HEADER BLOCK

```
000000                *Lori Cook ..... ...........August 1981
000000                *This contains the floppy-disk control routines
000000                *for GEMINI (HP-87). These routines were taken
000000                *from the Capricorn mass storage rom and modified.
000000                *Extras have been incorporated such as the READ#
000000                *and PRINT# string arrays and the primitive
000000                *printer driver.
000000                *Release scheduled week August 1981.
000000                      ABS
000000                      LOC 60000      ROM START ADDRESS
060000                INTRSC  DAD 177500      INTERRUPTING S.C. ADDRESS
060000                ***!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!***
060000                *   HEADER BLOCK                              *
060000                ***********************************************
060000                MYROM#  EQU 320
060000                *Steps for creating EROMs and ROMS:
060000                * 1. Change MYROM# to 320,2'S compliment to 60.
060000                * 2. Enable JZR CKSMOK code, disable JMP CKSMOK
060000                IOROM#  EQU 300
060000 320            BASE    VAL MYROM#          ROM NUMBER
060001 060                    OCT 60              2'S COMPLEMENT ROM NUMB
060002 012 140                DEF RUNTIM          ADDRESS OF LIST OF RUN
060004                *                           ROUTINES' ADDRESSES
060004 215 140                DEF TOKENS          ROM RESERVED WORD
060006                *                           (TOKEN) TABLE
060006 130 140                DEF PARSES          PARSE ROUTINES' TABLE
060010                *                           OF ADDRESSES
060010 040 167                DEF ERMESS          ERROR MESSAGE TABLE
060012 231 167        RUNTIM  DEF INITIT          ADDRESS OF INITIALIZ-
060014                *                           ATION ROUTINE
```

```
==========================================================================
        RUN TIME ADDRESSES
060122 062 ?05            DEF PRSTR.        36  PRINT STRING
060124            RDATOK  EQU 37D           37. READ ARRAY
060124 106 160            DEF RDARR.
060126            PRA#TK  EQU 38D           38 PRINT STRING ARRAY
060126 370 160            DEF PRARR$
060130            RDA#TK  EQU 39D           39.READ STRING ARRAY
060130 312 160    PARSES  DEF RDARR$
060132            *NOTE: PRINT tokens must be even to force a comma
060132            *       (not a semi) when listed.
060132            *-----------------------------------------------------
```

```
        MASS STORAGE ROM TOKEN LIST
060215                    ******************************************
060215                    *   FOLLOWING IS A LIST OF TOKENS DEFINED     *
060215                    *   BY THIS ROM.  EACH ASCII REPRESENTATION *
060215                    *   HAS THE HIGH BIT (BIT 7) OF ITS LAST      *
060215                    *   CHARACTER SET  (E.G. OF ASSIGN, N (116) *
060215                    *   BECOMES (316)).                           *
060215                    *                                             *
060215                    *   TOKENS AFTER THE 377 (END OF TOKEN        *
060215                    *   LIST) ARE USED FOR DECOMPILING BUT ARE   *
060215                    *   NOT PARSED BY THE SYSTEM                  *
060215                    ******************************************
060215 101 123 123 TOKENS   ASP 7,ASSIGN#        TOKEN 1
060220 111 107 116
060223 243
060224 103 101 324          ASP 3,CAT            TOKEN 2
060227 103 110 105          ASP 15D,CHECK READ OFF#  TOKEN 3
060232 103 113 040
060235 122 105
060237 101 104 040
060242 117 106 106
060245 243
060246 103 110 105          ASC 10D,CHECK READ    TOKEN 4
060251 103 113 040
060254 122 105
060256 101 104
060260 243                  OCT 243              BIASED #
060261 200                  OCT 200              TOKEN 5
060262 103 117 120          ASP 4,COPY            TOKEN 6
060265 331
060266 103 122 105          ASP 6,CREATE         TOKEN 7
060271 101 124 305
060274 111 116 111          ASP 10D,INITIALIZE TOKEN 8
060277 124 111 101
060302 114 111
060304 132 305
060306 103 110 101          ASP 5,CHAIN          TOKEN 9
060311 111 316
060313 114 117 101          ASP 7,LOADBIN        TOKEN 10
060316 104 102 111
060321 316
060322 114 117 101          ASP 4,LOAD           TOKEN 11
060325 304
060326 115 101 123          ASP 15D,MASS STORAGE IS   TOKEN 12
060331 123 040 123
060334 124 117
060336 122 101 107
060341 105 040 111
060344 323
060345 120 122 111          ASP 6,PRINT#         TOKEN 13
060350 116 124 243
060353 123 105 103          ASP 6,SECURE         TOKEN 14
060356 125 122 305
060361 125 116 123          ASP 8,UNSECURE       TOKEN 15
```

```
060364 105 103 125
060367 122 305
060371 120 125 122         ASP 5,PURGE         TOKEN 16
060374 107 305
060376 122 105 101         ASP 5,READ#         TOKEN 17
060401 104 243
060403 122 105 116         ASP 6,RENAME        TOKEN 18
060406 101 115 305
060411 123 124 117         ASP 30,STOREBIN     TOKEN 19
060414 122 105 102
060417 111 316
060421 123 124 117         ASP 5,STORE         TOKEN 20
060424 122 305
060426 120 101 103         ASP 4,PACK          TOKEN 21
060431 313
060432 126 117 114         ASP 6,VOLUME        TOKEN 22
060435 125 115 305
060440 107 114 117         ASP 5,GLOAD         #24
060443 101 304
060445 107 123 124         ASP 6,GSTORE        #25
060450 117 122 305
060453 105 122 122         ASP 5,ERROM         #26
060456 117 315
060460 105 122 122         ASP 5,ERRSC         #27
060463 123 303
060465 124 131 320         ASP 3,TYP           #28
060470             ISTOK   EQU 28D             'IS' PARSES THIS ROM
060470 040 111 323         ASP 3, IS           TOKEN #28D
060473             *       OCT 240             BIASED BLANK (8/24/81 RO)
060473 377                 OCT 377             #29:END OF TOKENS MARKER
060474             *
060474             * FOLLOWING ARE TOKENS WHICH LIST AND EXECUTE,
060474             * BUT DON'T PARSE (IN THIS ROM)
060474             *
060474             RTOTOK  EQU 30D             ROM TOKEN # 30 DECIMAL
060474 040 124 117         ASC 3, TO
060477 240                 OCT 240           BIASED BLANK
060500             *---------- INITIALIZE PARSE ROUTINE
060500             *--
060500             INITIA  BSS 0
060500 143 006 344         PUBD R43,+R6        SAVE 'INITIALIZE' TOKEN
060503 316 377 377         JSB =ROMJSB
060506 377 377             DEF STREX+          CK FOR ANY PARAMS
060510 000                 OCT 0
060511 371 040             JEZ GOINDO          JIF NO PARAMS
060513 114 310 054         CMB R14,=COMMA      CK FOR COMMA
060516 366 033             JNZ GOINDO          NO COMMA,NO MORE PARAMS
060520 316 137 161         JSB =GETSTRING      GET A STRING
060523 114 310 054         CMB R14,=COMMA      CK FOR COMMA
060526 366 023             JNZ GOINDO          NO MORE PARAMS
060530 316 123 161         JSB =GETNUM         GET A NUMBER
060533 114 310 054         CMB R14,=COMMA      CK FOR COMMA
060536 366 013             JNZ GOINDO          NO MORE PARAMS
```

```
                        PASS: STORAGE ROM TOKEN LAST
060540 750 041                  JMP  CHCK+          ELSE GETNUM
060542                  *---------- END OF INITIALIZE PARSING
060542
060542
060542
060542
060542
060542
060542
060542
060542                  *---------- CAT PARSE ROUTINE
060542                  *--
060542                  *--
060542                  CAT     BSS  0
060542                  PACK    BSS  0
060542 143 006 344              PUBD R43,+R6        PUSH ROM TOKEN
060545 316 377 377              JSB  =ROMJSB
060550 377 377                  DEF  STREX+         GET A STRING
060552 000                      OCT  0
060553 760 971          GOINDO  JMP  IND5
060555                  *---------- END OF CAT PARSE ROUTINE
```

```
            CREATE strex,numex[,numex]
     060555                 *----------- CREATE PARSE ROUTINE
     060555                 *--
     060555                 CREATE  BSS 0
     060555 143 006 244             PUBD R43,+R6         SAVE 'CREATE' TOKEN
     060560 316 377 377             JSB =ROMJSB
     060563 377 377                 DEF G$N+NN           GET STRING & 1 OR 2
     060565                 *                            NUMERICS
     060565 000                     OCT 0
     060566 760 052                 JMP FURPAR           RESTORE TOKEN&GEN.CODE
     060570                 *----------- END OF CREATE PARSE ROUTINE
     060570
     060570
     060570
     060570
     060570
     060570
     060570
     060570
     060570                 *----------- PARSE <COMMAND> strex
     060570                 *-- WHERE <COMMAND> IS ONE OF:
     060570                 *--    LOAD,LOADBIN,STORE,STOREBIN,
     060570                 *--    CHAIN,MASS STORAGE IS,
     060570                 *--    GSTORE,GLOAD
     060570                 *--
     060570                 CHAIN   BSS 0
     060570                 GLOAD   BSS 0
     060570                 GSTORE  BSS 0
     060570                 LOAD    BSS 0
     060570                 LOADBI  BSS 0
     060570                 STORE   BSS 0
     060570                 STOREB  BSS 0
     060570                 MASSST  BSS 0
     060570 143 006 244             PUBD R43,+R6
     060573 316 137 161             JSB =GETSTR          GET A STRING
     060576 760 046                 JMP INDS
     060600                 *-----------  END OF <COMMAND> strex PARSING
```

```
060600                 *----------  CHECKR PARSE ROUTINE
060600                 *--
060600                 *--
060600                 CHKOF    BSS 0
060600                 CHECKR   BSS 0
060600 143 006 344              PUBD R43,+R6        SAVE ROM TOKEN
060603 316 123 161     CHCK+    JSB =GETNUM
060606 760 036                  JMP IND6
060610                 *----------  END OF CHECKR PARSE ROUTINE
060610
060610
060610
060610
060610
060610
060610
060610
060610                 *---------- SECURE (UNSECURE) PARSE ROUTINE
060610                 *--
060610                 *--
060610                 SECURE   BSS 0
060610                 UNSECU   BSS 0
060610 143 006 344              PUBD R43,+R6        PUSH ROM TOKEN
060613 316 137 161              JSB =GETSTR         GET A STRING
060616 316 156 161              JSB =GETCOM         GET A COMMA
060621 316 137 161              JSB =GETSTR         GET A STRING
060624 316 156 161              JSB =GETCOM         GET A COMMA
060627 760 352                  JMP CHCK+           RESTORE TOK & GEN. CODE
060631                 *---------- END OF SECURE PARSING
060631
060631
060631
060631
060631
060631
060631
060631
060631                 *---------- PURGE PARSE ROUTINE
060631                 *--
060631                 PURGE    BSS 0
060631 143 006 344              PUBD R43,+R6        SAVE 'PURGE' TOKEN
060634 316 377 377              JSB =ROMJSB
060637 377 377                  DEF GET$N?          GET STRING [& 1 NUM]
060641 000                      OCT 0
060642 153 270 316     PURPAR   LDBI R53,=PTR2+      UNDO AUTO TOKEN GEN.
060645 377
060646 760 160         IND6     JMP IND7            RESTORE TOK & GEN CODE
060650                 *---------- END OF PURGE PARSE ROUTINE
```

```
                *>>>        READ# PARSE ROUTINE
    060650                  *---------- READ# PARSE ROUTINE
    060650                  *--
231 060650                  READ#   BSS 0
232 060650                  *PARSE READ# ROUTINE
233 060650 143 006 344              PUBO R43,+R6        SAVE READ# TOKEN
234 060653 316 377 377              JSB =ROMJSB
235 060656 377 377                  DEF G1OR2N          GET 1 OR 2 PARAMS
236 060660 100                      OCT 0
237 060661 153 270 316              LDBI R53,=PTR2+     UNDO G- AUTO CODE GEN
237 060664 377
239 060665 006 342                  POBO R53,-R6        RESTORE TOKEN INFO
239 060667 316 014 142              JSB =PSHRON         GENERATE READ# CODE
240 060672 114 310 047              CMB R14,=SEMI       SEMICOLON SEPARATOR?
241 060675 367 013                  JZR GOTSEM          JIF YES
242 060677 134 310 002 DEMND2       CMB R34,=2          NO SEMI, NEED 2 PARAMS
243 060702 766 002                  JNZ ER22D           BUFFER/RECORD ERROR
244 060704 760 130                  JMP GORRTN          LEAVE ROM
245 060706 316 170 161 ER22D         JSB =SYSER+        SYS ERROR IN PARSING
246 060711 133                      OCT 91D            MISSING PARAM
2   060712                  *----------
 38 060712                  *SEMI SEEN,GET <READ#-LIST>
299 060712 316 377 377 GOTSEM   JSB =ROMJSB
300 060715 377 377              DEF PUSH1A              PUSH SEMI AND SCAN
301 060717 100                  OCT 0
302 060720                  *SEMI IS PUSHED FOR DECOMPILER ONLY
303 060720 316 343 141 RD#LST   JSB =RD#ITM             GET <READ#-ITEM>
304 060723 371 012              JEZ ERR23D               PARAMETER LIST ERROR
    060725 316 377 377          JSB =ROMJSB
3.2 060730 377 377              DEF GETCM?             MORE TO BE FOUND?
307 060732 100                  OCT 0
308 060733 370 763              JEN RD#LST              MORE IN READLIST
309 060735 760 977              JMP GORRTN              LEAVE ROM
310 060737                  *----------
311 060737 316 170 161 ERR23D    JSB =SYSER+
312 060742 134                  OCT 92D                 SYNTAX ERROR
313 060743                  *----------
314 060743                  RD#ITM  BSS 0               PARSE <READ#-ITEM>
315 060743                  *<FORMAL STRING ARRAY>?
3   060743 316 173 142              JSB =STRARR         IS IT A STRING ARRY?
 17 060746                  *                           USED TO LET A JMP REACH
318 060746 153 250 047              LDB R53,=RDA$TK
319 060751 370 041                  JEN PUSHRD
320 060753                  *<FORMAL-ARRAY>?
321 060753 316 377 377              JSB =ROMJSB
322 060756 377 377                  DEF FORMAR          FIND FORMAL ARRAY
323 060760 100                      OCT 0
324 060761 153 250 045              LDB R53,=RDATOK     GET ARRAY RD TOK
325 060764 370 026                  JEN PUSHRD          GEN CODE IF ARRAY FOUND
326 060766                  *<STRING-REFERENCE>?
327 060766 316 377 377              JSB =ROMJSB
328 060771 377 377                  DEF STRREF          STRING REFERENCE?
329 060773 100                      OCT 0
330 060774 153 250 041              LDB R53,=RDSTOK     GET STRING RD TOK
```

```
>>>>>>          READ# PARSE ROUTINE
   060777 370 013                JEN PUSHRD          GEN CODE IF STR FOUND
332 061001                  *<NUMERIC-VALUE>?
333 061001 316 377 377            JSB =RCMJSB
334 061004 377 377                DEF REFNUM          NUMERIC?
335 061006 000                    OCT 0
336 061007 371 016                JEZ REDRTN          JIF NOT NUMERIC
337 061011 153 250 037            LDB R53,=RDNTOK     GET NUMERIC RD TOK
338 061014                  PUSHRD  BSS 0             GENERATE ROM TOK CODE
339 061014 176              PSHROM  DRP 76            THE 'CALL-ROM' TOKEN
340 061015 251                    OCT 251
341 061016 320                    VAL MYROM#          LDM R76,=320,370
342 061017 370                    OCT 370
343 061020 273 315 377            STMI R76,=PTR2-     CALL MASS STO. ROM CODE
344 061023 153 272 315            STBI R53,=PTR2-     MASS STO. TOKEN
344 061026 377
345 061027 236              REDRTN  RTN
346 061030                  *----------  END OF READ# PARSING
347 061030
348 061030
349 061030
 50 061030
351 061030 153 006 342 IND?       POBD R53,-R6        RESTORE ROM TOKEN
352 061033 316 014 142 PUSHIT     JSB =PSHROM         GEN ROM CODE
353 061036 104 251 377 GORTN      GTO ROMRTN          LEAVE ROM
353 061041 377
```

```
                        PRINT# PARSE ROUTINE
    061042                      *---------- PRINT# PARSE ROUTINE
    061042                      *--
357 061042                      PRINT#  BSS 0
358 061042                      *
359 061042                      * PARSE PRINT#
360 061042                      *
361 061042 143 056 344                  PUBD R43,+R6        PUSH ROM TOKEN
362 061045 316 377 377                  JSB =ROMJSB
363 061050 377 377                      DEF G1OR2N          GET 1 OR 2 NUMERICS
364 061052 000                          OCT 0
365 061053 153 270 316                  LDBI R53,=PTR2+     UNDO AUTO. CODE GEN
365 061056 377
366 061057 056 342                      POBD R53,-R6        RESTORE ROM TOKEN
367 061061 316 014 142                  JSB =PSHROM         GENERATE PRINT# CODE
368 061064 014 310 047                  CMB R14,=SEMI       SEMI SEPARATOR?
369 061067 266 206                      JNZ DEMND2          IF NOT, NEED 2 PARAMS
370 061071 316 377 377                  JSB =ROMJSB
371 061074 377 377                      DEF PUSHIM          PUSH SEMI & SCAN
372 061076 000                          OCT 0
373 061077                      * GET <PRINT#-LIST>
 74 061077 316 121 142 PR#LST   JSB =PR#ITM         GET <PRINT#-ITEM>
375 061102 371 233                      JEZ ERR230          PARAMETER ERROR
376 061104 316 377 377                  JSB =ROMJSB
377 061107 377 377                      DEF GETCM?          MORE TO BE FOUND?
378 061111 000                          OCT 0
379 061112 370 363                      JEN PR#LST          YES, MORE PARAMETERS
380 061114 153 250 043                  LDB R53,=PREOL      GET EOL TOK
    061117 260 312                      JMP PUSHIT         GEN CODE & DONE
382 061121                      *----------
383 061121                      PR#ITM  BSS 0               GETS A <PRINT#-ITEM>
384 061121                      *STRING-ARRAY?
385 061121 316 173 142                  JSB =STRARR         IS IT A STRING ARRAY?
386 061124 153 250 046                  LDB R53,=PRA$TK
387 061127 370 263                      JEN PUSHRD
388 061131                      *ARRAY?
389 061131 316 377 377                  JSB =ROMJSB
390 061134 377 377                      DEF FORMAR          <FORMAL-ARRAY>?
391 061136 000                          OCT 0
392 061137 153 250 040                  LDB R53,=PRATOK     ARRAY PRINT TOKEN
 93 061142 370 250                      JEN PUSHRD          GENERATE CODE
394 061144                      *NUMERIC?
395 061144 316 377 377                  JSB =ROMJSB
396 061147 377 377                      DEF NUMVAL          <NUMERIC-VALUE>?
397 061151 000                          OCT 0
398 061152 153 250 042                  LDB R53,=PRNTOK     NUMERIC PRINT TOKEN
399 061155 370 235                      JEN PUSHRD          GENERATE CODE
400 061157                      *STRING?
401 061157 316 377 377                  JSB =ROMJSB
402 061162 377 377                      DEF STREXP          <STRING-VALUE>?
403 061164 000                          OCT 0
404 061165 153 250 044                  LDB R53,=PRSTOK     STRING PRINT TOKEN
405 061170 370 222                      JEN PUSHRD          GENERATE CODE
406 061172 236                          RTN
```

```
                PRINT# PARSE ROUTINE
061173
061173
061173                  STRARR  BSS 0
061173 235                      CLE
061174 214 310 040              CMB  R14,=40        IS IT A STRING TOKEN
061177 266 371                  RNZ                 RTN NOW IF NOT
061201 175 261 314              LDMD R75,=PTR2      SAVE ADDR FOR LATER
061204 377
061205 316 377 377              JSB =RCMJSB
061210 377 377                  DEF FORM$A          FORMAL ARRAY?
061212 000                      OCT 0
061213 371 355                  REZ                 RETURN IF NO ARRAY
061215 316 377 377              JSB =REPLTK         REPLACE NUMERIC TOKEN
061220 236                      RTN
061221                  *---------- END OF PRINT# PARSING
```

```
***          (RENAME,COPY) strex TO strex
061221                 RENAME  BSS 0
061221                 COPY    BSS 0
424 061221 143 006 344         PUBD R43,+R6        SAVE ROM TOKEN
425 061224 316 137 161         JSB =GETSTR         GET A STRING
426 061227 316 147 161         JSB =GETTO          GET 'TO'
427 061232 316 137 161         JSB =GETSTR         GET 2ND STRING
428 061235 260 035             JMP TO+TOK          OUTPUT TO + COMMAND TOKS
429 061237             *-----------
```

```
                    Parse ASSIGN#, VOLUME
   061237                   *----------  ASSIGN PARSE ROUTINE
   061237                   *--
   061237                   ASSIGN  BSS 0                   PARSE ASSIGN
   061237 143 006 344               PUBD R43,+R6           SAVE ROM TOKEN
   061242 316 123 161               JSB =GETNUM            GET A NUM
   061245 316 147 161               JSB =GETTO             GET 'TO'
   061250 120 310 052               CMB R20,=STAR          IS NEXT CHAR A STAR?
   061253                   *                              I.E. ASSIGN - TO * ?
   061253 367 033                   JZR CLOSF              JIF SO
   061255 316 137 161               JSB =GETSTR            GET A STRING
   061260 114 310 100               CMB R14,=100           @ FOLLOWS ASSIGN# ?
   061263 366 007                   JNZ TO+TOK             NOPE
   061265 116 220                   TSB R16                CALC MODE?
   061267 363 003                   JEV TO+TOK             NOPEd CALC --> ERROR!
   061271                   *DONT ALLOW ANYTHING AFTER ASSIGN# IN CALC MODE!
   061271                   * AS OF JULY 30,1981 ALLOW @ IN CALC MODE
   061271                   *       JSB =DMNDCR            DEMAND CR OR !
   061271 006 342                   POBD R#,-R6            POP TOKEN
   061273 236                       RTN
   061274 153 250 036       TO+TOK  LDB R53,=RTOTOK        GET 'TO' TOK
   061277 316 014 142       IS+TOK  JSB =PSHROM            OUTPUT IT
   061302 006 342                   POBD R#,-R6                COMMAND TOKEN
   061304 104 251 032               GTO PUSHIT           GENERATE CODE
   061307 142
   061310                   *----------
   061310 145 251 052       CLOSF   LDM R45,=52,1,6        UNQUOTED STRING
   061313 001 006
   061315 273 315 377               STMI R45,=PTR2-        OUTPUT THE CODE
   061320 316 377 377               JSB =ROMJSB
   061323 377 377                   DEF SCAN+              ADVANCE PAST *
   061325 100                       OCT 0
   061326 360 344                   JMP TO+TOK
   061330                   *----------
   061330                   *
   061330
   061330
   061330
   061330
   061330
   061330
   061330
   061330
   061330                   VOLUME  BSS 0
   061330 143 006 344               PUBD R43,+R6           SAVE ROM TOKEN
   061333 316 137 161               JSB =GETSTR            GET A STRING
   061336 143 310 034               CMB R43,=ISTOK         FIND IS
   061341 367 003                   JZR VOK                JIF IS FOUND
   061343 316 133 161               JSB =ER91D+            MISSING TO/IS
   061346 316 137 161       VOK     JSB =GETSTR            GET 2ND STRING
   061351 153 250 034               LDB R53,=ISTOK         GET IS TOKEN
   061354 360 321                   JMP IS+TOK
```

```
GMA 03/14/81 ================================================================
ITEM    LOC    OBJECT CODE   SRC=LCGMA2 OBJ=GCMGMA    9/11/1981  3:03 PM  PG 10
================================================================
                    CATALOG [msus/volume label]
 74 061356 126 012 241 GETMSU  LDM R26,R12        HOW MANY PARAMETERS?
475 061361 325 344 203         EBMD R26,=TOS      IF R12-TOS=0 THEN NONE
476 061364 367 013             JZR NOPAR          MUST HAVE NO PARAMS
477 061366 140 250 002 DCD-    LDB R40,=2         ALLOW MSUS ONLY
478 061371 316 366 161 DCDFIL  JSB =DECODE
479 061374 144 263 160 DCD+    STMD R44,=ACTMSU    NEW ACTIVE MSUS
479 061377 207
480 061400 236                 RTN
481 061401 144 261 077 NOPAR   LDMD R44,=DEFMSU    GET DEFAULT MSUS
481 061404 207
482 061405 360 365             JMP DCD+
483 061407 140 222     DCDZER  CLB R40            ANY OLD SPECIFIER
484 061411 360 356             JMP DCDFIL
485 061413                     *-------------------------------------------------
486 061413 241                 OCT 241
487 061414                     MSCAT.  BSS 0
488 061414 024                 ARP 24
489 061415 316 245 161         JSB =MSINHK        RUNTIME INIT
490 061420 316 356 142         JSB =GETMSU        INIT & GET MSUS
491 061423                     ORP !44
492 061423 220                 TSB R44            IS THERE A MSUS ON LINE
493 061424 366 003             JNZ MSOK
494 061426 316 334 161         JSB =ERR20D        "NO MSUS DEVICE"
495 061431                 *DO A CATALOG OF DISC ADDRESSED R45,46,47
496 061431                 *PRINT VOLUME LABEL
497 061431 316 360 146 MSOK    JSB =GETDIR        GET VOL.LAB,1ST DIR.ENTRY
498 061434 114 026 243         STM R14,R26        PRTBUF OFFSET ZERO
499 061437 152 261 377         LDMD R52,=VOLHED   VOLUME HEADING
499 061442 377
500 061443 345                 PUMD R52,+R26      PUSH IT
501 061444 261 377 377         LDMD R52,=VLHED2   2ND HALF OF HEADER
502 061447 345                 PUMD R52,+R26
503 061450 261 127 207         LDMD R52,=SPECIF GET VOL.NAME
504 061453 345                 PUMD R52,+R26      PUSH VOLUME NAME
505 061454 250 015             LDB R52,=15        CARRIAGE RETURN
506 061456 344                 PUBD R52,+R26      OUTPUT IT TOO
507 061457 136 251 023         LDM R36,=19D,0     VOLHED LENGTH
507 061462 000
508 061463 316 313 143         JSB =DRIVER        OUTPUT IT
509 061466                 *MOVE HEADING FROM ROM TO RAM FOR OUTPUT
510 061466 124 251 074         LDM R24,=CATHED    ROM SOURCE
510 061471 144
511 061472 114 026 243         STM  R14,R26       SINK ADDRESS
512 061475 122 251 037         LDM R22,=31D,0     30 BYTES
512 061500 000
513 061501 316 377 377         JSB =MOVUP         MOVE ROM TO RAM
514 061504                 *PRINT HEADING
515 061504 316 307 143         JSB =DRIVE-        OUTPUT BUFFER
516 061507 316 037 147         JSB =RBUF36        RESTORE DIR.PTR
517 061512                 *NOW OUTPUT EACH CAT ENTRY
518 061512                 CATLOP  BSS 0
519 061512 126 261 162         LDMD R26,=SVCWRD   EXIT IF KEY PRESSED
519 061515 200
```

```
 <<>>        CATALOG [msus/volume label]
    061516 762 166             JOD CATDON
 52 061520 014 241             LDM R26,R14        PRTBUF OFFSET ZERO
522 061522              *PRINT OUT TYPE,BYTES/LREC,LRECS
523 061522 316 236 143         JSB =GETTYP        GET FILE TYPE
524 061525 156 223             CLM R56        CLEAR 57 FOR TYPE INDEXING
525 061527 030 240             LDB R56,R30        FILE TYPE
526 061531              *
527 061531              *Summary of disk file types and type-table lookup:
528 061531              *  type=377 --> next available file
529 061531              *  type=0   --> empty (hole) file
530 061531              *  type bit set (msb7 to lsb0)
531 061531              *    2        --> **** extended file type(GRAF,ASCI,etc.)
532 061531              *    3        --> BPGM
533 061531              *    4        --> DATA
534 061531              *    5        --> PROG
535 061531              *
536 061531 310 377             CMB R56,=377       LAST FILE?
537 061533 367 151             JZR CATDON         'TWAS
538 061535 143 261 377         LDMD R43,=BLANKS   BLANKS IF EMPTY
538 061540 377
539 061541 063 243             STM R43,R63       (OR LIST SECURED)
540 061543 156 220             TSB R56            CHECK DATA SECURED
541 061545 362 015             JOD BLDUN          JIF SECURED
542 061547 367 011             JZR NULL           JIF EMPTY
543 061551 143 036 245         LDMD R43,R36       GET THE NAME
544 061554 163 265 005         LDMD R63,X36,SEC1/2 2ND 1/2 NAME
544 061557 000
5   061560 760 002             JMP BLDUN
546 061562              NULL   ORP 156
547 061562 250 100             LDB R56,=100       SET NULL BIT FOR TYPE CHK
548 061564 143 026 345 BLDUN  PUMD R43,+R26      NAME TO PRTBUF
549 061567 163 345             PUMD R63,+R26      2ND HALF NAME
550 061571 142 261 377         LDMD R42,=BLANKS   R42,3:=BLANK
550 061574 377
551 061575 156 206             LRB R56            NULL->40,ALL->2
552 061577 206                 LRB R56            NULL->20,ALL->1
553 061600 763 011             JEV NOTEXT         JIF NOT EXTENDED
554 061602              *For interchange files must create BYTES/LOG.REC
5   061602              *field for CAT 'Bytes' and 'Recs' to be correct
556 061602              *now all files except data files assumed to be
557 061602              *256 bytes per record August 1980 GEMINI MSROM
558 061602 212                 DCB R56            PEAL OFF EXTEND FLAG
559 061603 204                 LLB R56            MAKE MULT OF 4
560 061604 144 056 265         LDMD R44,X56,EXTFIL
560 061607 010 220
                               JMP  DNSH+
561 061611 760 014      NOTEXT JLN DNSHFT
562 061613 375 005
563 061615              *      LRB R#             PROG->4,ALL->0
564 061615              *      LLB R#             PROG->10,ALL->0
565 061615 204                 LLB R#             PROG->20,ALL->0
566 061616 374 002             JLZ DNSHFT         JIF DATA,BPGM,ALL
567 061620 250 014             LDB R#,=14         FORCE PROG TYPE
568 061622              DNSHFT BSS 0
```

```
GENMA  03/14/81=================================================================
ITEM    LOC    OBJECT CODE  SRC=_COMAS OBJ=GENOMA    9/11/1981  3:03 PM  PG 19
================================================================================
>>>>        CATALOG [msus:volume label]                              <<<<<<<<
 59 061622 144 056 265          LDMD R44,X56,CNTTAB  GET TYPE
569 061625 377 377
570 061627 142 026 345 ONSH+    FUMD R42,+R26        TYPE TO PRTBUF
571 061632 136 006 345          FUMD R36,+R6         SAVE DIR PTR
572 061635 156 344              FUBD R56,+R6
573 061637 310 010              CMB  R56,=10         DATA FILE?
574 061641 136                  OPP  36
575 061642 766 006              JNZ  REC256          JIF NOT
576 061644 036 265 036          LDMD R36,X36,D.B/RC BYTES/REC
576 061647 000
577 061650 760 003              JMP  NMASC
578 061652 251 000 001 REC256   LDM  R#,=0,1         BYTES PER REC=256
579 061655 316 033 144 NMASC    JSB =NM2ASC          NUM->ASCII
580 061660 156 006 342          FOBD R56,-R6         RESTORE TYPE FLAG
581 061663 136 343              FOMD R36,-R6         RESTORE DIR PTR
582 061665 316 042 144          JSB =LOGREC          COMPUTE LOG.RECS
583 061670 126 030 241          LDM R26,R30          ASCII GOES HERE
584 061673 316 033 144          JSB =NM2ASC           NUM->ASCII
585 061676 316 307 143          JSB =DRIVE-          OUTPUT BUFFER
586 061701 316 044 147          JSB =NXTENT          GET NXT DIR.ENTRY
587 061704 371 204              JEZ CATLOP
588 061706 236            CATDON  RTN
589 061707                *---------------------------------------------------
590 061707                T.ASCI  EQU 4D              FOR SAVE/GET INTERCHANGE
591 061707                T.GRAF  EQU 14              FOR GLOAD, GSAVE
592 061707                *ALL EXTENTED TYPES ARE THE SAME. THIS MEANS COPY
593 061707                *MUST MAP DIFFERENT TAPE EXTENDED TYPES TO ONE BYTE
594 061707                *OF DISC TYPE WITH ONE BIT EXTENDED.
595 061707                *---------------------------------------------------
596 061707 136 251 037 DRIVE-    LDM R36,=31D,0     SEND OUTPUT
596 061712 000
597 061713                DRIVER  BSS 0               CALL SYSTEM OUTPUT DRIVER
598 061713 126 014 241          LDM R26,R14         PRTBUF OFFSET ZERO
599 061716 316 377 377          JSB =ROMJSB
600 061721 377 377              DEF DISP.           INITIALIZE DRIVER
601 061723 000                  OCT 0
602 061724 316 377 377          JSB =ROMJSB
603 061727 377 377              DEF DRV12.          DO OUTPUT
604 061731 000                  OCT 0
605 061732 316 277 161          JSB =INIT14         DRIVER DESTROYS R14
606 061735 236                  RTN
607 061736                *---------------------------------------------------
608 061736                GETTYP  BSS 0
609 061736                *CONVERTS 2-BYTE INTERCHANGE DISC TYPE TO
610 061736                *1-BYTE DISC TYPE.
611 061736                *THERE ARE 3 SPECIAL CASES:
612 061736                * 1: 0 MEANS PURGED FILE, STAYS 0
613 061736                * 2: 1 MEANS ASCII FILE, BECOMES T.ASCI
614 061736                * 3: -1 MEANS NEXT-AVAIL FILE, STAYS -1
615 061736                *NOTE: -16001 <= INTERCHANGE CAPR TYPES <= -20000 (octal
616 061736                *       i.e. CAPR FILE TYPE - 20000 --> INTERCHANGE CAPR
617 061736 130 036 265          LDMD R30,X36,D.FTYP   GET INTERCHANGE TYPE
617 061741 012 000
```

```
>>>>>       CATALOG [msus/volume label]                              <<<<<<<<
3 061743 316 147 147          JSB =SWAP+          SWAP BYTES
  061746 130 221              TSN R30
620 061750 367 014            JZR GETLST          JIF PURGED FILE
621 061752 311 377 377        CMM R30=377,377     NEXT-AVAIL FILE?
622 061755 367 007            JZR GETLST          JIF YES
623 061757 311 001 000        CMM R30,=1,0        ASCI FILE?
624 061762 766 004            JNZ GETCAP          JIF REGULAR FILE
625 061764 250 004            LDB R30,=T.ASCI     INTERCHNG ASCII --> CAPR ASCI
626 061766 220          GETLST TSB R#             R30
627 061767 236                RTN
628 061770 313 000 040 GETCAP ADM R#,=CVDOFF      INTERCHNG CAPR --> CAPR CAPR
629 061773             CVDOFF DAD 20000           CORVALLIS DIV OFFSET
630 061773 236                RTN
631 061774                    *-------------------------------------------------
632 061774                    PUTTYP BSS 0
633 061774                    *REVERSE OF GETTYP
634 061774 131 222            CLB R31             FOR INTERCHANGE CONVERSION
635 061776 130 220            TSB R30             WHAT KIND CAPR FILE?
636 062000 367 020            JZR PUTLST          JIF PURGED FILE
7 062002 210                  ICB R30             NEXT-AVAIL FILE?
8 062003 367 014              JZR PUTNXT          JIF SO
639 062005 212                DCB R30             RESTORE TYPE
640 062006 310 004            CMB R30,=T.ASCI     ASCII FILE TYPE?
641 062010 367 005            JZR PUTASC          YES
642 062012                    *ITS A REGULAR CAPR FILE TYPE:
643 062012 315 000 040        EBM R30,=CVDOFF     CONVERT CAPR TO INTERCHANGE
644 062015 760 003            JMP PUTLST
) 062017 250 002        PUTASC LDB R#,=2          ASCII + 1
6 062021 213            PUTNXT DCM R#             ASCII-->1,NEXT-AV-->-1
647 062022 316 147 147 PUTLST JSB =SWAP+          SWAP R30,R31
648 062025 130 036 267        STMD R30,X36,D.FTYP READY FOR INTERCHANGE DISC
648 062030 012 000
649 062032 236                RTN
650 062033                    *-------------------------------------------------
651 062033                    NM2ASC BSS 0
652 062033 316 377 377        JSB =RCMJSB         TRANSLATE INTEGER R36
653 062036 377 377            DEF CATCO-          TO ASCII STR ADDR R26
654 062040 000                OCT 0
5 062041 236                  RTN
6 062042                      *-------------------------------------------------
657 062042                    LOGREC BSS 0
658 062042                    *COMPUT LOG REC COUNT
659 062042 146                OPP 46
660 062043 316 165 147        JSB =LDRECS
661 062046 132 251 000        LDM R32,=0,1        BYTES PER REC=256
661 062051 001
662 062052 156 310 010        CMB R56,=10         DATA FILE?
663 062055 766 005            JNZ LOGRE+          JIF NOT
664 062057 132 036 265        LDMO R32,X36,D.B/RC BYTES/REC
664 062062 036 000
665 062064 145 222      LOGRE+ CLB R45            SEE ABOVE
666 062066 134 222            CLB R34
667 062070 316 377 377        JSB =COMLO+         COMPUTE LOG.RECS.
```

```
ARNA 03/14/81 ================================================================================
ITEM    LOC   OBJECT CODE  SRC=LCGMA3 OBJ=GEXGMA   9/11/1981  3:03 PM  PG 20
================================================================================
>>>>>>      CATALOG [msus/volume label]                                    <<<<<<
672 062073 236                  RTN
673 062074              ;-------------------------------------------------------
679 062074              ^VOLHED  ASC 8D,Volume:
671 062074              ^VOL HEADING HAS BEEN MOVED TO THE SYSTEM DUE TO CRUNCH
672 062074 116 141 155 CATHED  ASC 30D,Name          Type  Bytes   Recs
672 062077 145 040 040
672 062102 040 040
673 062104 040 040 040
673 062107 040 124 171
673 062112 160 145
674 062114 040 040 102
674 062117 171 164 145
674 062122 163 040
675 062124 040 040 122
675 062127 145 143 163
676 062132 015                  OCT 15              CARRIAGE RETURN
```

```
                INIT[label[,msus[,entries[,intlv]]]]
062133                  *SECTOR OFFSETS FOR DISC INITIALIZATION
062133                  *VOLUME OFFSETS
062133                  V.ID    EQU 0               DISC ID
062133                  V.LABL  EQU 2               VOLUME LABEL
062133                  V.DBEG  EQU 10D             START OF DIRECTORY
062133                  V.DLEN  EQU 18D             LENGTH OF DIRECTORY
062133                  V.1000   EQU 12D            SYSTEM 3000 CONSTANT !?!
062133                  *DIRECTORY OFFSETS (<=32D)
062133                  D.FTYP  EQU 10D       FILE TYPE IN DIR. ENTRY
062133                  D.RECS  EQU 18D       FILE LENGTH IN SECTORS
062133                  D.TIME  EQU 20D       TIME OF CREATION,99999 FOR GEN
062133                  D.BYTS  EQU 28D       FILE LENGTH IN BYTES
062133                  D.FBEG  EQU 14D       START OF FILE (SECTOR#)
062133                  D.B/RC  EQU 30D       BYTES/LOGICAL RECORD FOR DATA
062133                  D.FLAG  EQU 31D        CAPRICORN,GEMINI=1
062133                  D.VOL   EQU 26D             ENTIRE FILE IN THIS VOLUME
062133                  *
062133 241                      OCT 241
062134                  INITI.  BSS 0
062134 023                      ARP 23
062135 316 245 161              JSB =MSINHK         RUNTIME INIT
062140 144 251 077              LDMD R44,=DEFMSU    DEFAULT IF NO MSUS
062143 207
062144 263 160 207              STMD R44,=ACTMSU
062147                  *PREPARE VOLLAB TEMPLATE W/DEFAULTS
062147 316 377 377              JSB =CLRSEC         ZERO SECTOR
062152 316 037 147              JSB =RBUF36         BUFFER BASE
062155 152 250 200              LDB R52,=200        THE VOLUME ID
062160 036 246                  STBD R52,R36        SINCE V.ID=0
062162 261 377 377              LDMD R52,=BLANKS    DEF VOL.NAME
062165 267 002 000              STMD R52,X36,V.LABL
062170 155 251 000              LDM R55,=0,2,20   DEF DIR. START SECTOR#
062173 002 020
062175                  *NOTE! *this is 2D  in INTERCHANGE format (ie swapped)
062175 267 012 000              STMD R55,X36,V.DBEG
062200 132 251 000              LDM R32,=0,14D      INTCHNG DEF DIRLEN
062203 016
062204 267 022 000              STMD R32,X36,V.DLEN  ...INTO VOLUME LABEL
062207 251 016 000              LDM R32,=14D,0      CAPR FORMAT DEF DIRLEN
062212 006 345                  PUMD R32,+R6        FOR DIR.INIT.LOOP
062214 250 006                  LDB R32,=6          DEFAULT INTERLEAVE FACTOR
062216 345                      PUMD R32,+R6        SAVE FOR FORMAT
062217                  *HOW MANY PARAMS?
062217 012 241                  LDM R32,R12
062221 325 344 203              SBMD R32,=TOS       # BYTES ON STACK
062224 310 023                  CMB R32,=19D        interleave factor?
062226 765 015                  JPS ISINTL          JIF SO
062230 310 013                  CMB R32,=11D        directory sectors?
062232 765 021                  JPS ISENT           JIF SO
062234 310 006                  CMB R32,=6D         msus?
062236 765 036                  JPS ISMSUS          JIF SO
062240 320                      TSB R32             volume label?
062241 766 043                  JNZ VOLLON          JIF SO
```

```
>>>      INIT[label[,msus[,entries[,intlv]]]]
 25 062243 760 101                JMP INIDSK          OTHERWISE NO PARAMS
729 062245                    *
730 062245 316 377 377 ISINTL   JSB =ONEB            GET INTEGER
731 062250 136 006 343          POMD R36,-R5         POP DEFAULT INTERLEAVE
732 062253 146 345              PUMD R46,+R5         SAVE USER SUPPLIED
733 062255 316 377 377 ISENT    JSB =ONEB            GET INTEGER
734 062260 766 003              JNZ ENTSOK
735 062262 316 302 152 INVAL    JSB =ER39D           INVALID PARAM
736 062265 154 006 343 ENTSOK   POMD R54,-R5         REPLACE 2ND PARAM ON STAK)
737 062270 146 054 243          STM R46,R54         REPLACE ENTRIES PARAM
738 062273 154 006 345          PUMD R54,+R5         NOW SWITCHED, PUSH BOTH PARAM
739 062276                    ISMSUS  BSS 0          ALLOW ANY MSUS
740 062276 316 007 143          JSB =DCDZER          DECODE MSUS
741 062301                    *       TSB R44        TAPE OR DISC?
742 062301 766 003              JNZ VOLLON           JIF DISC
743 062303 316 334 161          JSB =ER20D           NO MSUS CONNECTED
744 062306                    *
745 062306                    VOLLON  BSS 0
746 062306 316 304 161          JSB =GETNA!          6-CH NAME BLANKFILLED
747 062311 154 006 343          POMD R54,-R5         54:=ENTRIES,56:=INTLV
748 062314 345                  PUMD R54,+R5         BACK FOR LOOPS
749 062315                    *SWAP R54 ENTRIES INTO R56
750 062315 057 242              STB R54,R57
751 062317 155 056 242          STB R55,R56          SWAP DONE
752 062322 316 037 147          JSB =RBUF36
753 062325 142 310 056          CMB R42,=PERIOD      DOES LABEL START WITH "."
754 062330 767 330              JZR INVAL            JIF YES INVALID PARAM
755 062332 310 072              CMB R42,=COLON       DOES IT START WITH ","
756 062334 767 324              JZR INVAL            JIF YES INVALID PARAM
757 062336 036 267 002          STMD R42,X36,V.LABL SAVE VOLUME LABEL
757 062341 000
758 062342 156 267 022          STMD R56,X36,V.DLEN SAVE DIR LENGTH
758 062345 000
759 062346                    *
760 062346                    INIDSK  BSS 0          INITIALIZE A DISC
761 062346 130 006 343          POMD R30,-R5         POP INTERLEAVE
762 062351 316 221 173          JSB =HLFMT           FORMAT THE DISC (ACT.MSUS)
763 062354 132 223              CLM R32
764 062356 316 032 176          JSB =PUTBUF          OUTPUT VOLUME LABEL
765 062361                    *CREATE,WRITE NULL DIRECTORY
766 062361 316 377 377          JSB =CLRSEC          CLEAR SECTOR
767 062364                    *OUTPUT ONE RECORD OF 0's FOR INTERCHANGE
768 062364 316 170 176          JSB =PUTBU+          THE 2ND SECTOR
769 062367                    *NULL DIRECTORY CREATION
770 062367 140 250 010          LDB R40,=8D          8 ENTRIES/SECTOR COUNT
771 062372 316 037 147          JSB =RBUF36          SECTOR BASE
772 062375 142 250 231          LDB R42,=231         BCD  99
773 062400 143 042 241          LDM R43,R42          PROPIGATE
774 062403 156 223              CLM R56              EACH FILE FLAGGED LAST
775 062405 213                  DCM R56
776 062406 166 251 177          LDM R66,=177,377     AND HAS INFINITE LENGTH
776 062411 377
777 062412                    *Note: ^ this is INTERCHANGE for MAXINT
```

```
                INIT[label[,msus[,entries[,intlv]]]]
062412 156 036 267 ILOP1   STMD R56,X36,D.FTYP MAKE 'LAST' FILE
062415 012 000
062417 142 267 024         STMD R42,X36,D.TIME INIT TIME
062422 000
062423 166                 DPP 56
062424 316 140 147         JSB =STRECS
062427 136 313 040         ADM R36,=32D,0      BUMP DIR PTR
062432 000
062433 140 212             DCB R40             COUNTER
062435 766 353             JNZ ILOP1           DO FOR EACH ENTRY
062437                 *WRITE 'entries' OF THESE EMPTY SECTORS
062437 316 170 176 ILOP2   JSB =PUT3U+         WRITE ONE DIR SECTOR
062442 136 006 343         POMD R36,-R6        RESTORE COUNTER
062445 213                 DCM R36
062446 345                 FUND R36,+R6        AND SAVE IT
062447 766 366             JNZ ILOP2           REPEAT FOR entries SECTORS
062451 343                 POMD R36,-R6        DELETE COUNTER
062452 236                 RTN
```

```
                CHAIN file specifier
062453 241                      OCT 241
062454                  MSCHA.  BSS 0
062454 100                      ARP 0
062455 316 313 161              JSB =TAPD3-         DECODE FILE & CHECK MSUS
062460 316 377 377              JSB =RCMJSB
062463 377 377                  DEF SETTR!          RESET TRACE INFO
062465 100                      OCT 0
062466 155 261 003              LDMD R55,=FWPRGM
062471 200
062472 263 006 200              STMD R55,=FWCURR
062475 316 267 147              JSB =LOAD+           GO LOAD IT
062500 316 260 147              JSB =LDCOMN          CLEAN UP LOAD
062503 316 377 377              JSB =RCMJSB
062506 377 377                  DEF CHAIN+          TAPE CHAIN POST-PROCESSING
062510 100                      OCT 0
062511 236              SECQUI  RTN
```

```
                STOREBIN File specifier                          <<<<<<<<
062512 241                    OCT 241
062513                 MSSTB. BSS 0
062513 001                    ARP 1
062514 316 313 161            JSB =TAPOS-        DECODE FILE & CHECK MSUS
062517 250 010                LDB R#,=BPGMTY
062521 262 271 203            STBD R#,=FILTYP    SET FILE TYPE
062524                 *jan 23, 1980: add STOREBIN security
062524                 *MAY 1981: MULTIPLE BINARIES
062524 316 377 377            JSB  =BINAM        DOES BINARY NAME EXIST?
062527 235                    CLE                FOR SECUR+
062530                        DRP !36
062530 055 243                STM R36,R55        FOR STORB-
062532 100 250 200            LDB R0,=200        STORE-SECURITY
062535 316 377 377            JSB =SECUR+        SYSTEM CK SECURE
062540 370 347                JEN SECQUI         BINARY WAS SECURED!
062542 175 261 310            LDMD R75,=PTR1
062545 377
062546 006 345                PUMD R75,+R6
062550 036 265 004            LDMD R75,X36,BINLEN    LAST WORD BINARY
062553 000
062554 177 222                CLB  R77
062556 175 055 303            ADM  R75,R55           COMPUTE LAST BINARY
062561 316 313 145            JSB =STOR5-
062564 155 006 343            POMD R55,-R6
062567 263 310 377            STMD R55,=PTR1
062572 236                    RTN
062573
062573
```

```
                        STORE file specifier                    <<<<<<<<
062573 141                      OCT 141
062574                  MSSTO.  BSS 0
062574 002                      ARP 2
062575 316 313 161              JSB =TAPOS-             DECODE FILE & CHECK MSUS
062600 100 250 200              LDB R0,=200             TEST STORE PROTECT
062603 316 377 377              JSB =SECUR?             PROG SECURED?
062606 370 301                  JEN SECQUI              JIF SECURED FOR STORE
062610 250 040                  LDB R#,=CAPRTY          SET CAPR TYPE FILE
062612 262 271 203              STBD R#,=FILTYP         STORE TYPE
062615                  *---------------------------------------------------
062615 316 377 377               JSB =ROMJSB            LOAD PREPROCESSING
062620 377 377                   DEF INIPGM
062622 000                       OCT 0
062623 316 377 377               JSB =ST240+
062626 316 377 377               JSB =ROMJSB
062631 377 377                   DEF FXDIR              DEALLOC PGMS
062633 000                       OCT 0
062634 145 261 022              LDMD R45,=NXTMEM
062637 200
062640 263 341 203              STMD R45,=STSIZE
062643 316 377 377              JSB =RELMEM         RELEASE TEMP MEM
062646                  *(such as in LOAD A$&B$)
062646 316 277 161               JSB =INIT14
062651 145 261 006              LDMD R45,=FWCURR    BEGIN OF PGRM
062654 200
062655 055 243                  STM  R45,R55
062657 075 243                  STM  R45,R75
062661 315 007 000              SBM  R45,=7,0,0     GET PROGRAM LENGTH
062664 100
062665 263 310 377              STMD R45,=PTR1
062670 271 313 377              LDMI R45,=PTR1-+    R45 = PGRM LENGTH
062673 230              STORB%  BIN                 FOR ADDS THAT FOLLOW
062674 166 250 020              LDB  R66,=20
062677 167 270 310              LDBI R57,=PTR1      LOAD PRGM HEADER
062702 377
062703 066 224                  ORS  R57,R66        SET GEMINI PRGM FLAG
062705 272 310 377              STBI R57,=PTR1      AND STORE
062710 175 045 305              SBM 75,45           COMPUTE LAST DATA ADDR
062713 263 250 203 STORB-       STMD R#,=LSTDAT
062716 155 263 245 STORB%       STMD R55,=NXTDAT    SET 1ST WORD TO STORE
062721 203
062722 316 346 145               JSB =FILOK?        SCAN FILE,CK TYPE,LEN
062725 316 251 150               JSB =SETFLG        SET   FLAG
062730 316 377 377 DO.BUF        JSB =GETBUF        GET SOME PROG
062733 371 005                   JEZ NOBUFR         DONE IF E=0
062735 316 176 176               JSB =PUTSE+        TO DISC
062740 760 366                   JMP DO.BUF
062742 316 257 150 NOBUFR        JSB =CLRFLG        CLEAR FLAG
062745 236                       RTN
062746                  *---------------------------------------------------
062746                  *Does a disk file exist? If so, is it of correct
062746                  *type and length for writing onto?
062746                  * Incorrect type-->causes error 58 wrong file type
```

```
                STORE file specifier
062746                  * Soft write protect error 60, soft w/protect
062746                  *If file exists but required length is too small,
062746                  *old entry is purged and new file created. Required
062746                  *length is obtained from routine GETCNT.
062746                  *SEEK is performed to ready file data writes.
062746                  *For update of old, new file creation:
062746                  * D.RECS from GETCNT routine
062746                  * D.TYPE from FILTYP
062746                  * D.BYTS from LSTOAT-NXTDAT
062746                  * D.B/RC is   256
062746                  * name    from FNAM,FNAM+5(x14)
062746 316 162 146 FILOK? JSB =DIRSCN           SCAN DIR. FOR NAME
062751 371 110          JE2 GOTNAM              FOUND NAME
062753 316 246 146      JSB =MTSCAN             LOOK FOR EMPTYS
062756 760 054          JMP MT?                 EMPTY OR MARK?
062760 145 261 151 BADLEN LDMD R45,=DENTRY      GET OLD DIR INFO
062763 207
062764 014 247          STMD 45,R14             KEEP OLD DENTRY,DADDR
062766 316 246 146      JSB =MTSCAN             LOOK FOR EMPTYS
062771 316 223 163      JSB =WRTDIR             SAVE NEW(UPDATED)DIR
062774 145 261 151       LDMD R45,=DENTRY       GET NEW DIR INFO
062777 207
063000 006 345           PUMD R45,+R5           SAVE IT
063002 014 245           LDMD R45,R14           RESTORE OLD DIR INFO
063004 263 151 207       STMD R45,=DENTRY END RESTORE
063007 132 046 241       LDM R32,R46            GET SECTOR# DADDR
063012 316 120 176       JSB =GETBUF            RESTORE OLD DIRECTORY
063015 316 307 151       JSB =PURFIL            PURGE OLD FILE
063020 145 006 343       POMD R45,-R5           RESTORE NEW DIR INFO
063023 263 151 207       STMD R45,=DENTRY
063026 146 032 243       STM R46,R32            GETSEC TARGET SECTOR
063031 316 120 176       JSB =GETBUF            RESTORE NEW DIRECTORY
063034 316 106 147 MT?   JSB =DIRPTR            R36:=DIR PTR(DENTRY,DADDR)
063037 034 243           STM R#,R34            FOR PUSHING NAME
063041 143 261 103       LDMD R43,=FNAM  NAME 1ST HALF
063044 207
063045 345               PUMD R43,+R34          TO DIRECTORY
063046 261 110 207       LDMD R43,=FNAM+5 NAME 2ND HALF
063051 345               PUMD R43,+R34          TO DIR
063052 130 260 271       LDBD R30,=FILTYP       GET FILE TYPE
063055 203
063056 316 374 143       JSB =PUTTYP            UPDATE FILE TYPE
063061 760 023           JMP TYPOK              NOW WRITE FILE AND DIR.
063063                  *************************************************
063063 316 336 143 GOTNAM JSB =GETTYP           DO TYPE CHECK
063066 206               LR3 R#                 CHECK SOFT W/PROTECT
063067 763 003           JEV GOTNA+
063071 316 000 151       JSB =ERR22D
063074 204           GOTNA+ LL3 R#              SHIFT BACK FOR AND
063075 320 271 203       CMBD R#,=FILTYP        PROGRAM TYPE FILE?
063100 367 004           JZR TYPOK
063102 316 225 161 ER68D JSB =MSERR+            SYSTEM ERROR
063105 104               OCT 68D                WRONG FILE TYPE
```

```
>>>>>>          STORE file specifier                              <<<<<<<
:  063105                  TYPOK  BSS 0
 35 063106 316 377 377            JSB =GETCNT          GET PROG LENGTH
936 063111 134                    DRP 34
937 063112 316 165 147            JSB =LDRECS
938 063115 032 305                SBM R#,32            FILE BIG ENOUGH?
939 063117 372 237                JNC BADLEN
940 063121 145 220                TSB R45
941 063123 367 002                JZR STOW
942 063125 146 213                DCN R46
943 063127 145 036 267 STOW       STMD R45,X36,D.BYTS
943 063132 034 000
944 063134 132 250 001            LDB R32,=1           DONE FOR CAPR CATS.
945 063137 266 037 000            STBD R32,X36,D.FLAG  UPDATE LOG.REC.LEN.
946 063142 316 132 147            JSB =LDFBEG
947 063145 132 006 345            PUMD R32,+R5         STASH SEEK ADDR
948 063150 316 223 163            JSB =WRTDIR          REWRITE DIR
949 063153 130 006 343            POMD R30,-R5         RESTORE SEEK ADDR
950 063156 316 230 176            JSB =HLSEEK            GO TO IT
951 063161 236                    RTN
```

```
                DIRECTORY SUBROUTINES
063162                  DIRSCN  EQS 0
063162                  *SCAN DIRECTORY OF ACTIVE MSUS FOR NAME IN
063162                  * FNAM,FNAM+5                   RETURNS:
063162                  *                       E=0 IF NAME FOUND
063162                  *                       E=1 IF NAME NOT FOUND
063162                  *                       R36 IS DIR. POINTER IF E=0
063162 316 360 146              JSB =GETDIR             GET 1ST DIR SECTR
063165 235                      CLE                     FOUND FLAG
063166 316 336 143 DLOOP        JSB =GETTYP             FILE TYPE
063171 267 032                  JZR NXTPAS              JIF NULL FILE
063173 210                      ICB R#                  TEST FOR LAST FILE
063174 367 034                  JZR DDONE               NO MORE FILES
063176 143 036 245              LDMD R43,R36            LOAD 1ST 1/2 NAME
063201 153 261 103              LDMD R53,=FNAM  SAME SO FAR?
063204 207
063205 043 301                  CMM R53,R43             SAME SO FAR?
063207 766 014                  JNZ NXTPAS              DIFFRENT FIRST HALVES
063211 261 110 207              LDMD R53,=FNAM+5 SAME THRU-OUT?
063214 143 036 265              LDMD R43,X36,SEC1/2  LOAD 2ND HALF
063217 005 000
063221 053 301                  CMM R43,R53             SAME THRU-OUT?
063223 367 006                  JZR DFOUND              NAME MATCH
063225 316 044 147 NXTPAS       JSB =NXTENT             GET NEXT DIR ENTRY
063230 371 334                  JEZ DLOOP               JIF NOT DIR.END
063232 234              DDONE   ICE                     FLAG FAILURE
063233 236              DFOUND  RTN
063234                  *---------------------------------------------------------
063234                  *FIND THE 1ST HOLE--USED IN PACK
063234 130 223          HOLE#1  CLM R30                 0-SZ HOLE FOR MTSCAN
063236 263 276 203              STMD R#,=CR.CNT         FOR FILE SIZE
063241 250 020                  LDB R#,=DATATY
063243 262 271 203              STBD R#,=FILTYP
063246                  *FALL THRU TO MTSCAN
063246                  MTSCAN  BSS 0
063246                  *SCAN DIR OF ACTIVE MSUS TO FIND A HOLE BIG ENUF
063246                  *FOR A STORE. ERROR EXIT IF NO HOLE BIG ENUF.
063246 316 360 146              JSB =GETDIR             FIRST DIR ENTRY
063251                  NXTONE  BSS 0
063251 120 261 154              LDMD R20,=CUMREC CUMMUL.REC.CNT
063254 207
063255 122                      DRP 22
063256 316 165 147              JSB =LDRECS
063261 316 377 377              JSB =GETCNT R32:=REC FILE SIZE
063264 316 336 143              JSB =GETTYP             FILE TYPE
063267 210                      ICB R#                  LAST FILE?
063270 367 027                  JZR LASTF1              YES
063272 212                      DCB R#                  HOLE FILE?
063273 766 005                  JNZ *EMPTY              NOT A HOLE
063275 122 032 301              CMM R22,R32             FILE LEN. - PROG LEN.
063300 373 331                  JCY DFOUND              HOLE BIG ENUF
063302                  *EMPTY  BSS 0
063302 120 022 303              ADM R20,R22             CUMREC:=CUMREC+FILE LEN
063305 263 154 207              STMD R20,=CUMREC
```

DIRECTORY SUBROUTINES

```
063310 316 044 147            JSB =NXTENT           NXT DIR ENTRY
063313 371 334                JEZ NXTONE            JIF NOT DIR.END
063315 316 212 161            JSB =MSERR
063320 030                    OCT 24D               DIR OVERFLOW
063321                 *MAKE SURE ENTIRE FILE WILL FIT IN THIS DISC VOLUME
063321                 LASTFI BSS 0
063321 132                    DRP 32                FILERECS:=NEC FILE SIZE
063322 316 140 147            JSB =STRECS
063325 120                    DRP 20                FILEORG:=CUMREC
063326 316 121 147            JSB =STFBEG
063331 032 303                 ADM R#,R32     BEG NEW FILE + FILE SIZE
063333 361 002                JNO CKDISK      CHECK DISK SIZE
063335 360 015                JMP ER28D       DEFINITE SIZE PROBLEM!
063337 306 345         CKDISK PUMD R#,+R6     SAVE FROM TOID
063341 316 035 175            JSB =TOID
063344 134 006 343            POMD R34,-R6    RESTORE AFTER TOID
063347 146 034 301            CMM R46,R34  MEDIUM SIZE - END NEW FILE
063352 365 132                JPS DIRPTR            IT FITS. RESTORE DIR PTR
063354 316 212 161 ER28D      JSB =MSERR
063357 034                    OCT 28D               DISC SIZE TOO SMALL
063360                 *--------------------------------------------------
063360                 GETDIR  BSS 0
063360                 *GETS 1ST SECTOR OF DIR. OF ACTIVE MSUS
063360                 *DIR POINTER (X14)DENTRY POINTS AT 1ST DIR ENTRY
063360                 *Note! Performs INTERCHANGE-->CAPR conversion for
063360                 *V.DBEG and V.DLEN in volume label.
063360 132 223                CLM R32                 SECTOR#0
063362 316 034 147            JSB =GETD+            READ VOL LABEL
063365 142 036 265            LDMD R42,X36,V.LABL GET VOL NAME
063370 002 000
063372 263 127 207            STMD R42,=SPECIF STASH FOR CAT
063375 146 265 012            LDMD R46,X36,V.DBEG WHERE DIR STARTS
063400 000
063401 316 147 147            JSB =SWAP+            SWAP 46,47
063404 145 222                CLB R45               INIT DENTRY
063406 263 151 207            STMD R45,=DENTRY MARK IT
063411 132 046 241            LDM R32,R46             FIRST DIR SECTOR
063414 130 036 265            LDMD R30,X36,V.DLEN  FIND END OF DIRECTORY
063417 022 000
063421 316 147 147            JSB =SWAP+
063424                        DRP !30
063424 046 303                ADM R30,R46               NEXT FILE ORIGIN
063426 263 154 207            STMD R30,=CUMREC   (FOR MTSCAN)
063431 263 156 207            STMD R30,=LSTDIR   DIR.END FOR NXTENT
063434 316 120 176 GETD+      JSB =GETBUF           GET 1ST DIR.SECTOR
063437 136 251 200 RBUF36     LDM R36,=REDBUF       POINT AT DIR
063442 205
063443 236                    RTN
063444                 *--------------------------------------------------
063444                 NXTENT  BSS 0
063444                 *Advances to next entry (if one exists then E:=0
063444                 *else E:=1).
063444                 *Directory scanning must follow the sequence:
```

```
GMA 03.14.R1 ================================================================
ITEM    LOC    OBJECT CODE   SRC=LOCMA3 OBJ=GEXGMA   9/11/1981  3:03 PM  PG 31
================================================================
                    DIRECTORY SUBROUTINES
    063444                  *  GETDIR
    063444                  *  zero or more NXTENTs
1054 063444                 *  read/write sny directory info
1055 063444                 *  zero or more NXTENTs
1056 063444                 *  WRTDIR
1057 063444 235                     CLE                 DIR.END REACHED IF E=1
1058 063445 316 106 147             JSB =DIRPTR         R36:=DIR PTR
1059 063450                 *MOVES DIR POINTER TO NEXT DIR ENTRY
1060 063450 145 261 151             LDMD R45,=DENTRY GET PTR
1060 063453 207
1061 063454 312 040                 ADB R45,=32D        BUMP TO NEXT ENTRY
1062 063456 262 151 207             STBD R45,=DENTRY UPDATE
1063 063461 766 023                 JNZ NXTFIN         DONT NEED NEW SECTOR
1064 063463 146 211                 ICM R46             DADDR
1065 063465 263 152 207             STMD R46,=DADDR
1066 063470 130 261 156             LDMD R30,=LSTDIR LAST DIR SECTOR
1066 063473 207
1067 063474 046 301                 CMM R30,R46         PASSED IT?
1068 063476 367 017                 JZR ENDDIR          YES: LSTDIR=DENTRY
1069 063500 132 241                 LDM R32,R46         NEXT DIR SECTOR
1069 063502 316 120 176             JSB =GETBUF         ANOTHER DIR SECTOR
1071 063505 235                     CLE                 AFTER LOW-LEVEL
1072 063506                 NXTFIN  BSS 0
1073 063506 136 223         DIRPTR  CLM R36
1074 063510 260 151 207             LDBD R36,=DENTRY DIR ENTRY
1075 063513 313 200 205             ADM R36,=RECBUF     DIR BASE
1076 063516 236                     RTN
1077 063517                 *-------------------------------------------------
1078 063517 234             ENDDIR  ICE
1079 063520 236                     RTN
1080 063521                 *-------------------------------------------------
1081 063521                 *The following routines swap 2 bytes before storing/
1082 063521                 *after reading in order to perform INTERCHANGE<-->CAPR
1083 063521                 *conversion. (Interchange format calls for most
1084 063521                 *significant byte to precede least.)
1085 063521                 STFBEG  BSS 0           SWAP-STORE R#,X36,D.FBEG
1086 063521 316 147 147             JSB =SWAP+
1087 063524 036 267 016             STMD R#,X36,D.FBEG     R# NOW SWAPPED
1087 063527 100
1088 063530 360 015                 JMP SWAP+              UNSWAP R#
1089 063532                 *------------
1090 063532                 *------------
1091 063532                 LDFBEG  BSS 0           SWAP-LOAD R#,X36,D.FBEG
1092 063532 036 265 016             LDMD R#,X36,D.FBEG
1092 063535 100
1093 063536 360 007                 JMP SWAP+
1094 063540 316 147 147 STRECS      JSB =SWAP+          SWAP STORE R#,X36,D.RECS
1095 063543 036 267 022             STMD R#,X36,D.RECS
1095 063546 100
1096 063547 232             SWAP+   SAD
1097 063550 070 243         SWAP1   STM R#,R70
1098 063552 077 242                 STB R#,R77          SWAP R70,R71
1099 063554 170 071 240             LDB R70,R71
```

```
ARIA 03/14/81=================================================================
ITEM    LOC   OBJECT CODE  SRC=LCGMAS OBJ=GEXGNA   9/11/1981  3:03 PM  PG 33
=================================================================
>>>>>>      DIRECTORY SUBROUTINES                              <<<<<<<<
1100  063557 177 242              STB R77,R71        SWAP COMPLETE
1101  063561 237                  PAD
1102  063562 970 241              LDM R#,R79
1103  063564 236                  RTN
1104  063565                *------------
1105  063565                LDRECS  BSS 0          SWAP-LOAD R#,X36,D.RECS
1106  063565 036 265 022           LDMD R#,X36,D.RECS
1106  063570 000
1107  063571 760 354              JMP SWAP+
```

```
                        LOADBIN file specifier
063573 241                      OCT 241
063574                  MSLDB.  BSS 0
063574 003                      ARP 3
063575 316 313 161              JSB =TAPOS-         DECODE FILE & CHECK MSUS
063600 250 010          MSLOD   LDB  R#,=BFGMTY
063602 262 271 203              STBD R#,=FILTYP
063605                  * MAY 1981: MULTIPLE BINARIES
063605 316 377 377              JSB =BINMOV
063610 345                      PUMD R#,+R#         LOAD BIN PTR
063611 316 306 147              JSB =LOADB+          GO DO LOAD
063614 145 006 343              PCMD R45,-R6         RESTORE LOADBIN PTR
063617                  *NOW BRANCH TO (TAPE)COMMON LOAD POST-PROCESSING
063617 316 377 377              JSB =ROMJSB          BINARY LOAD CLEANUP
063622 377 377                  DEF LDB.+
063624 000                      OCT 0
063625 236                       RTN
```

```
>>> LOAD file specifier                                                        <<<<<<<<
063626 141                           OCT 141
063627                   MSLOD.      BSS 0
063627 104                           ARP 4
063630 316 313 161                   JSB =TAPDS-           DECODE FILE & CHECK MSUS
063633 250 005                       LDB R#,=5             ROMINI #5
063635 262 065 210                   STBD R#,=ROMFL
063640 316 377 377                   JSB =ROMJSB
063643 377 377                       DEF SCRA.-            STORE AND SCRATCH
063645 000                           OCT 0
063646 316 277 161 AUTOST            JSB =INIT14           SYSTEM DESTROYS!
063651                   *( PROVISION FOR DISC AUTOSTART)
063651 116 210                       ICB R16                FOR ERRORS
063653 316 267 147                   JSB =LOAD+             FOR ERROR INTERCEPT
063656 116 222                       CLB R16
063660 316 377 377 LDCOMN            JSB  =ROMJSB
063663 377 377                       DEF  LDFNSH
063665 000                           OCT 0
063666 236               LOCOMX      RTN
063667                   *--------------------------------------------------------------
063667 155 250 040 LOAD+  LDB R55,=CAPRTY        SET FILE TYPE
063672 262 271 203        STBD R55,=FILTYP
063675 261 025 200        LDMD R55,=LAVAIL       SET LOAD LIMIT
063700 263 250 203        STMD R55,=LSTDAT
063703 261 006 200        LDMD R55,=FWCURR       SET LOAD ADDRESS
063706 263 245 203 LOADB+ STMD R#,=NXTDAT
063711 106 263 066 LOADB& STMD R6,=SAVER6     FOR ERRORS IN LOAD,CHAIN
063714 210
063715                   *in case failure occurs in LOAD,CHAIN, or LOADBIN
063715                   *report error then return to CALLING routine (not
063715                   *system) to do corrective SCRATCH, or delete binary
063715                   *NOW LOAD MEM. BOUNDED BY NXTDAT<LSTDAT FROM DISC
063715 316 164 150          .        JSB =GETFIL         DIR.SCAN,TYPE CK,SEEK
063720 316 002 165                   JSB =BYTES          #BYTES LAST REC,#RECS
063723 306 345                       PUMD R#,+R6         4 BYTES @ R44
063725 156 260 271                   LDBD R56,=FILTYP
063730 203
063731 314 010                       SBB  R56,=BPGMTY
063733 145                           DRP  45
063734 366 010                       JNZ  SWITCH
063736 261 250 203                   LDMD R45,=LSTDAT   UPPER MEM LIMIT
063741 325 245 203                   SBMD R45,=NXTDAT   -LOWER-->SPACE AVAIL
063744 360 006                       JMP  PRGSIZ
063746                   SWITCH      DRP  !45
063746 261 245 203                   LDMD R45,=NXTDAT
063751 325 250 203                   SBMD R45,=LSTDAT
063754 075 305           PRGSIZ      SBM R#,R75          -PROG SIZE
063756 365 004                       JPS MEMOVF              ENUF SPACE!
063760 316 225 161                   JSB =MSERR+
063763 023                           OCT 19D                MEMORY OVERFLOW
063764 175 221           MEMOVF      TSM R75         D.BYTS=0-->BAD FILE(E.G.COPY)
063766 366 004                       JNZ OKFILE        #BYTES IN FILE SENSIBLE
063770 316 225 161                   JSB =MSERR+       SYSTEM TAPE ERROR
063773 100                           OCT 64D           EMPTY FILE!
```

```
                        LOAD file specified
0637 74 250 361         OKFILE  LDB R#,=1           FROM NOW ON ..
0637 76 262 331 203             STBD R#,=SCRTYP     ITS A SCRATCH ERROR
1179 064001 316 251 150         JSB =SETFLG
1180 064004 132 006 343 NXTREC  PUMD R32,-R6        R32:= GET SECTOR CNT
1181 064007 213                 DCM R#                UPDATE IT
1182 064010 367 044             JZR LSTREC            DO LAST REC
1183 064012 345                 PUMD R#,+R6         SAVE FOR NXT PASS
1184 064013 156 344             PUBD R56,+R6         SAVE FLAG
1185 064015 145 261 245         LDMD R45,=NXTDAT     DATA ADDR.
1186 064020 203
1186 064021 263 314 377         STMD R45,=PTR2        STORE PTR
1187 064024 316 163 176         JSB =GETSE+          READ A CHUNK
1188 064027 145 261 245         LDMD R45,=NXTDAT     UPDATE...
1188 064032 203
1189 064033 156 006 342         POBD R56,-R6         POP FLAG
1190 064036 145                 DRP 45
1191 064037 367 004             JZR  AR02
1192 064041 315 000 002         SBM  R#,=0,2,0      SUBTRACT TWICE AS NEEDED
1192 064044 000
1193 064045                 *   JMP  AR03           AND FALL THROUGH TO ADD
1194 064045 313 000 001 AR02    ADM R#,=0,1,0        256 BYTES TRANSFERRED
1194 064050 000
1195 064051 263 245 203 AR03    STMD R#,=NXTDAT      ... DATA ADDR
1196 064054 260 326             JMP NXTREC           DO SOME MORE
1197 064056                 *BUFFER (PORTION OF) LAST SECTOR BEFORE TRANSFERRING
1198 064056 156 006 344 LSTREC  PUBD R56,+R6         LAST SECTOR
1199 064061 316 133 176         JSB =GETBU+
1200 064064 316 257 150         JSB =CLRFLG
1201 064067 145 261 245         LDMD R45,=NXTDAT     DEST
1201 064072 203
1202 064073 263 314 377         STMD R45,=PTR2
1203 064076 251 200 205         LDM  R45,=RECBUF      SOURCE
1204 064101 000                 OCT 0                 PAD ADDRESS
1205 064102 156 006 342         POBD R56,-R6
1206 064105 145                 DRP 45
1207 064106 367 004             JZR  AR04
1208 064110 313 000 001         ADM  R#,=0,1,0
1208 064113 000
1209 064114 263 310 377 AR04    STMD R#,=PTR1
1210 064117 223                 CLN  R#
1211 064120 122 343             POMD R22,-R#        BYTE COUNT LST.REC
1212 064122 045 243             STN  R22,R45
1213 064124 156 220             TSB  R56
1214 064126 367 005             JZR  AR05
1215 064130 316 377 377         JSB =EMOVUP          FROM BUFFER TO PROG
1216 064133 260 003             JMP AR06
1217 064135 316 377 377 AR05    JSB =EMOVDN          BINARY MOVE
1218 064140                 *
1219 064140                 *LOAD NOW COMPLETED--CLEAN UP AFTER
1220 064140                 *
1221 064140                 *??REMOVE NXT LINE IF MOVUP KILLS 26
1222 064140 175 261 314 AR06    LDMD R75,=PTR2
1222 064143 377
```

```
                LOAD file specifier
    064144 263 245 203          STMD  R#,=NXTDAT
    064147 136 260 271          LDBD  R36,=FILTYP
    064152 203
    064153 310 010              IMB   R36,=BPGMTY     SKIP THIS IF BIN PRGM
    064155 767 004              JZR   OVEREX
    064157 175 263 022          STMD  R75,=NXTMEM     RESET NEXTMEM
    064162 200
    064163 236          OVEREX  RTN
```

```
>>>>>>>      GETFIL                                          <<<<<<<<
      064164 316 162 146 GETFIL  JSB =DIRSCN        NAME IN DIR?
      064167 371 003             JEZ LNAME          LOAD NAME EXISTS
      064171 316 244 151         JSB =ER57D         SYS.FILE NOT FOUND
      064174 316 336 143 LNAME   JSB =GETTYP        FILE TYPE CHECK
      064177 047 242             STB R#,R47         FOR 'ANM'
      064201 147 317 374         ANM R47,=374       TRASH 2 LOWER BITS
      064204 321 271 203         CMMD R47,=FILTYP
      064207 367 003             JZR TYPOK2         TYPE MATCHES
      064211 316 102 146         JSB =ER68D         WRONG FILE TYPE
      064214 136 006 345 TYPOK2  PUMD R36,+R5       SAVE DIR.PTR.
      064217 130                 ORP 30
      064220 316 132 147         JSB =LDFBEG        R30=File begin
      064223 316 230 176         JSB =HLSEEK        GO THERE
      064226 136 006 343         POMD R36,-R5       RESTORE DIR.PTR.
      064231 236                 RTN
```

```
                        MASS STORAGE IS <msus/volume label>
064232 241                      OCT 241
064233                  MASSS.  BSS 0
064233 031                      ARP 3!
064234 316 245 161              JSB =MSINHK         RUNTIME INIT
064237 140 250 002              LDB R40,=2          ALLOW MSUS ONLY
064242 316 371 142              JSB =DCDFIL         DECODE MSUS
064245                          ORP !44
064245 263 077 207              STMD R44,=DEFMSU    NEW DEFAULT MSUS
064250 236                      RTN
064251
064251                  *-------------------------------------------------
064251
064251 250 001          SETFLG  LDB R#,=1
064253 262 224 210      STRFLG  STBD R#,=DFLAG
064256 236                      RTN
064257
064257                  *-------------------------------------------------
064257 122              CLRFLG  CLB R#
064260 360 371                  JMP STRFLG
```

```
                        [UN]SECURE s-type,s-code,file-specifier
064262 241                      OCT 241
064263                  MSSEC.  BSS 0
064263 316 065 151              JSB =SECURC          INTERCEPT TAPE
064266 032                      ARP 32
064267 316 313 161              JSB =TAPOS-          DECODE FILE&MSUS
064272 316 116 151              JSB =DISCOP          READ DIR, ETC
064275 373 060                  JCY NAMESC           NAME SECURE
064277                  P/BTYP EQU 50     BINARY OR PROGRAM TYPE
064277                  E/DTYP EQU 24     DATA OR EXTENDED TYPE
064277                  *add binary program security
064277 133 317 024              ANM R33,=E/DTYP      PGM OR BINARY FILE?
064302 766 074                  JNZ ERR22D            JIF NO
064304 316 146 151              JSB =SREADR          READ RECORD
064307 766 067                  JNZ ERR22D            JIF ALREADY SECURED
064311 316 377 377              JSB =SBYTE           BUILD SECURE BYTE
064314                  STFLAG BSS 0                  SET FLAGS
064314 130 261 200              LDMD R30,=SECURN     LOAD SECURE CODE
064317 200
064320 133 220                  TSB  R33             CAPRICORN OR GEMINI?
064322 130                      DRP  30
064323 367 012                  JZR CAPSEC           JIF CAPRICORN
064325 056 267 033              STMD R#,X56,P.SCOG    STORE GEMINI SECUR
064330 000
064331 137 267 032              STMD R37,XR#,P.SFLG
064334 000
064335 760 010                  JMP CURSTO
064337 056 267 026 CAPSEC STMD R#,X56,P.SCOC      SET CAPRI SECUR
064342 000
064343 137 267 025              STMD R37,XR#,P.SFLC
064346 000
064347 132 261 147 CURSTO LDMD R32,=CURREC      SECTOR # ON DISC
064352 210
064353 316 032 176              JSB =PUTBUF          REWRITE 1ST FILE SECTOR
064356 236              SECEXT RTN
064357 133 047 224 NAMESC ORB R33,R47            ADD BITS
064362 760 005                  JMP NAME+
064364 147 216          NAMEUN NCB R47               COMPLIMENT MASK
064366 133 047 307              ANM R33,R47          TAKE OUT BIT
064371 133 030 242 NAME+  STB R33,R30            FILE TYPE FOR PUTTYP
064374 104 251 313              GTO PURGE+           REWRITE DIR WITH SECURITY
064377 151
064400                  *------------------------------------------------
064400 316 225 161 ERR22D  JSB =MSERR+
064403 026                      OCT 22D              SECURITY
064404                  *--------------------------------------------
064404 141                      OCT 141
064405                  MSUNS.  BSS 0
064405 316 065 151              JSB =SECURC          INTERCEPT TAPE
064410 033                      ARP 33
064411 316 313 161              JSB =TAPOS-          DECODE FILE & MSUS
064414 316 116 151              JSB =DISCOP          DIR,ETC
064417 373 343                  JCY NAMEUN           UNSECURE IF 2,3
064421 133 317 024              ANM R33,=E/DTYP      PRGM OR BINARY TYPE?
```

```
APNH 02/14/81 ================================================================
ITEM    LOC   OBJECT CODE  SRC=LOGMA3 OBJ=GEXGMA   9/11/1981  3:03 PM  PG 40
================================================================
          >>>        [UNSECURE s-type,s-code,file-specifier                <<<<<<<
1312 064424 766 352              JNE ERR22D           JIF NO
1313 064426 316 146 151          JSB =SREADR          READ RECORD
1314 064431 367 323              JZR SECERT           ALREADY UNSECURED
1315 064433 133 220              TSB R33              CAPRICORN OR GEMINI
1316 064435 134                  DRP 34
1317 064436 367 005              JZR CAPUNS           JIF CAPRICORN
1318 064440 265 033 000          LDMD R#,XR#,P.SCOG    UNSEC GEMINI PRGM
1319 064443 760 003              JMP COMPAR
1320 064445 265 026 000 CAPUNS   LDMD R#,XR#,P.SCOC UNSEC CAPRICORN PRGM
1321 064450 321 200 200 COMPAR   CMMD R#,=SECURN     SAME?
1322 064453 766 323              JNE ERR22D           JIF NO
1323 064455 316 377 377          JSB =SBYTE           BUILD SECURE BYTE
1324 064460 216                  NCB R#
1325 064461 030 307              ANM R#,R30           DROP BIT(S)
1326 064463 760 227              JMP STFLAG
1327 064465                  *-------------------------------------------------
1328 064465 316 377 377 SECURC   JSB =ONEB            GET SECURE #
1329 064470 146 317 003          ANM R46,=3,0         ISOLATE 3 BITS
1330 064473 000
1331 064474 262 174 200          STBD R46,=SEC#       SAVE EM
1331 064477 316 304 161          JSB =GETNA!          POP SECURE CODE
1332 064502 136 042 241          LDM R36,R42          USE 1ST TWO
1333 064505 217                  NCM R36              SCRAMBLE THE CODE
1334 064506 205                  LLM R36              LEFT ROTATE
1335 064507 372 001              JNC SHIFTD            THE
1336 064511 211                  ICM R36               BITS
1337 064512 263 200 200 SHIFTD   STMD R#,=SECURN      SAVE EM
1338 064515 236                   RTN
1339 064516                  *-------------------------------------------------
1340 064516 316 162 146 DISCOP   JSB =DIRSCN          LOOK FOR FILE
1341 064521 371 003              JEZ SECNAM
1342 064523 316 244 151          JSB =ER67D           NO NAME FOUND
1343 064526 316 336 143 SECNAM   JSB =GETTYP          FILE TYPE TO R30
1344 064531 130 033 242           STB R30,R33
1345 064534                  *    ORP 30
1346 064534 316 132 147           JSB =LDFBEG
1347 064537 263 147 210           STMD R#,=CURREC    SAVE FOR I/O
1348 064542 104 251 377          GTO SECNA+           USE SYSTEM ROUTINE
1349 064545 377
1349 064546                  *-------------------------------------------------
1350 064546 132 261 147 SREADR   LDMD R32,=CURREC    1ST FILE SECTOR
1350 064551 210
1351 064552 316 120 176          JSB =GETBUF         READ 1ST SECTOR
1352 064555 156 251 200 SREAD+   LDM R56,=RECBUF     GET SECURE FLAG
1352 064560 205
1353 064561 133 056 264          LDBD R33,X56,P.TYPE GET TYPE
1353 064564 006 000
1354 064566 317 020              ANM  R33,=20         SET UP FLAG
1355 064570 130                  DRP  30
1356 064571 367 004              JZR  SREADC
1357 064573 264 032 000          LDBD R30,X56,P.SFLG
1358 064576 236                  RTN
1359 064577 264 025 000 SREADC   LDBD R#,XR#,P.SFLC
```

```
GMA 03/14/81 ================================================================
ITEM    LOC   OBJECT CODE  SRC=LOGMAS OBJ=GEXGMA   9/11/1981  3:03 PM  PG 41
================================================================
>>>>>>      [UNISECURE s-type,s-code,file-specifier                  <<<<<<<
3  064602 036                  RTN
```

```
                PURGE file specifier [,purge code]
064603 241                      OCT 241
064604                  MSPUR.  BSS 0
064604 025                      ARP 25
064605 316 245 161              JSB =MSINHK             RUNTIME INIT
064610 146 250 001              LDB R46,=1              FLAG NO PURGE CODE
064613 126 012 241              LDM R26,R12             CHECK FOR PURGE CODE
064616 325 344 203              EBMD R26,=TOS
064621 310 006                  CMB R26,=6              5 IS STRING ONLY
064623 764 003                  JNG NOPARM              -1 IF STRING ONLY
064625 316 377 377              JSB =ONEB               GET PURGE CODE
064630 146 262 200 NOPARM       STBD R46,=MTFLAG        DISC PURGE CODE
064633 200
064634 316 320 161              JSB =TAPDS              DECODE FILE & CHECK MSUS
064637                  *PURGE A DISC FILE
064637 316 162 146              JSB =DIRSCN             FIND THE FILE
064642 371 004                  JEZ PURCOD              JIF NAME FOUND
064644 316 225 161 ER67D        JSB =MSERR+
064647 103                      OCT 67D                 SYSTEM FILE NOT FOUND
064650                  *
064650 130 260 200 PURCOD       LDBD R30,=MTFLAG         PURGE REMAINING(=0)?
064653 200
064654 766 031                  JNZ PURFIL              NOPE
064656                  PUR+    BSS 0
064656 130 251 377              LDM R30,=377,177        LARGEST INT
064661 177
064662 316 140 147              JSB =STRECS
064665 130 250 377              LDB R30,=377            'LAST' FILE TYPE
064670 316 314 151              JSB =PURGE+             REWRITE DIRECTORY
064673 316 044 147              JSB =NXTENT             FOR ALL REMAINING FILES
064676 316 336 143              JSB =GETTYP             FILE TYPE
064701 210                      ICB R#                  LAST FILE?
064702 367 002                  JZR PURDUN              JIF LAST FILE
064704 371 350                  JEZ PUR+                JIF MORE FILES
064706 236              PURDUN  RTN
064707                  *
064707                  PURFIL  BSS 0
064707                  *PURGE THE CURRENT FILE:
064707                  * DADDR - DISC DIR. SECTOR
064707                  * DENTRY - OFFSET FOR ENTRY IN THIS SECTOR
064707 316 106 147              JSB =DIRPTR             R36:=DIR PTR(DENTRY,DADDR)
064712 130 222                  CLB R30                 NULL FILE TYPE
064714 316 374 143 PURGE+       JSB =PUTTYP             UPDATE FILE TYPE
064717 316 223 163 PURG-+       JSB =WRTDIR             REWRITE DIRECTORY
064722 236                      RTN
```

```
>>>>>>>      RENAME old filespecifier,new filename                    <<<<<<<<
    064723 241                    OCT 241
    064724                MSREN.  BSS 0
1406 064724 034                   ARP 24
1407 064725 316 245 161           JSB =MSINHK          RUNTIME INIT
1408 064730 140 250 001           LDB R40,=1           ALLOW NEW FILENAME ONLY
1409 064733 316 266 161           JSB =DECODE          GET FILENAME
1410 064736 143 261 103           LDMD R43,=FNAM  GET 2ND NAME
1410 064741 207
1411 064742 263 115 207           STMD R43,=GNAM  AND SAVE IT
1412 064745 261 110 207           LDMD R43,=FNAM+5 & REMAINDER
1413 064750 263 122 207           STMD R43,=GNAM+5
1414 064753 316 326 161           JSB =TAPDS+          DECODE FILE & CHECK MSUS
1415 064756                    *DONT ALLOW RENAMING A FILE SO 2 WOULD HAVE SAME NAME
1416 064756 112 014 245           LDMD R12,R14  RESTORE PARAMS
1417 064761 316 266 161           JSB =DECODE   FNAM:=NEW FILENAME, ACTMSUS SAME
1418 064764 316 162 146           JSB =DIRSCN   NEW NAME REALLY NEW?
1419 064767 370 004               JEN NODUPE    ITS NEW
1420 064771 316 225 161 ER63D     JSB =MSERR+   SYSTEM ERROR
1421 064774 977                   OCT 63D       DUPE NAME
1422 064775 316 320 161 NODUPE    JSB =TAPDS    FNAM:=OLD FILESPECIFIER
  23 065000                    *RENAME A DISC FILE
1424 065000 316 162 146           JSB =DIRSCN          SEARCH FILE NAMES
1425 065003 371 003               JEZ RENFIL           JIF FILE FOUND
1426 065005 316 244 151           JSB =ER57D           FILE NOT FOUND
1427 065010 143 261 115 RENFIL    LDMD R43,=GNAM    1ST HALF
1427 065013 207
1428 065014 036 345               FUMD R43,+R36        ...REPLACED
1429 065016 261 122 207           LDMD R43,=GNAM+5     2ND HALF
1430 065021 345                   FUMD R43,+R36        ...REPLACED
1431 065022 360 273               JMP PURG++           REWRITE DIR
```

```
                ERROM, ERRSC
    065024 000 055              OCT 0,55
  4 065026 136 260 121 ERROM.   LDBD R36,=ERROM#
1434 065031 200
1435 065032 760 042             JMP RESULT
1436 065034
1437 065034                 *-----------------------------------------
1438 065034 000 055             OCT 0,55
1439 065036 136 260 141 ERRSC.  LDBD R36,=ERRSC
1439 065041 202
1440 065042 760 032             JMP RESULT
```

```
                        TYP( buffer# )
065044 020 055                  OCT 20,55
065046                  TYP.    BSS 0
065046 035                      ARP 35
065047 316 245 161              JSB =MSINHK         RUNTIME INIT
065052 316 166 152              JSB =GET#           GET BUFFER #
065055 765 003                  JFS TYPOPN          JIF FILE OPEN
065057 316 373 154              JSB =ER66D          FILE NOT OPEN
065062                  TYPOPN  BSS 0               NUMBER TYPE
065062 316 252 155              JSB =ONEBC          SETUP FILE POINTERS
065065 270 314 377              LDBI R#,=PTR2       D32,A65
065070 210                      ICB R#              ADJUST RIGHT DIGIT
065071 376 014                  JRZ ITSA$           JIF ITS A STRING
065073 136 250 001              LDB R36,=1          ITS A NUMBER
065076 137 222          RESULT  CLB R37             TO SAVE BYTES BELOW
065100 316 377 377              JSB =CONBIN         R36 TO R40 REAL
065103 140 012 345              PUMD R40,+R12       ONTO RESULT STACK
065106 236                      RTN
065107 766 005          ITSA$   JNZ TRY340
065111 136 250 003              LDB R36,=3          377 MEANS EOF
065114 260 360                  JMP RESULT
065116 310 340          TRY340  CMB R#,=340         ENTIRE STRING?
065120 766 005                  JNZ NENTIR
065122 136 250 002              LDB R36,=2          ENTIRE STRING
065125 260 347                  JMP RESULT
065127 310 320          NENTIR  CMB R#,=320         START STRING?
065131 766 005                  JNZ NSTART
065133 136 250 010              LDB R36,=8D         START STRING
065136 260 336                  JMP RESULT
065140 310 200          NSTART  CMB R#,=200         MIDDLE STRING?
065142 766 005                  JNZ NMIDDL
065144 136 250 011              LDB R36,=9D         MIDDLE STRING
065147 260 325                  JMP RESULT
065151 220              NMIDDL  TSB R#              END STRING?
065152 764 005                  JNG NEND$
065154 136 250 012              LDB R36,=10D        END STRING
065157 260 315                  JMP RESULT
065161 136 250 004      NEND$   LDB R36,=4          EOR TYP VALUE
065164 260 310                  JMP RESULT
065166                  *---------------------------------------------------
065166 316 377 377      GET#    JSB =ROMJSB
065171 377 377                  DEF GET#1           GET BUFFER#
065173 000                      OCT 0
065174 236                      RTN                 ROMJSB NEEDED FOR ERRORS!
```

```
                        CREATE file specifier,#recs[,#bytes]
065175 241                      OCT 241
065176                  MSCRE.  BSS 0
065176 007                      APP 7
065177 316 245 161              JSB =MSINHK         RUNTIME INIT
065202 162 263 332      MSCRE+   STMD R12,=INCRA     FOR TAPE TRANSFER
065205 203
065206                  *USE INCRA (NOT R14) BECAUSE COPY USES R14
065206                  *MSCRE+ CALLED FROM COPY
065206 146 251 000              LDM R46,=0,1          DEFAULT 256 B/REC
065211 001
065212 263 245 203              STMD R46,=B/REC
065215 316 377 377              JSB =ONEB           GET FIRST PARAM
065220 370 060                  JEN ER89D           JIF PARAM OVERFLOW
065222 126 012 241              LDM R26,R12         1 OR 2 PARAMS?
065225 325 344 203              EBMD R26,=TOS
065230 311 006 000              CMM R26,=6,0        STRING TAKES 5 BYTES
065233 764 011                  JNG JSTREC          JIF #RECORDS ONLY
065235 146 263 245              STMD R46,=B/REC     1ST OF 2 PARAMS
065240 203
065241 316 377 377              JSB =ONEB           GET # RECORDS
065244 370 034                  JEN ER89D           JIF PARAM OVRFLO
065246                  JSTREC  BSS 0               SAVE #RECORDS
065246 146 263 273              STMD R46,=R/FILE    REALLY save it!
065251 203
065252 316 326 161              JSB =TAPDS+         DECODE FILE & CHECK MSUS
065255                  *CREATE A DISC FILE
065255 146 261 273              LDMD R46,=R/FILE    GET RECS BACK
065260 203
065261 367 017                  JZR ER89D           0 RECORDS IS ERROR
065263 764 015                  JNG ER89D            - IS ERROR
065265 066 243                  STM 46,66           SET FOR INTMUL
065267 176 261 245              LDMD 76,=B/REC      GET BYTES PER REC
065272 203
065273 764 005                  JNG ER89D            - IS ERROR
065275 311 004 000              CMM R76,=4,0        <4 BYTES/REC IS ERROR
065300 373 004                  JCY MULTIP
065302 316 225 161      ER89D    JSB =MSERR+
065305 131                      OCT 89D             INVALID PARAMS
065306 316 377 377      MULTIP  JSB =INTMUL
065311 154 220                  TSB 54              MAKE RESULT MULT OF 256
065313 367 002                  JZR R55OK
065315 155 211                  ICM 55
065317 157 220          R55OK   TSB R57             JIF TOO MANY
065321 766 357                  JNZ ER89D
065323 166 055 241              LDM R66,R55         MOVE 2 BYTE
065326 764 352                  JNG ER89D           JIF TOO MANY
065330 367 350                  JZR ER89D            INCREMENT TO ZERO?
065332 263 276 203              STMD 66,=CR.CNT     COUNT FOR GETBU!
065335 316 162 146              JSB =DIRSCN
065340 370 003                  JEN LIKNAM          NAME FOUND IS ERROR
065342 316 371 151              JSB =ER63D          DUPE FILE NAME
065345 141 250 020 LIKNAM LDB 41,=DATATY            SET TYPE FOR GETCNT
065350 262 271 203              STBD 41,=FILTYP       IN MTSCAN
```

```
                CREATE file specifier,#recs[,#bytes]
065353 316 246 146          JSB =MTSCAN          FIND EMPTY OR NXTAVAIL
065356                    *DIR ENTRY IS BIG ENUF:UPDATE & REWRITE IT
065356 130 261 245          LDMD R30,=B/REC      LOG.BYTES/REC
065361 203
065362 036 267 036          STMD R30,X36,D.B/RC
065365 000
065366 316 165 147          JSB =LDRECS
065371 006 345              PUMD R#,+R6          STASH FOR DATAFILE INIT
065373 316 132 147          JSB =LDFBEG
065376                    *************************************************
065376 006 345              PUMD R#,+R6          STSH FOR LOOP
065400 250 020              LDB R#,=DATATY       ITS DATA FILE
065402 316 374 143          JSB =PUTTYP           INTO DIRECTORY
065405 134 036 241          LDM R34,R36           TRANSFER NAME
065410 143 261 103          LDMD R43,=FNAM   1ST HALF
065413 207
065414 034 345              PUMD R43,+R34         DONE
065416 261 110 207          LDMD R43,=FNAM+5 2ND HALF
065421 345                  PUMD R43,+R34         DONE
065422 316 223 163          JSB =WRTDIR           REWRITE DIRECTORY
065425                    *INIT DATA FILE WITH ALL EOF (377)
065425 130 006 343          POMD R30,-R6          FIRST SECTOR OF FILE
065430                    *COPY speedup: if COPY calls CREATE dont bother
065430                    * doing data file init (fill with 377s):
065430 120 270 310          LDBI R20,=PTR1        LAST RUN TOKEN
065433 377
065434 310 006              CMB R20,=COPYTK       COPY OR CREATE?
065436 367 922              JZR NOTDAT        COPY: NO DATA INIT
065440 316 230 176          JSB =HLSEEK           GO THERE
065443 316 377 377          JSB =FIL377           SYS: RECBUF := ALL 377s
065446 130 006 343 LOP377   POMD R30,-R6         RECORD COUNT
065451 367 011              JZR EX_-_N        LAST REC
065453 213                  DCM R30
065454 345                  PUMD R30,+R6          RESTORE SECTOR COUNT
065455 316 170 176          JSB =PUTBU+           WRITE NEXT RECORD
065460 360 364              JMP LOP377
065462                    ************************
065462 006 343            NOTDAT  PUMD R#,-R6     JUNK REC COUNT
065464 236                EX_-_N  RTN
```

```
                        ASSIGN TABLE FORMAT
065465                  *  ASNTBL IS A 20 BYTE TABLE OF RELATIVE
065465                  *  ADDRESSES REFERENCED TO RTNSTK,POINTING
065465                  *  TO EACH ASSIGNED FILE'S TABLE AND BUFFER.
065465                  *
065465                  *    2 BYTES = 0 IF CLOSED
065465                  * OR 2 BYTES REL. ADDRESS(SEE ABOVE) IF OPEN
065465                  *
065465                  *************************************************
065465                  *  DISC DATA BUFFER POINTER ALLOCATION
065465                  *
065465                  *************************************************
065465                  * CURLOC 0     -  FWCURR OF PGM WHO OPENED
065465                  * A.PGMU 2     -  FWCURR OF CURRENT USING PGM
065465                  * A.PEND 4     -  BUFFER PENDING FLAG
065465                  * A.FILE 5     -  FILE #      THIS SECTOR
065465                  * A.ERFL 6     -  OVF,ERROR FLAG
065465                  * A.PPTR 7     -  PHYSICAL POINTER
065465                  * A.PREC 10    -  PHYSICAL RECORD # (LOCAL TO FILE)
065465                  * A.LPTR 12    -  LOGICAL POINTER
065465                  * A.LREC 14    -  LOGICAL RECORD #
065465                  * A.#R/F 16    -  # LOG.RECORDS / FILE
065465                  * A.#B/R 20    -  # LOG.BYTES / RECORD
065465                  * A.MSUS 22    -  MASS STORAGE UNIT SPEC.
065465                  *A.BSEC  EQU 26  1ST SECTOR OF DISC FILE
065465                  A.CHEK  EQU 30   CHECK READ [OFF] FLAG
065465                  * A.DATA 34    -  256 BYTES DATA
065465                  *************************************************
065465                  * A.RESL 434
065465                  * ASNLEN 12 ( 10 FILES ALLOWED)
065465                  *************************************************
```

```
>>>>>>      ASSIGN# buffer no., file specifier                    <<<<<<<
      065465 241                    OCT 241
      065466                ASSIG.  BSS 0
1602  065466 010                    ARP 10
1603  065467 316 245 161            JSB =MSINHK
1604  065472                *CHECK FOR FILENAME="*" WHICH MEANS CLOSE
1605  065472 175 012 343            FOMD R75,-R12       FILENAME STR ADDR
1606  065475 263 314 377            STMD R#,=PTR2
1607  065500 132 343                FOMD R32,-R12       STR LENGTH
1608  065502 311 001 000            CMM R32,=1,0        LENGTH(FILENAME)=1?
1609  065505 766 034                JNZ NOT"*"          JIF NOT
1610  065507 270 315 377            LDB( R#,=PTR2-      GET THE SINGLE CHAR
1611  065512 310 052                CMB R#,=STAR        IS IT * ?
1612  065514 766 025                JNZ NOT"*"          JIF NOT
1613  065516                *ITS A CLOSE; ASSIGN# buffer no TO *
1614  065516 316 166 152            JSB =GET#           GET BUFF#
1615  065521 765 005                JPS DOCLOS          JIF FILE OPEN
1616  065523 316 377 377            JSB =WARN
1617  065526 102                    OCT 56D             FILE NOT OPEN
1618  065527 236            EXASIN  RTN                 BUFFER WASNT OPEN
1619  065530 165 263 314    DOCLOS  STMD R65,=PTR2
      065533 377
1620  065534 316 143 154            JSB =WRTDA-         DUMP THE BUFFER
1621  065537                *       JSB =RCMJSB
1622  065537 316 377 377            JSB =CLOSE+      (TAPE)BUFFER POST PROCESS
1623  065542                *       OCT 0
1624  065542 236                    RTN
1625  065543                *
1626  065543                NOT"*"  BSS 0
1627  065543 112 014 245            LDMD R12,R14        RESTORE PARAMS
1628  065546 316 326 161            JSB =TAPDS+         DECODE FILE &CHECK MSUS
1629  065551 316 162 146            JSB =DIRSCN         FIND THE NAME
1630  065554 371 003                JEZ ASNNAM          E=0 --> NAME FOUND
1631  065556 316 244 151            JSB =ER67D  FILE NAME
1632  065561 136 263 010    ASNNAM  STMD 36,=CURCAT     SAVE CUR. CAT. LOC.
1632  065564 207
1633  065565 316 336 143            JSB =GETTYP         GET FILE TYPE
1634  065570 262 201 200            STBD R#,=DTAWPR     SAVE FOR SOFT PTCT
1635  065573 133 030 240            LDB  R33,R30        COPY FILE TYPE
1636  065576 317 023                ANN  R33,=23        DATA TYPE+SECURITY
1637  065600 300                    CMB  R33,R30        SHOULD BE THE SAME
1638  065601 367 003                JZR OKSOFR
1639  065603 316 102 146            JSB =ER68D          WRONG FILE TYPE
1640  065606 316 166 152    OKSOFR  JSB =GET#           POP STACK VALUE
1641  065611 764 007                JNS ITSCLS          ZERO IF CLOSED
1642  065613 316 377 377            JSB =SAVACN         CLOSE IT & SAVE MSUS
1643  065616 106 263 066            STMD R6,=SAVER6     RESET SAVER6
1643  065621 210
1644  065622 136 261 010    ITSCLS  LDMD R36,=CURCAT    LOAD DIRECTORY LOC.
1644  065625 207
1645  065626 155 251 034            LDM  R55,=34,1,0
1645  065631 001 000
1646  065633 316 377 377            JSB =ASIGNL         COMPUTE FWA ASSIGN BUFFS
1647  065636 170 220                TSB R70
```

ASSIGN# buffer no., file specifier

```
065640 165                 DRP 65
065641 367 002             JZR CURLO+
065643 055 303             ADM  R#,R55
065645 263 053 210 CURLO+  STMD R#,=CURLOC
065650 145 261 025         LDMD R45,=LAVAIL
065653 200
065654 065 305             SBM  R45,R65
065656 316 377 377         JSB =GETMEM          MOVE ROUTINE
065661 370 244             JEN EXASIN
065663 146 261 006         LDMD R46,=CURASN     LOAD INDEX
065666 207
065667 175 261 033         LDMD R75,=RTNSTK
065672 200
065673 155 261 053         LDMD R55,=CURLOC
065676 210
065677 263 314 377         STMD R55,=PTR2
065702 075 305             SBM R55,R75
065704 132 055 241         LDM R32,R55
065707 046 247             STMD R32,R46         STORE REL. ADDRESS
065711 140 261 006         LDMD 40,=FWCURR      MAKE ASSIGN ENTRIES
065714 200
065715 142 040 241         LDM 42,40            OPENER IS USER
065720 144 223             CLM 44
065722 145 260 201         LDBD R45,=DTAWPR      CHECK SOFT PROTECT
065725 200
065726 206                 LRB
065727 206                 LRB                  SHIFT BIT TO CARRY
065730 232                 SAD                  SAVE BIT
065731 222                 CLB
065732 237                 PAD                  RESTORE BIT
065733 202                 ERB                  CARRY TO MSB
065734 140 273 316         STMI 40,=PTR2+       1ST EIGHT BYTES
065737 377
065740 223                 CLM 40
065741 146 210             ICB R46
065743 142 273 316         STMI R42,=PTR2+
065746 377
065747 146 222             CLB  R46
065751 263 147 210         STMD R46,=CURREC
065754 156 250 010         LDB  R56,=10          FLAG DATA FILE
065757 316 042 144         JSB  =LOGREC
065762 136 273 316         STMI R36,=PTR2+
065765 377
065766 132 273 316         STMI R32,=PTR2+
065771 377
065772 144 261 160         LDMD R44,=ACTMSU  ACT.MSUS THIS FILE
065775 207
065776 273 316 377         STMI R44,=PTR2+        INTO TABLE
066001 136 261 010         LDMD R36,=CURCAT
066004 207
066005 130                 DRP  30
066006 316 132 147         JSB  =LDFBEG
066011 273 316 377         STMI R#,=PTR2+
```

```
-RMM 03/10/81 ==========================================================
ITEN    LOC   OBJECT CODE  SRC=LCGMA3 OBJ=GENGMA   9/11/1981  3:03 PM  PG 51
==========================================================
>>>>>       ASSIGN# buffer no., file specifier                  <<<<<<<
  066014 122                    CLB  R#
  066015 272 316 377            STBI R#,=PTR2+
1691 066020 316 377 377         JSB =SETFER          SET FILE ERROR
1692 066023 316 032 154         JSB =RDATA          INITIALIZE DATA BUFFER
1693 066026 316 377 377          JSB =CLRERF         SYS CLEAR ERROR FLAG
1694 066031 236                  RTN
```

```
                        READ/WRITE DATA BUFFER
 056032                 *The following two routines perform the disk data
 056032                 *buffer i/o. Verification is performed for WRITE.
056032 316 115 154 RDATA    JSB =R/WPRM         PUTSEC PARAMS
056035 316 126 176          JSB =GETSEC
056040 236          WRTRTN  RTN
056041              *-------------------------------------------------------
056041 316 115 154 WDATA    JSB =R/WPRM         GETSEC PARAMS
056044 316 040 176          JSB =PUTSEC
056047 316 115 154          JSB =R/WPRM         INCASE VERIFY
056052                      DRP 154
056052 220                  TSB R54
056053 767 763              JZR WRTRTN          JIF CHECKREAD OFF
056055 316 120 176          JSB =GETBUF         REREAD THE DATA
056060 316 115 154          JSB =R/WPRM         REMAKE DATA BUFF ADDR
056063 316 037 147          JSB =RBUF36         FOR VERIFY
056066 134 250 040          LDB R34,=32D        256 BYTES/8 AT A TIME
056071              *FALL INTO DATA VERIFICATION ROUTINE
056071              *-------------------------------------------------------
056071              *VERIFY THAT R30,R32 CONTAIN THE SAME # BYTES
056071              *(IN R34) OF IDENTICAL DATA. ERROR IF NOT.
056071 140 271 316 CKDAT?   LDMI R40,=PTR2+     ARE THESE 2 BYTES...
056074 377
056075 150 036 341          POMD R50,+R36       ...IDENTICAL?
056100 040 301              CMM R50,R40
056102 766 005              JNZ DATDIF          JIF NOT SAME DATA
056104 134 212              DCB R34             BYTE COUNT
056106 766 361              JNZ CKDAT?          CHECK 1 BYTE MORE
056110 236                  RTN                 ALL BYTES VERIFIED!
056111 316 212 161 DATDIF   JSB =MSERR
056114 033                  OCT 27D             READ VERIFY ERROR
056115              *-------------------------------------------------------
056115              *Set up for disk i/o:
056115              *PTR2 := data buffer address
056115              * R32 := disk sector #
056115              *CURLOC:=ASSIGN# table base address
056115              R/WPRM  BSS  0               CURRENT BUFFER TABLE
056115 316 352 161          JSB =BUFA.M         CURLOC+A.MSUS
056120 144 271 316          LDMI R44,=PTR2+
056123 377
056124 263 160 207          STMD R44,=ACTMSU
056127 132 271 316          LDMI R32,=PTR2+
056132 377
056133 323 147 210          ADMD R32,=CURREC
056136 154 271 316          LDMI R54,=PTR2+     R54=CHECK READ FLAG
056141 377
056142 236                  RTN                AND PTR2=BEGINNING OF BUFFER
```

```
>>>>>          WRTDAT: BUFFER OUTPUT & UPDATE                    <<<<<<<<
      056143             ^WRTDA- STMD R6,=SAVER6   SO CLOSE RETURNS
      056143             WRTDA- BSS 0
1741  056143             *                                  TO RELEASE MEM.
1742  056143 316 377 377 WRTDAT JSB =SETF#
1743  056146 316 100 161        JSB =PRECOR
1744  056151 146 271 314        LDMI 46,=PTR2               AND RECORD#
1744  056154 377
1745  056155 263 147 210        STMD R#,=CURREC
1746  056160 316 377 377        JSB =PENDIN
1747  056163                    DRP 165
1748  056163 220                TSB R65
1749  056164             *      LD3I R#,=PTR2               BUFFER PENDING?
1750  056164 367 031            JZR SERIAL                  JIF NO TO EXIT
1751  056166 371 003            JE2 NOSPTC                  JIF NOT SOFT PTCT
1752  056170 316 000 151        JSB =ERR22D                SOFT WRITE PROTECT
1753  056173 316 377 377 NOSPTC JSB =ERFLAG
1754  056176 270 314 377        LD3I R#,=PTR2               CHECK ERROR FLAG
1755  056201 365 003            JPS NPTCT                   JIF NO FILE ERROR
1756  056203 316 377 377        JSB =ER72D
1757  056206 316 377 377 NPTCT   JSB =SETFER                SET FILE ERROR
1758  056211 316 041 154        JSB =WDATA                  WRITE DATA BUFFER
1759  056214 316 377 377         JSB =NPTC+5                CLEAR FILE,PENDING BITS
1760  056217 236          SERIAL RTN
```

```
OMA 07/14/81 ================================================================
ITEM    LOC    OBJECT CODE  SRC=LCOMA2 OBJ=GENCMA   9/11/1981  3:03 PM  PG 54
================================================================
>>>>>        READ#  SET UP ROUTINE                                  <<<<<<<
 2 066220                  **********READ # ATTRIBUTES TABLE**************
 3 066220 241                    OCT 241
 4 066221                  ********************************************
1765 066221                MSPRNT BSS 0
1766 066221                READ.  BSS 0
1767 066221 011                   ARP 11
1768 066222 316 245 161           JSB =MSINHK      RUNTIME INIT
1769 066225 126 222               CLB R26
1770 066227 262 177 200           STBD 26,=RANDOM     SET TO SERIAL
1771 066232 212                   DCB R26             SET INTERCEPT
1772 066233 262 330 203           STBD R26,=SCT+7
1773 066236 012 241               LDN 26,12           SERIAL OR RANDOM?
1774 066240 325 344 203           SSMD 26,=TOS
1775 066243 220                   TSB R26             CAN BE 10 OR 20
1776 066244 375 025               JLN RNDPR#          20  = 2 VALS = RND
1777 066246 316 266 154           JSB =TOTAP?        GET ASSIGN #
1778 066251 316 377 377           JSB =ERFLAG        GET CURRENT ERFLAG
1779 066254 270 314 377           LDBI R#,=PTR2       CURRENT REC# VALID?
1780 066257 265 003               JPS NOFERR          NO FILE ERROR
1781 066261 316 377 377 FILERR    JSB =ER72D          FILE ERROR
1782 066264 206           NOFERR  LRB R#              CHECK FILE OVF
1783 066265 362 372               JOD FILERR          JIF FILE OVF
1784 066267 316 377 377           JSB =CLRERF         CLEAR ERROR FLAG
1785 066272                *FOR DISC IO: active msus:= msus of current buffer
1786 066272 236                   RTN
1787 066273 316 377 377 RNDPR#    JSB =ONEB           GET REC#
1788 066276 146 024 243           STM 46,24           SAVE IT
1789 066301 266 003               JNZ GORND
1790 066303 316 302 152           JSB =ER89D           RANDOM READ TO 0 ERROR
1791 066306 316 266 154 GORND     JSB =TOTAP?        GET ASSIGN #
1792 066311 316 377 377           JSB =ERFLAG        GET CURRENT ERFLAG
1793 066314 250 001               LDB R#,=1           SET RANDOM <>0
1794 066316 262 177 200           STBD R#,=RANDOM
1795 066321 270 314 377           LDBI R#,=PTR2      ZERO ERFL IF NO FILE
1796 066324 265 023               JPS CLEARE          -RECORD ERROR
1797 066326 136 271 317           LDMI R36,=PTR2-+    DATA PENDING?
1797 066331 377
1798 066332 220                   TSB  R36
1799 066333 266 014               JNZ CLEARE          JIF YES, TRY WRITING AGAIN
1800 066335 223                   CLM R36             SET INVALID PHYS. REC#
1801 066336 213                   DCM R36
1802 066337 316 100 161           JSB =FRECOR
1803 066342 136 273 314           STMI R36,=PTR2      FORCE RE-READ OF REC
1803 066345 377
1804 066346 316 377 377           JSB =ERFLAG
1805 066351                CLEARE BSS 0
1806 066351 222                   CLB R#
1807 066352 272 316 377           STBI R#,=PTR2+        CLEAR
1808 066355 151 271 316           LDMI R51,=PTR2+     GARBAGE TO SET PTR2 FOR JMP
1808 066360 377
1809 066361 136 024 241           LDM  R36,R24        MOVE FOR OUTLR
1810 066364 760 125               JMP OUTLR+
1811 066366
```

```
                READ#  SET UP ROUTINE
!2  056366              * COMMENTS:  SERIAL PRNT#, AND READ#, CRASH
    056366              *            ON HARD FILE ERRORS (ERFL NEG.).
!314 056366             *
!315 056366             *            SERIAL ACCESS CRASHES ON FILE OVF
!316 056366             *
!317 056366             TOTAP?  BSS 0
!318 056366 316 !66 !52         JSB =GET#           GET ASSIGN #
!319 056371 765 004             JFS ITSOPN          BUFFER OPENED
!320 056373 316 225 !61 ER66D   JSB =MSERR+
!321 056376 !02                 OCT 66D             FILE CLOSED
!322 056377 236         ITSOPN  RTN
!323 066400             *            RANDOM ACCESS CLEARS THE LSB OF ERFL
!324 056400             *
```

```
                        READ/WRITE 1 BYTE & ADVANCE
  26 066400 316 252 155 RD+ADV  JSB =ONEBC1          CHECK ERROR,COMPUTE ADDRESS
  27 066403 272 314 377         STBI R#,=PTR2        R32,R66
1826 066406 250 001             LDB R#,=1            SET PENDING FLAG
1827 066410 232                 SAB
1830 066411 316 377 377         JSB =PENDIN
1831 066414 237                 PAD
1832 066415 272 314 377         STBI R#,=PTR2
1833 066420 260 011             JMP ADVANC
1834 066422
1835 066422 316 252 155 RD+ADV  JSB =ONEBC1          CHECK ERROR,COMPUTE ADDRESS
1836 066425 270 314 377         LDBI R#,=PTR2        R32,R66
1837 066430 262 174 200         STBD R#,=DATUM       SAVE TILL EXIT
1838 066433 154 210     ADVANC  ICB R54
1839 066435 266 003             JNZ NOPOVF           JIF NO PHYS OVF
1840 066437 316 143 154         JSB =WRTDAT          WRITE IF NECESSARY
1841 066442 316 107 161 NOPOVF  JSB =LRECOR
1842 066445 172 271 314         LDMI R72,=PTR2         R76=BYTES/RECORD
1842 066450 377
1843 066451 166 271 317         LDMI R66,=PTR2-+     R66=L PTR.LEAVES PTR2 AT LREC
1844 066454 377
1845 066455 211                 ICM R66               INCR. LOG PTR
1845 066456 076 301             CMM R66,R76          OVF LOG RECORD?
1846 066460 373 010             JCY OUTLR            JIF OUTSIDE L.RECORD
1847 066462 166 064 243 RSPTR   STM R66,R64           COPY L PTR TO 64
1848 066465 271 314 377         LDMI R#,=PTR2      PUT L REC# IN 66
1849 066470 260 050             JMP LROK+
1850 066472
1851 066472 316 377 377 OUTLR   JSB =ERFLAG
1852 066475 260 177 200         LDBD R#,=RANDOM
1853 066500 272 314 377         STBI R#,=PTR2        0 IF SERIAL,1 IF RANDOM
1854 066503 316 107 161 REDNXT  JSB =LRECOR
1855 066506 136 271 316         LDMI R36,=PTR2+       INCR. LOG REC#
1855 066511 377
1856 066512 211                 ICM R36
1857 066513 166 271 314 OUTLR+  LDMI R66,=PTR2        COMPARE TO RECS/FILE
1857 066516 377
1858 066517 036 301             CMM R66,R36
1859 066521 373 012             JCY LROK             JIF NO OVF FILE
1860 066523 316 377 377         JSB =ERFLAG
1861 066526 250 002             LDB  R#,=2           NOTE FILE OVF
1862 066530 272 314 377         STBI R#,=PTR2        ADD FILE OVF TO ERFL
1863 066533 260 110             JMP REST32
1864 066535
1865 066535 164 223     LROK    CLM R64              0 LPTR
1866 066537 166 036 241         LDM R66,R36          MOVE LREC
1867 066542 316 377 377 LROK+   JSB =LPOINT
1868 066545 164 273 316         STMI R64,=PTR2+       SET LPTR,LREC
1868 066550 377
1869 066551 166 213             DCM R66              FOR INTMUL
1870 066553 174 271 314         LDMI R74,=PTR2        76=BYTES/REC
1870 066556 377
1871 066557 316 377 377         JSB =INTMUL
1872 066562 164 271 315         LDMI R64,=PTR2-       ADD LPTR OFFSET
```

```
>>>>>         READ/WRITE 1 BYTE & ADVANCE                                  <<<<<<<<
    066565 377
    066566 166 223              CLM R66
1874 066570 154 064 303         ADM 54,64                 INTO PHYS RECORD
1875 066573 166 271 315         LDMI 66,=PTR2-           SAME PRECORD?
1875 066576 377
1876 066577 055 301             CMM 66,55
1877 066601 154                 DRP 54
1878 066602 367 036             JZR NOR/W                NO NEED TO ADVANCE
1879 066604 306 345             PUMD 54,+R6              SAVE POINTERS
1880 066606 316 143 154         JSB =WRTDAT              WRT CUR PREC IF NECESSARY
1881 066611 155 006 342         POBD 55,-R6              TRASH MSBYTE
1882 066614 343                 POMD 55,-R6              RESTORE POINTERS
1883 066615 316 377 377         JSB =PPOINT
1884 066620 155 273 314         STMI 55,=PTR2            R55=PREC POINTER
1884 066623 377
1885 066624 156 263 147         STMD 56,=CURREC          R56,57=PHYS REC#
1885 066627 210
1886 066630 316 377 377         JSB =SETF#               SET FILE #
1887 066633 316 032 154         JSB =RDATA              READ DATA BUFFER
1888 066636 316 377 377         JSB =CLFBIT              CLEAR ERROR BIT
1889 066641 236                 RTN
1890 066642 272 315 377 NOR/W  STBI R#,=PTR2-           SET PPTR
1891 066645 132 260 174 REST32 LDBD R32,=DATUM          RESTORE R32 FOR READS
1891 066650 200
1892 066651 236                 RTN
1893 066652                  *-----------------------------------------------
1894 066652 316 377 377 ONEBC!  JSB =ROMJSB
1895 066655 377 377              DEF ONEBCM
1896 066657 000                  OCT 0
1897 066660 236                  RTN
1898 066661                      LST
```

```
>>>>>>          PRINT STRING RUNTIME                                  <<<<<<<<
000 056651                  ENTIR$ EQU 337
001 056651                  START$ EQU 317
002 056651                  MIDDL$ EQU 177
003 056651                  END#   EQU 157
004 056651                  EOR    EQU 357
005 056651 036                     OCT 36
006 056652                  PRSTR. BSS 0
007 056652 014                     ARP 14
008 056653 316 245 161             JSB  =MSINHK
009 056656 143 012 343             PCMD R43,-R12        STRING LEN & ADDR
010 056661                  PRSTR+ BSS 0                CHECK MSUS
011 056661 316 340 161             JSB  =CKMSUS
012 056664 155 043 241             LDM  55,43
013 056667 157 222                 CLB  57
014 056701 145 213                 DCM  R45             BACKUP TO DATA
015 056703 263 060 210             STMD R45,=RESDAT     R43=LEN,R45=ADDR ON ENTRY
016 056706 055 305                 SBM 45,55            COMPUTE ECDATA
017 056710 263 161 210             STMD 45,=RESEND
018 056713 223                     CLM  R45
019 056714 213                     DCM  R45
020 056715 263 245 203             STMD R45,=NXTDAT     OFFSET FOR LOOP
021 056720 146 250 337             LDB 46,=ENTIR$
022 056723 262 173 200             STBD 46,=DATYPE      INIT HEADER TYPE
023 056726 043 241                 LDM 46,43
024 056730 313 003 000 DOCHK       ADM R#,=3,0          ADD FOR HEADER BYTES
025 056733 263 056 210             STMD R#,=DATLEN
026 056736 026 243                 STM  R#,R26          FOR LENCHK NULL STRINGS
027 056740 316 377 377             JSB =LENCHK          OVF LREC?
028 056743 373 070                 JCY INBNO$
029 056745                         DRP !56
030 056745 076 305                 SBM R#,R76           AMT DATA WHICH FITS
031 056747 263 056 210             STMD R#,=DATLEN
032 056752 311 004 000             CMM R#,=4,0          SHOULD I EVEN START?
033 056755 373 016                 JCY ITFITS           JIF YES
034 056757 316 103 155             JSB =REDNXT          ADVANCE TO NXT LREC
035 056762 316 377 377             JSB =ERFLAG          BUFFER LOC
036 056765 270 314 377             LDBI R#,=PTR2          FILE OVF?
037 056770 367 027                 JZR DOBUMP           JIF NO
038 056772 316 377 377             JSB =ER72D           OVF,NEED RECORD#
039 056775 132 260 173 ITFITS      LDBD R32,=DATYPE
039 057000 200
040 057001 310 337                 CMB 32,=ENTIR$       1ST TIME SET START
041 057003 367 004                 JZR USTART           USE START
042 057005 250 177                 LDB 32,=MIDDL$       ELSE SET MIDDLE STRING
043 057007 360 002                 JMP OVER
044 057011 250 317         USTART  LDB R#,=START$
045 057013 262 173 200     OVER    STBD R#,=DATYPE
046 057016 316 047 156             JSB =USEHOL          WRITE TYPE,STRING
047 057021 155 261 060 DOBUMP      LDMD 55,=RESDAT      COMPUTE REMAINDER
047 057024 210
048 057025 325 161 210             SSMD 55,=RESEND
049 057030 146 055 241             LDM R46,R55
050 057033 360 273                 JMP DOCHK
```

```
                        PRINT STRING RUNTIME
1951 067035 132 260 173 INBND# LDBD 32,=DATYPE      USE WHOLE STRING
     067040 200
1952 067041 310 337            CMB 32,=ENTIR$       OR END STRING TYPE
1953 067043 367 002            JZR USEHOL
1954 067045 250 157            LDB 32,=END$
1955 067047 316 000 155 USEHOL JSB =WT+ADV          WRITE TYPE
1956 067052 165 261 060        LDMD R65,=RESDAT     WRITE LENGTH
1956 067055 210
1957 067056 325 161 210        SBMD R65,=RESEND
1958 067061 132 065 241        LDM  R32,R65
1959 067064 133 006 344        PUSD 33+R6           SAVE SECOND HALF
1960 067067 316 000 155        JSB =WT+ADV
1961 067072 006 342            POSD R#,-R6          RESTORE SEC HALF
1962 067074 316 000 155        JSB =WT+ADV
1963 067077 126 261 056        LDMD 26,=DATLEN      LENGTH=LENGTH-3 NOW
1963 067102 210
1964 067103 315 003 000        SBM 26,=3,0
1965 067106 263 056 210        STMD 26,=DATLEN
1966 067111 126 261 056 WSTRNG LDMD 26,=DATLEN
1966 067114 210
1967 067115 367 074            JZR WREOR!
1968 067117 316 340 157        JSB =MOVTST          SEE IF CAN MOVE ALL AT ONCE
1969 067122 373 042            JCY WSTRN1           JIF NO
1970 067124 165 261 310        LDMD R65,=PTR1       SAVE PROGRAM POINTER
1970 067127 377
1971 067130 145 263 314        STMD R45,=PTR2       R45=>SINK ADDR (ASSIGN BUFF)
1971 067133 377
1972 067134 155 263 310 AWRITE STMD R55,=PTR1       R55=>SOURCE ADDR
1972 067137 377
1973 067140 101 271 310        LDMI R*,=PTR1        R0 DETERMINES # OF BYTES LOADED
1973 067143 377
1974 067144 273 316 377        STMI R*,=PTR2+
1975 067147 155 323 245        ADMD R55,=NXTDAT     MOVE SOURCE ADDR
1975 067152 203
1976 067153 321 161 210        CMMD R55,=RESEND     MOVED ALL DATA?
1977 067156 266 354            JNZ  AWRITE          JIF NO
1978 067160 165 263 310        STMD R65,=PTR1       RESTORE PROGRAM PTR
1978 067163 377
1979 067164 260 025            JMP  WREOR!          FINISH
1980 067166
1981 067166 126 213      WSTRN1 DCM 26                 DECR DATA COUNT
1982 067170 263 056 210        STMD 26,=DATLEN
1983 067173 316 067 160        JSB  =RESSET         RESET PTR2 AND RESDAT
1984 067176 132 270 314        LDBI R32,=PTR2         GET DATA BYTE
1984 067201 377
1985 067202 316 000 155        JSB =WT+ADV            SEND THE BYTE
1986 067205 126 261 056        LDMD R26,=DATLEN
1986 067210 210
1987 067211 266 353            JNZ WSTRN1
1988 067213 316 377 377 WREOR!  JSB =WREOR
1989 067216 236                 RTN
```

```
>>>>>          PRINT# NUMERIC                                          <<<<<<<
      067217 036                    OCT 36
1992  067220                PRNUM.  BSS 0
1993  067220 012                    ARG 12
1994  067221 316 245 161            JSB =MSINHK
1995  067224 316 377 377            JSB =ONER               POP ONE REAL
1996  067227 316 340 161            JSB =CKMSUS             CHECK MSUS
1997  067232                PR#DSK  BSS 0
1998  067232 126 251 010            LDM 26,=10,0            RESERVE 10 BYTES
1998  067235 000
1999  067236 263 056 210            STMD 26,=DATLEN         SAVE DATA LENGTH
2000  067241 155 223                CLM  R55
2001  067243 211                    ICM  R55
2002  067244 263 245 203            STMD R55,=NXTDAT        OFFSET FOR NUMERICS
2003  067247 251 200 205            LDM R55,=RECBUF         TEMP DATA LOC
2004  067252 000                    OCT 0
2005  067253 263 161 210            STMD R55,=RESEND
2006  067256 026 304                SBB  R55,R26
2007  067260 263 060 210            STMD R55,=RESDAT      BEGINNING OF TEMP DATA LOC
2008  067263 140 055 247 RD=17      STMD 40,55              PUT THE DATA THERE
2009  067266 210                    ICB R40                 DONT LET STRING FLAG THRU
2010  067267 376 372                JRZ RD=17               JIF "STRING FLAG"
2011  067271 316 377 377            JSB =LENCHK             CHECK FOR OVF LOG. REC.
2012  067274 373 213                JCY WSTRNG
2013  067276 311 010 000            CHM R#,=10,0            WILL IT FIT EVER?
2014  067301 372 005                JNC R#ER              JIF NO
2015  067303 316 103 155            JSB =REDNXT             READ THE NEXT REC.
2016  067306 760 201                JMP WSTRNG
2017  067310
2018  067310
2019  067310 316 377 377 R#ER       JSB =ER72D              OVF,NEED REC #
```

```
     >>>>        READ STRING ROUTINE                                             <<<<<<<<
067313 044                  OCT 44
067314              RDSTR.  BSZ 0
067314 020                  ARP 20
067315 316 245 161          JSB =MSINHK
067320 316 340 161          JSB =CKMSUS          CHECK MSUS
067323 360 003               JMP RD#$
067325 316 072 155 EOR$     JSB =OUTLR           ADVANCE ON EOR
067330 316 331 157 RD#$     JSB =R#COM!
067333 377 174              JRN ER33D           ALL STRING TYPES GIVE RDZ
067335 220                  TSB R#               FIX POSITIVE!!
067336 365 020              JPS ALL$             JIF END$ (157+1)
067340 310 360              CMB R#,=360
067342 367 361              JZR EOR$             ADVANCE ON EOR
067344 310 340              CMB R#,=340         ENTIRE STRING?
067346 367 010              JZR ALL$             JIF YES
067350 260 177 200          LDBD R#,=RANDOM     CHECK RANDOM
067353 367 003              JZR ALL$             JIF NOT RANDOM
067355 316 377 377          JSB =ER69D           RANDOM OVF
067360 316 377 377 ALL$     JSB =LPOINT
067363 140 271 314          LDMI 40,=PTR2
067366 377
067367 146 040 305          SBM 46,40            #BYTES/REC - LOG. PTR.
067372 311 004 000          CMM 46,=4,0          ADVANCE ON HEADER ONLY
067375 372 326              JNC EOR$
067377 316 046 160          JSB =S$HDR           HANDLE STRING HEADER
067402                      DRP !56
067402 012 345              PUMD R#,+R12         SAVE BYTE COUNT
067404 055 243              STM  R#,R55          MAKE INTO 3 BYTE COUNT
067406 157 222              CLB  R57
067410 316 377 377          JSB =RSMEM-          RESERVE SPACE FOR STRING
067413 160 222              CLB  R60             BECOMES 3RD BYTE FOR LENGTH
067415 156 012 343          POMD R56,-R12        CLEAN OFF STACK
067420 370 042              JEN NOPE$            NO ROOM FOR STRING
067422 345                  PUMD 56,+12          PUSH STRING LENGTH
067423 165 345              PUMD 65,+12          PUSH STRING ADDRESS
067425 213                  DCM  R65             BACK UP FOR LOOP
067426 263 060 210          STMD 65,=RESDAT
067431 056 305              SBM  R65,R56         USING R60 AS 3RD BYTE
067433 263 161 210          STMD R65,=RESEND
067436 223                  CLM  R65
067437 213                  DCM  R65
067440 263 245 203          STMD R65,=NXTDAT     OFFSET FOR STRINGS=-1
067443 156                  DRP  56              SET DRP FOR GETDAT
067444 316 177 157          JSB =GETDAT          FETCH THE STRING
067447             *        JSB =TURN+           TURN DATA AROUND
067447 316 377 377          JSB =ROMJSB          LET SOMEONE STORE IT
067452 377 377              DEF STOST
067454 000                  OCT 0
067455 316 377 377          JSB =ROMJSB
067460 377 377              DEF RELMEM
067462 000                  OCT 0
067463 236                  RTN
067464             *
```

```
                READ STRING ROUTINE                                  <<<<<<<<
2073  067464 316 107 161 NOPE$  JSR =LRECOR          DO IT THIS WAY FOR RSPTR
2  4  067467 166 271 317         LDMI R66,=PTR2-+     GET L PTR VALUE
2074  067472 377
2075  067473 213                 DCM                  BACK UP THREE BYTES
2076  067474 213                 DCM
2077  067475 213                 DCM
2078  067476 104 251 061         GTU RSPTR            RESET POINTERS
2078  067501 155
```

```
>>>>>>>       READ NUMERIC                                              <<<<<<<<
     067502 044                   OCT 44
     067503               RDNUM.  BSS 0
     067503 016                   ARP 16
     067504 316 245 161           JSB =MSINHK
     067507 316 340 161           JSB =CKMSUS        CHECK MSUS
     067512 360 003               JMP RD#DSK
     067514 316 072 155 EORN      JSB =OUTLR
     067517 316 331 157 RD#DSK    JSB =R#COM!        COMMON TO READN/$
     067522 377 011               JRN ITSDAT
     067524 132 310 360           CMB 32,=360        EOR?
     067527 367 363               JZR EORN           YES, ADVANCE TO NXT REC
     067531 316 225 161 ER33D JSB =MSERR+
     067534 041                   OCT 33D            BAD DATA TYPE
     067535               ITSDAT  BSS 0
     067535 175 223               CLM  R75
     067537 045 243               STM  R75,R45
     067541 211                   ICM  R75
     067542 263 245 203           STMD R75,=NXTDAT   OFFSET FOR NUMERIC
     067545 250 010               LDB  R75,=10
     067547 112 243               STM  R12,R45       DATA ADDR
     067551 075 303               ADM  R12,R75       BUMP FOR SYSTEM
     067553 145 263 060           STMD R45,=RESDAT   SAVE DATA ADDR
     067556 210
     067557 303                   ADM  R45,R75
     067560 263 161 210           STMD R45,=RESEND
     067563 156 241               LDM  R56,R75       SET DATA LEN
     067565 316 177 157           JSB =GETDAT        GET 8 BYTES
     067570 316 377 377           JSB =ROMJSB        GIVE TO SYSTEM
     067573 377 377               DEF STOSV
     067575 000                   OCT 0
     067576 236                   RTN
     067577                        .
     067577 263 056 210 GETDAT STMD R#,=DATLEN   STORE DATA LENGTH
     067602 367 124               JZR EORERN         JIF NULL STRING
     067604 126 251 004           LDM R26,=4,0       FOR LENCHK NULL STRING
     067607 000
     067610 316 377 377           JSB =LENCHK
     067613 373 015               JCY INBNDR         SPAN LRECORD?
     067615                       DRP !56
     067615 076 305               SBM 56,76          YES IMPLIES STRING
     067617                 *************************************************
     067617                 * LENCHK HAS A CHECK AGAINST NULL STRINGS BEING PRINTED
     067617                 *** ON LOGICAL RECORD BOUNDRIES. UNFORTUNATELY, WHEN
     067617                 *** READ# string IS DONE, THE 3 BYTE HEADER IS STRIPPED
     067617                 *** BEFORE COMING TO THIS ROUTINE AND HENCE TO LENCHK.
     067617                 *** THEREFORE ANY STRING WITH A LENGTH OF 1 TO 3 BYTES
     067617                 *** PRINTED TO A LOGICAL BOUNDRY CAUSES PROBLEMS.THIS
     067617                 *** CHECK IS TO SEE IF THIS IS THE CASE.
     067617                 *************************************************
     067617                 *         CMMD R56,=DATLEN   SAME AS BEFORE
     067617                 *         JZR  INBNDR        JIF YES
     067617 263 056 210               STMD 56,=DATLEN
     067622 316 232 157               JSB =INBNDR        READ SEGMENT OF DATA
```

```
>>>>>>          READ NUMERIC                                                <<<<<<
067625 316 046 160              JSB =STHDR            HANDLE STRING HEADER
067630 360 345                  JMP GETDAT
067632
067632 126 261 056 INBHDR LDMD R26,=DATLEN
067635 210
067636 316 340 157              JSB  =MOVTST          CAN I DO ALL AT ONCE
067641 373 041                  JCY  INBHD1           JIF NO
067643 165 261 310              LDMD R65,=PTR1        SAVE PROGRAM POINTER
067646 377
067647 145 263 310              STMD R45,=PTR1        R45=>SOURCE(ASSIGN BUFF)
067652 377
067653 155 263 314 AREAD        STMD R55,=PTR2        R55=>DATA SINK
067656 377
067657 101 271 312              LDMI R*,=PTR1+
067662 377
067663 273 314 377              STMI R*,=PTR2
067666 155 323 245              ADMD R55,=NXTDAT
067671 203
067672 321 161 210              CMMD R55,=RESEND      ANY MORE TO MOVE?
067675 766 354                  JNZ  AREAD            JIF YES
067677 165 263 310              STMD R65,=PTR1        RESTORE PROGRAM POINTER
067702 377
067703 236                      RTN
067704
067704 316 022 155 INBHD1       JSB =RD+ADV           GET BYTE,ADVANCE
067707 316 067 160              JSB =RESSET           RESET PTR2 AND RESDAT
067712 132 272 314              STBI 32,=PTR2
067715 377
067716 126 261 056              LDMD 26,=DATLEN
067721 210
067722 213                      DCM 26                DECR. DATA COUNT
067723 263 056 210              STMD 26,=DATLEN
067726 766 354                  JNZ INBHD1
067730 236          EORERN      RTN
067731 316 377 377 R#COM!       JSB =ROMJSB
067734 377 377                  DEF R#COMN
067736 000                      OCT 0
067737 236                      RTN
067740
```

```
                        READ# AND PRINT# UTILITIES
  067740                MOVTST BSS 0                  TEST TO SEE IF MOVE CAN BE
  067740                *                             DONE WITHOUT RD+ADV OR WT+ADV
2164 067740 316 377 377         JSB  =PPOINT          GET PHYSICAL PTR
2165 067743 136 223             CLM  R36
2166 067745 270 316 377         LDBI R36,=PTR2+       LOAD PTR VALUE
2167 067750 026 303             ADM  R36,R26          ADD DATA LENGTH
2168 067752 311 000 001         CMM  R36,=0,1         OVF PHYSICAL REC?
2169 067755 373 360             RCY                   RTN IF YES
2170 067757 144 271 316         LDMI R44,=PTR2+       R46,=LOGICAL PTR
2170 067762 377
2171 067763 152 271 314         LDMI R52,=PTR2        R56=BYTES PER REC
2171 067766 377
2172 067767 146 303             ADM  R46,R26          ADD DATA LENGTH
2173 067771 056 301             CMM  R46,R56          OVF LOGICAL REC?
2174 067773 373 342             RCY                   RTN IF YES
2175 067775 144 273 315         STMI R44,=PTR2-       UPDATE LOGICAL PTR
2175 070000 377
2176 070001 145 223             CLM  R45
2177 070003 270 315 377         LDBI R45,=PTR2-       RETRIEVE PHYSICAL PTR
2178 070006 323 053 210         ADMD R45,=CURLOC      MAKE ABSOLUTE
2179 070011 313 034 000         ADM  R45,=34,0,0
2179 070014 000
2180 070015 100 250 047         LDB  R0,=47
2181 070020 155 261 060         LDMD R55,=RESDAT
2181 070023 210
2182 070024 136 272 314         STBI R36,=PTR2        STORE UPDATED POINTER
2182 070027 377
2183 070030 260 245 203         LDBD R36,=NXTDAT      DOING A STRING?
2184 070033 364 302             RNG                   RTN IF YES
2185 070035 250 010             LDB  R36,=10          FOR NUMBERS
2186 070037 262 245 203         STBD R36,=NXTDAT
2187 070042 100 250 040         LDB  R0,=40           8 BYTES AT A TIME
2188 070045 236                 RTN
2189 070046                     *-------------------------------------------------
2190 070046 316 022 155 S$HDR   JSB =RD+ADV           SKIP STRING TYPE
2191 070051 316 022 155         JSB =RD+ADV           GET STRING LENGTH
2192 070054 006 344             PUBD R#,+R5           SAVE LOW ORDER BYT
2193 070056 316 022 155         JSB =RD+ADV           GET HIGH ORDER BYTE
2194 070061 057 242             STB R#,R57            MOVE R32 TO R57
2195 070063 156 006 342         POBD R56,-R6          POP LOW ORDER BYTE
2196 070066 236                 RTN
2197 070067                     *-------------------------------------------------
2198 070067                     RESSET BSS 0
2199 070067 145 261 060         LDMD R45,=RESDAT
2199 070072 210
2200 070073 263 314 377         STMD R45,=PTR2
2201 070076 323 245 203         ADMD R45,=NXTDAT
2202 070101 263 060 210         STMD R45,=RESDAT
2203 070104 236                 RTN
```

```
                        READ ARRAY NUMERIC
070105                  ***     READ ARRAY ATTRIBUTES **********************
070105 044              .       OCT 44
070106                  **************************************************
070106                  RDARR.  BSS  0
070106 017              .       ARF  17
070107 316 245 161      .       JSB  =MSINHK
070112 316 340 161      .       JSB  =CKMSUS        CHECK MSUS
070115 316 243 160      .       JSB  =TRUCAL       CALCULATE TRUE ARR SIZE
070120 006 345          .       PUMD R#,+R5         SAVE ADDRS,NAME
070122 316 377 377      .       JSB  =RDARRX         DOES THE ABOVE CODE
070125 316 145 160      .       JSB  =LENCA!         CALC DATA LENGTH
070130 166 006 345      .       PUMD R66,+R6         SAVE DATA LENGTH
070133 155 345          .       PUMD R55,+R6         SAVE IT
070135 316 117 157      NXTELR  JSB  =RD#DSK         READ A NUMERIC
070140 316 377 377      .       JSB  =RDARRC         DOES THE ABOVE CODE
070143 260 370          .       JMP  NXTELR          READ NEXT ELEMENT
070145                  LENCA!  BSS  0
070145 147 046 242      .       STB  R47,R46
070150 317 060          .       ANM  R47,=60
070152 262 015 204      .       STBD R47,=DIMFLG
070155 316 377 377      .       JSB  =ELSZ           CALC DATA LEN
070160 147 222          .       CLB  R47
070162 262 015 204      .       STBD R47,=DIMFLG     CLR DIMFLG FOR ALLOCATION
070165 236              .       RTN
```

```
                        PRINT ARRAY ELEMENT
2230 070166                     *****  PRINT ARRAY ATTRIBUTE TABLE  ****************
2231 070166 036                         OCT 36
2232 070167                     *************************************************
2233 070167                     ***THIS CODE NEEDS SOMETHING FOR REMOTE VARIABLES*****
2234 070167                     PRARR.  BSS 0
2235 070167 013                         ARP 13
2236 070170 316 245 161                 JSB =MSINHK
2237 070173 316 340 161                 JSB =CKMSUS             CHECK MSUS
2238 070176 316 243 160                 JSB =TRUCAL           CALCULATE TRUE ARRAY SIZE
2239 070201 006 345                     PUMD R#,+R5             SAVE ADDR,NAME
2240 070203 175 041 241                 LDM 75,41               1ST ELEM ADDRESS
2241 070206 316 145 160                 JSB =LENCA!             CALC DATA LENGTH
2242 070211 166 006 345                 PUMD 66,+R5             SAVE DATA LENGTH
2243 070214 155 345                     PUMD 55,+R5             SAVE IT
2244 070216 175 263 314 NXTELP  STMD R75,=PTR2          FOR FNUM
2244 070221 377
2245 070222 316 377 377                 JSB =ROMJSB             FETCH NUMBER (R60)
2246 070225                     *-----  DEF FNUM
2247 070225 377 377                     DEF FETSVX              FETCH NUM & CHECK FOR NULL
2248 070227 000                         OCT 0
2249 070230 140 060 241                 LDM 40,60               FOR PR#DSK
2250 070233 316 232 156                 JSB =PR#DSK
2251 070236 316 377 377                 JSB =PRARRC             DOES THE ABOVE CODE
2252 070241 260 353                     JMP NXTELP
2253 070243                     *-----------------------------------------------
2254 070243                     *CALCULATE REDIMENSIONED ARRAY SIZE FOR READ,PRINT
2255 070243 141 012 343 TRUCAL  POMD R41,-R12           R41=ELEM ADDR,R47=NAME
2256 070246                     *Fix option base depending on CALCMODE, and
2257 070246                     * PROGRAM or COMMON array variable, as in matrix RON.
2258 070246 116 006 344                 PUBD R16,+R5            SAVE R16
2259 070251 155 261 022                 LDMD 55,=NXTMEM
2259 070254 200
2260 070255 041 301                     CMM R55,R41 PROG OR COMMON VAR?
2261 070257 373 002                     JCY CALCVB              JIF CALC VARIABLE
2262 070261 116 222                     CLB R16
2263 070263 316 377 377 CALCVB  JSB =CUROPT             COMPUTE CURRENT OPT BASE
2264 070266 116 006 342                 POBD R16,-R5            RESTORE R16
2265 070271                     *-----------END OF CURRENT OPTION BASE FIX-------------
2266 070271 155 041 241                 LDM  R55,R41            LOAD ADDR
2267 070274 263 314 377                 STMD R55,=PTR2
2268 070277 151 271 314                 LDMI R51,=PTR2          ARRAY COL,ROW,LEN
2268 070302 377
2269 070303 161 223                     CLM  R61
2270 070305 250 004                     LDB  R61,=4              DONE FOR STRING ARRY
2271 070307 141                         ORP 41    FOR R#,R# AFTER TRUCAL CALL
2272 070310 236                         RTN
```

```
>>>>>>      READ# STRING ARRAY
070311                 *****  READ STRING ARRAY ATTRIBUTES  ***************<<<<<<
070311 044                            OCT 44
070312                 ******************************************************
070312                 RDARR$         BSS 0
070312 021                            ARP 21
070313 316 245 161                    JSB =MSINHK
070316 316 340 161                    JSB =CKMSUS
070321 316 243 160                    JSB =TRUCAL
070324 061 305                        SBM  R#,R61         BACKUP TO 1ST ELEM
070326 006 345                        PUMD R#,+R6         SAVE ADDR,NAME
070330 316 377 377                    JSB =RDARX+         TST TRACE
070333 136 271 315                    LDMI R36,=PTR2-     GET MAX ELEM LEN
070336 377
070337 053 243                        STM  R36,R53        SAVE FOR STOST
070341 211                            ICM  R36
070342 211                            ICM                 ACCOUNT FOR 2 BYTE LEN
070343 066 345                        PUMD R36,+R6        SAVE MAX LENGTH
070345 222                            CLB
070346 344                            PUBD R36,+R6        MAKE 3 BYTES LONG
070347 155 345                        PUMD R55,+R6
070351 041 241                        LDM  R55,R41        COPY BASE ADDR
070353 153 012 345                    PUMD R53,+R12       MAX LEN,BASE ADDR
070356 345                            PUMD R53,+R12       SINK LEN,SINK ADDR
070357 316 330 156 NXTR$              JSB =RD#$           READ STRING
070362 316 377 377                    JSB =RDAR$C         DO DATA MANIPULATION
070365 760 370                        JMP NXTR$           IF RTN READ ANOTHER
```

```
                        PRINT# STRING ARRAY
070367                        **** PRINT STRING ARRAY ATTRIBUTES *************
070367 036                         OCT 36
070370                        ****************************************************
070370                        PRARR$ BSS 0
070370 015                         ARP 15
070371 316 245 161                 JSB =MSINHK
070374 316 340 161                 JSB =CKMSUS
070377 316 243 160                 JSB =TRUCAL
070402 061 305                     SBM  R#,R61            BACKUP TO 1ST ELEM
070404 066 345                     PUMD R#,+R6            SAVE ADDR,NAME
070406 175 041 241                 LDM  R75,R41
070411 263 314 377                 STMD R75,=PTR2
070414 144 271 314                 LDMI R44,=PTR2         GET MAX ELEM LEN
070417 377
070420 146 211                     ICM  R46
070422 211                         ICM                    ADJUST 2 EXTRA
070423 066 345                     PUMD R46,+R6
070425 222                         CLB
070426 344                         PUBD R46,+R6
070427 155 345                     PUMD R55,+R6
070431 146 271 314 NXTP$           LDMI R46,=PTR2
070434 377
070435 311 377 377                 CMM  R46,=377,377       IS IT UNDEFINED
070440 366 001                     JNZ  NXTR46             NO KEEP CURRENT LENGTH
070442 223                         CLM  R46                MAKE NULL STRING
070443                    NXTR46   ARP  !46
070443 043 243                     STM  R46,R43            PLACE IN CORRECT REG
070445 175 045 243                 STM  R75,R45
070450 143 012 345                 PUMD R43,+R12
070453 316 262 155                 JSB =PRSTR.
070456 316 377 377                 JSB =PRAR$C
070461 360 346                     JMP NXTP$               IF RTN, PRINT ANOTHER
```

```
<<<<         PRINT# EOL                                                               <<<<<<<<
070463                    ****  PRINT EOL ATTRIBUTES TABLE  *****************
070463 035                       OCT 35
070464                    ***************************************************************
070464                    PREOL.   BSS 0
070464 022                       APP 22
070465 316 245 161               JSB =MSINHK
070470 116 220                   TSB R16                  CALC MODE?
070472 363 003                   JEV BUFFEX               EXIT IF NO
070474 316 261 167               JSB =MSDUMP              DUMP MS BUFFERS
070477 236                BUFFEX   RTN
070500
070500
070500                    ***********************************************************************
070500                    PRECOR   BSS 0
070500 316 377 377               JSB  =PPOINT
070503 270 316 377               LDBI R#,=PTR2+        NOW POINTING AT PHY. REC
070506 236                       RTN
070507
070507                    LRECOR   BSS 0
070507 155 261 053               LDMD R55,=CURLOC
070512 210
070513 313 014 000               ADM  R55,=14,0,0     OFFSET FOR LOGICAL.REC
070516 000
070517 263 314 377               STMD R55,=PTR2
070522 236                       RTN
070523
```

```
070523                          ***************************************************
070523                          *    UTILITY ROUTINES USED BY THIS ROM            *
070523                          ***************************************************
070523                          *THE FOLLOWING ROUTINES ARE FOR PARSING:
070523                          ***************************************************
070523                          GETNUM  BSS 0
070523 316 377 377                      JSB =ROMJSB
070526 377 377                          DEF NUMVA+          GET A NUMBER
070530 000                              OCT 0
070531 370 344                  ERR???  JEN BUFFEX          FOUND NUMBER
070533 316 163 161              ER91D+  JSB =SYSERP
070536 130                              OCT 88D             MISSING PARAM
070537                          *-------------------------------------------------
070537                          GETSTR  BSS 0
070537 316 377 377                      JSB =ROMJSB
070542 377 377                          DEF STREX+          GET A STRING
070544 000                              OCT 0
070545 360 362                          JMP ERR???          FOUND STRING
070547                          *-------------------------------------------------
070547                          TOTOK   EQU 244             SYSTEM 'TO' TOKEN
070547                          GETTO   BSS 0
070547 114 310 244                      CMB R14,=TOTOK      FIND 'TO'
070552 367 323                  ER???2  JZR BUFFEX          'TO' FOUND
070554 360 355                          JMP ER91D+
070556                          *-------------------------------------------------
070556                          GETCOM  BSS 0
070556 114 310 054                      CMB R14,=COMMA            FIND COMMA
070561 360 367                          JMP ER???2          COMMA FOUND
070563                          *-------------------------------------------------
070563                          SYSERP  BSS 0
070563 143 006 343                      POMD R43,-R6        2 RTN ADDRESS, 1 TOKEN#
070566 360 003                          JMP SYS+1           NOW FIX STACK
070570 146 006 343              SYSER+  POMD R46,-R6        TOKEN # AND 1 RTN ADDR
070573 130 251 377              SYS+1   LDM R30,=ROMRTN      ERROR+ RETURNS HERE
070576 377
070577 106 345                          PUMD R30,+R6        PUT ON RETURN STACK
070601 146 345                          PUMD R46,+R6        RESTORE ERROR # ADDR
070603 360 034                          JMP ERR!            RETURN TO ROMRTN VIA ERROR+
070605                          *-------------------------------------------------
070605 044                              OCT 44
070606 316 170 161              ERRORX  JSB =SYSER+
070611 017                              OCT 15D
070612                          *-------------------------------------------------
070612                          *THE FOLLOWING ROUTINES ARE ERROR CALLS FROM
070612                          * RUNTIME:
070612                          *-------------------------------------------------
070612 130 260 123              MSERR   LDBD R30,=ERRORS     GET ERROR FLAG
070615 200
070616 366 005                          JNZ MSERR+          SKIP IF ERROR ALREADY
070620 250 320                          LDB R#,=MYROM#      GET MASS STOR. ROM#
070622 262 120 200                      STBD R#,=ERRROM     FOR ERROR ROUTINE
070625 136 006 343              MSERR+  POMD R36,-R6        GET ERROR # ADDR
070630 106 261 066                      LDMD R6,=SAVER6     CAUSE QUICK EXIT
```

```
                >>>>        MAS: STORAGE ROM UTILITY ROUTINES                    <<<<<<<<
070633 210
070634 136 345              PUMD R3S,+R5        RESTORE ERROR # ADDR
070636 316 257 150          JSB  =CLRFLG        CLEAR LOAD/STORE SRST FLG
070641 104 251 377 ERR:     GTO ERROR+          RETURN TO SYSTEM
070644 377
070645                      *---------------------------------------------------
070645                      *MISC. RUNTIME ROUTINES:
070645                      *---------------------------------------------------
070645                      MSINHK  BSS 0
070645 232                          SAD                 SAVE ARP FLAG FOR HOOK
070646 316 364 207                  JSB =MSHIGH         JMP TO HIGH LEVEL HOOK
070651 237                          PAD                 RESTORE R6 IF WE RTN
070652                      MSIN    BSS 0
070652                      *called by all runtime routines for initialization
070652 316 277 161                  JSB =INIT14
070655 112 014 247                  STMD R12,R14        SAVE PARAMS PTR
070660 230                          BIN                 FOR ALL LOCAL ARITHMETIC
070661 130 222                      CLB R30             START OUT NOT
070663 262 331 203                  STBD R30,=SCRTYP    ...NOT SCRATCH TYPE
070666 130 006 241 ININIT   LDM R30,R5          GET SP
070671                      *ININIT called by INITIT to set SAVER5, R14
070671                      *but cannot set scratchtype(=SAVNAM in tape dscan)!
070671 130 213              DEC30   DCM R30             ADJUST FOR EXTRA JSB
070673 213                          DCM R30             ADJUST FOR 2 BYTES
070674 263 066 210                  STMD R30,=SAVER6     FOR ERROR RETURNS
070677 114 261 012 INIT14   LDMD R14,=MSBASE  MS RAM BASE ADDR
070702 207
070703 236                          RTN
070704                      *---------------------------------------------------
070704 316 377 377 GETNA!   JSB =ROMJSB
070707 377 377              DEF GETNAM
070711 000                  OCT 0
070712 236                  RTN
070713                      *---------------------------------------------------
070713                      TAPOS-  BSS 0
070713 232                          SAD                 SAVE ARP FLAG
070714 316 364 207                  JSB =MSHIGH         JMP TO HIGH LEVEL HOOK
070717 237                          PAD                 RESTORE R6 IF WE RTN
070720                      TAPOS   BSS 0
070720 316 252 161                  JSB =MSIN           INIT SYSTEM
070723 316 271 161                  JSB =DEC30          ADJUST FOR EXTRA RTN
070726                      TAPOS+  BSS 0
070726 316 007 143                  JSB =DCDZER         DECODE FILE
070731                              DRP !44
070731 220                          TSB R44
070732 766 015                      JNZ GOOD
070734 316 212 161 ERR20D   JSB =MSERR           "NO MSUS ON LINE"
070737 024                  OCT 20D
070740                      *---------------------------------------------------
070740                      CKMSUS  BSS 0
070740 230                          BIN
070741 316 352 161                  JSB =BUFA.M           CURLOC+A.MSUS
070744                              DRP !55
```

```
>>>>>        MASS STORAGE ROM UTILITY ROUTINES                                <<<<<<
2457 070744 270 314 377           LDBI R55,=PTR2        GET MSUS FLAG
2458 070747 367 763               JZR ERR20D             JIF NO MSUS
2459 070751 236           GOOD    RTN
2460 070752               *---------------------------------------------------------
2461 070752               BUFA.M  BSS 0
2462 070752 155 261 053           LDMD R55,=CURLOC
2462 070755 210
2463 070756 313 022 000           ADM  R55,=A.MSUS
2464 070761 000                   OCT  0
2465 070762 263 314 377           STMD R55,=PTR2
2466 070765 236                   RTN
```

```
>>>>          FETCH AND PARSE FILE SPECIFIER                          <<<<<<<
070766                    *GETS A STRING OFF THE STACK AND PARSES AS A
070766                    *FILE SPECIFIER OR MSUS ADDR/VOLUME LABEL
070766                    *RETURNS FILENAME IN FNAME AND TAPE/DISC FLAG
070766                    *AND ADDRESS IN R44-47 AS PER ROUTINE MS2AD.
070766                    *DOES AUTOMATIC VOLUME SEARCH IF SPECIFIER CONTAINS
070766                    *A VOLUME LABEL.
070766                    *RETURNS DEFAULT MSUS IF SPECIFIER CONTAINS NEITHER
070766                    *VOLUME LABEL NOR MSUS ADDRESS.
070766                    *ON INPUT R40 IS USED TO FLAG FILENAME OR MSUS:
070766                    *         0: AT LEAST 1 OF FILE NAME OR MSUS
070766                    *         1: FILENAME ONLY (I.E. NO MSUS ALLOWED)
070766                    *         2: MSUS ONLY (I.E. NO FILENAME ALLOWED)
070766                    *
070766                    DECODE  BSS 0
070766 230                        BIN               FOR ADDS
070767 175 012 343                POMD R75,-R12     POP STRING ADDRESS
070772 263 314 377                STMD R75,=PTR2    STORE STRING ADDR
070775 132 343                    POMD R32,-R12     POP STRING LENGTH
070777 366 003                    JNZ NAMEOK        CAN'T BE NULL STRING
071001 316 302 152                JSB =ER99D       INVALID PARAM
071004                    *
071004 143 261 377 NAMEOK  LDMD R43,=BLANKS   GET 5 BLANKS
071007 377
071010 263 103 207                STMD R43,=FNAM   STORE FIRST 5 OF 10
071013 263 110 207                STMD R43,=FNAM+5 STORE LAST 5 OF 10
071016 235                        CLE               SET VOL LABEL FLAG
071017 137 250 012                LDB R37,=10D      MAX FILENAME LENGTH
071022 126 251 103                LDM R26,=FNAM     STARTING ADDRESS
071025 207
071026 132 210                    ICB R32           ADJUST ST LENGTH
071030 132 212            NXTCH   DCB R32           CHARACTER COUNT
071032 367 110                    JZR ITSDEF        JIF STRING EXHAUSTED
071034 130 270 315                LDBI R30,=PTR2-   GET NEXT CHAR
071037 377
071040 310 056                    CMB R30,=PERIOD   BEGIN VOL LABEL?
071042 367 030                    JZR ITSVOL        JIF VOL SPECIFIER
071044 310 072                    CMB R30,=COLON    BEGIN MSU SPECIFIER?
071046 367 023                    JZR ITSMSU        JIF MSU
071050                    *OTHERWISE ITS ANOTHER CHAR IN FILENAME:
071050 140 310 002                CMB R40,=2        MSUS ONLY ALLOWED?
071053 366 004                    JNZ FNAMOK        JIF FILENAME OK
071055 316 212 161 ERR21D  JSB =MSERR
071060 032                        OCT 26D           MSUS EXPECTED
071061                    FNAMOK  BSS 0
071061 137 220                    TSB R37           CHECK FILENAME OVERFLO
071063 367 343                    JZR NXTCH         TOO LONG NAMES TRUNCATED
071065 212                        DCB R37           ANOTHER CHAR TO ADD
071066 130 026 344                PUBD R30,+R26     STORE IT
071071 360 335                    JMP NXTCH         NEXT CHARACTER...
071073                    *
071073 234                ITSMSU  ICE               MSUS FLAG
071074                    ITSVOL  BSS 0
071074 140 212                    DCB R40           FILENAME ONLY ALLOWED?
```

```
2519          ;FETCH AND PARSE FILE SPECIFIER                    <<<<<<<<
2520 071076 766 003           JNZ MSUSOK          JIF MSUS OK
2521 071100 316 244 151       JSB =ER570           FILE NOT FOUND
2520 071103                MSUSOK  BSS 0
2521 071103                *MSUS STRING NOW ADDRESSED BY PTR2
2522 071103                *MSUS CHAR COUNT (+1) IN R32
2523 071103 132 213           DCM R32             ADJUST LEN COUNT
2524 071105 370 042           JEN MS2AD           JIF MSUS ADDRESS
2525 071107
2526 071107                *SYSTEM ROUTINE GETS BLANKFILLED 6-CHR STR
2527 071107 112 313 012       SBM  R12,=12,0      FOR RENAME
2527 071112 000
2528 071113 132 012 345       PUMD R32,+R12       STR ONTO STAK
2529 071116 175 261 314       LDMD R75,=PTR2      GET ADDR OF STRING
2529 071121 377
2530 071122 345               PUMD R75,+R12
2531 071123 316 304 161       JSB =GETNA!         -R12 BLANKFILLED TO R42
2532 071126 137 006 344       PUBD R37,+R5        SAVE WAS-MSUS FLAG
2533 071131 316 277 176       JSB =VOL2AD         PERFORM VOL SEARCH
2534 071134 137 006 342       POBD R37,-R5        RESTORE WAS-MSUS FLAG
2535 071137 112 315 012       SBM  R12,=12,0      SUB EXTRA (FOR RENAME)
2535 071142 000
2536 071143 236               RTN
2537 071144 144 261 977 ITSDEF  LDMD R44,=DEFMSU DEFAULT MSUS
2537 071147 207
2538 071150 236               RTN
```

```
                        MSUS TO INTERNAL ADDRESS
071151                  MSUAD   BSS 0
071151                  *TAKES ASCII MSUS ADDR AND RETURNS 4 BYTES:
071151                  * R44: DISC (1) FLAG
071151                  * R45: 0<=SELECT CODE<=3
071151                  * R46: 0<=CONTROLLER ADDRESS<=7
071151                  * R47: 0<=UNIT#<=3
071151                  *EXAMPLES: "M321" RETURNS 1,0,2,1
071151                  *          "n1023"  ->    1,7,2,3
071151                  *
071151 144 223                  CLM R44         DISC DEF.ADDR.
071153 210                      ICB R44         FLAG DISC
071154 130 270 315              LDBI R30,=PTR2-  1ST CHAR OF MSUS ADDR
071157 377
071160 132 314 004              SBB R32,=4      3 DIGITS (+ 1 LETTER)?
071163 367 014                  JZR DIG3        YES
071165 212                      DCB R32         4 DIGITS?
071166 766 265                  JNZ ERR21D       BAD MSUS
071170 316 216 162              JSB =NXTDIG     S.C. TENS PLACE
071173 212                      DCB R#          TEST FOR TEN
071174 766 257                  JNZ ERR21D       S.C.>10
071176 145 250 012              LDB R45,=10D    LOAD TEN
071201 316 216 162 DIG3         JSB =NXTDIG     S.C. ONES PLACE
071204 145 047 302              ADB R45,R47     TENS*10+ONES
071207 314 003                  SBB R45,=3      10>>7,3>>0
071211 316 216 162              JSB =NXTDIG     CONTROLLER#
071214 046 242                  STB R#,R46
071216                  *..FALL THRU TO NXTDIG FOR UNIT#
071216                  NXTDIG  BSS 0
071216                  *NXTDIG GETS NEXT DIGIT AND BINARYS IT INTO R47
071216 147 270 315              LDBI R47,=PTR2-  GET NEXT CHAR
071221 377
071222 314 060                  SBB R47,=ZERO   CHAR < '0'
071224 764 227                  JNG ERR21D       CHAR IS < '0'
071226 310 012                  CMB R#,=10D     CHAR ABOVE '9'?
071230 765 223                  JPS ERR21D       CHAR IS > '9'
071232 236                      RTN
071233                  *
071233                  ZERO    EQU 48D         ASCII '0'
```

```
ATMW 03/14/81 ================================================================
ITEM    LOC   OBJECT CODE  SRC=LOCMAS OBJ=GENGMA   9/11/1981  3:03 PM  PG 77
================================================================================
                    PACK [(msus)]
 071233 241                     OCT 241
 071234                 PACK.   BSS 0
 071234 626                     ARP 26
 071235 316 245 161             JSB =MSINHK          RUNTIME INIT
 071240 316 256 142             JSB =GETMSU          GET MSUS
 071243 266 003                 JNZ PACKDK           JIF DISC
 071245 316 334 161             JSB =ERR20D          NO MSUS
 071250                 PACKDK  BSS 0
 071250                         ORP 144
 071250 263 202 207             STMD R44,=SRCMSU
 071253 263 206 207             STMD R44,=DSTMSU
 071256                 *1. FIND THE FIRST HOLE. ERROR IF NONE
 071256 316 234 146             JSB =HOLE#1          1ST HOLE
 071261                 *2. PACK THE FILES, LEAVING DIRECTORY INTACT
 071261 316 044 147 PFILES      JSB =NXTENT          NEXT DIR. ENTRY
 071264 370 065                 JEN PACKDN           JIF ALL FILES PACKED
 071266 316 336 143             JSB =GETTYP          FILE TYPE
 071271 367 366                 JZR PFILES           SKIP HOLE FILES
 071273 210                     ICB R#               LAST FILE?
 071274 367 055                 JZR PACKDN           JIF FILES ALL PACKED
 071276                 *MOVE A FILE
 071276 316 132 147             JSB =LDFBEG
 071301 130 263 170             STMD R30,=SRCORG  FOR COPY
 071304 207
 071305 316 165 147             JSB =LDRECS
 071310 130 263 276             STMD R30,=CR.CNT     FOR MVFILE ROUTINE
 071313 203
 071314 132 261 154             LDMD R32,=CUMREC DEST. ORIGIN
 071317 207
 071320 263 172 207             STMD R32,=DSTORG  FOR COPY
 071323 316 121 147             JSB =STFBEG
 071326 132 030 303             ADM R32,R30          UPDATE CUMREC
 071331 263 154 207             STMD R32,=CUMREC  CUMREC:=CUMREC+CPYSIZ
 071334 316 223 163             JSB =WRTDIR          REWRITE DIR
 071337 316 206 164             JSB =MVFILE          MOVE FILE CONTENTS
 071342                 *RESTORE DIR INFO FOR CALL TO NXTENT
 071342 132 261 152             LDMD R32,=DADDR DIR SECTOR#
 071345 207
 071346 316 120 176             JSB =GETBUF
 071351 260 306                 JMP PFILES           DO SOME MORE
 071353                 *--------------------------------------------------
 071353                 PACKDN  BSS 0
 071353                 *ALL FILES NOW RELOCATED--DELETE DIR. HOLES
 071353 316 234 146             JSB =HOLE#1          BACK TO 1ST HOLE
 071356 145 261 151             LDMD R45,=DENTRY HOLE DIR INFO
 071361 207
 071362 263 174 207             STMD R45,=DENTD  DEST. DIR ENTRY
 071365 316 044 147 NEWSRC      JSB =NXTENT          FILE TO MOVE
 071370 145 261 151             LDMD R45,=DENTRY  ENTRY TO BE MOVED
 071373 207
 071374 263 177 207             STMD R45,=DENTS   SOURCE DIR ENTRY
 071377 370 122                 JEN PDELET           PURGE END HOLES
 071401 316 336 143             JSB =GETTYP          FILE TYPE
```

```
A??? 03/14/81:==================================================================
ITEM    LOC   OBJECT CODE  SRC=LOGMAS OBJ=GENSMA   9/11/1981  3:03 PM  PG 78
================================================================================
<<<<<<       PACK [<msus>]                                             <<<<<<<<
2?   071404 367 357               JZR NEWSRC        .DONT COPY HOLES!
2?6  071406 210                   ICB R#             LAST FILE?
2627 071407 367 112               JZR PDELET         NOTHING TO PACK
2628 071411 316 162 163           JSB =GETINF        GET SOURCE DIR ENTRY
2629 071414                *RESTORE FIRST DEST DIR SECTOR
2630 071414 132 261 175           LDMD R32,=DADDD   DEST DIR SECTOR
2630 071417 207
2631 071420 316 120 176           JSB =GETBUF        GET IT
2632 071423 145 261 174           LDMD R45,=DENTD   RESTORE DIR PTRS
2632 071426 207
2633 071427 263 151 207           STMD R45,=DENTRY
2634 071432 316 106 147           JSB =DIRPTR         R36 FROM DENTRY,DADDR
2635 071435 316 206 163           JSB =PUTINF         TRANSFER DIR ENTRY
2636 071440 316 056 163 NXTTRN    JSB =NXTSRC         NEXT SOURCE DIR ENTRY
2637 071443 316 162 163           JSB =GETINF         PRTBUF:=SRC DIR INFO
2638 071446 316 132 163           JSB =NXTDST         NEXT DEST DIR ENTRY
2639 071451 316 206 163           JSB =PUTINF         DST DIR INFO:=PRTBUF
2640 071454 360 362               JMP NXTTRN          LOOP TERMINATES IN NXTSRC
2641 071456                *-----------------------------------------------
26?  071456 145 261 177 NXTSRC    LDMD R45,=DENTS   RESTORE DIR PTRS
2?2  071461 207
2643 071462 263 151 207           STMD R45,=DENTRY
2644 071465 132 046 241           LDM R32,R46         RESTORE DIR SECTOR
2645 071470 316 120 176           JSB =GETBUF
2646 071473 316 044 147 AD2NTF    JSB =NXTENT         ADVANCE TO NXT FILE
2647 071476 370 020               JEN PAKDON          DIR HOLES ALL MOVED
2648 071500 316 336 143           JSB =GETTYP         FILE TYPE
2649 071503 367 366               JZR AD2NTF          HOLE ENTRIES NOT MOVED
2650 071505 210                   ICB R#              LAST FILE?
2651 071506 367 010               JZR PAKDON       VALID ENTRIES ALL MOVED
2652 071510                *NEXT SOURCE ENTRY TO MOVE:
2653 071510 145 261 151           LDMD R45,=DENTRY   SAVE ...
2653 071513 207
2654 071514 263 177 207           STMD R45,=DENTS    ... NEW SOURCE PTRS
2655 071517 236                   RTN
2656 071520 130 006 343 PAKDON    POMD R30,-R6        FOR EXTRA JSB
2657 071523                *MAKE THE LAST DEST FILE THE real LAST!
2658 071523                *DELETE HOLES NOW LOCATED AT END OF DIR
26?  071523 316 234 146 PDELET    JSB =HOLE#1         FIND FIRST HOLE
2??0 071526 316 256 151           JSB =PUR+           REMAINING ENTRIES 'LAST'ED
2??1 071531 236                   RTN
2662 071532                *-----------------------------------------------
2663 071532             NXTDST    BSS 0  MOVE TO NEXT DEST DIR ENTRY
2664 071532 145 261 174           LDMD R45,=DENTD   RESTORE DEST DIR PTRS
2664 071535 207
2665 071536 263 151 207           STMD R45,=DENTRY
2666 071541 132 046 241           LDM R32,R46
2667 071544 316 120 176           JSB =GETBUF         RESTORE DIR SECTOR
2668 071547 316 044 147           JSB =NXTENT         MOVE TO NEXT FILE
2669 071552 145 261 151           LDMD R45,=DENTRY   SAVE NEW DEST PTRS
2669 071555 207
2670 071556 263 174 207           STMD R45,=DENTD
2671 071561 236                   RTN
```

```
071562             *---------------------------------------------------------
071562             GETINF  BSS 0              SOURCE DIR TO BUFFER
071562 136 024 243         STM R36,R24
071565 114 026 243         STM R14,R26
071570 122 251 040         LDM R22,=32D,0        32 BYTES/DIR ENTRY
071573 000
071574 316 377 377         JSB =MOVUP            PRTBUF:=SRC DIR INFO
071577 130 222             CLB R30               'HOLE' FILE TYPE
071601 316 374 143         JSB =PUTTYP           INTO DIR
071604 260 015             JMP WRTDIR
071606             *---------------------------------------------------------
071606             PUTINF  BSS 0             BUFFER TO DIRECTORY
071606 114 024 243         STM  R14,R24
071611 136 026 243         STM R36,R26
071614 122 251 040         LDM R22,=32D,0       32 BYTES/DIR ENTRY
071617 000
071620 316 377 377         JSB =MOVUP           DST DIR INFO:=PRTBUF
071623             *FALL THRU TO WRTDIR
071623             *-----------------------------------------------
071623             WRTDIR  BSS 0
071623             *REPLACES ONE DIRECTORY SECTOR
071623             * RECBUF: CAPR ADDR OF MODIFIED SECTOR
071623             * DADDR: DISC ADDRESS OF DIR SECTOR TO REPLACE
071623             *Note: interchange format requires volume field
071623             * (in each dir entry) to indicate file is
071623             * contained entirely in this volume.
071623 316 106 147         JSB =DIRPTR         R36:=DIR PTR
071626 132 251 200         LDM R32,=200,1      ENTIRE FILE THIS VOLUME
071631 001
071632 036 267 032         STMD R32,X36,D.VOL
071635 000
071636 261 152 207         LDMD R32,=DADDR   DISC ADDRESS
071641 316 032 176         JSB =PUTBUF         REWRITE THE DIR
071644 236                 RTN
```

```
      ???>      COPY source, destination                                   <<<<<<<
071645               *Depending on 'source', COPY will perform one of:
071645               *1.Volume copy. If 'source' is an msus (i.e. filename
071645               *  is blank), COPY will expect dest. to be an msus
071645               *  and copy an entire medium.
071645               *2.File copy. If 'source' is a file specifier (i.e.
071645               *  filename non-blank), COPY will expect dest. to be
071645               *  the same and copy only a single file.
071645 241                   OCT 241
071646               MSCPY.  BSS 0
071646 027                   ARP 27
071647 316 245 161           JSB =MSINHK         RUNTIME INIT
071652 140 250 020           LDB R40,=DATATY
071655 262 271 203           STBD R40,=FILTYP    FOR MTSCAN
071660               *       CLB R40             ALLOW ANY OLD FILE SPEC.
071660 316 007 143           JSB =DCDZER         DECODE DEST.  FILE
071663                       DRP !44
071663 263 206 207           STMD R44,=DSTMSU  STASH DEST MSUS
071666               *CHECK FOR VOLUME COPY
071666 137 310 012           CMB R37,=10D        DCDFIL: WAS-MSUS?
071671 366 076               JNZ FILCPY          IT WAS NOT
071673               *ITS A VOLUME COPY
071673 316 366 142           JSB =DCD-           PARSE 2ND MSUS
071676                       DRP !44
071676 263 202 207           STMD R44,=SRCMSU SAVE SRC. MSUS
071701               *GET FIRST FILE TO COPY
071701 316 360 146           JSB =GETDIR         GET 1ST DISC FILE
071704 316 317 164           JSB =SKHOL+         SKIP HOLES
071707 370 057               JEN VLDONE          JIF END OF DIR
071711               ONEFIL  DRP !45
071711 014 247               STMD R45,R14         SAVE DIR INFO
071713               *COPY ONE FILE
071713 143 036 245           LDMD R43,R36
071716 263 103 207           STMD R43,=FNAM   STORE FOR DIR SCAN
071721 263 115 207           STMD R43,=GNAM    AND COPY
071724 265 005 000           LDMD R43,X36,SEC1/2
071727 263 110 207           STMD R43,=FNAM+5
071732 263 122 207           STMD R43,=GNAM+5
071735 144 261 206           LDMD R44,=DSTMSU  GET DEST MSUS
071740 207
071741 263 160 207           STMD R44,=ACTMSU   MAKE ACTIVE
071744 316 162 146           JSB  =DIRSCN           DOES FILE ALREADY EXIST?
071747 371 025               JEZ  ERR63D            JIF ERROR DUPL. NAME
071751 144 261 202           LDMD R44,=SRCMSU  GO BACK TO SOURCE
071754 207
071755 263 160 207           STMD R44,=ACTMSU
071760 316 024 164           JSB =FILSCN         PERFORM FILE COPY
071763 316 267 164           JSB =NXTFIL         SCAN NEXT SRC FILE
071766 371 321               JEZ ONEFIL          COPY ANOTHER FILE
071770 236           VLDONE  RTN
071771               *ITS A FILE COPY--------------------------------------
071771               FILCPY  BSS 0
071771 316 162 146           JSB =DIRSCN         SCAN FOR DEST. NAME
071774 370 003               JEN FILNOT          FILE NOT FOUND
```

```
                        COPY source, destination
071776 316 371 151 ERR63D  JSB =ER63D          DUPLICATE FILE NAME
072001 143 261 103 FILNOT  LDMD R43,=FNAM  SAVE DEST NAME
2755 072004 207
2756 072005 263 115 207          STMD R43,=GNAM
2757 072010 261 110 207          LDMD R43,=FNAM+5
2758 072013 263 122 207          STMD R43,=GNAM+5
2759 072016              *       CLB  R40
2760 072016 316 007 143          JSB  =DCDZER        DECODE SOURCE FILE NAME
2761 072021                      DRP  144
2762 072021 263 202 207          STMD R44,=SRCMSU  SAVE SOURCE MSUS
2763 072024 316 162 146 FILSCN   JSB  =DIRSCN         IF SOURCE FILE THERE?
2764 072027 371 003              JEZ  FILFND          JIF YES
2765 072031 316 244 151          JSB  =ER67D
2766 072034 132             FILFND DRP  32
2767 072035 316 132 147          JSB  =LDFBES         GET SOURCE ORIGIN
2768 072040                      DRP  132
2769 072040 263 170 207          STMD R32,=SRCORG SOURCE FILE ORIGIN
2770 072043 144 036 265          LDMD R44,X36,D.BYTS GET #OF BYTES & B/REC
2771 072046 034 000
2772 072050 012 345              PUMD R44,+R12       SAVE
2773 072052 316 336 143          JSB  =GETTYP        GET FILE TYPE I UNDERSTAND
2774 072055                      DRP  130
2775 072055 012 344              PUBD R30,+R12       SAVE
2776 072057 206                  LRB                 STRIP SECURE FLAGS
2777 072060 206                  LRB
2778 072061 204                  LLB
2779 072062 204                  LLB
2780 072063 310 020              CMB  R30,=DATATY    DATA FILE?
2781 072065 367 036              JZR  DATCOP
2782 072067 316 002 165          JSB  =BYTES        COMPUTE # OF RECS TO COPY
2783 072072 146 263 276          STMD R46,=CR.CNT    SAVE COUNT
2784 072075 203
2785 072076 130 317 004          ANM R30,=T.ASCI    EXTENDED FILE?
2786 072101 000
2787 072102 766 027              JNZ DUPOK2          NOT PROG--> DUP IS OK
2788 072104 132 261 170          LDMD R32,=SRCORG  PROG FILE ORIGIN
2789 072107 207
2790 072110 316 120 176          JSB =GETBUF         READ 1ST SECTOR
2791 072113 316 155 151          JSB =SREAD+        SYSTEM CK SECUR BIT
2792 072116 765 013              JPS DUPOK2          NO DUPLIC. PROTECTION
2793 072120 112 261 344          LDMD R12,=TOS       CLEAN OFF STACK
2794 072123 203
2795 072124 236                  RTN                 COPY NOT PERFORMED
2796 072125              DATCOP  BSS  0
2797 072125 316 165 147          JSB  =LDRECS        # OF RECS TO COPY
2798 072130              *FILE NOT DUPLICATION-PROTECTED, PROCEED WITH COPY
2799 072130 263 276 203          STMD R#,=CR.CNT  COPY COUNT
2800 072133 144 261 206 DUPOK2   LDMD R44,=DSTMSU FOR TRANSFER
2801 072136 207
2802 072137 263 160 207          STMD R44,=ACTMSU
2803 072142 316 246 146          JSB =MTSCAN         FIND DIR ENTRY
2804 072145 143 261 115          LDMD R43,=GNAM  COPY SOURCE INFO
2805 072150 207
```

```
>>>>>          COPY source, destination                                     <<<<<<<
072151 036 247              STMD R43,R36
072153 261 122 207          LDMD R43,=GNAM+5
072156 267 005 000          STMD R43,X36,SEC1/2
072161 130 012 342          POBD R30,-R12      POP FILE TYPE
072164 144 343              POMD R44,-R12
072166 036 267 034          STMD R44,X36,D.BYTS #OF BYTES & B/REC
072171 000
072172 316 374 143          JSB  =PUTTYP
072175 316 132 147          JSB  =LDFBEG       GET DEST ORIGIN
072200 263 172 207          STMD R#,=DSTORG  DEST. ORIGIN
072203                *REWRITE DIR
072203 316 223 163          JSB =WRTDIR        WRITE IT
072206                *COPY FILE FROM SOURCE TO DEST.
072206                *fall thru to MVFILE...
072206                *-------------------------------------------------------
072206                MVFILE  BSS 0
072206                *READ ONE REC FROM SOURCE
072206 144 261 202          LDMD R44,=SRCMSU  SOURCE MSUS
072211 207
072212 263 160 207          STMD R44,=ACTMSU  MAKE IT ACTIVE
072215 132 261 170          LDMD R32,=SRCORG   ...AND ORIGIN
072220 207
072221 211                  ICM R32
072222 263 170 207          STMD R32,=SRCORG UPDATE SECTR CNT
072225 213                  DCM R32            READJUST
072226 316 120 176          JSB =GETBUF        READ A SECTOR
072231                *WRITE ONE RECORD TO DEST.
072231 132 261 172          LDMD R32,=DSTORG DEST. ORIGIN
072234 207
072235 211                  ICM R32
072236 263 172 207          STMD R32,=DSTORG UPDATE SECTR CNT
072241 213                  DCM R32            READJUST
072242 144 261 206          LDMD R44,=DSTMSU  DEST MSUS
072245 207
072246 263 160 207          STMD R44,=ACTMSU  FOR LOW LEVEL
072251 316 032 176          JSB =PUTBUF        DISC WRITE
072254 132 261 276          LDMD R32,=CR.CNT    RECS TO COPY
072257 203
072260 213                  DCM R32
072261 263 276 203          STMD R32,=CR.CNT   UPDATE IT
072264 366 320              JNZ MVFILE         MORE TO TRANSFER
072266 236                  RTN         TRANSFER COMPLETE
072267                *-------------------------------------------------------
072267                *Get next directory entry for copying files.
072267                *Returns: E=1 directory is exhausted,
072267                *          E=0 R36 is valid directory pointer.
072267 145 014 245 NXTFIL   LDMD R45,R14          RESTORE DIR INFO
072272 263 151 207          STMD R45,=DENTRY  ...RESTORED
072275 154 261 202          LDMD R54,=SRCMSU  RESTORE FOR GETSEC
072300 207
072301 263 160 207          STMD R54,=ACTMSU  ...RESTORED
072304                ****     TSB R54             TAPE OR DISC?
072304                *        STMD R45,=DENTRY ...RESTORED
```

```
ROM 03/14/81 ================================================================
ITEM    LOC   OBJECT CODE  SRC=LCGMA? OBJ=GENGMA   9/11/1981  3:03 PM  PG 37
================================================================
>>>>>>>      COPY source, destination                          <<<<<<<
2845 072304 146 032 243          STM R46,R32         RESTORE DIR SECTOR
2846 072307 316 120 176          JSB =GETBUF         RESTORE DIR SEGMENT
2847 072312 316 044 147 SKHOLE   JSB =NXTENT         SCAN NEXT FILE
2848 072315 370 017              JEN NXTDR+          NO MORE FILES!
2849 072317 234          SKHOL+  ICE                 FLAG LAST FILE
2850 072320 316 336 143          JSB =GETTYP         CK FILE TYPE
2851 072323 210                  ICB R#              LAST FILE?
2852 072324 367 010              JZR NXTDR+          JIF LAST FILE
2853 072326 212                  DCB R#              HOLE FILE?
2854 072327 367 361              JZR SKHOLE          THEN SKIP IT!
2855 072331 235                  CLE                 FLAG GOOD FILE
2856 072332 145 261 151          LDMD R45,=DENTRY NEW DENTRY
2856 072335 207
2857 072336 236          NXTDR+  RTN
2858 072337              *-------------------------------------------------
```

```
>>>>>>       VOLUME msus,vol.label                                  <<<<<<<
2361 072337 241                    OCT 241
2361 072340                VOLUM.  BSS 0
2362 072340 030                    APP 30
2363 072341 316 245 161            JSB =MSINHK          RUNTIME INIT
2364 072344 316 304 161            JSB =GETNAI          GET 6-CHAR LABEL
2365 072347 142 014 247            STMD R42,R14          STASH IT
2366 072352                *       LDB R40,=2           ALLOW MSUS ONLY
2367 072352 316 266 142            JSB =DCD-            PARSE MSUS
2368 072355 132 223                CLM R32              SECTOR#0
2369 072357 316 120 176            JSB =GETBUF          READ IT
2370 072362 316 037 147            JSB  =RBUF36         BUFFER BASE
2371 072365 142 014 245            LDMD R42,R14          GET NEW VOL.LABEL
2372 072370 036 267 002            STMD R42,X36,V.LABL  UPDATE IT
2372 072373 000
2373 072374 132 223                CLM R32              SECTOR#0
2374 072376 316 032 176            JSB =PUTBUF          REPLACE IT
2375 072401 236                    RTN
2376 072402                *-------------------------------------------------
2377 072402                BYTES   BSS 0
2378 072402                *USE D.BYTS TO COMPUTE #-OF-SIGNIFICANT-RECS,
2379 072402                * #-BYTES-IN-LAST-RECORD IN A FILE.
2380 072402 175 036 265            LDMD R75,X36,D.BYTS
2380 072405 034 000
2381 072407 144 223                CLM R44
2382 072411 975 240                LDB R44,R75
2383 072413 266 002                JNZ BYT1
2384 072415 145 210                ICB R45
2385 072417                *NOW R44,5 = BYTES IN LAST REC
2386 072417 146 076 241 BYT1       LDM R46,R76
2387 072422 175 220                TSB R75
2388 072424 367 002                JZR BYT2
2389 072426 146 211                ICM R46
2390 072430                *NOW R46,7=# SIGNIFICANT RECORDS IN FILE
2391 072430 144            BYT2    DRP 44
2392 072431 236                    RTN
```

```
                        CHECKREAD (OFF) Buffer#
072432 241                      OCT 241
072433                  CHECK.  BSS 0
072433 036                      APP 36
072434 316 245 161              JSB =MSINHK         RUNTIME INIT
072437 316 166 152              JSB =GET#           GET BUFFER#
072442 235                      CLE                 FLAG CHECK ON
072443 765 003          CKCOMN  JPS CKOPEN          JIF BUFFER OPEN
072445 316 373 154              JSB =ER66D          FILE NOT OPEN
072450 155 261 053      CKOPEN  LDMD R55,=CURLOC    BUFFER BEGINNING
072453 210
072454 313 030 000              ADM  R55,=A.CHEK    CHECK OFF OFFSET
072457 000                      OCT 0
072460 263 314 377              STMD R55,=PTR2      EXTEND MEM PTR.
072463 222                      CLB R#
072464 370 001                  JEN CKON            JIF CHECK FLAG ON
072466 210                      ICB R#              FLAG CHECK OFF
072467 272 314 377      CKON    STB1 R#,=PTR2       ASSIGN# CHECK FLAG
072472 236                      RTN
072473                  *-------------------------------------------------------
072473 241                      OCT 241
072474                  CHKOF.  BSS 0
072474 037                      APP 37
072475 316 245 161              JSB =MSINHK         RUNTIME INIT
072500 316 166 152              JSB =GET#           GET BUFFER#
072503 235                      CLE
072504 234                      ICE                 FLAG CHECK OFF
072505 760 334                  JMP CKCOMN          CHECK COMMON ROUTINE
```

```
                        GLOAD,GSTORE file specifier
2921 072507 241                     OCT 241
2922 072510                 GLOAD.  BSS 0
2923 072510 005                     APP 5
2924 072511 316 245 161             JSB =MSINHK         RUNTIME INIT
2925 072514 316 265 165             JSB =GCOMN          INIT & PARSE FILE
2926 072517 260 302 377             LDBD R#,=CRTSTS     WHAT MODE AM I IN?
2927 072522 764 010                 JNG  GRFIL          GRAPHICS OK
2928 072524 310 100                 CMB  R#,=100        ALPHA ALL?
2929 072526 372 004                 JNC  GRFIL          JUST ALPHA OK
2930 072530 316 225 161             JSB  =MSERR+        WRONG MODE!
2931 072533 030                     OCT  24D
2932 072534 316 164 150 GRFIL       JSB =GETFIL         SCAN,CK TYPE,SEEK
2933 072537                 *NOW DO THE LOAD
2934 072537 156 222                 CLB R56
2935 072541 316 133 176             JSB =GETBU+
2936 072544 126 251 340             LDM R26,=340,56     # OF BYTES GRAPH MODE
2936 072547 056
2937 072550 146 260 200             LDBD R46,=RECBUF    GET CRT SAVED CRT STATUS
2937 072553 205
2938 072554 310 300                 CMB  R46,=300       GRAPH OR GRAPH ALL?
2939 072556 126                     DRP  26
2940 072557 232                     SAD
2941 072560 372 003                 JNC  LCOUNT         JIF GRAPH
2942 072562 251 300 077             LDM  R#,=300,77     # FOR GRAPH ALL
2943 072565 263 177 200 LCOUNT      STMD R#,=RANDOM
2944 072570 313 340 020             ADM  R#,=GBASE      ADD GBASE OFFSET
2945 072573 263 342 211             STMD R#,=GSIZE      UPDATE
2946 072576 237                     PAD
2947 072577 372 005                 JNC  JSTGRF         JUST GRAPHICS MODE
2948 072601 316 377 377             JSB  =GRAFA.        SET GRAPH ALL
2949 072604 260 003                 JMP  LOADLP
2950 072606 316 377 377 JSTGRF      JSB  =GRAPH.        SET GRAPHICS MODE
2951 072611             LOADLP      BSS  0
2952 072611 316 275 165             JSB  =GBADDR        RESET GRAPHICS BASE
2953 072614 316 037 147             JSB  =RBUF36        RECLAIM PTR
2954 072617                         DRP  !36
2955 072617 211                     ICM  R36            UPDATE TO DATA
2956 072620 126 260 177             LDBD R26,=RANDOM      1ST REC NOT TOT 256 BYTES
2957 072623 200
2958 072624 316 377 377 LLOOP2      JSB =CHKSTS         WAIT FOR CRT
2958 072627 036 340                 FOBD R#,+R36        GET A BYTE FRM BUFFER
2959 072631 262 303 377             STBD R#,=CRTDAT     WRITE TO CRT
2960 072634 126 212                 DCB R26             256 COUNT
2961 072636 766 364                 JNZ LLOOP2          MORE IN BUFFER
2962 072640 260 200 200             LDBD R#,=MTFLAG     REC COUNT
2963 072643 367 225                 RZR                 RTN IF DONE
2964 072645 212                     DCB R#
2965 072646 262 200 200             STBD R#,=MTFLAG     STORE FOR LATER
2966 072651 156 222                 CLB  R56
2967 072653 316 133 176             JSB =GETBU+         GET ANOTHER REC
2968 072656 316 037 147             JSB =RBUF36         BEGIN OF BUFFER
2969 072661 126 222                 CLB R26             256 BYTES
2970 072663 260 337                 JMP LLOOP2          GET MORE GRAFURE
```

```
                        *LOAD,*STORE file specifier
 072665                 *-------------------------------------------------
 072665                 GCOMN   BSS 0                   DO GRAPHICS COMMON
2973 072665 316 326 161         JSB =TAPD3+             DECODE & CHECK MSUS
2974 072670 250 014             LDB R#,=T.GRAF           FLAG GRAPHICS FILE
2975 072672 262 271 203         STBD R#,=FILTYP
2976 072675 316 377 377 GBADDR  JSB  =CHK3TS            WAIT FOR CRT NOT BUSY
2977 072700 176 251 340         LDM  R76,=GBASE         GRAPHICS BASE
2977 072703 020
2978 072704 263 301 377         STMD R76,=CRTBAD
2979 072707 236                 RTN
2980 072710                     *-------------------------------------------------
2981 072710 241                 OCT 241
2982 072711             GSTOR.  BSS 0
2983 072711 006                 ARP 6
2984 072712 316 245 161         JSB =MSINHK             RUNTIME INIT
2985 072715 316 265 165         JSB =GCOMN              PARSE FILE
2986 072720 316 377 377         JSB =GRAPH              SET GRAPHICS MODE
2987 072723 152 223             CLM R52                 FOR FILE SIZE CK
2988 072725 126 261 342         LDMD R26,=GSIZE         SCREEN=GSIZE-GBASE
2989 072730 211
2989 072731 315 340 020         SBM  R26,=GBASE
2990 072734 263 177 200         STMD R26,=RANDOM        REMEMBER FOR STORE
2991 072737 211                 ICM                     INCLUDE CRTSTS BYTE
2992 072740 052 243             STM  R26,R52            TRANSFER SIZE
2993 072742 152 263 245         STMD R52,=NXTDAT        FILE SIZE
2993 072745 203
2994 072746 316 346 145         JSB =FILOK?             EXIST,TYPE,LENGTH,ETC
2995 072751             *NOW DO THE STORE
2996 072751 316 037 147         JSB  =RBUF36            BUFFER ADDR
2997 072754 126 260 302         LDBD R26,=CRTSTS        GET CRT STATUS
2997 072757 377
2998 072760 036 344             PUBD R26,+R36           SAVE FOR LOAD
2999 072762 260 177 200         LDBD R26,=RANDOM        1ST REC NOT FULL 256
3000 072765 316 377 377 SLOOP   JSB =INCHR              GET CHAR FROM SCREEN
3001 072770 036 344             PUBD R#,+R36            OUTPUT TO BUFFER
3002 072772 126 212             DCB R26                 RECORD DONE?
3003 072774 266 367             JNZ SLOOP               MORE IN REC
3004 072776 316 170 176         JSB =PUTBU+             OUTPUT TO DISC
3005 073001 260 200 200         LDBD R#,=MTFLAG          24 REC COUNT
3006 073004 267 301             RZR                     RTN IF DONE
3007 073006 212                 DCB R#
3008 073007 262 200 200         STBD R#,=MTFLAG
3009 073012 316 037 147         JSB  =RBUF36            BEGIN OF BUFFER
3010 073015 126 222             CLB  R26                256 BYTES
3011 073017 260 344             JMP SLOOP               MORE RECS TO GET
3012 073021
3013 073021 077 000     GMARK.  DEF GMARK
```

```
                        PRINTER DRIVER
3015  073023 126 263 060 PRDRVR  STMD R26,=RESDAT    SAVE POINTER
3015  073026 210
3016  073027 136 263 056         STMD R36,=DATLEN    SAVE COUNT
3016  073032 210
3017  073033 156 223             CLM R56             FOR LINE LENGTH
3018  073035 270 314 203         LDBI R56,=LINELN
3019  073040 036 301             CMM R56,R36         COMPARE TO JOB LENGTH
3020  073042 373 034             JCY LINEL>          JIF SHORTER THAN LINE
3021  073044 136 006 345         PUMD R36,+R6        SAVE TOTAL JOB
3022  073047 056 241             LDM R36,R56         SET MAX LENGTH
3023  073051 263 056 210         STMD R36,=DATLEN    STORE IT TOO
3024  073054 316 100 166         JSB =LINEL>         SUBROUTINE JOB
3025  073057 156 223             CLM R56
3026  073061 270 314 203         LDBI R56,=LINELN    FETCH LINE LENGTH
3027  073064 136 006 343         POMD R36,-R6
3028  073067 056 305             SBM R36,R56         REMAINING JOB
3029  073071 126 261 060         LDMD R26,=RESDAT
3029  073074 210
3030  073075 303                 ADM R26,R56         NEXT DATA ADDRESS
3031  073076 260 323             JMP PRDRVR
3032  073100
3033  073100 156 026 241 LINEL>  LDM R56,R26
3034  073103 036 303             ADM R56,R36         END OF DATA
3035  073105 132 056 342         POBD R32,-R56       LAST DATA BYTE
3036  073110 310 015             CMB R32,=CR
3037  073112 367 023             JZR LOADSC          JIF GOT CR
3038  073114 316 137 166         JSB =LOADSC         SUBROUTINE JOB
3039  073117 117 310 300         CMB R17,=300
3040  073122 373 012             JCY OOPS
3041  073124 146 250 107 SNDEOL  LDB R46,=107        SEND EOL SEQUENCE
3042  073127 134 251 300         LDM R34,=300,10     COMMAND HANDSHAKE
3042  073132 010
3043  073133 316 072 170         JSB =CMDOUT         READ PSR
3044  073136 236           OOPS  RTN
```

```
                PROCESS ADDRESSING
3046 073137 316 266 161 LOADSC  JSB =ININIT
3047 073142                     ORP  14
3048 073142 262 147 207         STBO R14,=PRDVFG  SET PRNTER DRVR FLAG
3049 073145 160 261 321         LDMD R60,=SCTEMP   BEEN HERE BEFORE?
3049 073150 203
3050 073151 267 044             JZR OUTBUS        JIF YES, NO ADDRESSING
3051 073153 316 377 377         JSB =ROMJSB
3052 073156 054 143             DEF PRADDR        GET ADDRESS
3053 073160 001                 OCT 1
3054 073161 117 310 300         CMB R17,=300      ANY ERRORS?
3055 073164 373 350             JCY OOPS
3056 073166 316 277 161 PREXIT  JSB  =INIT14
3057 073171 122 046 240         LDB R22,R46       GET DEVICE
3058 073174 145 056 240         LDB R45,R56
3059 073177 314 003             SBB R45,=3
3060 073201 316 222 172         JSB =SETCC-       SET R22,R34,R14
3061 073204 126 263 212         STMD R26,=SAVE26
3061 073207 207
3062 073210 140 223             CLM R40
3063 073212 263 321 203         STMD R40,=SCTEMP  CLR SCTEMP FOR NXT TIME
3064 073215 371 011             JEZ SENDB+        JIF ADDRESSING
3065 073217                 OUTBUS  RSS  0
3066 073217 126 261 212         LDMD R26,=SAVE26  RECOVER R26
3066 073222 207
3067 073223 316 231 172         JSB =SETSC+       SET R34
3068 073226 260 036             JMP SEND++        SEND WITH NO ADDRESSING
3069 073230
3070 073230 122 312 040 SENDB+  ADB R22,=40       DEVICE IS LSTNER
3071 073233 220                 TSB R22
3072 073234 264 030             JNG SEND++        JIF RS232
3073 073236 316 306 166         JSB =LLINI+       UNL,UNTALK
3074 073241 146 250 105         LDB R46,=105      CAPRICORN TALKER
3075 073244 262 147 207         STBO R46,=PRDVFG  LLINI+ CLEARED
3076 073247 134 223             CLM R34
3077 073251 316 072 170         JSB =CMDOUT
3078 073254                 *   JSB =LADTAD       SEND LISTEN ADDRESS
3079 073254 166 251 273 LADTAD  LDM R66,=R/FILE   TEMP BUFF
3079 073257 203
3080 073260 122 066 246         STBO R22,R66      STORE LAD OR TAD
3081 073263 316 312 166         JSB =LLIN++       SEND IT
3082 073266 166 261 060 SEND++  LDMD R66,=RESDAT  START
3082 073271 210
3083 073272 262 147 207         STBO R66,=PRDVFG  LLIN++ CLEARED
3084 073275 136 261 056         LDMD R36,=DATLEN  LENGTH
3084 073300 210
3085 073301 146 250 240 SENDCR  LDB R46,=240      DATA OUT COMMAND
3086 073304 260 012             JMP DATOT
```

```
                        REGISTER CONVENTIONS
3089 073306                     *       R14                     MSBASE
3090 073306                     *       R22                     DEVICE
3090 073306                     *       R26                     CCR ADDRESS
3091 073306                     *       R76                     IO BUFFER ADDRESS
3092 073306                     *       R66                     DATA POINTER
3093 073306                     *       R75                     STATUS
3094 073306                     *       R32                     DATA
3095 073306                     *       R34                     TIME-OUT COUNT
3096 073306                     *       R36                     DATA COUNTER
3097 073306                     *       R46                     COMMAND BYTE
3098 073306
3099 073306 166 251 021 LLINI+  LDM R66,=QMARK.
3099 073311 166
3100 073312 136 223     LLIN++  CLM R36
3101 073314 211                 ICM R36
3102 073315 146 250 260         LDB R46,=260
3103 073320             DATOT   BSS 0
3104 073320 316 261 170         JSB =CMDCED             CMD, THEN CED=0
3105 073323 132 066 340 LOP-03  POBD R32,+R66           POP DATA BYTE
3106 073326 134 223             CLM R34
3107 073330 316 220 170         JSB =OCKLOP             WAIT FOR OBF
3108 073333 132 310 015         CMB R32,=CR             CARRIAGE RETURN?
3109 073336 266 011             JNZ NCR                 JIF NO
3110 073340 316 124 166         JSB =SNDEOL             SEND EOL SEQUENCE
3111 073343 136 213             DCM R36                 MORE BYTES?
3112 073345 266 332             JNZ SENDCR              JIF YES,SET DATA AGAIN
3113 073347 360 027             JMP JOBF
3114 073351             NCR     BSS 0
3115 073351 146 260 150         LDB R46,=PRDVF+         LOOKING FOR SPEC. CHARS?
3115 073354 207
3116 073355 132                 ORP  32
3117 073356 267 012             JZR  NCR++              JIF NO
3118 073360 310 040             CMB  R32,=40            CHAR<BLANK?
3119 073362 372 004             JNC  BLANK              YES, GO BLANK
3120 073364 310 177             CMB  R32,=177
3121 073366 372 002             JNC  NCR++              OUTPUT GOOD CHAR
3122 073370             BLANK   ORP  !32
3123 073370 250 040             LDB  R32,=40
3124 073372 076 246     NCR++   STBD R#,R76             WRITE DATA BYTE
3125 073374 136 213             DCM R36                 DECR. COUNT
3126 073376 266 323             JNZ LOP-03              JIF MORE
3127 073400 316 075 170 JOBF    JSB =OBFCK
3128 073403 122                 CLB R#
3129 073404 262 147 207         STBD R#,=PRDVFG  CLEAR PRNTR DRVR FLAG
3130 073407 236                 RTN
3131 073410
```

```
                SET I/O SUPPORT
3    073410
3    073410                 CNTOUT  BSS   0
3135 073410 106 263 066             STMD  R6,=SAVER6     REMEMBER RETURN ADDR
3135 073413 210
3136 073414 100 223                 CLM   R0
3137 073416 006 345                 PUMD  R0,+R6         PAD STACK
3138 073420 345                     PUMD  R0,+R6
3139 073421 157 250 001             LDB   R57,=1         ONE CONTROL BYTE
3140 073424 166 251 135             LDM   R65,=DATAB     ADDRESS OF CONTRL DATA
3140 073427 207
3141 073430 316 277 170             JSB   =DATOUT        CALL HELPFUL  ROUTINE
3142 073433 100 006 343             POMD  R0,-R6         IF RTN, CLEAN STACK
3143 073436 343                     POMD  R0,-R6
3144 073437 236                     RTN
```

```
3146 073440                 *999999999999999999999999999999999999999**
3147 073440                 *   ERROR AND INITIALIZATION MESSAGES        :
3148 073440                 *   FOLLOW.                                  *
3149 073440                 ******************************************
3150 073440                 ERMESS  BSS 0
3151 073440                 *-------------------------------------------------
3152 073440                 *To all ROM modifiers: Please note the following
3153 073440                 *OUTRAGEOUS code. VOLHED has been underscored and
3154 073440                 *moved to the error msg table to save 7 bytes.
3155 073440                 *VOLHED MOVED TO SYSTEM 12-17-80
3156 073440                 *OLHED  VAL V
3157 073440                 *       OCT 357        o
3158 073440                 *       OCT 354        l
3159 073440                 *       OCT 365        u
3160 073440                 *       OCT 355        m
3161 073440                 *       OCT 345        e
3162 073440                 *       OCT 272        :
3163 073440                         REP 10D
3163 073440 200                     OCT 200        SYS DEF
3163 073441 200                     OCT 200        SYS DEF
3163 073442 200                     OCT 200        SYS DEF
3163 073443 200                     OCT 200        SYS DEF
3163 073444 200                     OCT 200        SYS DEF
3163 073445 200                     OCT 200        SYS DEF
3163 073446 200                     OCT 200        SYS DEF
3163 073447 200                     OCT 200        SYS DEF
3163 073450 200                     OCT 200        SYS DEF
3163 073451 200                     OCT 200        SYS DEF
3164 073452                 *10D:
3165 073452 111 057 117             ASP 8,I/O CARD
3165 073455 040 103 101
3165 073460 122 304
3166 073462                 *11D:
3167 073462 111 117 320             ASP 3,IOP
3168 073465                 *12D:
3169 073465 115 056 123             ASP 7,M.S.ROM
3169 073470 056 122 117
3169 073473 315
3170 073474                 *13D:
3171 073474 240                     OCT 240
3172 073475                 *14D:
3173 073475 240                     OCT 240
3174 073476                 *15D:
3175 073476 240                     OCT 240
3176 073477                 *16D:
3177 073477 240                     OCT 240
3178 073500                 *17D:
3179 073500 240                     OCT 240
3180 073501                 *18D:
3181 073501 240                     OCT 240
3182 073502                 *19D
3183 073502 240                     OCT 240
3184 073503                 *20D
```

```
                        ERROR MESSAGES
073503 016 117 040              ASP 13D,NO M.S.DEVICE
073506 015 056 123
3185 073511 056 104
3186 073513 105 126 111
3186 073516 103 305
3187 073520                     *21D
3188 073520 240                 OCT 240
3189 073521                     *       ASP 4,ADDR
3190 073521                     *22D
3191 073521 240                 OCT 240
3192 073522                     *       ASP 11D,SELECT CODE
3193 073522                     *-------------------------------------------------------
3194 073522                     *Mass Storage ROM error messages begin # 18d
3195 073522                     *23D:
3196 073522 240                 OCT 240
3197 073523                     *24D:
3198 073523 106 111 114         ASP 5,FILES
3198 073526 105 323
3199 073530                     *25D:
3200 073530 126 117 114         ASP 6,VOLUME
3201 073533 125 115 305
3201 073536                     *26D:
3202 073536 115 123 125         ASP 4,MSUS
3202 073541 323
3203 073542                     *27D:
3204 073542 122 105 101         ASP 8D,READ VFY
3204 073545 104 040 126
3205 073550 106 331
3205 073552                     *28D:
3206 073552 106 125 114         ASP 4,FULL          NO SPACE THIS DISC
3206 073555 314
3207 073556                     *29D:
3208 073556 115 105 104         ASP 6,MEDIUM
3208 073561 111 125 315
3209 073564                     *30D:
3210 073564 104 111 123         ASP 4,DISC
3210 073567 303
3211 073570                     *31D:
3212 073570 124 111 115         ASP 8D,TIME-OUT
3212 073573 105 055 117
3213 073576 125 324
3213 073600 377                 OCT 377
```

```
073601                 *****************************************************
073601                 *    THE INITIALIZATION ROUTINE FOLLOWS             *
073601                 *    THE VARIOUS TYPES OF INITIALIZE                *
073601                 *    ARE DESIGNATED BY ROMFL, VIZ.:                 *
073601                 *   ROMFL                                           *
073601                 *     0   POWER ON                                  *
073601                 *     1   RESET                                     *
073601                 *     2   SCRATCH  (DE-INITIALIZE)                  *
073601                 *     3   LOAD BIN                                  *
073601                 *     4   RUN, INIT                                 *
073601                 *     5   LOAD (DOES SCRATCH SO DE-INIT)            *
073601                 *     6   STOP                                      *
073601                 *     7   CHAIN                                     *
073601                 *    10 ALLOCATE - CLASS > 56                       *
073601                 *    11 DEALLOCATE - CLASS > 56                     *
073601                 *    12 DECOMPILE - CLASS > 56                      *
073601                 *                                                   *
073601                 *    CLASS = LEAST SIG. 6 BITS  OF ATTRIBUTE        *
073601                 *    BIT 7 = BASIC/NON-BASIC    (STMT TYPE)         *
073601                 *    BIT 6 = LEGAL AFTER THEN                       *
073601                 *****************************************************
073601                 MSDUMP  BSS 0
073601 316 266 161             JSB =ININIT
073604                 *WHEN PAUSE OCCURS DUMP ALL DISC BUFFERS
073604                 *ALSO WHEN I/O OCCURS IN CALC MODE.
073604 146 251 012             LDM R46,=12,0        TEN ENTRIES
073607 000
073610 006 345         CLRAS+  PUMD R#,+R6          SAVE COUNTER
073612 316 377 377             JSB =GET#1+         GET 0(CLOSED) OR CURLOC
073615 364 003                 JNG NEXTCL           JIF CLOSED
073617 316 143 154             JSB =WRTDA-          DUMP A BUFFER TO DISC
073622 146 006 343     NEXTCL  POMD R46,-R6         POP COUNTER
073625 213                     DCM R46
073626 366 360                 JNZ CLRAS+
073630 236                     RTN
073631                 *-----------------------------------------------
073631                 INITIT  BSS 0
073631 230                     BIN                  FOR ADDS
073632 134 260 065             LDBD R34,=ROMFL      WHAT KIND INIT?
073635 210
073636 367 011                 JZR YESPOW           POWER ON IF ROMFL = 0
073640 310 006                 CMB R34,=6
073642 367 335                 JZR MSDUMP           PAUSE IF ROMFL = 6
073644 310 001                 CMB R34,=1
073646 367 076                 JZR NOTPAU           RESET IF ROMFL = 1
073650 236                     RTN
073651                 ****** ******* PAUSE (ROMFL = 6)
073651                 ****** ******* POWER ON (ROMFL = 0)
073651                 YESPOW  BSS 0
073651 316 266 161             JSB =ININIT
073654                 *Perform checksum verification:
073654 132 251 000             LDM R32,=R60K         START ADDR =24K
073657 140
```

```
ITEM    LOC   OBJECT CODE  SRC=LCOMNS OBJ=GENSMA   9/11/1981  3:03 PM  PG 55
                      ROM INITIALIZATION
3266 073660 316 377 377          JSB =RSUMSK          SYSTEM CHECKSUM
     073663 267 004              JZR CKSMOK           CHECKSUM OK
3267 073665                  *REMOVE * IN PRECEDING LINE FOR RELEASED ROM!!!!!!!!!!
3268 073665                  *       JMP CKSMOK        FOR RAM ROM!!!!!!!!!!!!
3269 073665 316 212 161          JSB =MSERR
3270 073670 014                  OCT 12D              CHECKSUM FAIL
3271 073671                  CKSMOK  BSS 0
3272 073671                  *************************************************
3273 073671                  *POWER-ON INITIALIZATION
3274 073671                  * GET RAM FOR MS ROM READ/WRITE VARIABLES, AND THEIR
3275 073671                  *       RAM OFFSETS FROM THE BASE ADDRESS MSBASE:
3276 073671                  *PRTBUF EQU 0                 32-BYTE CAT PRINTER BUFFER
3277 073671                  *DEFMSU EQU 32D               4-BYTE DEFAULT MSUS
3278 073671                  *       FILENAME: 10-CHAR FILE NAME
3279 073671                  *FNAM    EQU 36D             ORIGIN
3280 073671                  *FNAM+5  EQU 41D             2ND HALF
3281 073671                  NAMLEN EQU 10D               FILENAME LENGTH(NOTE FNAM+NAM
3282 073671                  SEC1/2 EQU 5D                NAMLEN/2
3283 073671                  *       2ND FILENAME: ALSO 10 CHARS
3284 073671                  *GNAM     EQU 46D            ORIGIN
3285 073671                  *GNAM+5   EQU 51D            2ND HALFOF NAME
3286 073671                  *       SPECIF:6-CHAR VOL LABEL/MSUS
3287 073671                  *SPECIF EQU 56D
3288 073671                  *       DATA BUFFER FOR TRANSLATOR LOW-LEVEL DRIVERS
3289 073671                  *                    (10D BYTES)
3290 073671                  *DATAB  EQU 62D
3291 073671                  *PRDVFG EQU 72D          2 BYTE TO WARD OFF TIME-OUT
3292 073671                  *PRDVF+ EQU 73D
3293 073671                  *PRDVF+  DAD 103550
3294 073671                  *DSJBYT EQU 74D********
3295 073671                  *NPARAM EQU 75D********
3296 073671                  *IOBADD EQU 76D********  (2 BYTE IO BUFFER ADDRESS FOR IN
3297 073671                  *STATDD EQU 78D********  4-BYTE R/W SECTOR STAT. VAR
3298 073671                  *DENTRY EQU 62D          1-BYTE CURRENT DIR ENTRY OFFSET
3299 073671                  *DADDR  EQU 83D          2-BYTE CURRENT DIR SECTOR
3300 073671                  *CUMREC EQU 85D          2-BYTE CUMMULATIVE RECORD CNT (M
3301 073671                  *LSTDIR EQU 87D          2-BYTE LAST SECTOR OF DIRECTORY
3302 073671                  *SECTDD EQU 89D********  (2 BYTE SECTOR VALUE (BINARY))
3303 073671                  *RWFLG  EQU 91D********  (1 BYTE READ/WRITE FLAG)
3304 073671                  *CCRTEM EQU 92D********  (1 BYTE TEMPORAYR CCR IMAGE)
3305 073671                  *ACTMSU EQU 93D          4-BYTE ACTIVE MSUS
3306 073671                  *PARAMS EQU 97D          2-BYTE PARAMETER TEMP
3307 073671                  *PARAM2 EQU 99D********  2-BYTE PARAMETER TEMP
3308 073671                  *-------------------------------------------------
3309 073671                  *CPYSIZ EQU 101D             # SECTORS TO COPY
3310 073671                  *SRCORG EQU 103D             SOURCE FILE ORIGIN
3311 073671                  *DSTORG EQU 105D             DESTINATION FILE ORIGIN
3312 073671                  *DENTD  EQU 107D             DEST FILE DIR.ENTRY#
3313 073671                  *DADDD  EQU 108D             DEST FILE DIR.SECTOR#
3314 073671                  *DENTS  EQU 110D             SRCE FILE DIR.ENTRY#
3315 073671                  *DADDS  EQU 111D*********    SRCE FILE DIR.SECTOR#
3316 073671                  *SRCMSU EQU 113D             SOURCE FILE MSUS
3317 073671                  *DSTMSU EQU 117D             DEST FILE MSUS-4byte
```

ROM INITIALIZATION

```
073671                *SRCTYP EQU 121D            SOURCE FILE TYPE
073671                *Total bytes used so far: 121d
073671                ***********************************************
073671                *SFAR#  EQU 123D
073671                *ERRRAM EQU 124D
073671                *OCTER# EQU 127D
073671                ***********************************************
073671                *
073671                * STEP 2: OBTAIN RAM ADDRESS AND INTIALIZE FILE TABLE
073671                *         SPACE RESERVED IN GLOBALS FOR MSROM
073671                *
073671                *       LDM  R14,=4,0       TABLE ENTRY FOR "GRAF"
073671 144 251 107            LDM  R44,=107,122,101,106
073674 122 101 106
073677                *       STMD R44,X14,EXTFIL  STORE ASCII "GRAF"
073677 263 014 220            STMD R44,=EXTGRF     STORE ASCII "GRAF"
073702                EXTGRF  DAD  110014          EXTFIL=110010 & 4
073702 114 251 037            LDM  R14,=MSPTR
073705 207
073706 263 012 207            STMD R14,=MSBASE   RAM BASE ADDRESS
073711                *
073711                *
073711                * SET TIME-OUT DELAY FOR LOW-LEVEL
073711                *       LDBD R76,=IR020
073711 316 020 170    COMN-   JSB  =ISRHOK
073714                *
073714                * STEP 1:
073714                *              ESTABLISH ISR HOOK
073714                *              CLEAR IOBITS AND ERRSC
073714                *              RESET IOPS
073714                *           ESTABLISH ISR HOOK
073714                *
073714 146 223                CLM R46
073716 263 140 202            STMD R46,=IOBITS   CLEAR IOBITS AND ERRSC
073721 316 353 167            JSB =IOPRST        RESET IOPS
073724                DONOTR  BSZ 0
073724 146 223                CLM R46
073726 263 147 207            STMD R46,=PRDVFG    CLR PRNTR DRVR FLAG
073731 262 224 210            STBD R46,=DFLAG     CLR BURST IN FLAG
073734                *
073734                * SET THE INITIAL MSUS (SEARCH FOR A DISC CONTROLLER)
073734 316 277 176            JSB =VOL2AD
073737                RSR.A.  BSZ 0
073737                *
073737                ***********************************************
073737                *  This return address is used by VOL2AD
073737                *  in a manne specific to this location
073737                *  DO NOT use VOL2AD for initialization any other
073737                *  place except here!
073737                ***********************************************
073737                *
073737                * SET DMSUS ON RAM
073737                *
```

```
                    ;ROM INITIALIZATION                                  <<<<<<<<
3369 073737 316 165 177          JSB =SETMSU
3370 073742                      DRG 144
3371 073742 263 977 207          STMD R44,=DEFMSU
3372 073745 236                  RTN
3373 073746                  *-----------------------------------------
3374 073746                  NOTPAU BSS 0
3375 073746                  *
3376 073746                  * ESTABLISH ISR HOOK
3377 073746                  *                        RESET IOPS
3378 073746                  *
3379 073746 316 266 161          JSB =ININIT
3380 073751                  * AND RETURN THROUGH COMMON AREA
3381 073751 760 336              JMP COMN-
3382 073753                  *-------------------------------------------------
3383 073753                  IOPRST  BSS 0
3384 073753                  *
3385 073753                  *  RESET IOPS
3386 073753                  *
3387 073753 130 250 200          LDB R30,=200
3388 073756 262 001 377          STBD R#,=GINTDS   GLOBAL INTERRUPT DISABLE
3389 073761 126 251 120          LDM R26,=120,377  CCR BASE
3389 073764 377
3390 073765                  RSLOOP  BSS 0
3391 073765 130 026 246          STBD R30,R26    WRITE RESET TO CCR
3392 073770                  *  HOLD FOR > 7 MICSEC
3393 073770 131 250 010          LDB R31,=10
3394 073773                  WLOP-+  BSS 0
3395 073773 212                  DCB R#
3396 073774 766 375              JNZ WLOP-+
3397 073776                  * NOW RELEASE RESET LINE
3398 073776 026 344              PUBD R#,+R26
3399 074000 126 211              ICM R26
3400 074002 310 140              CMB R26,=140
3401 074004 372 357              JNC RSLOOP
3402 074006                  * AND ENABLE THE RESET IOP TO INTERRUPT
3403 074006 316 272 171          JSB =RE--EN         ENABLE IO INT, AND GLOBAL
3404 074011                  * NOW WAIT FOR IOPS TO INTERRUPT
3405 074011 130 250 020 WAIT20   LDB R30,=20
3406 074014 316 377 377          JSB =CNTRTR   WAIT 20 RETRACES (>250 MILLSEC)
3407 074017                  * AND RETURN
3408 074017 236                  RTN
3409 074020                  *-----------------------------------------
3410 074020                  ISRHOK  BSS 0
3411 074020                  *
3412 074020                  * ESTABLISH ISR HOOK
3413 074020                  *
3414 074020 141 262 001          STBD R41,=GINTDS   DISABLE INTERRUPTS
3414 074023 377
3415 074024 251                  OCT 251             LDM R41,=NEXT 7 BYTES
3416 074025 232                  BAD                   OF CODE
3417 074026 262 001 377          STBD R#,=GINTDS
3418 074031 316 377 377          JSB =ROMJSB
3419 074034 263 342 207          STMD R#,=IR020   SET FIRST SEVEN BYTES OF
```

```
074037                    *                      INTERRUPT VECTOR
074037 140                       ORP 40
074040 251                       OCT 251         LDM R#,=NEXT 8 BYTES OF
074041 176 171                   DEF ISR              CODE
074043 320                       VAL MYROM#
074044 262 000 377               STBD R#,=GINTEN
074047 237                       PAD
074050 236                       RTN
074051 263 351 207               STMD R#,=IR020+
074054                    *************
074054                    *  LOAD IOP ERROR REPORT CODE IN RAM
074054 143                       ORP 43           LOAD JSB =ERROR ...
074055 251                       OCT 251
074056 316                       OCT 316
074057 377 377                   DEF ERROR
074061 000 236                   OCT 0,236
074063 263 215 207               STMD R#,=ERRRAM
074066                    *
074066                    *************
074066 262 000 377               STBD R#,=GINTEN
074071 236                       RTN
074072                    *-----------------------------------------------
```

```
                TRANSLATOR LOW-LEVEL DRIVERS
074072  ********************************************************************
074072  *********                                                  *********
074072  *********                                                  *********
074072  *********      LOW-LEVEL DISC DRIVERS                      *********
074072  *********                                                  *********
074072  ********************************************************************
074072  *
074072  *  THESE DRIVERS PROVIDE A RELATIVELY ERROR-RESISTANT
074072  *    CONCEPTUALLY SIMPLE VIEW OF THE DISC AS A
074072  *    SEQUENCE OF SECTORS WHICH CAN BE RANDOMLY, OR
074072  *    SEQUENTIALLY ACCESSED.  THERE IS A HIGHLY
074072  *    RESTRICTED SET OF ENTRY POINTS TO THIS SECTION.
074072  *    THE ENTRY POINTS ARE:
074072  *           TOID                VOL2AD
074072  *           GETSEC              PUTSEC
074072  *           GETBUF              PUTBUF
074072  *           GETSE+              PUTSE+
074072  *           HLSEEK      and     HLFMT.
074072  *
074072  *  LOW LEVEL DRIVERS ARE DIVIDED INTO THREE GROUPS:
074072  *    GROUP 1 IMPLEMENT THE CAPRICORN TO
074072  *    IO PROCESSOR PROTOCOL.  ROUTINES IN THIS
074072  *     GROUP ARE:      CMDOUT  Command out
074072  *                     CMDHS   Command Handshake
074072  *                     DATOUT  Data out
074072  *                     DATAIN  Data in
074072  *                     BSTOUT  Burst out
074072  *                     BSTIN   Burst in
074072  *                     OPSU#1  Optimization # 1
074072  *           and ISR the Interrupt service routine.
074072  *           Additionally, there are several loop
074072  *           routines. Every loop in the low-level
074072  *           has a time-out count to prevent low-
074072  *           level hangs.
074072  *              Loop 4 (BUSY check) is in DATAIN,
074072  *           and loop 6 (FPOLL check) is in PPOLL.
074072  *           The other loops are in the following
074072  *           routines*
074072  *                     OBFCK   Check OBF and BUSY
074072  *                     OCK     Check OBF
074072  *                     ICK     Check IBF
074072  *                   and SFLGCK  Check SFLG.
074072  *
074072  *    GROUP 2  DEALS WITH HPIB SEQUENCES USED TO
074072  *     DRIVE THE DISC HARDWARE. THESE ROUTINES SEND
074072  *     HPIB SEQUENCES THROUGH THE IOP, USING
074072  *     THE 1 ROUTINES.  ROUTINES IN THIS GROUP
074072  *     ARE:      LLINIT    Establish register env
074072  *               EX__IT    Restore IOP EOL count
074072  *               FORM.T    Format a disk
074072  *               DSJ       Device specified jump
074072  *               PP        Parallel poll
074072  *               RDBUFF    Read-buffered
```

```
                        TRANSLATOR LOW-LEVEL DRIVERS
074072                  *               WRBUFF   Write-buffered.
074072                  *               TO_ID    Time-out identify.
074072                  *               STATUS   Get 2 status words.
074072                  *               SNDMTA   (& SNDMLA) Send my talk addr?
074072                  *               FORMAT   Format and verify disc
074072                  *               HLTIOP   (& SRTIOP) Suspend IOPs
074072                  *               VERIFY   Check medium's integrity
074072                  *               INITZE   Mark defective track
074072                  *               REDDA    Request Disc address
074072                  *               LLPARI   Add parity to command
074072                  *          and  UNTALI   send UNTALK and UNLISTEN
074072                  *
074072                  *     THE MASS STORAGE ROM DEALS WITH PROTOCOL
074072                  *     ERRORS OVER THE HPIB IN AN OCCASIONALLY RADICAL
074072                  *     MANNER.   WHEN COMMUNICATION WITH AN IOP IS
074072                  *     BROKEN (TYPICALLY DUE TO AN INCORRECT CONTROLLER
074072                  *     ADDRESS), THEN THE MS ROM RESETS THE IOP TO
074072                  *     REGAIN CONTROL.  THIS CAUSES THE IOP TO RE-
074072                  *     ESTABLISH ITS POWER -ON STATE.  INFORMATION ABOUT
074072                  *     THE HP-IB CARD AND IO PROCESSOR PROTOCOL IS
074072                  *     FOUND IN THE "HP-IB CARD ERS"  (DAVE SWEETSER DCD).
074072                  *          ALL NORMAL HP-IB SEQUENCES ARE TERMINATED WITH
074072                  *     UNTALK AND UNLISTEN.  AT POWER-ON OF THE
074072                  *     CAPRICORN THE ROUTINE VOL2AD IS CALLED TO DETERMINE
074072                  *     WHETHER A DISC IS ON LINE.  IF ONE IS AT POWER-ON
074072                  *     THEN ITS ADDRESS IS SET AS THE DEFAULT MSUS.  IF
074072                  *     SEVERAL DISC S ARE ON-LINE THEN THE CONTROLLER
074072                  *     WITH TE LOWEST S.C. AND THEN LOWEST C.A. BECOMES
074072                  *     THE DEFAULT CONTROLLER AT POWER-ON.
074072                  *     IF THIS HAPPENS THEN THE MASS STORAGE ROM SENDS
074072                  *
074072                  *
074072                  *     GROUP 3 ARE HIGH-LEVEL ENTRY POINTS.  THESE
074072                  *      ARE ENTERED FROM THE REST OF THE MSROM TO
074072                  *      PERFORM DISC OPERATIONS. PARAMETER PASSING
074072                  *      TECHNIQUES FOR EACH ROUTINE WILL BE DISCUSSED
074072                  *      IMMEDIATELY PRIOR TO EACH. GROUP 3 CONTAINS:
074072                  *          TOID     Time-out identify with LLINIT and
074072                  *                   ER__IT
074072                  *          HLSEEK   High-level seek to conceptual
074072                  *                   sector (in R30)
074072                  *          HLFMT    High-level Format a disk (stagger
074072                  *                   in R30)
074072                  *          GETSEC   Get sector (256 bytes) from disk,
074072                  *                   Conceptual target sector in R32,
074072                  *                   data address in R30.
074072                  *          PUTSEC   Put sector (see GETSEC)
074072                  *          GETSE+   Get sector without seek (data
074072                  *                   address in R30).  Used with auto-
074072                  *                   increment for multi-sector bursts.
074072                  *          PUTSE+   Put sector without seek.
074072                  *          GETBUF   Read sector into tape buffer
074072                  *          PUTBUF   Write sector from tape buffer
```

```
074072          *          VOLZAD  Translate Volume label into
074072          *                  hardware msus (involves medium
074072          *                  search).
074072          *          HLINIT  Record entry point and incoming
074072          *                  parameters for future retry.
074072          *          LLERR   Low-level error recovery. 10 simple
074072          *                  retries, then head move-retry.
074072          *          HDMOVE. Move disc head one track in (then
074072          *                  out) and back.
074072          *
074072          *
074072          * ALL ROUTINES IN BOTH GROUPS ASSUME
074072          * A FAIRLY RIGID REGISTER ENVIRONMENT, I.E.:
074072          *    E        ...  SRT/HLTIOP FLAG
074072          *    R0,1     ...  TIME-OUT COUNT
074072          *    R14,15   ...  MSROM'S RAM BASE ADDR.
074072          *    R20,21   ...  RETRY ADDRESS
074072          *    R22      ...  DEVICE
074072          *    R23      ...  UNIT
074072          *    R24,25   ...  (WORKING REG-PAIR)
074072          *    R26,27   ...  CCR ADDRESS
074072          *    R30,31   ...  TARGET SECTOR (conceptual)
074072          *                  or  ADDRESS (for R/W SECTOR)
074072          *                  or  STAGGAR (for FORMAT)
074072          *    R32      ...  USED TO STORE CCR IMAGE
074072          *    R33      ...  REFORMAT FLAG (ALSO USED BY VOL2AD
074072          *                  AND AS INPUT PARAMETER (GET+PUTBUF))
074072          *    R34,35   ...  TIME-OUT COUNT
074072          *    R36      ...  RD-WR TEMP FLAG
074072          *    R37      ...  TIME-OUT IDENTIFY RSLT FLAG
074072          *    R40      ...  TYPE OF CURRENT DISC
074072          *    R42,47   ...  (USED FOR DATA MOVEMENT)
074072          *    R45      ...  FORMAT TYPE
074072          *    R46      ...  (USED FOR COMMAND BYTE)
074072          *    R50,51   ...  DISC PARAMETERS (CYL & SEC/TRACK)
074072          *    R52      ...  FULL/SHORT PROTOCOL FLAG
074072          *    R54      ...  RETRY COUNT
074072          *    R55      ...  (USED FOR NEW CCR)
074072          *    R56      ...  TEMP IN FORMAT
074072          *                  IOBITS IN HLT/SRTIOP
074072          *    R57      ...  (USED TO TELL NUMBER
074072          *                  OF BYTES TO SEND IOP)
074072          *    R60      ...  DISC TYPE
074072          *    R61      ...  VOL2AD S.C. COUNT
074072          *    R62      ...  VOL2AD C.A. COUNT
074072          *    R63      ...  VOL2AD UNIT COUNT
074072          *    R64      ...  # OF SPARE TRACKS LEFT
074072          *    R65      ...  SRTIOP S.C. COUNT
074072          *    R66,67   ...  DATA TO BE TRANSFERRED ADDRESS
074072          *    R71      ...  RETRY HEAD MOVE STATUS
074072          *    R72,73   ...  ERROR RECOVERY
074072          *    R74      ...  RETRY COUNT
074072          *    R75      ...  TEMP CCR
```

<<<<< TRANSLATOR LOW-LEVEL DRIVERS <<<<<<<

```
3602 074072                *     R76,77  ..    IO BUFFER ADDRESS
3603 074072                *     R72-77  ...   TEMP USE IN VOL2AD (LLINIT FOLLOWING
3604 074072                *
3605 074072                OUT.CO    EQU 260
3606 074072                OUT.DO    EQU 240
3607 074072                OUT.S4    EQU 104
3608 074072                OUT.S5    EQU 105
3609 074072                OUT.S6    EQU 106
3610 074072                OUT.EN    EQU 20
```

```
074072               *************************************************
074072               ***                                           ***
074072               ***     IO PROCESSOR PROTOCOL DRIVERS         ***
074072               ***                                           ***
074072               *************************************************
074072               ***
074072               ***
074072               ***  Communication with the IOP is synchronized
074072               *** with a pair of dedicated registers (memory
074072               *** locations).  One is an I/O buffer, and the
074072               *** second is a control register.  The control
074072               *** register is referred to as the Calculator-
074072               *** Control-Register (CCR) on write operations,
074072               *** and the Processor-Status-Register (PSR) on
074072               *** read.  The important conceptual bitys in the
074072               *** control register are:
074072               ***
074072               ***  ---------------------------------------
074072               ***  ! reset ! d ! d ! d ! d ! CED ! COM ! INT !
074072               ***  ------------------------------------------
074072               ***  CCR
074072               ***      7      6   5   4   3    2     1     0
074072               ***  PSR
074072               ***  ---------------------------------------------
074072               ***  ! OBF ! d ! FD ! SFLG ! PACK ! PED ! BUSY ! IBF !
074072               ***  ------------------------------------------------
074072               ***     7    6    5     4      3     2      1      0
074072               ***
074072               * FIRST THE GROUP 1 ROUTINES
074072               *
074072               CMDOUT   BSS 0
074072               * This routine sends one command byte to the IOP.
074072               * FIRST SEND COMMAND
074072 316 247 170            JSB =CMDHS
074075               * NOW LOOP UNTIL BUSY=OBF=0
074075               *NOTE: This routine does no interrupt control I O.
074075               *  IO protocol dealing with interrupts
074075               * is  not implemented.
074075               * If the IOP is busy, then the MS ROM waits.
074075               *       FALL THROUGH!!!
```

```
                LOOP ROUTINES
074075                  * THE Utility routine OBFCK waits for BUSY + OBF
074075                  * (bits 1 and 7) both to become 0.
074075                  OBFCK   BSS 0
074075 100 034 243              STM R0,R34
074100                  OBFLOP  BSS 0
074100 316 220 170              JSB =OCKLOP
074103 206                      LRS R#              BUSY IS NOW BIT 0
074104 362 372                  JOD OBFLOP            LOOP IF BUSY=1
074106 236                      RTN                   OTHERWISE RETURN
074107                  CNTERR  BSS 0
074107 146 260 147              LDBD R46,=PRDVFG    ONLY TIME SET IS DURING
074112 207
074113 266 103                  JNZ OCKLOP              PRNTR DRVR FROM OCKLOP
074115 316 002 210              JSB  =NSTIME            TIME-OUT HOOK
074120                  * "OUT-TO-LUNCH" IOP
074120                  *
074120                  *   1. RESET OFFENDING CARD (HOLD 50MICSECS)
074120                  *   2. TIME-OUT FOR IOP TO LOGIN INT
074120                  **
074120                  *  NOTE: ASSUME R26 HAS CCR ADDR FOR BAD CARD
074120                  *
074120 146 250 200              LDB R46,=200  10 000 000 PATTERN (RESET)
074123 026 246                  STBD R#,R26   PUT IN CCR
074125 250 100                  LDB R#,=100
074127 212              LOPW+0  DCB R#
074130 266 375                  JNZ LOPW+0    WAIT LOOP HOLDING RESET
074132 246                      STBD R#,R#    RELEASE RESET LINE
074133                  * NOW WAIT FOR IOP TO INTERRUPT AND LOGIN
074133 316 011 170              JSB =WAIT20
074136                  * RESET EOL COUNT TO 2  (DEFAULT)
074136 250 002                  LDB R#,=2
074140 262 214 207              STBD R#,=SPAR#
074143                  * IF IDENTIFY THEN RETURN
074143                  *              ELSE ERROR
074143                  * THIS ROUTINE EXAMINES THE R.A. STACK (R6)
074143                  * LOOKING FOR A R.A. IN IDENTIFY.  IF FOUND THEN
074143                  * THIS ROUTINE RETURNS TO THAT ADDRESS, ELSE
074143                  * IT REPORTS A DISC-TIME-OUT ERROR.
074143 146 251 141              LDN R46,=RTNADD   R.A. FROM IDENTIFY
074146 175
074147 156 251 126              LDN R56,=RTNAD2   R.A. FROM IDENTIFY
074152 175
074153 106 072 243              STM R6,R72
074156 145 250 006              LDB R45,=6
074161 124 072 343 LO.+P        POND R24,-R72
074164 046 301                  CMM R24,R46
074166 267 014                  JZR IORTN
074170 056 301                  CMM R24,R56
074172 267 010                  JZR IORTN
074174 145 212                  DCB R45
074176 266 361                  JNZ LO.+P
074200                  **********ERROR !!!!!!!!!!!
074200 316 212 161              JSB =MSERR
```

```
                    LOOP ROUTINES
 074203 027                         OCT 31D          "TIME-OUT"
 074204                    *-------------------------
 074204                    IDRTN    BSS 0     RETURN TO IDENTIFY
 074204                    *                  FIX R6
 074204 106 072 241                 LDM R6,R72
 074207 146 006 345                 PUMD R46,+R6
 074212 137 210                     ICR R37
 074214                    *                        TIME-OUT (FALSE)
 074214 236                         RTN                AND USE IT
 074215                    *-----------------------------------------
 074215                    *-----------------------------------------
 074215                    *
 074215                    * THE ROUTINE OCK CHECKS OBF WAITING FOR THE
 074215                    * OUTPUT BUFFER TO EMPTY (OBF=0)
 074215                    OCK      BSS 0
 074215 100 034 243                 STM R0,R34
 074220                    OCKLOP   BSS 0
 074220 134 213                     DCM R34     DECREMENT COUNT
 074222 367 263                     JZR CNTERR    IF COUNT EXPIRED THEN ERROR
 074224 175 026 244                 LDBD R75,R26    READ PSR
 074227 364 367                     JNG OCKLOP      LOOP IF OBF=1
 074231 236                         RTN
 074232                    *-----------------------------------------
 074232                    *-----------------------------------------
 074232                    *
 074232                    * ROUTINE ICK WAITS FOR THE INPUT BUFFER TO
 074232                    * FILL (I.E. FOR IBF = 1)
 074232                    ICK      BSS 0
 074232 100 034 243                 STM R0,R34
 074235                    ICKLOP   BSS 0
 074235 134 213                     DCM R34     DECREMENT TIME-OUT COUNT
 074237 367 246                     JZR CNTERR  IF COUNT EXPIRED THEN ERROR
 074241 175 026 244                 LDBD R75,R26      READ PSR
 074244 363 367                     JEV ICKLOP        LOOP IF IBF = 0
 074246 236                         RTN
 074247                    *-----------------------------------------
 074247                    *-----------------------------------------
```

```
>>>>>>>>      Command Handshake  <CMDHS>                                              <<<<<<<<
3741 074247                   CMDHS    BSS 0
3742 074247                   * This routine sends one command byte from R46 to
3743 074247                   * the IOP.
3744 074247                   * MAKE SURE THE IOP IS NOT BUSY AND OBF IS EMPTY
3745 074247 316 075 170                JSB =OBFCK
3746 074252                   * SET COM=1 (I.E. IS COMMAND BYTE IN OBF)
3747 074252 250 002                   LDB R#,=2          BINARY 00000010
3748 074254 246                       STBD R#,R#         STORE TO CCR
3749 074255                   * NOW SEND COMMAND BYTE
3750 074255 146 076 246                STBD R46,R76
3751 074260 236                       RTN
```

```
                        Data Out (DATOUT)
074261                  CMDCED  BSS 0
074261 100 223                  CLM R0
074263                  DATOU+  BSS 0
074263                  *  Sends one command byte to the IOP (from R46)
074263                  * and follows it with one or more data bytes
074263                  * (len in R57) This routine is used to implement
074263                  * data transfer IOP protocols, where the IOP is
074263                  * to transfer      bytes to the HPIB
074263                  * This routine is also used in wait for data
074263                  * commands dealing only with IOP communication.
074263                  *     R24,25 are used to store DATA BUFFER add.
074263                  * FIRST SEND THE COMMAND BYTE
074263 316 247 170              JSB =CMDHS
074266                  * NOW LOOP UNTIL OBF=0 (OUTPUT BUFFER IS EMPTY)
074266 316 215 170              JSB =OCK        CHECK OBF
074271                  * NOW SET COM=CED=0
074271 222                      CLB R#
074272 246                      STBD R#,R#          STORE TO CCR
074273 124 066 241              LDM R24,R66
074276 236                      RTN
074277                  ********************************************
074277                  DATOUT  BSS 0
074277 316 263 170              JSB =DATOU+
074302                  * FETCH DATA BYTE
074302                  LOP-04  BSS 0
074302 146 024 340              POBD R46,+R24       POP DATA BYTE
074305 157 212                  DCB R57             DECREMENT COUNTER
074307 267 010                  JZR DONOUT          IF COUNTER=0 THEN DONE
074311                  * WAIT FOR OBF=0
074311 316 215 170              JSB =OCK            CHECK OBFBF=1
074314                  * WRITE DATA TO OUTPUT BUFFER
074314 146 076 246              STBD R46,R76        STORE DATA TO OBUFFER
074317 260 361                  JMP LOP-04          GET NEXT DATA BYTE
074321                  * WAIT FOR OBF=0
074321                  DONOUT  BSS 0
074321 316 215 170              JSB =OCK        TIME-OUT OBF CHECK
074324                  * SET CED = 1 (CED IS BIT 2)
074324 250 004                  LDB R#,=4       00 000 100
074326 246                      STBD R#,R#      STORE TO CCR
074327                  * WRITE DATA
074327 146 076 246              STBD R46,R76        STORE TO OBUFFER
074332                  * WAIT FOR OBF=BUSY=0
074332 316 075 170              JSB =OBFCK
074335                  * RESET CED TO 0
074335                  OPLB++  BSS 0
074335 157 222                  CLB R57
074337 026 246                  STBD R#,R26
074341 236                      RTN
074342                  ********************************************
074342                  *   PATH TO ERROR RECOVERY
074342                  ERR--R  BSS 0
074342 104 251 106              GTO CNTERR
074345 170
```

```
>>>>>>          DATAIN                                              <<<<<<<
                        DATAIN   BSS 0
                        *  Sends one command byte to IOP
                        *  then accepts data bytes
                        *  and stores them in the RAM data buffer.
                        *  Assumes R57 has
                        *  expected number of bytes.
                        *     Uses R24,25 as temp DATA BUFFER address area
                        * FIRST SEND THE  COMMAND BYTE
074346 316 263 170              JSB =DATOU+
                        * WAIT FOR IO BUFFER TO FILL
074351                  LOP-10   BSS 0
074351 316 232 170              JSB =ICK          TIME-OUT IBF CHECKBIT 0=0
                        * CHECK PED (PROCESSOR END DATA = BIT 2)
074354                          LDBD R#,R#       READ CCR
074355 206                      LRB R#               CCR BIT2 >>> BIT1
074356 206                      LRB R#               CCR BIT1 >>> BIT0
074357 362 016                  JOD DONIN            PED=1 IMPLIES DONE
                        * READ IB AND CHECK IF LAST BYTE DESIRED
074361 146 076 244              LDBD R46,R76         READ IBUFFER
074364 024 344                  PUBD R46,+R24        PUSH ON DATA STACK
074366 157 212                  DCB R57              DECREMENT COUNTER
074370 367 016                  JZR STTCED           IF COUNT=0 THEN DONE
                        * FAKE WRITE TO OBUFFER
074372 146 076 246              STBD R46,R76         STORE TO OBUFFER
074375 260 352                  JMP LOP-10         LOOP TO GET NEXT BYTE
074377                  DONIN    BSS 0
                        * READ IBUFFER AND STORE TO DATA BUFFER
074377 146 076 244              LDBD R46,R76         READ OBUFFER
074402 024 344                  PUBD R46,+R24        STORE TO DATA BUFFER
074404 157 212                  DCB R57              DECREMENT COUNT
074406 260 004                  JMP PT..1.           FINISH PROTOCOL
074410                  * SET CED = 1
074410                  STTCED   BSS 0
074410 250 004                  LDB R#,=4
074412 026 246                  STBD R#,R26          STORE TO CCR
074414                  * NOW WAIT FOR BUSY=0
074414                  PT..1.   BSS 0
074414 134 000 241              LDM R34,R0               LOAD TIME-OUT COUNT
074417 175 026 244 LOP-11       LDBD R75,R26         READ PSR
074422 134 213                  DCM R34              DECREMENT COUNT
074424 367 314                  JZR ERR--R           IF STILL > 0 THEN GOON
074426 175 206                  LRB R75              BUSY=BIT1 >>> BIT0
074430 362 365                  JOD LOP-11           LOOP IF BUSY = 1
074432                  * RESET CED TO 0
074432                  *       CLB R#
074432                  *       STBD R#,R#      STORE TO CCR
074432                  * RESET DATA BUFFER ADDRESS (R65)
074432 260 301                  JMP OPLB++
```

```
>>>>>          BSTIN Burst-in loop                                        <<<<<<<<
074434                        * This routine sends one command byte,
074434                        * then enters an
074434                        * infinite burst-in loop.  CAPRICORN relies on
074434                        * an interrupt from the IOP
074434                        * (after one sector is transferred) to
074434                        * escape the Burst-in loop.
074434                        * SEND COMMAND BYTE
074434                        BSTIN-   BSS 0
074434 316 072 170                 JSB =CMDOUT
074437                        * HALT NON-BURST IOPS TO PREVENT STRAY INTERRUPTS
074437 316 022 172                 JSB =HLTIOP
074442                        * NOW ENTER INFINITE (19 CYCLE) BURST-IN LOOP
074442 130 261 332                 LDMD R30,=INCRA
074445 203
074446 030                         ARP 30
074447 146 260 224                 LDBD R46,=DFLAG
074452 210
074453 236                         RTN
074454                        BSTIN    BSS 0
074454 316 034 171                 JSB =BSTIN-
074457 367 016                     JZR BRSTBN
074461                             DRP !46
074461 260 271 203                 LDBD R46,=FILTYP
074464 310 010                     CMB  R46,=10
074466 367 007                     JZR BRSTBN
074470                        * DO DUMMY WRITE TO IOBUFFER TO START BURST
074470 246                         STBD R#,R#                 DUMMY WRITER
074471 244                    LOP-23   LDBD R#,R#            READ IBUFFER
074472 272 315 377                 STBI R#,=PTR2-        PUSH DATA
074475 760 372                     JMP LOP-23            LOOP
074477                        *
074477 246                    BRSTBN   STBD R#,R#
074500 244                    LOP-2B   LDBD R#,R#
074501 272 316 377                 STBI R#,=PTR2+
074504 760 372                     JMP  LOP-2B
074506                        * THERE IS NO RETURN FROM THIS ROUTINE
```

```
                        BSTOUT Burst-out loop                          <<<<<<<<
074506                  * This routine sends one command byte,
074506                  * then bursts out one sector
074506                  * (256 bytes) to the disk.  Again, the burst-out
074506                  * loop is infinite and will be escaped by an IOP
074506                  * interrupt.
074506                  * FIRST SEND ONE COMMAND BYTE
074506                  BSTOUT   BSS  0
074506 316 034 171              JSB  =BSTIN-
074511 367 015                  JZR  BSTBIN
074513                          DRP  !46
074513 260 271 203              LDBD R46,=FILTYP
074516 310 010                  CMB  R46,=10
074520 367 006                  JZR  BSTBIN
074522                  ************************ ENTER BURST OUT LOOP
074522 270 315 377 LOP-21       LDBI R#,=PTR2-                  POP BYTE
074525 246                      STBD R#,R#              STORE TO STRING
074526 760 372                  JMP  LOP-21             LOOP
074530                  *
074530 270 316 377 BSTBIN        LDBI R#,=PTR2+
074533 246                       STBD R#,R#
074534 760 372                   JMP  BSTBIN
074536                  * THIS ROUTINE DOES NOT HAVE A RETURN
```

```
3914 074536                    * THIS SUBROUTINE ADDS THE UNIT (P22) TO
3915 074536                    * AN HPIB PRIMARY IN R46.  R46 THEN HAS PARITY
3916 074536                    * ADDED.  PRIMARY AND SECONDARY IN R46-7 ARE
3917 074536                    * PLACED IN THE DATA BUFFER, THEN SENT OVER
3918 074536                    * THE HPIB WITH ATN SET TO 1.
3919 074536                    OPSU#1   BSS 0
3920 074536 146 022 302                 ADB R46,R22
3921 074541 316 254 172                 JSB =LLPARI          ADD PARITY
3922 074544                    OPSU#2   BSS 0
3923 074544 146 066 247                 STMD R46,R66      PUT IN DATA BUFFER
3924 074547 250 260                     LDB R46,=OUT.CO     SEND WITH ATN=1
3925 074551 157 250 002        D__OUT   LDB R57,=2          SEND TWO BYTES
3926 074554 316 277 170                 JSB =DATOUT          SEND
3927 074557 236                         RTN
3928 074560                    *********************************************
3929 074560                    S2#3     BSS 0
3930 074563 316 104 173                 LDB R57,=2
3931 074566                             JSB =OPSU#3
3932 074566                    *********************************************
3933 074566 146 251 137        UNTALI   BSS 0
3934 074571 077                         LDM R46,=137,77
3935 074572 316 144 171                 JSB =OPSU#2
3936 074575 236                         RTN
```

```
                        ISR Interrupt Service Routine
                        * The Interrupt Service Routine should be
                        * entered ONLY to complete BURST mode and
                        * LOGIN operations, and IOP error report.
                        * Any other entry
                        * context results in an error and abort.
                        * FIRST READ THE SC FROM 177500
                        ISR     BSS 0
                                PUMD R40,+R6        SAVE REGISTERS
                                BCD
                                ELB R40
                                PUBD R40,+R6          SAVE E
                                BIN
                        *       LDMD R40,=MSBASE    GET OUR RAM BASE
                                CLM R42             GENERATE 120,377
                                DCM R42
                                LDBD R42,=INTRSC
                                STB R42,R47         SAVE A COPY
                                CLB R46
                                ICB R46             MASK FOR IOBITS
                                ANM R47,=16         ISOLATE SC
                                STB R47,R44         SAVE A COPY FOR ERROR
                                JZR R460K           MASK OK
                                LRS R47             SC*2/2
                        SHFT46  LLB R46
                                DCB R47             GENERATE MASK
                                JNZ SHFT46
                        R460K   BSS  0
                                CLB  R40          TIMEOUT IN 9MS
                        R460+   BSS  0
                                DCB  R40
                                JZR  IBFERR      NO IBF=1
                                LDBD R47,R42
                                JEV R460+           WAIT FOR IB
                                POBD R47,+R42       BUMP ADDRESS
                                LDBD R47,R42        FETCH BYTE
                                DCM R42             BACK UP ADDRESS
                                ICB R45             1ST OR 2ND TIME
                                JZR R460K           JIF 1ST TIME
                                DCB R47             CHECK FOR ERROR,LOG IN
                                JNZ ISRERR
                                POMD R41,-R6        CLEAN UP STACK BY SEVEN
                                POMD R41,-R6        CLEAN UP STACK BY SEVEN
                                POMD R41,-R6        CLEAN UP STACK
                        RE--EN  STBD R#,=INTRSC
                                STBD R#,=GINTEN
                                RTN

                        ********************************
                        *  IBF NEVER EQUALED ONE
                        IBFERR  BSS  0
                                LDB  R#,=376        NEGATIVE TWO
                                JMP  ISE+++
                        ********************************************
```

```
                        ISR Interrupt Service Routine
074705                  ISREPR  BSS 0
074705                  * DECREMENT R45 TWICE TO SEE IF IT WAS A
074705                  * TYPE 3 INTERRUPT (EOL E.G. POWER ON)
074705 206                      LRB R#
074706 206                      LRB R#
074707 367 066                  JZR CSSSSG   IF TYPE 3 INTERRUPT
074711                  *                    THEN LOGIN CARD
074711                  * ERROR!!  SET ERRSC TO ERROR CARD'S NO
074711                  *
074711 310 070                  CMB R#,=70      WAS ERROR NEGATIVE?
074713 372 002                  JNC ISE+++      JIF NO
074715 312 300                  ADB R#,=300     PUT UPPER BITSD BACK
074717                  ISE+++  BSS 0
074717 312 014                  ADB R#,=14      12D OFFSET
074721 262 220 207              STBD R#,=OCTER#     SET ERROR #
074724 144 206                  LRB R44
074726 312 003                  ADB R44,=3      BUMP FOR 3-10 SC
074730 262 141 202              STBD R44,=ERRSC SET ERROR SCSC
074733 260 123 200              LDBD R44,=ERRORS  GET ERROR FLAG AND CHECK
074736 266 005                  JNZ NOE+++  IF ERROR ALREADY PRESENT
074740 250 320                  LDB R44,=MYROM#
074742 262 120 200              STBD R44,=ERRROM
074745                  NOE+++  BSS 0
074745                  *       LDM  R46,=ERRRAM
074745                  *       JSB X46,ZRO      ERR ROUTINE IN RAM
074745 316 215 207              JSB  =ERRRAM     ERR ROUTINE IN RAM
074750 316 257 150              JSB  =CLRFLG     CLR LOAD/STORE BRST FLG
074753 146 261 072              LDMD R46,=INTPR-   FETCH ADDR FROM R6
074756 204
074757 311 256 001              CMM R46,=INTRAD     CAME FROM INTERP?
074762 266 024                  JNZ LOP-32        JIF NO
074764 106 251 074              LDM R6,=INTPR6     INTERPRETER RTN
074767 204
074770 316 272 171              JSB =RE--EN      RE-ENABLE CARDS
074773 104 251 377              GTO ROMRTN
074776 377
074777                  CSSSSG  BSS 0
074777                  *  LOGIN INTERRUPTING IOP
074777                  *     SET RELEVANT BIT IN IOBITS (GLOBAL) AND
074777                  *     REENTER FOR END OF ISR SERVICING AT LOP-32
074777                  *
074777                  *  SC IS ASSUMED IN R47.  SHIFT 00 000 001 MASK
074777                  *  LEFT WHILE DECREMENTING SC  THEN MERGE MASK
074777                  *  WITH IOBITS.
074777 147 260 140              LDBD R47,=IOBITS   GET OLD IOBITS
075002 202
075003 046 224                  ORS R47,R46        OR WITH THIS SC'S MASK
075005 262 140 202              STBD R47,=IOBITS   PUT BACK IN IOBITS
075010 140 006 342 LOP-32       POBD R40,-R6       RESTORE E
075013 231                      BCD
075014 202                      ERB R40
075015 343                      POMD R40,-R6
075016 262 100 377              STBD R#,=INTRSC
```

>>>> ISR Interrupt Service Routine <<<<<<<<<

075021 236 RTN

0040 075022 * ISR CLEANUP

```
                        SRTIOP
075022                  HLTIOP  BSS 0
075022                  *  The routine HLTIOP suspends all IO Processors
075022                  *  on line, except the IOP of the active msus.
075022                  *  This prevents stray interrupts which would
075022                  *  interfere with the burst-transfer procedure.
075022                  *  HLTIOP interrupts each non-primary IOP and
075022                  *  leaves that IOP in mid-protocol waiting
075022                  *  for Capricorn.  The sister routine SRTIOP
075022                  *  finishes the protocols begun by HLTIOP and
075022                  *  thus releases the non-primary IOPs.
075022 235                       CLE
075023 316 166 172               JSB =DONSRT
075026 260 002                   JMP CO++-+
075030                  SRTIOP   BSS 0
075030 235                       CLE
075031 234                       ICE
075032                  CO++-+   BSS 0
075032                  *  This routine restarts all except the active
075032                  *  SC's IOP.  They are assumed to have been
075032                  *  suspended by the routine HLTIOP.  This routine
075032                  *  finishes the protocol begun by HLTIOP  and
075032                  *  thereby starts those IOPs again.
075032                  *  SRTIOP is called after each sector burst, just
075032                  *  as HLTIOP is called before each sector burst.
075032 156 260 140               LDBD R56,=IOBITS    GET IOP LOGIN VECTOR
075035 202
075036                  *
075036                  *  FOR SC = 0 TO 7 DO
075036                  *
075036 165 250 377               LDB R65,=377   START COUNT AT -1
075041 165 210           SRTLOP  ICB R65
075043 310 010                   CMB R65,=8
075045 267 115                   JZR DONHSI     TEST FOR LAST SC
075047 156 220                   TSB R56
075051 263 105                   JEV SRWIND     TEST FOR LOGGED-IN IOP
075053 316 165 177               JSB =SETMSU    SET ACTIVE MSUS
075056 145 065 300               CMB R45,R65
075061 267 075                   JZR SRWIND     TEST FOR ACTIVE SC
075063                  *
075063                  *  IS AN IOP TO RESTART-HALT
075063                  *
075063                  *  STROBE INT
075063 240                       LDB R#,R#
075064 316 222 172               JSB =SETCC-
075067 157 250 001               LDB R57,=1
075072 262 001 377               STBD R57,=GINTDS
075075 026 246                   STBD R57,R26   PUT A 1 IN CCR
075077 371 006                   JEZ CIR.01
075101                  *
075101                  *  SMALL WAIT LOOP
075101 250 100                   LDB R57,=100
075103 212              WA--LP   DCB R#
075104 266 375                   JNZ WA--LP
```

```
                SRTIOP
                *
                *  RESET INT TO 0
                        STBD R#,R#
                *
                *  WAIT FOR PACK =0
                CIR.01  BSS 0
                        LDN R34,R0
                PACK=1  BSS 0
                        DCN R34
                        JNZ CN--OK
                        GTO CNTERR    IF TIME-OUT THEN ERROR
                CN--OK  BSS 0
                        STBD R#,=GINTEN
                        DRP 47
                        JEZ CIR.02
                        LDBD R47,R26   READ PSR
                        ANM R47,=10
                        JZR SRWIND
                        JMP PACK=1
                CIR.02  BSS 0
                        DRP 147
                        LDBD R47,R26
                        ANM R47,=10
                        JZR PACK=1
                        LDBD R46,R76    TRASH IB
                        LDB R46,=60
                        STBD R46,R76    SEND COMMAND
                        LDB R#,=2
                        STBD R#,R26     RESET INT =0  AND COM=1
                *
                *   NOW PACK = 0
                *
                *  DONE WITH THIS IOP, GO ON TO NEXT ONE
                SRWIND  BSS 0
                        LRB R56
                        JMP SRTLOP
                **********************
                DONHSI  JEZ SETCR
                *******END OF FOR LOOP
                DONSRT  BSS 0
                * DISABLE-REENABLE TIMERS AND KEYBOARD
                * INTERRUPTS.
                        JEZ CIR.03
                        LDB R55,=2
                        JSB =TINW-+
                        LDB R55,=1
                        JMP CIR.04
                CIR.03  LDB R55,=1
                        JSB =TINW-+
                        LDB R55,=2
                CIR.04  BSS 0
                        DRP 155
```

```
                    SRTIOP
 075214 262 302 377          STBD R55,=KEYSTS
 075217                *
4149 075217            *  RESET CCR AND RETURN
4149 075217            SETCCR  BSS 0
4150 075217 316 165 177        JSB =SETMSU
4151 075222 126 251 120 SETCC- LDM R26,=120,377
4151 075225 377
4152 075226 045 302            ADB R26,R45
4153 075230 302                ADB R26,R45
4154 075231 176 026 241 SETSC+ LDM R76,R26
4155 075234 211                ICM R76
4156 075235            *
4157 075235 236                RTN
4158 075236            ********************************
4159 075236            TIMW-+  BSS 0
4160 075236            * OPTIMIZATION LOOP DISABLES-ENABLES THE
4161 075236            * FOUR TIMERS
4162 075236 173 250 003         LDB R73,=3
4163 075241 316 377 377 GO++LP JSB =TIMWST
4164 075244 155 312 100        ADS R55,=100
4165 075247 173 212            DCB R73
4166 075251 766 366            JNZ GO++LP
4167 075253 236                RTN
4168 075254            ********************************
```

```
                    LLPARI Parity
075254                  LLPARI  BSS 0
075254                  *  This routine adds parity to R46 (using bit 7),
075254                  *  FIRST, COUNT THE ONES IN R46 (MOD 2).
075254 157 222                  CLB R57         NO ONES TO START
075256 156 046 240              LDB R56,R46     GET WORKING COPY
075261 767 010          LOP-70  JZR DONCNT      IF WORKING COPY IS = 0
075263                  *                          THEN DONE COUNT
075263 763 002                  JEV DONTCM         IF BIT 0 = 0 GOON
075265 157 216                  NCB R57         ADD 1 TO COUNT (MOD 2)
075267                  DONTCM  BSS 0           DON'T COMPLEMENT COUNT
075267 156 206                  LRB R56       GET NEXT BIT TO CHECK
075271 760 766                  JMP LOP-70
075273                  * NOW ADD UPPER 1 BIT IF ODD PARITY (TO CREATE
075273                  * EVEN PARITY IN R46)
075273                  DONCNT  BSS 0
075273 157 220                  TSB R57          TEST R57 (COUNT OF PARITY)
075275 763 005                  JEV DONPAR       IF ALREADY EVEN PARITY
075277                  *                           THEN DONE
075277 250 200                  LDB R#,=200
075301 146 057 224              ORS R46,R57         ELSE SET BIT 7 TO 1
075304                  DONPAR  BSS 0
075304                  * AND FINALLY, RETURN
075304 236                      RTN
```

```
4194 075305            *************************************************
4195 075305            ***                                            ***
4196 075305            ***    GROUP TWO                               ***
4197 075305            ***       HP-IB drivers                        ***
4198 075305            ***                                            ***
4199 075305            *************************************************
4200 075305            ***
4201 075305            * Now the group two routines.
4202 075305            * First the utility initilizing routine LLINIT.
4203 075305            * LLINIT creates the reg. state necessary to use
4204 075305            * the rest of the low-level package.
4205 075305            *      LLINIT also initialized the input and output
4206 075305            * conditions of the IOP.  (This is done with:
4207 075305            *         WRITEREG        0,0   (NULL OUTPUT EOL)
4208 075305            *     and set IOP parity to no parity
4209 075305            LLLNIT   BSS 0
4210 075305 230                 BIN
4211 075306            * NOW LOAD THE DATA BUFFER BASE ADDRESS
4212 075306 166 251 135         LDM R66,=DATAB      GET DATA BUFFER
4212 075311 207
4213 075312            * NOW SET THE CCR/PSR ADDRESS
4214 075312            * GET ACTIVE MSUS AND ADD SC TO ADDRESS
4215 075312            * AND THE I/O BUFFER ADDRESS
4216 075312 316 217 172         JSB =SETCCR
4217 075315            * SET THIS ADDRESS IN INCRA FOR BURST USE
4218 075315 176 263 332         STMD R76,=INCRA   INCRA >> IOBUFFER ADD
4218 075320 203
4219 075321            * SET THE UNIT AND DEVICE IN R23 AND R22
4220 075321 146 022 243         STM R46,R22
4221 075324            * GET TIME-OUT COUNT
4222 075324 100 251 300         LDM R0,=300,10
4222 075327 010
4223 075330 236                 RTN
4224 075331
4225 075331
4226 075331            LLINIT   BSS 0
4227 075331 316 305 172         JSB =LLLNIT
4228 075334            * TEST FOR LONG OR SHORT INIT
4229 075334            * SHORT PROTOCOL IF IOP STATE HAS ALREADY
4230 075334            * BEEN ESTABLISHED.
4231 075334 152 220             TSB R52
4232 075336 366 051             JNZ NOWRR1
4233 075340            * WRITE CONTROL TO REG 0 (DDDD 0100) TO SET
4234 075340            *  EVEN PARITY
4235 075340 146 222             CLB R46
4236 075342 316 033 173         JSB =OPSU#4
4237 075345            * SET BURST COUNT
4238 075345 146 251 000         LDM R46,=0,1
4238 075350 001
4239 075351 066 247             STMD R46,R66
4240 075353 250 231             LDB R46,=231
4241 075355 316 151 171         JSB =D__OUT
4242 075360            * POINT TO IOP EOL REG (INTERNAL REG 700CT)
```

```
075360 146 250 070             LDB R46,=70
075363 066 246                 STBD R46,R66
075365 250 346                 LDB R46,=346  WRITE AUX POINTER
075367 316 024 173             JSB =DATO1
075372               * READ AND SAVE EOL COUNT AND EOI ENABLE
075372 146 250 167             LDB R46,=167
075375 157 250 001             LDB R57,=1
075400 316 346 170             JSB =DATAIN
075403 170 066 244             LDBD R70,R66    SAVE EOL COUNT
075406 262 214 207             STBD R70,=SPAR#    SAVE
075411               * WRITE OUR EOL COUNT (0) AND ENABLE EOI
075411               NOURR1    BSS 0
075411 146 250 200             LDB R46,=200
075414 316 066 175             JSB =WR_EOL     SEND
075417 236                     RTN
075420               *
075420               * FINALLY, RETURN
075420 066 246       OUTCH1    STBD R#,R66       PUT IN DATA BUFFER
075422 250 260                 LDB R#,=OUT.CO    SEND WITH ATN=1
075424 157 250 001   DATO1     LDB R57,=1
075427 316 277 170   DATO      JSB =DATOUT
075432 236                     RTN
075433               **************
075433               OPSU#4    BSS 0
075433 146 066 246             STBD R46,R66
075436 250 200                 LDB R46,=200
075440 360 362                 JMP DATO1
```

```
                        FORM.T
    075442                      * The first group two routine is FORMAT,
    075442                      * which prepares a diskette for use
4273 075442                     * (detecting bad tracks and renumbering).
4274 075442                     * The structure of this command is:
4275 075442                     *          1. send MY TALK ADDRESS and
4276 075442                     *                UNLISTEN commands
4277 075442                     *          2. send primary and secondary
4278 075442                     *          3. send staggar,unit,   ,   .
4279 075442                     *          4. send UNLISTEN  + UNTALK
4280 075442                     *
4281 075442                     FORM.T   BSS 0
4282 075442                     * SEND MY TALK ADDRESS
4283 075442 316 274 175                 JSB =SNDMTA                 SEND COMMAND
4284 075445                     * NOW SEND PRIMARY AND SECONDARY
4285 075445 146 251 040                 LDM R46,=40,154    PRIM AND SEC LISTEN
4285 075450 154
4286 075451 316 136 171                 JSB =OPSU#1
4287 075454                     * SEND DATA WITH ATN=0 (USING DATA OUT (HEX A0))
4288 075454 143 250 030                 LDB R43,=30
4289 075457 147 250 333                 LDB R47,=333
4290 075462                     * SET UNIT AND STAGGAR
4291 075462                     *   UNIT IS BYTE 2 AND STAGGER IS BYTE 4
4292 075462 144 023 240                 LDB R44,R23       LOAD UNIT
4293 075465                     *  GET STAGGAR FROM R30
4294 075465 146 260 164                 LDBD R46,=PARAMS
4294 075470 207
4295 075471                     * R45 IS THE FORMAT TYPE (202 ... HP OVERRIDE OLD)
4296 075471 143 066 247                 STMD R43,R66       STORE TO DATA BUFFER
4297 075474                     * REG 47 = 0 IS THE DATA TO FILL EACH SECTOR BYTE
4298 075474 157 250 005                 LDB R57,=5          SEND 5 DATA BYTES
4299 075477 316 104 173                 JSB =OPSU#3
4300 075502                     * WAIT FOR FORMAT TO COMPLETE
4301 075502 360 040                     JMP WAI++T
4302 075504                     ********************************************
4303 075504                     ***
4304 075504                     OPSU#3   BSS 0
4305 075504 146 250 240                 LDB R46,=OUT.DO
4306 075507 360 316                     JMP DATO
```

```
                    VERIFY
075511                      JERIFT   BSS 0
075511                      * THIS ROUTINE PERFORMS A VERIFY TO
075511                      * THE DESIRED DISC.
075511                      * THE VERIFY BEGINS AT THE CURRENT TARGET ADDRESS.
075511                      *
075511                      * SEND PRIMARY AND SECONDARY LISTEN AND CAPR T.A
075511 316 262 175                  JSB =S40350
075514                      * NOW SEND OPCODE (7) AND UNIT AND SECTOR COUNT
075514 144 251 007                  LDM R44,=7,0,177,0   LARGEST COUNT
075517 000 177 000
075522 145 023 240                  LDB R45,R23          UNIT
075525 144 066 247                  STMD R44,R66      PUT IN DATA BUFFER
075530 157 250 004                  LDB R57,=4        SEND 4 BYTES
075533 316 104 173                  JSB =OPSU#3     SEND
075536                      * AND WAIT
075536 146 251 040                  LDM R46,=40,0      WAIT 2 MINUTES (HARD DISC)
075541 000
075542 360 004                      JMP WAI++_
075544                      WAI++T  BSS 0
075544 146 251 005                  LDM R46,=5,0
075547 000
075550                      WAI++_  BSS 0
075550 316 024 174                  JSB =PPOLL
075553                      * AND FINISH
075553 316 166 171                  JSB =UNTALI
075556 236                          RTN
```

LOG WAS HAD 0 ERRORS 0 WARNINGS 894 LABELS LAST ERROR AT 0

```
ASMB 02.14.01  ================================================================
ITEM    LOC   OBJECT CODE  SRC=LOGMAS OBJ=GENGMA   9/11/1981  3:03 PM  PG123
================================================================================
>>>>>>>         INITIALIZE                                              <<<<<<<
4333 075537                INITZE   BSS 0
4334 075537                * THIS ROUTINE INITIALIZES THE CURRENT TARGET SECTOR
4335 075537                *  SETTING D BIT   ALL SECTORS ON CURRENT TRACK ARE
4336 075537                *   MARKED DEFECTIVE.
4337 075537                *
4338 075537 316 262 175            JSB =S40350
4339 075562                * SEND OPCODE (13 + D >>> 53)
4340 075562                * AND UNIT
4341 075562 147 023 240            LDB R47,R23     UNIT
4342 075565 146 250 053            LDB R46,=53     OPCODE
4343 075570 066 247                STND R46,R66    PUT IN DATA BUFFER
4344 075572                * AND FINISH INITIALIZE REQUEST
4345 075572 316 160 171            JSB =S2#3
4346 075575                * NOW SEND FAKE DATA
4347 075575 146 251 040             LDM R46,=40,140
4347 075600 140
4348 075601 316 271 175             JSB =S40+
4349 075604                * SEND FAKE DATA
4350 075604 157 250 001             LDB R57,=1
4351 075607 316 104 173             JSB =OPSU#3       SEND
4352 075612                * AND WAIT
4353 075612 316 024 174             JSB =PPOLL
4354 075615                * FINALLY, COMPLETE PROTOCOL
4355 075615 316 166 171             JSB =UNTALI
4356 075620 236                    RTN
```

)

>>> FORMAT <<<<

```
4358 075621                          *
4359 075621                          * THE HIGH-LEVEL ENTRY "HLFMT" IS HERE AS AN
4360 075621                          * OPTIMIZATION OF CODE SPACE.
4361 075621                          *
4362 075621                          HLFMT    BSS 0
4363 075621 004                      ARP 4
4364 075622 316 366 175              JSB =HLINI
4365 075625                          * THIS ROUTINE FORMATS A NEW DISCETTE
4366 075625                          * VERIFY AND INITIALIZE ARE USED TO MARK
4367 075625                          * BAD TRACKS TO BE MADE INVISIBLE.
4368 075625 316 035 175              JSB =TOID
4369 075630 144 050 242              STB R44,R50    SAVE # CYLINDERS
4370 075633 137 220                  TSB R37
4371 075635 266 163                  JNZ 00_-P4
4372 075637 316 241 175              JSB =L.D.S    REMOVE POSSIBLE HOLDOFF
4373 075642 145 250 202              LDB R45,=202
4374 075645 316 042 173              JSB =FORM.T    FORMAT DISC
4375 075650 316 377 173              JSB =DSJ      CHECK RESULT OF FORMAT
4376 075653 263 003                  JEV LLOK#2
4377 075655 316 172 177              JSB =LLERR
4378 075660                          LLOK#2  BSS 0
4379 075660                          * IDENTIFY AND GET #CYLINDERS AND # SECTORS/TRK
4380 075660                          *   SET # SPARES
4381 075660 164 250 004              LDB R64,=4
4382 075663                          * SET REFORMAT FLAG TO FALSE
4383 075663                          FMTLOP  BSS 0
4384 075663 133 222                  CLB R33
4385 075665                          * SEEK TO SECTOR 0
4386 075665 130 223                  CLM R30
4387 075667 316 112 174              JSB =SEEK
4388 075672 267 004                  JZR OK.1A
4389 075674 104 251 245              GTO BADDDD  !Schwerer Fehler bei Seek 0 nicht moglich
4389 075677 177
4390 075700                          OK.1A   BSS 0
4391 075700                          * NOW ENTER VERIFY LOOP
4392 075700                          VFYL.P  BSS 0
4393 075700 316 111 173              JSB =VERIFT
4394 075703                          * AND REQUEST DISC ADDRESS
4395 075703 316 310 175              JSB =READA
4396 075706                          * COMPARE TO TARGET LAST ADDRESS
4397 075706 145 050 304              SBB R45,R50
4398 075711 264 020                  JNG ERRVER
4399 075713                          * DISC SURFACE IS O.K., EXIT
4400 075713 133 220                  TSB R33
4401 075715 267 010                  JZR NOFM..
4402 075717 172 250 002              LDB R72,=2
4403 075722 316 042 173              JSB =FORM.T
4404 075725 260 334                  JMP FMTLOP          TRY TO VERIFY AGAIN
4405 075727                          NOFM..  BSS 0
4406 075727 316 052 175              JSB =EX__IT
4407 075732 236                      RTN
4408 075733                          ********************************************
4409 075733                          ERRVER  BSS 0
```

```
                FORMAT                                          <<<<<<<<<
 075733                 * DECREMENT SPARE TRACK COUNT
 075733 164 212                 DCB R64
 075735 764 031                 JNG BADETT
 075737                 * VERIFY ERROR, MARK DEFECTIVE TRACK
 075737 316 157 173             JSB =INITZE   INITIALIZE BAD TRACK
 075742 133 210                 ICB R33
 075744                 *                                SET REFORMAT FLAG = T
 075744                 *
 075744                 * SEEK TO NEXT SECTOR
 075744                 *   GET DISC ADDRESS
 075744 316 310 175             JSB =REQDA
 075747 147 222                 CLB R47   SECTOR << 0
 075751 146 210                 ICB R46
 075753 320 222 207             CMBD R46,=HEADCT
 075756 766 003                 JNZ OK.HHH
 075760                 * HEAD IS 2, GO TO NEXT CYLINDER
 075760 222                     CLB R46
 075761 145 210                 ICB R45
 075763                 * AND SEEK TO NEXT TRACK
 075763                 OK.HHH  BSS 0
 075763 316 106 174             JSB =SEEK2
 075766                 * AND LOOP FOR THE REST OF THE DISCETTE
 075766 760 310                 JMP VFYL.P
 075770                 ********************************
 075770                 BADETT  BSS 0
 075770 316 052 175             JSB =EX__IT
 075773 316 212 161             JSB =MSERR
 075776 035                     OCT 29D      "BAD MEDIUM"
 075777                 *---------------------------------------
```

```
>>>>>>>          DSJ Device Specified Jump                                 <<<<<<<<
075777                    * This routine accepts one DSJ byte .  The
075777                    * structure of this routine is:
075777                    *      1. UNTALK,UNLISTEN (using CMDOUT)
075777                    *      2. Primary TALK, Secondary
075777                    *      3. Send My Listen Address (CAPR addr)
075777                    *      4. Get DSJ byte (using DATAIN)
075777                    *      5. UNTALK
075777                    * Returns with DSJ byte in R46.
075777                    DSJ     BSS 0
075777                    * NOW SEND PRIMARY AND SECONDARY
075777 146 251 100                LDM R46,=100,360
076002 360
076003 316 136 171                JSB =OPSU#1
076006                    * SEND CAPRICORN LISTEN ADDRESS
076006 316 301 175                JSB =SNDMLA                        SEND
076011                    * NOW GET DATA BYTE
076011 157 250 001                LDB R57,=1        WANT ONE BYTE ONLY
076014 316 343 175                JSB =GET_BY       GET DATA
076017 144 220                    TSB R44
076021 236                        RTN
076022                    *----------------------------------------
076022 360 056            OO_-P*  JMP OO__P*
076024                    *--------------------
```

```
                        PPOLL  parallel poll
076024                  * This performs a parallel poll and returns the
076024                  * resulting byte in R46.  The structure of this
076024                  * routine is:
076024                  * 1. Interface control (parallel poll)
076024                  * 2. Shift PPOLL byte until bit device is right
076024                  * 3. If bit not set then loop,else return
076024                  * This routine actually constitutes a wait loop,
076024                  * waiting for the device specified
076024                  * by R22 to respond to Parallel
076024                  * Poll.
076024                  FPOLL    BSS 0
076024 100 044 243               STM R0,R44    TIME-OUT COUNT
076027                  LOP-40   BSS 0
076027 144 006 345               PUMD R44,+R6
076032 157 250 001               LDB R57,=1      GET ONE BYTE
076035 146 250 104               LDB R46,=OUT.S4   PARALLEL POLL
076040 316 346 170               JSB =DATAIN       GET THE PPOLL BYTE
076043 143 066 244               LDBD R43,R66      PUT IT IN R43
076046                  * CHECK FOR RELEVANT BIT SET (I.E. BIT OF DEVICE)
076046                  * THE DEVICE # IS ASSUMED IN R22.
076046                  * HERE FOLLOWS A LOOP WHICH SHIFTS THE PPOLL BYTE
076046                  * RIGHT (LOGICAL) DEVICE BITS.
076046                  *
076046                  * DETERMINE BIT RESPONSE EXPECTED
076046 144 250 007               LDB R44,=7
076051 022 304                   SBB R44,R22                SUBTRACT DEVICE
076053                  * DEVICES ARE NUMBERED S.T. DEV 7 IS BIT 0 ...
076053                  LOP-41   BSS 0
076053 367 006                   JZR SHFTDN    IF DEVICE=0 THEN DONE
076055 143 206                   LRB R43       RIGHT SHIFT
076057 144 212                   DCB R44       DECREMENT DEVICE
076061 760 370                   JMP LOP-41    LOOP
076063                  SHFTDN   BSS 0
076063 144 006 343               POMD R44,-R6
076066 213                       DCM R44
076067 764 006                   JNG PP_FAL
076071 143 220                   TSB R43
076073                  * CHECK TO SEE THAT BIT 0 (I.E. BIT DEVICE) = 1
076073 220                       TSB R43         CHECKING PPOLL BYTE
076074 763 331                   JEV LOP-40      IF NOT PRESENT POLL AGAIN
076076 236                       RTN               ELSE RETURN
076077                  ********************************
076077                  PP_FAL   BSS 0
076077 316 166 171               JSB =UNTALI      CLEAR BUS TALKERS
076102 104 251 106      OO__P*   GTO CNTERR
076105 170
076106                  ************NO RETURN**********************
```

```
                SEEK Seek sector
4511 076106                     * This routine issues a SEEK command.
4512 076106                     *     The target sector (conceptual) is given
4513 076106                     * as a binary integer in R30,31.  The UNIT to
4514 076106                     * send the SEEK to is given in R43.
4515 076106                     *     Formulae for finding the actual cylinder,
4516 076106                     * head, and sector issued by this routine are:
4517 076106                     *  CYLINDER = con-SECTOR DIV S
4518 076106                     *  HEAD= (con-SECTOR DIV  E) MOD T
4519 076106                     *  SECTOR = con-SECTOR MOD E
4520 076106                     *           WHERE:   S = SECTORS/CYL
4521 076106                     *                    E = SECTORS/TRACK
4522 076106                     *              AND   T = TRACKS/CYL = NO HEADS
4523 076106                     *
4524 076106                     SEEK2   BSS 0
4525 076106                     *
4526 076106                     *  SEEK2 PERFORMS A SEEK USING TARGET ADDRESS
4527 076106                     *  IN R44-47
4528 076106 141 222             *
4529 076110                     *
4530 076110 360 015                     CLB R41   FLAG INDICATING TARGET ADDRESS
4531 076112                     SEEK    BSS 0      ALREADY IN R44-47
4532 076112 316 166 171                 JMP PT.02-
4533 076115 316 101 175                 JSB =UNTALI
4534 076120 137 220                     JSB =TO_ID
4535 076122 266 356                     TSB R37
4536 076124 141 250 001                 JNZ OO__P*
4537 076127 144 006 345 PT.02-          LDB R41,=1  FLAG INDICATES TARGET ADDRESS IN R3
4538 076132 316 166 171                 PUMD R44,+R6    PUSH STACK FOR LATER POP
4539 076135                     *       JSB =UNTALI   UNTALK AND UNLISTEN
4540 076135                     * SEND THE PRIMARY AND SECONDARY ADDRESS NOW
4541 076135                     * SEND MY TALK ADDRESS                  IF NOT THEN
4542 076135 316 265 175                 JSB =S40            SEND
4543 076140                     * NOW PREPARE TO SEND THE SEEK INFORMATION:
4544 076140                     *    6 BYTES .....   002,UNIT,BYTE1CYLINDER,BYTE2-
4545 076140                     *                    CYLINDER,HEAD,SECTOR
4546 076140                     * THESE WILL BE BUILT IN R42-47
4547 076140 142 250 002                 LDB R42,=2                      SEEK OPCODE
4548 076143 143 023 240                 LDB R43,R23                     UNIT
4549 076146                     * IF THE ADDRESS INFO IS ALREADY IN R44-47 THEN
4550 076146                     * SEND IT DIRECTLY.  IF NOT, THEN GENERATE IT
4551 076146                     * (USE R41 FLAG TO DECIDE)
4552 076146 144 006 343                 PUMD R44,-R6     RESTORE SEEK2 ADDRESS
4553 076151 141 220                     TSB R41
4554 076153 367 064                     JZR EASYAD   JUMP IF ADDRESS IN R44-47
4555 076155                     *  LOAD PARAMETERS FROM PREVIOUS TOID
4556 076155 124 044 241                 LDM R24,R44    PUT IN R24,25
4557 076160                     * NOW CALCULATE TARGET CYLINDER (= SECTOR DIV (2*S))
4558 076160                     *   WHERE S = #SECTORS / TRACK
4559 076160 144 223                     CLM R44      START DIV AT 0
4560 076162 156 223                     CLM R56
4561 076164 125 056 242                 STB R25,R56 EXTEND SEC-CYL TO 2-BYTES
4562 076167                     DIVLOP  BSS 0
```

```
                SEEK Seek sector                                    <<<<<<<<
076167 130 056 305          SBM R30,R56    SUBTRACT ONCE
076172 364 004              JNC DONDIV
4565 076174 146 211         ICM R46        AND INCREMENT DIV
4566 076176 360 367         JMP DIVLOP      LOOP
4567 076200                 *--------------------------
4568 076200                 DONDIV   BSS 0
4569 076200                 *  DIVISION DONE, FIRST ERASE LAST SUBTRACT
4570 076200                 *  TO GET POSITIVE REMAINDER
4571 076200 130 056 303             ADM R30,R56
4572 076203                 *  NOW MOD IS IN R30,31 AND DIV IS IN R46,47
4573 076203                 *  PUT TARGET CYLINDER (DIV) IN R44,45.
4574 076203                 *  R44 HAS ALREADY BEEN CLEARED (=0).
4575 076203 145 046 240             LDB R45,R46
4576 076206                 *  NOW PUT HEAD IN R46 AND SECTOR IN R47
4577 076206 146 222                 CLB R46      TRY HEAD 0
4578 076210 147 260 222             LDBD R47,=HEADCT  # OF HEAD/CYL
4578 076213 207
4579 076214 157 210         TRAC+   ICB R57      R57 << SECTORS/TRACK
4580 076216 125 047 304             SBB R25,R47  SECTORS/CYL-(#OF HEADS)
076221 364 002              JNC HDCNT     IN CASE OF ERRORS
32 076223 366 367           JNZ TRAC+     JIF MORE TO SUBTRACT
4583 076225 130 057 300 HDCNT   CMB R30,R57
4584 076230 364 005             JNC HD__00    IF >0 THEN HEAD SHOULD BE +1
4585 076232 304                 SBB R30,R57   SUB ONE TRACKS WORTH
4586 076233 146 210             ICB R46
4587 076235 360 366             JMP HDCNT     TEST AGAIN
4588 076237                 HD__00   BSS 0
076237 047 242              STB R#,R47    STORE SECTOR NUMBER
076241                      * ALL ADDRESS INFO IS NOW READY
4591 076241                 * NOW STORE TO DATA BUFFER
4592 076241                 EASYAD   BSS 0
4593 076241 142 066 247             STMD R42,R66
4594 076244 157 250 006             LDB R57,=6       SIX DATA BYTES
4595 076247 316 104 173             JSB =DFSU#3          SEND
4596 076252                 * NOW WAIT FOR SEEK TO COMPLETE
4597 076252 316 144 173             JSB =WAI++T
4598 076255 316 377 173             JSB =DSJ
4599 076260 236                     RTN
076261                      ********************************
```

```
ASMA 03/14/81 ================================================================
 ITEM    LOC    OBJECT CODE  SRC=LCGMA2 OBJ=GENGMA   9/11/1981  3:03 PM  PG130
================================================================================
>>>>>>>         WRBUFF  write buffered                                  <<<<<<<
4600 076261              * This routine performs the burst out-in
4601 076261              * of one sector (256 bytes) from-to the string
4602 076261              * addressed by R31,30.  The structure
4603 076261              * of this routine is:
4604 076261              *      1.  UNTALK send MTA
4605 076261              *      2.  UNLISTEN,UNTALK
4606 076261              *      3.  send Primary and Secondary (LISTEN)
4607 076261              *      4.  send OPCODE,UNIT
4608 076261              *      5.  UNLISTEN,UNTALK
4609 076261              *      6.  Primary and Secondary (LISTEN,TALK)
4610 076261              *      7.  Start burst mode
4611 076261              *      8.  UNLISTEN,UNTALK
4612 076261              *  The unit number is assumed to be in R23
4613 076261              RDBUFF  BSS 0
4614 076261 136 222            CLB R36
4615 076263 360 003            JMP GO+.1
4616 076265              ********************************
4617 076265              WRBUFF  BSS 0
4618 076265 136 250 040        LDB R36,=40
4619 076270              GO+.1   BSS 0
4620 076270              * SEND PRIMARY AND SECONDARY LISTEN ADDRESSES
4621 076270 146 251 040        LDM R46,=40,151    PRIMARY AND SECONDARY
4622 076273 151
4623 076274 136 220            TSB R36
4624 076276 366 002            JNZ GO+.2
4625 076300 147 210            ICB R47
4626 076302              GO+.2   BSS 0
4627 076302              * SEND MY TALK ADDRESS
4628 076302 316 271 175        JSB =S40+          SEND
4629 076305              * NOW SEND OPCODE (10-5) AND UNIT
4630 076305 146 250 010        LDB R46,=10        PCODE
4631 076310 136 220            TSB R36
4632 076312 366 003            JNZ GO+.3
4633 076314 146 250 005        LDB R46,=5
4634 076317              GO+.3   BSS 0
4635 076317 147 023 240        LDB R47,R23        UNIT
4636 076322 146 066 247        STMD R46,R66       STORE TO DATA BUFFER
4637 076325 157 250 002        LDB R57,=2         TWO DATA BYTES
4638 076330 316 104 173        JSB =OPSU#3        SEND
4639 076333              * NOW WAIT FOR DISC CONTROLLER TO ENABLE PPOLL
4640 076333 136 220            TSB R36
4641 076335 366 005            JNZ XNOPPX
4642 076337 146 223            CLM R46
4643 076341 316 024 174        JSB =PPOLL
4644 076344 316 166 171 XNOPPX JSB =UNTALI
4645 076347              * SEND PRIMARY AND SECONDARY LISTEN ADDRESSES
4646 076347 146 251 100        LDM R46,=100,140         BURST OUT
4647 076352 140
4648 076353 036 304            SBB R46,R36          (OR IN)
4649 076355 316 136 171        JSB =OPSU#1
4650 076360              * SEND MY TALK ADDRESS
4651 076360 136 220            TSB R36
4652 076362 366 005            JNZ GO+.5
```

```
GPIB 03/14/81 ================================================================
ITEM    LOC   OBJECT CODE  SRC=LOGPAS OBJ=GENGPA   9/11/1981  3:03 PM  PG(5)
================================================================================
>>>>        WRBUFF  write buffered                                  <<<<<<<<
4653 076364 316 301 175          JSB =SNDMLA
4654 076367 760 003              JMP GO++--
4655 076371                 GO+.5    BSS 0
4656 076371 316 274 175          JSB =SNDMTA          SEND
4657 076374                 GO++--   BSS 0
4658 076374                 * NOW ENTER BURST LOOP
4659 076374 146 250 041          LDB R46,=41
4660 076377 136 220              TSB R36
4661 076401 766 005              JNZ GO+.6
4662 076403                 *        LDB R46,=41
4663 076403 316 054 171          JSB =BSTIN
4664 076406 760 005              JMP GO+.7
4665 076410                 GO+.6    BSS 0
4666 076410 146 212              DCB R46
4667 076412                 *        LDB R46,=40          HEX 20 (BURST OUTPUT)
4668 076412 316 106 171          JSB =BSTOUT          ENTER BURST LOOP
4669 076415                 GO+.7    BSS 0
4670 076415                 **************************************
4671 076415                 * THIS LOCATION IS ENTERED FROM THE ISR
4672 076415 316 030 172          JSB =SRTIOP
4673 076420 316 166 171          JSB =UNTALI          UNTALK UNLISTEN
4674 076423 136 220              TSB R36
4675 076425 367 005              JZR YNOPPY
4676 076427 146 223              CLN R46
4677 076431 316 024 174          JSB =PPOLL
4678 076434 236             YNOPPY   RTN
```

```
                        TOID Time-out identify                          <<<<<<<
076435                  TOID    BSS 0
4681 076435             * TOID IS A HIGH-LEVEL ENTRY POINT!
4682 076435             * Time out identify.  This routine performs an identify
4683 076435             * command to the device specified by ANSUS.  If the
4684 076435             * identify times-out (i.e. no device present) then
4685 076435             * LLERR returns control to TOID.  TOID returns allways
4686 076435             * therefore.
4687 076435             *
4688 076435             * TO_ID (and TOID) are called from: VOL2AD, SEEK,
4689 076435             * FORM_T , and from the high-level.
4690 076435             *   TOID returns:   R37 --  0- good
4691 076435             *                           1- no disc controller
4692 076435             *                   R44 --  # cylinders
4693 076435             *                   R45 --  # sectors / cylinder
4694 076435             *                   R46-7 --  total # sectors on disc
4695 076435             *
4696 076435 012                 ARP 12
4697 076436 316 024 176         JSB =HOOK+
4698 076441 316 331 172         JSB =LLINIT     LOW-LEVEL INIT
4[illegible] 076444 316 166 171 JSB =UNTALI
4699 076447 316 101 175         JSB =TO_ID
4701 076452             **********
4702 076452             *   THE COMMON EXIT ROUTINE EX__IT OCCURS HERE IN LINE
4703 076452             EX__IT  BSS 0
4704 076452 146 006 345         PUMD R46,+R6    SAVE R46,47
4705 076455 250 111             LDB R46,=111    RESUME IO
4706 076457 316 072 170         JSB =CMDOUT     SEND
4707 076462 146 260 214         LDBD R46,=SPAR#
4707 076465 207,
4708 076466             *         RESET EOI ENABLE AND EOL COUNT
4709 076466             WR_EOL  BSS 0
4710 076466 066 246             STBD R#,R66
4711 076470 250 220             LDB R#,=220
4712 076472 316 024 173         JSB =DATOI
4713 076475 146 006 343         POMD R46,-R6  RESTORE R46,47
4714 076500 236                 RTN
4715 076501             ********************************
4716 076501             ***
4717 076501             * SEND CAPRICORN LISTEN ADDRESS
4718 076501             TO_ID   BSS 0
4719 076501 316 301 175         JSB =SNDMLA       SEND
4720 076504             * SEND UNTALK (137)
4721 076504 146 250 137         LDB R46,=137    UNTALK
4722 076507 316 020 173         JSB =OUTCMI       SEND
4723 076512             * SEND SECONDARY PTIC DEVICE ]
4724 076512             *-----------------------------------------------
4725 076512             * DA'S MIRACLE TIME-OUT FIX:
4726 076512             *         PUMD R46,+R6
4727 076512             *         LDM  R46,=0,10
4728 076512             *TO_FIX   DRP  !46
4729 076512             *         DCM  R46
4730 076512             *         JNZ  TO_FIX
4731 076512             *         POMD R46,-R6
```

```
                TOID Time-out identify
076512                  *-----------------------------------------
076512 146 250 140              LDB R46,=140
076515 022 302                  ADB R#,R22      ADD DEVICE
076517 316 254 172              JSB =LLPARI     ADD PARITY BIT
076522 146                      DRP 46
076523 316 020 173              JSB =OUTCHI       SEND
076526                  RTNAD2  BSS 0
076526                  * GET 2 BYTES BACK
076526 157 250 002              LDB R57,=2      GET 2 BYTES
076531 146 250 020              LDB R46,=OUT.EN   ENTER BYTES FROM HP-IB
076534 137 222                  CLB R37
076536                  *                       CLEAR IDENTIFY ERROR FLAG
076536 316 346 170              JSB =DATAIN       GET ID BYTES
076541                  *                       TRY TO GET DATA
076541                  RTNADD  BSS 0           R.A. FOR ERROR RETURN FROM
076541                  *                        CNTERR (COUNT ERROR)
076541                  *
076541                  *    SAVE R53-7 AND USE TO GATHER DISC PARAMETERS
076541                  *
076541 153 006 345              PUND R53,+R6
076544 137 220                  TSB R37
076546 366 052                  JNZ TOIDER      IF TIME-OUT THEN RETURN ERROR
076550 146 066 245              LDMD R46,R66    GET IDENTIFY BYTES
076553 153 251 002              LDM R53,=2,41,40,40,4     MINI FLOPPY PARAMETER
076556 041 040 040
076561 004
076562                  *                                  2 =  2D   -- HEADS
076562                  *                                 41 = 33D   -- CYLINDERS
076562                  *                                 40 = 32D   -- SECTORS/CYLIN
076562                  *                               1056D = 2040 = 4,40
076562                  *                                      -- TOTAL SECTORS
076562 146 311 001              CMM R46,=1,4
076565 004
076566 367 034                  JZR OOOK.                 IF MINI CONTROLLER
076570 153 251 002              LDM R53,=2,113,74,224,21    8" FLOPPY PARAMETERS
076573 113 074 224
076576 021
076577                  *                                  2 =  2D  -- HEADS
076577                  *                                113 = 75D  -- CYLINDERS
076577                  *                                 74 = 60D  -- SEC/CYL
076577                  *                      4500D = 10624 = 21,224 -- SECTORS TOT
076577 146 311 000              CMM R46,=0,201
076602 201
076603 367 017                  JZR OOOK.       R40 << 1 IF 8" CONTROLLER
076605 153 251 004              LDM R53,=4,231,174,34,112 HARD DISC PARAMETERS
076610 231 174 034
076613 112
076614                  *                                  4 =  4D -- HEADS
076614                  *                                231 =153D -- CYL
076614                  *                                174 =124D -- SEC/CYL
076614                  *                   18972D = 45034 = 112,34    -- TOT SEC
076614 146 311 001              CMM R46,=1,6
076617 006
```

```
                    TOID Time-out Identity                          <<<<.<<<
4775 076620 767 002             JZR OOOK.       R40 << 2 IF QUAD DENSITY MINI
4777 076622                     *               THIS IS A HOOK FOR FUTURE USE
4778 076622                     *       SOMETHING THERE, BUT NOT A DISC CONTROLLER
4779 076622                     *       THUS, ERROR
4780 076622                     TOIDER  BSS 0
4781 076622 137 210             ICB R37
4782 076624                     *-----------------------------------
4783 076624                     OOOK.   BSS 0
4784 076624                     *   PUT TOID INFO IN R44-7 AND RESTORE R54-7
4785 076624 154 044 243             STM R54,R44
4786 076627 153 262 222             STBD R53,=HEADCT  SAVE HEAD COUNT
4786 076632 207
4787 076633 006 343                 PDMD R53,-R6
4788 076635 316 354 175             JSB =UNTA.!     SEND UNTALK AND UNLISTEN
4789 076640 236                     RTN             RETURN
4790 076641                     *-------------------------------
```

```
>>>>>          STATUS                                          <<<<<<<
4    076641                 L.D.S    BSS 0
+    076641                 **********************
4794 076641                 *  L.D.S is an optimization routine.
4795 076641 316 331 172            JSB =LLINIT
4796 076644 316 377 173            JSB =DSJ
4797 076647                 STATUS   BSS 0
4798 076647                 * The STATUS routine gets 2 status words from a
4799 076647                 * controller.  Status word 1 gives info on the
4800 076647                 * most recent operation and status 2 gives info
4801 076647                 * on the current state of a unit.
4802 076647                 * Bytes returned are:
4803 076647                 *    R44    Status 1 Byte 1
4804 076647                 *    R45    Status 1 Byte 2
4805 076647                 *    R46    Status 2 Byte 1
4806 076647                 *    R47    Status 2 byte 2
4807 076647                 *
4808 076647                 *   The comand sequence involves*
4809 076647                 *       First a REQUEST STATUS then
4810 076647                 *             a RECIEVE STATUS.
48   076647                 *
4  2 076647                 * FIRST request status
4813 076647                 *
4814 076647 316 262 175            JSB =S40350   SEND UNTALK AND UNLISTEN
4815 076652                 *
4816 076652                 *   SEND PRIMARY LISTEN AND SECONDARY
4817 076652                 *
4818 076652                 * SEND CAPRICORN TALK ADDRESS
4    076652                 * NOW SEND OPCODE AND UNIT
4820 076652                 *          WITH ATN =0
4821 076652 147 023 240           LDB R47,R23         UNIT FROM R23
4822 076655 146 250 003          LDB R46,=3        OPCODE =3
4823 076660 760 034              JMP RE9+++
```

```
>>>>>>          SNDMLA                                          <<<<<<<
4826 076662                    *
4827 076662                    * SEND MY LISTEN ADDRESS
4828 076662                    *      (ALSO INCLUDES SEND MY TALK ADDRESS
4829 076662                    *       ENTRY SNDMTA)
4830 076662 316 166 171 $40350    JSB =UNTALI
4831 076665 146 251 040 $40       LDM R46,=40,350
4831 076670 350
4832 076671 316 136 171 $40+      JSB =OPSU#1          SEND
4833 076674                    ***************    ***********************
4834 076674                    SNDMTA   BSS 0
4835 076674 146 250 105             LDB R46,=OUT.S5
4836 076677 360 003                 JMP CO++MN
4837 076701                    ***************************************
4838 076701                    SNDMLA   BSS 0
4839 076701 146 250 106             LDB R46,=OUT.S6
4840 076704                    ***************    ***********************
4841 076704                    CO++MN   BSS 0
4842 076704                    * AND SEND MY ISTEN ADDRESS
4843 076704 316 072 170             JSB =CMDOUT          SEND
4844 076707                    * AND FINALLY, RETURN
4845 076707 236                     RTN
```

```
>>>>          REQUEST DISC ADDRESS (LOGICAL)                    <<<<<<<<
 076710                 REGDA    BSS 0
 076710                 * THIS ROUTINE PERFORMS A REQUEST DISC LOGICAL
 076710                 * ADDRESS OF THE DISC CONTROLLER (LAST UNIT
 076710                 * ACCESSED).  THIS ADDRESS (CYLINDER BYTE1,
 076710                 * CYLINDER BYTE2, HEAD, SECTOR) IS RETURNED
 076710                 * IN R44-47
 076710 316 262 175              JSB =S40350  SEND UNTALK AND UNLISTEN
 076713                 * SEND PRIMARY AND SECONDARY LISTEN
 076713                 * SEND CAPR TALK ADDRESS
 076713                 * SEND OPCODE (24) AND DUMMY EOI BYTE
 076713 146 250 024              LDB R46,=24
 076716 066 247         REQ+++   STND R#,R66    PUT IN DATA BUFFER
 076720 316 160 171              JSB =S2#3
 076723                 * AND UNTALK UNLISTEN
 076723 316 144 173              JSB =WAI++T
 076726                 * NOW GET FOUR ADDRESS BYTES
 076726                 *
 076726                 * SEND PRIMARY AND SECONDARY TALK
 076726 316 301 175              JSB =SNDMLA   SEND CAPR LISTEN ADDRESS
 076731 146 251 100              LDM R46,=100,350
 076734 350
 076735 316 136 171              JSB =OPSU#1     SEND
 076740                 *
 076740                 * AND GET FOUR BYTES
 076740 157 250 004              LDB R57,=4
 076743 146 250 020 GET_BY       LDB R46,=OUT.EN
 076746 316 346 170              JSB =DATAIN      GET FOUR BYTES
 076751                 * AND TERMINATE WITH UNTALK UNLISTEN AND RETRUN
 076751 144 066 245              LDND R44,R66   GET ADDRESS
 076754 144 006 345 UNTA.I       PUND R44,+R6
 076757 316 166 171              JSB =UNTALI
 076762 144 006 343              PUND R44,-R6
 076765 236                      RTN
```

```
                        HLINI  High Level initialize
 4380 076766                   ********************************************
 4381 076766                   ***                                      ***
 4382 076766                   ***      GROUP THREE                     ***
 4383 076766                   ***         CONCEPTUAL LEVEL DISC UTILITIES ***
 4384 076766                   ***                                      ***
 4385 076766                   ********************************************
 4386 076766                   ***
 4387 076766                   * The third group of low-level routines follows.
 4388 076766                   * All routines in this group call HLINI as their
 4389 076766                   * first act, for the purpose of future retries.
 4390 076766                   * HLINI records the information (HL entry point,
 4391 076766                   * input parameter values, and retry count)
 4392 076766                   * necessary to retry following a bad disk operation
 4393 076766                   * (DSJ = 1).
 4394 076766                   HLINI   BSS 0
 4395 076766 232               .       SAD             SAVE ARP FLAG
 4396 076767 316 373 207       .       JSB =MSLOW      CALL LOW LEVEL HOOK
 4397 076772 237               .       PAD             RESTORE R6 IF WE RTN
 4398 076773                   * SET RETRY VECTOR IN R72,73,74
 4399 076773 171 222           .       CLB R71     CLEAR NO HEAD MOVES
 4900 076775 152 222           .       CLB R52     SET FULL PROTOCOL
 4901 076777 174 250 010       .       LDB R74,=10     RETRY COUNTER
 4902 077002                   * SAVE RETRY ADDRESS (FROM R.A. STACK) IN R20,21
 4903 077002 120 006 343       .       POMD R20,-R6    POP R.A.
 4904 077005 345               .       PUMD R20,+R6    RESTORE R.A. FOR RETURN
 4905 077006                   * AND SAVE INPUT PARAMETERS (SAVE R30-33)
 4906 077006 144 030 241       SAV_PA  LDM  R44,R30
 4907 077011 263 164 207       .       STMD R44,=PARAMS    SAVE R32,33
 4908 077014 145 261 314       .       LDMD R45,=PTR2
 4909 077017 377
 4910 077020 263 223 207       .       STMD R45,=PTRADR    SAVE BURST IN ADDR
 4911 077023                   * AND RETURN
 4912 077023 236               .       RTN
 4913 077024
 4914 077024
 4915 077024                   ********************************************
 4916 077024                   * EXTRA ROUTINE TO MAKE SURE THE CORRECT NUMBER  *
 4917 077024                   * OF BYTES ARE ON THE R6 STACK.                  *
 4918 077024                   ********************************************
 4919 077024                   HOOK+   BSS 0
 4920 077024 232               .       SAD         SAVE THE ARP FLAG
 4921 077025 316 373 207       .       JSB =MSLOW  CALL LOW LEVEL HOOK
 4922 077030 237               .       PAD         RESTORE IF WE RTN
 4923 077031 236               .       RTN         RTN TO CALLING ROUTINE
```

```
4000 03/14/81 ================================================================
ITEM    LOC   OBJECT CODE  SRC=LOGMA3 OBJ=GENGMA   9/11/1981  3:03 PM  PG139
================================================================================
>>>>>>          PUTSEC RAM sector >> Disk                              <<<<<<<
4926 077032                * the routine PUTSEC transfers one sector (256
4927 077032                * bytes) TO the disk (sector number specified
4928 077032                * in R32,33) FROM Capricorn RAM (address in R30,
4929 077032                * 31).  Each suboperation (SEEK,WRBUFF) is
4930 077032                * followed by a DSJ. If DSJ returns a 1 result
4931 077032                * then this routine aborts through MSERR.
4932 077032                * The structure of this routine is:
4933 077032                *      1. save R30,31, and initialize
4934 077032                *      2. SEEK target sector
4935 077032                *      3. DSJ (and exit if error)
4936 077032                *      4. WRBUFF FROM RAM specified
4937 077032                *      5. DSJ (and exit if error)
4938 077032                *
4939 077032                * ENTRY POINTS ARE:
4940 077032                *                        PUTBUF
4941 077032                *                        PUTSEC
4942 077032                *                        GETBUF
4943 077032                *                        GETSEC
4944 077032                *                        GETBU+
4945 077032                *                        GETSE+
4946 077032                *                        PUTBU+
4947 077032                *                  AND   PUTSE+
4948 077032                *
4949 077032                PUTBUF   BSS 0
4950 077032                *        LDM R55,=RECBUF    TAPE BUFFER ADDR
4951 077032                *        OCT 0
4952 077032                *        STMD R55,=PTR2     STORE BUFFER ADDR
4953 077032                *        LDM  R30,R55
4954 077032 316 214 176            JSB  =BUFF55
4955 077035 006                    ARP  6
4956 077036 760 001                JMP  PUTCOM
4957 077040                PUTSEC   BSS 0
4958 077040 010                    ARP  10
4959 077041 136 250 001 PUTCOM     LDB R36,=1
4960 077044                PGCOM+   BSS 0
4961 077044 316 366 175            JSB =HLINI
4962 077047 316 331 172            JSB =LLINIT
4963 077052                * FIRST SAVE THE RAM ADDRESS AND INITIALIZE
4964 077052 316 006 176            JSB =SAV_PA
4965 077055                *                                 SAVE TARGET SECTOR
4966 077055                * NOW SEEK AND DSJ
4967 077055 132 030 243            STM R32,R30             LOAD TARGET SECTOR
4968 077060 316 112 174            JSB =SEEK          PERFORM DISC SEEK
4969 077063 367 003                JZR OK.3A
4970 077065 316 172 177            JSB =LLERR
4971 077070                OK.3A    BSS 0
4972 077070                *        LDMD R30,=PARAMS  LOAD RAM ADDR
4973 077070                * NOW PERFORM BURST OUTPUT AND DSJ
4974 077070                LLOK     BSS 0
4975 077070 136 220                TSB R36              JIF GETSECTOR
4976 077072 367 005                JZR GETS++
4977 077074 316 265 174             JSB =WRBUFF       PERFORM WRITE BUFFERED
4978 077077 760 003                JMP PGCONT          CONTINUE
```

```
==============================================================================
>>>>>          PUTSEC RAM sector >>> Disk                                  <<<<<<<<
077101 316 261 174 GETS++   JSB =RDBUFF         PERFORM READ BUFFERED
077104 316 377 173 PGCONT    JSB =DSJ          GET OPERATION STATUS
077107 263 003               JEV LLOK2               ELSE GO-ON
077111 316 172 177           JSB =LLERR                  ERROR EXIT
077114                     * DONE SUCCESSFULLY, RETURN
077114                     LLOK2     BSS 0
077114 316 052 175                   JSB =EX__IT
077117 236                           RTN
077120                     *-------------------------------------------------------
077120                     GETBUF    BSS 0
077120                     *         LDM R45,=RECBUF
077120                     *         OCT 0
077120                     *         STMD R45,=PTR2
077120                     *         LDM  R30,R45
077120 316 214 176                   JSB  =BUFF55
077123 000                           ARP  0
077124 360 001                       JMP  GETCO+
077126                     GETSEC  BSS 0
077126                     *Same as PUTSEC but does a read from disk
077126                     *to capricorn.
077126 002                         ARP 2
077127 136 222             GETCO+  CLB R36              FLAG GETSECTOR
077131 360 311                     JMP PGCOM+
077133                     *-----------------------------------------------------------
077133                     GETBU+  BSS 0
077133 165 251 200                 LDM R65,=RECBUF
077136 205
077137 000                         OCT 0
077140 156 220                     TSB R56
077142 367 005                     JZR STPTR
077144 165 313 000                 ADM R65,=0,1,0
077147 001 000
077151 165 263 314 STPTR           STMD R65,=PTR2
077154 377
077155 130 065 241                 LDM  R30,R65
077160 001                         ARP  1
077161 360 001                     JMP  GETSCO
077163                     GETSE+  BSS 0
077163 003                         ARP 3
077164 136 222             GETSCO  CLB R36
077166                     *SAME AS GETSEC BUT NO SEEK
077166 360 012                     JMP GETP+-
077170                     *-----------------------------------------------------------
077170                     PUTBU+  BSS 0
077170                     *       LDM R55,=RECBUF
077170                     *       OCT 0
077170                     *       STMD R55,=PTR2
077170                     *       LDM  R30,R55
077170 316 214 176                 JSB  =BUFF55
077173 007                         ARP  7
077174 360 001                     JMP  PUT++
077176                     PUTSE+  BSS 0
077176 011                         ARP 11
```

```
ARMA 03/14/81 ================================================================
 ITEM    LOC    OBJECT CODE   SRC=LCGMA? OBJ=GENGMA    9/11/1981  3:03 PM  PG141
================================================================================
>>>>>>          PUTSEC RAM sector >>. Disk                          <<<<<<<
    077177 136 250 001 PUT++    LDB R36,=1
    077202                GETP+-  BSS 0
5030 077202 316 366 175           JSB =HLINI   HIGH-LEVEL INIT
5031 077205 152 210               ICB R52
5032 077207 316 331 172           JSB =LLINIT
5033 077212                   *SAME AS PUTSEC BUT NO SEEK
5034 077212 760 254               JMP LLOK
5035 077214                   *-----------------------------------------------
5036 077214                   BUFF55  BSS  0
5037 077214 155 251 200           LDM  R55,=RECBUF
5037 077217 205
5038 077220 000                   OCT  0
5039 077221 263 314 377           STMD R55,=PTR2
5040 077224 130 055 241           LDM  R30,R55
5041 077227 236                   RTN
```

```
>>>>      HLFMT   HIGH-LEVEL FORMAT                                  <<<<<<<
5043 077230              ********************************************
5044 077230              *                                          *
5045 077230              *    SEE FORMAT                            *
5046 077230              *                                          *
5047 077230              ********************************************
5048 077230              *
5049 077230              * FORMAT then DSJ and check DSJ byte.
5050 077230              *LFMT   BSS 0      HIGH LEVEL ENTRY POINT
5051 077230              * FORMAT CALLS LLINIT
5052 077230              *       JSB =HLINI   HIGH-LEVEL INIT
5053 077230              *TFORM  BSS 0        RETRY ENTRY POINT
5054 077230              *       JSB =FORMAT  PERFORM FORMAT
5055 077230              *       JMP PGCONT
```

```
                HLSEEK  HIGH-LEVEL SEEK                          <<<<<<<
077230                  *
077230                  * This routine performs a seek to the
077230                  * conceptual sector in R30,31.  The call
077230                  * to SEEK is preceeded by a call to LLINIT,
077230                  * and followed by a checking DSJ call.
077230                  HLSEEK  BSS 0
077230 005              ARP 5
077231 316 366 175      JSB =HLINI
077234 316 331 172      JSB =LLINIT
077237 316 112 174      JSB =SEEK       SEEK
077242 760 240          JMP PGCONT
077244                  *****************************************
077244                  *
077244                  *         ++++  THIS ROUTINE BELONGS TO VOL2AD ++++
077244                  *
077244                  ESCFOR  BSS 0   END OF SC FOR
077244 316 264 176      JSB =CK_-RA     CHECK FOR CALL FROM RESET
077247 766 007          JNZ ERXXX
077251 144 223          CLM R44
077253 144 263 160 PUTMSU  STMD R44,=ACTMSU     SAVE MSUS
077256 207
077257 236              RTN
077260                  ***********************************
077260                  ERXXX   BSS 0
077260 316 212 161      JSB =MSERR
077263 031              OCT 25D     "NO SUCH VOLUME"
077264                  ***********************************
077264                  CK_-RA  BSS 0
077264 144 006 343      POMD R44,-R5    GET SECOND R.A. IN STACK
077267 345              PUMD R44,+R6
077270 130 251 337      LDM R30,=RSR.A.  GET RESET PWO R.A.
077273 167
077274 044 301          CMM R30,R44     COMPARE THE TWO
077276 236              RTN
```

```
>>>>         VOL2AD Translate vol label                                   <<<<<<<<
077277                        VOL2AD   BSS 0
077277                        *  This routine accepts a volume label in ram
077277                        *  (SPECIF) and converts that to an MSUS (in
077277                        *  AMSUS).  This is accomplished by searching the
077277                        *  media present for the given volume label.  Search
077277                        *  is in order of ascending Controller address, and
077277                        *  ascending SC (thus if duplicate labels exist then
077277                        *  the one residing in the lower address MSUS will be
077277                        *  found).  If the volume label specified is not
077277                        *  then an error return occurs.
077277                        *
077277                        * FOR SC := 0 TO 7 DO
077277                        *
077277 230                             BIN
077300 142 263 127                     STMD R42,=SPECIF   SAVE TARGET LABEL IN RAM
077303 207
077304 013                             ARP  13
077305 316 024 176                     JSB  =HOOK+
077310 161 250 377                     LDB R61,=377   START COUNT AT -1
077313                        SCFOR    BSS 0     *********START OF FOR*******
077313 161 210                         ICB R61           INCREMENT COUNT
077315 310 010                         CMB R61,=8
077317 367 323                         JZR ESCFOR             AND END IF EQUAL
077321                        *  CHECK GLOBAL VECTOR IOBITS TO ENSURE THAT AN
077321                        *  IOP IS LOGGED IN FOR THIS SC
077321                        *
077321                        *  SC COPY IN R47 IS DECREMENTED AS IOBITS COPY IN
077321                        *  R46 IS LOGICAL RIGHT SHIFTED.  WHEN SC IS
077321                        *  REDUCED TO 0 THEN IOBITS MUST BE ODD.
077321 146 260 140                     LDBD R46,=IOBITS       GET COPY OPF IOBITS
077324 202
077325 147 061 240                     LDB R47,R61            GET COPY OF SC
077330 367 006                LOPBIT   JZR CKBITA        IF SC=0 THEN DONE SHIFTING
077332 146 206                         LRB R46                   ELSE SHIFT IOBITS
077334 147 212                         DCB R47                        AND DEC SC
077336 360 370                         JMP LOPBIT                     AND LOOP
077340                        CKBITA   BSS 0
077340 146 220                         TSB R46
077342 363 347                         JEV SCFOR         IF IOBITS EVEN THEN DONE
077344                        *                                 WITH THIS SC
077344                        *
077344                        *   FOR CONTROLLER-ADDRESS :=0 TO 7
077344                        *
077344                        *        CLB R52
077344 162 250 377                     LDB R62,=377   START COUNT AT -1
077347                        CONFOR   BSS 0     **********START OF CA FOR*******
077347 152 222                         CLB R52
077351 162 210                         ICB R62                INCREMENT COUNT
077353 310 010                         CMB R62,=8
077355 367 334                         JZR SCFOR                  ELSE END OF C.A. FOR
077357                        *
077357                        *        DO BEGIN (* CA FOR LOOP *)
077357                        *             TIME OUT IDENTIFY
```

```
>>>>>>      VOL2AL Translate vol label                                                  <<<<<<<<
077357                   *                   IF DISC CONTROLLER THEN
077357                   *
077357                   * store test MSUS in AMSUS for TOID
077357 160 250 001               LDB R60,=1      DISC FLAG
077362 163 222                   CLB R63         DUMMY UNIT (0)
077364 144 060 241               LDM R44,R50
077367 316 253 176               JSB =PUTMSU                       ONLY NEED SC
077372 316 305 172               JSB =LLLNIT
077375                   *****************************************************
077375                   * CODE TO DETERMINE IF I AM TALKING TO AN HP-IB   *
077375                   *****************************************************
077375                   CKHPIB  BSS 0
077375 146 222                   CLB R46
077377 157 250 001               LDB R57,=1
077402 316 346 170               JSB =DATAIN
077405 146 066 244               LDBD R46,R66
077410 310 001                   CMB  R46,=1           AM I TALKING TO HP-IB?
077412 367 004                   JZR  CKIBOK           JIF YES
077414 310 006                   CMB  R46,=6           TALKING TO LIBRA IB?
077416 366 273                   JNZ  SCFOR            JIF NO
077420                   CKIBOK  DRP  !46
077420 250 005                   LDB  R46,=5
077422 157 250 001               LDB  R57,=1
077425 316 346 170               JSB  =DATAIN
077430 146 066 244               LDBD R46,R66
077433 204                       LLB  R46
077434 204                       LLB  R46
077435 220                       TSB  R46               HP-IB?
077436 765 253                   JPS  SCFOR             JIF NO
077440 316 035 175               JSB =TOID            TIME-OUT IDENTIFY
077443                   *  TIME-OUT IDENTIFY (TOID) ALLWAYS RETURNS
077443                   *  THE SUCCESS OR FAILURE (DISC CONTROLLER OR NOT)]
077443                   *  IS INDICATED IN RAM FLAG TOIDFG
077443 137 220                   TSB R37
077445 766 300                   JNZ CONFOR      IF FAILURE (NO DISC CONTROLLER
077447                   *
077447                   *  DISC CONTROLLER FOUND
077447                   *  IF (FROM RESET)
077447                   *     THEN RETURN
077447                   *  (Look in RA stack for RSR.A.)
077447                   *
077447 316 264 176               JSB =CK_-RA
077452 766 004                   JNZ NOTRST      IF UNEQUAL THEN NOT RESET
077454 316 052 175               JSB =EX__IT
077457 236                       RTN                             ELSE RETURN
077460                   NOTRST  BSS 0
077460                   *                                 THEN (GET NEXT C.A.)
077460                   *
077460                   *                 BEGIN (* IS DISC CONTROLLER *)
077460                   *                 FOR UNIT :=0 TO 3 DO
077460                   *                     BEGIN
077460                   *                       IF STATUS O.K. THEN
077460                   *                          READ VOLUME LABEL
```

```
=================================================================================
>>>>>            VOL2AC Translate vol label                                      <<<<<<<<
077460                     *                           IF VOL LABEL MATCHES
077460                     *                              THEN RETURN SUCCESS
077460                     *                                   SET R44-47
077460                     *                           END
077460                     *                      END (* IS DISC CONTROLLER *)
077460                     *
077460 163 250 377                 LDB R63,=377   START UNIT
077463             UNFOR   BSS 0       **********START OF UNIT FOR *******
077463 163 210                     ICB R63        INCREMENT COUNT
077465 310 004                     CMB R63,=4
077467 367 256                     JZR CONFOR     IF EQUAL THEN END OF FOR
077471                     * STORE MSUS IN AMSUS
077471 160 250 001                 LDB R60,=1              DISC FLAG (TYPE)
077474 144 060 241                 LDM R44,R60
077477 316 253 176                 JSB =PUTMSU              PUT IN RAM
077502                     *                                     *
077502                     *  CHECK STATUS ON THIS UNIT
077502                     *  IF BYTE1 STATUS2 IS ODD THEN STATUS 2 ERROR
077502                     *                                GET NEXT UNIT
077502                     *
077502 316 241 175                 JSB =L.D.S     CLEAR POSSIBLE HOLDOFF
077505 152 210                     ICB R52        CHANGE TO SHORT PROTOCOL
077507 146 220                     TSB R46
077511 364 350                     JNG UNFOR     ERROR IF STATUS2 ERROR BIT SET
077513 130 223                     CLM R30
077515 316 112 174                 JSB =SEEK
077520 366 341                     JNZ UNFOR      IF NEGATIVE THEN NEXT UNIT
077522                     * AT THIS POINT WE HAVE FOUND A DISC CONTROLLER
077522                     * AND A GOOD UNIT, SO GET THE VOLUME LABEL.
077522 155 012 345                 PUMD R55,+R12      SAVE REG 45-47
077525 316 214 176                 JSB =BUFF55        SET PTR2
077530 155 012 343                 POMD R55,-R12      RESTORE R45-47
077533 316 261 174                 JSB =RDBUFF
077536                     * IF DESIRED LABEL THEN RETURN
077536 142 261 127                 LDMD R42,=SPECIF   GET TARGET LABEL
077541 207
077542 130 251 200                 LDM R30,=RECBUF
077545 205
077546 172 030 265                 LDMD R72,XR30,V.LABL   GET MEDIUM'S LABEL
077551 002 000
077553 042 301                     CMM R72,R42     COMPARE THE TWO LABELS
077555 366 304                     JNZ UNFOR       IF NOT SAME THEN GET NEXT UNIT
077557                     *************************
077557                     *
077557                     *  SUCCESSFULL RETURN >>>>>>>>
077557                     *
077557                     *    SET R44-47
077557 316 331 172                 JSB =LLINIT
077562 316 052 175                 JSB =EX__IT
077565 144 261 160 SETMSU  LDMD R44,=ACTMSU     LOAD AMSUS
077570 207
077571 236                         RTN
077572                     *
```

>>>>>>>> VOL2AD Translate vol label
077572 **********

```
                        .LERR   LOW-LEVEL ERROR RECOVERY                <<<<<<<<
5245 077572                     * This routine performs the clean-up necessary
5246 077572                     * before a retry (after a DSJ error).  First,
5247 077572                     * the retry count is incremented and tested.
5248 077572                     * If the count  is not satisfied then restore
5249 077572                     * the input parameters, and recall the high-level
5250 077572                     * entry point (from R20,21) to jump to.
5251 077572                     *
5252 077572                     * FIRST DECREMENT THE RETRY COUNT
5253 077572                     LLERR   BSS 0
5254 077572 316 247 175                 JSB =STATUS
5255 077575                     * RETURNS WITH STATUS BYTES IN R44-47
5256 077575                     * TEST STATUS 1 BYTE 1 (PUT COPY OF R47 IN R46)
5257 077575 147 046 242                 STB R47,R46
5258 077600 144 310 007                 CMB R44,=7      CYLINDER COMPARE ERROR
5259 077603 367 017                     JZR BAD...
5260 077605 310 021                     CMB R44,=21     DEFECTIVE TRACE OR SECTOR
5261 077607 367 013                     JZR BAD...
5262 077611 310 023                     CMB R44,=23     STATUS TWO ERROR
5263 077613 367 007                     JZR BAD...
5264 077615 147 317 004                 ANM R47,=4       0000 0100 SEEK CHECK BIT
5265 077620 366 002                     JNZ BAD...
5266 077622 360 033                     JMP OK..01
5267 077624                     BAD...  BSS 0
5268 077624 147 317 010                 ANM R47,=10      CHECK FOR FIRST STATUS HOLDOFF
5269 077627 366 026                     JNZ OK..01       BIT SET, RE-TRY
5270 077631                     * DO NOT RETRY
5271 077631                     * CHECK FOR WRITE PROTECT
5272 077631 146 317 100                 ANM R46,=100,0     WRITE PROTECT BIT
5272 077634 000
5273 077635 367 007                     JZR BADDDD
5274 077637 316 052 175                 JSB =EX__IT
5275 077642 316 225 161                 JSB =MSERR+
5276 077645 974                         OCT 60D      "WRITE PROTECT"
5277 077646                     *                    USE SYSTEM ERROR
5278 077646                     *---------------------
5279 077646                     BADDDD  BSS 0
5280 077646 171 222                     CLB R71     CLEAR HEAD POSITION
5281 077650 316 052 175                 JSB =EX__IT
5282 077653 316 212 161                 JSB =MSERR
5283 077656 036                         OCT 30D      "DISC ERROR"
5284 077657                     *-------------------------
5285 077657                     OK..01  BSS 0
5286 077657                     * DECREMENT RETRY COUNT AND TEST
5287 077657 174 212                     DCB R74
5288 077661 367 026                     JZR RTCNTO
5289 077663                     * RETRY AREA
5290 077663                     RETRY   BSS 0
5291 077663 144 261 164                 LDMD R44,=PARAMS
5291 077666 207
5292 077667 030 243                     STM R44,R30
5293 077671 145 261 223                 LDMD R45,=PTRADR    BURST ADDR
5293 077674 207
5294 077675 263 314 377                 STMD R45,=PTR2
```

```
>>>>>          LLERR  LOW-LEVEL ERROR RECOVERY                          <<<<<<
 077700                      * INPUT PARAMETERS RESTORED, FOR CALING
 077700                      * R.A. AND REPLACE WITH HIGH LEVEL ENTRY
 077700 146 006 343                  POMD R46,-R6
 077703                      * H.L. R.A. WAS SAVED IN R20,21 BY HLINIT
 077703 120 345                      PUMD R20,+R6
 077705                      * AND RETURN TO TRY AGAIN
 077705 316 052 175                  JSB =EX__IT
 077710 236                          RTN
 077711                      *-------------------------
 077711                      * RETRY COUNT EXPIRED, CHECK HEAD STATE
 077711                      RTCNTO  BSS 0
 077711 174 250 010                  LDB R74,=10    RESET RETRY COUNT
 077714                      * TEST R71    (IF < 2 THEN MOVE HEAD AND RETRY)
 077714 171 310 002                  CMB R71,=2
 077717 367 325                      JZR BADDDD
 077721                      * MOVE HEAD AND RETRY
 077721 316 326 177                  JSB =HDMOVE
 077724 360 335                      JMP RETRY
 077726                      *----------------------------
```

```
>>>>>>>>        HEAD MOVE                                                  <<<<<<<<
077726                  HDMOVE  BSS 0
077726                  * MOVE HEAD  IN IF R71=0
077726                  *            OUT IF R71=1
077726                  *                     AND MOVE HEAD BACK
077726                  * THEN INCREMENT R71
077726                  *
077726                  * FIRST, GET DISC ADDRESS
077726 316 310 175              JSB =REQDA
077731                  * RETURNS WITH ADDRESS IN R44-47
077731                  *
077731 124 045 240              LDB R24,R45     LSB OF CYLINDER COUNT
077734 171 220                  TSB R71
077736 366 007                  JNZ MOVOUT        IF <>0 THEN MOVEOUT
077740 124 220                  TSB R24
077742 367 024                  JZR DON.E   CAN'T MOVE IN IF CYL 0,
077744                  *                          SO DON'T
077744                  * DECREMENT CYLINDER AND SEEK
077744 212                      DCB R#
077745 360 002                  JMP MOV-..
077747                  * INCREMENT CYLINDER AND SEEK
077747                  MOVOUT  BSS 0
077747 124 210                  ICB R24
077751 144 006 345      MOV-..  PUMD R44,+R6   SAVE ADDRESS
077754 124 045 242              STB R24,R45
077757 316 106 174              JSB =SEEK2
077762                  * RETURN TO DESIRED CYLINDER
077762 144 006 343              POMD R44,-R6   RESTORE ADDRESS
077765 316 106 174              JSB =SEEK2
077770                  ******************************
077770                  DON.E   BSS 0
077770                  * INCREMENT R71  (HEAD STATUS)
077770 171 210                  ICB R71
077772                  * AND RETURN
077772 236                      RTN
077773                  ******************************************
077773                  *                                        *
077773                  *    END OF MASS STORAGE ROM             *
077773                  *                                        *
077773                  ******************************************
077773                          LST CURRENT SPACE REMAINING
077773                  *       UNIL
077773
```

EXTERNALS

```
5358 077773          EXT  ASIGNL
5359 077773          EXT  BINMOV
5360 077773          EXT  BINAM
5361 077773          EXT  BLANKS
5362 077773          EXT  CATCO-
5363 077773          EXT  CATTAB
5364 077773          EXT  CHAIN+
5365 077773          EXT  CHKSTS
5366 077773          EXT  CLFBIT
5367 077773          EXT  CLOSE+
5368 077773          EXT  CLRERF
5369 077773          EXT  CLRSEC
5370 077773          EXT  CNTRTR
5371 077773          EXT  COMLO+
5372 077773          EXT  CONBIN
5373 077773          EXT  CUROPT
5374 077773          EXT  DISP.
5375 077773          EXT  DMNDCR
5376 077773          EXT  DRV12.
5377 077773          EXT  ENQVDN
5378 077773          EXT  ENQVUP
5379 077773          EXT  ERFLAG
5380 077773          EXT  ERROR
5381 077773          EXT  ERROR+
5382 077773          EXT  ERS9D
5383 077773          EXT  ER72D
5384 077773          EXT  ELSZ
5385 077773          EXT  FETSVX
5386 077773          EXT  FIL377
5387 077773          EXT  FNUM
5388 077773          EXT  FORM$A
5389 077773          EXT  FORMAR
5390 077773          EXT  FXDIR
5391 077773          EXT  GET#1
5392 077773          EXT  GET#1+
5393 077773          EXT  GETBU!
5394 077773          EXT  GETCN?
5395 077773          EXT  GETCNT
5396 077773          EXT  GETMEM
5397 077773          EXT  GETNAM
5398 077773          EXT  GET#N?
5399 077773          EXT  GRAFA.
5400 077773          EXT  GRAPH
5401 077773          EXT  GRAPH.
5402 077773          EXT  G10R2N
5403 077773          EXT  G$N+NN
5404 077773          EXT  INCHR
5405 077773          EXT  INIPGN
5406 077773          EXT  INTMUL
5407 077773          EXT  LDB.+
5408 077773          EXT  LDFNSH
5409 077773          EXT  LENCHK
5410 077773          EXT  LPOINT
```

```
                EXTERNALS
                        EXT   MOVUP
                        EXT   NPTC+5
                        EXT   NUMVAL
                        EXT   NUMVA+
                        EXT   ONEB
                        EXT   ONEBCM
                        EXT   ONER
                        EXT   PENDIN
                        EXT   PPGINT
PRADDR                  DAD   61454
                        EXT   PRAR$C
                        EXT   PRARRC
                        EXT   PUSH1A
                        EXT   RDAR$C
                        EXT   RDARRC
                        EXT   RDARRX
                        EXT   RDARX+
                        EXT   REFNUM
                        EXT   RELMEM
                        EXT   REPLTK
                        EXT   RCMJSB
                        EXT   RCMRTN
                        EXT   RSMEM-
                        EXT   RSUMSK
                        EXT   R#CONN
                        EXT   SAVACM
                        EXT   SBYTE
                        EXT   SCAN+
                        EXT   SCRA.-
                        EXT   SECNA+
                        EXT   SECUR+
                        EXT   SECUR?
                        EXT   SETF#
                        EXT   SETFER
                        EXT   SETTR1
                        EXT   STOST
                        EXT   STOSV
                        EXT   STREX+
                        EXT   STREXP
                        EXT   STRREF
                        EXT   ST240+
                        EXT   TIMWST
                        EXT   VLHED2
                        EXT   VOLHED
                        EXT   WARN
                        EXT   WREOR
INTRAD                  DAD   656
```

```
>>>>>>>>        ENTRIES                                                          <<<<<<<<
5[illegible] 077773
5[illegible] 077773                              ENT   ASSIG.
5462 077773                              ENT   AUTOST
5463 077773                              ENT   CMDCED
5464 077773                              ENT   CHTOUT
5465 077773                              ENT   DOCLOS
5466 077773                              ENT   GETDAT
5467 077773                              ENT   INITI.
5468 077773                              ENT   LOADB&
5469 077773                              ENT   MASSS.
5470 077773                              ENT   MSERR+
5471 077773                              ENT   MSCAT.
5472 077773                              ENT   MSCRE.
5473 077773                              ENT   MSIN
5474 077773                              ENT   MSLOD
5475 077773                              ENT   MSPRNT
5476 077773                              ENT   MSPUR.
5477 077773                              ENT   NAMEOK
5478 077773                              ENT   OBFCX
54[illegible]9 077773                              ENT   OUTLR
5[illegible]0 077773                              ENT   PRDRVR
5481 077773                              ENT   PRSTR+
5482 077773                              ENT   RDSTR.
5483 077773                              ENT   S$HDR
5484 077773                              ENT   SETCC-
5485 077773                              ENT   STORB%
5486 077773                              ENT   STORB&
5[illegible] 077773                              ENT   TAPDS+
5488 077773                              FIN
```

```
A.CHEK  000030      EQU   1
A.ASUS  000021  G   EQU   1
AC  SU  103560  G   LAD   12
AL  ATF 071473      LCL   1
ADVANC  066433      LCL   1
ALL$    067360      LCL   3
APEAD   067657      LCL   1
AR02    064045      LCL   1
AR03    064051      LCL   0   <--- NOT REFERENCED??
AR04    064114      LCL   1
AR05    064135      LCL   1
AR06    064140      LCL   1
ASIGNL  177777      EXT   1
ASNNAM  065561      LCL   1
ASSIG.  065465      ENT   2
ASSIGN  061237      LCL   1
AUTOST  063643      ENT   1
AWRITE  067134      LCL   1
B/REC   101645  G   LAD   4
BAD...  077624      LCL   4
BADDDD  077643      LCL   3
BAD  TT 075770      LCL   1
B/  EN  062760      LCL   1
BASE    060004      LCL   0   <--- NOT REFERENCED??
BINAM   177777      EXT   1
BINLEN  000004  G   EQU   1
BINMOV  177777      EXT   1
BLANK   073370      LCL   1
BLANKS  177777      EXT   4
BLOUN   061564      LCL   2
BPGMTY  000010  G   EQU   4
BRSTBN  074477      LCL   2
BSTBIN  074530      LCL   3
BSTIN   074454      LCL   1
BSTIN-  074434      LCL   2
BSTOUT  074506      LCL   1
BUFA.M  070752      LCL   2
BUFF55  077214      LCL   4
BUFFEX  070477      LCL   3
BYT1    072417      LCL   1
BYT2    072430      LCL   1
BY  S   072402      LCL   2
CAL.VB  070263      LCL   1
CAPRTY  000040  G   EQU   2
CAPSEC  064337      LCL   1
CAPUNS  064445      LCL   1
CAT     060542      LCL   1
CATCO-  177777      EXT   1
CATDON  061706      LCL   2
CATHED  062074      LCL   1
CATLOP  061512      LCL   1
CATTAB  177777      EXT   1
CHAIN   060570      LCL   1
CHAIN+  177777      EXT   1
```

```
CHCK+   060607    LCL    2
CHECK.  072437    LCL    1
CHECKR  060601    LCL    1
CHKF    060600    LCL    1
CHKOF.  072474    LCL    1
CHKSTS  177777    EXT    2
CIR.01  075197    LCL    1
CIR.02  075140    LCL    1
CIR.03  075203    LCL    1
CIR.04  075214    LCL    1
CKBITA  077340    LCL    1
CKCOMN  072443    LCL    1
CKD4T?  066071    LCL    1
CKDISK  063337    LCL    1
CKHPIB  077375    LCL    0   <--- NOT REFERENCED??
CKIBOK  077429    LCL    1
CKMSUS  070740    LCL    8
CKON    072467    LCL    1
CKOPEN  072450    LCL    1
CKSMOK  073671    LCL    1
CK_-RA  077264    LCL    2
CLEARE  066351    LCL    2
CLOSIT  177777    EXT    1
CLOSE+  177777    EXT    1
CLOSF   061310    LCL    1
CLRAS+  073610    LCL    1
CLRERF  177777    EXT    2
CLRFLG  064257    LCL    4
CLRSEC  177777    EXT    2
CMDED   074261    ENT    2
CMDHS   074247    LCL    2
CMDOUT  074072    LCL    5
CN--OK  075122    LCL    1
CNTERR  074107    LCL    5
CNTOUT  073410    ENT    1
CNTRTR  177777    EXT    1
CO++-+  075032    LCL    1
CO++MN  076704    LCL    1
COLON   000072  G EQU    2
COMLO+  177777    EXT    1
COMMA   000054  G EQU    4
COPY-   073711    LCL    1
COPAR   064450    LCL    1
CONBIN  177777    EXT    1
CONFOR  077347    LCL    2
COPY    061221    LCL    1
COPYTK  000006    EQU    1
CR      000015  G EQU    2
CR.CNT  101676  G DAD    7
CREATE  060555    LCL    1
CRTBAD  177701  G DAD    1
CRTDAT  177703  G DAD    1
CRTSTS  177702  G DAD    2
CSSSSG  074777    LCL    1
```

```
CUMREC  103554  G DAD     5
CURASN  103406  G DAD     1
CURCAT  103410  G DAD     3
CURLO+  065645    LCL     1
CURLOC  104053  G DAD     6
CUROPT  177777    EXT     1
CURREC  104147  G DAD     7
CURSTO  064347    LCL     1
CVDOFF  020000    DAD     2
D.B/RC  000036    EQU     3
D.BYTS  000034    EQU     4
D.FBEG  000016    EQU     2
D.FLAG  000037    EQU     1
D.FTYP  000012    EQU     3
D.RECS  000022    EQU     2
D.TIME  000024    EQU     1
D.VOL   000032    EQU     1
DADDD   103575  G DAD     1
DADDR   103552  G DAD     3
DATAB   103535  G DAD     2
DATAIN  074346    LCL     6
DATATY  000020  G EQU     5
D  COP  072125    LCL     1
DATDIF  066111    LCL     1
DATLEN  104056  G DAD    16
DATO    075427    LCL     1
DATO1   075424    LCL     3
DATOT   073320    LCL     1
DATOU+  074263    LCL     2
DATOUT  074277    LCL     3
DATUN   100174  G DAD     2
DATYPE  100173  G DAD     4
DCD+    061374    LCL     1
DCD-    061356    LCL     2
DCDFIL  061371    LCL     2
DCDZER  061407    LCL     4
DDONE   063232    LCL     1
DEC30   070671    LCL     1
DECODE  070765    LCL     3
DEFMSU  103477  G DAD     5
DE  D2  060677    LCL     1
DF  TD  103574  G DAD     4
DENTRY  103551  G DAD    17
DENTS   103577  G DAD     3
DFLAG   104224  G DAD     3
DFOUND  063233    LCL     2
DIG3    071201    LCL     1
DINFLG  102015  G DAD     2
DIRPTR  063506    LCL     6
DIRSCN  063162    LCL    11
DISCOP  064516    LCL     2
DISP.   177777    EXT     1
DIVLOP  076167    LCL     1
DLOOP   063166    LCL     1
```

SYMBOL	VALUE		TYPE	COUNT	
[illegible]MHDER	177777		EXT	0	<--- NOT REFERENCED??
[illegible]+	061627		LCL	1	
[illegible]FT	061622		LCL	2	
[illegible]UF	062730		LCL	1	
DOBUMP	067021		LCL	1	
DOCHK	066730		LCL	1	
DOCLOS	065530		ENT	2	
DON.E	077770		LCL	1	
DONCNT	075273		LCL	1	
DONDIV	076205		LCL	1	
DONHSI	075164		LCL	1	
DONIN	074377		LCL	1	
DONOTR	073724		LCL	0	<--- NOT REFERENCED??
DONOUT	074321		LCL	1	
DONPAR	075304		LCL	1	
DONSRT	075166		LCL	1	
DONTCM	075267		LCL	1	
DRIVE-	061707		LCL	2	
DRIVER	061713		LCL	1	
DRV12.	177777		EXT	1	
DSJ	075777		LCL	4	
D[illegible]ISU	103606	G	LAD	5	
D[illegible]ORG	103572	G	LAD	4	
DTAWPR	100201	G	LAD	2	
DUPOK2	072133		LCL	2	
D__OUT	074551		LCL	1	
E/DTYP	000024		EQU	2	
EASYAD	076241		LCL	1	
EI SZ	177777		EXT	1	
E[illegible]VDN	177777		EXT	1	
EMOVUP	177777		EXT	1	
END#	000157		EQU	1	
ENDDIR	063517		LCL	1	
ENTIR$	000337		EQU	3	
ENTSOK	062265		LCL	1	
EOR	000357		EQU	0	<--- NOT REFERENCED??
EOR#	067325		LCL	2	
EORERN	067730		LCL	1	
EORN	067514		LCL	1	
ER22D	060706		LCL	1	
ER[illegible]D	063354		LCL	1	
E[illegible]3D	067531		LCL	1	
ER63D	064771		LCL	2	
ER66D	066373		LCL	2	
ER67D	064644		LCL	6	
ER68D	063102		LCL	2	
ER69D	177777		EXT	1	
ER72D	177777		EXT	4	
ER89D	065302		LCL	11	
ER91D+	070533		LCL	2	
ER???2	070552		LCL	1	
ERFLAG	177777		EXT	7	
ERMESS	073440		LCL	1	
ERR!	070641		LCL	1	

```
ERR--R  074342    LCL    1
ERR20D  070734    LCL    4
ERR21D  071057    LCL    4
ERR22D  064401    LCL    6
ERR23D  060737    LCL    2
ERR63D  071776    LCL    1
ERR???  070531    LCL    1
ERROM#  100121  G GAD    1
ERROM.  065026    LCL    1
ERROR   177777    EXT    1
ERROR+  177777    EXT    1
ERRORS  100123  G GAD    2
ERRORX  070606    LCL    2
ERRRAM  103615  G GAD    2
ERRROM  100120  G GAD    2
ERRSC   101141  G GAD    2
ERRSC.  065036    LCL    1
ERRVER  075733    LCL    1
ERRXXX  077260    LCL    1
ESCFOR  077244    LCL    1
EXASIN  065527    LCL    1
EXFIL   110010  G GAD    1
EXGRF   110014    GAD    1
EX_-_N  065464    LCL    1
EX__IT  076452    LCL    8
FETSVX  177777    EXT    1
FIL377  177777    EXT    1
FILCPY  071771    LCL    1
FILERR  066261    LCL    1
FILFND  072034    LCL    1
FILNOT  072001    LCL    1
FILOK?  062746    LCL    2
FILSCN  072024    LCL    1
FILTYP  101671  G GAD   15
FMTLOP  075663    LCL    1
FNAM    103503  G GAD    8
FNAM+5  103510  G GAD    7
FNAMOK  071061    LCL    1
FNUM    177777    EXT    0   <--- NOT REFERENCED??
FORM$A  177777    EXT    1
FORMAT  075442    LCL    2
FORMAR  177777    EXT    2
FWGJRR  100006  G GAD    4
FWPRGM  100003  G GAD    1
FXDIR   177777    EXT    1
G$N+NN  177777    EXT    1
G10R2N  177777    EXT    2
GBADDR  072675    LCL    1
GBASE   010340  G EQU    3
GCOMN   072665    LCL    2
GET#    065166    LCL    6
GET#1   177777    EXT    1
GET#1+  177777    EXT    1
GET$N?  177777    EXT    1
```

SYMBOL	VALUE	TYPE	COUNT
GETBU!	177777	EXT	1
[illegible]SU+	977133	LCL	3
[illegible]UF	977120	LCL	14
[illegible]AP	061779	LCL	1
GETCM?	177777	EXT	2
GETCNT	177777	EXT	2
GETCO+	977127	LCL	1
GETCOM	970556	LCL	2
GETD+	063434	LCL	1
GETDAT	067577	ENT	4
GETDIR	063360	LCL	4
GETFIL	064164	LCL	2
GETINF	071562	LCL	2
GETLST	061756	LCL	2
GETMEM	177777	EXT	1
GETMSU	061356	LCL	2
GETNA!	970704	LCL	4
GETNAM	177777	EXT	1
GETNUM	970523	LCL	3
GETP+-	977202	LCL	1
GETS++	977101	LCL	1
G[illegible]SCO	977164	LCL	1
G [illegible]SE+	977163	LCL	1
GETSEC	977126	LCL	1
GETSTR	970537	LCL	9
GETTO	970547	LCL	2
GETTYP	061736	LCL	13
GET_BY	976743	LCL	1
GINTDS	177401	G %AD	4
[illegible]TEN	177400	G %AD	4
GLOAD	060570	LCL	1
GLOAD.	972510	LCL	1
GNAM	103515	G %AD	5
GNAM+5	103522	G %AD	5
GO++--	976374	LCL	1
GO++LP	975241	LCL	1
GO+.1	976279	LCL	1
GO+.2	976302	LCL	1
GO+.3	976317	LCL	1
GO+.5	976371	LCL	1
G[illegible].6	976410	LCL	1
G[illegible].7	976415	LCL	1
GOINDO	060557	LCL	4
GOOD	970751	LCL	1
GORND	066306	LCL	1
GORRTN	061036	LCL	2
GOTNA+	063074	LCL	1
GOTNAM	063063	LCL	1
GOTSEM	060712	LCL	1
GRAFA.	177777	EXT	1
GRAPH	177777	EXT	1
GRAPH.	177777	EXT	1
GRFIL	972534	LCL	2
GSIZE	104742	G %AD	2

GSTOR.	072711		LCL	1
GSTORE	060570		LCL	1
HCNT	076225		LCL	2
HLMOVE	077726		LCL	1
HD__00	076237		LCL	1
HEADCT	103622	G	DAD	3
HLFMT	075621		LCL	1
HLINI	076766		LCL	4
HLSEEK	077230		LCL	3
HLTIOP	075022		LCL	1
HOLE#1	063234		LCL	3
HOOK+	077024		LCL	2
IBFERR	074701		LCL	1
ICK	074232		LCL	1
ICKLOP	074235		LCL	1
IDRTN	074204		LCL	2
ILOP1	062412		LCL	1
ILOP2	062437		LCL	1
INBND#	067035		LCL	1
INBND1	067704		LCL	2
INBNDR	067632		LCL	2
INCHR	177777		EXT	1
INCRA	101732	G	DAD	3
IND6	060646		LCL	3
IND7	061030		LCL	1
INIDSK	062346		LCL	1
ININIT	070665		LCL	4
INIPGN	177777		EXT	1
INIT14	070677		LCL	5
INITI.	062134		ENT	2
INITIA	060500		LCL	1
INITIT	073631		LCL	1
INITZE	075557		LCL	1
INTMUL	177777		EXT	2
INTPR-	102072	G	DAD	1
INTPR6	102074	G	DAD	1
INTRAD	000656		DAD	1
INTRSC	177500		DAD	3
INVAL	062262		LCL	2
IOBITS	101140	G	DAD	5
IOCRST	073753		LCL	1
IOCOM#	000300		EQU	0
IRQ20	103742	G	DAD	1
IRQ20+	103751	G	DAD	1
IS+TOK	061277		LCL	1
IS.	060214		LCL	1
ISE+++	074717		LCL	2
ISENT	062255		LCL	1
ISINTL	062245		LCL	1
ISMSUS	062276		LCL	1
ISR	074576		LCL	1
ISRERR	074705		LCL	1
ISRHOK	074020		LCL	1
ISTOK	000034		EQU	2

)

```
ITFITS  066775      LCL   1
ITSA$   065197      LCL   1
IT  S   065622      LCL   1
I   T   067535      LCL   1
ITSDEF  071144      LCL   1
ITSMSU  071073      LCL   1
ITSOPN  066377      LCL   1
ITSVOL  071074      LCL   1
JOBF    073400      LCL   1
JSTGRF  072606      LCL   1
JSTREC  065246      LCL   1
KEYSTS  177402  G   SAD   1
L.D.S   076641      LCL   2
LADTAD  073254      LCL   0   <--- NOT REFERENCED??
LASTFI  063321      LCL   1
LAVAIL  100025  G   SAD   2
LCOUNT  072565      LCL   1
LDB.+   177777      EXT   1
LDCOMN  063660      LCL   1
LDFBEG  063532      LCL   8
LDFNSH  177777      EXT   1
LDFECS  063565      LCL   6
LF  A!  070145      LCL   2
LL.CHK  177777      EXT   3
LIKNAM  065345      LCL   1
LINEL>  073100      LCL   2
LINELN  101714  G   SAD   2
LLERR   077572      LCL   3
LLIN++  073312      LCL   1
L   II+ 073306      LCL   1
LL.NIT  075331      LCL   6
LLLNIT  075305      LCL   2
LLOK    077079      LCL   1
LLOK#2  075660      LCL   1
LLOK2   077114      LCL   1
LLOOP2  072624      LCL   2
LLPARI  075254      LCL   2
LNAME   064174      LCL   1
LO.+P   074161      LCL   1
LOAD    060579      LCL   1
LOAD+   063667      LCL   2
LO  B%  063711      ENT   1
L   DB+ 063706      LCL   1
LOADBI  060579      LCL   1
LOADLP  072611      LCL   1
LOADSC  073137      LCL   2
LOCOMX  063665      LCL   0   <--- NOT REFERENCED??
LOGRE+  062064      LCL   1
LOGREC  062042      LCL   2
LOP-03  073323      LCL   1
LOP-04  074302      LCL   1
LOP-10  074351      LCL   1
LOP-11  074417      LCL   1
LOP-21  074523      LCL   1
```

SYMBOL	VALUE	TYPE	COUNT	
LOP-23	074471	LCL	1	
LOP-26	074507	LCL	1	
LOP-32	075010	LCL	1	
LOP-40	076027	LCL	1	
LOP-41	076053	LCL	1	
LOP-70	075261	LCL	1	
LOP377	065446	LCL	1	
LOPSIT	077330	LCL	1	
LOPW+0	074127	LCL	1	
LPOINT	177777	EXT	2	
LRECOR	070507	LCL	3	
LROK	066535	LCL	1	
LROK+	066542	LCL	1	
LSTDAT	101650	G DAD	4	
LSTDIR	103556	G DAD	2	
LSTREC	064056	LCL	1	
MASSS.	064233	ENT	2	
MASSST	060570	LCL	1	
MEMOVF	063764	LCL	1	
MIDDL$	000177	EQU	1	
MOV-..	077751	LCL	1	
MOVOUT	077747	LCL	1	
MOVTST	067740	LCL	2	
MS.UP	177777	EXT	3	
MS2AD	071151	LCL	1	
MSBASE	103412	G DAD	2	
MSCAT.	061414	ENT	2	
MSCHA.	062454	LCL	1	
MSCPY.	071646	LCL	1	
MSCRE+	065202	LCL	0	<--- NOT REFERENCED??
MSCRE.	065176	ENT	2	
MSDUMP	073601	LCL	2	
MSERR	070612	LCL	10	
MSERR+	070625	ENT	13	
MSHIGH	103764	G DAD	2	
MSIN	070652	ENT	2	
MSINHK	070645	LCL	25	
MSLDB.	063574	LCL	1	
MSLOD	063600	ENT	1	
MSLOD.	063627	LCL	1	
MSLOW	103773	G DAD	2	
MSR.	061431	LCL	1	
MSRNT	066221	ENT	2	
MSPTR	103437	G DAD	1	
MSPUR.	064604	ENT	2	
MSREN.	064724	LCL	1	
MSSEC.	064263	LCL	1	
MSSTB.	062513	LCL	1	
MSSTO.	062574	LCL	1	
MSTIME	104002	G DAD	1	
MSUNS.	064405	LCL	1	
MSUSOK	071103	LCL	1	
MT?	063034	LCL	1	
MTFLAG	100200	G DAD	6	

```
*TSCNN  063245    LCL    4
N[illegible]TP  065300    LCL    1
[illegible]LE  072200    LCL    2
N[illegible]M#  000320    EQU    5
NAME+  064371    LCL    1
NAMEOK  071004    ENT    2
NAMESC  064357    LCL    1
NAMEUN  064364    LCL    1
NAMLEN  000012    EQU    0    <--- NOT REFERENCED??
NCR  073351    LCL    1
NCR++  073372    LCL    2
NEMPTY  063302    LCL    1
NEND$  065161    LCL    1
NENTIR  065127    LCL    1
NEWSRC  071365    LCL    1
NEXTCL  073622    LCL    1
NM24SC  062033    LCL    2
NMASC  061655    LCL    1
NMIDDL  065151    LCL    1
NOBUFR  062742    LCL    1
NO[illegible]PE  064775    LCL    1
N[illegible]+++  074745    LCL    1
NO[illegible]ERR  066264    LCL    1
NOFM..  075727    LCL    1
NOPAR  061401    LCL    1
NOPARM  064630    LCL    1
NOPE$  067464    LCL    1
NOPOVF  066442    LCL    1
NOR/W  066642    LCL    1
N[illegible]PTC  066173    LCL    1
NOT"*"  065543    LCL    2
NOTDAT  065462    LCL    1
NOTEXT  061613    LCL    1
NOTPAU  073746    LCL    1
NOTRST  077460    LCL    1
NOWRR1  075411    LCL    1
NPTC+5  177777    EXT    1
NPTCT  066206    LCL    1
NSTART  065140    LCL    1
NU[illegible]  061562    LCL    1
N[illegible]A+  177777    EXT    1
N[illegible]VAL  177777    EXT    1
NXTCH  071030    LCL    2
NXTDAT  101645  G  DAD  19
NXTDIG  071216    LCL    3
NXTDR+  072334    LCL    2
NXTDST  071532    LCL    1
NXTELP  070216    LCL    1
NXTELR  070135    LCL    1
NXTENT  063444    LCL    9
NXTFIL  072267    LCL    1
NXTFIN  063506    LCL    1
NXTMEM  100022  G  DAD  3
NXTONE  063251    LCL    1
```

SYMBOL VALUE TYPE COUNT Symbol Table ?/11/198? 3:03 PM PG164

SYMBOL	VALUE	TYPE	COUNT	
NXTP#	070431	LCL	1	
NXTPA3	063225	LCL	2	
NXT[illegible]#	070357	LCL	1	
NX[illegible]4S	070443	LCL	1	
NX.REC	064004	LCL	1	
NXTSRC	071456	LCL	1	
NXTTRN	071440	LCL	1	
OBFCK	074075	ENT	4	
OBFLOP	074100	LCL	1	
OCK	074215	LCL	3	
OCKLOP	074225	LCL	4	
OCTER#	103620	G SAD	1	
OK..01	077657	LCL	2	
OK.1A	075705	LCL	1	
OK.3A	077070	LCL	1	
OK.HHH	075763	LCL	1	
OKFILE	063774	LCL	1	
OKSOFR	065606	LCL	1	
ONEB	177777	EXT	7	
ONEBC/	066652	LCL	3	
ONEBCM	177777	EXT	1	
ONEFIL	071711	LCL	1	
ON[illegible]	177777	EXT	1	
OO[illegible]	076624	LCL	3	
OOPS	073136	LCL	2	
OO_-P*	076022	LCL	1	
OO__P*	076102	LCL	2	
OPLB++	074335	LCL	1	
OPSU#1	074536	LCL	5	
OPSU#2	074544	LCL	1	
OPSU#3	075504	LCL	6	
OPSU#4	075433	LCL	1	
OUT.CO	000260	EQU	2	
OUT.DO	000240	EQU	1	
OUT.EN	000220	EQU	2	
OUT.S4	000104	EQU	1	
OUT.S5	000105	EQU	1	
OUT.S6	000106	EQU	1	
OUTBUS	073217	LCL	1	
OUTCM1	075420	LCL	2	
OUTLR	066472	ENT	4	
OUT[illegible]+	066513	LCL	1	
OV[illegible]	067013	LCL	1	
OVEREX	064163	LCL	1	
P.SCOC	000026	G EQU	2	
P.SCOG	000033	G EQU	2	
P.SFLC	000025	G EQU	2	
P.SFLG	000032	G EQU	2	
P.TYPE	000006	G EQU	1	
P/BTYP	000050	EQU	0	<--- NOT REFERENCED??
PACK	060542	LCL	1	
PACK.	071234	LCL	1	
PACK=1	075112	LCL	2	
PACKDK	071250	LCL	1	

```
PACKDN  071353      LCL   2
PACOON  071526      LCL   2
PAMS    103564  G   SAD   3
PA .ES  060130      LCL   1
PDELET  071523      LCL   2
PENDIN  177777      EXT   2
PERIOD  000056  G   EQU   2
PFILES  071261      LCL   2
PGCOM+  077044      LCL   1
PGCONT  077104      LCL   2
PPOINT  177777      EXT   3
PPOLL   076024      LCL   4
PP_FAL  076077      LCL   1
PR#DSK  067232      LCL   1
PR#ITM  061121      LCL   1
PR#LST  061077      LCL   1
PRA$TK  000046      EQU   1
PRADDR  061454      SAD   1
PRAR$C  177777      EXT   1
PRARR$  070379      LCL   1
PRARR.  070167      LCL   1
PRRRC   177777      EXT   1
P  TOK  000040      EQU   1
PRDRVR  073023      ENT   2
PRDVF+  103550  G   SAD   1
PRDVFG  103547  G   SAD   6
PRECOR  070500      LCL   2
PREOL   000043      EQU   1
PREOL.  070464      LCL   1
P  IT   073166      LCL   0   <--- NOT REFERENCED??
PRGSIZ  063754      LCL   1
PRINT#  061042      LCL   1
PRNTOK  000042      EQU   1
PRNUM.  067226      LCL   1
PRSTOK  000044      EQU   1
PRSTR+  066671      ENT   1
PRSTR.  066662      LCL   2
PSHRON  061014      LCL   4
PT..1.  074414      LCL   1
PT.02-  076127      LCL   1
PT      177710  G   SAD   14
PT 1+   177712  G   SAD   1
PTR1-+  177713  G   SAD   1
PTR2    177714  G   SAD   50
PTR2+   177716  G   SAD   24
PTR2-   177715  G   SAD   15
PTR2-+  177717  G   SAD   3
PTRADR  103623  G   SAD   2
PUR+    064656      LCL   2
PURCOD  064650      LCL   1
PURDUN  064706      LCL   1
PURFIL  064707      LCL   2
PURG++  064717      LCL   1
PURGE   060631      LCL   1
```

SYMBOL	VALUE	TYPE	COUNT
[illegible]PGE+	064714	LCL	2
[illegible]URFPR	060642	LCL	1
[illegible]1H	177777	EXT	2
[illegible]IT	061033	LCL	2
PUSHRD	061014	LCL	7
PUT++	977177	LCL	1
PUTASC	062017	LCL	1
PUTBU+	977170	LCL	4
PUTBUF	977032	LCL	5
PUTCOM	977041	LCL	1
PUTINF	971606	LCL	2
PUTLST	062023	LCL	2
PUTMSU	977253	LCL	2
PUTNXT	062021	LCL	1
PUTSE+	977176	LCL	1
PUTSEC	977040	LCL	1
PUTTYP	061774	LCL	5
QMARK	000077	G EQU	1
QMARK.	073021	LCL	1
R#COM!	067731	LCL	2
R#COMN	177777	EXT	1
R#[illegible]	067310	LCL	1
R[illegible]LE	101673	G SAD	3
R[illegible]wPRM	066115	LCL	4
R460+	074640	LCL	1
R460K	074636	LCL	2
R550K	065317	LCL	1
R60K	060006	G SAD	1
RANDOM	100177	G SAD	8
RBUF36	063437	LCL	10
RD#$	067330	LCL	2
RD#DSK	067517	LCL	2
RD#ITM	060743	LCL	1
RD#LST	060720	LCL	1
RD+ADV	066422	LCL	4
RD=17	067263	LCL	1
RDA$TK	000047	EQU	1
RDAR$C	177777	EXT	1
RDARR$	970312	LCL	1
RDARR.	970106	LCL	1
RD[illegible]RC	177777	EXT	1
RD[illegible]RX	177777	EXT	1
RD[illegible]RX+	177777	EXT	1
RDATA	066032	LCL	2
RDATOK	000045	EQU	1
RDBUFF	076261	LCL	2
RDNTOK	000037	EQU	1
RDNUM.	067503	LCL	1
RDSTOK	000041	EQU	1
RDSTR.	067314	ENT	2
RE--EN	074672	LCL	2
READ#	060650	LCL	1
READ.	066221	LCL	1
REC256	061652	LCL	1

SYMBOL	VALUE	TYPE	COUNT
RECBUF	10266	G DAD	9
REDNXT	066503	LCL	2
RETN	061027	LCL	1
RETUM	177777	EXT	1
RELMEM	177777	EXT	2
RENAME	061221	LCL	1
RENFIL	065010	LCL	1
REPLTK	177777	EXT	1
REQ+++	076716	LCL	1
REQDA	076710	LCL	3
RESDAT	104060	G DAD	12
RESEND	104161	G DAD	8
RESSET	070067	LCL	2
REST32	066645	LCL	1
RESULT	065076	LCL	8
RETRY	077667	LCL	1
RHDPR#	066273	LCL	1
ROMFL	104065	G DAD	2
ROMJSB	177777	EXT	40
ROMRTN	177777	EXT	3
RSLOOP	073765	LCL	1
RSMEM-	177777	EXT	1
RSPTR	066462	LCL	1
RSR.A.	073737	LCL	1
RSUM8K	177777	EXT	1
RTCNTO	077711	LCL	1
RTNAD2	076526	LCL	1
RTNADD	076541	LCL	1
RTNSTK	100033	G DAD	1
RTOK	000036	EQU	1
RUNTIM	060012	LCL	1
S$HDR	070046	ENT	3
S2#3	074560	LCL	2
S40	076665	LCL	1
S40+	076671	LCL	2
S40350	076662	LCL	4
SAVACM	177777	EXT	1
SAVE26	103612	G DAD	2
SAVER6	104066	G DAD	5
SAV_PA	077006	LCL	1
SBYTE	177777	EXT	2
SCAN+	177777	EXT	1
SCOR	077313	LCL	4
SCRA.-	177777	EXT	1
SCRTYP	101731	G DAD	2
SCT+7	101730	G DAD	1
SCTEMP	101721	G DAD	2
SEC#	100174	G DAD	1
SEC1/2	000005	EQU	4
SECEXT	064356	LCL	1
SECNA+	177777	EXT	1
SECNAM	064526	LCL	1
SECQUI	062511	LCL	2
SECUR+	177777	EXT	1

SYMBOL VALUE TYPE COUNT Symbol Table 9/11/1981 3:03 PM PG16S

SECUR?	177777	EXT	1
SECURC	064465	LCL	2
SECURE	060610	LCL	1
SECURN	100206	G DAD	3
SEEK	076112	LCL	4
SEEK2	076106	LCL	3
SEMI	000047	G EQU	2
SEND++	073265	LCL	2
SENDB+	073230	LCL	1
SENDCR	073301	LCL	1
SERIAL	066217	LCL	1
SETCC-	075222	ENT	3
SETCCR	075217	LCL	2
SETF#	177777	EXT	2
SETFER	177777	EXT	2
SETFLG	064251	LCL	2
SETMSU	077565	LCL	3
SETSC+	075231	LCL	1
SETTR1	177777	EXT	1
SHFT46	074630	LCL	1
SHFTDN	076062	LCL	1
SKPTD	064512	LCL	1
SKPOL+	072317	LCL	1
SKHOLE	072312	LCL	1
SLOOP	072765	LCL	2
SNDEOL	073124	LCL	1
SNDMLA	076701	LCL	4
SNDMTA	076674	LCL	2
SPAR#	103614	G DAD	3
SPECIF	103527	G DAD	4
SRCMSU	103602	G DAD	6
SRCORG	103570	G DAD	5
SREAD+	064555	LCL	1
SREADC	064577	LCL	1
SREADR	064546	LCL	2
SRTIOP	075030	LCL	1
SRTLOP	075041	LCL	1
SRWIND	075160	LCL	3
ST240+	177777	EXT	1
STAR	000052	G EQU	2
STATS$	000317	EQU	1
STATUS	076647	LCL	1
STFBEG	063521	LCL	2
STFLAG	064314	LCL	1
STORB%	062716	ENT	1
STORB&	062673	ENT	1
STORB-	062713	LCL	1
STORE	060570	LCL	1
STOREB	060570	LCL	1
STOST	177777	EXT	1
STOSV	177777	EXT	1
STOW	063127	LCL	1
STPTR	077151	LCL	1
STRARR	061173	LCL	2

```
STPECS  063540    LCL    3
[illegible]  177777    EXT    3
[illegible]  177777    EXT    1
[illegible]  064253    LCL    1
STRREF  177777    EXT    1
STSIZE  101741  G DAD    1
STTCED  074410    LCL    1
SVCWRD  100162  G DAD    1
SWAP+   063547    LCL    9
SWAP1   063550    LCL    0   <--- NOT REFERENCED??
SWITCH  063746    LCL    1
SYS+I   070573    LCL    1
SYSER+  070579    LCL    3
SYSERP  070562    LCL    1
T.ASCI  000004    EQU    3
T.GRAF  000014    EQU    1
TAPOS   070720    LCL    2
TAPOS+  070726    ENT    5
TAPOS-  070713    LCL    7
TIMW-+  075236    LCL    2
TIMWST  177777    EXT    1
TO[illegible]OK  061274    LCL    4
TO.     060214    LCL    2
TOID    076435    LCL    3
TOIDER  076622    LCL    1
TOKENS  060215    LCL    1
TOS     101744  G DAD    6
TOTAP?  066366    LCL    2
TOTOK   000244    EQU    1
[illegible]D  076501    LCL    2
TRAC+   076214    LCL    1
TRUCAL  070243    LCL    4
TRY340  065116    LCL    1
TYP.    065046    LCL    1
TYPOK   063106    LCL    2
TYPOK2  064214    LCL    1
TYPOPN  065062    LCL    1
UNFOR   077462    LCL    3
UNSECU  060610    LCL    1
UNTA.!  076754    LCL    1
UN[illegible]LI  074566    LCL   10
USEHOL  067047    LCL    2
USTART  067011    LCL    1
V.1000  000014    EQU    0   <--- NOT REFERENCED??
V.DBEG  000012    EQU    2
V.DLEN  000022    EQU    3
V.ID    000000    EQU    0   <--- NOT REFERENCED??
V.LABL  000002    EQU    5
VERIFT  075511    LCL    1
VFYL.P  075700    LCL    1
VLDONE  071779    LCL    1
VLHED2  177777    EXT    1
VOK     061346    LCL    1
VOL2AD  077277    LCL    2
```

SYMBOL	VALUE	TYPE	COUNT
VOLHED	177777	EXT	1
[illegible]ON	062306	LCL	2
[illegible]M	072340	LCL	1
VOLUME	061330	LCL	1
WA--LP	075103	LCL	1
WAI++T	075544	LCL	3
WAI++_	075550	LCL	1
WAIT20	074011	LCL	1
WARN	177777	EXT	1
WDATA	066041	LCL	1
WLOP-+	073773	LCL	1
WRBUFP	076265	LCL	1
WREOR	177777	EXT	1
WREOR!	067213	LCL	2
WRTDA-	066143	LCL	2
WRTDAT	066143	LCL	2
WRTDIR	071623	LCL	7
WRTRTN	066040	LCL	1
WR_EOL	076466	LCL	1
WSTRN1	067165	LCL	2
WSTRNG	067111	LCL	2
WT[illegible]DY	066400	LCL	4
XNOPPX	076344	LCL	1
YESPOW	073851	LCL	1
YNOPPY	076434	LCL	1
ZERO	000060	EQU	1
rtn001	061172	LCL	2
rtn002	067737	LCL	3
rtn003	072472	LCL	1
rtn004	072707	LCL	1