********
                ;
                ;   QT1 READS FROM DISK THE INTERIM DATA STRUCTURES (EMATRIX.EMA)
                ;   GENERATED BY QT0.  THE 4-WORD FILE DESCRIPTOR (FDSC) OF EMATRIX.EMA
                ;   IS SENT TO QT1 BY QTS.
                ;   LAYOUT OF EMATRIX.EMA IS AS FOLLOWS:
                ;
                ;       VIRTUAL BLOCK 1          : VI
                ;       VIRTUAL BLOCKS 2-12.     : EMA
                ;
                ;   BASED ON EMA AND VI, QT1 BUILDS THE "TERM DETECTOR
                ;   CONTROL TABLE" (TDCT) FOR FSA-A, WRITES THIS STRUCTURE TO DISK
                ;   (FILE NAME = TDCTA     .FSA), SENDS TO QTS THE 4-WORD FDSC OF
                ;   THE FILE AND EXITS.
                ;
                ;   ASSEMALY PARAMETERS:
                ;       MAC>QT1=P,M,T,QT1
                ;
                ;   TASK BUILD PARAMETERS:
                ;       MCR>TKB @QT1TKB.CMD
                ;       TKB>QT1,LP=QT1,MSGOUT
                ;       TKB>/
                ;       TKB>TASK=QT1
                ;       TKB>/
                ;
                ;
```

```
                                        .SBTTL  TASK IMMEDIATES AND VARIABLES
                                ;
                                ;
                                ;
000004                          LUNFIL  =       4                       ;LUN DEFINITION FOR FCS
000002                          EFN.2   =       2                       ;EVENT FLAG TWO
001000                          BLOCK   =       512.                    ;BLOCK LENGTH (BYTES)
100000                          BITNIB  ==      BIT15                   ;IF 0 : FIRST NIBBLE
                                                                        ;IF 1 : SECOND NIBBLE
                                ;
                                ;
000030                          TDCT1   =       30              ;TDCT OFFSET OF FIRST FSA STATE
000036                          FREE1   =       36              ;TDCT OFFSET OF FIRST FREE STATE
                                ;
177632                          ER102   =       -102.                   ;NODE NOT IN QUEUE
177630                          ER104   =       -104.                   ;NODE ADDRESS NOT IN REGION
177616                          ER114   =       -114.                   ;NO FREE NODES
177614                          ER116   =       -116.                   ;NODE NOT LONG ENOUGH
000524                          STPBLK  ==      <N:BUFW-4>/3            ;# STATES PER BLOCK
```

```
                        .SBTTL  LOCAL MACROS
                ;
                ;
                ;
                .MACRO  GETNXT  A,?B,?C,?D
                        MOV     NXTOFF,A                ;REG = OFFSET OF NEXT FREE TDCT STATE
                        CMP     A,TDCMAX                ;TDCT BUFFER OVERFLOW?
                        BLT     B                       ;BRANCH IF NO
                        CMP     NXTADD,#TDAMAD          ;TD MEMORY OVERFLOW?
                        BLT     C                       ;BRANCH IF NO
                        JSR     PC,ERROR2               ;MEMORY OVERFLOW
                C:      CMP     LOWOFF,#<N.BUFB-10>     ;ARE ALL STATES IN BLOCK 1 COMPLETE?
                        BGE     D                       ;BRANCH IF YES
                        JSR     PC,ERROR2               ;TDCT BUFFER NOT BIG ENOUGH
                D:      JSR     PC,WRTTDC               ;WRITE NEXT BLOCK OF TDCT
                        JSR     PC,MOVTDC               ;COMPRESS TDCT BUFFER
                        MOV     NXTOFF,A                ;RESTORE NEXT FREE TDCT STATE
                B:
                .ENDM
```

```
                                        .SBTTL  TASK BUFFERS
                                ;
000000                                  .PSECT  DATA
                                ;
                                ;
000000                          IOST:   .BLKW   2               ;IO STATUS BUFFER
000004  000000                 PARBUF: .WORD   0               ;.XQIO PARAM: 1 - BUFFER ADDS
000006  000000                         .WORD   0               ;     "     "   2 - BUFFER LENGTH (BYTES)
000010  000000                         .WORD   0               ;     "     "   3
000012  000000                         .WORD   0               ;     "     "   4 - VIRTUAL DISK ADDS (HI)
000014  000001                         .WORD   1               ;     "     "   5 -     "       "     " (LO)
000016  000000                 NIBMSK: .WORD   0               ;NIBBLE (1 OR 2) MASK
000020  000000                 NIBPOS: .WORD   0               ;CURRENT NIBBLE POSITION NUMBER
000022  000000                 R10:    .WORD   0               ;COLUMN COUNTER
000024  000000                 R11:    .WORD   0               ;CURRENT TDCT STATE'S OFFSET
000026  000000                 R12:    .WORD   0               ;CURRENT TDCT STATE'S ADDRESS
000030                          VVEC::  .BLKW   VI1MSZ+4        ;TEMPORARY BUFFER FOR V-VECTOR
001040  066563  000000          QTS:    .RAD50  /QTS   /
001044                          RECBUF: .BLKW   2               ;SENDING TASK'S NAME
001050                          SNDBUF: .BLKW   13.
```

```
                                        .SBTTL  DPB AND FDB DEFINITIONS
                                ;
                                ;
                                ;
                                        .MCALL  MOUT$S,EXIT$S
                                        .MCALL  GTIM$S,RCVD$C,SDAT$C,RSUM$C
                                ;
                                ;
                                ;
                                ;   FDB DEFINITIONS
                                ;
                                        .MCALL  FDBDF$,FDAT$A,FDRC$A,FDBK$A,FDOP$A,FDOP$R,FSRSZ$
                                        .MCALL  FINIT$,OFNB$W,OPEN$R,CLOSE$
                                        .MCALL  NMBLK$,FDBK$R,FDAT$R,WRITE$,WAIT$
                                ;
                                ;
                                ;   FDB FOR EMATRIX AND TDCTA
                                ;
001102                          EMAFDB::FDBDF$
001242                                  FDAT$A  R,VAR,FD,BLK,,50.
001242                                  FDRC$A  FD,RWM
001242                                  FDBK$A  EMALGT,512,,,EFN,2,IOST
001242                                  FDOP$A  LUNFIL,,EMADNB
001242                          EMADNB: NMBLK$  EMATRIX,EMA,,SY,0
001300                                  FSRSZ$  0,,DATA
```

```
                                        .SBTTL  MAIN PROCESSING LOOP
000000                                  .PSECT  CODE
                                ;
                                ;   RECEIVE PACKET FROM QTS
                                ;
000000                          START:
000000                                  RCVD$C  QTS,RECBUF,CODE,DIRERR
000014                                  FINIT$
                                ;
                                ;   READ EMATRIX.EMA FILE
                                ;
000020  012700  001102'                 MOV     #EMAFDB,R0              ;R0->FDB
                                ;
                                ;   ZERO OUT FILENAME BLOCK
                                ;
000024  062700  000102                  ADD     #F.FNB,R0               ;R0->FNB
000030  012701  000036                  MOV     #S.FNB,R1               ;R1 = BYTE LENGTH OF FNB
000034  006201                          ASR     R1                      ;R1 = WORD LENGTH OF FNB
000036  005020                  5$:     CLR     (R0)+
000040  077102                          SOB     R1,5$
                                ;
                                ;   LOAD EMA'S FID, DEVICE NAME AND UNIT NUMBER INTO FNB
                                ;
000042  012700  001102'                 MOV     #EMAFDB,R0              ;R0-> FDB
000046  062700  000102                  ADD     #F.FNB,R0               ;R0-> FNB
000052  012701  001850'                 MOV     #SNDBUF,R1              ;R1-> DATA RECEIVED FROM QTS
000056  016160  000000  000000          MOV     N.FID(R1),N.FID(R0)     ;FID
000064  016160  000002  000002          MOV     N.FID+2(R1),N.FID+2(R0)
000072  016160  000004  000032          MOV     N.FID+4(R1),N.DVNM(R0)  ;DEVICE NAME
000100  016160  000016  000034          MOV     N.FVER(R1),N.UNIT(R0)   ;DEVICE UNIT NUMBER
                                ;
                                ;   OPEN EMATRIX.EMA FOR READ
                                ;
000106                                  OPEN$R  #EMAFDB                 ;OPEN FILE FOR READ
000124  103010                          BCC     1$
000126  116001  000052                  MOVB    F.ERR(R0),R1            ;R1 = FCS ERROR
000132  010167  001566'                 MOV     R1,PAR2
000136                                  CALL    FCSERR
000142  000167  006612                  JMP     EXIT
                                ;
                                ;   READ VI
                                ;
000146  012767  000010' 000004' 1$:     MOV     #EMALGT,PARBUF          ;PAR 1 = BUFFER ADDS
000154  012767  001000  000006'         MOV     #512.,PARBUF+2          ;PAR 2 = BUFFER LENGTH (BYTES)
000162  012701  000000G                 MOV     #IO.RVB,R1              ;R1 = IO FUNCTION CODE
000166  012702  000005                  MOV     #5,R2                   ;R2 = NO. OF QIO PARAMETERS
000172  012703  000004'                 MOV     #PARBUF,R3              ;R3->QIO PARAMETERS
000176                                  CALL    .XQIO
000202  103010                          BCC     2$
000204  116001  000052                  MOVB    F.ERR(R0),R1            ;R1 = IO ERROR
000210  010167  001566'                 MOV     R1,PAR2
000214                                  CALL    FCSERR
000220  000167  006534                  JMP     EXIT
                                ;
                                ;   TEST TO SEE IF THERE IS AN EMA
                                ;
000224  005767  000010'         2$:     TST     EMALGT                  ;BLANK EMA?
```

```
                                        BNE     4$                      ;BRANCH IF NO
                                ;
                                ;   CLOSE EMA FILE
                                ;
000232                                  CLOSE$  #EMAFDB
                                ;
                                ;   SET UP BLANK TDCTA
                                ;
000242 016702 001686'                   MOV     FSAOFF,R2               ;R2 = NEXT FREE TDCT OFFSET
000246 012762 000000  001630'           MOV     #0,TDCT(R2)             ;SEG 1
000254 016762 001610' 001632'           MOV     NXTADD,TDCT+SEG2(R2)
000262 012762 040000  001634'           MOV     #ST$INX,TDCT+SEG3(R2)
                                ;
                                ;   WRITE TDCTA
                                ;
000270                                  CALL    OPNTDC                  ;OPEN TDCTA FILE
000274                                  CALL    WRTTDC                  ;CREATE TDCTA
                                ;
                                ;   SET UP SNDBUF FOR QTS
                                ;
000300 005067 000000C                   CLR     SNDBUF+SD.SEC
000304 005067 000000C                   CLR     SNDBUF+SD.TIC
000310 016767 001610' 000000C           MOV     NXTADD,SNDBUF+SD.FSA
000316 000167 000524                    JMP     QT1.7
                                ;
                                ;   READ EMA
                                ;
000322 012767 001010' 000004' 4$:       MOV     #EMA,PARBUF             ;PAR 1 = BUFFER ADDS
000330 016701 000010'                   MOV     EMALGT,R1               ;R1 = EMA LENGTH IN BLOCKS
000334 070127 001000                    MUL     #512.,R1                ;R1 = BYTE LENGTH
000340 010167 000006'                   MOV     R1,PARBUF+2             ;PAR 2 = FILE LENGTH
000344 012767 000002  000014'           MOV     #2,PARBUF+10            ;START READ AT VIRTUAL BLOCK 2
000352 012701 000000G                   MOV     #IO.RVB,R1              ;R1 = IO FUNCTION CODE
000356 012702 000005                    MOV     #5,R2                   ;R2 = NO. OF QIO PARAM'S
000362 012703 000004'                   MOV     #PARBUF,R3              ;R3-> PARAMETERES
000366                                  CALL    .XQIO
000372 103010                           BCC     3$
000374 116001 000052                    MOVB    F.ERR(R0),R1            ;R1 = IO ERROR
000400 010167 001566'                   MOV     R1,PAR2
000404                                  CALL    FCSERR
000410 000167 006344                    JMP     EXIT
                                ;
                                ;   CLOSE FILE
                                ;
000414                          3$:     CLOSE$  R0
                                ;
                                ; OPEN FILE FOR TDCTA,FSA
                                ;
000420                                  CALL    OPNTDC
                                ;
                                ;   INITIALIZE TRANSLATION PROCESS
                                ;
000424                          QT1.1:  GTIM$S  #GTIM1                  ;GET TIME PARAM'S AT START
000436 012703 000000'                   MOV     #VMASK,R3               ;R3->VMASK = VI
000442 016763 001570' 000006            MOV     ELSEDF,N.ELSE(R3)       ;LOAD ELSE DEF. ADDS INTO VMASK
000450 005063 000010                    CLR     N.POS(R3)               ;ZERO OUT NIBBLE POSITION NO. FIELD
                                ;
```

```
                                        ;   FOR FIRST TDCT STATE GO DIRECTLY TO MASKEM
                                        ;       R3-> VMASK
                                        ;
000454  000167  000070                          JMP     QT1.5
                                        ;
                                        ;   INNER LOOP TO PROCESS INCOMPLETE TDCT STATES WITH SAME NIBBLE
                                        ;   POSITION NUMBER.
                                        ;
                                        ;   TEST TO SEE IF FSA STATE POINTED TO BY FSAOFF IS COMPLETE.
                                        ;
000460  016700  001606'          QT1.2:  MOV     FSAOFF,R0               ;R0 = OFFSET OF CURRENT TDCT STATE
000464  032760  000001  001634'          BIT     #BIT0,TDCT+SEG3(R0)     ;COMPLETED STATE?
000472                                  BOFF    QT1.6                   ;BRANCH IF YES
                                        ;
                                        ;   CURRENT STATE IS INCOMPLETE.  CHECK ITS POSITION NUMBER.
                                        ;
000474  122760  000100  001635'          CMPB    #100,TDCT+SEG3+1(R0)    ;SHOULD BE AN INDEX STATE
000502  001405                          BEQ     1$                      ;BRANCH IF OK
000504  012767  000001  001564'          MOV     #1,PAR1
000512  000167  006222                  JMP     ERRINT                  ;ERROR !!!!
000516  016003  001632'          1$:     MOV     TDCT+SEG2(R0),R3        ;R3->NODE CONTAINING V-VECTOR
000522  126763  000020' 000010          CMPB    NIBPOS,N.POS(R3)        ;DO NIBBLE POSITION NO'S MATCH?
000530  001011                          BNE     QT1.6                   ;BRANCH IF NO
000532                                  CALL    GVMASK                  ;GENERATE NEW VMASK
000536  103004                          BCC     QT1.5
000540                                  CALL    NODERR
000544  000167  006210                  JMP     EXIT
                                        ;
                                        ;   ROUTINE TO GENERATE NEW TDCT STATES BY MASKING THE E-MATRIX
                                        ;   WITH THE NEW VMASK.
                                        ;
000550                          QT1.5:  CALL    MASKEM                  ;MASK E-MATRIX
                                        ;
                                        ;   CURRENT TDCT STATE IS COMPLETED, GO TO NEXT STATE.
                                        ;
000554  062767  000006  001606' QT1.6:  ADD     #L$STAT,FSAOFF          ;FSAOFF=OFFSET OF NEXT TDCT STATE
000562  005267  001604'                 INC     FSAADD                  ;INCREMENT CURRENT TDCT ADD'S
                                        ;
                                        ;   DO UNTIL FSAOFF IS > OR = NXTOFF
                                        ;
000566  026767  001606' 001612'          CMP     FSAOFF,NXTOFF
000574  002731                          BLT     QT1.2                   ;DO INNER LOOP
                                        ;
                                        ;   CURRENT NIBBLE POSITION IS DONE
                                        ;
000576  005267  000020'          QT1.4:  INC     NIBPOS                  ;INCREMENT NIBBLE POSITION COUNTER
000602                                  CALL    SWAPNP                  ;SWAP NODE POOLS
                                        ;
                                        ;   DO UNTIL LOWOFF POINTS TO AN INCOMPLETE TDCT STATE, OR UNTIL
                                        ;   LOWOFF > OR = NXTOFF
                                        ;
000606  016700  001602'                 MOV     LOWOFF,R0               ;R0 = OFFSET OF LOWEST COMPLETED STATE
000612  032760  000001  001634' 1$:     BIT     #BIT0,TDCT+SEG3(R0)     ;STATE INCOMPLETE?
000620                                  BON     2$                      ;BRANCH IF YES
                                        ;
                                        ;   INCREMENT LOWOFF TO POINT TO NEXT TDCT STATE
                                        ;
```

```
 299 000622  062700  000006                ADD     #L$STAT,R0              ;R0 = OFFSET OF NEXT TDCT STATE
 300 000626  005267  001600'               INC     LOWADD                  ;INC. ADD'S.
 301 000632  020067  001612'               CMP     R0,NXTOFF               ;IS LOWOFF > OR = NXTOFF ?
 302 000636  002765                        BLT     1$                      ;LOOP IF NO
 303 000640  000406                        BR      QT1.3                   ;TRANSLATION IS DONE
 304                                ;
 305                                ;   R0 = OFFSET OF LOWEST INCOMPLETE TDCT STATE
 306                                ;
 307 000642  010067  001602'        2$:     MOV     R0,LOWOFF
 308 000646  010067  001606'               MOV     R0,FSAOFF
 309 000652  000167  177602                JMP     QT1.2                   ;LOOP ON LOWOFF
 310                                ;
 311                                ;   TRANSLATION PROCESS IS COMPLETED.
 312                                ;
 313 000656                         QT1.3:  GTIM$S  #SECBUF
 314                                ;
 315                                ;   WRITE FIRST OF REMAINING TDCT BLOCKS
 316                                ;
 317 000670                              CALL    WRTTDC                  ;WRITE 1ST BUFFER
 318                                ;
 319                                ;   WRITE REMAINING TDCT BLOCKS
 320                                ;
 321 000674  012702  000003              MOV     #<TDABLK-1>,R2          ;R2 = # BLOCKS REMAINING
 322 000700  012703  001630'             MOV     #TDCT,R3                ;R3 = TDCT
 323 000704  066703  001612'             ADD     NXTOFF,R3               ;R3 = END OF TDCT
 324 000710  012701  001620'             MOV     #TDCBUF,R1              ;R1 = 1ST BLOCK OF TDCT BUFFER
 325 000714  062701  004000        3$:   ADD     #N.BUFB,R1              ;R1 = NEXT BLOCK OF TDCT
 326 000720  020301                      CMP     R3,R1                   ;ANY MORE DATA IN TDCT?
 327 000722  101436                      BLOS    2$                      ;NO
 328 000724  162701  000010              SUB     #10,R1                  ;YES: BACK UP 4 WDS FOR BLOCK HDR
 329 000730  016711  001620'             MOV     TDCBUF,(R1)             ;SET UP BLOCK HEADER
 330 000734  005061  000002              CLR     2(R1)
 331 000740  005061  000004              CLR     4(R1)
 332 000744  005061  000006              CLR     6(R1)
 333 000750                              WRITE$  #EMAFDB,R1              ;WRITE THIRD (AND LAST) BLOCK
 334 000764                              WAIT$   #EMAFDB
 335 000774  105767  000000'             TSTB    IOST
 336 001000  003006                      BGT     1$
 337 001002  116701  000000'             MOVB    IOST,R1
 338 001006  010167  001566'             MOV     R1,PAR2
 339 001012                              CALL    FCSERR
 340                                ;
 341                                ;   LOOP BACK TO WRITE NEXT BLOCK
 342                                ;
 343 001016  077242                 1$:  SOB     R2,3$
 344                                ;
 345                                ;   CALCULATE TIME ELAPSED DURING TRANSLATION
 346                                ;
 347 001020                         2$:  CALL    DELTIM
 348 001024  016767  000206' 000000C    MOV     SECBUF,SNDBUF+SD.SEC    ;LOAD ELAPSED SECONDS INTO SNDBUF
 349 001032  016767  000210' 000000C    MOV     SECBUF+2,SNDBUF+SD.TIC  ;  "      "    TICS      "      "
 350 001040  016767  001610' 000000C    MOV     NXTADD,SNDBUF+SD.FSA    ;LOAD NUMBER OF FSA STATES
 351                                ;
 352                                ;   NOTIFY QTS
 353                                ;
 354 001046                         QT1.7:  SDAT$C  QTS,SNDBUF,,CODE
 355 001054  103004                      BCC     1$
```

```
    356 001056                                      CALL    DIRERR
    357 001062  000167  005672                      JMP     EXIT
    358 001066                              1$:     RSUM$C  QTS,CODE
    359 001074  103002                              BCC     2$
    360 001076                                      CALL    DIRERR
    361                                     ;
    362                                     ;   CLOSE FILE
    363                                     ;
    364 001102                              2$:     CLOSE$  #EMAFDB
    365 001112                                      EXIT$S                          ;EXIT
```

```
                                        .SBTTL  OPEN TDCTA FILE
                                ;
                                ;   SUBROUTINE TO OPEN FILE FOR TDCTA.FSA
                                ;
                                ;
                                ;     BUILD FNB USING DATA RECEIVED FROM QTS IN SNDBUF
                                ;
001120                          OPNTDC: SAVE    R0,R1
001124  012700  001102'                 MOV     #EMAFDB,R0              ;R0->FDB
001130  062700  000102                   ADD     #F.FNB,R0               ;R0-> FNB
001134  012701  001050'                 MOV     #SNDBUF,R1              ;R1-> DATA RECEIVED FROM QTS
001140  005060  000000                   CLR     N.FID(R0)               ;CLEAR FID
001144  005060  000002                   CLR     N.FID+2(R0)
001150  005060  000004                   CLR     N.FID+4(R0)
001154  016160  000006  000006           MOV     N.FNAM(R1),N.FNAM(R0)   ;FILENAME
001162  016160  000010  000010           MOV     N.FNAM+2(R1),N.FNAM+2(R0)
001170  016160  000012  000012           MOV     N.FNAM+4(R1),N.FNAM+4(R0)
001176  016160  000014  000014           MOV     N.FTYP(R1),N.FTYP(R0)   ;FILE TYPE
001204  005060  000016                   CLR     N.FVER(R0)              ;NEXT AVAIL. VERSION NUMBER
001210  005060  000020                   CLR     N.STAT(R0)
001214  005060  000022                   CLR     N.NEXT(R0)
001220  016160  000024  000024           MOV     N.DID(R1),N.DID(R0)     ;UFD'S FID
001226  016160  000026  000026           MOV     N.DID+2(R1),N.DID+2(R0)
001234  016160  000030  000030           MOV     N.DID+4(R1),N.DID+4(R0)
001242  016160  000020  000032           MOV     N.STAT(R1),N.DVNM(R0)   ;DEVICE NAME
001250  016160  000022  000034           MOV     N.NEXT(R1),N.UNIT(R0)   ;DEVICE UNIT NUMBER
001256  012700  001102'                 MOV     #EMAFDB,R0              ;R0->FDB
                                ;
                                ;   SPECIFY BLOCK SIZE TO BE N.BUFB BYTES (2048.)
                                ;
001262                                  FDBK$R  R0,,#N.BUFB
                                ;
                                ;   ALLOCATE SPACE FOR NEW FILE - 48. SECTORS OR 12 BLOCKS
                                ;
001270                                  FDAT$R  R0,,,#48.
                                ;
                                ;   OPEN FILE FOR WRITE
                                ;
001276                                  OFNB$W  R0
001310  103010                         BCC     2$
001312  116001  000052                 MOVB    F.ERR(R0),R1            ;R1 = FCS ERROR
001316  010167  001566'                 MOV     R1,PAR2
001322                                  CALL    FCSERR
001326  000167  005426                 JMP     EXIT
                                ;
                                ;   LOAD FID OF NEW TDCTA FILE INTO SNDBUF FOR TRANSMISSION TO QTS
                                ;
001332  012701  001050'         2$:     MOV     #SNDBUF,R1              ;R1->SNDBUF
001336  016061  000102  000000           MOV     N.FID+F.FNB(R0),F.FID(R1)       ;FID
001344  016061  000104  000002           MOV     N.FID+F.FNB+2(R0),F.FID+2(R1)
001352  016061  000120  000004           MOV     N.FVER+F.FNB(R0),F.FVR(R1)      ;VERSION NO.
001360                                  RESTOR  R0,R1
001364                                  EXIT    OPNTDC
```

```
                                        .SBTTL  MASK E-MATRIX
                                ;
                                ;   PURPOSE OF THIS ROUTINE IS TO USE THE VECTOR IN VMASK TO MASK
                                ;   EACH COLUMN OF THE E-MATRIX AND GENERATE, FOR EACH COLUMN, ANOTHER
                                ;   VECTOR CONSISTING OF THE SEQUENCE NUMBERSOF THE E-MATRIX ROWS
                                ;   WHERE A BIT MATCH WAS DETECTED.  THE CORRESPONDING TRANSISTION
                                ;   SELECT BIT IN THE CURRENT FSA STATE WORD IS SET, UNLESS THE VECTOR
                                ;   LENGTH IS ZERO OR THE ELSE DEFAULT OR ELSE DEFAULT OVERRIDE APPLIES.
                                ;
                                ;   EACH VECTOR GENERATED IS FIRST COMPARED WITH THE ELSE DEFAULT VECTOR (ELSEDF).
                                ;   IF VECTORS MATCH, NO FURTHER ACTION IS NEEDED.  IF NO MATCH, AND ELSE DEF.
                                ;   OVERRIDE BIT IS SET IN CURRENT FSA STATE WORD, VECTOR IS COMPARED
                                ;   WITH ELSEDEFAULT OVERRIDE VECTOR (ELSEDO).  IF VECTORS MATCH, NO
                                ;   FURTHER ACTION IS NEEDED.  IF NO MATCH, VECTOR IS COMPARED WITH THE
                                ;   VECTORS CONTAINED IN THE NODE CHAIN OF THE CURRENT E-MATRIX COLUMN.
                                ;   IF A VECTOR MATCH IS FOUND, THE FSA STATE ASSIGNED TO THE NEWLY
                                ;   GENERATED VECTOR IS DESIGNATED AS A JUMP STATE, ITS NEXT STATE ADDRESS
                                ;   THE STATE ADDRESS OF THE MATCHING VECTOR.  IF NO MATCH, VECTOR
                                ;   IS INSERTED IN A BLANK NODE AND NODE IS MERGED INTO NODE CHAIN.
                                ;   (NODES IN CHAIN ARE ORDERED IN ASCENDING ORDER OF VECTOR LENGTH
                                ;   AND IN ASCENDING ORDER OF SEQ. NUMBERS WITHIN SAME LENGTH GROUPS)
                                ;
                                ;   ON ENTRY,
                                ;
                                ;       R3->VMASK
                                ;       FSAOFF = CURRENT TDCT OFFSET
                                ;       FSAADD = CURRENT TDCT ADDRESS
                                ;       NXTADD = NEXT AVAILBLE TDCT ADDRESS
                                ;       NXTOFF = NEXT AVAILBLE TDCT OFFSET
                                ;
001366                          MASKEM:
001366  032763  100000  000012          BIT     #BITNIB,N.LGT(R3)               ;TEST FOR 1ST OR 2ND NIBBLE
001374                                  BEQ     1$                              ;BRANCH IF 2ND NIBBLE
001376  005067  000016'                CLR     NIBMSK                          ;CURRENT VMASK REPRESENTS 1ST NIBBLE
001402  000403                          BR      2$
001404  012767  100000  000016' 1$:     MOV     #BITNIB,NIBMSK                  ;CURRENT VMASK REPRESENTS 2ND NIBBLE
                                ;
                                ;   CHECK TO SEE IF VLDC COLUMN HAS ALREADY BEEN CONSIDERED FOR
                                ;   CURRENT TDCT STATE.  (IF BIT 1 OF SEG3 IS SET, BYPASS VLDC PROCESSING)
                                ;
001412  016702  001606'         2$:     MOV     FSAOFF,R2               ;R2=OFFSET OF CURRENT TDCT STATE
001416  010267  000024'                 MOV     R2,R11                  ;R11 = CURRENT TDCT STATE'S OFFSET
001422  016767  001604' 000026'         MOV     FSAADD,R12              ;R12 = CURRENT TDCT STATE'S ADDRESS
001430  032762  000002  001634'         BIT     #BIT1,TDCT+SEG3(R2)     ;CHECK BIT 1 OF SEG3
001436                                  BOFF    3$                      ;BRANCH IF VLDC COLUMN HAS NOT
                                                                        ;YET BEEN CONSIDERED
001440  000414                          BR      5$                      ;JUMP IF VLDC COLUMN IS DONE
001442  016301  000012          3$:     MOV     N.LGT(R3),R1            ;R1 = NO. OF ENTRIES IN VMASK
001446  042701  100000                  BIC     #BITNIB,R1              ;CLEAR NIBBLE INDICATOR FLAG
001452  022701  000001                  CMP     #1,R1                   ;LENGTH = 1?
001456  001003                          BNE     4$                      ;BRANCH IF NO
001460                                  CALL    SEQSTA                  ;BUILD SEQUENTIAL STATES
001464  103051                          BCC     MSK.12                  ;BRANCH IF CURRENT STATE DONE
                                ;
                                ;   DO VLDC COLUMN OF EMATRIX
                                ;
001466                          4$:     CALL    VDC
```

```
                                ;
                                ;   DO FLDC COLUMN OF EMATRIX
                                ;
001472                      5$:     CALL    FDC
                                ;
                                ;   DO SEARCHABLE COLUMNS OF EMATRIX
                                ;
001476                      6$:     CALL    SCH
                                ;
                                ;   IF FIRST NIBBLE, PROCESS COLUMN OF BIT0 (INTERWORD BIT). OTHERWISE
                                ;   GO TO MSK.12
                                ;
001502  005767  000016'     MSK.11: TST     NIBMSK                  ;FIRST OR SECOND NIBBLE?
001506  001040                      BNE     MSK.12                  ;BRANCH IF 2ND NIBBLE
                                ;
                                ;   FIRST NIBBLE
                                ;
001510                      6$:     SAVE    R1                      ;R1 = LOOP COUNTER
001512  016301  000012              MOV     N.LGT(R3),R1            ;R1 = VMASK LENGTH
001516  042701  100000              BIC     #BITNIB,R1              ;CLEAR NIB 1/2 INDICATOR
001522  010300                      MOV     R3,R0                   ;R0->VMASK
001524  012705  000030'             MOV     #VVEC,R5                ;R5->V-VECTOR
001530  005065  000012              CLR     N.LGT(R5)               ;CLEAR LENGTH FIELD
001534  062700  000014              ADD     #N.VEC,R0               ;R0->1ST OFFSET IN VMASK
001540  062705  000014              ADD     #N.VEC,R5               ;R5-> 1ST OFFSET IN VVEC
001544  012702  000001              MOV     #BIT0,R2                ;R2 = BIT 0
001550  012004              2$:     MOV     (R0)+,R4                ;R4 = E-MATRIX OFFSET
001552  030264  001012'             BIT     R2,EMA+EMXNB1(R4)       ; BIT SET?
001556                              BOFF    1$                      ;BRANCH IF NO
001560  005267  000042'             INC     N.LGT+VVEC              ;INC LENGTH COUNT
001564  010425                      MOV     R4,(R5)+                ;SAVE OFFSET
001566  077110              1$:     SOB     R1,2$                   ;LOOP
001570                              RESTOR  R1
001572  005767  000042'             TST     N.LGT+VVEC              ;TEST NEW VECTOR'S LENGTH
001576  001404                      BEQ     MSK.12                  ;BRANCH IF ZERO
001600                              CALL    INTERW
                                ;
                                ;   PROCESS FLU MODIFIERS (DOC, TYPE, ZONE, SUBZONE) IF ANY.
                                ;
001604                              CALL    FLUMD
                                ;
001610                      MSK.12: EXIT    MASKEM
```

```
                                        ;
                                        ;               .SBTTL  VLDC COLUMN
                                        ;
                                        ;   CHECK VLDC COLUMN OF E-MATRIX, PROVIDED IT HAS NOT ALREADY
                                        ;   BEEN CONSIDERED.
                                        ;       R1 = NUMBER OF ENTRIES IN VMASK
                                        ;
001612  010300                  VDC:      MOV     R3,R0                   ;R0->VMASK
001614  012702  004000                    MOV     #EMXVDC,R2                      ;R2 = VLDC BIT
001620  012705  000030'                   MOV     #VVEC,R5                        ;R5-> V-VECTOR
001624  005065  000012                    CLR     N.LGT(R5)                       ;CLEAR LENGTH FIELD
001630  062700  000014                    ADD     #N.VEC,R0                       ;R0->1ST OFFSET IN VMASK
001634  062705  000014                    ADD     #N.VEC,R5                       ;R5-> 1ST OFFSET IN VVEC
001640  012004                  VDC.1:    MOV     (R0)+,R4                        ;R4 = E-MATRIX OFFSET
001642  030264  001010'                   BIT     R2,EMA(R4)                      ;VLDC BIT SET?
001646                                    BOFF    1$                              ;BRANCH IF NO
001650  005267  000042'                   INC     N.LGT+VVEC                      ;INC LENGTH COUNT
001654  010425                          MOV     R4,(R5)+                        ;SAVE OFFSET
001656  077110                  1$:       SOB     R1,VDC.1                        ;LOOP
001660  005767  000042'                   TST     N.LGT+VVEC                      ;TEST NEW VECTOR'S LENGTH
001664  001002                          BNE     10$                             ;BRANCH IF NOT ZERO
001666  000167  000644                    JMP     11$                             ;JUMP IF ZERO
001672  056767  000016' 000042' 10$:     BIS     NIBMSK,N.LGT+VVEC               ;INSERT NIBBLE INDICATOR FLAG
001700  116367  000010  000040'          MOVB    N.POS(R3),N.POS+VVEC            ;LOAD NIBBLE POSITION NO.
001706  105267  000040'                   INCB    N.POS+VVEC                      ;INCREMENT NIBBLE POS. NO.
                                        ;
                                        ;   NEW VECTOR GENERATED FOR VLDC COLUMN OF E-MATRIX.  NEW VECTOR IN VVEC
                                        ;
001712  012701  000032'                   MOV     #VLDC,R1                ;R1->NODE CHAIN
001716                                    CALL    NSCAN                   ;MATCH IN THE VLDC CHAIN?
001722  103114                          BCC     2$                      ;BRANCH IF NO MATCH
                                        ;
                                        ;   MATCHED A VLDC VECTOR.  SEE IF CURRENT ELSE DEF. STATE IS SAME AS
                                        ;   STATE ADDRESS IN MATCHING NODE.
                                        ;       R2->MATCHING NODE
                                        ;       R3->VMASK
                                        ;
001724  026362  000006  000006          CMP     N.ELSE(R3),N.ELSE(R2)   ;COMPARE ELSE DEF. ADDRESS
001732  001504                          BEQ     3$                      ;IF SAME
001734  010267  001576'                   MOV     R2,C.VLDC               ;RESET CURRENT VLDC NODE ADR
001740                                    GETNXT  R4                      ;GET NEXT OFFSET
002016  016701  000024'                   MOV     R11,R1                  ;R1=OLD TDCT     R4=NEW TDCT
002022  010467  000024'                   MOV     R4,R11                  ;RESET CURRENT OFFSET
002026  062701  001630'                   ADD     #TDCT,R1
002032  062704  001630'                   ADD     #TDCT,R4
002036  011124                          MOV     (R1),(R4)+              ;XFER SEG1
002040  016221  000006                    MOV     N.ELSE(R2),(R1)+        ;SEG1 OF CHANGE DEFAULT
002044                                    SAVE    R3
002046  016703  001610'                   MOV     NXTADD,R3               ;R3=NEXT ADDRESS
002052  010367  000026'                   MOV     R3,R12                  ;RESET CURRENT STATES ADR
002056  011105                          MOV     (R1),R5                 ;R5=NODE ADR OF CURRENT NODE
002060  010524                          MOV     R5,(R4)+                ;XFER SEG2
002062  010321                          MOV     R3,(R1)+                ;SEG1 OF CHANGE DEFAULT
002064  011114                          MOV     (R1),(R4)               ;XFER SEG3
002066  052724  000002                  BIS     #BIT1,(R4)+             ;FLAG VLDC SCAN AS COMPLETE
002072  012721  130000                  MOV     #ST$CNG!ELSBIT,(R1)+    ;SEG3 OF CHANGE DEFAULT
002076  005267  001610'                   INC     NXTADD                  ;ALLOCATE LOC
```

```
 578 002102  062767  000006  001612'          ADD     #6,NXTOFF
 579 002110  016265  000006  000006           MOV     N.ELSE(R2),N.ELSE(R5)   ;CHANGE ELSE DEFAULT IN NODE
 580 002116  016267  000006  000036'          MOV     N.ELSE(R2),VVEC+N.ELSE  ;CHANGE OTHER ELSE FIELDS ALSO
 581 002124  016267  000006  000006'          MOV     N.ELSE(R2),VMASK+N.ELSE
 582 002132  010365  000004                   MOV     R3,N.FSA(R5)            ;CHANGE NODE FSA LOC
 583 002136                                   RESTOR  R3
 584 002140  000167  000372                   JMP     11$
 585                                  ;
 586                                  ;   ELSE DEF. ADDRESSES ARE THE SAME. SAVE NODE ADDRESS OF MATCHING NODE
 587                                  ;   IN C.VLDC AND CONTINUE.
 588                                  ;
 589 002144  010267  001576'          3$:     MOV     R2,C.VLDC               ;SAVE CURRENT VLDC NODE ADDRESS
 590 002150  000167  000362                   JMP     11$
 591                                  ;   DID NOT MATCH A VLDC VECTOR.  LOAD CONTENTS OF CURRENT TDCT STATE INTO
 592                                  ;   NEXT FREE STATE AND MAKE CURRENT TDCT STATE A "CHANGE ELSE DEF." STATE.
 593                                  ;   CHANGE ELSE DEF. TO NEXT AVAILABLE STATE PLUS 1. POINT TO NEXT AVAILABLE
 594                                  ;   STATE AND MERGE VVEC INTO FLDC NODE CHAIN.  GENERATE ANOTHER VVEC TO
 595                                  ;   REPRESENT VLDC SEQ. NUMBERS PLUS SEQ. NUMBERS IMMEDIATELY FOLLOWING
 596                                  ;   EACH VLDC, AND MERGE THIS VVEC INTO FLDC CHAIN.
 597                                  ;       R2->NODE AFTER WHICH NEW NODE CONTAINING VECTOR SHOULD BE INSERTED
 598                                  ;       R3-> VMASK
 599                                  ;
 600 002154                           2$:     GETNXT  R4                      ;R4=OFFSET OF NEXT FREE TDCT STATE
 601 002232  016701  000024'                  MOV     R11,R1                  ;R1=OFFSET OF CURRENT TDCT STATE
 602 002236  016164  001630'  001630'         MOV     TDCT(R1),TDCT(R4)       ;SEG 1
 603 002244  016761  001610'  001630'         MOV     NXTADD,TDCT(R1)         ;SEG 1 = NEXT FREE STATE'S ADD'S
 604 002252  005261  001630'                  INC     TDCT(R1)                ;SEG 1 = NEW DEF. ADDRESS
 605 002256  022124                           CMP     (R1)+,(R4)+             ;R1 AND R4 ARE OFFSETS OF SEG 2
 606 002260  016164  001630'  001630'         MOV     TDCT(R1),TDCT(R4)       ;SEG 2
 607 002266                                   SAVE    R2,R3
 608 002272  016102  001630'                  MOV     TDCT(R1),R2             ;R2 -> NODE
 609 002276  016703  001610'                  MOV     NXTADD,R3               ;R3 = NEXT STATE'S ADDRESS
 610 002302  010361  001630'                  MOV     R3,TDCT(R1)             ;SEG 2 = NEXT STATE ADDRESS
 611 002306  022124                           CMP     (R1)+,(R4)+
 612 002310  016164  001630'  001630'         MOV     TDCT(R1),TDCT(R4)       ;SEG 3
 613 002316  052764  000002  001630'         BIS     #BIT1,TDCT(R4)          ;MARK SEG3 TO INDICATE THAT VLDC
 614                                                                          ;COLUMN IS DONE
 615 002324  012761  120000  001630'         MOV     #ST$CNG,TDCT(R1)        ;SEG 3 = CHANGE ELSE DEF. STATE
 616 002332  052761  010000  001630'         BIS     #ELSBIT,TDCT(R1)
 617 002340  022124                           CMP     (R1)+,(R4)+             ;R4 = OFFSET OF NEXT AVAILABLE TDCT STATE
 618 002342  016767  001612'  000024'         MOV     NXTOFF,R11              ;R11 = OFFSET OF CURRENT TDCT STATE
 619 002350  010467  001612'                  MOV     R4,NXTOFF               ;SAVE OFFSET OF NEXT FREE TDCT STATE
 620 002354  005203                           INC     R3                      ;R3 = FSA ADDRESS OF ELSE DEF. STATE
 621 002356  010362  000006                   MOV     R3,N.ELSE(R2)           ;RESET NODE'S ELSE DEF. ADDS
 622 002362  016762  001610'  000004         MOV     NXTADD,N.FSA(R2)        ;RESET NODE'S FSA ADDRESS
 623 002370  010367  000036'                  MOV     R3,N.ELSE+VVEC
 624 002374  010367  000034'                  MOV     R3,N.FSA+VVEC
 625 002400  010367  000006'                  MOV     R3,N.ELSE+VMASK
 626 002404  016767  001610'  000004'         MOV     NXTADD,N.FSA+VMASK
 627 002412                                   RESTOR  R2,R3
 628 002416  016767  001610'  000026'         MOV     NXTADD,R12              ;R12 = ADD'S OF CURRENT TDCT STATE
 629 002424  005267  001610'                  INC     NXTADD                  ;NEXT FREE STATE ADDRESS
 630                                  ;
 631                                  ;   MERGE VVEC INTO VLDC NODE CHAIN
 632                                  ;       R2->NODE THAT SHOULD PRECEED NEW NODE
 633                                  ;
 634 002430  012701  000032'                  MOV     #VLDC,R1                ;R1-> NODE CHAIN
```

```
635 002434                                          BCC     5$
636 002440  103005                                  MOV     R2,R0                   ;R0 = ERROR CODE
637 002442  010200                                  CALL    NODERR
638 002444                                          JMP     EXIT
639 002450  000167  004304                          MOV     R2,C.VLDC               ;SAVE CURRENT VLDC NODE'S ADDRESS
640 002454  010267  001576'             5$:        ;
641                                                 ;   SET ELSE DEFAULT OVERRIDE BIT IN CURRENT TDCT STATE WORD.
642                                                 ;
643                                                 BIS     #BIT15,N.POS+VVEC       ;MARK VVEC TO REPRESENT ELSE DEF
644 002460  052767  100000  000040'                ;OVERRIDE STATE
645                                                 MOV     R11,R4          ;R4=OFFSET OF CURRENT TDCT STATE WORD
646 002466  016704  000024'                         MOV     NXTADD,TDCT+SEG2(R4)    ;ASSIGN BASE ADDRESS
647 002472  016764  001610' 001632'                 MOV     #ST$INX,TDCT+SEG3(R4)   ;DESIGNATE CURRENT STATE AS INDEX STATE
648 002500  012764  040000  001634'                 BIS     #EOBIT,TDCT+SEG3(R4)    ;SET ELSE DEF, OVERRIDE BIT
649 002506  052764  010000  001634'                 MOV     #FLDC,R1                ;R1->FLDC NODE CHAIN
650 002514  012701  000040'                         CALL    NSCAN
651 002520                                          BCS     12$                     ;BRANCH IF MATCH
652 002524  103405
653                                                 ;
654                                                 ;   MAKE NEXT AVAILABLE TDCT STATE AN INDEX STATE, MARK IT INCOMPLETE (SET
655                                                 ;   BIT0 OF SEG3), INSERT VVEC INTO AN EMPTY NODE, INSERT THIS
656                                                 ;   NEW NODE INTO THE FLDC NODE CHAIN AND PUT ADDRESS OF NODE INTO SEG2 OF
657                                                 ;   ASSIGNED STATE.
658                                                 ;
659                                                 ;       R2-> NODE THAT SHOULD PRECEED NEW NODE
660                                                 ;
661 002526  012701  000040'                         MOV     #FLDC,R1                ;R1->NODE CHAIN INDEX
662 002532                                          CALL    NEWSTA
663                                                 ;
664                                                 ;       R1 = OFFSET OF NEW NODE
665                                                 ;       R2-> NEW NODE
666                                                 ;
667 002536                                  11$:    EXIT    VDC
668                                                 ;
669                                                 ;   MATCHED EXISTING FLDC NODE ??????
670                                                 ;
671 002540  012767  000002  001564'         12$:    MOV     #2,PAR1
672 002546  000167  004166                          JMP     ERRINT
```

```
                                        .SBTTL  FLDC COLUMN
                                ;
                                ;   CHECK FLDC COLUMN OF E-MATRIX.
                                ;
002552' 016701 000024'  FDC:    MOV     R11,R1          ;R1 = OFFSET OF CURRENT TDCT STATE
002556  032761 010000  001634'  BIT     #EOBIT,TDCT+SEG3(R1)    ;ELSE OVERRIDE BIT ALREADY SET?
002564                          BEQ     1$              ;BRANCH IF YES
002566  005061 001632'          CLR     TDCT+SEG2(R1)   ;CLEAR BASE ADDRESS
002572  012761 040000  001634'  MOV     #ST$INX,TDCT+SEG3(R1)   ;DESIGNATE CURRENT TDCT STATE AS AN
                                                        ;INDEX STATE
002600  016301 000012      1$:     MOV     N.LGT(R3),R1            ;R1 = VMASK LENGTH
002604  042701 100000              BIC     #BITNIB,R1              ;CLEAR NIBBLE INDICATOR FLAG
002610  010300                      MOV     R3,R0                   ;R0->VMASK
002612  012702 006000              MOV     #EMX!FDC!EMXVDC,R2      ;R2 = FLDC BIT
002616  012705 000030'              MOV     #VVEC,R5                ;R5-> V-VECTOR
002622  005065 000012              CLR     N.LGT(R5)               ;CLEAR LENGTH FIELD
002626  062700 000014              ADD     #N.VEC,R0               ;R0->1ST OFFSET IN VMASK
002632  062705 000014              ADD     #N.VEC,R5               ;R5-> 1ST OFFSET IN VVEC
002636  012004              FDC.1:  MOV     (R0)+,R4                ;R4 = E-MATRIX OFFSET
002640  030264 001010'              BIT     R2,EMA(R4)              ;FLDC BIT SET?
002644                          BOFF    1$                      ;BRANCH IF NO
002646  005267 000042'              INC     N.LGT+VVEC              ;INC LENGTH COUNT
002652  010425                      MOV     R4,(R5)+                ;SAVE OFFSET
002654  077110              1$:     SOB     R1,FDC.1                ;LOOP
002656  005767 000042'              TST     N.LGT+VVEC              ;TEST NEW VECTOR'S LENGTH
002662  001455                      BEQ     3$                      ;BRANCH IF ZERO
002664  056767 000016' 000042'      BIS     NIBMSK,N.LGT+VVEC       ;INSERT NIBBLE INDICATOR FLAG
002672  116367 000010  000040'      MOVB    N.POS(R3),N.POS+VVEC    ;LOAD NIBBLE POSITION NO.
002700  105267 000040'              INCB    N.POS+VVEC              ;INCREMENT NIBBLE POS. NO.
                                ;
                                ;   NEW VECTOR GENERATED FOR FLDC COLUMN OF EMA.
                                ;       R3->VMASK.
                                ;       VVEC->NEW VECTOR.
                                ;
                                ;   CHECK VLDC NODE CHAIN FOR MATCH.
                                ;
002704  016702 001576'              MOV     C.VLDC,R2               ;R2->VLDC NODE
002710  001405                      BEQ     4$                      ;BRANCH IF NO SUCH NODE
002712                              CALL    MATNOD                  ;MATCHED VLDC NODE?
002716  103002                      BCC     4$                      ;BRANCH IF NO MATCH
002720  000167  000072              JMP     3$                      ;JUMP IF MATCH
                                ;
                                ;   SET ELSE DEFAULT OVERRIDE BIT IN CURRENT TDCT STATE WORD (INDEX
                                ;   STATE, SEG 3).
                                ;
002724  016704 000024'      4$:     MOV     R11,R4                  ;R4= OFFSET OF CURRENT TDCT WORD
002730  052767 100000  000040'      BIS     #BIT15,N.POS+VVEC       ;MARK VVEC TO REPRESENT ELSE
                                                                    ;DEFAULT OVERRIDE STATE
002736  016764 001610' 001632'      MOV     NXTADD,TDCT+SEG2(R4)    ;ASSIGN BASE ADDRESS
002744  052764 010000  001634'      BIS     #EOBIT,TDCT+SEG3(R4)    ;SET ELSE DEF. OVERRIDE BIT
002752  012701 000040'              MOV     #FLDC,R1                ;R1-> FLDC NODE CHAIN
002756                              CALL    NSCAN                   ;MATCH IN THE FLDC CHAIN?
002762  103411                      BCS     2$                      ;BRANCH IF MATCH
                                ;
                                ;   NO MATCH.  MAKE NEXT AVAILABLE TDCT STATE AN INDEX STATE, MARK IT
                                ;   INCOMPLETE (SET BIT 0 OF SEG3).  INSERT VVEC INTO AN EMPTY NODE,
                                ;   INSERT THIS NEW NODE INTO THE FLDC NODE CHAIN AND PUT ADDRESS OF
```

```
                                    ;   NODE INTO SEG 2 OF ASSIGNED STATE AND SAVE ADDRESS OF NODE IN C.FLDC
                                    ;
                                    ;       R2->NODE THAT SHOULD PRECEED NEW NODE
                                    ;       R1-> NODE CHAIN INDEX
                                    ;
002764  105267  000040'                   INCB    N.POS+VVEC              ;INCREMENT NIBBLE POSITION NUMBER
                                                                            ;TO ALIGN IT WITH FIRST NIBBLE!!!
002770  012701  000040'                   MOV     #FLDC,R1
002774                                    CALL    NEWSTA
003000  010267  001574'                   MOV     R2,C.FLDC               ;SAVE CURRENT FLDC NODE'S ADDRESS
003004  000404                            BR      3$
                                    ;
                                    ;   MATCHED EXISTING NODE.  JUMP TO TDCT STATE REPRESENTED BY THIS
                                    ;   NODE
                                    ;       R2-> MATCHING NODE
                                    ;
003006  010267  001574'             2$:   MOV     R2,C.FLDC               ;SAVE CURRENT FLDC NODE'S ADDRESS
003012                                    CALL    JUMP
003016                              3$:   EXIT    FDC
```

```
                                        .SBTTL  SEARCHABLE CHARACTER COLUMNS
                                ;
                                ;   CHECK SEARCHABLE CHAR. COLUMNS OF EMATRIX
                                ;
                                ;   CHECK WHETHER 1ST OR 2ND NIBBLE
                                ;
                                ;       R3 -> VMASK
                                ;
003020  005067  000040'         SCH:    CLR     N.POS+VVEC              ;CLEAR NIBBLE POSITION NO. FIELD
003024  005767  000016'                 TST     NIBMSK                  ;FIRST OR SECOND NIBBLE?
003030  001005                         BNE     10$                     ;BRANCH IF 2ND NIBBLE
                                ;
                                ;   FIRST NIBBLE.  CHECK COLUMNS N15 THRU N1 OF E-MATRIX
                                ;
003032  012701  000017                 MOV     #15.,R1                 ;R1 = LOOP COUNTER
003036  012702  100000                 MOV     #BIT15,R2               ;R2 = BIT MASK
003042  000404                         BR      11$
                                ;
                                ;   SECOND NIBBLE.  CHECK COLUMNS N14 THRU N0 OF E-MATRIX
                                ;
003044  012701  000017          10$:    MOV     #15.,R1                 ;R1 = LOOP COUNTER
003050  012702  040000                 MOV     #BIT14,R2               ;R2 = BIT MASK
003054  005067  000022'          11$:    CLR     R10                     ;R10 = COLUMN COUNTER
003060                          SCH.5:  SAVE    R1                      ;R1 = LOOP COUNTER
003062  016301  000012                 MOV     N.LGT(R3),R1            ;R1 = VMASK LENGTH
003066  042701  100000                 BIC     #BITNIB,R1
003072  010300                         MOV     R3,R0                   ;R0 -> VMASK
003074  012705  000030'                 MOV     #VVEC,R5                ;R5-> V-VECTOR
003100  005065  000012                 CLR     N.LGT(R5)               ;CLEAR LENGTH FIELD
003104  062700  000014                 ADD     #N.VEC,R0               ;R0->1ST OFFSET IN VMASK
003110  062705  000014                 ADD     #N.VEC,R5               ;R5-> 1ST OFFSET IN VVEC
003114  012004                  SCH.6:  MOV     (R0)+,R4                ;R4 = E-MATRIX OFFSET
003116  005767  000016'                 TST     NIBMSK                  ;FIRST OR SECOND NIBBLE?
003122  001004                         BNE     12$                     ;BRANCH IF SECOND NIBBLE
                                ;
                                ;   FIRST NIBBLE
                                ;
003124  030264  001012'                 BIT     R2,EMA+EMXNB1(R4)       ; BIT SET?
003130                                  BOFF    1$                      ;BRANCH IF NO
003132  000403                         BR      13$
                                ;
                                ;   SECOND NIBBLE
                                ;
003134  030264  001014'          12$:   BIT     R2,EMA+EMXNB2(R4)       ;BIT SET?
003140                                  BOFF    1$                      ;BRANCH IF NO
003142  005267  000042'          13$:    INC     N.LGT+VVEC              ;INC LENGTH COUNT
003146  010425                         MOV     R4,(R5)+                ;SAVE OFFSET
003150  077117                  1$:     SOB     R1,SCH.6                ;LOOP
003152  005767  000042'                 TST     N.LGT+VVEC              ;TEST NEW VECTOR'S LENGTH
003156  001003                         BNE     7$                      ;BRANCH IF NOT ZERO
003160                                  RESTOR  R1
003162  000167  000164                 JMP     SCH.13
003166  056767  000016' 000042' 7$:     BIS     NIBMSK,N.LGT+VVEC       ;INSERT NIBBLE INDICATOR FLAG
003174  116367  000010  000040'         MOVB    N.POS(R3),N.POS+VVEC    ;LOAD NIBBLE POSITION NUMBER
003202  105267  000040'                 INCB    N.POS+VVEC              ;INCREMENT NIBBLE POSITION NUMBER
```

```
                                        ;
                                        ;   NEW VECTOR GENERATED.
                                        ;
                                        ;       R3 -> VMASK.
                                        ;       VVEC CONTAINS NEW VECTOR.
                                        ;
                                        ;   CHECK VLDC NODE CHAIN FOR MATCH.
                                        ;
003206                                          SAVE    R2                      ;R2 = EMATRIX COLUMN BIT MASK.
003210  016702  001576'                         MOV     C.VLDC,R2               ;R2->VLDC NODE.
003214                                          CALL    MATNOD                  ;MATCHED VLDC NODE?.
003220  103002                                  BCC     2$                      ;BRANCH IF NO MATCH.
                                        ;
                                        ;   MATCHED A VLDC VECTOR.  IGNORE CURRENT VVEC (GO GENERATE NEXT ONE)
                                        ;
003222  000167  000120                          JMP     SCH.10
                                        ;
                                        ;   CHECK ELSE DEF. OVERRIDE CHAIN. PROVIDED ELSE DEF. OVERRIDE BIT IS
                                        ;   SET IN CURRENT TDCT STATE WORD.
                                        ;
003226  016702  000024'         2$:             MOV     R11.R2          ;R2 = OFFSET OF CURRENT TDCT STATE
003232  032762  010000  001634'                 BIT     #EOBIT.TDCT+SEG3(R2)    ;TEST ELSE DEF. OVERRIDE BIT
                                                                        ;IN SEG. 3 OF INDEX STATE WORD
003240                                          BOFF    3$                      ;BRANCH IF NOT ON
003242  016702  001574'                         MOV     C.FLDC,R2               ;R2->FLDC NODE.
003246                                          CALL    MATNOD                  ;MATCHED FLDC NODE?.
003252  103002                                  BCC     3$                      ;BRANCH IF NO MATCH.
                                        ;
                                        ;   MATCHED AN FLDC VECTOR.  IGNORE CURRENT VVEC AND GO GENERATE
                                        ;   ANOTHER ONE.
                                        ;
003254  000167  000066                          JMP     SCH.10
```

```
                                        ;
                                        ;   SET TRANSITION SELECTOR BIT IN CURRENT TDCT STATE WORD (INDEX STATE, SEG 1)
                                        ;       R10 = COLUMN COUNTER
                                        ;
003260                          3$:     RESTOR  R2                      ;R2 = EMA COLUMN BIT MASK
003262  016704  000024'                MOV     R11,R4                  ;R4= OFFSET OF CURRENT TDCT STATE WORD
003266  050264  001630'                BIS     R2,TDCT+SEG1(R4)        ;SET TRANSITION SELECT BIT
003272  005764  001632'                TST     TDCT+SEG2(R4)           ;BASE ADD'S ALREADY ASSIGNED?
003276  001003                        BNE     15$                     ;BRANCH IF YES
                                        ;
                                        ;   BASE ADDRESS NOT YET ASSIGNED
                                        ;
003300  016764  001610'  001632'        MOV     NXTADD,TDCT+SEG2(R4)    ;ASSIGN BASE ADD'S
                                        ;
                                        ;   CHECK VECTOR CHAIN OF CURRENT E-MATRIX COLUMN
                                        ;
003306  016701  000022'         15$:    MOV     R10,R1                  ;R1 = COLUMN COUNTER
003312  070127  000006                    MUL     #N.LEN,R1               ;R1 = OFFSET INTO NODE CHAIN INDEX TABLE
003316  066701  000004'                ADD     NCITN,R1                ;R1-> "NEXT" NODE CHAIN INDEX
003322                                  SAVE    R2
003324                                  CALL    NSCAN
003330  103404                        BCS     4$                      ;BRANCH IF  MATCH
                                        ;
                                        ;   NO MATCH;  MAKE NEXT AVAILABLE TDCT STATE WORD AN INDEX
                                        ;   STATE, MARK IT INCOMPLETE (SET BIT 0 OF SEG 3), INSERT VVEC INTO
                                        ;   AN EMPTY NODE, INSERT THIS NEW NODE INTO THE APPROPRIATE NODE CHAIN
                                        ;   AND PUT ADDRESS OF NODE INTO SEG 2 OF ASSIGNED STATE.
                                        ;
                                        ;       R2-> NODE THAT SHOULD PRECEED NEW NODE
                                        ;       R1-> NODE CHAIN INDEX
                                        ;
                                                CALL    NEWSTA
003336  000167  000004                        JMP     SCH.10
                                        ;
                                        ;   MATCHED EXISTING NODE;  JUMP TO TDCT  STATE REPRESENTED BY THIS
                                        ;   NODE
                                        ;       R2->MATCHING NODE
                                        ;
003342                          4$:     CALL    JUMP
                                        ;
                                        ;   PROCESSING OF CURRENT E-MATRIX COLUMN IS COMPLETED;  MASK NEXT
                                        ;   COLUMN OF E-MATRIX TO GENERATE NEXT VVEC.
                                        ;
003346                          SCH.10: RESTOR  R1,R2                   ;R2 = EMATRIX COLUMN BIT MASK
                                                                        ;R1 = LOOP COUNTER
003352  005301                  SCH.13: DEC     R1
003354  001415                          BEQ     SCH.11                  ;BRANCH IF ALL COLUMNS PROCESSED
003356  003005                          BGT     1$
003360  012767  000003  001564'        MOV     #3,PAR1
003366  000167  003346                ERRINT                          ;BUG!!!!!!
003372  005267  000022'          1$:     INC     R10                     ;R10 = COLUMN COUNTER
003376  006202                          ASR     R2                      ;R2 = BIT MASK OF NEXT COLUMN
003400  042702  100000                  BIC     #BIT15,R2
003404  000167  177450                JMP     SCH.5                   ;PROCESS NEXT COLUMN
003410                          SCH.11: EXIT    SCH
```

```
                                        .SBTTL  INTERWORD COLUMN
                                ;
                                ;   NEW VECTOR GENERATED FOR INTERWORD COLUMN OF E-MATRIX.  MAKE NEXT
                                ;   AVAILABLE TDCT STATE A JUMP/SEQUENTIAL STATE AND POINT TO NEXT TDCT
                                ;   STATE WHICH SHOULD THEN BE MADE A MATCH REPORT STATE.
                                ;
003412  016704  000024'         INTERW: MOV     R11,R4          ;R4 = CURRENT STATE'S OFFSET
003416  005764  001632'                 TST     TDCT+SEG2(R4)           ;BASE ADDRESS ASSIGNED ALREADY?
003422  001044                          BNE     3$                      ;BRANCH IF YES
                                ;
                                ;   BASE ADDRESS NOT YET ASSIGNED.  MAKE CURRENT STATE A
                                ;   JUMP/SEQUENTIAL STATE.
                                ;
003424  005064  001630'                 CLR     TDCT+SEG1(R4)           ;SEG 1 = LOOK FOR INTERWORD (0)
003430  016764  001610' 001632'         MOV     NXTADD,TDCT+SEG2(R4)    ;SEG 2 = BASE ADDRESS
003436  012764  100000  001634'         MOV     #ST$JSQ,TDCT+SEG3(R4)   ;SEG 3
003444  052764  014000  001634'         BIS     #JMPBIT!TXTBIT,TDCT+SEG3(R4)
003452                                  GETNXT  R4                      ;R4 = NEXT AVAIL. TDCT OFFSET
003530  000167  000124                 JMP     5$
                                ;
                                ;   MAKE NEXT STATE AN INDEX STATE LOOKING FOR NIBBLE 0
                                ;
003534  052764  000001  001630'  3$:     BIS     #BIT0,TDCT+SEG1(R4)     ;SET TRANSITION SELECT BIT IN
                                                                        ;CURRENT STATE
003542                                  GETNXT  R4                      ;R4 = NEXT AVAILABLE OFFSET IN TDCT
003620  012764  000001  001630'         MOV     #BIT0,TDCT(R4)          ;SEG 1
003626  005724                          TST     (R4)+                   ;R4 = OFFSET OF SEG 2
003630  005267  001610'                 INC     NXTADD                  ;POINT TO NEXT AVAIL. TDCT STATE
003634  016764  001610' 001630'         MOV     NXTADD,TDCT(R4)         ;SEG 2 - NEXT STATE BASE ADDRESS
003642  005724                          TST     (R4)+                   ;R4 = OFFSET OF SEG 3
003644  012764  040000  001630'         MOV     #ST$INX,TDCT(R4)        ;SEG 3
003652  005724                          TST     (R4)+                   ;R4 = OFFSET OF NEXT AVAIL. TDCT STATE
003654  010467  001612'                 MOV     R4,NXTOFF               ;SAVE OFFSET
003660                          5$:     EXIT    INTERW
```

```
                                        .SBTTL  SEQSTA
                                ;
                                ;   SUBROUTINE TO GENERATE SEQUENTIAL STATES.  THIS ROUTINE IS CALLED
                                ;   FROM THE MAIN LOOP (MASKEM) WHEN THE LENGTH OF VMASK IS EQUAL TO ONE.
                                ;   THE ROUTINE FIRST CHECKS THE NEXT "FIRST" NIBBLE IN EMA.  IF NIBBLE IS
                                ;       A)  TERMINATING INTERWORD, CALLS INTERW AND RETURNS WITH CC-CLEAR
                                ;       B)  TRAILING VLDC, RETURNS WITH CC-SET
                                ;   IF TERM CONTAINS FLDC, IT RETURNS WITH CC-SET
                                ;   OTHERWISE, THE CURRENT TDCT STATE IS MADE AN INDEX STATE LOOKING FOR
                                ;   THE SPECIFIC NIBBLE, POINTING TO THE NEXT AVAILABLE TDCT STATE.
                                ;   THE REMAINING CHARACTERS OF THE TERM ARE THEN INSERTED IN SEQUENTIAL STATES
                                ;   UNTIL A TERMINATING INTERWORD OR A TRAILING VLDC ARE ENCOUNTERED, WHEN
                                ;   THIS HAPPENS, CONTROL IS TRANSFERRED TO A) OR B), RESPECTIVELY.
                                ;
                                ;   ON ENTRY,
                                ;       R3->VMASK
                                ;       R11 = CURRENT TDCT STATE'S OFFSET
                                ;
003662                          SEQSTA: SAVE    R1,R2,R4,R5
003672  016302  000014                  MOV     N.VEC(R3),R2            ;R2 = OFFSET INTO EMA
003676  032763  100000  000012          BIT     #BITNIB,N.LGT(R3)       ;SECOND NIBBLE?
003704                                  BON     7$                      ;BRANCH IF YES
003706  000167  001012                  JMP     10$                     ;BYPASS IF 1ST NIBBLE
003712  032762  004000  001016' 7$:     BIT     #EMXVDC,EMA+6(R2)       ;NEXT CHAR A VLDC??
003720                                  BOFF    12$                     ;BRANCH IF NO
003722  000167  000776                  JMP     10$
003726  032762  000001  001020' 12$:    BIT     #BIT0,EMA+6+EMXNB1(R2)  ;NEXT CHAR, TERMINATING INTERWORD?
003734                                  BOFF    11$                     ;BRANCH IF NO
003736  000167  000762                  JMP     10$                     ;BRANCH IF YES
                                ;
                                ;   LOOK FOR FLDC IN TERM
                                ;
003742  062702  000006          11$:    ADD     #6,R2                   ;R2 = EMA OFFSET OF NEXT CHAR.
003746  032762  002000  001010' 15$:    BIT     #EMXFDC,EMA(R2)         ;FLDC?
003754                                  BOFF    13$                     ;BRANCH IF NO
003756  000167  000742                  JMP     10$
003762  032762  004000  001010' 13$:    BIT     #EMXVDC,EMA(R2)         ;***VLDC?
003770                                  BOFF    14$                     ;***BRANCH IF NO
003772  000167  000726                  JMP     10$
003776                          14$:
003776  032762  000001  001020'          BIT     #BIT0,EMA+6+EMXNB1(R2)  ;NEXT CHAR, TERMINATING INTERWORD?
004004                                  BON     1$                      ;BRANCH IF YES
004006  062702  000006                  ADD     #6,R2                   ;R2 = EMA OFFSET OF NEXT CHARACTER
004012  000755                          BR      15$
                                ;
                                ;   MAKE CURRENT STATE AN INDEX STATE
                                ;
004014  016302  000014          1$:     MOV     N.VEC(R3),R2            ;R2 = EMA OFFSET OF CURRENT NIBBLE(2ND)
004020  016701  000024'                  MOV     R11,R1                  ;R1 = TDCT OFFSET OF CURRENT STATE
004024  016167  001630' 000706          MOV     TDCT(R1),SEQTMP         ;SAVE CONTENTS OF STATE
004032  016167  001632' 000702          MOV     TDCT+SEG2(R1),SEQTMP+2
004040  016167  001634' 000676          MOV     TDCT+SEG3(R1),SEQTMP+4
004046  032763  100000  000012          BIT     #BITNIB,N.LGT(R3)       ;SECOND NIBBLE?
004054                                  BON     2$                      ;BRANCH IF YES
004056  016261  001012' 001630'          MOV     EMA+EMXNB1(R2),TDCT(R1) ;SEG 1
004064  000403                          BR      3$
004066  016261  001014' 001630' 2$:     MOV     EMA+EMXNB2(R2),TDCT(R1) ;SEG 1
```

```
 988 004074  005721                    3$:     TST     (R1)+
 989 004076  016761  001610' 001630'            MOV     NXTADD,TDCT(R1)         ;SEG 2
 990 004104  005721                            TST     (R1)+
 991 004106  012761  040000  001630'            MOV     #ST$INX,TDCT(R1)        ;SEG 3
 992                                    ;
 993                                    ;   BUILD SEQUENTIAL STATE ENTRIES
 994                                    ;
 995                                    ;       R2 = EMA OFFSET
 996                                    ;       R3-> VMASK
 997                                    ;
 998 004114  062702  000006                    ADD     #6,R2                   ;R2 = EMA OFFSET OF NEXT CHAR
 999 004120                                    GETNXT  R4                      ;R4= NEXT FREE TDCT STATE
1000 004176  012764  150000  001634'  4$:     MOV     #ST$SEQ!TXTBIT,TDCT+SEG3(R4)    ;LOAD STATE TYPE IN SEG 3
1001 004204  052764  004000  001634'            BIS     #S3BIT,TDCT+SEG3(R4)    ;END AFTER 3 BYTES
1002                                    ;
1003                                    ;   BYTE 1
1004                                    ;
1005 004212  116264  001010' 001630'            MOVB    EMA(R2),TDCT(R4)        ;BYTE 1
1006 004220  062702  000006                    ADD     #6,R2                   ;R2 = OFFSET OF NEXT CHAR, IN EMA
1007 004224  032762  004000  001010'            BIT     #EMXVDC,EMA(R2)         ;VLDC?
1008 004232                                    BOFF    30$                     ;BRANCH IF NO
1009 004234  052764  001000  001634'            BIS     #S1BIT,TDCT+SEG3(R4)    ;END AFTER 1 BYTE
1010 004242  062704  000006                    ADD     #6,R4                   ;R4 = OFFSET OF NEXT FREE TDCT STATE
1011 004246  010467  001612'                    MOV     R4,NXTOFF               ;SAVE IT
1012 004252  005267  001610'                    INC     NXTADD                  ;NEXT AVAIL, TDCT ADDRESS
1013 004256  000517                            BR      27$
1014 004260  032762  000001  001012' 30$:     BIT     #BIT0,EMA+EMXNB1(R2)    ;INTERWORD?
1015 004266                                    BOFF    31$                     ;BRANCH IF NO
1016 004270  052764  001000  001634'            BIS     #S1BIT,TDCT+SEG3(R4)    ;END AFTER 1 BYTE
1017 004276  062704  000006                    ADD     #6,R4                   ;R4 = OFFSET OF NEXT FREE TDCT STATE
1018 004302  010467  001612'                    MOV     R4,NXTOFF               ;SAVE IT
1019 004306  005267  001610'                    INC     NXTADD                  ;NEXT AVAIL TDCT ADDRESS
1020 004312  000555                            BR      20$
1021                                    ;
1022                                    ;   BYTE 2
1023                                    ;
1024 004314  116264  001010' 001631' 31$:     MOVB    EMA(R2),TDCT+1(R4)      ;BYTE 2
1025 004322  062702  000006                    ADD     #6,R2                   ;R2 = OFFSET OF NEXT CHAR, IN EMA
1026 004326  032762  004000  001010'            BIT     #EMXVDC,EMA(R2)         ;VLDC?
1027 004334                                    BOFF    32$                     ;BRANCH IF NO
1028 004336  052764  002000  001634'            BIS     #S2BIT,TDCT+SEG3(R4)    ;END AFTER 2 BYTES
1029 004344  062704  000006                    ADD     #6,R4                   ;R4 = OFFSET OF NEXT FREE TDCT STATE
1030 004350  010467  001612'                    MOV     R4,NXTOFF               ;SAVE IT
1031 004354  005267  001610'                    INC     NXTADD                  ;NEXT AVAIL, TDCT ADDRESS
1032 004360  000456                            BR      27$
1033 004362  032762  000001  001012' 32$:     BIT     #BIT0,EMA+EMXNB1(R2)    ;INTERWORD?
1034 004370                                    BOFF    33$                     ;BRANCH IF NO
1035 004372  052764  002000  001634'            BIS     #S2BIT,TDCT+SEG3(R4)    ;END AFTER 2 BYTES
1036 004400  062704  000006                    ADD     #6,R4                   ;R4 = OFFSET OF NEXT FREE TDCT STATE
1037 004404  010467  001612'                    MOV     R4,NXTOFF               ;SAVE IT
1038 004410  005267  001610'                    INC     NXTADD                  ;NEXT AVAIL TDCT ADDRESS
1039 004414  000514                            BR      20$
1040                                    ;
1041                                    ;   BYTE 3
1042                                    ;
1043 004416  116264  001010' 001632' 33$:     MOVB    EMA(R2),TDCT+SEG2(R4)   ;BYTE 3
1044 004424  062702  000006                    ADD     #6,R2                   ;R2 = OFFSET OF NEXT CHAR, IN EMA
```

```
004430  032762  004000  001010'         BIT     #EMXVDC,EMA(R2)         ;VLDC?
004436                                  BOFF    34$                     ;BRANCH IF NO
004440  062704  000006                  ADD     #6,R4                   ;R4 = OFFSET OF NEXT FREE TDCT STATE
004444  010467  001612'                 MOV     R4,NXTOFF               ;SAVE IT
004450  005267  001610'                 INC     NXTADD                  ;NEXT AVAIL. TDCT ADDRESS
004454  000420                          BR      27$
004456  032762  000001  001012' 34$:    BIT     #BIT0,EMA+EMXNB1(R2)    ;INTERWORD?
004464                                  BOFF    35$                     ;BRANCH IF NO
004466  062704  000006                  ADD     #6,R4                   ;R4 = OFFSET OF NEXT FREE TDCT STATE
004472  010467  001612'                 MOV     R4,NXTOFF               ;SAVE IT
004476  005267  001610'                 INC     NXTADD                  ;NEXT AVAIL TDCT ADDRESS
004502  000461                          BR      20$
004504  062704  000006          35$:    ADD     #6,R4                   ;R4 = TDCT OFFSET OF NEXT STATE
004510  005267  001610'                 INC     NXTADD
004514  000630                          BR      4$                      ;LOOP
                                ;
                                ;       TRAILING VLDC
                                ;
004516                          27$:    GETNXT  R4                      ;R4 = NEXT FREE TDCT OFFSET
004574  010467  000024'                 MOV     R4,R11
004600  016764  000134  001630'         MOV     SEQTMP,TDCT(R4)         ;SEG 1
004606  005724                          TST     (R4)+
004610  016764  000126  001630'         MOV     SEQTMP+2,TDCT(R4)       ;SEG 2
004616  005724                          TST     (R4)+
004620  016764  000120  001630'         MOV     SEQTMP+4,TDCT(R4)       ;SEG 3
004626  005724                          TST     (R4)+
004630  010467  001612'                 MOV     R4,NXTOFF
004634  005267  001610'                 INC     NXTADD
004640  010263  000014                  MOV     R2,N.VEC(R3)            ;LOAD VLDC'S EMA OFFSET INTO VMASK
004644  000427                          BR      10$
                                ;
                                ;       TERMINATING INTERWORD
                                ;
004646  010467  000024'         20$:    MOV     R4,R11                  ;R11 = CURRENT TDCT STATE'S OFFSET
004652  062704  000006                  ADD     #6,R4                   ;R4 = OFFSET OF NEXT TDCT STATE
004656  010467  001612'                 MOV     R4,NXTOFF
004662  005267  001610'                 INC     NXTADD
004666  016704  000024'         21$:    MOV     R11,R4
004672  005064  001632'                 CLR     TDCT+SEG2(R4)           ;INDICATE THAT TDCT ADD'S IS UNASSIGNED
004676  012767  000001  000042'         MOV     #1,VVEC+N.LGT           ;BUILD VVEC
004704  010267  000044'                 MOV     R2,VVEC+N.VEC
004710                                  CALL    INTERW                  ;PROCESS INTERWORD
004714                                  CALL    FLUMD                   ;PROCESS FLU MODIFIERS
004720  000241                          CLC
004722  000401                          BR      26$
                                ;
                                ;       RETURN
                                ;
004724  000261                  10$:    SEC
004726                          26$:    RESTOR  R1,R2,R4,R5
004736                                  EXIT    SEQSTA
                                ;
                                ;
                                ;
004740                          SEQTMP: .BLKW   3
```

```
                                        .SBTTL  SUBROUTINE "JUMP"
                                ;
                                ;   MATCHED EXISTING NODE.  COMPARE ELSE DEFAULT STATE ADDRESS OF MATCHED
                                ;   NODE WITH ELSE DEFAULT STATE ADDRESS OF VMASK.  IF THEY ARE THE
                                ;   SAME, MAKE NEXT AVAILABLE TDCT STATE A JUMP STATE,
                                ;   AND POINT TO TDCT ADDRESS OF MATCHED NODE FROM THIS STATE.
                                ;   IF THEY ARE DIFFERENT, USE "CHANGE ELSE DEFAULT" STATE TO JUMP BACK.
                                ;
                                ;       R2->MATCHING NODE
                                ;       R3->VMASK
                                ;
1112 004746  016704  000024'    JUMP:   MOV     R11,R4          ;R4=TDCT OFFSET OF CURRENT STATE
1113 004752  005764  001632'            TST     TDCT+SEG2(R4)           ;BASE ADDRESS ALREADY ASSIGNED?
1114 004756  001003                     BNE     1$                      ;BRANCH IF YES
                                ;
                                ;   BASE ADDRESS NOT YET ASSIGNED
                                ;
1118 004760  016764  001610' 001632'            MOV     NXTADD,TDCT+SEG2(R4)    ;ASSIGN BASE ADDRESS
                                ;
                                ;   COMPARE ELSE DEF. ADDRESSES
                                ;
1122 004766                     1$:     GETNXT  R4              ;R4=NEXT AVAILABLE OFFSET IN TDCT
1123 005044  026362  000006  000006             CMP     N.ELSE(R3),N.ELSE(R2)   ;COMPARE ELSE DEF. ADD'S
1124 005052  001417                             BEQ     2$                      ;BRANCH IF EQUAL
                                ;
                                ;   ELSE DEF. STATE ADDRESSES ARE NOT EQUAL.  USE "CHANGE ELSE DEF. "
                                ;   STATE TO JUMP BACK.
                                ;
1129 005054  016264  000006  001630'            MOV     N.ELSE(R2),TDCT(R4)     ;SEG 1 = NEW ELSE DEF. ADDRESS
1130 005062  005724                             TST     (R4)+                   ;R4 = OFFSET OF SEG 2
1131 005064  016264  000004  001630'            MOV     N.FSA(R2),TDCT(R4)      ;SEG 2 = NEXT STATE ADDRESS
1132 005072  005724                             TST     (R4)+                   ;R4 = OFFSET OF SEG 3
1133 005074  012764  120000  001630'            MOV     #ST$CNG,TDCT(R4)        ;SEG 3 = CHANGE ELSE DEF. STATE
1134 005102  052764  010000  001630'            BIS     #ELSBIT,TDCT(R4)
1135 005110  000412                             BR      3$
                                ;
                                ;   ELSE DEF. STATE ADDRESSES ARE  EQUAL.  USE "JUMP" STATE TO JUMP BACK.
                                ;
1139 005112  005064  001630'    2$:     CLR     TDCT(R4)                ;SEG 1
1140 005116  005724                     TST     (R4)+                   ;R4 = OFFSET OF SEG 2
1141 005120  016264  000004  001630'            MOV     N.FSA(R2),TDCT(R4)      ;SEG 2 - NEXT STATE ADDRESS
1142 005126  005724                             TST     (R4)+                   ;R4 = OFFSET OF SEG 3
1143 005130  012764  100000  001630'            MOV     #ST$JSQ,TDCT(R4)        ;SEG 3 - JUMP STATE
1144 005136  005724                     3$:     TST     (R4)+                   ;R4 = OFFSET OF NEXT AVAILABLE TDCT STARTATE
1145 005140  010467  001612'                    MOV     R4,NXTOFF               ;SAVE OFFSET
1146 005144  005267  001610'                    INC     NXTADD                  ;INC. NEXT AVAIL. TDCT ADDRESS
1147 005150                                     EXIT    JUMP
```

```
                                        .SBTTL  NEWSTA
                                ;
                                ;   A NEW FSA STATE HAS BEEN IDENTIFIED WHEN A UNIQUE VVEC WAS GENERATED.
                                ;   THE PURPOSE OF THIS SUBROUTINE IS TO MAKE THE NEXT AVAILABLE TDCT
                                ;   STATE WORD AN INDEX STATE, MARK IT INCOMPLETE (SET BIT0 OF SEG 3),
                                ;   INSERT VVEC INTO AN EMPTY NODE, INSERT THIS NEW NODE INTO THE
                                ;   APPROPRIATE NODE CHAIN (IN "NEXT" POOL) AND PUT ADDRESS OF NODE INTO SEG2
                                ;   OF ASSIGNED STATE.
                                ;
                                ;   ON ENTRY,
                                ;       R2-> NODE THAT SHOULD PRECEED NEW NODE
                                ;       R1->NODE CHAIN INDEX
                                ;
                                ;   ON RETURN,
                                ;       R1 = TDCT OFFSET OF NEW STATE
                                ;       R2-> NEW NODE
                                ;
005152                          NEWSTA: SAVE    R4
005154                                  GETNXT  R4                      ;R4=NEXT AVAILABLE TDCT OFFSET
005232  026767  001612' 000024'         CMP     NXTOFF,R11              ;ARE CURRENT AND NEXT FREE STATES THE SAME?
005240  001402                          BEQ     1$                      ;BRANCH IF YES
005242  005064  001630'                 CLR     TDCT(R4)                ;SEG 1
005246  005724                  1$:     TST     (R4)+                   ;R4 = OFFSET OF SEG 2
005250  005064  001630'                 CLR     TDCT(R4)                ;SEG 2
005254  005724                          TST     (R4)+                   ;R4 = OFFSET OF SEG 3
005256  012764  040000  001630'         MOV     #ST$INX,TDCT(R4)        ;SEG 3
005264  052764  000001  001630'         BIS     #BIT0,TDCT(R4)          ;MARK TDCT STATE AS INCOMPLETE
005272  005724                          TST     (R4)+                   ;R4 = OFFSET OF NEXT AVAIL, TDCT STATE
005274  016767  001610' 000034'         MOV     NXTADD,N.FSA+VVEC       ;SAVE TDCT STATE ADDRESS IN VVEC
005302  005267  001610'                 INC     NXTADD                  ;INC, NEXT AVAILABLE TDCT STATE ADDRESS
005306  016767  000006' 000036'         MOV     N.ELSE+VMASK,N.ELSE+VVEC        ;LOAD ELSE DEF, STATE ADDS
                                ;
                                ;       R2->NODE THAT SHOULD PRECEED NEW NODE
                                ;       R1->NODE CHAIN INDEX
                                ;
005314  020127  000040'                 CMP     R1,#FLDC                ;VLDC OR FLDC?
005320  101011                          BHI     3$                      ;BRANCH IF NO
005322                                  CALL    MERGE                   ;INSERT VVEC INTO NODE CHAIN
005326  103017                          BCC     2$
005330  010200                          MOV     R2,R0                   ;R0 = ERROR CODE
005332                                  RESTOR  R2
005334                                  CALL    NODERR
005340  000167  001414                  JMP     EXIT
005344                          3$:     CALL    MRG                     ;MERGE INTO "NEXT" NODE POOL
005350  103006                          BCC     2$
005352  010200                          MOV     R2,R0                   ;R0 = ERROR CODE
005354                                  RESTOR  R2
005356                                  CALL    NODERR
005362  000167  001372                  JMP     EXIT
                                ;
                                ;       R4 = OFFSET OF STATE IMMEDIATELY FOLLOWING TDCT STATE OF INTEREST
                                ;       R2-> NEW NODE
                                ;
005366  016701  001612'         2$:     MOV     NXTOFF,R1               ;R1=OFFSET OF NEW TDCT STATE
005372  010467  001612'                 MOV     R4,NXTOFF               ;R4=OFFSET OF NEXT FREE TDCT STATE
005376  010261  001632'                 MOV     R2,TDCT+SEG2(R1)        ;LOAD NODE ADDRESS INTO SEG2
005402                                  RESTOR  R4
```

```
1206 005404                             EXIT    NEWSTA
```

```
                                        .SBTTL  SWAP NODE POOLS
                                ;
                                ;   SUBROUTINE TO SWAP NODE POOLS WHEN TRANSLATOR IS DONE WITH CURRENT
                                ;   NIBBLE POSITION NUMBER AND WANTS TO START ON NEW POSITION NUMBER.
                                ;   THE NODE POOL WHICH WAS USED TO STORE VECTORS OF "NEXT" STATES
                                ;   BECOMES THE "CURRENT" STATE NODE POOL.  THE OTHER NODE POOL IS
                                ;   REINITIALIZED AND BECOMES THE NODE POOL FOR THE NEW SET OF "NEXT"
                                ;   STATE NODES.
                                ;
                                ;   ON ENTRY, LOCATION BNPN CONTAINS POINTER TO BEGINNING OF NODE POOL
                                ;   THAT WAS USED FOR "NEXT" STATE NODES.
                                ;
005406                          SWAPNP: SAVE    R0,R1
005412  016746  000014'                 MOV     BNPC,-(SP)              ;SAVE BEGINNING OF CURRENT NODE POOL
005416  016746  000016'                 MOV     ENPC,-(SP)              ;SAVE END OF      "       "     "
005422  016746  000020'                 MOV     NCITC,-(SP)             ;SAVE NODE CHAIN INDEX TABLE ADD'S
                                ;
                                ;   MAKE WHAT USED TO BE THE "NEXT" POOL THE "CURRENT" POOL
                                ;
005426  012700  000000'                 MOV     #BNPN,R0                ;R0->"NEXT" POOL
005432  012701  000014'                 MOV     #BNPC,R1                ;R1->"CURRENT" POOL
005436  012021                         MOV     (R0)+,(R1)+             ;START OF NODE POOL
005440  012021                         MOV     (R0)+,(R1)+             ;END OF    "    "
005442  012021                         MOV     (R0)+,(R1)+             ;NODE CHAIN INDEX TABLE ADD'S
005444  012021                         MOV     (R0)+,(R1)+             ;NO. OF NODES IN USE
005446  012021                         MOV     (R0)+,(R1)+             ;POINTER TO 1ST NODE
005450  012021                         MOV     (R0)+,(R1)+             ;POINTER TO NEXT AVAIL. ADD'S IN FREEQ
                                ;
                                ;   SET UP NEW "NEXT" POOL
                                ;
005452  012700  000000'                 MOV     #BNPN,R0                ;R0->"NEXT" NODE POOL HEADER
005456  012660  000004                 MOV     (SP)+,4(R0)             ;NODE CHAIN INDEX TABLE ADD'S
005462  012660  000002                 MOV     (SP)+,2(R0)             ;END OF NODE POOL
005466  011610                         MOV     (SP),(R0)               ;START OF NODE POOL
005470  062700  000006                 ADD     #6,R0                   ;R0->NO. OF NODES IN USE
005474  005020                         CLR     (R0)+
005476  011620                         MOV     (SP),(R0)+              ;POINTER TO 1ST NODE
005500  012610                         MOV     (SP)+,(R0)              ;POINTER TO NEXT AVAIL. ADD'S
                                ;
                                ;   REINITIALIZE NODE CHAIN INDEX OF NEW "NEXT" POOL
                                ;
005502  016700  000004'                 MOV     NCITN,R0                ;R0->NODE CHAIN INDEX TABLE
005506  162700  000002                 SUB     #2,R0                   ;R0->NODE COUNTER OF 1ST CHAIN
005512  012701  000020                 MOV     #16.,R1                 ;R1 = LOOP COUNTER
005516  005020                 1$:     CLR     (R0)+
005520  010010                         MOV     R0,(R0)
005522  010060  000002                 MOV     R0,2(R0)
005526  062700  000004                 ADD     #4,R0
005532  077107                         SOB     R1,1$
                                ;
                                ;   RETURN
                                ;
005534                                  RESTOR  R0,R1
005540                                  EXIT    SWAPNP
                                ;
000000                                  .PSECT  NODES
                                ;   THE LOGICAL MAPPING OF THE "CURRENT" AND "NEXT" NODE POOLS
```

```
                                ;    INTO THE TWO PHYSICAL NODE POOLS, POOL 1 AND POOL 2, IS DONE
                                ;    THROUGH THE POOL HEADER TABLES DEFINED BELOW.
                                ;
                                ;    REASSIGNMENT OF PHYSICAL TO LOGICAL POOLS IS DONE BY THE
                                ;    SWAPNP ROUTINE.
                                ;
                                ;    NODE HEADER TABLE FOR "NEXT" NODE POOL
                                ;        (INITIALLY = POOL 2)
                                ;
000000  030366'  BNPN:   .WORD   NP2             ;BEGINNING OF NODE POOL
000002  040366'  ENPN:   .WORD   NP2E            ;END OF NODE POOL
000004  030230'  NCITN:  .WORD   NCIT2           ;POINTER TO NODE CHAIN INDEX TABLE
000006  000000           .WORD   0               ;NO. OF NODES IN USE
000010  030366'  FRNPN:  .WORD   NP2             ;POINTER TO 1ST NODE
000012  030366'          .WORD   NP2             ;POINTER TO NEXT AVAIL, ADDS IN FREE POOL
                                ;
                                ;    NODE HEADER TABLE FOR "CURRENT" NODE POOL
                                ;
000014  020226'  BNPC:   .WORD   NP1             ;BEGINNING OF NODE POOL
000016  030226'  ENPC:   .WORD   NP1E            ;END OF NODE POOL
000020  020070'  NCITC:  .WORD   NCIT1           ;POINTER TO NODE CHAIN INDEX TABLE
000022  000000           .WORD   0               ;NO. OF NODES IN USE
000024  020226'  FRNPC:  .WORD   NP1             ;POINTER TO 1ST NODE
000026  020226'          .WORD   NP1             ;POINTER TO NEXT AVAIL, ADDS IN FREE POOL
```

```
                                        .SBTTL  SCAN NODE CHAIN
005542                                  .PSECT  CODE
                                ;
                                ;   SUBROUTINE TO SCAN INDICATED NODE CHAIN FOR A VECTOR THAT MATCHES
                                ;   VECTOR IN VVEC BUFFER.
                                ;
                                ;   ON ENTRY,
                                ;       R1->NODE CHAIN TO BE SCANNED
                                ;
                                ;   ON RETURN,
                                ;       R1 IS UNCHANGED
                                ;       CC IS SET IF MATCH WAS FOUND
                                ;       R2->MATCHING NODE
                                ;
                                ;       CC IS CLEARED IF NO MATCH
                                ;       R2->NODE AFTER WHICH NEW NODE CONTAINING VECTOR SHOULD BE INSERTED
                                ;
005542                          NSCAN:  SAVE    R0,R3,R4,R5
005552  005761  177776                  TST     N.NODE(R1)              ;ANY NODES IN CHAIN?
005556  001003                          BNE     1$                      ;BRANCH IF YES
                                ;
                                ;   CHAIN IS EMPTY.  NEW NODE SHOULD BE INSERTED AT HEAD OF CHAIN
                                ;
005560  010102                          MOV     R1,R2                   ;R2->NODE CHAIN INDEX
005562  000167  000120                  JMP     10$
                                ;
                                ;
005566  011102                  1$:     MOV     (R1),R2                 ;R2->1ST NODE
005570  012703  000030'                  MOV     #VVEC,R3                ;R3->NEW VECTOR
005574  016300  000012          7$:     MOV     N.LGT(R3),R0            ;R0 = VVEC LENGTH
005600  042700  100000                  BIC     #BITNIB,R0              ;CLEAR NIBBLE INDICATOR FLAG
005604  016205  000012                  MOV     N.LGT(R2),R5            ;NODE'S VECTOR LENGTH
005610  042705  100000                  BIC     #BITNIB,R5
005614  020005                          CMP     R0,R5                   ;COMPARE VECTOR LENGTHS
005616  003024                          BGT     3$                      ;BRANCH IF NEW VECTOR'S LENGTH IS GREATEER
005620  001404                          BEQ     2$                      ;BRANCH IF SAME LENGTH
                                ;
                                ;   LENGTH OF NEW VECTOR IS LESS.  THEREFORE NO POSSIBLE MATCH.  NEW NODE
                                ;   SHOULD BE INSERTED BEFORE CURRENT NODE.
                                ;
005622  016202  000002          4$:     MOV     N.BACK(R2),R2           ;R2->PREVIOUS NODE
005626  000167  000054                  JMP     10$
                                ;
                                ;   LENGTHS MATCHED.  COMPARE ENTRIES OF VECTORS.
                                ;       R2-> CURRENT NODE
                                ;       R3-> NEW VECTOR
                                ;
005632  010204                  2$:     MOV     R2,R4                   ;R4-> CURRENT NODE
005634  062704  000014                  ADD     #N.VEC,R4               ;R4->CURRENT VECTOR
005640  010305                          MOV     R3,R5                   ;R5->NEW VECTOR
005642  062705  000014                  ADD     #N.VEC,R5               ;R5->NEW VECTOR'S ENTRIES
005646  022524                  5$:     CMP     (R5)+,(R4)+             ;COMPARE ENTRIES
005650  003007                          BGT     3$                      ;BRANCH IF NEW VECTOR'S ENTRY IS GREATER
005652  001402                          BEQ     6$                      ;BRANCH IF ENTRIES MATCH
                                ;
                                ;   NEW VECTOR'S ENTRY IS SMALLER.  NO POSSIBLE MATCH.
                                ;
```

```
 1347 005654  000167  177742                    JMP     4$
 1348                                 ;
 1349                                 ;   CONTINUE
 1350                                 ;
 1351 005660  077006                  6$:     SOB     R0,5$                   ;LOOK AT NEXT ENTRIES
 1352                                 ;
 1353                                 ;   VECTORS MATCHED !!!
 1354                                 ;       R2->MATCHED NODE
 1355                                 ;
 1356 005662  000261                          SEC
 1357 005664  000167  000020                  JMP     11$
 1358                                 ;
 1359                                 ;   LOOK AT NEXT NODE
 1360                                 ;
 1361 005670  026201  000000          3$:     CMP     N.FWD(R2),R1            ;CURRENT NODE LAST IN CHAIN?
 1362 005674  001404                          BEQ     10$                     ;BRANCH IF YES
 1363 005676  016202  000000                  MOV     N.FWD(R2),R2            ;R2->NEXT NODE
 1364 005702  000167  177666                  JMP     7$
 1365                                 ;
 1366                                 ;   NO MATCH,  NEW NODE TO BE LAST IN CHAIN.
 1367                                 ;       R2->LAST NODE
 1368                                 ;
 1369 005706  000241                  10$:    CLC
 1370 005710                          11$:    RESTOR  R0,R3,R4,R5
 1371 005720                                  EXIT    NSCAN
```

```
 1373                                                   .SBTTL  MATCH NODE
 1374                                           ;
 1375                                           ;   SUBROUTINE TO COMPARE A GIVEN V-VECTOR WITH THE V-VECTOR OF
 1376                                           ;   INDICATED NODE.
 1377                                           ;   ON ENTRY,
 1378                                           ;       R2-> NODE
 1379                                           ;   ON RETURN,
 1380                                           ;       MATCH:          CC = 1          R2->MATCHING NODE
 1381                                           ;       NO MATCH:       CC = 0
 1382                                           ;
 1383 005722                                   MATNOD: SAVE    R0,R3,R4,R5
 1384 005732  005702                                   TST     R2                      ;IS THERE A NODE?
 1385 005734  001426                                   BEQ     2$                      ;BRANCH IF NO
 1386 005736  012703  000030'                          MOV     #VVEC,R3                ;R3->VVEC
 1387 005742  016300  000012                           MOV     N.LGT(R3),R0            ;R0 = VVEC'S VECTOR LENGTH
 1388 005746  042700  100000                           BIC     #BITNIB,R0
 1389 005752  016204  000012                           MOV     N.LGT(R2),R4            ;R4 = NODE'S VECTOR LENGTH
 1390 005756  042704  100000                           BIC     #BITNIB,R4
 1391 005762  020004                                   CMP     R0,R4                   ;COMPARE VECTOR LENGTHS
 1392 005764  001012                                   BNE     2$                      ;BRANCH IF LENGTHS ARE NOT EQUAL
 1393 005766  010204                                   MOV     R2,R4                   ;R4->NODE
 1394 005770  062704  000014                           ADD     #N.VEC,R4               ;R4-> VECTOR IN NODE
 1395 005774  062703  000014                           ADD     #N.VEC,R3               ;R3-> NEW VECTOR
 1396 006000  022324                           1$:     CMP     (R3)+,(R4)+             ;COMPARE ENTRIES
 1397 006002  001003                                   BNE     2$                      ;BRANCH IF NOT EQUAL
 1398 006004  077003                                   SOB     R0,1$
 1399 006006  000261                                   SEC
 1400 006010  000401                                   BR      3$
 1401 006012  000241                           2$:     CLC
 1402 006014                                   3$:     RESTOR  R0,R3,R4,R5
 1403 006024                                           EXIT    MATNOD
```

```
1405                                            .SBTTL  MERGE VARIABLE LENGTH NODES
1406                                    ;
1407                                    ;
1408                                    ;   ON ENTRY,
1409                                    ;       R1-> NODE CHAIN INDEX
1410                                    ;       R2-> NODE THAT SHOULD PRECEED NEW NODE
1411                                    ;       VVEC -> SOURCE NODE
1412                                    ;
1413                                    ;   ON RETURN,
1414                                    ;       CC = 0          R2->NEW NODE
1415                                    ;       CC = 1          R2 = ERROR CODE         ER114 = NO FREE NODES
1416                                    ;
1417 006026                            MRG:    SAVE    R0,R1
1418 006032  016701  000012'                   MOV     FRNPN+2,R1              ;R1-> NEXT AVAIL, ENTRY IN FREEQ
1419 006036  166701  000000'                   SUB     BNPN,R1                 ;R1 = LENGTH OF NODE POOL IN USE
1420 006042  022701  010000                    CMP     #NP1MSZ,R1                      ;END OF NODE POOL REGION?
1421 006046  003004                            BGT     1$                      ;BRANCH IF NO
1422 006050  012700  177616                    MOV     #ER114,R0               ;NO FREE NODES
1423 006054  000261                            SEC
1424 006056  000442                            BR      10$
1425 006060  016701  000012'           1$:     MOV     FRNPN+2,R1              ;R1->NEXT AVAIL, ENTRY IN FREEQ
1426 006064                            2$:     SAVE    R0,R1,R2                ;R1-> NEW NODE
1427 006072  012700  000030'                   MOV     #VVEC,R0                ;R0 = SOURCE ADDS
1428 006076  016702  000042'                   MOV     N.LGT+VVEC,R2           ;R2 = VECTOR LENGTH (WORDS)
1429 006102  042702  100000                    BIC     #BITNIB,R2              ;CLEAR NIBBLE INDICATOR FLAG
1430 006106  006302                            ASL     R2                      ;R2 = VECTOR LENGTH (BYTES)
1431 006110  062702  000014                    ADD     #N.HEAD,R2              ;R2 = NO. OF BYTES TO BE MOVED
1432 006114                                    CALL    MOVDT                   ;COPY VVEC INTO FREE NODE
1433 006120  010167  000012'                   MOV     R1,FRNPN+2              ;SAVE NEXT AVAIL, ENTRY IN FREEQ
1434 006124  005267  000006'                   INC     FRNPN-2                 ;INCREMENT NODE COUNT
1435 006130                                    RESTOR  R0,R1,R2                ;R1->NEW NODE
1436 006136  010100                            MOV     R1,R0                   ;R0-> NEW NODE
1437 006140                                    RESTOR  R1                      ;R1->NODE CHAIN INDEX
1438                                                                           ;R2->NODE AFTER WHICH INSERT SHOULD BE MADE
1439 006142                                    CALL    ADDNOD                  ;ADD NODE TO NODE CHAIN
1440 006146  103003                            BCC     4$
1441 006150  010002                            MOV     R0,R2                   ;R2 = ERROR CODE
1442 006152  000261                            SEC
1443 006154  000405                            BR      11$
1444 006156  010002                    4$:     MOV     R0,R2                   ;R2->NEW NODE
1445 006160  000241                            CLC
1446 006162  000402                            BR      11$
1447 006164  010002                    10$:    MOV     R0,R2                   ;R2 = ERROR CODE
1448 006166                                    RESTOR  R1
1449 006170                            11$:    RESTOR  R0
1450 006172                                    EXIT    MRG
```

```
                                        .SBTTL  MERGE FIXED LENGTH NODE

                                ;
                                ;   SUBROUTINE TO GET A FREE NODE FROM THE FREE QUEUE, COPY CONTENTS OF
                                ;   THE SOURCE NODE INTO IT, REMOVE NODE FROM FREE Q AND ADD IT TO
                                ;   INDICATED NODE CHAIN.
                                ;
                                ;   ON ENTRY,
                                ;       R1-> NODE CHAIN INDEX
                                ;       R2-> NODE THAT SHOULD PRECEED NEW NODE
                                ;       VVEC -> SOURCE NODE
                                ;
                                ;   ON RETURN,
                                ;       CC = 0          R2->NEW NODE
                                ;       CC = 1          R2 = ERROR CODE         ER114 = NO FREE NODES
                                ;
1467 006174                     MERGE:  SAVE    R0,R1
1468 006200  012701  000052'            MOV     #FREEQ,R1               ;R1->FREE Q
1469 006204  005761  177776             TST     N.NODE(R1)              ;ANY NODES LEFT?
1470 006210  003007                     BGT     1$                      ;BRANCH IF YES
1471 006212                             CALL    FRENOD                  ;DELETE OLD NODES
1472 006216  103004                     BCC     1$
1473 006220  012700  177616             MOV     #ER114,R0               ;NO FREE NODES
1474 006224  000261                     SEC
1475 006226  000456                     BR      10$
1476 006230  011100             1$:     MOV     (R1),R0                 ;R0->FREE NODE
1477                                                                     ;R1->FREE Q
1478 006232                             CALL    REMND                   ;REMOVE NODE FROM FREE Q
1479 006236  103002                     BCC     2$
1480 006240  000261                     SEC                             ;R0 HAS ERROR CODE
1481 006242  000450                     BR      10$
1482 006244                     2$:     SAVE    R0,R1,R2
1483 006252  010001                     MOV     R0,R1                   ;R1->FREE NODE
1484 006254  062701  000004             ADD     #4,R1                   ;R1 = DESTINATION ADDRESS
1485 006260  012700  000034'            MOV     #<VVEC+4>,R0            ;R0 = SOURCE ADDS
1486 006264  016702  000042'            MOV     N.LGT+VVEC,R2           ;R2 = VECTOR LENGTH (WORDS)
1487 006270  042702  100000             BIC     #BITNIB,R2              ;CLEAR NIBBLE INDICATOR FLAG
1488 006274  006302                     ASL     R2                      ;R2 = VECTOR LENGTH (BYTES)
1489 006276  020227  000106             CMP     R2,#NPEVSZ*2            ;COUNT MUST BE LESS THAN NODE LENGTH
1490 006302  003407                     BLE     3$                      ;BRANCH IF LESS
1491 006304                             RESTOR  R0,R1,R2
1492 006312  012700  177614             MOV     #ER116,R0               ;DATA LONGER THAN NODE
1493 006316  000261                     SEC
1494 006320  000421                     BR      10$
1495 006322  062702  000020     3$:     ADD     #<N.HEAD+4>,R2          ;R2 = NO. OF BYTES TO BE MOVED
1496 006326                             CALL    MOVDT                   ;COPY VVEC INTO FREE NODE
1497 006332                             RESTOR  R0,R1,R2                ;R0->NEW NODE
1498 006340                             RESTOR  R1                      ;R1->NODE CHAIN INDEX
1499                                                                     ;R2->NODE AFTER WHICH INSERT SHOULD BE MADE
1500 006342                             CALL    ADDMID                  ;ADD NODE TO NODE CHAIN
1501 006346  103003                     BCC     4$
1502 006350  010002                     MOV     R0,R2                   ;R2 = ERROR CODE
1503 006352  000261                     SEC
1504 006354  000405                     BR      11$
1505 006356  010002             4$:     MOV     R0,R2                   ;R2->NEW NODE
1506 006360  000241                     CLC
1507 006362  000402                     BR      11$
1508 006364  010002             10$:    MOV     R0,R2                   ;R2 = ERROR CODE
```

```
1509 006366                             RESTOR  R1
1510 006370                     11$:    RESTOR  R0
1511 006372                             EXIT    MERGE
```

```
                                        .SBTTL  FREE UP NODES

                                ;
                                ;   SUBROUTINE TO DELETE THOSE NODES FROM NODE CHAIN FLDC
                                ;   WHOSE NIBBLE POSITION NUMBER IS 2 OR MORE LESS THAN THE NIBBLE
                                ;   POSITION NUMBER OF THE CURRENT NODE.  THE NODES THUS RELEASED
                                ;   ARE RETURNED TO THE FREE NODE CHAIN.
                                ;   ON ENTRY,
                                ;       R1 -> FREEQ
                                ;       VVEC->  SOURCE NODE
                                ;   ON RETURN,
                                ;       CC =0           FREE NODE COUNT GREATER THAN ZERO
                                ;       CC =1           COULD NOT FREE ANY NODES
                                ;
1526 006374                     FRENOD: SAVE    R0,R1,R2,R3,R4,R5
1527 006410  116703  000040'           MOVB    N.POS+VVEC,R3           ;R3 = CHAR. POSITION NUMBER
1528 006414  162703  000003            SUB     #3,R3
1529 006420  012701  000040'    1$:     MOV     #FLDC,R1                ;R1->NODE CHAIN INDEX
1530 006424  016105  177776            MOV     N.NODE(R1),R5           ;R5 = NO. OF NODES IN CHAIN
1531 006430  001411                    BEQ     6$
1532 006432  011100                    MOV     (R1),R0                 ;R0-> 1ST NODE
1533 006434  120360  000010    2$:     CMPB    R3,N.POS(R0)            ;TEST NODE'S CHAR. POS. NUMBER
1534 006440  002403                    BLT     3$                      ;BRANCH IF POS. NO. NOT LOW ENOUGH
1535                            ;
1536                            ;   FOUND NODE THAT CAN BE DELETED
1537                            ;       R0-> NODE
1538                            ;       R1-> NODE CHAIN INDEX
1539                            ;
1540 006442                             CALL    DELNOD
1541 006446  000764                     BR      1$
1542 006450  011000             3$:     MOV     (R0),R0                 ;R0-> NEXT NODE
1543 006452  077510                     SOB     R5,2$
1544 006454                     6$:
1545 006454                             RESTOR  R0,R1,R2,R3,R4,R5
1546 006470  005761  177776            TST     N.NODE(R1)
1547 006474  003002                    BGT     4$
1548 006476  000261                    SEC
1549 006500  000401                    BR      5$
1550 006502  000241             4$:    CLC
1551 006504                      5$:    EXIT    FRENOD
```

```
                                        .SBTTL  WRITE TDCTA TO DISK
                                ;
                                ;   THE PURPOSE OF THIS SUBROUTINE IS TO WRITE ONE BLOCK OF
                                ;   THE TERM DETECTOR CONTROL TABLE TO DISK.
                                ;
1553                            WRTTDC::
1554                                    WRITE$  #EMAFDB,#TDCBUF         ;WRITE 1ST BUFFER BLOCK TO DISK
1555                                    WAIT$   #EMAFDB
1556                                    TSTB    IOST                    ;ANY ERRORS?
1557                                    BGT     1$                      ;BRANCH IF NO
1558 006506                            MOVB    IOST,R1
1559 006506                            MOV     R1,PAR2
1560 006524                            CALL    FCSERR
1561 006534  105767  000000'           JMP     EXIT
1562 006540  003010            1$:     EXIT    WRTTDC
1563 006542  116701  000000'
1564 006546  010167  001566'
1565 006552
1566 006556  000167  000176
1567 006562
```

```
                                        .SBTTL  ERROR HANDLING ROUTINE
                                ;
                                ;   DIRECTIVE ERROR
                                ;
1573 006564  011667  001564'    DIRERR::MOV     (SP),PAR1               ;PC AT DIRECTIVE ERROR
1574 006570  016767  000000G 001566'    MOV     $DSW,PAR2               ;$DSW
1575 006576                             MOUT$S  #MSG1,#PAR1
1576 006616  005726                     TST     (SP)+
1577 006620  000167  000134             JMP     EXIT                    ;EXIT
1578                            ;
1579                            ;   FCS ERROR
1580                            ;
1581 006624  011667  001564'    FCSERR::MOV     (SP),PAR1               ;PC
1582 006630                             MOUT$S  #MSG2,#PAR1
1583 006650  005726                     TST     (SP)+                   ;RESTORE STACK
1584 006652  000167  000102             JMP     EXIT
1585                            ;
1586                            ;   TDCT OVERFLOW
1587                            ;
1588 006656                     TDCERR:
1589 006656  011667  001564'    ERROR2::MOV     (SP),PAR1
1590 006662                             MOUT$S  #MSG5,#PAR1
1591 006702  000167  000052             JMP     EXIT
1592                            ;
1593                            ;   NODE ERROR
1594                            ;
1595 006706  011667  001564'    NODERR::MOV     (SP),PAR1
1596 006712  010067  001566'            MOV     R0,PAR2
1597 006716                             MOUT$S  #MSG4,#PAR1
1598 006736                             EXIT    NODERR
1599                            ;
1600 006740                     ERRINT: MOUT$S  #MSG6,#PAR1
1601                            ;
1602 006760                     EXIT::
1603 006760  012767  177777  000000C         MOV     #-1,SNDBUF+SD.FSA
1604 006766  000167  172054             JMP     QT1.7
1605                            ;
1606 001300                             .PSECT  DATA
1607                            ;   ERROR MESSAGES ARE PRINTED ON TI BY MSGOUT
1608                            ;   STRING DESCRIPTORS
1609                            ;
1610 001300  000041             MSG1:   .WORD   LN1E-LN1                ;LENGTH OF FORMAT STRING
1611 001302  001330'                    .WORD   LN1                     ;ADDS OF FORMAT STRING
1612 001304  000040             MSG2:   .WORD   LN2E-LN2
1613 001306  001371'                    .WORD   LN2
1614 001310  000014             MSG3:   .WORD   LN3E-LN3
1615 001312  001431'                    .WORD   LN3
1616 001314  000041             MSG4:   .WORD   LN4E-LN4
1617 001316  001445'                    .WORD   LN4
1618 001320  000032             MSG5:   .WORD   LN5E-LN5
1619 001322  001506'                    .WORD   LN5
1620 001324  000023             MSG6:   .WORD   LN6E-LN6
1621 001326  001540'                    .WORD   LN6
1622                            ;
1623                            ;   FORMAT STRINGS
1624                            ;
1625 001330  104  111  122      LN1:    .ASCIZ  /DIR. ERROR, PC = %10, $DSW = %1D/
```

```
1626 001371                            LN1E:
1627 001371     106     103     123    LN2:    .ASCIZ  /FCS  ERROR, PC = %1O, ERR = %1D/
1628 001431                            LN2E:
1629 001431     121     124     061    LN3:    .ASCIZ  /QT1 EXITING/
1630 001445                            LN3E:
1631 001445     116     117     104    LN4:    .ASCIZ  /NODE ERROR, PC = %1O, CODE = %1D/
1632 001506                            LN4E:
1633 001506     124     104     103    LN5:    .ASCIZ  /TDCT OVERFLOW.   PC = %1O/
1634 001540                            LN5E:
1635 001540     111     116     124    LN6:    .ASCIZ  /INTERNAL ERROR=%1D/
1636 001563                            LN6E:
1637                                    .EVEN
1638                                ;
1639                                ;   PARAMETERS
1640                                ;
1641 001564  000000                    PAR1:   .WORD   0                       ;PC
1642 001566  000000                    PAR2::  .WORD   0                       ;$DSW OR F.ERR
```

```
                                        .SBTTL  TERM DETECTOR CONTROL TABLE BUFFER
                                ;
                                ;   TERM DETECTOR CONTROL TABLE BUFFER
                                ;
                                ;   LAYOUT OF INCOMPLETE STATE WORD:
                                ;       SEG1 =  0
                                ;       SEG2 =  NODE ADDRESS OF STATE'S VVEC
                                ;       SEG3 =  INDEX STATE, BIT0 = 1
                                ;               BIT1 = 1 MEANS THAT VLDC COLUMN HAS  ALREADY BEEN CONSIDERED
                                ;
001570  000002                 ELSEDF: .WORD   2                       ;ELSE DEFAULT STATE ADDS
001572  000003                 INTDEF::.WORD   3                       ;INTERWORD DEFAULT ADDS
001574  000000                 C.FLDC: .WORD   0                       ;ADDRESS OF CURRENT FLDC NODE
001576  000000                 C.VLDC: .WORD   0                       ;ADDRESS OF CURRENT VLDC NODE
                                ;
                                ;
                                ;
001600  000004                 LOWADD: .WORD   4                       ;ADDRESS OF LOWEST INCOMPLETE STATE
001602  000030                 LOWOFF::.WORD   TDCT1                   ;OFFSET OF LOWEST INCOMPLETE STATE
001604  000004                 FSAADD::.WORD   4                       ;ADDRESS OF CURRENT TDCT STATE
001606  000030                 FSAOFF::.WORD   TDCT1                   ;OFFSET OF CURRENT TDCT STATE
001610  000005                 NXTADD::.WORD   5                       ;NEXT AVAILABLE STATE ADDRESS
001612  000036                 NXTOFF::.WORD   FREE1                   ;NEXT AVAILABLE OFFSET IN TDCT
001614  017732                 TDCMAX::.WORD   TDCEND-TDCT-6           ;LARGEST AVAILABLE TDCT OFFSET
001616  001630'                TDCADR::.WORD   TDCT                    ;VIRTUAL START OF TDCT BUFFER
                                ;
                                ;   TDCT BUFFER
                                ;
001620     063     103          TDCBUF: .ASCII  /3C/
001622  000000                 TDCLGT: .WORD   0                       ;BLOCK LENGTH OF TDCT-A
001624                                  .BLKW   2                       ;FILLER TO ALIGN 3-WORD TDCT ENTRIES
                                                                        ;ON 1024. WORD BLOCK BOUNDARIES
                                ;
                                ;   TDCT STATE WORD 0
                                ;
001630  000002                 TDCT::  .WORD   2                       ;NEW ELSE DEF, ADD'S
001632  000002                         .WORD   2                       ;NEXT STATE ADD'S
001634  130000                         .WORD   130000                  ;STATE : CHANGE ELSE DEF,
                                ;
                                ;   TDCT STATE WORD 1
                                ;
001636  000003                         .WORD   3                       ;NEW INT, DEF, ADD'S
001640  000004                         .WORD   4                       ;NEXT STATE ADD'S
001642  124000                         .WORD   124000                  ;STATE : CHANGE INT, DEF
                                ;
                                ;   TDCT STATE WORD 2 (ELSE DEF, STATE WORD)
                                ;
001644  000000                         .WORD   0                       ;MATCH BYTE (INTERWORD CHAR,)
001646  000001                         .WORD   1                       ;NEXT STATE ADD'S
001650  114000                         .WORD   114000                  ;STATE : JUMP/SEQUENTIAL
                                ;
                                ;   TDCT STATE WORD 3 (INTERWORD DEF, STATE WORD)
                                ;
001652  000002                         .WORD   2                       ;NEW ELSE DEF, ADDS
001654  000001                         .WORD   1                       ;NEXT STATE ADD'S
001656  130000                         .WORD   130000                  ;STATE : CHANGE ELSE DEF,
                                ;
```

```
                                        .SBTTL  NODE HEADER DEFINITIONS
000030                                  .PSECT  NODES

                                ;
                                ;   NODE LENGTHS
                                ;
000014                          N.HEAD  =       12.     ;NODE HEADER LENGTH (BYTES )
000122                          N.TOT   =       N.HEAD+<NPEVSZ*2>;TOTAL NODE LENGTH (FIXED LENGTH NODES)
000006                          N.LEN   =       6       ;LENGTH OF INDEX HEADER
                                ;
                                ;   NODE HEADER OFFSET DEFINITIONS
                                ;
000000                          N.FWD   ==      0               ;FORWARD POINTER
000002                          N.BACK  ==      N.FWD+WORD1     ;BACKWARD POINTER
000004                          N.FSA   ==      N.BACK+WORD1    ;FSA STATE ADDRESS
000006                          N.ELSE  ==      N.FSA+WORD1     ;ELSE DEFAULT STATE ADDRESS
000010                          N.POS   ==      N.ELSE+WORD1    ;NIBBLE POSITION NUMBER (LOW BYTE)
                                                                ;BIT15 = 1 IMPLIES NODE REPRESENTS ELSE
                                                                ;DEFAULT OVERRIDE STATE
000012                          N.LGT   ==      N.POS+WORD1     ;VECTOR LENGTH
                                                                ;BIT15 = 0 : FIRST NIBBLE
                                                                ;BIT15 = 1 : SECOND NIBBLE
000014                          N.VEC   ==      N.LGT+WORD1     ;OFFSET OF START OF VECTOR
                                ;
                                ;   NODE CHAIN INDEX TABLE OFFSET DEFINITIONS
                                ;
177776                          N.NODE  =       -2              ;NUMBER OF NODES IN Q
                                ;
                                ;
                                        .SBTTL  E-MATRIX BUFFERS
000000                                  .PSECT  EMA
                                ;
                                ;   VMASK BUFFER (IT OVERLAYS EMALGT AND VI)
                                ;
000000                          VMASK:: .BLKB   N.POS
                                ;
                                ;   VI
                                ;
000010  000000                  EMALGT: .WORD   0                       ;BLOCK LENGTH OF EMA
000012  000000                  VILGT:  .WORD   0                       ;NO. OF ENTRIES IN VI
000014                          VI::    .BLKW   VI1MSZ-2
                                ;
                                ;   EMA
                                ;
001010                          EMA::   .BLKW   EMAMSZ
```

```
                                        .SBTTL  DON'T CARE NODE POOL
                                        .PSECT  NODES
                                ;
                                ;   NODE CHAIN INDEX TABLE FOR VLDC AND FLDC
                                ;
                                ;
                                ;   VLDC NODE CHAIN INDEX
                                ;
                                        .WORD   0       ;NO. OF NODES IN CHAIN
                                VLDC:   .WORD   .       ;PT TO 1ST NODE
                                        .WORD   .-2     ;PT TO LAST NODE
                                ;
                                ;   FLDC NODE CHAIN INDEX
                                ;
                                        .WORD   0       ;NO. OF NODES IN NODE CHAIN
                                FLDC:   .WORD   .       ;PT TO 1ST NODE
                                        .WORD   .-2     ;PT TO LAST NODE
                                ;
                                ;   DON'T CARE (VLDC AND FLDC) NODE POOL AND INDEX
                                ;
                                BPOOL:  .WORD NPOOL     ;BEGINNING OF NODE POOL
                                EPOOL:  .WORD NPOOLE    ;END
                                        .WORD NPECNT    ;# OF FREE NODES
                                FREEQ:  .WORD NPOOL     ;PT TO 1ST NODE
                                        .WORD NPOOLE-N.TOT  ;PT TO LAST NODE
                                ;
                                ;   NODE POOL REGION
                                ;
                                NPOOL:  .WORD .+N.TOT   ;PT TO 2ND NODE
                                        .WORD FREEQ     ;PT TO PREV, INDEX
                                        .BLKW <N.TOT-4>/2  ;LEN OF NODE DATA
                                        .REPT NPECNT-2  ;REPT
                                        .WORD .+N.TOT   ;FRWD,PT TO NEXT
                                        .WORD .-N.TOT-2 ;BACK, PT TO PREV
                                        .BLKW <N.TOT-4>/2  ;DATA
                                        .ENDM
                                        .WORD FREEQ     ;FRWD, PT TO INDEX
                                        .WORD .-N.TOT-2  ;BACK, PT TO PREV
                                        .BLKW <N.TOT-4>/2
                                NPOOLE = .              ;END OF REGION + 1
```

```
                                        .SBTTL  NIBBLE NODE POOLS (VARIABLE LENGTH NODES)
                                ;
                                ;   NODE CHAIN INDEX TABLE - POOL 1
                                ;   (ONE CHAIN PER NIBBLE VALUE - 16 NIBBLES)
                                ;
020066  000000                          .WORD   0               ;NO. OF NODES IN CHAIN
020070  020070'                 NCIT1:  .WORD   .               ;FORWARD POINTER
020072  020070'                         .WORD   .-2             ;BACKWARD POINTER
        000017                          .REPT   15.
                                        .WORD   0               ;NO. OF NODES IN CHAIN
                                        .WORD   .               ;FORWARD POINTER
                                        .WORD   .-2             ;BACKWARD POINTER
                                        .ENDM
                                ;
                                ;   NODE POOL REGION - POOL 1
                                ;
020226                          NP1:    .BLKB   NP1MSZ
        030226'                 NP1E    =       .               ;END OF REGION + 1
                                ;
                                ;   NODE CHAIN INDEX TABLE - POOL 2
                                ;   (ONE CHAIN PER NIBBLE VALUE - 16 NIBBLES)
                                ;
030226  000000                          .WORD   0               ;NO. OF NODES IN CHAIN
030230  030230'                 NCIT2:  .WORD   .               ;FORWARD POINTER
030232  030230'                         .WORD   .-2             ;BACKWARD POINTER
        000017                          .REPT   15.
                                        .WORD   0               ;NO. OF NODES IN CHAIN
                                        .WORD   .               ;FORWARD POINTER
                                        .WORD   .-2             ;BACKWARD POINTER
                                        .ENDM
                                ;
                                ;   NODE POOL REGION - POOL 2
                                ;
030366                          NP2:    .BLKB   NP1MSZ
        040366'                 NP2E    =       .               ;END OF REGION + 1
```

```
                                        .SBTTL  REMOVE NODE FROM QUEUE (FIXED LENGTH NODES)
006772                                  .PSECT  CODE
                                ;
                                ;
                                ;
                                ;REMOVE NODE FROM CHAIN - THIS ROUTINE ACTUALLY DELETES A NODE
                                ;     FROM A Q.  THE NODE MAY BE RANDOM WITHIN A Q.
                                ;
                                ;INPUT: R0= NODE ADDR
                                ;       Ri= Q INDEX ADDR
                                ;  JSR PC,REMND
                                ;
                                ;OUTPUT: IF C=0, OK
                                ;        IF C=1 R0= 102 NODE NOT IN Q, 113 THRESHOLD REACHED (FREE Q ONLY)
                                ; R0-R5 RESTORE
                                ; INDEX NODE CNT DECREMENTED
                                ;
006772  010246                  REMND:  MOV     R2,-(SP)
006774  004767  000360                  JSR     PC,VALND
007000  103412                         BCS     REMND2
007002  005361  177776                  DEC     N.NODE(R1)      ;DECR ENTRY CNT
007006  011070  000002                  MOV     (R0),@2(R0)     ;MOV CUR FWD PTR TO THE PREVIOUS NODE
007012  011002                         MOV     (R0),R2         ;ADDR OF NEXT NODE
007014  016062  000002  000002          MOV     2(R0),2(R2)     ;NEXT NODE PT TO PREVIOUS, OVER THE CURRENT
007022  000241                         CLC
007024  000401                         BR      REMND3
007026  000261                  REMND2: SEC
007030  012602                  REMND3: MOV     (SP)+,R2
007032  000207                         RTS     PC
```

```
1858                                    .SBTTL  ADD NODE TO MIDDLE OF Q (VARIABLE LENGTH NODES)
1859                                ;
1860                                ;ADD NODE TO MIDDLE OF Q - THIS ROUTINE ACTUALLY PERFORMS
1861                                ;LINKING
1862                                ;
1863                                ;INPUT:   R2-->NODE AFTER WHICH INSERT TO BE MADE
1864                                ;         R1-->Q INDEX
1865                                ;         R0-->NODE TO BE INSERTED
1866                                ;
1867                                ;JSR   PC,ADDMID
1868                                ;
1869                                ;OUTPUT:    IF C=0, NODE ADDED AND INDEX NODE COUNT INCREMENTED
1870                                ;           IF C=1, R0=104,  NODE ADDR OUT OF REGION
1871                                ;
1872 007034                        ADDNOD:
1873 007034  026700  000000'               CMP     BNPN,R0                 ;REGION START
1874 007040  101016                        BHI     1$                      ;NODE OUT OF REGION
1875 007042  020067  000002'               CMP     R0,ENPN                 ;END OF REGION
1876 007046  101013                        BHI     1$                      ;NODE OUT OF REGION
1877 007050  011210                        MOV     (R2),(R0)               ;NEW NODE FWD PTR
1878 007052  010260  000002                MOV     R2,2(R0)                ;NEW NODE BACK POINTER
1879 007056  010012                        MOV     R0,(R2)                 ;PREV NODE FWD PTR
1880 007060  011002                        MOV     (R0),R2
1881 007062  010062  000002                MOV     R0,2(R2)                ;NEXT NODE BACK POINTER
1882 007066  005261  177776                INC     N.NODE(R1)              ;INCREMENT NODE COUNT
1883 007072  000241                        CLC
1884 007074  000403                        BR      2$
1885 007076  012700  177630        1$:     MOV     #ER104,R0               ;NODE OUT OF REGION
1886 007102  000261                        SEC
1887 007104  000207                2$:     RTS     PC
```

```
                                        .SBTTL  ADD NODE TO TOP OF Q (FIXED LENGTH NODES)
                                ;
                                ;
                                ;ADD NODE TO TOP OF CHAIN
                                ;
                                ;INPUT: R0=NODE ADDR
                                ;       R1= Q INDEX
                                ;
                                ;       JSR PC,ADDTP
                                ;
                                ;OUTPUT: IF C=0, R0-R5 RESTORED
                                ;       INDEX NODE CNT INCREMENTED
                                ;       IF C=1, R0=104 OUT OF REGION
                                ;
007106  010246                  ADDTP:  MOV     R2,-(SP)
007110  020067  000044'                 CMP     R0,BPOOL        ;REGION START
007114  103420                          BLO     2$              ;OUT OF REG
007116  020067  000046'                 CMP     R0,EPOOL        ;REGION END
007122  103015                          BHIS    2$              ;OUT OF REG
007124  004767  000116                  JSR     PC,ZEROFN       ;ZERO OUT NODE WHEN RETURNING TO FREE QUEUE
007130  011102                          MOV     (R1),R2         ;PREV TOP PTR, FROM INDEX
007132  010011                          MOV     R0,(R1)         ;NEW FRWD , IN INDEX
007134  010210                          MOV     R2,(R0)         ;FRWD PTER IN NEW NODE
007136  010062  000002                  MOV     R0,2(R2)        ;PT PREV TOP TO NEW TOP; BACK
007142  010160  000002                  MOV     R1,2(R0)        ;NEW NODE PT TO INDEX, BACK
007146  005261  177776                  INC     N.NODE(R1)      ;NODE CNT
007152  000241                          CLC
007154  000403                          BR      1$
007156  012700  177630          2$:     MOV     #ER104,R0       ;OUT OF REGION
007162  000261                          SEC
007164  012602                  1$:     MOV     (SP)+,R2
007166  000207                          RTS     PC
```

```
                                        .SBTTL  ADD NODE TO MIDDLE OF Q (FIXED LENGTH NODES)
                                ;
                                ;
                                ;
                                ;ADD NODE TO MIDDLE OF Q - THIS ROUTINE ACTUALLY PERFORMS
                                ;LINKING
                                ;
                                ;
                                ;INPUT:   R2-->NODE AFTER WHICH INSERT TO BE MADE
                                ;         R1-->Q INDEX
                                ;         R0-->NODE TO BE INSERTED
                                ;
                                ;
                                ;JSR   PC,ADDMID
                                ;
                                ;
                                ;OUTPUT:    IF C=0, NODE ADDED AND INDEX NODE COUNT INCREMENTED
                                ;          IF C=1, R0=104,  NODE ADDR OUT OF REGION
                                ;
                                ;
1942 007170                     ADDMID:
1943 007170  020067  000044'            CMP     R0,BPOOL                ;REGION START
1944 007174  103420                     BLO     1$                      ;NODE OUT OF REGION
1945 007176  020067  000046'            CMP     R0,EPOOL                ;END OF REGION
1946 007202  103015                     BHIS    1$                      ;NODE OUT OF REGION
1947 007204  004767  000036            JSR     PC,ZEROFN               ;ZERO OUT FREE QUEUE NODES
1948 007210  011210                     MOV     (R2),(R0)               ;NEW NODE FWD PTR
1949 007212  010260  000002            MOV     R2,2(R0)                ;NEW NODE BACK POINTER
1950 007216  010012                     MOV     R0,(R2)                 ;PREV NODE FWD PTR
1951 007220  011002                     MOV     (R0),R2
1952 007222  010062  000002            MOV     R0,2(R2)                ;NEXT NODE BACK POINTER
1953 007226  005261  177776            INC     N.NODE(R1)              ;INCREMENT NODE COUNT
1954 007232  000241                     CLC
1955 007234  000403                     BR      2$
1956 007236  012700  177630     1$:    MOV     #ER104,R0               ;NODE OUT OF REGION
1957 007242  000261                     SEC
1958 007244  000207             2$:    RTS     PC
```

```
                                        .SBTTL  ZERO OUT FREE QUEUE NODE
                                ;
                                ;
                                ;
                                ;THIS ROUTINE CHECKS TO SEE IF A NODE IS BEING ADDED TO THE FREE QUEUE.
                                ;IF SO, THE NODE IS ZEROED OUT BEFOREHAND.
                                ;
                                ;
                                ;INPUT:  R0-->NODE        R1-->QUEUE
                                ;OUTPUT: R0-->NODE        R1-->QUEUE
                                ;
                                ;
                                ;R0, R2 ARE SAVED AND RESTORED
                                ;
                                ;
007246                          ZEROFN:
007246  020127  000052'                 CMP     R1,#FREEQ       ;IS THE FREE QUEUE HAVING A NODE RETURNED?
007252  001010                          BNE     2$              ;NO, EXIT
007254  010046                          MOV     R0,-(SP)
007256  010246                          MOV     R2,-(SP)
007260  012702  000122                  MOV     #N.TOT,R2       ;R2=TOTAL BYTES IN A NODE
007264                          1$:
007264  105020                          CLRB    (R0)+           ;CLEAR BYTE
007266  077202                          SOB     R2,1$           ;LOOP TILL ALL BYTES CLEARED
007270  012602                          MOV     (SP)+,R2
007272  012600                          MOV     (SP)+,R0
007274                          2$:
007274  000207                          RTS     PC
```

```
                                                .SBTTL  DELETE NODE
                                        ;
                                        ;
                                        ;   DELETE A NODE FROM INDICATED NODE CHAIN:
                                        ;
                                        ;INPUT  R0-->NODE TO BE DELETED
                                        ;       R1-->NODE INDEX
                                        ;
                                        ;OUTPUT
                                        ;       C=0, OPERATION SUCCESSFUL, NO FURTHER PROCESSING NOW
                                        ;
                                        ;
1989                                    DELNOD:
1990 007276  010146                             MOV     R1,-(SP)        ;SAVE INDEX VALUE WHICH IS CLOBERRED IN ROUTINE
1991 007300  004767  177466                     JSR     PC,REMND        ;DELETE NODE
1992 007304  103004                             BCC     1$
1993 007306  004767  177374                     JSR     PC,NODERR
1994 007312  000167  177442                     JMP     EXIT
1995 007316                             1$:
1996 007316  012701  000052'                    MOV     #FREEQ,R1
1997 007322  004767  177560                     JSR     PC,ADDTP        ;RETURN NODE TO FREE Q
1998 007326  103004                             BCC     2$
1999 007330  004767  177352                     JSR     PC,NODERR
2000 007334  000167  177420                     JMP     EXIT
2001 007340  000241                    2$:     CLC
2002 007342  012601                            MOV     (SP)+,R1
2003 007344  000207                            RTS     PC
```

```
                                                .SBTTL  MOVE DATA BY COUNT
                                        ;
                                        ;MOVE DATA BY CNT
                                        ;
                                        ;INPUT:  R0= SOURCE ADDR
                                        ;        R1= DEST ADDR
                                        ;        R2= COUNT (BYTES)
                                        ;
                                        ;        JSR PC,MOVDT
                                        ;
                                        ;OUTPUT: R0-R2 CHANGED
                                        ;
007346  005702                  MOVDT:  TST     R2
007350  001402                          BEQ     2$
007352  112021                  1$:     MOVB    (R0)+,(R1)+
007354  077202                          SOB     R2,1$
007356                          2$:
007356  000207                          RTS     PC
```

```
                                        .SBTTL  VALIDATE Q NODE ADDRESS
                                ;
                                ;
                                ;DETERMINE IF NODE ADDR IS VALID
                                ;    I.E. CORRECT Q AND CORRECT REGION
                                ;
                                ;INPUT: R0= NODE ADDR
                                ;       R1= Q INDEX
                                ;    JSR PC,VALND
                                ;
                                ;OUTPUT: R2-R5 RESTORED
                                ;      IF C =0,OK,R0 + R1 RESTORED
                                ;       IF C = 1,ERROR,R0=102-NODE NOT IN Q
                                ;                      R0=104, NODE ADDR NOT IN REGION
                                ;
2051 007360  010146             VALND:  MOV     R1,-(SP)
2052 007362  010246                     MOV     R2,-(SP)
2053 007364  105761  177776             TSTB    N.NODE(R1)              ;ANY ENTRIES
2054 007370  001423                     BEQ     4$                      ;=0,GO
2055 007372  020067  000044'            CMP     R0,BPOOL                ;REGION START
2056 007376  103414                     BLO     3$                      ;OUT OF REGION
2057 007400  020067  000046'            CMP     R0,EPOOL                ;REGION END +1
2058 007404  103011                     BHIS    3$                      ;OUT OF REGION
2059 007406  116102  177776             MOVB    N.NODE(R1),R2           ;# OF ENTRIES
2060 007412  011101             1$:     MOV     (R1),R1                 ;1ST NODE ADDR
2061 007414  020100                     CMP     R1,R0                   ;VALID NODE
2062 007416  001402                     BEQ     2$                      ;YES,GO
2063 007420  077204                     SOB     R2,1$                   ;LOOP
2064 007422  000406                     BR      4$                      ;ERROR
2065 007424  000241             2$:     CLC
2066 007426  000407                     BR      5$
2067 007430  000261             3$:     SEC
2068 007432  012700  177630             MOV     #ER104,R0
2069 007436  000403                     BR      5$
2070 007440  000261             4$:     SEC
2071 007442  012700  177632             MOV     #ER102,R0
2072 007446  012602             5$:     MOV     (SP)+,R2
2073 007450  012601                     MOV     (SP)+,R1
2074 007452  000207                     RTS     PC                      ;RET
```

```
                                        .SBTTL  COMPRESS TDCT
                                ;
                                ;
007454                          MOVTDC::SAVE    R0,R1,R2
007462  012700  001630'                MOV     #TDCT,R0                ;R0-> TDCT BUFFER
007466  010001                         MOV     R0,R1                   ;R1 = DESTINATION ADDRESS
007470  062700  003770                 ADD     #N,BUFB-10,R0           ;R0 = SOURCE ADDRESS
007474  012702  003770                 MOV     #<N,BUFB-10>,R2         ;R2 = BYTE COUNT OF TDCT WRITTEN TO DISK
007500  160267  001602'                SUB     R2,LOWOFF               ;RESET LOWOFF
007504  160267  001606'                SUB     R2,FSAOFF               ;RESET FSAOFF
007510  160267  001612'                SUB     R2,NXTOFF               ;RESET NXTOFF
007514  160267  001616'                SUB     R2,TDCADR               ;ADJUST VIRTUAL TDCT BUF START
007520  016702  001612'                MOV     NXTOFF,R2               ;R2 = BYTE COUNT OF ACTIVE TDCT
007524  016767  001606'  000024'       MOV     FSAOFF,R11              ;UPDATE R11
                                ;
                                ;   MOVE ACTIVE TDCT TO START OF BUFFER
                                ;
007532                                  CALL    MOVDT
                                ;
                                ;   CLEAR UNUSED PORTION OF TDCT BUFFER
                                ;       R1 -> ONE WORD BEYOND ACTIVE TDCT
                                ;
007536  012702  021570'                 MOV     #TDCEND,R2              ;R2->END OF TDCT BUFFER
007542  160102                         SUB     R1,R2                   ;R2=BYTE LENGTH OF FREED UP BUFFER SPACE
007544  006202                         ASR     R2                      ;R2=WORD    "       "       "     "
007546  005021                  1$:    CLR     (R1)+                   ;ZERO OUT THIS SPACE
007550  077202                         SOB     R2,1$
007552                                 RESTOR  R0,R1,R2
007560                                 EXIT    MOVTDC
        000000'                        .END    START
```

```
ADDMID  007170R     017 B.HBLK  000120      010 EMAMSZ= 016000 G       FN.DBS  000022      011 F.MBFG= 000856
ADDNOD  007034R     017 B.HDOC  000114      010 EMBCLS= 000004 G       FN.DHR  000040      011 F.NRBD= 000024
ADDTP   007106R     017 B.HRLP  000126      010 EMBCUT= 001130 G       FN.EMA  000012      011 F.NREC= 000030
ASTKMX= 000400 G        B.HRLR  000122      010 EMBMSZ= 004400 G       FN.EMB  000014      011 F.OVBS= 000030
AVELGT= 000025 G        B.HRLW  000124      010 EMCMSZ= 001000 G       FN.EMC  000016      011 F.RACC= 000016
BITNIB= 100000 G        B.NMBR  000052      010 EMXCHF= 020000 G       FN.FSA  000000      011 F.RATT= 000001
BITVAL= 000000          B.NQRY  000232      010 EMXDTF= 010000 G       FN.FSB  000002      011 F.RCNM= 000034
BIT0  = 000001          B.QLSZ  000106      010 EMXEXF= 000001 G       FN.FSC  000004      011 F.RCTL= 000017
BIT1  = 000002          B.QMAP  000234      010 EMXFDC= 002000 G       FN.LGQ  000034      011 F.RSIZ= 000002
BIT10 = 002000          B.QSPL  000316      010 EMXMCD= 000002 G       FN.LGU  000036      011 F.RTYP= 000000
BIT11 = 004000          B.QTTM  000076      010 EMXMCF= 001000 G       FN.MFO  000024      011 F.SEQN= 000100
BIT12 = 010000          B.QUQP  000056      010 EMXMSZ= 016000 G       FN.MHR  000010      011 F.SPDV= 000072
BIT13 = 020000          B.SFDB  000010      010 EMXMTV= 040000 G       FN.NMB  000044      011 F.SPUN= 000074
BIT14 = 040000          B.SIZE  000772      010 EMXNB1= 000002 G       FN.QLS  000006      011 F.STBK= 000036
BIT15 = 100000          B.SNDP  000012      010 EMXNB2= 000004 G       FN.ORY  000020      011 F.UNIT= 000136
BIT2  = 000004          B.SSQ   000004      010 EMXNFD= 000400 G       FN.SF0  000030      011 F.URBD= 000020
BIT3  = 000010          B.SSQF  000050      010 EMXNSQ= 100000 G       FN.SF1  000032      011 F.VBN = 000064
BIT4  = 000020          B.STAT  000044      010 EMXNVD= 001000 G       FN.SHD  000042      011 F.VBSZ= 000060
BIT5  = 000040          B.STTE  000053      010 EMXSZF= 002000 G       FO.RD = ****** GX       GTIM1   000220RG    014
BIT6  = 000100          B.UDOC  000110      010 EMXTRL= 010000 G       FO.WRT= ****** GX       GVMASK= ****** GX
BIT7  = 000200          CBIT  = 010000 G        EMXVDC= 004000 G       FREEQ   000052R     021 G.TICP= 000016
BIT8  = 000400          CF.B0 = 000070          EMXVVV= 000001 G       FREE1 = 000036          G.TICT= 000014
BIT9  = 001000          CF.B2 = 000067          EMXZNF= 004000 G       FRENOD  006374R     017 G.TIDA= 000004
BLOCK = 001000          CF.B4 = 000066          EMX0VD= 000200 G       FRNPC   000024R     021 G.TIHR= 000006
BNPC    000014R     021 CF.B6 = 000065          ENCNT = 000004 G       FRNPN   000010R     021 G.TIMI= 000010
BNPN    000000R     021 CF.DR0= 000064          ENPC    000016R     021 FSAADD  001604RG  015 G.TIMO= 000002
BPOOL   000044R     021 CF.DR1= 000063          ENPN    000002R     021 FSAOFF  001606RG  015 G.TISC= 000012
BS.CLS= 000002          CPIXMK= 177770 G        EOBIT = 010000 G       F.ACTL= 000076          G.TIYR= 000000
BS.DBU= 000004          CPIXSZ= 000010 G        EPOOL   000046R     021 F.ALOC= 000040          HOUR.1= 000020
BS.INA= 000000          C.FLDC  001574R     015 ERRINT  006740R     017 F.BBFS= 000062          HOUR.2= 000006
BS.OPN= 000001          C.VLDC  001576R     015 ERROR2  006656RG    017 F.BDB = 000070          HSTKMX= 000100 G
BS.SRC= 000003          DBSLEN= 000116          ER102 = 177632          F.BGBC= 000057          INTDEF  001572RG    015
BYTE0 = 000000          DELNOD  007276R     017 ER104 = 177630          F.BKDN= 000026          INTERW  003412R     017
BYTE1 = 000001          DELTIM  000000RG    014 ER114 = 177616          F.BKDS= 000020          IOST    000000R     015
BYTE2 = 000002          DH.BF0  000002      005 ER116 = 177614          F.BKEF= 000050          IO.RVB= ****** GX
BYTE3 = 000003          DH.BF1  000004      005 EXIT    006760RG    017 F.BKP1= 000051          IWBIT = 004000 G
BYTE4 = 000004          DH.CTL  000000      005 FCSERR  006624RG    017 F.BKST= 000024          JMPBIT= 004000 G
BYTE5 = 000005          DH.DMC  000010      005 FDC     002552R     017 F.BKVB= 000064          JUMP    004746R     017
BYTE6 = 000006          DH.FLG  000006      005 FDC.1   002636R     017 F.CHR = 000075          LN1     001330R     015
BYTE7 = 000007          DIRERR  006564RG    017 FD.BLK= ****** GX       F.CNTG= 000034          LN1E    001371R     015
BYTE8 = 000010          DN.DCK  000000      013 FD.FID  000000      003 F.DFNB= 000046          LN2     001371R     015
BYTE9 = 000011          DN.NTP  000004      013 FD.FNB  000006      003 F.DSPT= 000044          LN2E    001431R     015
BYTVAL= 000012          DN.NXT  000006      013 FD.FVR  000004      003 F.DVNM= 000134          LN3     001431R     015
B.BSTA  000054      010 DN.ROT  000002      013 FD.LEN  000010      003 F.EFBK= 000010          LN3E    001445R     015
B.CNTX  000046      010 DN.SIZ  000010      013 FD.RWM= ****** GX       F.EFN = 000050          LN4     001445R     015
B.CQUQ  000060      010 EFN.2 = 000002          FLDC    000040R     021 F.EOBB= 000032          LN4E    001506R     015
B.FEMA  000132      010 EIXCNT= 000010 G        FLIXMK= 177740 G        F.ERR = 000052          LN5     001506R     015
B.FEMB  000142      010 ELSBIT= 010000 G        FLIXSZ= 000040 G        F.FACC= 000043          LN5E    001540R     015
B.FEMC  000152      010 ELSCLS= 000007 G        FLUCLS= 000001 G        F.FFBY= 000014          LN6     001540R     015
B.FFSA  000202      010 ELSEDF  001570R     015 FLUCUT= 000016 G        F.FNAM= 000110          LN6E    001563R     015
B.FFSB  000212      010 ELSMSK= 177761 G        FLUMD = ****** GX       F.FNB = 000102          LOGCLS= 000002 G
B.FFSC  000222      010 EMA     001010RG    022 FMEPSZ= 000400 G        F.FTYP= 000116          LOWADD  001600R     015
B.FMAR  000172      010 EMACLS= 000003 G        FMIXSZ= 000001 G        F.FVER= 000120          LOWOFF  001602RG    015
B.FQLS  000162      010 EMACUT= 001700 G        FMNPSZ= 002260 G        F.HIBK= 000004          LUNFIL= 000004
B.FSAZ  000100      010 EMADNB  001242R     015 FNPOSZ= 000400 G        F.LUN = 000042          L$STAT= 000006 G
B.FSBZ  000102      010 EMAFDB  001102RG    015 FNPSZ = 005734 G        F.MBCT= 000054          M       = 000062
B.FSCZ  000104      010 EMALGT  000010R     022 FN.DBR  000026      011 F.MBC1= 000055          MASKEM  001366R     017
```

```
MATNOD  005722R     017 N.FTYP= 000014          SD.SEC= 000012 G    003 ST.QRY  000002      006 T.SBY1= 000000 G
MERGE   006174R     017 N.FVER= 000016          SD.TIC= 000014 G    003 ST.QSZ  000034      006 T.SBY2= 000001 G
MIN.1 = 000022          N.FWD = 000000 G        SECBUF  000206RG    014 ST.SCH  000040      006 T.SBY3= 000002 G
MIN.2 = 000010          N.HEAD= 000014          SEC.1 = 000024          ST.UHL  000004      006 T.STAD= 000002 G
MOVDT   007346R     017 N.LEN = 000006          SEC.2 = 000012          ST.XLT  000014      006 T.TRAN= 000000 G
MOVTDC  007454RG    017 N.LGT = 000012 G        SEG1  = 000000 G        SU.DBU= 000004          T.TYPW= 000004 G
MRG     006026R     017 N.NEXT= 000022          SEG2  = 000002 G        SU.DON= 000006          VALND   007360R     017
MSGOUT= ****** GX       N.NODE= 177776          SEG3  = 000004 G        SU.IDL= 000000          VDC     001612R     017
MSG1    001300R     015 N.PKSZ= 000020          SEQSTA  003662R     017 SU.LOD= 000001          VDC.1   001640R     017
MSG2    001304R     015 N.PKTS= 000043          SEQTMP  004740R     017 SU.SRC= 000002          VEC1MX= 000375 G
MSG3    001310R     015 N.POS = 000010 G        SNDBUF  001050R     015 SU.SRR= 000005          VEC2MX= 000375 G
MSG4    001314R     015 N.QURY= 000031          SR.ARE  000114      002 SU.XPD= 000003          VEC3MX= 000125 G
MSG5    001320R     015 N.STAT= 000020          SR.ARS  000106      002 SWAPNP  005406R     017 VI      000014RG    022
MSG6    001324R     015 N.SUNT= 000002          SR.DAY  000010      002 S.BFHD= 000020          VIBCUT= 000036 G
MSK.11  001502R     017 N.TOT = 000122          SR.DLT  000014      002 S.DABA= 000006          VILGT   000012R     022
MSK.12  001610R     017 N.UNIT= 000034          SR.ECB  000047      002 S.DAEF= 000010          VI0MSZ= 000400 G
N       000002          N.VEC = 000014 G        SR.ECH  000046      002 S.DATN= 000002          VI1MSZ= 000400 G
NCITC   000020R     021 OPNTDC  001120R     017 SR.ECL  000050      002 S.FATT= 000016          VI2MSZ= 000400 G
NCITN   000004R     021 PARBUF  000004R     015 SR.FIB  000012      002 S.FDB = 000140          VI3MSZ= 000400 G
NCIT1   020070R     021 PAR$$$= 000027          SR.GRE  000100      002 S.FNAM= 000006          VLDC    000032R     021
NCIT2   030230R     021 PAR1    001564R     015 SR.GRS  000072      002 S.FNB = 000036          VMASK   000000RG    022
NDBCLS= 000006 G        PAR2    001566RG    015 SR.LEN  000122      002 S.FNBW= 000017          VVEC    000030RG    015
NEWSTA  005152R     017 PSTKMX= 000024 G        SR.LIN  000066      002 S.FNTY= 000004          WN.NTP  000004      012
NIBMSK  000016R     015 QE.RQ1= 000144          SR.LIP  000062      002 S.FTYP= 000002          WN.NXT  000006      012
NIBPOS  000020R     015 QNDCNT= 001000 G        SR.MON  000006      002 S.HRL = 000240          WN.ROT  000002      012
NIXCNT= 000020 G        QRYSZ = 002000 G        SR.NDC  000042      002 S.NFEN= 000020          WN.SIZ  000010      012
NNMASK= 177760 G        QTS     001040R     015 SR.NDS  000036      002 S1BIT = 001000 G        WN.SRC  000000      012
NODERR  006706RG    017 QTI.1   000424R     017 SR.NIN  000030      002 S2BIT = 002000 G        WN.TYP  000001      012
NPECNT= 000144 G        QTI.2   000460R     017 SR.NIP  000022      002 S3BIT = 004000 G        WORD0 = 000000
NPEVSZ= 000043 G        QTI.3   000656R     017 SR.SDB  000032      002 TAEBIT= 020000 G        WORD1 = 000002
NPOOL   000056R     021 QTI.4   000576R     017 SR.SRC  000002      002 TBIT  = 004000 G        WORD2 = 000004
NPOOLE= 020066R     021 QTI.5   000550R     017 SR.SUN  000000      002 TDABLK= 000004 G        WORD3 = 000006
NP1     020226R     021 QTI.6   000554R     017 SR.TWS  000056      002 TDABMX= 007764 G        WORD4 = 000010
NP1E  = 030226R     021 QTI.7   001046R     017 SR.WSL  000052      002 TDAMAD= 007760 G        WORD5 = 000012
NP1MSZ= 010000 G        Q.FDSC  000004      007 SR.YR   000004      002 TDBBLK= 000003 G        WORD6 = 000014
NP2     030366R     021 Q.NQBK  000000      007 SR.1IN  000024      002 TDBCLS= 000005 G        WORD7 = 000016
NP2E  = 040366R     021 Q.NUHL  000002          SR.1IP  000016      002 TDBMAD= 007775 G        WORD8 = 000020
NP2MSZ= 036000 G        Q.SIZE  000014      007 SSBIT = 004000 G        TDBOSZ= 000074 G        WORD9 = 000022
NP2OSZ= 000153 G        RECBUF  001044R     015 SS.FID  000002      004 TDCADR  001616RG    015 WRDVAL= 000024
NP3MSZ= 000524 G        REMND   006772R     017 SS.FNB  000010      004 TDCBLK= 000010 G        WRTTDC  006506RG    017
NP3OSZ= 000125 G        REMND2  007026R     017 SS.FVR  000006      004 TDCBUF  001620R     015 XBATCH= 000013
NSCAN   005542R     017 REMND3  007030R     017 SS.LEN  000012      004 TDCEND  021570R     015 XDBLOA= 000004
NXTADD  001610RG    015 R.SUTN= 000002          SS.STT  000000      004 TDCERR  006656R     017 XDBPRO= 000012
NXTOFF  001612RG    015 R.VAR = ****** GX       START   000000R     017 TDCLGT  001622R     015 XDMCIN= 000006
N.BACK= 000002 G        R.VDBA= 000006          STPBLK= 000524 G        TDCMAX  001614RG    015 XFOSMR= 000007
N.BFAC= 000004          R.VDTN= 000002          ST$CNG= 120000 G        TDCMSZ= 004000 G        XGTSRE= 000014
N.BHGH= 000006          R10     000022R     015 ST$INR= 060000 G        TDCT    001630RG    015 XHITSK= 000011
N.BTCH= 000004          R11     000024R     015 ST$INX= 040000 G        TDCT1 = 000030          XHLMER= 000002
N.BUFB= 004000          R12     000026R     015 ST$JSQ= 100000 G        TIC.1 = 000026          XHOTSK= 000010
N.BUFW= 002000          SCH     003620R     017 ST$MAT= 160000 G        TIC.2 = 000014          XMSCHE= 000000
N.DID = 000024          SCH.10  003346R     017 ST$SEQ= 140000 G        TRMCUT= 000040 G        XQTS  = 000003
N.DVNM= 000032          SCH.11  003410R     017 ST.ASZ  000020      006 TSTKMX= 000400 G        XQT0  = 000001
N.ELSE= 000006 G        SCH.13  003352R     017 ST.BSZ  000024      006 TXTBIT= 010000 G        XSULOA= 000005
N.FID = 000000          SCH.5   003060R     017 ST.BTC  000000      006 T.JSBY= 000000 G        ZEROFN  007246R     017
N.FNAM= 000006          SCH.6   003114R     017 ST.CSZ  000030      006 T.MATC= 000000 G        $DSW  = ****** GX
N.FOS = 000764          SD.FSA= 000010 G    003 ST.HRL  000010      006 T.NBAS= 000002 G        $$$   = 000022R     020
N.FSA = 000004 G        SD.NPS= 000016 G    003 ST.LEN  000044      006 T.NDEF= 000000 G        $$$OST= 000006
```

```
FSA-A TRANSLATOR (QT1)  MACRO M1110  27-MAR-80 13:07  PAGE 53-3
SYMBOL TABLE

$$$T1 = 000003          .FSRCB= ****** G        .WAIT = ****** G        ...PC1= 001102R     015 ...PC3= 001102R     015
.CLOSE= ****** G        .OPEN = ****** G        .WRITE= ****** G        ...PC2= 001276R     015 ...TPC= 000020
.FINIT= ****** G        .OPFNB= ****** G        .XQIO = ****** GX

. ABS.  000000     000
        000000     001
SRCOFF  000122     002
FDSCOF  000010     003
SUSOFF  000012     004
DHROFF  000012     005
STTOFF  000044     006
QSPLOF  000014     007
BSTOFF  000772     010
FNOFFS  000044     011
WNODOF  000010     012
DNODOF  000010     013
TRNOFF  000240     014
DATA    021570     015
$$FSR1  000000     016
CODE    007562     017
$DPB$$  000030     020
NODES   040366     021
EMA     035010     022
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  8988 WORDS  (36 PAGES)
DYNAMIC MEMORY:  10196 WORDS  (39 PAGES)
ELAPSED TIME:  00:02:13
QT1,QT1/-SP/NL:ME:8EX=[20,1]P,M,T,QTSIZE,QT1
```

```
                                ; QTSIZE.MAC    "QUERY TRANSLATORS BUFFER SIZE CONFIGUATION FILE"
                                ;       THIS FILE CONTROLS THE SIZE OF ALL BUFFERS THAT CAN VARY
                                ;       IN SIZE DUE TO THE AMOUNT OF QUERIES OR OTHER SUCH PARAMETERS
                                ;       THAT WOULD BE CHARACTERISTIC OF A SITE.  THESE BUFFERS ARE
                                ;       IN QT0, QT1, QT2, OR QT3.
                                ;
                                ;---QT3---BUFFERS-----------------------------------------------------------
                                ;
000125                          VEC3MX==85.                             ;MAXIMUM # CWP'S IN BATCH
000025                          AVELGT==3*2*7/2                         ;AVERAGE CWP SIZE IN TDCT (BYTES)
000400                          VI3MSZ==VEC3MX+2/256.+1*256.            ;SIZE OF VI (NEAREST BLOCK IN BYTES)
001000                          EMCMSZ==VEC3MX/51.+1*256.               ;SIZE OF EMC (NEAREST BLOCK)
004000                          TDCMSZ==VEC3MX*AVELGT+8./N.BUFW+1*N.BUFW;SIZE OF TDCT
000010                          TDCBLK==TDCMSZ/256.                     ;VIRTUAL BLOCK SIZE OF TDCT
000524                          NP3MSZ==VEC3MX*4                        ;SIZE OF NODE POOL
000125                          NP3OSZ==VEC3MX                          ;SIZE OF NODE POOL OVERFLOW AREA
                                ;
                                ;-----QT2----BUFFERS-------------------------------------------------------
                                ;
000375                          VEC2MX==253.                            ;MAXIMUM # VECTORS
000400                          VI2MSZ==VEC2MX+2/256.+1*256.            ;SIZE OF VI
004400                          EMBMSZ==9.*256.                         ;SIZE OF EMB
000003                          TDBBLK==3                               ;# BLOCKS (N.BUFW) IN TDCT BUFFER
000074                          TDBOSZ==3*20.                           ;SIZE OF TDCT BUFFER OVERFLOW
007775                          TDBMAD==4093.                           ;MAXIMUM ADDRESS (48 BITS) IN TDCT
000020                          NIXCNT==16.                             ;# INDEX POINTERS FOR NORMAL NODES
177760                          NNMASK==177760                          ;MASK FOR NORMAL INDEX
000010                          EIXCNT==8.                              ;# INDEX PTRS FOR ELSE NODES
177761                          ELSMSK==177761                          ;MASK FOR ELSE INDEX
000004                          ENCNT==4                                ;# ENTRIES TO USE FOR HASH
036000                          NP2MSZ==256.*60.                        ;SIZE OF NODE POOL
000153                          NP2OSZ==7+100.                          ;SIZE OF NODE POOL OVERFLOW AREAR
                                ;
                                ;------QT1-----BUFFERS------------------------------------------------------
                                ;
000375                          VEC1MX==253.                            ;MAX. # TERMS IN FSA-A
000400                          VI1MSZ==VEC2MX+2/256.+1*256.            ;SIZE OF VI
000004                          TDABLK==4                               ;# BLOCKS (N.BUFW) IN TDCT BUFFER
007764                          TDABMX==<TDABLK*<N.BUFW-4>>+4
016000                          EMAMSZ==7.*4*256.                       ;SIZE OF EMA
007760                          TDAMAD==340.*12.                        ;MAX. ADDRESS IN TDCT
010000                          NP1MSZ==4096.                           ;SIZE OF NORMAL NODE POOL
000144                          NPECNT==100.                            ;# NODES IN ELSE POOL
000043                          NPEVSZ==35.                             ;# VECTORS IN ELSE NODES
                                ;
                                ;-----FLU---MODIFIER----BUFFERS---------------------------------------------
                                ;
002260                          FMNPSZ==1200.                           ;SIZE OF NORMAL FLU MOD NODE POOL
000400                          FMEPSZ==256.                            ;SIZE OF ELSE FLU MOD NODE POOL
                                ;
                                ;------QT0------BUFFERS-------------------------------------------------
                                ;
002000                          QRYSZ==N.BUFW                           ;SIZE OF QUERY BUFFER
000040                          FLIXSZ==32.                             ;# INDEX PTRS FOR TERMS
177740                          FLIXMK==177740                          ;MASK FOR TERM INDEX
000010                          CPIXSZ==8.                              ;# INDEX PTRS FOR CWP'S
177770                          CPIXMK==177770                          ;MASK FOR CWP INDEX
```

```
                                FMIXSZ==1                               ;#·INDEX·PTRS·FOR·FLU-MOD'S·
                                FNPSZ==3036.                            ;FLU·NODE·POOL·SIZE·
                                FNPOSZ==256.                            ;SIZE·OF·POOL·OVERFLOW·AREA·
                                TSTKMX==256.                            ;MAX. #·TOKENS·TO·BE PUSHED·
                                ASTKMX==256.                            ;MAX. #·ARGUMENTS·TO BE·PUSHED
                                PSTKMX==20.                             ;MAX. #·PROX·NODES·IN·A·CHAIN·
                                HSTKMX==64.                             ;MAX. NESTING·OF·QUERY·
                                QNDCNT==512.                            ;#·NODES·IN·LOGIC·POOL·
                                VI0MSZ==VI1MSZ·                         ;ASSUME·MAX·VI·IS·IN QT1
                                .IIF·LT,VI0MSZ-VI2MSZ,VI0MSZ==VI2MSZ·   ; IF·NOT, RESET·QT0'S·TO·QT2'S VI·SIZE·
                                EMXMSZ==EMAMSZ·                         ;ASSUME·MAX·EMX·IS·IT·QT1
                                .IIF·LT,EMXMSZ-EMBMSZ,EMXMSZ==EMBMSZ·   ; IF·NOT, RESET·TO·QT2'S·
                                ;
                                ;-----ERROR·AND·CLOSE·CODES---------------------------------------
                                ;
                                TRMCUT==32.     ;TERM·CUT-OFF·
                                FLUCUT==14.     ;FLU CUT-OFF·
                                EMACUT==TRMCUT*30.      ;EMA·CUT-OFF·
                                VIBCUT==30.             ;VIB·CUT-OFF·
                                EMBCUT==VIBCUT*20.      ;EMB·CUT-OFF·
                                ;
                                FLUCLS==1       ;CLOSED·DUE·TO FLU·COUNT
                                LOGCLS==2·      ;CLOSED·DUE·TO LOGIC·COUNT·(QLB,SDLB,QEX)
                                EMACLS==3       ;CLOSED·DUE·TO EMA·SIZE·
                                EMBCLS==4       ;CLOSED·DUE·TO EMB·SIZE·
                                TDBCLS==5       ;CLOSED·DUE·TO TDCTB·SIZE·
                                NDBCLS==6       ;CLOSED·DUE·TO QT2·NODE·POOL·SIZE·
                                ELSCLS==7       ;CLOSED·DUE·TO OTHER·CONDITIONS·(FLU·POOL,ETC)
                                ;
```

```
                                        .TITLE  FLU-MOD CODE
                                        .MCALL  MOUT$S
                                ;   TRACE MACRO FOR TROUBLE SHOOTING
                                ;
                                .MACRO  TRCTST,?X
                                        CMPB    #1,-1(R1)
                                        BNE     X
                                        INC     TRCFLG
                                X:
                                .ENDM
000000  000000                  TRCFLG: .WORD   0
000002  000046  000006'         MSG12:  .WORD   LN12E-LN12,LN12
000006      126     115     101 LN12:   .ASCIZ  /VMASK=%10,%10,%10,%10,%10,%10,%10,%10/
000054                          LN12E:
                                        .EVEN
                                ;
                                ;   THIS IS THE DATA AREA FOR FLU-MODIFIER CODE.
                                ;   EQUATES
                                ;
        000010                  MCTYP=10        ;FLU-MOD TYPE AT END OF SCAN
                                ;
000000                                  .PSECT  FMDATA
                                ;   POOLS
                                ;
000000                          FMNPS:  .BLKW   FMNPSZ  ;NORMAL NODE POOL (START)
004540                          FMNPE:                  ;               (END)
004540                          FMEPS:  .BLKW   FMEPSZ  ;ELSE NODE POOL (START)
005540                          FMEPE:                  ;              (END)
                                ;
                                ;   MULTI-FLU-MOD VARIABLES
                                ;
005540                          MIXFMC::.BLKW   40.     ;VEC. STORAGE FOR FMTST1 & FMTST2
005660  000000                  FMEXF2: .WORD   0       ;DIFFERENT FLU NODES
005662  005664'                 VALVPT: .WORD   VALVEC  ;PTR TO NEXT VECTOR SAVED LOCATION
005664                          VALVEC: .BLKW   20.     ;SAVE AREA FOR VECTORS
005734  000000                  LEVEL:  .WORD   0       ;LEVEL OF NESTING OF EXP. VECTORS
                                ;
                                ;   CONVERSION TABLES
                                ;
005736  000000G 000000G 000000G FMTYP:  .WORD   EMXDTF,EMXDTF,EMXZNF,EMXZNF,EMXSZF,EMXSZF,EMXMCF,EMXMCF
005756  002000  002000  001000  FMTSC:  .WORD   BIT10,BIT10,BIT9,BIT9,BIT10!BIT9,BIT10!BIT9,0,0
                                ;
                                ;   POINTERS
                                ;
005776  000000                  CFMN:   .WORD   0       ;CURRENT FLU-MOD NODE
006000  000000                  NFMN:   .WORD   0       ;NEXT FLU-MOD NODE
006002  000000                  FMNI:   .WORD   0       ;FLU-MOD NODE INDEX
006004  000000                  CFME:   .WORD   0       ;CURRENT ELSE NODE
006006  000000                  NFME:   .WORD   0       ;NEXT ELSE NODE
006010  000000                  FMEI:   .WORD   0       ;ELSE NODE INDEX
                                ;
                                ;   STORAGE WORDS
                                ;
006012  000000                  CFSA:   .WORD   0       ;CURRENT FSA LOC
006014  000000                  CFMT:   .WORD   0       ;CURRENT FMT FOR 'GETNDE'
006016  000000                  FMT:    .WORD   0       ;FLU-MOD TYPE FOR 'GVMASK'
006020  000000                  FMTSUM: .WORD   0       ;FMT SUMMARY FOR 'IMASK' & 'FMSEQ'
```

```
     58 006022  000000                    FMSC:   .WORD   0       ;SOURCE CODE FOR FMT
     59 006024  000000                    FMEXL:  .WORD   0       ;EMA POINTER FOR FLU-MOD VECTOR STARTS
     60 006026  000000                    FMECT:  .WORD   0       ;COUNT OF FLU-MOD VECTOR STARTS
     61 006030  000000                    GVSUB:  .WORD   0       ;'GVMASK' SUB-SUBROUTINE POINTER
     62 006032  000000                    OLDMC:  .WORD   0       ;OLD MATCH CODE
     63 006034                            FMPARI: .BLKW   4       ;FLU-MOD ERROR STATISTICS
     64                                   ;
     65                                   ;   FLAG WORDS
     66                                   ;
     67 006044  000000                    FMLOG:  .WORD   0       ;FLU-MOD SCAN LOGIC ENABLE
     68 006046  000000                    FMEXF:  .WORD   0       ;FLU-MOD TYPE (-2,-1,0,1=MC,MS,S,C)
     69                                                           ;M=MULTIPLE S=SIMPLE C=COMPLEX
     70 006050  000000                    FMIN:   .WORD   0       ;ELSE VECTOR
     71 006052  000000                    FVAL:   .WORD   0       ;NORMAL VECTOR
     72 006054  000000                    FMAT:   .WORD   0       ;MATCH VECTOR
     73 006056  000000                    ENFLAG: .WORD   0       ;ELSE SEARCH ENABLE 'GETNDE'
     74 006060  000000                    NEST:   .WORD   0       ;NESTED INPUT TO GVMASK SUB-SUBROUTINE FLAG
     75                                   ;
     76 006062  000053  006072'           MSG10:  .WORD   LN7E-LN7,LN7     ;ERROR MESSAGE
     77 006066  000046  006145'           MSG11:  .WORD   LN11E-LN11,LN11
     78 006072      106     114     125   LN7:    .ASCIZ  /FLU-MOD NODE POOL OVERFLOW %1D,%1D,%1D,%1D/
     79 006145                            LN7E:
     80 006145      106     114     125   LN11:   .ASCIZ  /FLU-MOD VALVEC OR MIXFMC OVERFLOW %1D/
     81 006213                            LN11E:
     82                                           .EVEN
```

```
                        ;
                        .MACRO  GETNXT  A,?B,?C,?D
                                MOV     NXTOFF,A                ;REG = OFFSET OF NEXT FREE TDCT STATE
                                CMP     A,TDCMAX                ;TDCT BUFFER OVERFLOW?
                                BLT     B                       ;BRANCH IF NO
                                CMP     NXTADD,#TDAMAD          ;TD MEMORY OVERFLOW?
                                BLT     C                       ;BRANCH IF NO
                                JSR     PC,ERROR2               ;MEMORY OVERFLOW
                        C:      CMP     LOWOFF,#<N,BUFB-10>     ;ARE ALL STATES IN BLOCK 1 COMPLETE?
                                BGE     D                       ;BRANCH IF YES
                                JSR     PC,ERROR2               ;TDCT BUFFER NOT BIG ENOUGH
                        D:      MOV     R0,-(SP)
                                JSR     PC,WRTTDC               ;WRITE NEXT BLOCK OF TDCT
                                MOV     (SP)+,R0
                                JSR     PC,MOVTDC               ;COMPRESS TDCT BUFFER
                                MOV     NXTOFF,A                ;RESET NEXT FREE TDCT STATE
                        B:
                        .ENDM
```

```
103 000000                              .PSECT··FMCODE···;
104                                     ;   THIS·SUBROUTINE·IS·THE·MAIN·CONTROL·SUBROUTINE·FOR·
105                                     ;   FLU-MOD·PROCESSING.·WHEN·IT·IS·CALLED,·BUFFER·VVEC·
106                                     ;   CONTAINS·THE·VECTOR·FOR·THE·INTERWORD·CHARACTER·WHICH·
107                                     ;   DESIGNATES·THE·END·OF·THE·FLU·AND·THE·START·OF·THE·
108                                     ;   THE·FLU-MODIFIER.·NOTE:·REGISTERS·HAVE·NO·MEANING·
109                                     ;   AT·THE·TIME·FLUMD·IS·CALLED.
110                                     ;
111 000000  012703  000000G·        FLUMD::  MOV·    #VVEC,R3                ;VVEC·CONTAINS·VECTOR·
112·000004  016300  000000G·                 MOV·    N.LGT(R3),R0            ;GET·NODE·SIZE·
113 000010  042700  100000                   BIC·    #100000,R0
114 000014  020027  000001                   CMP·    R0,#1                   ;ONLY·1·VECTOR?·
115 000020  001435                           BEQ·    100$                    ;YES·
116 000022· 012767  177777  006046'         MOV·    #-1,FMEXF·              ;NO:·FLAG·AS·DIFFERENT·MATCH·CODES·
117 000030  005267  005660'                  INC·    FMEXF2·         ;FLAG·AS·MULTIPLE·NODES·
118 000034  010067  006034'                  MOV·    R0,FMPAR1       ;STORE·#·TERMS·
119 000040  020027  000047                   CMP·    R0,#39.         ;DON'T·LET·OVERFLOW·
120 000044  101402·                          BLOS·   41$
121 000046  000167  005614                   JMP·    VVOVF·
122·000052· 010067  005540'          41$:    MOV·    R0,MIXFMC·      ;SAVE·COUNT·AND·LOCATION·
123 000056  006367  005540'                  ASL·    MIXFMC·         ;ADJUST·COUNT·AS·IF·EXP.·VECTOR·
124 000062· 012767  005540' 006024'          MOV·    #MIXFMC,FMEXL·  ;LOCATION·
125 000070                                   SAVE·   R0
126 000072· 012702· 005542'                  MOV·    #MIXFMC+2,R2·   ;SET·UP·TO·SAVE·VECTOR·
127 000076  010305                           MOV·    R3,R5
128·000100  062705  000000G·                 ADD·    #N.VEC,R5
129 000104  012522·                  40$:    MOV·    (R5)+,(R2)+     ;SAVE·VECTOR·
130 000106  077002·                          SOB·    R0,40$
131 000110                                   RESTOR· R0
132·000112· 000407                           BR      101$
133 000114  005067  006046'          100$:   CLR·    FMEXF·                  ;FLAG·AS·SINGLE·MATCH·CODE·
134 000120  005067  005660'                  CLR·    FMEXF2·
135 000124  012767  000001  006034'         MOV·    #1,FMPAR1
136 000132· 052763  000000G·000000G· 101$:   BIS·    #BITNIB,N.LGT(R3)       ;FORCE·VECTOR·STEPPING·
137 000140  005367  006044'                  DEC·    FMLOG·                  ;TRIGGER·FLU-MOD·PROCESSING·
138 000144  005063  000000G·                 CLR·    N.POS(R3)               ;FLU-MOD·TYPE·=·DOCUMENT/NIB1
139 000150  016763  000000G·000000G·         MOV·    NXTADD,N.ELSE(R3)       ;STORE·TDCT·ADDRESS·
140 000156  005267  000000G·                 INC·    NXTADD·                 ;ALLOCATE·ADDRESS·
141 000162·                                  GETNXT· R4
142·000244  062767  000006  0000000G·        ADD·    #6,NXTOFF·
143 000252· 012767  000000G·006016'          MOV·    #EMXDTF,FMT·    ;SCAN·FOR·DOC-TYPE
144 000260  005067  006010'                  CLR·    FMEI            ;RESET·INDEX·POINTERS·
145 000264  005067  006002'                  CLR·    FMNI
146 000270  005067  006014'                  CLR·    CFMT            ;SCAN·FOR·DOC-TYPE
147 000274  012767  004540' 006004'         MOV·    #FMEPS,CFME·    ;RESET·POOL·POINTERS·
148 000302· 012767  004540' 006006'         MOV·    #FMEPS,NFME·
149 000310  012767  000000' 005776'         MOV·    #FMNPS,CFMN·
150 000316  012767  000000' 006000'         MOV·    #FMNPS,NFMN·
151 000324  012767  002000  006022'         MOV·    #BIT10,FMSC·    ;SOURCE·CODE=DOC·TYPE·
152·000332· 004767  000020           1$:     JSR·    PC,GVMASK·      ;STEP·VECTOR·
153 000336  004767  000712                   JSR·    PC,IMASK·       ;SPLIT·VECTOR·
154 000342· 004767  005652                   JSR·    PC,GETNDE·      ;GET·NEXT·VECTOR·
155 000346  103371                           BCC·    1$              ;IF·MORE·VECTORS·
156 000350  005067  006044'                  CLR·    FMLOG·          ;SWITCH·BACK·TO·TEXT·CHARS·
157 000354  000207                           RTS·    PC·             ;IF·NO·MORE·VECTORS·
```

```
                                        ;
                                        ;   GVMASK IS THE SUBROUTINE THAT GIVEN A VECTOR FOR THE
                                        ;   CURRENT NIBBLE POSITION, GENERATES A VECTOR FOR THE
                                        ;   NEXT NIBBLE POSITION. GVMASK HANDLES TEXT CHARACTERS
                                        ;   AS WELL AS FLU-MOD TERMS.
                                        ;       INPUT                   OUTPUT
                                        ;   R3= NODE ADR OF GIVEN       ADR OF GENERATED VECTOR
                                        ;       VECTOR                  (VMASK)
                                        ;
                                        ;   NO OTHER REGISTERS HAVE MEANING FOR I/O
                                        ;
000356                                  GVMASK::SAVE    R0
000360  010301                                 MOV     R3,R1           ;INITIALIZE REGISTERS
000362  010305                                 MOV     R3,R5
000364  012703  000000C                        MOV     #VMASK+N.VEC,R3
000370  016100  000000G                        MOV     N.LGT(R1),R0
000374  042700  000000G                        BIC     #BITNIB,R0
000400  062701  000000G                        ADD     #N.VEC,R1
000404  005002                                 CLR     R2
000406  005067  006060'                       CLR     NEST
000412                                  1$:     SAVE    R5
000414  032765  100000  000000G                BIT     #BIT15,N.POS(R5)        ;IS ELSE OVERIDE SET?
000422                                         BEQ     1$                      ;YES
000424  032765  000000G 000000G                BIT     #BITNIB,N.LGT(R5)       ;IS IT SECOND NIBBLE?
000432                                         BEQ     1$                      ;YES
000434  012767  000552' 006030'               MOV     #GV1,GVSUB      ;SIMPLE NIB1 TO NIB2 ADVANCE
000442  000420                                 BR      3$
000444  005767  006044'                       TST     FMLOG           ;PROCESSIG FLU-MOD'S?
000450  001006                                 BNE     2$              ;YES
000452  012767  000572' 006030'               MOV     #GV2,GVSUB      ;STEP VECTORS FOR TEXT CHAR
000460  005067  000000C                        CLR     VMASK+N.LGT
000464  000407                                 BR      3$
000466  012767  000656' 006030'       2$:     MOV     #GV3,GVSUB      ;STEP VECTORS FOR FLU-MOD
000474  005067  006020'                       CLR     FMTSUM          ;INITIALIZE VARIABLES
000500  005067  000000C                        CLR     VMASK+N.LGT
000504  004777  006030'               3$:     JSR     PC,@GVSUB       ;EXECUTE THE SELECTED SUB
000510                                         RESTOR  R5
000512  012703  000000G                        MOV     #VMASK,R3
000516  050263  000000G                        BIS     R2,N.LGT(R3)    ;ADD VECTOR LENGTH TO NIB FLAG
000522  005063  000000G                        CLR     N.POS(R3)
000526  116563  000000G 000000G                MOVB    N.POS(R5),N.POS(R3)     ;MAINTAIN FMT CODE
000534  016563  000000G 000000G                MOV     N.ELSE(R5),N.ELSE(R3)   ; & ELSE STATE OR ADR
000542  005063  000000G                        CLR     N.FSA(R3)
000546                                         RESTOR  R0
000550  000207                          4$:    RTS     PC
```

```
                                        ;
                                        ;   SUBROUTINE GV1 HANDLES SIMPLE NIBBLE1 TO NIBBLE2 ADVANCE
                                        ;   IN THIS CASE, VECTORS ARE IDENTICAL.
                                        ;
000552  016502  000000G          GV1:    MOV     N.LGT(R5),R2    ;VECTOR LENGTH SAME AS BEFORE
000556  012123                   1$:     MOV     (R1)+,(R3)+     ;SO ARE VECTORS
000560  077002                           SOB     R0,1$
000562  012767  000000G 000000C          MOV     #BITNIB,VMASK+N.LGT     ;SET NIB2 FLAG
000570  000207                           RTS     PC
                                        ;
                                        ;   SUBROUTINE GV2 HANDLES STEPPING VECTORS FOR TEXT CHAR'S.
                                        ;
000572  012104                   GV2:    MOV     (R1)+,R4        ;GET EMA POINTER
000574  032764  000000G 000000G          BIT     #EMXVDC,EMA(R4) ;IS EMA ENTRY A VLDC?
000602                                   BOFF    1$              ;NO
000604  010423                           MOV     R4,(R3)+        ;XFER VLDC & NEXT ENTRY
000606  062704  000006                   ADD     #6,R4
000612  010423                           MOV     R4,(R3)+
000614  062702  000002                   ADD     #2,R2           ;2 MORE ENTRIES IN VECTOR
000620  000414                           BR      2$
000622  062704  000006           1$:     ADD     #6,R4           ;XFER NEXT ENTRY
000626  010423                           MOV     R4,(R3)+
000630  005202                           INC     R2              ;1 MORE ENTRY IN VECTOR
000632  032764  000000G 000000G          BIT     #EMXVDC,EMA(R4) ;IS EMA ENTRY A VLDC?
000640                                   BOFF    2$              ;NO
000642  062704  000006                   ADD     #6,R4           ;YES:XFER NEXT ENTRY ALSO
000646  010423                           MOV     R4,(R3)+
000650  005202                           INC     R2              ;1 MORE ENTRY IN VECTOR
000652  077031                   2$:     SOB     R0,GV2          ;DO FOR ALL ENTRIES
000654  000207                           RTS     PC
                                        ;
                                        ;   SUBROUTINE GV3 HANDLES STEPPING OF VECTOR FOR FLU-MOD'S.
                                        ;
000656  012104                   GV3:    MOV     (R1)+,R4        ;GET EMA POINTER
000660  100406                           BMI     1$              ;IF NEGATIVE MAKE POSITIVE
000662  005767  006060'                  TST     NEST            ;IF NESTED,DON'T STEP VECTOR
000666  001004                           BNE     2$              ;NESTED
000670  062704  000006                   ADD     #6,R4           ;IF POS & NOT NESTED, STEP ENTRY
000674  000401                           BR      2$
000676  005404                   1$:     NEG     R4
000700  016405  000000G          2$:     MOV     EMA(R4),R5      ;GET ENTRY
000704  100411                           BMI     3$              ;IF NON-SEQ FLOW...
000706  036705  006016'                  BIT     FMT,R5          ;ENTRY OF TYPE DESIRED?
000712                                   BON     4$              ;YES
000714  005404                           NEG     R4              ;NO: NEGATE
000716  050567  006020'           4$:     BIS     R5,FMTSUM       ;NOTE TYPE IN SUMMARY
000722  010423                           MOV     R4,(R3)+        ;XFER ENTRY
000724  005202                           INC     R2              ;1 MORE ENTRY IN VECTOR
000726  000433                           BR      5$
000730  032705  040000           3$:     BIT     #BIT14,R5       ;MULTIPLE FLU-MOD'S
000734                                   BON     31$             ;YES
000736  005767  006046'                  TST     FMEXF           ;NO:HAD IT OCCURRED BEFORE?
000742  100004                           BPL     32$             ;NO
000744  012767  177776  006046'         MOV     #-2,FMEXF       ;FLAG AS UNKNOWN
000752  000417                           BR      33$
000754  012767  000001  006046' 32$:     MOV     #1,FMEXF        ;FLAG AS SAME MATCH CODE
000762  000413                           BR      33$
```

```
262 000764  005767  005660'          31$:    TST     FMEXF2
263 000770  100414                           BMI     34$             ;IF DIFF FLU NODES
264 000772  012767  177777  006046'         MOV     #-1,FMEXF       ;FLAG AS ALWAYS DIFFER
265 001000  010467  006024'                  MOV     R4,FMEXL        ;SAVE EXPANSION VECTOR LOCATION
266 001004  062767  000000G 006024'         ADD     #EMA,FMEXL
267 001012  004767  000062           33$:    JSR     PC,NSQSUB       ;EXPAND VECTOR
268 001016  077061                   5$:     SOB     R0,GV3          ;DO FOR ALL ENTRIES
269 001020  000207                           RTS     PC
270 001022                           34$:    SAVE    R4,R0,R3        ;IF DIFFERENT FLU-MOD NODES
271 001030  016703  006024'                  MOV     FMEXL,R3
272 001034  012300                           MOV     (R3)+,R0        ;GET COUNT OF VECTORS
273 001036  006200                           ASR     R0
274 001040  020423                   300$:   CMP     R4,(R3)+        ;FIND ENTRY FOR THIS POINTER
275 001042  103404                           BLO     301$            ;FOUND
276 001044  077003                           SOB     R0,300$
277 001046  162703  000002                   SUB     #2,R3           ;FOUND: PRIOR ENTRY
278 001052  000402                          BR      302$
279 001054  162703  000004           301$:   SUB     #4,R3           ;PRIOR PRIOR ENTRY
280 001060  062704  000000G          302$:   ADD     #EMA,R4         ;CONVERT OFFSET TO LOCATION
281 001064  005404                           NEG     R4              ;FLAG AND RE-ENTER IT
282 001066  010413                           MOV     R4,(R3)
283 001070                                   RESTOR  R4,R0,R3
284 001076  000745                           BR      33$
```

```
                                        ;
                                        ;   SUBROUTINE·IMASK IS·RESPONSIBLE·FOR·THE·MAJOR·PORTION·OF
                                        ;   FLU-MOD·PROCESSING. IT·SPLITS·A·VECTOR·INTO·SUCCESIVE·VECTORS·
                                        ;   IF·MULTIPLE·PATHS·EXIST·FROM·THAT·STATE. IT·ALSO·CREATES THE·
                                        ;   TDCT·STATES·FOR·THE·VECTORS.
                                        ;
                                        ;   ONLY·R3 IS·USED·FOR·I/O. AS·INPUT·IT POINTS·TO THE·VECTOR·TO·
                                        ;   BE·SPLIT·WHICH· IMASK·ALSO·KNOWS·IS·VMASK. NO·REGISTERS·HAVE·
                                        ;   MEANING·FOR·OUTPUT.
                                        ;
001254  016367  000000G·006012' IMASK:  MOV·    N.ELSE(R3),CFSA·;STORE·FSA·ADR·FOR NODE·
001262· 016301  000000G·                MOV·    N.LGT(R3),R1    ;R1=VECTOR·SIZE·
001266  062703  000000G·                ADD·    #N.VEC,R3       ;R3=INPUT·VECTOR·ENTRIES·
001272· 012705  000000C·                MOV·    #VVEC+N.VEC,R5  ;R5=OUTPUT·VECTOR·ENTRIES·
001276  005701                          TST·    R1              ;NIB2?·
001300  100012·                        BPL·    10$             ;NO·
001302·                         1$:     SAVE·   R1,R3
001306  016705  006012'          2$:     MOV·    CFSA,R5         ;CONVERT FSA·LOC·TO·TDCT·ADR.
001312· 070527  000006                  MUL·    #6,R5
001316· 066705  000000G·                ADD·    TDCADR,R5
001322· 000167  000526                  JMP·    1000$
001326                          10$:    SAVE·   R1,R3
001332· 005000                          CLR·    R0              ;R0=ENTRIES·IN·ELSE·VECTOR·
001334· 022767  000000G·006016'         CMP·    #EMXMCF,FMT·    ;MATCH·STATE·SCAN?
001342· 001761                          BEQ·    2$              ;YES·
001344  012304                  11$:    MOV·    (R3)+,R4        ;GET·EMA POINTER·
001346  100002·                        BPL·    12$             ;XFER·&·COUNT·ONLY ELSE'S·
001350  010425                          MOV·    R4,(R5)+
001352· 005200                          INC·    R0
001354  077105                  12$:    SOB·    R1,11$
001356  016705  006012'                 MOV·    CFSA,R5         ;CONVERT FSA·LOC·TO·TDCT·ADR.
001362· 070527  000006                  MUL·    #6,R5
001366  066705  000000G·                ADD·    TDCADR,R5
001372· 005700                          TST·    R0              ;ANY·ELSE·VECTOR·
001374  001537                          BEQ·    100$            ;NO·
001376  166705  000000G·                SUB·    TDCADR,R5       ;ALLOW·FOR·BUFFER·SHIFT·
001402·                                 GETNXT· R1              ;ALLOCATE·NEXT·FSA LOC·
001464  066705  000000G·                ADD·    TDCADR,R5
001470  016725  000000G·                MOV·    NXTADD,(R5)+    ;POINT·CHANGE·STATE·TO·IT·
001474  062767  000006  000000G·       ADD·    #6,NXTOFF·
001502· 016767  000000G·000000C·       MOV·    NXTADD,VVEC+N.ELSE·     ;ENTER·NODE'S·FSA·LOC·
001510  005267  000000G·                INC·    NXTADD·
001514  016767  000014' 000000C·       MOV·    CFMT,VVEC+N.POS·;ENTER·FMT·
001522· 062767  000002  000000C·       ADD·    #2,VVEC+N.POS·
001530  010067  000000C·                MOV·    R0,VVEC+N.LGT·  ;ENTER·SIZE·
001534  052767  000000G·000000C·       BIS·    #BITNIB,VVEC+N.LGT·     ;NIBBLE2·
001542· 012703  000010'                 MOV·    #FMEI,R3        ;CHECK·FOR·DUPLICATE·
001546  012702· 000000G·                MOV·    #VVEC,R2·
001552· 004767  002754                  JSR·    PC,SCAN·
001556  016703  000000C·                MOV·    VMASK+N.LGT,R3  ;GET·OLD SIZE·
001562· 042703  000000G·                BIC·    #BITNIB,R3
001566  103445                          BCS·    14$             ;NEW·VECTOR·
001570  004767  003356                  JSR·    PC,FMTST1       ;DO·NON-ELSE·ENTRIES·MATTER?
001574  103424                          BCS·    16$             ;NO·
001576  005367  000000G·                DEC·    NXTADD·         ;OLD·VECTOR: DEALLOCATE·FSA·LOC·
001602· 162767  000006  000000G·       SUB·    #6,NXTOFF·
001610  020003                          CMP·    R0,R3           ;WAS·IT·ELSE·ONLY?
```

```
 366 001612· 001404                                 BEQ·    13$             ;YES·
 367 001614  016465  000000G·177776                 MOV·    N.ELSE(R4),-2(R5)       ;REPOINT·TO·OLD·FSA·LOC·
 368 001622· 000434                                 BR      700$
 369 001624  005065  177776          13$:           CLR·    -2(R5)          ;CREATE·JUMP·STATE TO·FSA·LOC·
 370 001630  016425  000000G·                       MOV·    N.ELSE(R4),(R5)+
 371 001634  012725  000000G·                       MOV·    #ST$JSQ,(R5)+
 372·001640                                         RESTOR· R1,R3
 373 001644  000207                                 RTS·    PC·
 374 001646  016764  006012' 000000G· 16$:         MOV·    CFSA,N.ELSE(R4) ;REUSE·THIS·FSA·LOC·
 375 001654  005367  000000G·                       DEC·    NXTADD·         ;DEALLOCATE·FSA·LOC·
 376 001660  162767  000006  000000G·               SUB·    #6,NXTOFF·
 377 001666                                         RESTOR· R1,R3
 378 001672· 000207                                 RTS·    PC·
 379 001674  016725  000000G·          100$:        MOV·    INTDEF,(R5)+    ;XFER·INTERNAL·DEFAULT·
 380 001700  000405                                 BR      700$
 381 001702· 020003                    14$:         CMP·    R0,R3           ;ELSE·ONLY?·
 382·001704  001760                                 BEQ·    16$             ;YES·
 383 001706  004767  003240                         JSR·    PC,FMTST1       ;DO·NON-ELSE·ENTRIES·MATTER?
 384 001712· 103755                                 BCS·    16$             ;NO·
 385 001714  016725  000000G·          700$:        MOV·    NXTADD,(R5)+    ;INDEX·STATE·FOR·POSITION·
 386 001720  012725  000000C·                       MOV·    #ST$CNG!IWBIT,(R5)+     ;CHANGE·ELSE·STATE·
 387 001724  016767  000000G·006012'               MOV·    NXTADD,CFSA·    ;UPDATE·CURRENT·FSA·LOC·
 388 001732· 005267  000000G·                       INC·    NXTADD·         ;ALLOCATE·FSA·LOC·
 389 001736                                         GETNXT· R5
 390 002020  062705  000000G·                       ADD·    #TDCT,R5
 391 002024  062767  000006  000000G·               ADD·    #6,NXTOFF·
 392·002032· 022766  000001  000002·                CMP·    #1,2(SP)        ;SEQUENCIAL·POSSIBLE?·
 393 002040  001005                                 BNE·    1000$           ;NO·
 394 002042·                                        RESTOR· R1,R3           ;YES·
 395 002046  004767  000720                         JSR·    PC,FMSEQ·       ;COMPLETE·VECTOR·WITH·SEQ·STATES·
 396 002052· 000207                                 RTS·    PC·
 397 002054  022767  000000G·006016' 1000$:         CMP·    #EMXMCF,FMT·    ;MATCH·STATE·SCAN?
 398 002062· 001005                                 BNE·    1001$           ;NO·
 399 002064                                         RESTOR· R1,R3
 400 002070  004767  002204                         JSR·    PC,MATRP2·      ;CREATE·MATCH·STATE(S)
 401 002074  000207                                 RTS·    PC·
 402·002076  011603                    1001$:       MOV·    (SP),R3         ;R3=VECTOR·ENTRIES ADR·
 403 002100  016601  000002                         MOV·    2(SP),R1        ;R1=VECTOR·SIZE·
 404 002104  042701  000000G·                       BIC·    #BITNIB,R1      ;REMOVE·NIBBLE·INDICATOR·
 405 002110                                         SAVE·   R5              ;TDCT·ADR·
 406 002112· 005004                                 CLR·    R4              ;INIT·FOR·SINGLE·EXIT·TEST·
 407 002114  005005                                 CLR·    R5
 408 002116  005000                                 CLR·    R0
 409 002120  012302·                   1002$:       MOV·    (R3)+,R2·       ;SUMMARIZE·BITS·IN VECTOR·
 410 002122· 100405                                 BMI·    1003$           ;EXCEPT·ELSE·ENTRIES·
 411 002124  056204  000002G·                       BIS·    EMA+2(R2),R4
 412·002130  056205  000004G·                       BIS·    EMA+4(R2),R5
 413 002134  005200                                 INC·    R0              ;COUNT·NON-ELSE·ENTRIES·
 414 002136  077110                    1003$:       SOB·    R1,1002$
 415 002140  004767  003546                         JSR·    PC,FMTST2·      ;TEST·FOR·SINGLE·EXIT·CONDITION·
 416 002144  103116                                 BCC·    1100$           ;NOT·SINGLE·EXIT·
 417 002146                                         RESTOR· R1,R3,R0
 418 002154  005701                                 TST·    R1              ;WHICH·NIBBLE?·
 419 002156  100402·                                BMI·    1004$           ;NIBBLE2
 420 002160  010420                                 MOV·    R4,(R0)+        ;XFER·NIBBLE1
 421 002162· 000403                                 BR      1005$
 422·002164  010520                    1004$:       MOV·    R5,(R0)+        ;XFER·NIBBL2·
```

```
 423 002166  042701  000000G                        BIC     #BITNIB,R1      ;REMOVE FLAG
 424 002172  016720  000000G              1005$:    MOV     NXTADD,(R0)+    ;COMPLETE SINGLE EXIT STATE
 425 002176  016767  000000G 000000C               MOV     NXTADD,VVEC+N.ELSE      ;& ENTER NODE FOR NEXT STATE
 426 002204  005267  000000G                        INC     NXTADD
 427 002210  012710  000000C                        MOV     #ST$INX!SSBIT,(R0)
 428 002214  056720  006022'                        BIS     FMSC,(R0)+              ;SOURCE CODE
 429 002220  016767  000000C 000000C               MOV     VMASK+N.POS,VVEC+N.POS  ;FMT
 430 002226  105267  000000C                        INCB    N.POS+VVEC
 431 002232  016767  000000C 000000C               MOV     VMASK+N.LGT,VVEC+N.LGT  ;VECTOR SIZE
 432 002240  012702  000000C                        MOV     #VMASK+N.VEC,R2         ;XFER VECTOR
 433 002244  012703  000000C                        MOV     #VVEC+N.VEC,R3
 434 002250  012223                       1007$:    MOV     (R2)+,(R3)+
 435 002252  077102                               SOB     R1,1007$
 436 002254                                       GETNXT  R5
 437 002336  062705  000000G                        ADD     #TDCT,R5
 438 002342  062767  000006  000000G               ADD     #6,NXTOFF
 439 002350  012703  006002'                        MOV     #FMNI,R3        ;SCAN FOR VECTOR DUPLICATE
 440 002354  012702  000000G                        MOV     #VVEC,R2
 441 002360  004767  002146                         JSR     PC,SCAN
 442 002364  103005                               BCC     1006$           ;NEW VECTOR
 443 002366  005025                               CLR     (R5)+           ;DUPLICATE, JUMP TO IT
 444 002370  016425  000000G                        MOV     N.ELSE(R4),(R5)+
 445 002374  012725  000000G                        MOV     #ST$JSQ,(R5)+
 446 002400  000207                       1006$:  RTS     PC
 447 002402                               1100$:  RESTOR  R2              ;R2=TDCT ADR
 448 002404  011603                               MOV     (SP),R3         ;R3=VECTOR ENTRY ADR
 449 002406  016601  000002                        MOV     2(SP),R1        ;R1=VECTOR SIZE
 450 002412  005701                               TST     R1              ;WHICH NIBBLE?
 451 002414  100401                               BMI     1101$           ;NIBBLE2
 452 002416  010405                               MOV     R4,R5           ;NIBBLE1, R5=NIBBLE VALUE
 453 002420  010522                       1101$:  MOV     R5,(R2)+        ;XFER INDEX
 454 002422  016722  000000G                        MOV     NXTADD,(R2)+    ;COMPLETE INDEX STATE
 455 002426  012712  000000G                        MOV     #ST$INX,(R2)
 456 002432  056722  006022'                        BIS     FMSC,(R2)+
 457 002436  012704  000000C                        MOV     #VVEC+N.VEC,R4  ;R4=VECTOR ENTRIES ADR
 458 002442  016767  000000C 000000C               MOV     VMASK+N.POS,VVEC+N.POS  ;XFER FMT
 459 002450  005267  000000C                        INC     VVEC+N.POS              ;STEP FMT
 460 002454  016767  000000G 000000C               MOV     NXTADD,VVEC+N.ELSE      ;FSA LOC
 461 002462  005367  000000C                        DEC     VVEC+N.ELSE     ;INC'ED LATER
 462 002466  012700  000020                         MOV     #16.,R0         ;#BITS TO SCAN
 463 002472  005002                               CLR     R2              ;R2=SIZE
 464 002474  000261                               SEC
 465 002476  006002                       1102$:  ROR     R2              ;MASK
 466 002500  006305                               ASL     R5              ;VECTOR SUMMARY
 467 002502  103404                               BCS     1110$           ;IF SET IN SUM, THEN CREATE NODE
 468 002504  077004                       1103$:  SOB     R0,1102$
 469 002506                                       RESTOR  R1,R3
 470 002512  000207                               RTS     PC
 471 002514                               1110$:  SAVE    R0,R2,R3,R4,R5
 472 002526  005267  000000C                        INC     VVEC+N.ELSE     ;NEXT FSA LOC
 473 002532  016601  000014                         MOV     14(SP),R1       ;GET NIB INDICATOR
 474 002536  100414                               BMI     1120$           ;IF NIBBLE2
 475 002540  005867  000000C                        CLR     VVEC+N.LGT
 476 002544  012305                       1111$:  MOV     (R3)+,R5        ;GET ENTRY
 477 002546  100403                               BMI     1112$           ;XFER IF ELSE OR
 478 002550  030265  000000C                        BIT     R2,EMA+EMXNB1(R5)       ;IF MASK MATCHES
 479 002554                                       BOFF    1113$
```

```
  480 002556  010524                    1112$:  MOV·     R5,(R4)+
  481 002560  005267  000000C·                  INC·     VVEC+N.LGT·
  482·002564  077111                    1113$:  SOB·     R1,1111$
  483 002566  000416                            BR       1130$
  484 002570  012767  100000  000000C·  1120$:  MOV·     #100000,VVEC+N.LGT·      ;CLR·COUNT &·SET·AS·NIB2·
  485 002576  042701  000000G·                  BIC·     #BITNIB,R1
  486 002602· 012305                    1121$:  MOV·     (R3)+,R5         ;LIKE·LOOP·1111$ FOR·NIB2·
  487 002604  100403                            BMI·     1122$
  488 002606  030265  000000C·                  BIT·     R2,EMA+EMXNB2(R5)
  489 002612·                                   BOFF·    1123$
  490 002614  010524                    1122$:  MOV·     R5,(R4)+
  491 002616  005267  000000C·                  INC·     VVEC+N.LGT·
  492·002622· 077111                    1123$:  SOB·     R1,1121$
  493 002624                            1130$:  GETNXT·  R5               ;ALLOCATE·FSA·LOC·
  494 002706  062705  000000G·                  ADD·     #TDCT,R5
  495 002712· 012703  006002'                   MOV·     #FMNI,R3         ;SEARCH·FOR·DUP·OR ENTER·VECTOR·
  496 002716  012702· 000000G·                  MOV·     #VVEC,R2·
  497 002722· 004767  001604                    JSR·     PC,SCAN·
  498 002726  103405                            BCS·     1131$            ;NEW·VECTOR·
  499 002730  005025                            CLR·     (R5)+            ;DUP, JUMP·TO·IT·
  500 002732· 016425  000000G·                  MOV·     N.ELSE(R4),(R5)+
  501 002736  012725  000000G·                  MOV·     #ST$JSQ,(R5)+
  502·002742· 005267  000000G·          1131$:  INC·     NXTADD·          ;COMPLETE·ALLOCATION·
  503 002746  062767  000006  000000G·        ADD·     #6,NXTOFF·
  504 002754                                    RESTOR·  R0,R2,R3,R4,R5
  505 002766  000241                            CLC·                      ;DO·ADD·BIT·TO·MASK·
  506 002770  000645                            BR       1103$
```

```
                                        ;
                                        ;   FMSEQ IS A SUBROUTINE TO CREATE SEQUENCIAL STATES FOR SINGLE
                                        ;   ENTRY VECTORS.
                                        ;
                                        ;   R3 IS USED FOR INPUT AS A POINTER TO THE SINGLE ENTRY IN THE
                                        ;   VECTOR. NO REGISTERS ARE USED FOR OUTPUT.
                                        ;
002772  016701  006012'         FMSEQ:  MOV     CFSA,R1         ;CONVERT FSA LOC TO TDCT ADR
002776  070127  000006                  MUL     #6,R1
003002  066701  000000G                 ADD     TDCADR,R1
003006  011303                          MOV     (R3),R3         ;GET ENTRY
003010  100001                          BPL     1$
003012  005403                          NEG     R3              ;IF NEGATIVE, MAKE POSITIVE
003014  016305  000000G         1$:     MOV     EMA(R3),R5      ;GET EMA ENTRY
003020  032705  000000G                 BIT     #EMXMCF,R5      ;IS IT A MATCH CODE?
003024                                  BOFF    6$              ;NO
003026  000167  000734                  JMP     700$            ;YES
003032  032705  000000G         6$:     BIT     #EMXDTF,R5      ;R2=SOURCE CODE FOR FMT
003036                                  BON     2$
003040  032705  000000G                 BIT     #EMXZNF,R5
003044                                  BON     3$
003046  012702  003000                  MOV     #BIT10!BIT9,R2
003052  000405                          BR      4$
003054  012702  002000          2$:     MOV     #BIT10,R2
003060  000402                          BR      4$
003062  012702  001000          3$:     MOV     #BIT9,R2
003066  032705  000000G         4$:     BIT     #EMXMTV,R5      ;MULTIPLE VALUES FOR ENTRY?
003072                                  BOFF    7$              ;NO
003074  000167  001040                  JMP     100$            ;YES
003100  010100                  7$:     MOV     R1,R0           ;IS STATE BEING APENDED TO TDCT?
003102  062700  000000C                 ADD     #6-TDCT,R0
003106  020067  000000G                 CMP     R0,NXTOFF
003112  103442                          BLO     5$              ;NO
003114  001544                          BEQ     20$             ;CLOSE ENOUGH
003116                                  GETNXT  R1              ;YES: ALLOCATE NEXT ADR
003200  005267  000000G                 INC     NXTADD
003204  062767  000006  000000G        ADD     #6,NXTOFF
003212  062701  000000G                 ADD     #TDCT,R1        ;CONVERT OFFSET TO ADR
003216  000503                          BR      20$
003220  110521                  5$:     MOVB    R5,(R1)+        ;CREATE JUMP/SEQ STATE
003222  105021                          CLRB    (R1)+
003224  016721  000000G                 MOV     NXTADD,(R1)+
003230  012711  000000C                 MOV     #ST$JSQ!JMPBIT,(R1)
003234  050211                          BIS     R2,(R1)
003236  062703  000006          10$:    ADD     #6,R3           ;STEP TO NEXT EMA ENTRY
003242  016305  000000G                 MOV     EMA(R3),R5      ;GET ENTRY
003246                                  GETNXT  R1              ;ALLOCATE NEXT TDCT ADR
003330  062701  000000G                 ADD     #TDCT,R1
003334  005267  000000G                 INC     NXTADD
003340  062767  000006  000000G        ADD     #6,NXTOFF
003346  032705  000000G                 BIT     #EMXMCF,R5      ;IS ENTRY A MATCH CODE?
003352                                  BOFF    17$             ;NO
003354  000167  000406                  JMP     700$            ;YES
003360  032705  000000G         17$:    BIT     #EMXDTF,R5      ;SET UP SOURCE CODE
003364                                  BON     11$
003366  032705  000000G                 BIT     #EMXZNF,R5
003372                                  BON     12$
```

```
  565 003374  012702· 003000                  MOV·    #BIT10!BIT9,R2
  566 003400  000405                          BR      13$
  567 003402· 012702· 002000          11$:    MOV·    #BIT10,R2·
  568 003406  000402·                         BR      13$
  569 003410  012702· 001000          12$:    MOV·    #BIT9,R2·
  570 003414  032705  000000G·        13$:    BIT·    #EMXMTV,R5      ;MULTIPLE·VALUES?·
  571 003420                                  BOFF·   20$             ;NO·
  572·003422· 000167  000512                  JMP·    100$            ;YES·
  573 003426  012761  000000C·000004 20$:    MOV·    #ST$SEQ!S3BIT,4(R1)     ;CREATE·SEQ·STATE·
  574 003434  110511                          MOVB·   R5,(R1)         ;XFER·1ST·BYTE·OF·SEQ·
  575 003436  006202·                         ASR·    R2·
  576 003440  050261  000002                  BIS·    R2,2(R1)        ;ENTER·SOURCE·CODE
  577 003444  062703  000006                  ADD·    #6,R3           ;STEP·TO·NEXT·EMA·ENTRY·
  578 003450  016305  000000G·                MOV·    EMA(R3),R5      ;GET·NEXT·ENTRY·
  579 003454  032705  000000C·                BIT·    #EMXMCF!EMXMTV,R5       ;SEQ·OVER?
  580 003460                                  BOFF·   21$                     ;NO·
  581 003462· 052761  000000G·000004          BIS·    #S1BIT,4(R1)            ;YES: END·ON·1ST·BYTE·
  582·003470  032705  000000G·                BIT·    #EMXMCF,R5      ;MATCH·CODE?·
  583 003474                                  BON·    702$            ;YES·
  584 003476  032705  000000G·        21$:    BIT·    #EMXDTF,R5      ;SET·UP·SOURCE·CODE·
  585 003502·                                 BON·    22$
  586 003504  032705  000000G·                BIT·    #EMXZNF,R5
  587 003510                                  BON·    23$
  588 003512· 012702· 003000                  MOV·    #BIT10!BIT9,R2
  589 003516  000405                          BR      24$
  590 003520  012702· 002000          22$:    MOV·    #BIT10,R2·
  591 003524  000402·                         BR      24$
  592·003526  012702· 001000          23$:    MOV·    #BIT9,R2·
  593 003532· 032705  000000G·        24$:    BIT·    #EMXMTV,R5      ;MULTIPLE·VALUES?·
  594 003536                                  BON·    102$            ;YES·
  595 003540  110561  000001                  MOVB·   R5,1(R1)        ;XFER·2ND·BYTE·OF·SEQ·
  596 003544  006302·                         ASL·    R2·
  597 003546  050261  000002                  BIS·    R2,2(R1)        ;ENTER·SOURCE·CODE
  598 003552· 062703  000006                  ADD·    #6,R3           ;SET·TO·NEXT·EMA·ENTRY·
  599 003556  016305  000000G·                MOV·    EMA(R3),R5      ;GET·ENTRY·
  600 003562· 032705  000000C·                BIT·    #EMXMCF!EMXMTV,R5       ;SEQ·OVER?
  601 003566                                  BOFF·   25$                     ;NO·
  602·003570  052761  000000G·000004          BIS·    #S2BIT,4(R1)            ;YES: END·ON·2ND·BYTE·
  603 003576  032705  000000G·                BIT·    #EMXMCF,R5      ;MATCH· CODE?·
  604 003602·                                 BON·    702$            ;YES·
  605 003604  032705  000000G·        25$:    BIT·    #EMXDTF,R5      ;SET·UP·SOURCE·CODE·
  606 003610                                  BON·    26$
  607 003612· 032705  000000G·                BIT·    #EMXZNF,R5
  608 003616                                  BON·    27$
  609 003620  012702· 003000                  MOV·    #BIT10!BIT9,R2
  610 003624  000405                          BR      30$
  611 003626  012702· 002000          26$:    MOV·    #BIT10,R2·
  612·003632· 000402·                         BR      30$
  613 003634  012702· 001000          27$:    MOV·    #BIT9,R2·
  614 003640  032705  000000G·        30$:    BIT·    #EMXMTV,R5      ;MULTIPLE·VALUES?·
  615 003644                                  BON·    102$            ;YES·
  616 003646  110561  000002                  MOVB·   R5,2(R1)        ;XFER·3RD·BYTE·OF·SEQ·
  617 003652· 072227  000003                  ASH·    #3,R2·
  618 003656  050261  000002                  BIS·    R2,2(R1)        ;ENTER·  SOURCE·CODE·
  619 003662· 062703  000006                  ADD·    #6,R3           ;STEP·TO·MATCH·CODE·EMA·ENTRY·
  620 003666                          702$:   GETNXT· R1              ;ALLOCATE·FSA·LOC·
  621 003750  062701  000000G·                ADD·    #TDCT,R1
```

```
622 003754  062767  000006  000000G          ADD     #6,NXTOFF
623 003762  005267  000000G                  INC     NXTADD
624 003766  016311  000000C          700$:   MOV     EMA+EMXMCD(R3),(R1)      ;CREATE MATCH STATE
625 003772  042721  140000                   BIC     #BIT15!BIT14,(R1)+
626 003776  016721  000000G                  MOV     INTDEF,(R1)+
627 004002  012711  000000G                  MOV     #ST$MAT,(R1)
628 004006  032763  100000  000000C          BIT     #BIT15,EMA+EMXMCD(R3)    ;SET TERM TYPE
629 004014                                   BOFF    701$
630 004016  052711  000000G                  BIS     #CBIT,(R1)
631 004022  032763  040000  000000C 701$:    BIT     #BIT14,EMA+EMXMCD(R3)
632 004030                                   BOFF    703$
633 004032  052711  000000G                  BIS     #TBIT,(R1)
634 004036  000207                  703$:    RTS     PC
635 004040                           102$:   GETNXT  R1              ;ALLOCATE FSA LOC
636 004122  062701  000000G                  ADD     #TDCT,R1
637 004126  062767  000006  000000G          ADD     #6,NXTOFF
638 004134  005267  000000G                  INC     NXTADD
639 004140  016321  000000C          100$:   MOV     EMA+EMXNB1(R3),(R1)+     ;CREATE SINGLE EXIT INDEX
640 004144  016721  000000G                  MOV     NXTADD,(R1)+             ;FOR NIBBLE1 AND
641 004150  012711  000000C                  MOV     #ST$INX!SSBIT,(R1)
642 004154  050211                           BIS     R2,(R1)
643 004156                           GETNXT  R1
644 004240  062701  000000G                  ADD     #TDCT,R1
645 004244  005267  000000G                  INC     NXTADD
646 004250  062767  000006  000000G          ADD     #6,NXTOFF
647 004256  016321  000000C                  MOV     EMA+EMXNB2(R3),(R1)+     ;  FOR NIBBLE2
648 004262  016721  000000G                  MOV     NXTADD,(R1)+
649 004266  012711  000000C                  MOV     #ST$INX!SSBIT,(R1)
650 004272  050211                           BIS     R2,(R1)
651 004274  000167  176736                   JMP     10$
```

```
                                                ;
                                                ;   SUBROUTINE·NSQSUB·EXPANDS·VECTORS·DUE·TO·NON-SEQ·FLOW·
                                                ;
001100                                  NSQSUB: ;NO·MERGE·ENTRY·POSSIBLE, DON'T·CHECK·IN·QT1
001100                                          SAVE·    R0,R1
001104  005267  006060'                        INC·     NEST              ;FLAG·AS NESTED·
001110  010401                                 MOV·     R4,R1             ;ADJUST·EXPAND·VECTOR·AS· INPUT·
001112· 042705  000000C·                       BIC·     #EMXNSQ!BIT14!EMXEXF,R5
001116  006205                                 ASR·     R5
001120  010500                                 MOV·     R5,R0
001122· 001411                                 BEQ·     1$
001124  062701  000002G·                       ADD·     #EMA+2,R1         ;R1,R0=VECT0R·ADR, &·SIZE·
001130  004777  006030'                        JSR·     PC,@GVSUB·        ;RECALL·CALLER·
001134  005367  006060'                        DEC·     NEST              ;REMOVE·1 LEVEL·OF NEST· INDICATION·
001140                                         RESTOR·  R0,R1
001144  000207                                 RTS·     PC·
001146                                  1$:    MOUT$S·  #MSG13,FMEXL·
001166  000167  000000G·                       JMP·     EXIT
001172· 000056  001176'                 MSG13:  .WORD·   LNE13-LN13,LN13
     105     130      120               LN13:   .ASCIZ·  /EXP. CNT=0, EXP. VEC.=%1P,%1P,%1P,%1P,%1P,%1P/·
001254                                  LNE13:
                                                .EVEN·
```

```
                                        ;
                                        ;   GENERATE·MATCH·REPORT(S) !!!!!
                                        ;   CREATE A·SEPARATE·MATCH·REPORT·FOR·EACH·ENTRY·IN·VVEC.  INSERT·IN·
                                        ;   SEG·1 OF·MATCH·REPORT·STATE·THE·FLU/TERM·ID·FOUND·IN·NB1 OF·EMA.
                                        ;
                                        ;       R5 =·NEXT·AVAILABLE·TDCT·STATE·
                                        ;
004300  000440                  MATRP2::BR      1$
004302· 005267  000000G·         4$:    INC·    NXTADD·         ;ALLOCATE·NEXT·FSA LOC·
004306                                  GETNXT· R5
004370  062705  000000G·                 ADD·    #TDCT,R5
004374  062767  000006  000000G·        ADD·    #6,NXTOFF·
004402· 012302·                  1$:    MOV·    (R3)+,R2·       ;GET·EMA OFFSET·
004404  016202· 000000C·                 MOV·    EMA+EMXMCD(R2),R2·      ;R2=EMA·ADR·OF·MATCH CODE·
004410  020267  006032'                  CMP·    R2,OLDMC·       ;SAME·CODE·AS·BEFORE?·
004414  001435                          BEQ·    5$              ;YES·
004416  010267  006032'                 MOV·    R2,OLDMC·       ;NO: UPDATE·OLD·
004422· 010215                          MOV·    R2,(R5)         ;XFER·MATCH·CODE·
004424  042725  140000                  BIC·    #BIT15!BIT14,(R5)+      ;REMOVE·FLAGS·
004430  016725  000000G·                MOV·    NXTADD,(R5)+    ;ASSUME·ANOTHER·MATCH·CODE·
004434  012715  000000G·                MOV·    #ST$MAT,(R5)    ;DECLAR·AS·MATCH·STATE·
004440  032702· 100000                  BIT·    #BIT15,R2·      ;CWP?·
004444                                  BOFF·   2$              ;NO·
004446  052715  000000G·                BIS·    #CBIT,(R5)      ;YES·
004452· 032702· 040000           2$:    BIT·    #BIT14,R2·      ;TERM?·
004456                                  BOFF·   3$              ;NO·
004460  052715  000000G·                BIS·    #TBIT,(R5)      ;YES·
004464  010504                   3$:    MOV·    R5,R4           ;SAVE·CURRENT·TDCT STATE·ADR
004466  166704  000000G·                SUB·    TDCADR,R4       ;ALLOW·FOR·BUFFER·SHIFT·
004472· 077175                          SOB·    R1,4$           ;DO·FOR·ALL·MATCH·REPORTS·
004474  016765  000000G· 177776          MOV·    INTDEF,-2(R5)   ;RESET·LAST·STATES XFER·ADR·
004502· 005067  006032'                 CLR·    OLDMC·          ;PREVENT MATCH·WITH·NEXT·CODE·
004506  000207                          RTS·    PC·
004510  162767  000006  000000G· 5$:    SUB·    #6,NXTOFF·      ;DEALLOCATE·FSA·LOC·
004516  005367  000000G·                DEC·    NXTADD·
004522· 010405                          MOV·    R4,R5           ;RESTORE LAST·TDCT ADR·
004524  066705  000000G·                ADD·    TDCADR,R5       ;ALLOW·FOR·BUFFER·SHIFT·
004530  000755                          BR      3$
```

```
                                        ;
                                        ; SUBROUTINE TO SCAN NODE POOL FOR NODE AND MERGE INTO POOL IF NOT FOUND
                                        ;
                                        ;               INPUT                           OUTPUT
                                        ;       R0=     ..............UNUSED:..................
                                        ;       R1=     ..............UNUSED:..................
                                        ;       R2=     NODE TO SCAN FOR                (SAME)
                                        ;       R3=     POOL HEADER                     SCRATCH
                                        ;       R4=     SCRATCH                         ADR NODE IN POOL(OLD OR NEW)
                                        ;       R5=     (SAVED)                         (RESTORED)
                                        ;
004532  011304                  SCAN:  MOV     (R3),R4          ;GET 1ST NODE ADR
004534  001460                         BEQ     MERGE            ;NONE: MERGE WITH POOL HEADER
004536                                  SAVE    R3,R5
004542  016203  000000G                 MOV     N.LGT(R2),R3     ;R3=NODE SIZE
004546  026403  000000G           1$:   CMP     N.LGT(R4),R3     ;CMP CURRENT NODE SIZE WITH TEST NODE
004552  101045                         BHI     410$             ;PAST WITHOUT FINDING
004554  001404                         BEQ     11$              ;EQUAL: CHECK FURTHER
004556  011405                         MOV     (R4),R5          ;LESS THAN: STEP TO NEXT NODE
004560  001444                         BEQ     401$             ;IF NO MORE NODES ...
004562  010504                         MOV     R5,R4            ;IF MORE NODES
004564  000770                         BR      1$
004566  042703  000000G           11$:  BIC     #BITNIB,R3       ;REMOVE FLAG FROM SIZE
004572                                  SAVE    R2,R3,R4
004600  062702  000000G                 ADD     #N.VEC,R2        ;STEP TO CONTENT PORTION OF NODE
004604  062704  000000G                 ADD     #N.VEC,R4
004610  005703                         TST     R3               ;ZERO LENGTH COMPARES ALWAYS WORK
004612  001404                         BEQ     1100$
004614  022422                  110$:  CMP     (R4)+,(R2)+      ;CMP NODE CONTENTS
004616  103411                         BLO     111$             ;LESS THAN
004620  101017                         BHI     112$             ;PAST WITHOUT FINDING
004622  077304                         SOB     R3,110$          ;EQUAL: CONTINUE UNTIL ALL CHECKED
004624                          1100$: RESTORE R2,R3,R4         ;ALL
004632                                 RESTORE R3,R5            ;THE NODE MATCHES!!!!!!!
004636  000241                         CLC
004640  000207                         RTS     PC
004642                          111$:  RESTORE R2,R3,R4
004650  011405                         MOV     (R4),R5          ;STEP TO NEXT NODE
004652  001407                         BEQ     401$             ;IF NO MORE NODES ...
004654  010504                         MOV     R5,R4            ;IF MORE NODES
004656  000733                         BR      1$
004660                          112$:  RESTORE R2,R3,R4
004666  016404  000000G         410$:  MOV     N.BACK(R4),R4    ;BACK UP TO PREVIOUS NODE
004672                          401$:  RESTORE R3,R5
004676                          MERGE: SAVE    R4,R5,R2
004704  016205  000000G                 MOV     N.LGT(R2),R5     ;R5=NODE VECTOR SIZE (BYTES)
004710  042705  000000G                 BIC     #BITNIB,R5
004714  006305                         ASL     R5
004716  062705  000000G                 ADD     #N.VEC,R5
004722  020327  006002'                 CMP     R3,#FMNI                ;UPDATE NEXT POINTER & GET NEW NODE ADR
004726  001011                         BNE     1$
004730  016704  006000'                 MOV     NFMN,R4
004734  060567  006000'                 ADD     R5,NFMN
004740  026727  006000' 004540'        CMP     NFMN,#FMNPE      ;POOL OVERFLOW?
004746  101046                         BHI     12$              ;YES
004750  000410                         BR      2$
004752  016704  006006'           1$:   MOV     NFME,R4
```

```
  749 004756  060567  006006'                    ADD·    R5,NFME·
  750 004762· 026727  006006' 005540'            CMP·    NFME,#FMEPE·     ;POOL·OVERFLOW?·
  751 004770  101035                              BHI·    12$              ;YES·
  752·004772· 010416                    2$:       MOV·    R4,(SP)          ;ON·RESTOR·R2=NEW·NODE·
  753 004774  006205                              ASR·    R5               ;CONVERT TO·WORD·COUNT·
  754 004776  012224                    3$:       MOV·    (R2)+,(R4)+      ;XFER·VVEC·TO·NEW·NODE··
  755 005000  077502·                             SOB·    R5,3$
  756 005002·                                     RESTOR· R4,R5,R2·
  757 005010  005704                              TST·    R4               ;R4=POOL HEADER?·
  758 005012· 001410                              BEQ·    10$              ;YES·
  759 005014  011403                              MOV·    (R4),R3          ;NO:ADJUST·LINKAGES·
  760 005016  010312·                             MOV·    R3,(R2)
  761 005020  010462· 000000G·                    MOV·    R4,N.BACK(R2)
  762·005024  010214                              MOV·    R2,(R4)
  763 005026  010263  000000G·                    MOV·    R2,N.BACK(R3)
  764 005032· 000411                              BR      11$
  765 005034  010462· 000000G·          10$:      MOV·    R4,N.BACK(R2)    ;ADJUST·LINKAGES·
  766 005040  011304                              MOV·    (R3),R4
  767 005042· 010412·                             MOV·    R4,(R2)
  768 005044  010213                              MOV·    R2,(R3)
  769 005046  005704                              TST·    R4               ;AT·HEADER?·
  770 005050  001402·                             BEQ·    11$              ;YES: NO BACK·LINK
  771 005052· 010264  000000G·                    MOV·    R2,N.BACK(R4)    ;NO: SET BACK·LINK
  772·005056  010204                    11$:      MOV·    R2,R4            ;R4=MERGED·NODE·
  773 005060  000261                              SEC·                     ;SET·MERGE·FLAG·
  774 005062· 000207                              RTS·    PC·
  775 005064  016705  006000'           12$:      MOV·    NFMN,R5          ;ENTER·NODE·SIZES·
  776 005070  162705  000000'                     SUB·    #FMNPS,R5
  777 005074  006205                              ASR·    R5
  778 005076  010567  006036'                     MOV·    R5,FMPAR1+2·
  779 005102· 016705  006006'                     MOV·    NFME,R5
  780 005106  162705  004540'                     SUB·    #FMEPS,R5
  781 005112· 006205                              ASR·    R5
  782·005114  010567  006040'                     MOV·    R5,FMPAR1+4
  783 005120  016767  006014' 006042'            MOV·    CFMT,FMPAR1+6    ;ENTER·FMTYP·BEING SCANNED·
  784 005126                                      MOUT$S· #MSG10,#FMPAR1           ;REPORT·ERROR·
  785 005146  000167  000000G·                    JMP·    EXIT
```

```
                                        ;
                                        ;   SUBROUTINE FMTST1'S ONLY FUNCTION IS TO MAKE A COMPLEX DECISION.
                                        ;   IT MUST DECIDE WHEITHER THE REMAINING NON-ELSE VECTORS SHOULD BE
                                        ;   IGNORED OR CONSIDERED. COMPLEX FLU-MOD'S MAY BE CONSIDERABLE
                                        ;   SIMPLIFIED BY IGNORING SUBSEQUENT MATCH CRITERIA ON A PATH ONCE
                                        ;   IT IS KNOWN THAT THAT PATH MUST MATCH. HOWEVER, SINCE THE VECTOR
                                        ;   MAY CONSIST OF MULTIPLE COMPLEX FLU-MOD'S, THIS DECISION CAN BE
                                        ;   QUITE INVOLVED.
                                        ;
005152  032767  000000G 006020' FMTST1: BIT     #EMXMCF,FMTSUM  ;HAVE ANY CRITERIA BEEN STATISFIED?
005160                                  BOFF    210$            ;NO: MUST CONSIDER OTHER CRITERIA
005162  022767  177777  006046'       CMP     #-1,FMEXF       ;YES: WHAT DOES VECTOR REPESENT?
005170  001576                        BEQ     21$             ;MULTIPLE SIMPLE FLU MOD'S, CONSIDER REST
005172  002434                        BLT     310$            ;SINGLE COMPLEX FLU-MOD, IGNORE REST
005174                                SAVE    R4,R5           ;MULTIPLE COMPLEX FLU-MOD'S,CHECK FURTHER
005200  016601  000010                MOV     10(SP),R1       ;R1=VECTOR SIZE
005204  016603  000006                MOV     6(SP),R3        ;R3=START OF VECTORS
005210  016704  006024'                MOV     FMEXL,R4        ;R4=POINTER TO FLU-MOD VECTO STARTS
005214  012405                        MOV     (R4)+,R5        ;R5=# FLU-MOD'S
005216  006205                        ASR     R5
005220  042705  160000                BIC     #160000,R5
005224  005067  006054'                CLR     FMAT            ;CLEAR INDICATORS
005230  005067  006050'                CLR     FMIN
005234  005067  006052'                CLR     FVAL
005240  012767  005664' 005662'       MOV     #VALVEC,VALVPT  ;RESET VALUE VECTOR PTR
005246  005067  005734'                CLR     LEVEL
005252  012302                1$:     MOV     (R3)+,R2        ;GET VECTOR ENTRY
005254  100407                        BMI     2$              ;IF ELSE
005256  005067  006050'                CLR     FMIN            ;INDICATE POSITIVE
005262  000407                        BR      3$
005264  000167  000326        310$:   JMP     31$
005270  000167  000272        210$:   JMP     21$
005274  005402                2$:     NEG     R2              ;MAKE POSITIVE
005276  105267  006050'                INCB    FMIN            ;INDICATE NEGATIVE
005302  012400                3$:     MOV     (R4)+,R0        ;FIND FLU-MOD FOR VECTOR ENTRY
005304  100013                        BPL     40$
005306                                SAVE    R4,R5           ;IF NEG, ENTRY IS NESTED
005312  005267  005734'                INC     LEVEL           ;INDDICATE LEVEL NESTED
005316  005400                        NEG     R0              ;CONVERT TO POINTER
005320  010004                        MOV     R0,R4
005322  012405                        MOV     (R4)+,R5        ;GET COUNT
005324  006205                        ASR     R5
005326  042705  160000                BIC     #160000,R5
005332  012400                        MOV     (R4)+,R0        ;GET ENTRY
005334  020200                40$:    CMP     R2,R0
005336  103426                        BLO     10$             ;FOUND
005340  077520                4$:     SOB     R5,3$
005342  005767  005734'                TST     LEVEL
005346  001405                        BEQ     41$             ;IF NOT NESTED
005350  005367  005734'                DEC     LEVEL           ;IF NESTED, RETURN TO PRIOR LEVEL
005354                                RESTOR  R4,R5
005360  000767                        BR      4$
005362  012700  177777        41$:    MOV     #-1,R0          ;LAST ENTRY
005366  000412                        BR      10$
005370  012302                5$:     MOV     (R3)+,R2        ;GET NEXT VECTOR ENTRY
005372  100403                        BMI     6$              ;ELSE
005374  105067  006050'                CLRB    FMIN            ;INDICATE POSITIVE
```

```
844 005400  000403                          BR      7$
845 005402  005402                  6$:     NEG     R2              ;MAKE POSITIVE
846 005404  105267  006050'                 INCB    FMIN            ;INDICATE NEGATIVE
847 005410  020200                  7$:     CMP     R2,R0           ;DOES THIS VECTOR ENTRY USE SAME FLU-MOD
848 005412  103041                          BHIS    13$             ;NO
849 005414  105767  006050'          10$:    TSTB    FMIN            ;YES: ELSE ENTRY?
850 005420  001410                          BEQ     11$             ;NO
851 005422  032762  000000G 000000G          BIT     #EMXMCF,EMA(R2) ;YES: MATCH CODE ENTRY?
852 005430                                  BOFF    12$             ;NO: SKIP
853 005432  005402                          NEG     R2              ;YES: STORE MATCH CODE
854 005434  010267  006054'                 MOV     R2,FMAT
855 005440  000416                          BR      12$
856 005442  010302                  11$:    MOV     R3,R2           ;SAVE VECTOR LOCATION
857 005444  162702  000002                  SUB     #2,R2
858 005450  010277  005662'                 MOV     R2,@VALVPT
859 005454  062767  000002  005662'         ADD     #2,VALVPT
860 005462  026727  005662' 005732'         CMP     VALVPT,#VALVEC+38.      ;DON'T LET OVERFLOW
861 005470  101076                          BHI     VVOVF
862 005472  005267  006052'                 INC     FVAL            ;OK
863 005476  077144                  12$:    SOB     R1,5$           ;CHECK ALL VECTOR ENTRIES
864 005500  005767  006052'                 TST     FVAL            ;WAS NON-ELSE FOUND?
865 005504  001434                          BEQ     30$             ;NO: NON-ELSE CAN BE IGNORED!!!!!!
866 005506  005767  006054'                 TST     FMAT            ;YES: WAS MATCH FOUND?
867 005512  001027                          BNE     300$            ;YES: NON-ELSE CAN BE IGNORED!!!!!!!
868 005514  000414                          BR      20$             ;NO: OTHER CRITERIA MUST BE CONSIDERED!!
869 005516  105767  006052'          13$:    TSTB    FVAL            ;WAS NON-ELSE FOUND?
870 005522  001406                          BEQ     14$             ;NO: SKIP
871 005524  105767  006054'                 TSTB    FMAT            ;YES: WAS MATCH FOUND?
872 005530  001406                          BEQ     20$             ;NO: OTHER CRITERIA MUST BE CONSIDERED!!
873 005532  004767  000064                  JSR     PC,400$         ;REMOVE DUPLICATE CRITERIA
874 005536  000700                          BR      4$
875 005540  005067  006054'          14$:    CLR     FMAT            ;CLEAR INDICATERS FOR NEXT FLU-MOD
876 005544  000675                          BR      4$              ;TEST NEXT FLU-MOD
877 005546                           20$:    RESTOR  R4,R5
878 005552  005767  005734'                 TST     LEVEL
879 005556  001403                          BEQ     21$
880 005560  005367  005734'                 DEC     LEVEL           ;IF NESTED, RETURN TO PRIOR LEVEL
881 005564  000770                          BR      20$
882 005566  000241                  21$:    CLC                     ;INDICATE THAT OTHER CRITERIA MUST BE
883 005570  000207                          RTS     PC              ; CONSIDERED
884 005572  004767  000024          300$:   JSR     PC,400$         ;REMOVE DUP CRITERIA
885 005576                           30$:    RESTOR  R4,R5
886 005602  005767  005734'                 TST     LEVEL
887 005606  001403                          BEQ     31$
888 005610  005367  005734'                 DEC     LEVEL           ;IF NESTED, RETURN TO PRIOR LEVEL
889 005614  000770                          BR      30$
890 005616  000261                  31$:    SEC                     ;INDICATE THAT NON-ELSE CAN BE IGNORED
891 005620  000207                          RTS     PC
892                                  ;
893 005622                           400$:   SAVE    R2
894 005624  016700  006054'                 MOV     FMAT,R0         ;REPLACE ALL STORED VECTOR LOCATIONS
895 005630  016701  006052'                 MOV     FVAL,R1         ;WITH MATCH CODE ENTRY
896 005634  012702  005664'                 MOV     #VALVEC,R2
897 005640  010032                  401$:   MOV     R0,@(R2)+
898 005642  077102                          SOB     R1,401$
899 005644  005067  006054'                 CLR     FMAT
900 005650  005067  006052'                 CLR     FVAL
```

```
                                        ;
                                        ;   SUBROUTINE FMTST2'S ONLY FUNCTION IS TO MAKE A COMPLEX DECISION;
                                        ;   IT MUST DECIDE WEITHER A SINGE EXIT STATE IS POSSIBLE FOR THIS STATE;
                                        ;
005712  020027  000001          FMTST2: CMP     R0,#1           ;SINGLE ENTRY VECTOR?
005716  003536                  .       BLE     30$             ;YES: CAN ALWAYS BE SINGLE EXIT
005720  032767  000000G 006016'  .       BIT     #EMXSZF,FMT     ;NO: PROCESSING SUB-ZONES?
005726                          .       BOFF    20$             ;NO: PROBABLY CAN'T BE SINGLE EXIT
005730  005766  000004          .       TST     4(SP)           ;YES: NIBBLE2?
005734  100414                  .       BMI     1$              ;YES: CHECK IF SINGLE FLU-MOD
005736  022705  000001          .       CMP     #1,R5           ;NO: CHECK IF NIBBLE2 HAS SINGLE VALUE
005742  001411                  .       BEQ     1$              ;SINGLE VALUE
005744  022705  000002          .       CMP     #2,R5
005750  001406                  .       BEQ     1$
005752  022705  000004          .       CMP     #4,R5
005756  001403                  .       BEQ     1$
005760  022705  000010          .       CMP     #10,R5
005764  001101                  .       BNE     20$             ;MULTI-VALUE, CAN'T BE SINGLE EXIT
005766  022767  177777  006046' 1$:     CMP     #-1,FMEXF       ;WHAT TYPE OF FLU-MOD?
005774  002507                  .       BLT     30$             ;SINGLE COMPLEX FLU-MOD: SINGLE EXIT
005776  001474                  .       BEQ     20$             ;MULTIPLE SIMPLE FLU-MOD'S: CAN'T USE
006000                          .       SAVE    R4,R5           ;MULTIPLE COPLEX FLU-MOD'S
006004  016601  000012          .       MOV     12(SP),R1       ;R1=VECTOR SIZE
006010  016603  000010          .       MOV     10(SP),R3       ;R3=START OF VECTORS
006014  016704  006024'          .       MOV     FMEXL,R4        ;R4=POINTER TO FLU-MOD VECTOR STARTS
006020  012405                  .       MOV     (R4)+,R5        ;R5=# FLU-MOD'S
006022  006205                  .       ASR     R5
006024  042705  160000          .       BIC     #160000,R5
006030  005205                  .       INC     R5
006032  005067  005734'          .       CLR     LEVEL
006036  042701  100000          .       BIC     #100000,R1
006042  005301                  10$:    DEC     R1              ;GET 1ST NON-ELSE VECTOR ENTRY
006044  012300                  .       MOV     (R3)+,R0
006046  100775                  .       BMI     10$
006050  005305                  11$:    DEC     R5              ;FLU-MOD COUNT
006052  001426                  .       BEQ     32$             ;LAST FLU-MOD, ALL SAME, SINGLE EXIT
006054  012402                  .       MOV     (R4)+,R2        ;FIND FLU-MOD NOD FOR THIS ENTRY
006056  100013                  .       BPL     40$
006060                          .       SAVE    R4,R5           ;IF NEG, ENTRY IS POINTER
006064  005267  005734'          .       INC     LEVEL           ;NEST A LEVEL
006070  005402                  .       NEG     R2
006072  010204                  .       MOV     R2,R4           ;GET LOCATION AND COUNT
006074  012405                  .       MOV     (R4)+,R5
006076  006205                  .       ASR     R5
006100  042705  160000          .       BIC     #160000,R5
006104  012402                  .       MOV     (R4)+,R2        ;GET ENTRY
006106  020002                  40$:    CMP     R0,R2
006110  103357                  .       BHIS    11$             ;FOUND
006112  005301                  12$:    DEC     R1              ;VECTOR COUNT
006114  100427                  .       BMI     31$             ;NO MORE ENTRIES,ALL SAME,SINGLE EXIT
006116  012300                  .       MOV     (R3)+,R0        ;GET NEXT NON-ELSE VECTOR
006120  100774                  .       BMI     12$             ;ELSE
006122  020002                  .       CMP     R0,R2           ;SAME FLU-MOD?
006124  103772                  .       BLO     12$             ;YES
006126  000410                  .       BR      21$             ;NO
006130  005767  005734'          32$:    TST     LEVEL
006134  001417                  .       BEQ     31$
```

```
  965 006136  005367  005734'                DEC·····LEVEL·          ;IF·NESTED,RETURN·
  966 006142·                                RESTOR· R4,R5
  967 006146  000740                         BR      11$
  968 006150                         21$:    RESTOR· R4,R5           ;NO·CAN'T·USE·SINGLE·EXIT·
  969 006154  005767  005734'                TST·    LEVEL·
  970 006160  001403                         BEQ·    20$
  971 006162· 005367  005734'                DEC·    LEVEL·          ;IF·NESTED,RETURN·
  972·006166  000770                         BR      21$
  973 006170  000241                 20$:    CLC·
  974 006172· 000207                         RTS·    PC·
  975 006174                         31$:    RESTOR· R4,R5           ;USE·SINGLE·EXIT·
  976 006200  005767  005734'                TST·    LEVEL·
  977 006204  001403                         BEQ·    30$
  978 006206  005367  005734'                DEC·    LEVEL·          ;IF·NESTED,RETURN·
  979 006212· 000770                         BR      31$
  980 006214  000261                 30$:    SEC·
  981 006216  000207                         RTS·    PC·
```

```
                                        ;
                                        ;   SUBROUTINE·GETNDE·GETS· THE NEXT·FLU-MOD·NODE·THAT·HAS·NOT·YET·BEEN
                                        ;   PROCESSED.
                                        ;
                                        ;   ON·OUTPUT·R3 =·NODE·TO·BE·PROCESSED·
                                        ;
006220  005767  006056'         GETNDE: TST·    ENFLAG·          ;ELSE·OR NORMAL·NODE·SEARCH?
006224  001062·                         BNE·    11$              ;ELSE·
006226  016703  005776'         1$:     MOV·    CFMN,R3          ;NORMAL: GET·NODE·
006232· 020367  006000'                 CMP·    R3,NFMN·         ;END·OF·POOL?·
006236  001432·                         BEQ·    2$               ;YES·
006240  126763  006014' 000000G·        CMPB·   CFMT,N,POS(R3)   ;NO: DO·FLU-MOD·TYPES·MATCH?
006246  001437                         BEQ·    3$               ;YES·
006250  005067  006002'                 CLR·    FMNI             ;DELETE·PREVIOUS·NORMAL·NODES·
006254  005267  006014'                 INC·    CFMT             ;STEP·TO NEXT·FLU-MOD·TYPE·
006260  032767  000001  006014'        BIT·    #1,CFMT·         ;SHOULD·WE·SWITCH·TO·ELSE·NODES?·
006266                                 BON·    1$               ;NO·
006270  016703  006014'         4$:     MOV·    CFMT,R3          ;CONVERT TYPE·TO·SOURCE·CODE
006274  006303                         ASL·    R3
006276· 016367  005756' 006022'        MOV·    FMTSC(R3),FMSC
006304  016367  005736' 006016'        MOV·    FMTYP(R3),FMT·   ;&·BIT·FLAG·
006312· 005067  006010'                 CLR·    FMEI             ;DELETE·PREVIOUS·ELSE·NODES·FROM· INDEX
006316  005267  006056'                 INC·    ENFLAG·          ;NOTE·ELSE·SEARCH·
006322· 000423                         BR      11$
006324  062767  000002  006014' 2$:     ADD·    #2,CFMT·         ;STEP·TO NEXT·FLU-MOD·TYPE·FOR·ELSE·
006332· 042767  000001  006014'        BIC·    #1,CFMT·
006340  005067  006002'                 CLR·    FMNI             ;DELETE·NORMAL·NODES·FROM· INDEX·
006344  000751                         BR      4$
006346  016302· 000000G·         3$:     MOV·    N,LGT(R3),R2·    ;STEP·PAST·NODE·
006352· 006302·                         ASL·    R2·
006354  060302·                         ADD·    R3,R2·
006356  062702· 000000G·                ADD·    #N,VEC,R2·
006362· 010267  005776'                 MOV·    R2,CFMN·         ;UPDATE·CURRENT·NODE·POINTER
006366  000241                         CLC·
006370  000207                         RTS·    PC·
006372· 016703  006004'         11$:    MOV·    CFME,R3          ;GET·CURRENT·ELSE·NODE·
006376  020367  006006'                 CMP·    R3,NFME·         ;END·OF·POOL?·
006402· 001407                         BEQ·    12$              ;YES·
006404  126763  006014' 000000G·        CMPB·   CFMT,N,POS(R3)   ;NO: DO·FLU-MOD·TYPES·MATCH?
006412· 001411                         BEQ·    13$              ;YES·
006414  005067  006056'         14$:    CLR·    ENFLAG·          ;SWITCH·TO·NORMAL·NODE·SEARCH·
006420  000702·                         BR      1$
006422· 026727  006014' 000010  12$:    CMP·    CFMT,#MCTYP·     ;END·OF·SEARCH?·
006430  002771                         BLT·    14$              ;NO·
006432· 000261                         SEC·                     ;YES: NO NODE,SEARCH·COMPLETE·
006434  000207                         RTS·    PC·
006436  016302· 000000G·         13$:    MOV·    N,LGT(R3),R2·    ;STEP·PAST·NODE·
006442· 006302·                         ASL·    R2·
006444  060302·                         ADD·    R3,R2·
006446  062702· 000000G·                ADD·    #N,VEC,R2·
006452· 010267  006004'                 MOV·    R2,CFME·
006456  000241                         CLC·
006460  000207                         RTS·    PC·
                                        ;
000001                                  .END·
```

```
ASTKMX=·000400 G·
AVELGT=·000025 G·
BITNIB=·****** GX·
BITVAL=·000000
BIT0  =·000001
BIT1  =·000002·
BIT10 =·002000
BIT11 =·004000
BIT12·=·010000
BIT13 =·020000
BIT14 =·040000
BIT15 =·100000
BIT2·=·000004
BIT3  =·000010
BIT4  =·000020
BIT5  =·000040
BIT6  =·000100
BIT7  =·000200
BIT8·=·000400
BIT9  =·001000
BS.CLS=·000002·
BS.DBU=·000004
BS.INA=·000000
BS.OPN=·000001
BS.SRC=·000003
BYTE0 =·000000
BYTE1 =·000001
BYTE2·=·000002·
BYTE3 =·000003
BYTE4 =·000004
BYTE5 =·000005
BYTE6 =·000006
BYTE7 =·000007
BYTE8·=·000010
BYTE9 =·000011
BYTVAL=·000012·
B.BSTA·000054
B.CNTX·000046
B.COUQ·000060
B.FEMA·000132·
B.FEMB·000142·
B.FEMC·000152·
B.FFSA·000202·
B.FFSB·000212·
B.FFSC·000222·
B.FMHR·000172·
B.FQLS·000162·
B.FSAZ·000100
B.FSBZ·000102·
B.FSCZ·000104
B.HBLK·000120
B.HDOC·000114
B.HRLP·000126
B.HRLR·000122·
B.HRLW·000124
B.NMBR·000052·
B.NQRY·000232·
B.QLSZ  000106
B.QMAP  000234
B.QSPL  000316
B.QTTM  000076
B.QUQP  000056
B.SFDB  000010
B.SIZE  000772·
B.SNDP  000012·
B.SSQ·  000004
B.SSQF  000050
B.STAT  000044
B.STTE  000053
B.UDOC  000110
CBIT·  =·****** GX·
CFME·    006004R·
CFMN·    005776R·
CFMT·    006014R·
CFSA·    006012R·
CF.B0 =·000070
CF.B2·=·000067
CF.B4 =·000066
CF.B6 =·000065
CF.DR0=·000064
CF.DR1=·000063
CPIXMK=·177770 G
CPIXSZ=·000010 G
DBSLEN=·000116
DH.BF0  000002·
DH.BF1  000004
DH.CTL  000000
DH.DMC  000010
DH.FLG  000006
DN.DCK  000000
DN.NTP  000004
DN.NXT  000006
DN.ROT  000002·
010 DN.SIZ  000010
010 EIXCNT=·000010 G
010 ELSCLS=·000007 G
010 ELSMSK=·177761 G
010 EMA·   =·****** GX·
010 EMACLS=·000003 G
010'EMACUT=·001700 G
010 EMAMSZ=·016000 G
010 EMBCLS=·000004 G
010 EMBCUT=·001130 G
010 EMBMSZ=·004400 G
010 EMCMSZ=·001000 G
010 EMXDTF=·****** GX·
010 EMXEXF=·****** GX·
010 EMXMCD=·****** GX·
010 EMXMCF=·****** GX·
010 EMXMSZ=·016000 G
010 EMXMTV=·****** GX·
010 EMXNB1=·****** GX·
010 EMXNB2=·****** GX·
010 EMXNSQ=·****** GX·
010 EMXSZF=·****** GX·
010 EMXVDC=·****** GX·
010 EMXZNF=·****** GX·
010 ENCNT·=·000004 G·
010 ENFLAG·006056R·
010 ERROR2=·****** GX·
010 EXIT·  =·****** GX·
010 FD.FID·000000
010 FD.FNB·000006
010 FD.FVR·000004
010 FD.LEN·000010
010 FLIXMK=·177740 G·
010 FLIXSZ=·000040 G·
    FLUCLS=·000001 G·
014 FLUCUT=·000016 G·
014 FLUMD·  000000RG·
014 FMAT·   006054R·
014 FMECT·  006026R·
    FMEI·   006010R·
    FMEPE·  005540R·
    FMEPS·  004540R·
    FMEPSZ=·000400 G·
    FMEXF·  006046R·
    FMEXF2· 005660R·
    FMEXL·  006024R·
    FMIN·   006050R·
    FMIXSZ=·000001 G·
005 FMLOG·  006044R·
005 FMNI·   006002R·
005 FMNPE·  004540R·
005 FMNPS·  000000R·
005 FMNPSZ=·002260 G·
013 FMPAR1  006034R·
013 FMSC·   006022R·
013 FMSEQ·  002772R·
013 FMT·    006016R·
013 FMTSC·  005756R·
    FMTST1  005152R·
    FMTST2· 005712R·
    FMTSUM· 006020R·
    FMTYP·  005736R·
    FNPOSZ=·000400 G·
    FNPSZ·=·005734 G·
    FN.DBR· 000026
    FN.DBS· 000022·
    FN.DHR· 000040
    FN.EMA· 000012·
    FN.EMB· 000014
    FN.EMC· 000016
    FN.FSA· 000000
    FN.FSB· 000002·
    FN.FSC· 000004
    FN.LGQ· 000034
    FN.LGU· 000036
    FN.MFO· 000024
    FN.MHR· 000010
    FN.NMB· 000044
    FN.QLS· 000006
    FN.QRY· 000020
    FN.SF0  000030
    FN.SF1  000032·
014 FN.SHD· 000042·
    FVAL·   006052R·
    GETNDE· 006220R·
003 GVMASK· 000356RG·
003 GVSUB·  006030R·
003 GV1     000552R·
003 GV2·    000572R·
    GV3     000656R·
    HSTKMX=·000100 G·
    IMASK·  001254R·
    INTDEF=·****** GX·
015 IWBIT·=·****** GX·
014 JMPBIT=·****** GX·
014 LEVEL·  005734R·
014 LNE13   001254R·
014 LN11    006145R·
014 LN11E·  006213R·
    LN12·   000006R·
014 LN12E·  000054R·
014 LN13    001176R·
014 LN7     006072R·
014 LN7E·   006145R·
    LOGCLS=·000002·G·
014 LOWOFF=·****** GX·
014 M·     =·000062·
014 MATRP2· 004300RG·
014 MCTYP·=·000010
    MERGE·  004676R·
014 MIXFMC· 005540RG·
014 MOVTDC=·****** GX·
015 MSGOUT=·****** GX·
014 MSG10   006062R·
014 MSG11   006066R·
015 MSG12·  000002R·
015 MSG13   001172R·
014 N·     =·000002·
014 NDBCLS=·000006 G·
    NEST·   006060R·
    NFME·   006006R·
011 NFMN·   006000R·
011 NIXCNT=·000020 G·
011 NNMASK=·177760 G·
011 NPECNT=·000144 G·
011 NPEVSZ=·000043 G·
011 NP1MSZ=·010000 G·
011 NP2MSZ=·036000 G·
011 NP2OSZ=·000153 G·
011 NP3MSZ=·000524 G·
011 NP3OSZ=·000125 G·
011 NSQSUB· 001100R·
011 NXTADD=·****** GX·
011 NXTOFF=·****** GX·
011 N.BACK=·****** GX·
011 N.BFAC=·000004
011 N.BHGH=·000006
011 N.BTCH=·000004
011 N.BUFB=·004000
011 N.BUFW=·002000
014 N.ELSE=·****** GX·
015 N.FOS·=·000764
015 N.FSA·=·****** GX·
014 N.LGT·=·****** GX·
015 N.PKSZ=·000020
015 N.PKTS=·000043
015 N.POS·=·****** GX·
    N.QURY=·000031
015 N.SUNT=·000002
    N.VEC·=·****** GX·
    OLDMC·  006032R·    014
    PSTKMX=·000024 G·
014 OE.RO1=·000144
015 QNDCNT=·001000 G·
014 QRYSZ·=·002000 G·
014 Q.FDSC· 000004      007
    Q.NQBK· 000000      007
    Q.NUHL· 000002      007
015 Q.SIZE· 000014      007
014 SCAN    004532R·    015
014 SR.ARE· 000114      002·
    SR.ARS· 000106      002·
    SR.DAY· 000010      002·
    SR.DLT· 000014      002·
015 SR.ECB· 000047      002·
    SR.ECH· 000046      002·
015 SR.ECL· 000050      002·
014 SR.FIB· 000012      002·
    SR.GRE· 000100      002·
    SR.GRS· 000072      002·
014 SR.LEN· 000122      002·
014 SR.LIN· 000066      002·
    SR.LIP· 000062      002·
015 SR.MON· 000006      002·
    SR.NDC· 000042      002·
    SR.NDS· 000036      002·
014 SR.NIN· 000030      002·
014 SR.NIP· 000022      002·
014 SR.SDB· 000032      002·
    SR.SRC· 000002      002·
    SR.SUN· 000000      002·
    SR.TWS· 000056      002·
    SR.WSL· 000052      002·
    SR.YR·  000004      002·
    SR.1IN· 000024      002·
    SR.1IP· 000016      002·
    SSBIT·=·****** GX·
    SS.FID· 000002      004
015 SS.FNB· 000010      004
    SS.FVR· 000006      004
    SS.LEN· 000012      004
    SS.STT· 000000      004·
```

```
FLU-MOD-CODE··· MACRO·M1110  27-MAR-80 13:10  PAGE 23-2
SYMBOL·TABLE·

ST$CNG=·****** GX·      SU.IDL=·000000          TDCBLK=·000010 G·       VMASK·=·****** GX·      WORD8 =·000020
ST$INX=·****** GX·      SU.LOD=·000001          TDCMAX=·****** GX·      VVEC·  =·****** GX·      WORD9 =·000022
ST$JSQ=·****** GX·      SU.SRC=·000002·         TDCMSZ=·004000 G·       VVOVF·  005666R·    015 WRDVAL=·000024
ST$MAT=·****** GX·      SU.SRR=·000005          TDCT·  =·****** GX·     WN.NTP·  000004     012·WRTTDC=·****** GX·
ST$SEQ=·****** GX·      SU.XPD=·000003          TRCFLG·  000000R·       WN.NXT·  000006     012·XBATCH=·000013
ST.ASZ·  000020      006 S.HRL·=·000240          TRMCUT=·000040 G·       WN.ROT·  000002·    012·XDBLOA=·000004
ST.BSZ·  000024      006 S1BIT·=·****** GX·      TSTKMX=·000400 G·       WN.SIZ·  000010     012·XDBPRO=·000012
ST.BTC·  000000      006 S2BIT·=·****** GX·      VALVEC·  005664R·   014 WN.SRC·  000000     012·XDMCIN=·000006
ST.CSZ·  000030      006 S3BIT·=·****** GX·      VALVPT·  005662R·   014 WN.TYP·  000001     012·XFDSMR=·000007
ST.HRL·  000010      006 TBIT·  =·****** GX·     VEC1MX=·000375 G·       WORD0 =·000000          XGTSRE=·000014
ST.LEN·  000044      006 TDABLK=·000004 G        VEC2MX=·000375 G·       WORD1 =·000002·         XHITSK=·000011
ST.QRY·  000002·     006 TDABMX=·007764 G        VEC3MX=·000125 G·       WORD2·=·000004          XHLMER=·000002
ST.QSZ·  000034      006 TDAMAD=·007760 G        VIBCUT=·000036 G·       WORD3 =·000006          XHOTSK=·000010
ST.SCH·  000040      006 TDBBLK=·000003 G        VI0MSZ=·000400 G·       WORD4 =·000010          XMSCHE=·000000
ST.UHL·  000004      006 TDBCLS=·000005 G        VI1MSZ=·000400 G·       WORD5 =·000012·         XQTS   =·000003
ST.XLT·  000014      006 TDBMAD=·007775 G        VI2MSZ=·000400 G·       WORD6 =·000014          XQT0   =·000001
SU.DBU=·000004           TDBOSZ=·000074 G        VI3MSZ=·000400 G·       WORD7 =·000016          XSULOA=·000005
SU.DON=·000006           TDCADR=·****** GX·

. ABS.  000000     000
        000054     001
SRCOFF·  000122·    002·
FDSCOF·  000010     003
SUSOFF·  000012·    004
DHROFF·  000012·    005
STTOFF·  000044     006
QSPLOF·  000014     007
BSTOFF·  000772·    010
FNOFFS·  000044     011
WNODOF·  000010     012·
DNODOF·  000010     013
FMDATA·  006214     014
FMCODE·  006462·    015
ERRORS·DETECTED:  0

VIRTUAL·MEMORY·USED:  4314 WORDS·  (·17 PAGES)
DYNAMIC·MEMORY:  4916 WORDS·  ( 18 PAGES)
ELAPSED·TIME:  00:01:01
FMCDE1,FMCDE1/NL:ME:BEX/-SP=[20,1]P,M,QTSIZE,FMCDE1
```

```
TASK    NAME    : QT1
PARTITION NAME  : HSTSPR
IDENTIFICATION  : 0736
TASK  UIC       : [20,3]
STACK    LIMITS: 000212 001007 000576 00382.
PRG XFR ADDRESS: 001754
TOTAL ADDRESS WINDOWS: 2.
TASK  IMAGE  SIZE   : 26560. WORDS
TASK ADDRESS LIMITS: 000000 147517
R-W DISK BLK LIMITS: 000002 000151 000150 00104.

*** ROOT SEGMENT: QT1


R/W MEM  LIMITS: 000000 147517 147520 53072.
DISK BLK LIMITS: 000002 000151 000150 00104.


MEMORY ALLOCATION SYNOPSIS:

SECTION                                          TITLE  IDENT  FILE
-------                                          -----  -----  ----
. BLK.:(RW,I,LCL,REL,CON) 001010 000744 00484.
                          001010 000054 00044.   FLU            FMCDE1.OBJ;1
BSTOFF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000.   FSA            QT1.OBJ;1
                          000000 000000 00000.   FLU            FMCDE1.OBJ;1
CODE  :(RW,I,LCL,REL,CON) 001754 007562 03954.
                          001754 007562 03954.   FSA            QT1.OBJ;1
DATA  :(RW,I,LCL,REL,CON) 011536 021570 09080.
                          011536 021570 09080.   FSA            QT1.OBJ;1
DHROFF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000.   FSA            QT1.OBJ;1
                          000000 000000 00000.   FLU            FMCDE1.OBJ;1
DNODOF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000.   FSA            QT1.OBJ;1
                          000000 000000 00000.   FLU            FMCDE1.OBJ;1
EMA   :(RW,I,LCL,REL,CON) 033326 035010 14856.
                          033326 035010 14856.   FSA            QT1.OBJ;1
FDSCOF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000.   FSA            QT1.OBJ;1
                          000000 000000 00000.   FLU            FMCDE1.OBJ;1
FMCODE:(RW,I,LCL,REL,CON) 070336 006462 03378.
                          070336 006462 03378.   FLU            FMCDE1.OBJ;1
FMDATA:(RW,I,LCL,REL,CON) 077020 006214 03212.
                          077020 006214 03212.   FLU            FMCDE1.OBJ;1
FNOFFS:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000.   FSA            QT1.OBJ;1
                          000000 000000 00000.   FLU            FMCDE1.OBJ;1
MSGOUT:(RW,I,LCL,REL,CON) 105234 001214 00652.
                          105234 001214 00652.   MESSAG        MSGOUT.OBJ;1
NODES :(RW,I,LCL,REL,CON) 106450 040366 16630.
                          106450 040366 16630.   FSA            QT1.OBJ;1
QSPLOF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000.   FSA            QT1.OBJ;1
```

```
                          000000 000000 00000. FLU          FMCDE1.OBJ;1
SRCOFF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000. FSA          QT1.OBJ;1
                          000000 000000 00000. FLU          FMCDE1.OBJ;1
STTOFF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000. FSA          QT1.OBJ;1
                          000000 000000 00000. FLU          FMCDE1.OBJ;1
SUSOFF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000. FSA          QT1.OBJ;1
                          000000 000000 00000. FLU          FMCDE1.OBJ;1
TRNOFF:(RW,I,LCL,REL,CON) 147036 000240 00160.
                          147036 000240 00160. FSA          QT1.OBJ;1
WNODOF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000. FSA          QT1.OBJ;1
                          000000 000000 00000. FLU          FMCDE1.OBJ;1
$DPB$$:(RW,I,LCL,REL,CON) 147276 000030 00024.
                          147276 000030 00024. FSA          QT1.OBJ;1
$$FSR1:(RW,D,GBL,REL,OVR) 147326 000000 00000.
                          147326 000000 00000. FSA          QT1.OBJ;1
$$FSR2:(RW,D,GBL,REL,CON) 147326 000104 00068.
$$RESL:(RW,I,LCL,REL,CON) 147432 000066 00054.
$$RESM:(RW,I,LCL,REL,CON) 160000 015600 07040.


GLOBAL SYMBOLS:

ASTKMX 000400    EMXDTF 010000    FCSERR 010600-R  MATRP2 074636-R  N.POS  000010    S1BIT  001000    T.NBAS 000002
AVELGT 000025    EMXEXF 000001    FLIXMK 177740    MIXFMC 104560-R  N.VEC  000014    S2BIT  002000    T.NDEF 000000
BITNIB 100000    EMXFDC 002000    FLIXSZ 000040    MOVTDC 011430-R  PAR2   013324-R  S3BIT  004000    T.SBY1 000000
CBIT   010000    EMXMCD 000002    FLUCLS 000001    MSGOUT 105436-R  PSTKMX 000024    TAEBIT 020000    T.SBY2 000001
CPIXMK 177770    EMXMCF 001000    FLUCUT 000016    NDBCLS 000006    QNDCNT 001000    TBIT   004000    T.SBY3 000002
CPIXSZ 000010    EMXMSZ 016000    FLUMD  070336-R  NIXCNT 000020    QRYSZ  002000    TDABLK 000004    T.STAD 000002
DELTIM 147036-R  EMXMTV 040000    FMEPSZ 000400    NNMASK 177760    SD.FSA 000010    TDABMX 007764    T.TRAN 000000
DIRERR 010540-R  EMXNB1 000002    FMIXSZ 000001    NODERR 010662-R  SD.NPS 000016    TDAMAD 007760    T.TYPW 000004
EIXCNT 000010    EMXNB2 000004    FMNPSZ 002260    NPECNT 000144    SD.SEC 000012    TDBBLK 000003    VEC1MX 000375
ELSBIT 010000    EMXNFD 000400    FNPOSZ 000400    NPEVSZ 000043    SD.TIC 000014    TDBCLS 000005    VEC2MX 000375
ELSCLS 000007    EMXNSQ 100000    FNPSZ  005734    NP1MSZ 010000    SECBUF 147244-R  TDBMAD 007775    VEC3MX 000125
ELSMSK 177761    EMXNVD 001000    FSAADD 013342-R  NP2MSZ 036000    SEG1   000000    TDBOSZ 000074    VI     033342-R
EMA    034336-R  EMXSZF 002000    FSAOFF 013344-R  NP2OSZ 000153    SEG2   000002    TDCADR 013354-R  VIBCUT 000036
EMACLS 000003    EMXTRL 010000    GTIM1  147256-R  NP3MSZ 000524    SEG3   000004    TDCBLK 000010    VI0MSZ 000400
EMACUT 001700    EMXVDC 004000    GVMASK 070714-R  NP3OSZ 000125    SSBIT  004000    TDCMAX 013352-R  VI1MSZ 000400
EMAFDB 012640-R  EMXVVV 000001    HSTKMX 000100    NXTADD 013346-R  STPBLK 000524    TDCMSZ 004000    VI2MSZ 000400
EMAMSZ 016000    EMXZNF 004000    INTDEF 013330-R  NXTOFF 013350-R  ST$CNG 120000    TDCT   013366-R  VI3MSZ 000400
EMBCLS 000004    EMX0VD 000200    IWBIT  004000    N.BACK 000002    ST$INR 060000    TRMCUT 000040    VMASK  033326-R
EMBCUT 001130    ENCNT  000004    JMPBIT 004000    N.ELSE 000006    ST$INX 040000    TSTKMX 000400    VVEC   011566-R
EMBMSZ 004400    EOBIT  010000    LOGCLS 000002    N.FSA  000004    ST$JSQ 100000    TXTBIT 010000    WRTTDC 010462-R
EMCMSZ 001000    ERROR2 010632-R  LOWOFF 013340-R  N.FWD  000000    ST$MAT 160000    T.JSBY 000000    $EDMSG 105664-R
EMXCHF 020000    EXIT   010734-R  L$STAT 000006    N.LGT  000012    ST$SEQ 140000    T.MATC 000000
```

*** TASK BUILDER STATISTICS:

TOTAL WORK FILE REFERENCES: 27982.
WORK FILE READS: 0.
WORK FILE WRITES: 0.
SIZE OF CORE POOL: 6634. WORDS (25. PAGES)
SIZE OF WORK FILE: 2816. WORDS (11. PAGES)

ELAPSED TIME:00:00:18

```
                                ; QTSIZE.MAC    "QUERY TRANSLATORS BUFFER SIZE CONFIGUATION FILE"
                                ;       THIS FILE CONTROLS THE SIZE OF ALL BUFFERS THAT CAN VARY
                                ;       IN SIZE DUE TO THE AMOUNT OF QUERIES OR OTHER SUCH PARAMETERS
                                ;       THAT WOULD BE CHARACTERISTIC OF A SITE.  THESE BUFFERS ARE
                                ;       IN QT0, QT1, QT2, OR QT3.
                                ;
                                ;---QT3---BUFFERS-----------------------------------------------------------
                                ;
000125                          VEC3MX==85.                             ;MAXIMUM # CWP'S IN BATCH
000025                          AVELGT==3*2*7/2                         ;AVERAGE CWP SIZE IN TDCT (BYTES)
000400                          VI3MSZ==VEC3MX+2/256.+1*256.            ;SIZE OF VI (NEAREST BLOCK IN BYTES)
001000                          EMCMSZ==VEC3MX/51.+1*256.               ;SIZE OF EMC (NEAREST BLOCK)
004000                          TDCMSZ==VEC3MX*AVELGT+8./N.BUFW+1*N.BUFW;SIZE OF TDCT
000010                          TDCBLK==TDCMSZ/256.                     ;VIRTUAL BLOCK SIZE OF TDCT
000524                          NP3MSZ==VEC3MX*4                        ;SIZE OF NODE POOL
000125                          NP3OSZ==VEC3MX                          ;SIZE OF NODE POOL OVERFLOW AREA
                                ;
                                ;-----QT2----BUFFERS-------------------------------------------------------
                                ;
000375                          VEC2MX==253.                            ;MAXIMUM # VECTORS
000400                          VI2MSZ==VEC2MX+2/256.+1*256.            ;SIZE OF VI
004400                          EMBMSZ==9.*256.                         ;SIZE OF EMB
000003                          TDBBLK==3                               ;# BLOCKS (N.BUFW) IN TDCT BUFFER
000074                          TDBOSZ==3*20.                           ;SIZE OF TDCT BUFFER OVERFLOW
007775                          TDBMAD==4093.                           ;MAXIMUM ADDRESS (48 BITS) IN TDCT
000020                          NIXCNT==16.                             ;# INDEX POINTERS FOR NORMAL NODES
177760                          NNMASK==177760                          ;MASK FOR NORMAL INDEX
000010                          EIXCNT==8.                              ;# INDEX PTRS FOR ELSE NODES
177761                          ELSMSK==177761                          ;MASK FOR ELSE INDEX
000004                          ENCNT==4                                ;# ENTRIES TO USE FOR HASH
036000                          NP2MSZ==256.*60.                        ;SIZE OF NODE POOL
000153                          NP2OSZ==7+100.                          ;SIZE OF NODE POOL OVERFLOW AREAR
                                ;
                                ;------QT1-----BUFFERS------------------------------------------------------
                                ;
000375                          VEC1MX==253.                            ;MAX. # TERMS IN FSA-A
000400                          VI1MSZ==VEC2MX+2/256.+1*256.            ;SIZE OF VI
000004                          TDABLK==4                               ;# BLOCKS (N.BUFW) IN TDCT BUFFER
007764                          TDABMX==<TDABLK*<N.BUFW-4>>+4
016000                          EMAMSZ==7.*4*256.                       ;SIZE OF EMA
007760                          TDAMAD==340.*12.                        ;MAX. ADDRESS IN TDCT
010000                          NP1MSZ==4096.                           ;SIZE OF NORMAL NODE POOL
000144                          NPECNT==100.                            ;# NODES IN ELSE POOL
000043                          NPEVSZ==35.                             ;# VECTORS IN ELSE NODES
                                ;
                                ;-----FLU---MODIFIER----BUFFERS---------------------------------------------
                                ;
002260                          FMNPSZ==1200.                           ;SIZE OF NORMAL FLU MOD NODE POOL
000400                          FMEPSZ==256.                            ;SIZE OF ELSE FLU MOD NODE POOL
                                ;
                                ;------QT0------BUFFERS-----------------------------------------------------
                                ;
002000                          QRYSZ==N.BUFW                           ;SIZE OF QUERY BUFFER
000040                          FLIXSZ==32.                             ;# INDEX PTRS FOR TERMS
177740                          FLIXMK==177740                          ;MASK FOR TERM INDEX
000010                          CPIXSZ==8.                              ;# INDEX PTRS FOR CWP'S
177770                          CPIXMK==177770                          ;MASK FOR CWP INDEX
```

```
FSA-B TRANSLATOR (QT2)  MACRO M1110  27-MAR-80 13:11  PAGE 1(4)
SUBROUTINE DELTIM
```

```
     58          000001                 FMIXSZ==1                                ;# INDEX PTRS FOR FLU-MOD'S
     59          005734                 FNPSZ==3036.                             ;FLU NODE POOL SIZE
     60          000400                 FNPOSZ==256.                             ;SIZE OF POOL OVERFLOW AREA
     61          000400                 TSTKMX==256.                             ;MAX. # TOKENS TO BE PUSHED
     62          000400                 ASTKMX==256.                             ;MAX. # ARGUMENTS TO BE PUSHED
     63          000024                 PSTKMX==20.                              ;MAX. # PROX NODES IN A CHAIN
     64          000100                 HSTKMX==64.                              ;MAX. NESTING OF QUERY
     65          001000                 QNDCNT==512.                             ;# NODES IN LOGIC POOL
     66          000400                 VI0MSZ==VI1MSZ                           ;ASSUME MAX VI IS IN QT1
     67                                 .IIF LT,VI0MSZ-VI2MSZ,VI0MSZ==VI2MSZ    ; IF NOT, RESET QT0'S TO QT2'S VI SIZE
     68          016000                 EMXMSZ==EMAMSZ                           ;ASSUME MAX EMX IS IT QT1
     69                                 .IIF LT,EMXMSZ-EMBMSZ,EMXMSZ==EMBMSZ    ; IF NOT, RESET TO QT2'S
     70                                 ;
     71                                 ;-----ERROR AND CLOSE CODES---------------------------------------
     72                                 ;
     73          000040                 TRMCUT==32.      ;TERM CUT-OFF
     74          000016                 FLUCUT==14.      ;FLU CUT-OFF
     75          001700                 EMACUT==TRMCUT*30.       ;EMA CUT-OFF
     76          000036                 VIBCUT==30.              ;VIB CUT-OFF
     77          001130                 EMBCUT==VIBCUT*20.       ;EMB CUT-OFF
     78                                 ;
     79          000001                 FLUCLS==1        ;CLOSED DUE TO FLU COUNT
     80          000002                 LOGCLS==2        ;CLOSED DUE TO LOGIC COUNT (QLB,SDLB,QEX)
     81          000003                 EMACLS==3        ;CLOSED DUE TO EMA SIZE
     82          000004                 EMBCLS==4        ;CLOSED DUE TO EMB SIZE
     83          000005                 TDBCLS==5        ;CLOSED DUE TO TDCTB SIZE
     84          000006                 NDBCLS==6        ;CLOSED DUE TO QT2 NODE POOL SIZE
     85          000007                 ELSCLS==7        ;CLOSED DUE TO OTHER CONDITIONS (FLU POOL,ETC)
     86                                 ;
```

```
                        .TITLE  FSA-B TRANSLATOR (QT2)
                        .SBTTL
                        .SBTTL  MODULE NAME:     QUERY TRANSLATOR (FSA-B)
                        .SBTTL  AUTHOR:
                        .SBTTL
                        .SBTTL
                        .SBTTL
                        .SBTTL
                        .SBTTL  RELEASE DATE:    7 MAY 1979
                        .SBTTL  RELEASE NUMBER: SL120024
                        .SBTTL
                        .SBTTL  ***********
                        .SBTTL  FSA-B TRANSLATOR (QT2)
                        .SBTTL  ***********

                ;
                ;   QT2 READS FROM DISK THE INTERIM DATA STRUCTURES (EMATRIX.EMB)
                ;   GENERATED BY QT0.  THE 4-WORD FILE DESCRIPTOR (FDSC) OF EMATRIX.EMB
                ;   IS SENT TO QT2 BY QTS.
                ;   LAYOUT OF EMATRIX.EMB IS AS FOLLOWS:
                ;
                ;       VIRTUAL BLOCK 1          : VI
                ;       VIRTUAL BLOCKS 2-12,     : EMB
                ;
                ;   BASED ON EMB AND VI, QT2 BUILDS THE "TERM DETECTOR
                ;   CONTROL TABLE" (TDCT) FOR FSA-B, WRITES THIS STRUCTURE TO DISK
                ;   (FILE NAME = TDCTB    .FSA), SENDS TO QTS THE 4-WORD FDSC OF
                ;   THE FILE AND EXITS.
                ;
                ;   ASSEMBLY PARAMETERS:
                ;       MAC>QT2=P,M,T,QT2
                ;
                ;   TASK BUILD PARAMETERS:
                ;       MCR>TKB @QT2TKB.CMD
                ;       TKB>QT2,LP=QT2,MSGOUT,FMCDE2
                ;       TKB>/
                ;       TKB>TASK=QT2
                ;       TKB>/
                ;
```

```
                                        .SBTTL  TASK BUFFERS
000000                                  .PSECT  DATA
                                ;
000000                          IOST:   .BLKW   2                       ;IO STATUS BUFFER
000004  000000                  PARBUF: .WORD   0                       ;.XQIO PARAM. 1 - BUFFER ADDS
000006  000000                          .WORD   0                       ;    "     "    2 - BUFFER LENGTH (BYTES)
000010  000000                          .WORD   0                       ;    "     "    3
000012  000000                          .WORD   0                       ;    "     "    4 - VIRTUAL DISK ADDS (HI)
000014  000001                          .WORD   1                       ;    "     "    5 -    "       "    "  (LO)
000016  066563  000000          QTS:    .RAD50  /QTS   /
000022                          RECBUF: .BLKW   2                       ;SENDING TASK'S NAME
000026                          SNDBUF: .BLKW   13.
                                ;
                                        .MCALL  MOUT$S,EXIT$S
                                        .MCALL  GTIM$S,RCVD$C,SDAT$C,RSUM$C
                                        .MCALL  FDBDF$,FDAT$A,FDRC$A,FDBK$A,FDOP$A,FDOP$R,FSRSZ$
                                        .MCALL  FINIT$,OFNB$W,OPEN$R,CLOSE$
                                        .MCALL  NMBLK$,FDBK$R,FDAT$R,WRITE$,WAIT$
                                ;
                                ;   FDB FOR EMATRIX AND TDCTB
                                ;
000060                          EMAFDB::FDBDF$
000220                                  FDAT$A  R,VAR,FD,BLK,,50.
000220                                  FDRC$A  FD,RWM
000220                                  FDBK$A  EMALGT,512,,,EFN,2,IOST
000220                                  FDOP$A  LUNFIL,,EMADNB
000220                          EMADNB: NMBLK$  EMATRIX,EMA,,SY,0
000256                                  FSRSZ$  0,,DATA
```

```
                                        .SBTTL  TASK IMMEDIATES AND VARIABLES
000256                                  .PSECT  DATA
                                ;
        000004          LUNFIL  =       4                       ;LUN DEFINITION FOR FCS
        000002          EFN.2   =       2                       ;EVENT FLAG TWO
        001000          BLOCK   =       512.                    ;BLOCK LENGTH (BYTES)
                                ;
        000000          CCW     =       0       ;CONTROL WORD (GENVEC)
        000002          CN2     =       2       ;NIBBLE 2 (GENVEC)
        000004          CN1     =       4       ;NIBBLE 1 (GENVEC) <USE TO GENERATE VECTOR>
                                ;
000256  000000          NPHI::  .WORD   0       ;NEXT NODE POOL HEADER INDEX
000260  000000          PHI::   .WORD   0       ;THIS NODE POOL HEADER INDEX
000262  000124'          PRIOR:  .WORD   ELSDFN  ;PRIOR ELSE NODE ADR
000264  000000          REESET: .WORD   0       ;TRANSFER ADR OF PRIOR FSA STATE
000266  000000          NMLCNT: .WORD   0       ;NORMAL ENTRY COUNT (EN)
000270  000000          DCNT:   .WORD   0       ;DON'T CARE ENTRY COUNT (EN)
000272  000000          TVCNT:  .WORD   0       ;TRAILING VLDC ENTRY COUNT (EN)
000274  000000          CTRSUM: .WORD   0       ;CONTROL SUMMARY
000276  000000          TASUM:  .WORD   0       ;TRY AGAIN SUMMARY
000300  000000          NMLSUM: .WORD   0       ;NORMAL SUMMARY
000302  000000          TAFLG:  .WORD   0       ;TRY AGAIN ENTRY FLAG
000304  000000          FAILST: .WORD   0       ;ADR OF FAIL STATE FOR ALL FLDC VECTOR
                                ;
        000524          STPBK1  =  <N.BUFW-4>/3                 ;# STATES PER BLOCK 1
        000014          LOC2    =  2*6                          ;FSA LOC 2 OFFSET
        000030          LOC4    =  4*6                          ;FSA LOC 4 OFFSET
        000006          BEGFSA  =  6                            ;1ST FREE FSA LOC
                                ;
000306  000000          CURBLK: .WORD   0                       ;CURRENT BLOCK OFFSET
000310  001776          BUFMAX: .WORD   <<TDBBLK*N.BUFW>-4>/3   ;CURRENT MAX FSA LOC IN BUFFER
000312  000616'          TDCADR::.WORD   TDCT                    ;CURRENT VIRTUAL TDCT BUF START
000314  000000          LOWADR::.WORD   0                       ;LOW FSA LOCATION
                                ;
                                ; NODE HEADER OFFSETS
                                ;
                                ;XXXXX  =  0             ;FORWARD POINTER
        000002          N.BACK  ==  2           ;BACKWARD POINTER
        000004          N.PHI   ==  N.BACK+2    ;NODE POOL HEADER INDEX (BYTE)
        000005          N.FLG   ==  N.PHI+1     ;NODE FLAGS (BYTE) (N1P,N2P,ENP)
        000005          N.FMT   ==  N.FLG       ;FLU-MOD TYPE (BYTE) (FME,FMN1,FMN2)
        000006          N.LGT   ==  N.PHI+2     ;NODE VECTOR SIZE
        000010          N.FSA   ==  N.LGT+2     ;NODE FSA LOC
        000012          N.SUM   ==  N.FSA+2     ;NODE SUMMARY (XFER BITS) (N1P,N2P)
        000012          N.EID   ==  N.SUM       ;ELSE ID (BYTE) (ENP)
        000013          N.TVCT  ==  N.EID+1     ;TRAILING VLDC COUNT (BYTE) (ENP)
        000012          N.FMHS  ==  N.SUM       ;HEADER SIZE OF FLU-MOD NODES (FME,FMN1,FMN2)
        000014          N.ELSE  ==  N.SUM+2     ;ELSE NODE ADDRESS OR ID FOR NODE (N1P,N2P)
        000014          N.CIDP  ==  N.ELSE      ;CID FSA LOC STORAGE (ENP)
        000016          N.NHS   ==  N.ELSE+2    ;NODE HEADER SIZE (N1P,N2P,ENP)
                                ;
                                ; N.FLG VALUES
                                ;
        000200          NDSKIP  =  BIT7         ;SKIP NODE
        000100          NDTAD   =  BIT6         ;TRY AGAIN DIFFER
        000040          NDEON   =  BIT5         ;ELSE OVERRIDE FOR NUMERICS
        000020          NDSE    =  BIT4         ;SINGLE EXIT
```

```
126          000010                  NDJSC   =  BIT3          ;JUMP/SEQ
127          000004                  NDCID   =  BIT2          ;CHANGE INTER-WORD DEFAULT
128                                  ;
129                                  ; N.PHI VALUES
130                                  ;
131          000000                  N1P     == 0     ;NIBBLE 1 NODE
132          000002                  N2P     == 2     ;NIBBLE 2 NODE
133          000004                  ENP     == 4     ;ELSE NODE
134          000006                  FME     == 6     ;FLU-MOD ELSE NODE
135          000010                  FMN1    == 10    ;FLU-MOD NIBBLE 1 NODE
136          000012                  FMN2    == 12    ;FLU-MOD NIBBLE 2 NODE
137                                  ;
```

```
                                        ;
                                        ;  ERROR MESSAGE STRING DEFINITIONS, PRINTED ON TI VIA MOUT$S,
                                        ;
000316  000041                  MSG1:   .WORD   LN1E-LN1                ;LENGTH OF FORMAT STRING
000320  000342'                         .WORD   LN1                     ;ADDS OF FORMAT STRING
000322  000040                  MSG2:   .WORD   LN2E-LN2
000324  000403'                         .WORD   LN2
000326  000032                  MSG4:   .WORD   LN4E-LN4
000330  000443'                         .WORD   LN4
000332  000042                  MSG5:   .WORD   LN5E-LN5
000334  000475'                         .WORD   LN5
000336  000034                  MSG6:   .WORD   LN6E-LN6
000340  000537'                         .WORD   LN6
                                        ;
                                        ;   FORMAT STRINGS
                                        ;
000342      104     111     122 LN1:    .ASCIZ  /DIR. ERROR, PC = %O, $DSW = %D/
000403                          LN1E:
000403      106     103     123 LN2:    .ASCIZ  /FCS  ERROR, PC = %O, ERR = %D/
000443                          LN2E:
000443      116     117     104 LN4:    .ASCIZ  /NODE OVERFLOW, TYPE = %D/
000475                          LN4E:
000475      126     105     103 LN5:    .ASCIZ  /VECTORS EXCEEDED MAX ALLOWED, %D/
000537                          LN5E:
000537      124     104     103 LN6:    .ASCIZ  /TDCT OVERFLOW: BLOCKS = %D/
000573                          LN6E:
                                        .EVEN
                                        ;
                                        ;   PARAMETERS
                                        ;
000574  000000                  PAR1::  .WORD   0                       ;PC
000576  000000                  PAR2::  .WORD   0                       ;$DSW OR F.ERR
```

```
                                        .SBTTL  TERM DETECTOR CONTROL TABLE BUFFER
                                ;
000600  000003                INTDEF::.WORD   3                       ;INTERWORD DEFAULT ADDS
000602  000007                NXTADD::.WORD   BEGFSA+1                ;NEXT AVAILABLE STATE ADDRESS
000604  000001                FIRST:  .WORD   1                       ;FIRST STATE FLAG
                                ;
                                ;   TDCT BUFFER
                                ;
000606      064     103         TDCBUF:.ASCII  /4C/
000610  000000                TDCLGT: .WORD   0                       ;BLOCK LENGTH OF TDCT-A
000612                                  .BLKW   2                       ;FILLER TO ALIGN 3-WORD TDCT ENTRIES
                                                                        ;ON 1024. WORD BLOCK BOUNDARIES
                                ;   TDCT STATE WORD 0
                                ;
000616  000002                TDCT:   .WORD   2                       ;NEW ELSE DEF, ADD'S
000620  000001                        .WORD   1                       ;NEXT STATE ADD'S
000622  130000                        .WORD   130000                  ;STATE : CHANGE ELSE DEF,
                                ;
                                ;   TDCT STATE WORD 1
                                ;
000624  000004                         .WORD   4                       ;NEW INT, DEF, ADD'S
000626  000002                         .WORD   2                       ;NEXT STATE ADD'S
000630  124000                         .WORD   124000                  ;STATE : CHANGE INT, DEF
                                ;
                                ;   TDCT STATE WORD 2 (ELSE DEF, STATE WORD)
                                ;
000632  000000                         .WORD   0                       ;MATCH BYTE (INTERWORD CHAR.)
000634  000006                         .WORD   6                       ;NEXT STATE ADD'S
000636  114000                         .WORD   114000                  ;STATE : JUMP/SEQUENTIAL
                                ;
                                ;   TDCT STATE WORD 3 (INTERWORD RESET STATE WORD)
000640  000004                         .WORD   4                       ;OLD INTERWORD DEF, ADR
000642  000004                         .WORD   4                       ;NEXT STATE ADR
000644  124000                         .WORD   124000                  ;STATE : CHANGE INTERWORD DEF,
                                ;
                                ;    TDCT STATE WORD 4 (INTERWORD DEF, STATE WORD)
                                ;
000646  000002                         .WORD   2                       ;NEW ELSE DEF, ADDS
000650  000006                         .WORD   6                       ;NEXT STATE ADD'S
000652  130000                         .WORD   130000                  ;STATE : CHANGE ELSE DEF,
                                ;
                                ;   TDCT STATE WORD 5 (RESERVED STATE-CHANGE DEF,)
                                ;
000654  000000  000000  000000          .WORD   0,0,0
                                ;
                                ;   THE REST OF THE TDCT IS FILLED IN DURING TRANSLATION
                                ;
000662                                  .BLKW   TDBBLK*N.BUFW-22.       ;TDCT BUFFER SIZE
014606                          TDCEND: .BLKW   TDBOSZ                  ;TDCT OVERFLOW AREA
                                ;
                                        .SBTTL  E-MATRIX BUFFERS
                                ;
014776  000000  000000  000004  VMASK:: .WORD   0,0,ENP,0,BEGFSA
                                ;
                                ; START OF EMATRIX FILE EMATRIX.EMB
                                ;
015010  000000                  EMALGT: .WORD   0                       ;BLOCK LENGTH OF EMA
```

```
    229 015012  000000                  VILGT:  .WORD   0                       ;NO. OF ENTRIES IN VI
    230 015014                          VI:     .BLKW   VI2MSZ-2                ;INITIAL VECTOR
    231 016010                          EMA::   .BLKW   EMBMSZ                  ;EMATRIX
    232                                 ;
    233 027010                          VVEC::  .BLKW   VI2MSZ+5                ;TEMP VECTOR BUFFER
    234 000000                                  .PSECT  NODES
    235                                 ;
    236                                 ; NODE POOL AND INDICES
    237                                 ;
    238 000000                          ELSIDX: .BLKW   EIXCNT                  ;ELSE NODE INDICES
    239 000020                          N1IDX:  .BLKW   NIXCNT                  ;NIBBLE 1 INDICES
    240 000060                          N2IDX:  .BLKW   NIXCNT                  ;NIBBLE 2 INDICES
    241                                 ;
    242 000120  000146'                 NNPE::  .WORD   NP                      ;NEXT AVAILABLE NODE ADR
    243 000122  000146'                 SNPE:   .WORD   NP                      ;NODE ADR TO RESUME SCAN
    244                                 ;
    245 000124  000000  000000  000004  ELSDFN::.WORD   0,0,ENP,0,2,0,ELSDFN,0  ;ELSE DEFAULT NODE
    246 000144  000001                  ELSEID: .WORD   1                       ;ELSE ID
    247                                 ;
    248 000146                          NP::    .BLKW   NP2MSZ                  ;NODE POOL
    249 074146                          NPE::   .BLKW   NP2OSZ                  ;NODE POOL OVERFLOW AREA
```

```
                                        .SBTTL  MAIN PROCESSING LOOP
000000                                  .PSECT  CODE
                                ;
                                ;   RECEIVE PACKET FROM QTS
                                ;
000000                          START:
000000                                  RCVD$C  QTS,RECBUF,CODE,DIRERR
000014                                  FINIT$
                                ;
                                ;   READ EMATRIX.EMB FILE
                                ;
000020  012700  000060'                 MOV     #EMAFDB,R0              ;R0->FDB
                                ;
                                ;   ZERO OUT FILENAME BLOCK
                                ;
000024  062700  000102                  ADD     #F.FNB,R0               ;R0->FNB
000030  012701  000036                  MOV     #S.FNB,R1               ;R1 = BYTE LENGTH OF FNB
000034  006201                          ASR     R1                      ;R1 = WORD LENGTH OF FNB
000036  005020                  5$:     CLR     (R0)+
000040  077102                          SOB     R1,5$
                                ;
                                ;   LOAD EMA'S FID, DEVICE NAME AND UNIT NUMBER INTO FNB
                                ;
000042  012700  000060'                 MOV     #EMAFDB,R0              ;R0-> FDB
000046  062700  000102                  ADD     #F.FNB,R0               ;R0-> FNB
000052  012701  000026'                 MOV     #SNDBUF,R1              ;R1-> DATA RECEIVED FROM QTS
000056  016160  000000  000000          MOV     N.FID(R1),N.FID(R0)     ;FID
000064  016160  000002  000002          MOV     N.FID+2(R1),N.FID+2(R0)
000072  016160  000004  000032          MOV     N.FID+4(R1),N.DVNM(R0)  ;DEVICE NAME
000100  016160  000016  000034          MOV     N.FVER(R1),N.UNIT(R0)   ;DEVICE UNIT NUMBER
                                ;
                                ;   OPEN EMATRIX.EMB FOR READ
                                ;
000106                                  OPEN$R  #EMAFDB                 ;OPEN FILE FOR READ
000124  103010                          BCC     1$
000126  116001  000052                  MOVB    F.ERR(R0),R1            ;R1 = FCS ERROR
000132  010167  000576'                 MOV     R1,PAR2
000136                                  CALL    FCSERR
000142  000167  001366                  JMP     EXIT
                                ;
                                ;   READ VI
                                ;
000146  012767  015010' 000004' 1$:     MOV     #EMALGT,PARBUF          ;PAR 1 = BUFFER ADDS
000154  012767  001000  000006'         MOV     #512.,PARBUF+2          ;PAR 2 = BUFFER LENGTH (BYTES)
000162  012701  000000G                 MOV     #IO.RVB,R1              ;R1 = IO FUNCTION CODE
000166  012702  000005                  MOV     #5,R2                   ;R2 = NO. OF QIO PARAMETERS
000172  012703  000004'                 MOV     #PARBUF,R3              ;R3->QIO PARAMETERS
000176                                  CALL    .XQIO
000202  103010                          BCC     2$
000204  116001  000052                  MOVB    F.ERR(R0),R1            ;R1 = IO ERROR
000210  010167  000576'                 MOV     R1,PAR2
000214                                  CALL    FCSERR
000220  000167  001310                  JMP     EXIT
                                ;
                                ;   TEST TO SEE IF THERE IS AN EMA
                                ;
000224  005767  015010'         2$:     TST     EMALGT                  ;BLANK EMA?
```

```
000230  001033                          BNE     4$                      ;BRANCH IF NO
                                ;
                                ;   CLOSE EMA FILE
                                ;
000232                                  CLOSE$  #EMAFDB
                                ;
                                ;   SET UP BLANK TDCTB
                                ;
000242  012700  000662'                 MOV     #TDCT+44,R0
000246  005020                          CLR     (R0)+
000250  005020                          CLR     (R0)+
000252  012720  040000                  MOV     #ST$INX,(R0)+
                                ;
                                ;   WRITE TDCTB
                                ;
000256                                  CALL    OPNTDC                  ;OPEN TDCTB FILE
000262  012702  000001                  MOV     #1,R2
000266  012703  000606'                 MOV     #TDCBUF,R3
000272                                  CALL    WRTTDC                  ;CREATE TDCTB
                                ;
                                ;   SET UP SNDBUF FOR QTS
                                ;
000276  005067  000000C                 CLR     SNDBUF+SD.SEC
000302  005067  000000C                 CLR     SNDBUF+SD.TIC
000306  016767  000602' 000000C         MOV     NXTADD,SNDBUF+SD.FSA
000314  000167  000406                  JMP     QT2.7
                                ;
                                ;   READ EMA
                                ;
000320  012767  016010' 000004' 4$:     MOV     #EMA,PARBUF             ;PAR 1 = BUFFER ADDS
000326  016701  015010'                 MOV     EMALGT,R1               ;R1 = EMA LENGTH IN BLOCKS
000332  070127  001000                  MUL     #512.,R1                ;R1 = BYTE LENGTH
000336  020127  011000                  CMP     R1,#EMBMSZ*2            ;DON'T LET OVERFLOW
000342  101020                          BHI     8$
000344  010167  000006'                 MOV     R1,PARBUF+2             ;PAR 2 = FILE LENGTH
000350  012767  000002  000014'         MOV     #2,PARBUF+10            ;START READ AT VIRTUAL BLOCK 2
000356  012701  000000G                 MOV     #IO.RVB,R1              ;R1 = IO FUNCTION CODE
000362  012702  000005                  MOV     #5,R2                   ;R2 = NO. OF QIO PARAM'S
000366  012703  000004'                 MOV     #PARBUF,R3              ;R3-> PARAMETERES
000372                                  CALL    .XQIO
000376  103010                          BCC     3$
000400  116001  000052                  MOVB    F.ERR(R0),R1            ;R1 = IO ERROR
000404  010167  000576'         8$:     MOV     R1,PAR2
000410                                  CALL    FCSERR
000414  000167  001114                  JMP     EXIT
                                ;
                                ;   CLOSE FILE
                                ;
000420                          3$:     CLOSE$  R0
                                ;
                                ; OPEN FILE FOR TDCTB,FSA
                                ;
000424                                  CALL    OPNTDC
                                ;
                                ;   INITIALIZE TRANSLATION PROCESS
                                ;
000430                          QT2.1:  GTIM$S  #GTIM1                  ;GET TIME PARAM'S AT START
```

```
365 000442  012703  014776'                     MOV     #VMASK,R3               ;R3->VMASK = VI
366 000446  016763  015012' 000006              MOV     VILGT,N.LGT(R3)         ;SET VECTOR LENGTH
367 000454  005063  000012                      CLR     N.SUM(R3)               ;INIT SUMMARY
368 000460  012763  000124' 000014              MOV     #ELSDFN,N.ELSE(R3)      ;SET ELSE TO DEFAULT
369 000466  004767  003106                      JSR     PC,EN                   ;CREATE ELSE & NML NODE
370 000472  005067  000604'                     CLR     FIRST
371 000476  026767  000122' 000120' 1$:        CMP     SNPE,NNPE               ;ANY MORE NODES?
372 000504  001437                            BEQ     QT2.3                   ;NO: FINISHED
373 000506  016703  000122'                     MOV     SNPE,R3                 ;GET NEXT NODE
374 000512  016301  000006                      MOV     N.LGT(R3),R1            ;CALCULATE NODE SIZE
375 000516  006301                            ASL     R1
376 000520  062701  000016                      ADD     #N.NHS,R1
377 000524  060167  000122'                     ADD     R1,SNPE                 ;NEXT NODE
378 000530  132763  000200  000005            BITB    #NDSKIP,N.FLG(R3)       ;NODE TO BE SKIPPED?
379 000536                                    BON     1$                      ;YES
380 000540  016300  000010                      MOV     N.FSA(R3),R0            ;R1=LOWEST FSA ADR
381 000544  010067  000314'                     MOV     R0,LOWADR               ;SAVE IT
382 000550  020027  000354                      CMP     R0,#354
383 000554  001001                            BNE     10$
384 000556  000240                            NOP
385 000560                              10$:
386 000560  004767  000762                      JSR     PC,TDCHK                ;TDCT BUFFER OVERFLOW CHECK
387 000564  116300  000004                      MOVB    N.PHI(R3),R0            ;NODE TYPE
388 000570  004770  000576'                     JSR     PC,@3$(R0)              ;PROCESS ACCORDING TO TYPE
389 000574  000740                            BR      1$
390                                     ;
391 000576  002006'                     3$:     SPLTN1          ;N1P    SPLIT NIBBLE 1 NODE
392 000600  002624'                             SPLTN2          ;N2P    SPLIT NIBBLE 2 NODE
393 000602  001746'                             PRCELS          ;ENP    PROCESS ELSE NODE
394                                     ;
395                                     ;   TRANSLATION PROCESS IS COMPLETED.
396                                     ;
397 000604                              QT2.3:  GTIM$S  #SECBUF
398                                     ;
399                                     ;   WRITE REMAINING TDCT BLOCKS
400                                     ;
401 000616  016702  000602'                     MOV     NXTADD,R2               ;CALCULATE # BLOCKS
402 000622  070227  000003                      MUL     #3,R2
403 000626  162703  000004                      SUB     #4,R3
404 000632  005602                            SBC     R2
405 000634  071227  002000                      DIV     #N.BUFW,R2
406 000640  005703                            TST     R3
407 000642  001401                            BEQ     1$
408 000644  005202                            INC     R2
409 000646  166702  000306'             1$:     SUB     CURBLK,R2               ;R2=# REMAINING BLOCKS
410 000652  012703  000606'                     MOV     #TDCBUF,R3              ;R3=START OF BLOCKS
411 000656  004767  000364                      JSR     PC,WRTTDC               ;WRITE BLOCKS
412                                     ;
413                                     ;   CALCULATE TIME ELAPSED DURING TRANSLATION
414                                     ;
415 000662                              2$:     CALL    DELTIM
416 000666  016767  000206' 000000C            MOV     SECBUF,SNDBUF+SD,SEC    ;LOAD ELAPSED SECONDS INTO SNDBUF
417 000674  016767  000210' 000000C            MOV     SECBUF+2,SNDBUF+SD,TIC  ;  "      "    TICS     "      "
418 000702  016767  000602' 000000C            MOV     NXTADD,SNDBUF+SD,FSA    ;LOAD NUMBER OF FSA STATES
419                                     ;
420 000710  016705  000120'                     MOV     NNPE,R5                 ;STORE NODE POOL SIZE USED
421 000714  162705  000146'                     SUB     #NP,R5
```

```
 422 000720  006205                        ASR     R5
 423 000722  010567  000000C               MOV     R5,SNDBUF+SD.NPS
 424                                ;
 425                                ;   NOTIFY QTS
 426                                ;
 427 000726                         QT2.7:  SDAT$C  QTS,SNDBUF,,CODE
 428 000734  103004                         BCC     1$
 429 000736                                 CALL    DIRERR
 430 000742  000167  000566                 JMP     EXIT
 431 000746                         1$:     RSUM$C  QTS,CODE
 432 000754  103002                         BCC     2$
 433 000756                                 CALL    DIRERR
 434                                ;
 435                                ;   CLOSE FILE
 436                                ;
 437 000762                         2$:     CLOSE$  #EMAFDB
 438 000772                                 EXIT$S                          ;EXIT
```

```
                                        .SBTTL  OPEN TDCTB FILE
                                ;
                                ;   SUBROUTINE TO OPEN FILE FOR TDCTB.FSA
                                ;
                                ;
                                ;     BUILD FNB USING DATA RECEIVED FROM QTS IN SNDBUF
                                ;
001000                          OPNTDC: SAVE    R0,R1
001004  012700  000060'                 MOV     #EMAFDB,R0              ;R0->FDB
001010  062700  000102                  ADD     #F.FNB,R0               ;R0-> FNB
001014  012701  000026'                 MOV     #SNDBUF,R1              ;R1-> DATA RECEIVED FROM QTS
001020  005060  000000                  CLR     N.FID(R0)               ;CLEAR FID
001024  005060  000002                  CLR     N.FID+2(R0)
001030  005060  000004                  CLR     N.FID+4(R0)
001034  016160  000006  000006          MOV     N.FNAM(R1),N.FNAM(R0)   ;FILENAME
001042  016160  000010  000010          MOV     N.FNAM+2(R1),N.FNAM+2(R0)
001050  016160  000012  000012          MOV     N.FNAM+4(R1),N.FNAM+4(R0)
001056  016160  000014  000014          MOV     N.FTYP(R1),N.FTYP(R0)   ;FILE TYPE
001064  005060  000016                  CLR     N.FVER(R0)              ;NEXT AVAIL. VERSION NUMBER
001070  005060  000020                  CLR     N.STAT(R0)
001074  005060  000022                  CLR     N.NEXT(R0)
001100  016160  000024  000024          MOV     N.DID(R1),N.DID(R0)     ;UFD'S FID
001106  016160  000026  000026          MOV     N.DID+2(R1),N.DID+2(R0)
001114  016160  000030  000030          MOV     N.DID+4(R1),N.DID+4(R0)
001122  016160  000020  000032          MOV     N.STAT(R1),N.DVNM(R0)   ;DEVICE NAME
001130  016160  000022  000034          MOV     N.NEXT(R1),N.UNIT(R0)   ;DEVICE UNIT NUMBER
001136  012700  000060'                 MOV     #EMAFDB,R0              ;R0->FDB
                                ;
                                ;   SPECIFY BLOCK SIZE TO BE N.BUFB BYTES (2048.)
                                ;
001142                                  FDBK$R  R0,,#N.BUFB
                                ;
                                ;   ALLOCATE SPACE FOR NEW FILE -- 48. SECTORS OR 12 BLOCKS
                                ;
001150                                  FDAT$R  R0,,,#48.
                                ;
                                ;   OPEN FILE FOR WRITE
                                ;
001156                                  OFNB$W  R0
001170  103010                         BCC     2$
001172  116001  000052                  MOVB    F.ERR(R0),R1            ;R1 = FCS ERROR
001176  010167  000576'                 MOV     R1,PAR2
001202                                  CALL    FCSERR
001206  000167  000322                  JMP     EXIT
                                ;
                                ;   LOAD FID OF NEW TDCTB FILE INTO SNDBUF FOR TRANSMISSION TO QTS
                                ;
001212  012701  000026'         2$:     MOV     #SNDBUF,R1              ;R1->SNDBUF
001216  016061  000102  000000          MOV     N.FID+F.FNB(R0),FD.FID(R1)      ;FID
001224  016061  000104  000002          MOV     N.FID+F.FNB+2(R0),FD.FID+2(R1)
001232  016061  000120  000004          MOV     N.FVER+F.FNB(R0),FD.FVR(R1)     ;VERSION NO.
001240                                  RESTOR  R0,R1
001244                                  EXIT    OPNTDC
```

```
                                        .SBTTL  WRITE TDCTB TO DISK
                                ;
                                ;   THE PURPOSE OF THIS SUBROUTINE IS TO WRITE ONE BLOCK OF
                                ;   THE TERM DETECTOR CONTROL TABLE TO DISK.
                                ;
                                WRTTDC:
                                        WRITE$  #EMAFDB,R3              ;WRITE 1ST BUFFER BLOCK TO DISK
                                        WAIT$   #EMAFDB
                                        TSTB    IOST                    ;ANY ERRORS?
                                        BGT     1$                      ;BRANCH IF NO
                                        MOVB    IOST,R1
                                        MOV     R1,PAR2
                                        CALL    FCSERR
                                        JMP     EXIT
                                1$:     ADD     #N,BUFB,R3              ;NEXT BLOCK START
                                        SOB     R2,WRTTDC               ;WRITE ALL REQUESTED BLOCKS
                                        EXIT    WRTTDC
```

```
                                        .SBTTL  ERROR HANDLING ROUTINE
                                ;
                                ;   DIRECTIVE ERROR
                                ;
001330  011667  000574'         DIRERR::MOV     (SP),PAR1               ;PC AT DIRECTIVE ERROR
001334  016767  000000G 000576'         MOV     $DSW,PAR2               ;$DSW
001342                                  MOUT$S  #MSG1,#PAR1
001362  005726                          TST     (SP)+
001364  000167  000144                 JMP     EXIT                    ;EXIT
                                ;
                                ;   FCS ERROR
                                ;
001370  011667  000574'         FCSERR::MOV     (SP),PAR1               ;PC
001374                                  MOUT$S  #MSG2,#PAR1
001414  005726                          TST     (SP)+                   ;RESTORE STACK
001416  000167  000112                 JMP     EXIT
                                ;
                                ;   NODE ERROR
                                ;
001422  010067  000574'         NODERR::MOV     R0,PAR1
001426                                  MOUT$S  #MSG4,#PAR1
001446  005726                          TST     (SP)+
001450  000167  000060                 JMP     EXIT
001454  010067  000574'         TDCERR: MOV     R0,PAR1
001460                                  MOUT$S  #MSG6,#PAR1
001500  005726                          TST     (SP)+
001502  000167  000026                 JMP     EXIT
001506                          VIERR:: MOUT$S  #MSG5,#PAR1
001526  005726                          TST     (SP)+
001530  000167  000000                 JMP     EXIT
                                ;
001534  012767  177777  000000G EXIT:  MOV     #-1,SNDBUF+SD,FSA
001542  000167  177160                 JMP     QT2,7
```

```
                                        ;
                                        ;  SUBROUTINE TO CHECK FOR TDCT OVERFLOW OR TDCT BUFFER OVERFLOW.
                                        ;        INPUT                      OUTPUT
                                        ;    R0= CURRENT LOW FSA LOC        SCRATCH
                                        ;    R1= SCRATCH                    SCRATCH
                                        ;    R2= (SAVED)                    (RESTORED)
                                        ;    R3= (SAVED)                    (RESTORED)
                                        ;    R4= ..........NOT USED.................
                                        ;    R5= ..........NOT USED.................
                                        ;
                                        ;
001546  016705  000602'         TDCHK:: MOV     NXTADD,R5       ;GET LAST TDCT ADR
001552  020567  000310'                 CMP     R5,BUFMAX       ;BUFFER OVERFLOW?
001556  103466                         BLO     2$              ;NO
001560  020527  007775                 CMP     R5,#TDBMAD      ;TDCT OVERFLOW?
001564  101066                         BHI     4$              ;YES
001566                                 SAVE    R2,R3           ;NO
001572  070027  000003                 MUL     #3,R0
001576  162701  000004                 SUB     #4,R1
001602  005600                         SBC     R0
001604  071027  002000                 DIV     #N.BUFW,R0
001610  166700  000306'                 SUB     CURBLK,R0
001614  001450                         BEQ     41$             ;       NO BLOCKS TO WRITE (FATAL)
001616  010002                         MOV     R0,R2           ;R2=# BLOCKS TO WRITE
001620  060067  000306'                 ADD     R0,CURBLK       ;NOTE ADDITION BLOCKS OFFSET
001624  012703  000606'                 MOV     #TDCBUF,R3      ;R3=ADR OF 1ST BUFFER TO WRITE
001630  004767  177412                 JSR     PC,WRTTDC       ;WRITE BLOCKS
001634  012701  014776'                 MOV     #TDCEND+TDBOSZ+TDBOSZ,R1        ;SHIFT TDCT DOWN
001640  160301                         SUB     R3,R1
001642  006201                         ASR     R1              ;R1=# WORDS TO XFER
001644  012700  000606'                 MOV     #TDCBUF,R0      ;R0=TO ADR    R3=FROM ADR
001650  012320                 3$:     MOV     (R3)+,(R0)+     ;SHIFT
001652  077102                         SOB     R1,3$
001654  016700  000306'                 MOV     CURBLK,R0       ;COMPUTER VIRTUAL START OF TDCT BUFFER
001660  070027  004000                 MUL     #N.BUFB,R0
001664  012700  000606'                 MOV     #TDCBUF,R0
001670  160100                         SUB     R1,R0
001672  010067  000312'                 MOV     R0,TDCADR       ;VIRTUAL START
001676  016700  000306'                 MOV     CURBLK,R0       ;COMPUTE BUFFER END FSA LOC
001702  062700  000003                 ADD     #TDBBLK,R0
001706  070027  002000                 MUL     #N.BUFW,R0
001712  162701  000004                 SUB     #4,R1
001716  005600                         SBC     R0
001720  071027  000003                 DIV     #3,R0
001724  010067  000310'                 MOV     R0,BUFMAX       ;END BUFFER FSA LOC
001730                                 RESTOR  R2,R3
001734  000207                 2$:     RTS     PC
001736  012700  000003         41$:    MOV     #TDBBLK,R0
001742  004767  177506         4$:     JSR     PC,TDCERR       ;REPORT ERROR
                                ;       EXIT
```

```
                                        ;
                                        ;  SUBROUTINE TO PROCCESS ELSE NODE.
                                        ;
                                        ;  R3 = INPUT NODE
                                        ;
001746  005067  000264'         PRCELS: CLR     REESET          ;CAN'T ALTER PRIOR STATE
001752  010367  000262'                 MOV     R3,PRIOR        ;SAVE PRIOR ELSE LOC
001756  004767  000000G                 JSR     PC,GVMASK       ;STEP VECTOR
001762  004767  001612                  JSR     PC,EN           ;CREATE NEW ELSE AND NORMAL NODE
001766  103406                          BCS     11$             ;IF NO NEW NORMAL NODE
001770  152764  000200  000005         BISB    #NDSKIP,N.FLG(R4)       ;FLAG NEW NORMAL NODE TO BE SKIPPED
001776  010403                          MOV     R4,R3           ;PROCESS NEW NORMAL NODE
002000  004767  000002                  JSR     PC,SPLTN1
002004  000207                  11$:    RTS     PC
```

```
                                        ;
                                        ;  SUBROUTINE TO PROCESS NIBBLE 1 NODES. IT SPLITS THE NODE AND CREATES
                                        ;  NIBBLE 2 NODES.
                                        ;
                                        ;  R3 = INPUT NODE
                                        ;
002006  016305  000010        SPLTN1: MOV     N.FSA(R3),R5    ;R5=FSA LOC
002012  070527  000006                MUL     #6,R5
002016  066705  000312'               ADD     TDCADR,R5
002022  005067  000264'               CLR     REESET          ;CAN'T ALTER PRIOR STATES
002026  116304  000005                MOVB    N.FLG(R3),R4    ;R4=NODE FLAGS
002032  132704  000030                BITB    #NDSE!NDJSC,R4  ;SINGLE EXIT OR JUMP/SEQ STATE?
002036                                BOFF    10$             ;NO
002040  016325  000012                MOV     N.SUM(R3),(R5)+ ;XFER BITS OR VALUE CONDITIONAL ON
002044  010567  000264'               MOV     R5,REESET       ;THIS STATE CAN BE ALTERED
002050  016725  000602'               MOV     NXTADD,(R5)+    ;XFER NEXT STATE
002054                                SAVE    N.FSA(R3),R3    ;SAVE NODE VALUES
002062  016763  000602' 000010        MOV     NXTADD,N.FSA(R3);ALTER NODE FSA LOCATION
002070  005267  000602'               INC     NXTADD          ;ALLOCATE STATE
002074  132704  000020                BITB    #NDSE,R4        ;INDEX OR JUMP STATE?
002100                                BOFF    1$              ;JUMP
002102  012725  044000                MOV     #ST$INX!SSBIT,(R5)+     ;SINGLE EXIT INDEX
002106  022763  000001  000012        CMP     #1,N.SUM(R3)    ;INTERWORD?
002114  001443                        BEQ     3$              ;YES
002116  004767  000000G               JSR     PC,GVMASK       ;STEP VECTOR
002122                                RESTOR  N.FSA(R0),R0    ;RESTORE NODE VALUES
002130  004767  003264                JSR     PC,NCN          ;CREATE NIBBLE 2 NODE(S)
002134  000207                        RTS     PC
002136  012725  114000        1$:     MOV     #ST$JSQ!JMPBIT!TXTBIT,(R5)+     ;JUMP/SEQ STATE
002142  112763  000002  000004        MOVB    #N2P,N.PHI(R3)          ;ALTER OLD NODE
002150  005763  000012                TST     N.SUM(R3)       ;INTERWORD?
002154  001415                        BEQ     2$              ;YES
002156  004767  003416                JSR     PC,MRGEN        ;MERGE ELSE VECTOR INTO NODE
002162  004767  000000G               JSR     PC,GVMASK       ;STEP VECTOR
002166                                RESTOR  N.FSA(R0),R0    ;RESTORE NODE VALUES
002174  112760  000000  000004        MOVB    #N1P,N.PHI(R0)
002202  004767  001372                JSR     PC,EN           ;CREATE ELSE & NIB1 NODES
002206  000207                        RTS     PC
002210  004767  000000G       2$:     JSR     PC,FLUMD                ;CREATE FLU-MODIFIERS
002214                                RESTOR  N.FSA(R0),R0    ;RESTORE NODE VALUES
002222  000207                        RTS     PC
002224                        3$:     SAVE    N.PHI(R3),N.ELSE(R3),R3 ;SAVE NODE VALUES
002236  004767  000746                JSR     PC,INTERW               ;COMPLETE VECTOR
002242                                RESTOR  N.PHI(R0),N.ELSE(R0),R0 ;RESTORE NODE VALUES
002254                                RESTOR  N.FSA(R0),R0
002262  000207                        RTS     PC
002264  132763  000004  000005  10$:  BITB    #NDCID,N.FLG(R3);CHANGE INDERWORD DEFAULT REQUIRED?
002272                                BOFF    20$             ;NO
002274  016304  000014                MOV     N.ELSE(R3),R4   ;CHECK IF CID FSA LOC SET?
002300  005764  000014                TST     N.CIDP(R4)
002304  001403                        BEQ     22$             ;IF NOT SET, CREATE NEW
002306  016425  000014                MOV     N.CIDP(R4),(R5)+;IF SET, POINT THIS CID STATE TO IT
002312  000425                        BR      23$
002314  016725  000602'       22$:    MOV     NXTADD,(R5)+    ;CREATE CHANGE INDERWORD DEF STATE
002320                                SAVE    R3,R5
002324  016764  000602' 000014        MOV     NXTADD,N.CIDP(R4)       ;POINT OTHER CID STATES HERE
002332  016705  000602'               MOV     NXTADD,R5
```

```
  669 002336  005267  000602'                    INC     NXTADD          ;ALLOCATE THE FSA LOC
  670 002342  012701  000004                     MOV     #CN1,R1         ;CREATE NIB1 VECTOR
  671 002346  012702  000001                     MOV     #1,R2           ;FOR INTERWORD BIT
  672 002352  004767  001044                     JSR     PC,GENVEC       ;GENERATE VECTOR
  673 002356  004767  000000G                    JSR     PC,FLUMD        ;CREATE FLU MOD STATES
  674 002362                                     RESTOR  R3,R5
  675 002366  016725  000602'          23$:      MOV     NXTADD,(R5)+    ;COMPLETE CID STATE
  676 002372  012725  124000                     MOV     #ST$CNG!IWBIT,(R5)+
  677 002376  016705  000602'                    MOV     NXTADD,R5       ;SET UP FOR INDEX STATE
  678 002402  070527  000006                     MUL     #6,R5
  679 002406  066705  000312'                    ADD     TDCADR,R5
  680 002412  005267  000602'                    INC     NXTADD
  681 002416  016325  000012           20$:      MOV     N.SUM(R3),(R5)+ ;CREATE INDEX STATE
  682 002422  016725  000602'                    MOV     NXTADD,(R5)+
  683 002426  012725  040000                     MOV     #ST$INX,(R5)+
  684 002432  132763  000100  000005             BITB    #NDTAD,N.FLG(R3)        ;TRY AGAIN ELSE?
  685 002440                                     BOFF    21$
  686 002442  052765  020000  177776             BIS     #TAEBIT,-2(R5)          ;YES: ALTER STATE
  687 002450  004767  000722           21$:      JSR     PC,BITCNT       ;COUNT BITS IN SUMMARY
  688 002454  022702  000001                     CMP     #1,R2           ;ONLY 1?
  689 002460  103404                             BLO     24$             ;NO: MULTIPLE BITS
  690 002462  162705  000004                     SUB     #4,R5           ;YES: CAN ALTER THIS STATE
  691 002466  010567  000264'                    MOV     R5,REESET
  692 002472  016705  000602'          24$:      MOV     NXTADD,R5       ;R5=START ADR OF INDEXED STATES
  693 002476  060267  000602'                    ADD     R2,NXTADD       ;ALLOCATE INDEXED STATES
  694 002502  012701  000017                     MOV     #15.,R1
  695 002506  016304  000012                     MOV     N.SUM(R3),R4    ;R4=BIT SUMMARY
  696 002512  005002                             CLR     R2      ;SET UP TO SCAN SUMMARY FOR BIT POSITIONS
  697 002514  000261                             SEC             ;NEEDING NODE GENERATION
  698 002516  006002           30$:      ROR     R2
  699 002520  006304                             ASL     R4
  700 002522  103413                             BCS     34$     ;NEEDS NODE
  701 002524  077104           31$:      SOB     R1,30$
  702 002526  006202                             ASR     R2      ;CHECK INTERWORD BIT
  703 002530  006304                             ASL     R4
  704 002532  103006                             BCC     33$     ;NOT SET
  705 002534  012701  000004                     MOV     #CN1,R1 ;CREATE INTERWORD VECTOR AND NODE AND STATES
  706 002540  004767  000656                     JSR     PC,GENVEC
  707 002544  004767  000440                     JSR     PC,INTERW
  708 002550  000207           33$:      RTS     PC
  709 002552                     34$:      SAVE    R1,R2,R3,R4,R5
  710 002564  012701  000004                     MOV     #CN1,R1         ;CREATE NIB1 VECTOR
  711 002570  004767  000626                     JSR     PC,GENVEC
  712 002574  004767  000000G                    JSR     PC,GVMASK       ;STEP VECTOR
  713 002600  004767  002614                     JSR     PC,NCN          ;CREATE NIB2 NODE
  714 002604                                     RESTOR  R1,R2,R3,R4,R5
  715 002616  005205                             INC     R5              ;STEP TO NEXT INDEXED STATE
  716 002620  000241                             CLC
  717 002622  000740                             BR      31$
```

```
                                        ;
                                        ;  SUBROUTINE TO PROCESS NIBBLE 2 NODES. IT SPLITS THE NODE AND CREATES
                                        ;  ELSE AND NIBBLE 1 NODES
                                        ;  R3 = INPUT NODE
                                        ;
002624  016305  000010          SPLTN2: MOV     N.FSA(R3),R5    ;R5=FSA LOC
002630  070527  000006                  MUL     #6,R5
002634  066705  000312'                 ADD     TDCADR,R5
002640  005067  000264'                 CLR     REESET          ;PRIOR STATE CAN'T BE ALTERED
002644  116304  000005                  MOVB    N.FLG(R3),R4    ;R4=NODE FLAGS
002650  132704  000020                  BITB    #NDSE,R4        ;SINGLE EXIT?
002654                                  BOFF    1$              ;NO
002656  016325  000012                  MOV     N.SUM(R3),(R5)+ ;CREATE SINGLE EXIT INDEX STATE
002662  010567  000264'                 MOV     R5,REESET       ;THIS STATE CAN BE ALTERED
002666  016725  000602'                 MOV     NXTADD,(R5)+
002672                                  SAVE    N.FSA(R3),R3    ;SAVE NODE VALUES
002700  016763  000602' 000010          MOV     NXTADD,N.FSA(R3);ALTER NODE FSA LOC
002706  005267  000602'                 INC     NXTADD
002712  012725  044000                  MOV     #ST$INX!SSBIT,(R5)+
002716  004767  002656                  JSR     PC,MRGEN        ;MERGE ELSE WITH VECTOR
002722  004767  000000G                 JSR     PC,GVMASK       ;STEP VECTOR
002726                                  RESTOR  N.FSA(R0),R0    ;RESTOR NODE VALUES
002734  004767  000640                  JSR     PC,EN           ;CREATE ELSE & NIB1 NODES
002740  000207                          RTS     PC
002742  016325  000012          1$:     MOV     N.SUM(R3),(R5)+ ;CREATE INDEX STATE
002746  016725  000602'                 MOV     NXTADD,(R5)+
002752  012725  040000                  MOV     #ST$INX,(R5)+
002756  004767  000414                  JSR     PC,BITCNT       ;COUNT BITS IN SUMMARY
002762  132704  000040                  BITB    #NDEON,R4       ;ELSE OVERRIDE?
002766                                  BOFF    2$              ;NO
002770  052765  010000  177776          BIS     #EOBIT,-2(R5)   ;YES: ALTER STATE
002776  005202                          INC     R2              ;ONE MORE FOR EO
003000  022702  000001          2$:     CMP     #1,R2           ;ONLY ONE BIT?
003004  103405                          BLO     3$              ;NO: MULTIPLE BITS
003006  101046                          BHI     32$             ;NO: NO BITS, DON'T CREATE NODE
003010  162705  000004                  SUB     #4,R5           ;THIS STATE CAN BE ALTERED
003014  010567  000264'                 MOV     R5,REESET
003020  016705  000602'          3$:     MOV     NXTADD,R5       ;START ADDRESS OF INDEXED STATES
003024  060267  000602'                 ADD     R2,NXTADD       ;ALLOCATE INDEXED STATES
003030  132704  000040                  BITB    #NDEON,R4       ;ELSE OVERRIDE?
003034                                  BOFF    4$              ;NO
003036                                  SAVE    R3,R5           ;YES
003042  012701  000000                  MOV     #CCW,R1         ;CREATE VECTOR FOR NUMERIC DON'T CARES
003046  012702  001400                  MOV     #EMXNFD!EMXNVD,R2
003052  004767  000344                  JSR     PC,GENVEC
003056  004767  002516                  JSR     PC,MRGEN        ;MERGE ELSE
003062  004767  000000G                 JSR     PC,GVMASK       ;STEP VECTOR
003066  004767  000506                  JSR     PC,EN           ;CREATE NODES
003072                                  RESTOR  R3,R5
003076  005205                          INC     R5              ;STEP TO NEXT INDEXED STATE
003100  005002                  4$:     CLR     R2
003102  016304  000012                  MOV     N.SUM(R3),R4    ;SET UP TO CHECK FOR BIT POSITIONS
003106  012701  000020                  MOV     #16.,R1         ;IN SUMMARY FOR WHICH NODES MUST
003112  000261                          SEC                     ;BE CREATED
003114  006002                  30$:    ROR     R2
003116  006304                          ASL     R4
003120  103402                          BCS     34$             ;CREATE NODE
```

```
    776 003122  077104                  31$:    SOB     R1,30$
    777 003124  000207                  32$:    RTS     PC
    778 003126                           34$:    SAVE    R1,R2,R3,R4,R5
    779 003140  012701  000002                  MOV     #CN2,R1         ;CREATE NIB2 VECTOR
    780 003144  004767  000252                  JSR     PC,GENVEC
    781 003150  004767  002424                  JSR     PC,MRGEN        ;MERGE ELSE
    782 003154  004767  000000G                 JSR     PC,GVMASK       ;STEP VECTOR
    783 003160  004767  000414                  JSR     PC,EN           ;CREATE NODES
    784 003164                                  RESTOR  R1,R2,R3,R4,R5
    785 003176  005205                          INC     R5              ;STEP TO NEXT INDEXED STATE
    786 003200  000241                          CLC
    787 003202  005067  000264'                 CLR     REESET
    788 003206  000745                          BR      31$
```

```
                                        ;
                                        ; THIS SUBROUTINE COMPLETES A VECTOR BY CREATING ANY INTERWORD
                                        ; STATES NECESSARY AND FLU-MODIFIERS IF UNIQUE PATH.
                                        ; R3=INPUT NODE.
                                        ;
003210  016305  000010          INTERW: MOV     N.FSA(R3),R5    ;R5=FSA LOC.
003214  070527  000006                  MUL     #6,R5
003220  066705  000312'                 ADD     TDCADR,R5
003224  112763  000002  000004          MOVB    #N2P,N.PHI(R3)  ;ALTER NODE.
003232  012763  000124' 000014          MOV     #ELSDFN,N.ELSE(R3)
003240  010302                          MOV     R3,R2
003242  012703  000060'                 MOV     #N2IDX,R3       ;SCAN FOR NIB2 NODE.
003246  012700  000002                  MOV     #N2P,R0
003252  004767  000000G                 JSR     PC,SCAN
003256  103417                          BCS     2$              ;IF NEW NODE.
003260  005767  000264'                 TST     REESET          ;IF OLD NODE, PRIOR STATE ALTERABLE?
003264  001406                          BEQ     1$              ;NO.
003266  016477  000010  000264'        MOV     N.FSA(R4),@REESET;YES: ALTER PRIOR STATE.
003274  005367  000602'                 DEC     NXTADD
003300  000207                          RTS     PC
003302  005025                  1$:     CLR     (R5)+           ;CREATE JUMP STATE
003304  016425  000010                  MOV     N.FSA(R4),(R5)+
003310  012725  100000                  MOV     #ST$JSQ,(R5)+
003314  000207                          RTS     PC
003316  012725  000001          2$:     MOV     #1,(R5)+        ;CREATE INDEX STATE.
003322  016725  000602'                 MOV     NXTADD,(R5)+
003326  012725  040000                  MOV     #ST$INX,(R5)+
003332  152764  000200  000005        BISB    #NDSKIP,N.FLG(R4)       ;SKIP NODE
003340  010403                          MOV     R4,R3           ;ALTER NODE.
003342                                  SAVE    N.FSA(R3),R3
003350  016763  000602' 000010          MOV     NXTADD,N.FSA(R3)
003356  005267  000602'                 INC     NXTADD          ;ALLOCATE FSA LOC.
003362  004767  000000G                 JSR     PC,FLUMD        ;CREATE FLU-MOD STATES.
003366                                  RESTOR  N.FSA(R0),R0
003374  000207                          RTS     PC
```

```
826                                     ;
827                                     ;  COUNT BITS IN SUMMARY WORD OF NODE.
828                                     ;  R3=NODE INPUT.
829                                     ;  R2=COUNT OUTPUT.
830                                     ;
831 003376  016300  000012          BITCNT: MOV     N.SUM(R3),R0    ;R0=SUMMARY.
832 003402  012701  000020                  MOV     #16.,R1         ;R1=# BIT POSITIONS.
833 003406  005002                          CLR     R2              ;R2=COUNT.
834 003410  006300                  22$:    ASL     R0
835 003412  103001                          BCC     23$
836 003414  005202                          INC     R2      ;COUNT POSITION.
837 003416  077104                  23$:    SOB     R1,22$
838 003420  000207                          RTS     PC
```

```
                                ;
                                ;  SUBROUTINE TO GENERATE VECTORS.
                                ;
                                ;        INPUT                      OUTPUT
                                ;   R0=  SCRATCH                    0
                                ;   R1=  SCAN TYPE (SEE 10$)       VECTOR SIZE
                                ;   R2=  BITS TO MASK FOR          (SAME)
                                ;   R3=  INPUT NODE                OUTPUT NODE
                                ;   R4=  ..........SCRATCH:.................
                                ;   R5=  ..........SCRATCH:.................
                                ;
003422  016704  000120'         GENVEC: MOV     NNPE,R4         ;NODE ADDRESS
003426  016164  003474' 000004          MOV     11$(R1),N.PHI(R4)       ;SET UP NODE HEADER
003434  010564  000010                  MOV     R5,N.FSA(R4)
003440  016364  000014  000014          MOV     N.ELSE(R3),N.ELSE(R4)
003446  016300  000006                  MOV     N.LGT(R3),R0    ;INPUT VECTOR SIZE
003452  062703  000016                  ADD     #N.NHS,R3
003456  062704  000016                  ADD     #N.NHS,R4
003462  000171  003466'                 JMP     @10$(R1)        ;PROCESS ACCORDING TO SCAN TYPE
003466  003502'                 10$:    20$     ;CONTROL WORD   (CCW)
003470  003524'                         30$     ;NIBBLE 2       (CN2)
003472  003546'                         40$     ;NIBBLE 1       (CN1)
003474  000002                  11$:    N2P     ;CCW
003476  000002                          N2P     ;CN2
003500  000000                          N1P     ;CN1
003502  005001                  20$:    CLR     R1
003504  012305                  21$:    MOV     (R3)+,R5        ;GET ENTRY
003506  030265  016010'                 BIT     R2,EMA(R5)      ;TEST BITS AGAINST CONTROL WORD
003512                                  BOFF    22$             ;NOT THIS ENTRY
003514  005201                          INC     R1      ;COUNT & XFER THIS ENTRY
003516  010524                          MOV     R5,(R4)+
003520  077007                  22$:    SOB     R0,21$
003522  000421                          BR      50$
003524  005001                  30$:    CLR     R1
003526  012305                  31$:    MOV     (R3)+,R5
003530  030265  016014'                 BIT     R2,EMA+EMXNB2(R5)       ;TEST AGAINST NIBBLE 2
003534                                  BOFF    32$
003536  005201                          INC     R1
003540  010524                          MOV     R5,(R4)+
003542  077007                  32$:    SOB     R0,31$
003544  000410                          BR      50$
003546  005001                  40$:    CLR     R1
003550  012305                  41$:    MOV     (R3)+,R5
003552  030265  016012'                 BIT     R2,EMA+EMXNB1(R5)       ;TEST AGAINST NIBBLE 1
003556                                  BOFF    42$
003560  005201                          INC     R1
003562  010524                          MOV     R5,(R4)+
003564  077007                  42$:    SOB     R0,41$
003566  016703  000120'         50$:    MOV     NNPE,R3         ;R3=NEW NODE
003572  010163  000006                  MOV     R1,N.LGT(R3)    ;SET VECTOR LENGTH
003576  000207                          RTS     PC
```

```
                                        ;
                                        ;  SUBROUTINE EN FIRST CREATES AN ELSE NODE, CHECKS FOR DUPLICATE,
                                        ;  AND CREATES ANY NECCESSARY CHANGE DEFAULT STATES.  IT THEN CREATES
                                        ;  A NORMAL NODE AND AGAIN CHECKS FOR DUPLICATES.
                                        ;  R3=INPUT NODE.
                                        ;  C-BIT IS SET ON OUTPUT IF NORMAL NODE CREATED WAS A DUPLICATE
                                        ;
003600  016704  000120'          EN:     MOV     NNPE,R4         ;R4=NEW ELSE NODE ADR
003604                                  SAVE    R4
003606  005067  000302'                  CLR     TAFLG           ;INIT VARABLES
003612  005067  000270'                  CLR     DCNT
003616  005067  000266'                  CLR     NMLCNT
003622  005067  000272'                  CLR     TVCNT
003626  005067  000274'                  CLR     CTRSUM
003632  005067  000276'                  CLR     TASUM
003636  005067  000300'                  CLR     NMLSUM
003642  012764  000004  000004          MOV     #ENP,N.PHI(R4)  ;INIT NODE HEADERS
003650  012767  000000  027014'          MOV     #NIP,VVEC+N.PHI
003656  016300  000006                  MOV     N.LGT(R3),R0    ;GET VECTOR SIZE
003662  126327  000004  000004          CMPB    N.PHI(R3),#ENP  ;IF ENP NODE, USE IT'S ADR AS PRIOR ELSE
003670  001403                          BEQ     6$
003672  016367  000014  000262'         MOV     N.ELSE(R3),PRIOR;GET PRIOR ELSE
003700  012701  027026'          6$:     MOV     #VVEC+N.NHS,R1  ;R1=NORMAL NODE VECTOR POINTER
003704  062704  000016                  ADD     #N.NHS,R4       ;R4=ELSE NODE VECTOR POINTER
003710  062703  000016                  ADD     #N.NHS,R3       ;R3=INPUT NODE VECTOR POINTER
003714  012305                  1$:     MOV     (R3)+,R5        ;GET NEXT (1ST) ENTRY
003716  016502  016010'                  MOV     EMA(R5),R2      ;GET CONTROL WORD
003722  032702  040000                  BIT     #EMXMTV,R2      ;MULTIPLE VALUES FOR ENTRY?
003726                                  BOFF    4$              ;NO
003730  050267  000274'                  BIS     R2,CTRSUM       ;ADD TO CONTROL SUMMARY
003734  032702  004000                  BIT     #EMXVDC,R2      ;VLDC?
003740                                  BOFF    2$              ;NO
003742  005267  000302'                  INC     TAFLG           ;YES; NEXT ENTRY SHOULD BE TRY AGAIN
003746  032702  010000                  BIT     #EMXTRL,R2      ;IF TRAILING VLDC, COUNT IT
003752                                  BOFF    3$
003754  005267  000272'                  INC     TVCNT
003760  000403                          BR      3$
003762  032702  002000          2$:     BIT     #EMXFDC,R2      ;FLDC?
003766                                  BOFF    4$              ;NO
003770  010524                  3$:     MOV     R5,(R4)+        ;ADD ENTRY TO ELSE VECTOR
003772  005267  000270'                  INC     DCNT
003776  000417                          BR      7$
004000  010521                  4$:     MOV     R5,(R1)+        ;ADD ENTRY TO NORMAL VECTOR
004002  005267  000266'                  INC     NMLCNT
004006  005767  000302'                  TST     TAFLG           ;TRY AGAIN?
004012                                  BOFF    5$              ;NO
004014  056567  016012'  000276'        BIS     EMA+EMXNB1(R5),TASUM    ;ADD TO TRY AGAIN SUMMARY
004022  005067  000302'                  CLR     TAFLG
004026  000403                          BR      7$
004030  056567  016012'  000300'  5$:     BIS     EMA+EMXNB1(R5),NMLSUM   ;ADD TO NORMAL SUMMARY
004036  077052                  7$:     SOB     R0,1$           ;DO FOR ALL ENTRIES OF INPUT VECTOR
004040                                  RESTOR  R4              ;R4=ELSE NODE ADR
004042  016705  015006'                  MOV     VMASK+N.FSA,R5  ;BEFORE SCAN GET FSA LOC
004046  070527  000006                  MUL     #6,R5
004052  066705  000312'                  ADD     TDCADR,R5
004056  016764  000270'  000006        MOV     DCNT,N.LGT(R4)  ;STORE VECTOR LENGTH OF ELSE
004064  001570                          BEQ     17$             ;IF NONE, USE DEFAULT ELSE NODE
```

```
 949 004066                                         SAVE    R4
 950 004070  062704  000016                         ADD     #N.NHS,R4       ;CALCULATE HASH INDEX FOR ELSE VECTOR
 951 004074  005003                                 CLR     R3
 952 004076  016701  000270'                        MOV     DCNT,R1         ; HASK = X-BITS OF THE SUM OF THE
 953 004102  020127  000004                         CMP     R1,#ENCNT       ;       FIRST N ENTRIES OF ELSE VECTOR
 954 004106  101402                                 BLOS    10$
 955 004110  012701  000004                         MOV     #ENCNT,R1
 956 004114  062403                     10$:        ADD     (R4)+,R3
 957 004116  077102                                 SOB     R1,10$
 958 004120  042703  177761                         BIC     #ELSMSK,R3
 959 004124  062703  000000'                        ADD     #ELSIDX,R3      ;R3=ELSE INDEX POINTER
 960 004130                                         RESTOR  R2              ;SCAN FOR ELSE DUPLICATE
 961 004132  012700  000004                         MOV     #ENP,R0
 962 004136  004767  0000006                        JSR     PC,SCAN
 963 004142  012703  027010'                        MOV     #VVEC,R3        ;R3=NORMAL NODE ADR
 964 004146  103410                                 BCS     11$             ;IF NEW NODE
 965 004150  020467  000262'                        CMP     R4,PRIOR        ;IF DUP NODE, SAME AS BEFORE?
 966 004154  001161                                 BNE     171$            ;NO
 967 004156  122767  000004  015002'                CMPB    #ENP,VMASK+N.PHI;INPUT NODE AN ELSE?
 968 004164  001517                                 BEQ     121$            ;YES
 969 004166  000525                                 BR      1440$           ;NO
 970 004170  005767  000604'             11$:       TST     FIRST
 971 004174  001402                                 BEQ     1110$
 972 004176  000167  001002                         JMP     400$            ;IF FIRST STATE, SPECIAL PROCESS
 973 004202  016725  000602'             1110$:     MOV     NXTADD,(R5)+    ;CREATE CHANGE ELSE DEFAULT STATE
 974 004206  010567  000264'                        MOV     R5,REESET       ;STATE IS ALTERABLE
 975 004212  016764  000602'  000010               MOV     NXTADD,N.FSA(R4)
 976 004220  116764  000144'  000012               MOVB    ELSEID,N.EID(R4)        ;ALTER ELSE NODE (NEW)
 977 004226  116764  000272'  000013               MOVB    TVCNT,N.TVCT(R4)
 978 004234  005267  000144'                        INC     ELSEID
 979 004240  005267  000602'                        INC     NXTADD
 980 004244  016701  000262'                        MOV     PRIOR,R1        ;SET CID FSA LOC POINTER
 981 004250  126164  000013  000013               CMPB    N.TVCT(R1),N.TVCT(R4)
 982 004256  001403                                 BEQ     111$            ;IF TRAILING VLDC COUNT SAME, CIDP SAME
 983 004260  005064  000014                         CLR     N.CIDP(R4)      ;IF DIFFER, RESET AS UNDEFINED
 984 004264  000403                                 BR      112$
 985 004266  016164  000014  000014  111$:         MOV     N.CIDP(R1),N.CIDP(R4)
 986 004274  016725  000602'             112$:      MOV     NXTADD,(R5)+
 987 004300  012725  130000                         MOV     #ST$CNG!ELSBIT,(R5)+
 988 004304  016767  000602'  015006'               MOV     NXTADD,VMASK+N.FSA
 989 004312  005267  000602'                        INC     NXTADD
 990 004316  032767  003600  000274'               BIT     #EMXFDC!EMX0VD!EMXNFD!EMXNVD,CTRSUM
 991 004324                                         BON     143$            ;SEPARATE STATES NEEDED
 992 004326  005767  000300'                        TST     NMLSUM          ; IF DON'T CARE OR ADDITION PATHS PRESENT
 993 004332  001117                                 BNE     142$
 994 004334  152764  000200  000005               BISB    #NDSKIP,N.FLG(R4)       ;ELSE=NORMAL NODE, IGNORE ELSE NODE
 995 004342  016565  177772  177774               MOV     -6(R5),-4(R5)
 996 004350  016567  177772  015006'               MOV     -6(R5),VMASK+N.FSA      ;NORMAL NODE FSA LOC STORED HERE
 997 004356  005367  000602'                        DEC     NXTADD
 998 004362  022767  000001  000276'               CMP     #1,TASUM        ;IF ONLY INTERWORD, DON'T CREATE CID
 999 004370  001420                                 BEQ     12$
1000 004372  032767  000001  000276'  143$:        BIT     #1,TASUM        ;IF INTERWORD, CREATE CID AND REMOVE
1001 004400                                         BOFF    12$             ; FROM TRY AGAIN SUMMMARY
1002 004402  016700  000262'                        MOV     PRIOR,R0        ;HAVE MORE TRAILING VLDC'S BEEN FOUND
1003 004406  126760  000272'  000013               CMPB    TVCNT,N.TVCT(R0)
1004 004414  001403                                 BEQ     121$            ;NO: DON'T BOTHER WITH CID
1005 004416  152763  000004  000005               BISB    #NDCID,N.FLG(R3)
```

```
1006 004424  042767  000001  000276' 121$:   BIC     #1,TASUM
1007 004432  056767  000276' 000300' 12$:    BIS     TASUM,NMLSUM    ;ADD TRY AGAIN TO NORMAL (SAME STATE)
1008 004440  000505                           BR      22$
1009 004442  000167  000156          1440$:  JMP     144$
1010 004446  012704  000124'        17$:    MOV     #ELSDFN,R4      ;ELSE NODE = DEFAULT
1011 004452  012703  027010'                MOV     #VVEC,R3        ;R3=NORMAL NODE ADR
1012 004456  020467  000262'                CMP     R4,PRIOR        ;WAS PRIOR DEFAULT?
1013 004462  001474                         BEQ     22$             ;YES
1014 004464  016425  000010                 MOV     N.FSA(R4),(R5)+ ;CREATE CHANGE ELSE DEFAULT
1015 004470  010567  000264'                MOV     R5,REESET       ;STATE IS ALTERABLE
1016 004474  016725  000602'                MOV     NXTADD,(R5)+
1017 004500  012725  130000                 MOV     #ST$CNG!ELSBIT,(R5)+
1018 004504  016767  000602' 015006'        MOV     NXTADD,VMASK+N.FSA      ;NORMAL NODE FSA LOC
1019 004512  005267  000602'                INC     NXTADD                  ;ALLOCATE
1020 004516  000456                         BR      22$
1021 004520  016425  000010          171$:  MOV     N.FSA(R4),(R5)+ ;CREATE CHANGE ELSE DEFAULT
1022 004524  010567  000264'                MOV     R5,REESET       ;STATE IS ALTERABLE
1023 004530  016725  000602'                MOV     NXTADD,(R5)+
1024 004534  012725  130000                 MOV     #ST$CNG!ELSBIT,(R5)+
1025 004540  016767  000602' 015006'        MOV     NXTADD,VMASK+N.FSA      ;NORMAL NODE FSA LOC
1026 004546  005267  000602'                INC     NXTADD                  ;ALLOCATE
1027 004552  032767  003600  000274'        BIT     #EMXFDC!EMX0VD!EMXNFD!EMXNVD,CTRSUM     ;IF DON'T CARE
1028 004560                                  BON     143$                    ;TAD INVALID
1029 004562  122767  000004  015002'        CMPB    #ENP,VMASK+N.PHI        ;INPUT NODE AN ELSE?
1030 004570  001700                         BEQ     143$                    ;YES
1031 004572  032767  000001  000276' 142$:  BIT     #1,TASUM        ;IF INTERWORD, CREATE CID AND REMOVE
1032 004600                                  BOFF    13$             ; FROM TRY AGAIN SUMMARY
1033 004602  016700  000262'                MOV     PRIOR,R0        ;HAVE MORE TRAILING VLDC'S BEEN FOUND
1034 004606  126760  000272' 000013        CMPB    TVCNT,N.TVCT(R0)
1035 004614  001403                         BEQ     144$            ;NO: DON'T BOTHER WITH CID
1036 004616  152763  000004  000005        BISB    #NDCID,N.FLG(R3)
1037 004624  042767  000001  000276' 144$:  BIC     #1,TASUM
1038 004632  016702  000300'          13$:  MOV     NMLSUM,R2       ;ARE THERE ANY UNIQUE TRY AGAIN BITS?
1039 004636  005102                         COM     R2
1040 004640  030267  000276'                BIT     R2,TASUM
1041 004644                                  BOFF    22$             ;NO
1042 004646  152763  000100  000005        BISB    #NDTAD,N.FLG(R3);YES: FLAG TO CREATE SPECIAL INDEX STATE
1043 004654  010463  000014          22$:   MOV     R4,N.ELSE(R3)   ;FILL IN NODE HEADER & GET FSA ADR
1044 004660  016705  015006'                MOV     VMASK+N.FSA,R5
1045 004664  010563  000010                 MOV     R5,N.FSA(R3)
1046 004670  070527  000006                 MUL     #6,R5
1047 004674  066705  000312'                ADD     TDCADR,R5
1048 004700  016763  000300' 000012        MOV     NMLSUM,N.SUM(R3)
1049 004706  001012                         BNE     221$            ;IF PATHS, CONTINUE
1050 004710  005767  000276'                TST     TASUM
1051 004714  001041                         BNE     220$
1052 004716  132763  000004  000005        BITB    #NDCID,N.FLG(R3);IF NO PATHS, DOES INTERWORD NEED HANDLING?
1053 004724                                  BOFF    222$            ;NO: CREATE FAIL STATE
1054 004726  012763  000001  000012        MOV     #1,N.SUM(R3)    ;LOOK FOR INTERWORD
1055 004734  016763  000266' 000006  221$:  MOV     NMLCNT,N.LGT(R3)
1056 004742  026727  000266' 000001        CMP     NMLCNT,#1       ;IF ONLY 1 ENTRY, FLAG AS SINGLE EXIT
1057 004750  001077                         BNE     24$             ;MULTIPLE ENTRIES
1058 004752  016302  000016                 MOV     N.NHS(R3),R2    ;GET ENTRY
1059 004756  016202  016010'                MOV     EMA(R2),R2      ;GET CONTROL WORD
1060 004762  032702  040000                 BIT     #EMXMTV,R2      ;IF SINGLE VALUE &-TRY AGAIN, CAN USE JUMP
1061 004766                                  BON     23$
1062 004770  132763  000100  000005        BITB    #NDTAD,N.FLG(R3)
```

```
1063 004776                                        BON     23$
1064 005000  152763  000010  000005                BISB    #NDJSC,N.FLG(R3)        ;USE JUMP/SEQ
1065 005006  042702  177400                        BIC     #177400,R2              ;AND PUT COND BYTE IN SUMMARY
1066 005012  010263  000012                        MOV     R2,N.SUM(R3)
1067 005016  000454                                BR      24$
1068 005020  016302  000014          220$:         MOV     N.ELSE(R3),R2   ;NODE SHOULD BE NIBBLE 1 NODE FOR ELSE
1069 005024  016305  000010                        MOV     N.FSA(R3),R5    ;GET FSA ADR FROM ELSE NODE
1070 005030  010563  000010                        MOV     R5,N.FSA(R3)    ;RESET NODE FSA LOC
1071 005034  010577  000264'                       MOV     R5,@REESET      ;AND REPOINT TO NEW ADR
1072 005040  070527  000006                        MUL     #6,R5           ;CONVERT LOC TO ADR
1073 005044  066705  000312'                       ADD     TDCADR,R5
1074 005050  056763  000276' 000012                BIS     TASUM,N.SUM(R3) ;ADJUST SUMMARY
1075 005056  142763  000100  000005                BICB    #NDTAD,N.FLG(R3)
1076 005064  000723                                BR      221$
1077 005066  005767  000304'          222$:         TST     FAILST          ;IF NO FAIL STATE YET, CREATE ONE
1078 005072  001412                                BEQ     223$
1079 005074  005767  000264'                       TST     REESET          ;IF NO ALTERABLE STATE, CREATE NEW FAIL
1080 005100  001407                                BEQ     223$
1081 005102  016777  000304' 000264'               MOV     FAILST,@REESET  ;USE OLD FAIL STATE
1082 005110  005367  000602'                       DEC     NXTADD
1083 005114  000261                                SEC
1084 005116  000207                                RTS     PC
1085 005120  016367  000010  000304' 223$:         MOV     N.FSA(R3),FAILST        ;NOTE FAIL STATE LOC
1086 005126  005025                                CLR     (R5)+                   ;CREATE FAIL STATE
1087 005130  005025                                CLR     (R5)+
1088 005132  012725  040000                        MOV     #ST$INX,(R5)+
1089 005136  000261                                SEC
1090 005140  000207                                RTS     PC
1091 005142  152763  000020  000005  23$:          BISB    #NDSE,N.FLG(R3)         ;USE SINGLE EXIT INDEX STATE
1092 005150  010302                        24$:    MOV     R3,R2           ;CALCULATE INDEX FROM ELSE ID
1093 005152  016203  000014                        MOV     N.ELSE(R2),R3
1094 005156  116303  000012                        MOVB    N.EID(R3),R3
1095 005162  042703  177760                        BIC     #NNMASK,R3
1096 005166  006303                                ASL     R3
1097 005170  062703  000020'                       ADD     #N1IDX,R3
1098 005174  012700  000000                        MOV     #N1P,R0         ;SCAN FOR NIB1 NODE
1099 005200  000167  000142                        JMP     NCOM
1100                                        ;
1101 005204  005067  000604'          400$:         CLR     FIRST           ;NO MORE FIRST STATE
1102 005210  032767  003600  000274'                BIT     #EMXFDC!EMX0VD!EMXNFD!EMXNVD,CTRSUM     ;NEXT=ELSE?
1103 005216                                        BOFF    401$            ;YES
1104 005220  012767  000004  000634'                MOV     #4,TDCT+LOC2+2  ;NO: RESET PRIOR STATES TO LOC 5
1105 005226  012767  000005  000646'                MOV     #5,TDCT+LOC4
1106 005234  000406                                BR      402$
1107 005236  012767  000004  000634' 401$:         MOV     #4,TDCT+LOC2+2  ;RESET PRIOR STATES TO LOC 6
1108 005244  012767  000006  000646'                MOV     #6,TDCT+LOC4
1109 005252  012764  000006  000010  402$:         MOV     #6,N.FSA(R4)    ;INIT NODE
1110 005260  116764  000144' 000012                MOVB    ELSEID,N.EID(R4)
1111 005266  116764  000272' 000013                MOVB    TVCNT,N.TVCT(R4)
1112 005274  016164  000014  000014                MOV     N.CIDP(R1),N.CIDP(R4)
1113 005302  005267  000144'                       INC     ELSEID          ;USED
1114 005306  056767  000276' 000300'               BIS     TASUM,NMLSUM
1115 005314  032767  003600  000274'                BIT     #EMXFDC!EMX0VD!EMXNFD!EMXNVD,CTRSUM     ;NEXT=ELSE?
1116 005322                                        BOFF    403$            ;NO
1117 005324  005364  000010                        DEC     N.FSA(R4)       ;ELSE NODE = 5
1118 005330  000167  177320                        JMP     22$
1119 005334  152764  000200  000005  403$:         BISB    #NDSKIP,N.FLG(R4)       ;SKIP ELSE NODE PROCESSING
```

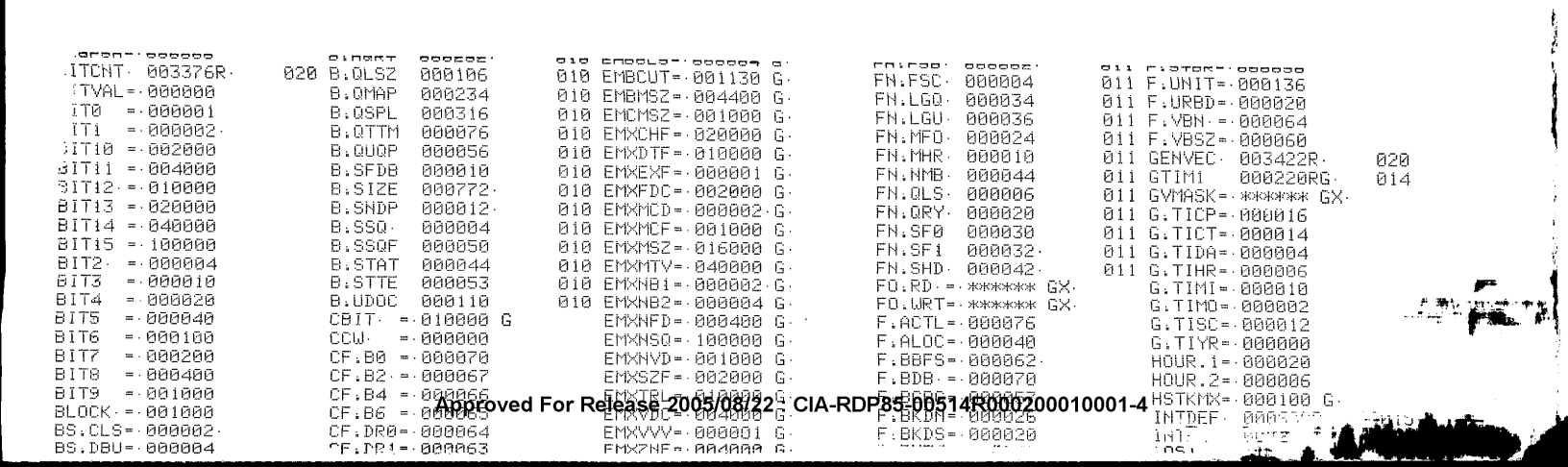
```
 1120 005342  000167  177306                    JMP     22$
 1121                                ;
 1122 005346  004767  000000G          NCOM:    JSR     PC,SCAN
 1123 005352  103420                            BCS     32$             ;NEW NODE
 1124 005354  005767  000264'                    TST     REESET          ;DUP NODE: CAN PRIOR STATE BE ALTERED?
 1125 005360  001006                            BNE     25$             ;YES
 1126 005362  005025                            CLR     (R5)+           ;NO: CREATE JUMP STATE
 1127 005364  016425  000010                    MOV     N.FSA(R4),(R5)+
 1128 005370  012725  100000                    MOV     #ST$JSQ,(R5)+
 1129 005374  000405                            BR      31$
 1130 005376  016477  000010  000264' 25$:      MOV     N.FSA(R4),@REESET       ;ALTER PRIOR STATE
 1131 005404  005367  000602'                    DEC     NXTADD                  ;DEALLOCATE FSA LOC
 1132 005410  000261                    31$:    SEC                     ;DUP NODE
 1133 005412  000207                            RTS     PC
 1134 005414  000241                    32$:    CLC                     ;NEW NODE
 1135 005416  000207                            RTS     PC
```

```
1137
1138                                    ;  SUBROUTINE NCN CREATES NIB2 STATE AND ELSE AND NIB1 NODES.
1139                                    ;  R3 = INPUT NODE
1140                                    ;
1141 005420  016300  000006           NCN:    MOV     N.LGT(R3),R0    ;SET UP FOR NUMERIC DON'T CARE CHECK
1142 005424  005001                            CLR     R1
1143 005426  005004                            CLR     R4
1144 005430  016305  000010                    MOV     N.FSA(R3),R5
1145 005434  070527  000006                    MUL     #6,R5
1146 005440  066705  000312'                   ADD     TDCADR,R5
1147 005444  062703  000016                    ADD     #N.NHS,R3
1148 005450  012302                   1$:     MOV     (R3)+,R2        ;GET ENTRY
1149 005452  032762  001400  016010'          BIT     #EMXNFD!EMXNVD,EMA(R2)  ;NUMERIC DON'T CARE?
1150 005460                                    BOFF    2$                      ;NO
1151 005462  152704  000040                    BISB    #NDEON,R4               ;YES: FLAG OCCURRENCE
1152 005466  000402                            BR      3$
1153 005470  056201  016014'           2$:     BIS     EMA+EMXNB2(R2),R1       ;ADD TO SUMMARY
1154 005474  077013                   3$:     SOB     R0,1$           ;DO FOR ALL ENTRIES
1155 005476  022701  007774                    CMP     #7774,R1        ;IF ALL PATHS, REMOVE ELSE OVERRIDE
1156 005502  001001                            BNE     4$
1157 005504  005004                            CLR     R4
1158 005506  012702  014776'           4$:     MOV     #VMASK,R2       ;R2=NODE ADR
1159 005512  012762  000002  000004           MOV     #N2P,N.PHI(R2)  ;FILL IN NODE TYPE & CLR FLAGS
1160 005520  010162  000012                    MOV     R1,N.SUM(R2)    ;FILL IN SUMMARY
1161 005524  022762  000001  000006           CMP     #1,N.LGT(R2)    ;IF LGT=1 & -ED SET SINGLE EXIT FLAG
1162 005532  001004                            BNE     5$
1163 005534  005704                            TST     R4
1164 005536  001002                            BNE     5$
1165 005540  112704  000020                    MOVB    #NDSE,R4
1166 005544  150462  000005           5$:     BISB    R4,N.FLG(R2)
1167 005550  012700  000002                    MOV     #N2P,R0         ;SCAN FOR NIB2 NODE
1168 005554  016203  000014                    MOV     N.ELSE(R2),R3   ;CALCULATE INDEX FROM ELSE ID
1169 005560  116303  000012                    MOVB    N.EID(R3),R3
1170 005564  042703  177760                    BIC     #NNMASK,R3
1171 005570  006303                            ASL     R3
1172 005572  062703  000060'                   ADD     #N2IDX,R3
1173 005576  000663                            BR      NCOM            ;SCAN FOR NODE
```

```
                                        ;
                                        ;  SUBROUTINE MRGEN MERGES ELSE NODE AND NORMAL NODE EMX POINTERS.
                                        ;  R3=INPUT NORMAL NODE.
                                        ;  R3=OUTPUT CREATED NODE.
                                        ;
005600  016300  000014          MRGEN:  MOV     N.ELSE(R3),R0   ;GET ELSE NODE.
005604  016001  000006                  MOV     N.LGT(R0),R1    ;GET ELSE VECTOR SIZE.
005610  001455                          BEQ     34$             ;IF NULL, NO NEED TO MERGE.
005612  012702  027010'                  MOV     #VVEC,R2        ;R2=CREATED NODE.
005616  012762  000002  000004          MOV     #N2P,N.PHI(R2)  ;FILL IN HEADER.
005624  016362  000010  000010          MOV     N.FSA(R3),N.FSA(R2)
005632  016362  000014  000014          MOV     N.ELSE(R3),N.ELSE(R2)
005640  016304  000006                  MOV     N.LGT(R3),R4
005644  062703  000016                  ADD     #N.NHS,R3       ;STEP TO VECTOR PORTION OF NODES.
005650  062700  000016                  ADD     #N.NHS,R0
005654  062702  000016                  ADD     #N.NHS,R2
005660  005005                          CLR     R5
005662  005704                          TST     R4              ;IF NORMAL NODE NULL, USE ELSE VECTOR.
005664  001413                          BEQ     24$
005666  005205                  1$:     INC     R5              ;ONE MORE ENTRY WILL BE ADDED.
005670  020527  000375                  CMP     R5,#VEC2MX      ;EXCEED LIMIT?.
005674  101024                          BHI     700$            ;YES.
005676  021013                          CMP     (R0),(R3)       ;ELSE OR NORMAL ENTRY NEXT?.
005700  101003                          BHI     20$             ;NORMAL.
005702  012022                  10$:    MOV     (R0)+,(R2)+     ;ELSE.
005704  077110                          SOB     R1,1$                   ;ELSE VECTOR END.
005706  000404                          BR      30$
005710  012322                  20$:    MOV     (R3)+,(R2)+     ;XFER NORMAL ENTRY
005712  077413                          SOB     R4,1$                   ;NORMAL VECTOR END.
005714  010003                  24$:    MOV     R0,R3                   ;COMPLETE WITH ELSE VECTOR.
005716  010104                          MOV     R1,R4
005720  012322                  30$:    MOV     (R3)+,(R2)+     ;COMPLETE WITH REMAINING VECTOR.
005722  005205                          INC     R5
005724  020527  000375                  CMP     R5,#VEC2MX      ;EXCEED LIMIT?.
005730  101006                          BHI     700$            ;YES.
005732  077406                          SOB     R4,30$
005734  012703  027010'                  MOV     #VVEC,R3        ;R3=NEW NODE.
005740  010563  000006                  MOV     R5,N.LGT(R3)    ;FILL IN SIZE.
005744  000207                  34$:    RTS     PC.
005746  010567  000574'          700$:   MOV     R5,PAR1
005752  000167  173530                  JMP     VIERR.
                                        ;
000000'                                 .END    START.
```



```
ASTKMX= 000400 G        B.HRLW  000124      010 EMALGT  015010R      015 FN.EMC  000016      011 F.SPDV= 000072
AVELGT= 000025 G        B.NMBR  000052      010 EMAMSZ= 016000 G         FN.FSA  000000      011 F.SPUN= 000074
```



```
BS.INA=·000000          [illegible]              EMX0VD=·000200 G·        [illegible]              [illegible] GX·
BS.OPN=·000001          CN2·   =·000002·         EN       003600R·    020 F.BKST=·000024           IWBIT·=·004000 G·
BS.SRC=·000003          CPIXMK=·177770 G         ENCNT·=·000004 G·        F.BKVB=·000064           JMPBIT=·004000 G·
BUFMAX· 000310R·    015 CPIXSZ=·[illegible]      [illegible] G·           [illegible]              LN1      000342R·    015
BYTE0  =·000000         CTRSUM  000274R·     015 EOBIT·=·010000 G·        F.CNTG=·000034           LN1E     000403R·    015
BYTE1  =·000001         CURBLK  000306R·     015 EXIT·    001534R·    020 F.DFNB=·000046           LN2·     000403R·    015
BYTE2·=·000002·         DBSLEN=·000116           FAILST·  000304R·    015 F.DSPT=·000044           LN2E     000443R·    015
BYTE3  =·000003         DCNT·   000270R·     015 FCSERR·  001370RG·   020 F.DVNM=·000134           LN4      000443R·    015
BYTE4  =·000004         DELTIM  000000RG     014 FD.BLK=·****** GX·       F.EFBK=·000010           LN4E     000475R·    015
BYTE5  =·000005         DH.BF0  000002·      005 FD.FID·  000000      003 F.EFN·=·000050           LN5      000475R·    015
BYTE6  =·000006         DH.BF1  000004       005 FD.FNB·  000006      003 F.EOBB=·000032·          LN5E     000537R·    015
BYTE7  =·000007         DH.CTL  000000       005 FD.FVR·  000004      003 F.ERR·=·000052·          LN6      000537R·    015
BYTE8  =·000010         DH.DMC  000010       005 FD.LEN·  000010      003 F.FACC=·000043           LN6E     000573R·    015
BYTE9  =·000011         DH.FLG  000006       005 FD.RWM=·****** GX·       F.FFBY=·000014           LOC2   =·000014
BYTVAL=·000012·         DIRERR  001330RG     020 FIRST·   000604R·    015 F.FNAM=·000110           LOC4   =·000030
B.BSTA·  000054     010 DN.DCK  000000       013 FLIXMK=·177740 G·        F.FNB·=·000102·          LOGCLS=·000002 G·
B.CNTX·  000046     010 DN.NTP  000004       013 FLIXSZ=·000040 G·        F.FTYP=·000116           LOWADR·  000314RG·   015
B.CQUQ·  000060     010 DN.NXT  000006       013 FLUCLS=·000001 G·        F.FVER=·000120           LUNFIL=·000004
B.FEMA·  000132·    010 DN.ROT  000002·      013 FLUCUT=·000016 G·        F.HIBK=·000004           L$STAT=·000006 G·
B.FEMB·  000142·    010 DN.SIZ  000010       013 FLUMD·=·****** GX·       F.LUN·=·000042·          M·     =·000062
B.FEMC·  000152·    010 EFN.2·=·000002·          FME·  =·000006 G·        F.MBCT=·000054           MIN.1  =·000022
B.FFSA·  000202·    010 EIXCNT=·000010 G         FMEPSZ=·000400 G·        F.MBC1=·000055           MIN.2·=·000010
B.FFSB·  000212·    010 ELSBIT=·010000 G         FMIXSZ=·000001 G·        F.MBFG=·000056           MRGEN·   005600R·    020
B.FFSC·  000222·    010 ELSCLS=·000007 G         FMNPSZ=·002260 G·        F.NRBD=·000024           MSGOUT=·****** GX·
B.FMHR·  000172·    010 ELSDFN  000124RG     017 FMN1  =·000010 G·        F.NREC=·000030           MSG1     000316R·    015
B.FQLS·  000162·    010 ELSEID  000144R·     017 FMN2· =·000012·G·        F.OVBS=·000030           MSG2     000322R·    015
B.FSAZ·  000100     010 ELSIDX  000000R·     017 FNPOSZ=·000400 G·        F.RACC=·000016           MSG4     000326R·    015
B.FSBZ·  000102·    010 ELSMSK=·177761 G         FNPSZ·=·005734 G·        F.RATT=·000001           MSG5     000332R·    015
B.FSCZ·  000104     010 EMA·    016010RG     015 FN.DBR·  000026      011 F.RCNM=·000034           MSG6     000336R·    015
B.HBLK·  000120     010 EMACLS=·000003 G         FN.DBS·  000022·     011 F.RCTL=·000017           N·     =·000002
B.HDOC·  000114     010 EMACUT=·001700 G         FN.DHR·  000040      011 F.RSIZ=·000002·          NCN·     005420R·    020
B.HRLP·  000126     010 EMADNB  000220R·     015 FN.EMA·  000012·     011 F.RTYP=·000000           NCOM     005346R·    020
B.HRLR·  000122·    010 EMAFDB  000060RG     015 FN.EMB·  000014      011 F.SEQN=·000100           NDBCLS=·000006 G·
```

```
NDCID = 000004
NDEON = 000040
NDJSC = 000010
NDSE  = 000020
NDSKIP= 000200
NDTAD = 000100
NIXCNT= 000020 G
NMLCNT  000266R     015 PAR$$$= 000027
NMLSUM  000300R     015 PAR1    000574RG
NNMASK= 177760 G        PAR2    000576RG
NNPE    000120RG    017 PHI     000260RG
NODERR  001422RG    020 PRCELS  001746R
NP      000146RG    017 PRIOR   000262R
NPE     074146RG    017 PSTKMX= 000024 G
NPECNT= 000144 G        QE.RO1= 000144
NPEVSZ= 000043 G        QNDCNT= 001000 G
NPHI    000256RG    015 QRYSZ = 002000 G
NP1MSZ= 010000 G        QTS     000016R
NP2MSZ= 036000 G        QT2.1   000430R
NP2OSZ= 000153 G        QT2.3   000604R
NP3MSZ= 000524 G        QT2.7   000726R
NP3OSZ= 000125 G        Q.FDSC  000004
NXTADD  000602RG    015 Q.NQBK  000000
N.BACK= 000002 G        Q.NUHL  000002
N.BFAC= 000004          Q.SIZE  000014
N.BHGH= 000006          RECBUF  000022R
N.BTCH= 000004          REESET  000264R
N.BUFB= 004000          R.SUTN= 000002
N.BUFW= 002000          R.VAR = ****** GX
N.CIDP= 000014 G        R.VDBA= 000006
N.DID = 000024          R.VDTN= 000002
N.DVNM= 000032          SCAN  = ****** GX
N.EID = 000012 G        SD.FSA= 000010 G
N.ELSE= 000014 G        SD.NPS= 000016 G
N.FID = 000000          SD.SEC= 000012 G
N.FLG = 000005 G        SD.TIC= 000014 G
N.FMHS= 000012 G        SECBUF  000206RG
N.FMT = 000005 G        SEC.1 = 000024
N.FNAM= 000006          SEC.2 = 000012
N.FOS = 000764          SEG1  = 000000 G
N.FSA = 000010 G        SEG2  = 000002 G
N.FTYP= 000014          SEG3  = 000004 G
N.FVER= 000016          SNDBUF  000026R
N.LGT = 000006 G        SNPE    000122R
N.NEXT= 000022          SPLTN1  002006R
N.NHS = 000016 G        SPLTN2  002624R
N.PHI = 000004 G        SR.ARE  000114
N.PKSZ= 000020          SR.ARS  000106
N.PKTS= 000043          SR.DAY  000010
N.QURY= 000031          SR.DLT  000014
N.STAT= 000020          SR.ECB  000047
N.SUM = 000012 G        SR.ECH  000046
N.SUNT= 000002          SR.ECL  000050
N.TVCT= 000013 G        SR.FIB  000012

N.UNIT= 000034          SR.GRE  000100
NiIDX   000020R     017 SR.GRS  000072
NiP   = 000000 G        SR.LEN  000122
N2IDX   000060R     017 SR.LIN  000066
N2P   = 000002 G        SR.LIP  000062
OPNTDC  001000R     020 SR.MON  000006
PARBUF  000004R     015 SR.NDC  000042
                        SR.NDS  000036
                    020 SR.NIN  000030
                    015 SR.NIP  000022
                    015 SR.SDB  000032
                    020 SR.SRC  000002
                    015 SR.SUN  000000
                        SR.TWS  000056
                        SR.WSL  000052
                        SR.YR   000004
                        SR.1IN  000024
                    015 SR.1IP  000016
                    020 SSBIT = 004000 G
                    020 SS.FID  000002
                    020 SS.FNB  000010
                    007 SS.FVR  000006
                    007 SS.LEN  000012
                    007 SS.STT  000000
                    007 START   000000R
                    015 STPBK1= 000524
                    015 ST$CNG= 120000 G
                        ST$INR= 060000 G
                        ST$INX= 040000 G
                        ST$JSQ= 100000 G
                        ST$MAT= 160000 G
                        ST$SEQ= 140000 G
                    003 ST.ASZ  000020
                    003 ST.BSZ  000024
                    003 ST.BTC  000000
                    003 ST.CSZ  000030
                    014 ST.HRL  000010
                        ST.LEN  000044
                        ST.QRY  000002
                        ST.QSZ  000034
                        ST.SCH  000040
                        ST.UHL  000004
                    015 ST.XLT  000014
                    017 SU.DBU= 000004
                    020 SU.DON= 000006
                    020 SU.IDL= 000000
                    002 SU.LOD= 000001
                    002 SU.SRC= 000002
                    002 SU.SRR= 000005
                    002 SU.XPD= 000003
                    002 S.BFHD= 000020
                    002 S.DABA= 000006
                    002 S.DAEF= 000010
                    002 S.DATN= 000002

002 S.FATT= 000016          VILGT   015012R     015
002 S.FDB = 000140          VI0MSZ= 000400 G
002 S.FNAM= 000006          VI1MSZ= 000400 G
002 S.FNB = 000036          VI2MSZ= 000400 G
002 S.FNBW= 000017          VI3MSZ= 000400 G
002 S.FNTY= 000004          VMASK   014776RG    015
002 S.FTYP= 000002          VVEC    027010RG    015
002 S.HRL = 000240          WN.NTP  000004      012
002 S.NFEN= 000020          WN.NXT  000006      012
002 S1BIT = 001000 G        WN.ROT  000002      012
002 S2BIT = 002000 G        WN.SIZ  000010      012
002 S3BIT = 004000 G        WN.SRC  000000      012
002 TAEBIT= 020000 G        WN.TYP  000001      012
002 TAFLG   000302R     015 WORD0 = 000000
002 TASUM   000276R     015 WORD1 = 000002
002 TBIT  = 004000 G        WORD2 = 000004
002 TDABLK= 000004 G        WORD3 = 000006
002 TDABMX= 007764 G        WORD4 = 000010
    TDAMAD= 007760 G        WORD5 = 000012
004 TDBBLK= 000003 G        WORD6 = 000014
004 TDBCLS= 000005 G        WORD7 = 000016
004 TDBMAD= 007775 G        WORD8 = 000020
004 TDBOSZ= 000074 G        WORD9 = 000022
004 TDCADR  000312RG    015 WRDVAL= 000024
020 TDCBLK= 000010 G        WRTTDC  001246R     020
    TDCBUF  000606R     015 XBATCH= 000013
    TDCEND  014606R     015 XDBLOA= 000004
    TDCERR  001454R     020 XDBPRO= 000012
    TDCHK   001546RG    020 XDMCIN= 000006
    TDCLGT  000610R     015 XFOSMR= 000007
    TDCMSZ= 004000 G        XGTSRE= 000014
    TDCT    000616R     015 XHITSK= 000011
006 TIC.1 = 000026          XHLMER= 000002
006 TIC.2 = 000014          XHOTSK= 000010
006 TRMCUT= 000040 G        XMSCHE= 000000
006 TSTKMX= 000400 G        XQTS  = 000003
006 TVCNT   000272R     015 XQT0  = 000001
006 TXTBIT= 010000 G        XSULOA= 000005
006 T.JSBY= 000000 G        $DSW  = ****** GX
006 T.MATC= 000000 G        $$$   = 000022R     021
006 T.NBAS= 000002 G        $$$OST= 000006
006 T.NDEF= 000000 G        $$$T1 = 000003
006 T.SBY1= 000000 G        .CLOSE= ****** G
    T.SBY2= 000001 G        .FINIT= ****** G
    T.SBY3= 000002 G        .FSRCB= ****** G
    T.STAD= 000002 G        .OPEN = ****** G
    T.TRAN= 000000 G        .OPFNB= ****** G
    T.TYPW= 000004 G        .WAIT = ****** G
    VEC1MX= 000375 G        .WRITE= ****** G
    VEC2MX= 000375 G        .XQIO = ****** GX
    VEC3MX= 000125 G        ...PC1= 000060R     015
    VI      015014R     015 ...PC2= 000254R     015
    VIBCUT= 000036 G        ...PC3= 000060R     015
    VIERR   001506RG    020 ...TPC= 000020

. ABS.  000000      000
        000000      001
```

```
SRCOFF  000122     002
FDSCOF  000010     003
SUSOFF  000012     004
DHROFF  000012     005
STTOFF  000044     006
QSPLOF  000014     007
BSTOFF  000772     010
FNOFFS  000044     011
WNODOF  000010     012
DNODOF  000010     013
TRNOFF  000240     014
DATA    030022     015
$$FSR1  000000     016
NODES   074474     017
CODE    005756     020
$DPB$$  000030     021
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  8345 WORDS  (33 PAGES)
DYNAMIC MEMORY:  9140 WORDS  ( 35 PAGES)
ELAPSED TIME:  00:01:38
QT2,QT2/-SP/NL:BEX:ME=[20,1]P,M,T,QTSIZE,QT2
```

```
                        ; QTSIZE.MAC    "QUERY TRANSLATORS BUFFER SIZE CONFIGUATION FILE"
                        ;       THIS FILE CONTROLS THE SIZE OF ALL BUFFERS THAT CAN VARY
                        ;       IN SIZE DUE TO THE AMOUNT OF QUERIES OR OTHER SUCH PARAMETERS
                        ;       THAT WOULD BE CHARACTERISTIC OF A SITE.  THESE BUFFERS ARE
                        ;       IN QT0, QT1, QT2, OR QT3.
                        ;
                        ;---QT3---BUFFERS-------------------------------------------------------
                        ;
000125                  VEC3MX==85.                             ;MAXIMUM # CWP'S IN BATCH
000025                  AVELGT==3*2*7/2                         ;AVERAGE CWP SIZE IN TDCT (BYTES)
000400                  VI3MSZ==VEC3MX+2/256.+1*256.            ;SIZE OF VI (NEAREST BLOCK IN BYTES)
001000                  EMCMSZ==VEC3MX/51.+1*256.               ;SIZE OF EMC (NEAREST BLOCK)
004000                  TDCMSZ==VEC3MX*AVELGT+8./N.BUFW+1*N.BUFW;SIZE OF TDCT
000010                  TDCBLK==TDCMSZ/256.                     ;VIRTUAL BLOCK SIZE OF TDCT
000524                  NP3MSZ==VEC3MX*4                        ;SIZE OF NODE POOL
000125                  NP3OSZ==VEC3MX                          ;SIZE OF NODE POOL OVERFLOW AREA
                        ;
                        ;-----QT2----BUFFERS---------------------------------------------------
                        ;
000375                  VEC2MX==253.                            ;MAXIMUM # VECTORS
000400                  VI2MSZ==VEC2MX+2/256.+1*256.            ;SIZE OF VI
004400                  EMBMSZ==9.*256.                         ;SIZE OF EMB
000003                  TDBBLK==3                               ;#BLOCKS (N.BUFW) IN TDCT BUFFER
000074                  TDBOSZ==3*20.                           ;SIZE OF TDCT BUFFER OVERFLOW
007775                  TDBMAD==4093.                           ;MAXIMUM ADDRESS (48 BITS) IN TDCT
000020                  NIXCNT==16.                             ;#INDEX POINTERS FOR NORMAL NODES
177760                  NNMASK==177760                          ;MASK FOR NORMAL INDEX
000010                  EIXCNT==8.                              ;#INDEX PTRS FOR ELSE NODES
177761                  ELSMSK==177761                          ;MASK FOR ELSE INDEX
000004                  ENCNT==4                                ;#ENTRIES TO USE FOR HASH
036000                  NP2MSZ==256.*60.                        ;SIZE OF NODE POOL
000153                  NP2OSZ==7+100.                          ;SIZE OF NODE POOL OVERFLOW AREAR
                        ;
                        ;------QT1-----BUFFERS--------------------------------------------------
                        ;
000375                  VEC1MX==253.                            ;MAX. #TERMS IN FSA-A
000400                  VI1MSZ==VEC2MX+2/256.+1*256.            ;SIZE OF VI
000004                  TDABLK==4                               ;#BLOCKS (N.BUFW) IN TDCT BUFFER
007764                  TDABMX==<TDABLK*<N.BUFW-4>>+4
016000                  EMAMSZ==7.*4*256.                       ;SIZE OF EMA
007760                  TDAMAD==340.*12.                        ;MAX. ADDRESS IN TDCT
010000                  NP1MSZ==4096.                           ;SIZE OF NORMAL NODE POOL
000144                  NPECNT==100.                            ;#NODES IN ELSE POOL
000043                  NPEVSZ==35.                             ;#VECTORS IN ELSE NODES
                        ;
                        ;-----FLU---MODIFIER----BUFFERS-----------------------------------------
                        ;
002260                  FMNPSZ==1200.                           ;SIZE OF NORMAL FLU MOD NODE POOL
000400                  FMEPSZ==256.                            ;SIZE OF ELSE FLU MOD NODE POOL
                        ;
                        ;------QT0------BUFFERS-----------------------------------------------
                        ;
002000                  QRYSZ==N.BUFW                           ;SIZE OF QUERY BUFFER
000040                  FLIXSZ==32.                             ;#INDEX PTRS FOR TERMS
177740                  FLIXMK==177740                          ;MASK FOR TERM INDEX
000010                  CPIXSZ==8.                              ;#INDEX PTRS FOR CWP'S
177770                  CPIXMK==177770                          ;MASK FOR CWP INDEX
```

```
                        FMIXSZ==1                               ;#· INDEX·PTRS·FOR·FLU-MOD'S·
                        FNPSZ==3036.                            ;FLU·NODE·POOL·SIZE·
                        FNPOSZ==256.                            ;SIZE·OF·POOL·OVERFLOW·AREA·
                        TSTKMX==256.                            ;MAX. #·TOKENS·TO·BE PUSHED·
                        ASTKMX==256.                            ;MAX. #·ARGUMENTS·TO BE·PUSHED
                        PSTKMX==20.                             ;MAX. #·PROX·NODES·IN·A·CHAIN·
                        HSTKMX==64.                             ;MAX. NESTING·OF·QUERY·
                        QNDCNT==512.                            ;#·NODES·IN·LOGIC·POOL·
                        VI0MSZ==VI1MSZ·                         ;ASSUME·MAX·VI·IS·IN QT1
                        .IIF·LT,VI0MSZ-VI2MSZ,VI0MSZ==VI2MSZ·   ; IF·NOT, RESET·QT0'S·TO·QT2'S VI·SIZE·
                        EMXMSZ==EMAMSZ·                         ;ASSUME·MAX·EMX·IS·IT·QT1
                        .IIF·LT,EMXMSZ-EMBMSZ,EMXMSZ==EMBMSZ·   ; IF·NOT, RESET·TO·QT2'S·
                        ;
                        ;-----ERROR·AND·CLOSE·CODES----------------------------------------
                        ;
                        TRMCUT==32.     ;TERM·CUT-OFF·
                        FLUCUT==14.     ;FLU CUT-OFF·
                        EMACUT==TRMCUT*30.      ;EMA·CUT-OFF·
                        VIBCUT==30.             ;VIB·CUT-OFF·
                        EMBCUT==VIBCUT*20.      ;EMB·CUT-OFF·
                        ;
                        FLUCLS==1       ;CLOSED·DUE·TO FLU·COUNT
                        LOGCLS==2·      ;CLOSED·DUE·TO LOGIC·COUNT·(QLB,SDLB,QEX)
                        EMACLS==3       ;CLOSED·DUE·TO EMA·SIZE·
                        EMBCLS==4       ;CLOSED·DUE·TO EMB·SIZE·
                        TDBCLS==5       ;CLOSED·DUE·TO TDCTB·SIZE·
                        NDBCLS==6       ;CLOSED·DUE·TO QT2·NODE·POOL·SIZE·
                        ELSCLS==7       ;CLOSED·DUE·TO OTHER·CONDITIONS·(FLU·POOL,ETC)
                        ;
```

```
                                        .TITLE  FLU-MOD CODE
                                        .MCALL  MOUT$S
                                ;
                                ;   THIS IS THE DATA AREA FOR FLU-MODIFIER CODE.
                                ;   EQUATES
                                ;
          000010                MCTYP=10          ;FLU-MOD TYPE AT END OF SCAN
                                ;
 000000                                 .PSECT  FMDATA
                                ;   POOLS
                                ;
 000000                         FMNPS:  .BLKW   FMNPSZ  ;NORMAL NODE POOL (START)
 004540                         FMNPE:                  ;               (END)
 004540                         FMEPS:  .BLKW   FMEPSZ  ;ELSE NODE POOL (START)
 005540                         FMEPE:                  ;              (END)
                                ;
                                ;    VECTOR STORAGE FOR FMTST1 & FMTST2
                                ;
 005540                         MIXFMC: .BLKW   20.
                                ;
                                ;   CONVERSION TABLES
                                ;
 005610  000000G 000000G 000000G FMTYP:  .WORD   EMXDTF,EMXDTF,EMXZNF,EMXZNF,EMXSZF,EMXSZF,EMXMCF,EMXMCF
 005630  002000  002000  001000  FMTSC:  .WORD   BIT10,BIT10,BIT9,BIT9,BIT10!BIT9,BIT10!BIT9,0,0
                                ;
                                ;   POINTERS
                                ;
 005650  000000                 CFMN:   .WORD   0       ;CURRENT FLU-MOD NODE
 005652  000000                 NFMN:   .WORD   0       ;NEXT FLU-MOD NODE
 005654  000000                 FMNI:   .WORD   0       ;FLU-MOD NODE INDEX
 005656  000000                 CFME:   .WORD   0       ;CURRENT ELSE NODE
 005660  000000                 NFME:   .WORD   0       ;NEXT ELSE NODE
 005662  000000                 FMEI:   .WORD   0       ;ELSE NODE INDEX
 005664  000000                 LVLDC:: .WORD   0       ;LAST VLDC EMX LOC (GVMASK)
                                ;
                                ;   STORAGE WORDS
                                ;
 005666  000000                 CFSA:   .WORD   0       ;CURRENT FSA LOC
 005670  000000                 CFMT:   .WORD   0       ;CURRENT FMT FOR 'GETNDE'
 005672  000000                 FMT:    .WORD   0       ;FLU-MOD TYPE FOR 'GVMASK'
 005674  000000                 FMTSUM: .WORD   0       ;FMT SUMMARY FOR 'IMASK' & 'FMSEQ'
 005676  000000                 FMSC:   .WORD   0       ;SOURCE CODE FOR FMT
 005700  000000                 FMEXL:  .WORD   0       ;EMA POINTER FOR FLU-MOD VECTOR STARTS
 005702  000000                 GVSUB:  .WORD   0       ;'GVMASK' SUB-SUBROUTINE POINTER
 005704  000000                 OLDMC:  .WORD   0       ;OLD MATCH CODE
                                ;
                                ;   FLAG WORDS
                                ;
 005706  000000                 FMLOG:  .WORD   0       ;FLU-MOD SCAN LOGIC ENABLE
 005710  000000                 FMEXF:  .WORD   0       ;FLU-MOD TYPE (-2,-1,0,1=MC,MS,S,C)
                                                        ;M=MULTIPLE S=SIMPLE C=COMPLEX
 005712  000000                 FMEXF2: .WORD   0       ;DIFFERENT FLU NODES
 005714  000000                 FMIN:   .WORD   0       ;ELSE VECTOR
 005716  000000                 FVAL:   .WORD   0       ;NORMAL VECTOR
 005720  000000                 FMAT:   .WORD   0       ;MATCH VECTOR
 005722  005724'                VALVPT: .WORD   VALVEC  ;POINTER TO NEXT VECTOR SAVE LOCATION
 005724                         VALVEC: .BLKW   20.     ;SAVE AREA FOR VECTORS
```

```
     58 005774  000000                ENFLAG: .WORD   0       ;ELSE SEARCH ENABLE 'GETNDE'
     59 005776  000000                NEST:   .WORD   0       ;NESTED INPUT TO GVMASK SUB-SUBROUTINE FLAG
     60 006000  000000                LEVEL:  .WORD   0       ;LEVEL OF NESTING OF EXP. VECTORS
     61                               ;
```

```
                                        .PSECT  FMCODE
                                ;   THIS SUBROUTINE IS THE MAIN CONTROL SUBROUTINE FOR
                                ;   FLU-MOD PROCESSING, WHEN IT IS CALLED, BUFFER VVEC
                                ;   CONTAINS THE VECTOR FOR THE INTERWORD CHARACTER WHICH
                                ;   DESIGNATES THE END OF THE FLU AND THE START OF THE
                                ;   THE FLU-MODIFIER.  NOTE: REGISTERS HAVE NO MEANING AT
                                ;   THE TIME FLUMD IS CALLED, EXCEPT R3, WHICH POINTS TO
                                ;   THE NODE THAT IS TO BE PROCESSED.
000000                          FLUMD::
000000  005367  005706'                DEC     FMLOG                   ;TRIGGER FLU-MOD PROCESSING
000004  012704  000000G                MOV     #VVEC,R4                ;R4=FLU-MOD NODE
000010  016364  000000G 000000G        MOV     N.FSA(R3),N.FSA(R4)     ;XFER FSA LOC
000016  012764  000000G 000000G        MOV     #FME,N.PHI(R4)          ;NODE TYPE = ELSE NODE
000024  016300  000000G                MOV     N.LGT(R3),R0            ;R0 = VECTOR SIZE
000030  020027  000001                 CMP     R0,#1                   ;SINGLE VECTOR?
000034  001426                        BEQ     100$                    ;YES
000036  012767  177777  005710'        MOV     #-1,FMEXF               ;NO: FLAG AS DIFFERENT MATCH'S
000044  005267  005712'                INC     FMEXF2                  ;FLAG AS MULTIPLE NODES
000050  010067  005540'                MOV     R0,MIXFMC               ;SAVE COUNT AND LOCATION
000054  006367  005540'                ASL     MIXFMC                  ;ADJUST COUNT AS IF EXP, VECTOR
000060  012767  005540' 005700'        MOV     #MIXFMC,FMEXL           ;LOCATION
000066                                  SAVE    R0
000070  012702  005542'                MOV     #MIXFMC+2,R2            ;SET UP TO SAVE VECTOR
000074  010305                        MOV     R3,R5
000076  062705  000000G                ADD     #N.NHS,R5
000102  012522                 40$:    MOV     (R5)+,(R2)+
000104  077002                        SOB     R0,40$                  ;SAVE VECTOR
000106                                  RESTOR  R0
000110  000404                        BR      101$
000112  005067  005710'         100$:  CLR     FMEXF
000116  005067  005712'                CLR     FMEXF2
000122  010064  000000G         101$:  MOV     R0,N.LGT(R4)            ;XFER NODE SIZE
000126  105064  000000G                CLRB    N.FMT(R4)               ;FLU-MOD TYPE = DOCUMENT/NIB1
000132  062704  000000G                ADD     #N.FMHS,R4              ;XFER VECTOR
000136  062703  000000G                ADD     #N.NHS,R3
000142  012324                 4$:    MOV     (R3)+,(R4)+
000144  077002                        SOB     R0,4$
000146  012703  000000G                MOV     #VVEC,R3                ;R3=FLU-MOD NODE
000152  012767  000000G 005672'        MOV     #EMXDTF,FMT     ;SCAN FOR DOC-TYPE
000160  005067  005662'                CLR     FMEI            ;RESET INDEX POINTERS
000164  005067  005654'                CLR     FMNI
000170  005067  005670'                CLR     CFMT            ;SCAN FOR DOC-TYPE
000174  012767  004540' 005656'        MOV     #FMEPS,CFME     ;RESET POOL POINTERS
000202  012767  004540' 005660'        MOV     #FMEPS,NFME
000210  012767  000000' 005650'        MOV     #FMNPS,CFMN
000216  012767  000000' 005652'        MOV     #FMNPS,NFMN
000224  012767  002000  005676'        MOV     #BIT10,FMSC     ;SOURCE CODE=DOC TYPE
000232  004767  000032          1$:    JSR     PC,GVMASK       ;STEP VECTOR
000236  004767  001024                JSR     PC,IMASK        ;SPLIT VECTOR
000242  004767  004752                JSR     PC,GETNDE       ;GET NEXT VECTOR
000246  103405                        BCS     10$             ;IF FINISHED
000250  016700  000000G                MOV     LOWADR,R0       ;R1 = MINIMUM ADDRESS
000254  004767  000000G                JSR     PC,TDCHK        ;CHECK FOR TDCT OVERFLOW
000260  000764                        BR      1$
000262  005067  005706'         10$:   CLR     FMLOG           ;SWITCH BACK TO TEXT CHARS
000266  000207                        RTS     PC              ;IF NO MORE VECTORS
```

```
                                ;
                                ;   GVMASK IS THE SUBROUTINE THAT GIVEN A VECTOR FOR THE
                                ;   CURRENT NIBBLE POSITION, GENERATES A VECTOR FOR THE
                                ;   NEXT NIBBLE POSITION. GVMASK HANDLES TEXT CHARACTERS
                                ;   AS WELL AS FLU-MOD TERMS.
                                ;       INPUT                   OUTPUT
                                ;   R3= NODE ADR OF GIVEN       ADR OF GENERATED VECTOR
                                ;       VECTOR                  (VMASK)
                                ;
                                ;   NO OTHER REGISTERS HAVE MEANING FOR I/O
                                ;
000270                          GVMASK::
000270  010301                          MOV     R3,R1           ;INITIALIZE REGISTERS
000272  016100  000000G                 MOV     N.LGT(R1),R0
000276  001467                          BEQ     10$
000300  016367  000000G 000000C         MOV     N.PHI(R3),VMASK+N.PHI
000306  016367  000000G 000000C         MOV     N.ELSE(R3),VMASK+N.ELSE ;COPY HEADER FIELDS
000314  016367  000000G 000000C         MOV     N.FSA(R3),VMASK+N.FSA
000322  016367  000000G 000000C         MOV     N.EID(R3),VMASK+N.EID
000330  116304  000000G                 MOVB    N.PHI(R3),R4
000334  066401  004144'                 ADD     NDSIZ(R4),R1
000340  012705  000000G                 MOV     #VMASK,R5
000344  066405  004144'                 ADD     NDSIZ(R4),R5
000350  005002                          CLR     R2
000352  005067  005776'                 CLR     NEST
000356  000174  000362'                 JMP     @100$(R4)       ;PROCESS ACCORDING TO NODE TYPE
000362  000376'                 100$:   1$      ;NIB1 NODE
000364  000416'                         2$      ;NIB2 NODE
000366  000416'                         2$      ;ELSE NODE
000370  000426'                         3$      ;FLU MOD ELSE NODE
000372  000426'                         3$      ;FLU MOD NIB1 NODE
000374  000376'                         1$      ;FLU MOD NIB2 NODE
000376  012125                  1$:     MOV     (R1)+,(R5)+     ;TRANSFER VECTOR
000400  077002                          SOB     R0,1$
000402  016367  000000G 000000C         MOV     N.LGT(R3),VMASK+N.LGT   ;TRANSFER SIZE
000410  012703  000000G                 MOV     #VMASK,R3       ;R3=NEW NODE
000414  000207                          RTS     PC
000416  012767  000462' 005702' 2$:     MOV     #GV2,GVSUB      ;USE GV2 SUBROUTINE
000424  000405                          BR      4$
000426  012767  000654' 005702' 3$:     MOV     #GV3,GVSUB      ;USE GV3 SUBROUTINE
000434  005067  005674'                 CLR     FMTSUM          ;INIT SUMMARY
000440  004777  005702'          4$:    JSR     PC,@GVSUB       ;CALL SUB
000444  010267  000000C                 MOV     R2,VMASK+N.LGT  ;TRANSFER SIZE
000450  012703  000000G                 MOV     #VMASK,R3       ;R3=NEW NODE
000454  000207                          RTS     PC
000456  000167  177775          10$:    JMP     .+1
```

```
                                        ;
                                        ;   SUBROUTINE·GV2·HANDLES·STEPPING·VECTORS·FOR·TEXT·CHAR'S.
                                        ;
000462· 012104                  GV2:    MOV·    (R1)+,R4        ;GET·EMA POINTER·
000464  016403  000000G·                MOV·    EMA(R4),R3      ;GET·CONTROL·WORD·
000470  100462·                         BMI·    10$             ;IF·NON-SEQ·FLOW·
000472· 005767  005776'                 TST·    NEST            ;IF·NESTED, DON'T·STEP·VECTORS·
000476  001050                          BNE·    2$              ;NESTED·
000500  032703  000000G·                BIT·    #EMXMTV,R3      ;DON'T·CARE?·
000504                                  BOFF·   1$              ;NO·
000506  032703  000000C·                BIT·    #EMXVDC!EMX0VD!EMXNVD,R3        ;IS·IT·VAR·LENGTH?·
000512·                                 BOFF·   1$              ;NO·
000514  032703  000000G·        20$:    BIT·    #EMXVDC,R3      ;IS·IT·A VLDC?·
000520                                  BOFF·   22$             ;NO·
000522· 010567  005664'          21$:   MOV·    R5,LVLDC·       ;SAVE·VLDC·LOCATION·
000526  010425                  22$:    MOV·    R4,(R5)+        ;TRANSFER·ENTRY·
000530  005202·                         INC·    R2·
000532· 020227  000375                  CMP·    R2,#VEC2MX·     ;EXCEED·LIMIT?·
000536  101042·                         BHI·    VIER
000540  062704  000006          1$:     ADD·    #6,R4           ;STEP·TO NEXT·ENTRY·
000544  016403  000000G·                MOV·    EMA(R4),R3      ;GET·CONTROL·WORD·
000550  100432·                         BMI·    10$             ;IF·NON-SEQ·
000552· 032703  000000G·                BIT·    #EMXMTV,R3      ;DON'T·CARE?·
000556                                  BOFF·   2$              ;NO·
000560  032703  000000C·                BIT·    #EMXVDC!EMX0VD!EMXNVD,R3        ;IF·VAR·LENGTH·
000564                                  BOFF·   2$
000566  032703  000000G·                BIT·    #EMXVDC,R3      ;IS·IT·A VLDC?·
000572·                                 BOFF·   22$             ;NO·
000574  032703  000000G·                BIT·    #EMXVVV,R3      ;WAS·THERE·A·PREVIOUS·ONE·IN TERM?·
000600                                  BOFF·   21$             ;NO·
000602· 166705  005664'                 SUB·    LVLDC,R5        ;YES: WIPE·OUT·ALL PRIOR·ENTRIES·FOR·TERM·
000606  006205                          ASR·    R5
000610  160502·                         SUB·    R5,R2·          ;REMOVE·PROIR·ENTRIES·FROM·COUNT·
000612· 016705  005664'                 MOV·    LVLDC,R5        ;REMOVE·FROM·VECTOR·
000616  000743                          BR      22$
000620  010425                  2$:     MOV·    R4,(R5)+        ;TRANSFER·ENTRY·
000622· 005202·                         INC·    R2·
000624  020227  000375                  CMP·    R2,#VEC2MX·     ;EXCEED·LIMIT·
000630  101005                          BHI·    VIER
000632· 077065                  3$:     SOB·    R0,GV2·         ;CONTINUE·UNTIL·FINISHED·VECTOR·
000634  000207                          RTS·    PC·
000636  004767  000242          10$:    JSR·    PC,NSQSUB·      ;PROCESS NON-SEQ·ENTRY·
000642· 000773                          BR      3$
000644  010267  000000G·        VIER:   MOV·    R2,PAR1
000650  000167  000000G·                JMP·    VIERR·
                                        ;
                                        ;   SUBROUTINE·GV3 HANDLES·STEPPING·OF·VECTOR·FOR·FLU-MOD'S.
                                        ;
000654  012104                  GV3:    MOV·    (R1)+,R4        ;GET·EMA POINTER·
000656  100406                          BMI·    1$              ;IF·NEGATIVE·MAKE·POSITIVE·
000660  005767  005776'                 TST·    NEST            ;IF·NESTED,DON'T·STEP·VECTOR
000664  001004                          BNE·    2$              ;NESTED·
000666  062704  000006                  ADD·    #6,R4           ;IF·POS·&·NOT·NESTED, STEP·ENTRY·
000672· 000401                          BR      2$
000674  005404                  1$:     NEG·    R4
000676· 016403  000000G·        2$:     MOV·    EMA(R4),R3      ;GET·ENTRY·
000702· 100414                          BMI·    3$              ;IF·NON-SEQ·FLOW...
```

```
  224 000704  036703  005672'                BIT·    FMT,R3          ;ENTRY·OF·TYPE·DESIRED?·
  225 000710                                 BON·    4$              ;YES·
  226 000712· 005404                         NEG·    R4              ;NO: NEGATE·
  227 000714  050367  005674'          4$:   BIS·    R3,FMTSUM·      ;NOTE·TYPE·IN·SUMMARY·
  228·000720  010425                         MOV·    R4,(R5)+        ;XFER·ENTRY·
  229 000722· 005202·                        INC·    R2·             ;1 MORE·ENTRY·IN·VECTOR·
  230 000724  020227  000375                 CMP·    R2,#VEC2MX·     ;EXCEED·LIMIT·
  231 000730  101345                         BHI·    VIER            ;YES·
  232·000732· 000433                         BR      5$
  233 000734  032703  040000           3$:   BIT·    #BIT14,R3       ;MULTIPLE·FLU-MOD'S·
  234 000740                                 BON·    31$             ;YES·
  235 000742· 005767  005710'                TST·    FMEXF·          ;NO:HAD·IT·OCCURRED·BEFORE?·
  236 000746  100004                         BPL·    32$             ;NO·
  237 000750  012767  177776  005710'       MOV·    #-2,FMEXF·      ;FLAG·AS UNKNOWN·
  238 000756  000417                         BR      33$
  239 000760  012767  000001  005710' 32$:   MOV·    #1,FMEXF·       ;FLAG·AS SAME·MATCH·CODE·
  240.000766  000413                         BR      33$
  241 000770  005767  005712'          31$:  TST·    FMEXF2·
  242·000774  100414                         BMI·    34$             ;IF·DIFFERENT·FLU·NODES·
  243 000776  012767  177777  005710'       MOV·    #-1,FMEXF·      ;FLAG·AS ALWAYS·DIFFER·
  244 001004  010467  005700'                MOV·    R4,FMEXL·       ;SAVE·EXPANSION·VECTOR·LOCATION·
  245 001010  062767  000000G·005700'        ADD·    #EMA,FMEXL·
  246 001016  004767  000062           33$:  JSR·    PC,NSQSUB·      ;EXPAND·VECTOR·
  247 001022· 077064                   5$:   SOB·    R0,GV3          ;DO·FOR·ALL·ENTRIES·
  248 001024  000207                         RTS·    PC·
  249 001026                           34$:  SAVE·   R4,R0,R5         ;IF·DIFFERENT·
  250 001034  016705  005700'                MOV·    FMEXL,R5
  251 001040  012500                         MOV·    (R5)+,R0        ;GET·COUNT·IF·VECTORS·
  252·001042· 006200                         ASR·    R0
  253 001044  020425                   300$: CMP·    R4,(R5)+        ;FIND·ENTRY·FOOR·THIS·POINTER·
  254 001046  103404                         BLO·    301$            ;FOUND·
  255 001050  077003                         SOB·    R0,300$
  256 001052· 162705  000002                 SUB·    #2,R5           ;FOUND: PRIOR·ENTRY·
  257 001056  000402·                        BR      302$
  258 001060  162705  000004           301$: SUB·    #4,R5           ;PRIOR·PRIOR·ENTRY
  259 001064  062704  000000G·         302$: ADD·    #EMA,R4         ;CONVERT OFFSET·TO LOCATION·
  260 001070  005404                         NEG·    R4              ;FLAG·AND·RE-ENTER IT·
  261 001072· 010415                         MOV·    R4,(R5)
  262·001074                                 RESTOR· R4,R0,R5
  263 001102· 000745                         BR      33$
```

```
                                        ;
                                        ;   SUBROUTINE·NSQSUB·EXPANDS·VECTORS·DUE·TO·NON-SEQ·FLOW·
                                        ;
001104                          NSQSUB: SAVE·   R0,R1
001110  032703  000000G·                BIT·    #EMXEXF,R3      ;MERGE?·
001114                                  BOFF·   1$              ;YES·
001116  005267  005776'                 INC·    NEST            ;FLAG·AS NESTED·
001122· 010401                          MOV·    R4,R1           ;ADJUST·EXPAND·VECTOR·AS· INPUT·
001124  042703  000000C·                BIC·    #EMXNSQ!BIT14!EMXEXF,R3
001130  006203                          ASR·    R3
001132· 010300                          MOV·    R3,R0
001134  062701  000002G·                ADD·    #EMA+2,R1       ;R1,R0=VECT0R·ADR, &·SIZE·
001140  004777  005702'                 JSR·    PC,@GVSUB·      ;RECALL·CALLER·
001144  005367  005776'                 DEC·    NEST            ;REMOVE·1 LEVEL·OF NEST· INDICATION·
001150                                  RESTOR· R0,R1
001154  000207                          RTS·    PC·
001156                          1$:     SAVE·   R5
001160  042703  000000G·                BIC·    #EMXNSQ,R3      ;REMOVE·FLAG·
001164  010304                          MOV·    R3,R4           ;STANDARDIZE· INPUT
001166  016403  000000G·        5$:     MOV·    EMA(R4),R3      ;GET·NEW ENTRY·
001172· 100423                          BMI·    10$             ;IF·ANOTHER·MERGE·
001174  012700  000000C·                MOV·    #VMASK+N.NHS,R0 ;POINT·TO·1ST·ENTRY·
001200  020500                  2$:     CMP·    R5,R0           ;ALL·ENTRIES·CHECKED?·
001202· 101403                          BLOS·   4$              ;YES·
001204  024504                          CMP·    -(R5),R4        ;DOES·PRIOR·ENTRY·MATCH?·
001206  001423                          BEQ·    3$              ;YES; THEN·DON'T·TRANSFER·
001210  000773                          BR      2$              ;NO·
001212·                         4$:     RESTOR· R5
001214  010425                          MOV·    R4,(R5)+        ;TRANSFER·ENTRY·
001216  005202·                         INC·    R2·
001220  032703  000000G·                BIT·    #EMXMTV,R3      ;DON'T·CARE?·
001224                                  BOFF·   6$              ;NO·
001226  032703  000000C·                BIT·    #EMX0VD!EMXNVD,R3       ;VARIBLE·LENGTH?·
001232·                                 BOFF·   6$              ;NO·
001234  062704  000006                  ADD·    #6,R4           ;REPEAT·FOR·NEXT·ENTRY·
001240  000752·                         BR      5$
001242·                         10$:    RESTOR· R5
001244  004767  177634                  JSR·    PC,NSQSUB·      ;PROCESS MERGE·
001250                          6$:     RESTOR· R0,R1
001254  000207                          RTS·    PC·
001256                          3$:     RESTOR· R0,R1,R5
001264  000207                          RTS·    PC·
```

```
                                ;
                                ;   SUBROUTINE IMASK IS RESPONSIBLE FOR THE MAJOR PORTION OF
                                ;   FLU-MOD PROCESSING.  IT SPLITS A VECTOR INTO SUCCESIVE VECTORS
                                ;   IF MULTIPLE PATHS EXIST FROM THAT STATE.  IT ALSO CREATES THE
                                ;   TDCT STATES FOR THE VECTORS.
                                ;
                                ;   ONLY R3 IS USED FOR I/O.  AS INPUT IT POINTS TO THE VECTOR TO
                                ;   BE SPLIT WHICH IMASK ALSO KNOWS IS VMASK.  NO REGISTERS HAVE
                                ;   MEANING FOR OUTPUT.
                                ;
001266  016367  000000G 005666' IMASK:  MOV     N.FSA(R3),CFSA  ;STORE FSA ADR FOR NODE
001274  016301  000000G                 MOV     N.LGT(R3),R1    ;R1=VECTOR SIZE
001300  016367  000000G 000000G         MOV     N.PHI(R3),PHI   ;STORE NODE TYPE
001306  062703  000000G                 ADD     #N.FMHS,R3      ;R3=INPUT VECTOR ENTRIES
001312  012705  000000C                 MOV     #VVEC+N.FMHS,R5 ;R5=OUTPUT VECTOR ENTRIES
001316  126727  000000G 000000G         CMPB    PHI,#FMN2       ;NIB2?
001324  001015                         BNE     10$             ;NO
001326                          1$:     SAVE    R1,R3
001332  012767  000000G 000000G         MOV     #FMN1,NPHI      ;NEW NODE TYPE WILL BE NIB1
001340  016705  005666'          2$:    MOV     CFSA,R5         ;CONVERT FSA LOC TO TDCT ADR.
001344  070527  000006                  MUL     #6,R5
001350  066705  000000G                 ADD     TDCADR,R5
001354  000167  000344                  JMP     1000$
001360                          10$:    SAVE    R1,R3
001364  012767  000000G 000000G         MOV     #FMN2,NPHI      ;NEW NODE TYPE WILL BE NIB2
001372  005000                          CLR     R0              ;R0=ENTRIES IN ELSE VECTOR
001374  022767  000000G 005672'         CMP     #EMXMCF,FMT     ;MATCH STATE SCAN?
001402  001756                          BEQ     2$              ;YES
001404  012304                  11$:    MOV     (R3)+,R4        ;GET EMA POINTER
001406  100002                          BPL     12$             ;XFER & COUNT ONLY ELSE'S
001410  010425                          MOV     R4,(R5)+
001412  005200                          INC     R0
001414  077105                  12$:    SOB     R1,11$
001416  016705  005666'                 MOV     CFSA,R5         ;CONVERT FSA LOC TO TDCT ADR.
001422  070527  000006                  MUL     #6,R5
001426  066705  000000G                 ADD     TDCADR,R5
001432  005700                          TST     R0              ;ANY ELSE VECTOR
001434  001474                          BEQ     100$            ;NO
001436  016725  000000G                 MOV     NXTADD,(R5)+    ;POINT CHANGE STATE TO IT
001442  016767  000000G 000000C         MOV     NXTADD,VVEC+N.FSA       ;ENTER NODE'S FSA LOC
001450  005267  000000G                 INC     NXTADD
001454  116767  005670' 000000C         MOVB    CFMT,VVEC+N.FMT ;ENTER FMT
001462  105267  000000C                 INCB    VVEC+N.FMT
001466  105267  000000C                 INCB    VVEC+N.FMT
001472  010067  000000C                 MOV     R0,VVEC+N.LGT   ;ENTER SIZE
001476  112767  000000G 000000C         MOVB    #FME,VVEC+N.PHI ;NODE TYPE = ELSE
001504  012703  005662'                 MOV     #FMEI,R3        ;CHECK FOR DUPLICATE
001510  012702  000000G                 MOV     #VVEC,R2
001514                                  SAVE    R0
001516  012700  000000G                 MOV     #FME,R0         ;NODE TYPE = ELSE
001522  004767  002076                  JSR     PC,SCAN
001526                                  RESTOR  R0
001530  016703  000000C                 MOV     VMASK+N.LGT,R3  ;GET OLD SIZE
001534  103437                          BCS     14$             ;NEW VECTOR
001536  004767  002446                  JSR     PC,FMTST1       ;DO NON-ELSE ENTRIES MATTER?
001542  103421                          BCS     16$             ;NO
001544  005367  000000G                 DEC     NXTADD          ;OLD VECTOR: DEALLOCATE FSA LOC
```

```
 365 001550  020003                                    CMP     R0,R3           ;WAS IT ELSE ONLY?
 366 001552  001404                                    BEQ     13$             ;YES
 367 001554  016465  000000G 177776                    MOV     N.FSA(R4),-2(R5)        ;REPOINT TO OLD FSA LOC
 368 001562  000431                                    BR      700$
 369 001564  005065  177776          13$:              CLR     -2(R5)          ;CREATE JUMP STATE TO FSA LOC
 370 001570  016425  000000G                           MOV     N.FSA(R4),(R5)+
 371 001574  012725  000000G                           MOV     #ST$JSQ,(R5)+
 372 001600                                            RESTOR  R1,R3
 373 001604  000207                                    RTS     PC
 374 001606  016764  005666' 000000G 16$:              MOV     CFSA,N.FSA(R4)  ;REUSE THIS FSA LOC
 375 001614  005367  000000G                           DEC     NXTADD          ;DEALLOCATE FSA LOC
 376 001620                                            RESTOR  R1,R3
 377 001624  000207                                    RTS     PC
 378 001626  016725  000000G         100$:             MOV     INTDEF,(R5)+    ;XFER INTERNAL DEFAULT
 379 001632  000405                                    BR      700$
 380 001634  020003                  14$:              CMP     R0,R3           ;ELSE ONLY?
 381 001636  001763                                    BEQ     16$             ;YES
 382 001640  004767  002344                            JSR     PC,FMTST1       ;DO NON-ELSE ENTRIES MATTER?
 383 001644  103760                                    BCS     16$             ;NO
 384 001646  016725  000000G         700$:             MOV     NXTADD,(R5)+    ;INDEX STATE FOR POSITION
 385 001652  012725  000000C                           MOV     #ST$CNG!IWBIT,(R5)+     ;CHANGE ELSE STATE
 386 001656  016705  000000G                           MOV     NXTADD,R5       ;R5=TDCT OFFSET
 387 001662  010567  005666'                           MOV     R5,CFSA         ;UPDATE CURRENT FSA LOC
 388 001666  070527  000006                            MUL     #6,R5
 389 001672  066705  000000G                           ADD     TDCADR,R5
 390 001676  005267  000000G                           INC     NXTADD          ;ALLOCATE FSA LOC
 391 001702  022766  000001  000002                    CMP     #1,2(SP)        ;SEQUENCIAL POSSIBLE?
 392 001710  001005                                    BNE     1000$           ;NO
 393 001712                                            RESTOR  R1,R3           ;YES
 394 001716  004767  000606                            JSR     PC,FMSEQ        ;COMPLETE VECTOR WITH SEQ STATES
 395 001722  000207                                    RTS     PC
 396 001724  022767  000000G 005672' 1000$:           CMP     #EMXMCF,FMT     ;MATCH STATE SCAN?
 397 001732  001005                                    BNE     1001$           ;NO
 398 001734                                            RESTOR  R1,R3
 399 001740  004767  001524                            JSR     PC,MATRP2       ;CREATE MATCH STATE(S)
 400 001744  000207                                    RTS     PC
 401 001746  011603                  1001$:            MOV     (SP),R3         ;R3=VECTOR ENTRIES ADR
 402 001750  016601  000002                            MOV     2(SP),R1        ;R1=VECTOR SIZE
 403 001754                                            SAVE    R5              ;TDCT ADR
 404 001756  005004                                    CLR     R4              ;INIT FOR SINGLE EXIT TEST
 405 001760  005005                                    CLR     R5
 406 001762  005000                                    CLR     R0
 407 001764  012302                  1002$:            MOV     (R3)+,R2        ;SUMMARIZE BITS IN VECTOR
 408 001766  100405                                    BMI     1003$           ;EXCEPT ELSE ENTRIES
 409 001770  056204  000002G                           BIS     EMA+2(R2),R4
 410 001774  056205  000004G                           BIS     EMA+4(R2),R5
 411 002000  005200                                    INC     R0              ;COUNT NON-ELSE ENTRIES
 412 002002  077110                  1003$:            SOB     R1,1002$
 413 002004  004767  002704                            JSR     PC,FMTST2       ;TEST FOR SINGLE EXIT CONDITION
 414 002010  103073                                    BCC     1100$           ;NOT SINGLE EXIT
 415 002012                                            RESTOR  R1,R3,R0
 416 002020  126727  000000G 000000G                   CMPB    PHI,#FMN2       ;NIB2?
 417 002026  001402                                    BEQ     1004$           ;YES
 418 002030  010420                                    MOV     R4,(R0)+        ;XFER NIBBLE1
 419 002032  000401                                    BR      1005$
 420 002034  010520                  1004$:            MOV     R5,(R0)+        ;XFER NIBBL2
 421 002036  016720  000000G         1005$:            MOV     NXTADD,(R0)+    ;COMPLETE SINGLE EXIT STATE
```

```
422 002042 016767 000000G 000000C          MOV     NXTADD,VVEC+N.FSA       ;& ENTER NODE FOR NEXT STATE
423 002050 012710 000000C                  MOV     #ST$INX!SSBIT,(R0)
424 002054 056720 005676'                  BIS     FMSC,(R0)+              ;SOURCE CODE
425 002060 116767 000000C 000000C          MOVB    VMASK+N.FMT,VVEC+N.FMT  ;FMT
426 002066 105267 000000C                  INCB    N.FMT+VVEC
427 002072 016767 000000C 000000C          MOV     VMASK+N.LGT,VVEC+N.LGT  ;VECTOR SIZE
428 002100 116767 000000G 000000C          MOVB    NPHI,VVEC+N.PHI         ;NODE TYPE
429 002106 012702 000000C                  MOV     #VMASK+N.FMHS,R2        ;XFER VECTOR
430 002112 012703 000000C                  MOV     #VVEC+N.FMHS,R3
431 002116 012223                 1007$:   MOV     (R2)+,(R3)+
432 002120 077102                          SOB     R1,1007$
433 002122 016705 000000G                  MOV     NXTADD,R5       ;R5=TDCT OFFSET
434 002126 070527 000006                   MUL     #6,R5
435 002132 066705 000000G                  ADD     TDCADR,R5
436 002136 005267 000000G                  INC     NXTADD
437 002142 012703 005654'                  MOV     #FMNI,R3        ;SCAN FOR VECTOR DUPLICATE
438 002146 012702 000000G                  MOV     #VVEC,R2
439 002152 016700 000000G                  MOV     NPHI,R0         ;NODE TYPE
440 002156 004767 001442                   JSR     PC,SCAN
441 002162 103005                          BCC     1006$           ;NEW VECTOR
442 002164 005025                          CLR     (R5)+           ;DUPLICATE, JUMP TO IT
443 002166 016425 000000G                  MOV     N.FSA(R4),(R5)+
444 002172 012725 000000G                  MOV     #ST$JSQ,(R5)+
445 002176 000207                 1006$:   RTS     PC
446 002200                        1100$:   RESTOR  R2              ;R2=TDCT ADR
447 002202 011603                          MOV     (SP),R3         ;R3=VECTOR ENTRY ADR
448 002204 016601 000002                   MOV     2(SP),R1        ;R1=VECTOR SIZE
449 002210 126727 000000G 000000G          CMPB    PHI,#FMN2       ;NIB2?
450 002216 001401                          BEQ     1101$           ;YES
451 002220 010405                          MOV     R4,R5           ;NIBBLE1, R5=NIBBLE VALUE
452 002222 010522                 1101$:   MOV     R5,(R2)+        ;XFER INDEX
453 002224 016722 000000G                  MOV     NXTADD,(R2)+    ;COMPLETE INDEX STATE
454 002230 012712 000000G                  MOV     #ST$INX,(R2)
455 002234 056722 005676'                  BIS     FMSC,(R2)+
456 002240 012704 000000C                  MOV     #VVEC+N.FMHS,R4 ;R4=VECTOR ENTRIES ADR
457 002244 116767 000000C 000000C          MOVB    VMASK+N.FMT,VVEC+N.FMT  ;XFER FMT
458 002252 105267 000000C                  INCB    VVEC+N.FMT              ;STEP FMT
459 002256 016767 000000G 000000C          MOV     NXTADD,VVEC+N.FSA       ;FSA LOC
460 002264 005367 000000C                  DEC     VVEC+N.FSA      ;INC'ED LATER
461 002270 116767 000000G 000000C          MOVB    NPHI,VVEC+N.PHI ;       ;POOL HEADER INDEX
462 002276 012700 000020                   MOV     #16.,R0         ;#BITS TO SCAN
463 002302 005002                          CLR     R2              ;R2=SIZE
464 002304 000261                          SEC
465 002306 006002                 1102$:   ROR     R2              ;MASK
466 002310 006305                          ASL     R5              ;VECTOR SUMMARY
467 002312 103404                          BCS     1110$           ;IF SET IN SUM, THEN CREATE NODE
468 002314 077004                 1103$:   SOB     R0,1102$
469 002316                                 RESTOR  R1,R3
470 002322 000207                          RTS     PC
471 002324                        1110$:   SAVE    R0,R2,R3,R4,R5
472 002336 005267 000000C                  INC     VVEC+N.FSA      ;NEXT FSA LOC
473 002342 005067 000000C                  CLR     VVEC+N.LGT      ;INIT SIZE
474 002346 016601 000014                   MOV     14(SP),R1       ;GET SIZE
475 002352 116767 000000G 000000C          MOVB    NPHI,VVEC+N.PHI ;SET POOL HEADER INDEX
476 002360 126727 000000G 000000G          CMPB    PHI,#FMN2       ;NIB2?
477 002366 001412                          BEQ     1121$           ;YES
478 002370 012305                 1111$:   MOV     (R3)+,R5        ;GET ENTRY
```

```
  479 002372· 100403                        BMI·    1112$           ;XFER·IF ELSE·OR·
  480 002374  030265  000000C·              BIT·    R2,EMA+EMXNB1(R5)       ;IF·MASK·MATCHES·
  481 002400                                BOFF·   1113$
  482·002402· 010524                1112$:  MOV·    R5,(R4)+
  483 002404  005267  000000C·              INC·    VVEC+N,LGT·
  484 002410  077111                1113$:  SOB·    R1,1111$
  485 002412· 000411                        BR      1130$
  486 002414  012305                1121$:  MOV·    (R3)+,R5        ;LIKE·LOOP·1111$ FOR·NIB2·
  487 002416  100403                        BMI·    1122$
  488 002420  030265  000000C·              BIT·    R2,EMA+EMXNB2(R5)
  489 002424                                BOFF·   1123$
  490 002426  010524                1122$:  MOV·    R5,(R4)+
  491 002430  005267  000000C·              INC·    VVEC+N,LGT·
  492·002434  077111                1123$:  SOB·    R1,1121$
  493 002436  016705  000000G·      1130$:  MOV·    NXTADD,R5       ;ALLOCATE·FSA·LOC·
  494 002442· 070527  000006                MUL·    #6,R5
  495 002446  066705  000000G·              ADD·    TDCADR,R5
  496 002452· 005267  000000G·              INC·    NXTADD·
  497 002456  012703  005654'               MOV·    #FMNI,R3        ;SEARCH·FOR·DUP·OR ENTER·VECTOR·
  498 002462· 012702· 000000G·              MOV·    #VVEC,R2·
  499 002466  016700  000000G·              MOV·    NPHI,R0         ;NODE·TYPE·
  500 002472· 004767  001126                JSR·    PC,SCAN·
  501 002476  103405                        BCS·    1131$           ;NEW·VECTOR·
  502·002500  005025                        CLR·    (R5)+           ;DUP, JUMP·TO·IT·
  503 002502· 016425  000000G·              MOV·    N,FSA(R4),(R5)+
  504 002506  012725  000000G·              MOV·    #ST$JSD,(R5)+
  505 002512·                       1131$:  RESTOR· R0,R2,R3,R4,R5
  506 002524  000241                        CLC·                    ;DO·ADD·BIT·TO·MASK·
  507 002526  000672·                       BR      1103$
```

```
                                                ;
                                                ;   FMSEQ IS A SUBROUTINE TO CREATE SEQUENCIAL STATES FOR SINGLE
                                                ;   ENTRY VECTORS.
                                                ;
                                                ;   R3 IS USED FOR INPUT AS A POINTER TO THE SINGLE ENTRY IN THE
                                                ;   VECTOR. NO REGISTERS ARE USED FOR OUTPUT.
                                                ;
002530  016701  005666'          FMSEQ:  MOV     CFSA,R1         ;CONVERT FSA LOC TO TDCT ADR
002534  070127  000006                  MUL     #6,R1
002540  066701  000000G                 ADD     TDCADR,R1
002544  011303                          MOV     (R3),R3         ;GET ENTRY
002546  100001                          BPL     1$
002550  005403                          NEG     R3              ;IF NEGATIVE, MAKE POSITIVE
002552  016305  000000G         1$:     MOV     EMA(R3),R5      ;GET EMA ENTRY
002556  032705  000000G                 BIT     #EMXMCF,R5      ;IS IT A MATCH CODE?
002562                                  BOFF    6$              ;NO
002564  000167  000526                  JMP     700$            ;YES
002570  032705  000000G         6$:     BIT     #EMXDTF,R5      ;R2=SOURCE CODE FOR FMT
002574                                  BON     2$
002576  032705  000000G                 BIT     #EMXZNF,R5
002602                                  BON     3$
002604  012702  003000                  MOV     #BIT10!BIT9,R2
002610  000405                          BR      4$
002612  012702  002000          2$:     MOV     #BIT10,R2
002616  000402                          BR      4$
002620  012702  001000          3$:     MOV     #BIT9,R2
002624  032705  000000G         4$:     BIT     #EMXMTV,R5      ;MULTIPLE VALUES FOR ENTRY?
002630                                  BOFF    7$              ;NO
002632  000167  000552                  JMP     100$            ;YES
002636                          7$:     SAVE    R1              ;IS STATE BEING APENDED TO TDCT?
002640  166701  000000G                 SUB     TDCADR,R1
002644  005000                          CLR     R0
002646  071027  000006                  DIV     #6,R0
002652  005200                          INC     R0
002654                                  RESTOR  R1
002656  020067  000000G                 CMP     R0,NXTADD
002662  103412                          BLO     5$              ;NO
002664  001464                          BEQ     20$             ;CLOSE ENOUGH
002666  016701  000000G                 MOV     NXTADD,R1       ;ALLOCATE NEXT FSA LOC
002672  070127  000006                  MUL     #6,R1
002676  066701  000000G                 ADD     TDCADR,R1
002702  005267  000000G                 INC     NXTADD
002706  000453                          BR      20$
002710  110521                  5$:     MOVB    R5,(R1)+        ;CREATE JUMP/SEQ STATE
002712  105021                          CLRB    (R1)+
002714  016721  000000G                 MOV     NXTADD,(R1)+
002720  012711  000000C                 MOV     #ST$JSQ!JMPBIT,(R1)
002724  050211                          BIS     R2,(R1)
002726  062703  000006          10$:    ADD     #6,R3           ;STEP TO NEXT EMA ENTRY
002732  016305  000000G                 MOV     EMA(R3),R5      ;GET ENTRY
002736  016701  000000G                 MOV     NXTADD,R1       ;ALLOCATE NEXT FSA LOC
002742  070127  000006                  MUL     #6,R1
002746  066701  000000G                 ADD     TDCADR,R1
002752  005267  000000G                 INC     NXTADD
002756  032705  000000G                 BIT     #EMXMCF,R5      ;IS ENTRY A MATCH CODE?
002762                                  BOFF    17$             ;NO
002764  000167  000326                  JMP     700$            ;YES
```

```
 566 002770  032705  000000G·          17$:    BIT·    #EMXDTF,R5        ;SET·UP·SOURCE·CODE·
 567 002774                                    BON·    11$
 568 002776· 032705  000000G·                  BIT·    #EMXZNF,R5
 569 003002·                                   BON·    12$
 570 003004  012702· 003000                    MOV·    #BIT10!BIT9,R2
 571 003010  000405                            BR      13$
 572·003012· 012702· 002000            11$:    MOV·    #BIT10,R2·
 573 003016  000402·                           BR      13$
 574 003020  012702· 001000            12$:    MOV·    #BIT9,R2·
 575 003024  032705  000000G·          13$:    BIT·    #EMXMTV,R5        ;MULTIPLE·VALUES?·
 576 003030                                    BOFF·   20$               ;NO·
 577 003032· 000167  000352                    JMP·    100$              ;YES·
 578 003036  012761  000000C·000004   20$:    MOV·    #ST$SEQ!S3BIT,4(R1)       ;CREATE·SEQ·STATE·
 579 003044  110511                            MOVB·   R5,(R1)           ;XFER·1ST·BYTE·OF·SEQ·
 580 003046  006202·                           ASR·    R2·
 581 003050  050261  000002                    BIS·    R2,2(R1)          ;ENTER·SOURCE·CODE
 582·003054  062703  000006                    ADD·    #6,R3             ;STEP·TO·NEXT·EMA·ENTRY·
 583 003060  016305  000000G·                  MOV·    EMA(R3),R5        ;GET·NEXT·ENTRY·
 584 003064  032705  000000C·                  BIT·    #EMXMCF!EMXMTV,R5         ;SEQ·OVER?
 585 003070                                    BOFF·   21$                       ;NO·
 586 003072· 052761  000000G·000004            BIS·    #S1BIT,4(R1)              ;YES:  END·ON·1ST·BYTE·
 587 003100  032705  000000G·                  BIT·    #EMXMCF,R5        ;MATCH·CODE?·
 588 003104                                    BON·    702$              ;YES·
 589 003106  032705  000000G·          21$:    BIT·    #EMXDTF,R5        ;SET·UP·SOURCE·CODE·
 590 003112·                                   BON·    22$
 591 003114  032705  000000G·                  BIT·    #EMXZNF,R5
 592·003120                                    BON·    23$
 593 003122· 012702· 003000                    MOV·    #BIT10!BIT9,R2
 594 003126  000405                            BR      24$
 595 003130  012702· 002000            22$:    MOV·    #BIT10,R2·
 596 003134  000402·                           BR      24$
 597 003136  012702· 001000            23$:    MOV·    #BIT9,R2·
 598 003142· 032705  000000G·          24$:    BIT·    #EMXMTV,R5        ;MULTIPLE·VALUES?·
 599 003146                                    BON·    102$              ;YES·
 600 003150  110561  000001                    MOVB·   R5,1(R1)          ;XFER·2ND·BYTE·OF·SEQ·
 601 003154  006302·                           ASL·    R2·
 602·003156  050261  000002                    BIS·    R2,2(R1)          ;ENTER·SOURCE·CODE
 603 003162· 062703  000006                    ADD·    #6,R3             ;SET·TO·NEXT·EMA·ENTRY·
 604 003166  016305  000000G·                  MOV·    EMA(R3),R5        ;GET·ENTRY·
 605 003172· 032705  000000C·                  BIT·    #EMXMCF!EMXMTV,R5         ;SEQ·OVER?
 606 003176                                    BOFF·   25$                       ;NO·
 607 003200  052761  000000G·000004            BIS·    #S2BIT,4(R1)              ;YES:  END·ON·2ND·BYTE·
 608 003206  032705  000000G·                  BIT·    #EMXMCF,R5        ;MATCH·  CODE?·
 609 003212·                                   BON·    702$              ;YES·
 610 003214  032705  000000G·          25$:    BIT·    #EMXDTF,R5        ;SET·UP·SOURCE·CODE·
 611 003220                                    BON·    26$
 612·003222· 032705  000000G·                  BIT·    #EMXZNF,R5
 613 003226                                    BON·    27$
 614 003230  012702· 003000                    MOV·    #BIT10!BIT9,R2
 615 003234  000405                            BR      30$
 616 003236  012702· 002000            26$:    MOV·    #BIT10,R2·
 617 003242· 000402·                           BR      30$
 618 003244  012702· 001000            27$:    MOV·    #BIT9,R2·
 619 003250  032705  000000G·          30$:    BIT·    #EMXMTV,R5        ;MULTIPLE·VALUES?·
 620 003254                                    BON·    102$              ;YES·
 621 003256  110561  000002                    MOVB·   R5,2(R1)          ;XFER·3RD·BYTE·OF·SEQ·
 622·003262· 072227  000003                    ASH·    #3,R2·
```

```
623 003266  050261  000002                      BIS·    R2,2(R1)          ;ENTER· SOURCE·CODE·
624 003272· 062703  000006                      ADD·    #6,R3             ;STEP·TO MATCH·CODE·EMA·ENTRY·
625 003276  016701  000000G·          702$:     MOV·    NXTADD,R1         ;ALLOCATE·FSA·LOC·
626 003302· 070127  000006                      MUL·    #6,R1
627 003306  066701  000000G·                    ADD·    TDCADR,R1
628·003312· 005267  000000G·                    INC·    NXTADD·
629 003316  016311  000000C·          700$:     MOV·    EMA+EMXMCD(R3),(R1)       ;CREATE·MATCH·STATE·
630 003322· 042721  140000                      BIC·    #BIT15!BIT14,(R1)+
631 003326  016721  000000G·                    MOV·    INTDEF,(R1)+
632·003332· 012711  000000G·                    MOV·    #ST$MAT,(R1)
633 003336  032763  100000  000000C·           BIT·    #BIT15,EMA+EMXMCD(R3)     ;SET·TERM·TYPE·
634 003344                                      BOFF·   701$
635 003346  052711  000000G·                    BIS·    #CBIT,(R1)
636 003352· 032763  040000  000000C·701$:     BIT·    #BIT14,EMA+EMXMCD(R3)
637 003360                                      BOFF·   703$
638 003362· 052711  000000G·                    BIS·    #TBIT,(R1)
639 003366  000207                  703$:     RTS·    PC·
640 003370  016701  000000G·          102$:     MOV·    NXTADD,R1         ;ALLOCATE·FSA·LOC·
641 003374  070127  000006                      MUL·    #6,R1
642·003400  066701  000000G·                    ADD·    TDCADR,R1
643 003404  005267  000000G·                    INC·    NXTADD·
644 003410  016321  000000C·          100$:     MOV·    EMA+EMXNB1(R3),(R1)+      ;CREATE·SINGLE·EXIT·INDEX·
645 003414  016721  000000G·                    MOV·    NXTADD,(R1)+              ;FOR·NIBBLE1 AND·
646 003420  012711  000000C·                    MOV·    #ST$INX!SSBIT,(R1)
647 003424  050211                              BIS·    R2,(R1)
648 003426  016701  000000G·                    MOV·    NXTADD,R1         ;ALLOCATE·NEXT·FSA LOC·
649 003432· 070127  000006                      MUL·    #6,R1
650 003436  066701  000000G·                    ADD·    TDCADR,R1
651 003442· 005267  000000G·                    INC·    NXTADD·
652·003446  016321  000000C·                    MOV·    EMA+EMXNB2(R3),(R1)+      ;  FOR·NIBBLE2·
653 003452· 016721  000000G·                    MOV·    NXTADD,(R1)+
654 003456  012711  000000C·                    MOV·    #ST$INX!SSBIT,(R1)
655 003462· 050211                              BIS·    R2,(R1)
656 003464  000167  177236                      JMP·    10$
```

```
                                        ;
                                        ;   GENERATE·MATCH·REPORT(S) !!!!!
                                        ;   CREATE A·SEPARATE·MATCH·REPORT·FOR·EACH·ENTRY·IN·VVEC.  INSERT·IN·
                                        ;   SEG·1 OF·MATCH·REPORT·STATE·THE·FLU/TERM·ID·FOUND·IN·NB1 OF·EMA.
                                        ;
                                        ;       R5 =·NEXT·AVAILABLE·TDCT·STATE·
                                        ;
003470  000410                  MATRP2::BR      1$
003472· 016705  000000G·         4$:    MOV·    NXTADD,R5       ;ALLOCATE·NEXT·FSA LOC·
003476  070527  000006                   MUL·    #6,R5
003502· 066705  000000G·                 ADD·    TDCADR,R5
003506  005267  000000G·                 INC·    NXTADD·
003512· 012302·                  1$:    MOV·    (R3)+,R2·       ;GET·EMA OFFSET·
003514  016202· 000000C·                 MOV·    EMA+EMXMCD(R2),R2·      ;R2=EMA·ADR·OF·MATCH CODE·
003520  020267  005704'                   CMP·    R2,OLDMC·       ;SAME·CODE·AS·BEFORE?·
003524  001433                           BEQ·    5$              ;YES·
003526  010267  005704'                   MOV·    R2,OLDMC·       ;NO: UPDATE·OLD·
003532· 010215                           MOV·    R2,(R5)         ;XFER·MATCH·CODE·
003534  042725  140000                   BIC·    #BIT15!BIT14,(R5)+      ;REMOVE·FLAGS·
003540  016725  000000G·                 MOV·    NXTADD,(R5)+    ;ASSUME·ANOTHER·MATCH·CODE·
003544  012715  000000G·                 MOV·    #ST$MAT,(R5)    ;DECLAR·AS·MATCH·STATE·
003550  032702· 100000                   BIT·    #BIT15,R2·      ;CWP?·
003554                                   BOFF·   2$              ;NO·
003556  052715  000000G·                 BIS·    #CBIT,(R5)      ;YES·
003562· 032702· 040000           2$:    BIT·    #BIT14,R2·      ;TERM?·
003566                                   BOFF·   3$              ;NO·
003570  052715  000000G·                 BIS·    #TBIT,(R5)      ;YES·
003574  010500                   3$:    MOV·    R5,R0           ;SAVE·CURRENT·TDCT STATE·ADR
003576  077143                           SOB·    R1,4$           ;DO·FOR·ALL·MATCH·REPORTS·
003600  016765  000000G· 177776         MOV·    INTDEF,-2(R5)   ;RESET·LAST·STATES XFER·ADR·
003606  005067  005704'                   CLR·    OLDMC·          ;PREVENT MATCH·WITH·NEXT·CODE·
003612· 000207                           RTS·    PC·
003614  005367  000000G·         5$:    DEC·    NXTADD·         ;DEALLOCATE·FSA·LOC·
003620  010005                           MOV·    R0,R5           ;RESTORE LAST·TDCT ADR·
003622· 000764                           BR      3$
```

```
                                        ;
                                        ; SUBROUTINE·TO·SCAN NODE·POOL FOR·NODE·AND·MERGE·INTO·POOL·IF·NOT·FOUND·
                                        ;
                                        ;                INPUT·                              OUTPUT·
                                        ;       R0=·     NODE TYPE·                          (SAME)
                                        ;       R1=·     (SAVED)                             (RESTORED)
                                        ;       R2=·     NODE TO·SCAN·FOR·                   (SAME)
                                        ;       R3=·     POOL HEADER·                        SCRATCH·
                                        ;       R4=·     SCRATCH·                            ADR·NODE·IN·POOL(OLD·OR·NEW)
                                        ;       R5=·     (SAVED)                             (RESTORED)
                                        ;
003624                                  SCAN::  SAVE·    R1
003626  016001  004144'                         MOV·     NDSIZ(R0),R1      ;R1=NODE SIZE·
003632· 011304                                  MOV·     (R3),R4           ;GET·1ST NODE·
003634  001463                                  BEQ·     MERGE·            ;IF·NONE, MERGE·
003636                                          SAVE·    R3,R5
003642· 016203  000000G·                        MOV·     N.LGT(R2),R3      ;R3=NODE SIZE·
003646  020027  000000G·                1$:     CMP·     R0,#N2P·          ;ELSE·CHECK·NECCESSARY(NIB1 OR·NIB2)
003652· 101011                                  BHI·     10$               ;NO·
003654  026462· 000000G·000000G·                CMP·     N.ELSE(R4),N.ELSE(R2)
003662· 103444                                  BLO·     410$              ;ELSE·IN DEC·ORDER, IF·LESS·FINISHED·
003664  001404                                  BEQ·     10$
003666  011405                          2$:     MOV·     (R4),R5           ;STEP·TO NEXT·NODE
003670  001443                                  BEQ·     401$              ;NO·MORE
003672· 010504                                  MOV·     R5,R4
003674  000764                                  BR       1$
003676  026403  000000G·                10$:    CMP·     N.LGT(R4),R3      ;ARE·SIZES·SAME?·
003702· 101034                                  BHI·     410$              ;NO: INC ORDER, FINISHED·
003704  001401                                  BEQ·     11$
003706  103767                                  BLO·     2$
003710  005703                          11$:    TST·     R3                ;ZERO·LENGTH·VECTORS·ALWAYS·MATCH·
003712· 001414                                  BEQ·     1100$
003714                                          SAVE·    R2,R3,R4
003722· 060102·                                 ADD·     R1,R2·            ;STEP·TO VECTOR·PORTION·OF·NODE·
003724  060104                                  ADD·     R1,R4
003726  022224                          110$:   CMP·     (R2)+,(R4)+       ;VECTORS ARE·IN·DEC·ORDER·
003730  101016                                  BHI·     112$              ;HIGH: GET·NEXT·NODE·
003732· 103411                                  BLO·     111$              ;LOW: FINISHED·
003734  077304                                  SOB·     R3,110$           ;CMP·ALL ENTRIES·
003736                                          RESTOR·  R2,R3,R4
003744                                  1100$:  RESTOR·  R1,R3,R5
003752· 000241                                  CLC·                       ;MATCH!!!!!!!!!!!!!!
003754  000207                                  RTS·     PC·
003756                                  111$:   RESTOR·  R2,R3,R4          ;STEP·TO NEXT·VECTOR·
003764  000740                                  BR       2$
003766                                  112$:   RESTOR·  R2,R3,R4
003774  016404  000000G·                410$:   MOV·     N.BACK(R4),R4     ;NO·MATCH, BACK·UP TO·PRIOR·NODE·
004000                                  401$:   RESTOR·  R3,R5
004004                                  MERGE:  SAVE·    R4,R5,R2·
004012· 016205  000000G·                        MOV·     N.LGT(R2),R5      ;R5=NODE SIZE·
004016  006305                                  ASL·     R5
004020  060105                                  ADD·     R1,R5
004022· 016001  004160'                         MOV·     NPLOC(R0),R1      ;R1=NEXT POOL·ENTRY·ADR·
004026  011104                                  MOV·     (R1),R4
004030  060511                                  ADD·     R5,(R1)           ;GET·ADR AND·ALLOCATE·THIS·NODE·
004032· 021160  004174'                         CMP·     (R1),NPOVF(R0)    ;NODE·POOL·OVERFLOW?·
004036  101040                                  BHI·     12$               ;YES·
```

```
 751 004040  020402·                        CMP·    R4,R2·          ;IS·NODE AT·THIS·LOC?·
 752·004042· 001404                         BEQ·    31$             ;YES: THEN·DON'T·COPY·
 753 004044  010416                         MOV·    R4,(SP)         ;REPLACE NODE·LOC·
 754 004046  006205                         ASR·    R5              ;R5=·NODE·SIZE(WORDS)
 755 004050  012224                  3$:    MOV·    (R2)+,(R4)+     ;XFER·VECTOR·TO·NEW·NODE·
 756 004052· 077502·                        SOB·    R5,3$
 757 004054                          31$:   RESTOR· R1,R4,R5,R2·
 758 004064  005704                         TST·    R4              ;R4=POOL HEADER?·
 759 004066  001410                         BEQ·    10$             ;YES·
 760 004070  011403                         MOV·    (R4),R3         ;NO:ADJUST·LINKAGES·
 761 004072· 010312·                        MOV·    R3,(R2)
 762·004074  010462· 000000G·               MOV·    R4,N.BACK(R2)
 763 004100  010214                         MOV·    R2,(R4)
 764 004102· 010263  000000G·               MOV·    R2,N.BACK(R3)
 765 004106  000411                         BR      11$
 766 004110  010462· 000000G·          10$: MOV·    R4,N.BACK(R2)   ;ADJUST·LINKAGES·
 767 004114  011304                         MOV·    (R3),R4
 768 004116  010412·                        MOV·    R4,(R2)
 769 004120  010213                         MOV·    R2,(R3)
 770 004122· 005704                         TST·    R4              ;AT·HEADER?·
 771 004124  001402·                        BEQ·    11$             ;YES: NO BACK·LINK
 772·004126  010264  000000G·               MOV·    R2,N.BACK(R4)   ;NO: SET BACK·LINK
 773 004132· 010204                  11$:   MOV·    R2,R4           ;R4=MERGED·NODE·
 774 004134  000261                         SEC·                    ;SET·MERGE·FLAG·
 775 004136  000207                         RTS·    PC·
 776 004140  004767  000000G·         12$:  JSR·    PC,NODERR·      ;REPORT·ERROR·
 777                                 ;      EXIT·
 778 004144  000000G·                NDSIZ:: N.NHS·  ;N1P
 779 004146  000000G·                        N.NHS·  ;N2P
 780 004150  000000G·                        N.NHS·  ;ENP
 781 004152· 000000G·                        N.FMHS· ;FME
 782·004154  000000G·                        N.FMHS· ;FMN1
 783 004156  000000G·                        N.FMHS· ;FMN2·
 784 004160  000000G·                NPLOC:  NNPE·   ;N1P
 785 004162· 000000G·                        NNPE·   ;N2P
 786 004164  000000G·                        NNPE·   ;ENP
 787 004166  005660'                         NFME·   ;FME
 788 004170  005652'                         NFMN·   ;FMN1
 789 004172· 005652'                         NFMN·   ;FMN2·
 790 004174  000000G·                NPOVF:  NPE·    ;N1P
 791 004176  000000G·                        NPE·    ;N2P
 792·004200  000000G·                        NPE·    ;ENP
 793 004202· 005540'                         FMEPE·  ;FME
 794 004204  004540'                         FMNPE·  ;FMN1
 795 004206  004540'                         FMNPE·  ;FMN2·
```

```
                                        ;
                                        ;   SUBROUTINE FMTST1'S ONLY FUNCTION IS TO MAKE A COMPLEX DECISION.
                                        ;   IT MUST DECIDE WHEITHER THE REMAINING NON-ELSE VECTORS SHOULD BE
                                        ;   IGNORED OR CONSIDERED. COMPLEX FLU-MOD'S MAY BE CONSIDERABLE.
                                        ;   SIMPLIFIED BY IGNORING SUBSEQUENT MATCH CRITERIA ON A PATH ONCE
                                        ;   IT IS KNOWN THAT THAT PATH MUST MATCH. HOWEVER, SINCE THE VECTOR
                                        ;   MAY CONSIST OF MULTIPLE COMPLEX FLU-MOD'S, THIS DECISION CAN BE
                                        ;   QUITE INVOLVED.
                                        ;
004210  032767  000000G 005674' FMTST1: BIT     #EMXMCF,FMTSUM  ;HAVE ANY CRITERIA BEEN STATISFIED?
004216                                  BOFF    21$             ;NO: MUST CONSIDER OTHER CRITERIA.
004220  022767  177777  005710'       CMP     #-1,FMEXF       ;YES: WHAT DOES VECTOR REPESENT?
004226  001570                        BEQ     21$             ;MULTIPLE SIMPLE FLU MOD'S, CONSIDER REST
004230  002434                        BLT     310$            ;SINGLE COMPLEX FLU-MOD, IGNORE REST.
004232                                SAVE    R4,R5           ;MULTIPLE COMPLEX FLU-MOD'S,CHECK FURTHER
004236  016601  000010                MOV     10(SP),R1       ;R1=VECTOR SIZE.
004242  016603  000006                MOV     6(SP),R3        ;R3=START OF VECTORS.
004246  016704  005700'               MOV     FMEXL,R4        ;R4=POINTER TO FLU-MOD VECTOR STARTS.
004252  012405                        MOV     (R4)+,R5        ;R5=# FLU-MOD'S.
004254  006205                        ASR     R5
004256  042705  160000                BIC     #160000,R5
004262  005067  005720'               CLR     FMAT            ;CLEAR INDICATORS
004266  005067  005714'               CLR     FMIN
004272  005067  005716'               CLR     FVAL
004276  012767  005724' 005722'       MOV     #VALVEC,VALVPT  ;RESET VALUE VECTOR POINTER
004304  005067  006000'               CLR     LEVEL
004310  012302                1$:     MOV     (R3)+,R2        ;GET VECTOR ENTRY
004312  100405                        BMI     2$              ;IF ELSE
004314  005067  005714'               CLR     FMIN            ;INDICATE POSITIVE
004320  000405                        BR      3$
004322  000167  000312        310$:   JMP     31$
004326  005402                2$:     NEG     R2              ;MAKE POSITIVE
004330  005267  005714'               INC     FMIN            ;INDICATE NEGATIVE
004334  012400                3$:     MOV     (R4)+,R0        ;FIND FLU-MOD FOR VECTOR ENTRY
004336  100013                        BPL     40$
004340                                SAVE    R4,R5           ;IF NEG, ENTRY IS NESTED
004344  005267  006000'               INC     LEVEL           ;INDICATED LEVEL NESTING
004350  005400                        NEG     R0              ;CONVERT TO POINTER
004352  010004                        MOV     R0,R4
004354  012405                        MOV     (R4)+,R5        ;GET COUNT
004356  006205                        ASR     R5
004360  042705  160000                BIC     #160000,R5
004364  012400                        MOV     (R4)+,R0        ;GET ENTRY
004366  020200                40$:    CMP     R2,R0
004370  103426                        BLO     10$             ;FOUND
004372  077520                4$:     SOB     R5,3$
004374  005767  006000'               TST     LEVEL
004400  001405                        BEQ     41$             ;IF NOT NESTED
004402  005367  006000'               DEC     LEVEL           ;IF NESTED, RETURN TO PRIOR LEVEL
004406                                RESTOR  R4,R5
004412  000767                        BR      4$
004414  012700  177777        41$:    MOV     #-1,R0          ;LAST ENTRY
004420  000412                        BR      10$
004422  012302                5$:     MOV     (R3)+,R2        ;GET NEXT VECTOR ENTRY
004424  100403                        BMI     6$              ;ELSE
004426  005067  005714'               CLR     FMIN            ;INDICATE POSITIVE
004432  000403                        BR      7$
```

```
854 004434  005402·                        6$:     NEG·    R2·             ;MAKE·POSITIVE·
855 004436  005267  005714'                 7$:     INC·    FMIN            ;INDICATE·NEGATIVE
856 004442· 020200                          10$:    CMP·    R2,R0           ;DOES·THIS·VECTOR·ENTRY·USE·SAME·FLU-MOD·
857 004444  103035                          11$:    BHIS·   13$             ;NO·
858 004446  105767  005714'                 12$:    TSTB·   FMIN            ;YES: ELSE·ENTRY?·
859 004452· 001410                          13$:    BEQ·    11$             ;NO·
860 004454  032762· 000000G·000000G·        14$:    BIT·    #EMXMCF,EMA(R2) ;YES: MATCH·CODE·ENTRY?·
861 004462·                                 20$:    BOFF·   12$             ;NO: SKIP·
862·004464  005402·                         21$:    NEG·    R2·             ;YES: STORE·MATCH·OFFSET·
863 004466  010267  005720'                 300$:   MOV·    R2,FMAT·
864 004472· 000412·                         30$:    BR      12$
865 004474  010302·                         31$:    MOV·    R3,R2·          ;SAVE·VECTOR·LOCATION·
866 004476  162702· 000002                  400$:   SUB·    #2,R2·
867 004502· 010277  005722'                 401$:   MOV·    R2,@VALVPT·
868 004506  062767  000002  005722'        ;       ADD·    #2,VALVPT·
869 004514  005267  005716'                 ;       INC·    FVAL
870 004520  077140                         ;       SOB·    R1,5$           ;CHECK·ALL·VECTOR·ENTRIES·
871 004522· 005767  005716'                 ;       TST·    FVAL            ;WAS·NON-ELSE·FOUND?·
872·004526  001434                         ;       BEQ·    30$             ;NO: NON-ELSE·CAN·BE·IGNORED!!!!!!
873 004530  005767  005720'                 ;       TST·    FMAT            ;YES: WAS·MATCH·FOUND?·
874 004534  001027                         ;       BNE·    300$            ;YES: NON-ELSE·CAN·BE·IGNORED!!!!!!!!
875 004536  000414                         ;       BR      20$             ;NO: OTHER·CRITERIA·MUST·BE·CONSIDERED!!
876 004540  005767  005716'                ;       TST·    FVAL            ;WAS·NON-ELSE·FOUND?·
877 004544  001406                         ;       BEQ·    14$             ;NO: SKIP·
878 004546  005767  005720'                ;       TST·    FMAT            ;YES: WAS·MATCH·FOUND?·
879 004552· 001406                         ;       BEQ·    20$             ;NO: OTHER·CRITERIA·MUST·BE·CONSIDERED!!
880 004554  004767  000064                 ;       JSR·    PC,400$         ;REMOVE·DUPLICATE·CRITERIA·
881 004560  000704                         ;       BR      4$
882·004562· 005067  005720'                ;14$:   CLR·    FMAT            ;CLEAR·INDICATERS·FOR·NEXT·FLU-MOD·
883 004566  000701                         ;       BR      4$              ;TEST·NEXT·FLU-MOD
884 004570                                  ;20$:   RESTOR· R4,R5
885 004574  005767  006000'                ;       TST·    LEVEL·
886 004600  001403                         ;       BEQ·    21$
887 004602· 005367  006000'                ;       DEC·    LEVEL·          ;IF·NESTED, RETURN TO·PRIOR·LEVEL·
888 004606  000770                         ;       BR      20$
889 004610  000241                         ;21$:   CLC·                    ;INDICATE·THAT·OTHER·CRITERIA·MUST·BE·
890 004612· 000207                         ;       RTS·    PC·             ; CONSIDERED·
891 004614  004767  000024                 ;300$:  JSR·    PC,400$         ;REMOVE·DUP·CRITERIA·
892·004620                                  ;30$:   RESTOR· R4,R5
893 004624  005767  006000'                ;       TST·    LEVEL·
894 004630  001403                         ;       BEQ·    31$
895 004632· 005367  006000'                ;       DEC·    LEVEL·
896 004636  000770                         ;       BR      30$
897 004640  000261                         ;31$:   SEC·                    ;INDICATE·THAT·NON-ELSE·CAN·BE· IGNORED
898 004642· 000207                         ;       RTS·    PC·
899                                         ;
900 004644                                  ;400$:  SAVE·   R2,R1
901 004650  016700  005720'                ;       MOV·    FMAT,R0         ;REPLACE ALL·STORED·VECTOR·LOCATIONS·
902·004654  016701  005716'                ;       MOV·    FVAL,R1         ;WITH·MATCH·CODE·ENTRY·
903 004660  012702· 005724'                ;       MOV·    #VALVEC,R2·
904 004664  010032·                        ;401$:  MOV·    R0,@(R2)+
905 004666  077102·                        ;       SOB·    R1,401$
906 004670  005067  005716'                ;       CLR·    FVAL
907 004674  005067  005720'                ;       CLR·    FMAT
908 004700  012767  005724'  005722'       ;       MOV·    #VALVEC,VALVPT
909 004706                                 ;       RESTOR· R2,R1
910 004712· 000207                         ;       RTS·    PC·
```

```
                                        ;
                                        ;   SUBROUTINE FMTST2'S ONLY FUNCTION IS TO MAKE A COMPLEX DECISION.
                                        ;   IT MUST DECIDE WEITHER A SINGE EXIT STATE IS POSSIBLE FOR THIS STATE.
                                        ;
004714  020027  000001        FMTST2: CMP     R0,#1           ;SINGLE ENTRY VECTOR?
004720  003535                        BLE     30$             ;YES: CAN ALWAYS BE SINGLE EXIT
004722  032767  000000G 005672'       BIT     #EMXSZF,FMT     ;NO: PROCESSING SUB-ZONES?
004730                                BOFF    20$             ;NO: PROBABLY CAN'T BE SINGLE EXIT
004732  122767  000000G 000000G       CMPB    #FMN2,PHI       ;YES: NIBBLE2?
004740  001414                        BEQ     1$              ;YES: CHECK IF SINGLE FLU-MOD
004742  022705  000001                CMP     #1,R5           ;NO: CHECK IF NIBBLE2 HAS SINGLE VALUE
004746  001411                        BEQ     1$              ;SINGLE VALUE
004750  022705  000002                CMP     #2,R5
004754  001406                        BEQ     1$
004756  022705  000004                CMP     #4,R5
004762  001403                        BEQ     1$
004764  022705  000010                CMP     #10,R5
004770  001077                        BNE     20$             ;MULTI-VALUE, CAN'T BE SINGLE EXIT
004772  022767  177777  005710' 1$:   CMP     #-1,FMEXF       ;WHAT TYPE OF FLU-MOD?
005000  002505                        BLT     30$             ;SINGLE COMPLEX FLU-MOD: SINGLE EXIT
005002  001472                        BEQ     20$             ;MULTIPLE SIMPLE FLU-MOD'S: CAN'T USE
005004                                SAVE    R4,R5           ;MULTIPLE COPLEX FLU-MOD'S
005010  016601  000012                MOV     12(SP),R1       ;R1=VECTOR SIZE
005014  016603  000010                MOV     10(SP),R3       ;R3=START OF VECTORS
005020  016704  005700'               MOV     FMEXL,R4        ;R4=POINTER TO FLU-MOD VECTOR STARTS
005024  012405                        MOV     (R4)+,R5        ;R5=# FLU-MOD'S
005026  006205                        ASR     R5
005030  042705  160000                BIC     #160000,R5
005034  005205                        INC     R5
005036  005067  006000'               CLR     LEVEL
005042  005301                10$:    DEC     R1              ;GET 1ST NON-ELSE VECTOR ENTRY
005044  012300                        MOV     (R3)+,R0
005046  100775                        BMI     10$
005050  005305                11$:    DEC     R5              ;FLU-MOD COUNT
005052  001426                        BEQ     32$             ;LAST FLU-MOD,ALL SAME,SINGLE EXIT
005054  012402                        MOV     (R4)+,R2        ;FIND FLU-MOD NOD FOR THIS ENTRY
005056  100013                        BPL     40$
005060                                SAVE    R4,R5           ;IF NEG, ENTRY IS POINTER
005064  005267  006000'               INC     LEVEL           ;NEST A LEVEL
005070  005402                        NEG     R2
005072  010204                        MOV     R2,R4           ;GET LOCATION
005074  012405                        MOV     (R4)+,R5        ;GET COUNT
005076  006205                        ASR     R5
005100  042705  160000                BIC     #160000,R5
005104  012402                        MOV     (R4)+,R2        ;GET ENTRY
005106  020002                40$:    CMP     R0,R2
005110  103357                        BHIS    11$             ;FOUND
005112  005301                12$:    DEC     R1              ;VECTOR COUNT
005114  100427                        BMI     31$             ;NO MORE ENTRIES,ALL SAME,SINGLE EXIT
005116  012300                        MOV     (R3)+,R0        ;GET NEXT NON-ELSE VECTOR
005120  100774                        BMI     12$             ;ELSE
005122  020002                        CMP     R0,R2           ;SAME FLU-MOD?
005124  103772                        BLO     12$             ;YES
005126  000410                        BR      21$             ;NO
005130  005767  006000'        32$:    TST     LEVEL
005134  001417                        BEQ     31$
005136  005367  006000'               DEC     LEVEL           ;IF NESTED, RETURN TO PRIOR LEVEL
```

```
    969 005142·                                 RESTOR· R4,R5
    970 005146  000740                          BR      11$
    971 005150                          21$:    RESTOR· R4,R5           ;NO: CAN'T·USE·SINGLE·EXIT·
    972·005154  005767  006000'                 TST·    LEVEL·
    973 005160  001403                          BEQ·    20$
    974 005162· 005367  006000'                 DEC·    LEVEL·          ;IF·NESTED, RETURN TO·PRIOR·LEVEL·
    975 005166  000770                          BR      21$
    976 005170  000241                  20$:    CLC·
    977 005172· 000207                          RTS·    PC·
    978 005174                          31$:    RESTOR· R4,R5           ;USE·SINGLE·EXIT·
    979 005200  005767  006000'                 TST·    LEVEL·
    980 005204  001403                          BEQ·    30$
    981 005206  005367  006000'                 DEC·    LEVEL·          ;IF·NESTED, RETURN TO·PRIOR·LEVEL·
    982·005212· 000770                          BR      31$
    983 005214  000261                  30$:    SEC·
    984 005216  000207                          RTS·    PC·
```

```
                                        ;
                                        ;   SUBROUTINE·GETNDE·GETS·THE NEXT·FLU-MOD·NODE·THAT·HAS·NOT·YET·BEEN
                                        ;   PROCESSED.
                                        ;
                                        ;   ON·OUTPUT·R3 =·NODE·TO·BE·PROCESSED·
                                        ;
005220  005767  005774'         GETNDE: TST·    ENFLAG·          ;ELSE·OR NORMAL·NODE·SEARCH?
005224  001062·                         BNE·    11$              ;ELSE·
005226  016703  005650'          1$:    MOV·    CFMN,R3          ;NORMAL: GET·NODE·
005232· 020367  005652'                 CMP·    R3,NFMN·         ;END·OF·POOL?·
005236  001432·                         BEQ·    2$               ;YES·
005240  126763  005670' 000000G·        CMPB·   CFMT,N,FMT(R3)   ;NO: DO·FLU-MOD·TYPES·MATCH?
005246  001437                         BEQ·    3$               ;YES·
005250  005067  005654'                 CLR·    FMNI             ;DELETE·PREVIOUS·NORMAL·NODES·
005254  005267  005670'                 INC·    CFMT             ;STEP·TO NEXT·FLU-MOD·TYPE·
005260  032767  000001  005670'        BIT·    #1,CFMT·         ;SHOULD·WE·SWITCH·TO·ELSE·NODES?·
005266'                                 BON·    1$               ;NO·
005270  016703  005670'          4$:    MOV·    CFMT,R3          ;CONVERT TYPE·TO·SOURCE·CODE
005274  006303                         ASL·    R3
005276  016367  005630' 005676'        MOV·    FMTSC(R3),FMSC
005304  016367  005610' 005672'        MOV·    FMTYP(R3),FMT·   ;&·BIT·FLAG·
005312· 005067  005662'                 CLR·    FMEI             ;DELETE·PREVIOUS·ELSE·NODES·FROM· INDEX
005316  005267  005774'                 INC·    ENFLAG·          ;NOTE·ELSE·SEARCH·
005322· 000423                         BR      11$
005324  062767  000002  005670' 2$:    ADD·    #2,CFMT·         ;STEP·TO NEXT·FLU-MOD·TYPE·FOR·ELSE·
005332· 042767  000001  005670'        BIC·    #1,CFMT·
005340  005067  005654'                 CLR·    FMNI             ;DELETE·NORMAL·NODES·FROM· INDEX·
005344  000751                         BR      4$
005346  016302· 000000G·          3$:    MOV·    N,LGT(R3),R2·    ;STEP·PAST·NODE·
005352· 006302·                         ASL·    R2·
005354  060302·                         ADD·    R3,R2·
005356  062702· 000000G·                ADD·    #N,FMHS,R2·
005362· 010267  005650'                 MOV·    R2,CFMN·         ;UPDATE·CURRENT·NODE·POINTER
005366  000241                         CLC·
005370  000207                         RTS·    PC·
005372· 016703  005656'          11$:   MOV·    CFME,R3          ;GET·CURRENT·ELSE·NODE·
005376· 020367  005660'                 CMP·    R3,NFME·         ;END·OF·POOL?·
005402· 001407                         BEQ·    12$              ;YES·
005404  126763  005670' 000000G·        CMPB·   CFMT,N,FMT(R3)   ;NO: DO·FLU-MOD·TYPES·MATCH?
005412· 001411                         BEQ·    13$              ;YES·
005414  005067  005774'          14$:   CLR·    ENFLAG·          ;SWITCH·TO·NORMAL·NODE·SEARCH·
005420  000702·                         BR      1$
005422· 026727  005670' 000010  12$:    CMP·    CFMT,#MCTYP·     ;END·OF·SEARCH?·
005430  002771                         BLT·    14$              ;NO·
005432· 000261                         SEC·                     ;YES: NO NODE,SEARCH·COMPLETE·
005434  000207                         RTS·    PC·
005436· 016302· 000000G·          13$:   MOV·    N,LGT(R3),R2·    ;STEP·PAST·NODE·
005442· 006302·                         ASL·    R2·
005444  060302·                         ADD·    R3,R2·
005446  062702· 000000G·                ADD·    #N,FMHS,R2·
005452· 010267  005656'                 MOV·    R2,CFME·
005456  000241                         CLC·
005460  000207                         RTS·    PC·
                                        ;
000001                                  .END·
```

```
ASTKMX=·000400 G·
AVELGT=·000025 G·
BITVAL=·000000
BIT0  =·000001
BIT1  =·000002·
BIT10 =·002000
BIT11 =·004000
BIT12·=·010000
BIT13 =·020000
BIT14 =·040000
BIT15 =·100000
BIT2·  =·000004
BIT3  =·000010
BIT4  =·000020
BIT5  =·000040
BIT6  =·000100
BIT7  =·000200
BIT8  =·000400
BIT9  =·001000
BS.CLS=·000002·
BS.DBU=·000004
BS.INA=·000000
BS.OPN=·000001
BS.SRC=·000003
BYTE0 =·000000
BYTE1 =·000001
BYTE2·=·000002·
BYTE3 =·000003
BYTE4 =·000004
BYTE5 =·000005
BYTE6 =·000006
BYTE7 =·000007
BYTE8 =·000010
BYTE9 =·000011
BYTVAL=·000012·
B.BSTA·  000054
B.CNTX·  000046
B.COUQ·  000060
B.FEMA·  000132·
B.FEMB·  000142·
B.FEMC·  000152·
B.FFSA·  000202·
B.FFSB·  000212·
B.FFSC·  000222·
B.FMHR·  000172·
B.FQLS·  000162·
B.FSAZ·  000100
B.FSBZ·  000102·
B.FSCZ·  000104
B.HBLK·  000120
B.HDOC·  000114
B.HRLP·  000126
B.HRLR·  000122·
B.HRLW·  000124
B.NMBR·  000052·
B.NQRY·  000232·
B.QLSZ·  000106
B.QMAP   000234
B.QSPL   000316
B.QTTM   000076
B.QUQP   000056
B.SFDB   000010
B.SIZE   000772·
B.SNDP   000012·
B.SSQ·   000004
B.SSQF   000050
B.STAT   000044
B.STTE   000053
B.UDOC   000110
CBIT·  =·****** GX·
CFME·    005656R·
CFMN·    005650R·
CFMT·    005670R·
CFSA·    005666R·
CF.B0 =·000070
CF.B2·=·000067
CF.B4 =·000066
CF.B6 =·000065
CF.DR0=·000064
CF.DR1=·000063
CPIXMK=·177770 G
CPIXSZ=·000010 G
DBSLEN=·000116
DH.BF0   000002·
DH.BF1   000004
DH.CTL   000000
DH.DMC   000010
DH.FLG   000006
DN.DCK   000000
DN.NTP   000004
DN.NXT   000006
DN.ROT   000002·
010 DN.SIZ  000010
010 EIXCNT=·000010 G
010 ELSCLS=·000007 G
010 ELSMSK=·177761 G
010 EMA·   =·****** GX·
010 EMACLS=·000003 G
010 EMACUT=·001700 G
010 EMAMSZ=·016000 G
010 EMBCLS=·000004 G
010 EMBCUT=·001130 G
010 EMBMSZ=·004400 G
010 EMCMSZ=·001000 G
010 EMXDTF=·****** GX·
010 EMXEXF=·****** GX·
010 EMXMCD=·****** GX·
010 EMXMCF=·****** GX·
010 EMXMSZ=·016000 G
010 EMXMTV=·****** GX·
010 EMXNB1=·****** GX·
010 EMXNB2=·****** GX·
010 EMXNSQ=·****** GX·
010 EMXNVD=·****** GX·
010 EMXSZF=·****** GX·
010 EMXVDC=·****** GX·
010 EMXVVV=·****** GX·
010 EMXZNF=·****** GX·
010 EMX0VD=·****** GX·
010 ENCNT·=·000004 G·
010 ENFLAG·  005774R·
010 FD.FID·  000000
010 FD.FNB·  000006
010 FD.FVR·  000004
010 FD.LEN·  000010
010 FLIXMK=·177740 G·
    FLIXSZ=·000040 G·
014 FLUCLS=·000001 G·
014 FLUCUT=·000016 G·
014 FLUMD·   000000RG·
014 FMAT·    005720R·
    FME·   =·****** GX·
    FMEI·    005662R·
    FMEPE·   005540R·
    FMEPS·   004540R·
    FMEPSZ=·000400 G·
    FMEXF·   005710R·
    FMEXF2·  005712R·
    FMEXL·   005700R·
    FMIN·    005714R·
005 FMIXSZ=·000001 G·
005 FMLOG·   005706R·
005 FMNI·    005654R·
005 FMNPE·   004540R·
005 FMNPS·   000000R·
013 FMNPSZ=·002260 G·
013 FMNI   =·****** GX·
013 FMN2·  =·****** GX·
013 FMSC·    005676R·
013 FMSEQ·   002530R·
    FMT·     005672R·
    FMTSC·   005630R·
    FMTST1   004210R·
    FMTST2·  004714R·
    FMTSUM·  005674R·
    FMTYP·   005610R·
    FNPOSZ=·000400 G·
    FNPSZ·=·005734 G·
    FN.DBR·  000026
    FN.DBS·  000022·
    FN.DHR·  000040
    FN.EMA·  000012·
    FN.EMB·  000014
    FN.EMC·  000016
    FN.FSA·  000000
    FN.FSB·  000002·
    FN.FSC·  000004
    FN.LGO·  000034
    FN.LGU·  000036
    FN.MFO·  000024
    FN.MHR·  000010
    FN.NMB·  000044
    FN.QLS·  000006
    FN.QRY·  000020
    FN.SF0·  000030
    FN.SF1·  000032·
    FN.SHD·  000042·
014 FVAL·    005716R·
003 GETNDE·  005220R·
003 GVMASK·  000270RG·
003 GVSUB·   005702R·
003 GV2·     000462R·
    GV3      000654R·
    HSTKMX=·000100 G·
    IMASK·   001266R·
    INTDEF=·****** GX·
015 IWBIT·=·****** GX·
014 JMPBIT=·****** GX·
    LEVEL·   006000R·
014 LOGCLS=·000002·G·
014 LOWADR=·****** GX·
014 LVLDC·   005664RG·
    M·     =·000062·
014 MATRP2·  003470RG·
014 MCTYP·=·000010
014 MERGE·   004004R·
014 MIXFMC·  005540R·
    N·     =·000002·
014 NDBCLS=·000006 G·
014 NDSIZ·   004144RG·
014 NEST·    005776R·
014 NFME·    005660R·
    NFMN·    005652R·
    NIXCNT=·000020 G·
    NNMASK=·177760 G·
014 NNPE·  =·****** GX·
015 NODERR=·****** GX·
014 NPE·   =·****** GX·
014 NPECNT=·000144 G·
015 NPEVSZ=·000043 G·
015 NPHI·  =·****** GX·
014 NPLOC·   004160R·
014 NPOVF·   004174R·
    NP1MSZ=·010000 G·
    NP2MSZ=·036000 G·
011 NP2OSZ=·000153 G·
011 NP3MSZ=·000524 G·
011 NP3OSZ=·000125 G·
011 NSQSUB·  001104R·
011 NXTADD=·****** GX·
011 N.BACK=·****** GX·
011 N.BFAC=·000004
011 N.BHGH=·000006
011 N.BTCH=·000004
011 N.BUFB=·004000
011 N.BUFW=·002000
011 N.EID·=·****** GX·
011 N.ELSE=·****** GX·
011 N.FMHS=·****** GX·
011 N.FMT·=·****** GX·
011 N.FOS·=·000764
011 N.FSA·=·****** GX·
011 N.LGT·=·****** GX·
011 N.NHS·=·****** GX·
014 N.PHI·=·****** GX·
015 N.PKSZ=·000020
015 N.PKTS=·000043
014 N.QURY=·000031
015 N.SUNT=·000002
015 N2P·   =·****** GX·
    OLDMC·   005704R·     014
015 PARI   =·****** GX·
    PHI·   =·****** GX·
    PSTKMX=·000024 G·
    QE.RO1=·000144
014 QNDCNT=·001000 G·
    QRYSZ·=·002000 G·
    Q.FDSC·  000004       007
014 Q.NQBK·  000000       007
    Q.NUHL·  000002       007
015 Q.SIZE·  000014       007
    SCAN     003624RG·    015
015 SR.ARE·  000114       002·
014 SR.ARS·  000106       002·
    SR.DAY·  000010       002·
    SR.DLT·  000014       002·
015 SR.ECB·  000047       002·
014 SR.ECH·  000046       002·
014 SR.ECL·  000050       002·
014 SR.FIB·  000012       002·
    SR.GRE·  000100       002·
    SR.GRS·  000072       002·
    SR.LEN·  000122       002·
    SR.LIN·  000066       002·
    SR.LIP·  000062       002·
    SR.MON·  000006       002·
    SR.NDC·  000042       002·
    SR.NDS·  000036       002·
015 SR.NIN·  000030       002·
015 SR.NIP·  000022       002·
    SR.SDB·  000032       002·
    SR.SRC·  000002       002·
    SR.SUN·  000000       002·
    SR.TWS·  000056       002·
    SR.WSL·  000052       002·
015 SR.YR·   000004       002·
    SR.1IN·  000024       002·
    SR.1IP·  000016       002·
    SSBIT.=·******* GX·
    SS.FID·  000002       004
    SS.FNB·  000010       004
    SS.FVR·  000006       004
    SS.LEN·  000012       004
    SS.STT·  000000       004
    ST$CNG=·****** GX·
```

```
ST$INX=·****** GX·      SU.IDL=·000000          TDCADR=·****** GX·      VI3MSZ=·000400 G·       WORD8 =·000020
ST$JSQ=·****** GX·      SU.LOD=·000001          TDCBLK=·000010 G·       VMASK·=·****** GX·      WORD9 =·000022
ST$MAT=·****** GX·      SU.SRC=·000002·         TDCHK·=·****** GX·      VVEC· =·****** GX·      WRDVAL=·000024
ST$SEQ=·****** GX·      SU.SRR=·000005          TDCMSZ=·004000 G·       WN.NTP· 000004      012·XBATCH=·000013
ST.ASZ· 000020      006 SU.XPD=·000003          TRMCUT=·000040 G·       WN.NXT· 000006      012·XDBLOA=·000004
ST.BSZ· 000024      006 S.HRL·=·000240          TSTKMX=·000400 G·       WN.ROT· 000002·     012·XDBPRO=·000012
ST.BTC· 000000      006 S1BIT·=·****** GX·      VALVEC· 005724R·    014 WN.SIZ· 000010      012·XDMCIN=·000006
ST.CSZ· 000030      006 S2BIT·=·****** GX·      VALVPT· 005722R·    014 WN.SRC· 000000      012·XFOSMR=·000007
ST.HRL· 000010      006 S3BIT·=·****** GX·      VEC1MX=·000375 G·       WN.TYP· 000001      012·XGTSRE=·000014
ST.LEN· 000044      006 TBIT· =·****** GX·      VEC2MX=·000375 G·       WORD0 =·000000          XHITSK=·000011
ST.QRY· 000002·     006 TDABLK=·000004 G        VEC3MX=·000125 G·       WORD1 =·000002·         XHLMER=·000002
ST.QSZ· 000034      006 TDABMX=·007764 G        VIBCUT=·000036 G·       WORD2·=·000004          XHOTSK=·000010
ST.SCH· 000040      006 TDAMAD=·007760 G        VIER·   000644R·    015 WORD3 =·000006          XMSCHE=·000000
ST.UHL· 000004      006 TDBBLK=·000003 G        VIERR·=·****** GX·      WORD4 =·000010          XQTS  =·000003
ST.XLT· 000014      006 TDBCLS=·000005 G        VI0MSZ=·000400 G·       WORD5 =·000012·         XQT0  =·000001
SU.DBU=·000004          TDBMAD=·007775 G        VI1MSZ=·000400 G·       WORD6 =·000014          XSULOA=·000005
SU.DON=·000006          TDBOSZ=·000074 G        VI2MSZ=·000400 G·       WORD7 =·000016

. ABS.  000000     000
        000000     001
SRCOFF· 000122·    002·
FDSCOF· 000010     003
SUSOFF· 000012·    004
DHROFF· 000012·    005
STTOFF· 000044     006
QSPLOF· 000014     007
BSTOFF· 000772·    010
FNOFFS· 000044 °   011
WNODOF· 000010     012·
DNODOF· 000010     013
FMDATA· 006002·    014
FMCODE· 005462·    015
ERRORS·DETECTED:  0

VIRTUAL·MEMORY·USED:  3709 WORDS· (·15 PAGES)
DYNAMIC·MEMORY:  4916 WORDS· ( 18 PAGES)
ELAPSED·TIME:  00:00:57
FMCDE2,FMCDE2/NL:ME:BEX/-SP=[20,1]JP,M,QTSIZE,FMCDE2·
```

```
TASK    NAME    : QT2
PARTITION NAME  : HSTSPR
IDENTIFICATION  : 0736
TASK  UIC       : [20,3]
STACK    LIMITS: 000212 000611 000400 00256.
PRG XFR ADDRESS: 001502
TOTAL ADDRESS WINDOWS: 2.
TASK  IMAGE  SIZE  : 27072. WORDS
TASK ADDRESS LIMITS: 000000 151557
R-W DISK BLK LIMITS: 000002 000153 000152 00106.

*** ROOT SEGMENT: QT2

R/W MEM  LIMITS: 000000 151557 151560 54128.
DISK BLK LIMITS: 000002 000153 000152 00106.


MEMORY ALLOCATION SYNOPSIS:

SECTION                                          TITLE  IDENT  FILE
-------                                          -----  -----  ----
. BLK.:(RW,I,LCL,REL,CON) 000612 000670 00440.
BSTOFF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000.  FSA             QT2.OBJ;1
                          000000 000000 00000.  FLU             FMCDE2.OBJ;1
CODE  :(RW,I,LCL,REL,CON) 001502 005756 03054.
                          001502 005756 03054.  FSA             QT2.OBJ;1
DATA  :(RW,I,LCL,REL,CON) 007460 030022 12306.
                          007460 030022 12306.  FSA             QT2.OBJ;1
DHROFF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000.  FSA             QT2.OBJ;1
                          000000 000000 00000.  FLU             FMCDE2.OBJ;1
DNODOF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000.  FSA             QT2.OBJ;1
                          000000 000000 00000.  FLU             FMCDE2.OBJ;1
FDSCOF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000.  FSA             QT2.OBJ;1
                          000000 000000 00000.  FLU             FMCDE2.OBJ;1
FMCODE:(RW,I,LCL,REL,CON) 037502 005462 02866.
                          037502 005462 02866.  FLU             FMCDE2.OBJ;1
FMDATA:(RW,I,LCL,REL,CON) 045164 006002 03074.
                          045164 006002 03074.  FLU             FMCDE2.OBJ;1
FNOFFS:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000.  FSA             QT2.OBJ;1
                          000000 000000 00000.  FLU             FMCDE2.OBJ;1
MSGOUT:(RW,I,LCL,REL,CON) 053166 001214 00652.
                          053166 001214 00652.  MESSAG          MSGOUT.OBJ;1
NODES :(RW,I,LCL,REL,CON) 054402 074474 31036.
                          054402 074474 31036.  FSA             QT2.OBJ;1
QSPLOF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000.  FSA             QT2.OBJ;1
                          000000 000000 00000.  FLU             FMCDE2.OBJ;1
SRCOFF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000.  FSA             QT2.OBJ;1
```

```
                                000000 000000 00000.  FLU          FMCDE2.OBJ;1
STTOFF:(RW,I,LCL,ABS,CON)       000000 000000 00000.
                                000000 000000 00000.  FSA          QT2.OBJ;1
                                000000 000000 00000.  FLU          FMCDE2.OBJ;1
SUSOFF:(RW,I,LCL,ABS,CON)       000000 000000 00000.
                                000000 000000 00000.  FSA          QT2.OBJ;1
                                000000 000000 00000.  FLU          FMCDE2.OBJ;1
TRNOFF:(RW,I,LCL,REL,CON)       151076 000240 00160.
                                151076 000240 00160.  FSA          QT2.OBJ;1
WNODOF:(RW,I,LCL,ABS,CON)       000000 000000 00000.
                                000000 000000 00000.  FSA          QT2.OBJ;1
                                000000 000000 00000.  FLU          FMCDE2.OBJ;1
$DPB$$:(RW,I,LCL,REL,CON)       151336 000030 00024.
                                151336 000030 00024.  FSA          QT2.OBJ;1
$$FSR1:(RW,D,GBL,REL,OVR)       151366 000000 00000.
                                151366 000000 00000.  FSA          QT2.OBJ;1
$$FSR2:(RW,D,GBL,REL,CON)       151366 000104 00068.
$$RESL:(RW,I,LCL,REL,CON)       151472 000066 00054.
$$RESM:(RW,I,LCL,REL,CON)       160000 015600 07040.


GLOBAL SYMBOLS:

ASTKMX 000400    EMXFDC 002000    FLUMD  037502-R  NNMASK 177760    N.PHI  000004    ST$JSQ 100000    T.NBAS 000002
AVELGT 000025    EMXMCD 000002    FME    000006    NNPE   054522-R  N.SUM  000012    ST$MAT 160000    T.NDEF 000000
CBIT   010000    EMXMCF 001000    FMEPSZ 000400    NODERR 003124-R  N.TVCT 000013    ST$SEQ 140000    T.SBY1 000000
CPIXMK 177770    EMXMSZ 016000    FMIXSZ 000001    NP     054550-R  N1P    000000    S1BIT  001000    T.SBY2 000001
CPIXSZ 000010    EMXMTV 040000    FMNPSZ 002260    NPE    150550-R  N2P    000002    S2BIT  002000    T.SBY3 000002
DELTIM 151076-R  EMXNB1 000002    FMN1   000010    NPECNT 000144    PAR1   010254-R  S3BIT  004000    T.STAD 000002
DIRERR 003032-R  EMXNB2 000004    FMN2   000012    NPEVSZ 000043    PAR2   010256-R  TAEBIT 020000    T.TRAN 000000
EIXCNT 000010    EMXNFD 000400    FNPOSZ 000400    NPHI   007736-R  PHI    007740-R  TBIT   004000    T.TYPW 000004
ELSBIT 010000    EMXNSQ 100000    FNPSZ  005734    NP1MSZ 010000    PSTKMX 000024    TDABLK 000004    VEC1MX 000375
ELSCLS 000007    EMXNVD 001000    GTIMI  151316-R  NP2MSZ 036000    QNDCNT 001000    TDABMX 007764    VEC2MX 000375
ELSDFN 054526-R  EMXSZF 002000    GVMASK 037772-R  NP2OSZ 000153    QRYSZ  002000    TDAMAD 007760    VEC3MX 000125
ELSMSK 177761    EMXTRL 010000    HSTKMX 000100    NP3MSZ 000524    SCAN   043326-R  TDBBLK 000003    VIBCUT 000036
EMA    025470-R  EMXVDC 004000    INTDEF 010260-R  NP3OSZ 000125    SD.FSA 000010    TDBCLS 000005    VIERR  003210-R
EMACLS 000003    EMXVVV 000001    IWBIT  004000    NXTADD 010262-R  SD.NPS 000016    TDBMAD 007775    VI0MSZ 000400
EMACUT 001700    EMXZNF 004000    JMPBIT 004000    N.BACK 000002    SD.SEC 000012    TDBOSZ 000074    VI1MSZ 000400
EMAFDB 007540-R  EMX0VD 000200    LOGCLS 000002    N.CIDP 000014    SD.TIC 000014    TDCADR 007772-R  VI2MSZ 000400
EMAMSZ 016000    ENCNT  000004    LOWADR 007774-R  N.EID  000012    SECBUF 151304-R  TDCBLK 000010    VI3MSZ 000400
EMBCLS 000004    ENP    000004    LVLDC  053050-R  N.ELSE 000014    SEG1   000000    TDCHK  003250-R  VMASK  024456-R
EMBCUT 001130    EOBIT  010000    L$STAT 000006    N.FLG  000005    SEG2   000002    TDCMSZ 004000    VVEC   036470-R
EMBMSZ 004400    FCSERR 003072-R  MATRP2 043172-R  N.FMHS 000012    SEG3   000004    TRMCUT 000040    $EDMSG 053616-R
EMCMSZ 001000    FLIXMK 177740    MSGOUT 053370-R  N.FMT  000005    SSBIT  004000    TSTKMX 000400
EMXCHF 020000    FLIXSZ 000040    NDBCLS 000006    N.FSA  000010    ST$CNG 120000    TXTBIT 010000
EMXDTF 010000    FLUCLS 000001    NDSIZ  043646-R  N.LGT  000006    ST$INR 060000    T.JSBY 000000
EMXEXF 000001    FLUCUT 000016    NIXCNT 000020    N.NHS  000016    ST$INX 040000    T.MATC 000000


*** TASK BUILDER STATISTICS:

    TOTAL WORK FILE REFERENCES: 27677.
```

```
    WORK FILE READS: 0.
    WORK FILE WRITES: 0.
    SIZE OF CORE POOL: 6634. WORDS (25. PAGES)
    SIZE OF WORK FILE: 2816. WORDS (11. PAGES)

    ELAPSED TIME:00:00:16
```

```
                                ; QTSIZE.MAC    "QUERY TRANSLATORS BUFFER SIZE CONFIGUATION FILE"
                                ;       THIS FILE CONTROLS THE SIZE OF ALL BUFFERS THAT CAN VARY
                                ;       IN SIZE DUE TO THE AMOUNT OF QUERIES OR OTHER SUCH PARAMETERS
                                ;       THAT WOULD BE CHARACTERISTIC OF A SITE.  THESE BUFFERS ARE
                                ;       IN QT0, QT1, QT2, OR QT3.
                                ;
                                ;---QT3---BUFFERS-----------------------------------------------------
                                ;
000125                          VEC3MX==85.                             ;MAXIMUM # CWP'S IN BATCH
000025                          AVELGT==3*2*7/2                         ;AVERAGE CWP SIZE IN TDCT (BYTES)
000400                          VI3MSZ==VEC3MX+2/256.+1*256.            ;SIZE OF VI (NEAREST BLOCK IN BYTES)
001000                          EMCMSZ==VEC3MX/51.+1*256.               ;SIZE OF EMC (NEAREST BLOCK)
004000                          TDCMSZ==VEC3MX*AVELGT+8./N.BUFW+1*N.BUFW;SIZE OF TDCT
000010                          TDCBLK==TDCMSZ/256.                     ;VIRTUAL BLOCK SIZE OF TDCT
000524                          NP3MSZ==VEC3MX*4                        ;SIZE OF NODE POOL
000125                          NP3OSZ==VEC3MX                          ;SIZE OF NODE POOL OVERFLOW AREA
                                ;
                                ;-----QT2----BUFFERS-------------------------------------------------
                                ;
000375                          VEC2MX==253.                            ;MAXIMUM # VECTORS
000400                          VI2MSZ==VEC2MX+2/256.+1*256.            ;SIZE OF VI
004400                          EMBMSZ==9.*256.                         ;SIZE OF EMB
000003                          TDBBLK==3                               ;# BLOCKS (N.BUFW) IN TDCT BUFFER
000074                          TDBOSZ==3*20.                           ;SIZE OF TDCT BUFFER OVERFLOW
007775                          TDBMAD==4093.                           ;MAXIMUM ADDRESS (48 BITS) IN TDCT
000020                          NIXCNT==16.                             ;# INDEX POINTERS FOR NORMAL NODES
177760                          NNMASK==177760                          ;MASK FOR NORMAL INDEX
000010                          EIXCNT==8.                              ;# INDEX PTRS FOR ELSE NODES
177761                          ELSMSK==177761                          ;MASK FOR ELSE INDEX
000004                          ENCNT==4                                ;#ENTRIES TO USE FOR HASH
036000                          NP2MSZ==256.*60.                        ;SIZE OF NODE POOL
000153                          NP2OSZ==7+100.                          ;SIZE OF NODE POOL OVERFLOW AREAR
                                ;
                                ;------QT1-----BUFFERS-----------------------------------------------
                                ;
000375                          VEC1MX==253.                            ;MAX. # TERMS IN FSA-A
000400                          VI1MSZ==VEC2MX+2/256.+1*256.            ;SIZE OF VI
000004                          TDABLK==4                               ;# BLOCKS (N.BUFW) IN TDCT BUFFER
007764                          TDABMX==<TDABLK*<N.BUFW-4>>+4
016000                          EMAMSZ==7.*4*256.                       ;SIZE OF EMA
007760                          TDAMAD==340.*12.                        ;MAX. ADDRESS IN TDCT
010000                          NP1MSZ==4096.                           ;SIZE OF NORMAL NODE POOL
000144                          NPECNT==100.                            ;# NODES IN ELSE POOL
000043                          NPEVSZ==35.                             ;# VECTORS IN ELSE NODES
                                ;
                                ;-----FLU---MODIFIER----BUFFERS--------------------------------------
                                ;
002260                          FMNPSZ==1200.                           ;SIZE OF NORMAL FLU MOD NODE POOL
000400                          FMEPSZ==256.                            ;SIZE OF ELSE FLU MOD NODE POOL
                                ;
                                ;------QT0------BUFFERS-----------------------------------------------
                                ;
002000                          QRYSZ==N.BUFW                           ;SIZE OF QUERY BUFFER
000040                          FLIXSZ==32.                             ;# INDEX PTRS FOR TERMS
177740                          FLIXMK==177740                          ;MASK FOR TERM INDEX
000010                          CPIXSZ==8.                              ;# INDEX PTRS FOR CWP'S
177770                          CPIXMK==177770                          ;MASK FOR CWP INDEX
```

```
                                FMIXSZ==1                               ;#.INDEX.PTRS.FOR.FLU-MOD'S.
                                FNPSZ==3036.                            ;FLU.NODE.POOL.SIZE.
                                FNPOSZ==256.                            ;SIZE.OF.POOL.OVERFLOW.AREA.
                                TSTKMX==256.                            ;MAX. #.TOKENS.TO.BE PUSHED.
                                ASTKMX==256.                            ;MAX. #.ARGUMENTS.TO BE.PUSHED
                                PSTKMX==20.                             ;MAX. #.PROX.NODES.IN.A.CHAIN.
                                HSTKMX==64.                             ;MAX. NESTING.OF.QUERY.
                                QNDCNT==512.                            ;#.NODES.IN.LOGIC.POOL.
                                VI0MSZ==VI1MSZ.                         ;ASSUME.MAX.VI.IS.IN QTi
                                .IIF.LT,VI0MSZ-VI2MSZ,VI0MSZ==VI2MSZ.   ; IF.NOT, RESET.QT0'S.TO.QT2'S VI.SIZE.
                                EMXMSZ==EMAMSZ.                         ;ASSUME.MAX.EMX.IS.IT.QTi
                                .IIF.LT,EMXMSZ-EMBMSZ,EMXMSZ==EMBMSZ.   ; IF.NOT, RESET.TO.QT2'S.
                                ;
                                ;-----ERROR.AND.CLOSE.CODES--------------------------------------
                                ;
                                TRMCUT==32.     ;TERM.CUT-OFF.
                                FLUCUT==14.     ;FLU CUT-OFF.
                                EMACUT==TRMCUT*30.      ;EMA.CUT-OFF.
                                VIBCUT==30.             ;VIB.CUT-OFF.
                                EMBCUT==VIBCUT*20.      ;EMB.CUT-OFF.
                                ;
                                FLUCLS==1       ;CLOSED.DUE.TO FLU.COUNT
                                LOGCLS==2.      ;CLOSED.DUE.TO LOGIC.COUNT.(QLB,SDLB,QEX)
                                EMACLS==3       ;CLOSED.DUE.TO EMA.SIZE.
                                EMBCLS==4       ;CLOSED.DUE.TO EMB.SIZE.
                                TDBCLS==5       ;CLOSED.DUE.TO TDCTB.SIZE.
                                NDBCLS==6       ;CLOSED.DUE.TO QT2.NODE.POOL.SIZE.
                                ELSCLS==7       ;CLOSED.DUE.TO OTHER.CONDITIONS.(FLU.POOL,ETC)
                                ;
```

```
                        .TITLE  FSA-C TRANSLATOR (QT3)
                        .SBTTL
                        .SBTTL  MODULE NAME:     QUERY TRANSLATOR (FSA-C)
                        .SBTTL  AUTHOR:
                        .SBTTL
                        .SBTTL
                        .SBTTL
                        .SBTTL
                        .SBTTL  RELEASE DATE:    24 APR 1978
                        .SBTTL  RELEASE NUMBER: SL120026
                        .SBTTL
                        .SBTTL  **********
                        .SBTTL  FSA-C TRANSLATOR (QT3)
                        .SBTTL  **********
                ;
                ;       QT3 WAS EXTENSIVELY RE-WRITTEN ON 4/24/78
                ;                                  AND 11/15/78
                ;   QT3 READS FROM DISK THE INTERIM DATA STRUCTURES (EMATRIX.EMC)
                ;   GENERATED BY QT0.
                ;   LAYOUT OF EMATRIX.EMC IS AS FOLLOWS:
                ;
                ;       VIRTUAL BLOCK 1          : VI
                ;       VIRTUAL BLOCKS 2-12.     : EMC
                ;
                ;   BASED ON EMC AND VI VECTORS, QT3 BUILDS THE "TERM DETECTOR
                ;   CONTROL TABLE" (TDCT) FOR FSA-C, WRITES THIS STRUCTURE TO DISK
                ;   (FILE NAME = TDCTC.FSA) AND EXITS.
                ;
```

STAT

```
                                .SBTTL  TASK IMMEDIATES AND VARIABLES
000004          LUNFIL  =       4               ;LUN DEFINITION FOR FCS
000002          LUN2    =       2               ;LUN FOR TDCT
000045          MODL    =       45              ;PDP 11/45
000002          EFN.2   =       2               ;EVENT FLAG TWO
001000          BLOCK   =       512.            ;BLOCK LENGTH (BYTES)
000000          SEG1=0                  ;SEGMENT OFFSET DEFINITIONS
000002          SEG2=SEG1+WORD1
000004          SEG3=SEG2+WORD1
000006          SEG4=SEG3+WORD1
```

```
                                        .SBTTL  TASK BUFFERS
000000                                  .PSECT  DATA
                                ;
                                ;
                                ;
000000                          IOST:   .BLKW   2               ;IO STATUS BUFFER
000004  000000                 PARBUF: .WORD   0               ;.XQIO PARAM: 1 - BUFFER ADDS
000006  000000                         .WORD   0               ;     "     "   2 - BUFFER LENGTH (BYTES)
000010  000000                         .WORD   0               ;     "     "   3
000012  000000                         .WORD   0               ;     "     "   4 - VIRTUAL DISK ADDS (HI)
000014  000001                         .WORD   1               ;     "     "   5 -     "       "    (LO)
000016  066563  000000         QTS:    .RAD50  /QTS    /
000022  000000                 BAT.NO: .WORD   0               ;BATCH NUMBER
000024                                  .BLKW   12.             ;REMAINDER OF SEND/RECEIVE BUFFER
                                ;
        177741                  NIBMSK=177741           ;MASK TO ISOLATE ANY STRAY BITS FROM NIBBLE NUMBER
000054  000000                  CURNIB: .WORD 0         ;CURRENT NIBBLE NUMBER
000056  000000                  NXTNIB: .WORD 0         ;NEXT NIBBLE NUMBER
000060                          SEQVAL: .BLKB 4         ;STORAGE FOR NIBBLE VALUES FOR EACH NIBBLE NUMBER
000064                          NIBCP:  .BLKW 16.       ;COUNTER/POINTER FOR EACH NIBBLE VALUE
000124  000000                  SEQCNT: .WORD 0         ;COUNTER OF SEQUENCIAL STATES (# OF NIBBLES)
000126  000000                  OLDNDE: .WORD 0         ;SAVE LOCATION FOR NODE ADDRESS BEING PROCESSED
000130  000001  000002  000004  INDCNV: .WORD 1,2,4,10,20,40,100,200,400,1000,2000,4000,10000,20000,40000
000166  100000                          .WORD 100000    ;INDEX CONVERT (VALUE TO BIT POSITION)
```

```
                                        .SBTTL  DPB AND FDB DEFINITIONS
                                        .MCALL  MOUT$S,EXIT$S
                                        .MCALL  GTIM$S,RCVD$C,SDAT$C,RSUM$C
                                        .MCALL  FDBDF$,FDAT$A,FDRC$A,FDBK$A,FDOP$A,FDOP$R,FSRSZ$
                                        .MCALL  FINIT$,OPEN$W,OPEN$R,CLOSE$,OFNB$R,OFNB$W,WRITE$,WAIT$
                                        .MCALL  NMBLK$,FDBF$A,FDAT$R
                                ;
                                ;   FDB FOR EMATRIX AND TDCTC
                                ;
000170                          EMCFDB::FDBDF$
000330                                  FDRC$A  FD,RWM
000330                                  FDBK$A  EMCLGT,512,,,EFN,2,IOST
000330                                  FDOP$A  LUNFIL
000330                                  FSRSZ$  0,,DATA
000330                          TDCFDB: FDBDF$
000470                                  FDRC$A  FD,RWM
000470                                  FDBK$A  TDCHDR,N,BUFB,,EFN,2,IOST
000470                                  FDOP$A  LUN2
```

```
                                        .SBTTL  READ E-MATRIX
000000                                  .PSECT  CODE
                                ;
                                ;   START OUT BY READING THE VI, VT VECTORS AND THE EMC MATRIX
                                ;
000000                          START:  RCVD$C   QTS,QTS,CODE,DIRERR
000014  012700  000170'                 MOV     #EMCFDB,R0      ;OPEN THE EMATRIX FILE
000020  016701  000022'                 MOV     BAT.NO,R1       ;       FIND THE FDSC
000024  016101  000000G                 MOV     BSTPTR(R1),R1
000030  062701  000152                  ADD     #B.FEMC,R1
000034  004767  000000G                 JSR     PC,BLDEFL       ;       BUILD FILE NAME BLOCK
000040                                  OFNB$R
000052  103445                          BCS     4$
                                ;
                                ;   READ VI
                                ;
000054  012767  001016' 000004' 1$:     MOV     #EMCLGT,PARBUF          ;PAR 1 = BUFFER ADDS
000062  012767  001000  000006'         MOV     #512.,PARBUF+2          ;PAR 2 = BUFFER LENGTH (BYTES)
000070  012701  000000G                 MOV     #IO.RVB,R1              ;R1 = IO FUNCTION CODE
000074  012702  000005                  MOV     #5,R2                   ;R2 = NO. OF QIO PARAMETERS
000100  012703  000004'                 MOV     #PARBUF,R3              ;R3->QIO PARAMETERS
000104                                  CALL    .XQIO
000110  103426                          BCS     4$                      ;IF ERROR
                                ;
                                ;   READ EMC
                                ;
000112  012767  002016' 000004'         MOV     #EMC,PARBUF             ;PAR 1 = BUFFER ADDRESS
000120  016701  001016'                 MOV     EMCLGT,R1               ;R1 = EMC LENGTH IN BLOCKS
000124  001432                          BEQ     10$                     ;BRANCH IF ZERO LENGTH
000126  070127  001000                  MUL     #512.,R1                ;R1 = BYTE LENGTH
000132  010167  000006'                 MOV     R1,PARBUF+2             ;PAR 2 = FILE LENGTH
000136  012767  000002  000014'         MOV     #2,PARBUF+10            ;START READ AT VIRTUAL BLOCK 2
000144  012701  000000G                 MOV     #IO.RVB,R1              ;R1 = IO FUNCTION CODE
000150  012702  000005                  MOV     #5,R2                   ;R2 = NO. OF QIO PARAM'S
000154  012703  000004'                 MOV     #PARBUF,R3              ;R3-> PARAMETERES
000160                                  CALL    .XQIO
000164  103004                          BCC     3$
000166                          4$:     CALL    FCSERR
000172  000167  003006                  JMP     EXIT
                                ;
                                ;   CLOSE FILE
                                ;
000176                          3$:     CLOSE$  #EMCFDB
000206  000167  000024                  JMP     QT3.1
                                ;
                                ;   ZERO LENGTH EMC, CREATE DUMMY TDCT
                                ;
000212                          10$:    GTIM$S  #GTIM1          ;START TIME
000224  012767  000001  004036'         MOV     #1,TDCT+2       ;CREATE SUICIDE TABLE
000232  000167  000536                  JMP     QT3.3
```

```
                                        .SBTTL  INITIALIZE TRANSLATION PROCESS
                                ;
                                ;
                                ;
000236                          QT3.1:  GTIM$S  #GTIM1          ;START TIME
000250  012703  001016'                MOV     #VMASK,R3               ;R3->VMASK = VI
000254  016363  000002  000000         MOV     N.NIB(R3),N.LGT(R3)     ;MOVE VECTOR LENGTH TO ITS PROPER PLACE
000262  021327  000125                 CMP     (R3),#VEC3MX            ;DO VECTORS EXCEED DESIGN LIMIT?
000266  103402                        BLO     1$                      ;NO -- CONTINUE
000270  004767  002654                 JSR     PC,LIMIT                ;YES -- ISSUE WARNING
000274  005063  000002          1$:    CLR     N.NIB(R3)               ;CURRENT NIBBLE = FIRST
000300  012767  014032' 014030'        MOV     #NPOOL,FREEQ            ;RESET NEXT FREE NODE PTR TO POOL START
000306  012767  000002  004036'        MOV     #2,TDCT+2               ;RESTORE NORMAL START STATE
```

```
                                        .SBTTL  MAIN TRANSLATION PROCESSING LOOP
                                ;
                                ;   MAIN TRANSLATION PROCESSING LOOP
                                ;
000314  016301  000002         NIBCHK: MOV     N.NIB(R3),R1    ;GET CURRENT NIBBLE FOR NODE
000320  010167  000054'                 MOV     R1,CURNIB       ;SET CURRENT NIBBLE
000324  005201                         INC     R1              ;CALCULATE AND SET NEXT NIBBLE
000326  042701  177774                 BIC     #177774,R1
000332  010167  000056'                 MOV     R1,NXTNIB
000336  010367  000126'                 MOV     R3,OLDNDE       ;SAVE NODE ADDRESS
000342  022713  000001                 CMP     #1,(R3)         ;DOES VECTOR LENGTH=1?
000346  001004                         BNE     1$              ;NO -- CONTINUE
000350  004767  001616                 JSR     PC,SEQST        ;YES - FINISH VECTOR WITH SEQUENCIAL STATES
000354  000167  000330                 JMP     MSK.12
000360  011301                 1$:     MOV     (R3),R1         ;R1=VECTOR LENGTH
000362  062703  000004                 ADD     #N.VEC,R3       ;R3=START OF VECTORS IN NODE
000366  016700  000054'                 MOV     CURNIB,R0       ;R0=CURRENT NIBBLE NUMBER
000372  011302                         MOV     (R3),R2         ;R2=ADDRESS OF EMC ENTRY
000374  062702  002016'                 ADD     #EMC,R2
000400  032700  000002                 BIT     #2,R0           ;WHAT IS NIBBLE POSITION?
000404                                 BON     2$
000406  111260  000060'                 MOVB    (R2),SEQVAL(R0) ;  0 OR 1, SO STORE LOW ORDER BYTE
000412  000403                         BR      3$
000414  116260  000001  000060' 2$:     MOVB    1(R2),SEQVAL(R0); 2 OR 3, SO STORE HIGH ORDER BYTE
000422  006300                 3$:     ASL     R0              ;R0=CURRENT NIBBLE (WORD OFFSET)
000424  012702  000064'                 MOV     #NIBCP,R2       ;R2=ADDRESS OF NIBBLE COUNTERS
000430  012704  000020                 MOV     #16.,R4         ;R4=# NIBBLE COUNTERS
000434  005022                 4$:     CLR     (R2)+           ;RESET ALL COUNTERS
000436  077402                         SOB     R4,4$
                                ;
                                ;  THE FOLLOWING LOOP IS THE MOST TIME CRITICAL LOOP IN THIS TASK, IN ORDER
                                ;  TO SPEED IT UP, IT HAS BEEN SPLIT INTO 4 LOOPS SO THAT THE SHIFT COUNT
                                ;  COULD BE IMMEDIATE ACCESS, THIS RESULTS IN A 7 TO 19% SPEED UP.
                                ;   THE PURPOSE OF THE LOOP IS TO STORE THE OCCURRENCE COUNT FOR EACH NIB VALUE.
                                ;
000440  000170  000444'                 JMP     @5$(R0)         ;CHOOSE APPROPRIATE LOOP
000444  000454'                 5$:     10$             ;FIRST NIBBLE
000446  000500'                         11$             ;SECOND NIBBLE
000450  000526'                         12$             ;THIRD NIBBLE
000452  000552'                         13$             ;LAST NIBBLE
                                ;
000454  012304                 10$:    MOV     (R3)+,R4        ;R4=NIBBLE VALUE (WORD OFFSET)
000456  016404  002016'                 MOV     EMC(R4),R4
000462  006304                         ASL     R4
000464  042704  177741                 BIC     #NIBMSK,R4
000470  005264  000064'                 INC     NIBCP(R4)       ;STEP COUNTER FOR NIBBLE VALUE
000474  077111                         SOB     R1,10$          ;REPEAT FOR EVERY VECTOR
000476  000437                         BR      20$
                                ;
000500  012304                 11$:    MOV     (R3)+,R4        ;LOOP FOR 2ND NIBBLE
000502  016404  002016'                 MOV     EMC(R4),R4
000506  072427  177775                 ASH     #-3,R4
000512  042704  177741                 BIC     #NIBMSK,R4
000516  005264  000064'                 INC     NIBCP(R4)
000522  077112                         SOB     R1,11$
000524  000424                         BR      20$
000526  012304                 12$:    MOV     (R3)+,R4        ;LOOP FOR 3RD NIBBLE
```

```
 207 000530  116404  002017'                MOVB    EMC+1(R4),R4
 208 000534  006304                          ASL     R4
 209 000536  042704  177741                  BIC     #NIBMSK,R4
 210 000542  005264  000064'                 INC     NIBCP(R4)
 211 000546  077111                          SOB     R1,12$
 212 000550  000412                          BR      20$
 213 000552  012304                13$:      MOV     (R3)+,R4        ;LOOP FOR 4TH NIBBLE
 214 000554  116404  002017'                MOVB    EMC+1(R4),R4
 215 000560  072427  177775                  ASH     #-3,R4
 216 000564  042704  177741                  BIC     #NIBMSK,R4
 217 000570  005264  000064'                 INC     NIBCP(R4)
 218 000574  077112                          SOB     R1,13$
 219                                 ;
 220 000576  012702  000064'        20$:    MOV     #NIBCP,R2       ;R2=ADDRESS OF NIBBLE COUNTERS
 221 000602  012704  000020                  MOV     #16.,R4         ;R4=# NIBBLE COUNTERS
 222 000606  005001                          CLR     R1              ;R1=# UNIQUE NIBBLE VALUES
 223 000610  005722                21$:      TST     (R2)+           ;WAS THIS VALUE PRESENT?
 224 000612  001401                          BEQ     22$             ;NO -- IGNORE
 225 000614  005201                          INC     R1              ;YES - RECORD IT
 226 000616  077404                22$:      SOB     R4,21$          ;TEST ALL COUNTERS
 227 000620  022701  000001                  CMP     #1,R1           ;WAS ONLY ONE VALUE PRESENT?
 228 000624  001022                          BNE     30$             ;NO -- INDEX STATE NEEDED
 229 000626  005267  000124'                 INC     SEQCNT          ;YES - STEP COUNTER OF SEQUENCIAL STATES
 230 000632  016703  000126'                 MOV     OLDNDE,R3       ;RESTORE NODE ADDRESS
 231 000636  016763  000056'  000002        MOV     NXTNIB,N.NIB(R3);STEP TO NEXT NIBBLE
 232 000644  001402                          BEQ     23$             ;** LAST NIBBLE, GENERATE REQUIRED STATE
 233 000646  000167  177442                  JMP     NIBCHK          ;** INTERMEDIATE NIBBLE, CONTINUE CHECKING
 234                                 ;   AT THIS POINT WE ARE AT THE END OF THE VECTOR AND SOME SEQUENCIAL
 235                                 ;   STATES MUST BE GENERATED BEFORE WE CONTINUE WITH NEXT VECTOR.
 236 000652  005267  000054'        23$:    INC     CURNIB          ;CURNIB=4, BEGINNING OF NEXT VECTOR
 237 000656  004767  000654                  JSR     PC,SEQPAD       ;GENERATE REQUIRED STATES
 238 000662  004767  001474                  JSR     PC,GVMASK       ;STEP TO NEXT VECTORS
 239 000666  000167  177422                  JMP     NIBCHK          ;CONTINUE PROCESSING THIS NODE
 240                                 ;   AT THIS POINT WE MUST GENERATE A INDEX STATE PLUS ANY LEADING SEQ STATES.
 241 000672  005767  000124'        30$:    TST     SEQCNT          ;WERE ANY SEQUENCIAL STATES DETECTED?
 242 000676  001402                          BEQ     31$             ;NO -- JUST GENERATE THE INDEX STATE
 243 000700  004767  000632                  JSR     PC,SEQPAD       ;YES - GENERATE REQUIRED SEQ STATES
 244 000704  004767  000172        31$:    JSR     PC,INDEX        ;GENERATE INDEX STATE
 245                                 ;
 246                                 ;   AT THIS POINT THE NODE WE WERE PROCESSING HAS BEEN SPLIT INTO MULTIPLE
 247                                 ;   NODES. THE TDCT MUST BE SCANNED TO FIND AN INCOMPLETED INDEX STATE TO
 248                                 ;   CONTINUE PROCESSING WITH.
 249                                 ;
 250 000710  016700  004016'        MSK.12: MOV     CHKOFF,R0       ;R0=LAST COMPLETED STATE IN TDCT
 251 000714  020067  004024'        1$:     CMP     R0,TDCNXT       ;ARE WE AT THE END OF THE TDCT?
 252 000720  103025                          BHIS    QT3.3           ;YES - WE ARE FINISHED THE TRANSLATION
 253 000722  032760  000001  004040'        BIT     #BIT0,TDCT+SEG3(R0);NO--IS STATE INCOMPLETE?
 254 000730                                  BON     2$              ;YES - FOUND NODE TO PROCESS
 255 000732  062700  000006                  ADD     #L$STAT,R0      ;NO -- STEP TO NEXT TDCT ENTRY
 256 000736  000766                          BR      1$
 257                                 ;   AT THIS POINT ANOTHER NODE HAS BEEN FOUND TO PROCESS.
 258 000740  016003  004036'        2$:     MOV     TDCT+SEG2(R0),R3;GET NODE ADDRESS FROM TDCT ENTRY
 259 000744  010067  004020'                MOV     R0,FSAOFF       ;SET OFFSET OF TDCT ENTRY TO BE PROCESSED
 260 000750  010067  004016'                MOV     R0,CHKOFF       ;SET OFFSET OF LAST COMPLETED TDCT ENTRY
 261 000754  005763  000002                  TST     N.NIB(R3)       ;LAST NIBBLE?
 262 000760  001003                          BNE     3$              ;NO -- INTERMEDIATE NIBBLE, CONTINUE
 263 000762  004767  001374                  JSR     PC,GVMASK       ;YES - STEP TO NEXT VECTORS
```

```
    264 000766  103750                        BCS     MSK.12          ;JUST A MATCH STATE, NO NODE LEFT TO PROCESS
    265 000770  000167  177320          3$:   JMP     NIBCHK
```

```
                                        .SBTTL  TRANSLATION COMPLETE
                                ;
                                ;       TRANSLATION PROCESS IS COMPLETED.
                                ;
000774                          QT3.3:  GTIM$S  #SECBUF          ;END TIME
001006  004767  000000'                JSR     PC,DELTIM        ;COMPUTE DELTA TIME
001012  016767  000206' 000000C        MOV     SECBUF,BAT.NO+SD.SEC    ;TRANSFER DELTA TIME TO SEND BUFFER
001020  016767  000210' 000000C        MOV     SECBUF+2,BAT.NO+SD.TIC
001026  016767  004022' 000000C        MOV     NXTADD,BAT.NO+SD.FSA
001034                                  CALL    WRTTDC
                                ;
                                ;   NOTIFY QTS:
                                ;
001040                                  SDAT$C  QTS,BAT.NO,,CODE
001046  103004                          BCC     1$
001050                                  CALL    DIRERR
001054  000167  002124                  JMP     EXIT
001060                          1$:     RSUM$C  QTS,CODE                ;RESUME QTS
001066  103002                          BCC     2$
001070                                  CALL    DIRERR
001074                          2$:     EXIT$S
```

```
                                        .SBTTL  INDEX STATE GENERATION
                                ;
                                ;       INDEX STATE GENERATION
                                ;
001102  012704  000124'        INDEX:  MOV     #NIBCP+40,R4    ;R4=ADDRESS OF END OF NIBBLE COUNTERS
001106  012702  000020                  MOV     #16.,R2         ;R2=# NIBBLE COUNTERS
001112  005001                          CLR     R1              ;R1=INDEX MASK (SEG1 OF TDCT ENTRY)
001114  005744                  1$:     TST     -(R4)           ;WAS THIS NIBBLE VALUE PRESENT?
001116  001401                          BEQ     2$              ;NO -- THEN DON'T MARK THIS POSITION
001120  000261                          SEC                     ;YES - THEN MARK THIS POSITION
001122  006101                  2$:     ROL     R1              ;ENTER POSITION INTO MASK
001124  077205                          SOB     R2,1$           ;REPEAT FOR ALL NIBBLE VALUES
001126  016704  004020'                MOV     FSAOFF,R4       ;R4=OFFSET TO TDCT ENTRY BEING PROCESSED
001132  020467  004024'                CMP     R4,TDCNXT       ;ARE WE APPENDING TO TDCT?
001136  103405                          BLO     200$            ;NO -- CONTINUE
001140  062767  000006  004024'        ADD     #6,TDCNXT       ;YES - THEN UPDATE POINTERS
001146  005267  004022'                INC     NXTADD
001152  062704  004034'        200$:   ADD     #TDCT,R4        ;R4=ADDRESS OF TDCT ENTRY
001156  010124                          MOV     R1,(R4)+        ;TDCT STATE=R1,NXTADD,#ST$INX
001160  016724  004022'                MOV     NXTADD,(R4)+
001164  012714  040000                MOV     #ST$INX,(R4)
001170  016703  004024'                MOV     TDCNXT,R3       ;R3=ADDRESS OF NEXT AVAILABLE TDCT ENTRY
001174  062703  004034'                ADD     #TDCT,R3
001200  012704  000124'                MOV     #NIBCP+40,R4    ;R4=ADDRESS OF END OF NIBBLE COUNTERS
001204  012701  000020                MOV     #16.,R1         ;R1=# NIBBLE COUNTERS
001210  016702  014030'                MOV     FREEQ,R2        ;R2=ADDRESS OF NEXT FREE NODE
001214  020227  015302'        20$:    CMP     R2,#NPOOLE      ;HAVE WE REACHED THE END OF THE POOL?
001220  101402                          BLOS    21$             ;NO -- CONTINUE
001222  012702  014032'                MOV     #NPOOL,R2       ;YES - RESET TO BEGINNING OF POOL
001226  014412                  21$:    MOV     -(R4),(R2)      ;TRANSFER VECTOR LENGTH TO NODE
001230  001002                          BNE     22$             ;IF VECTOR LENGTH NON-ZERO, COMPLETE NODE
001232  077103                          SOB     R1,21$          ;OTHERWISE, TRY NEXT NIBBLE COUNTER
001234  000417                          BR      23$             ;IF FINISHED ALL COUNTERS, READY FOR NEXT STEP
                                ;   AT THIS POINT A NODE AND TDCT STATE NEED TO BE CREATED FOR THIS INDEX VALUE
001236  012200                  22$:    MOV     (R2)+,R0        ;R0=SIZE OF VECTOR AREA (BYTES)
001240  006300                          ASL     R0
001242  016722  000056'                MOV     NXTNIB,(R2)+    ;NEXT NODE ENTRY = NEXT NIBBLE NUMBER
001246  010214                          MOV     R2,(R4)         ;THIS NIBBLE COUNTER BECOMES A NODE VECTOR PTR
001250  005023                          CLR     (R3)+           ;NOW CREATE TDCT STATE, INDEX MASK=0
001252  010213                          MOV     R2,(R3)         ;NEXT SEGMENT POINTS TO NODE
001254  162723  000004                SUB     #N.VEC,(R3)+
001260  012723  040001                MOV     #ST$INX!BIT0,(R3)+;DEFINE STATE AS INCOMPLETE INDEX STATE
001264  060002                          ADD     R0,R2           ;ADD VECTOR AREA TO NODE POSITION POINTER
001266  005267  004022'                INC     NXTADD          ;STEP TDCT STATE ADDRESS
001272  077130                          SOB     R1,20$          ;FINISH ALL COUNTERS
001274  162703  004034'        23$:    SUB     #TDCT,R3        ;CONVERT ADDRESS TO OFFSET
001300  010367  004024'                MOV     R3,TDCNXT       ;UPDATE NEXT TDCT OFFSET POINTER
001304  020367  004026'                CMP     R3,TDCMAX       ;HAVE WE OVERFLOWED THE TDCT?
001310  103404                          BLO     24$             ;NO -- CONTINUE
001312  004767  001604                JSR     PC,TDCERR       ;YES - REPORT CONDITION AND EXIT
001316  000167  001662                JMP     EXIT
001322  010267  014030'        24$:    MOV     R2,FREEQ        ;UPDATE FREE NODE ADDRESS POINTER
001326  016703  000126'                MOV     OLDNDE,R3       ;R3=ADDRESS OF NODE DEFINING VECTORS
001332  011301                          MOV     (R3),R1         ;R1=VECTOR LENGTH
001334  062703  000004                ADD     #N.VEC,R3       ;R3=ADDRESS OF VECTORS
001340  016700  000054'                MOV     CURNIB,R0       ;R0=NIBBLE NUMBER (WORD OFFSET)
001344  006300                          ASL     R0
```

```
                                        ;
                                        ;   THE FOLLOWING LOOP IS THE SECOND MOST TIME CRITICAL LOOP IN THIS TASK,
                                        ;   IN ORDER TO SPEED IT UP, IT HAS BEEN SPLIT INTO 4 LOOPS SO THAT THE SHIFT
                                        ;   COUNT COULD BE IMMEDIATE ACCESS, THIS RESULTS IN A 5 TO 13% SPEED UP,
                                        ;   THE PURPOSE OF THE LOOP IS TO FILL IN THE VECTOR AREA FOR EACH NODE,
                                        ;
001346  000170  001352'                        JMP     @25$(R0)        ;CHOOSE APPROPRIATE LOOP
001352  001362'                        25$:    30$             ;FIRST NIBBLE
001354  001414'                                31$             ;SECOND NIBBLE
001356  001450'                                32$             ;THIRD NIBBLE
001360  001502'                                33$             ;LAST NIBBLE
                                        ;
001362  012302                          30$:    MOV     (R3)+,R2        ;R2=VECTOR
001364  016204  002016'                        MOV     EMC(R2),R4      ;R4=NIBBLE VALUE (WORD OFFSET)
001370  006304                                ASL     R4
001372  042704  177741                        BIC     #NIBMSK,R4
001376  010274  000064'                        MOV     R2,@NIBCP(R4)   ;TRANSFER VECTOR TO NEXT VECTOR ENTRY OF NODE
001402  062764  000002  000064'               ADD     #2,NIBCP(R4)    ; FOR THIS NIBBLE VALUE, ALSO STEP VECTOR PTR
001410  077114                                SOB     R1,30$          ;REPEAT FOR ALL VECTORS
001412  000207                                RTS     PC
                                        ;
001414  012302                          31$:    MOV     (R3)+,R2        ;LOOP FOR 2ND NIBBLE
001416  016204  002016'                        MOV     EMC(R2),R4
001422  072427  177775                        ASH     #-3,R4
001426  042704  177741                        BIC     #NIBMSK,R4
001432  010274  000064'                        MOV     R2,@NIBCP(R4)
001436  062764  000002  000064'               ADD     #2,NIBCP(R4)
001444  077115                                SOB     R1,31$
001446  000207                                RTS     PC
001450  012302                          32$:    MOV     (R3)+,R2        ;LOOP FOR 3RD NIBBLE
001452  116204  002017'                        MOVB    EMC+1(R2),R4
001456  006304                                ASL     R4
001460  042704  177741                        BIC     #NIBMSK,R4
001464  010274  000064'                        MOV     R2,@NIBCP(R4)
001470  062764  000002  000064'               ADD     #2,NIBCP(R4)
001476  077114                                SOB     R1,32$
001500  000207                                RTS     PC
001502  012302                          33$:    MOV     (R3)+,R2        ;LOOP FOR 4TH NIBBLE
001504  116204  002017'                        MOVB    EMC+1(R2),R4
001510  072427  177775                        ASH     #-3,R4
001514  042704  177741                        BIC     #NIBMSK,R4
001520  010274  000064'                        MOV     R2,@NIBCP(R4)
001524  062764  000002  000064'               ADD     #2,NIBCP(R4)
001532  077115                                SOB     R1,33$
001534  000207                                RTS     PC
```

```
                                        .SBTTL  SEQUENCIAL STATE GENERATION
                                ;
                                ;   SEQUENCIAL STATE GENERATION
                                ;
001536                          SEQPAD: ;THIS SUBROUTINE GENERATES LEADING SEQUENCIAL STATES
001536  005002                          CLR     R2              ;R2=TDCT APPEND INDICATOR (0=NOT APPENDING)
001540  026767  004020' 004024'        CMP     FSAOFF,TDCNXT   ;ARE WE AT END OF TDCT?
001546  001001                        BNE     2$              ;NO -- THEN WE ARE REPLACING AN INTERNAL STATE
001550  005202                        INC     R2              ;YES - THEN WE ARE APPENDING (R2=1)
001552  016704  004020'          2$:   MOV     FSAOFF,R4       ;R4=ADDRESS OF TDCT STATE BEING PROCESSED
001556  062704  004034'                ADD     #TDCT,R4
001562  016700  000054'                MOV     CURNIB,R0       ;R0=NIBBLE NUMBER OF FIRST SEQ POSSIBLE
001566  166700  000124'                SUB     SEQCNT,R0
001572  016701  000124'                MOV     SEQCNT,R1       ;R1=# NIBBLES FOR WHICH SEQ STATES ARE POSSIBLE
001576  006301                        ASL     R1              ;R1= (WORD OFFSET)
001600  000171  001604'                JMP     @1$(R1)         ;CHOOSE APPROPRIATE ROUTINE TO HANDLE CONDITION
001604  000001                  1$:   1               ;ILLEGAL CONDITION, ABORT IF IT OCCURS
001606  001616'                       10$             ;ONLY 1 NIBBLE HAD SINGLE TRANSITION
001610  001624'                       20$             ;2 CONSECUTIVE NIBBLES HAD SINGLE TRANSITION
001612  001666'                       30$             ;3 CONSECUTIVE NIBBLES HAD SINGLE TRANSITION
001614  001750'                       40$             ;ENTIRE VECTOR HAD ONLY ONE TRANSITION
                                ;   ONLY ONE NIBBLE, MUST GENERATE INDEX SEQUENCIAL
001616  004767  000276          10$:  JSR     PC,STISQ        ;GENERATE INDEX SEQUENCIAL STATE
001622  000477                        BR      100$
                                ;   2 NIBBLES, BUT ARE THEY PART OF THE SAME BYTE?
001624  032700  000001          20$:  BIT     #1,R0           ;IS FIRST NIBBLE NUMBER=0 OR 2?
001630                                BON     21$             ;NO -- NOT SAME BYTE
001632  004767  000232                JSR     PC,STJSQ        ;YES - GENERATE JUMP SEQUENCIAL STATE
001636  000471                        BR      100$
001640  004767  000254          21$:  JSR     PC,STISQ        ;  GENERATE 2 INDEX SEQUENCIAL STATES
001644  012702  000001                MOV     #1,R2           ;WE ARE NOW APPENDING TO TDCT
001650  016704  004024'                MOV     TDCNXT,R4
001654  062704  004034'                ADD     #TDCT,R4
001660  004767  000234                JSR     PC,STISQ
001664  000456                        BR      100$
                                ;   3 NIBBLES, MUST GENERATE INDEX AND JUMP STATE, BUT IN WHAT ORDER?
001666  032700  000001          30$:  BIT     #1,R0           ;IS FIRST NIBBLE NUMBER=0?
001672                                BON     31$             ;NO -- INDEX STATE FIRST
001674  004767  000170                JSR     PC,STJSQ        ;YES - GENERATE JUMP SEQ THEN INDEX SEQ
001700  012702  000001                MOV     #1,R2           ;WE ARE NOW APPENDING TO TDCT
001704  016704  004024'                MOV     TDCNXT,R4
001710  062704  004034'                ADD     #TDCT,R4
001714  004767  000200                JSR     PC,STISQ
001720  000440                        BR      100$
001722  004767  000172          31$:  JSR     PC,STISQ        ;  GENERATE INDEX SEQ THEN JUMP SEQ
001726  012702  000001                MOV     #1,R2           ;WE ARE NOW APPENDING TO TDCT?
001732  016704  004024'                MOV     TDCNXT,R4
001736  062704  004034'                ADD     #TDCT,R4
001742  004767  000122                JSR     PC,STJSQ
001746  000425                        BR      100$
                                ;   ENTIRE VECTOR, BUT IS IT AT END OF TDCT?
001750  005702                  40$:  TST     R2              ;ARE WE APPENDING?
001752  001412                        BEQ     41$             ;NO -- CANNOT USE SEQ STATE, NEXT ADDRESS WRONG
001754  116724  000060'              MOVB    SEQVAL,(R4)+    ;YES - GENERATE SEQ STATE
001760  116724  000062'              MOVB    SEQVAL+2,(R4)+
001764  005024                        CLR     (R4)+
001766  012724  156000                MOV     #ST$SEQ!TXTBIT!S3BIT!S2BIT,(R4)+
```

```
 449 001772  005267  004022'                INC     NXTADD
 450 001776  000411                         BR      100$
 451 002000  004767  000064          41$:   JSR     PC,STJSQ        ;GENERATE 2 JUMP SEQ STATES
 452 002004  005202                         INC     R2              ;WE ARE NOW APPENDING TO TDCT
 453 002006  016704  004024'                MOV     TDCNXT,R4
 454 002012  062704  004034'                ADD     #TDCT,R4
 455 002016  004767  000046                 JSR     PC,STJSQ
 456                                 ;
 457 002022  005702                 100$:   TST     R2              ;WERE ANY STATES APPENDED TO TDCT?
 458 002024  001413                         BEQ     101$            ;NO
 459 002026  162704  004034'                SUB     #TDCT,R4        ;YES - R4=NEXT TDCT ENTRY OFFSET
 460 002032  010467  004024'                MOV     R4,TDCNXT       ;UPDATE NEXT TDCT ENTRY OFFSET PTR
 461 002036  020467  004026'                CMP     R4,TDCMAX       ;HAVE WE OVERFLOWED TDCT?
 462 002042  103404                         BLO     101$            ;NO -- CONTINUE
 463 002044  004767  001052                 JSR     PC,TDCERR       ;YES - REPORT CONDITION AND EXIT
 464 002050  000167  001130                 JMP     EXIT
 465 002054  016767  004024' 004020' 101$:  MOV     TDCNXT,FSAOFF   ;ANY MORE STATES MUST BE APPENDED TO TDCT
 466 002062  005067  000124'                CLR     SEQCNT          ;ALL DETECTED SEQ STATES HAVE BEEN CREATED
 467 002066  000207                         RTS     PC              ;FINISHED
 468                                 ;
 469 002070                          STJSQ:  ;THIS SUBROUTINE ENTERS JUMP SEQ STATES INTO TDCT
 470 002070  116024  000060'                MOVB    SEQVAL(R0),(R4)+;ENTER BYTE JUMP IS CONDITIONAL ON
 471 002074  105024                         CLRB    (R4)+
 472 002076  060267  004022'                ADD     R2,NXTADD       ;IF APPENDING, STEP NEXT ADDRESS
 473 002102  016724  004022'                MOV     NXTADD,(R4)+    ;ENTER NEXT STATE ADDRESS
 474 002106  012724  114000                 MOV     #ST$JSQ!JMPBIT!TXTBIT,(R4)+;ENTER STATE OF COND, JUMP SEQ
 475 002112  062700  000002                 ADD     #2,R0           ;STEP PAST NIBBLES HANDLED BY THIS STATE
 476 002116  000207                         RTS     PC
 477                                 ;
 478 002120                          STISQ:  ;THIS SUBROUTINE ENTERS INDEX SEQ STATES INTO TDCT
 479 002120  116001  000060'                MOVB    SEQVAL(R0),R1
 480 002124  005200                         INC     R0              ;STEP PAST NIBBLE HANDLED BY THIS STATE
 481 002126  032700  000001                 BIT     #1,R0           ;WHICH NIBBLE OF SAVED BYTE ARE WE AFTER?
 482 002132                                 BON     1$              ;LOW ORDER NIBBLE
 483 002134  072127  177775                 ASH     #-3,R1          ;HIGH ORDER NIBBLE
 484 002140  000401                         BR      2$
 485 002142  006301                  1$:    ASL     R1
 486 002144  042701  177741          2$:    BIC     #NIBMSK,R1      ;R1=NIBBLE VALUE (WORD OFFSET)
 487 002150  016124  000130'                MOV     INDCNV(R1),(R4)+;ENTER INDEX MASK
 488 002154  060267  004022'                ADD     R2,NXTADD       ;IF APPENDING, STEP NEXT ADDRESS
 489 002160  016724  004022'                MOV     NXTADD,(R4)+    ;ENTER NEXT ADDRESS
 490 002164  012724  040000                 MOV     #ST$INX,(R4)+   ;ENTER STATE OF INDEX STATE COMPLETE
 491 002170  000207                         RTS     PC
 492                                 ;
 493 002172                          SEQST:  ;THIS SUBROUTINE ENTERS TRAILING SEQ STATES (VECTOR LENGTH=1)
 494 002172  012767  000004  000124'        MOV     #4,SEQCNT       ;USE SEQPAD TO HANDLE INITIAL STATES
 495 002200  166767  000054' 000124'        SUB     CURNIB,SEQCNT   ;SEQCNT=# SEQ STATES POSIBLE (NIBBLE COUNT)
 496 002206  012767  000004  000054'        MOV     #4,CURNIB       ;CURNIB=LAST (FIRST OF NEXT VECTOR)
 497 002214  016703  000126'                MOV     OLDNDE,R3
 498 002220  062703  000004                 ADD     #N,VEC,R3       ;R3=ADDRESS OF VECOTOR IN NODE
 499 002224  011305                         MOV     (R3),R5         ;R5=VECTOR
 500 002226  062705  002016'                ADD     #EMC,R5         ;R5=EMC ADDRESS
 501 002232  111567  000060'                MOVB    (R5),SEQVAL     ;SET UP SAVED NIBBLE VALUES
 502 002236  112567  000061'                MOVB    (R5)+,SEQVAL+1
 503 002242  111567  000062'                MOVB    (R5),SEQVAL+2
 504 002246  112567  000063'                MOVB    (R5)+,SEQVAL+3
 505 002252  004767  177260                 JSR     PC,SEQPAD       ;CREATE TRANSITION STATES
```

```
 506 002256  016704  004024'                MOV     TDCNXT,R4       ;R4=NEXT AVAILABLE STATE ADDRESS
 507 002262  062704  004034'                ADD     #TDCT,R4
 508 002266  005715                  1$:    TST     (R5)            ;IS NEXT STATE A MATCH STATE?
 509 002270  100407                         BMI     20$             ;YES -- CREATE MATCH STATE
 510 002272  012524                         MOV     (R5)+,(R4)+     ;ENTER SEQ VALUE
 511 002274  005024                         CLR     (R4)+
 512 002276  012724  156000                MOV     #ST$SEQ!TXTBIT!S3BIT!S2BIT,(R4)+;DEFINE STATE AS SEQ STATE
 513 002302  005267  004022'                INC     NXTADD          ;UPDATE NEXT STATE ADDRESS
 514 002306  000767                         BR      1$              ;CONTINUE UNTIL MATCH STATE FOUND
 515 002310  011514                  20$:   MOV     (R5),(R4)       ;ENTER MATCH CODE VALUE
 516 002312  042724  100000                BIC     #100000,(R4)+   ;REMOVE MATCH CODE INDICATOR
 517 002316  012724  000001                MOV     #SUISTA,(R4)+   ;ENTER NEXT ADDRESS=SUISIDE STATE
 518 002322  012724  164000                MOV     #ST$MAT!TBIT,(R4)+;ENTER STATE OF NORMAL TERM, MATCH STATE
 519 002326  005267  004022'                INC     NXTADD          ;UPDATE NEXT STATE ADDRESS
 520 002332  162704  004034'                SUB     #TDCT,R4        ;CONVERT ADDRESS TO OFFSET
 521 002336  010467  004024'                MOV     R4,TDCNXT       ;UPDATE NEXT AVAILABLE TDCT ENTRY OFFSET PTR
 522 002342  020467  004026'                CMP     R4,TDCMAX       ;TDCT OVERFLOW?
 523 002346  103404                         BLO     21$             ;NO -- FINISHED
 524 002350  004767  000546                JSR     PC,TDCERR       ;YES - REPORT CONDITION AND EXIT
 525 002354  000167  000624                JMP     EXIT
 526 002360  000207                  21$:   RTS     PC
```

```
                                        .SBTTL  GVMASK & MATCHT SUBROUTINES

                                ;
                                ;   SUBROUTINE TO STEP VECTORS. ALL NIBBLES OF PREVIOUS VECTORS HAVE BEEN
                                ;   PROCESSED
                                ;
002362  010301                  GVMASK: MOV     R3,R1           ;R1=ADDRESS OF NODE FOR VECTOR STEPPING
002364  011300                          MOV     (R3),R0         ;R0=VECTOR LENGTH
002366  012703  001016'                  MOV     #VMASK,R3       ;R3=ADDRESS TO GENERATE NEW NODE
002372  005002                          CLR     R2              ;R2=NEW VECTOR LENGTH
002374  012123                          MOV     (R1)+,(R3)+     ;TRANSFER VECTOR LENGTHS
002376  012123                          MOV     (R1)+,(R3)+     ;TRANSFER NIBBLE NUMBER
002400  012104                  1$:     MOV     (R1)+,R4        ;R4=OFFSET OF E-MATRIX ENTRY
002402  062704  000002                  ADD     #2,R4
002406  005764  002016'                  TST     EMC(R4)         ;IS E-MATRIX ENTRY A MATCH STATE?
002412  100003                          BPL     2$              ;NO -- THEN STEP VECTOR
002414  004767  000030                  JSR     PC,MATCHT       ;YES - THEN GENERATE MATCH STATE IN TDCT
002420  000402                          BR      3$
002422  010423                  2$:     MOV     R4,(R3)+        ;TRANSFER VECTOR
002424  005202                          INC     R2              ;KEEP COUNT OF VECTORS TRANSFERRED
002426  077014                  3$:     SOB     R0,1$           ;REPEAT FOR ALL VECTORS
002430  005702                          TST     R2              ;WERE ANY VECTORS TRANSFERRED?
002432  001404                          BEQ     4$              ;NO -- FLAG CONDITION
002434  012703  001016'                  MOV     #VMASK,R3       ;RESET TO BEGINNING OF NODE
002440  010213                          MOV     R2,(R3)         ;TRANSFER VECTOR LENGTH TO NEW NODE
002442  000207                          RTS     PC
002444  000261                  4$:     SEC                     ;FLAG CONDITION
002446  000207                          RTS     PC
                                ;
                                ;   SUBROUTINE TO GENERATE MATCH STATE IN TDCT
                                ;
002450                          MATCHT: SAVE    R0,R2,R3
002456  016446  002016'                  MOV     EMC(R4),-(SP)   ;STORE MATCH CODE ON STACK
002462  042716  100000                  BIC     #100000,(SP)    ;REMOVE MATCH CODE INDICATOR
002466  016703  004020'                  MOV     FSAOFF,R3       ;R3=ADDRESS OF TDCT ENTRY BEING PROCESSED
002472  062703  004034'                  ADD     #TDCT,R3
002476  022700  000001                  CMP     #1,R0           ;IS THIS THE ONLY VECTOR?
002502  001010                          BNE     2$              ;NO -- MUST GENERATE 2 STATES
002504  005702                          TST     R2
002506  001006                          BNE     2$              ;NO --
002510  012623                          MOV     (SP)+,(R3)+     ;YES - GENERATE SINGLE MATCH STATE
002512  012723  000001                  MOV     #SUISTA,(R3)+   ;ENTER MATCH CODE AND NEXT ADDRESS = SUISIDE
002516  012723  164000                  MOV     #ST$MAT!TBIT,(R3)+;DEFINE AS NORMAL TERM MATCH STATE
002522  000425                          BR      3$
                                ;
002524  016704  004024'          2$:     MOV     TDCNXT,R4       ;R4=NEXT AVAILABLE TDCT STATE
002530  010467  004020'                  MOV     R4,FSAOFF       ;THIS NEW STATE IS ALSO TO BE THE CURRENT
002534  062704  004034'                  ADD     #TDCT,R4
002540  020403                          CMP     R4,R3           ;ARE WE APPENDING TO TDCT?
002542  001010                          BNE     21$             ;NO
002544  005267  004022'                  INC     NXTADD          ;ADJUST FOR NEW STATE
002550  062767  000006  004024'          ADD     #6,TDCNXT
002556  062767  000006  004020'          ADD     #6,FSAOFF
002564  012623                  21$:    MOV     (SP)+,(R3)+     ;GENERATE MATCH STATE, ENTER MATCH CODE
002566  016723  004022'                  MOV     NXTADD,(R3)+    ;ENTER NEXT ADDRESS = OLD STATE (NEW LOCATION)
002572  012723  164000                  MOV     #ST$MAT!TBIT,(R3)+;DEFINE AS NORMAL TERM MATCH STATE
                                ;
002576                          3$:     RESTOR  R0,R2,R3
```

```
    585 002604  000207                          RTS     PC
```

```
                                        .SBTTL  WRITE EMX TO DISK
                                ;
                                ;   THE PURPOSE OF THIS SUBROUTINE IS TO OPEN THE TDCT-C FILE, WRITE
                                ;   THE TERM DETECTOR CONTROL TABLE TO DISK AND CLOSE THE FILE.
                                ;
002606                          WRTTDC::
                                ;
                                ;   OPEN TDCT FILE
                                ;
002606  012700  000330'             MOV     #TDCFDB,R0              ;R0->FDB
002612  012701  000004              MOV     #FN.FSC,R1              ;R1->FILE NUMBER
002616  004767  000000G             JSR     PC,BLDNFL               ;BUILD FILE NAME BLOCK
002622                              FDAT$R  ,,,,#TDCBLK
002630                              OFNB$W                          ;OPEN FILE
002642  103447                      BCS     6$
002644  016702  000022'             MOV     BAT.NO,R2               ;FIND FDSC ENTRY IN MCOM
002650  016202  000000G             MOV     BSTPTR(R2),R2
002654  062702  000222              ADD     #B.FFSC,R2
002660  016722  000432'             MOV     TDCFDB+F.FNB+N.FID,(R2)+        ;TRANSFER FDSC
002664  016722  000434'             MOV     TDCFDB+F.FNB+N.FID+2,(R2)+
002670  016722  000450'             MOV     TDCFDB+F.FNB+N.FVER,(R2)+
002674  012722  000004              MOV     #FN.FSC,(R2)+
                                ;
                                ;   DETERMINE BLOCK LENGTH OF TDC AND INSERT IN TDCLGT
                                ;
002700                          2$:
002700  005002                      CLR     R2
002702  016703  004024'             MOV     TDCNXT,R3               ;R3 = BYTE LENGTH OF TDCT
002706  071227  004000              DIV     #N.BUFB,R2              ;R2 = QUOTIENT
                                                                    ;R3 = REMAINDER
002712  005703                      TST     R3                      ;TEST REMAINDER
002714  003401                      BLE     3$                      ;BRANCH IF ZERO
002716  005202                      INC     R2
002720  020227  000002          3$: CMP     R2,#TDCOVR              ;TDCT TOO BIG?
002724  003016                      BGT     6$                      ;YES
002726                          4$: WRITE$                          ;WRITE TDCT
002732  103413                      BCS     6$
002734                              WAIT$
002740  062767  004000  000416'     ADD     #N.BUFB,TDCFDB+F.BKVB+2
002746  077211                      SOB     R2,4$                   ;   ALL OF IT
002750                              CLOSE$  #TDCFDB
002760  000207                      RTS     PC
002762                          6$: CALL    FCSERR
002766  000167  000212              JMP     EXIT
```

```
                                        .SBTTL  ERROR HANDLING ROUTINE
                                        .NLIST  MEB
                                ;
                                ;   DIRECTIVE ERROR
                                ;
002772  011667  001010'         DIRERR::MOV     (SP),PAR1               ;PC AT DIRECTIVE ERROR
002776  016767  000000G 001012'         MOV     $DSW,PAR2               ;DSW
003004                                  MOUT$S  #MSG1,#PAR1
003024  005726                         TST     (SP)+
003026  000167  000152                 JMP     EXIT                    ;EXIT
                                ;
                                ;   FCS ERROR
                                ;
003032  011667  001010'         FCSERR::MOV     (SP),PAR1               ;PC
003036  116001  000052                 MOVB    F.ERR(R0),R1            ;ERROR CODE
003042  010167  001012'                 MOV     R1,PAR2
003046                                  MOUT$S  #MSG2,#PAR1
003066  005726                         TST     (SP)+                   ;RESTORE STACK
003070  000167  000110                 JMP     EXIT
                                ;
                                ;   ILLEGAL SEARCH CHARACTER
                                ;
003074  011667  001010'         ERROR2: MOV     (SP),PAR1
003100                                  MOUT$S  #MSG5,#PAR1
003120                                  EXIT    ERROR2
                                ;
                                ;   TDCT OVERFLOW
                                ;
003122  011667  001010'         TDCERR: MOV     (SP),PAR1
003126                                  MOUT$S  #MSG6,#PAR1
003146                                  EXIT    TDCERR
                                ;
                                ;   MORE CWP'S RECEIVED THAN DESIGNED FOR
                                ;
003150  011367  001010'         LIMIT:  MOV     (R3),PAR1       ;VECTORS RECEIVED
003154  012767  000125  001012'         MOV     #VEC3MX,PAR2    ;DESIGN LIMIT
003162                                  MOUT$S  #MSG4,#PAR1
003202  000207                         RTS     PC
                                ;
003204                          EXIT:   MOUT$S  #MSG3
                                ;
                                ;   CLOSE ALL FILES
                                ;
003222                          EXIT1:  CLOSE$  #EMCFDB
003232                                  CLOSE$  #TDCFDB
003242  012767  177777  000022'         MOV     #-1,BAT.NO
003250                                  SDAT$C  QTS,BAT.NO,,CODE
003256                                  RSUM$C  QTS,CODE
003264                                  EXIT$S
```

```
                                        .PSECT  DATA
                                ;
                                ;   ERROR MESSAGES ARE PRINTED ON TI BY MO:...
                                ;   STRING DESCRIPTORS
                                ;
000470  000040                MSG1:   .WORD   LN1E-LN1                ;LENGTH OF FORMAT STRING
000472  000520'                        .WORD   LN1                     ;ADDS OF FORMAT STRING
000474  000040                MSG2:   .WORD   LN2E-LN2
000476  000560'                        .WORD   LN2
000500  000014                MSG3:   .WORD   LN3E-LN3
000502  000620'                        .WORD   LN3
000504  000067                MSG4:   .WORD   LN4E-LN4
000506  000634'                        .WORD   LN4
000510  000033                MSG5:   .WORD   LN5E-LN5
000512  000723'                        .WORD   LN5
000514  000031                MSG6:   .WORD   LN6E-LN6
000516  000756'                        .WORD   LN6
                                ;
                                ;   FORMAT STRINGS
                                ;
000520      104     111     122 LN1:    .ASCIZ  /DIR. ERROR, PC = %10, DSW = %1D/
000560                          LN1E:
000560      106     103     123 LN2:    .ASCIZ  /FCS  ERROR, PC = %10, ERR = %1D/
000620                          LN2E:
000620      121     124     063 LN3:    .ASCIZ  /QT3 EXITING/
000634                          LN3E:
000634      052     052     040 LN4:    .ASCIZ  /** WARNING--# CWP'S (%1D) EXCEED DESIGN LIMIT (%1D)!!!/
000723                          LN4E:
000723      132     105     122 LN5:    .ASCIZ  /ZERO LENGTH EMC.  PC = %10/
000756                          LN5E:
000756      124     104     103 LN6:    .ASCIZ  /TDCT OVERFLOW.  PC = %10/
001007                          LN6E:
                                        .EVEN
                                ;
                                ;   PARAMETERS
                                ;
001010  000000                  PAR1:   .WORD   0                       ;PC
001012  000000                  PAR2::  .WORD   0                       ;DSW OR F.ERR
                                ;
                                ;   MO:... LUN ASSIGNMENT
                                ;
001014                          .MOLUN::.BLKW   1
```

```
                                        .SBTTL  E-MATRIX BUFFERS

                                ;
                                ;   VMASK BUFFER (IT OVERLAYS EMCLGT AND VI)
                                ;
001016                          VMASK:
                                ;
                                ;   VI
                                ;
001016  000000                  EMCLGT: .WORD   0                       ;BLOCK LENGTH OF EMC
001020  000000                  VILGT:  .WORD   0                       ;NO. OF ENTRIES IN VI
001022                          VI:     .BLKW   VI3MSZ-2
                                ;
                                ;   EMC
                                ;
002016                          EMC:    .BLKW   EMCMSZ
```

```
                                        .SBTTL  TERM DETECTOR CONTROL TABLE BUFFER

                                ;
                                ;   TERM DETECTOR CONTROL TABLE BUFFER
                                ;
000001                          SUISTA=1                                ;SUICIDE STATE ADDRESS
                                ;
                                ;
004016  000014                  CHKOFF: .WORD   14              ;SAME AS FSAOFF INITIALLY
004020  000014                  FSAOFF: .WORD   14                      ;OFFSET OF CURRENT TDCT STATE
004022  000002                  NXTADD: .WORD   2                       ;NEXT AVAILABLE STATE ADDRESS
004024  000014                  TDCNXT: .WORD   14                      ;NEXT AVAILABLE OFFSET IN TDCT
                                ;
                                ;
004026  007770                  TDCMAX: .WORD   TDCEND-TDCT-4           ;LARGEST AVAILABLE TDCT OFFSET
                                ;
                                ;
                                ;
004030     065     103          TDCHDR: .ASCII  /SC/                    ;TDCT HEADER
004032  000000                          .WORD   0
                                ;
                                ;   TDCT STATE WORD 0
                                ;
004034  000001                  TDCT:   .WORD   1                       ;NEW ELSE DEF, ADD'S
004036  000002                          .WORD   2                       ;NEXT STATE ADD'S
004040  130000                          .WORD   130000                  ;STATE : CHANGE ELSE DEF,
                                ;
                                ;   TDCT STATE WORD 1 (ELSE DEF, STATE)
                                ;
004042  000000                          .WORD   0                       ;NO INDEX BITS SET, MUST FAIL
004044  000001                          .WORD   1                       ;NEXT STATE = THIS ONE
004046  040000                          .WORD   40000                   ;INDEX STATE ON CHAR CODE
                                ;
                                ;   THE REST OF THE TDCT IS FILLED IN DURING TRANSLATION
                                ;
004050                                  .BLKW   TDCMSZ-8.
014030                          TDCEND:         ;END OF TDCT
000002                          TDCOVR= TDCBLK/4                ;SIZE IN BLOCKS
                                ;
                                ;   NODE DESCRIPTION
                                ;
000000                          N.LGT=0                 ;VECTOR LENGTH
000002                          N.NIB=N.LGT+WORD1       ;NIBBLE POSITION FOR NODE
000004                          N.VEC=N.NIB+WORD1       ;START OF VECTORS
                                ;
014030  014032'                 FREEQ:  .WORD NPOOL     ;PT TO 1ST NODE
                                ;
                                ;       NODE POOL REGION
                                ;
014032                          NPOOL:  .BLKW   NP3MSZ
015302                          NPOOLE: .BLKW   NP3OSZ  ;END OF POOL, START OF POOL OVERFLOW AREA
000000'                                 .END    START
```

```
ASTKMX= 000400 G
AVELGT= 000025 G
BAT.NO  000022R     015
BITVAL= 000000
BIT0  = 000001
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BLDEFL= ****** GX
BLDNFL= ****** GX
BLOCK = 001000
BSTPTR= ****** GX
BS.CLS= 000002
BS.DBU= 000004
BS.INA= 000000
BS.OPN= 000001
BS.SRC= 000003
BYTE0 = 000000
BYTE1 = 000001
BYTE2 = 000002
BYTE3 = 000003
BYTE4 = 000004
BYTE5 = 000005
BYTE6 = 000006
BYTE7 = 000007
BYTE8 = 000010
BYTE9 = 000011
BYTVAL= 000012
B.BSTA  000054     010
B.CNTX  000046     010
B.COUO  000060     010
B.FEMA  000132     010
B.FEMB  000142     010
B.FEMC  000152     010
B.FFSA  000202     010
B.FFSB  000212     010
B.FFSC  000222     010
B.FMHR  000172     010
B.FQLS  000162     010
B.FSAZ  000100     010
B.FSBZ  000102     010
B.FSCZ  000104     010
B.HBLK  000120     010
B.HDOC  000114     010
B.HRLP  000126     010
B.HRLR  000122     010
B.HRLW  000124     010
B.NMBR  000052     010
B.NORY  000232     010
B.QLSZ  000106     010
B.QMAP  000234     010
B.QSPL  000316     010
B.QTTM  000076     010
B.QUQP  000056     010
B.SFDB  000010     010
B.SIZE  000772     010
B.SNDP  000012     010
B.SSQ   000004     010
B.SSQF  000050     010
B.STAT  000044     010
B.STTE  000053     010
B.UDOC  000110     010
CBIT  = 010000 G
CF.B0 = 000070
CF.B2 = 000067
CF.B4 = 000066
CF.B6 = 000065
CF.DR0= 000064
CF.DR1= 000063
CHKOFF  004016R     015
CPIXMK= 177770 G
CPIXSZ= 000010 G
CURNIB  000054R     015
DBSLEN= 000116
DELTIM  000000RG    014
DH.BF0  000002     005
DH.BF1  000004     005
DH.CTL  000000     005
DH.DMC  000010     005
DH.FLG  000006     005
DIRERR  002772RG    017
DN.DCK  000000     013
DN.NTP  000004     013
DN.NXT  000006     013
DN.ROT  000002     013
DN.SIZ  000010     013
EFN.2 = 000002
EIXCNT= 000010 G
ELSBIT= 010000 G
ELSCLS= 000007 G
ELSMSK= 177761 G
EMACLS= 000003 G
EMACUT= 001700 G
EMAMSZ= 016000 G
EMBCLS= 000004 G
EMBCUT= 001130 G
EMBMSZ= 004400 G
EMC     002016R     015
EMCFDB  000170RG    015
EMCLGT  001016R     015
EMCMSZ= 001000 G
EMXCHF= 020000 G
EMXDTF= 010000 G
EMXEXF= 000001 G
EMXFDC= 002000 G
EMXMCD= 000002 G
EMXMCF= 001000 G
EMXMSZ= 016000 G
EMXMTV= 040000 G
EMXNB1= 000002 G
EMXNB2= 000004 G
EMXNFD= 000400 G
EMXNSQ= 100000 G
EMXNVD= 001000 G
EMXSZF= 002000 G
EMXTRL= 010000 G
EMXVDC= 004000 G
EMXVVV= 000001 G
EMXZNF= 004000 G
EMX0VD= 000200 G
ENCNT = 000004 G
EOBIT = 010000 G
ERROR2  003074R     017
EXIT    003204R     017
EXIT1   003222R     017
FCSERR  003032RG    017
FD.FID  000000     003
FD.FNB  000006     003
FD.FVR  000004     003
FD.LEN  000010     003
FD.RWM= ****** GX
FLIXMK= 177740 G
FLIXSZ= 000040 G
FLUCLS= 000001 G
FLUCUT= 000016 G
FMEPSZ= 000400 G
FMIXSZ= 000001 G
FMNPSZ= 002260 G
FNPOSZ= 000400 G
FNPSZ = 005734 G
FN.DBR  000026     011
FN.DBS  000022     011
FN.DHR  000040     011
FN.EMA  000012     011
FN.EMB  000014     011
FN.EMC  000016     011
FN.FSA  000000     011
FN.FSB  000002     011
FN.FSC  000004     011
FN.LGO  000034     011
FN.LGU  000036     011
FN.MFO  000024     011
FN.MHR  000010     011
FN.NMB  000044     011
FN.QLS  000006     011
FN.QRY  000020     011
FN.SF0  000030     011
FN.SF1  000032     011
FN.SHD  000042     011
FO.RD = ****** GX
FO.WRT= ****** GX
FREEQ   014030R     015
FSAOFF  004020R     015
F.ACTL= 000076
F.ALOC= 000040
F.BBFS= 000062
F.BDB = 000070
F.BGBC= 000057
F.BKDN= 000026
F.BKDS= 000020
F.BKEF= 000050
F.BKPI= 000051
F.BKST= 000024
F.BKVB= 000064
F.CHR = 000075
F.CNTG= 000034
F.DFNB= 000046
F.DSPT= 000044
F.DVNM= 000134
F.EFBK= 000010     017
F.EFN = 000050     017
F.EOBB= 000032     017
F.ERR = 000052     017
F.FACC= 000043     003
F.FFBY= 000014     003
F.FNAM= 000110     003
F.FNB = 000102     003
F.FTYP= 000116
F.FVER= 000120
F.HIBK= 000004
F.LUN = 000042
F.MBCT= 000054
F.MBCI= 000055
F.MBFG= 000056
F.NRBD= 000024
F.NREC= 000030
F.OVBS= 000030
F.RACC= 000016     011
F.RATT= 000001     011
F.RCNM= 000034     011
F.RCTL= 000017     011
F.RSIZ= 000002     011
F.RTYP= 000000     011
F.SEQN= 000100     011
F.SPDV= 000072     011
F.SPUN= 000074     011
F.STBK= 000036     011
F.UNIT= 000136     011
F.URBD= 000020     011
F.VBN = 000064     011
F.VBSZ= 000060     011
GTIMI   000220RG    014     011
GVMASK  002362R     017     011
G.TICP= 000016     011
G.TICT= 000014     011
G.TIDA= 000004     011
G.TIHR= 000006
G.TIMI= 000010
G.TIMO= 000002
G.TISC= 000012
G.TIYR= 000000
HOUR.1= 000020
HOUR.2= 000006
HSTKMX= 000100 G
INDCNV  000130R     015
INDEX   001102R     017
IOST    000000R     015
IO.RVB= ****** GX
IWBIT = 004000 G
JMPBIT= 004000 G
LIMIT   003150R     017
LN1     000520R     015
LN1E    000560R     015
LN2     000560R     015
LN2E    000620R     015
LN3     000620R     015
LN3E    000634R     015
LN4     000634R     015
LN4E    000723R     015
LN5     000723R     015
LN5E    000756R     015
LN6     000756R     015
LN6E    001007R     015
LOGCLS= 000002 G
LUNFIL= 000004
LUN2  = 000002
L$STAT= 000006 G
M     = 000062
MATCHT  002450R     017
MIN.1 = 000022
MIN.2 = 000010
MODL  = 000045
MSGOUT= ****** GX
MSG1    000470R     015
MSG2    000474R     015
MSG3    000500R     015
MSG4    000504R     015
MSG5    000510R     015
MSG6    000514R     015
MSK.12  000710R     017
N     = 000002
NDBCLS= 000006 G
NIBCHK  000314R     017
NIBCP   000064R     015
NIBMSK= 177741
NIXCNT= 000020 G
NNMASK= 177760 G
NPECNT= 000144 G
NPEVSZ= 000043 G
NPOOL   014032R     015
NPOOLE  015302R     015
NP1MSZ= 010000 G
NP2MSZ= 036000 G
```

```
NP2OSZ= 000153 G      SD.FSA= 000010 G    003 SS.STT  000000      004 TAEBIT= 020000 G      WN.NTP  000004      012
NP3MSZ= 000524 G      SD.NPS= 000016 G    003 START   000000R     017 TBIT  = 004000 G      WN.NXT  000006      012
NP3OSZ= 000125 G      SD.SEC= 000012 G    003 STISQ   002120R     017 TDABLK= 000004 G      WN.ROT  000002      012
NXTADD  004022R   015 SD.TIC= 000014 G    003 STJSQ   002070R     017 TDABMX= 007764 G      WN.SIZ  000010      012
NXTNIB  000056R   015 SECBUF  000206RG    014 ST$CNG= 120000 G          TDAMAD= 007760 G      WN.SRC  000000      012
N.BFAC= 000004        SEC.1 = 000024            ST$INR= 060000 G          TDBBLK= 000003 G      WN.TYP  000001      012
N.BHGH= 000006        SEC.2 = 000012            ST$INX= 040000 G          TDBCLS= 000005 G      WORD0 = 000000
N.BTCH= 000004        SEG1  = 000000 G          ST$JSQ= 100000 G          TDBMAD= 007775 G      WORD1 = 000002
N.BUFB= 004000        SEG2  = 000002 G          ST$MAT= 160000 G          TDBOSZ= 000074 G      WORD2 = 000004
N.BUFW= 002000        SEG3  = 000004 G          ST$SEQ= 140000 G          TDCBLK= 000010 G      WORD3 = 000006
N.DID = 000024        SEG4  = 000006            ST.ASZ  000020      006 TDCEND  014030R   015 WORD4 = 000010
N.DVNM= 000032        SEQCNT  000124R     015 ST.BSZ  000024      006 TDCERR  003122R   017 WORD5 = 000012
N.FID = 000000        SEQPAD  001536R     017 ST.BTC  000000      006 TDCFDB  000330R   015 WORD6 = 000014
N.FNAM= 000006        SEQST   002172R     017 ST.CSZ  000030      006 TDCHDR  004030R   015 WORD7 = 000016
N.FOS = 000764        SEQVAL  000060R     015 ST.HRL  000010      006 TDCMAX  004026R   015 WORD8 = 000020
N.FTYP= 000014        SR.ARE  000114      002 ST.LEN  000044      006 TDCMSZ= 004000 G          WORD9 = 000022
N.FVER= 000016        SR.ARS  000106      002 ST.QRY  000002      006 TDCNXT  004024R   015 WRDVAL= 000024
N.LGT = 000000        SR.DAY  000010      002 ST.OSZ  000034      006 TDCOVR= 000002            WRTTDC  002606RG    017
N.NEXT= 000022        SR.DLT  000014      002 ST.SCH  000040      006 TDCT    004034R   015 XBATCH= 000013
N.NIB = 000002        SR.ECB  000047      002 ST.UHL  000004      006 TIC.1 = 000026            XDBLOA= 000004
N.PKSZ= 000020        SR.ECH  000046      002 ST.XLT  000014      006 TIC.2 = 000014            XDBPRO= 000012
N.PKTS= 000043        SR.ECL  000050      002 SUISTA= 000001          TRMCUT= 000040 G          XDMCIN= 000006
N.QURY= 000031        SR.FIB  000012      002 SU.DBU= 000004          TSTKMX= 000400 G          XFOSMR= 000007
N.STAT= 000020        SR.GRE  000100      002 SU.DON= 000006          TXTBIT= 010000 G          XGTSRE= 000014
N.SUNT= 000002        SR.GRS  000072      002 SU.IDL= 000000          T.JSBY= 000000 G          XHITSK= 000011
N.UNIT= 000034        SR.LEN  000122      002 SU.LOD= 000001          T.MATC= 000000 G          XHLMER= 000002
N.VEC = 000004        SR.LIN  000066      002 SU.SRC= 000002          T.NBAS= 000002 G          XHOTSK= 000010
OLDNDE  000126R   015 SR.LIP  000062      002 SU.SRR= 000005          T.NDEF= 000000 G          XMSCHE= 000000
PARBUF  000004R   015 SR.MON  000006      002 SU.XPD= 000003          T.SBYi= 000000 G          XQTS  = 000003
PAR$$$= 000027        SR.NDC  000042      002 S.BFHD= 000020          T.SBY2= 000001 G          XQT0  = 000001
PAR1    001010R   015 SR.NDS  000036      002 S.DABA= 000006          T.SBY3= 000002 G          XSULOA= 000005
PAR2    001012RG  015 SR.NIN  000030      002 S.DAEF= 000010          T.STAD= 000002 G          $DSW  = ****** GX
PSTKMX= 000024 G      SR.NIP  000022      002 S.DATN= 000002          T.TRAN= 000000 G          $$$   = 000042R     020
QE.RO1= 000144        SR.SDB  000032      002 S.FATT= 000016          T.TYPW= 000004 G          $$$OST= 000006
QNDCNT= 001000 G      SR.SRC  000002      002 S.FDB = 000140          VEC1MX= 000375 G          $$$T1 = 000003
QRYSZ = 002000 G      SR.SUN  000000      002 S.FNAM= 000006          VEC2MX= 000375 G          .CLOSE= ****** G
QTS     000016R   015 SR.TWS  000056      002 S.FNB = 000036          VEC3MX= 000125 G          .FSRCB= ****** G
QT3.1   000236R   017 SR.WSL  000052      002 S.FNBW= 000017          VI      001022R   015 .MOLUN  001014RG    015
QT3.3   000774R   017 SR.YR   000004      002 S.FNTY= 000004          VIBCUT= 000036 G          .OPFNB= ****** G
Q.FDSC  000004    007 SR.1IN  000024      002 S.FTYP= 000002          VILGT   001020R   015 .WAIT = ****** G
Q.NQBK  000000    007 SR.1IP  000016      002 S.HRL = 000240          VI0MSZ= 000400 G          .WRITE= ****** G
Q.NUHL  000002    007 SSBIT = 004000 G          S.NFEN= 000020          VI1MSZ= 000400 G          .XQIO = ****** GX
Q.SIZE  000014    007 SS.FID  000002      004 S1BIT = 001000 G          VI2MSZ= 000400 G          ...PC1= 000330R     015
R.SUTN= 000002        SS.FNB  000010      004 S2BIT = 002000 G          VI3MSZ= 000400 G          ...PC2= 000470R     015
R.VDBA= 000006        SS.FVR  000006      004 S3BIT = 004000 G          VMASK   001016R   015 ...TPC= 000020
R.VDTN= 000002        SS.LEN  000012      004

. ABS.  000000  000
        000000  001
SRCOFF  000122  002
FDSCOF  000010  003
SUSOFF  000012  004
DHROFF  000012  005
STTOFF  000044  006
QSPLOF  000014  007
BSTOFF  000772  010
FNOFFS  000044  011
```

SYMBOL TABLE


```
WNODOF  000010     012
DNODOF  000010     013
TRNOFF  000240     014
DATA    015554     015
$$FSR1  000000     016
CODE    003272     017
$DPB$$  000050     020
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  8000 WORDS  (32 PAGES)
DYNAMIC MEMORY:  9140 WORDS  ( 35 PAGES)
ELAPSED TIME:  00:01:22
QT3,QT3/-SP/NL:BEX:ME=[20,1]P,M,T,QTSIZE,QT3
```

```
TASK    NAME    : QT3
PARTITION NAME  : HSTSPR
IDENTIFICATION  : 0736
TASK  UIC       : [20,3]
STACK    LIMITS: 000236 001235 001000 00512.
PRG XFR ADDRESS: 002126
TOTAL ADDRESS WINDOWS: 3.
TASK  IMAGE  SIZE   : 5440. WORDS
TASK ADDRESS LIMITS: 000000 025113
R-W DISK BLK LIMITS: 000002 000027 000026 00022.

*** ROOT SEGMENT: MSGOUT


R/W MEM  LIMITS: 000000 025113 025114 10828.
DISK BLK LIMITS: 000002 000027 000026 00022.


MEMORY ALLOCATION SYNOPSIS:

SECTION                                          TITLE   IDENT   FILE
-------                                          -----   -----   ----
. BLK.  :(RW,I,LCL,REL,CON) 001236 000670 00440.
BSTOFF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000. FSA             QT3.OBJ;1
CODE    :(RW,I,LCL,REL,CON) 002126 003272 01722.
                          002126 003272 01722. FSA             QT3.OBJ;1
DATA    :(RW,I,LCL,REL,CON) 005420 015554 07020.
                          005420 015554 07020. FSA             QT3.OBJ;1
DHROFF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000. FSA             QT3.OBJ;1
DNODOF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000. FSA             QT3.OBJ;1
FDSCOF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000. FSA             QT3.OBJ;1
FNOFFS:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000. FSA             QT3.OBJ;1
MSGOUT:(RW,I,LCL,REL,CON) 023174 001214 00652.
                          023174 001214 00652. MESSAG          MSGOUT.OBJ;1
QSPLOF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000. FSA             QT3.OBJ;1
SRCOFF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000. FSA             QT3.OBJ;1
STTOFF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000. FSA             QT3.OBJ;1
SUSOFF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000. FSA             QT3.OBJ;1
TRNOFF:(RW,I,LCL,REL,CON) 024410 000240 00160.
                          024410 000240 00160. FSA             QT3.OBJ;1
WNODOF:(RW,I,LCL,ABS,CON) 000000 000000 00000.
                          000000 000000 00000. FSA             QT3.OBJ;1
$DPB$$:(RW,I,LCL,REL,CON) 024650 000050 00040.
                          024650 000050 00040. FSA             QT3.OBJ;1
$$FSR1:(RW,D,GBL,REL,OVR) 024720 000000 00000.
                          024720 000000 00000. FSA             QT3.OBJ;1
```

```
$$FSR2:(RW,D,GBL,REL,CON) 024720 000104 00068.
$$RESL:(RW,I,LCL,REL,CON) 025024 000066 00054.
$$RESM:(RW,I,LCL,REL,CON) 140000 015600 07040.


GLOBAL SYMBOLS:

ASTKMX 000400    EMCMSZ 001000    EMXZNF 004000    LOGCLS 000002    SD.NPS 000016    TBIT  004000    T.SBY2 000001
AVELGT 000025    EMXCHF 020000    EMX0VD 000200    L$STAT 000006    SD.SEC 000012    TDABLK 000004    T.SBY3 000002
CBIT   010000    EMXDTF 010000    ENCNT  000004    MSGOUT 023376-R  SD.TIC 000014    TDABMX 007764    T.STAD 000002
CPIXMK 177770    EMXEXF 000001    EOBIT  010000    NDBCLS 000006    SECBUF 024616-R  TDAMAD 007760    T.TRAN 000000
CPIXSZ 000010    EMXFDC 002000    FCSERR 005160-R  NIXCNT 000020    SEG1   000000    TDBBLK 000003    T.TYPW 000004
DELTIM 024410-R  EMXMCD 000002    FLIXMK 177740    NNMASK 177760    SEG2   000002    TDBCLS 000005    VEC1MX 000375
DIRERR 005120-R  EMXMCF 001000    FLIXSZ 000040    NPECNT 000144    SEG3   000004    TDBMAD 007775    VEC2MX 000375
EIXCNT 000010    EMXMSZ 016000    FLUCLS 000001    NPEVSZ 000043    SSBIT  004000    TDBOSZ 000074    VEC3MX 000125
ELSBIT 010000    EMXMTV 040000    FLUCUT 000016    NP1MSZ 010000    ST$CNG 120000    TDCBLK 000010    VIBCUT 000036
ELSCLS 000007    EMXNB1 000002    FMEPSZ 000400    NP2MSZ 036000    ST$INR 060000    TDCMSZ 004000    VI0MSZ 000400
ELSMSK 177761    EMXNB2 000004    FMIXSZ 000001    NP2OSZ 000153    ST$INX 040000    TRMCUT 000040    VI1MSZ 000400
EMACLS 000003    EMXNFD 000400    FMNPSZ 002260    NP3MSZ 000524    ST$JSQ 100000    TSTKMX 000400    VI2MSZ 000400
EMACUT 001700    EMXNSQ 100000    FNPOSZ 000400    NP3OSZ 000125    ST$MAT 160000    TXTBIT 010000    VI3MSZ 000400
EMAMSZ 016000    EMXNVD 001000    FNPSZ  005734    PAR2   006432-R  ST$SEQ 140000    T.JSBY 000000    WRTTDC 004734-R
EMBCLS 000004    EMXSZF 002000    GTIM1  024630-R  PSTKMX 000024    S1BIT  001000    T.MATC 000000    $EDMSG 023624-R
EMBCUT 001130    EMXTRL 010000    HSTKMX 000100    QNDCNT 001000    S2BIT  002000    T.NBAS 000002    .MOLUN 006434-R
EMBMSZ 004400    EMXVDC 004000    IWBIT  004000    QRYSZ  002000    S3BIT  004000    T.NDEF 000000
EMCFDB 005610-R  EMXVVV 000001    JMPBIT 004000    SD.FSA 000010    TAEBIT 020000    T.SBY1 000000


*** TASK BUILDER STATISTICS:

    TOTAL WORK FILE REFERENCES: 17521.
    WORK  FILE  READS: 0.
    WORK  FILE WRITES: 0.
    SIZE OF CORE POOL: 6634. WORDS (25. PAGES)
    SIZE OF WORK FILE: 2816. WORDS (11. PAGES)

    ELAPSED TIME:00:00:11
```

```
        .NLIST.....
        .SBTTL. E-MATRIX EQUATES.
;
        .PSECT. TRNOFF.
;
;  E-MATRIX (FSA-A &FSA-B) OFFSETS AND BIT DEFINITIONS.
;
;=======BYTE0    ;CHARACTER CODE (BINARY VALUE)
;=======BYTE0    ;DOCUMENT TYPE  (BINARY VALUE)
;=======BYTE0    ;ZONE           (BINARY VALUE)
;=======BYTE0    ;SUBZONE        (BINARY VALUE)
;=======WORD0    ;# VECTORS THAT EMX EXPANDS TO (BYTES)
;=======WORDO    ;VECTOR EMX MERGES TO (OFFSET FROM EMX START)
EMXMCD==WORD1   ;MATCH CODE     (BINARY VALUE)
EMXNB1==WORD1   ;NIBBLE 1       (BIT MATRIX)
EMXNB2==WORD2   ;NIBBLE 2       (BIT MATRIX)
;
;  EMX FLAG DEFINITIONS IN CONTROL WORD
;
EMXNSQ==BIT15   ;NON-SEQUENCIAL EMX FLOW
EMXEXF==BIT0    ;EMX FLOW EXPANDS
        ;FOLLOWING VALID ONLY IF EMXNSQ=0
        ;NOTE: ALL IN NEXT GROUP ARE MUTUALLY EXCLUSIVE EXCEPT
        ;       EMXMTV WHICH CAN OCCUR WITH ALL EXCEPT EMXMCF
EMXMTV==BIT14   ;THIS EMX ENTRY REPRESENTS MULTIPLE VALUES
EMXCHF==BIT13   ;THIS EMX ENTRY IS A CHARACTER
EMXDTF==BIT12   ;THIS EMX ENTRY IS A DOCUMENT TYPE
EMXZNF==BIT11   ;THIS EMX ENTRY IS A ZONE
EMXSZF==BIT10   ;THIS EMX ENTRY IS A SUBZONE
EMXMCF==BIT9    ;THIS EMX ENTRY IS A MATCH CODE
        ;FOLLOWING VALID ONLY IF EMXMTV=1
EMXVDC==BIT11   ;VLDC  (VARIABLE LENGTH DON'T CARE)
EMXFDC==BIT10   ;FLDC  (FIXED LENGTH DON'T CARE)
EMXNVD==BIT9    ;NVLDC (NUMERICAL VLDC)
EMXNFD==BIT8    ;NFLDC (NUMERICAL FLDC)
EMX0VD==BIT7    ;ZVLDC (ZERO VLDC)
        ;FOLLOWING VALID ONLY IF EMXVDC=1
EMXTRL==BIT12   ;THIS ENTRY IS FOLLOWED BY AN INTERWORD
EMXVVV==BIT0    ;SECESSIVE VLDC
.PAGE
        .SBTTL. TERM DETECTOR CONTROL TABLE EQUATES
;
;
;       THE FOLLOWING ARE THE SEGMENT, WORD, AND BYTE
;       OFFSETS AND THE BIT DEFINITIONS FOR THE
;       TERM DETECTOR STATE WORDS.
;
;
;       TDCT CONTROL TABLE OFFSETS
;
SEG1 ==.0               ;RELATIVE TDCT OFFSET TO SEGMENT 1
SEG2 ==.SEG1+WORD1      ;    TO SEG2
SEG3 ==.SEG2+WORD1      ;    TO SEG3
;
;
L$STAT  ==      WORD3   ;BYTE LENGTH OF TDCT STATE WORD
;
;       WORD OFFSETS IN TDCT
;
T.TRAN ==.SEG1          ;TRANSITION BITS SEGMENT (INDEX)
T.NBAS ==.SEG2          ;NEXT BASE ADDRESS SEGMENT (INDEX)
```

```
T.TYPW ==  [illegible]          ;TYPE CODE SEGMENT (ALL)
T.NDEF ==  SEG1                 ;NEXT DEFAULT ADDRESS (CHANGE DEF)
T.STAD ==  SEG2                 ;NEXT STATE ADDRESS (CHANGE DEF, JMP/SEQ, MATCH)
T.MATC ==  SEG1                 ;MATCH CODE SEGMENT (MATCH)
;
;       BYTE OFFSETS IN TDCT
;
T.JSBY ==  0                    ;JMP/SEQ MATCH BYTE OFFSET
T.SBY1 ==  0                    ;SEQ MATCH BYTE OFFSET 1
T.SBY2 ==  T.SBY1+BYTE1         ;SEQ MATCH BYTE OFFSET 2
T.SBY3 ==  T.SBY2+BYTE1         ;SEQ MATCH BYTE OFFSET 3
;
;       BIT POSITIONS IN TDCT SEGMENT 3 (T.TYPWD)
;
TAEBIT  == BIT13                ;INDEX RETRY BIT (INDEX)
EOBIT ==  BIT12                 ;ELSE OVERRIDE BIT (INDEX)
SSBIT ==  BIT11                 ;SINGLE SUCCESS BIT (INDEX)
JMPBIT ==  BIT11                ;JUMP IMMEDIATELY BIT (JMP/SEQ)
TXTBIT  == BIT12                ;TEXT INPUT
ELSBIT ==  BIT12                ;ELSE DEFAULT BIT (CHANGE DEF)
IWBIT ==  BIT11                 ;INTERWORD DEFAULT BIT (CHANGE DEF)
CBIT ==  BIT12                  ;POTENTIAL CWP BIT (MATCH)
TBIT ==  BIT11                  ;TERM BIT (MATCH)
S1BIT ==  BIT9                  ;END SEQ BIT FOR MATCH BYTE1 (SEQUENTIAL)
S2BIT ==  BIT10                 ;END SEQ BIT FOR MATCH BYTE2 (SEQUENTIAL)
S3BIT ==  BIT11                 ;END SEQ BIT FOR MATCH BYTE3 (SEQUENTIAL)
;
;   STATE TYPE CODE BITS
;
ST$INX  ==      BIT14                   ;INDEX STATE
ST$INR  ==      BIT13+BIT14             ;INDEX RETRY STATE
ST$JSQ  ==      BIT15                   ;JUMP/SEQUENTIAL
ST$CNG  ==      BIT13+BIT15             ;CHANGE DEFAULT
ST$SEQ  ==      BIT14+BIT15             ;SEQUENTIAL STATE
ST$MAT  ==      BIT13+BIT14+BIT15       ;MATCH REPORT
;
        .PAGE
        .SBTTL  'SEND' BUFFER OFFSETS
;
;   THE FOLLOWING ARE THE OFFSET DEFINITIONS OF THE "SEND"
;   DIRECTIVE BUFFER USED BY THE QUERY TRANSLATOR AND QTS TO
;   EXCHANGE FDSC AND TRANSLATOR PERFORMANCE DATA.
;
;FD.FID ==       0               ;FDSC ENTRIES (FID)
;FD.FVR ==       FD.FID+WORD2
;FD.FNB ==       FD.FVR+WORD1
SD.FSA  ==      FD.LEN                  ;NO. OF FSA STATES
SD.SEC  ==      SD.FSA+WORD1            ;ELAPSED SECONDS
SD.TIC  ==      SD.SEC+WORD1            ;   "    TICS
SD.NPS  ==      SD.TIC+WORD1            ;NODE POOL SIZE
        .PAGE
        .SBTTL  SUBROUTINE DELTIM
;
;   SUBROUTINE TO CALCULATE ELAPSED TIME FROM TIME MEASUREMENT VALUES
;   OBTAINED THROUGH THE GTIM$ DIRECTIVE.
;
;   ON RETURN,
;       SECBUF = ELAPSED SECONDS
;       SECBUF+2 = ELAPSED TICS
;
DELTIM::SAVE    R1,R2,R3
        MOV     #SECBUF,R1
```

```
        MOV     #SECBUF,R2              ;R2->ELAPSED SECONDS
        CMP     TIC.2(R1),TIC.1(R1)     ;COMPARE TIC VALUES
        BGE     1$
        ADD     #60.,TIC.2(R1)
        INC     SEC.1(R1)               ;CARRY 1
1$:     SUB     TIC.1(R1),TIC.2(R1)
        MOV     TIC.2(R1),2(R2)         ;ELAPSED TICS
        CMP     SEC.2(R1),SEC.1(R1)
        BGE     2$
        ADD     #60.,SEC.2(R1)
        INC     MIN.1(R1)               ;CARRY ONE
2$:     SUB     SEC.1(R1),SEC.2(R1)
        MOV     SEC.2(R1),(R2)          ;R2->ELAPSED SECONDS
        CMP     MIN.2(R1),MIN.1(R1)
        BGE     3$
        ADD     #60.,MIN.2(R1)
        INC     HOUR.1(R1)
3$:     MOV     MIN.2(R1),R3
        SUB     MIN.1(R1),R3            ;R3 == ELAPSED MINUTES
        MUL     #60.,R3                 ;R1 == SECONDS
        ADD     R3,(R2)
        CMP     HOUR.2(R1),HOUR.1(R1)
        BLT     4$
        MOV     HOUR.2(R1),R3
        SUB     HOUR.1(R1),R3           ;R3 == ELAPSED HOURS
        MUL     #3600.,R3               ;CONVER TO SECONDS
        ADD     R3,(R2)
4$:
        RESTOR  R1,R2,R3
        EXIT    DELTIM
;
;   TEMPORARY BUFFER FOR ELAPSED SECONDS AND TICKS
;
SECBUF::.BLKW   5.
GTIM1:: .BLKW   8.
;
;   OFFSET DEFINITIONS FOR GET TIME PARAMETERS BUFFER
;
HOUR.2  =       WORD3
MIN.2   =       HOUR.2+WORD1
SEC.2   =       MIN.2+WORD1
TIC.2   =       SEC.2+WORD1
HOUR.1  =       TIC.2+WORD2
MIN.1   =       HOUR.1+WORD1
SEC.1   =       MIN.1+WORD1
TIC.1   =       SEC.1+WORD1
        .LIST
```

```
;...
; MACRO TO PRINT BUFFER
; CREATES SPOOL FILE RECORDS
;
.MACRO  PRT,START,END
        MOV     R4,-(SP)
        MOV     START,R4
        MOV     R4,LOCAT
        MOV     END,ENDLOC
        JSR     PC,PRINT
        MOV     (SP)+,R4
.ENDM
.MACRO  PRTH,START,END
        MOV     R4,-(SP)
        MOV     START,R4
        MOV     R4,LOCAT
        MOV     END,ENDLOC
        JSR     PC,PRNTH
        MOV     (SP)+,R4
.ENDM
;
; MACRO TO TEST CHARACTER SIGNIFICANCE
; PARAMETERS ARE CONDITION,TRUE,AND FALSE
;       CONDITION=THE TEST CONDITION
;       TRUE     =PATH TO TAKE IF CONDITION TRUE
;       FALSE    =PATH TO TAKE IF CONDITION FALSE
; ANY PARAMETER CAN BE PROCEEDED BY "!" WHICH CAUSES THAT
;  PARAMETER TO BE A SUBROUTINE CALL
;
        .NLIST
.MACRO  CHECK,COND,TRUE,FALSE
        .IF B COND
          .ERROR        "CONDITION CANNOT BE OMITTED"
        .ENDC
        .IRPC X,<COND>
          .IF IDN <X>,<!>
            JSR PC,0'COND               ;CHECK !COND,?,?
            .IF NB TRUE
              .IRPC Y,<TRUE>
                .IF IDN <Y>,<!>
                  .IF B FALSE
                    BCS .+6             ;CHECK !COND,!TRUE
                    JSR PC,0'TRUE
                    .MEXIT
                  .ENDC
                  .IRPC Z,<FALSE>
                    .IF IDN <Z>,<!>
                      BCS .+10          ;CHECK !COND,!TRUE,!FALSE
                      JSR PC,0'TRUE
                      BR .+6
                      JSR PC,0'FALSE
                      .MEXIT
                    .ENDC
                    BCS FALSE           ;CHECK !COND,!TRUE,FALSE
                    JSR PC,0'TRUE
                    .MEXIT
                  .ENDM
                  .MEXIT
                .ENDC
                .IF B FALSE
                  BCC TRUE              ;CHECK !COND,TRUE
```

```
        .MEXIT
      .ENDC
      .IRPC Z,<FALSE>
        .IF IDN <Z>,<!>
          BCC TRUE              ;CHECK !COND,TRUE,!FALSE
          JSR PC,@#FALSE
          .MEXIT
        .ENDC
        BCC TRUE                ;CHECK !COND,TRUE,FALSE
        BCS FALSE
        .MEXIT
      .ENDM
      .MEXIT
    .ENDM
    .MEXIT
  .ENDC
  .IF B FALSE
    .MEXIT                      ;CHECK !COND
  .ENDC
  .IRPC Z,<FALSE>
    .IF IDN <Z>,<!>
      BCC .+6                   ;CHECK !COND,,!FALSE
      JSR PC,@#FALSE
      .MEXIT
    .ENDC
    BCS FALSE                   ;CHECK !COND,,FALSE
    .MEXIT
  .ENDM
  .MEXIT
.ENDC
BIT #COND,R0                    ;CHECK COND,?,?
.IF NB TRUE
  .IRPC Y,<TRUE>
    .IF IDN <Y>,<!>
      .IF B FALSE
        BEQ .+6                 ;CHECK COND,!TRUE
        JSR PC,@#TRUE
        .MEXIT
      .ENDC
      .IRPC Z,<FALSE>
        .IF IDN <Z>,<!>
          BNE .+10              ;CHECK COND,!TRUE,!FALSE
          JSR PC,@#TRUE
          BR .+6
          JSR PC,@#FALSE
          .MEXIT
        .ENDC
        BEQ FALSE               ;CHECK COND,!TRUE,FALSE
        JSR PC,@#TRUE
        .MEXIT
      .ENDM
      .MEXIT
    .ENDC
    .IF B FALSE
      BNE TRUE                  ;CHECK COND,TRUE
      .MEXIT
    .ENDC
    .IRPC Z,<FALSE>
      .IF IDN <Z>,<!>
        BNE TRUE                ;CHECK COND,TRUE,!FALSE
        JSR PC,@#FALSE
        .MEXIT
```

```
            .ENDC...
            BNE·TRUE·                   ;CHECK·COND,TRUE,FALSE·
            BEQ·FALSE·
            .MEXIT·
          .ENDM·
          .MEXIT
        .ENDM·
        .MEXIT·
      .ENDC·
      .IF·B·FALSE·
        .MEXIT·                         ;CHECK·COND·
      .ENDC·
      .IRPC·Z,<FALSE>
        .IF·IDN·<Z>,<!>
          BNE·.+6                       ;CHECK·COND,,!FALSE·
          JSR·PC,@'FALSE·
          .MEXIT
        .ENDC·
        BEQ·FALSE·                      ;CHECK·COND,,FALSE·
        .MEXIT
      .ENDM·
      .MEXIT·
    .ENDM·
.ENDM·  CHECK·
      .LIST·····
```