RA80-** UDA DIAG DATA FILE AH-S834A-MC OCT 1981
CZUDDAO FICHE 1 OF 3 COPYRIGHT© 1981
MADE IN USA

RA80·** UDA DIAG DATA FILE AH-S834A-MC OCT 1981
CZUDDA0 FICHE 2 OF 3 COPYRIGHT© 1981
MADE IN USA

RA80-** UDA DIAG DATA FILE AH-S834A-MC
CZUDDAO FICHE 3 OF 3

1

.REM &

IDENTIFICATION

PRODUCT CODE: AC-S833A-MC

PRODUCT NAME: CZUDDAO UDA DIAGNOSTIC DATA FILE

PRODUCT DATE: 10-JULY-81

MAINTAINER: DIAGNOSTIC ENGINEERING

AUTHOR: DALE KECK

TABLE OF CONTENTS

1.0 GENERAL INFORMATION

1.1 PROGRAM ABSTRACT

THE UDA HOST RESIDENT DIAGNOSTIC CAUSES THE EXECUTION OF THE UDA AND DRIVE RESIDENT DIAGNOSTIC INDIVIDUAL PROGRAMS AND REPORTS THE RESULTS ON THE PDP-11 CONSOLE TERMINAL OR LINE PRINTER. EXERCISES ALSO PERFORMS TO VERIFY THAT:

1. THE UDA CAN PROPERLY COMMUNICATE WITH THE PDP-11 PROCESSOR AND CAN TRANSFER BLOCKS OF DATA TO AND FROM UNIBUS MEMORY.

2. THE UDA AND THE DISK DRIVES CAN COMMUNICATE AND TRANSFER DATA PROPERLY.

3. THE DISK DRIVES CAN FUNCTION PROPERLY AS DEFINED BY THE SDI.

4. THE DISK DRIVES CAN SEEK, READ AND WRITE TO ALL BLOCKS ON THE DISK AND ALLOW THE MEASUREMENT OF THE DRIVES' PERFORMANCE IN A RATIO OF ERRORS PER MILLION BITS TRANSFERRED.

THE UDA HOST RESIDENT DIAGNOSTIC CONSISTS OF ONE PDP-11 DIAGNOSTIC SUPERVISOR PROGRAM THAT RUNS IN THE PDP-11 PROCESSOR AND FOUR PROGRAMS THAT RUNS IN THE UDA'S BUFFER MEMORY THROUGH AN INTERPRETER CALLED THE "DIAGNOSTIC MACHINE" WHICH RESIDES IN THE UDA. THE PDP-11 PROGRAM MAINLY IS RESPONSIBLE FOR DOWNLINE LOADING THE "DIAGNOSTIC MACHINE" PROGRAMS INTO THE UDA AND STARTING THEIR EXECUTION. THE "DIAGNOSTIC MACHINE" PROGRAM CONTROLS THE TESTING FROM THAT POINT BY REQUESTING THE PDP-11 PROCESSOR TO SUPPLY INFORMATION, PRINT ERROR MESSAGES AND UPDATE STATISTICS. THE "DIAGNOSTIC MACHINE" PROGRAM INFORMS THE PDP-11 PROCESSOR WHEN A TEST IS COMPLETE.

THE NUMBER OF UDAS AND DISK DRIVES THAT CAN BE TESTED AT ONE TIME BY THIS DIAGNOSTIC PROGRAM IS LIMITED BY THE PDP-11 MEMORY SIZE. WITH THE MINIMUM MEMORY SIZE OF 28K WORDS, AT LEAST TWO UDAS WITH FOUR DISK DRIVES EACH ARE SELECTABLE. THE NUMBER OF UDAS AND DISK DRIVES SELECTABLE BY THE DIAGNOSTIC INCREASES WITH ADDITIONAL MEMORY, BUT THERE IS NO GOAL IMPLIED FOR THE MAXIMUM POSSIBLE CONFIGURATION.

THIS DIAGNOSTIC HAS BEEN WRITTEN FOR USE WITH THE DIAGNOSTIC RUNTIME SERVICES SOFTWARE (SUPERVISOR). THESE SERVICES PROVIDE THE INTERFACE TO THE OPERATOR AND TO THE SOFTWARE ENVIRONMENT. THIS PROGRAM CAN BE USED WITH XXDP+, ACT, APT, SLIDE AND PAPER TAPE. FOR A COMPLETE DESCRIPTION OF THE RUNTIME SERVICES, REFER TO THE XXDP+ USER'S MANUAL. THERE IS A BRIEF DESCRIPTION OF THE RUNTIME SERVICES IN SECTION 2 OF THIS DOCUMENT.

1.2 SYSTEM REQUIREMENTS

THIS PROGRAM WILL BE DESIGNED USING THE PDP-11 SUPERVISOR REVISION C. RUN TIME ENVIRONMENTS ARE DETERMINED BY THE SUPERVISOR AND MAY CHANGE AS NEW VERSIONS OF THE SUPERVISOR ARE DEVELOPED. THE INITIAL VERSION WILL REQUIRE THE FOLLOWING:

PDP-11 PROCESSOR
28K WORDS OF MEMORY (MINIMUM)
XXDP+ LOAD MEDIA
ONE OR MORE UDA SUB-SYSTEMS
LINE CLOCK - EITHER TYPE L OR P

THE LINE CLOCK WILL BE USED FOR ALL TIMED LOOPS IN THE PROGRAM. THE DIAGNOSTIC WILL RUN ON A SYSTEM WITH NO CLOCK BUT WILL HANG WHENEVER AN EVENT FOR WHICH THE PROGRAM IS WAITING DOES NOT HAPPEN (I.E., A TIME-OUT ERROR MESSAGE WILL NOT RESULT).

2.0 OPERATING INSTRUCTIONS

THIS SECTION CONTAINS A BRIEF DESCRIPTION OF THE RUNTIME SERVICES. FOR DETAILED INFORMATION, REFER TO THE XXDP+ USER'S MANUAL (CHQUS).

2.1 COMMANDS

THERE ARE ELEVEN LEGAL COMMANDS FOR THE DIAGNOSTIC RUNTIME SERVICES (SUPERVISOR). THIS SECTION LISTS THE COMMANDS AND GIVES A VERY BRIEF DESCRIPTION OF THEM. THE XXDP+ USER'S MANUAL HAS MORE DETAILS.

COMMAND	EFFECT
START	START THE DIAGNOSTIC FROM AN INITIAL STATE
RESTART	START THE DIAGNOSTIC WITHOUT INITIALIZING
CONTINUE	CONTINUE AT TEST THAT WAS INTERRUPTED (AFTER ^C)
PROCEED	CONTINUE FROM AN ERROR HALT
EXIT	RETURN TO XXDP+ MONITOR (XXDP+ OPERATION ONLY.)
ADD	ACTIVATE A UNIT FOR TESTING (ALL UNITS ARE CONSIDERED TO BE ACTIVE AT START TIME
DROP	DEACTIVATE A UNIT
PRINT	PRINT STATISTICAL INFORMATION (IF IMPLEMENTED BY THE DIAGNOSTIC - SECTION 4.0)
DISPLAY	TYPE A LIST OF ALL DEVICE INFORMATION
FLAGS	TYPE THE STATE OF ALL FLAGS (SEE SECTION 2.3)
ZFLAGS	CLEAR ALL FLAGS (SEE SECTION 2.3)

A COMMAND CAN BE RECOGNIZED BY THE FIRST THREE CHARACTERS. SO YOU MAY, FOR EXAMPLE, TYPE "STA" INSTEAD OF "START".

2.2 SWITCHES

THERE ARE SEVERAL SWITCHES WHICH ARE USED TO MODIFY SUPERVISOR OPERATION. THESE SWITCHES ARE APPENDED TO THE LEGAL COMMANDS. ALL OF THE LEGAL SWITCHES ARE TABULATED BELOW WITH A BRIEF DESCRIPTION OF EACH. IN THE DESCRIPTIONS BELOW, A DECIMAL NUMBER IS DESIGNATED BY "DDDDD".

SWITCH	EFFECT
/TESTS:LIST	EXECUTE ONLY THOSE TESTS SPECIFIED IN THE LIST. LIST IS A STRING OF TEST NUMBERS, FOR EXAMPLE - /TESTS:1:5:7-10. THIS LIST WILL CAUSE TESTS 1,5,7,8,9,10 TO BE RUN. ALL OTHER TESTS WILL NOT BE RUN.

/PASS:DDDDD	EXECUTE DDDDD PASSES (DDDDD = 1 TO 64000)
/FLAGS:FLGS	SET SPECIFIED FLAGS. FLAGS ARE DESCRIBED IN SECTION 2.3.
/EOP:DDDDD	REPORT END OF PASS MESSAGE AFTER EVERY DDDDD PASSES ONLY. (DDDDD = 1 TO 64000)
/UNITS:LIST	TEST/ADD/DROP ONLY THOSE UNITS SPECIFIED IN THE LIST. LIST EXAMPLE - /UNITS:0:5:10-12 USE UNITS 0,5,10,11,12 (UNIT NUMBERS = 0-63)

EXAMPLE OF SWITCH USAGE:

START/TESTS:1-5/PASS:1000/EOP:100

THE EFFECT OF THIS COMMAND WILL BE: 1) TESTS 1 THROUGH 5 WILL BE EXECUTED, 2) ALL UNITS WILL TESTED 1000 TIMES AND 3) THE END OF PASS MESSAGES WILL BE PRINTED AFTER EACH 100 PASSES ONLY. A SWITCH CAN BE RECOGNIZED BY THE FIRST THREE CHARACTERS. YOU MAY, FOR EXAMPLE, TYPE ''/TES:1-5'' INSTEAD OF ''/TESTS:1-5''.

BELOW IS A TABLE THAT SPECIFIES WHICH SWITCHES CAN BE USED BY EACH COMMAND.

	TESTS	PASS	FLAGS	EOP	UNITS
START	X	X	X	X	X
RESTART	X	X	X	X	X
CONTINUE		X	X	X	
PROCEED			X		
DROP					X
ADD					X
PRINT					
DISPLAY					X
FLAGS					
ZFLAGS					
EXIT					

2.3 FLAGS

FLAGS ARE USED TO SET UP CERTAIN OPERATIONAL PARAMETERS SUCH AS LOOPING ON ERROR. ALL FLAGS ARE CLEARED AT STARTUP AND REMAIN CLEARED UNTIL EXPLICITLY SET USING THE FLAGS SWITCH. FLAGS ARE ALSO CLEARED AFTER A START COMMAND UNLESS SET USING THE FLAG SWITCH. THE ZFLAGS COMMAND MAY ALSO BE USED TO CLEAR ALL FLAGS. WITH THE EXCEPTION OF THE START AND ZFLAGS COMMANDS, NO COMMANDS AFFECT THE STATE OF THE FLAGS; THEY REMAIN SET OR CLEARED AS SPECIFIED BY THE LAST FLAG SWITCH.

FLAG	EFFECT
HOE	HALT ON ERROR - CONTROL IS RETURNED TO RUNTIME SERVICES COMMAND MODE
LOE	LOOP ON ERROR
IER*	INHIBIT ALL ERROR REPORTS
IBR*	INHIBIT ALL ERROR REPORTS EXCEPT FIRST LEVEL (FIRST LEVEL CONTAINS ERROR TYPE, NUMBER, PC, TEST AND UNIT)

IXR*	INHIBIT EXTENDED ERROR REPORTS (THOSE CALLED BY PRINTX MACRO'S)
PRI	DIRECT MESSAGES TO LINE PRINTER
PNT	PRINT TEST NUMBER AS TEST EXECUTES
BOE	"BELL" ON ERROR
UAM	UNATTENDED MODE (NO MANUAL INTERVENTION)
ISR	INHIBIT STATISTICAL REPORTS (DOES NOT APPLY TO DIAGNOSTICS WHICH DO NOT SUPPORT STATISTICAL REPORTING)
IDR	INHIBIT PROGRAM DROPPING OF UNITS
ADR	EXECUTE AUTODROP CODE
LOT	LOOP ON TEST
EVL	EXECUTE EVALUATION (ON DIAGNOSTICS WHICH HAVE EVALUATION SUPPORT)

*ERROR MESSAGES ARE DESCRIBED IN SECTION 3.1

SEE THE XXDP+ USER'S MANUAL FOR MORE DETAILS ON FLAGS. YOU MAY SPECIFY MORE THAN ONE FLAG WITH THE FLAG SWITCH. FOR EXAMPLE, TO CAUSE THE PROGRAM TO LOOP ON ERROR, INHIBIT ERROR REPORTS AND TYPE A "BELL" ON ERROR, YOU MAY USE THE FOLLOWING STRING:

```
/FLAGS:LOE:IER:BOE
```

2.4 HARDWARE QUESTIONS

WHEN A DIAGNOSTIC IS STARTED, THE RUNTIME SERVICES WILL PROMPT THE USER FOR HARDWARE INFORMATION BY TYPING "CHANGE HW (L) ?" YOU MUST ANSWER "Y" AFTER A START COMMAND UNLESS THE HARDWARE INFORMATION HAS BEEN "PRELOADED" USING THE SETUP UTILITY (SEE CHAPTER 6 OF THE XXDP+ USER'S MANUAL). WHEN YOU ANSWER THIS QUESTION WITH A "Y", THE RUNTIME SERVICES WILL ASK FOR THE NUMBER OF UNITS (IN DECIMAL). YOU WILL THEN BE ASKED THE FOLLOWING QUESTIONS FOR EACH UNIT.

THE PROGRAM WILL ASK THE FOLLOWING QUESTIONS IN RESPONSE TO A START COMMAND (NON-SCRIPT):

CHANGE HW?

ANSWER NO TO USE THE PRE-BUILT ANSWERS FOR ALL HARDWARE QUESTIONS. THIS PROGRAM WILL BE RELEASED PRE-BUILT TO TEST ONE UNIT WITH THE DEFAULT ANSWERS SHOWN BELOW. THE PRE-BUILT ANSWERS MAY BE CHANGED AT ANY TIME WITH THE SETUP UTILITY. ANSWER 'YES' TO BE ASKED ALL THE HARDWARE QUESTIONS.

UNITS (D) ?

ANSWER WITH THE NUMBER OF UNITS TO BE TESTED (NO DEFAULT). THIS ANSWER WILL DETERMINE HOW MANY TIMES THE FOLLOWING QUESTIONS ARE ASKED. A UNIT IS A LOGICAL DISK DRIVE ON A UDA. 1 TO 64 UNITS MAY BE SPECIFIED.

UNIBUS ADDRESS OF UDA (O) 177510 ?

ANSWER WITH THE ADDRESS OF THE UDAIP REGISTER OF ONE UDA AS ADDRESSED BY THE PROCESSOR WITH MEMORY MANAGEMENT TURNED OFF (I.E., AN EVEN 16-BIT ADDRESS IN THE RANGE OF 160000 TO 177774).

VECTOR (O) 154 ?

ANSWER WITH THE INTERRUPT VECTOR ADDRESS OF THE UDA. A VECTOR ADDRESS IN THE RANGE OF 4 TO 774 MAY BE SPECIFIED.

BR LEVEL (D) 5 ?

ANSWER WITH THE INTERRUPT PRIORITY USED BY THE UDA. LEVELS 4 TO 7 ARE ACCEPTED.

UNIBUS BURST RATE (D) 0 ?

THE UDA ALLOWS THE ABILITY TO CONTROL THE MAXIMUM NUMBER OF WORDS TRANSFERRED ACROSS THE UNIBUS EACH TIME THE UDA BECOMES MASTER. ANSWER WITH THE VALUE YOUR OPERATING SYSTEM USES OR USE ZERO WHICH WILL TELL THE UDA TO SUPPLY A VALUE. A DECIMAL NUMBER IN THE RANGE OF 0 TO 63 MAY BE SPECIFIED. THE VALUE WILL BE PASSED DIRECTLY TO THE UDA DURING INITIALIZATION.

DRIVE NUMBER (D) 0 ?

ANSWER WITH THE LOGICAL DRIVE NUMBER ON THE FRONT OF THE DRIVE YOU WISH TO TEST. ON A MULTI-UNIT DRIVE, EACH SUB-UNIT NUMBER ON THE DRIVE MUST BE TESTED AS A SEPARATE UNIT TO COMPLETELY TEST THE DRIVE. A MAXIMUM OF EIGHT LOGICAL DRIVES MAY BE TESTED ON ONE UDA AT A TIME (UDA CONFIGURATION LIMIT).

EXERCISE ON CUSTOMER DATA AREA IN TEST 4 (L) N ?

ANSWER NO TO HAVE TEST 4 (DRIVE EXERCISER) RUN ON THE DIAGNOSTIC AREA OF THE DISK. ANSWER YES TO RUN ON THE CUSTOMER DATA AREA. A YES ANSWER WILL DESTROY ANY CUSTOMER DATA THAT MAY BE ON THE DISK. A WARNING MESSAGE WILL BE PRINTED IF THIS QUESTION IS ANSWERED YES (SEE SECTION 4.3).

2.5 SOFTWARE QUESTIONS

AFTER YOU HAVE ANSWERED THE HARDWARE QUESTIONS OR AFTER A RESTART OR CONTINUE COMMAND, THE RUNTIME SERVICES WILL ASK FOR SOFTWARE PARAMETERS. THESE PARAMETERS WILL GOVERN SOME DIAGNOSTIC SPECIFIC OPERATION MODES. YOU WILL BE PROMPTED BY "CHANGE SW (L) ?" IF YOU WISH TO CHANGE ANY PARAMETERS, ANSWER BY TYPING "Y". THE SOFTWARE QUESTIONS AND THE DEFAULT VALUES ARE DESCRIBED IN THE NEXT PARAGRAPH(S).

THE PROGRAM WILL ASK THE FOLLOWING QUESTIONS IN RESPONSE TO THE START,

RESTART, AND CONTINUE COMMANDS.

CHANGE SW ?

ANSWER NO TO BYPASS THE FOLLOWING QUESTIONS IN THIS SECTION. A YES ANSWER WILL CAUSE THE QUESTIONS TO BE ASKED AND ALLOW THE DEFAULT ANSWERS TO BE CHANGED.

ENTER MANUAL INTERVENTION MODE FOR SPECIAL DIAGNOSIS (L) N ?

TESTS 2 AND 4 HAVE MANUAL INTERVENTION MODES WHICH ALLOW ADDITIONAL PARAMETERS TO BE INPUT TO ALTER THE NORMAL TESTING OF A DISK DRIVE. THIS QUESTION SHOULD NORMALLY BE ANSWERED NO WHEN THIS DIAGNOSTIC IS FIRST RUN. THEN, DEPENDING ON THE ERRORS DETECTED, IT MAY BE DESIRABLE TO CHANGE THIS ANSWER TO YES AND ALTER THE TESTING TO FURTHER ISOLATE THE PROBLEM. IF THIS QUESTION IS ANSWERED YES, AND THE UAM (UNATTENDED MODE OPERATION) FLAG IS SET, TESTS 2 AND 4 WILL PRINT A WARNING MESSAGE THAT THE MODE CANNOT BE ENTERED AND WILL PROCEED AS IF ANSWERED NO. SEE THE DESCRIPTION OF THE INDIVIDUAL TESTS FOR MORE INFORMATION.

REMAINING SOFTWARE QUESTIONS APPLY TO TEST 4 ONLY

THIS INFORMATIONAL MESSAGE IS PRINTED TO DESCRIBE THE USE OF THE REMAINING QUESTIONS. IF TEST 4 IS NOT BEING RUN, A ''CONTROL Z'' CAN BE TYPED TO BYPASS THEM.

ERROR LIMIT (D) 32 ?

ENTER THE NUMBER OF HARD ERRORS ALLOWED BEFORE A DRIVE IS DROPPED FROM EXERCISE BY TEST #4. A NUMBER IN THE RANGE OF 1 TO 65535 WILL BE ACCEPTED.

READ TRANSFER LIMIT IN MEGABYTES - 0 FOR NO LIMIT (D) 0 ?

WHEN THE SPECIFIED NUMBER OF BYTES HAVE BEEN READ FROM A DRIVE BY TEST #4, THE DRIVE WILL BE DROPPED FROM TESTING. WHEN ALL DRIVES ARE DROPPED, AN END OF PASS WILL BE INDICATED AND THE SELECTED TESTS WILL BE RUN AGAIN. THIS IS THE METHOD USED TO DETERMINE HOW LONG TEST #4 IS TO RUN. ANSWER WITH A ZERO TO PREVENT TEST FROM ENDING. THE ONLY OTHER WAY TEST #4 CAN END IS TO HAVE ALL DRIVES DROPPED BECAUSE THE ERROR LIMIT ON EACH IS EXCEEDED. OF COURSE, THE OPERATOR CAN ALWAYS STOP TEST #4 BY TYPING A CONTROL-C.

SUPPRESS PRINTING SOFT ERRORS (L) Y?

WHEN TEST #4 NEEDS TO PERFORM RETRIES, SOFT ERROR REPORTS WILL BE PRINTED TO GIVE AS MUCH INFORMATION AS POSSIBLE. THESE ACTIONS ARE CONSIDERED NORMAL OPERATION AND ARE NOT ERROR CONDITIONS UNTIL THE RETRIES FAIL. WHEN THE TEST IS BEING RUN ONLY TO SEE HOW RELIABLE THE DRIVE PERFORMS, THIS QUESTION SHOULD BE ANSWERED YES SO THEY ARE NOT CONFUSED WITH HARD ERRORS. THE NUMBER OF THESE SOFT ERRORS IS ALWAYS REPORTED IN THE STATISTICAL REPORT. ANSWER NO TO SEE ALL THE SOFT ERROR REPORTS.

DO INITIAL WRITE ON START (L) Y ?

IF TEST #4 IS TO DO DATA COMPARES, THE DRIVE WILL NEED TO BE WRITTEN WITH DATA PATTERNS READABLE BY THE PROGRAM.

IF THE DIAGNOSTIC AREA IS SELECTED FOR TESTING, THE INITIAL WRITE IS ALWAYS PERFORMED (REGARDLESS OF HOW THIS QUESTION IS ANSWERED).

IF THE CUSTOMER DATA AREA IS SELECTED FOR TESTING, THE INITIAL WRITE WILL BE PERFORMED WHEN ALL OF THE FOLLOWING ARE TRUE:

1. THIS QUESTION IS ANSWERED YES.

2. THIS IS THE FIRST TIME TEST #4 IS BEING RUN AFTER A START COMMAND.

3. THE DISK IS WRITE ENABLED.

ANSWERING THIS QUESTION NO WHEN TESTING ON THE CUSTOMER DATA AREA WILL NORMALLY RESULT IN DATA COMPARISON ERRORS IF THE DISK WAS NOT PREVIOUSLY WRITTEN BY THIS DIAGNOSTIC OR THE FORMATTER.

NOTE THAT WRITE CHECKS ARE NOT PERFORMED DURING THE INITIAL WRITE.

ENABLE ERROR LOG (L) N ?

A YES ANSWER WILL CAUSE ERROR MESSAGES IN TEST #4 TO BE STORED IN A LOG BUFFER. ONCE THE LOG BUFFER IS FULL, ADDITIONAL ERROR INFORMATION IS LOST. THE CONTENTS OF THE LOG BUFFER WILL BE PRINTED WHEN TEST #4 IS STOPPED AND A STATISTICAL REPORT REQUESTED. THIS LOG FEATURE IS INTENDED TO ALLOW THE DIGITAL DIAGNOSIS CENTER (DDC) TO START TEST #4 THEN HANG UP FROM THE SYSTEM AND LET IT RUN FOR SOME PERIOD OF TIME. DDC CAN CALL THE SYSTEM BACK LATER, TYPE CONTROL-C, THEN PRINT AND SEE THE ERRORS THAT HAVE OCCURRED (UP TO THE LIMIT OF THE LOG BUFFER). A MESSAGE WILL BE PRINTED TO INDICATE NO ERRORS HAVE OCCURRED IF THE LOG BUFFER IS EMPTY. TEST #4 WILL NOT BE ALLOWED TO END WHILE THE ERROR LOG IS ENABLED EVEN THOUGH TESTING MAY ACTUALLY STOP. THE LOG BUFFER WILL HOLD A MINIMUM OF 30 ERROR MESSAGES.

2.6 EXTENDED P-TABLE DIALOGUE

THE FOLLOWING WARNING WILL BE PRINTED IN RESPONSE TO A START COMMAND IF ANY DRIVE IS SELECTED FOR TESTING ON THE CUSTOMER DATA AREA. THE WARNING WILL APPEAR IMMEDIATELY AFTER THE SOFTWARE QUESTIONING IS COMPLETED.

CUSTOMER DATA WILL BE DESTROYED ON:

UNIT	UDA AT	DRIVE
XX	XXXXXX	XXX

UNLESS THE DIAGNOSTIC IS BEING RUN IN UNATTENDED MODE (E.G., START/FLAG:UAM TO SUPERVISOR PROMPT), A CONFIRMATION WILL ALSO BE REQUIRED AS FOLLOWS:

ARE YOU SURE CUSTOMER DATA CAN BE DESTROYED (L) ?

IF THE ABOVE QUESTION IS ANSWERED NO, THE ENTIRE DIAGNOSTIC WILL STOP AND THE SUPERVISOR PROMPT WILL BE DISPLAYED. NO DEFAULT ANSWER IS PROVIDED FOR THIS QUESTION.

2.7 QUICK START-UP PROCEDURE (XXDP+)

TO START-UP THIS PROGRAM:

1. BOOT XXDP+
2. GIVE THE DATE AND ANSWER THE LSI AND 50HZ (IF THERE IS A CLOCK) QUESTIONS
3. TYPE 'R NAME', WHERE NAME IS THE NAME OF THE BIN OR BIC FILE FOR THIS PROGRAM
4. TYPE 'START'
5. ANSWER THE 'CHANGE HW' QUESTION WITH 'Y'
6. ANSWER ALL THE HARDWARE QUESTIONS
7. ANSWER THE 'CHANGE SW' QUESTION WITH 'N'

WHEN YOU FOLLOW THIS PROCEDURE YOU WILL BE USING ONLY THE DEFAULTS FOR FLAGS AND SOFTWARE PARAMETERS. THESE DEFAULTS ARE DESCRIBED IN SECTIONS 2.3 AND 2.5.

3.0 ERROR INFORMATION

3.1 TYPES OF ERROR MESSAGES

THERE ARE THREE LEVELS OF ERROR MESSAGES THAT MAY BE ISSUED BY A DIAGNOSTIC: GENERAL, BASIC AND EXTENDED. GENERAL ERROR MESSAGES ARE ALWAYS PRINTED UNLESS THE 'IER' FLAG IS SET (SECTION 2.3). THE GENERAL ERROR MESSAGE IS OF THE FORM:

```
NAME  TYPE  NUMBER  ON UNIT NUMBER  TST NUMBER PC:XXXXXX
ERROR MESSAGE
```

WHERE: NAME = DIAGNOSTIC NAME
TYPE - ERROR TYPE (SYS FATAL DEV FATAL, HARD OR SOFT)
NUMBER = ERROR NUMBER
UNIT NUMBER = 0 - N (N IS LAST UNIT IN PTABLE)
TST NUMBER = TEST AND SUBTEST WHERE ERROR OCCURRED
PC:XXXXXX = ADDRESS OF ERROR MESSAGE CALL

BASIC ERROR MESSAGES ARE MESSAGES THAT CONTAIN SOME ADDITIONAL INFORMATION ABOUT THE ERROR. THESE ARE ALWAYS PRINTED UNLESS THE "IER" OR "IBR" FLAGS ARE SET (SECTION 2.3). THESE MESSAGES ARE PRINTED AFTER THE ASSOCIATED GENERAL MESSAGE.

EXTENDED ERROR MESSAGES CONTAIN SUPPLEMENTARY ERROR INFORMATION SUCH AS REGISTER CONTENTS OR GOOD/BAD DATA. THESE ARE ALWAYS PRINTED UNLESS THE "IER", "IBR" OR "IXR" FLAGS ARE SET (SECTION 2.3). THESE MESSAGES ARE PRINTED AFTER THE ASSOCIATED GENERAL ERROR MESSAGE AND ANY ASSOCIATED BASIC ERROR MESSAGES.

4.0 PERFORMANCE AND PROGRESS REPORTS

A STATISTICAL REPORT WILL AUTOMATICALLY BE PRINTED PERIODICALLY (APPROXIMATELY EVERY SEVENTEEN MINUTES) AND AT THE END OF TEST #4. IT CAN BE SUPPRESSED BY SETTING THE INHIBIT STATISTICAL REPORT FLAG (E.G. START/FLAGS:ISR). THIS IS THE SAME REPORT THAT CAN BE PRINTED ON DEMAND WITH THE PRINT COMMAND.

DURING TEST 1, 2, AND 3, THE REPORT WILL LOOK LIKE THE FOLLOWING EXAMPLE:

```
TEST 1 IN PROGRESS    RUN TIME 2:24:10
```

DURING TEST #4, THE REPORT WILL CONTAIN STATISTICS ON EACH DRIVE FOR THE CURRENT PASS OF THE TEST; FOR EXAMPLE:

```
TEST 4 IN PROGRESS    RUN TIME 2:24:10

UNIT   DRIVE   SERIAL-NUMBER   SEEKS   MBYTES READ   MBYTES WRITTEN   HARD ERRORS   SOFT ERRORS   ECC
0      0             1002      12000   36            22               0             0             1
1      4      67342102'12      14000   42            29               0             2             0
```

5.0 TEST SUMMARIES

TEST # 1 - UNIBUSS ADDRESSING TEST

THE PURPOSE OF TEST #1 IS TO COMPLETE THE TESTING OF THE UNIBUS INTERFACE IN THE UDA. THE UDA RESIDENT DIAGNOSTIC IS NOT ABLE TO COMPLETELY TEST THE UNIBUS INTERFACE BECAUSE COMMUNICATION WITH THE PDP-11 PROCESSOR IS NECESSARY. SPECIFICALLY, THIS TEST WILL:

1. CHECK THAT EVERY ADDRESS LINE ON THE UNIBUS CAN BE DRIVEN TO BOTH ONE AND ZERO STATES.

2. CHECK THAT THE UDA CAN INTERRUPT THE PDP-11 PROCESSOR AT THE PROPER PRIORITY LEVEL AND VECTOR.

3. EXERCISE THE UNIBUS INTERFACE BY TRANSFERRING BLOCKS OF DATA TO AND FROM UNIBUS MEMORY.

THIS TEST ASSUMES THAT THE FOLLOWING ARE BEING TESTED BY THE UDA RESIDENT DIAGNOSTIC:

1. ALL DATA BITS CAN BE WRITTEN AND READ CORRECTLY.

2. NPR CYCLES CAN BE EXECUTED CORRECTLY.

ONE AT A TIME, EACH UDA SELECTED FOR TESTING WILL BE INITIALIZED AND BROUGHT ON-LINE BY FOLLOWING THE INITIALIZATION PROTOCOL. SEVERAL SIZES OF THE HOST COMMUNICATIONS AREA WILL BE SUPPLIED TO ALLOW THE UDA RESIDENT DIAGNOSTIC TO DO THE MOST UNIBUS ADDRESS TESTING POSSIBLE. INTERRUPTS WILL BE DISABLED. ANY UDA RESIDENT DIAGNOSTIC ERRORS WILL BE REPORTED.

THE UDA WILL THEN BE INITIALIZED AGAIN, THIS TIME WITH INTERRUPTS ENABLED. THE VECTOR ADDRESS AND PRIORITY LEVEL WILL BE DETERMINED SOLELY FROM THE ANSWERS TO THE HARDWARE QUESTIONS. IF THE HARDWARE VECTORS TO THE WRONG ADDRESS, IT IS IMPOSSIBLE TO DETERMINE THE RESULT. A DESCRIPTIVE ERROR MESSAGE OF THE PROBLEM WILL NOT OCCUR (THE PROGRAM OR PROCESSOR MAY HANG OR AN UNRELATED MESSAGE MAY OCCUR). THEREFORE, THE MESSAGE "TESTING INTERRUPT ABILITY OF UDA AT ADR XXXXXX VEC XXX..." WILL BE PRINTED JUST BEFORE THE UDA IS REQUESTED TO CAUSE AN INTERRUPT AND THE WORD "COMPLETED" WILL BE PRINTED (ON THE SAME LINE) WHEN THE INTERRUPT TEST IS COMPLETED. IF THE WORD "COMPLETED" DOES NOT FOLLOW THE FIRST MESSAGE, IT SHOULD BE APPARENT THAT THE INTERRUPT CAUSED THE DIAGNOSTIC OR PROCESSOR TO GO ASTRAY. THE PRIORITY LEVEL OF THE INTERRUPT REQUEST WILL ALSO BE VERIFIED.

A "DIAGNOSTIC MACHINE" PROGRAM WILL THEN BE DOWNLINE LOADED IN THE UDA FROM THE MEMORY SPACE INCLUDED IN THE HOST COMMUNICATIONS AREA WHEN THE UDA WAS FIRST INITIALIZED. THE UDA RESIDENT DIAGNOSTIC HAS ALREADY VERIFIED THAT IT CAN ACCESS THESE MEMORY ADDRESSES, SO THE DOWNLINE LOAD COMMAND SHOULD PERFORM PROPERLY. THE "DIAGNOSTIC MACHINE" PROGRAM IS THEN STARTED.

THE "DIAGNOSTIC MACHINE" PROGRAM WILL ASK THE PDP-11 PROGRAM TO FILL FREE MEMORY (THAT MEMORY AVAILABLE TO THE PDP-11 PROGRAM THAT IS NOT BEING USED BY THE PROGRAM OR THE PDP-11 SUPERVISOR) WITH AN ADDRESSING PATTERN AND REPORT THE LOCATION AND SIZE OF THE FREE MEMORY. EVERY LOCATION OF FREE MEMORY WILL BE READ AND THE DATA CHECKED. THEN, ONE BY ONE, EACH ADDRESS LINE WILL BE TESTED AS FOLLOWS:

1. DETERMINE A TEST ADDRESS BY TAKING THE FIRST ADDRESS OF FREE MEMORY AND COMPLIMENTING THE ADDRESS BIT TO BE TESTED.

2. READ FROM THE TEST ADDRESS.

3. IF A NON-EXISTANT MEMORY ERROR OCCURRS, THE TEST IS COMPLETE.

4. WRITE ALL ONES TO THE FIRST ADDRESS OF FREE MEMORY THEN READ FROM THE TEST ADDRESS. IF DATA READ IS NOT ALL ONES, THEN TEST IS COMPLETE.

5. WRITE ZEROS TO THE FIRST ADDRESS OF FREE MEMORY THEN READ FROM THE TEST ADDRESS. IF DATA READ IS NOT ZEROS, THEN TEST IS COMPLETE.

6. REPORT UNIBUS ADDRESSING ERROR.

WHEN ALL ADDRESS BITS HAVE BEEN TESTED, THEN BLOCK TRANSFERS TO AND

FROM MEMORY WILL BE TESTED WITH DIFFERENT DATA PATTERNS. THIS DATA WILL BE TRANSFERRED AT THE RATE DISK DATA IS TRANSFERRED TO AND FROM MEMORY DURING NORMAL UDA OPERATION.

THE NEXT UDA SELECTED FOR TESTING WILL THEN BE TESTED IN THE SAME MANNER. WHEN ALL UDAS HAVE BEEN TESTED, TEST #1 WILL END.

TEST #2 - DISK RESIDENT DIAGNOSTIC TEST

THE PURPOSE OF TEST #2 IS TO EXECUTE THE DIAGNOSTICS THAT RUN IN EACH DISK DRIVE. THESE DIAGNOSTIC PROGRAMS MAY BE RESIDENT IN THE DISK DRIVE OR REQUIRE DOWNLINE LOADING FROM THE XXDP+ LOAD DEVICE. THESE DIAGNOSTIC PROGRAMS THAT RUN IN THE DISK DRIVES ARE NOT PART OF THIS DIAGNOSTIC PRODUCT, BUT ARE PRODUCED BY THE DISK DEVELOPMENT GROUP. THIS UDA DIAGNOSTIC PROGRAM ONLY KNOWS THE PROCEDURE TO EXECUTE THE DISK RESIDENT DIAGNOSTICS AND HOW TO DETERMINE WHETHER A TEST PASSED OR FAILED.

ONE AT A TIME, EACH UDA SELECTED FOR TESTING WILL BE INITIALIZED AND A "DIAGNOSTIC MACHINE" PROGRAM DOWNLINE LOADED. THE "DIAGNOSTIC MACHINE" PROGRAM WILL ASK WHAT DRIVES ARE TO BE TESTED, THEN WILL ISSUE SEVERAL ECHO FRAMES TO THE DISK DRIVE AND CHECK FOR THE CORRECT RESPONSE FROM THE DRIVE. THIS SHOULD SERVE AS A GOOD INDICATOR THAT THE UDA AND DISK DRIVE CAN COMMUNICATE.

A DIAGNOSE COMMAND WILL THEN BE ISSUED TO THE DRIVE TO REQUEST THE DRIVE RUN ALL OF ITS DIAGNOSTICS. IF THE DISK DRIVE REQUESTS A DOWNLINE LOAD OF A DRIVE DIAGNOSTIC, THE DIAGNOSTIC PROGRAM WILL BE READ FROM THE XXDP+ LOAD DEVICE, DOWNLINE LOAD THE FILE INTO THE DISK DRIVE AND START ITS EXECUTION. THERE IS NO LIMIT TO THE NUMBER OF DOWNLINE LOADS THAT CAN BE REQUESTED BY A DRIVE.

IF THE "MANUAL INTERVENTION MODE" SOFTWARE QUESTION WAS ANSWERED NO (DEFAULT) TESTING WILL PROCEED TO THE NEXT DRIVE. WHEN ALL DRIVES ON THE UDA HAVE BEEN TESTED, THE NEXT UDA SELECTED FOR TESTING WILL THEN BE TESTED IN THE SAME MANNER. WHEN ALL UDA'S HAVE BEEN TESTED, TEST #2 WILL END.

IF THE "MANUAL INTERVENTION MODE" SOFTWARE QUESTION WAS ANSWERED YES, AN INTERACTIVE MODE WILL BE ENTERED TO ALLOW THE OPERATOR TO PERFORM DIAGNOSTIC ACTIVITIES ON THE DISK DRIVE AS DESIRED. THE SERVICE MANUAL FOR THE DISK DRIVE MUST BE USED TO DETERMINE WHAT DIAGNOSTIC CAPABILITIES ARE AVAILABLE.

FIRST, A BRIEF DESCRIPTION OF AVAILABLE COMMANDS WILL BE PRINTED AS FOLLOWS:

```
TEST #2 MANUAL INTERVENTION ON UNIT xx UDA AT xxxxxx DRIVE xxx
TO WRITE AND READ MEMORY:
  W DATA REGION OFFSET
  R REGION OFFSET
TO RUN A DIAGNOSTIC:
  D REGION
TO EXIT QUESTIONING:
```

E
DATA, REGION AND OFFSET ARE HEX VALUES.
?

COMMANDS MAY BE TYPED AFTER THE QUESTION MARK PROMPT. EACH COMMAND WILL BE PROCESSED AS ENTERED AND RESULTS DISPLAYED IMMEDIATELY. THE EXIT COMMAND WILL ALLOW THE DIAGNOSTIC TO PROCEED.

READ AND WRITE COMMANDS WILL REMEMBER THE REGION AND OFFSET VALUES. SUCCESSIVE READ AND SUCCESSIVE WRITE COMMANDS WILL AUTOMATICALLY INCREMENT TO THE NEXT OFFSET IF THE REGION AND OFFSET VALUES ARE NOT TYPED. IF A REGION IS TYPED BUT NOT AN OFFSET, OFFSET ZERO WILL BE USED.

ONE TO FOUR BYTES OF DATA MAY BE ENTERED BY A SINGLE WRITE COMMAND, DEPENDING ON THE NUMBER OF DIGITS TYPED IN THE HEX VALUE. A READ COMMAND WILL ALWAYS RETURN FOUR BYTES OF DATA. EXAMPLES:

```
1.      W FF FFFC 4
2.      W 010203
3.      R FFFC 0004
          FFFC 0004/ FF 01 02 03
4.      W F0F1F2F3 FFFC
5.      R
          FFFC 0000/ F0 F1 F2 F3
6.      R
          FFFC 0004/ FF 01 02 03
```

COMMAND 1 WRITES ONE BYTE (FF) INTO REGION FFFC, OFFSET 4. COMMAND 2 WRITES THREE BYTES (01, 02 AND 03) INTO THE SAME REGION WITH OFFSETS 5, 6 AND 7. COMMAND 3 READS FOUR BYTES STARTING AT REGION FFFC OFFSET 4. COMMAND 4 WRITES FOUR BYTES AT REGION FFFC OFFSET 0. COMMANDS 5 AND 6 READ THE EIGHT BYTES.

THE DIAGNOSE COMMAND WILL REMEMBER THE REGION FROM PREVIOUS DIAGNOSE COMMANDS ONLY, BECAUSE THE REGION CONTAINING THE DIAGNOSTIC IS GENERALLY NOT THE SAME REGION USED TO WRITE PARAMETERS OR READ RESULTS. IF THE DIAGNOSTIC RETURNS ANY DATA, THE DATA WILL BE PRINTED IMMEDIATELY.

TEST #3 - DISK FUNCTION TEST

THE PURPOSE OF TEST #3 IS TO FUNCTIONALLY TEST THE DISK DRIVE. ON A DRIVE THAT IS WELL DIAGNOSED BY ITS DISK RESIDENT DIAGNOSTICS (EXECUTED BY TEST #2) THESE FUNCTIONAL TESTS WILL HAVE LITTLE VALUE. ON A DRIVE THAT HAS NO OR MINIMAL RESIDENT DIAGNOSTICS, THESE FUNCTIONAL TESTS WILL HAVE MORE VALUE.

TEST #3 WILL START BY INITIALIZING EACH UDA SELECTED FOR TESTING AND THEN DOWNLINE LOAD A "DIAGNOSTIC MACHINE" PROGRAM INTO EACH UDA. ONCE ALL UDAS HAVE BEEN STARTED, THE PDP-11 PROGRAM WILL RESPOND TO REQUESTS FROM ALL UDAS. WHEN ALL THE UDAS HAVE INDICATED THE END OF TESTING, TEST #3 WILL END.

THE "DIAGNOSTIC MACHINE" PROGRAM WILL PERFORM THE FOLLOWING FUNCTIONS

ON EACH DRIVE:

1. ISSUE A DRIVE CLEAR COMMAND.

2. ISSUE INITIATE RECALIBRATE COMMAND.

3. ISSUE A CHANGE MODE COMMAND TO ENABLE DIAGNOSTIC CYLINDER ACCESS AND SET THE DRIVE TO 512 BYTE SECTOR SIZE.

4. ISSUE INITIATE SEEK COMMAND TO LAST DIAGNOSTIC CYLINDER.

5. READ ALL FACTORY FORMATTED SECTOR HEADERS. IF NO HEADERS ON A TRACK CAN BE READ, REPORT THE ERROR, OTHERWISE CONTINUE.

6. STARTING WITH CYLINDER 0, GROUP 0 AND INCREMENTING THROUGH EVERY GROUP ON THE DISK, SEEK TO THE SELECTED GROUP, READ A HEADER ON TRACK 0 AND THEN SEEK TO THE FACTORY FORMATTED DIAGNOSTIC CYLINDER. READ FROM THE DIAGNOSTIC CYLINDER TO VERIFY DISK POSITIONED CORRECTLY.

7. ISSUE A CHANGE MODE COMMAND TO ENABLE FORMATTING OPERATIONS.

8. FORMAT ALL WRITABLE DBNS IN 512 BYTE FORMAT.

9. WRITE AND READ SEVERAL DATA PATTERNS TO EACH WRITABLE DBN. REPORT AN ERROR IF ALL DBNS ON ONE TRACK HAVE AN ERROR.

TEST #4 - DISK EXERCISER

THE PURPOSE OF TEST #4 IS TO EXERCISE THE DISK DRIVES IN A MANNER SIMILAR TO NORMAL USAGE UNDER STANDARD OPERATING SYSTEMS. EXECUTION OF THIS TEST SHOULD GIVE AN INDICATION OF THE PERFORMANCE OF THE DISK DRIVE. THIS TEST MAY BE RUN FOR LONG OR SHORT PERIODS OF TIME, DEPENDING ON HOW THE SOFTWARE QUESTIONS ARE ANSWERED.

THESE ARE TWO MODES OF OPERATION FOR TEST #4:

1. DEFAULT OPERATION ON THE ENTIRE AREA SELECTED (CUSTOMER OR DIAGNOSTIC) WITH ALL PARAMETERS SELECTED FOR RANDOM OPERATION AS SHOWN BY DEFAULT ANSWERS BELOW.

2. MANUAL INTERVENTION MODE WHERE A NUMBER OF QUESTIONS ARE ASKED AND OPERATION IS CONTROLLED BY THEIR ANSWERS.

WHICH MODE IS ENTIRELY DETERMINED BY THE ANSWER TO THE FIRST SOFTWARE QUESTION ASKING, 'ENTER MANUAL INTERVENTION MODE FOR SPECIAL DIAGNOSIS?'' THIS QUESTION WOULD NORMALLY HAVE BEEN ANSWERED NO (DEFAULT) AND TESTING WILL BEGIN IMMEDIATELY. IF ANSWERED YES, THE FOLLOWING SERIES OF QUESTIONS WILL BE ASKED FOR EACH UNIT SELECTED FOR TESTING:

THE FOLLOWING QUESTIONS REFER TO UNIT XX UDA AT XXXXXX DRIVE XXX

THIS MESSAGE WILL IDENTIFY TO WHICH DRIVE THE QUESTIONS ARE BEING ASKED. THE ENTIRE SERIES OF QUESTIONS WILL BE ASKED FOR EACH DRIVE, THERE IS NO SHORT WAY TO ANSWER LIKE IN THE

HARDWARE QUESTIONS.

NUMBER OF BAD BLOCKS (D) 0 ?

AN ANSWER IN THE RANGE OF 1 TO 16 WILL ALLOW THAT MANY BAD BLOCK NUMBERS TO BE ENTERED. THE PROGRAM WILL ALLOW WRITES AND READS TO THESE BLOCKS BUT NO ERROR MESSAGES WILL BE PRINTED FOR THESE BLOCKS. ERRORS ENCOUNTERED ON THESE BLOCKS WILL NOT APPEAR IN THE STATISTICS. ANSWER ZERO TO BYPASS ENTERING BAD BLOCKS.

BAD BLOCK (A) ?

THIS QUESTION WILL BE ASKED THE NUMBER OF TIMES REQUESTED BY THE PREVIOUS ANSWER. ANY DECIMAL NUMBER THAT CAN BE CONVERTED INTO A 28-BIT BINARY VALUE WILL BE ACCEPTED. NO OTHER ERROR CHECKING WILL BE MADE AT THIS TIME TO DETERMINE IF THE BLOCK NUMBER ACTUALLY EXISTS ON THE DISK.

READ ONLY (L) N ?

ANSWER YES TO DICTATE READ ONLY AND PREVENT TEST #4 FROM PERFORMING ANY WRITES TO THE DISK. NOTE THAT TEST #3 WILL STILL WRITE TO THE DIAGNOSTIC CYLINDERS.

WRITE ONLY (L) N ?

THIS QUESTION WILL ONLY BE ASKED IF THE PREVIOUS QUESTION WAS ANSWERED NO. ANSWER YES TO DICTATE WRITE ONLY.

CHECK ALL WRITES BY READING (L) N ?

ANSWER YES TO CAUSE ALL WRITES TO BE CHECKED BY READING THE DATA IMMEDIATELY AFTER THE WRITE OPERATION.

RANDOMLY CHECK WRITES BY READING (L) Y ?

THIS QUESTION WILL ONLY BE ASKED IF THE PREVIOUS QUESTION WAS ANSWERED NO. ANSWER YES FOR THE WRITE CHECK TO BE PERFORMED RANDOMLY. ANSWER NO IF WRITE CHECKS ARE NOT DESIRED.

DATA PATTERN - 0 FOR RANDOM SELECTION (D) 0 ?

THERE ARE 16 DATA PATTERNS AVAILABLE, SELECTED AS 1 TO 16. PATTERN NUMBER 0 WILL CAUSE PATTERNS 1 TO 15 TO BE RANDOMLY SELECTED FOR EACH WRITE. IF PATTERN NUMBER 16 IS SELECTED, THE FOLLOWING SET OF QUESTIONS WILL BE ASKED FOR A PATTERN TO BE INPUT.

ENABLE ECC DATA CORRECTION (L) Y ?

A YES ANSWER WILL ENABLE THE USE OF ECC TO CORRECT DATA ERRORS. NO ERROR MESSAGE WILL BE PRINTED WHEN THE ECC PROPERLY CORRECTS THE DATA ON THE FIRST READ OF ANY DISK BLOCK. READ RETRIES WILL END WHEN A RE-READ PRODUCES AN ECC CORRECTABLE ERROR. THE USE OF ECC CORRECTION WILL APPEAR ONLY IN THE STATISTICAL REPORT FOR THE DRIVE.

A NO ANSWER WILL PREVENT THE USE OF ECC. ALL ECC ERRORS WILL CAUSE AN ERROR MESSAGE TO BE PRINTED AND RETRIES TO BE ATTEMPTED.

COMPARE ALL DATA READ (L) N ?

ANSWER YES TO CAUSE A DATA COMPARE AFTER EVERY READ.

RANDOMLY COMPARE DATA READ (L) Y ?

THIS QUESTION WILL ONLY BE ASKED IF THE PREVIOUS QUESTION WAS ANSWERED NO. ANSWER YES FOR THE DATA COMPARE TO BE PERFORMED ON RANDOM RECORDS. ANSWER NO IF DATA COMPARES ARE NOT DESIRED.

ENABLE RETRIES (L) Y

A YES ANSWER WILL ENABLE RETRIES TO BE PERFORMED ON DISK ERRORS.

RANDOM SEEK MODE (L) Y ?

ANSWER YES TO CAUSE BLOCK NUMBERS TO BE CHOSEN RANDOMLY. ANSWER NO TO CAUSE BLOCK NUMBERS TO BE SELECTED SEQUENTIALLY UP AND DOWN THE DISK SURFACE.

DO YOU WISH TO:
   0 - TEST ENTIRE AREA SELECTED
   1 - SPECIFY BEGIN/END SETS TO TEST
   2 - SPECIFY TRACKS AND CYLINDERS TO TEST
   3 - SPECIFY GROUPS AND CYLINDERS TO TEST
   4 - SPECIFY CYLINDERS TO TEST
(D) 0 ?

THIS QUESTION SPECIFIES THE OPTIONS AVAILABLE TO LIMIT TESTING TO A PORTION OF THE SELECTED AREA (CUSTOMER OR DIAGNOSTIC) OF THE DISK. A ZERO ANSWER IS THE DEFAULT WHICH SPECIFIES TO USE THE ENTIRE AREA FOR THE TEST. OTHER ANSWERS WILL CAUSE ADDITIONAL QUESTIONS TO BE ASKED.

NUMBER OF BEGIN/END SETS (D) 1 ?
BEGIN BLOCK (A) 0 ?
END BLOCK (A) 0 ?

THESE QUESTIONS ARE ASKED IF BEGIN/END SETS WERE SELECTED TO LIMIT THE TESTING AREA (ANSWER 1). ONE TO FOUR SETS MAY BE SPECIFIED. THE BEGIN BLOCK AND END BLOCK QUESTIONS ARE ASKED AS MANY TIMES AS NEEDED.

NUMBER OF TRACKS TO TEST (D) 1 ?
TRACK (D) 0 ?

NUMBER OF GROUPS TO TEST (D) 1 ?
GROUP (D) 0 ?

ONE OF THESE SETS OF QUESTIONS IS ASKED IF EITHER TRACKS AND

CYLINDERS OR GROUPS AND CYLINDERS WAS SPECIFIED TO LIMIT THE TESTING AREA (ANSWERS 2 OR 3). UP TO SEVEN TRACKS OR GROUPS MAY BE SPECIFIED ON WHICH TESTING WILL BE LIMITED.

DO YOU WISH TO LIMIT THE CYLINDERS TESTED (L) N ?

THIS QUESTION IS ASKED ONLY AFTER THE TRACKS OR GROUPS HAVE BEEN SPECIFIED ABOVE. IF TESTING IS TO BE FURTHER LIMITED TO A SET OF CYLINDERS, ANSWER YES AND THE FOLLOWING TWO QUESTIONS WILL BE ASKED:

STARTING CYLINDER (A) 0 ?
ENDING CYLINDER (A) 0 ?

THESE QUESTIONS ARE ASKED IF THE QUESTION IMMEDIATELY ABOVE WAS ANSWERED YES OR IF CYLINDERS WERE SELECTED TO LIMIT THE TESTING AREA (ANSWER 4). ONE SET OF CYLINDER NUMBERS MAY BE SPECIFIED TO LIMIT THE TESTING AREA.

AFTER THE ABOVE QUESTIONS HAVE BEEN ASKED FOR ALL DRIVES SELECTED FOR TESTING, THE FOLLOWING QUESTIONS WILL BE ASKED IF DATA PATTERN 16 WAS SELECTED FOR ANY DRIVE:

NUMBER OF WORDS IN DATA PATTERN 16 (D) 1 ?
DATA WORD (O) 0 ?

DATA PATTERN 16 CAN BE INPUT BY THESE QUESTIONS. A DATA PATTERN CONSISTS OF A BUFFER OF ONE TO 16 WORDS WHICH IS REPEATED THROUGHOUT THE DATA PORTION OF THE DISK BLOCK. ENTER THE CONTENTS OF THE DATA PATTERN BUFFER. THE DATA WORD QUESTION WILL BE REPEATED AS NEEDED.

TEST #4 WILL START BY INITIALIZING EACH UDA SELECTED FOR TESTING AND THEN DOWNLINE LOADING A "DIAGNOSTIC MACHINE" PROGRAM INTO EACH UDA. THE "DIAGNOSTIC MACHINE" PROGRAM WILL ASK WHAT DRIVES ARE TO BE TESTED AND THEN WILL ASK FOR THE PARAMETERS FOR EACH DRIVE (THE ANSWERS TO THE MANJAL INTERVENTION QUESTIONS OR THEIR DEFAULTS). ONCE ALL UDAS HAVE BEEN STARTED, THE PDP-11 PROGRAM WILL RESPOND TO REQUESTS FROM ALL UDAS.

THE DISK WILL THEN BE EXERCISED ACCORDING TO THE PARAMETERS. THE EXERCISE CONSISTS OF SELECTING A DISK SECTOR, SEEKING TO THE PROPER CYLINDER, THEN READING OR WRITING THE SECTOR. THE PARAMETERS WILL CONTROL HOW THE DISK SECTOR IS SELECTED, WHETHER THE SECTOR IS WRITTEN OR READ AND WHETHER A WRITE IS FOLLOWED BY A READ (WRITE CHECK).

THE "DIAGNOSTIC MACHINE" PROGRAM WILL PERIODICALLY SEND STATISTICS TO THE PDP-11 PROGRAM. THESE STATISTICS WILL INCLUDE COUNTS OF READS, WRITES, SEEKS AND ERRORS ON A PER DRIVE BASIS. THE PDP-11 PROGRAM WILL ACCUMULATE THE STATISTICS FROM ALL THE UDAS AND WATCH FOR THE TRANSFER LIMIT TO BE EXCEEDED. AS LONG AS THE ERROR LOG IS NOT ENABLED, THE EXCEEDING OF THE TRANSFER LIMIT WILL CAUSE THE END OF TEST #4.

EACH TIME AN ERROR OCCURS, THE "DIAGNOSTIC MACHINE" WILL TELL THE PDP-11 PROGRAM. A MESSAGE WILL BE PRINTED (OR STORED IN THE LOG BUFFER) AND THEN THE ERROR LIMIT FOR THE DRIVE WILL BE CHECKED. IF THE

ERROR LIMIT HAS BEEN REACHED. THE DRIVE WILL BE DROPPED FROM TESTING. IF NO MORE DRIVES REMAIN TO BE TESTED, TEST #4 WILL END (UNLESS THE ERROR LOG IS ENABLED).

WHEN THE END OF TEST #4 OCCURS, THE ACCUMULATED STATISTICS FOR EACH DRIVE WILL BE PRINTED. THIS STATISTICAL REPORT CAN BE PRINTED AT ANY TIME DURING TEST #4 BY TYPING CONTROL-C THEN THE PRINT COMMAND.

THE DATA PATTERNS TO BE USED BY TEST #4 ARE INDICATED BELOW. EACH PATTERN IS GENERATED BY WRITING THE PATTERN NUMBER IN EACH 4-BIT NIBBLE OF THE FIRST WORD, THEN REPEATING THE DATA PATTERN (SEQUENCE OF ONE TO 16 WORDS) THROUGHOUT THE REST OF THE DATA BUFFER. PATTERN NUMBER 16 WRITES NIBBLES OF ZEROS. WHEN PATTERN NUMBER ZERO IS USED, THE ACTUAL PATTERN NUMBER WRITTEN (1 TO 15) IS PLACED IN THE NIBBLES.

PATTERN 0 THIS PATTERN NUMBER IS USED TO INDICATE ANY PATTERN NUMBER 1 TO 15 CHOSEN AT RANDOM.

PATTERN 1 WORDS IN PATTERN SEQUENCE - 1

SEQUENCE (OCTAL) 105613
SEQUENCE (HEX) 8888

PATTERN 2 WORDS IN PATTERN SEQUENCE - 1

SEQUENCE (OCTAL) 031463
SEQUENCE (HEX) 3333

PATTERN 3 WORDS IN PATTERN SEQUENCE - 1

SEQUENCE (OCTAL) 030221
SEQUENCE (HEX) 3091

PATTERN 4 WORDS IN PATTERN SEQUENCE - 16 (SHIFTING ONES)

SEQUENCE (OCTAL) 000001, 000003, 000007, 000017, 000037,
000077, 000177, 000377, 000777, 001777,
003777, 007777, 017777, 037777, 077777,
177777

SEQUENCE (HEX) 0001, 0003, 0007, 000F, 001F, 003F,
007F, 00FF, 01FF, 03FF, 07FF, 0FFF,
1FFF, 3FFF, 7FFF, FFFF

PATTERN 5 WORDS IN PATTERN SEQUENCE - 16 (SHIFTING ZEROS)

SEQUENCE (OCTAL) 177776, 177774, 177770, 177760, 177740,
177700, 177600, 177400, 177000, 176000,
174000, 170000, 160000, 140000, 100000,
000000

SEQUENCE (HEX) FFFE, FFFC, FFF8, FFF0, FFE0, FFC0,
FF80, FF00, FE00, FC00, F800, F000,
E000, C000, 8000, 0000

PATTERN 6 WORDS IN PATTERN SEQUENCE - 16

SEQUENCE (OCTAL) 000000, 000000, 000000, 177777, 177777,
177777, 000000, 000000, 177777, 177777,
000000, 177777, 000000, 177777, 000000,
177777

SEQUENCE (HEX) 0000, 0000, 0000, FFFF, FFFF, FFFF,
0000, 0000, FFFF, FFFF, 0000, FFFF,
0000, FFFF, 0000, FFFF

PATTERN 7 WORDS IN PATTERN SEQUENCE - (BINARY 1011011011011001)

SEQUENCE (OCTAL) 133331
SEQUENCE (HEX) B6D9

PATTERN 8 WORDS IN PATTERN SEQUENCE - 16

SEQUENCE (OCTAL) 052525, 052525, 052525, 125252, 125252,
125252, 052525, 052525, 125252, 125252,
052525, 125252, 052525, 125252, 052525,
125252

SEQUENCE (HEX) 5555, 5555, 5555, AAAA, AAAA, AAAA,
5555, 5555, AAAA, AAAA, 5555, AAAA,
5555, AAAA, 5555, AAAA

PATTERN 9 WORDS IN PATTERN SEQUENCE - 1 (BINARY 1101101101101100)

SEQUENCE (OCTAL) 155554
SEQUENCE (HEX) DB6C

PATTERN 10 WORDS IN PATTERN SEQUENCE - 16

SEQUENCE (OCTAL) 026455, 026455, 026455, 151322, 151322,
151322, 026455, 026455, 151322, 151322,
026455, 151322, 026455, 151322, 026455,
151322

SEQUENCE (HEX) 2D2D, 2D2D, 2D2D, D2D2, D2D2, D2D2,
2D2D, 2D2D, D2D2, D2D2, 2D2D, D2D2,
2D2D, D2D2, 2D2D, D2D2

PATTERN 11 WORDS IN PATTERN SEQUENCE - 1 (BINARY 0110110110110110)

SEQUENCE (OCTAL) 066666
SEQUENCE (HEX) 6DD6

PATTERN 12 WORDS IN PATTERN SEQUENCE - 16 (RIPPLE ONE)

SEQUENCE (OCTAL) 000001, 000002, 000004, 000010, 000020,
000040, 000100, 000200, 000400, 001000,
002000, 004000, 010000, 020000, 040000,
100000

SEQUENCE (HEX) 0001, 0002, 0004, 0008, 0010, 0020,
0040, 0080, 0100, 0200, 0400, 0800,
1000, 2000, 4000, 8000

```
                        PATTERN 13      WORDS IN PATTERN SEQUENCE - 16  (RIPPLE ZERO)

                        SEQUENCE (OCTAL)        177776, 177775, 177773, 177767, 177757,
                        177737, 177677, 177577, 177377, 176777,
                        175777, 173777, 167777, 157777, 137777,
                        077777

                        SEQUENCE (HEX)  FFFE, FFFD, FFFB, FFF7, FFEF, FFDF,
                        FFBF, FF7F, FEFF, FDFF, FBFF, F7FF,
                        EFFF, DFFF, BFFF, 7FFF

                        PATTERN 14      WORDS IN PATTERN SEQUENCE - 3

                        SEQUENCE (OCTAL)        155555, 133333, 155555
                        SEQUENCE (HEX)  DB6D, B6DB, DB6D

                        PATTERN 15      WORDS IN PATTERN SEQUENCE - 16

                        SEQUENCE (OCTAL)        133331, 133331, 133331, 155554, 155554,
                        155554, 133331, 133331, 155554, 155554,
                        133331, 155554, 133331, 155554, 133331,
                        155554

                        SEQUENCE (HEX)  B6D9, B6D9, B6D9, DB6C, DB6C, DB6C,
                        B6D9, B6D9, DB6C, DB6C, B6D9, DB6C,
                        B6D9, DB6C, B6D9, DB6C

                        PATTERN 16      THIS IS THE OPERATOR SELECTABLE PATTERN IN MANUAL
                        INTERVENTION MODE.  QUESTIONS ARE ASKED WHEN TEST #4 IS
                        STARTED FOR THE OPERATOR TO INPUT THE NUMBER OF WORDS IN
                        THE SEQUENCE AND THE CONTENTS OF THE WORDS.


                                                                &

1067            000001                  .END
```

SYMBOL TABLE

```
. ABS.  000000     000
        000000     001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  8192 WORDS  ( 32 PAGES)
DYNAMIC MEMORY AVAILABLE FOR  72 PAGES
,B:ZUDDA0=B:ZUDDA0.DOC


UDAT1   UNIBUS ADDRESSING       MACRO X04.00 9-JUL-81 11:25:16
TABLE OF CONTENTS

    3-   1       UDA DM PROGRAM PARAMETERS
    7-   1       TEST 4 SPECIFIC INFORMATION
    9-   1       MACRO DEFINITIONS
   20-   1       START OF TEST CODE
   21-   1       RDSTAT - GET DRIVE'S REAL TIME DRIVE STATE
   22-   1       HOSTRQ - HOST REQUEST - REPORT ERRORS, MEGABYTES TRANSFERRED, ETC.
   32-   1       FREE MEMORY CHECK
```

```
                                .TITLE  UDAT1   UNIBUS ADDRESSING


                        ;
                        ; COPYRIGHT (C)  1980
                        ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
                        ;
                        ;
                        ; THIS  SOFTWARE  IS  FURNISHED  UNDER A LICENSE FOR USE ONLY ON A
                        ; SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION
                        ; OF  THE  ABOVE COPYRIGHT  NOTICE.  THIS SOFTWARE, OR  ANY  OTHER
                        ; COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE  MADE AVAILABLE
                        ; TO  ANY  OTHER PERSON  EXCEPT  FOR USE ON SUCH SYSTEM AND TO ONE
                        ; WHO AGREES TO THESE LICENSE TERMS.  TITLE  TO  AND  OWNERSHIP OF
                        ; THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DIGITAL.
                        ;
                        ; THE  INFORMATION  IN THIS DOCUMENT IS SUBJECT  TO CHANGE WITHOUT
                        ; NOTICE AND SHOULD NOT BE CONSTRUED AS A  COMMITMENT  BY  DIGITAL
                        ; EQUIPMENT CORPORATION.
                        ;
                        ; DIGITAL ASSUMES NO  RESPONSIBILITY FOR THE USE OR RELIABILITY OF
                        ; ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;THIS PROGRAM SHALL BE ASSEMBLED WITH THE PROGRAM DMACRO
                        ;USING A COMMAND LINE SIMILAR TO:
                        ;
                        ;  UDAT1.BIN,UDAT1/C=[1,2]DMACRO,UDAT1T,UDATP,UDAT1M,UDATR,UDAT1S

000000                  TEST4   =       0                       ; THIS IS NOT TEST4
000000                          .ENABL  ABS
000000                          DMCODE  UDAT1,0,714,3,0
```

```
                        .SBTTL  UDA DM PROGRAM PARAMETERS
                                .LIST MEB
                        ;
                        ;       EQUATES
                        ;
                        ;       HIGHEST USABLE LOCATION OF UDA MEMORY + 1
                        ;
010000                  HIMEM   =       10000                   ; HIGH MEMORY+1
007774                  OVSTRT  =       7774                    ; OVERLAY ADDRESS LOCATION
                        ;
                        ;
                        ;       OFFSETS FOR FORMAT TRACK TABLE
                        ;
000000                  FT.BUF  =       0.                      ;BUFFER POINTER OFFSET
000001                  FT.HI   =       1.                      ;HI ORDER HEADER OFFSET
000002                  FT.LOW  =       2.                      ;LOW ORDER HEADER OFFSET
                        ;
                        ;
                        ;       OFFSETS FOR FORMAT TRACK BUFFER
                        ;
000000                  FB.DAT  =         0.                    ;FIRST DATA WORD OFFSET
000400                  FB.EDC  =       256.                    ;EDC WORD OFFSET
                        ;
                        ;
                        ;       OFFSETS FOR READ/WRITE I/O CHAIN TABLES
                        ;
000000                  RW.STAT =       0.                      ;STATUS AND NEXT BUFFER POINTER OFFSET
000001                  RW.BUF  =       1.                      ;POINTER TO DATA BUFFER
000002                  RW.LOW  =       2.                      ;HI ORDER EXPECTED HEADER
000003                  RW.HI   =       3.                      ;LOW ORDER EXPECTED HEADER
000004                  RW.CMD  =       4.                      ;SDI COMMAND AND HEAD ADDRESS
000005                  RW.SDI  =       5.                      ;DUMMY SDI CONTROL BLOCK POINTER
000006                  RW.ANG  =       6.                      ;THETA FROM INDEX
                        ;
                        ;       CONSTANTS FOR READ AND WRITE XFC'S
                        ;
140000                  WSTOP   =       140000                  ; LAST ENTRY IN CHAIN FOR WRITE
040000                  WCONT   =       40000                   ; WRITE CONTINUE
100000                  RSTOP   =       100000                  ; LAST ENTRY IN CHAIN FOR READ
000000                  RCONT   =       0                       ; READ CONTINUE
100000                  FSTOP   =       100000                  ; LAST ENTRY IN CHAIN FOR FORMAT
122400                  WREAL   =       122400                  ; WRITE REAL TIME ECOMMAND
013400                  RREAL   =       13400                   ; READ REAL TIME COMMAND
010000                  ECCFLG  =       10000                   ; ECC ERROR IN BUFFER BIT
100000                  EOC     =       100000                  ; END OF CHAIN
040000                  BUFFLG  =       40000                   ; BUFFER FULL OR EMPTY FLAG
                        ;
                        ;
                        ;       HEADER CODES
                        ;
000000                  HD.LBN  =       000000                  ;GOOD LBN
060000                  HD.RBN  =       060000                  ;GOOD RBN. PERHAPS UNUSED
030000                  HD.REV  =       030000                  ;REVECTORED LBN
110000                  HD.BAD  =       110000                  ;BAD BLOCK
050000                  HD.PRV  =       050000                  ;PRIMARY REVECTORED BLOCK
120000                  HD.XBN  =       120000                  ;XBN BLOCK
```

```
      58          140000          HD.DBN  =       140000                  ;DBN BLOCK
      59
      60                          ;       OFFSETS FOR DATA BUFFERS
      61
      62          000000          BF.DAT  =         0.                    ;DATA
      63          000400          BF.EDC  =       256.                    ;FRROR DETECTION CODE
      64          000401          BF.ECC  =       257.                    ;LAST 17 ECC RESIDUES
      65
      66                          ;       BUFFER AND READ/WRITE CHAIN LINK SIZES
      67
      68          000401          WBUFLN  =       257.                    ; WRITE BUFFER SIZE
      69          000415          RBUFLN  =       WBUFLN+12.              ; READ BUFFER SIZE
      70          000007          LINKLN  =       7.                      ; LINK SIZE
```

```
                        ;       XFC DEFINITION EQUATES
                        ;
000000                  BREAK   =       0.              ;BREAKPOINT XFC CODE
000001                  FORMAT  =       1.              ;FORMAT TRACK XFC CODE
000002                  XREAD   =       2.              ;READ N SECTORS XFC CODE
000003                  XWRITE  =       3.              ;WRITE N SECTORS XFC CODE
000004                  SEND    =       4.              ;SEND SDI COMMAND XFC CODE
000005                  RCV     =       5.              ;RECEIVE SDI MESSAGE XFC CODE
000006                  COMPARE =       6.              ;COMPARE DATA PATTERN TO BUFFER
000007                  STATUS  =       7.              ;RETURN DRIVE STATUS XFC CODE
000010                  ECHO    =       8.              ;ECHO DATA TO DRIVE XFC CODE
000011                  DINIT   =       9.              ;DRIVE INITIALIZE XFC CODE
000012                  WAITSI  =       10.             ;WAIT FOR SECTOR OR INDEX PULSE
000013                  UREAD   =       11.             ;READ UNIBUS MEMORY XFC CODE
000014                  UWRITE  =       12.             ;WRITE UNIBUS MEMORY XFC CODE
000015                  ECC     =       13.             ;DO ECC ON BUFFER XFC CODE
000016                  MRD     =       14.             ;SEND BUFFER TO MAINTENANCE READ COMMAND
000017                  MWR     =       15.             ;GET BUFFER FROM MAINTENANCE WRITE COMMAND
000020                  CVT     =       16.             ;CONVERT TO PHYSICAL ADDRESS XFC CODE
000021                  EXIT    =       17.             ;TERMINATE DM PROGRAM XFC CODE
                        ;
                        ;       GET STATUS OFFSETS
                        ;
000000                  ST.UNT  =       0.              ;UNIT NUMBER
000000                  ST.MSK  =       0.              ;SUBUNIT MASK
000001                  ST.STA  =       1.              ;STATUS BYTE
000001                  ST.MOD  =       1.              ;MODE BYTE
000002                  ST.ERR  =       2.              ;ERROR BYTE
000002                  ST.CON  =       2.              ;CONTROLLER BYTE
000002                  ST.C    =       2.              ;C BITS
000003                  ST.RTY  =       3.              ;RETRY COUNT/FAILURE CODE
                        ;
                        ;       STATUS BIT DEFINITIONS
                        ;
000200                  ST.OA   =       200             ; ONLINE TO ANOTHER (SET IF DRIVE UNAVAILABLE)
000100                  ST.RR   =       100             ; READJUSTMENT BIT (SET IF RECALIBRATION REQUIRED)
000040                  ST.DR   =       40              ; DIAGNOSTIC REQUEST (SET IF DIAGNOSTIC REQUESTED)
000020                  ST.SR   =       20              ; SPINDLE READY (SET IF SPINDLE READY)
000002                  ST.PS   =       2               ; PORT SWITCH (SET IF PORT SWITCH IN)
000001                  ST.RU   =       1               ; RUN/STOP SWITCH (SET IF RUN/STOP SWITCH IN)
000200                  ST.FE   =       200             ; FATAL ERROR (SET IF FATAL ERROR OCCURRED)
000100                  ST.RE   =       100             ; RETRIABLE ERROR (SET IF RETRIABLE ERROR OCCURRED)
000040                  ST.PE   =       40              ; PROTOCOL ERROR (SET IF PROTOCOL ERROR OCCURRED)
000020                  ST.DF   =       20              ; INITIALIZATION FAILURE (SET IF INIT FAILED)
000010                  ST.WE   =       10              ; WRITE ENABLE (SET IF WRT ATTEMPTED ON PROT DISK)
002000                  ST.FO   =       2000            ; FORMATTING (SET IF FORMATTING ENABLED)
001000                  ST.DB   =       1000            ; DIAGNOSTIC CYLS (SET IF DIAG CYL ACCESS ENABLED)
000400                  ST.S7           400             ; SECTOR SIZE (SET FOR 576 BYTE SECTORS)
```

```
                                ;       GET COMMON CHARACTERISTICS OFFSETS
                                ;
000000                          SHRTTO  -       0.              ;SHORT TIMEOUT <3:0>
000000                          SDIVER  =       0.              ;SDI VERSION <7:4>
000000                          XFERRT  =       0.              ;TRANSFER RATE <15:0>
000001                          LONGTO  =       1.              ;LONG TIMEOUT <3:0>
000001                          RETS    =       1.              ;RETRIES <7:4>
000001                          RCTCPS  =       1.              ;F/RCT COPIES <11:8>
000001                          SS      =       1.              ;SECTOR SIZE <15:15>
000002                          ERLEV   =       2.              ;ERROR RETRY LEVELS <7:0>
000002                          ECCRSH  =       2.              ;ECC THRESHOLD <15:8>
000003                          MICREV  =       3.              ;MICROCODE REVISION NUMBER <7:0>
000003                          HRDREV  =       3.              ;HARDWARE REVISION NUMBER <15:8>
000004                          DRVID   =       4.              ;UNIQUE DRIVE ID <47:0>
000007                          DRTYPE  =       7.              ;DRIVE TYPE IDENTIFIER <7:0>
000007                          REVS    =       7.              ;REVS/SECOND <15:8>
                                ;
                                ;       GET SUBUNIT CHARACTERISTICS OFFSETS
                                ;
                                ;THESE OFFSETS ARE CURRENTLY GIVEN AS FOLLOWING THE COMMON CHARACTERISTICS
                                ;
000013                          SUB     =       11.             ;OFFSET TO PUT SUBUNIT AFTER COMMON;
000000                          LBNCYL  =       0.              ;NUMBER OF CYLINDERS IN LBN AREA <31:0>
000001                          HICYL   =       1.              ;HI ORDER CYLINDER BITS <15:12>
000002                          GRPCYL  =       2.              ;GROUPS PER CYLINDER <7:0>
000002                          HILBN   -       2.              ;HI STARTING LBN <11:8>
000002                          HIXBN   =       2.              ;HI STARTING XBN <15:12>
000003                          TRKGRP  -       3.              ;TRACKS PER GROUP <7:0>
000003                          HIRBN   -       3.              ;HI STARTING RBN <11:8>
000003                          HIDBN   =       3.              ;HI STARTING DBN <15:12>
000004                          RBNTRK  =       4.              ;RBNS PER TRACK <6:0>
000004                          RM              4.              ;REMOVABLE MEDIA <7:7> 1=REMOVEABLE
000005                          DATPRE  -       5.              ;DATA PREAMBLE SIZE IN WORDS <7:0>
000005                          HDRPRE  -       5.              ;HEADER PREAMBLE SIZE IN WORDS <15:8>
000006                          MEDTYP  =       6.              ;MEDIA TYPE <31:0>
000010                          FCTSIZ  =       8.              ;FCT COPY SIZE <15:0>
000011                          LBNTRK  =       9.              ;LBNS PER TRACK <7:0>
000011                          GRPOFF  =       9.              ;GROUP OFFSET (SECTORS) <15:8>
000012                          LBNHST  =       10.             ;LBNS IN HOST AREA <31:0>
000014                          RCTCSZ  =       12.             ;RCT COPY SIZE <15:0>
000021                          XBNCYL  =       17.             ;CYLS IN XBN AREA <15:0>
000022                          DBNCYL  -       18.             ;CYLS IN DBN AREA <15:8>
                                ;
                                ;       UNIT CODES
                                ;
000001                          UNIT0   =       1.              ;UNIT ZERO CODE
000002                          UNIT1   =       2.              ;UNIT ONE CODE
000004                          UNIT2   =       4.              ;UNIT TWO CODE
000010                          UNIT3   =       8.              ;UNIT THREE CODE
                                ;
                                ;       BIT MASK DEFINITIONS
                                ;
                                ;
177400                          HIBYTE  -       177400          ;HIGH BYTE MASK
000377                          LOBYTE  =       000377          ;LOW BYTE MASK
007777                          HBHINB  =       7777            ;HI BYTE, HI NIBBLE MASK
170377                          HBLONB          170377          ;HI BYTE, LO NIBBLE MASK
```

UDA DM PROGRAM PARAMETERS

```
58          177417          LBHINB  =       177417          ;LO BYTE, HI NIBBLE MASK
59          177760          LBLONB  =       177760          ;LO BYTE, LO NIBBLE MASK
60                          ;
61          000001          TIMEOUT =       1.              ;DRIVE TIMEOUT CODE
62          000002          HEADER  =       2.              ;HEADER COMPARE FAILURE CODE
63          000004          REVECT  =       4.              ;REVECTOR NEEDED CODE
64                          ;
65          000002          WRONG   =       2.              ;FIRST WORD NOT START FRAME CODE
66          000004          FRAME   =       4.              ;FRAMING ERROR CODE
67          000010          CHECK   =       8.              ;CHECKSUM ERROR CODE
68                          ;
69          000001          TOOBIG  =       1.              ;NUMBER OF WORDS EXCEEDS 7064
70          000002          LOW     =       2.              ;DM BUFFER ADDRESS IS LESS THAN 714
71                          ;
72                          ;       BIT MASK
73                          ;
74          000001          LARGE   =       1.              ;BLOCK NUMBER TOO LARGE
75          000002          OVERFL  =       2.              ;SECTOR NUMBER LARGER THAN 16 BITS
```

```
                        ;MAINTANENCE READ/WRITE REQUEST NUMBERS
                        ;
000000                  T1MSIZ  =       0.      ;GET FREE MEMORY PARAMETERS
000001                  T2DLL   =       1.      ;DOWNLINE LOAD DRIVE DIAGNOSTIC
000002                  T2CMD   =       2.      ;MANUAL INTERVENTION TEST 2 PROTOCOL
000003                  T4MPRM  =       3.      ;GET MASTER PARAMETERS FROM SW QUESTIONS
000004                  T4UPRM  =       4.      ;GET UNIT PARAMETERS FROM HW QUESTIONS
000005                  T4BB1   =       5.      ;GET BAD BLOCKS (1 THRU 14)
000006                  T4BB2   =       6.      ;GET REST OF BAD BLOCKS (15 AND 16)
000007                  T4SOFT  =       7.      ;ADD TO SOFT ERROR AND ECC COUNT
000010                  T4SEEK  =       8.      ;ADD 1 TO SEEK COUNT
000011                  T4MXFR  =       9.      ;ADD TO MEGABITS READ AND WRITTEN
000012                  UTOTST  =       10.     ;GET UNITS TO TEST
000013                  ERRMES  =       11.     ;PRINT ERROR MESSAGE
000014                  ERRMC   =       12.     ;TEST 4 ERROR REPORTING
000015                  MESSAG  =       13.     ;INFORMATION MESSAGE
000016                  DONE    =       14.     ;MARK DM PROGRAM AS NO LONGER RUNNING
                        ;
                        ;       OTHER BIT DEFINITIIONS
                        ;
000001                  RCVRDY  =       1       ; RECIEVER READY 1 - READY
000002                  ATTN    =       2       ; ATTENTION BIT FOR RETURN DRIVE SIGNALS XFC
000004                  RCVERR  =       4       ; RECIEVER ERROR
000100                  AVAIL   =       100     ; AVAILABLE 1 = AVAILABLE
000400                  XMTERR          400     ; TRANSMIT ERROR
100000                  RWRDY   =       100000  ; IF SET, UDA IS ABLE TO READ AND/OR WRITE TO DRIVE
                        ;
                        ;       SDI COMMANDS AND RESPONSES
                        ;
000204                  DISCON  =       204     ; DISCONNECT DRIVE
000006                  ERECOV  =       6       ; ERROR RECOVERY
000201                  CHGMOD  =       201     ; CHANGE MODE
000213                  DRVONL  =       213     ; DRIVE ONLINE
000014                  DRVRUN  =       14      ; DRIVE RUN
000005                  DRVCLR  =       5       ; DRIVE CLEAR OPCODE
000207                  GETCHR  =       207     ; GET CHARACTERISTICS
000210                  GETSUB  =       210     ; GET SUBUNIT CHARACTERISTICS
000011                  GETSTA  =       11      ; GET STATUS
000216                  IRECLB  =       216     ; RECALIBRATE
000012                  INSEEK  =       12      ; INITIATE SEEK
000176                  COMPLT  =       176     ; SUCCESSFUL COMPLETION
000175                  UNSSUC  =       175     ; UNSUCCESSFUL COMPLETION
000170                  CHRRES  =       170     ; GET CHARACTERISTICS RESPONSE
000167                  SBCRES  =       167     ; GET SUBUNIT CHARACTERISTICS RESPONSE
000366                  STSRES  =       366     ; GET STATUS RESPONSE
000350                  ECHOC           350     ; DIAGNOSTIC ECHO COMMAND AND RESPONSE
                        ;
                        ;       ERROR CODES
                        ;
000000                  FTLSYS  =       0       ; SYSTEM FATAL ERROR
000400                  FTLDEV  =       400     ; DEVICE FATAL
001000                  ERHARD  =       1000    ; HARD ERROR
001400                  ERSOFT  =       1400    ; SOFT ERROR
```

TEST 4 SPECIFIC INFORMATION

```
                        .SBTTL  TEST 4 SPECIFIC INFORMATION
                        ;
                        ;       TEST 4 SPECIFIC INFORMATION
                        ;
                        ;
                        ;       CONSTANTS
                        ;
000377          SCTWRD  =       255.            ; NUMBER OF WORDS IN SECTOR TO FILL
000105          INTEDC  =       69.             ; INITIAL EDC VALUE
000061          TLEN.U  =       U.LGRP+1        ; UNIT PARAMETER LENGTH
007717          FIRSTU  =       HIMEM-TLEN.U    ; LOCATION OF FIRST UNIT PARAMETER BLOCK
                        ;
                        ;       UNIT PARAMETER OFFSETS
                        ;
000000          U.NEXT  =       0.              ; POINTER TO NEXT UNIT (RING LINKED LIST)
000001          U.SUBP  =       U.NEXT+1        ; 4 WORDS OF SUBUNIT PARAMETER POINTERS
000005          U.TIMO  =       U.SUBP+4        ; AREA TO STORE VARIOUS TIMEOUT VALUES
000006          U.RWTO  =       U.TIMO+1        ; READ/WRITE TIMEOUT AREA
000007          U.SEEK  =       U.RWTO+1        ; NUMBER OF SEEKS ISSUED
000010          U.NFUN  =       U.SEEK+1        ; NEXT FUNCTION ADDRESS (FOR DEFERRED CALLS)
000011          U.PAT   -       U.NFUN+1        ; PATTERN NUMBER TO WRITE
000012          U.CCNT  -       U.PAT+1         ; CURRENT COUNT OF T/G LOOPS
000014          U.PCTG  =       U.CCNT+2        ; POINTER TO CURRENT TRACK OR GROUP
000015          U.CTRK  =       U.PCTG+1        ; TRACK COUNT FOR GROUP OPERATIONS
000016          U.NSEC  -       U.CTRK+1        ; NUMBER OF SECTORS R/W THIS TRY
000017          U.MSEC  =       U.NSEC+1        ; NUMBER OF SECTORS TO BE R/W
000020          U.TSEC  =       U.MSEC+1        ; NUMBER OF SECTORS TO BE R/W THIS OP
000021          U.CSEC  =       U.TSEC+1        ; COUNT OF SECTORS R/W SO FAR
000022          U.MASK  =       U.CSEC+1        ; UNIT MASK FOR XFC CALLS (0001 - 1000)
000023          U.WRIT  =       U.MASK+1        ; WRITE PROTECTION STATUS
000024          U.ELEV  =       U.WRIT+1        ; CURRENT ERROR RECOVERY LEVEL
000025          U.RTRY  -       U.ELEV+1        ; MAXIMUM NUMBER OF READ RETRIES
000026          U.MLEV  -       U.RTRY+1        ; MAXIMUM NUMBER OF ERROR RECOVERY LEVELS
000027          U.ECCT  =       U.MLEV+1        ; ECC THRESHOLD
000030          U.SDIS          U.ECCT+1        ; SDI SHORT TIMEOUT
000031          U.SDIL  =       U.SDIS+1        ; SDI LONG TIMEOUT
000032          U.WPRT  -       U.SDIL+1        ; MASK TO WRITE PROTECT READ-ONLY DRIVES
000033          U.PARM  =       U.WPRT+1        ; UNIT PARAMETER WORD
000034          U.SUBU  -       U.PARM+1        ; SUBUNIT OFFSET (0 - 3)
000035          U.MBN   =       U.SUBU+1        ; MASTER L/DBN
000037          U.CBN   =       U.MBN+2         ; CURRENT L/DBN FOR START OF CHAIN
000041          U.RBN   =       U.CBN+2         ; RBN TO BE READ/WRITTEN IF LBN REVECTORED
000043          U.COPY  -       U.RBN+2         ; NUMBER OF RCT COPIES ON EACH SUBUNIT
000044          U.CCOP  -       U.COPY+1        ; CURRENT RCT COPY THAT WE'RE WORKING ON
000045          U.RWER  -       U.CCOP+1        ; ERROR (IF ANY) ON THE LAST R/W
000046          U.RVER  =       U.RWER+1        ; ERROR THAT OCCURRED BEFOR THE REVECTOR OPERATION
000047          U.SNUM  -       U.RVER+1        ; 4 WORDS THAT HOLD THE SUBUNIT LOGICAL NUMBERS
000053          U.CCYL  =       U.SNUM+4        ; CURRENT CYLINDER
000055          U.CGRP  -       U.CCYL+2
000056          U.LCYL  =       U.CGRP+1        ; LAST CYLINDER SEEKED TO
000060          U.LGRP  =       U.LCYL+2        ; LAST GROUP SEEKED TO
                        ;
                        ;       SUBUNIT PARAMETER OFFSET
                        ;
000000          S.PARM  =       0.              ; SUBUNIT PARAMETER WORD
000001          S.SDCL  -       S.PARM+1        ; STARTING DIAGNOSTIC CYLINDER
000003          S.PAT   -       S.SDCL+2        ; PATTERN TO USE FOR WRITES
```

```
                000004          S.TRKL  =       S.PAT+1                 ; NUMBER OF SECTORS IN ONE TRACK
                000005          S.SCHR  =       S.TRKL+1                ; POINTER TO SUBUNIT CHARACTERISTICS
                000006          S.MEGR  =       S.SCHR+1                ; SECTORS READ (UP TO 245)
                000007          S.MEGW  =       S.MEGR+1                ; SECTORS WRITTEN (UP TO 245)
                000010          S.BADP  =       S.MEGW+1                ; POINTER TO BAD BLOCK AREA
                000011          S.BESS  =       S.BADP+1                ; START OF BEGIN/END SETS
                                ;
                                ;       IF TRACK/GROUP LIMITS ARE GIVEN, THE SUBUNIT PARAMETERS HAVE THE
                                ;       FOLLOWING FIELDS ADDED TO THEM
                                ;
                000011          S.MCNT  =       S.BESS                  ; MAXIMUM TRACK/GROUP COUNT
                000013          S.TGOF  =       S.MCNT+2                ; ORIGINAL TRACK/GROUP OFFSET
                000015          S.TGSS  =       S.TGOF+2                ; START OF TRACK/GROUP SETS
```

```
                        ;
                        ;       DUMMY SDI CONTROL BLOCK OFFSETS
                        ;
000001          D.LIMT  =       1                       ; DUMMY SDI SEARCH LIMIT
000002          D.SCHR  =       2                       ; DUMMY POINTER TO SUBUNIT CHAR-5
                        ;
                        ;
                        ;       UNIT PARAMETER BITS (U.PARM(R5))
                        ;
100000          DROP    =       100000                  ; DROP BIT (SET IF UNIT OR SUBUNIT DROPPED)
040000          INITW   =       40000                   ; INITIAL WRITE (SET IF INITIAL WRITE IN PROG)
020000          RESEEK  =       20000                   ; IF 1, INDICATES THAT A SEEK IS NECESSARY
010000          DIREC   =       10000                   ; DIRECTION (SET IF SEQUENTIAL ACCESSES DECREASING)
004000          NEWSUB  =       4000                    ; SET IF SEQUENTIAL SEEKS MOVED TO NEW SUBUNIT
002000          SEKINP  =       2000                    ; SEEK IN PROGRESS - SET IF TRUE
001000          FTIME   =       1000                    ; FIRST TIME FLAG - SET FOR INIT CODE
000400          REVEC   =       400                     ; REVECTOR BIT (SET IF BLOCK REVECTORED)
000200          RBNSBN  =       200                     ; SEE IF WORKING ON RBN
000100          REDWRT  =       100                     ; REDWRT (READ OR WRITE IN PROGRESS SET IF WRITE)
000040          REVINP  =       40                      ; REVECTORING OPERATION IN PROGRESS
000020          DATERR  =       20                      ; DATA ERROR IF SET
000004          RETRY   =       4                       ; IF CLEAR, START RETRIES AT ZERO
000001          RCLB    =       1                       ; RECALIBRATION BIT (SET IF RECALIBRATE JUST DONE)
                        ;
                        ;       SUBUNIT PARAMETER BITS (S.PARM(R4))
                        ;
020000          DCYLS   =       20000                   ; DIAGNOSTIC CYLINDER FLAG (SET IF DBNS)
010000          ECCCHK  =       10000                   ; 1 IF ECC CORRECTION ALLOWED   MASTER BITS
004000          RONLY   =       4000                    ; READ ONLY (SET IF READ ONLY)
002000          WONLY   =       2000                    ; WRITE ONLY (SET IF WRITE ONLY)
001000          RTRIES  =       1000                    ; 1 IF RETRIES ALLOWED
000200          ONLYCL  =       200                     ; SET IF ONLY CYLINDERS SPECIFIED
                                                        ; USED DURING SETUP ONLY
000100          SEQ.EK  =       100                     ; SEQUENTIAL SEEK (START UP TESTING ONLY)
000040          BEUSED  =       40                      ; BEGIN/END SETS (USED IF SET)
000020          TRACKS  =       20                      ; TRACKS OR GROUPS (TRACK IF SET)
000010          WCHECK  =       10                      ; WRITE CHECK BIT (IF SET, WRITE CHECK WILL BE DONE)
                        ;                       WCHECK IS ALSO USED FOR THE UNIT PARAMETERS
000004          WCHKAL  =       4                       ; SET IF WRITE CHECK ALWAYS TO BE DONE
000002          DATCMP  =       2                       ; DATA COMPARE (SET IF DATA COMPARE TO BE DONE)
                        ;                       DATCMP IS ALSO USED FOR THE UNIT PARAMETERS
000001          DCMPAL  =       1                       ; SET IF DATA COMPARE ALWAYS TO BE DONE
```

MACRO DEFINITIONS

```
                        .SBTTL  MACRO DEFINITIONS
                        ;
                        ;       DIAGNOSTIC MACRO FOR TEST4 OVERLAYS
                        ;
                                .MACRO  DIAG$$
                                TST     $$DIAG+$DIAG$
                                BEQ     .+6
                                MOV     #60000,R0
                                MOV     R0,@$$DIAG+$DIAG$
                                BR      .+1
                        $DIAG$  =       $DIAG$ + 1
                                .ENDM
```

MACRO DEFINITIONS

```
                        ;       MESSAGE CONTROL TABLE MACRO
                        ;
                        .MACRO MSG CMDBUF,CMDSZ,RPLBUF,RPLSZ,SUCCOM
                        .WORD CMDBUF                    ;ADDRESS OF COMMAND
                        .WORD CMDSZ                     ;SIZE OF COMMAND IN BYTES
                        .WORD RPLBUF                    ;ADDRESS OF REPLY
                        .WORD RPLSZ                     ;SIZE OF REPLY IN WORDS
                        .IF NB NUMBER
                        .WORD SUCCOM                    ; SUCCESSFUL COMPLETION CODE
                        .ENDC
                        .ENDM
```

MACRO DEFINITIONS

```
      1                                 .MACRO  BCS     LAB..
      2                                                         BCC     .+2
      3                                                         BR      LAB..
      4                                 .ENDM
```

MACRO DEFINITIONS

```
                        ;       PUSH REGISTER MACRO
                        ;
                                .MACRO PUSH R9
                                .IRP X,<R9>
                                                        MOV X,-(SP)
                                .ENDR
                                .ENDM
                        ;
                        ;       POP REGISTER MACRO
                        ;
                                .MACRO POP R9
                                .IRP X,<R9>
                                                        MOV (SP)+,X
                                .ENDR
                                .ENDM
```

MACRO DEFINITIONS

```
                        ;ERROR MACROS
                        ;THESE MACROS ARE CALLED TO REPORT ERRORS TO THE HOST PROGRAM.
                        ;THE MACRO NAMES ARE : ERRSF, ERRDF, ERRHRD, ERRSFT. EACH RESULTS IN THE HOST
                        ;BEING REQUESTED TO REPORT THE ERROR.
                        ;ARGUMENTS:     1 (MS$) MESSAGE POINTER
                        ;               2 (P1$) PARAMETER #1
                        ;               3 (P2$) PARAMETER #2
                        ;               4 (P3$) PARAMETER #3
                        ;               5 (P4$) PARAMETER #4
                        ;               6 (P5$) PARAMETER #5
                        ;               7 (P6$) PARAMETER #6
                        ;               8 (P7$) PARAMETER #7
                        ;               9 (P8$) PARAMETER #8
                        ;
                        ;THE MESSAGE POINTER MUST POINT TO AN ADDRESS IN THE OVERLAY 'MS' IMMEDIATELY
                        ;FOLLOWING THE MAIN CODE. ANY ADDRESS MODE MAY BE USED (E.G. #MS1, @R2).
                        ;THE ADDRESS MUST CONTAIN AN ASCII FORMAT STRING TO DETERMINE THE MESSAGE
                        ;TO PRINT.
                        ;THE PARAMETER ARGUMENTS ARE OPTIONAL. THEY SHOULD BE SUPPLIED ONLY WHEN
                        ;THERE IS DATA TO BE PASSED TO THE HOST THAT WILL BE USED IN PRINTING THE
                        ;MESSAGE. THESE PARAMETER ARGUMENTS ARE THE ADDRESS OF DATA TO BE PASSED
                        ;USING ANY ADDRESSING MODE DESIRED.
                        ;ALL REGISTERS ARE RETURNED UNCHANGED. IT SHOULD BE NOTED THAT ARGUMENTS
                        ;CONTAINING SOMETHING OTHER THAN A REGISTER NAME (E.G. #100 OR MEMADR)
                        ;ASSEMBLE TO INSTRUCTIONS THAT SAVE AND RESTORE A REGISTER ON THE STACK.

                                .MACRO ERRSF MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ 0,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .ENDM

                                .MACRO ERRDF MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                 NARG ARGS$
                                .RADIX 10
                                ERROR$ 1,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .ENDM

                                .MACRO ERRHRD MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NLIST
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ 2,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .LIST
                                .ENDM

                                .MACRO ERRSFT MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ 3,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .ENDM
```

MACRO DEFINITIONS

```
                        ;THE FOLLOWING MACRO ACTUALLY PROCESSES THE ERROR CALL TO THE HOST PROGRAM

                        .MACRO ERROR$ ET$,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,ERRN$
                          .RADIX 8
                          PRMS=ARGS$-1
                          .IF LT,<PRMS>
                            .ERROR;NOT ENOUGH ARGUMENTS IN ERROR CALL
                          .ENDC
                          REGS$=-1
                          .IIF GE,<PRMS-8.>,PARGS. P8$
                          .IIF GE,<PRMS-7.>,PARGS. P7$
                          .IIF GE,<PRMS-6.>,PARGS. P6$
                          .IIF GE,<PRMS-5.>,PARGS. P5$
                          .IIF GE,<PRMS-4.>,PARGS. P4$
                          .IIF GE,<PRMS-3.>,PARGS. P3$
                          .IIF GE,<PRMS-2.>,PARGS. P2$
                          .IIF GE,<PRMS-1.>,PARGS. P1$
                          .IF GE REGS$
                            RSTR$ \REGS$
                          .ENDC
                          .RADIX 10
                          .LIST
                                                                        CALL RERROR     ;ERROR # ERRN$'.
                          .NLIST
                          .RADIX 8
                          .LIST
                                                                        .WORD <PRMS*2000>+<ET$*400>+ERRN
                                                                        .WORD MS$
                          .NLIST
                          ERRN=ERRN+1
                        .ENDM

                        .MACRO PARGS.,ADDR$
                          .NTYPE PTYPE$,ADDR$
                          .IF EQ,<PTYPE$&70>
                            .IIF EQ,<PTYPE$&7>-REGS$,RSTR$ \REGS$
                            .LIST
                                                                        MOV ADDR$,-(SP)
                            .NLIST
                          .IFF
                            .IF EQ,<PTYPE$&7>-1          ;PICK A REGISTER TO USE
                              REGU$=2                    ;SELECT R2 IF R1 IS USED IN PARAMETER FETCH
                            .IFF
                              REGU$=1                    ;OTHERWISE USE R1
                            .ENDC
                            .IF NE,<REGU$-REGS$>         ;IF REGISTER NOT ALREADY SAVED
                              .IF GE,REGS$
                                RSTR$ \REGS$             ;RESTORE CURRENT SAVED REGISTER
                              .ENDC
                              SAVR$ \REGU$               ;THEN SAVE SELECTED REGISTER
                            .ENDC
                            GETP$ \REGS$,ADDR$
                          .ENDC
                        .ENDM
```

MACRO DEFINITIONS

```
                        .MACRO SAVR$ REGN
                          .LIST
                                                        MOV R'REGN,SAVREG
                          .NLIST
                          REGS$-REGN
                        .ENDM

                        .MACRO RSTR$ REGN
                          .LIST
                                                        MOV SAVREG,R'REGN
                          .NLIST
                          REGS$=-1
                        .ENDM

                        .MACRO GETP$ REGN,ADDR$
                          .LIST
                                                        MOV ADDR$,R'REGN
                                                        MOV R'REGN,-(SP)
                          .NLIST
                        .ENDM
```

MACRO DEFINITIONS

```
                        ;
                        ;       PRIMARY ERROR REPORTING (TEST 4)
                        ;
                        .MACRO  SOFTER  NUM,ARGS
                        ERROR   #ERSOFT,NUM,<ARGS>
                                                        MOV     #ERRMES,R2
                                                        MOV     R2,OUT.RQ

                        .ENDM

                        .MACRO  REPSFT  SFTFLG,ECCFG
                        .IF     NB,SFTFLG
                                                        MOV     #1,OUT.02

                        .IFF
                                                        CLR     OUT.02

                        .ENDC
                        .IF     NB,ECCFG
                                                        MOV     #1,OUT.03

                        .IFF
                                                        CLR     OUT.03

                        .ENDC
                                                        PUSH    R0
                                                        MOV     #U.SNUM,R0
                                                        ADD     R5,R0
                                                        ADD     U.SUBU(R5),R0
                                                        MOV     (R0),OUT.01
                                                        MOV     #T4SOFT,R0
                                                        CALL    HOSTRQ
                                                        POP     R0

                        .ENDM

                        .MACRO  HARDER  NUM,ARGS
                        ERROR   #ERHARD,NUM,<ARGS>
                                                        MOV     #ERRMC,R2
                                                        MOV     R2,OUT.RQ

                        .ENDM

                        .MACRO  DEVFTL  NUM,ARGS
                        ERROR   #FTLDEV,NUM,<ARGS>
                                                        MOV     #ERRMC,R2
                                                        MOV     R2,OUT.RQ

                        .ENDM

                        .MACRO  SYSFTL  NUM,ARGS
                        ERROR   #FTLSYS,NUM,<ARGS>
                                                        MOV     #ERRMC,R2
                                                        MOV     R2,OUT.RQ

                        .ENDM

                        .MACRO  ERROR   TYPE,NUM,ARGS
                        .RADIX  10
                        NUMPTR  -       5
                        MOVMSG  #ER'NUM,4
                        .IF     NB,<ARGS>
                        .IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
                        NUMPTR  =       NUMPTR + 1
                        .ENDR
```

MACRO DEFINITIONS

```
                        .ENDC
                                                        MOV     #NUM,OUT.02
                                                        BIS     TYPE,OUT.02
                                                        MOV     #.,OUT.01

                        .RADIX
                        .ENDM

                        .MACRO  CERROR  STNUM,ARGS
                        .RADIX  10
                NUMPTR  =       STNUM
                        .IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
                NUMPTR  =       NUMPTR + 1
                        .ENDR
                        .RADIX
                        .ENDM

                        .MACRO  ERRORC  ARGS
                        .RADIX  10
                        .IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
                NUMPTR  =       NUMPTR + 1
                        .ENDR
                        .RADIX
                        .ENDM

                        .MACRO  MOVMSG  ARG,INDX

                        .IF     LT,INDX-10
                                                        MOV     ARG,OUT.0'INDX
                        .IFF
                                                        MOV     ARG,OUT.'INDX
                        .ENDC
                        .ENDM

                        .MACRO  ENDERR  POS
                        .IF     NB,POS
                NDERR   POS
                        .IFF
                NDERR   \NUMPTR
                        .ENDC
                        .ENDM

                        .MACRO  NDERR   POS
                        .IF     NE,POS
                                                        BIS     #POS,ERRPOS     ; SET THE POSITION
                        .IFF
                                                        CLR     ERRPOS          ; CLEAR THE POSITION
                        .ENDC
                        .ENDM

;
;       MESSAGE REPORTING MACRO
;
                        .MACRO  MSSG    NUM,ARGS
                        .RADIX  '0
                NUMPTR          3
                MOVMSG  #MS'NUM,2
```

MACRO DEFINITIONS

```
                        .IF     NB,<ARGS>
                        .IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
                        NUMPTR  -       NUMPTR + 1
                        .ENDR
                        .ENDC

                                                        PUSH    <R0,R1>
                                                        MOV     #U.SNUM,R1
                                                        ADD     R5,R1
                                                        ADD     U.SUSU(R5),R1
                                                        MOV     (R1),OUT.01
                                                        MOV     #MESSAG,R0
                                                        CALL    HOSTRQ
                                                        POP     <R1,R0>

                        .RADIX
                        .ENDM
                ;
                ;
                        .MACRO  MSSGE   NUM,ARGS
                        .RADIX  10
                        NUMPTR  -       3
                        MOVMSG  #'NUM,2
                        .IF     NB,<ARGS>
                        .IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
                        NUMPTR  -       NUMPTR + 1
                        .ENDR
                        .ENDC

                                                        PUSH    <R0,R1>
                                                        MOV     #MESSAG,R0
                                                        CALL    HOSTRQ
                                                        POP     <R1,R0>

                        .RADIX
                        .ENDM
```

MACRO DEFINITIONS

```
                        ;RETURN DRIVE STATUS MACRO WITH ERROR REPORTING

                                .MACRO  DSTAT,LAB$,E1,E2
                                .NLIST
                                .NLIST  MEB
                                .LIST   MF
                                .LIST
                                CALL    RDSTAT                  ; GET DRIVE STATUS
                                BIT     #10000,R1               ; SEE IF ANY ERRORS
                                BEQ     2$                      ; IF NO ERROR, BRANCH
                                BIT     #4000,R1                ; SEE IF XMIT ERROR
                                BEQ     1$                      ; IF SO, BRANCH
                                ERRHRD  E1                      ; REPORT INVALID STATUS ERROR
                                BR      LAB$                    ; BRANCH TO DONE
                        1$:     ERRHRD  E2                      ; REPORT XMIT ERROR
                                BR      LAB$                    ; BRANCH TO DONE
                        2$:
                                .NLIST
                                .NLIST  ME
                                .LIST   MEB
                                .LIST
                                .ENDM
```

MACRO DEFINITIONS

```
1                                       ;
2                                       ;       XOR THE CONTENTS OF TWO REGISTERS
3                                       ;
4                                               .MACRO  RXOR    REG1,REG2
5                                               MOV     REG2,-(SP)              ; SAVE REGISTER REG2
6                                               BIC     REG1,REG2               ; CLEAR COORESPONDING BITS IN REG2
7                                               BIC     (SP)+,REG1              ; CLEAR COORESPONDING BITS IN REG1
8                                               BIS     REG1,REG2               ; OR WHAT'S LEFT
9                                               .ENDM
```

MACRO DEFINITIONS

```
1                               ;       SDI INTERCHANGE WITH DRIVE WITH ERROR REPORTING
2
3                                       .MACRO  TALKX   ERRLAB,E1,E2
4                                       .NLIST
5                                       .NLIST  MEB
6                                       .LIST   MF
7                                       .LIST
8                                       CALL    TALKER                  ; INITIATE SDI INTERCHANGE
9                                       TST     R3                      ; SEE IF ERROR OCCURRED
10                                      BEQ     12$                     ; IF NOT, BRANCH
11                                      BPL     11$                     ; IF SO, BRANCH
12                                      ERRHRD  E1;SEND COMMAND ERROR
13                                      BR      ERRLAB
14                              11$:    ERRHRD  E2;RECEIVE COMMAND ERROR
15                                      BR      ERRLAB
16                              12$:
17                                      .NLIST
18                                      .NLIST  ME
19                                      .LIST   MEB
20                                      .LIST
21                                      .ENDM
```

START OF TEST CODE

```
                                .SBTTL  START OF TEST CODE
                                ;THE FOLLOWING IS FOR DEBUG PURPOSES ONLY. CAN BE NOP OR BREAKPOINT.
                                
000714  114007                          CLR R0                          ;CHANGE TO BREAKPOINT FOR DEBUG
                                
                                ;INITIALIZE STACK
                                
000715  104206  001357                  MOV #STACK,SP                   ;SET UP STACK POINTER
000717  001360                          BR      START                     BRANCH OVER SUPPORT CODE
```

```
                                        .SBTTL  RDSTAT - GET DRIVE'S REAL TIME DRIVE STATE
000720                                  RDSTAT:
                                        ;
                                        ;RETURN DRIVE STATUS
                                        ;STATUS RETURNED IN DM REGISTER 1
                                        ;
000720                                          PUSH    <R3,R0>                 ; SAVE R3 AND R0
000720  100463                                                                         MOV R3,-(SP)
000721  100467                                                                         MOV R0,-(SP)
000722  104203  000003                         MOV     #3,R3                   ; ALLOW ONLY 3 ERRORS
000724  060007                          STATLP: XFC     STATUS                  ; GET DRIVE'S STATUS
000725  103201  014000                          BIC     #14000,R1               ; CLEAR ERROR PASSING BITS
000727  102201  000400                          BIT     #XMTERR,R1              ; CHECK XMIT ERRORS
000731  010737                                  BEQ     STATOK                  ; IF NO ERRORS, BRANCH
000732  117403                                  DEC     R3                      ; DECREMENT TRANSMIT ERROR COUNT
000733  050724                                  BNE     STATLP                  ; IF ERROR COUNT INCOMPLETE, BRANCH
000734  104201  010000                          MOV     #10000,R1               ; FLAG AS TRANSMIT ERROR
000736  000746                                  BR      STATEX                  ; BRANCH
000737  102201  000004                  STATOK: BIT     #RCVERR,R1              ; RECIEVER ERRORS
000741  050746                                  BNE     STATEX                  ; IF VALID, BRANCH
000742  117403                                  DEC     R3                      ; DECREMENT ERROR COUNT
000743  050724                                  BNE     STATLP                  ; IF ERROR COUNT NON-ZERO, BRANCH
000744  104201  014000                          MOV     #14000,R1               ; FLAG AS INVALID STATUS ERROR
000746                                  STATEX: POP     <R0,R3>                 ; RESTORE R0, R3
000746  104267                                                                         MOV (SP)+,R0
000747  104263                                                                         MOV (SP)+,R3
000750  000000                                  RETURN                          ; RETURN TO CALLING MODULE
```

```
                                .SBTTL  HOSTRQ - HOST REQUEST - REPORT ERRORS, MEGABYTES TRANSFERRED, ETC.
000751                          HOSTRQ:
                                ;
                                ;SEND REQUEST BUFFER TO HOST AND WAIT FOR RESPONSE.
                                ;CLEAR ARGUMENT AREA OF OUT BUFFER IN PREPARATION
                                ;FOR NEXT HOSTRQ CALL.
                                ;
                                ;INPUTS:
                                ;       R0 - HOST REQUEST NUMBER
                                ;       OUT BUFFER LOADED WITH DATA

000751                          PUSH <R0,R1,R2>
000751  100467                                                                  MOV R0,-(SP)
000752  100461                                                                  MOV R1,-(SP)
000753  100462                                                                  MOV R2,-(SP)
000754  104070  001006          MOV R0,OUT.RQ                   ;STORE REQUEST NUMBER IN BUFFER
000756  104207  001006  SNDAGN: MOV #OUT.RQ,R0                 ;SEND BUFFER TO HOST
000760  104201  000036          MOV #BUFSIZ,R1
000762  060016                  XFC MRD
000763  115001                  TST R1                          ;CHECK FOR ERROR
000764  050756                  BNE SNDAGN                      ;IF ERROR, TRY AGAIN
000765  104207  001044          MOV #IN.RQ,R0                   ;WAIT FOR RESPONSE FROM HOST
000767  104201  000036          MOV #BUFSIZ,R1
000771  060017                  XFC MWR
000772  104207  001007          MOV #OUT.01,R0                  ;CLEAR ARGUMENT WORDS IN BUFFER
000774  104201  000034          MOV #OUT.29-OUT.01,R1
000776  114002                  CLR R2
000777  100272  CLRBUF: MOV R2,(R0)+
001000  117401                  DEC R1
001001  030777                  BPL CLRBUF
001002                          POP <R2,R1,R0>
001002  104262                                                                  MOV (SP)+,R2
001003  104261                                                                  MOV (SP)+,R1
001004  104267                                                                  MOV (SP)+,R0
001005  000000                  RETURN
```

```
                                ;STORAGE AREA FOR MAINTENANCE WRITE AND READ BUFFERS
                                ;
                                ;OUT BUFFER - DATA TO SEND TO HOST
                                ;
001006  000000                  OUT.RQ: .WORD 0                 ;HOST REQUEST CODE
001007  000000                  OUT.01: .WORD 0                 ;DATA ARGUMENT  1
001010  000000                  OUT.02: .WORD 0                 ;DATA ARGUMENT  2
001011  000000                  OUT.03: .WORD 0                 ;DATA ARGUMENT  3
001012  000000                  OUT.04: .WORD 0                 ;DATA ARGUMENT  4
001013  000000                  OUT.05: .WORD 0                 ;DATA ARGUMENT  5
001014  000000                  OUT.06: .WORD 0                 ;DATA ARGUMENT  6
001015  000000                  OUT.07: .WORD 0                 ;DATA ARGUMENT  7
001016  000000                  OUT.08: .WORD 0                 ;DATA ARGUMENT  8
001017  000000                  OUT.09: .WORD 0                 ;DATA ARGUMENT  9
001020  000000                  OUT.10: .WORD 0                 ;DATA ARGUMENT 10
001021  000000                  OUT.11: .WORD 0                 ;DATA ARGUMENT 11
001022  000000                  OUT.12: .WORD 0                 ;DATA ARGUMENT 12
001023  000000                  OUT.13: .WORD 0                 ;DATA ARGUMENT 13
001024  000000                  OUT.14: .WORD 0                 ;DATA ARGUMENT 14
001025  000000                  OUT.15: .WORD 0                 ;DATA ARGUMENT 15
001026  000000                  OUT.16: .WORD 0                 ;DATA ARGUMENT 16
001027  000000                  OUT.17: .WORD 0                 ;DATA ARGUMENT 17
001030  000000                  OUT.18: .WORD 0                 ;DATA ARGUMENT 18
001031  000000                  OUT.19: .WORD 0                 ;DATA ARGUMENT 19
001032  000000                  OUT.20: .WORD 0                 ;DATA ARGUMENT 20
001033  000000                  OUT.21: .WORD 0                 ;DATA ARGUMENT 21
001034  000000                  OUT.22: .WORD 0                 ;DATA ARGUMENT 22
001035  000000                  OUT.23: .WORD 0                 ;DATA ARGUMENT 23
001036  000000                  OUT.24: .WORD 0                 ;DATA ARGUMENT 24
001037  000000                  OUT.25: .WORD 0                 ;DATA ARGUMENT 25
001040  000000                  OUT.26: .WORD 0                 ;DATA ARGUMENT 26
001041  000000                  OUT.27: .WORD 0                 ;DATA ARGUMENT 27
001042  000000                  OUT.28: .WORD 0                 ;DATA ARGUMENT 28
001043  000000                  OUT.29: .WORD 0                 ;DATA ARGUMENT 29
                                ;
                                ;IN BUFFER - DATA RECEIVED FROM HOST
                                ;
001044  000000                  IN.RQ:  .WORD 0                 ;HOST REQUEST CODE (ECHO)
001045  000000                  IN.01:  .WORD 0                 ;DATA ARGUMENT  1
001046  000000                  IN.02:  .WORD 0                 ;DATA ARGUMENT  2
001047  000000                  IN.03:  .WORD 0                 ;DATA ARGUMENT  3
001050  000000                  IN.04:  .WORD 0                 ;DATA ARGUMENT  4
001051  000000                  IN.05:  .WORD 0                 ;DATA ARGUMENT  5
001052  000000                  IN.06:  .WORD 0                 ;DATA ARGUMENT  6
001053  000000                  IN.07:  .WORD 0                 ;DATA ARGUMENT  7
001054  000000                  IN.08:  .WORD 0                 ;DATA ARGUMENT  8
001055  000000                  IN.09:  .WORD 0                 ;DATA ARGUMENT  9
001056  000000                  IN.10:  .WORD 0                 ;DATA ARGUMENT 10
001057  000000                  IN.11:  .WORD 0                 ;DATA ARGUMENT 11
001060  000000                  IN.12:  .WORD 0                 ;DATA ARGUMENT 12
001061  000000                  IN.13:  .WORD 0                 ;DATA ARGUMENT 13
001062  000000                  IN.14:  .WORD 0                 ;DATA ARGUMENT 14
001063  000000                  IN.15:  .WORD 0                 ;DATA ARGUMENT 15
001064  000000                  IN.16:  .WORD 0                 ;DATA ARGUMENT 16
001065  000000                  IN.17:  .WORD 0                 ;DATA ARGUMENT 17
001066  000000                  IN.18:  .WORD 0                 ;DATA ARGUMENT 18
001067  000000                  IN.19:  .WORD 0                 ;DATA ARGUMENT 19
```

```
     58 001070  000000                IN.20:  .WORD 0                 ;DATA ARGUMENT 20
     59 001071  000000                IN.21:  .WORD 0                 ;DATA ARGUMENT 21
     60 001072  000000                IN.22:  .WORD 0                 ;DATA ARGUMENT 22
     61 001073  000000                IN.23:  .WORD 0                 ;DATA ARGUMENT 23
     62 001074  000000                IN.24:  .WORD 0                 ;DATA ARGUMENT 24
     63 001075  000000                IN.25:  .WORD 0                 ;DATA ARGUMENT 25
     64 001076  000000                IN.26:  .WORD 0                 ;DATA ARGUMENT 26
     65 001077  000000                IN.27:  .WORD 0                 ;DATA ARGUMENT 27
     66 001100  000000                IN.28:  .WORD 0                 ;DATA ARGUMENT 28
     67 001101  000000                IN.29:  .WORD 0                 ;DATA ARGUMENT 29
     68         000036                BUFSIZ  =       . - IN.RQ       ;SIZE OF BUFFER
```

```
                                        .IF     NE,TEST4
                                        .NLIST
                                        .IFF
                        ;
                        ;       TALKER SENDS AND RECEIVES AN SDI INTERCHANGE
                        ;
                        ;INPUTS:
                        ;       R2 - SDI INTERCONNECT
                        ;       R3 - POINTER TO COMMAND TABLE CONTAINING APPROPRIATE COMMAND
                        ;       SDILTO/SDISTO SDI LONG AND SHORT TIMEOUTS, RESPECTIVELY
                        ;OUTPUTS:
                        ;       R0 - RETURN OP CODE FROM UNIT
                        ;       R3 - ERROR CODE - 0 = NO ERROR, 1 = RECEIVE ERROR, 100001 = SEND ERROR
                        ;
001102                  TALKER: PUSH    <R1,R4>         ; SAVE REGISTERS
001102  100461                                                                 MOV R1,-(SP)
001103  100464                                                                 MOV R4,-(SP)
001104  104237                  MOV     (R3)+,R0        ; SET ADR OF SDI COMMAND BUFFER
001105  104231                  MOV     (R3)+,R1        ; SET BUFFER LENGTH
001106  060004                  XFC     SEND            ; SEND COMMAND
001107  115001                  TST     R1              ; DID UNIT ACCEPT COMMAND
001110  011114                  BEQ     TALK1A          ; IF SO, BRANCH
001111  104203  100001          MOV     #100001,R3      ; FLAG AS SEND ERROR
001113  001145                  BR      TALK2B          ; BRANCH TO EXIT
001114  106203  001360  TALK1A: CMP     #LONG,R3        ; SEE IF LONG TIMEOUT TO BE USED
001116  071122                  BMI     1$              ; IF SO, BRANCH
001117  104304  001150          MOV     SDISTO,R4       ; SET UP SHORT TIMEOUT
001121  001124                  BR      TALK1B          ; BRANCH
001122  104304  001151  1$:     MOV     SDILTO,R4       ; SET UP LONG TIMEOUT
001124  104137          TALK1B: MOV     (R3),R0         ; SET DATA BUFFER ADDRESS
001125  104631  000001          MOV     1(R3),R1        ; SET BUFFER LENGTH
001127  060005                  XFC     RCV             ; SEND RECEIVE SDI COMMAND
001130  115001                  TST     R1              ; DID ERROR OCCUR
001131  011144                  BEQ     TALK2A          ; IF NOT, BRANCH
001132  106201  000001          CMP     #1,R1           ; SEE IF TIMEOUT
001134  011137                  BEQ     1$              ; IF SO, BRANCH
001135  104013                  MOV     R1,R3           ; MOVE ERROR TYPE TO R3 FOR REPORTING
001136  001145                  BR      TALK2B          ; EXIT
001137  117404  1$:             DEC     R4              ; DECREMENT TIMEOUT VALUE
001140  051124                  BNE     TALK1B          ; IF NOT TIMEOUT, BRANCH
001141  104203  000001          MOV     #1,R3           ; FLAG AS RECIEVE ERROR
001143  001145                  BR      TALK2B          ; BRANCH TO EXIT
001144  114003  TALK2A:         CLR     R3              ; FLAG AS NO ERRORS
001145  TALK2B:                 POP     <R4,R1>         ; RESTORE R4, R1
001145  104264                                                                 MOV (SP)+,R4
001146  104261                                                                 MOV (SP)+,R1
001147  000000                  RETURN
                        ;
001150  000012  SDISTO: .WORD   10.             ; SDI SHORT TIMEOUT
001151  000024  SDILTO: .WORD   20.             ; SDI LONG TIMEOUT
```

```
                                ;MREAD
                                ;
                                ;READ ONE WORD FROM UNIBUS MEMORY
                                ;INPUTS:
                                ;       R5 - ADDRESS OF WORD TO READ (LOW 16 BITS)
                                ;       R4 -    ''           (HIGH 2 BITS)
                                ;OUTPUTS:
                                ;       R0 - DATA READ
                                ;
001152  100462                  MREAD:  PUSH <R2,R3>                    ;SAVE SOME REGISTERS
                                                                                MOV R2,-(SP)
001153  100463                                                                 MOV R3,-(SP)
001154  104057                          MOV R5,R0                       ;PUT UNIBUS ADDRESS IN R0 AND R1
001155  104041                          MOV R4,R1
001156  104202  000001                  MOV #1,R2                       ;TRANSFER ONE WORD
001160  104203  001210                  MOV #UDATA,R3                   ;LOCATION TO PUT DATA
001162  060013                          XFC UREAD                       ;DO THE READ
001163  104307  001210                  MOV UDATA,R0                    ;GET DATA READ
001165                                  POP <R3,R2>                     ;RESTORE OTHER REGISTERS
001165  104263                                                                  MOV (SP)+,R3
001166  104262                                                                  MOV (SP)+,R2
001167  000000                          RETURN
```

```
                                ;MWRITE
                                ;
                                ;WRITE ONE WORD TO UNIBUS MEMORY
                                ;INPUTS:
                                ;       R5 - ADDRESS OF WORD TO WRITE (LOW 16 BITS)
                                ;       R4 -    ''           (HIGH 2 BITS)
                                ;       R0 - DATA TO WRITE
                                ;OUTPUTS:

001170                          MWRITE: PUSH <R0,R2,R3>                 ;SAVE SOME REGISTERS
001170  100467                                                                  MOV R0,-(SP)
001171  100462                                                                  MOV R2,-(SP)
001172  100463                                                                  MOV R3,-(SP)
001173  104070  001210                  MOV R0,UDATA                    ;PUT DATA TO BE WRITTEN INTO BUFFER
001175  104057                          MOV R5,R0                       ;PUT UNIBUS ADDRESS IN R0 AND R1
001176  104041                          MOV R4,R1
001177  104202  000001                  MOV #1,R2                       ;TRANSFER ONE WORD
001201  104203  001210                  MOV #UDATA,R3                   ;ADDRESS OF DATA WORD
001203  060014                          XFC UWRITE                      ;DO THE WRITE
001204                                  POP <R3,R2,R0>                  ;RESTORE THE REGISTERS
001204  104263                                                                  MOV (SP)+,R3
001205  104262                                                                  MOV (SP)+,R2
001206  104267                                                                  MOV (SP)+,R0
001207  000000                          RETURN

001210  000000                  UDATA:  .WORD 0                         ;DATA BUFFER FOR TRANSFERS TO AND FROM
                                                                        ;UNIBUS MEMORY
```

```
      1                                 ;XOR
      2                                 ;
      3                                 ;PERFORM XOR LOGIC FUNCTION ON TWO REGISTERS
      4                                 ;INPUTS:
      5                                 ;       R1, R2 - DATA TO BE XOR'ED
      6                                 ;OUTPUTS:
      7                                 ;       R1 - UNCHANGED
      8                                 ;       R2 - XOR OF TWO INPUTS
      9
     10 001211                          XOR:    PUSH R3                         ,SAVE R3
        001211  100463                                                          MOV R3,-(SP)
     11 001212  104013                          MOV R1,R3
     12 001213  103023                          BIC R2,R3
     13 001214  103012                          BIC R1,R2
     14 001215  101032                          BIS R3,R2
     15 001216                                  POP R3                          ;RESTORE R3
        001216  104263                                                          MOV (SP)+,R3
     16 001217  115002                          TST R2                          ;SET CONDITION CODES
     17 001220  000000                          RETURN
```

```
                                TO:
                                ;
                                ;       CALCULATE THE TIMEOUT IN 9SEC INTERVALS (SDI RECEIVE XFC TAKES
                                ;       9 SEC)
                                ;
001221  104201  000001                  MOV     #1,R1           ; SET UP LOG2 SHIFTER
001223  110201                  1$:     ROL     R1              ; DOUBLE THE TIMEOUT VALUE
001224  103201  000001                  BIC     #1,R1           ; CLEAR THE LOW BIT
001226  117407                          DEC     R0              ; DECREMENT COUNT
001227  051223                          BNE     1$              ; IF COUNT INCOMPLETE, BRANCH
001230  114007                          CLR     R0              ; CLEAR 9SEC COUNT
001231  115407                  2$:     INC     R0              ; INCREMENT 9 SEC COUNT
001232  107201  000011                  SUB     #9.,R1          ; SUBTRACT 9 SEC FROM TIMEOUT
001234  031231                          BPL     2$              ; IF MORE TIME TO GO, BRANCH
001235  000000                          RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                        ;RERROR
                        ;
                        ;REPORT ERROR TO HOST PROGRAM
                        ;THIS ROUTINE IS CALLED BY THE ERROR MACROS:
                        ;ERRSF, ERRDF, ERRHRD AND ERRSFT

001236                  RERROR: PUSH R0                         ;SAVE ONE REGISTER
001236  100467                                                          MOV R0,-(SP)
001237  104067                  MOV SP,R0                       ;GET STACK POINTER
001240                          PUSH <R1,R2,R3,R4>              ;SAVE MORE REGISTERS
001240  100461                                                          MOV R1,-(SP)
001241  100462                                                          MOV R2,-(SP)
001242  100463                                                          MOV R3,-(SP)
001243  100464                                                          MOV R4,-(SP)
001244  115407                  INC R0                          ;CHANGE SAVED STACK POINTER TO POINT TO
                                                                ; ADDRESS OF LOCATION AFTER CALL
001245  104271                  MOV (R0)+,R1                    ;GET RETURN PC
001246  104202  001007          MOV #OUT.01,R2                  ;GET ADDRESS OF WHERE TO PUT DATA
001250  117401                  DEC R1                          ;REDUCE TO PC OF ERROR CALL
001251  100221                  MOV R1,(R2)+                    ;PUT PC IN OUT BUFFER
001252  115401                  INC R1                          ;GET BACK TO RETURN PC
001253  104113                  MOV (R1),R3                     ;GET ERROR NUMBER AND TYPE
001254  103203  176000          BIC #^C001777,R3                ;CLEAR OTHER BITS
001256  100223                  MOV R3,(R2)+                    ;PUT IN BUFFER
001257  104303  001315          MOV LUNIT,R3                    ;PUT UNIT NUMBER IN BUFFER
001261  100223                  MOV R3,(R2)+
001262  104214                  MOV (R1)+,R4                    ;GET COUNT OF PARAMETERS
001263  104213                  MOV (R1)+,R3                    ;GET MESSAGE POINTER
001264  100223                  MOV R3,(2)+                     ;PUT IN OUT BUFFER
                                                                ;R1 IS NOW POINTING TO INSTRUCTION AFTER ERROR CALL
001265  110704                  SWAB R4
001266  110604                  ROR R4                          ;EXTRACT NUMBER OF PARAMETERS FROM ERROR MACRO
001267  110604                  ROR R4
001270  103204  177700          BIC #177700,R4
001272  104040  001313          MOV R4,SPADJU+1                 ;SAVE FOR LATER ADJUSTMENT OF STACK POINTER
001274  011301                  BEQ RERRCA                      ;BRANCH IF NO PARAMETERS
001275  104273          RERRPA: MOV (R0)+,R3                    ;GET PARAMETER
001276  100223                  MOV R3,(R2)+                    ;STORE PARAMETER IN OUT BUFFER
001277  117404                  DEC R4                          ;COUNT THE PARAMETERS
001300  051275                  BNE RERRPA                      ;GET NEXT IF MORE
001301  100471          RERRCA: MOV R1,-(R0)                    ;PUT RETURN ADDRESS ON STACK
001302  104207  000013          MOV #ERRMES,R0                  ;SEND ERROR PACKET TO HOST PROGRAM
001304  020751                  CALL HOSTRQ
001305                          POP <R4,R3,R2,R1,R0>            ;RESTORE REGISTERS
001305  104264                                                          MOV (SP)+,R4
001306  104263                                                          MOV (SP)+,R3
001307  104262                                                          MOV (SP)+,R2
001310  104261                                                          MOV (SP)+,R1
001311  104267                                                          MOV (SP)+,R0
001312  105206  000000  SPADJU: ADD #0,SP                       ;ADJUST STACK OVER PARAMETERS
                                                                ;VALUE CHANGED ABOVE
001314  000000                  RETURN

001315  177777  LUNIT:  .WORD -1                               ;LOGICAL UNIT NUMBER (-1 FOR NOT AVAILABLE)

001316  000000  SAVREG: .WORD 0                               ;STORAGE FOR REGISTER AT CALL TIME
                        .IFT
```

```
48                                      .LIST
49                                      .ENDC
```

```
                                ;STACK AREA
                                
001317  123456                        .WORD 123456            ;END MARKER FOR STACK
001320                              .BLKW 31.               ;STACK
001357  123456                STACK:  .WORD 123456            ;MARKER FOR STACK UNDERFLOW
```

```
                                .SBTTL  FREE MEMORY CHECK

000001                                  FRRN=1.                         ;START ERROR NUMBERS FROM 1.

                                ;ASK HOST WHERE FREE MEMORY IS AND TO FILL IT WITH AN ADDRESS PATTERN

001360                          LONG:                                   ;UNUSED LABEL THAT MUST BE DEFINED
001360  104207  000000          START:  MOV #T1MSIZ,R0                  ;GET REQUEST NUMBER
001362  020751                          CALL HOSTRQ                     ;ASK HOST
001363  104207  001045                  MOV #IN.01,R0                   ;TRANSFER DATA FROM HOST
001365  104201  002057                  MOV #FWADR,R1                   ; TO STORAGE
001367  104202  000010                  MOV #8.,R2
001371  104273                  FMEMFL: MOV (R0)+,R3
001372  100213                          MOV R3,(R1)+
001373  117402                          DEC R2
001374  051371                          BNE FMEMFL

                                ;READ ALL OF SPECIFIED MEMORY AND CHECK FOR DATA SAME AS ADDRESS

001375  104305  002057                  MOV FWADR,R5                    ;GET STARTING ADDRESS
001377  104304  002060                  MOV FWADRH,R4
001401  021152                  ACHK1:  CALL MREAD                      ;READ FROM UNIBUS MEMORY

                                ;COMPARE DATA READ WITH EXPECTED

001402  106075                  ACHK2:  CMP R0,R5                       ;COMPARE DATA READ WITH ADDRESS
001403  011413                          BEQ ACHK3                       ;BRANCH IF A MATCH
001404                                  ERRHRD MS3,R5,R4,R5,R0 ;UNIBUS ADDRESSING ERROR - INCORRECT DATA READ
001404  100467                                                                  MOV R0,-(SP)
001405  100465                                                                  MOV R5,-(SP)
001406  100464                                                                  MOV R4,-(SP)
001407  100465                                                                  MOV R5,-(SP)
001410  021236                                                                  CALL RERROR     ;ERROR # 1.
001411  011001                                                                  .WORD <PRMS*2000>+<2*400>+ERRN
001412  000123                                                                  .WORD MS3

                                ;INCREMENT TO NEXT LOCATION UNTIL ALL TESTED

001413  106305  002061          ACHK3:  CMP LWADR,R5                    ;CHECK IF AT LAST ADDRESS
001415  051421                          BNE ACHK4
001416  106304  002062                  CMP LWADRH,R4
001420  011426                          BEQ BCHK                        ;GO TO NEXT TEST IF SO
001421  105205  000002          ACHK4:  ADD #2,R5                       ;INCREMENT TEST ADDRESS
001423  041401                          BCC ACHK1
001424  115404                          INC R4                          ;CARRY TO HIGH BITS
001425  001401                          BR ACHK1                        ;LOOP IF STILL IN SPECIFIED MEMORY
```

```
                                ;CHECK EACH ADDRESS LINE BY LOOKING AT TWO LOCATIONS WITH ONLY THAT
                                ;ADDRESS LINE DIFFERENT AND VERIFYING TWO LOCATIONS ARE ACTUALLY ACCESSED
                                
001426  104307  002060  BCHK:   MOV FWADRH,R0          ;LOAD TEST ADDRESS HIGH BITS
001430  104070  002070          MOV R0,TADRH
001432  104203  000002          MOV #2,R3             ;GET STARTING ADDRESS BIT TO TEST
001434  104032          BCHK1:  MOV R3,R2             ;GET ADDRESS BIT TO TEST
001435  104301  002057          MOV FWADR,R1          ;GET FIRST ADDRESS OF FREE MEMORY
001437  021211                  CALL XOR              ;CHANGE JUST THE ONE BIT IN ADDRESS
001440  104020  002067          MOV R2,TADR           ;STORE TEST ADDRESS
001442  021742                  CALL BCHKM            ;TEST THE MEMORY ADDRESS
001443  105033                  ADD R3,R3             ;GET NEW ADDRESS BIT TO TEST
001444  051434                  BNE BCHK1             ;LOOP IF STILL AN ADDRESS BIT TO TEST

001445  104307  002057          MOV FWADR,R0          ;NOW MOVE TO HIGH TWO BITS
001447  104070  002067          MOV R0,TADR           ;COPY LOW BITS TO TEST ADDRESS
001451  104203  000001          MOV #1,R3             ;START BIT
001453  104032          BCHK2:  MOV R3,R2             ;GET TEST BIT
001454  104301  002060          MOV FWADRH,R1         ;GET FIRST ADDRESS
001456  021211                  CALL XOR              ;CHANGE ONLY THE ONE BIT
001457  104020  002070          MOV R2,TADRH          ;STORE AS TEST ADDRESS
001461  021742                  CALL BCHKM            ;TEST THE MEMORY ADDRESS
001462  105033                  ADD R3,R3             ;CHANGE TEST BIT
001463  102203  000004          BIT #4,R3             ;CHECK IF PAST TWO ADDRESS BITS
001465  011453                  BEQ BCHK2             ;REPEAT FOR OTHER BIT
```

```
                                ;TRANSFER LARGE BUFFERS OF DATA TO AND FROM THE HOST MEMORY.
                                ;
                                ;THE HOST MEMORY WILL BE DIVIDED INTO SEVERAL BUFFERS. ALL BUFFERS WILL
                                ;BE WRITTEN WITH PATTERN 0 THEN READ AND THE DATA COMPARED TO PATTERN 0.
                                ;EACH BUFFER WILL THEN BE WRITTEN TO PATTERN 1 WITH A READ OF ALL BUFFERS
                                ;AFTER EACH WRITE. WHEN ALL BUFFERS HAVE BEEN WRITTEN WITH PATTERN 1, THEN
                                ;THE SEQUENCE REPEATS BY WRITING EACH BUFFER WITH PATTERN 2 AND THEN
                                ;WITH PATTERN 3.
                                ;
                                ;DATA PATTERNS: (EACH IS A REPETITION OF THREE WORDS)
                                ;0 - 111111     1 - 177400      2 - 155555      3 - 000377
                                ;    044444         007760          133333          170017
                                ;    022222         000377          066666          177400

                                ;BREAK THE HOST MEMORY INTO BUFFERS

001466  104202  010000      CCHK:   MOV #10000,R2           ;COMPUTE SIZE OF DATA
001470  107202  002103              SUB #FREE,R2            ;  BUFFER IN M MEMORY
001472  105022                      ADD R2,R2               ;CHANGE TO BYTE COUNT
001473  104205  002103              MOV #FREE,R5            ;GET ADDRESS OF FIRST TABLE ENTRY
001475  104304  002057              MOV FWADR,R4            ;GET ADDRESS OF FIRST
001477  104303  002060              MOV FWADRH,R3           ;  BUFFER IN HOST
001501  114001                      CLR R1                  ;INIT COUNT OF BUFFERS
001502  100254              2$:     MOV R4,(R5)+            ;STORE HOST BUFFER ADDRESS
001503  100253                      MOV R3,(R5)+            ;  IN TABE ENTRY
001504  107202  000004              SUB #4,R2               ;REDUCE BUFFER SIZE BY TABLE ENTRY
001506  105024                      ADD R2,R4               ;INCREADE HOST ADDRESS TO
001507  041511                      BCC 1$                  ;  NEXT BUFFER
001510  115403                      INC R3
001511  106303  002062      1$:     CMP LWADRH,R3           ;CHECK IF BUFFER LARGER
001513  071523                      BMI 4$                  ;  THEN HOST MEMORY REMAINING
001514  051521                      BNE 3$
001515  106304  002061              CMP LWADR,R4
001517  041521                      BCC 3$
001520  001523                      BR 4$
001521  115401              3$:     INC R1                  ;COUNT THE TABLE ENTRY
001522  001502                      BR 2$                   ;GO BACK AND DO AGAIN

001523  104050  002074      4$:     MOV R5,DATBUF           ;SAVE ADDRESS OF DATA BUFFER
001525  105207  000000              ADD #0,R0               ;CLEAR CARRY
001527  110602                      ROR R2                  ;SAVE SIZE OF DATA BUFFER
001530  104020  002075              MOV R2,SIZBUF           ;  IN WORDS
001532  104010  002071              MOV R1,BUFCNT           ;SAVE COUNT OF BUFFERS
```

```
                                ;WRITE ALL BUFFERS TO PATTERN 0
                                
001534  114007                          CLR R0                  ;LOAD PATTERN 0
001535  104070  002073                  MOV R0,CURPAT           ;  IN SAVE WORD
001537  104070  002072          10$:    MOV R0,CURBUF           ;SELECT BUFFER 0
001541  021601                          CALL WRITE              ;WRITE THE BUFFER
001542  104307  002072                  MOV CURBUF,R0           ;INCREMENT TO NEXT BUFFER
001544  115407                          INC R0
001545  106307  002071                  CMP BUFCNT,R0           ;WRITE ANOTHER IF
001547  051537                          BNE 10$                 ;  NOT AT LAST
001550  021651                          CALL READ               ;READ ALL HOST BUFFERS
                                
001551  104307  002073          12$:    MOV CURPAT,R0           ;INCREMENT PATTERN NUMBER
001553  115407                          INC R0
001554  106207  000004                  CMP #4,R0               ;EXIT SUBTEST IF NO
001556  011575                          BEQ DONECD              ;  NO PATTERNS REMAINING
001557  104070  002073                  MOV R0,CURPAT
                                
001561  114007                          CLR R0                  ;POINT TO BUFFER 0
001562  104070  002072          14$:    MOV R0,CURBUF
001564  021601                          CALL WRITE              ;WRITE A BUFFER
001565  021651                          CALL READ               ;READ ALL HOST BUFFERS
001566  104307  002072                  MOV CURBUF,R0           ;INCREMENT TO NEXT BUFFER
001570  115407                          INC R0
001571  106307  002071                  CMP BUFCNT,R0           ;CHECK IF AT LAST
001573  051562                          BNE 14$                 ;  WRITE NEXT BUFFER
001574  001551                          BR 12$                  ;  WRITE NEXT PATTERN
```

FREE MEMORY CHECK

```
      1 001575  104207  000016          DONECD: MOV #DONE,R0            ;END OF PROGRAM, TELL PDP-11
      2 001577  020751                          CALL HOSTRQ
      3 001600  001575                          BR DONECD               ;KEEP SENDING IF RETURNED
```

```
                                ;FILL A DATA BUFFER BEGINNING AT ADDRESS IN DATBUF AND WHOSE SIZE
                                ;IS IN SIZBUF WITH A DATA PATTERN SPECIFIED BY CONTENTS OF CURPAT.
                                ;WRITE THIS BUFFER TO HOST STARTING AT ADDRESS IN TWO WORDS POINTED
                                ;TO BY CURBUF. THEN PLACE THE PATTERN NUMBER IN THE SECOND WORD OF THE
                                ;HOST ADDRESS

                                ;FILL BUFFER WITH DATA PATTERN

001601  104307  002073    WRITE:  MOV CURPAT,R0           ;GET PATTERN NUMBER
001603  105077                    ADD R0,R0               ;  TIMES FOUR
001604  105077                    ADD R0,R0
001605  104072                    MOV R0,R2               ;SAVE FOR ADDING TO TABLE ENTRY
001606  105207  001723            ADD #PAT0,R0            ;ADD START OF PATTERN TABLES
001610  104273                    MOV (R0)+,R3            ;GET PATTERN WORDS
001611  104274                    MOV (R0)+,R4
001612  104275                    MOV (R0)+,R5
001613  104307  002074            MOV DATBUF,R0           ;GET ADDRESS OF BUFFER
001615  104301  002075            MOV SIZBUF,R1           ;GET SIZE OF BUFFER

001617  100273    1$:             MOV R3,(R0)+            ;LOAD ONE WORD
001620  117401                    DEC R1                  ;COUNT THE WORDS
001621  011630                    BEQ 2$
001622  100274                    MOV R4,(R0)+            ;LOAD ONE WORD
001623  117401                    DEC R1                  ;COUNT THE WORDS
001624  011630                    BEQ 2$
001625  100275                    MOV R5,(R0)+            ;LOAD ONE WORD
001626  117401                    DEC R1                  ;COUNT THE WORDS
001627  051617                    BNE 1$

                                ;WRITE THE BUFFER TO HOST MEMORY

001630  104305  002072    2$:     MOV CURBUF,R5           ;GET POINTER TO HOST ADDRESS
001632  105055                    ADD R5,R5               ;  COUNT TIMES TWO
001633  105205  002103            ADD #FREE,R5            ;  PLUS START ADDRESS
001635  104257                    MOV (R5)+,R0            ;GET LOW ADDRESS BITS
001636  104151                    MOV (R5),R1             ;GET HIGH ADDRESS BITS
001637  103201  177774            BIC #177774,R1          ;CLEAR CURRENT PATTERN
001641  101012                    BIS R1,R2               ;SET IN NEW PATTERN
001642  100152                    MOV R2,(R5)             ;STORE IN TABLE ENTRY
001643  104302  002075            MOV SIZBUF,R2           ;GET WORDS TO TRANSFER
001645  104303  002074            MOV DATBUF,R3           ;GET DM ADDRESS
001647  060014                    XFC UWRITE              ;WRITE TO THE HOST MEMORY
001650  000000                    RETURN
```

```
                                        ;READ AND PERFORM A DATA COMPARE ON ALL THE HOST BUFFERS. TWO WORD
                                        ;TABLE ENTRIES STARTING AT FREE CONTAIN THE HOST ADDRESS OF THE
                                        ;BUFFERS AND THE PATTERN NUMBER LAST WRITTEN TO THE BUFFER. THE
                                        ;NUMBER OF BUFFERS IS IN BUFCNT AND THE SIZE OF THE BUFFERS IS IN
                                        ;SIZBUF. THE DATA CAN BE READ INTO THE DM AT ADDRESS IN DATBUF.

001651  104205  002103          READ:   MOV #FREE,R5                    ;GET ADDRESS OF TABLE ENTRIES
001653  104304  002071                  MOV BUFCNT,R4                   ;GET COUNT OF BUFFERS

                                        ;READ DATA FROM HOST BUFFER INTO DM MEMORY

001655  104257                  1$:     MOV (R5)+,R0                    ;GET BUFFER ADDRESS
001656  104151                          MOV (R5),R1
001657  103201  177774                  BIC #177774,R1                  ;CLEAR PATTERN NUMBER
001661  104302  002075                  MOV SIZBUF,R2                   ;SIZE OF BUFFER
001663  104303  002074                  MOV DATBUF,R3                   ;PLACE TO READ DATA
001665  060013                          XFC UREAD                       ;READ THE DATA FROM HOST MEMORY

                                        ;COMPARE DATA READ WITH DATA PATTERN

001666  104207  002076                  MOV #CMPSIZ,R0                  ;GET STORAGE ADDRESS OF
001670  104071                          MOV R0,R1                       ;  DATA COMPARE DATA
001671  104303  002075                  MOV SIZBUF,R3                   ;STORE BUFFER SIZE
001673  100213                          MOV R3,(R1)+
001674  104303  002074                  MOV DATBUF,R3                   ;STORE ADDRESS
001676  100213                          MOV R3,(R1)+
001677  104252                          MOV (R5)+,R2                    ;GET PATTERN NUMBER
001700  103202  177763                  BIC #177763,R2                  ;  AND COMPUTE PATTERN
001702  105202  001723                  ADD #PAT0,R2                    ;  ADDRESS
001704  104223                          MOV (R2)+,R3                    ;MOVE DATA PATTERN WORDS
001705  100213                          MOV R3,(R1)+                    ;  TO DATA COMPARE
001706  104223                          MOV (R2)+,R3                    ;  STORAGE AREA
001707  100213                          MOV R3,(R1)+
001710  104223                          MOV (R2)+,R3
001711  100213                          MOV R3,(R1)+
001712  060006                          XFC COMPARE                     ;COMPARE DATA IN BUFFER TO PATTERN
001713  115001                          TST R1                          ;CHECK FOR ERRORS
001714  011720                          BEQ 3$
001715                                  ERRHRD MS6 ;DATA COMPARE FAILED AFTER WRITE THEN READ FROM UNIBUS
001715  021236                                                          CALL RERROR     ;ERROR # 2.
001716  001002                                                          .WORD <PRMS*2000>+<2*400>+ERRN
001717  000411                                                          .WORD MS6

                                        ;CYCLE THROUGH ALL HOST BUFFERS

001720  117404                  3$:     DEC R4                          ;COUNT BUFFERS
001721  051655                          BNE 1$                          ;REPEAT FOR ALL BUFFERS
001722  000000                          RETURN
```

```
                                ;DATA PATTERN TABLES
                        
001723  111111          PAT0:   .WORD ^B1001001001001001
001724  044444                  .WORD ^B0100100100100100
001725  022222                  .WORD ^B0010010010010010
001726  000000                  .WORD 0
                        
001727  177400          PAT1:   .WORD ^B1111111100000000
001730  007760                  .WORD ^B0000111111110000
001731  000377                  .WORD ^B0000000011111111
001732  000000                  .WORD 0
                        
001733  155555          PAT2:   .WORD ^B1101101101101101
001734  133333                  .WORD ^B1011011011011011
001735  066666                  .WORD ^B0110110110110110
001736  000000                  .WORD 0
                        
001737  000377          PAT3:   .WORD ^B0000000011111111
001740  170017                  .WORD ^B1111000000001111
001741  177400                  .WORD ^B1111111100000000
```

```
                                        ;CHECK IF TEST ADDRESS IS IN BOUNDS OF READABLE MEMORY

001742  106300  002070  002064  BCHKM:  CMP TADRH,FRADRH                ;COMPARE TEST ADDRESS WITH FIRST READABE ADDRESS
001745  011750                          BEQ 1$                          ;BRANCH IF EQUAL
001746  041755                          BCC 2$                          ;BRANCH IF TEST ADDRESS HIGHER
001747  002056                          BR BCHKMX                       ;BRANCH IF TEST ADDRESS LOWER
001750  106300  002067  002063  1$:     CMP TADR,FRADR
001753  041755                          BCC 2$                          ;BRANCH IF HIGHER OR SAME
001754  002056                          BR BCHKMX                       ;BRANCH IF LOWER
001755  106300  002066  002070  2$:     CMP LRADRH,TADRH                ;COMPARE TEST ADRESS WITH LAST READABLE ADDRESS
001760  011763                          BEQ 3$                          ;BRANCH IF EQUAL
001761  041770                          BCC 4$                          ;BRANCH IF READABLE ADDRESS HIGHER
001762  002056                          BR BCHKMX                       ;BRANCH IF READABLE ADDRESS LOWER
001763  106300  002065  002067  3$:     CMP LRADR,TADR
001766  041770                          BCC 4$                          ;BRANCH IF HIGHER OR SAME
001767  002056                          BR BCHKMX                       ;BRANCH IF LOWER

001770  104305  002057          4$:     MOV FWADR,R5                    ;WRITE ONES INTO FIRST ADDRESS
001772  104304  002060                  MOV FWADRH,R4
001774  104207  177777                  MOV #177777,R0
001776  021170                          CALL MWRITE
001777  104305  002067                  MOV TADR,R5                     ;READ FROM TEST ADDRESS
002001  104304  002070                  MOV TADRH,R4
002003  021152                          CALL MREAD
002004  106207  177777                  CMP #177777,R0                  ;CHECK DATA READ FOR ALL ONES
002006  052056                          BNE BCHKMX                      ;GO TO NEXT BIT IF NOT ALL ONES

002007  104305  002057                  MOV FWADR,R5                    ;WRITE ALL ZEROS TO FIRST ADDRESS
002011  104304  002060                  MOV FWADRH,R4
002013  114007                          CLR R0
002014  021170                          CALL MWRITE
002015  104305  002067                  MOV TADR,R5                     ;READ FROM TEST ADDRESS
002017  104304  002070                  MOV TADRH,R4
002021  021152                          CALL MREAD
002022  115007                          TST R0                          ;CHECK DATA READ FOR ALL ZEROS
002023  052056                          BNE BCHKMX                      ;GO TO NEXT BIT IF NOT ALL ZEROS

002024  104301  002057                  MOV FWADR,R1                    ;COMPUTE XOR OF FIRST ADDRESS
002026  104052                          MOV R5,R2                       ; AND TEST ADDRESS
002027  021211                          CALL XOR
002030  104027                          MOV R2,R0                       ;RESULT TO R0
002031  104301  002060                  MOV FWADRH,R1                   ;NOW DO IT FOR HIGH BITS
002033  104042                          MOV R4,R2
002034  021211                          CALL XOR
002035                                  ERRHRD MS7,FWADR,FWADRH,R5,R4,R0,R2 ;UNIBUS ADDRESSING ERROR. TWO ADDRESSES READ SAM
002035  100462                                                                  MOV R2,-(SP)
002036  100467                                                                  MOV R0,-(SP)
002037  100464                                                                  MOV R4,-(SP)
002040  100465                                                                  MOV R5,-(SP)
002041  104010  001316                                                          MOV R1,SAVREG
002043  104301  002060                                                          MOV FWADRH,R1
002045  100461                                                                  MOV R1,-(SP)
002046  104301  002057                                                          MOV FWADR,R1
002050  100461                                                                  MOV R1,-(SP)
002051  104301  001316                                                          MOV SAVREG,R1
002053  021236                                                                  CALL RERROR     ;ERROR # 3.
002054  015003                                                                  .WORD <PRMS*2000>+<2*400>+ERRN
```

FREE MEMORY CHECK

```
002055  000446                                          .WORD MS7
46
47 002056  000000                  BCHKMX: RETURN
```

```
                                ;PROGRAM VARIABLES
                                
002057  000000                  FWADR:  .WORD 0                 ;FIRST ADDRESS CONTAINING ADDRESS DATA
002060  000000                  FWADRH: .WORD 0
002061  000000                  LWADR:  .WORD 0                 ;LAST ADDRESS CONTAINING ADDRESS DATA
002062  000000                  LWADRH: .WORD 0
                                
002063  000000                  FRADR:  .WORD 0                 ;FIRST ADDRESS READABLE
002064  000000                  FRADRH: .WORD 0
002065  000000                  LRADR:  .WORD 0                 ;LAST ADDRESS READABLE
002066  000000                  LRADRH: .WORD 0
                                
002067  000000                  TADR:   .WORD 0                 ;TEST ADDRESS
002070  000000                  TADRH:  .WORD 0
                                
002071  000000                  BUFCNT: .WORD 0                 ;COUNT OF BUFFERS IN HOST MEMORY
002072  000000                  CURBUF: .WORD 0                 ;CURRENT BUFFER BEING WRITTEN IN HOST
002073  000000                  CURPAT: .WORD 0                 ;CURRENT DATA PATTERN BEING WRITTEN
002074  000000                  DATBUF: .WORD 0                 ;ADDRESS OF DATA BUFFER IN DM MEMORY
002075  000000                  SIZBUF: .WORD 0                 ;SIZE OF DATA BUFFER IN HOST MEMORY
                                
                                ;HOST BUFFER TABLE ENTRIES ARE STORED AFTER THE PROGRAM AT FREE.
                                ;       TWO WORDS PER TABLE
                                ;               FIRST WORD  - LOW 16 BITS OF HOST ADDRESS
                                ;               SECOND WORD - BITS 1-0 HIGH TWO BITS OF HOST ADDRESS
                                ;                             BITS 3-2 PATTERN LAST WRITTEN
                                
002076  000000                  CMPSIZ: .WORD 0                 ;TABLE FOR COMPARE DATA PATTERN XFC
002077  000000                  CMPADR: .WORD 0
002100  000000                  CMP1:   .WORD 0
002101  000000                  CMP2:   .WORD 0
002102  000000                  CMP3:   .WORD 0
```

```
                                ;MESSAGE STORAGE OVERLAY

002103                          FREE:                                   ; FREE SPACE BEGINS HERE
                                        DMOVLY MS,0
002103  154530                          .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                        .NLIST BEX

000000  042  116  117  MS1:     .ASCII\'NON-EXISTANT MEMORY ERROR TRYING TO READ FROM UNIBUS.'N\
000034  042  011  101          .ASCII\''       ADDRESS         '018N\
000044  000                     .BYTE 0
000045  042  120  101  MS2:     .ASCII\'PARITY ERROR ON READ FROM UNIBUS.'N\
000067  042  011  101          .ASCII\''       ADDRESS         '018N\
000077  042  011  104          .ASCII\''       DATA READ       '016N\
000107  042  011  104          .ASCII\''       DATA EXPECTED   '016N\
000122  000                     .BYTE 0
000123  042  125  116  MS3:     .ASCII\'UNIBUS ADDRESSING ERROR - INCORRECT DATA READ.'N\
000153  042  115  105          .ASCII\'MEMORY LOCATION SHOULD CONTAIN OWN ADDRESS.'N\
000202  042  011  101          .ASCII\''       ADDRESS         '018N\
000212  042  011  104          .ASCII\''       DATA READ       '016N\
000223  042  011  104          .ASCII\''       DATA EXPECTED   '016N\
000235  000                     .BYTE 0
000236  042  125  116  MS4:     .ASCII\'UNIBUS WRITE FAILED.'N\
000251  042  011  105          .ASCII\''       ERROR CODE FROM R1      '016N\
000266  042  011  106          .ASCII\''       FIRST ADDRESS WRITTEN   '018N\
000305  042  011  127          .ASCII\''       WORDS TRYING TO WRITE   'D''.'N\
000324  000                     .BYTE 0
000325  042  125  116  MS5:     .ASCII\'UNIBUS READ FAILED.'N\
000340  042  011  105          .ASCII\''       ERROR CODE FROM R1      '016N\
000355  042  011  106          .ASCII\''       FIRST ADDRESS READ      '018N\
000372  042  011  127          .ASCII\''       WORDS TRYING TO READ    'D''.'N\
000410  000                     .BYTE 0
000411  042  104  101  MS6:     .ASCII\'DATA COMPARE FAILED AFTER WRITE THEN READ FROM UNIBUS.'N\
000445  000                     .BYTE 0
000446  042  125  116  MS7:     .ASCII\'UNIBUS ADDRESSING ERROR. TWO ADDRESSES READ SAME LOCATION.'N\
000504  042  011  113          .ASCII\''       KNOWN GOOD ADDRESS      '018N\
000521  042  011  105          .ASCII\''       ERROR ADDRESS           '018N\
000534  042  011  101          .ASCII\''       ADDRESS BIT IN ERROR    '018N\
000552  000                     .BYTE 0

000553                          DMEND
000553  066640                  .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
        000001                  .END
```

SYMBOL TABLE

```
ACHK1   001401        ECCFLG= 010000        INITW = 040000        MS3     000123        RBUFLN= 000415
ACHK2   001402        ECCRSH= 000002        INSEEK= 000012        MS4     000236        RCLB  = 000001
ACHK3   001413        ECHO  = 000010        INTEDC= 000105        MS5     000325        RCONT = 000000
ACHK4   001421        ECHOC = 000350        IN.RQ   001044        MS6     000411        RCTCPS= 000001
ARGS$ = 000007        EOC   = 100000        IN.01   001045        MS7     000446        RCTCSZ= 000014
ATTN  = 000002        ERECOV= 000006        IN.02   001046        MWR   = 000017        RCV   = 000005
AVAIL = 000100        ERHARD= 001000        IN.03   001047        MWRITE  001170        RCVERR= 000004
BCHK    001426        ERLEV = 000002        IN.04   001050        NEWSUB= 004000        RCVRDY= 000001
BCHKM   001742        ERRMC = 000014        IN.05   001051        ONLYCL= 000200        RDSTAT  000720
BCHKMX  002056        ERRMES= 000013        IN.06   001052        OUT.RQ  001006        READ    001651
BCHK1   001434        ERRN  = 000004        IN.07   001053        OUT.01  001007        REDWRT= 000100
BCHK2   001453        ERSOFT= 001400        IN.08   001054        OUT.02  001010        REGS$ = 177777
BEUSED= 000040        EXIT  = 000021        IN.09   001055        OUT.03  001011        REGU$ = 000001
BF.DAT= 000000        FB.DAT= 000000        IN.10   001056        OUT.04  001012        RERRCA  001301
BF.ECC= 000401        FB.EDC= 000400        IN.11   001057        OUT.05  001013        RERROR  001236
BF.EDC= 000400        FCTSIZ= 000010        IN.12   001060        OUT.06  001014        RERRPA  001275
BREAK = 000000        FIRSTU= 007717        IN.13   001061        OUT.07  001015        RESEEK= 020000
BUFCNT  002071        FMEMFL  001371        IN.14   001062        OUT.08  001016        RETRY = 000004
BUFFLG= 040000        FORMAT= 000001        IN.15   001063        OUT.09  001017        RETS  = 000001
BUFSIZ= 000036        FRADR   002063        IN.16   001064        OUT.10  001020        REVEC = 000400
CCHK    001466        FRADRH  002064        IN.17   001065        OUT.11  001021        REVECT= 000004
CHECK = 000010        FRAME = 000004        IN.18   001066        OUT.12  001022        REVINP- 000040
CHGMOD= 000201        FREE    002103        IN.19   001067        OUT.13  001023        REVS  = 000007
CHRRES= 000170        FSTOP = 100000        IN.20   001070        OUT.14  001024        RM    = 000004
CLRBUF  000777        FTIME = 001000        IN.21   001071        OUT.15  001025        RONLY = 004000
CMPADR  002077        FTLDEV= 000400        IN.22   001072        OUT.16  001026        RREAL = 013400
CMPSIZ  002076        FTLSYS= 000000        IN.23   001073        OUT.17  001027        RSTOP = 100000
CMP1    002100        FT.BUF= 000000        IN.24   001074        OUT.18  001030        RTRIES= 001000
CMP2    002101        FT.HI = 000001        IN.25   001075        OUT.19  001031        RWRDY = 100000
CMP3    002102        FT.LOW= 000002        IN.26   001076        OUT.20  001032        RW.ANG= 000006
COMPAR= 000006        FWADR   002057        IN.27   001077        OUT.21  001033        RW.BUF= 000001
COMPLT= 000176        FWADRH  002060        IN.28   001100        OUT.22  001034        RW.CMD= 000004
CURBUF  002072        GETCHR= 000207        IN.29   001101        OUT.23  001035        RW.HI = 000003
CURPAT  002073        GETSTA= 000011        IRECLB= 000216        OUT.24  001036        RW.LOW= 000002
CVT   = 000020        GETSUB- 000210        LARGE = 000001        OUT.25  001037        RW.SDI= 000005
DATBUF  002074        GRPCYL= 000002        LBHINB= 177417        OUT.26  001040        RW.STA= 000000
DATCMP- 000002        GRPOFF= 000011        LBLONB= 177760        OUT.27  001041        SAVREG  001316
DATERR= 000020        HBHINB= 007777        LBNCYL= 000000        OUT.28  001042        SBCRES= 000167
DATPRE= 000005        HBLONB= 170377        LBNHST- 000012        OUT.29  001043        SCTWRD= 000377
DBNCYL= 000022        HDRPRE= 000005        LBNTRK= 000011        OVERFL= 000002        SDILTO  001151
DCMPAL- 000001        HD.BAD- 110000        LINKLN= 000007        OVE.MN= 000714        SDISTO  001150
DCYLS = 020000        HD.DBN= 140000        LOBYTE= 000377        OVE.MS= 000000        SDIVER= 000000
DINIT = 000011        HD.LBN= 000000        LONG    001360        OVL.MN= 001170 G      SEKINP= 002000
DIREC - 010000        HD.PRV= 050000        LONGTO= 000001        OVL.MS= 000554 G      SEND  = 000004
DISCON= 000204        HD.RBN- 060000        LOW   = 000002        OVL...= 001744        SEQSEK= 000100
DONE  - 000016        HD.REV= 030000        LRADR   002065        OVSTRT= 007774        SHRTTO= 000000
DONECD  001575        HD.XBN= 120000        LRADRH  002066        OVS.MN= 001040 G      SIZBUF  002075
DROP  = 100000        HEADER= 000002        LUNIT   001315        OVS.MS= 003420 G      SNDAGN  000756
DRTYPE= 000007        HIBYTE= 177400        LWADR   002061        OV... = 002364        SPADJU  001312
DRVCLR= 000005        HICYL = 000001        LWADRH  002062        PAT0    001723        SS    = 000001
DRVID = 000004        HIDBN = 000003        MEDTYP= 000006        PAT1    001727        STACK   001357
DRVONL= 000213        HILBN = 000002        MESSAG= 000015        PAT2    001733        START   001360
DRVRUN= 000014        HIMEM = 010000        MICREV= 000003        PAT3    001737        STATEX  000746
D.LIMT- 000001        HIRBN = 000003        MRD   - 000016        PRMS  = 000006        STATLP  000724
D.SCHR= 000002        HIXBN = 000002        MREAD   001152        PTYPE$= 000030        STATOK  000737
ECC   = 000015        HOSTRQ  000751        MS1     000000        RBNBN - 000200        STATUS= 000007
ECCCHR= 010000        HRDREV- 000003        MS2     000045        RBNTRK- 000004        STSRES= 000366
```

SYMBOL TABLE

```
ST.C   = 000002        S.MCNT= 000011        T1MSIZ= 000000        U.COPY= 000043        U.SEEK= 000007
ST.CON= 000002        S.MEGR= 000006        T2CMD = 000002        U.CSEC= 000021        U.SNUM= 000047
ST.DB  = 001000        S.MEGW= 000007        T2DLL = 000001        U.CTRK= 000015        U.SUBP= 000001
ST.DF  = 000020        S.PARM= 000000        T4BB1 = 000005        U.ECCT= 000027        U.SUBU= 000034
ST.DR  = 000040        S.PAT = 000003        T4BB2 = 000006        U.ELEV= 000024        U.TIMO= 000005
ST.ERR= 000002        S.SCHR= 000005        T4MPRM= 000003        U.LCYL= 000056        U.TSEC= 000020
ST.FE  = 000200        S.SDCL= 000001        T4MXFR= 000011        U.LGRP  000060        U.WPRT= 000032
ST.FO  = 002000        S.TGOF= 000013        T4SEEK= 000010        U.MASK= 000022        U.WRIT= 000023
ST.MOD= 000001        S.TGSS= 000015        T4SOFT= 000007        U.MBN  = 000035        WAITSI= 000012
ST.MSK= 000000        S.TRKL= 000004        T4UPRM= 000004        U.MLEV= 000026        WBUFLN= 000401
ST.OA  = 000200        TADR    002067        UDATA   001210        U.MSEC= 000017        WCHECK= 000010
ST.PE  = 000040        TADRH   002070        UNIT0 = 000001        U.NEXT= 000000        WCHKAL= 000004
ST.PS  = 000002        TALKER  001102        UNIT1 = 000002        U.NFUN= 000010        WCUNT = 040000
ST.RE  = 000100        TALK1A  001114        UNIT2 = 000004        U.NSEC= 000016        WONLY = 002000
ST.RR  = 000100        TALK1B  001124        UNIT3 = 000010        U.PARM= 000033        WREAL = 122400
ST.RTY= 000003        TALK2A  001144        UNSSUC= 000175        U.PAT  = 000011        WRITE   001601
ST.RU  = 000001        TALK2B  001145        UREAD = 000013        U.PCTG= 000014        WRONG = 000002
ST.SR  = 000020        TEST4 = 000000        UTOTST= 000012        U.RBN  = 000041        WSTOP = 140000
ST.STA= 000001        TIMEOU= 000001        UWRITE= 000014        U.RTRY= 000025        XBNCYL= 000021
ST.ST  = 000400        TLEN.U= 000061        U.CBN  = 000037        U.RVER= 000046        XFERRT= 000000
ST.UNT= 000000        TO      001227        U.CCNT= 000012        U.RWER= 000045        XMTERR= 000400
ST.WE  = 000010        TOOBIG= 000001        U.CCOP= 000044        U.RWTO= 000006        XOR     001211
SUB    = 000013        TRACKS= 000020        U.CCYL= 000053        U.SDIL= 000031        XREAD = 000002
S.BADP= 000010        TRKGRP= 000003        U.CGRP= 000055        U.SDIS= 000030        XWRITE= 000003
S.BESS= 000011

. ABS.  004750      000
        000000      001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  4286 WORDS  ( 17 PAGES)
DYNAMIC MEMORY AVAILABLE FOR  69 PAGES
A:UDAT1.BIN,SY:UDAT1/C=[1,33]DMACRO,UDAT1T,UDATP,UDATR,UDAT1M,UDAT1S
```

CROSS REFERENCE TABLE (CREF V01-05)

CROSS REFERENCE TABLE (CREF V01-05)

```
ECCCHK   8-28#
ECCFLG   3-45#
ECCRSH   5-11#
ECHO     4-11#
ECHOC    6-46#
EOC      3-46#
ERECOV   6-31#
ERHARD   6-52#
ERLEV    5-10#
ERRMC    6-15#
ERRMES   6-14#   29-37
ERRN    32-3#    32-28   32-28   32-28   32-28#  38-39   38-39   38-39   38-39#  40-45   40-45   40-45   40-45#
ERSOFT   6-53#
EXIT     4-20#
FB.DAT   3-22#
FB.EDC   3-23#
FCTSIZ   5-36#
FIRSTU   7-11#
FMEMFL  32-13#   32-16
FORMAT   4-4#
FRADR   40-7*    41-8#
FRADRH  40-3*    41-9#
FRAME    5-66#
FREE    34-18    34-20   37-34   38-7    42-3#
FSTOP    3-42#
FT.BUF   3-15#
FT.HI    3-16#
FT.LOW   3-17#
FTIME    8-16#
FTLDEV   6-51#
FTLSYS   6-50#
FWADR   32-11    32-20   33-8    33-15   34-21   40-18   40-28   40-38   40-45   40-45   41-3#
FWADRH  32-21    33-4    33-19   34-22   40-19   40-29   40-42   40-45   40-45   41-4#
GETCHR   6-36#
GETSTA   6-38#
GETSUB   6-37#
GRPCYL   5-25#
GRPOFF   5-38#
HBHINB   5-56#
HBLONB   5-57#
HD.BAD   3-55#
HD.DBN   3-58#
HD.LBN   3-52#
HD.PRV   3-56#
HD.RBN   3-53#
HD.REV   3-54#
HD.XBN   3-57#
HDRPRE   5-34#
HEADER   5-62#
HIBYTE   5-54#
HICYL    5-24#
HIDBN    5-30#
HILBN    5-26#
HIMEM    3-9#     7-11
HIRBN    5-29#
HIXBN    5-27#
HOSTRQ  22-2#    29-38   32-9    36-2
```

CROSS REFERENCE TABLE (CREF V01-05)

```
HRDREV    5-13#
IN.01    23-39#    32-10
IN.02    23-40#
IN.03    23-41#
IN.04    23-42#
IN.05    23-43#
IN.06    23-44#
IN.07    23-45#
IN.08    23-46#
IN.09    23-47#
IN.10    23-48#
IN.11    23-49#
IN.12    23-50#
IN.13    23-51#
IN.14    23-52#
IN.15    23-53#
IN.16    23-54#
IN.17    23-55#
IN.18    23-56#
IN.19    23-57#
IN.20    23-58#
IN.21    23-59#
IN.22    23-60#
IN.23    23-61#
IN.24    23-62#
IN.25    23-63#
IN.26    23-64#
IN.27    23-65#
IN.28    23-66#
IN.29    23-67#
IN.RQ    22-19     23-38#    23-68
INITW     8-11#
INSEEK    6-40#
INTEDC    7-9#
IRECLB    6-39#
LARGE     5-74#
LBHINB    5-58#
LBLONB    5-59#
LBNCYL    5-23#
LBNHST    5-39#
LBNTRK    5-37#
LINKLN    3-70#
LOBYTE    5-55#
LONG     24-23     32-7#
LONGTO    5-6#
LOW       5-70#
LRADR    40-14     41-10#
LRADRH   40-10     41-11#
LUNIT    29-20     29-44#
LWADR    32-32     34-33     41-5#
LWADRH   32-34     34-30     41-6#
MEDTYP    5-35#
MESSAG    6-16#
MICREV    5-12#
MRD       4-17#    22-16
MREAD    25-10#    32-22     40-24     40-34
MS1      42-7#
```

CROSS REFERENCE TABLE (CREF V01-05)

```
MS2      42-10#
MS3      32-28    42-15#
MS4      42-21#
MS5      42-26#
MS6      38-39    42-31#
MS7      40-45    42-33#
MWP       4-18#   22-21
MWRITE   26-10#   40-21    40-31
NEWSUB    8-14#
ONLYCL    8-32#
OUT.01   22-22    22-23    23-6#    29-13
OUT.02   23-7#
OUT.03   23-8#
OUT.04   23-9#
OUT.05   23-10#
OUT.06   23-11#
OUT.07   23-12#
OUT.08   23-13#
OUT.09   23-14#
OUT.10   23-15#
OUT.11   23-16#
OUT.12   23-17#
OUT.13   23-18#
OUT.14   23-19#
OUT.15   23-20#
OUT.16   23-21#
OUT.17   23-22#
OUT.18   23-23#
OUT.19   23-24#
OUT.20   23-25#
OUT.21   23-26#
OUT.22   23-27#
OUT.23   23-28#
OUT.24   23-29#
OUT.25   23-30#
OUT.26   23-31#
   .27   23-32#
   .28   23-33#
   .29   22-23    23-34#
OUT.RQ   22-13*   22-14    23-5#
OV...     2-38     2-38     2-38#   42-4    42-4    42-4    42-4#   42-39   42-39#
OVE.MN    2-38#   42-4
OVE.MS   42-4#    42-39
OVERFL    5-75#
OVL...    2-38     2-38#   42-4    42-4#   42-39   42-39#
OVL.MN    2-38     2-38    42-4    42-4    42-4#
OVL.MS   42-39    42-39    42-39#
OVS.MN    2-38#
OVS.MS   42-4#
OVSTRT    3-10#
PAT0     37-13    38-29    39-3#
PAT1     39-8#
PAT2     39-13#
PAT3     39-18#
PRMS     32-28    32-28    32-28    32-28    32-28    32-28    32-28    32-28    32-28    32-28    32-28#   38-39    38-39    38-39
         38-39    38-39    38-39    38-39    38-39    38-39    38-39    38-39#   40-45    40-45    40-45    40-45    40-45    40-45
         40-45    40-45    40-45    40-45    40-45#
```

CROSS REFERENCE TABLE (CREF V01-05)

```
PTYPE$  32-28   32-28   32-28   32-28   32-28   32-28   32-28   32-28   32-28#  32-28#  32-28#  32-28#  40-45   40-45
        40-45   40-45   40-45   40-45   40-45   40-45   40-45   40-45   40-45   40-45   40-45#  40-45#  40-45#  40-45#
        40-45#  40-45#
RBNBN    8-18#
RBNTRK   5-31#
RBUFLN   3-69#
RCLB     8-23#
RCONT    3-41#
RCTCPS   5-8#
RCTCSZ   5-40#
RCV      4-8#    24-30
RCVERR   6-23#   21-17
RCVRDY   6-21#
RDSTAT  21-2#
READ    35-11   35-22   38-7#
REDWRT   8-19#
REGS$   32-28   32-28   32-28   32-28   32-28   32-28#  38-39   38-39#  40-45   40-45   40-45   40-45   40-45   40-45
        40-45   40-45   40-45   40-45   40-45   40-45#  40-45#  40-45#
REGU$   40-45   40-45   40-45   40-45#  40-45#
RERRCA  29-31   29-36#
RERROR  29-7#   32-28   38-39   40-45
RERRPA  29-32#  29-35
RESEEK   8-12#
RETRY    8-22#
RETS     5-7#
REVEC    8-17#
REVECT   5-63#
REVINP   8-20#
REVS     5-16#
RM       5-32#
RONLY    8-29#
RREAL    3-44#
RSTOP    3-40#
RTRIES   8-31#
RW.ANG   3-34#
RW.BUF   3-29#
RW.CMD   3-32#
RW.HI    3-31#
RW.LOW   3-30#
RW.SDI   3-33#
RW.STA   3-28#
RWRDY    6-26#
S.BADP   7-62#   7-63
S.BESS   7-63#   7-68
S.MCNT   7-68#   7-69
S.MEGR   7-60#   7-61
S.MEGW   7-61#   7-62
S.PARM   7-55#   7-56
S.PAT    7-57#   7-58
S.SCHR   7-59#   7-60
S.SDCL   7-56#   7-57
S.TGOF   7-69#   7-70
S.TGSS   7-70#
S.TRKL   7-58#   7-59
SAVREG  29-46#  40-45   40-45*
SBCRES   6-44#
SCTWRD   7-8#
```

CROSS REFERENCE TABLE (CREF V01-05)

```
SDILTO  24-27    24-46#
SDISTO  24-25    24-45#
SDIVER   5-4#
SEKINP   8-15#
SEND     4-7#    24-18
SEQSEK   8-34#
SHRTTO   5-3#
SIZBUF  34-42*   37-18    37-40    38-15    38-23    41-20#
SNDAGN  22-14#   22-18
SPADJU  29-30*   29-40#
SS       5-9#
ST.C     4-30#
ST.CON   4-29#
ST.DB    4-47#
ST.DF    4-44#
ST.DR    4-37#
ST.ERR   4-28#
ST.FE    4-41#
ST.FO    4-46#
ST.MOD   4-27#
ST.MSK   4-25#
ST.OA    4-35#
ST.PE    4-43#
ST.PS    4-39#
ST.RE    4-42#
ST.RR    4-36#
ST.RTY   4-31#
ST.RU    4-40#
ST.S7    4-48#
ST.SR    4-38#
ST.STA   4-26#
ST.UNT   4-24#
ST.WE    4-45#
STACK   20-8     30-5#
START   20-9     32-8#
STATEX  21-16    21-18    21-22#
STATLP  21-9#    21-14    21-20
STATOK  21-12    21-17#
STATUS   4-10#   21-9
STSRES   6-45#
SUB      5-22#
T1MSIZ   6-3#    32-8
T2CMD    6-5#
T2DLL    6-4#
T4BB1    6-8#
T4BB2    6-9#
T4MPRM   6-6#
T4MXFR   6-12#
T4SEEK   6-11#
T4SOFT   6-10#
T4UPRM   6-7#
TADR    33-10*   33-16*   40-7     40-14*   40-22    40-32    41-13#
TADRH   33-5*    33-21*   40-3     40-10*   40-23    40-33    41-14#
TALK1A  24-20    24-23#
TALK1B  24-26    24-28#   24-38
TALK2A  24-32    24-41#
TALK2B  24-22    24-36    24-40    24-42#
```

CROSS REFERENCE TABLE (CREF V01-05)

```
TALKER  24-15#
TEST4    2-36#    24-1
TIMEOU   5-61#
TLEN.U   7-10#     7-11
TO      28-1#
TOOBIG   5-69#
TRACKS   8-36#
TRKGRP   5-28#
U.CBN    7-41#     7-42
U.CCNT   7-22#     7-23
U.CCOP   7-44#     7-45
U.CCYL   7-48#     7-49
U.CGRP   7-49#     7-50
U.COPY   7-43#     7-44
U.CSEC   7-28#     7-29
U.CTRK   7-24#     7-25
U.ECCT   7-34#     7-35
U.ELEV   7-31#     7-32
U.LCYL   7-50#     7-51
U.LGRP   7-10      7-51#
U.MASK   7-29#     7-30
U.MBN    7-40#     7-41
U.MLEV   7-33#     7-34
U.MSEC   7-26#     7-27
U.NEXT   7-15#     7-16
U.NFUN   7-20#     7-21
U.NSEC   7-25#     7-26
U.PARM   7-38#     7-39
U.PAT    7-21#     7-22
U.PCTG   7-23#     7-24
U.RBN    7-42#     7-43
U.RTRY   7-32#     7-33
U.RVER   7-46#     7-47
U.RWER   7-45#     7-46
U.RWTO   7-18#     7-19
U.SDIL   7-36#     7-37
U.SDIS   7-35#     7-36
U.SEEK   7-19#     7-20
U.SNUM   7-47#     7-48
U.SUBP   7-16#     7-17
U.SUBU   7-39#     7-40
U.TIMO   7-17#     7-18
U.TSEC   7-27#     7-28
U.WPRT   7-37#     7-38
U.WRIT   7-30#     7-31
UDATA   25-14     25-16     26-11*    26-15     26-20#
UNIT0    5-46#
UNIT1    5-47#
UNIT2    5-48#
UNIT3    5-49#
UNSSUC   6-42#
UREAD    4-14#    25-15     38-17
UTOTST   6-13#
UWRITE   4-15#    26-16     37-42
WAITSI   4-13#
WBUFLN   3-68#     3-69
WCHECK   8-37#
```

CROSS REFERENCE TABLE (CREF V01-05)

```
WCHKAL    8-39#
WCONT     3-39#
WONLY     8-30#
WREAL     3-43#
WRITE    35-6      35-21     37-9#
WRONG     5-65#
WSTOP     3-38#
XBNCYL    5-41#
XFERRT    5-5#
XMTERR    6-25#     21-11
XOR      27-10#     33-9      33-20     40-40     40-44
XREAD     4-5#
XWRITE    4-6#
```

CROSS REFERENCE TABLE (CREF V01-05)

UDAT2 DISK RESIDENT MACRO X04.00 9-JUL-81 11:39:12

TABLE OF CONTENTS

```
47-   2       READ MEMORY SUBROUTINE
48-   1       WRITE MEMORY SUBROUTINE
49-   1       CONVERT MEMORY -- SKEWED BY BYTE
50-   1       GET BYTE COUNT
51-   1       TYPE WHAT KIND OF RECEIVE ERROR
52-   1       GET UNITS FROM HOST
54-   1       GET STATUS SUBROUTINE
55-   1       CLEAR DRIVE SUBROUTINE
56-   1       DIVIDE BY OCTAL 50 AND FIND ASCII EQUIVALENT
```

```
                                        .TITLE  UDAT2   DISK RESIDENT

                                 ;
                                 ;
                                 ; COPYRIGHT (C)  1981
                                 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
                                 ;
                                 ;
                                 ; THIS  SOFTWARE  IS  FURNISHED  UNDER A LICENSE FOR USE ONLY ON A
                                 ; SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION
                                 ; OF  THE  ABOVE COPYRIGHT  NOTICE.  THIS SOFTWARE, OR  ANY  OTHER
                                 ; COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE  MADE AVAILABLE
                                 ; TO  ANY  OTHER PERSON  EXCEPT  FOR USE ON SUCH SYSTEM AND TO ONE
                                 ; WHO AGREES TO THESE LICENSE TERMS.  TITLE  TO  AND  OWNERSHIP OF
                                 ; THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DIGITAL.
                                 ;
                                 ; THE  INFORMATION  IN THIS DOCUMENT IS SUBJECT  TO CHANGE WITHOUT
                                 ; NOTICE AND SHOULD NOT BE CONSTRUED AS A  COMMITMENT  BY  DIGITAL
                                 ; EQUIPMENT CORPORATION.
                                 ;
                                 ; DIGITAL ASSUMES NO  RESPONSIBILITY FOR THE USE OR RELIABILITY OF
                                 ; ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
                                 ;
                                 ;
                                 ;
                                 ;
                                 ;
                                 ;
                                 ;
                                 ;
                                 ;
                                 ;THIS PROGRAM SHALL BE ASSEMBLED WITH THE PROGRAM DMACRO
                                 ;USING A COMMAND LINE SIMILAR TO:
                                 ;
                                 ;  UDAT2.BIN,UDAT2/C=[1,2]DMACRO,UDAT2T,UDATP,UDAT2M,UDATR,UDAT2S

               000000            TEST4   =       0                       ; THIS IS NOT TEST4
 000000                                  .ENABL  ABS
 000000                                  DMCODE  UDAT2,0,714,3,0
```

```
UDA DM PROGRAM PARAMETERS

                                .SBTTL  UDA DM PROGRAM PARAMETERS

                                        .LIST MEB,TOC

                                ;
                                ;       EQUATES
                                ;
                                ;       HIGHEST USABLE LOCATION OF UDA MEMORY + 1
                                ;
010000                          HIMEM   =       10000                   ; HIGH MEMORY+1
007774                          OVSTRT  =       7774                    ; OVERLAY ADDRESS LOCATION
                                ;
                                ;       OFFSETS FOR FORMAT TRACK TABLE
                                ;
000000                          FT.BUF  =       0.                      ;BUFFER POINTER OFFSET
000001                          FT.HI   =       1.                      ;HI ORDER HEADER OFFSET
000002                          FT.LOW  =       2.                      ;LOW ORDER HEADER OFFSET
                                ;
                                ;       OFFSETS FOR FORMAT TRACK BUFFER
                                ;
000000                          FB.DAT  =       0.                      ;FIRST DATA WORD OFFSET
000400                          FB.EDC  =       256.                    ;EDC WORD OFFSET
                                ;
                                ;       OFFSETS FOR READ/WRITE I/O CHAIN TABLES
                                ;
000000                          RW.STAT =       0.                      ;STATUS AND NEXT BUFFER POINTER OFFSET
000001                          RW.BUF  =       1.                      ;POINTER TO DATA BUFFER
000002                          RW.LOW  =       2.                      ;HI ORDER EXPECTED HEADER
000003                          RW.HI   =       3.                      ;LOW ORDER EXPECTED HEADER
000004                          RW.CMD  =       4.                      ;SDI COMMAND AND HEAD ADDRESS
000005                          RW.SDI  =       5.                      ;DUMMY SDI CONTROL BLOCK POINTER
000006                          RW.ANG  =       6.                      ;THETA FROM INDEX
                                ;
                                ;       CONSTANTS FOR READ AND WRITE XFC'S
                                ;
140000                          WSTOP   =       140000                  ; LAST ENTRY IN CHAIN FOR WRITE
040000                          WCONT           40000                   ; WRITE CONTINUE
100000                          RSTOP   =       100000                  ; LAST ENTRY IN CHAIN FOR READ
000000                          RCONT   =       0                       ; READ CONTINUE
100000                          FSTOP   =       100000                  ; LAST ENTRY IN CHAIN FOR FORMAT
122400                          WREAL   =       122400                  ; WRITE REAL TIME ECOMMAND
013400                          RREAL   =       13400                   ; READ REAL TIME COMMAND
010000                          ECCFLG          10000                   ; ECC ERROR IN BUFFER BIT
100000                          EOC             100000                  ; END OF CHAIN
040000                          BUFFLG  =       40000                   ; BUFFER FULL OR EMPTY FLAG
                                ;
                                ;       HEADER CODES
                                ;
000000                          HD.LBN  =       000000                  ;GOOD LBN
060000                          HD.RBN  =       060000                  ;GOOD RBN, PERHAPS UNUSED
030000                          HD.REV  =       030000                  ;REVECTORED LBN
110000                          HD.BAD  =       110000                  ;BAD BLOCK
050000                          HD.PRV  =       050000                  ;PRIMARY REVECTORED BLOCK
120000                          HD.XBN          120000                  ;XBN BLOCK
```

```
                140000          HD.DBN  -       140000                  ;DBN BLOCK

                                ;       OFFSETS FOR DATA BUFFERS

                000000          BF.DAT  =         0.                    ;DATA
                000400          BF.EDC  =       256.                    ;ERROR DETECTION CODE
                000401          BF.ECC  =       257.                    ;LAST 17 ECC RESIDUES

                                ;       BUFFER AND READ/WRITE CHAIN LINK SIZES

                000401          WBUFLN  -       257.                    ; WRITE BUFFER SIZE
                000415          RBUFLN  -       WBUFLN+12.              ; READ BUFFER SIZE
                000007          LINKLN  =       7.                      ; LINK SIZE
```

```
                                ;       XFC DEFINITION EQUATES
                                ;
000000                          BREAK   =       0.      ;BREAKPOINT XFC CODE
000001                          FORMAT  =       1.      ;FORMAT TRACK XFC CODE
000002                          XREAD   =       2.      ;READ N SECTORS XFC CODE
000003                          XWRITE  =       3.      ;WRITE N SECTORS XFC CODE
000004                          SEND    =       4.      ;SEND SDI COMMAND XFC CODE
000005                          RCV     =       5.      ;RECEIVE SDI MESSAGE XFC CODE
000006                          COMPARE =       6.      ;COMPARE DATA PATTERN TO BUFFER
000007                          STATUS  =       7.      ;RETURN DRIVE STATUS XFC CODE
000010                          ECHO    =       8.      ;ECHO DATA TO DRIVE XFC CODE
000011                          DINIT   =       9.      ;DRIVE INITIALIZE XFC CODE
000012                          WAITSI  =       10.     ;WAIT FOR SECTOR OR INDEX PULSE
000013                          UREAD   =       11.     ;READ UNIBUS MEMORY XFC CODE
000014                          UWRITE  =       12.     ;WRITE UNIBUS MEMORY XFC CODE
000015                          ECC     =       13.     ;DO ECC ON BUFFER XFC CODE
000016                          MRD     =       14.     ;SEND BUFFER TO MAINTENANCE READ COMMAND
000017                          MWR     =       15.     ;GET BUFFER FROM MAINTENANCE WRITE COMMAND
000020                          CVT     =       16.     ;CONVERT TO PHYSICAL ADDRESS XFC CODE
000021                          EXIT    =       17.     ;TERMINATE DM PROGRAM XFC CODE
                                ;
                                ;       GET STATUS OFFSETS
                                ;
000000                          ST.UNT  =       0.      ;UNIT NUMBER
000000                          ST.MSK  =       0.      ;SUBUNIT MASK
000001                          ST.STA  =       1.      ;STATUS BYTE
000001                          ST.MOD  =       1.      ;MODE BYTE
000002                          ST.ERR  =       2.      ;ERROR BYTE
000002                          ST.CON  =       2.      ;CONTROLLER BYTE
000002                          ST.C    =       2.      ;C BITS
000003                          ST.RTY  =       3.      ;RETRY COUNT/FAILURE CODE
                                ;
                                ;       STATUS BIT DEFINITIONS
                                ;
000200                          ST.OA   =       200     ; ONLINE TO ANOTHER (SET IF DRIVE UNAVAILABLE)
000100                          ST.RR   =       100     ; READJUSTMENT BIT (SET IF RECALIBRATION REQUIRED)
000040                          ST.DR   =       40      ; DIAGNOSTIC REQUEST (SET IF DIAGNOSTIC REQUESTED)
000020                          ST.SR   =       20      ; SPINDLE READY (SET IF SPINDLE READY)
000002                          ST.PS   =       2       ; PORT SWITCH (SET IF PORT SWITCH IN)
000001                          ST.RU   =       1       ; RUN/STOP SWITCH (SET IF RUN/STOP SWITCH IN)
000200                          ST.FE   =       200     ; FATAL ERROR (SET IF FATAL ERROR OCCURRED)
000100                          ST.RE   =       100     ; RETRIABLE ERROR (SET IF RETRIABLE ERROR OCCURRED)
000040                          ST.PE   =       40      ; PROTOCOL ERROR (SET IF PROTOCOL ERROR OCCURRED)
000020                          ST.DF   =       20      ; INITIALIZATION FAILURE (SET IF INIT FAILED)
000010                          ST.WE   =       10      ; WRITE ENABLE (SET IF WRT ATTEMPTED ON PROT DISK)
002000                          ST.FO   =       2000    ; FORMATTING (SET IF FORMATTING ENABLED)
001000                          ST.DB   =       1000    ; DIAGNOSTIC CYLS (SET IF DIAG CYL ACCESS ENABLED)
000400                          ST.S7   =       400     ; SECTOR SIZE (SET FOR 576 BYTE SECTORS)
```

```
                        ;       GET COMMON CHARACTERISTICS OFFSETS
                        ;
000000                  SHRTTO  -       0.              ;SHORT TIMEOUT <3:0>
000000                  SDIVER  =       0.              ;SDI VERSION <7:4>
000000                  XFERRT  =       0.              ;TRANSFER RATE <15:0>
000001                  LONGTO  =       1.              ;LONG TIMEOUT <3:0>
000001                  RETS    =       1.              ;RETRIES <7:4>
000001                  RCTCPS  =       1.              ;F/RCT COPIES <11:8>
000001                  SS      =       1.              ;SECTOR SIZE <15:15>
000002                  ERLEV   =       2.              ;ERROR RETRY LEVELS <7:0>
000002                  ECCRSH  =       2.              ;ECC THRESHOLD <15:8>
000003                  MICREV  =       3.              ;MICROCODE REVISION NUMBER <7:0>
000003                  HRDREV  =       3.              ;HARDWARE REVISION NUMBER <15:8>
000004                  DRVID   =       4.              ;UNIQUE DRIVE ID <47:0>
000007                  DRTYPE  =       7.              ;DRIVE TYPE IDENTIFIER <7:0>
0000C7                  REVS    =       7.              ;REVS/SECOND <15:8>
                        ;
                        ;       GET SUBUNIT CHARACTERISTICS OFFSETS
                        ;
                        ;THESE OFFSETS ARE CURRENTLY GIVEN AS FOLLOWING THE COMMON CHARACTERISTICS
                        ;
000013                  SUB     =       11.             ;OFFSET TO PUT SUBUNIT AFTER COMMON;
000000                  LBNCYL  =       0.              ;NUMBER OF CYLINDERS IN LBN AREA <31:0>
000001                  HICYL           1.              ;HI ORDER CYLINDER BITS <15:12>
000002                  GRPCYL  -       2.              ;GROUPS PER CYLINDER <7:0>
000002                  HILBN   -       2.              ;HI STARTING LBN <11:8>
000002                  HIXBN   -       2.              ;HI STARTING XBN <15:12>
000003                  TRKGRP  -       3.              ;TRACKS PER GROUP <7:0>
000003                  HIRBN   =       3.              ;HI STARTING RBN <11:8>
000003                  HIDBN   -       3.              ;HI STARTING DBN <15:12>
000004                  RBNTRK  -       4.              ;RBNS PER TRACK <6:0>
000004                  RM              4.              ;REMOVABLE MEDIA <7:7> 1=REMOVEABLE
000005                  DATPRE  -       5.              ;DATA PREAMBLE SIZE IN WORDS <7:0>
000005                  HDRPRE  =       5.              ;HEADER PREAMBLE SIZE IN WORDS <15:8>
000006                  MEDTYP          6.              ;MEDIA TYPE <31:0>
000010                  FCTSIZ  =       8.              ;FCT COPY SIZE <15:0>
000011                  LBNTRK  =       9.              ;LBNS PER TRACK <7:0>
000011                  GRPOFF          9.              ;GROUP OFFSET (SECTORS) <15:8>
000012                  LBNHST  -       10.             ;LBNS IN HOST AREA <31:0>
000014                  RCTCSZ  =       12.             ;RCT COPY SIZE <15:0>
000021                  XBNCYL  =       17.             ;CYLS IN XBN AREA <15:0>
000022                  DBNCYL  =       18.             ;CYLS IN DBN AREA <15:8>
                        ;
                        ;       UNIT CODES
                        ;
000001                  UNIT0   -       1.              ;UNIT ZERO CODE
000002                  UNIT1   =       2.              ;UNIT ONE CODE
000004                  UNIT2   =       4.              ;UNIT TWO CODE
000010                  UNIT3   -       8.              ;UNIT THREE CODE
                        ;
                        ;       BIT MASK DEFINITIONS
                        ;
                        ;
177400                  HIBYTE          177400          ;HIGH BYTE MASK
000377                  LOBYTE  -       000377          ;LOW BYTE MASK
007777                  HBHINB          7777            ;HI BYTE, HI NIBBLE MASK
170377                  HBLONB  -       170377          ;HI BYTE, LO NIBBLE MASK
```

JDA DM PROGRAM PARAMETERS

```
      58          177417                LBHINB  =       177417          ;LO BYTE, HI NIBBLE MASK
      59          177760                LBLONB  =       177760          ;LO BYTE, LO NIBBLE MASK
      60                                ;
      61          000001                TIMEOUT =       1.              ;DRIVE TIMEOUT CODE
      62          000002                HEADER  =       2.              ;HEADER COMPARE FAILURE CODE
      63          000004                REVECT  =       4.              ;REVECTOR NEEDED CODE
      64                                ;
      65          000002                WRONG   =       2.              ;FIRST WORD NOT START FRAME CODE
      66          000004                FRAME   =       4.              ;FRAMING ERROR CODE
      67          000010                CHECK   =       8.              ;CHECKSUM ERROR CODE
      68                                ;
      69          000001                TOOBIG  =       1.              ;NUMBER OF WORDS EXCEEDS 7064
      70          000002                LOW     =       2.              ;DM BUFFER ADDRESS IS LESS THAN 714
      71                                ;
      72                                ;       BIT MASK
      73                                ;
      74          000001                LARGE   =       1.              ;BLOCK NUMBER TOO LARGE
      75          000002                OVERFL  =       2.              ;SECTOR NUMBER LARGER THAN 16 BITS
```

```
                        ;MAINTANENCE READ/WRITE REQUEST NUMBERS
                        ;
000000                  T1MSIZ  =       0.              ;GET FREE MEMORY PARAMETERS
000001                  T2DLL   =       1.              ;DOWNLINE LOAD DRIVE DIAGNOSTIC
000002                  T2CMD   =       2.              ;MANUAL INTERVENTION TEST 2 PROTOCOL
000003                  T4MPRM  =       3.              ;GET MASTER PARAMETERS FROM SW QUESTIONS
000004                  T4UPRM  =       4.              ;GET UNIT PARAMETERS FROM HW QUESTIONS
000005                  T4BB1   =       5.              ;GET BAD BLOCKS (1 THRU 14)
000006                  T4BB2   =       6.              ;GET REST OF BAD BLOCKS (15 AND 16)
000007                  T4SOFT  =       7.              ;ADD TO SOFT ERROR AND ECC COUNT
000010                  T4SEEK  =       8.              ;ADD 1 TO SEEK COUNT
000011                  T4MXFR  =       9.              ;ADD TO MEGABITS READ AND WRITTEN
000012                  UTOTST  =       10.             ;GET UNITS TO TEST
000013                  ERRMES  =       11.             ;PRINT ERROR MESSAGE
000014                  ERRMC   =       12.             ;TEST 4 ERROR REPORTING
000015                  MESSAG  =       13.             ;INFORMATION MESSAGE
000016                  DONE    =       14.             ;MARK DM PROGRAM AS NO LONGER RUNNING
                        ;
                        ;       OTHER BIT DEFINITIIONS
                        ;
000001                  RCVRDY  =       1               ; RECIEVER READY 1 - READY
000002                  ATTN    =       2               ; ATTENTION BIT FOR RETURN DRIVE SIGNALS XFC
000004                  RCVERR  =       4               ; RECIEVER ERROR
000100                  AVAIL   =       100             ; AVAILABLE 1 - AVAILABLE
000400                  XMTERR  =       400             ; TRANSMIT ERROR
100000                  RWRDY   =       100000          ; IF SET, UDA IS ABLE TO READ AND/OR WRITE TO DRIVE
                        ;
                        ;       SDI COMMANDS AND RESPONSES
                        ;
000204                  DISCON  =       204             ; DISCONNECT DRIVE
000006                  ERECOV  =       6               ; ERROR RECOVERY
000201                  CHGMOD  =       201             ; CHANGE MODE
000213                  DRVONL  =       213             ; DRIVE ONLINE
000014                  DRVRUN  =       14              ; DRIVE RUN
000005                  DRVCLR  =       5               ; DRIVE CLEAR OPCODE
000207                  GETCHR  =       207             ; GET CHARACTERISTICS
000210                  GETSUB  =       210             ; GET SUBUNIT CHARACTERISTICS
000011                  GETSTA  =       11              ; GET STATUS
000216                  IRECLB  =       216             ; RECALIBRATE
000012                  INSEEK  =       12              ; INITIATE SEEK
000176                  COMPLT  =       176             ; SUCCESSFUL COMPLETION
000175                  UNSSUC  =       175             ; UNSUCCESSFUL COMPLETION
000170                  CHRRES  =       170             ; GET CHARACTERISTICS RESPONSE
000167                  SBCRES  =       167             ; GET SUBUNIT CHARACTERISTICS RESPONSE
000366                  STSRES  =       366             ; GET STATUS RESPONSE
000350                  ECHOC   =       350             ; DIAGNOSTIC ECHO COMMAND AND RESPONSE
                        ;
                        ;       ERROR CODES
                        ;
000000                  FTLSYS  =       0               ; SYSTEM FATAL ERROR
000400                  FTLDEV  =       400             ; DEVICE FATAL
001000                  ERHARD  =       1000            ; HARD ERROR
001400                  ERSOFT  =       1400            ; SOFT ERROR
```

```
                        .SBTTL  TEST 4 SPECIFIC INFORMATION
                        ;
                        ;       TEST 4 SPECIFIC INFORMATION
                        ;
                        ;
                        ;       CONSTANTS
                        ;
000377          SCTWRD  =       255.                    ; NUMBER OF WORDS IN SECTOR TO FI..
000105          INTEDC  =       69.                     ; INITIAL EDC VALUE
000055          TLEN.U  =       U.LGRP+1                ; UNIT PARAMETER LENGTH
007723          FIRSTU  =       HIMEM-TLEN.U            ; LOCATION OF FIRST UNIT PARAMETER BLOCK
                        ;
                        ;       UNIT PARAMETER OFFSETS
                        ;
000000          U.NEXT  =       0.                      ; POINTER TO NEXT UNIT (RING LINKED LIST)
000001          U.SUBP  =       U.NEXT+1                ; 4 WORDS OF SUBUNIT PARAMETER POINTERS
000005          U.TIMO  =       U.SUBP+4                ; AREA TO STORE VARIOUS TIMEOUT VALUES
000006          U.RWTO  =       U.TIMO+1                ; READ/WRITE TIMEOUT AREA
000007          U.SEEK  =       U.RWTO+1                ; NUMBER OF SEEKS ISSUED
000010          U.NFUN  =       U.SEEK+1                ; NEXT FUNCTION ADDRESS (FOR DEFERRED CALLS)
000011          U.PAT   =       U.NFUN+1                ; PATTERN NUMBER TO WRITE
000012          U.CCNT  =       U.PAT+1                 ; CURRENT COUNT OF T/G LOOPS
000014          U.PCTG  =       U.CCNT+2                ; POINTER TO CURRENT TRACK OR GROUP
000015          U.CTRK  =       U.PCTG+1                ; TRACK COUNT FOR GROUP OPERATIONS
000016          U.NSEC  =       U.CTRK+1                ; NUMBER OF SECTORS R/W THIS TRY
000017          U.MSEC  =       U.NSEC+1                ; NUMBER OF SECTORS TO BE R/W
000020          U.TSEC  =       U.MSEC+1                ; NUMBER OF SECTORS TO BE R/W THIS OP
000021          U.CSEC  =       U.TSEC+1                ; COUNT OF SECTORS R/W SO FAR
000022          U.MASK  =       U.CSEC+1                ; UNIT MASK FOR XFC CALLS (0001 - 1000)
000023          U.WRIT  =       U.MASK+1                ; WRITE PROTECTION STATUS
000024          U.ELEV  =       U.WRIT+1                ; CURRENT ERROR RECOVERY LEVEL
000025          U.RTRY  =       U.ELEV+1                ; MAXIMUM NUMBER OF READ RETRIES
000026          U.MLEV  =       U.RTRY+1                ; MAXIMUM NUMBER OF ERROR RECOVERY LEVELS
000027          U.ECCT  =       U.MLEV+1                ; ECC THRESHOLD
000030          U.SDIS  =       U.ECCT+1                ; SDI SHORT TIMEOUT
000031          U.SDIL  =       U.SDIS+1                ; SDI LONG TIMEOUT
000032          U.WPRT  =       U.SDIL+1                ; MASK TO WRITE PROTECT READ-ONLY DRIVES
000033          U.PARM  =       U.WPRT+1                ; UNIT PARAMETER WORD
000034          U.SUBU  =       U.PARM+1                ; SUBUNIT OFFSET (0 - 3)
000035          U.MBN   =       U.SUBU+1                ; MASTER L/DBN
000037          U.CBN   =       U.MBN+2                 ; CURRENT L/DBN FOR START OF CHAIN
000041          U.RBN   =       U.CBN+2                 ; RBN TO BE READ/WRITTEN IF LBN REVECTORED
000043          U.SNUM  =       U.RBN+2                 ; 4 WORDS THAT HOLD THE SUBUNIT LOGICAL NUMBERS
000047          U.CCYL  =       U.SNUM+4                ; CURRENT CYLINDER
000051          U.CGRP  =       U.CCYL+2
000052          U.LCYL  =       U.CGRP+1                ; LAST CYLINDER SEEKED TO
000054          U.LGRP  =       U.LCYL+2                ; LAST GROUP SEEKED TO
                        ;
                        ;       SUBUNIT PARAMETER OFFSET
                        ;
000000          S.PARM  =       0.                      ; SUBUNIT PARAMETER WORD
000001          S.SDCL  =       S.PARM+1                ; STARTING DIAGNOSTIC CYLINDER
000003          S.PAT   =       S.SDCL+2                ; PATTERN TO USE FOR WRITES
000004          S.TRKL  =       S.PAT+1                 ; NUMBER OF SECTORS IN ONE TRACK
000005          S.SCHR  =       S.TRKL+1                ; POINTER TO SUBUNIT CHARACTERISTICS
000006          S.MEGR  =       S.SCHR+1                ; SECTORS READ (UP TO 245)
000007          S.MEGW  =       S.MEGR+1                ; SECTORS WRITTEN (UP TO 245)
```

```
      58          00001C                S.BADP  =       S.MEGW+1                ; POINTER TO BAD BLOCK AREA
      59          000011                S.BESS  =       S.BADP+1                ; START OF BEGIN/END SETS
      60                                ;
      61                                ;       IF TRACK/GROUP LIMITS ARE GIVEN, THE SUBUNIT PARAMETERS HAVE THE
      62                                ;       FOLLOWING FIELDS ADDED TO THEM
      63                                ;
      64          000011                S.MCNT  =       S.BESS                  ; MAXIMUM TRACK/GROUP COUNT
      65          000013                S.TGOF  =       S.MCNT+2                ; ORIGINAL TRACK/GROUP OFFSET
      66          000015                S.TGSS  =       S.TGOF+2                ; START OF TRACK/GROUP SETS
```

```
                                ;
                                ;       DUMMY SDI CONTROL BLOCK OFFSETS
                                ;
000001                          D.LIMT  =       1               ; DUMMY SDI SEARCH LIMIT
000002                          D.SCHR  =       2               ; DUMMY POINTER TO SUBUNIT CHAR-5
                                ;
                                ;
                                ;       UNIT PARAMETER BITS (U.PARM(R5))
                                ;
100000                          DROP    =       100000          ; DROP BIT (SET IF UNIT OR SUBUNIT DROPPED)
040000                          INITW   =       40000           ; INITIAL WRITE (SET IF INITIAL WRITE IN PROG)
020000                          RESEEK  =       20000           ; IF 1, INDICATES THAT A SEEK IS NECESSARY
010000                          DIREC   =       10000           ; DIRECTION (SET IF SEQUENTIAL ACCESSES DECREASING)
004000                          NEWSUB  =       4000            ; SET IF SEQUENTIAL SEEKS MOVED TO NEW SUBUNIT
002000                          SEKINP  =       2000            ; SEEK IN PROGRESS - SET IF TRUE
001000                          FTIME   =       1000            ; FIRST TIME FLAG - SET FOR INI  CODE
000400                          REVEC   =       400             ; REVECTOR BIT (SET IF BLOCK REVECTORED)
000200                          RBNBN   =       200             ; SEE IF WORKING ON RBN
000100                          REDWRT  =       100             ; REDWRT (READ OR WRITE IN PROGRESS SET IF WRITE)
000020                          DATERR  =       20              ; DATA ERROR IF SET
000004                          RETRY   =       4               ; IF CLEAR, START RETRIES AT ZERO
                                ;
                                ;       SUBUNIT PARAMETER BITS (S.PARM(R4))
                                ;
020000                          DCYLS   =       20000           ; DIAGNOSTIC CYLINDER FLAG (SET IF DBNS)
010000                          ECCCHK  =       10000           ; 1 IF ECC CORRECTION ALLOWED   MASTER BITS
004000                          RONLY   =       4000            ; READ ONLY (SET IF READ ONLY)
002000                          WONLY   =       2000            ; WRITE ONLY (SET IF WRITE ONLY)
001000                          RTRIES  =       1000            ; 1 IF RETRIES ALLOWED
000200                          ONLYCL  =       200             ; SET IF ONLY CYLINDERS SPECIFIED
                                                                ; USED DURING SETUP ONLY
000100                          SEQSEK  =       100             ; SEQUENTIAL SEEK (START UP TESTING ONLY)
000040                          BEUSED  =       40              ; BEGIN/END SETS (USED IF SET)
000020                          TRACKS  =       20              ; TRACKS OR GROUPS (TRACK IF SET)
000010                          WCHECK  =       10              ; WRITE CHECK BIT (IF SET, WRITE CHECK WILL BE DONE)
                                ;                       WCHECK IS ALSO USED FOR THE UNIT PARAMETERS
000004                          WCHKAL  =       4               ; SET IF WRITE CHECK ALWAYS TO BE DONE
000002                          DATCMP  =       2               ; DATA COMPARE (SET IF DATA COMPARE TO BE DONE)
                                ;                       DATCMP IS ALSO USED FOR THE UNIT PARAMETERS
000001                          DCMPAL  =       1               ; SET IF DATA COMPARE ALWAYS TO BE DONE
```

MACRO DEFINITIONS

```
                        .SBTTL  MACRO DEFINITIONS
                        ;
                        ;       DIAGNOSTIC MACRO FOR TEST4 OVERLAYS
                        ;
                                .MACRO  DIAG$$
                                TST     $$DIAG+$DIAG$
                                BEQ     .+6
                                MOV     #60000,R0
                                MOV     R0,@$$DIAG+$DIAG$
                                BR      .+1
                        $DIAG$  =       $DIAG$ + 1
                                .ENDM
```

MACRO DEFINITIONS

```
                        ;       MESSAGE CONTROL TABLE MACRO
                        ;
                                .MACRO MSG CMDBUF,CMDSZ,RPLBUF,RPLSZ,SUCCOM
                                .WORD CMDBUF                    ;ADDRESS OF COMMAND
                                .WORD CMDSZ                     ;SIZE OF COMMAND IN BYTES
                                .WORD RPLBUF                    ;ADDRESS OF REPLY
                                .WORD RPLSZ                     ;SIZE OF REPLY IN WORDS
                                .IF NB NUMBER
                                .WORD SUCCOM                    ; SUCCESSFUL COMPLETION CODE
                                .ENDC
                                .ENDM
```

MACRO DEFINITIONS

```
      1                                .MACRO  BCS     LAB..
      2                                                        BCC     .+2
      3                                                        BR      LAB..
      4                                .ENDM
```

MACRO DEFINITIONS

```
                        ;       PUSH REGISTER MACRO
                        ;
                                .MACRO PUSH R9
                                .IRP X,<R9>
                                                        MOV X,-(SP)
                                .ENDR
                                .ENDM
                        ;
                        ;       POP REGISTER MACRO
                        ;
                                .MACRO POP R9
                                .IRP X,<R9>
                                                        MOV (SP)+,X
                                .ENDR
                                .ENDM
```

```
                        ;ERROR MACROS
                        ;THESE MACROS ARE CALLED TO REPORT ERRORS TO THE HOST PROGRAM.
                        ;THE MACRO NAMES ARE : ERRSF, ERRDF, ERRHRD, ERRSFT. EACH RESULTS IN THE HOST
                        ;BEING REQUESTED TO REPORT THE ERROR.
                        ;ARGUMENTS:     1 (MS$) MESSAGE POINTER
                        ;               2 (P1$) PARAMETER #1
                        ;               3 (P2$) PARAMETER #2
                        ;               4 (P3$) PARAMETER #3
                        ;               5 (P4$) PARAMETER #4
                        ;               6 (P5$) PARAMETER #5
                        ;               7 (P6$) PARAMETER #6
                        ;               8 (P7$) PARAMETER #7
                        ;               9 (P8$) PARAMETER #8
                        ;
                        ;THE MESSAGE POINTER MUST POINT TO AN ADDRESS IN THE OVERLAY 'MS' IMMEDIATELY
                        ;FOLLOWING THE MAIN CODE. ANY ADDRESS MODE MAY BE USED (E.G. #MS1, @R2).
                        ;THE ADDRESS MUST CONTAIN AN ASCII FORMAT STRING TO DETERMINE THE MESSAGE
                        ;TO PRINT.
                        ;THE PARAMETER ARGUMENTS ARE OPTIONAL. THEY SHOULD BE SUPPLIED ONLY WHEN
                        ;THERE IS DATA TO BE PASSED TO THE HOST THAT WILL BE USED IN PRINTING THE
                        ;MESSAGE. THESE PARAMETER ARGUMENTS ARE THE ADDRESS OF DATA TO BE PASSED
                        ;USING ANY ADDRESSING MODE DESIRED.
                        ;ALL REGISTERS ARE RETURNED UNCHANGED. IT SHOULD BE NOTED THAT ARGUMENTS
                        ;CONTAINING SOMETHING OTHER THAN A REGISTER NAME (E.G. #100 OR MEMADR)
                        ;ASSEMBLE TO INSTRUCTIONS THAT SAVE AND RESTORE A REGISTER ON THE STACK.

                                .MACRO ERRSF MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ 0,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .ENDM

                                .MACRO ERRDF MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ 1,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .ENDM

                                .MACRO ERRHRD MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NLIST
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ 2,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .LIST
                                .ENDM

                                .MACRO ERRSFT MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ 3,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .ENDM
```

MACRO DEFINITIONS

```
                        ;THE FOLLOWING MACRO ACTUALLY PROCESSES THE ERROR CALL TO THE HOST PROGRAM

                        .MACRO ERROR$ ET$,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,ERRN$
                          .RADIX 8
                          PRMS=ARGS$-1
                          .IF LT,<PRMS>
                            .ERROR;NOT ENOUGH ARGUMENTS IN ERROR CALL
                          .ENDC
                          REGS$=-1
                          .IIF GE,<PRMS-8.>,PARGS. P8$
                          .IIF GE,<PRMS-7.>,PARGS. P7$
                          .IIF GE,<PRMS-6.>,PARGS. P6$
                          .IIF GE,<PRMS-5.>,PARGS. P5$
                          .IIF GE,<PRMS-4.>,PARGS. P4$
                          .IIF GE,<PRMS-3.>,PARGS. P3$
                          .IIF GE,<PRMS-2.>,PARGS. P2$
                          .IIF GE,<PRMS-1.>,PARGS. P1$
                          .IF GE REGS$
                            RSTR$ \REGS$
                          .ENDC
                          .RADIX 10
                          .LIST
                                                                        CALL RERROR     ;ERROR # ERRN$'.
                          .NLIST
                          .RADIX 8
                          .LIST
                                                                        .WORD <PRMS*2000>+<ET$*400>+ERRN
                                                                        .WORD MS$
                          .NLIST
                          ERRN=ERRN+1
                        .ENDM

                        .MACRO PARGS.,ADDR$
                          .NTYPE PTYPE$,ADDR$
                          .IF EQ,<PTYPE$&70>
                            .IIF EQ,<PTYPE$&7>-REGS$,RSTR$ \REGS$
                            .LIST
                                                                          MOV ADDR$,-(SP)
                            .NLIST
                          .IFF
                            .IF EQ,<PTYPE$&7>-1         ;PICK A REGISTER TO USE
                              REGU$=2                   ;SELECT R2 IF R1 IS USED IN PARAMETER FETCH
                            .IFF
                              REGU$=1                   ;OTHERWISE USE R1
                            .ENDC
                            .IF NE,<REGU$-REGS$>        ;IF REGISTER NOT ALREADY SAVED
                              .IF GE,REGS$
                                RSTR$ \REGS$            ;RESTORE CURRENT SAVED REGISTER
                              .ENDC
                              SAVR$ \REGU$              ;THEN SAVE SELECTED REGISTER
                            .ENDC
                            GETP$ \REGS$,ADDR$
                          .ENDC
                        .ENDM
```

MACRO DEFINITIONS

```
                        .MACRO SAVR$ REGN
                          .LIST
                                                        MOV R'REGN,SAVREG
                          .NLIST
                          REGS$=REGN
                        .ENDM

                        .MACRO RSTR$ REGN
                          .LIST
                                                        MOV SAVREG,R'REGN
                          .NLIST
                          REGS$=-1
                        .ENDM

                        .MACRO GETP$ REGN,ADDR$
                          .LIST
                                                        MOV ADDR$,R'REGN
                                                        MOV R'REGN,-(SP)
                          .NLIST
                        .ENDM
```

MACRO DEFINITIONS

```
                        ;
                        ;       PRIMARY ERROR REPORTING (TEST 4)
                        ;
                        .MACRO  SOFTER  NUM,ARGS
                        ERROR   #ERSOFT,NUM,<ARGS>
                                                        MOV     #ERRMES,R2
                                                        MOV     R2,OUT.RQ
                        .ENDM

                        .MACRO  REPSFT  SFTFLG,ECCFG
                        .IF     NB,SFTFLG
                                                        MOV     #1,OUT.02
                        .ENDC
                        .IF     NB,ECCFG
                                                        MOV     #1,OUT.03
                        .ENDC
                                                        PUSH    R0
                                                        MOV     #U.SNUM,R0
                                                        ADD     R5,R0
                                                        ADD     U.SUBU(R5),R0
                                                        MOV     (R0),OUT.01
                                                        MOV     #T4SOFT,R0
                                                        CALL    HOSTRQ
                                                        POP     R0
                        .ENDM

                        .MACRO  HARDER  NUM,ARGS
                        ERROR   #ERHARD,NUM,<ARGS>
                                                        MOV     #ERRMC,R2
                                                        MOV     R2,OUT.RQ
                        .ENDM

                        .MACRO  DEVFTL  NUM,ARGS
                        ERROR   #FTLDEV,NUM,<ARGS>
                                                        MOV     #ERRMC,R2
                                                        MOV     R2,OUT.RQ
                        .ENDM

                        .MACRO  SYSFTL  NUM,ARGS
                        ERROR   #FTLSYS,NUM,<ARGS>
                                                        MOV     #ERRMC,R2
                                                        MOV     R2,OUT.RQ
                        .ENDM

                        .MACRO  ERROR   TYPE,NUM,ARGS
                        .RADIX  10
                        NUMPTR  =       5
                        MOVMSG  #ER'NUM,4
                        .IF     NB,<ARGS>
                        .IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
                        NUMPTR  =       NUMPTR + 1
                        .ENDR
                        .ENDC
                                                        MOV     #NUM,OUT.02
                                                        BIS     TYPE,OUT.02
                                                        MOV     #..,OUT.01
```

```
                                        .RADIX
                                        .ENDM

                                        .MACRO  CERROR  STNUM,ARGS
                                        .RADIX  10
                                NUMPTR  =       STNUM
                                        .IRP    X,<ARGS>
                                        MOVMSG  X,\NUMPTR
                                NUMPTR          NUMPTR + 1
                                        .ENDR
                                        .RADIX
                                        .ENDM

                                        .MACRO  ERRORC  ARGS
                                        .RADIX  10
                                        .IRP    X,<ARGS>
                                        MOVMSG  X,\NUMPTR
                                NUMPTR  =       NUMPTR + 1
                                        .ENDR
                                        .RADIX
                                        .ENDM

                                        .MACRO  MOVMSG  ARG,INDX

                                        .IF     LT,INDX-10
                                                                        MOV     ARG,OUT.0'INDX
                                        .IFF
                                                                        MOV     ARG,OUT.'INDX
                                        .ENDC
                                        .ENDM

                                        .MACRO  ENDERR  POS
                                        .IF     NB,POS
                                        NDERR   POS
                                        .IFF
                                        NDERR   \NUMPTR
                                        .ENDC
                                        .ENDM

                                        .MACRO  NDERR   POS
                                        .IF     NE,POS
                                                                        BIS     #POS,ERRPOS     ; SET THE POSITION
                                        .IFF
                                                                        CLR     ERRPOS          ; CLEAR THE POSITION
                                        .ENDC
                                        .ENDM

                                ;       MESSAGE REPORTING MACRO
                                ;

                                        .MACRO  MSSG    NUM,ARGS
                                        .RADIX  10
                                        NUMPTR  -       3
                                        MOVMSG  #MS'NUM,2
                                        .IF     NB,<ARGS>
                                        .IRP    X,<ARGS>
                                        MOVMSG  X,\NUMPTR
                                        NUMPTR  -       NUMPTR + 1
```

```
                                .ENDR
                                .ENDC
                                                        PUSH    <R0,R1>
                                                        MOV     #U.SNUM,R1
                                                        ADD     R5,R1
                                                        ADD     U.SUBU(R5),R1
                                                        MOV     (R1),OUT.01
                                                        MOV     #MESSAG,R0
                                                        CALL    HOSTRQ
                                                        POP     <R1,R0>
                                .RADIX
                                .ENDM
                                ;
                                ;
                                .MACRO  MSSGE   NUM,ARGS
                                .RADIX  10
                                NUMPTR          3
                                MOVMSG  #'NUM,2
                                .IF     NB,<ARGS>
                                .IRP    X,<ARGS>
                                MOVMSG  X,\NUMPTR
                                NUMPTR  =       NUMPTR + 1
                                .ENDR
                                .ENDC
                                                        PUSH    <R0,R1>
                                                        MOV     #MESSAG,R0
                                                        CALL    HOSTRQ
                                                        POP     <R1,R0>
                                .RADIX
                                .ENDM
```

MACRO DEFINITIONS

```
                        ;RETURN DRIVE STATUS MACRO WITH ERROR REPORTING

                                .MACRO  DSTAT,LAB$,E1,E2
                                .NLIST
                                .NLIST  MEB
                                .LIST   MF
                                .LIST
                                CALL    RDSTAT                  ; GET DRIVE STATUS
                                BIT     #10000,R1               ; SEE IF ANY ERRORS
                                BEQ     2$                      ; IF NO ERROR, BRANCH
                                BIT     #4000,R1                ; SEE IF XMIT ERROR
                                BEQ     1$                      ; IF SO, BRANCH
                                ERRHRD  E1                      ; REPORT INVALID STATUS ERROR
                                BR      LAB$                    ; BRANCH TO DONE
                        1$:     ERRHRD  E2                      ; REPORT XMIT ERROR
                                BR      LAB$                    ; BRANCH TO DONE
                        2$:
                                .NLIST
                                .NLIST  ME
                                .LIST   MEB
                                .LIST
                                .ENDM
```

MACRO DEFINITIONS

```
                        ;
                        ;       XOR THE CONTENTS OF TWO REGISTERS
                        ;
                        .MACRO  RXOR    REG1,REG2
                        MOV     REG2,-(SP)              ; SAVE REGISTER REG2
                        BIC     REG1,REG2               ; CLEAR COORESPONDING BITS IN REG2
                        BIC     (SP)+,REG1              ; CLEAR COORESPONDING BITS IN REG1
                        BIS     REG1,REG2               ; OR WHAT'S LEFT
                        .ENDM
```

MACRO DEFINITIONS

```
                        ;       SDI INTERCHANGE WITH DRIVE WITH ERROR REPORTING
                        
                                .MACRO  TALKX   ERRLAB,E1,E2
                                .NLIST
                                .NLIST  MEB
                                .LIST   MF
                                .LIST
                                CALL    TALKER                  ; INITIATE SDI INTERCHANGE
                                TST     R3                      ; SEE IF ERROR OCCURRED
                                BEQ     12$                     ; IF NOT, BRANCH
                                BPL     11$                     ; IF SO, BRANCH
                                ERRHRD  E1;SEND COMMAND ERROR
                                BR      ERRLAB
                        11$:    ERRHRD  E2;RECEIVE COMMAND ERROR
                                BR      ERRLAB
                        12$:
                                .NLIST
                                .NLIST  ME
                                .LIST   MEB
                                .LIST
                                .ENDM
```

START OF TEST CODE

```
                                .SBTTL  START OF TEST CODE
                                ;THE FOLLOWING IS FOR DEBUG PURPOSES ONLY. CAN BE NOP OR BREAKPOINT.
                                
000714  114007                        CLR R0                          ;CHANGE TO BREAKPOINT FOR DEBUG
                                
                                ;INITIALIZE STACK
                                
000715  104206  001357                 MOV #STACK,SP                   ;SET UP STACK POINTER
000717  001360                        BR      START                   ; BRANCH OVER SUPPORT CODE
```

```
                                .SBTTL  RDSTAT - GET DRIVE'S REAL TIME DRIVE STATE
000720                          RDSTAT:
                                ;
                                ;RETURN DRIVE STATUS
                                ;STATUS RETURNED IN DM REGISTER 1
                                ;
000720                                  PUSH    <R3,R0>                 ; SAVE R3 AND R0
000720  100463                                                                  MOV R3,-(SP)
000721  100467                                                                  MOV R0,-(SP)
000722  104203  000003                 MOV     #3,R3                   ; ALLOW ONLY 3 ERRORS
000724  060007                  STATLP: XFC     STATUS                  ; GET DRIVE'S STATUS
000725  103201  014000                  BIC     #14000,R1               ; CLEAR ERROR PASSING BITS
000727  102201  000400                  BIT     #XMTERR,R1              ; CHECK XMIT ERRORS
000731  010737                          BEQ     STATOK                  ; IF NO ERRORS, BRANCH
000732  117403                          DEC     R3                      ; DECREMENT TRANSMIT ERROR COUNT
000733  050724                          BNE     STATLP                  ; IF ERROR COUNT INCOMPLETE, BRANCH
000734  104201  010000                  MOV     #10000,R1               ; FLAG AS TRANSMIT ERROR
000736  000746                          BR      STATEX                  ; BRANCH
000737  102201  000004          STATOK: BIT     #RCVERR,R1              ; RECIEVER ERRORS
000741  050746                          BNE     STATEX                  ; IF VALID, BRANCH
000742  117403                          DEC     R3                      ; DECREMENT ERROR COUNT
000743  050724                          BNE     STATLP                  ; IF ERROR COUNT NON-ZERO, BRANCH
000744  104201  014000                  MOV     #14000,R1               ; FLAG AS INVALID STATUS ERROR
000746                          STATEX: POP     <R0,R3>                 ; RESTORE R0, R3
000746  104267                                                                  MOV (SP)+,R0
000747  104263                                                                  MOV (SP)+,R3
000750  000000                          RETURN                          ; RETURN TO CALLING MODULE
```

```
                                .SBTTL  HOSTRQ - HOST REQUEST - REPORT ERRORS, MEGABYTES TRANSFERRED, ETC.
000751                          HOSTRQ:
                                ;
                                ;SEND REQUEST BUFFER TO HOST AND WAIT FOR RESPONSE.
                                ;CLEAR ARGUMENT AREA OF OUT BUFFER IN PREPARATION
                                ;FOR NEXT HOSTRQ CALL.
                                ;
                                ;INPUTS:
                                ;       R0 - HOST REQUEST NUMBER
                                ;       OUT BUFFER LOADED WITH DATA

000751                                  PUSH <R0,R1,R2>
000751  100467                                                                  MOV R0,-(SP)
000752  100461                                                                  MOV R1,-(SP)
000753  100462                                                                  MOV R2,-(SP)
000754  104070  001006                 MOV R0,OUT.RQ                   ;STORE REQUEST NUMBER IN BUFFER
000756  104207  001006          SNDAGN: MOV #OUT.RQ,R0                  ;SEND BUFFER TO HOST
000760  104201  000036                  MOV #BUFSIZ,R1
000762  060016                          XFC MRD
000763  115001                          TST R1                          ;CHECK FOR ERROR
000764  050756                          BNE SNDAGN                      ;IF ERROR, TRY AGAIN
000765  104207  001044                  MOV #IN.RQ,R0                   ;WAIT FOR RESPONSE FROM HOST
000767  104201  000036                  MOV #BUFSIZ,R1
000771  060017                          XFC MWR
000772  104207  001007                  MOV #OUT.01,R0                  ;CLEAR ARGUMENT WORDS IN BUFFER
000774  104201  000012                  MOV #OUT.11-OUT.01,R1
000776  114002                          CLR R2
000777  100272          CLRBUF: MOV R2,(R0)+
001000  117401                          DEC R1
001001  030777                          BPL CLRBUF
001002                                  POP <R2,R1,R0>
001002  104262                                                                  MOV (SP)+,R2
001003  104261                                                                  MOV (SP)+,R1
001004  104267                                                                  MOV (SP)+,R0
001005  000000                          RETURN
```

```
                                ;STORAGE AREA FOR MAINTENANCE WRITE AND READ BUFFERS
                                ;
                                ;OUT BUFFER - DATA TO SEND TO HOST
                                ;
001006  000000  OUT.RQ: .WORD 0                 ;HOST REQUEST CODE
001007  000000  OUT.01: .WORD 0                 ;DATA ARGUMENT  1
001010  000000  OUT.02: .WORD 0                 ;DATA ARGUMENT  2
001011  000000  OUT.03: .WORD 0                 ;DATA ARGUMENT  3
001012  000000  OUT.04: .WORD 0                 ;DATA ARGUMENT  4
001013  000000  OUT.05: .WORD 0                 ;DATA ARGUMENT  5
001014  000000  OUT.06: .WORD 0                 ;DATA ARGUMENT  6
001015  000000  OUT.07: .WORD 0                 ;DATA ARGUMENT  7
001016  000000  OUT.08: .WORD 0                 ;DATA ARGUMENT  8
001017  000000  OUT.09: .WORD 0                 ;DATA ARGUMENT  9
001020  000000  OUT.10: .WORD 0                 ;DATA ARGUMENT 10
001021  000000  OUT.11: .WORD 0                 ;DATA ARGUMENT 11
001022  000000  OUT.12: .WORD 0                 ;DATA ARGUMENT 12
001023  000000  OUT.13: .WORD 0                 ;DATA ARGUMENT 13
001024  000000  OUT.14: .WORD 0                 ;DATA ARGUMENT 14
001025  000000  OUT.15: .WORD 0                 ;DATA ARGUMENT 15
001026  000000  OUT.16: .WORD 0                 ;DATA ARGUMENT 16
001027  000000  OUT.17: .WORD 0                 ;DATA ARGUMENT 17
001030  000000  OUT.18: .WORD 0                 ;DATA ARGUMENT 18
001031  000000  OUT.19: .WORD 0                 ;DATA ARGUMENT 19
001032  000000  OUT.20: .WORD 0                 ;DATA ARGUMENT 20
001033  000000  OUT.21: .WORD 0                 ;DATA ARGUMENT 21
001034  000000  OUT.22: .WORD 0                 ;DATA ARGUMENT 22
001035  000000  OUT.23: .WORD 0                 ;DATA ARGUMENT 23
001036  000000  OUT.24: .WORD 0                 ;DATA ARGUMENT 24
001037  000000  OUT.25: .WORD 0                 ;DATA ARGUMENT 25
001040  000000  OUT.26: .WORD 0                 ;DATA ARGUMENT 26
001041  000000  OUT.27: .WORD 0                 ;DATA ARGUMENT 27
001042  000000  OUT.28: .WORD 0                 ;DATA ARGUMENT 28
001043  000000  OUT.29: .WORD 0                 ;DATA ARGUMENT 29

                                ;IN BUFFER - DATA RECEIVED FROM HOST
                                ;
001044  000000  IN.RQ:  .WORD 0                 ;HOST REQUEST CODE (ECHO)
001045  000000  IN.01:  .WORD 0                 ;DATA ARGUMENT  1
001046  000000  IN.02:  .WORD 0                 ;DATA ARGUMENT  2
001047  000000  IN.03:  .WORD 0                 ;DATA ARGUMENT  3
001050  000000  IN.04:  .WORD 0                 ;DATA ARGUMENT  4
001051  000000  IN.05:  .WORD 0                 ;DATA ARGUMENT  5
001052  000000  IN.06:  .WORD 0                 ;DATA ARGUMENT  6
001053  000000  IN.07:  .WORD 0                 ;DATA ARGUMENT  7
001054  000000  IN.08:  .WORD 0                 ;DATA ARGUMENT  8
001055  000000  IN.09:  .WORD 0                 ;DATA ARGUMENT  9
001056  000000  IN.10:  .WORD 0                 ;DATA ARGUMENT 10
001057  000000  IN.11:  .WORD 0                 ;DATA ARGUMENT 11
001060  000000  IN.12:  .WORD 0                 ;DATA ARGUMENT 12
001061  000000  IN.13:  .WORD 0                 ;DATA ARGUMENT 13
001062  000000  IN.14:  .WORD 0                 ;DATA ARGUMENT 14
001063  000000  IN.15:  .WORD 0                 ;DATA ARGUMENT 15
001064  000000  IN.16:  .WORD 0                 ;DATA ARGUMENT 16
001065  000000  IN.17:  .WORD 0                 ;DATA ARGUMENT 17
001066  000000  IN.18:  .WORD 0                 ;DATA ARGUMENT 18
001067  000000  IN.19:  .WORD 0                 ;DATA ARGUMENT 19
```

```
    58 001070  000000C                IN.20:  .WORD 0                 ;DATA ARGUMENT 20
    59 001071  000000                 IN.21:  .WORD 0                 ;DATA ARGUMENT 21
    60 001072  000000                 IN.22:  .WORD 0                 ;DATA ARGUMENT 22
    61 001073  000000                 IN.23:  .WORD 0                 ;DATA ARGUMENT 23
    62 001074  000000                 IN.24:  .WORD 0                 ;DATA ARGUMENT 24
    63 001075  000000                 IN.25:  .WORD 0                 ;DATA ARGUMENT 25
    64 001076  000000                 IN.26:  .WORD 0                 ;DATA ARGUMENT 26
    65 001077  000000                 IN.27:  .WORD 0                 ;DATA ARGUMENT 27
    66 001100  000000                 IN.28:  .WORD 0                 ;DATA ARGUMENT 28
    67 001101  000000                 IN.29:  .WORD 0                 ;DATA ARGUMENT 29
    68         000036                 BUFSIZ  =       . - IN.RQ       ;SIZE OF BUFFER
```

```
                                        .IF     NE,TEST4
                                        .NLIST
                                        .IFF
                                ;
                                ;       TALKER SENDS AND RECEIVES AN SDI INTERCHANGE
                                ;
                                ;INPUTS:
                                ;       R2 - SDI INTERCONNECT
                                ;       R3 - POINTER TO COMMAND TABLE CONTAINING APPROPRIATE COMMAND
                                ;       SDILTO/SDISTO SDI LONG AND SHORT TIMEOUTS, RESPECTIVELY
                                ;OUTPUTS:
                                ;       R0 - RETURN OP CODE FROM UNIT
                                ;       R3 - ERROR CODE - 0 - NO ERROR, 1 = RECEIVE ERROR, 100001 = SEND ERROR
                                ;
001102                          TALKER: PUSH    <R1,R4>         ; SAVE REGISTERS
001102  100461                                                                  MOV R1,-(SP)
001103  100464                                                                  MOV R4,-(SP)
001104  104237                          MOV     (R3)+,R0        ; SET ADR OF SDI COMMAND BUFFER
001105  104231                          MOV     (R3)+,R1        ; SET BUFFER LENGTH
001106  060004                          XFC     SEND            ; SEND COMMAND
001107  115001                          TST     R1              ; DID UNIT ACCEPT COMMAND
001110  011114                          BEQ     TALK1A          ; IF SO, BRANCH
001111  104203  100001                  MOV     #100001,R3      ; FLAG AS SEND ERROR
001113  001145                          BR      TALK2B          ; BRANCH TO EXIT
001114  106203  004205          TALK1A: CMP     #LONG,R3        ; SEE IF LONG TIMEOUT TO BE USED
001116  071122                          BMI     1$              ; IF SO, BRANCH
001117  104304  001150                  MOV     SDISTO,R4       ; SET UP SHORT TIMEOUT
001121  001124                          BR      TALK1B          ; BRANCH
001122  104304  001151          1$:     MOV     SDILTO,R4       ; SET UP LONG TIMEOUT
001124  104137                  TALK1B: MOV     (R3),R0         ; SET DATA BUFFER ADDRESS
001125  104631  000001                  MOV     1(R3),R1        ; SET BUFFER LENGTH
001127  060005                          XFC     RCV             ; SEND RECEIVE SDI COMMAND
001130  115001                          TST     R1              ; DID ERROR OCCUR
001131  011144                          BEQ     TALK2A          ; IF NOT, BRANCH
001132  106201  000001                  CMP     #1,R1           ; SEE IF TIMEOUT
001134  011137                          BEQ     1$              ; IF SO, BRANCH
001135  104013                          MOV     R1,R3           ; MOVE ERROR TYPE TO R3 FOR REPORTING
001136  001145                          BR      TALK2B          ; EXIT
001137  117404                  1$:     DEC     R4              ; DECREMENT TIMEOUT VALUE
001140  051124                          BNE     TALK1B          ; IF NOT TIMEOUT, BRANCH
001141  104203  000001                  MOV     #1,R3           ; FLAG AS RECIEVE ERROR
001143  001145                          BR      TALK2B          ; BRANCH TO EXIT
001144  114003                  TALK2A: CLR     R3              ; FLAG AS NO ERRORS
001145                          TALK2B: POP     <R4,R1>         ; RESTORE R4, R1
001145  104264                                                                  MOV (SP)+,R4
001146  104261                                                                  MOV (SP)+,R1
001147  000000                          RETURN
                                ;
001150  000012                  SDISTO: .WORD   10.             ; SDI SHORT TIMEOUT
001151  000024                  SDILTO: .WORD   20.             ; SDI LONG TIMEOUT
```

```
                                ;MREAD
                                ;
                                ;READ ONE WORD FROM UNIBUS MEMORY
                                ;INPUTS:
                                ;       R5 - ADDRESS OF WORD TO READ (LOW 16 BITS)
                                ;       R4 -    ''          (HIGH 2 BITS)
                                ;OUTPUTS:
                                ;       R0 - DATA READ
                                ;
001152                          MREAD:  PUSH <R2,R3>                    ;SAVE SOME REGISTERS
001152  100462                                                                  MOV R2,-(SP)
001153  100463                                                                  MOV R3,-(SP)
001154  104057                          MOV R5,R0                       ;PUT UNIBUS ADDRESS IN R0 AND R1
001155  104041                          MOV R4,R1
001156  104202  000001                 MOV #1,R2                       ;TRANSFER ONE WORD
001160  104203  001210                 MOV #UDATA,R3                   ;LOCATION TO PUT DATA
001162  060013                          XFC UREAD                       ;DO THE READ
001163  104307  001210                 MOV UDATA,R0                    ;GET DATA READ
001165                                  POP <R3,R2>                     ;RESTORE OTHER REGISTERS
001165  104263                                                                  MOV (SP)+,R3
001166  104262                                                                  MOV (SP)+,R2
001167  000000                          RETURN
```

```
                                ;MWRITE
                                ;
                                ;WRITE ONE WORD TO UNIBUS MEMORY
                                ;INPUTS:
                                ;       R5 - ADDRESS OF WORD TO WRITE (LOW 16 BITS)
                                ;       R4 -    "            (HIGH 2 BITS)
                                ;       R0 - DATA TO WRITE
                                ;OUTPUTS:

001170                          MWRITE: PUSH <R0,R2,R3>                 ;SAVE SOME REGISTERS
001170  100467                                                                  MOV R0,-(SP)
001171  100462                                                                  MOV R2,-(SP)
001172  100463                                                                  MOV R3,-(SP)
001173  104070  001210                  MOV R0,UDATA                   ;PUT DATA TO BE WRITTEN INTO BUFFER
001175  104057                          MOV R5,R0                      ;PUT UNIBUS ADDRESS IN R0 AND R1
001176  104041                          MOV R4,R1
001177  104202  000001                  MOV #1,R2                      ;TRANSFER ONE WORD
001201  104203  001210                  MOV #UDATA,R3                  ;ADDRESS OF DATA WORD
001203  060014                          XFC UWRITE                     ;DO THE WRITE
001204                                  POP <R3,R2,R0>                 ;RESTORE THE REGISTERS
001204  104263                                                                  MOV (SP)+,R3
001205  104262                                                                  MOV (SP)+,R2
001206  104267                                                                  MOV (SP)+,R0
001207  000000                          RETURN

001210  000000                  UDATA:  .WORD 0                        ;DATA BUFFER FOR TRANSFERS TO AND FROM
                                                                       ;UNIBUS MEMORY
```

```
                                ;XOR
                                ;
                                ;PERFORM XOR LOGIC FUNCTION ON TWO REGISTERS
                                ;INPUTS:
                                ;       R1, R2 - DATA TO BE XOR'ED
                                ;OUTPUTS:
                                ;       R1 - UNCHANGED
                                ;       R2 - XOR OF TWO INPUTS
                                ;
001211                          XOR:    PUSH R3                         ;SAVE R3
001211  100463                                                                  MOV R3,-(SP)
001212  104013                          MOV R1,R3
001213  103023                          BIC R2,R3
001214  103012                          BIC R1,R2
001215  101032                          BIS R3,R2
001216                          POP R3                                  ;RESTORE R3
001216  104263                                                                  MOV (SP)+,R3
001217  115002                          TST R2                          ;SET CONDITION CODES
001220  000000                          RETURN
```

```
                                        TO:
                                        ;
                                        ;       CALCULATE THE TIMEOUT IN 9SEC INTERVALS (SDI RECEIVE XFC TAKES
                                        ;       9 SEC)
                                        ;
001221  104201  000001                  MOV     #1,R1           ; SET UP LOG2 SHIFTER
001223  110201                  1$:     ROL     R1              ; DOUBLE THE TIMEOUT VALUE
001224  103201  000001                  BIC     #1,R1           ; CLEAR THE LOW BIT
001226  117407                          DEC     R0              ; DECREMENT COUNT
001227  051223                          BNE     1$              ; IF COUNT INCOMPLETE, BRANCH
001230  114007                          CLR     R0              ; CLEAR 9SEC COUNT
001231  115407                  2$:     INC     R0              ; INCREMENT 9 SEC COUNT
001232  107201  000011                  SUB     #9.,R1          ; SUBTRACT 9 SEC FROM TIMEOUT
001234  031231                          BPL     2$              ; IF MORE TIME TO GO, BRANCH
001235  000000                          RETURN                  ; RETURN TO CALLING PROGRAM
```

HOSTRQ - HOST REQUEST - REPORT ERRORS, MEGABYTES TRANSFERRED,

```
                                ;RERROR
                                ;
                                ;REPORT ERROR TO HOST PROGRAM
                                ;THIS ROUTINE IS CALLED BY THE ERROR MACROS:
                                ;ERRSF, ERRDF, ERRHRD AND ERRSFT
                                ;
001236                          RERROR: PUSH R0                         ;SAVE ONE REGISTER
001236  100467                                                                  MOV R0,-.SP)
001237  104067                          MOV SP,R0                       ;GET STACK POINTER
001240                          PUSH <R1,R2,R3,R4>                      ;SAVE MORE REGISTERS
001240  100461                                                                  MOV R1,-(SP)
001241  100462                                                                  MOV R2,-(SP)
001242  100463                                                                  MOV R3,-(SP)
001243  100464                                                                  MOV R4,-(SP)
001244  115407                          INC R0                          ;CHANGE SAVED STACK POINTER TO POINT TO
                                                                        ; ADDRESS OF LOCATION AFTER CALL
001245  104271                          MOV (R0)+,R1                    ;GET RETURN PC
001246  104202  001007                  MOV #OUT.01,R2                  ;GET ADDRESS OF WHERE TO PUT DATA
001250  117401                          DEC R1                          ;REDUCE TO PC OF ERROR CALL
001251  100221                          MOV R1,(R2)+                    ;PUT PC IN OUT BUFFER
001252  115401                          INC R1                          ;GET BACK TO RETURN PC
001253  104113                          MOV (R1),R3                     ;GET ERROR NUMBER AND TYPE
001254  103203  176000                  BIC #^C001777,R3                ;CLEAR OTHER BITS
001256  100223                          MOV R3,(R2)+                    ;PUT IN BUFFER
001257  104303  001315                  MOV LUNIT,R3                    ;PUT UNIT NUMBER IN BUFFER
001261  100223                          MOV R3,(R2)+
001262  104214                          MOV (R1)+,R4                    ;GET COUNT OF PARAMETERS
001263  104213                          MOV (R1)+,R3                    ;GET MESSAGE POINTER
001264  100223                          MOV R3,(2)+                     ;PUT IN OUT BUFFER
                                                                        ;R1 IS NOW POINTING TO INSTRUCTION AFTER ERROR CALL
001265  110704                          SWAB R4
001266  110604                          ROR R4                          ;EXTRACT NUMBER OF PARAMETERS FROM ERROR MACRO
001267  110604                          ROR R4
001270  103204  177700                  BIC #177700,R4
001272  104040  001313                  MOV R4,SPADJU+1                 ;SAVE FOR LATER ADJUSTMENT OF STACK POINTER
001274  011301                          BEQ RERRCA                      ;BRANCH IF NO PARAMETERS
001275  104273                  RERRPA: MOV (R0)+,R3                    ;GET PARAMETER
001276  100223                          MOV R3,(R2)+                    ;STORE PARAMETER IN OUT BUFFER
001277  117404                          DEC R4                          ;COUNT THE PARAMETERS
001300  051275                          BNE RERRPA                      ;GET NEXT IF MORE
001301  100471                  RERRCA: MOV R1,-(R0)                    ;PUT RETURN ADDRESS ON STACK
001302  104207  000013                  MOV #ERRMES,R0                  ;SEND ERROR PACKET TO HOST PROGRAM
001304  020751                          CALL HOSTRQ
001305                          POP <R4,R3,R2,R1,R0>                    ;RESTORE REGISTERS
001305  104264                                                                  MOV (SP)+,R4
001306  104263                                                                  MOV (SP)+,R3
001307  104262                                                                  MOV (SP)+,R2
001310  104261                                                                  MOV (SP)+,R1
001311  104267                                                                  MOV (SP)+,R0
001312  105206  000000          SPADJU: ADD #0,SP                       ;ADJUST STACK OVER PARAMETERS
                                                                        ;VALUE CHANGED ABOVE
001314  000000                          RETURN

001315  177777                  LUNIT:  .WORD -1                        ;LOGICAL UNIT NUMBER (-1 FOR NOT AVAILABLE)

001316  000000                  SAVREG: .WORD 0                         ;STORAGE FOR REGISTER AT CALL TIME
                                        .IFT
```

```
48                                      .LIST
49                                      .ENDC
```

```
                                ;STACK AREA
                                
001317  123456                        .WORD 123456            ;END MARKER FOR STACK
001320                              .BLKW 31.               ;STACK
001357  123456                STACK:  .WORD 123456            ;MARKER FOR STACK UNDERFLOW
```

```
                        .SBTTL  DISK DRIVE SEQUENCER

000001                          ERRN=1.                         ;START ERROR NUMBERS AT 1.
                        ;GET WHICH UNITS TO TEST
001360  023361          START:  CALL    GETU                    ;GET UNIT

                        ;+
                        ;       FIND OUT IF HOST HAD TO INIT UDA TO GET A PROGRAM FROM AN ATTACHED DRIVE.
                        ;       THE PROGRAM WAS REQUESTED BY THE DRIVE TO BE DOWNLINE LOADED INTO ITS MEMORY.
                        ;       THE INDICATION COMES FROM THE UTOTST RESPONSE.
                        ;
                        ;       A WORD WITH MSB IS SET TO INDICATE THE LAST UNIT ENTRY.  AFTER THAT WORD
                        ;       ANOTHER WORD INDICATES WHEATHER OR NOT A FILE WAS SUPPOSE TO BE DOWN LINE LOADED.
                        ;       A '0' MEANS NORMAL PROCESSING, NO FILE WAS EXPECTED OR REQUESTED.
                        ;       A '1' MEANS A FILE WAS EXPECTED AND IS AVAILABLE.
                        ;       A '2' MEANS A FILE WAS EXPECTED AND IS NOT AVAILABLE.
                        ;
                        ;       R0 COMES BACK FROM GETU AS A POINTER TO THE WORD BEYOND THE LAST TABLE ENTRY
                        ;-
001361  104671  000001          MOV     1(R0),R1                ;WAS A FILE EXPECTED?
001363  011445                  BEQ     PORT0                   ;IF NOT, GO TO NORMAL PROCESSING
001364  115407                  INC     R0                      ;POINT TO INDICATOR IN IN.RQ BUFFER
001365  104075                  MOV     R0,R5                   ;R5 -> INDICATOR
001366  104151                  MOV     (R5),R1                 ;1ST WORD CONTAINS INDICATOR TO SEE IF A FILE WAS EX
001367  106201  000001          CMP     #1,R1                   ;WAS THE EXPECTED FILE AVAILABLE?
001371  052553                  BNE     DO.DI7                  ;IF NOT AVAILABLE, GO TO REPORT ERROR
                        ; *** FILE WAS THERE, NOW SEE WHERE IN THE TABLE IT WAS SO PORT INDICATOR IS THE SAME?
001372  104207  003676          MOV     #UNITS,R0               ;R0 -> UNIT TABLE
001374  104651  000001          MOV     1(R5),R1                ;R1 = UNIT NUMBER
001376  104202  000001          MOV     #UNIT0,R2               ;R2 = 1ST PORT
001400  114003                  CLR     R3                      ;R3 = UNIT TABLE INDICATOR
001401  114004          1$:     CLR     R4                      ;R4 KEEPS TRACK OF WHICH SUBUNITS
001402  104270  001316  2$:     MOV     (R0)+,SAVREG            ;STORE UNIT IN SAVE REG
001404  071431                  BMI     4$                      ;IF NEGATIVE VALUE, DON'T COMPARE
001405  103300  177400  001316  BIC     HIBYTE,SAVREG           ;CLEAR EXTANEOUS BITS
001410  106301  001316          CMP     SAVREG,R1               ;DID WE FIND THE DRIVE?
001412  011435                  BEQ     5$                      ;IF IT IS, WE HAVE FOUND IT
001413  115404                  INC     R4                      ;ELSE, INCREMENT SUBUNIT POINTER
001414  106204  000004          CMP     #4,R4                   ;ENDED WITH THIS UNIT ENTRY?
001416  051402                  BNE     2$                      ;IF NOT, CONTINUE
001417  105022          3$:     ADD     R2,R2                   ;ELSE, NEXT PORT SET IN R2
001420  115403                  INC     R3                      ;INCREMENT UNIT TABLE INDICATOR
001421  106203  000004          CMP     #4,R3                   ;DONE?
001423  051401                  BNE     1$                      ;IF THIS GETS TO 4, THEN ERROR(UNIT NOT FOUND)
001424                          ERRHRD  MS57,R1
                                                                        MOV R1,-(SP)
001424  100461                                                          CALL RERROR     ;ERROR # 1.
001425  021236                                                          .WORD <PRMS*2000>+<2*400>+ERRN
001426  003001                                                          .WORD MS57
001427  002547
001430  001445                  BR      PORT0                   ;START REGULAR TESTING
                        ; *** IF HERE, NEGATIVE ENTRY, GO TO NEXT UNIT ENTRY.  ADJUST R0 TO PROPER POINTER
001431  105207  000003  4$:     ADD     #3,R0                   ;R0 -> INTO NEXT UNIT\INCREMENTED BY (R0)+
001433  107047                  SUB     R4,R0                   ;REALIGN TO POINT TO BEGINNING OF POINTER
001434  001417                  BR      3$                      ;GO BACK INTO THE LOOP
                        ; *** IF HERE, FOUND THE ENTRY, SAVE PERTINENT VALUES, GO DOWN LINE LOAD PROGRAM
001435  104650  000001  001315  5$:     MOV     1(R5),LUNIT             ;SAVE LOGICAL UNIT NUMBER
```

```
     54 001440  104020  003717                  MOV     R2,SDI                  ;SAVE PORT VALUE
     55 001442  104030  003716                  MOV     R3,UNITNB               ;SAVE UNIT NUMBER OFFSET IN TABLE
     56 001444  002424                          BR      DO.DI2                  ;GO DOWN LINE LOAD PROGRAM
     57
     58                                 ; *** NORMAL PROCESSING
     59 001445                          PORT0:
     60                                 ;       MOV     #UNITS,R5               ;GET POINTER TO UNITS STORAGE BLOCK
     61                                 ;       MOV     #IN.01,R4               ;GET POINTER TO DATA FROM HOST
     62                                 ;       MOV     #4,R3                   ;GET COUNT OF PORTS
     63                                 ;
     64                                 ;PORT1: MOV     (R4)+,R0                ;GET FIRST NUMBER FROM HOST
     65                                 ;       BPL     PORT2                   ;IF NOT NEGATIVE, USE IT
     66                                 ;       MOV     (R4),R0                 ;OTHERWISE GET SECOND NUMBER FROM HOST
     67                                 ;PORT2: MOV     R0,(R5)+                ;STORE NUMBER IN TABLE
     68                                 ;       INC     R4                      ;BUMP POINTER
     69                                 ;       DEC     R3                      ;COUNT THE UNITS
     70                                 ;       BNE     PORT1                   ;REPEAT FOR ALL UNITS
     71
     72                                 ;SEQUENCE THE DIAGNOSTICS TO ALL UNITS SELECTED.
     73                                 ;TEST CODE WILL BE CALLED FOR EACH DISK DRIVE TO BE TESTED.
     74                                 ;  LUNIT WILL CONTAIN LOGICAL UNIT NUMBER OF DRIVE FOR ERROR REPORTS
     75                                 ;  SDI WILL CONTAIN SDI INTERCONNECT CODE FOR SELECTED DRIVE
     76
     77 001445  114001                          CLR     R1                      ;START WITH UNIT 0 INDEX
     78 001446  104202  000001                  MOV     #UNIT0,R2               ;SDI INTERCONNECT CODE
     79
     80 001450  104207  003676          PORT3:  MOV     #UNITS,R0               ;GET POINTER TO UNITS TABLE
     81 001452  105017                          ADD     R1,R0                   ;ADD INDEX
     82 001453  104173                          MOV     (R0),R3                 ;GET CONTENTS OF TABLE
     83 001454  071505                          BMI     PORT5                   ;BYPASS TESTING UNIT IF MINUS
     84 001455  114004                          CLR     R4                      ;R4 = INDEX COUNTER
     85 001456  104173                  1$:     MOV     (R0),R3                 ;CHECK UNIT
     86 001457  102203  040000                  BIT     #40000,R3               ;IS THE DRIVE TESTABLE?
     87 001461  011470                          BEQ     2$                      ;IF SO, CONTINE
     88 001462  115407                          INC     R0                      ;INCREMENT UNIT TABLE POINTER
     89 001463  115404                          INC     R4                      ;INCREMENT INDEX POINTER
     90 001464  106204  000004                  CMP     #4,R4                   ;IF = 4, POINTS TO NEXT ENTRY
     91 001466  011505                          BEQ     PORT5                   ;IF IT DOES, TRY NEXT UNIT
     92 001467  001456                          BR      1$                      ;ELSE TRY AGAIN
     93 001470  107047                  2$:     SUB     R4,R0                   ;R0-> HEAD OF ENTRY
     94 001471  100173                          MOV     R3,(R0)                 ;SAVE AT HEAD OF THE TABLE
     95 001472  104010  003716                  MOV     R1,UNITNB               ;STORE UNIT INDEX
     96 001474  104030  001315                  MOV     R3,LUNIT                ;STORE LOGICAL UNIT NUMBER FOR DRIVE
     97 001476  104020  003717                  MOV     R2,SDI                  ;STORE SDI INTERCONNECT CODE
     98 001500  001517                          BR      TEST                    ;PERFORM TEST ON THIS DRIVE
     99 001501  104301  003716          TESTX:  MOV     UNITNB,R1               ;GET UNIT INDEX
    100 001503  104302  003717                  MOV     SDI,R2                  ;GET SDI INTERCONNECT CODE
    101 001505  105201  000004          PORT5:  ADD     #4,R1                   ;INCREMENT INDEX
    102 001507  105022                          ADD     R2,R2                   ;COMPUTE NEW INTERCONNECT CODE
    103 001510  102202  000017                  BIT     #UNIT0+UNIT1+UNIT2+UNIT3,R2     ;CHECK IF LEGAL DRIVE SELECTED
    104 001512  051450                          BNE     PORT3                   ;REPEAT FOR ALL DRIVES
    105 001513  104207  000016          DONECD: MOV     #DONE,R0                ;END OF PROGRAM
    106 001515  020751                          CALL    HOSTRQ
    107 001516  001513                          BR      DONECD                  ;REPEAT IF RETURNED
```

```
                                .SBTTL  INITIALIZE DRIVE AND LOOK AT DRIVE SIGNALS

                                ;START OF TEST CODE
                                ;
                                ;INPUTS:
                                ;       LUNIT - LOGICAL UNIT NUMBER OF DRIVE UNDER TEST
                                ;       SDI - SDI INTERCONNECT CODE FOR DRIVE

                                ;INITIALIZE THE DRIVE

001517  104302  003717       TEST:   MOV     SDI,R2                  ;GET SDI SELECT CODE
001521  060011                         XFC     DINIT                   ;INITIALIZE THE DRIVE

                                ;WAIT FOR DRIVE TO ASSERT RECEIVER READY
                                ;TIME OUT AFTER ...?

001522  114005                         CLR     R5                      ;GET TIMEOUT COUNTER
001523                         ILOOP:   DSTAT   DONECD,MS1,MS2          ;LOOK AT DRIVE STATUS LINES
001523  020720                         CALL    RDSTAT                  ; GET DRIVE STATUS
001524  102201  010000                 BIT     #10000,R1               ; SEE IF ANY ERRORS
001526  011542                         BEQ     2$                      ; IF NO ERROR, BRANCH
001527  102201  004000                 BIT     #4000,R1                ; SEE IF XMIT ERROR
001531  011536                         BEQ     1$                      ; IF SO, BRANCH
001532                                 ERRHRD  MS1                     ; REPORT INVALID STATUS ERROR
001532  021236                                                                 CALL RERROR     ;ERROR # 2.
001533  001002                                                                 .WORD <PRMS*2000>+<2*400>+ERRN
001534  000000                                                                 .WORD MS1
001535  001513                         BR      DONECD                  ; BRANCH TO DONE
001536                         1$:     ERRHRD  MS2                     ; REPORT XMIT ERROR
001536  021236                                                                 CALL RERROR     ;ERROR # 3.
001537  001003                                                                 .WORD <PRMS*2000>+<2*400>+ERRN
001540  000050                                                                 .WORD MS2
001541  001513                         BR      DONECD                  ; BRANCH TO DONE
001542                         2$:
001542  102201  000001                 BIT     #RCVRDY,R1              ;CHECK RECEIVER READY LINE
001544  051553                         BNE     ECHO1                   ;ADVANCE IF ASSERTED
001545  117405                         DEC     R5                      ;DECREMENT TIME OUT COUNTER
001546  051523                         BNE     ILOOP                   ;STAY IN LOOP UNTIL SIGNAL SETS OR TIMEOUT
001547                                 ERRHRD  MS3                     ;REPORT ERROR
001547  021236                                                                 CALL RERROR     ;ERROR # 4.
001550  001004                                                                 .WORD <PRMS*2000>+<2*400>+ERRN
001551  000074                                                                 .WORD MS3
001552  002736                         BR      TESTEX                  ;EXIT TESTING THIS DRIVE
```

```
                        .SBTTL  ECHO DATA TO DRIVE

                        ;ECHO THE FOLLOWING DATA PATTERNS TO THE DRIVE THEN CHECK THE
                        ;RESPONSE FOR THE PROPER DATA RECEIVED:
                        ;       377
                        ;       000
                        ;       252
                        ;       360
                        ;       017

001553  104205  003722  ECHO1:  MOV     #ECHOD,R5               ;GET POINTER TO ECHO DATA
001555  104157          ECHO2:  MOV     (R5),R0                 ;GET ECHO DATA
001556  103207  177400          BIC     #HIBYTE,R0              ;CLEAR COMMAND FROM WORD
001560  060010                  XFC     ECHO                    ;PERFORM THE ECHO COMMAND

                        ;CHECK FOR TIMEOUT ERROR

001561  115001                  TST     R1                      ;CHECK FOR ERROR
001562  011600                  BEQ     ECHO4                   ;BRANCH IF NONE
001563  102201  000001          BIT     #1,R1                   ;CHECK IF SEND ERROR
001565  011573                  BEQ     ECHO3                   ;BRANCH IF RECEIVE ERROR
001566                          ERRHRD  MS4,R0                  ;REPORT SEND ERROR
001566  100467                                                          MOV R0,-(SP)
001567  021236                                                          CALL RERROR     ;ERROR # 5.
001570  003005                                                          .WORD <PRMS*2000>+<2*400>+ERRN
001571  000130                                                          .WORD MS4
001572  001614                  BR      ECHO5
001573                  ECHO3:  ERRHRD  MS5,R0                  ;REPORT RECEIVE ERROR
001573  100467                                                          MOV R0,-(SP)
001574  021236                                                          CALL RERROR     ;ERROR # 6.
001575  003006                                                          .WORD <PRMS*2000>+<2*400>+ERRN
001576  000170                                                          .WORD MS5
001577  001614                  BR      ECHO5

                        ;CHECK DATA RECEIVED

001600  106157          ECHO4:  CMP     (R5),R0                 ;COMPARE DATA RECEIVED WITH
001601  011614                  BEQ     ECHO5                   ; DATA SENT
001602                          ERRHRD  MS6,(R5),R0             ;REPORT DATA COMPARE ERROR
001602  100467                                                          MOV R0,-(SP)
001603  104010  001316                                                  MOV R1,SAVREG
001605  104151                                                          MOV (R5),R1
001606  100461                                                          MOV R1,-(SP)
001607  104301  001316                                                  MOV SAVREG,R1
001611  021236                                                          CALL RERROR     ;ERROR # 7.
001612  005007                                                          .WORD <PRMS*2000>+<2*400>+ERRN
001613  000233                                                          .WORD MS6

                        ;MOVE TO NEXT DATA PATTERN

001614  115405  ECHO5:  INC     R5                      ;BUMP TO NEXT DATA TO SEND
001615  104157          MOV     (R5),R0                 ;CHECK IF AT END OF TABLE
001616  071556          BMI     ECHO2                   ;SEND THIS DATA IF NOT
```

GET STATUS COMMAND

```
                                .SBTTL  GET STATUS COMMAND
                                ;ISSUE GET STATUS COMMAND AND CHECK THAT IT PERFORMS PROPERLY
001617  104204  000002                MOV     #2,R4                   ;COUNTER
001621  023522                        CALL    GTSTAT                  ;GET STATUS
001622  115003                        TST     R3                      ;WAS THERE AN ERROR?
001623  011625                        BEQ     DRCLR1                  ;IF NOT, CONTINUE
001624  002736                        BR      TESTEX                  ;ELSE, EXIT FROM TESTING ON THIS DRIVE
                                ;CLEAR DRIVE ERRORS
001625  023564                DRCLR1: CALL    CLRDRV                  ; CLEAR DRIVE
001626  115003                        TST     R3                      ; WAS THERE AN ERROR?
001627  011631                        BEQ     GSTS3                   ; IF NOT, CONTINUE
001630  002736                        BR      TESTEX
                                ;ISSUE GET STATUS COMMAND AND CHECK THAT IT PERFORMS PROPERLY
001631                          GSTS3:
001631  023522                        CALL    GTSTAT                  ;GET STATUS
001632  115003                        TST     R3                      ;WAS THERE AN ERROR?
001633  011635                        BEQ     GSTS6                   ;IF NOT, CONTINUE
001634  002736                        BR      TESTEX                  ;ELSE, STOP TESTING THIS DRIVE
                                ;LOOK AT DATA IN RESPONSE PACKET
001635  104307  004216          GSTS6:  MOV     ST+1,R0                 ;GET TWO WORDS FROM PACKET
001637  104301  004217                  MOV     ST+2,R1
001641  102201  000250                  BIT     #ST.WE+ST.PE+ST.FE,R1   ;CHECK ERROR BITS
001643  001662                        BR      GSTS7                   ;BRANCH IF ALL CLEAR
001644  117404                        DEC     R4                      ;TRY ONCE MORE?
001645  051625                        BNE     DRCLR1                  ;IF OK, TRY AGAIN
001646                                ERRHRD  MS16,R0,R1,ST+2
001646  104010  001316                                                          MOV R1,SAVREG
001650  104301  004217                                                          MOV ST+2,R1
001652  100461                                                                  MOV R1,-(SP)
001653  104301  001316                                                          MOV SAVREG,R1
001655  100461                                                                  MOV R1,-(SP)
001656  100467                                                                  MOV R0,-(SP)
001657  021236                                                                  CALL RERROH     ;ERROR # 8.
001660  007010                                                                  .WORD <PRMS*2000>+<2*400>+ERRN
001661  000655                                                                  .WORD MS16
                                ;CHECK IF DIAGNOSTIC REQUEST BIT IS SET
001662                          GSTS7:
001662  104070  003720                  MOV     R0,SAVSTA
```

```
                                        .SBTTL  GET DRIVE CHARACTERISTICS
                                ;GET DRIVE CHARACTERISTICS
                                ;
001664  104200  000012  001150  T:      MOV     #10.,SDISTO             ; SET UP TEMPORARY SHORT TIMEOUT VALUE
001667  104203  004177                  MOV     #CR.GCR,R3              ;POINT TO GET CHARS COMMAND
001671                                  TALKX   TESTEX,MS50,MS51        ; SDI INTERCHANGE
001671  021102                          CALL    TALKER                  ; INITIATE SDI INTERCHANGE
001672  115003                          TST     R3                      ; SEE IF ERROR OCCURRED
001673  011705                          BEQ     12$                     ; IF NOT, BRANCH
001674  031701                          BPL     11$                     ; IF SO, BRANCH
001675                                  ERRHRD  MS50;SEND COMMAND ERROR
001675  021236                                                                  CALL RERROR     ;ERROR # 9.
001676  001011                                                                  .WORD <PRMS*2000>+<2*400>+ERRN
001677  002165                                                                  .WORD MS50
001700  002736                          BR      TESTEX
001701                          11$:    ERRHRD  MS51;RECEIVE COMMAND ERROR
001701  021236                                                                  CALL RERROR     ;ERROR # 10.
001702  001012                                                                  .WORD <PRMS*2000>+<2*400>+ERRN
001703  002226                                                                  .WORD MS51
001704  002736                          BR      TESTEX
001705                          12$:
001705  106207  000170                  CMP     #CHRRES,R0              ;CHECK FOR SUCCESSFUL RESPONSE
001707  011724                          BEQ     T00
001710                                  ERRHRD  MS52,#CHRRES,R0         ;GET CHARACTERISTICS COMMAND FAILED
001710  100467                                                                  MOV R0,-(SP)
001711  104010  001316                                                          MOV R1,SAVREG
001713  104201  000170                                                          MOV #CHRRES,R1
001715  100461                                                                  MOV R1,-(SP)
001716  104301  001316                                                          MOV SAVREG,R1
001720  021236                                                                  CALL RERROR     ;ERROR # 11.
001721  005013                                                                  .WORD <PRMS*2000>+<2*400>+ERRN
001722  002271                                                                  .WORD MS52
001723  002736                          BR      TESTEX                  ; BRANCH TO END OF TEST

001724  104307  004215          T00:    MOV     ST+SHRTTO,R0            ; GET SHORT TIMEOUT
001726  103207  177760                  BIC     #LBLONB,R0              ; CLEAR UNUSED BITS
001730  021221                          CALL    TO                      ; SET UP TIMEOUT
001731  104070  001150                  MOV     R0,SDISTO               ; SAVE IN SHORT TIMEOUT
001733  104307  004216                  MOV     ST+LONGTO,R0            ; GET LONG TIMEOUT
001735  103207  177760                  BIC     #LBLONB,R0              ; CLEAR UNUSED BITS
001737  021221                          CALL    TO                      ; SET UP TIMEOUT
001740  104070  001151                  MOV     R0,SDILTO               ; SAVE IN LONG TIMEOUT

```

CHECK WHICH COMMAND HAS BEEN GIVEN

```
                                        .SBTTL  CHECK WHICH COMMAND HAS BEEN GIVEN
001742  104307  003720         ST.DIA: MOV     SAVSTA,R0               ;GET STORED STATUS WORD
001744  102207  000040                 BIT     #ST.DR,R0               ;WAS THE DR BIT SET?
001746  052340                         BNE     DO.DIX                  ;IF SO, GO HANDLE IT WITHIN DIAGNOSE CODE
                                ; *** NOW GO DIAGNOSE REGION ZERO
001747  114000  001007                 CLR     OUT.01                  ;CLEAR OUT BUFFER TO INITIAL DIAGNOSE COMMANDS
001751  114000  001010                 CLR     OUT.02                  ;
001753  114000  004416                 CLR     DIAG.1                  ;MAKE SURE DIAGNOSE COMMAND RETURNS OP-CODE = 0
                                                                        ; (FOR 1ST TIME THROUGH ONLY)
001755  104200  000017  001046         MOV     #17,IN.02               ;0FH IS DIAGNOSTIC ;****TEMP
                                ;       CLR     IN.02                   ;CLEAR REGION ID FOR 1ST DIAGNOSE COMMAND
001760  002110                         BR      DIAGNS                  ;DO DIAGNOSE COMMAND
001761                          GSTST8:                                 ;
001761  104207  000002                 MOV     #T2CMD,R0               ;SET REQUEST NUMBER
001763  020751                         CALL    HOSTRQ                  ;ASK HOST FOR PORTS
001764  104307  001045                 MOV     IN.01,R0                ;R0 = RESPONSE CODE;IN.02
                                                                        ; 0 = EXIT ; 1 = WRITE
                                                                        ; 2 = READ ; 3 = DIAGNOSE
001766  012736                         BEQ     TESTEX                  ; OP = 0 /EXIT
001767  117407                         DEC     R0                      ; R0 = 1?
001770  012010                         BEQ     WRITE                   ; IF SO, WRITE
001771  117407                         DEC     R0                      ; R0 = 2?
001772  012054                         BEQ     READ                    ; IF SO, READ
001773  117407                         DEC     R0                      ; R0 = 3?
001774  012110                         BEQ     DIAGNS                  ; IF SO, DIAGNOSE
001775                                 ERRHRD  MS20,IN.02              ; ELSE, ERROR=INVALID INPUT
001775  104070  001316                                                                MOV R1,SAVREG
001777  104301  001046                                                                MOV IN.02,R1
002001  100461                                                                        MOV R1,-(SP)
002002  104301  001316                                                                MOV SAVREG,R1
002004  021236                                                                        CALL RERROR     ;ERROR # 12.
002005  003014                                                                        .WORD <PRMS*2000>+<2*400>+ERRN
002006  001125                                                                        .WORD MS20
002007  001761                         BR      GSTST8
```

```
                                                .SBTTL  MEMORY WRITE
                                        ;:+
                                        ;
                                        ;       INPUT   IN.02 = REGION ID
                                        ;               IN.03 = OFFSET
                                        ;               IN.04 = DATA BYTE
                                        ;
                                        ;       OUTPUT  OUT.RQ HAS DAT SET FOR T2CMD
                                        ;:-
002010                                  WRITE:
002010  104300  001046  003753                MOV     IN.02,WRM+1             ;REGION ID
002013  104300  001047  003754                MOV     IN.03,WRM+2             ;OFFSET
002016  104307  001050                        MOV     IN.04,R0                ;R0 = BYTE OF DATA IN LO BYTE
002020  110707                                SWAB    R0                      ;R0 IS IN HI BYTE
002021  103207  000377                        BIC     #LOBYTE,R0              ;CLEAR LO BYTE
002023  101207  000001                        BIS     #1,R0                   ;SET BYTE COUNT
002025  104070  003755                        MOV     R0,WRM+5                ;SET WORD IN PACKET
002027  114000  003756                        CLR     WRM+4
002031  114000  003757                        CLR     WRM+5
002033  104200  000007  003746              MOV     #7,CR.WRM+1             ;SET COMMAND BYTE COUNT
                                        ;
002036  023013                          CALL    WRTMEM
002037  104200  000001  001010              MOV     #1,OUT.02               ;R0 = OP CODE
002042  104300  001315  001007              MOV     LUNIT,OUT.01            ;DRIVE NUMBER
002045  114000  001011                        CLR     OUT.03                  ;
002047  114000  001012                        CLR     OUT.04                  ;
002051  114000  001013                        CLR     OUT.05                  ;
002053  001761                          BR      GSTST8                  ;GO SEND REQUEST
```

MEMORY READ

```
                                        .SBTTL  MEMORY READ
                                ;:+
                                ;
                                ;       INPUT   IN.02 HAS REGION ID
                                ;               IN.03 HAS OFFSET
                                ;
                                ;       OFFSET  SETS UP OUTBUFFER FOR T2CMD
                                ;:-
002054                          READ:
002054  104300  001046  004167        MOV     IN.02,RDM+1             ;REGION ID
002057  104300  001047  004170        MOV     IN.03,RDM+2             ;OFFSET
002062  104200  000001  004171        MOV     #1,RDM+3                ;BYTE COUNT
                                ;
002065  022737                        CALL    RDMEM
                                ; *** SET UP RESPONSE PACKET
002066  104300  001315  001007        MOV     LUNIT,OUT.01            ;SET UP REQUEST
002071  104200  000002  001010        MOV     #2,OUT.02               ;RETURN OPCODE SET IN BUFFER
002074  104307  004215                MOV     ST,R0                   ;R0 = BYTE COUNT + DATA
002076  110707                        SWAB    R0                      ;DATA IN LO BYTE
002077  103207  177400                BIC     #HIBYTE,R0              ;CLEAR BYTE COUNT (1 BYTE ONLY SENT)
002101  104070  001011                MOV     R0,OUT.03               ;STORE IN BUFFER FOR HOST
002103  114000  001012                CLR     OUT.04                  ;
002105  114000  001013                CLR     OUT.05                  ;
002107  001761                        BR      GSTST8                  ;GO SEND REQUEST
```

```
                                        .SBTTL  DIAGNOSE
                                ;++
                                ;
                                ;       INPUT   IN.02 HAS REGION ID
                                ;
                                ;       OUTPUT  OUT.RQ IS SET UP FOR T2CMD
                                ;--
002110                          DIAGNS:
002110  104300  001046  004213          MOV     IN.02,DIA+1             ;SET UP MEMORY REGION ID
                                ;
                                ;       MSSGE   MS59,DIA+1              ;SEND MESSAGE
002113  104203  004205                  MOV     #CR.DIA,R3              ;R3->DIAGNOSE PACKET
002115  104302  003717                  MOV     SDI,R2                  ;R2 = PORT
002117  021102                          CALL    TALKER                  ;SEND COMMAND AND RECEIVE RESPONSE FROM DRIVE
002120  115003                          TST     R3                      ;SEE IF ERROR OCCURRED
002121  012135                          BEQ     3$                      ;IF NO ERRORS, BRANCH
002122  110203                          ROL     R3                      ;SHIFT HIGH BIT INTO CARRY BIT
002123  042130                          BCC     2$                      ;IF CARRY CLEAR (RECEIVE ERROR) BRANCH
                                ; *** REPORT TRANSMISSION ERROR
                                        ERRHRD  MS37
002124                                                                  CALL RERROR     ;ERROR # 13.
002124  021236                                                         .WORD <PRMS*2000>+<2*400>+ERRN
002125  001015                                                         .WORD MS37
002126  001526
002127  002313                          BR      11$
                                ; *** REPORT RECEPTION ERROR
002130                          2$:     ERRHRD  MS38
002130  021236                                                         CALL RERROR     ;ERROR # 14.
002131  001016                                                         .WORD <PRMS*2000>+<2*400>+ERRN
002132  001557                                                         .WORD MS38
002133  023237                          CALL    TYPERR
002134  002313                          BR      11$
                                ; *** CHECK RESPONSE OP-CODE FROM DRIVE
002135                          3$:
002135  106207  000374                  CMP     #DIA.EN,R0              ;CHECK RESPONSE CODE
002137  012154                          BEQ     4$                      ;IF EQUAL, BRANCH
002140                                  ERRHRD  MS36,#DIA.EN,R0
002140  100467                                                         MOV R0,-(SP)
002141  104010  001316                                                 MOV R1,SAVREG
002143  104201  000374                                                 MOV #DIA.EN,R1
002145  100461                                                         MOV R1,-(SP)
002146  104301  001316                                                 MOV SAVREG,R1
002150  021236                                                         CALL RERROR     ;ERROR # 15.
002151  005017                                                         .WORD <PRMS*2000>+<2*400>+ERRN
002152  001440                                                         .WORD MS36
002153  002313                          BR      11$
```

```
                                        .SBTTL  DIAGNOSE/READ MEMORY TO SEE IF ERROR OCCURRED
                                ; *** DO A READ MEMORY SDI COMMAND
002154  104300  004215  004167  4$:     MOV     ST,RDM+1                ;REGION ID SET
002157  114000  004170                  CLR     RDM+2                   ;CLEAR OFFSET
002161  104200  000010  004171          MOV     #8.,RDM+3               ;BYTE COUNT - 8
002164  022737                          CALL    RDMEM                   ;READ THE MEMORY
002165  115002                          TST     R2                      ;ALL OK?
002166  052313                          BNE     11$                     ;IF NOT, EXIT/TRY NEXT DRIVE
002167  104302  003717                  MOV     SDI,R2                  ;RESTORE PORT INDICATOR
002171  104204  001013                  MOV     #OUT.05,R4              ;R4 -> OUT BUFFER
002173  023054                          CALL    CONMEM                  ;CONVERT MEMORY
                                ; *** DATA NOW IN OUT.RQ
002174  104303  001015                  MOV     OUT.07,R3               ;R3 = ERROR NUMBER
002176  052242                          BNE     10$                     ;IF NOT 0, REPORT ERROR
002177  104303  001014                  MOV     OUT.06,R3               ;R3 HAD ET & DA FLAGS
002201  102203  100000                  BIT     #ERRTYP,R3              ;WAS ET SET? WITH NO ERROR NUMBER???
002203  012210                          BEQ     5$                      ;IF NOT, CONTINUE
002204                                  ERRHRD  MS40
002204  021236                                                          CALL RERROR     ;ERROR # 16.
002205  001020                                                          .WORD <PRMS*2000>+<2*400>+ERRN
002206  001613                                                          .WORD MS40
002207  002631                          BR      DO.DI9                  ;GET EXTENDED STATUS
                                ; *** CHECK IF DATA IS AVAILABLE FOR INFORMATION
002210  102203  040000          5$:     BIT     #DATAVL,R3              ;IS DATA AVAILABLE?
002212  012313                          BEQ     11$                     ;IF NOT, CONTINUE WITH THIS DRIVE
002213  104200  000010  004170          MOV     #8.,RDM+2               ;PREVIOUS BYTE COUNT
002216  023114                          CALL    GETBCN                  ;GET NEW BYTE COUNT
002217  022737                          CALL    RDMEM                   ;READ MEMORY XFC
002220  115002                          TST     R2                      ;ALL OK?
002221  052313                          BNE     11$                     ;IF NOT, EXIT/DROP DRIVE
002222  104302  003717                  MOV     SDI,R2                  ;RESTORE PORT INDICATOR
002224  104204  001017                  MOV     #OUT.09,R4              ;R4 -> OUT BUFFER
002226  023054                          CALL    CONMEM                  ;CONVERT MEMORY
002227                                  MSSGE   MS41
002227  104200  001715  001010                          MOV     #MS41,OUT.02
002232  100467                                                  MOV R0,-(SP)
002233  100461                                                  MOV R1,-(SP)
002234  104207  000015                          MOV     #MESSAG,R0
002236  020751                                  CALL    HOSTRQ
002237  104261                                                  MOV (SP)+,R1
002240  104267                                                  MOV (SP)+,R0
002241  002631                          BR      DO.DI9                  ;GET EXTENDED STATUS
                                ; *** REPORT AN ERROR
002242  104303  001014          10$:    MOV     OUT.06,R3               ;R3 HAS ET & DA FLAGS
002244  114000  004415                  CLR     S.ERR                   ;HARD ERROR ASSUMED
002246  102203  100000                  BIT     #ERRTYP,R3              ;WAS IT A HARD ERROR?
002250  052254                          BNE     6$                      ;NO, SKIP NEXT INSTRUCTION
002251  104200  177777  004415          MOV     #177777,S.ERR           ;SOFT ERROR SET
002254  102203  040000          6$:     BIT     #DATAVL,R3              ;DATA AVAILABLE?
002256  012276                          BEQ     7$                      ;IF NOT, CONTINE
002257  104200  000010  004170          MOV     #8.,RDM+2               ;PREVIOUS BYTE COUNT
002262  023114                          CALL    GETBCN                  ;GET NEW BYTE COUNT
002263  022737                          CALL    RDMEM                   ;READ MEMORY XFC
002264  115002                          TST     R2                      ;ALL OK?
002265  052313                          BNE     11$                     ;IF NOT, EXIT/DROP DRIVE
002266  104302  003717                  MOV     SDI,R2                  ;RESTORE PORT INDICATOR
002270  104204  001017                  MOV     #OUT.09,R4              ;R4 -> OUT BUFFER
```

```
48 002272  023054                        CALL    CONMEM          ;CONVERT MEMORY
49 002273  102203  040000                BIT     #DATAVL,R3      ;DATA AVAILABLE?
50 002275  052300                        BNE     8$              ;IF SO, SKIP NEXT INSTRUCTION
51 002276  114000  001016          7$:   CLR     OUT.08          ;CLEAR BINARY COUNT AND ASCII COUNT
52 002300  115000  004415          8$:   TST     S.ERR           ;SOFT ERROR?
53 002302  052307                        BNE     9$              ;IF NOT 0, SOFT ERROR
54                                 ; *** PRINT HARD ERROR MESSAGE
55 002303                                ERRHRD  MS42
   002303  021236                                                       CALL RERROR     ;ERROR # 17.
   002304  001021                                                       .WORD <PRMS*2000>+<2*400>+ERRN
   002305  002013                                                       .WORD MS42
56 002306  002631                        BR      DO.DI9          ;GET EXTENDED STATUS
57
58 002307                          9$:   ERRSFT  MS43
   002307  021236                                                       CALL RERROR     ;ERROR # 18.
   002310  001422                                                       .WORD <PRMS*2000>+<3*400>+ERRN
   002311  002100                                                       .WORD MS43
59 002312  002631                        BR      DO.DI9          ;GET EXTENDED STATUS
```

```
                                        .SBTTL  DIAGNOSE/DO A DRIVE CLEAR
                                ; *** DO DRIVE CLEAR, GET STATUS AND CHECK IF DR BIT IS SET
002313                          11$:
002313  104204  000004          DO.DC4: MOV     #4,R4                   ;R4 = # MAXIMUM DRIVE CLEAR TRIES
002315  104302  003717          DO.DCL: MOV     SDI,R2                  ;RESTORE PORT INDICATOR
002317  023564                          CALL    CLRDRV                  ;CALL DRIVE CLEAR
002320  115003                          TST     R3                      ;ERROR?
002321  012325                          BEQ     1$                      ;IF NOT, CONTINUE
002322  117404                          DEC     R4                      ;REACHED MAXIMUM # OF DRV CLR TRIES?
002323  052315                          BNE     DO.DCL                  ;IF NOT, TRY AGAIN
002324  002736                          BR      TESTEX                  ;ELSE DROP DRIVE
002325  023522                  1$:     CALL    GTSTAT                  ;CALL GET STATUS
002326  115003                          TST     R3                      ;ERROR?
002327  012333                          BEQ     2$                      ;IF NOT, CONTINUE
002330  117404                          DEC     R4                      ;REACHED MAXIMUM # OF DRV CLR TRIES?
002331  052315                          BNE     DO.DCL                  ;IF NOT, TRY AGAIN
002332  002736                          BR      TESTEX                  ;ELSE DROP DRIVE
002333  104307  004216          2$:     MOV     ST+1,R0                 ;GET STATUS WORD
002335  102207  000040                  BIT     #ST.DR,R0               ;DR SET?
002337  012706                          BEQ     DO.DIO                  ;IF NOT, EXIT
                                ; *** IF DR BIT IS SET, READ 6 BYTES FROM REGION ID FFFD
002340  104200  177775  004167  DO.DIX: MOV     #FFFD,RDM+1             ;FFFD (REGION ID) STORED IN READ MEM PACKET
002343  114000  004170                  CLR     RDM+2                   ;OFFSET = 0
002345  104200  000006  004171          MOV     #6,RDM+3                ;BYTE COUNT = 6
002350  022737                          CALL    RDMEM                   ;READ MEMORY
002351  115002                          TST     R2                      ;WAS THERE AN ERROR?
002352  012354                          BEQ     1$                      ;IF THERE WAS NOT, CONTINUE
                                ;       MSSGE   MS53
002353  002706                          BR      DO.DIO                  ;DO A DRIVE CLEAR
002354  104302  003717          1$:     MOV     SDI,R2                  ;RESTORE PORT INDICATOR
002356  104204  004215                  MOV     #ST,R4                  ;R4 -> ST
002360  023054                          CALL    CONMEM                  ;CONVERT MEMORY
                                ; *** GET REGION ID AND PROGRAM NAME IF ANY
002361  104300  004215  003721          MOV     ST,SAVRID               ;SAVE REGION ID
002364  115000  004216                  TST     ST+1                    ;IS CHARACTER ENCODED?
002366  052372                          BNE     DO.DI1                  ;IF SO, GO GET FILE FROM HOST
002367  115000  004217                  TST     ST+2                    ;IF 1ST WORD 0, IS THE SECOND?
002371  012547                          BEQ     DO.DI3                  ;IF SO, GO DIAGNOSE REGION
```

```
                                        .SBTTL  DIAGNOSE/GET PROGRAM NAME SPECIFIED BY DRIVE AND DOWNLINE LOAD
                                ; *** GET THE PROGRAM WHICH NAME WAS SPECIFIED BY THE DRIVE
002372                          DO.DI1:
002372  104300  003721  003753          MOV     SAVRID,WRM+1            ;STORE REGION ID IN THE WRITE PACKET
002375  104203  001045                  MOV     #IN.01,R3               ;R3 -> INPUT DATA
002377  104307  001315                  MOV     LUNIT,R0
002401  100237                          MOV     R0,(R3)+                ;SET UNIT NUMBER
002402  114007                          CLR     R0
002403  100237                          MOV     R0,(R3)+                ;CLEAR VALUE
002404  104307  004215                  MOV     ST,R0
002406  100237                          MOV     R0,(R3)+                ;SAVE REGION ID
002407  104307  004216                  MOV     ST+1,R0
002411  100237                          MOV     R0,(R3)+                ;1ST HALF NAME SAVED
002412  104307  004217                  MOV     ST+2,R0
002414  100237                          MOV     R0,(R3)+                ;2ND HALF NAME SAVED
002415  104207  000001                  MOV     #T2DLL,R0               ;SET R0 = TEST 2 DOWNLINE LOAD
002417  020751                          CALL    HOSTRQ
                                ; *** IF HERE, DIDN'T REINIT UDA TO GET DATA
002420  104205  001045                  MOV     #IN.01,R5               ;R5 -> INPUT BUFFER
002422  114000  003754                  CLR     WRM+2                   ;CLEAR OFFSET
                                ; *** DOWN LINE LOAD PROGRAM
002424  104157                  DO.DI2: MOV     (R5),R0                 ;IS THE PROGRAM THERE?
002425  117407                          DEC     R0
002426  052556                          BNE     DO.DI8                  ;IF NOT, REPORT ERROR
                                ; *** GET THE PROGRAM FROM THE HOST
002427  104657  000005          1$:     MOV     5(R5),R0                ;UNIBUS ADDRESS PA
002431  104651  000006                  MOV     6(R5),R1                ;UNIBUS ADDRESS EA
002433  104652  000007                  MOV     7(R5),R2                ;BYTE COUNT
002435  104203  004215                  MOV     #ST,R3                  ;POINTER TO INPUT BUFFER
002437  106202  000200                  CMP     #STSIZE,R2              ;R2 > MAX BUFFER SIZE?
002441  032444                          BPL     2$                      ;IF NOT, CONTINUE
002442  104202  000200                  MOV     #STSIZE,R2              ;SET MAX BUFFER SIZE IN R2
002444  060013                  2$:     XFC     UREAD                   ;DO A UNIBUS READ
                                ; *** SET UP SDI COMMAND PACKETS
002445  104020  003746                  MOV     R2,CR.WRM+1             ;SET BYTE COUNT IN WRM PACKET
002447  105200  000006  003746          ADD     #6,CR.WRM+1             ;ADD TO COUNT TO INCLUDE PACKET LENGTH
002452  104650  000004  003753          MOV     4(R5),WRM+1             ;SET REGION ID
002455  104020  003755                  MOV     R2,WRM+3                ;SET BYTE COUNT IN BUFFER
002457  104303  004215                  MOV     ST,R3                   ;1ST DATA WORD IN R3
002461  110703                          SWAB    R3                      ;1ST DATA WORD IN UPPER BYTE
002462  103203  000377                  BIC     #LOBYTE,R3              ;CLEAR LOW BYTE
002464  101030  003755                  BIS     R3,WRM+3                ;SET DATA IN 1ST BUFFER WORD OF WRT MEM PACKET
002466  104020  001316                  MOV     R2,SAVREG               ;SAVE BYTE COUNT
002470  117402                          DEC     R2                      ;ADJUST BYTE COUNT
002471  103200  000377  004215          BIC     #LOBYTE,ST              ;CLEAR LOW BYTE OF 1ST DATA WORD OF PROGRAM
002474  101020  004215                  BIS     R2,ST                   ;REPLACE IT BY THE BYTE COUNT OF THE REST
002476  104204  003756                  MOV     #WRM+4,R4               ;R4 -> OUT BUFFER
002500  023054                          CALL    CONMEM                  ;CONVERT MEMORY
                                ; *** SEND TO THE DRIVE
002501  104302  003717                  MOV     SDI,R2                  ;RESTORE PORT INDICATOR
002503  023013                          CALL    WRTMEM                  ;SEND TO DRIVE
002504  115002                          TST     R2                      ;ALL OK?
002505  052313                          BNE     DO.DC4                  ;IF NOT, GO CLEAR DRIVE
002506  104302  001316                  MOV     SAVREG,R2               ;R2 = BYTE COUNT
002510  105020  003754                  ADD     R2,WRM+2                ;ADJUST OFFSET
002512  104657  000007                  MOV     7(R5),R0
002514  107027                          SUB     R2,R0
```

```
     58 002515  100657  000007                    MOV     R0,7(R5)                ;DECREMENT TOTAL BYTE COUNT/DONE?
     59 002517  012536                             BEQ     4$                      ;IF 0, EXIT
     60 002520  072536                             BMI     4$                      ;IF NEG, EXIT
     61 002521  104657  000005                    MOV     5(R5),R0
     62 002523  105027                             ADD     R2,R0
     63 002524  100657  000005                    MOV     R0,5(R5)                ;ELSE, ADJUST UNIBUS ADDRESS TO READ FROM
     64 002526  106027                             CMP     R2,R0                   ;IS PA ADDRESS LESS THAN BUFFER SIZE?
     65 002527  072535                             BMI     3$                      ;IF IT IS NOT, CONTINUE
     66 002530  104657  000006                    MOV     6(R5),R0
     67 002532  115407                             INC     R0                      ;ELSE, INCREMENT EA ADDRESS (CROSSED 0 BOUNDARY)
     68 002533  100657  000006                    MOV     R0,6(R5)
     69 002535  002427                  3$:       BR      1$                      ;READ IN NEXT BUFFER
     70
     71                                         ; *** SET UP DIAGNOSE COMMAND
     72 002536  114000  003754          4$:       CLR     WRM+2                   ;REINIT OFFSET
     73 002540  104200  000007  003746           MOV     #7,CR.WRM+1             ;REINIT BYTE COUNT OF INSTRUCTION
     74 002543  104650  000004  001046           MOV     4(R5),IN.02             ;SET UP REGION ID
     75 002546  002110                             BR      DIAGNS                  ;SEND DIAGNOSE COMMAND
     76
     77                                         ; *** JUST RECEIVED THE REGION ID TO DIAGNOSE TO.  GO DIAGNOSE IT.
     78 002547  104300  003721  001046  DO.DI3:  MOV     SAVRID,IN.02            ;STORE REGION ID FOR DIAGNOSE COMMAND
     79 002552  002110                             BR      DIAGNS                  ;GO DIAGNOSE
```

```
                                        .SBTTL  DIAGNOSE/REPORT ERROR -- NO DOWNLINE LOAD PROGRAM
                                ; *** NO DOWN LINE LOAD PROGRAM AFTER REINITED UDA
002553  104650  000001  003716  DO.DI7: MOV     1(R5),UNITNB            ;SET UP LOGICAL UNIT NUMBER

                                ; *** NO DOWN LINE LOAD PROGRAM, REPORT ERROR
002556  104654  000010          DO.DI8: MOV     10(R5),R4               ;R4 = 1ST WORD OF PROGRAM NAME
002560  023630                          CALL    DIV50                   ;DIVIDE BY 50 AND GET 3RD CHARACTER
002561  110707                          SWAB    R0                      ;PUT IN PROPER BYTE
002562  104070  004420                  MOV     R0,NAM.2                ;SET 3RD CHARACTER
002564  023630                          CALL    DIV50                   ;GET 2ND CHARACTER
002565  104070  004417                  MOV     R0,NAM.1                ;SET 2ND CHARACTER
002567  104047                          MOV     R4,R0                   ;R0 = RAD50 OF 1ST CHARACTER
002570  023644                          CALL    FNDASC                  ;GET ASCII EQUIVALENT
002571  110707                          SWAB    R0                      ;PUT IN PROPER BYTE
002572  101070  004417                  BIS     R0,NAM.1                ;SET 1ST CHARACTER
002574  104654  000011                  MOV     11(R5),R4               ;R4 = 2ND WORD OF PROGRAM NAME
002576  023630                          CALL    DIV50                   ;GET 6TH CHARACTER
002577  104070  004421                  MOV     R0,NAM.3                ;SET 6TH CHARACTER
002601  023630                          CALL    DIV50                   ;GET 5TH CHARACTER
002602  110707                          SWAB    R0                      ;PUT IN PROPER BYTE
002603  101070  004421                  BIS     R0,NAM.3                ;SET 5TH CHARACTER
002605  104047                          MOV     R4,R0                   ;R0 = RAD50 OF 4TH CHARACTER
002606  023644                          CALL    FNDASC                  ;FIND ASCII EQUIVALENT
002607  101070  004420                  BIS     R0,NAM.2                ;SET 4TH CHARACTER
002611                                  ERRHRD  MS56,NAM.1,NAM.2,NAM.3  ;REPORT ERROR
002611  104010  001316                                                          MOV R1,SAVREG
002613  104301  004421                                                          MOV NAM.3,R1
002615  100461                                                                  MOV R1,-(SP)
002616  104301  004420                                                          MOV NAM.2,R1
002620  100461                                                                  MOV R1,-(SP)
002621  104301  004417                                                          MOV NAM.1,R1
002623  100461                                                                  MOV R1,-(SP)
002624  104301  001316                                                          MOV SAVREG,R1
002626  021236                                                                  CALL RERROR     ;ERROR # 19.
002627  007023                                                                  .WORD <PRMS*2000>+<2*400>+ERRN
002630  002511                                                                  .WORD MS56
```

```
                                        .SBTTL  DIAGNOSE/GET EXTENDED STATUS
002631                          DO.DI9:
                                ; *** GET EXTENDED STATUS
002631  104200  177776  004167        MOV     #FFFE,RDM+1             ;REGION ID SET
002634  114000  004170                CLR     RDM+2                   ;CLEAR OFFSET
002636  104200  000020  004171        MOV     #16.,RDM+3              ;BYTE COUNT - 8
002641  022737                        CALL    RDMEM                   ;READ MEMORY
002642  115002                        TST     R2                      ;OK?
002643  052706                        BNE     DO.DIO                  ;IF NOT, EXIT
002644  104302  003717                MOV     SDI,R2                  ;RESET PORT INDICATOR
002646  104204  001012                MOV     #OUT.04,R4              ;R4 -> OUT BUFFER
002650  023054                        CALL    CONMEM                  ;CONVERT MEMORY
002651  104307  001012                MOV     OUT.04,R0               ;R0 =BYTE COUNT DRIVE GAVE
002653  107207  000016                SUB     #14.,R0                 ;ANY EXTENED STATUS?
                                                                        ; (BYTE COUNT DOESN'T ACCOUNT FOR STORAGE OF ITSELF)
002655  012705                        BEQ     DO.DI6                  ;IF NOT, EXIT
002656  105200  000020  004170        ADD     #16.,RDM+2              ;ADJUST OFFSET
002661  104070  004171                MOV     R0,RDM+3                ;RDM+3 = FINAL BYTE COUNT
002663  022737                        CALL    RDMEM                   ;READ MEMORY
002664  115002                        TST     R2                      ;ALL OK?
002665  052706                        BNE     DO.DIO                  ;IF NOT, EXIT
002666  104302  003717                MOV     SDI,R2                  ;RESET PORT INDICATOR
002670  104204  001022                MOV     #OUT.12,R4              ;R4 -> OUTPUT BUFFER
002672  023054                        CALL    CONMEM                  ;CONVERT MEMORY
002673                                MSSGE   MS58                    ;REPORT EXTENDED STATUS
002673  104200  002613  001010                MOV     #MS58,OUT.02
002676  100467                                MOV R0,-(SP)
002677  100461                                MOV R1,-(SP)
002700  104207  000015                MOV     #MESSAG,R0
002702  020751                        CALL    HOSTRQ
002703  104261                                MOV (SP)+,R1
002704  104267                                MOV (SP)+,R0
002705  002313          DO.DI6: BR      DO.DC4                  ;DO A DRIVE CLEAR
```

```
                                        .SBTTL  DIAGNOSE/SET UP RESPONSE TO HOST AND EXIT
                                ; *** SET UP RESPONSE PACKET
002706  104302  003717          DO.DIO: MOV     SDI,R2                  ;RESTORE PORT INDICATOR
002710  104300  001315  001007          MOV     LUNIT,OUT.01            ;SAVE UNIT NUMBER
002713  115000  004416                  TST     DIAG.1                  ;IF DIAG.1 = 0?
002715  052721                          BNE     12$                     ;IF NOT, SKIP
002716  114000  001010                  CLR     OUT.02                  ;SET UP PROPER OP-CODE VALUE
002720  002724                          BR      13$                     ;CONTINUE
002721  104200  000003  001010  12$:    MOV     #3,OUT.02               ;SET UP REQUEST
002724                          13$:
002724  104300  004215  001011          MOV     ST,OUT.03               ;REGION ID SET
002727  114000  001012                  CLR     OUT.04                  ;
002731  104200  177777  004416          MOV     #177777,DIAG.1          ;MAKE SURE DIAG.1 HAS NONE ZERO VALUE IN IT
                                                                        ; FOR NEXT TIME THROUGH
002734  115007                          TST     R0                      ;XFC    BREAK
002735  001761                          BR      GSTST8                  ;GO SEND REQUEST


                                ;END OF TESTING THIS DRIVE

002736  001501                  TESTEX: BR      TESTX                   ;GO BACK TO DRIVE SEQUENCER
```

```
                                        .SBTTL  TEST 2 SPECIFIC ROUTINES
                                        .SBTTL  READ MEMORY SUBROUTINE
                                ;++
                                ;       ROUTINE NAME:   RDMEM
                                ;
                                ;       DESCRIPTION:
                                ;       THIS ROUTINE DOES THE XFC READ MEMORY (FROM A SPECIFIED REGION
                                ;       AND OFFSET) OF DATA FROM THE DRIVE.  THIS ROUTINE CALLS THE
                                ;       SEND AND RCV (TALKER) ROUTINE AND JUDGES IF THE SDI COMMAND
                                ;       WAS PROPERLY SENT.
                                ;
                                ;       INPUT:  ALL PARAMETERS FOR THE READ MEMORY COMMAND
                                ;               MUST BE SET.  THAT MEANS REGION ID, THE OFFSET,
                                ;               AND THE BYTE COUNT MUST BE GIVEN THE APPRORIATE VALUES
                                ;               AND STORED IN THE RDM PACKET
                                ;
                                ;       OUTPUT: R2 = 0 MEANS ALL IS OK
                                ;               R2 NOT = 0, READ FAILED (EITHER SEND OR RECIEVE OF THE COMMAND)
                                ;               ST HAS THE READ DATA
                                ;--
002737                          RDMEM:
002737                                  PUSH    <R0,R1,R3>              ;SAVE THESE REGISTERS
002737  100467                                                                  MOV R0,-(SP)
002740  100461                                                                  MOV R1,-(SP)
002741  100463                                                                  MOV R3,-(SP)
002742  104203  004161                 MOV     #CR.RDM,R3              ;R3 -> RDM PACKET
002744  104302  003717                 MOV     SDI,R2                  ;R2 = PORT
002746  021102                          CALL    TALKER                  ;SEND COMMAND AND RECEIVE RESPONSE
002747  115003                          TST     R3                      ;ERRORS?
002750  012764                          BEQ     3$                      ;IF NONE SO FAR, CONTINUE
002751  110203                          ROL     R3                      ;ERROR ON SEND?
002752  042757                          BCC     2$                      ;IF CARRY CLEAR (RECEIVE ERROR) BRANCH
                                ; *** REPORT TRANSMISSION ERROR
002753                                  ERRHRD  MS27
002753  021236                                                                  CALL RERROR     ;ERROR # 20.
002754  001024                                                                  .WORD <PRMS*2000>+<2*400>+ERRN
002755  001316                                                                  .WORD MS27
002756  003005                          BR      5$                      ;ERROR EXIT
                                ; *** REPORT RECEPTION ERROR
002757                          2$:     ERRHRD  MS28
002757  021236                                                                  CALL RERROR     ;ERROR # 21.
002760  001025                                                                  .WORD <PRMS*2000>+<2*400>+ERRN
002761  001350                                                                  .WORD MS28
002762  023237                          CALL    TYPERR                  ;GO CHECK TYPE OF ERROR
002763  003005                          BR      5$                      ;ERROR EXIT
                                ; *** CHECK RESPONSE OP-CODE FROM DRIVE
002764  106207  000162          3$:     CMP     #RDM.EN,R0              ;CHECK RESPONSE
002766  013003                          BEQ     4$                      ;IF IT MATCHES, OK, EXIT
002767                                  ERRHRD  MS26,#RDM.EN,R0
002767  100467                                                                  MOV R0,-(SP)
002770  104010  001316                                                          MOV R1,SAVREG
002772  104201  000162                                                          MOV #RDM.EN,R1
002774  100461                                                                  MOV R1,-(SP)
002775  104301  001316                                                          MOV SAVREG,R1
002777  021236                                                                  CALL RERROR     ;ERROR # 22.
003000  005026                                                                  .WORD <PRMS*2000>+<2*400>+ERRN
003001  001226                                                                  .WORD MS26
```

```
 41 003002  003005                      BR      5$                      ;ERROR EXIT
 42 003003  114002              4$:     CLR     R2                      ;ALL OK, EXIT
 43 003004  003007                      BR      6$                      ;
 44 003005  104202  000001      5$:     MOV     #1,R2                   ;ERROR OCCURED, SET R2
 45 003007                      6$:     POP     <R3,R1,R0>
    003007  104263                                                                  MOV (SP)+,R3
    003010  104261                                                                  MOV (SP)+,R1
    003011  104267                                                                  MOV (SP)+,R0
 46 003012  000000                      RETURN
```

```
                                        .SBTTL  WRITE MEMORY SUBROUTINE
                                ;++
                                ;       WRTMEM
                                ;
                                ;       DESCRIPTION:
                                ;       THIS ROUTINE DOES THE XFC WRITE MEMORY TO A SPECIFIED
                                ;       REGION ID AND OFFSET WITH A BYTE COUNT AND DATA.
                                ;       THIS ROUTINE CALLS THE SEND AND RCV (TALKER) ROUTINE
                                ;       AND JUDGES IF THE SDI COMMAND WAS PROPERLY SENT.
                                ;
                                ;       INPUT:  ALL PARAMETERS FOR THE WRITE MEMORY COMMAND MUST BE SET.
                                ;               REGION ID, OFFSET, BYTE COUNT AND THE DATA MUST BE STORED
                                ;               IN THE PACKET
                                ;--
003013  104203  003745  WRTMEM: MOV     #CR.WRM,R3              ;R3->MEM WRITE PACKET
003015  104302  003717          MOV     SDI,R2                  ;R2 = PORT
003017  021102                  CALL    TALKER                  ;SEND COMMAND AND RECEIVE RESPONSE FROM DRIVE
003020  115003                  TST     R3                      ;SEE IF ERROR OCCURRED
003021  013035                  BEQ     3$                      ;IF NO ERRORS, BRANCH
003022  110203                  ROL     R3                      ;SHIFT HIGH BIT INTO CARRY BIT
003023  043030                  BCC     2$                      ;IF CARRY CLEAR (RECEIVE ERROR) BRANCH
                        ; *** REPORT TRANSMISSION ERROR
003024                          ERRHRD  MS17
003024  021236                                                  CALL RERROR     ;ERROR # 23.
003025  001027                                                  .WORD <PRMS*2000>+<2*400>+ERRN
003026  000744                                                  .WORD MS17
003027  003053                  BR      4$
                        ; *** REPORT RECEPTION ERROR
003030                  2$:     ERRHRD  MS18
003030  021236                                                  CALL RERROR     ;ERROR # 24.
003031  001030                                                  .WORD <PRMS*2000>+<2*400>+ERRN
003032  000777                                                  .WORD MS18
003033  023237                  CALL    TYPERR                  ;FIND OUT WHAT TYPE OF ERROR OCCURED
003034  003053                  BR      4$
                        ; *** SET UP RESPONSE PACKET
003035  106207  000176  3$:     CMP     #COMPLT,R0              ;DID IT COMPLETE
003037  013053                  BEQ     4$                      ;IF SO CONTINUE
003040                          ERRHRD  MS19,#COMPLT,R0
003040  100467                                                  MOV R0,-(SP)
003041  104010  001316                                         MOV R1,SAVREG
003043  104201  000176                                         MOV #COMPLT,R1
003045  100461                                                  MOV R1,-(SP)
003046  104301  001316                                         MOV SAVREG,R1
003050  021236                                                  CALL RERROR     ;ERROR # 25.
003051  005031                                                  .WORD <PRMS*2000>+<2*400>+ERRN
003052  001035                                                  .WORD MS19
003053  000000  4$:     RETURN
```

CONVERT MEMORY -- SKEWED BY BYTE

```
                                        .SBTTL  CONVERT MEMORY -- SKEWED BY BYTE
                                ;:+
                                ;       ROUTINE NAME:   CONMEM
                                ;
                                ;       DESCRIPTION:
                                ;       MEMORY IS SENT TO THE DM ROUTINE STARTING ON THE ODD BYTE BOUNDARY.
                                ;       THIS ROUTINE SHIFTS THE DATA TO START ON THE EVEN BYTE BOUNDARY AND
                                ;       STORES EACH WORD INTO A SPECIFIED BUFFER AREA.  THE DATA IS STORED AS
                                ;       FOLLOWS:
                                ;               ST:     DATA BYTE 0     BYTE COUNT
                                ;                       DATA BYTE 2     DATA BYTE 1
                                ;                       DATA BYTE 4     DATA BYTE 3
                                ;                       .....           DATA BYTE 5
                                ;
                                ;       AFTER EXECUTION, THD DATA IS STORED LIKE THIS:
                                ;               OUT.XX: DATA BYTE 1     DATA BYTE 0
                                ;                       DATA BYTE 3     DATA BYTE 2
                                ;                       DATA BYTE 5     DATA BYTE 4
                                ;                       .....           DATA BYTE 6
                                ;
                                ;       INPUT:  R4 -> BUFFER AREA'S FIRST LOCATION
                                ;               ST HAS INPUT DATA
                                ;
                                ;       OUTPUT: OUTPUT BUFFER HAS SHIFTED DATA
                                ;
                                ;:-
003054                          CONMEM: PUSH    <R0,R1,R2,R3,R4>
003054  100467                                                         MOV R0,-(SP)
003055  100461                                                         MOV R1,-(SP)
003056  100462                                                         MOV R2,-(SP)
003057  100463                                                         MOV R3,-(SP)
003060  100464                                                         MOV R4,-(SP)
003061  104203  004215                 MOV     #ST,R3          ;R3 -> BYTE COUNT
003063  104137                  MOV     (R3),R0         ;R0 = BYTE COUNT
003054  103207  177400                 BIC     #HIBYTE,R0      ;STRIP OFF EXTRANEOUS
003066  104231                  1$:     MOV     (R3)+,R1        ;R1 = EVEN BYTE OF DATA
003067  103201  000377                 BIC     #LOBYTE,R1      ;STRIP OFF EXTRANEOUS
003071  117407                  DEC     R0              ;DONE?
003072  013104                  BEQ     2$              ;IF YES, GO STORE LAST BYTE
003073  104132                  MOV     (R3),R2         ;R2 = ODD BYTE
003074  103202  177400                 BIC     #HIBYTE,R2      ;STRIP OFF EXTRANEOUS
003076  101021                  BIS     R2,R1           ;R1 HAS WHOLE WORD(BYTES REVERSED)
003077  110701                  SWAB    R1              ;SWITCH BYTES
003100  100241                  MOV     R1,(R4)+        ;STORE DATA
003101  117407                  DEC     R0              ;DONE?
003102  013106                  BEQ     3$              ;IF SO, EXIT(EVEN # OF BYTES)
003103  003066                  BR      1$              ;ELSE, CONTINUE
003104  110701                  2$:     SWAB    R1              ;ODD # BYTES TO GET HERE
003105  100141                  MOV     R1,(R4)         ;STORE LAST BYTE
003106                          3$:     POP     <R4,R3,R2,R1,R0>
003106  104264                                                         MOV (SP)+,R4
003107  104263                                                         MOV (SP)+,R3
003110  104262                                                         MOV (SP)+,R2
003111  104261                                                         MOV (SP)+,R1
003112  104267                                                         MOV (SP)+,R0
003113  000000                  RETURN
```

```
                                .SBTTL  GET BYTE COUNT
                        ;:+
                        ;       ROUTINE NAME:   GETBCN
                        ;
                        ;       DESCRIPTION:
                        ;       DATA IS READ BY THE DM PROGRAM AFTER THE DIAGNOSE COMMAND.
                        ;       WITHIN THIS DATA, THE DRIVE SPECIFIES IF MORE DATA NEEDS TO
                        ;       BE READ.  IF IT IS, THE BYTE COUNT MUST BE CONVERTED FROM
                        ;       ITS PRESENT FORM WHICH FOLLOWS THIS FORMAT:
                        ;
                        ;       BIT 15                                          BIT 0
                        ;       +----------------------+----------------------+
                        ;OUT.08:.       ASCII COUNT     .    BINARY COUNT      !
                        ;       +----------------------+----------------------+
                        ;
                        ;       THE TOTAL BYTE COUNT IS FOUND BY THIS FORMULA:
                        ;                       TBC = AC + (4*BC)
                        ;       OR THE TOTAL BYTE COUNT EQUALS THE ASCII COUNT PLUS FOUR
                        ;       TIMES THE BINARY COUNT.  THE BINARY COUNT IS THE NUMBER OF
                        ;       32 BIT BINARY VALUES INCLUDED IN THE REPORT.  THE ASCII
                        ;       COUNT IS THE NUMBER OF ASCII CHARACTERS INCLUDED IN THE
                        ;       REPORT.
                        ;
                        ;       INPUT:  OUT.08 = ASCII COUNT AND BINARY COUNT IN THE FORM STATED ABOVE
                        ;
                        ;       OUTPUT: RDM+3 HAS UPDATED BYTE COUNT.
                        ;
                        ;:-
003114                  GETBCN: PUSH    <R0,R1,R2,R3,R4,R5>
003114  100467                                                          MOV R0,-(SP)
003115  100461                                                          MOV R1,-(SP)
003116  100462                                                          MOV R2,-(SP)
003117  100463                                                          MOV R3,-(SP)
003120  100464                                                          MOV R4,-(SP)
003121  100465                                                          MOV R5,-(SP)
003122  114007                          CLR     R0
003123  110207                          ROL     R0                      ;CLEAR CARRY
003124  104307  001016                  MOV     OUT.08,R0               ;GET BYTE COUNT
003126  103207  177400                  BIC     #HIBYTE,R0              ;R0 = BINARY COUNT  # 32 BIT WORDS
003130  110207                          ROL     R0                      ;
003131  110207                          ROL     R0                      ;R0 = BINARY COUNT IN BYTES
003132  104301  001016                  MOV     OUT.08,R1               ;R1 = BYTE COUNT
003134  110701                          SWAB    R1                      ;ASCII COUNT IN LO BYTE
003135  103201  177400                  BIC     #HIBYTE,R1              ;R1 = ASCII COUNT
003137  104012                          MOV     R1,R2                   ;SAVE IN R2 = ASCII COUNT
003140  105017                          ADD     R1,R0                   ;R0 = TOTAL BYTE COUNT
                                ; *** IS SIZE OF THE PACKET TOO BIG FOR BUFFER AVAILABLE?
003141  104201  000052                  MOV     #42.,R1                 ;42. IS THE MAXIMUM # OF BYTES LEFT
003143  106017                          CMP     R1,R0                   ;IS TOTAL BYTE COUNT > 42.?
003144  033226                          BPL     1$                      ;IF NOT, GO STORE TOTAL BYTE COUNT
                                ; *** SAVE OUT.RQ DATA
003145  104204  001007                  MOV     #OUT.01,R4              ;R4-> OUT.RQ DATA
003147  104205  004355                  MOV     #ST+140,R5              ;R5->SAVE AREA
003151  104241                  4$:     MOV     (R4)+,R1                ;SAVE
003152  100251                          MOV     R1,(R5)+                ;
003153  106204  001043                  CMP     #OUT.29,R4              ;DONE?
003155  053151                          BNE     4$                      ;IF NOT, CONTINUE
```

```
                                ;
                                        MSSGE   MS30
003156  104200  001406  001010                                          MOV     #MS30,OUT.02
003161  100467                                                                  MOV R0,-(SP)
003162  100461                                                                  MOV R1,-(SP)
003163  104207  000015                                                  MOV     #MESSAG,R0
003165  020751                                                          CALL    HOSTRQ
003166  104261                                                                  MOV (SP)+,R1
003167  104267                                                                  MOV (SP)+,R0
                                ; *** RESTORE OUT.RQ DATA
003170  104204  001007                MOV     #OUT.01,R4              ;R4-> OUT.RQ DATA
003172  104205  004355                MOV     #ST+140,R5              ;R5->SAVE AREA
003174  104251                5$:     MOV     (R5)+,R1                ;RESTORE
003175  100241                        MOV     R1,(R4)+                ;
003176  106204  001043                CMP     #OUT.29,R4              ;DONE?
003200  053174                        BNE     5$                      ;IF NOT, CONTINUE
                                ;
003201  104073                        MOV     R0,R3                   ;SAVE ASCII COUNT + 4*BINARY COUNT IN R3
003202  104207  000052                MOV     #42.,R0                 ;ELSE, MAXIMUM BYTE COUNT IS STORED.
                                ; *** ADJUST ASCII COUNT IF TOO LARGE
003204  107023                        SUB     R2,R3                   ;R3 = BINARY COUNT
003205  106073                        CMP     R0,R3                   ;IF BINARY COUNT ALONE > 42.?
003206  073215                        BMI     2$                      ;IF SO, TO CLEAR ASCII COUNT AND ADJUST BINARY COUNT
                                ; *** ASCII COUNT IS > 42. TO GET HERE
003207  105032                        ADD     R3,R2                   ;R2 = TOTAL BYTE COUNT
003210  107072                        SUB     R0,R2                   ;R2 = ASCII COUNT ADJUSTMENT
003211  110702                        SWAB    R2                      ;PUT ASCII COUNT ADJUSTMENT IN UPPER BYTE
003212  107020  001016                SUB     R2,OUT.08               ;SUBTRACT FROM OUT.08 FOR HOST
003214  003226                        BR      1$                      ;EXIT
                                ; *** ADJUST BINARY COUNT IF TOO LARGE/CLEAR OUT ASCII COUNT
003215  103200  177400  001016  2$:     BIC     #HIBYTE,OUT.08          ;CLEAR HI BYTE OF OUT.08
003220  107073                        SUB     R0,R3                   ;R3 = BINARY COUNT ALONE
003221  117400  001016        3$:     DEC     OUT.08                  ;ADJUST BINARY COUNT
003223  107203  000004                SUB     #4,R3
003225  033221                        BPL     3$
                                ; *** STORE IN READ MEMORY PACKET AND EXIT
003226                          1$:
003226  104070  004171                MOV     R0,RDM+3                ;STORE IN READ MEMORY PACKET
003230                                  POP     <R5,R4,R3,R2,R1,R0>
003230  104265                                                                  MOV (SP)+,R5
003231  104264                                                                  MOV (SP)+,R4
003232  104263                                                                  MOV (SP)+,R3
003233  104262                                                                  MOV (SP)+,R2
003234  104261                                                                  MOV (SP)+,R1
003235  104267                                                                  MOV (SP)+,R0
003236  000000                        RETURN
```

```
                                        .SBTTL  TYPE WHAT KIND OF RECEIVE ERROR
                                ;:+
                                ;       TYPE ERROR
                                ;
                                ;       DESCRIPTION:
                                ;       THIS ROUTINE PRINTS A REPORT TO TELL WHAT KIND OF ERROR OCCURED IF A
                                ;       RECEIVE XFC DID NOT SUCCESSFULLY COMPLETE.
                                ;
                                ;       INPUT:  R3 = ERROR VALUE FROM THE XFC SAVED IN TALKER ROUTINE SHIFTED LEFT ONCE
                                ;                    IF = 1 TIMEOUT
                                ;                    IF = 2 1ST WORD WAS NOT A START FRAME
                                ;                    IF = 4 FRAMING ERROR ON SDI LEVEL 0 READ
                                ;                    IF = 10 CHECKSUM ERROR ON SDI LEVEL 0 READ
                                ;                    IF = 20 BUFFER SIZE WAS SMALLER THAN RESPONSE
                                ;
                                ;:-
003237  110603          TYPERR: ROR     R3                      ;READJUST R3
003240  114001                  CLR     R1
003241  110201                  ROL     R1                      ;CLEAR CARRY
003242  104031                  MOV     R3,R1                   ;STORE IN R1
003243  110601                  ROR     R1                      ;ERROR?
003244  043260                  BCC     1$                      ;IF NOT, CONTINUE
003245                          MSSGE   MS60                    ;TIMEOUT
003245  104200  002677  001010                    MOV     #MS60,OUT.02
003250  100467                                            MOV R0,-(SP)
003251  100461                                            MOV R1,-(SP)
003252  104207  000015                            MOV     #MESSAG,R0
003254  020751                                    CALL    HOSTRQ
003255  104261                                            MOV (SP)+,R1
003256  104267                                            MOV (SP)+,R0
003257  003360                  BR      6$                      ;EXIT
003260  110601          1$:     ROR     R1                      ;ERROR?
003261  043275                  BCC     2$                      ;IF NOT, CONTINUE
003262                          MSSGE   MS61                    ;1ST WORD WAS NOT A START FRAME
003262  104200  002725  001010                    MOV     #MS61,OUT.02
003265  100467                                            MOV R0,-(SP)
003266  100461                                            MOV R1,-(SP)
003267  104207  000015                            MOV     #MESSAG,R0
003271  020751                                    CALL    HOSTRQ
003272  104261                                            MOV (SP)+,R1
003273  104267                                            MOV (SP)+,R0
003274  003360                  BR      6$                      ;EXIT
003275  110601          2$:     ROR     R1                      ;ERROR?
003276  043312                  BCC     3$                      ;IF NOT, CONTINUE
003277                          MSSGE   MS62                    ;FRAMING ERROR ON SDI LEVEL 0 READ
003277  104200  002755  001010                    MOV     #MS62,OUT.02
003302  100467                                            MOV R0,-(SP)
003303  100461                                            MOV R1,-(SP)
003304  104207  000015                            MOV     #MESSAG,R0
003306  020751                                    CALL    HOSTRQ
003307  104261                                            MOV (SP)+,R1
003310  104267                                            MOV (SP)+,R0
003311  003360                  BR      6$                      ;EXIT
003312  110601          3$:     ROR     R1                      ;ERROR?
003313  043327                  BCC     4$                      ;IF NOT, CONTINUE
003314                          MSSGE   MS63                    ;CHECKSUM ERROR ON SDI LEVEL 0 READ
003314  104200  003016  001010                    MOV     #MS63,OUT.02
003317  100467                                            MOV R0,-(SP)
```

```
                                                                        MOV R1,-(SP)
                                                                MOV     #MESSAG,R0
                                                                CALL    HOSTRQ
                                                                        MOV (SP)+,R1
                                                                        MOV (SP)+,R0
003320  100461
003321  104207  000015
003323  020751
003324  104261
003325  104267
35 003326  003360                      BR      6$              ;EXIT
36 003327  110601              4$:     ROR     R1              ;ERROR?
37 003330  043344                      BCC     5$              ;IF NOT, CONTINUE
38 003331                               MSSGE   MS64            ;BUFFER SIZE SMALLER THAN RESPONSE
003331  104200  003057  001010                                 MOV     #MS64,OUT.02
003334  100467                                                  MOV R0,-(SP)
003335  100461                                                  MOV R1,-(SP)
003336  104207  000015                                         MOV     #MESSAG,R0
003340  020751                                                 CALL    HOSTRQ
003341  104261                                                          MOV (SP)+,R1
003342  104267                                                          MOV (SP)+,R0
39 003343  003360                      BR      6$              ;EXIT
40 003344                       5$:     MSSGE   MS65,R3         ;ERROR WASN'T PROPERLY SPECIFIED
003344  104200  003114  001010                                 MOV     #MS65,OUT.02
003347  104030  001011                                         MOV     R3,OUT.03
003351  100467                                                  MOV R0,-(SP)
003352  100461                                                  MOV R1,-(SP)
003353  104207  000015                                         MOV     #MESSAG,R0
003355  020751                                                 CALL    HOSTRQ
003356  104261                                                          MOV (SP)+,R1
003357  104267                                                          MOV (SP)+,R0
41 003360  000000               6$:     RETURN
```

```
                                .SBTTL  GET UNITS FROM HOST
003361                  GETU:
                        ;
                        ;       GETU WILL GET THE UNITS TO TEST
                        ;
                        ;
                        ;       POLL ALL PORTS AND FILL IN A UDA PORT INFORMATION TABLE (UNITS)
                        ;
003361  104205  000001          MOV     #1,R5           ; MOVE INITIAL MASK TO R5
003363  104204  003676          MOV     #UNITS,R4       ; R4 POINTS TO UNIT TABLE
003365                  1$:     PUSH    R4              ; SAVE R4
003365  100464                                                          MOV R4,-(SP)
003366  104052                  MOV     R5,R2           ; MOVE MASK TO R2
003367  020720          2$:     CALL    RDSTAT          ; GET STATUS
003370  102201  010000          BIT     #10000,R1       ; SEE IF ERROR OCCURRED (NO DRIVE ATTACHED)
003372  053463                  BNE     6$              ; IF NO DRIVE, BRANCH
003373  060011                  XFC     DINIT           ; INITILIZE THE DRIVE
003374  104207  005000          MOV     #5000,R0        ; MOVE TIMEOUT TO R0
003376  020720          14$:    CALL    RDSTAT          ; GET STATUS
003377  102201  010000          BIT     #10000,R1       ; SEE IF STATUS IS VALID
003401  053405                  BNE     3$              ; IF NOT, BRANCH
003402  102201  000001          BIT     #RCVRDY,R1      ; SEE IF RECEIVER READY ASSERTED
003404  053410                  BNE     5$              ; IF SO, BRANCH
003405  117407          3$:     DEC     R0              ; DECREMENT COUNT
003406  053376                  BNE     14$             ; IF INCOMPLETE, BRANCH
003407  003463                  BR      6$              ; NO VALID STATE
003410  104203  003730          MOV     #CR.GST,R3      ; R3 POINTS TO GET STATUS COMMAND
003412  104237                  MOV     (R3)+,R0        ; SET ADR OF SDI COMMAND BUFFER
003413  104231                  MOV     (R3)+,R1        ; SET BUFFER LENGTH
003414  060004                  XFC     SEND            ; SEND COMMAND
003415  115001                  TST     R1              ; DID UNIT ACCEPT COMMAND
003416  053463                  BNE     6$              ; IF SO, BRANCH
003417                          PUSH    R4              ; SAVE R4
003417  100464                                                          MOV R4,-(SP)
003420  104204  000012          MOV     #10.,R4         ; SET UP SHORT TIMEOUT
003422  104137          11$:    MOV     (R3),R0         ; SET DATA BUFFER ADDRESS
003423  104631  000001          MOV     1(R3),R1        ; SET BUFFER LENGTH
003425  060005                  XFC     RCV             ; SEND RECEIVE SDI COMMAND
003426  115001                  TST     R1              ; DID ERROR OCCUR
003427  013437                  BEQ     13$             ; IF NOT, BRANCH
003430  106201  000001          CMP     #1,R1           ; SEE IF TIMEOUT
003432  053435                  BNE     12$             ; IF NOT, BRANCH
003433  117404                  DEC     R4              ; DECREMENT TIMEOUT VALUE
003434  053422                  BNE     11$             ; IF NOT TIMEOUT, BRANCH
003435                  12$:    POP     R4              ; RESTORE R4
003435  104264                                                          MOV (SP)+,R4
003436  003463                  BR      6$              ; BRANCH TO EXIT
003437                  13$:    POP     R4              ; RESTORE R4
003437  104264                                                          MOV (SP)+,R4
003440  104307  004215          MOV     ST,R0           ; R0 HAS UNIT NUMBER
003442  104072                  MOV     R0,R2           ; COPY R0 TO R2
003443  103207  170000          BIC     #^CHBHINB,R0    ; R0 HAS UNIT NUMBER
003445  101207  040000          BIS     #40000,R0       ; FLAG UNIT AS NOT TESTED
003447  110702                  SWAB    R2              ; SWAP R2'S BYTES
003450  110602                  ROR     R2              ; MOVE SUBUNIT MASK TO LO NIBBLE
003451  110602                  ROR     R2              ; MOVE SUBUNIT MASK TO LO NIBBLE
003452  110602                  ROR     R2              ; MOVE SUBUNIT MASK TO LO NIBBLE
```

GET UNITS FROM HOST

```
 54 003453  110602                        ROR     R2              ; MOVE SUBUNIT MASK TO LO NIBBLE
 55 003454  103202  177760                BIC     #LBLONB,R2      ; CLEAR ALL BUT SUBUNIT BITS
 56 003456  110602                4$:     ROR     R2              ; MOVE SUBUNIT BIT TO CARRY
 57 003457  043463                        BCC     6$              ; IF NO MORE SUBUNITS, BRANCH
 58 003460  100247                        MOV     R0,(R4)+        ; MOVE SUBUNIT NUMBER TO TABLE
 59 003461  115407                        INC     R0              ; INCREMENT SUBUNIT NUMBER
 60 003462  003456                        BR      4$              ; BRANCH
 61 003463                        6$:     POP     R4              ; R4 POINTS TO START OF UNIT JUST HANDLED
    003463  104264                                                                  MOV (SP)+,R4
 62 003464  105204  000004                ADD     #4,R4           ; R4 WILL POINT TO NEXT UNIT (CLEAR CARRY FOR ROL)
 63 003466  110205                        ROL     R5              ; R5 HAS NEXT UNIT PORT MASK
 64 003467  106205  000020                CMP     #20,R5          ; SEE IF ALL PORTS TESTED
 65 003471  053365                        BNE     1$              ; IF NOT, BRANCH
```

```
                                        ;
                                        ;       NOW GET THE PLUG NUMBERS TO TEST AND FIND THEM IN THE TABLE
                                        ;
003472  104207  000012                    MOV     #UTOTST,R0      ; GET WHAT SUBUNIT NUMBERS TO TEST REQUEST
003474  020751                            CALL    HOSTRQ          ; GET THE PLUG NUMBERS
003475  104207  001045                    MOV     #IN.01,R0       ; R0 POINTS TO UNIT NUMBERS TO TEST
003477  104201  003676            7$:     MOV     #UNITS,R1       ; R1 POINTS TO UDA PORT INFORMATION
003501  104112                    8$:     MOV     (R1),R2         ; R2 HAS UNIT
003502  073511                            BMI     9$              ; IF NONE, BRANCH
003503  103202  170000                    BIC     #^CHBHINB,R2    ; CLEAR THE 'NOT TESTED BIT' FOR COMPARISON
003505  106172                            CMP     (R0),R2         ; SEE IF IT IS A UNIT TO TEST
003506  053511                            BNE     9$              ; NO MATCH
003507  100112                            MOV     R2,(R1)         ; SAVE UNIT AS ONE TO TEST
003510  003516                            BR      10$             ; LOOK FOR THE NEXT ONE
003511  115401                    9$:     INC     R1              ; LOOK AT NEXT UNIT
003512  106201  003716                    CMP     #UNITS+16.,R1   ; SEE IF ENTIRE TABLE SEARCHED
003514  053501                            BNE     8$              ; IF NOT, BRANCH
003515  060000                            XFC     BREAK           ; ERROR ********
003516  115407                    10$:    INC     R0              ; POINT TO NEXT UNIT TO TEST
003517  104172                            MOV     (R0),R2         ; CHECK NEXT UNIT
003520  033477                            BPL     7$              ; FIND IN UDA PORT INFORMATION
003521  000000                            RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                        .SBTTL  GET STATUS SUBROUTINE
                                ;
                                ;       GET STATUS
                                ;
                                ;       OUTPUT  R3 = 0 IF OK
                                ;               R3 - 1 IF ERROR
                                ;
003522  104203  003730  GTSTAT: MOV     #CR.GST,R3              ;POINT TO GET STATUS COMMAND
003524  021102                  CALL    TALKER                  ; SEND AND RECEIVE COMMAND
003525  115003                  TST     R3                      ; SEE IF ERROR OCCURED
003526  013541                  BEQ     12$                     ; IF NOT, BRANCH
003527  110203                  ROL     R3                      ; SEE IF RECEIVE ERROR
003530  043535                  BCC     11$                     ; IF SO, BRANCH
003531                          ERRHRD  MS7
                                                                        CALL RERROR     ;ERROR # 26.
003531  021236                                                          .WORD <PRMS*2000>+<2*400>+ERRN
003532  001032                                                          .WORD MS7
003533  000316
003534  003557                  BR      ER.END
003535                  11$:    ERRHRD  MS8
                                                                        CALL RERROR     ;ERROR # 27.
003535  021236                                                          .WORD <PRMS*2000>+<2*400>+ERRN
003536  001033                                                          .WORD MS8
003537  000350
003540  003557                  BR      ER.END
                                ; *** LOOK AT RESPONSE OP-CODE FROM DRIVE -- SHOULD BE DATA RESPONSE
003541  106207  000366  12$:    CMP     #STSRES,R0              ;CHECK OP-CODE
003543  013562                  BEQ     OK.END                  ;BRANCH IF A MATCH
003544                          ERRHRD  MS9,#STSRES,R0          ;WRONG RESPONSE FROM DRIVE
                                                                        MOV R0,-(SP)
003544  100467                                                          MOV R1,SAVREG
003545  104010  001316                                                  MOV #STSRES,R1
003547  104201  000366                                                  MOV R1,-(SP)
003551  100461                                                          MOV SAVREG,R1
003552  104301  001316                                                  CALL RERROR     ;ERROR # 28.
003554  021236                                                          .WORD <PRMS*2000>+<2*400>+ERRN
003555  005034                                                          .WORD MS9
003556  000405
003557  104203  000001  ER.END: MOV     #1,R3                   ;R3 - NONE ZERO VALUE WHEN DONE(ERROR)
003561  003563                  BR      EX.END                  ;EXIT
003562  114003          OK.END: CLR     R3                      ;R3 - 0 WHEN DONE 'NO ERROR)
003563  000000          EX.END: RETURN
```

```
                                        .SBTTL  CLEAR DRIVE SUBROUTINE
                                ;
                                ;       DRIVE CLEAR
                                ;
                                ;       OUTPUT  R3 = 0 IF OK
                                ;               R3 = 1 IF ERROR
                                ;
 003564  104203  003736         CLRDRV: MOV     #CR.DRC,R3              ; POINT TO DRIVE CLEAR COMMAND
 003566  104301  004216                 MOV     ST+1,R1                 ; GET ERROR BYTE
 003570  103201  177400                 BIC     #HIBYTE,R1              ; CLEAR UNUSED BITS
 003572  104010  003744                 MOV     R1,DRCBYT               ; MOVE TO DRIVE CLEAR COMMAND
 003574  021102                         CALL    TALKER                  ; SEND AND RECEIVE COMMAND
 003575  115003                         TST     R3                      ; SEE IF ERROR OCCURED
 003576  013611                         BEQ     12$                     ; IF NOT, BRANCH
 003577  110203                         ROL     R3                      ; SEE IF RECEIVE ERROR
 003600  0436C5                         BCC     11$                     ; IF SO, BRANCH
 003601                                 ERRHRD  MS10
 003601  021236                                                                 CALL RERROR     ;ERROR # 29.
 003602  001035                                                                 .WORD <PRMS*2000>+<2*400>+ERRN
 003603  000475                                                                 .WORD MS10
 003604  003557                         BR      ER.END
 003605                         11$:    ERRHRD  MS11
 003605  021236                                                                 CALL RERROR     ;ERROR # 30.
 003606  001036                                                                 .WORD <PRMS*2000>+<2*400>+ERRN
 003607  000527                                                                 .WORD MS11
 003610  003557                         BR      ER.END
                                ; *** CHECK RESPONSE CODE
 003611  106207  000176         12$:    CMP     #COMPLT,R0              ; SEE IF CORRECT RESPONSE CODE RECEIVED
 003613  013562                         BEQ     OK.END                  ; IF SUCCESSFUL, BRANCH
 003614                                 ERRHRD  MS12,#COMPLT,R0         ;CLEAR ERROR FAILED
 003614  100467                                                                 MOV R0,-(SP)
 003615  104010  001316                                                         MOV R1,SAVREG
 003617  104201  000176                                                         MOV #COMPLT,R1
 003621  100461                                                                 MOV R1,-(SP)
 003622  104301  001316                                                         MOV SAVREG,R1
 003624  021236                                                                 CALL RERROR     ;ERROR # 31.
 003625  005037                                                                 .WORD <PRMS*2000>+<2*400>+ERRN
 003626  000564                                                                 .WORD MS12
 003627  003557                         BR      ER.END
```

```
DIVIDE BY OCTAL 50 AND FIND ASCII EQUIVALENT


                                        .SBTTL  DIVIDE BY OCTAL 50 AND FIND ASCII EQUIVALENT
                                ;++
                                ;       DIVIDE BY 50
                                ;
                                ;       DESCRIPTION:    RAD 50 VALUE IN R4 IS DIVIDED BY 50(OCTAL)
                                ;                       TO STRIP OFF A SINGLE CHARACTER INTO R0.
                                ;                       JUMP TC FNDASC TO FIND THE ASCII EQUIVALENT
                                ;
                                ;       INPUT:  R4 HAS RAD50 VALUE
                                ;
                                ;       OUTPUT: R0 HAS ASCII CHARACTER
                                ;--
003630                  DIV50:  PUSH    <R1>
003630  100461                                                                 MOV R1,-(SP)
003631  114001                  CLR     R1                      ;R1 = QUO
003632  115401          1$:     INC     R1                      ;KEEP TRACK OF # OF LOOPS
003633  107204  000050          SUB     #50,R4                  ;DONE?
003635  033632                  BPL     1$                      ;IF NOT, LOOP
003636  105204  000050          ADD     #50,R4                  ;R4 = REMAINDER
003640  104047                  MOV     R4,R0                   ;STORE REMAINDER IN R0
003641  117401                  DEC     R1                      ;R1 HAS 1 MORE THAN IT SHOULD
003642  104014                  MOV     R1,R4                   ;QUO IN R4
003643                          POP     <R1>
003643  10426                                                                   MOV (SP)+,R1

                                ;++
                                ;       FIND ASCII
                                ;
                                ;       INPUT:  R0 HAS RAD 50 CHARACTER
                                ;--
003644  115007          FNDASC: TST     R0                      ;IS = 0?
003645  053651                  BNE     3$                      ;IF NOT, CONTINUE
003646  104207  000040          MOV     #40,R0                  ;IF SO, SPACE SET
003650  000000                  RETURN
003651  106207  000032  3$:     CMP     #32,R0                  ; > LARGEST LETTER
003653  073657                  BMI     4$                      ;IF SO, CONTINUE
003654  105207  000100          ADD     #100,R0                 ;SET ASCII LETTER
003656  000000                  RETURN
003657  106207  000033  4$:     CMP     #33,R0                  ; = 33?
003661  053665                  BNE     5$                      ;IF NOT, CONTINUE
003662  104207  000044          MOV     #44,R0                  ;SET '$'
003664  000000                  RETURN
003665  106207  000034  5$:     CMP     #34,R0                  ; - 34?
003667  053673                  BNE     6$                      ;IF NOT, CONTINUE
003670  104207  000056          MOV     #56,R0                  ;SET '.'
003672  000000                  RETURN
003673  105207  000022  6$:     ADD     #22,R0                  ;ELSE ADD 22(OCTAL)FOR NUMERAL CHARACTER
003675  000000                  RETURN
```

```
                                        ;PROGRAM VARIABLES

                                        ;UNIT NUMBER STORAGE FOR DISK DRIVES TO TEST

003676  177777  000000  000000  UNITS:  .WORD   -1,0,0,0               ;LOGICAL UNIT NUMBER IF POSITIVE
003701  000000
003702  177777  000000  000000          .WORD   -1,0,0,0               ;DISK DRIVE NOT TO BE TESTED IF NEGATIVE
003705  000000
003706  177777  000000  000000          .WORD   -1,0,0,0
003711  000000
003712  177777  000000  000000          .WORD   -1,0,0,0
003715  000000

003716  000000                  UNITNB: .WORD   0                       ;NUMBER OF UNIT CURRENTLY UNDER TEST
                                                                        ;POINTER INTO TABLE ABOVE

003717  000000                  SDI:    .WORD   0                       ;SDI INTERCONNECT CODE FOR XFC CALLS

003720  000000                  SAVSTA: .WORD   0                       ;SAVE STATUS
003721  000000                  SAVRID: .WORD   0                       ;SAVE REGION ID

003722     377      350         ECHOD:  .BYTE   377,ECHOC               ;ECHO DATA TO SEND TO DRIVE
003723     000      350                 .BYTE   000,ECHOC
003724     252      350                 .BYTE   252,ECHOC
003725     360      350                 .BYTE   360,ECHOC
003726     017      350                 .BYTE   017,ECHOC
003727  000000                          .WORD   0                       ;END MARKER

                                        ;COMMAND BUFFERS FOR SEND XFC

003730                          CR.GST: MSG     GETST,1,ST,7                    ; GET STATUS SDI INFORMATION
003730  003735                          .WORD   GETST                           ;ADDRESS OF COMMAND
003731  000001                          .WORD   1                       ;SIZE OF COMMAND IN BYTES
003732  004215                          .WORD   ST                              ;ADDRESS OF REPLY
003733  000007                          .WORD   7                       ;SIZE OF REPLY IN WORDS
003734  000000                          .WORD                           ; SUCCESSFUL COMPLETION CODE
003735     000      011         GETST:  .BYTE   0,GETSTA
                                        ;
003736                          CR.DRC: MSG     DRC,2,ST,7                      ; DRIVE CLEAR SDI INFORMATION
003736  003743                          .WORD   DRC                             ;ADDRESS OF COMMAND
003737  000002                          .WORD   2                       ;SIZE OF COMMAND IN BYTES
003740  004215                          .WORD   ST                              ;ADDRESS OF REPLY
003741  000007                          .WORD   7                       ;SIZE OF REPLY IN WORDS
003742  000000                          .WORD                           ; SUCCESSFUL COMPLETION CODE
003743     000      005         DRC:    .BYTE   0,DRVCLR
003744  177777                  DRCBYT: .WORD   177777
                                        ;
003745                          CR.WRM: MSG     WRM,7,ST,7                      ; MEMORY WRITE COMMAND INFO
003745  003752                          .WORD   WRM                             ;ADDRESS OF COMMAND
003746  000007                          .WORD   7                       ;SIZE OF COMMAND IN BYTES
003747  004215                          .WORD   ST                              ;ADDRESS OF REPLY
003750  000007                          .WORD   7                       ;SIZE OF REPLY IN WORDS
003751  000000                          .WORD                           ; SUCCESSFUL COMPLETION CODE
003752     000      017         WRM:    .BYTE   0,WRM.OP
003753  000000  000000  000000          .WORD   0,0,0
003756  000000  000000  000000          .WORD   0,0,0
003761                          BUF1:   .BLKW   200
```

```
 39                                     ;
 40 004161                              CR.RDM: MSG     RDM,6,ST,STSIZE         ; MEMORY READ COMMAND INFO
    004161  004166                              .WORD RDM                       ;ADDRESS OF COMMAND
    004162  000006                              .WORD 6                 ;SIZE OF COMMAND IN BYTES
    004163  004215                              .WORD ST                        ;ADDRESS OF REPLY
    004164  000200                              .WORD STSIZE                    ;SIZE OF REPLY IN WORDS
    004165  000000                              .WORD                   ; SUCCESSFUL COMPLETION CODE
 41 004166     000      215                     RDM:    .BYTE   0,RDM.OP
 42 004167  000000  000000  000000              .WORD   0,0,0,0
    004172  000000
 43 004173  000000  000000  000000              .WORD   0,0,0,0
    004176  000000
 44                                     ;
 45 004177                              CR.GCR: MSG     GCR,1,ST,12.            ; GET CHARACTERISTICS
    004177  004204                              .WORD GCR                       ;ADDRESS OF COMMAND
    004200  000001                              .WORD 1                 ;SIZE OF COMMAND IN BYTES
    004201  004215                              .WORD ST                        ;ADDRESS OF REPLY
    004202  000014                              .WORD 12.                       ;SIZE OF REPLY IN WORDS
    004203  000000                              .WORD                   , SUCCESSFUL COMPLETION CODE
 46 004204     000      207                     GCR:    .BYTE   0,GETCHR        ; GET CHARACTERISTICS
 47                                     ;
 48 004205                              LONG:                                   ; ALL LONG TIMEOUT COMMANDS FOLLOW
 49                                     ;
 50 004205                              CR.DIA: MSG     DIA,3,ST,7              ; DIAGNOSE COMMAND INFORMATION
    004205  004212                              .WORD DIA                       ;ADDRESS OF COMMAND
    004206  000003                              .WORD 3                 ;SIZE OF COMMAND IN BYTES
    004207  004215                              .WORD ST                        ;ADDRESS OF REPLY
    004210  000007                              .WORD 7                 ;SIZE OF REPLY IN WORDS
    004211  000000                              .WORD                   ; SUCCESSFUL COMPLETION CODE
 51 004212     000      003                     DIA:    .BYTE   0,DIA.OP
 52 004213  000000                              .WORD   0
 53 004214  000000                              .WORD   0
 54
 55                                     ;DRIVE RESPONSE BUFFERS
 56             000200                  STSIZE  =       200
 57 004215                              ST:     .BLKW   200
 58
 59                                     ;
 60                                     ;       TEST 2 SPECIFIC MEMORY LOCATIONS
 61                                     ;
 62 004415  000000                      S.ERR:  .WORD   0                       ; = 0, HARD ERROR
 63                                                                             ; NOT = 0, SOFT ERROR
 64 004416  000000                      DIAG.1: .WORD   0                       ; = 0, 1ST DIAGNOSE COMMAND
 65                                                                             ; NOT = 0, DRIVE TESTED BEFORE
 66 004417  000000                      NAM.1:  .WORD   0                       ;HOLD PROGRAM NAME
 67 004420  000000                      NAM.2:  .WORD   0
 68 004421  000000                      NAM.3:  .WORD   0
 69 004422  000000                              .WORD   0
 70                                     ;
 71                                     ;       TEST 2 SPECIFIC OP CODES AND END CODES
 72                                     ;
 73             000017                  WRM.OP  =       17                      ; WRITE MEMORY COMMAND
 74             000215                  RDM.OP  =       215                     ; READ MEMORY COMMAND
 75             000003                  DIA.OP  =       3                       ; DIAGNOSE COMMAND
 76             000207                  GETCHR  =       207                     ; GET COMMON CHARACTERISTICS COMMAND
 77             000162                  RDM.EN  =       162                     ; READ MEMORY RESPONSE
 78             000374                  DIA.EN  =       374                     ; DIAGNOSE RESPONSE
```

```
                                GCC.EN  =       170             ; GET COMMON CHARACTERISTICS
                                ;
                                ;       DIAGNOSE ET AND DA
                                ;
100000                          ERRTYP  =       100000          ;ERROR TYPE ET
040000                          DATAVL  =        40000          ;DATA AVAILABLE
                                ;
                                ;       REGION ID'S
                                ;
177774                          FFFC    =       FFFD - 1
177775                          FFFD    =       177775
177776                          FFFE    =       FFFD + 1
```

```
                                ;MESSAGE STORAGE OVERLAY

004423                          FREE:                                   ; FREE AREA STARTS HERE
004423                                  DMOVLY MS,0
004423  150426                          .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                        .NLIST BEX

000000  042  103  101  MS1:     .ASCII\'CANNOT RECEIVE VALID DRIVE STATE FROM DRIVE'N\
000027  042  103  110           .ASCII\'CHECK IF DRIVE IS POWERED ON.'N\
000047  000                     .BYTE 0
000050  042  104  122  MS2:     .ASCII\'DRIVE STATE RECEIVED HAS BAD PARITY'N\
000073  000                     .BYTE 0
000074  042  104  122  MS3:     .ASCII\'DRIVE IS NOT ASSERTING RECEIVER READY IN DRIVE STATE'N\
000127  000                     .BYTE 0
000130  042  124  111  MS4:     .ASCII\'TIME-OUT ON SEND OF ECHO COMMAND TO DRIVE'N\
000156  042  040  040           .ASCII\'  ECHO DATA  'H8N\
000167  000                     .BYTE 0
000170  042  124  111  MS5:     .ASCII\'TIME-OUT ON RECEIVE OF ECHO RESPONSE FROM DRIVE'N\
000221  042  040  040           .ASCII\'  ECHO DATA  'H8N\
000232  000                     .BYTE 0
000233  042  105  103  MS6:     .ASCII\'ECHO COMMAND RESPONDED WITH DIFFERENT DATA'N\
000261  042  040  040           .ASCII\'  ECHO DATA SENT      'H16N\
000277  042  040  040           .ASCII\'  ECHO DATA RECEIVED  'H16N\
000315  000                     .BYTE 0
000316  042  124  111  MS7:     .ASCII\'TIME-OUT ON SEND OF GET STATUS COMMAND TO DRIVE'N\
000347  000                     .BYTE 0
000350  042  124  111  MS8:     .ASCII\'TIME-OUT ON RECEIVE OF GET STATUS RESPONSE FROM DRIVE'N\
000404  000                     .BYTE 0
000405  042  107  105  MS9:     .ASCII\'GET STATUS COMMAND DID NOT RETURN EXPECTED RESPONSE CODE'N\
000442  042  040  040           .ASCII\'  EXPECTED RESPONSE  'H8N\
000457  042  040  040           .ASCII\'  ACTUAL RESPONSE    'H8N\
000474  000                     .BYTE 0
000475  042  124  111  MS10:    .ASCII\'TIME-OUT ON SEND OF DRIVE CLEAR COMMAND TO DRIVE'N\
000526  000                     .BYTE 0
000527  042  124  111  MS11:    .ASCII\'TIME-OUT OF RECEIVE OF DRIVE CLEAR RESPONSE FROM DRIVE'N\
000563  000                     .BYTE 0
000564  042  104  122  MS12:    .ASCII\'DRIVE CLEAR COMMAND DID NOT RETURN EXPECTED RESPONSE CODE'N\
000622  042  040  040           .ASCII\'  EXPECTED RESPONSE  'H8N\
000637  042  040  040           .ASCII\'  ACTUAL RESPONE     'H8N\
000654  000                     .BYTE 0
000655  042  105  122  MS16:    .ASCII\'ERROR BIT SET IN GET STATUS RESPONSE AFTER DRIVE CLEAR COMMAND'N\
000715  042  040  040           .ASCII\'  GET STATUS RESPONSE  'H16N\
000734  123  062  063           .ASCII\S23H16N\
000737  123  062  063           .ASCII\S23H16N\
000743  000                     .BYTE 0
000744  042  124  111  MS17:    .ASCII\'TIME-OUT ON SEND OF MEMORY WRITE COMMAND TO DRIVE'N\
000776  000                     .BYTE 0
000777  042  105  122  MS18:    .ASCII\'ERROR DURING RECEIVE OF MEMORY WRITE RESPONSE FROM DRIVE'N\
001034  000                     .BYTE 0
001035  042  127  122  MS19:    .ASCII\'WRITE MEMORY COMMAND DID NOT RETURN EXPECTED RESPONSE CODE'N\
001073  042  040  040           .ASCII\'  EXPECTED RESPONSE 'H8N\
001110  042  040  040           .ASCII\'  ACTUAL RESPONSE   'H8N\
001124  000                     .BYTE   0
001125  042  111  116  MS20:    .ASCII\'INVALID INPUT.  HOST PROGRAM DID NOT GIVE PROPER RESPONSE'N\
001163  042  040  040           .ASCII\'  EXPECTED VALUE SHOULD BE BETWEEN 0 AND 3'N\
001211  042  040  040           .ASCII\'  ACTUAL VALUE WAS 'D8N\
001225  000                     .BYTE 0
```

```
     57 001226      042     122     105  MS26:   .ASCII\'READ MEMORY COMMAND DID NOT RETURN EXPECTED RESPONSE CODE'N\
     58 001264      042     040     040          .ASCII\'  EXPECTED RESPONSE  '2H8N\
     59 001300      042     040     040          .ASCII\'  ACTUAL RESPONSE    '2H8N\
     60 001315      000                          .BYTE 0
     61 001316      042     124     111  MS27:   .ASCII\'TIME-OUT ON SEND OF MEMORY READ COMMAND TO DRIVE'N\
     62 001347      000                          .BYTE 0
     63 001350      042     105     122  MS28:   .ASCII\'ERROR DURING RECEIVE OF MEMORY READ RESPONSE FROM DRIVE'N\
     64 001405      000                          .BYTE 0
     65 001406      042     106     117  MS30:   .ASCII\'FOLLOWING REPORT HAS BEEN TRUNCATED DUE TO SIZE'N\
     66 001437      000                          .BYTE 0
     67 001440      042     104     111  MS36:   .ASCII\'DIAGNOSE COMMAND DID NOT RETURN EXPECTED RESPONSE CODE'N\
     68 001474      042     040     040          .ASCII\'  EXPECTED RESPONSE  '2H8N\
     69 001511      042     040     040          .ASCII\'  ACTUAL RESPONSE    '2H8N\
     70 001525      000                          .BYTE 0
     71 001526      042     124     111  MS37:   .ASCII\'TIME-OUT ON SEND OF DIAGNOSE COMMAND TO DRIVE'N\
     72 001556      000                          .BYTE 0
     73 001557      042     105     122  MS38:   .ASCII\'ERROR DURING RECEIVE OF DIAGNOSE RESPONSE FROM DRIVE'N\
     74 001612      000                          .BYTE 0
     75 001613      042     125     116  MS40:   .ASCII\'UNDETERMINABLE RESPONSE FROM THE DRIVE GIVEN DIAGNOSTIC COMMAND 'N\
     76 001654      042     040     040          .ASCII\'  TEST NUMBER  '2H16N\
     77 001667      042     040     040          .ASCII\'  DRIVE TYPE    '2H8N\
     78 001701      042     040     040          .ASCII\'  ERROR NUMBER '2H16N\
     79 001713      122     062                  .ASCII\R2\
     80 001714      000                          .BYTE 0
     81 001715      042     111     116  MS41:   .ASCII\'INFORMATION SENT BACK FROM THE DRIVE IS BEING PRESENTED.'N\
     82 001752      042     040     040          .ASCII\'  TEST NUMBER  '2H16N\
     83 001765      042     040     040          .ASCII\'  DRIVE TYPE    '2H8N\
     84 001777      042     040     040          .ASCII\'  ERROR NUMBER '2H16N\
     85 002011      122     062                  .ASCII\R2\
     86 002012      000                          .BYTE 0
     87 002013      042     104     122  MS42:   .ASCII\'DRIVE DIAGNOSTIC REPORTS A HARD ERROR'N\
     88 002037      042     040     040          .ASCII\'  TEST NUMBER  '2H16N\
     89 002051      042     040     040          .ASCII\'  DRIVE TYPE    '2H8N\
     90 002063      042     040     040          .ASCII\'  ERROR NUMBER '2H16N\
     91 002076      122     062                  .ASCII\R2\
     92 002077      000                          .BYTE 0
     93 002100      042     104     122  MS43:   .ASCII\'DRIVE DIAGNOSTIC REPORTS A SOFT ERROR'N\
     94 002124      042     040     040          .ASCII\'  TEST NUMBER  '2H16N\
     95 002136      042     040     040          .ASCII\'  DRIVE TYPE    '2H8N\
     96 002150      042     040     040          .ASCII\'  ERROR NUMBER '2H16N\
     97 002163      122     062                  .ASCII\R2\
     98 002164      000                          .BYTE 0
     99 002165      042     124     111  MS50:   .ASCII\'TIME-OUT ON SEND OF GET UNIT CHARACTERISTICS COMMAND TO DRIVE'N\
    100 002225      000                          .BYTE   0
    101 002226      042     124     111  MS51:   .ASCII\'TIME-OUT ON RECEIVE OF GET UNIT CHARACTERISTICS COMMAND FROM DRIVE'N\
    102 002270      000                          .BYTE   0
    103 002271      042     107     105  MS52:   .ASCII\'GET UNIT CHARACTERISTICS COMMAND DID NOT RETURN EXPECTED RESPONSE CODE'N\
    104 002335      042     040     040          .ASCII\'  EXPECTED RESPONSE  '2H8N\
    105 002352      042     040     040          .ASCII\'  ACTUAL RESPONSE    '2H8N\
    106 002367      000                          .BYTE   0
    107 002370      116                  MS53:   .ASCII\N\
    108 002370      042     122     105          .ASCII\'READ MEMORY FROM DRIVE UNSUCCESSFUL'N\
    109 002413      042     101     124          .ASCII\'ATTEMPTING DRIVE CLEAR'N\
    110 002430      000                          .BYTE   0
    111 002431      116                  MS54:   .ASCII\N\
    112 002431      042     104     122          .ASCII\'DRIVE CLEAR FAILED'N\
    113 002444      042     101     124          .ASCII\'ATTEMPTING ATTEMPTING'N\
```

DIVIDE BY OCTAL 50 AND FIND ASCII EQUIVALENT

```
114 002460  000                          .BYTE   0
115 002461  116              MS55:       .ASCII\N\
116 002461  042  107  105                .ASCII\'GET STATUS FAILED'N\
117 002473  042  101  124                .ASCII\'ATTEMPTING DRIVE CLEAR'N\
118 002510  000                          .BYTE   0
119 002511  042  106  111  MS56:         .ASCII\'FILE ''A6''.??? REQUESTED BY THE DRIVE COULD NOT BE FOUND.'N\
120 002546  000                          .BYTE   0
121 002547  042  110  117  MS57:         .ASCII\'HOST SPECIFIED UNIT #'D16'' THAT CAN'T BE FOUND.'N\
122 002600  042  124  105                .ASCII\'TEST2 BEGAN AGAIN'N\
123 002612  000                          .BYTE   0
124 002613  116              MS58:       .ASCII\N\
125 002613  042  106  117                .ASCII\'FOLLOWING DATA IS THE STATUS AND EXTENDED STATUS FROM REGION FFFE 'N\
126 002656  122  064                     .ASCII\R4\
127 002657  000                          .BYTE   0
128 002660  042  104  111  MS59:         .ASCII\'DIAGNOSE OF REGION ID  'H8N\
129 002676  000                          .BYTE   0
130 002677  042  124  111  MS60:         .ASCII\'TIMEOUT ERROR OCCURED DURING RECEIVE XFC'N\
131 002724  000                          .BYTE   0
132 002725  042  061  123  MS61:         .ASCII\'1ST WORD NOT START FRAME DURING RECEIVE XFC'N\
133 002754  000                          .BYTE   0
134 002755  116              MS62:       .ASCII\N\
135 002755  042  106  122                .ASCII\'FRAMING ERROR OCCURED ON SDI LEVEL 0 READ DURING RECEIVE XFC'N\
136 003015  000                          .BYTE   0
137 003016  116              MS63:       .ASCII\N\
138 003016  042  103  110                .ASCII\'CHECKSUM ERROR OCCURED ON SDI LEVEL 0 READ DURING RECEIVE XFC'N\
139 003056  000                          .BYTE   0
140 003057  116              MS64:       .ASCII\N\
141 003057  042  102  125                .ASCII\'BUFFER SIZE SMALLER THEN RESPONSE DURING RFCEIVE XFC'N\
142 003113  000                          .BYTE   0
143 003114  116              MS65:       .ASCII\N\
144 003114  042  103  117                .ASCII\'CODE FROM RECEIVE XFC WAS UNINTELLIGIBLE FROM SUBSYSTEM  'D16N\
145 003154  000                          .BYTE   0
146
147
148 003154                               DMEND
    003155  020415                       .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
149         000001                       .END
```

SYMBOL TABLE

```
ARGS$ = 000003   DRCLR1  001625   GETSUB= 000210   IN.24   001074   MS59    002660
ATTN  = 000002   DROP   = 100000   GETU    003361   IN.25   001075   MS6     000233
AVAIL = 000100   DRTYPE= 000007   GRPCYL= 000002   IN.26   001076   MS60    002677
BEUSED= 000040   DRVCLR= 000005   GRPOFF= 000011   IN.27   001077   MS61    002725
BF.DAT= 000000   DRVID = 000004   GSTST8  001761   IN.28   001100   MS62    002755
BF.ECC= 000401   DRVONL= 000213   GSTS3   001631   IN.29   001101   MS63    003016
BF.EDC= 000400   DRVRUN= 000014   GSTS6   001635   IRECLB  000216   MS64    003057
BREAK = 000000   D.LIMT= 000001   GSTS7   001662   LARGE = 000001   MS65    003114
BUFFLG= 040000   D.SCHR= 000002   GTSTAT  003522   LBHINB= 177417   MS7     000316
BUFSIZ= 000036   ECC    = 000015   HBHINB= 007777   LBLONB= 177760   MS8     000350
BUF1    003761   ECCCHK= 010000   HBLONB= 170377   LBNCYL= 000000   MS9     000405
CHECK = 000010   ECCFLG= 010000   HDRPRE= 000005   LBNHST= 000012   MWR   = 000017
CHGMOD= 000201   ECCRSH= 000002   HD.BAD= 110000   LBNTRK= 000011   MWRITE  001170
CHRRES= 000170   ECHO   = 000010   HD.DBN= 140000   LINKLN= 000007   NAM.1   004417
CLRBUF  000777   ECHOC  = 000350   HD.LBN= 000000   LOBYTE= 000377   NAM.2   004420
CLRDRV  003564   ECHOD    003722   HD.PRV= 050000   LONG    004205   NAM.3   004421
COMPAR= 000006   ECHO1    001553   HD.RBN= 060000   LONGTO= 000001   NEWSUB= 004000
COMPLT= 000176   ECHO2    001556   HD.REV= 030000   LOW   = 000002   NUMPTR= 000004
CONMEM  003054   ECHO3    001573   HD.XBN= 120000   LUNIT   001315   OK.END  003562
CR.DIA  004205   ECHO4    001600   HEADER= 000002   MEDTYP= 000006   ONLYCL= 000200
CR.DRC  003736   ECHO5    001614   HIBYTE= 177400   MESSAG= 000015   OUT.RQ  001006
CR.GCR  004177   EOC    = 100000   HICYL = 000001   MICREV= 000003   OUT.01  001007
CR.GST  003730   ERECOV= 000006   HIDBN = 000003   MRD   = 000016   OUT.02  001010
CR.RDM  004161   ERHARD= 001000   HILBN = 000002   MREAD   001152   OUT.03  001011
CR.WRM  003745   ERLEV = 000002   HIMEM = 010000   MS1     000000   OUT.04  001012
CVT   = 000020   ERRMC = 000014   HIRBN = 000003   MS10    000475   OUT.05  001013
DATAVL= 040000   ERRMES= 000013   HIXBN = 000002   MS11    000527   OUT.06  001014
DATCMP= 000002   ERRN   = 000040   HOSTRQ  000751   MS12    000564   OUT.07  001015
DATERR= 000020   ERRTYP= 100000   HRDREV= 000003   MS16    000655   OUT.08  001016
DATPRE= 000005   ERSOFT= 001400   ILOOP   001523   MS17    000744   OUT.09  001017
DBNCYL= 000022   ER.END  003557   INITW = 040000   MS18    000777   OUT.10  001020
DCMPAL= 000001   EXIT   = 000021   INSEEK= 000012   MS19    001035   OUT.11  001021
DCYLS = 020000   EX.END  003563   INTEDC= 000105   MS2     000050   OUT.12  001022
DIA     004212   FB.DAT= 000000   IN.RQ   001044   MS20    001125   OUT.13  001023
DIAGNS  002110   FB.EDC= 000400   IN.01   001045   MS26    001226   OUT.14  001024
DIAG.1  0044 6   FCTSIZ= 000010   IN.02   001046   MS27    001316   OUT.15  001025
DIA.EN= 0003?4   FFFC   = 177774   IN.03   001047   MS28    001350   OUT.16  001026
DIA.OP= 000003   FFFD   = 177775   IN.04   001050   MS3     000074   OUT.17  001027
DINIT = 000011   FFFE   = 177776   IN.05   001051   MS30    001406   OUT.18  001030
DIREC = 010000   FIRSTU= 007723   IN.06   001052   MS36    001440   OUT.19  001031
DISCON= 000204   FNDASC  003644   IN.07   001053   MS37    001526   OUT.20  001032
DIV50   003630   FORMAT= 000001   IN.08   001054   MS38    001557   OUT.21  001033
DONE  = 000016   FRAME  = 000004   IN.09   001055   MS4     000130   OUT.22  001034
DONECD  001513   FREE     004423   IN.10   001056   MS40    001613   OUT.23  001035
DO.DCL  002315   FSTOP  = 100000   IN.11   001057   MS41    001715   OUT.24  001036
DO.DC4  002313   FTIME  = 001000   IN.12   001060   MS42    002013   OUT.25  001037
DO.DIX  002340   FTLDEV= 000400   IN.13   001061   MS43    002100   OUT.26  001040
DO.DI0  002706   FTLSYS= 000000   IN.14   001062   MS5     000170   OUT.27  001041
DO.DI1  002372   FT.BUF= 000000   IN.15   001063   MS50    002165   OUT.28  001042
DO.DI2  002424   FT.HI  = 000001   IN.16   001064   MS51    002226   OUT.29  001043
DO.DI3  002547   FT.LOW= 000002   IN.17   001065   MS52    002271   OVERFL= 000002
DO.DI6  002705   GCC.EN= 000170   IN.18   001066   MS53    002370   OVE.MN= 000714
DO.DI7  002553   GCR      004204   IN.19   001067   MS54    002431   OVE.MS= 000000
DO.DI8  002556   GETBCN   003114   IN.20   001070   MS55    002461   OVL.MN= 003510 G
DO.DI9  002631   GETCHR=  000207   IN.21   001071   MS56    002511   OVL.MS= 003156 G
DRC     003743   GETST    003735   IN.22   001072   MS57    002547   OVL...= 006666
DRCBYT  003744   GETSTA=  000011   IN.23   001073   MS58    002613   OVSTRT= 007774
```

UDAT2 DISK RESIDENT MACRO X04.00 9-JUL-81 11:39:12 PAGE 59-4 SEQ 0157
SYMBOL TABLE

```
OVS.MN= 001040 G        RW.ANG= 000006        ST.FO = 002000        TLEN.U= 000055        U.MLEV= 000026
OVS.MS= 010260 G        RW.BUF= 000001        ST.MOD= 000001        TO      001221        U.MSEC= 000017
OV... = 007306          RW.CMD= 000004        ST.MSK= 000000        TOOBIG= 000001        U.NEXT= 000000
PORT0   001445          RW.HI = 000003        ST.OA = 000200        TRACKS= 000020        U.NFUN= 000010
PORT3   001450          RW.LOW= 000002        ST.PE = 000040        TRKGRP= 000003        U.NSEC= 000016
PORT5   001505          RW.SDI= 000005        ST.PS = 000002        TYPERR  003237        U.PARM= 000033
PRMS  = 000002          RW.STA= 000000        ST.RE = 000100        TOO     001724        U.PAT = 000011
PTYPE$= 000020          SAVREG  001316        ST.RR = 000100        T1MSIZ= 000000        U.PCTG- 000014
RBNBN = 000200          SAVRID  003721        ST.RTY= 000003        T2CMD = 000002        U.RBN = 000041
RBNTRK= 000004          SAVSTA  003720        ST.RU = 000001        T2DLL = 000001        U.RTRY= 000025
RBUFLN= 000415          SBCRES= 000167        ST.SR = 000020        T4BB1 = 000005        U.RWTO= 000006
RCONT = 000000          SCTWRD= 000377        ST.STA= 000001        T4BB2 = 000006        U.SDIL= 000031
RCTCPS= 000001          SDI     003717        ST.S7 = 000400        T4MPRM= 000003        U.SDIS= 000030
RCTCSZ= 000014          SDILTO  001151        ST.UNT= 000000        T4MXFR= 000011        U.SEEK= 000007
RCV   = 000005          SDISTO  001150        ST.WE = 000010        T4SEEK= 000010        U.SNUM= 000043
RCVERR= 000004          SDIVER= 000000        SUB   = 000013        T4SOFT= 000007        U.SUBP= 000001
RCVRDY= 000001          SEKINP= 002000        S.BADP= 000010        T4UPRM= 000004        U.SUBU= 000034
RDM     004166          SEND  = 000004        S.BESS= 000011        UDATA   001210        U.TIMO= 000005
RDMEM   002737          SEQSEK= 000100        S.ERR   004415        UNITNB  003716        U.TSEC= 000020
RDM.EN= 000162          SHRTTO= 000000        S.MCNT= 000011        UNITS   003676        U.WPRT= 000032
RDM.OP= 000215          SNDAGN  000756        S.MEGR= 000006        UNIT0 = 000001        U.WRIT= 000023
RDSTAT  000720          SPADJU  001312        S.MEGW= 000007        UNIT1 = 000002        WAITSI= 000012
READ    002054          SS    = 000001        S.PARM= 000000        UNIT2 = 000004        WBUFLN= 000401
REDWRT= 000100          ST      004215        S.PAT = 000003        UNIT3 = 000010        WCHECK= 000010
REGS$ = 177777          STACK   001357        S.SCHR= 000005        UNSSUC= 000175        WCHKAL= 000004
REGU$ = 000001          START   001360        S.SDCL= 000001        UREAD = 000013        WCONT = 040000
RERRCA  001301          STATEX  000746        S.TGOF= 000013        UTOTST= 000012        WONLY = 002000
RERROR  001236          STATLP  000724        S.TGSS= 000015        UWRITE= 000014        WREAL = 122400
RFRRPA  001275          STATOK  000737        S.TRKL= 000004        U.CBN = 000037        WRITE   002010
RESEEK= 020000          STATUS= 000007        T       001664        U.CCNT= 000012        WRM     003752
RETRY = 000004          STSIZE= 000200        TALKER  001102        U.CCYL= 000047        WRM.OP= 000017
RETS  = 000001          STSRES= 000366        TALK1A  001114        U.CGRP= 000051        WRONG = 000002
REVEC = 000400          ST.C  = 000002        TALK1B  001124        U.CSEC= 000021        WRTMEM  003013
REVECT= 000004          ST.CON= 000002        TALK2A  001144        U.CTRK= 000015        WSTOP = 140000
REVS  = 000007          ST.DB - 001000        TALK2B  001145        U.ECCT= 000027        XBNCYL= 000021
RM    = 000004          ST.DF = 000020        TEST    001517        U.ELEV= 000024        XFERRT= 000000
RONLY = 004000          ST.DIA  001742        TESTEX  002736        U.LCYL= 000052        XMTERR= 000400
RREAL = 013400          ST.DR = 000040        TESTX   001501        U.LGRP- 000054        XOR     001211
RSTOP = 100000          ST.ERR= 000002        TEST4 = 000000        U.MASK= 000022        XREAD = 000002
RTRIES= 001000          ST.FE = 000200        TIMEOU= 000001        U.MBN = 000035        XWRITE= 000003
RWRDY - 100000

. ABS.  016614      000
        000000      001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  4377 WORDS  ( 18 PAGES)
DYNAMIC MEMORY AVAILABLE FOR  69 PAGES
UDAT2.BIN,A:UDAT2B/C=[1,33]DMACRO,UDAT2T,UDATP,UDATR,UDAT2M,UDAT2S
```

CROSS REFERENCE TABLE (CREF V01-05)

CROSS REFERENCE TABLE (CREF V01-05)

```
DO.DI8  43-24    44-6#
DO.DI9  41-19    41-32    41-56    41-59    45-2#
DO.DIX  37-4     42-22#
DONE     6-17#   32-105
DONECD  32-105#  32-107   33-18    33-18
DRC     57-30    57-31#
DRCBYT  55-11*   57-32#
DRCLR1  35-8     35-13#   35-34
DROP     8-10#
DRTYPE   5-15#
DRVCLR   6-35#   57-31
DRVID    5-14#
DRVONL   6-33#
DRVRUN   6-34#
ECC      4-16#
ECCCHK   8-26#
ECCFLG   3-45#
ECCRSH   5-11#
ECHO     4-11#   34-14
ECHO1   33-20    34-11#
ECHO2   34-13#   34-37
ECHO3   34-21    34-24#
ECHO4   34-19    34-29#
ECHO5   34-23    34-35    34-30    34-35#
ECHOC    6-46#   57-83    57-19    57-20    57-21    57-22
ECHOD   34-11    57-18#
EOC      3-46#
ER.END  54-15    54-17    54-22#   55-18    55-20    55-25
ERECOV   6-31#
ERHARD   6-52#
ERLEV    5-10#
ERRMC    6-15#
ERRMES   6-14#   29-37
ERRN    32-3#    32-46    32-46    32-46    32-46#   33-18    33-18    33-18    33-18    33-18    33-18    33-18#   33-18#   33-23
        33-23    33-23    33-23#   34-22    34-22    34-22    34-22#   34-24    34-24    34-24    34-24#   34-31    34-31    34-31
        34-31#   35-35    35-35    35-35    35-35#   36-6     36-6     36-6     36-6     36-6     36-6     36-6#    36-6#    36-9
        36-9     36-9     36-9#    37-26    37-26    37-26    37-26#   40-20    40-20    40-20    40-20#   40-23    40-23    40-23
        40-23#   40-30    40-30    40-30    40-30#   41-18    41-18    41-18    41-18#   41-55    41-55    41-55    41-55#   41-58
        41-58    41-58    41-58#   44-25    44-25    44-25    44-25#   47-31    47-31    47-31    47-31#   47-34    47-34    47-34
        47-34#   47-40    47-40    47-40    47-40#   48-23    48-23    48-23    48-23#   48-26    48-26    48-26    48-26#   48-32
        48-32    48-32    48-32#   54-14    54-14    54-14    54-14#   54-16    54-16    54-16    54-16#   54-21    54-21    54-21
        54-21#   55-17    55-17    55-17    55-17#   55-19    55-19    55-19    55-19#   55-24    55-24    55-24    55-24#
ERRTYP  41-16    41-36    57-83#
ERSOFT   6-53#
EX.END  54-23    54-25#
EXIT     4-20#
FB.DAT   3-22#
FB.EDC   3-23#
FCTSIZ   5-36#
FFFC    57-88#
FFFD    42-22    57-88    57-89#   57-90
FFFE    45-4     57-90#
FIRSTU   7-11#
FNDASC  44-13    44-23    56-29#
FORMAT   4-4#
FRAME    5-66#
FREE    59-3#
```

```
FSTOP    3-42#
FT.BUF   3-15#
FT.HI    3-16#
FT.LOW   3-17#
FTIME    8-16#
FTLDEV   6-51#
FTLSYS   6-50#
GCC.EN  57-79#
GCR     57-45    57-46#
GETBCN  41-24    41-42    50-29#
GETCHR   6-36#   57-46    57-76#
GETST   57-27    57-28#
GETSTA   6-38#   57-28
GETSUB   6-37#
GETU    32-6     52-2#
GRPCYL   5-25#
GRPOFF   5-38#
GSTS3   35-15    35-21#
GSTS6   35-24    35-29#
GSTS7   35-32    35-39#
GSTST8  37-13#   37-27    38-28    39-24    46-16
GTSTAT  35-6     35-22    42-12    54-8#
HBHINB   5-56#   52-48    53-10
HBLONB   5-57#
HD.BAD   3-55#
HD.DBN   3-58#
HD.LBN   3-52#
HD.PRV   3-56#
HD.RBN   3-53#
HD.REV   3-54#
HD.XBN   3-57#
HDRPRE   5-34#
HEADER   5-62#
HIBYTE   5-54#   32-36    34-13    39-20    49-30    49-36    50-33    50-38    50-75    55-10
HICYL    5-24#
HIDBN    5-30#
HILBN    5-26#
HIMEM    3-9#     7-11
HIPBN    5-29#
HIXBN    5-27#
HOSTRQ  22-2#    29-38    32-106   37-15    41-31    43-17    45-25    50-53    51-22    51-26    51-30    51-34    51-38    51-40
        53-5
HRDREV   5-13#
ILOOP   33-18#   33-22
IN.01   23-39#   37-16    43-5     43-19    53-6
IN.02   23-40#   37-10*   37-26    37-26    38-11    39-10    40-9     43-74*   43-78*
IN.03   23-41#   38-12    39-11
IN.04   23-42#   38-13
IN.05   23-43#
IN.06   23-44#
IN.07   23-45#
IN.08   23-46#
IN.09   23-47#
IN.10   23-48#
IN.11   23-49#
IN.12   23-50#
IN.13   23-51#
```

CROSS REFERENCE TABLE (CREF V01-05)

CROSS REFERENCE TABLE (CREF V01-05)

```
MS40     41-18     59-75#
MS41     41-31     59-81#
MS42     41-55     59-87#
MS43     41-58     59-93#
MS5      34-24     59-17#
MS50     36-6      59-99#
MS51     36-6      59-101#
MS52     36-9      59-103#
MS53     59-107#
MS54     59-111#
MS55     59-115#
MS56     44-25     59-119#
MS57     32-46     59-121#
MS58     45-25     59-124#
MS59     59-128#
MS6      34-31     59-20#
MS60     51-22     59-130#
MS61     51-26     59-132#
MS62     51-30     59-134#
MS63     51-34     59-137#
MS64     51-38     59-140#
MS65     51-40     59-143#
MS7      54-14     59-24#
MS8      54-16     59-26#
MS9      54-21     59-28#
MWR       4-18#    22-21
MWRITE   26-10#
NAM.1    44-11*    44-15*    44-25     44-25     57-66#
NAM.2    44-9*     44-24*    44-25     44-25     57-67#
NAM.3    44-18*    44-21*    44-25     44-25     57-68#
NEWSUB    8-14#
NUMPTR   41-31#    45-25#    50-53#    51-22*    51-26#    51-30#    51-34#    51-38#    51-40     51-40     51-40#    51-40#
OK.END   54-20     54-24#    55-23
ONLYCL    8-30#
OUT.01   22-22     22-23     23-6*     29-13     37-6*     38-24*    39-16*    46-4*     50-46     50-55
OUT.02   23-7#     37-7*     38-23*    39-17*    41-31*    45-25*    46-7*     46-9*     50-53*    51-22*    51-26*    51-30*    51-34*    51-38*
         51-40*
OUT.03   23-8#     38-25*    39-21*    46-11*    51-40*
OUT.04   23-9#     38-26*    39-22*    45-11     45-13     46-12*
OUT.05   23-10#    38-27*    39-23*    41-10
OUT.06   23-11#    41-15     41-34
OUT.07   23-12#    41-13
OUT.08   23-13#    41-51*    50-32     50-36     50-72*    50-75*    50-77*
OUT.09   23-14#    41-29     41-47
OUT.10   23-15#
OUT.11   22-23     23-16#
OUT.12   23-17#    45-23
OUT.13   23-18#
OUT.14   23-19#
OUT.15   23-20#
OUT.16   23-21#
OUT.17   23-22#
OUT.18   23-23#
OUT.19   23-24#
OUT.20   23-25#
OUT.21   23-26#
OUT.22   23-27#
```

CROSS REFERENCE TABLE (CREF V01-05)

CROSS REFERENCE TABLE (CREF V01-05)

```
S.SCHR   7-55#    7-56
S.SDCL   7-52#    7-53
S.TGOF   7-65#    7-66
S.TGSS   7-66#
S.TRKL   7-54#    7-55
SAVREG  29-46#   32-34*   32-36*   32-37    34-31    34-31*   35-35    35-35*   36-9     36-9*    37-26    37-26*   40-30    40-30*
        43-43*   43-54    44-25    44-25*   47-40    47-40*   48-32    48-32*   54-21    54-21*   55-24    55-24*
SAVRID  42-34*   43-4     43-78    57-16#
SAVSTA  35-40*   37-2     57-15#
SBCRES   6-44#
SCTWRD   7-8#
SDI     32-54*   32-97*   32-100   33-11    40-13    41-9     41-28    41-46    42-5     42-30    43-50    45-10    45-22    46-3
        47-24    48-16    57-13#
SDILTO  24-27    24-46#   36-19*
SDISTO  24-25    24-45#   36-4*    36-15*
SDIVER   5-4#
SEKINP   8-15#
SEND     4-7#    24-18    52-29
SEQSEK   8-32#
SHRTTO   5-3#    36-12
SNDAGN  22-14#   22-18
SPADJU  29-30*   29-40#
SS       5-9#
ST      35-29    35-30    35-35    35-35    36-12    36-16    39-18    41-3     42-18    42-31    42-34    42-35*   42-37*   43-10
        43-12    43-14    43-29    43-39    43-45*   43-46*   46-11    49-28    50-47    50-56    52-46    55-9     57-27    57-30
        57-34    57-40    57-45    57-50    57-57#
ST.C     4-30#
ST.CON   4-29#
ST.DB    4-47#
ST.DF    4-44#
ST.DIA  37-2#
ST.DR    4-37#    37-3     42-19
ST.ERR   4-28#
ST.FE    4-41#    35-31
ST.FO    4-46#
ST.MOD   4-27#
ST.MSK   4-25#
ST.OA    4-35#
ST.PE    4-43#    35-31
ST.PS    4-39#
ST.RE    4-42#
ST.RR    4-36#
ST.RTY   4-31#
ST.RU    4-40#
ST.S7    4-48#
ST.SR    4-38#
ST.STA   4-26#
ST.UNT   4-24#
ST.WE    4-45#    35-31
STACK   20-8      30-5#
START   20-9      32-6#
STATEX  21-16     21-18    21-22#
STATLP  21-9#     21-14    21-20
STATOK  21-12     21-17#
STATUS   4-10#    21-9
STSIZE  43-30     43-32    57-40    57-56#
STSRES   6-45#    54-19    54-21    54-21
```

CROSS REFERENCE TABLE (CREF V01-05)

```
SUB       5-22#
T        36-4#
T00      36-8      36-12#
T1MSIZ    6-3#
T2CMD     6-5#     37-14
T2DLL     6-4#     43-16
T4BB1     6-8#
T4BB2     6-9#
T4MPRM    6-6#
T4MXFR    6-12#
T4SEEK    6-11#
T4SOFT    6-10#
T4UPRM    6-7#
TALK1A   24-20     24-23#
TALK1B   24-26     24-28#    24-38
TALK2A   24-32     24-41#
TALK2B   24-22     24-36     24-40     24-42#
TALKER   24-15#    36-6      40-14     47-25     48-17     54-9      55-12
TEST     32-98     33-11#
TEST4     2-37#    24-1
TESTEX   33-24     35-9      35-16     35-25     36-6      36-6      36-10     37-19     42-11     42-17     46-21#
TESTX    32-99#    46-21
TIMEOU    5-61#
TLEN.U    7-10#     7-11
TO       28-1#     36-14     36-18
TOOBIG    5-69#
TRACKS    8-34#
TRKGRP    5-28#
TYPERR   40-24     47-35     48-27     51-16#
U.CBN     7-41#     7-42
U.CCNT    7-22#     7-23
U.CCYL    7-44#     7-45
U.CGRP    7-45#     7-46
U.CSEC    7-28#     7-29
U.CTRK    7-24#     7-25
U.ECCT    7-34#     7-35
U.ELEV    7-31#     7-32
U.LCYL    7-46#     7-47
U.LGRP    7-10      7-47#
U.MASK    7-29#     7-30
U.MBN     7-40#     7-41
U.MLEV    7-33#     7-34
U.MSEC    7-26#     7-27
U.NEXT    7-15#     7-16
U.NFUN    7-20#     7-21
U.NSEC    7-25#     7-26
U.PARM    7-38#     7-39
U.PAT     7-21#     7-22
U.PCTG    7-23#     7-24
U.RBN     7-42#     7-43
U.RTRY    7-32#     7-33
U.RWTO    7-18#     7-19
U.SDIL    7-36#     7-37
U.SDIS    7-35#     7-36
U.SEEK    7-19#     7-20
U.SNUM    7-43#     7-44
U.SUBP    7-16#     7-17
```

UDAT2 DISK RESIDENT MACRO X04.00 9-JUL-81 11:39:12 PAGE M-1 SEQ 0168
CROSS REFERENCE TABLE (CREF V01-05)

UDAT3 DISK FUNCTIONAL MACRO X04.00 9-JUL-81 01:13:55
TABLE OF CONTENTS

```
                                .TITLE  UDAT3   DISK FUNCTIONAL
                        ;
                        ;
                        ; COPYRIGHT (C)  1980
                        ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
                        ;
                        ;
                        ; THIS  SOFTWARE  IS  FURNISHED  UNDER A LICENSE FOR USE ONLY ON A
                        ; SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION
                        ; OF  THE  ABOVE COPYRIGHT  NOTICE.  THIS SOFTWARE, OR  ANY  OTHER
                        ; COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE  MADE AVAILABLE
                        ; TO  ANY  OTHER PERSON  EXCEPT  FOR USE ON SUCH SYSTEM AND TO ONE
                        ; WHO AGREES TO THESE LICENSE TERMS.  TITLE  TO  AND  OWNERSHIP OF
                        ; THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DIGITAL.
                        ;
                        ; THE  INFORMATION  IN THIS DOCUMENT IS SUBJECT  TO CHANGE WITHOUT
                        ; NOTICE AND SHOULD NOT BE CONSTRUED AS A  COMMITMENT  BY  DIGITAL
                        ; EQUIPMENT CORPORATION.
                        ;
                        ; DIGITAL ASSUMES NO  RESPONSIBILITY FOR THE USE OR RELIABILITY OF
                        ; ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;THIS PROGRAM SHALL BE ASSEMBLED WITH THE PROGRAM DMACRO
                        ;USING A COMMAND LINE SIMILAR TO:
                        ;
                        ;  UDAT3.BIN,UDAT3/C=[1,2]DMACRO,UDAT3T,UDATP,UDAT3M,UDATR,UDAT3S
                        ;
000000                  TEST4   =       0                       ; THIS IS NOT TEST4
000000                          .ENABL  ABS
000000                          DMCODE  UDAT3,0,714,3,0
```

```
                        .SBTTL  UDA DM PROGRAM PARAMETERS
                        .LIST MEB
                        ;
                        ;       EQUATES
                        ;
                        ;       HIGHEST USABLE LOCATION OF JDA MEMORY + 1
                        ;
010000          HIMEM   =       10000                   ; HIGH MEMORY+1
007774          OVSTRT  =       7774                    ; OVERLAY ADDRESS LOCATION
                        ;
                        ;       OFFSETS FOR FORMAT TRACK TABLE
                        ;
000000          FT.BUF  =       0.                      ;BUFFER POINTER OFFSET
000001          FT.HI   =       1.                      ;HI ORDER HEADER OFFSET
000002          FT.LOW  =       2.                      ;LOW ORDER HEADER OFFSET
                        ;
                        ;       OFFSETS FOR FORMAT TRACK BUFFER
                        ;
000000          FB.DAT  =         0.                    ;FIRST DATA WORD OFFSET
000400          FB.EDC  =       256.                    ;EDC WORD OFFSET
                        ;
                        ;       OFFSETS FOR READ/WRITE I/O CHAIN TABLES
                        ;
000000          RW.STAT =       0.                      ;STATUS AND NEXT BUFFER POINTER OFFSET
000001          RW.BUF  =       1.                      ;POINTER TO DATA BUFFER
000002          RW.LOW  =       2.                      ;HI ORDER EXPECTED HEADER
000003          RW.HI   =       3.                      ;LOW ORDER EXPECTED HEADER
000004          RW.CMD  =       4.                      ;SDI COMMAND AND HEAD ADDRESS
000005          RW.SDI  =       5.                      ;DUMMY SDI CONTROL BLOCK POINTER
000006          RW.ANG  =       6.                      ;THETA FROM INDEX
                        ;
                        ;       CONSTANTS FOR READ AND WRITE XFC'S
                        ;
140000          WSTOP   =       140000                  ; LAST ENTRY IN CHAIN FOR WRITE
040000          WCONT   =       40000                   ; WRITE CONTINUE
100000          RSTOP   =       100000                  ; LAST ENTRY IN CHAIN FOR READ
000000          RCONT   =       0                       ; READ CONTINUE
100000          FSTOP   =       100000                  ; LAST ENTRY IN CHAIN FOR FORMAT
122400          WREAL           122400                  ; WRITE REAL TIME ECOMMAND
013400          RREAL   =       13400                   ; READ REAL TIME COMMAND
010000          ECCFLG  =       10000                   ; ECC ERROR IN BUFFER BIT
100000          EOC             100000                  ; END OF CHAIN
040000          BUFFLG  =       40000                   ; BUFFER FULL OR EMPTY FLAG
                        ;
                        ;       HEADER CODES
                        ;
000000          HD.LBN  =       000000                  ;GOOD LBN
060000          HD.RBN  =       060000                  ;GOOD RBN, PERHAPS UNUSED
030000          HD.REV  =       030000                  ;REVECTORED LBN
110000          HD.BAD  =       110000                  ;BAD BLOCK
050000          HD.PRV          050000                  ;PRIMARY REVECTORED BLOCK
120000          HD.XBN          120000                  ;XBN BLOCK
```

```
                140000C                 HD.DBN  =       140000                  ;DBN BLOCK
                                        ;       OFFSETS FOR DATA BUFFERS
                000000                  BF.DAT  =         0.                    ;DATA
                000400                  BF.EDC  =       256.                    ;ERROR DETECTION CODE
                000401                  BF.ECC  =       257.                    ;LAST 17 ECC RESIDUES
                                        ;       BUFFER AND READ/WRITE CHAIN LINK SIZES
                000401                  WBUFLN  =       257.                    ; WRITE BUFFER SIZE
                000415                  RBUFLN  =       WBUFLN+12.              ; READ BUFFER SIZE
                000007                  LINKLN  =       7.                      ; LINK SIZE
```

```
                                ;       XFC DEFINITION EQUA ES

000000          BREAK   =       0.      ;BREAKPOINT XFC CODE
000001          FORMAT  =       1.      ;FORMAT TRACK XFC CODE
000002          XREAD   =       2.      ;READ N SECTORS XFC CODE
000003          XWRITE  =       3.      ;WRITE N SECTORS XFC CODE
000004          SEND    =       4.      ;SEND SDI COMMAND XFC CODE
000005          RCV     =       5.      ;RECEIVE SDI MESSAGE XFC CODE
000006          COMPARE =       6.      ;COMPARE DATA PATTERN TO BUFFER
000007          STATUS  =       7.      ;RETURN DRIVE STATUS XFC CODE
000010          ECHO    =       8.      ;ECHO DATA TO DRIVE XFC CODE
000011          DINIT   =       9.      ;DRIVE INITIALIZE XFC CODE
000012          WAITSI  =       10.     ;WAIT FOR SECTOR OR INDEX PULSE
000013          UREAD   =       11.     ;READ UNIBUS MEMORY XFC CODE
000014          UWRITE  =       12.     ;WRITE UNIBUS MEMORY XFC CODE
000015          ECC     =       13.     ;DO ECC ON BUFFER XFC CODE
000016          MRD     =       14.     ;SEND BUFFER TO MAINTENANCE READ COMMAND
000017          MWR     =       15.     ;GET BUFFER FROM MAINTENANCE WRITE COMMAND
000020          CVT     =       16.     ;CONVERT TO PHYSICAL ADDRESS XFC CODE
000021          EXIT    =       17.     ;TERMINATE DM PROGRAM XFC CODE
                                ;
                                ;       GET STATUS OFFSETS
                                ;
000000          ST.UNT  =       0.      ;UNIT NUMBER
000000          ST.MSK  =       0.      ;SUBUNIT MASK
000001          ST.STA  =       1.      ;STATUS BYTE
000001          ST.MOD  =       1.      ;MODE BYTE
000002          ST.ERR  =       2.      ;ERROR BYTE
000002          ST.CON  =       2.      ;CONTROLLER BYTE
000002          ST.C    =       2.      ;C BITS
000003          ST.RTY  =       3.      ;RETRY COUNT/FAILURE CODE
                                ;
                                ;       STATUS BIT DEFINITIONS
                                ;
000200          ST.OA   =       200     ; ONLINE TO ANOTHER (SET IF DRIVE UNAVAILABLE)
000100          ST.RR   =       100     ; READJUSTMENT BIT (SET IF RECALIBRATION REQUIRED)
000040          ST.DR   =       40      ; DIAGNOSTIC REQUEST (SET IF DIAGNOSTIC REQUESTED)
000020          ST.SR   =       20      ; SPINDLE READY (SET IF SPINDLE READY)
000002          ST.PS   =       2       ; PORT SWITCH (SET IF PORT SWITCH IN)
000001          ST.RU   =       1       ; RUN/STOP SWITCH (SET IF RUN/STOP SWITCH IN)
000200          ST.FE   =       200     ; FATAL ERROR (SET IF FATAL ERROR OCCURRED)
000100          ST.RE   =       100     ; RETRIABLE ERROR (SET IF RETRIABLE ERROR OCCURRED)
000040          ST.PE   =       40      ; PROTOCOL ERROR (SET IF PROTOCOL ERROR OCCURRED)
000020          ST.DF   =       20      ; INITIALIZATION FAILURE (SET IF INIT FAILED)
000010          ST.WE   =       10      ; WRITE ENABLE (SET IF WRT ATTEMPTED ON PROT DISK)
002000          ST.FO   =       2000    ; FORMATTING (SET IF FORMATTING ENABLED)
001000          ST.DB   =       1000    ; DIAGNOSTIC CYLS (SET IF DIAG CYL ACCESS ENABLED)
000400          ST.S7   =       400     ; SECTOR SIZE (SET FOR 576 BYTE SECTORS)
```

```
                        ;       GET COMMON CHARACTERISTICS OFFSETS
                        ;
000000  SHRTTO  =      0.       ;SHORT TIMEOUT <3:0>
000000  SDIVER  =      0.       ;SDI VERSION <7:4>
000000  XFERRT  =      0.       ;TRANSFER RATE <15:0>
000001  LONGTO  =      1.       ;LONG TIMEOUT <3:0>
000001  RETS    =      1.       ;RETRIES <7:4>
000001  RCTCPS  =      1.       ;F/RCT COPIES <11:8>
000001  SS      =      1.       ;SECTOR SIZE <15:15>
000002  ERLEV   =      2.       ;ERROR RETRY LEVELS <7:0>
000002  ECCRSH  =      2.       ;ECC THRESHOLD <15:8>
000003  MICREV  =      3.       ;MICROCODE REVISION NUMBER <7:0>
000003  HRDREV  =      3.       ;HARDWARE REVISION NUMBER <15:8>
000004  DRVID   =      4.       ;UNIQUE DRIVE ID <47:0>
000007  DRTYPE  =      7.       ;DRIVE TYPE IDENTIFIER <7:0>
0000C7  REVS    =      7.       ;REVS/SECOND <15:8>
                        ;
                        ;       GET SUBUNIT CHARACTERISTICS OFFSETS
                        ;
                        ;THESE OFFSETS ARE CURRENTLY GIVEN AS FOLLOWING THE COMMON CHARACTERISTICS
                        ;
000013  SUB     =      11.      ;OFFSET TO PUT SUBUNIT AFTER COMMON;
000000  LBNCYL  =      0.       ;NUMBER OF CYLINDERS IN LBN AREA <31:0>
000001  HICYL   =      1.       ;HI ORDER CYLINDER BITS <15:12>
000002  GRPCYL  =      2.       ;GROUPS PER CYLINDER <7:0>
000002  HILBN   =      2.       ;HI STARTING LBN <11:8>
000002  HIXBN   =      2.       ;HI STARTING XBN <15:12>
000003  TRKGRP  =      3.       ;TRACKS PER GROUP <7:0>
000003  HIRBN   =      3.       ;HI STARTING RBN <11:8>
000003  HIDBN   =      3.       ;HI STARTING DBN <15:12>
000004  RBNTRK  =      4.       ;RBNS PER TRACK <6:0>
000004  RM      =      4.       ;REMOVABLE MEDIA <7:7> 1-REMOVEABLE
000005  DATPRE  =      5.       ;DATA PREAMBLE SIZE IN WORDS <7:0>
000005  HDRPRE  =      5.       ;HEADER PREAMBLE SIZE IN WORDS <15:8>
000006  MEDTYP  =      6.       ;MEDIA TYPE <31:0>
000010  FCTSIZ  =      8.       ;FCT COPY SIZE <15:0>
000011  LBNTRK  =      9.       ;LBNS PER TRACK <7:0>
000011  GRPOFF  =      9.       ;GROUP OFFSET (SECTORS) <15:8>
000012  LBNHST  =     10.       ;LBNS IN HOST AREA <31:0>
000014  RCTCSZ  =     12.       ;RCT COPY SIZE <15:0>
000021  XBNCYL  =     17.       ;CYLS IN XBN AREA <15:0>
000022  DBNCYL  =     18.       ;CYLS IN DBN AREA <15:8>
                        ;
                        ;       UNIT CODES
                        ;
000001  UNIT0           1.       ;UNIT ZERO CODE
000002  UNIT1   =      2.       ;UNIT ONE CODE
000004  UNIT2   =      4.       ;UNIT TWO CODE
000010  UNIT3   =      8.       ;UNIT THREE CODE
                        ;
                        ;       BIT MASK DEFINITIONS
                        ;
                        ;
177400  HIBYTE  =    177400     ;HIGH BYTE MASK
000377  LOBYTE  =    000377     ;LOW BYTE MASK
007777  HBHINB          7777     ;HI BYTE, HI NIBBLE MASK
170377  HBLONB         170377     ;HI BYTE, LO NIBBLE MASK
```

```
177417          LBHINB  =       177417          ;LO BYTE, HI NIBBLE MASK
177760          LBLONB  =       177760          ;LO BYTE, LO NIBBLE MASK
                ;
000001          TIMEOUT =       1.              ;DRIVE TIMEOUT CODE
000002          HEADER  =       2.              ;HEADER COMPARE FAILURE CODE
000004          REVECT  =       4.              ;REVECTOR NEEDED CODE
                ;
000002          WRONG   -       2.              ;FIRST WORD NOT START FRAME CODE
000004          FRAME   =       4.              ;FRAMING ERROR CODE
000010          CHECK   =       8.              ;CHECKSUM ERROR CODE
                ;
000001          TOOBIG  =       1.              ;NUMBER OF WORDS EXCEEDS 7064
000002          LOW     =       2.              ;DM BUFFER ADDRESS IS LESS THAN 714
                ;
                ;       BIT MASK
                ;
000001          LARGE   =       1.              ;BLOCK NUMBER TOO LARGE
000002          OVERFL  =       2.              ;SECTOR NUMBER LARGER THAN 16 BITS
```

```
                        ;MAINTANENCE READ/WRITE REQUEST NUMBERS
                        ;
000000                  T1MSIZ  =       0.              ;GET FREE MEMORY PARAMETERS
000001                  T2DLL   =       1.              ;DOWNLINE LOAD DRIVE DIAGNOSTIC
000002                  T2CMD   =       2.              ;MANUAL INTERVENTION TEST 2 PROTOCOL
000003                  T4MPRM  =       3.              ;GET MASTER PARAMETERS FROM SW QUESTIONS
000004                  T4UPRM  =       4.              ;GET UNIT PARAMETERS FROM HW QUESTIONS
000005                  T4BB1   =       5.              ;GET BAD BLOCKS (1 THRU 14)
000006                  T4BB2   =       6.              ;GET REST OF BAD BLOCKS (15 AND 16)
000007                  T4SOFT  =       7.              ;ADD TO SOFT ERROR AND ECC COUNT
000010                  T4SEEK  =       8.              ;ADD 1 TO SEEK COUNT
000011                  T4MXFR  =       9.              ;ADD TO MEGABITS READ AND WRITTEN
000012                  UTOTST  =       10.             ;GET UNITS TO TEST
000013                  ERRMES  =       11.             ;PRINT ERROR MESSAGE
000014                  ERRMC   =       12.             ;TEST 4 ERROR REPORTING
000015                  MESSAG  =       13.             ;INFORMATION MESSAGE
000016                  DONE    =       14.             ;MARK DM PROGRAM AS NO LONGER RUNNING
                        ;
                        ;       OTHER BIT DEFINITIIONS
                        ;
000001                  RCVRDY  =       1               ; RECIEVER READY 1 - READY
000002                  ATTN    =       2               ; ATTENTION BIT FOR RETURN DRIVE SIGNALS XFC
000004                  RCVERR  =       4               ; RECIEVER ERROR
000100                  AVAIL   =       100             ; AVAILABLE 1 = AVAILABLE
000400                  XMTERR  =       400             ; TRANSMIT ERROR
100000                  RWRDY   =       100000          ; IF SET, UDA IS ABLE TO READ AND/OR WRITE TO DRIVE
                        ;
                        ;       SDI COMMANDS AND RESPONSES
                        ;
000204                  DISCON  =       204             ; DISCONNECT DRIVE
000006                  ERECOV  =       6               ; ERROR RECOVERY
000201                  CHGMOD  =       201             ; CHANGE MODE
000213                  DRVONL  =       213             ; DRIVE ONLINE
000014                  DRVRUN  =       14              ; DRIVE RUN
000005                  DRVCLR  =       5               ; DRIVE CLEAR OPCODE
000207                  GETCHR  =       207             ; GET CHARACTERISTICS
000210                  GETSUB  =       210             ; GET SUBUNIT CHARACTERISTICS
000011                  GETSTA  =       11              ; GET STATUS
000216                  IRECLB  =       216             ; RECALIBRATE
000012                  INSEEK  =       12              ; INITIATE SEEK
000176                  COMPLT  =       176             ; SUCCESSFUL COMPLETION
000175                  UNSSUC  =       175             ; UNSUCCESSFUL COMPLETION
000170                  CHRRES          170             ; GET CHARACTERISTICS RESPONSE
000167                  SBCRES  =       167             ; GET SUBUNIT CHARACTERISTICS RESPONSE
000366                  STSRES  =       366             ; GET STATUS RESPONSE
000350                  ECHOC           350             ; DIAGNOSTIC ECHO COMMAND AND RESPONSE
                        ;
                        ;       ERROR CODES
                        ;
000000                  FTLSYS  =       0               ; SYSTEM FATAL ERROR
000400                  FTLDEV  =       400             ; DEVICE FATAL
001000                  ERHARD  =       1000            ; HARD ERROR
001400                  ERSOFT  =       1400            ; SOFT ERROR
```

```
                        .SBTTL  TEST 4 SPECIFIC INFORMATION
                        ;
                        ;       TEST 4 SPECIFIC INFORMATION
                        ;
                        ;
                        ;       CONSTANTS
                        ;
000377                  SCTWRD  =       255.            ; NUMBER OF WORDS IN SECTOR TO FILL
000105                  INTEDC  =       69.             ; INITIAL EDC VALUE
000061                  TLEN.U  =       U.LGRP+1        ; UNIT PARAMETER LENGTH
007717                  FIRSTU  =       HIMEM-TLEN.U    ; LOCATION OF FIRST UNIT PARAMETER BLOCK
                        ;
                        ;       UNIT PARAMETER OFFSETS
                        ;
000000                  U.NEXT  =       0.              ; POINTER TO NEXT UNIT (RING LINKED LIST)
000001                  U.SUBP  =       U.NEXT+1        ; 4 WORDS OF SUBUNIT PARAMETER POINTERS
000005                  U.TIMO  =       U.SUBP+4        ; AREA TO STORE VARIOUS TIMEOUT VALUES
000006                  U.RWTO  =       U.TIMO+1        ; READ/WRITE TIMEOUT AREA
000007                  U.SEEK  =       U.RWTO+1        ; NUMBER OF SEEKS ISSUED
000010                  U.NFUN  =       U.SEEK+1        ; NEXT FUNCTION ADDRESS (FOR DEFERRED CALLS)
000011                  U.PAT   =       U.NFUN+1        ; PATTERN NUMBER TO WRITE
000012                  U.CCNT  =       U.PAT+1         ; CURRENT COUNT OF T/G LOOPS
000014                  U.PCTG  =       U.CCNT+2        ; POINTER TO CURRENT TRACK OR GROUP
000015                  U.CTRK  =       U.PCTG+1        ; TRACK COUNT FOR GROUP OPERATIONS
000016                  U.NSEC  =       U.CTRK+1        ; NUMBER OF SECTORS R/W THIS TRY
000017                  U.MSEC  =       U.NSEC+1        ; NUMBER OF SECTORS TO BE R/W
000020                  U.TSEC  =       U.MSEC+1        ; NUMBER OF SECTORS TO BE R/W THIS OP
000021                  U.CSEC  =       U.TSEC+1        ; COUNT OF SECTORS R/W SO FAR
000022                  U.MASK  =       U.CSEC+1        ; UNIT MASK FOR XFC CALLS (0001 - 1000)
000023                  U.WRIT  =       U.MASK+1        ; WRITE PROTECTION STATUS
000024                  U.ELEV  =       U.WRIT+1        ; CURRENT ERROR RECOVERY LEVEL
000025                  U.RTRY  =       U.ELEV+1        ; MAXIMUM NUMBER OF READ RETRIES
000026                  U.MLEV  =       U.RTRY+1        ; MAXIMUM NUMBER OF ERROR RECOVERY LEVELS
000027                  U.ECCT  =       U.MLEV+1        ; ECC THRESHOLD
000030                  U.SDIS  =       U.ECCT+1        ; SDI SHORT TIMEOUT
000031                  U.SDIL  =       U.SDIS+1        ; SDI LONG TIMEOUT
000032                  U.WPRT  =       U.SDIL+1        ; MASK TO WRITE PROTECT READ-ONLY DRIVES
000033                  U.PARM  =       U.WPRT+1        ; UNIT PARAMETER WORD
000034                  U.SUBU  =       U.PARM+1        ; SUBUNIT OFFSET (0 - 3)
000035                  U.MBN   =       U.SUBU+1        ; MASTER L/DBN
000037                  U.CBN   =       U.MBN+2         ; CURRENT L/DBN FOR START OF CHAIN
000041                  U.RBN   =       U.CBN+2         ; RBN TO BE READ/WRITTEN IF LBN REVECTORED
000043                  U.COPY          U.RBN+2         ; NUMBER OF RCT COPIES ON EACH SUBUNIT
000044                  U.CCOP          U.COPY+1        ; CURRENT RCT COPY THAT WE'RE WORKING ON
000045                  U.RWER  =       U.CCOP+1        ; ERROR (IF ANY) ON THE LAST R/W
000046                  U.RVER  =       U.RWER+1        ; ERROR THAT OCCURRED BEFOR THE REVECTOR OPERATION
000047                  U.SNUM          U.RVER+1        ; 4 WORDS THAT HOLD THE SUBUNIT LOGICAL NUMBERS
000053                  U.CCYL  =       U.SNUM+4        ; CURRENT CYLINDER
000055                  U.CGRP  =       U.CCYL+2
000056                  U.LCYL  =       U.CGRP+1        ; LAST CYLINDER SEEKED TO
000060                  U.LGRP  =       U.LCYL+2        ; LAST GROUP SEEKED TO
                        ;
                        ;       SUBUNIT PARAMETER OFFSET
                        ;
000000                  S.PARM  =       0.              ; SUBUNIT PARAMETER WORD
000001                  S.SDCL  =       S.PARM+1        ; STARTING DIAGNOSTIC CYLINDER
000003                  S.PAT   =       S.SDCL+2        ; PATTERN TO USE FOR WRITES
```

```
                                ;
                                ;       IF TRACK/GROUP LIMITS ARE GIVEN, THE SUBUNIT PARAMETERS HAVE THE
                                ;       FOLLOWING FIELDS ADDED TO THEM
                                ;
000011          S.MCNT  =       S.BESS          ; MAXIMUM TRACK/GROUP COUNT
000013          S.TGOF  =       S.MCNT+2        ; ORIGINAL TRACK/GROUP OFFSET
000015          S.TGSS  =       S.TGOF+2        ; START OF TRACK/GROUP SETS
```

```
                                ;
                                ;       DUMMY SDI CONTROL BLOCK OFFSETS
                                ;
000001                          D.LIMT  =       1                       ; DUMMY SDI SEARCH LIMIT
000002                          D.SCHR  =       2                       ; DUMMY POINTER TO SUBUNIT CHAR-5
                                ;
                                ;
                                ;       UNIT PARAMETER BITS (U.PARM(R5))
                                ;
100000                          DROP    =       100000                  ; DROP BIT (SET IF UNIT OR SUBUNIT DROPPED)
040000                          INITW   =       40000                   ; INITIAL WRITE (SET IF INITIAL WRITE IN PROG)
020000                          RESEEK  =       20000                   ; IF 1, INDICATES THAT A SEEK IS NECESSARY
010000                          DIREC   =       10000                   ; DIRECTION (SET IF SEQUENTIAL ACCESSES DECREASING)
004000                          NEWSUB  =       4000                    ; SET IF SEQUENTIAL SEEKS MOVED TO NEW SUBUNIT
002000                          SEKINP  =       2000                    ; SEEK IN PROGRESS - SET IF TRUE
001000                          FTIME   =       1000                    ; FIRST TIME FLAG - SET FOR INIT CODE
000400                          REVEC           400                     ; REVECTOR BIT (SET IF BLOCK REVECTORED)
000200                          RBNBN   =       200                     ; SEE IF WORKING ON RBN
000100                          REDWRT  =       100                     ; REDWRT (READ OR WRITE IN PROGRESS SET IF WRITE)
000040                          REVINP  =       40                      ; REVECTORING OPERATION IN PROGRESS
000020                          DATERR  =       20                      ; DATA ERROR IF SET
000004                          RETRY   =       4                       ; IF CLEAR, START RETRIES AT ZERO
000001                          RCLB    =       1                       ; RECALIBRATION BIT (SET IF RECALIBRATE JUST DONE)
                                ;
                                ;       SUBUNIT PARAMETER BITS (S.PARM(R4))
                                ;
020000                          DCYLS   =       20000                   ; DIAGNOSTIC CYLINDER FLAG (SET IF DBNS)
010000                          ECCCHK  =       10000                   ; 1 IF ECC CORRECTION ALLOWED   MASTER BITS
004000                          RONLY   =       4000                    ; READ ONLY (SET IF READ ONLY)
002000                          WONLY   =       2000                    ; WRITE ONLY (SET IF WRITE ONLY)
001000                          RTRIES  =       1000                    ; 1 IF RETRIES ALLOWED
000200                          ONLYCL  =       200                     ; SET IF ONLY CYLINDERS SPECIFIED
                                                                        ; USED DURING SETUP ONLY
000100                          SEQSEK  =       100                     ; SEQUENTIAL SEEK (START UP TESTING ONLY)
000040                          BEUSED  =       40                      ; BEGIN/END SETS (USED IF SET)
000020                          TRACKS  =       20                      ; TRACKS OR GROUPS (TRACK IF SET)
000010                          WCHECK  =       10                      ; WRITE CHECK BIT (IF SET, WRITE CHECK WILL BE DONE)
                                ;                               WCHECK IS ALSO USED FOR THE UNIT PARAMETERS
000004                          WCHKAL  =       4                       ; SET IF WRITE CHECK ALWAYS TO BE DONE
000002                          DATCMP  =       2                       ; DATA COMPARE (SET IF DATA COMPARE TO BE DONE)
                                ;                               DATCMP IS ALSO USED FOR THE UNIT PARAMETERS
000001                          DCMPAL  =       1                       ; SET IF DATA COMPARE ALWAYS TO BE DONE
```

```
                        .SBTTL  MACRO DEFINITIONS
                        ;
                        ;       DIAGNOSTIC MACRO FOR TEST4 OVERLAYS
                        ;
                                .MACRO  DIAG$$
                                TST     $$DIAG+$DIAG$
                                BEQ     .+6
                                MOV     #60000,R0
                                MOV     R0,@$$DIAG+$DIAG$
                                BR      .+1
                        $DIAG$  -       $DIAG$ + 1
                                .ENDM
```

```
                        ;       MESSAGE CONTROL TABLE MACRO
                        ;
                        .MACRO MSG CMDBUF,CMDSZ,RPLBUF,RPLSZ,SUCCOM
                        .WORD CMDBUF                    ;ADDRESS OF COMMAND
                        .WORD CMDSZ                     ;SIZE OF COMMAND IN BYTES
                        .WORD RPLBUF                    ;ADDRESS OF REPLY
                        .WORD RPLSZ                     ;SIZE OF REPLY IN WORDS
                        .IF NB NUMBER
                        .WORD SUCCOM                    ; SUCCESSFUL COMPLETION CODE
                        .ENDC
                        .ENDM
```

MACRO DEFINITIONS

```
      1                                .MACRO  BCS     LAB..
      2                                                        BCC     .+2
      3                                                        BR      LAB..
      4                                .ENDM
```

MACRO DEFINITIONS

```
                        ;       PUSH REGISTER MACRO
                        ;
                                .MACRO PUSH R9
                                .IRP X,<R9>
                                MOV X,-(SP)
                                .ENDR
                                .ENDM
                        ;
                        ;       POP REGISTER MACRO
                        ;
                                .MACRO POP R9
                                .IRP X,<R9>
                                MOV (SP)+,X
                                .ENDR
                                .ENDM
```

```
                        ;ERROR MACROS
                        ;THESE MACROS ARE CALLED TO REPORT ERRORS TO THE HOST PROGRAM.
                        ;THE MACRO NAMES ARE : ERRSF, ERRDF, ERRHRD, ERRSFT. EACH RESULTS IN THE HOST
                        ;BEING REQUESTED TO REPORT THE ERROR.
                        ;ARGUMENTS:     1 (MS$) MESSAGE POINTER
                        ;               2 (P1$) PARAMETER #1
                        ;               3 (P2$) PARAMETER #2
                        ;               4 (P3$) PARAMETER #3
                        ;               5 (P4$) PARAMETER #4
                        ;               6 (P5$) PARAMETER #5
                        ;               7 (P6$) PARAMETER #6
                        ;               8 (P7$) PARAMETER #7
                        ;               9 (P8$) PARAMETER #8
                        ;
                        ;THE MESSAGE POINTER MUST POINT TO AN ADDRESS IN THE OVERLAY 'MS' IMMEDIATELY
                        ;FOLLOWING THE MAIN CODE. ANY ADDRESS MODE MAY BE USED (E.G. #MS1, @R2).
                        ;THE ADDRESS MUST CONTAIN AN ASCII FORMAT STRING TO DETERMINE THE MESSAGE
                        ;TO PRINT.
                        ;THE PARAMETER ARGUMENTS ARE OPTIONAL. THEY SHOULD BE SUPPLIED ONLY WHEN
                        ;THERE IS DATA TO BE PASSED TO THE HOST THAT WILL BE USED IN PRINTING THE
                        ;MESSAGE. THESE PARAMETER ARGUMENTS ARE THE ADDRESS OF DATA TO BE PASSED
                        ;USING ANY ADDRESSING MODE DESIRED.
                        ;ALL REGISTERS ARE RETURNED UNCHANGED. IT SHOULD BE NOTED THAT ARGUMENTS
                        ;CONTAINING SOMETHING OTHER THAN A REGISTER NAME (E.G. #100 OR MEMADR)
                        ;ASSEMBLE TO INSTRUCTIONS THAT SAVE AND RESTORE A REGISTER ON THE STACK.

                                .MACRO ERRSF MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ 0,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .ENDM

                                .MACRO ERRDF MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ 1,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .ENDM

                                .MACRO ERRHRD MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NLIST
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ 2,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .LIST
                                .ENDM

                                .MACRO ERRSFT MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ 3,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .ENDM
```

```
                        ;THE FOLLOWING MACRO ACTUALLY PROCESSES THE ERROR CALL TO THE HOST PROGRAM

                .MACRO ERROR$ ET$,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,ERRN$
                  .RADIX 8
                  PRMS=ARGS$-1
                  .IF LT,<PRMS>
                    .ERROR;NOT ENOUGH ARGUMENTS IN ERROR CALL
                  .ENDC
                  REGS$=-1
                  .IIF GE,<PRMS-8.>,PARGS. P8$
                  .IIF GE,<PRMS-7.>,PARGS. P7$
                  .IIF GE,<PRMS-6.>,PARGS. P6$
                  .IIF GE,<PRMS-5.>,PARGS. P5$
                  .IIF GE,<PRMS-4.>,PARGS. P4$
                  .IIF GE,<PRMS-3.>,PARGS. P3$
                  .IIF GE,<PRMS-2.>,PARGS. P2$
                  .IIF GE,<PRMS-1.>,PARGS. P1$
                  .IF GE REGS$
                    RSTR$ \REGS$
                  .ENDC
                  .RADIX 10
                  .LIST
                                                                CALL RERROR     ;ERROR # ERRN$'.
                  .NLIST
                  .RADIX 8
                  .LIST
                                                                .WORD <PRMS*2000>+<ET$*400>+ERRN
                                                                .WORD MS$
                  .NLIST
                  ERRN=ERRN+1
                .ENDM

                .MACRO PARGS.,ADDR$
                  .NTYPE PTYPE$,ADDR$
                  .IF EQ,<PTYPE$&70>
                    .IIF EQ,<PTYPE$&7>-REGS$,RSTR$ \REGS$
                    .LIST
                                                                MOV ADDR$,-(SP)
                    .NLIST
                  .IFF
                    .IF EQ,<PTYPE$&7>-1         ;PICK A REGISTER TO USE
                      REGU$-2                   ;SELECT R2 IF R1 IS USED IN PARAMETER FETCH
                    .IFF
                      REGU$=1                   ;OTHERWISE USE R1
                    .ENDC
                    .IF NE,<REGU$-REGS$>        ;IF REGISTER NOT ALREADY SAVED
                      .IF GE,REGS$
                        RSTR$ \REGS$            ;RESTORE CURRENT SAVED REGISTER
                      .ENDC
                      SAVR$ \REGU$              ;THEN SAVE SELECTED REGISTER
                    .ENDC
                    GETP$ \REGS$,ADDR$
                  .ENDC
                .ENDM
```

MACRO DEFINITIONS

```
                        .MACRO SAVR$ REGN
                          .LIST
                                                        MOV R'REGN,SAVREG
                          .NLIST
                          REGS$=REGN
                        .ENDM

                        .MACRO RSTR$ REGN
                          .LIST
                                                        MOV SAVREG,R'REGN
                          .NLIST
                          REGS$=-1
                        .ENDM

                        .MACRO GETP$ REGN,ADDR$
                          .LIST
                                                        MOV ADDR$,R'REGN
                                                        MOV R'REGN,-(SP)
                          .NLIST
                        .ENDM
```

```
                        ;
                        ;       PRIMARY ERROR REPORTING (TEST 4)
                        ;

                        .MACRO  SOFTER  NUM,ARGS
                        ERROR   #ERSOFT,NUM,<ARGS>
                                                        MOV     #ERRMES,R2
                                                        MOV     R2,OUT.RQ

                        .ENDM

                        .MACRO  REPSFT  SFTFLG,ECCFG
                        .IF     NB,SFTFLG
                                                        MOV     #1,OUT.02

                        .IFF
                                                        CLR     OUT.02

                        .ENDC
                        .IF     NB,ECCFG
                                                        MOV     #1,OUT.03

                        .IFF
                                                        CLR     OUT.03

                        .ENDC
                                                        PUSH    R0
                                                        MOV     #U.SNUM,R0
                                                        ADD     R5,R0
                                                        ADD     U.SUBU(R5),R0
                                                        MOV     (R0),OUT.01
                                                        MOV     #T4SOFT,R0
                                                        CALL    HOSTRQ
                                                        POP     R0

                        .ENDM

                        .MACRO  HARDER  NUM,ARGS
                        ERROR   #ERHARD,NUM,<ARGS>
                                                        MOV     #ERRMC,R2
                                                        MOV     R2,OUT.RQ

                        .ENDM

                        .MACRO  DEVFTL  NUM,ARGS
                        ERROR   #FTLDEV,NUM,<ARGS>
                                                        MOV     #ERRMC,R2
                                                        MOV     R2,OUT.RQ

                        .ENDM

                        .MACRO  SYSFTL  NUM,ARGS
                        ERROR   #FTLSYS,NUM,<ARGS>
                                                        MOV     #ERRMC,R2
                                                        MOV     R2,OUT.RQ

                        .ENDM

                        .MACRO  ERROR   TYPE,NUM,ARGS
                        .RADIX  10
                        NUMPTR  =       5
                        MOVMSG  #ER'NUM,4
                        .IF     NB,<ARGS>
                        .IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
                        NUMPTR  =       NUMPTR + 1
                        .ENDR
```

MACRO DEFINITIONS

```
                        .ENDC
                                                        MOV     #NUM,OUT.02
                                                        BIS     TYPE,OUT.02
                                                        MOV     #.,OUT.01
                        .RADIX
                        .ENDM

                        .MACRO  CERROR  STNUM,ARGS
                        .RADIX  10
NUMPTR                  =       STNUM
                        .IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
NUMPTR                  =       NUMPTR + 1
                        .ENDR
                        .RADIX
                        .ENDM

                        .MACRO  ERRORC  ARGS
                        .RADIX  10
                        .IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
NUMPTR                  =       NUMPTR + 1
                        .ENDR
                        .RADIX
                        .ENDM

                        .MACRO  MOVMSG  ARG,INDX

                        .IF     LT,INDX-10
                                                        MOV     ARG,OUT.0'INDX
                        .IFF
                                                        MOV     ARG,OUT.'INDX
                        .ENDC
                        .ENDM

                        .MACRO  ENDERR  POS
                        .IF     NB,POS
                        NDERR   POS
                        .IFF
                        NDERR   \NUMPTR
                        .ENDC
                        .ENDM

                        .MACRO  NDERR   POS
                        .IF     NE,POS
                                                        BIS     #POS,ERRPOS     ; SET THE POSITION
                        .IFF
                                                        CLR     ERRPOS          ; CLEAR THE POSITION
                        .ENDC
                        .ENDM

                ;       MESSAGE REPORTING MACRO

                        .MACRO  MSSG    NUM,ARGS
                        .RADIX  10
                        NUMPTR  =       3
                        MOVMSG  #MS'NUM,2
```

MACRO DEFINITIONS

```
115                                     .IF     NB,<ARGS>
116                                     .IRP    X,<ARGS>
117                                     MOVMSG  X,\NUMPTR
118                                     NUMPTR  =       NUMPTR + 1
119                                     .ENDR
120                                     .ENDC
121                                                             PUSH    <R0,R1>
122                                                             MOV     #U.SNUM,R1
123                                                             ADD     R5,R1
124                                                             ADD     U.SUBU(R5),R1
125                                                             MOV     (R1),OUT.01
126                                                             MOV     #MESSAG,R0
127                                                             CALL    HOSTRQ
128                                                             POP     <R1,R0>
129                                     .RADIX
130                                     .ENDM
131                                     ;
132                                     ;
133                                     .MACRO  MSSGE   NUM,ARGS
134                                     .RADIX  10
135                                     NUMPTR  =       3
136                                     MOVMSG  #'NUM,?
137                                     .IF     NB,<ARGS>
138                                     .IRP    X,<ARGS>
139                                     MOVMSG  X,\NUMPTR
140                                     NUMPTR  -       NUMPTR + 1
141                                     .ENDR
142                                     .ENDC
143                                                             PUSH    <R0,R1>
144                                                             MOV     #MESSAG,R0
145                                                             CALL    HOSTRQ
146                                                             POP     <R1,R0>
147                                     .RADIX
148                                     .ENDM
```

MACRO DEFINITIONS

```
                        ;RETURN DRIVE STATUS MACRO WITH ERROR REPORTING

                                .MACRO  DSTAT,LAB$,E1,E2
                                .NLIST
                                .NLIST  MEB
                                .LIST   MF
                                .LIST
                                CALL    RDSTAT                  ; GET DRIVE STATUS
                                BIT     #10000,R1               ; SEE IF ANY ERRORS
                                BEQ     2$                      ; IF NO ERROR, BRANCH
                                BIT     #4000,R1                ; SEE IF XMIT ERROR
                                BEQ     1$                      ; IF SO, BRANCH
                                ERRHRD  E1                      ; REPORT INVALID STATUS ERROR
                                BR      LAB$                    ; BRANCH TO DONE
                        1$:     ERRHRD  E2                      ; REPORT XMIT ERROR
                                BR      LAB$                    ; BRANCH TO DONE
                        2$:
                                .NLIST
                                .NLIST  ME
                                .LIST   MEB
                                .LIST
                                .ENDM
```

MACRO DEFINITIONS

```
                        ;
                        ;       XOR THE CONTENTS OF TWO REGISTERS
                        ;
                        .MACRO  RXOR    REG1,REG2
                        MOV     REG2,-(SP)              ; SAVE REGISTER REG2
                        BIC     REG1,REG2               ; CLEAR COORESPONDING BITS IN REG2
                        BIC     (SP)+,REG1              ; CLEAR COORESPONDING BITS IN REG1
                        BIS     REG1,REG2               ; OR WHAT'S LEFT
                        .ENDM
```

MACRO DEFINITIONS

```
                        ;       SDI INTERCHANGE WITH DRIVE WITH ERROR REPORTING
                        
                                .MACRO  TALKX   ERRLAB,E1,E2
                                .NLIST
                                .NLIST  MEB
                                .LIST   MF
                                .LIST
                                CALL    TALKER                  ; INITIATE SDI INTERCHANGE
                                TST     R3                      ; SEE IF ERROP OCCURRED
                                BEQ     12$                     ; IF NOT, BRANCH
                                BPL     11$                     ; IF SO, BRANCH
                                ERRHRD  E1;SEND COMMAND ERROR
                                BR      ERRLAB
                        11$:    ERRHRD  E2;RECEIVE COMMAND ERROR
                                BR      ERRLAB
                        12$:
                                .NLIST
                                .NLIST  ME
                                .LIST   MEB
                                .LIST
                                .ENDM
```

```
                                .SBTTL  START OF TEST CODE
                                ;THE FOLLOWING IS FOR DEBUG PURPOSES ONLY. CAN BE NOP OR BREAKPOINT.
                                
000714  114007                          CLR R0                          ;CHANGE TO BREAKPOINT FOR DEBUG
                                
                                ;INITIALIZE STACK
                                
000715  104206  001357                  MOV #STACK,SP                   ;SET UP STACK POINTER
000717  001360                          BR      START                   ; BRANCH OVER SUPPORT CODE
```

RDSTAT - GET DRIVE'S REAL TIME DRIVE STATE

```
                                .SBTTL  RDSTAT - GET DRIVE'S REAL TIME DRIVE STATE
000720                          RDSTAT:
                                ;
                                ;RETURN DRIVE STATUS
                                ;STATUS RETURNED IN DM REGISTER 1
                                ;
000720                                  PUSH    <R3,R0>                 ; SAVE R3 AND R0
000720  100463                                                                  MOV R3,-(SP)
000721  100467                                                                  MOV R0,-(SP)
000722  104203  000003                 MOV     #3,R3                   ; ALLOW ONLY 3 ERRORS
000724  060007                  STATLP: XFC     STATUS                  ; GET DRIVE'S STATUS
000725  103201  014000                  BIC     #14000,R1               ; CLEAR ERROR PASSING BITS
000727  102201  000400                  BIT     #XMTERR,R1              ; CHECK XMIT ERRORS
000731  010737                          BEQ     STATOK                  ; IF NO ERRORS, BRANCH
000732  117403                          DEC     R3                      ; DECREMENT TRANSMIT ERROR COUNT
000733  050724                          BNE     STATLP                  ; IF ERROR COUNT INCOMPLETE, BRANCH
000734  104201  010000                  MOV     #10000,R1               ; FLAG AS TRANSMIT ERROR
000736  000746                          BR      STATEX                  ; BRANCH
000737  102201  000004          STATOK: BIT     #RCVERR,R1              ; RECIEVER ERRORS
000741  050746                          BNE     STATEX                  ; IF VALID, BRANCH
000742  117403                          DEC     R3                      ; DECREMENT ERROR COUNT
000743  050724                          BNE     STATLP                  ; IF ERROR COUNT NON-ZERO, BRANCH
000744  104201  014000                  MOV     #14000,R1               ; FLAG AS INVALID STATUS ERROR
000746                          STATEX: POP     <R0,R3>                 ; RESTORE R0, R3
000746  104267                                                                  MOV (SP)+,R0
000747  104263                                                                  MOV (SP)+,R3
000750  000000                          RETURN                          ; RETURN TO CALLING MODULE
```

```
                                .SBTTL  HOSTRQ - HOST REQUEST - REPORT ERRORS, MEGABYTES TRANSFERRED, ETC.
000751                          HOSTRQ:
                                ;
                                ;SEND REQUEST BUFFER TO HOST AND WAIT FOR RESPONSE.
                                ;CLEAR ARGUMENT AREA OF OUT BUFFER IN PREPARATION
                                ;FOR NEXT HOSTRQ CALL.
                                ;
                                ;INPUTS:
                                ;       R0 - HOST REQUEST NUMBER
                                ;       OUT BUFFER LOADED WITH DATA

000751                                  PUSH <R0,R1,R2>
000751  100467                                                                  MOV R0,-(SP)
000752  100461                                                                  MOV R1,-(SP)
000753  100462                                                                  MOV R2,-(SP)
000754  104070  001006                 MOV R0,OUT.RQ                   ;STORE REQUEST NUMBER IN BUFFER
000756  104207  001006          SNDAGN: MOV #OUT.RQ,R0                  ;SEND BUFFER TO HOST
000760  104201  000036                  MOV #BUFSIZ,R1
000762  060016                          XFC MRD
000763  115001                          TST R1                          ;CHECK FOR ERROR
000764  050756                          BNE SNDAGN                      ;IF ERROR, TRY AGAIN
000765  104207  001044                  MOV #IN.RQ,R0                   ;WAIT FOR RESPONSE FROM HOST
000767  104201  000036                  MOV #BUFSIZ,R1
000771  060017                          XFC MWR
000772  104207  001007                  MOV #OUT.01,R0                  ;CLEAR ARGUMENT WORDS IN BUFFER
000774  104201  000034                  MOV #OUT.29-OUT.01,R1
000776  114002                          CLR R2
000777  100272                  CLRBUF: MOV R2,(R0)+
001000  117401                          DEC R1
001001  030777                          BPL CLRBUF
001002                                  POP <R2,R1,R0>
001002  104262                                                                  MOV (SP)+,R2
001003  104261                                                                  MOV (SP)+,R1
001004  104267                                                                  MOV (SP)+,R0
001005  000000                          RETURN
```

```
                                ;STORAGE AREA FOR MAINTENANCE WRITE AND READ BUFFERS
                                ;
                                ;OUT BUFFER - DATA TO SEND TO HOST
                                ;
001006  000000  OUT.RQ: .WORD 0         ;HOST REQUEST CODE
001007  000000  OUT.01: .WORD 0         ;DATA ARGUMENT  1
001010  000000  OUT.02: .WORD 0         ;DATA ARGUMENT  2
001011  000000  OUT.03: .WORD 0         ;DATA ARGUMENT  3
001012  000000  OUT.04: .WORD 0         ;DATA ARGUMENT  4
001013  000000  OUT.05: .WORD 0         ;DATA ARGUMENT  5
001014  000000  OUT.06: .WORD 0         ;DATA ARGUMENT  6
001015  000000  OUT.07: .WORD 0         ;DATA ARGUMENT  7
001016  000000  OUT.08: .WORD 0         ;DATA ARGUMENT  8
001017  000000  OUT.09: .WORD 0         ;DATA ARGUMENT  9
001020  000000  OUT.10: .WORD 0         ;DATA ARGUMENT 10
001021  000000  OUT.11: .WORD 0         ;DATA ARGUMENT 11
001022  000000  OUT.12: .WORD 0         ;DATA ARGUMENT 12
001023  000000  OUT.13: .WORD 0         ;DATA ARGUMENT 13
001024  000000  OUT.14: .WORD 0         ;DATA ARGUMENT 14
001025  000000  OUT.15: .WORD 0         ;DATA ARGUMENT 15
001026  000000  OUT.16: .WORD 0         ;DATA ARGUMENT 16
001027  000000  OUT.17: .WORD 0         ;DATA ARGUMENT 17
001030  000000  OUT.18: .WORD 0         ;DATA ARGUMENT 18
001031  000000  OUT.19: .WORD 0         ;DATA ARGUMENT 19
001032  000000  OUT.20: .WORD 0         ;DATA ARGUMENT 20
001033  000000  OUT.21: .WORD 0         ;DATA ARGUMENT 21
001034  000000  OUT.22: .WORD 0         ;DATA ARGUMENT 22
001035  000000  OUT.23: .WORD 0         ;DATA ARGUMENT 23
001036  000000  OUT.24: .WORD 0         ;DATA ARGUMENT 24
001037  000000  OUT.25: .WORD 0         ;DATA ARGUMENT 25
001040  000000  OUT.26: .WORD 0         ;DATA ARGUMENT 26
001041  000000  OUT.27: .WORD 0         ;DATA ARGUMENT 27
001042  000000  OUT.28: .WORD 0         ;DATA ARGUMENT 28
001043  000000  OUT.29: .WORD 0         ;DATA ARGUMENT 29
                                ;
                                ;IN BUFFER - DATA RECEIVED FROM HOST
                                ;
001044  000000  IN.RQ:  .WORD 0         ;HOST REQUEST CODE (ECHO)
001045  000000  IN.01:  .WORD 0         ;DATA ARGUMENT  1
001046  000000  IN.02:  .WORD 0         ;DATA ARGUMENT  2
001047  000000  IN.03:  .WORD 0         ;DATA ARGUMENT  3
001050  000000  IN.04:  .WORD 0         ;DATA ARGUMENT  4
001051  000000  IN.05:  .WORD 0         ;DATA ARGUMENT  5
001052  000000  IN.06:  .WORD 0         ;DATA ARGUMENT  6
001053  000000  IN.07:  .WORD 0         ;DATA ARGUMENT  7
001054  000000  IN.08:  .WORD 0         ;DATA ARGUMENT  8
001055  000000  IN.09:  .WORD 0         ;DATA ARGUMENT  9
001056  000000  IN.10:  .WORD 0         ;DATA ARGUMENT 10
001057  000000  IN.11:  .WORD 0         ;DATA ARGUMENT 11
001060  000000  IN.12:  .WORD 0         ;DATA ARGUMENT 12
001061  000000  IN.13:  .WORD 0         ;DATA ARGUMENT 13
001062  000000  IN.14:  .WORD 0         ;DATA ARGUMENT 14
001063  000000  IN.15:  .WORD 0         ;DATA ARGUMENT 15
001064  000000  IN.16:  .WORD 0         ;DATA ARGUMENT 16
001065  000000  IN.17:  .WORD 0         ;DATA ARGUMENT 17
001066  000000  IN.18:  .WORD 0         ;DATA ARGUMENT 18
001067  000000  IN.19:  .WORD 0         ;DATA ARGUMENT 19
```

```
 58 001070  000000          IN.20:  .WORD 0                 ;DATA ARGUMENT 20
 59 001071  000000          IN.21:  .WORD 0                 ;DATA ARGUMENT 21
 60 001072  000000          IN.22:  .WORD 0                 ;DATA ARGUMENT 22
 61 001073  000000          IN.23:  .WORD 0                 ;DATA ARGUMENT 23
 62 001074  000000          IN.24:  .WORD 0                 ;DATA ARGUMENT 24
 63 001075  000000          IN.25:  .WORD 0                 ;DATA ARGUMENT 25
 64 001076  000000          IN.26:  .WORD 0                 ;DATA ARGUMENT 26
 65 001077  000000          IN.27:  .WORD 0                 ;DATA ARGUMENT 27
 66 001100  000000          IN.28:  .WORD 0                 ;DATA ARGUMENT 28
 67 001101  000000          IN.29:  .WORD 0                 ;DATA ARGUMENT 29
 68         000036          BUFSIZ  =       . - IN.RQ       ;SIZE OF BUFFER
```

```
                                .IF     NE,TEST4
                                .NLIST
                                .IFF
                        ;
                        ;       TALKER SENDS AND RECEIVES AN SDI INTERCHANGE
                        ;
                        ;INPUTS:
                        ;       R2 - SDI INTERCONNECT
                        ;       R3 - POINTER TO COMMAND TABLE CONTAINING APPROPRIATE COMMAND
                        ;       SDILTO/SDISTO SDI LONG AND SHORT TIMEOUTS, RESPECTIVELY
                        ;OUTPUTS:
                        ;       R0 - RETURN OP CODE FROM UNIT
                        ;       R3 - ERROR CODE - 0 = NO ERROR, 1 = RECEIVE ERROR, 100001 = SEND ERROR
                        ;
001102                  TALKER: PUSH    <R1,R4>         ; SAVE REGISTERS
001102  100461                                                                 MOV R1,-(SP)
001103  100464                                                                 MOV R4,-(SP)
001104  104237                  MOV     (R3)+,R0        ; SET ADR OF SDI COMMAND BUFFER
001105  104231                  MOV     (R3)+,R1        ; SET BUFFER LENGTH
001106  060004                  XFC     SEND            ; SEND COMMAND
001107  115001                  TST     R1              ; DID UNIT ACCEPT COMMAND
001110  011114                  BEQ     TALK1A          ; IF SO, BRANCH
001111  104203  100001          MOV     #100001,R3      ; FLAG AS SEND ERROR
001113  001145                  BR      TALK2B          ; BRANCH TO EXIT
001114  106203  004626  TALK1A: CMP     #LONG,R3        ; SEE IF LONG TIMEOUT TO BE USED
001116  071122                  BMI     1$              ; IF SO, BRANCH
001117  104304  001150          MOV     SDISTO,R4       ; SET UP SHORT TIMEOUT
001121  001124                  BR      TALK1B          ; BRANCH
001122  104304  001151  1$:     MOV     SDILTO,R4       ; SET UP LONG TIMEOUT
001124  104137          TALK1B: MOV     (R3),R0         ; SET DATA BUFFER ADDRESS
001125  104631  000001          MOV     1(R3),R1        ; SET BUFFER LENGTH
001127  060005                  XFC     RCV             ; SEND RECEIVE SDI COMMAND
001130  115001                  TST     R1              ; DID ERROR OCCUR
001131  011144                  BEQ     TALK2A          ; IF NOT, BRANCH
001132  106201  000001          CMP     #1,R1           ; SEE IF TIMEOUT
001134  011137                  BEQ     1$              ; IF SO, BRANCH
001135  104013                  MOV     R1,R3           ; MOVE ERROR TYPE TO R3 FOR REPORTING
001136  001145                  BR      TALK2B          ; EXIT
001137  117404          1$:     DEC     R4              ; DECREMENT TIMEOUT VALUE
001140  051124                  BNE     TALK1B          ; IF NOT TIMEOUT, BRANCH
001141  104203  000001          MOV     #1,R3           ; FLAG AS RECIEVE ERROR
001143  001145                  BR      TALK2B          ; BRANCH TO EXIT
001144  114003          TALK2A: CLR     R3              ; FLAG AS NO ERRORS
001145                  TALK2B: POP     <R4,R1>         ; RESTORE R4, R1
001145  104264                                                                 MOV (SP)+,R4
001146  104261                                                                 MOV (SP)+,R1
001147  000000                  RETURN
                        ;
001150  000012          SDISTO: .WORD   10.             ; SDI SHORT TIMEOUT
001151  000024          SDILTO: .WORD   20.             ; SDI LONG TIMEOUT
```

```
                                ;MREAD
                                ;
                                ;READ ONE WORD FROM UNIBUS MEMORY
                                ;INPUTS:
                                ;       R5 - ADDRESS OF WORD TO READ (LOW 16 BITS)
                                ;       R4 -    ''           (HIGH 2 BITS)
                                ;OUTPUTS:
                                ;       R0 - DATA READ
                                ;
001152                          MREAD:  PUSH <R2,R3>                    ;SAVE SOME REGISTERS
001152  010462                                                                 MOV R2,-(SP)
001153  010463                                                                 MOV R3,-(SP)
001154  104057                          MOV R5,R0                       ;PUT UNIBUS ADDRESS IN R0 AND R1
001155  104041                          MOV R4,R1
001156  104202  000001                 MOV #1,R2                       ;TRANSFER ONE WORD
001160  104203  001210                 MOV #UDATA,R3                   ;LOCATION TO PUT DATA
001162  060013                          XFC UREAD                       ;DO THE READ
001163  104307  001210                 MOV UDATA,R0                    ;GET DATA READ
001165                                  POP <R3,R2>                     ;RESTORE OTHER REGISTERS
001165  104263                                                                 MOV (SP)+,R3
001166  104262                                                                 MOV (SP)+,R2
001167  000000                          RETURN
```

```
                                ;MWRITE
                                ;
                                ;WRITE ONE WORD TO UNIBUS MEMORY
                                ;INPUTS:
                                ;       R5 - ADDRESS OF WORD TO WRITE (LOW 16 BITS)
                                ;       R4 -    "           (HIGH 2 BITS)
                                ;       R0 - DATA TO WRITE
                                ;OUTPUTS:

001170                          MWRITE: PUSH <R0,R2,R3>                 ;SAVE SOME REGISTERS
001170  100467                                                                  MOV R0,-(SP)
001171  100462                                                                  MOV R2,-(SP)
001172  100463                                                                  MOV R3,-(SP)
001173  104070  001210                  MOV R0,UDATA                   ;PUT DATA TO BE WRITTEN INTO BUFFER
001175  104057                          MOV R5,R0                      ;PUT UNIBUS ADDRESS IN R0 AND R1
001176  104041                          MOV R4,R1
001177  104202  000001                  MOV #1,R2                      ;TRANSFER ONE WORD
001201  104203  001210                  MOV #UDATA,R3                  ;ADDRESS OF DATA WORD
001203  060014                          XFC UWRITE                     ;DO THE WRITE
001204                                  POP <R3,R2,R0>                 ;RESTORE THE REGISTERS
001204  104263                                                                  MOV (SP)+,R3
001205  104262                                                                  MOV (SP)+,R2
001206  104267                                                                  MOV (SP)+,R0
001207  000000                          RETURN

001210  000000                  UDATA:  .WORD 0                        ;DATA BUFFER FOR TRANSFERS TO AND FROM
                                                                       ;UNIBUS MEMORY
```

```
                                ;XOR
                                ;
                                ;PERFORM XOR LOGIC FUNCTION ON TWO REGISTERS
                                ;INPUTS:
                                ;       R1, R2 - DATA TO BE XOR'ED
                                ;OUTPUTS:
                                ;       R1 - UNCHANGED
                                ;       R2 - XOR OF TWO INPUTS
                                ;
001211                          XOR:    PUSH R3                         ;SAVE R3
001211  100463                                                                  MOV R3,-(SP)
001212  104013                          MOV R1,R3
001213  103023                          BIC R2,R3
001214  103012                          BIC R1,R2
001215  101032                          BIS R3,R2
001216                          ;       POP R3                          ;RESTORE R3
001216  104263                                                                  MOV (SP)+,R3
001217  115002                          TST R2                          ;SET CONDITION CODES
001220  000000                          RETURN
```

```
                                        TO:
                                        ;
                                        ;       CALCULATE THE TIMEOUT IN 9SEC INTERVALS (SDI RECEIVE XFC TAKES
                                        ;       9 SEC)
                                        ;
001221  104201  000001                  MOV     #1,R1           ; SET UP LOG2 SHIFTER
001223  110201                  1$:     ROL     R1              ; DOUBLE THE TIMEOUT VALUE
001224  103201  000001                  BIC     #1,R1           ; CLEAR THE LOW BIT
001226  117407                          DEC     R0              ; DECREMENT COUNT
001227  051223                          BNE     1$              ; IF COUNT INCOMPLETE, BRANCH
001230  114007                          CLR     R0              ; CLEAR 9SEC COUNT
001231  115407                  2$:     INC     R0              ; INCREMENT 9 SEC COUNT
001232  107201  000011                  SUB     #9.,R1          ; SUBTRACT 9 SEC FROM TIMEOUT
001234  031231                          BPL     2$              ; IF MORE TIME TO GO, BRANCH
001235  000000                          RETURN                  ; RETURN TO CALLING PROGRAM
```

```
;HOSTRQ - HOST REQUEST - REPORT ERRORS, MEGABYTES TRANSFERRED,

                                        ;RERROR
                                        ;
                                        ;REPORT ERROR TO HOST PROGRAM
                                        ;THIS ROUTINE IS CALLED BY THE ERROR MACROS:
                                        ;ERRSF, ERRDF, ERRHRD AND ERRSFT
                                        ;
001236                          RERROR: PUSH R0                 ;SAVE ONE REGISTER
001236  100467                                                          MOV R0,-(SP)
001237  104067                          MOV SP,R0               ;GET STACK POINTER
001240                                  PUSH <R1,R2,R3,R4>      ;SAVE MORE REGISTERS
001240  100461                                                          MOV R1,-(SP)
001241  100462                                                          MOV R2,-(SP)
001242  100463                                                          MOV R3,-(SP)
001243  100464                                                          MOV R4,-(SP)
001244  115407                          INC R0                  ;CHANGE SAVED STACK POINTER TO POINT TO
                                                                ; ADDRESS OF LOCATION AFTER CALL
001245  104271                          MOV (R0)+,R1            ;GET RETURN PC
001246  104202  001007                  MOV #OUT.01,R2          ;GET ADDRESS OF WHERE TO PUT DATA
001250  117401                          DEC R1                  ;REDUCE TO PC OF ERROR CALL
001251  100221                          MOV R1,(R2)+            ;PUT PC IN OUT BUFFER
001252  115401                          INC R1                  ;GET BACK TO RETURN PC
001253  104113                          MOV (R1),R3             ;GET ERROR NUMBER AND TYPE
001254  103203  176000                  BIC #^C001777,R3        ;CLEAR OTHER BITS
001256  100223                          MOV R3,(R2)+            ;PUT IN BUFFER
001257  104303  001315                  MOV LUNIT,R3            ;PUT UNIT NUMBER IN BUFFER
001261  100223                          MOV R3,(R2)+
001262  104214                          MOV (R1)+,R4            ;GET COUNT OF PARAMETERS
001263  104213                          MOV (R1)+,R3            ;GET MESSAGE POINTER
001264  100223                          MOV R3,(2)+             ;PUT IN OUT BUFFER
                                                                ;R1 IS NOW POINTING TO INSTRUCTION AFTER ERROR CALL
001265  110704                          SWAB R4
001266  110604                          ROR R4                  ;EXTRACT NUMBER OF PARAMETERS FROM ERROR MACRO
001267  110604                          ROR R4
001270  103204  177700                  BIC #177700,R4
001272  104040  001313                  MOV R4,SPADJU+1         ;SAVE FOR LATER ADJUSTMENT OF STACK POINTER
001274  011301                          BEQ RERRCA              ;BRANCH IF NO PARAMETERS
001275  104273                  RERRPA: MOV (R0)+,R3            ;GET PARAMETER
001276  100223                          MOV R3,(R2)+            ;STORE PARAMETER IN OUT BUFFER
001277  117404                          DEC R4                  ;COUNT THE PARAMETERS
001300  051275                          BNE RERRPA              ;GET NEXT IF MORE
001301  100471                  RERRCA: MOV R1,-(R0)            ;PUT RETURN ADDRESS ON STACK
001302  104207  000013                  MOV #ERRMES,R0          ;SEND ERROR PACKET TO HOST PROGRAM
001304  020751                          CALL HOSTRQ
001305                                  POP <R4,R3,R2,R1,R0>    ;RESTORE REGISTERS
001305  104264                                                          MOV (SP)+,R4
001306  104263                                                          MOV (SP)+,R3
001307  104262                                                          MOV (SP)+,R2
001310  104261                                                          MOV (SP)+,R1
001311  104267                                                          MOV (SP)+,R0
001312  105206  000000          SPADJU: ADD #0,SP               ;ADJUST STACK OVER PARAMETERS
                                                                ;VALUE CHANGED ABOVE
001314  000000                          RETURN

001315  177777                  LUNIT:  .WORD -1                ,LOGICAL UNIT NUMBER (-1 FOR NOT AVAILABLE)

001316  000000                  SAVREG: .WORD 0                 ;STORAGE FOR REGISTER AT CALL TIME
                                        .IFT
```

```
48                                      .LIST
49                                      .ENDC
```

```
                                        ;STACK AREA
                                
001317  123456                                .WORD 123456            ;END MARKER FOR STACK
001320                                      .BLKW 31.               ;STACK
001357  123456                      STACK:    .WORD 123456            ;MARKER FOR STACK UNDERFLOW
```

```
                                        .SBTTL  DISK DRIVE SEQUENCER
                                ;
                        004706  SUB     =       SUB + CR                ;MODIFY SUB TO POINT AT SUBUNIT CHAR
                        000001          ERRN=1.                         ;START ERROR NUMBERS AT 1.
001360                          START:
                                ;
                                ;       GET THE UNITS TO TEST
                                ;
                                ;
                                ;       POLL ALL PORTS AND FILL IN A UDA PORT INFORMATION TABLE (UNITS)
                                ;
001360  104205  000001          1$:     MOV     #1,R5           ; MOVE INITIAL MASK TO R5
001362  104204  004527                  MOV     #UNITS,R4       ; R4 POINTS TO UNIT TABLE
001364                                  PUSH    R4              ; SAVE R4
001364  100464                                                          MOV R4,-(SP)
001365  104052                          MOV     R5,R2           ; MOVE MASK TO R2
001366  060011                          XFC     DINIT           ; INITILIZE THE DRIVE
001367  104207  040000                  MOV     #40000,R0       ; MOVE TIMEOUT TO R0
001371  020720          14$:    CALL    RDSTAT          ; GET STATUS
001372  104203  003732                  MOV     #SER10,R3       ; GET SECONDARY ERROR
001374  100643  000001                  MOV     R3,1(R4)        ; PUT ERROR IN TABLE
001376  102201  010000                  BIT     #10000,R1       ; SEE IF STATUS IS VALID
001400  051412                          BNE     3$              ; IF NOT, BRANCH
001401  102201  000001                  BIT     #RCVRDY,R1      ; SEE IF RECEIVER READY ASSERTED
001403  104203  004005                  MOV     #SER11,R3       ; GET SECONDARY ERROR
001405  100643  000001                  MOV     R3,1(R4)        ; PUT ERROR IN TABLE
001407  100641  000002                  MOV     R1,2(R4)        ; STORE STATE
001411  051415                          BNE     5$              ; IF SO, BRANCH
001412  117407          3$:     DEC     R0              ; DECREMENT COUNT
001413  051371                          BNE     14$             ; IF INCOMPLETE, BRANCH
001414  001534                          BR      6$              ; NO VALID STATE
001415  104203  004607          5$:     MOV     #C.GST,R3       ; R3 POINTS TO GET STATUS COMMAND
001417  104237                          MOV     (R3)+,R0        ; SET ADR OF SDI COMMAND BUFFER
001420  104231                          MOV     (R3)+,R1        ; SET BUFFER LENGTH
001421  060004                          XFC     SEND            ; SEND COMMAND
001422                                  PUSH    R3              ; SAVE COUNT
001422  100463                                                          MOV R3,-(SP)
001423  104203  004055                  MOV     #SER12,R3       ; GET ERROR NUMBER
001425  100643  000001                  MOV     R3,1(R4)        ; SAVE
001427                                  POP     R3              ; RESTORE R3
001427  104263                                                          MOV (SP)+,R3
001430  115001                          TST     R1              ; DID UNIT ACCEPT COMMAND
001431  051534                          BNE     6$              ; IF NOT, BRANCH
001432                                  PUSH    R4              ; SAVE R4
001432  100464                                                          MOV R4,-(SP)
001433  104204  000012                  MOV     #10.,R4         ; SET UP SHORT TIMEOUT
001435  104137          11$:    MOV     (R3),R0         ; SET DATA BUFFER ADDRESS
001436  104631  000001                  MOV     1(R3),R1        ; SET BUFFER LENGTH
001440  060005                          XFC     RCV             ; SEND RECEIVE SDI COMMAND
001441  115001                          TST     R1              ; DID ERROR OCCUR
001442  011502                          BEQ     13$             ; IF NOT, BRANCH
001443  106201  000001                  CMP     #1,R1           ; SEE IF TIMEOUT
001445  051454                          BNE     12$             ; IF NOT, BRANCH
001446  117404                          DEC     R4              ; DECREMENT TIMEOUT VALUE
001447  051435                          BNE     11$             ; IF NOT TIMEOUT, BRANCH
001450                                  POP     R4              ; RESTORE R4
001450  104264                                                          MOV (SP)+,R4
```

```
     53 001451  104203  004106                MOV     #SER13,R3       ; GET ERROR NUMBER
     54 001453  001477                        BR      23$             ; BRANCH TO EXIT
     55 001454                        12$:    POP     R4              ; RESTORE R4
        001454  104264                                                                  MOV (SP)+,R4
     56 001455  110601                        ROR     R1              ; ROTATE INTO POSITION TO TEST
     57 001456  110601                        ROR     R1              ; SEE IF FIRST WORD NOT START FRAME
     58 001457  041463                        BCC     20$             ; IF NOT, BRANCH
     59 001460  104203  004141                MOV     #SER14,R3       ; GET ERROR NUMBER
     60 001462  001477                        BR      23$             ; BRANCH TO END OF LOOP
     61 001463  110601                20$:    ROR     R1              ; SEE IF FRAMING ERROR
     62 001464  041470                        BCC     21$             ; IF NOT, BRANCH
     63 001465  104203  004176                MOV     #SER15,R3       ; GET ERROR NUMBER
     64 001467  001477                        BR      23$             ; BRANCH TO END OF LOOP
     65 001470  110601                21$:    ROR     R1              ; SEE IF CHECKSUM ERROR
     66 001471  041475                        BCC     22$             ; IF NOT, BRANCH
     67 001472  104203  004241                MOV     #SER16,R3       ; GET ERROR NUMBER
     68 001474  001477                        BR      23$             ; BRANCH TO END OF LOOP
     69 001475  104203  004304        22$:    MOV     #SER17,R3       ; GET ERROR NUMBER
     70 001477  100643  000001        23$:    MOV     R3,1(R4)        ; SAVE
     71 001501  001534                        BR      6$              ; BRANCH TO END OF LOOP
```

```
                                ;
                                ;       NOW FILL IN THE TABLE WITH ALL THE SUBUNIT NUMBERS
                                ;
001502                  13$:    POP     R4              ; RESTORE R4
001502  104264                                                         MOV (SP)+,R4
001503  104207  177777          MOV     #-1,R0          ; GET 'NO UNITS' FLAG
001505  100647  000001          MOV     R0,1(R4)        ; CLEAR ANY ERRORS THAT ARE FLAGGED
001507  100647  000002          MOV     R0,2(R4)        ; CLEAR ANY ERRORS THAT ARE FLAGGED
001511  104307  004664          MOV     ST,R0           ; R0 HAS UNIT NUMBER
001513  104072                  MOV     R0,R2           ; COPY R0 TO R2
001514  103207  170000          BIC     #^CHBHINB,R0    ; R0 HAS UNIT NUMBER
001516  101207  040000          BIS     #4^000,R0       ; FLAG UNIT AS NOT TESTED
001520  110702                  SWAB    R2              ; SWAP R2'S BYTES
001521  110602                  ROR     R2              ; MOVE SUBUNIT MASK TO LO NIBBLE
001522  110602                  ROR     R2              ; MOVE SUBUNIT MASK TO LO NIBBLE
001523  110602                  ROR     R2              ; MOVE SUBUNIT MASK TO LO NIBBLE
001524  110602                  ROR     R2              ; MOVE SUBUNIT MASK TO LO NIBBLE
001525  103202  177760          BIC     #LBLONB,R2      ; CLEAR ALL BUT SUBUNIT BITS
001527  110602          4$:     ROR     R2              ; MOVE SUBUNIT BIT TO CARRY
001530  041534                  BCC     6$              ; IF NO MORE SUBUNITS, BRANCH
001531  100247                  MOV     R0,(R4)+        ; MOVE SUBUNIT NUMBER TO TABLE
001532  115407                  INC     R0              ; INCREMENT SUBUNIT NUMBER
001533  001527                  BR      4$              ; BRANCH


001534                  6$:     POP     R4              ; R4 POINTS TO START OF UNIT JUST HANDLED
001534  104264                                                         MOV (SP)+,R4
001535  105204  000004          ADD     #4,R4           ; R4 WILL POINT TO NEXT UNIT (CLEAR CARRY FOR ROL)
001537  110205                  ROL     R5              ; R5 HAS NEXT UNIT PORT MASK
001540  106205  000020          CMP     #20,R5          ; SEE IF ALL PORTS TESTED
001542  051364                  BNE     1$              ; IF NOT, BRANCH
```

```
                                ;
                                ;       NOW GET THE PLUG NUMBERS TO TEST AND FIND THEM IN THE TABLE
                                ;
001543  104207  000012            MOV     #UTOTST,R0      ; GET WHAT SUBUNIT NUMBERS TO TEST REQUEST
001545  020751                    CALL    HOSTRQ          ; GET THE PLUG NUMBERS
001546  104207  001045            MOV     #IN.01,R0       ; R0 POINTS TO UNIT NUMBERS TO TEST
001550  104201  004527    7$:     MOV     #UNITS,R1       ; R1 POINTS TO UDA PORT INFORMATION
001552  104112    8$:     MOV     (R1),R2         ; R2 HAS UNIT
001553  071576            BMI     9$              ; IF NONE, BRANCH
001554                    PUSH    R1              ; SAVE R1
001554  100461                                                    MOV R1,-(SP)
001555  103202  170000    42$:    BIC     #^CHBHINB,R2    ; CLEAR THE 'NOT TESTED BIT' FOR COMPARISON
001557  106172            CMP     (R0),R2         ; SEE IF IT IS A UNIT TO TEST
001560  051564            BNE     40$             ; NO MATCH
001561  100112            MOV     R2,(R1)         ; SAVE UNIT AS ONE TO TEST
001562                    POP     R1              ; RESTORE STACK
001562  104261                                                    MOV (SP)+,R1
001563  001712            BR      10$             ; LOOK FOR THE NEXT ONE
001564  115401    40$:    INC     R1              ; POINT TO NEXT SUBUNIT
001565  104012            MOV     R1,R2           ; COPY TO R2
001566  107202  004527    SUB     #UNITS,R2       ; SUBTRACT STARTING ADDRESS
001570  102202  000003    BIT     #3,R2           ; SEE IF STILL ON SAME UNIT
001572  011575            BEQ     41$             ; IF NOT, BRANCH
001573  104112            MOV     (R1),R2         ; SEE IF ANY MORE SUBUNITS
001574  031555            BPL     42$             ; IF SO, BRANCH
001575            41$.    POP     R1              ; RESTORE R1
001575  104261                                                    MOV (SP)+,R1
001576  105201  000004    9$:     ADD     #4,R1           ; LOOK AT NEXT UNIT
001600  106201  004547            CMP     #UNITS+16.,R1   ; SEE IF ENTIRE TABLE SEARCHED
001602  051552            BNE     8$              ; IF NOT, BRANCH
```

```
                                ;
                                ;       DIDN'T FIND THE REQUESTED UNITS -- DUMP AL  KNOWLEDGE OF THE
                                ;       JDA PORTS AND DIE
                                ;
001603                          PUSH    R0              ; SAVE POINTER TO UNIT
001603  100467                                                          MOV R0,-(SP)
001604  104204  001013          MOV     #OUT.05,R4      ; R4 POINTS TO OUTPUT BUFFER
001606  104205  004527          MOV     #UNITS,R5       ; R5 POINTS TO UNIT TABLE
001610  104157          30$:    MOV     (R5),R0         ; GET FIRST WORD OF UNIT
001611  031624                  BPL     31$             ; IF VALID UNIT WAS FOUND, BRANCH
001612  104657  000001          MOV     1(R5),R0        ; GET POINTER TO ERROR MESSAGE
001614  100247                  MOV     R0,(R4)+        ; SAVE IN OUTPUT BUFFER
001615  106207  004005          CMP     #SER11,R0       ; SEE IF ITS THE RECIEVER RDY NEVER ASSERTED
001617  051652                  BNE     36$             ; IF NOT, BRANCH
001620  104657  000002          MOV     2(R5),R0        ; GET STATE THAT WAS FOUND
001622  100247                  MOV     R0,(R4)+        ; SAVE
001623  001652                  BR      36$             ; EXIT
001624                  31$:    PUSH    R5              ; SAVE POINTER TO UNIT TABLE
001624  100465                                                          MOV R5,-(SP)
001625  104207  004374          MOV     #SER18,R0       ; GET POINTER TO ERROR MESSAGE
001627  100247                  MOV     R0,(R4)+        ; SAVE
001630  114007                  CLR     R0              ; CLEAR COUNT
001631  115405                  INC     R5              ; SKIP FIRST WORD (WE KNOW IT'S POSITIVE)
001632  104251          32$:    MOV     (R5)+,R1        ; GET UNIT NUMBER
001633  071640                  BMI     33$             ; IF INVALID, BRANCH
001634  115407                  INC     R0              ; INCREMENT COUNT
001635  106207  000003          CMP     #3,R0           ; SEE IF MAX
001637  051632                  BNE     32$             ; IF NOT, LOOP
001640  104671  001716  33$:    MOV     ERTXT(R0),R1    ; GET POINTER TO CORRECT ERROR MESSAGE
001642  100241                  MOV     R1,(R4)+        ; MOVE INTO OUTPUT BUFFER
001643                          POP     R5              ; RESTORE R5
001643  104265                                                          MOV (SP)+,R5
001644                          PUSH    R5              ; SAVE R5
001644  100465                                                          MOV R5,-(SP)
001645  104251          34$:    MOV     (R5)+,R1        ; GET SUBUNIT NUMBER
001646  100241                  MOV     R1,(R4)+        ; SAVE
001647  117407                  DEC     R0              ; DECREMENT COUNT
001650  031645                  BPL     34$             ; IF COUNT INCOMPLETE, BRANCH
001651                          POP     R5              ; RESTORE R5
001651  104265                                                          MOV (SP)+,R5
001652  105205  000004  36$:    ADD     #4,R5           ; POINT TO NEXT UNIT TABLE
001654  106205  004547          CMP     #UNITS+16.,R5   ; SEE IF ENTIRE TABLE SEARCHED
001656  051610                  BNE     30$             ; IF NOT, BRANCH
001657                          DEVFTL  80.             ; REPORT FATAL ERROR
001657  104200  003577  001012                                  MOV     #ER80,OUT.04
001662  104200  000120  001010                                  MOV     #80,OUT.02
001665  101200  000400  001010                                  BIS     #FTLDEV,OUT.02
001670  104200  001670  001007                                  MOV     #.,OUT.01
001673  104202  000014                                          MOV     #ERRMC,R2
001675  104020  001006                                          MOV     R2,OUT.RQ
001677                          POP     R0              ; GET POINTER TO UNIT NUMBER
001677  104267                                                          MOV (SP)+,R0
001700  104172                  MOV     (R0),R2         ; GET UNIT NUMBER
001701  104020  001011          MOV     R2,OUT.03       ; SAVE IN OUTPUT
001703  104207  000014          MOV     #ERRMC,R0       ; SET UP FOR REPORT
001705  020751                  CALL    HOSTRQ          ; REPORT ERROR
001706  104207  000016          MOV     #DONE,R0        ; END TEST
```

```
                                        CALL    HOSTRQ          ; SEND END-OF-TEST TO HOST
                                        XFC     EXIT            ; TERMINATE DM

                            10$:        INC     R0              ; POINT TO NEXT UNIT TO TEST
                                        MOV     (R0),R2         ; CHECK NEXT UNIT
                                        BPL     7$              ; FIND IN UDA PORT INFORMATION
                                        BR      T3STRT          ; START TEST

                            ERTXT:      .WORD   SER18D          ; POINTERS TO THE DIFFERENT SUBUNIT
                                        .WORD   SER18C          ; REPORTING MESSAGES
                                        .WORD   SER18B
                                        .WORD   SER18A
```

```
                                ;SEQUENCE THE DIAGNOSTICS TO ALL UNITS SELECTED.
                                ;TEST CODE WILL BE CALLED FOR EACH DISK SUBUNIT TO BE TESTED.
                                ;  LUNIT WILL CONTAIN LOGICAL UNIT NUMBER OF DRIVE FOR ERROR REPORTS
                                ;  SDI WILL CONTAIN SDI INTERCONNECT CODE FOR SELECTED DRIVE
                                ;  UNITNB WILL CONTAIN AN EVEN NUMBER FOR TESTING FIRST SUBUNIT OF A DRIVE
                                ;                      AN ODD NUMBER FOR TESTING SECOND SUBUNIT OF A DRIVE

001722  114001          T3STRT: CLR     R1              ;START WITH UNIT 0 INDEX

001723  104207  004527  PORT2:  MOV     #UNITS,R0       ; GET POINTER TO UNITS TABLE
001725  105017                  ADD     R1,R0           ; ADD INDEX
001726  104173                  MOV     (R0),R3         ; GET CONTENTS OF TABLE
001727  031736                  BPL     1$              ; IF THIS UNIT IS PRESENT, BRANCH
001730  102201  000003          BIT     #3,R1           ; SEE IF ON SUBUNIT 0 OF UNIT
001732  051770                  BNE     PORT5           ; IF NOT, TEST NEXT SUBUNIT
001733  105201  000003          ADD     #3,R1           ; IF NO SUBUNIT 0, THEN NO UNIT - SKIP OTHER SUBUNITS
001735  001770                  BR      PORT5           ; BYPASS IF NO UNIT
001736  102203  040000  1$:     BIT     #40000,R3       ; SEE IF THIS UNIT IS TO BE TESTED
001740  051770                  BNE     PORT5           ; IF NOT, BRANCH
001741  104010  004547          MOV     R1,UNITNB       ; STORE UNIT INDEX
001743  104030  001315          MOV     R3,LUNIT        ; STORE LOGICAL UNIT NUMBER FOR DRIVE
001745  104202  000001          MOV     #UNIT0,R2       ; GET UNIT 0 INTERCONNECT CODE
001747  110601                  ROR     R1              ; DIVIDE UNITNB BY FOUR
001750  110601                  ROR     R1
001751  103201  177760          BIC     #LBLONB,R1      ; CLEAR UNUSED BITS
001753  117401          PORT3:  DEC     R1
001754  071761                  BMI     PORT4           ; FOR EACH DRIVE OVER 0 SHIFT R2 LEFT
001755  110202                  ROL     R2
001756  103202  000001          BIC     #1,R2           ; CLEAR CARRY ROTATED INTO REG (IF ANY)
001760  001753                  BR      PORT3
001761  104020  004550  PORT4:  MOV     R2,SDI          ; STORE SDI INTERCONNECT CODE
001763  002000                  BR      TEST            ; PERFORM TEST ON THIS DRIVE
                                                        ; TEST RETURNS TO TESTX
001764  104206  001357  TESTX:  MOV     #STACK,SP       ; RESET STACK DO TO JUMPS OUT OF
                                                        ; SUBROUTINES
001766  104301  004547          MOV     UNITNB,R1       ; GET UNIT INDEX
001770  115401          PORT5:  INC     R1              ; INCREMENT INDEX
001771  106201  000020          CMP     #16.,R1         ; CHECK IF 16 DRIVES ALREADY SELECTED
001773  051723                  BNE     PORT2           ; REPEAT FOR ALL DRIVES
001774  104207  000016  DONECD: MOV     #DONE,R0        ; END OF PROGRAM
001776  020751                  CALL    HOSTRQ
001777  001774                  BR      DONECD          ; REPEAT IF RETURNED
```

INITIALIZE DRIVE AND LOOK AT DRIVE SIGNALS

```
                                .SBTTL  INITIALIZE DRIVE AND LOOK AT DRIVE SIGNALS

                                ;START OF TEST CODE
                                ;
                                ;INPUTS:
                                ;       LUNIT - LOGICAL UNIT NUMBER OF DRIVE UNDER TEST
                                ;       SDI - SDI INTERCONNECT CODE FOR DRIVE

                                ;INITIALIZE THE DRIVE

002000  104302  004550  TEST:   MOV SDI,R2                      ;GET SDI SELECT CODE
002002  060011                  XFC DINIT                       ;INITIALIZE THE DRIVE

                                ;WAIT FOR DRIVE TO ASSERT RECEIVER READY
                                ;TIME OUT AFTER ...?

002003  114005                  CLR     R5                      ;GET TIMEOUT COUNTER
002004          ILOOP:  DSTAT   TESTEX,MS1,MS2                  ;LOOK AT DRIVE STATUS LINES
002004  020720                  CALL    RDSTAT                  ; GET DRIVE STATUS
002005  102201  010000          BIT     #10000,R1               ; SEE IF ANY ERRORS
002007  012023                  BEQ     2$                      ; IF NO ERROR, BRANCH
002010  102201  004000          BIT     #4000,R1                ; SEE IF XMIT ERROR
002012  012017                  BEQ     1$                      ; IF SO, BRANCH
002013                          ERRHRD  MS1                     ; REPORT INVALID STATUS ERROR
002013  021236                                                          CALL RERROR     ;ERROR # 1.
002014  001001                                                          .WORD <PRMS*2000>+<2*400>+ERRN
002015  000000                                                          .WORD MS1
002016  003443                  BR      TESTEX                  ; BRANCH TO DONE
002017          1$:     ERRHRD  MS2                             ; REPORT XMIT ERROR
002017  021236                                                          CALL RERROR     ;ERROR # 2.
002020  001002                                                          .WORD <PRMS*2000>+<2*400>+ERRN
002021  000045                                                          .WORD MS2
002022  003443                  BR      TESTEX                  ; BRANCH TO DONE
002023          2$:
002023  102201  000001          BIT #RCVRDY,R1                  ;CHECK RECEIVER READY LINES
002025  052034                  BNE T                           ;ADVANCE IF ASSERTED
002026  117405                  DEC R5                          ;DECREMENT TIME OUT COUNTER
002027  052004                  BNE ILOOP                       ;STAY IN LOOP UNTIL SIGNAL SETS OR TIMEOUT
002030                          ERRHRD MS3                      ;REPORT ERROR
002030  021236                                                          CALL RERROR     ;ERROR # 3.
002031  001003                                                          .WORD <PRMS*2000>+<2*400>+ERRN
002032  000071                                                          .WORD MS3
002033  003443                  BR TESTEX                       ;EXIT TESTING THIS DRIVE
```

```
                                ;GET DRIVE CHARACTERISTICS
                                
002034  104200  000012  001150  T:      MOV     #10.,SDISTO             ; SET UP TEMPORARY SHORT TIMEOUT VALUE
002037  104203  004575                  MOV #CR.GCR,R3                  ;POINT TO GET CHARS COMMAND
002041                                  TALKX   TESTEX,MS26,MS27        ; SDI INTERCHANGE
002041  021102                         CALL    TALKER                  ; INITIATE SDI INTERCHANGE
002042  115003                         TST     R3                      ; SEE IF ERROR OCCURRED
002043  012055                         BEQ     12$                     ; IF NOT, BRANCH
002044  032051                         BPL     11$                     ; IF SO, BRANCH
002045                                  ERRHRD  MS26;SEND COMMAND ERROR
002045  021236                                                                  CALL RERROR     ;ERROR # 4.
002046  001004                                                                  .WORD <PRMS*2000>+<2*400>+ERRN
002047  001344                                                                  .WORD MS26
002050  003443                         BR      TESTEX
002051                          11$:    ERRHRD  MS27;RECEIVE COMMAND ERROR
002051  021236                                                                  CALL RERROR     ;ERROR # 5.
002052  001005                                                                  .WORD <PRMS*2000>+<2*400>+ERRN
002053  001405                                                                  .WORD MS27
002054  003443                         BR      TESTEX
002055                          12$:
002055  106207  000170                 CMP #CHRRES,R0                  ;CHECK FOR SUCCESSFUL RESPONSE
002057  012074                         BEQ T00
002060                                  ERRHRD MS28,#CHRRES,R0;GET CHARACTERISTICS COMMAND FAILED
002060  100467                                                                  MOV R0,-(SP)
002061  104010  001316                                                          MOV R1,SAVREG
002063  104201  000170                                                          MOV #CHRRES,R1
002065  100461                                                                  MOV R1,-(SP)
002066  104301  001316                                                          MOV SAVREG,R1
002070  021236                                                                  CALL RERROR     ;ERROR # 6.
002071  005006                                                                  .WORD <PRMS*2000>+<2*400>+ERRN
002072  001451                                                                  .WORD MS28
002073  003443                 BR      TESTEX                  ; BRANCH TO END OF TEST

002074  104307  004673         T00:    MOV     CR+SHRTTO,R0    ; GET SHORT TIMEOUT
002076  103207  177760                 BIC     #LBLONB,R0      ; CLEAR UNUSED BITS
002100  021221                         CALL    TO              ; SET UP TIMEOUT
002101  104070  001150                 MOV     R0,SDISTO       ; SAVE IN SHORT TIMEOUT
002103  104307  004674                 MOV     CR+LONGTO,R0    ; GET LONG TIMEOUT
002105  103207  177760                 BIC     #LBLONB,R0      ; CLEAR UNUSED BITS
002107  021221                         CALL    TO              ; SET UP TIMEOUT
002110  104070  001151                 MOV     R0,SDILTO       ; SAVE IN LONG TIMEOUT
```

```
                                ;ISSUE ONLINE COMMAND AND CHECK RESPONSE
 1
 2
 3 002112  104203  004551              MOV     #CR.ONL,R3              ; POINT TO ONLINE COMMAND
 4 002114                               TALKX   TESTEX,MS4,MS5          ; SDI INTERCHANGE
   002114  021102                       CALL    TALKER                  ; INITIATE SDI INTERCHANGE
   002115  115003                       TST     R3                      ; SEE IF ERROR OCCURRED
   002116  012130                       BEQ     12$                     ; IF NOT, BRANCH
   002117  032124                       BPL     11$                     ; IF SO, BRANCH
   002120                               ERRHRD  MS4;SEND COMMAND ERROR
   002120  021236                                                               CALL RERROR     ;ERROR # 7.
   002121  001007                                                               .WORD <PRMS*2000>+<2*400>+ERRN
   002122  000124                                                               .WORD MS4
   002123  003443                       BR      TESTEX
   002124                       11$:    ERRHRD  MS5;RECEIVE COMMAND ERROR
   002124  021236                                                               CALL RERROR     ;ERROR # 8.
   002125  001010                                                               .WORD <PRMS*2000>+<2*400>+ERRN
   002126  000154                                                               .WORD MS5
   002127  003443                       BR      TESTEX
   002130                       12$:
 5 002130  106207  000176              CMP     #COMPLT,R0              ; CHECK FOR ERROR
 6 002132  012147                       BEQ     T0                      ; IF NO ERROR BRANCH
 7 002133                               ERRHRD  MS6,#COMPLT,R0          ; ONLINE COMMAND FAILED
   002133  100467                                                               MOV R0,-(SP)
   002134  104C10  001316                                                       MOV R1,SAVREG
   002136  104201  00017c                                                       MOV #COMPLT,R1
   002140  100461                                                               MOV R1,-(SP)
   002141  104301  001316                                                       MOV SAVREG,R1
   002143  021236                                                               CALL RERROR     ;ERROR # 9.
   002144  005011                                                               .WORD <PRMS*2000>+<2*400>+ERRN
   002145  000207                                                               .WORD MS6
 8 002146  003443                       BR      TESTEX                  ; EXIT TEST
```

```
                                ;ISSUE GET STATUS COMMAND AND CHECK THAT ATTENTION DROPS

002147  104203  004607    TO:   MOV #CR.GST,R3               ;POINT TO GET STATUS COMMAND
002151                          TALKX   TESTEX,MS7,MS8        ; SDI INTERCHANGE
002151  021102                  CALL    TALKER                ; INITIATE SDI INTERCHANGE
002152  115003                  TSTB    R3                    ; SEE IF ERROR OCCURRED
002153  012165                  BEQ     12$                   ; IF NOT, BRANCH
002154  032161                  BPL     11$                   ; IF SO, BRANCH
002155                          ERRHRD  MS7;SEND COMMAND ERROR
002155  021236                                                          CALL RERROR     ;ERROR # 10.
002156  001012                                                          .WORD <PRMS*2000>+<2*400>+ERRN
002157  000275                                                          .WORD MS7
002160  003443                  BR      TESTEX
002161                  11$:    ERRHRD  MS8;RECEIVE COMMAND ERROR
002161  021236                                                          CALL RERROR     ;ERROR # 11.
002162  001013                                                          .WORD <PRMS*2000>+<2*400>+ERRN
002163  000327                                                          .WORD MS8
002164  003443                  BR      TESTEX
002165                  12$:
002165  106207  000366            CMP #STSRES,R0              ;CHECK FOR ERROR
002167  012204                    BEQ TOA
002170                           ERRHRD MS9,#STSRES,R0       ;GET STATUS COMMAND REJECTED
002170  100467                                                          MOV R0,-(SP)
002171  104010  001316                                                  MOV R1,SAVREG
002173  104201  000366                                                  MOV #STSRES,R1
002175  100461                                                          MOV R1,-(SP)
002176  104301  001316                                                  MOV SAVREG,R1
002200  021236                                                          CALL RERROR     ;ERROR # 12.
002201  005014                                                          .WORD <PRMS*2000>+<2*400>+ERRN
002202  000364                                                          .WORD MS9
002203  003443                  BR      TESTEX
002204          TOA:    DSTAT   TESTEX,MS10,MS2             ;LOOK AT STATUS LINES
002204  020720                  CALL    RDSTAT              ; GET DRIVE STATUS
002205  102201  010000          BIT     #10000,R1           ; SEE IF ANY ERRORS
002207  012223                  BEQ     2$                  ; IF NO ERROR, BRANCH
002210  102201  004000          BIT     #4000,R1            ; SEE IF XMIT ERROR
002212  012217                  BEQ     1$                  ; IF SO, BRANCH
002213                          ERRHRD  MS10                ; REPORT INVALID STATUS ERROR
002213  021236                                                          CALL RERROR     ;ERROR # 13.
002214  001015                                                          .WORD <PRMS*2000>+<2*400>+ERRN
002215  000454                                                          .WORD MS10
002216  003443                  BR      TESTEX              ; BRANCH TO DONE
002217          1$:     ERRHRD  MS2                         ; REPORT XMIT ERROR
002217  021236                                                          CALL RERROR     ;ERROR # 14.
002220  001016                                                          .WORD <PRMS*2000>+<2*400>+ERRN
002221  000045                                                          .WORD MS2
002222  003443                  BR      TESTEX              ; BRANCH TO DONE
002223          2$:
002223  102201  000002            BIT #ATTN,R1                ;CHECK THE ATTENTION LINE, THE GETS TATUS
002225  012232                    BEQ TOB                     ;COMMAND SHOULD HAVE CLEARED IT
002226                           ERRHRD MS12;ATTN DID NOT DROP AFTER GET STATUS COMMAND
002226  021236                                                          CALL RERROR     ;ERROR # 15.
002227  001017                                                          .WORD <PRMS*2000>+<2*400>+ERRN
002230  000500                                                          .WORD MS12
002231  003443                  BR      TESTEX
002232          TOB:
```

```
                                ;ISSUE DRIVE CLEAR COMMAND
                                
002232  104303  004666  T1:     MOV ST+ST.ERR,R3                ;GET ERROR BYTE
002234  103203  177400          BIC #HIBYTE,R3                  ;CLEAR ALL NON-ERROR BITS
002236  104030  004647          MOV R3,ERRORS                   ;MOVE TO DRIVE CLEAR SDI AREA
002240  104203  004556          MOV #CR.CLR,R3                  ;POINT TO DRIVE CLEAR
002242                          TALKX   TESTEX,MS13,MS14        ; SDI INTERCHANGE
002242  021102                  CALL    TALKER                  ; INITIATE SDI INTERCHANGE
002243  115003                  TST     R3                      ; SEE IF ERROR OCCURRED
002244  012256                  BEQ     12$                     ; IF NOT, BRANCH
002245  032252                  BPL     11$                     ; IF SO, BRANCH
002246                          ERRHRD  MS13;SEND COMMAND ERROR
002246  021236                                                          CALL RERROR     ;ERROR # 16.
002247  001020                                                          .WORD <PRMS*2000>+<2*400>+ERRN
002250  000535                                                          .WORD MS13
002251  003443                  BR      TESTEX
002252                  11$:    ERRHRD  MS14;RECEIVE COMMAND ERROR
002252  021236                                                          CALL RERROR     ;ERROR # 17.
002253  001021                                                          .WORD <PRMS*2000>+<2*400>+ERRN
002254  000567                                                          .WORD MS14
002255  003443                  BR      TESTEX
002256                  12$:
002256  106207  000176          CMP #COMPLT,R0                  ;CHECK FOR SUCCESSFUL RESPONSE
002260  012275                  BEQ T2
002261                          ERRHRD MS15,#COMPLT,R0
002261  100467                                                          MOV R0,-(SP)
002262  104010  001316                                                  MOV R1,SAVREG
002264  104201  000176                                                  MOV #COMPLT,R1
002266  100461                                                          MOV R1,-(SP)
002267  104301  001316                                                  MOV SAVREG,R1
002271  021236                                                          CALL RERROR     ;ERROR # 18.
002272  005022                                                          .WORD <PRMS*2000>+<2*400>+ERRN
002273  000624                                                          .WORD MS15
002274  003443                  BR      TESTEX
```

```
                                ;ISSUE CHANGE MODE COMMAND TO ENABLE FORMATTING, DIAG CYL ACCESS AND 256 SECT.
                        
002275  104203  004614    T2:     MOV #CR.MOD,R3                  ;POINT TO CHANGE MODE COMMAND
002277                          TALKX   TESTEX,MS16,MS17        ; SDI INTERCHANGE
002277  021102                  CALL    TALKER                  ; INITIATE SDI INTERCHANGE
002300  115003                  TSTB    R3                      ; SEE IF ERROR OCCURRED
002301  012313                  BEQ     12$                     ; IF NOT, BRANCH
002302  032307                  BPL     11$                     ; IF SO, BRANCH
002303                          ERRHRD  MS16;SEND COMMAND ERROR
002303  021236                                                                  CALL RERROR     ;ERROR # 19.
002304  001023                                                                  .WORD <PRMS*2000>+<2*400>+ERRN
002305  000715                                                                  .WORD MS16
002306  003443                  BR      TESTEX
002307                  11$:    ERRHRD  MS17;RECEIVE COMMAND ERROR
002307  021236                                                                  CALL RERROR     ;ERROR # 20.
002310  001024                                                                  .WORD <PRMS*2000>+<2*400>+ERRN
002311  000747                                                                  .WORD MS17
002312  003443                  BR      TESTEX
002313                  12$:
002313  106207  000176            CMPB #COMPLT,R0                 ;CHECK FOR SUCCESSFUL RESPONSE
002315  012332                  BEQ T03
002316                          ERRHRD MS18,#COMPLT,R0
002316  100467                                                                  MOV R0,-(SP)
002317  104010  001316                                                          MOV R1,SAVREG
002321  104201  000176                                                          MOV #COMPLT,R1
002323  100461                                                                  MOV R1,-(SP)
002324  104301  001316                                                          MOV SAVREG,R1
002326  021236                                                                  CALL RERROR     ;ERROR # 21.
002327  005025                                                                  .WORD <PRMS*2000>+<2*400>+ERRN
002330  001004                                                                  .WORD MS18
002331  003443                  BR      TESTEX
```

INITIALIZE DRIVE AND LOOK AT DRIVE SIGNALS

```
                                        ;SPIN UP DRIVE
002332  104203  004633        T03:    MOV     #CR.RUN,R3              ; POINT TO RUN COMMAND
002334                                  TALKX   TESTEX,MS57,MS58        ; SDI INTERCHANGE
002334  021102                         CALL    TALKER                  ; INITIATE SDI INTERCHANGE
002335  115003                         TST     R3                      ; SEE IF ERROR OCCURRED
002336  012350                         BEQ     12$                     ; IF NOT, BRANCH
002337  032344                         BPL     11$                     ; IF SO, BRANCH
002340                                  ERRHRD  MS57;SEND COMMAND ERROR
002340  021236                                                                  CALL RERROR     ;ERROR # 22.
002341  001026                                                                  .WORD <PRMS*2000>+<2*400>+ERRN
002342  003436                                                                  .WORD MS57
002343  003443                         BR      TESTEX
002344                          11$:    ERRHRD  MS58;RECEIVE COMMAND ERROR
002344  021236                                                                  CALL RERROR     ;ERROR # 23.
002345  001027                                                                  .WORD <PRMS*2000>+<2*400>+ERRN
002346  003464                                                                  .WORD MS58
002347  003443                         BR      TESTEX
002350                          12$:
002350  106207  000176                 CMP     #COMPLT,R0              ; SEE IF COMPLETED NORMALLY
002352  012367                         BEQ     T3                      ; IF SO, BRANCH
002353                                  ERRHRD  MS59,#COMPLT,R0         ; REPORT ERROR
002353  100467                                                                  MOV R0,-(SP)
002354  104010  001316                                                         MOV R1,SAVREG
002356  104201  000176                                                         MOV #COMPLT,R1
002360  100461                                                                  MOV R1,-(SP)
002361  104301  001316                                                         MOV SAVREG,R1
002363  021236                                                                  CALL RERROR     ;ERROR # 24.
002364  005030                                                                  .WORD <PRMS*2000>+<2*400>+ERRN
002365  003515                                                                  .WORD MS59
002366  003443                 BR       TESTEX                  ; BRANCH
```

```
                                ;ISSUE INITIATE RECALIBRATE COMMAND
                                
002367  104203  004626    T3:     MOV #CR.INR,R3                  ;POINT TO RECALIBRATE COMMAND
002371                          TALKX   TESTEX,MS19,MS20        ; SDI INTERCHANGE
002371  021102                  CALL    TALKER                  ; INITIATE SDI INTERCHANGE
002372  115003                  TST     R3                      ; SEE IF ERROR OCCURRED
002373  012405                  BEQ     12$                     ; IF NOT, BRANCH
002374  032401                  BPL     11$                     ; IF SO, BRANCH
002375                          ERRHRD  MS19;SEND COMMAND ERROR
002375  021236                                                                  CALL RERROR     ;ERROR # 25.
002376  001031                                                                  .WORD <PRMS*2000>+<2*400>+ERRN
002377  001075                                                                  .WORD MS19
002400  003443                  BR      TESTEX
002401                  11$:    ERRHRD  MS20;RECEIVE COMMAND ERROR
002401  021236                                                                  CALL RERROR     ;ERROR # 26.
002402  001032                                                                  .WORD <PRMS*2000>+<2*400>+ERRN
002403  001127                                                                  .WORD MS20
002404  003443                  BR      TESTEX
002405                  12$:
002405  106207  000176            CMP #COMPLT,R0                  ;CHECK FOR SUCCESSFUL RESPONSE
002407  012424                  BEQ T3A
002410                          ERRHRD MS21,#COMPLT,R0          ;RECALIBRATE COMMAND REJECTED
002410  100467                                                                  MOV R0,-(SP)
002411  104010  001316                                                         MOV R1,SAVREG
002413  104201  000176                                                         MOV #COMPLT,R1
002415  100461                                                                  MOV R1,-(SP)
002416  104301  001316                                                         MOV SAVREG,R1
002420  021236                                                                  CALL RERROR     ;ERROR # 27.
002421  005033                                                                  .WORD <PRMS*2000>+<2*400>+ERRN
002422  001164                                                                  .WORD MS21
002423  003443                  BR      TESTEX
002424  104204  177716    T3A:    MOV #-50.,R4                    ;SET UP 30 SEC CLOCK
002426  104205  144062    T3B:    MOV #-14286.,R5                 ;SET UP 0.5 SEC CLOCK
002430                      T3C:    DSTAT   TESTEX,MS10,MS2         ;GET DRIVE SIGNALS
002430  020720                  CALL    RDSTAT                  ; GET DRIVE STATUS
002431  102201  010000            BIT     #10000,R1               ; SEE IF ANY ERRORS
002433  012447                  BEQ     2$                      ; IF NO ERROR, BRANCH
002434  102201  004000            BIT     #4000,R1                ; SEE IF XMIT ERROR
002436  012443                  BEQ     1$                      ; IF SO, BRANCH
002437                          ERRHRD  MS10                    ; REPORT INVALID STATUS ERROR
002437  021236                                                                  CALL RERROR     ;ERROR # 28.
002440  001034                                                                  .WORD <PRMS*2000>+<2*400>+ERRN
002441  000454                                                                  .WORD MS10
002442  003443                  BR      TESTEX                  ; BRANCH TO DONE
002443                  1$:     ERRHRD  MS2                     ; REPORT XMIT ERROR
002443  021236                                                                  CALL RERROR     ;ERROR # 29.
002444  001035                                                                  .WORD <PRMS*2000>+<2*400>+ERRN
002445  000045                                                                  .WORD MS2
002446  003443                  BR      TESTEX                  ; BRANCH TO DONE
002447                  2$:
002447  102201  000002            BIT #ATTN,R1                    ;DID ATTENTION SET?
002451  012456                  BEQ T3D                         ;  NO
002452                          ERRHRD MS24;ATTENTION SET AFTER RECALIBRATE COMMAND
002452  021236                                                                  CALL RERROR     ;ERROR # 30.
002453  001036                                                                  .WORD <PRMS*2000>+<2*400>+ERRN
002454  001255                                                                  .WORD MS24
002455  003443                  BR      TESTEX
```

INITIALIZE DRIVE AND LOOK AT DRIVE SIGNALS

```
    16
    17 002456  102201  100000          T3D:    BIT #RWRDY,R1                   ;DID R/W READY SET?
    18 002460  052471                          BNE T4A                         ;  YES
    19 002461  115405                          INC R5
    20 002462  052430                          BNE T3C
    21 002463  115404                          INC R4
    22 002464  052426                          BNE T3B
    23 002465                                  ERRHRD MS25;R/W READY DID NOT SET AFTER RECALIBRATE COMMAND
       002465  021236                                                          CALL RERROR     ;ERROR # 31.
       002466  001037                                                          .WORD <PRMS*2000>+<2*400>+ERRN
       002467  001303                                                          .WORD MS25
    24 002470  003443                          BR      TESTEX
```

```
                                ;       GET SUBUNIT CHARACTERISTICS
                                
 002471  104303  004547       T4A:    MOV     UNITNB,R3               ; GET UNIT OFFSET INTO UNITS TABLE
 002473  103203  177774               BIC     #177774,R3              ; R3 HAS SUBUNIT NUMBER (0 - 3)
 002475  104207  000010               MOV     #10,R0                  ; SETUP SUBUNIT MASK
 002477  110207                1$:     ROL     R0                      ; SHIFT MASK
 002500  117403                        DEC     R3                      ; DECREMENT SUBUNIT NUMBER
 002501  032477                        BPL     1$                      ; IF MASK NOT COMPUTED YET, BRANCH
 002502  103207  177417               BIC     #LBHINB,R0              ; CLEAR UNUSED BITS
 002504  104070  004652               MOV     R0,SUBUNT
 002506  104203  004602               MOV #CR.SCR,R3                  ; POINT TO GET SUBUNIT CHARACTERISTICS
 002510                               TALKX   TESTEX,MS29,MS30        ; SDI INTERCHANGE
 002510  021102                       CALL    TALKER                  ; INITIATE SDI INTERCHANGE
 002511  115003                       TST     R3                      ; SEE IF ERROR OCCURRED
 002512  012524                       BEQ     12$                     ; IF NOT, BRANCH
 002513  032520                       BPL     11$                     ; IF SO, BRANCH
 002514                               ERRHRD  MS29;SEND COMMAND ERROR
 002514  021236                                                               CALL RERROR     ;ERROR # 32.
 002515  001040                                                               .WORD <PRMS*2000>+<2*400>+ERRN
 002516  001550                                                               .WORD MS29
 002517  003443                       BR      TESTEX
 002520                         11$:  ERRHRD  MS30;RECEIVE COMMAND ERROR
 002520  021236                                                               CALL RERROR     ;ERROR # 33.
 002521  001041                                                               .WORD <PRMS*2000>+<2*400>+ERRN
 002522  001612                                                               .WORD MS30
 002523  003443                       BR      TESTEX
 002524                         12$:
 002524  106207  000167               CMP #SBCRES,R0                  ; TEST AGAINST RETURN CODE
 002526  012543                       BEQ T31                         ; IF EQUAL, BRANCH
 002527                               ERRHRD MS31,#SBCRES,R0; GET SUBUNIT CHARACTERISTICS FAILED
 002527  100467                                                               MOV R0,-(SP)
 002530  104010  001316                                                       MOV R1,SAVREG
 002532  104201  000167                                                       MOV #SBCRES,R1
 002534  100461                                                               MOV R1,-(SP)
 002535  104301  001316                                                       MOV SAVREG,R1
 002537  021236                                                               CALL RERROR     ;ERROR # 34.
 002540  005042                                                               .WORD <PRMS*2000>+<2*400>+ERRN
 002541  001657                                                               .WORD MS31
 002542  003443                       BR      TESTEX
```

```
                                        ;SEEK TO CYLINDER 0

                                        T31:    CLR R0                          ;LOAD CYLINDER 0, GROUP 0
002543  114007                                  MOV R0,CURCYL
002544  104070  004767                          MOV R0,CURCYL+2
002546  104070  004771                          MOV     SUB+LBNCYL+1,R0         ; GET Hi STARTING CYL NUMBER
002550  104307  004707                          BIC     #HBHINB,R0              ; CLEAR UNUSED BITS
002552  103207  007777                          MOV R0,CURCYL+1                 ; SAVE
002554  104070  004770                          MOV #CURCYL,R1                  ;POINT TO ZERO WORD
002556  104201  004767                          CALL SEEK
002560  024231
```

```
                                        ;       CALCULATE NUMBER OF SECTORS PER TRACK
                                        ;       NO. OF SECTORS/TRACK =(NO. OF LBN'S/TRACK) + (NO. OF RBN'S/TRACK)

002561  104301  004712        T4A1:   MOV SUB+RBNTRK,R1               ; GET RBN'S/TRACK
002563  103201  177600                BIC #177600,R1
002565  104307  004717                MOV SUB+LBNTRK,R0               ; GET LBN'S/TRACK
002567  103207  177400                BIC #HIBYTE,R0                  ; ZERO UPPER BITS
002571  105071                        ADD R0,R1                       ; ADD NUMBER OF LBN'S TO RBN'S
002572  104010  004753                MOV R1,SECTRK                   ; SAVE
002574  105011                        ADD R1,R1                       ; COMPUTE SECTORS TIMES TWO
002575  104010  004733                MOV R1,NSCSL                    ; AS SECTORS READ BEFORE DECLARING
                                                                      ; NO HEADER FOUND

                                ;COMPUTE FIRST CYLINDER IN XBN AREA

002577  104207  004706                MOV #SUB+LBNCYL,R0              ; POINT TO NUMBER OF LBN CYLINDERS
002601  104271                        MOV (R0)+,R1                    ; GET LO ORDER LBN CYL
002602  104010  004751                MOV R1,FXBNCYL                  ; FIRST XBN CYLINDER (LO)
002604  104177                        MOV (R0),R0                     ; GET HI ORDER LBN CYL
002605  104070  004752                MOV R0,FXBNCYL+2                ; FIRST XBN CYLINDER (HI)

                                ;COMPUTE FIRST CYLINDER IN DBN AREA

002607  105301  004727                ADD SUB+XBNCYL,R1               ; ADD NUMBER OF XBN CYLINDERS
002611  042613                        BCC T4B                         ; BRANCH IF NO CARRY
002612  115407                        INC R0                          ; INCREMENT HI ORDER CYL
002613  104010  004747        T4B:    MOV R1,FDIACYL                  ; STORE STARTING DBN CYLINDER
002615  104070  004750                MOV R0,FDIACYL+1

                                ;COMPUTE LAST CYLINDER IN DBN AREA

002617  104303  004730                MOV SUB+DBNCYL,R3               ;GET NUMBER OF DBN CYLINDERS
002621  110703                        SWAB R3
002622  103203  177400                BIC #HIBYTE,R3
002624  052631                        BNE     T4C
002625                                ERRHRD MS32;CHARACTERISTICS SAY LESS THAN 1 DIAGNOSTIC CYLINDER
002625  021236                                                          CALL RERROR     ;ERROR # 35.
002626  001043                                                          .WORD <PRMS*2000>+<2*400>+ERRN
002627  001760                                                          .WORD MS32
002630  003443                        BR      TESTEX
002631  117403        T4C:    DEC R3                                  ;REDUCE BY ONE AND ADD TO
002632  105031                        ADD R3,R1                       ; FIRST DBN CYLINDER TO
002633  042635                        BCC T4D                         ; COMPUTE LAST CYLINDER
002634  115407                        INC R0
002635  104010  004744        T4D:    MOV R1,LDIACYL                  ;STORE LAST DBN CYLINDER
002637  104070  004745                MOV R0,LDIACYL+1
002641  104303  004710                MOV     SUB+GRPCYL,R3           ; GET GROUPS/CYLINDER
002643  103203  177400                BIC     #HIBYTE,R3              ; CLEAR UNUSED BITS
002645  104305  004730                MOV     SUB+DBNCYL,R5           ; GET NUMBER OF RESERVED DBN GROUPS
002647  103205  177400                BIC     #HIBYTE,R5              ; CLEAR UNUSED BITS
002651  107053                        SUB     R5,R3                   ; R3 HAS FIRST RESERVED DBN CYL
002652  104030  004746                MOV R3,LDIACYL+2
002654  104201  004744                MOV #LDIACYL,R1
002656  024231                        CALL SEEK                       ;SEEK TO LAST DBN CYLINDER
```

```
                                        ;READ ALL FACTORY FORMATTED TRACKS.
                                        ;REPORT ERROR IF ALL SECTORS ON A TRACK READ WITH AN ERROR.

                                        ;COMPUTE DBN OF FIRST BLOCK ON FACTORY FORMATTED CYLINDER

002657  114005                  T6:     CLR     R5
002660  104303  004711                  MOV     SUB+TRKGRP,R3           ; TRACKS/GROUP
002662  103203  177400                  BIC     #HIBYTE,R3
002664  104304  004753                  MOV     SECTRK,R4               ; X SECTORS/TRACK
002666  105035                  T6A:    ADD     R3,R5                   ; TO COMPUTE NUMBER OF SECTORS PER GROUP
002667  117404                          DEC     R4
002670  052666                          BNE     T6A
002671  104050  004765                  MOV     R5,SECGRP               ; SAVE SECTORS/GROUP
002673  114001                          CLR     R1                      ; BY
002674                                  PUSH    R2                      ; SAVE R2
002674  100462                                                                  MOV R2,-(SP)
002675  104302  004710                  MOV     SUB+GRPCYL,R2           ; GET GROUPS/CYL
002677  103202  177400                  BIC     #HIBYTE,R2              ; CLEAR UNUSED BITS
002701  104301  004730                  MOV     SUB+DBNCYL,R1           ; GET CYLINDERS IN DBN AREA
002703  110701                          SWAB    R1
002704  103201  177400                  BIC     #HIBYTE,R1              ; TO COMPUTE BLOCK NUMBER OF FIRST BLOCK
002706  114005                          CLR     R5                      ; CLEAR PRODUCT
002707  105025                  1$:     ADD     R2,R5
002710  117401                          DEC     R1
002711  052707                          BNE     1$
002712                                  POP     R2                      ; RESTORE R2
002712  104262                                                                  MOV (SP)+,R2
002713  104307  004730                  MOV     SUB+DBNCYL,R0           ; GET RESERVED GROUPS
002715  103207  177400                  BIC     #HIBYTE,R0              ; CLEAR UNUSED BITS
002717  107075                          SUB     R0,R5                   ; R5 IS NOW NUMBER OF WRITEABLE GROUPS
002720  114007                          CLR     R0
002721  105301  004765          T6C:    ADD     SECGRP,R1
002723  042725                          BCC     T6D
002724  115407                          INC     R0
002725  117405                  T6D:    DEC     R5
002726  052721                          BNE     T6C
002727  104303  004711                  MOV     SUB+HIDBN,R3            ;GET HI DBN BITS
002731  110603                          ROR     R3                      ;SHIFT INTO POSITION
002732  110603                          ROR     R3
002733  110603                          ROR     R3
002734  110603                          ROR     R3
002735  103203  170377                  BIC     #HBLONB,R3              ;CLEAR UNUSED BITS
002737  101037                          BIS     R3,R0                   ;SET HI BITS
002740  101207  140000                  BIS     #HD.DBN,R0              ;SET IN DBN HEADER CODE
002742  104010  004763                  MOV     R1,CURBLK
002744  104070  004764                  MOV     R0,CURBLK+1
002746  104010  004742                  MOV     R1,LCDBN
002750  104070  004743                  MOV     R0,LCDBN+1
002752  104300  004746  004766          MOV     LDIACYL+2,CURGRP
002755  104205  004763                  MOV     #CURBLK,R5
002757  114004                          CLR     R4
002760  104303  004753                  MOV     SECTRK,R3
002762  024364                  T6E:    CALL    READ1                   ;READ EACH SECTOR TILL ONE READS OK
002763  115404                          INC     R4
002764  104301  004711                  MOV     SUB+TRKGRP,R1           ;UNTIL ALL TRACKS READ
002766  103201  177400                  BIC     #HIBYTE,R1
002770  106014                          CMP     R1,R4
```

```
;INITIALIZE DRIVE AND LOOK AT DRIVE SIGNALS

 56 002771  013001                          BEQ     T6E1
 57 002772  105300  004753  004763          ADD     SECTRK,CURBLK
 58 002775  043021                          BCC     T6F
 59 002776  115400  004764                  INC     CURBLK+1
 60 003000  003021                          BR      T6F
 61 003001  115400  004766          T6E1:   INC     CURGRP
 62 003003  104303  004710                  MOV     SUB+GRPCYL,R3
 63 003005  106030  004766                  CMP     R3,CURGRP
 64 003007  013022                          BEQ     T6X
 65 003010  104300  004766  004746          MOV     CURGRP,LDIACYL+2
 66 003013  104201  004744                  MOV     #LDIACYL,R1
 67 003015  024231                          CALL    SEEK
 68 003016  114004                          CLR     R4
 69 003017  104303  004753                  MOV     SECTRK,R3
 70 003021  002762                  T6F:    BR      T6E
 71 003022  104304  004730          T6X:    MOV     SUB+DBNCYL,R4           ; GET NUMBER OF RESERVED GROUPS
 72 003024  103204  177400                  BIC     #HIBYTE,R4
 73 003026  104301  004710                  MOV     SUB+GRPCYL,R1           ; GET GROUPS/CYLINDER
 74 003030  103201  177400                  BIC     #HIBYTE,R1
 75 003032  107041                          SUB     R4,R1                   ; R1 HAS FIRST RESERVED GROUP
 76 003033  104010  004746                  MOV     R1,LDIACYL+2            ; SAVE
```

INITIALIZE DRIVE AND LOOK AT DRIVE SIGNALS

```
                                        ;STARTING WITH CYLINDER 0, GROUP 0 AND INCREMENTING THROUGH EVERY GROUP
                                        ;ON THE DISK, PERFORM A SEEK TO THE SELECTED GROUP, READ A HEADER
                                        ;ON THAT GROUP AND THEN A SEEK TO THE FACTORY FORMATTED DIAGNOSTIC
                                        ;GROUP TO VERIFY HEADS POSITIONED CORRECTLY.

                                        ;COMPUTE LBNS PER GROUP

003035  114001                  T7:     CLR     R1                      ; LBN'S/TRK X TRK/GROUP
003036  104305  004711                  MOV     SUB+TRKGRP,R5
003040  103205  177400                  BIC     #HIBYTE,R5
003042  104303  004717                  MOV     SUB+LBNTRK,R3           ;GET LBN'S PER TRACK
003044  103203  177400                  BIC     #HIBYTE,R3              ;CLEAR UNUSED BITS
003046  105031                  T7A:    ADD     R3,R1
003047  117405                          DEC     R5
003050  053046                          BNE     T7A
003051  104010  004761                  MOV     R1,BLOCKG               ;SAVE IN BLOCKG
003053  104050  004756                  MOV     R5,TSTCYL
003055  104307  004707                  MOV     SUB+HICYL,R0
003057  103207  007777                  BIC     #HBHINB,R0              ;CLEAR UNUSED BITS
003061  104070  004757                  MOV     R0,TSTCYL+1
003063  104050  004760                  MOV     R5,TSTCYL+2             ;GROUP ZERO
003065  104050  004754                  MOV     R5,TSTBLK
003067  104307  004710                  MOV     SUB+HILBN,R0
003071  103207  170377                  BIC     #HBLONB,R0              ;CLEAR UNUSED BITS
003073  104070  004755                  MOV     R0,TSTBLK+1
003075  104307  004717                  MOV     SUB+LBNTRK,R0           ;GET LBN BLOCKS IN EACH TRACK
003077  103207  177400                  BIC     #HIBYTE,R0              ;CLEAR UNUSED BITS
003101  104070  004762                  MOV     R0,BLOCKT
003103  114004                          CLR     R4                      ;READ TRACK 0
003104  104205  004754          T7C:    MOV     #TSTBLK,R5              ;POINT TO LBN NUMBER
003106  104303  004762                  MOV     BLOCKT,R3               ;GET MAXIMUM NUMBER OF SECTOPS TO READ
003110  104201  004756                  MOV     #TSTCYL,R1              ;SEEK TO CYLINDER
003112  024231                          CALL    SEEK
003113  024364                          CALL    READ1                   ;READ UNTIL AT LEAST ONE RECORD READ WITHOUT ERROR
003114  104205  004742                  MOV     #LCDBN,R5               ;POINT TO DBN NUMBER
003116  104303  004753                  MOV     SECTRK,R3               ;GET MAXIMUM NUMBER OF SECTORS TO READ
003120  104201  004744                  MOV     #LDIACYL,R1             ;SEEK TO DIAGNOSTIC CYLINDER
003122  024231                          CALL    SEEK
003123  024364                          CALL    READ1                   ;READ UNTIL AT LEAST ONE RECORD READ WITHOUT ERROR
003124  104307  004760                  MOV     TSTCYL+2,R0
003126  115407                          INC     R0
003127  104301  004710                  MOV     SUB+GRPCYL,R1
003131  103201  177400                  BIC     #HIBYTE,R1
003133  106071                          CMP     R0,R1
003134  013140                          BEQ     T7C1
003135  104070  004760                  MOV     R0,TSTCYL+2
003137  003222                          BR      T7G
003140  104307  004756          T7C1:   MOV     TSTCYL,R0               ;GET CURRENT CYLINDER
003142  115407                          INC     R0                      ;INCREMENT TO NEXT
003143  104070  004756                  MOV     R0,TSTCYL               ;SAVE NEW CYLINDER
003145  104301  004757                  MOV     TSTCYL+1,R1
003147  043153                          BCC     T7D
003150  115401                          INC     R1
003151  104010  004757                  MOV     R1,TSTCYL+1
003153  114005                  T7D:    CLR     R5
003154  104050  004760                  MOV     R5,TSTCYL+2
003156  106307  004751                  CMP     FXBNCYL,R0              ;CHECK IF INCREMENTED TO XBN AREA
```

```
 58 003160  053214                        BNE     T7E
 59 003161  106301  004752                CMP     FXBNCYL+1,R1
 60 003163  053214                        BNE     T7E
 61 003164  114007                        CLR     R0
 62 003165  104070  004754                MOV     R0,TSTBLK               ; TO FIRST XBN
 63 003167  104307  004710                MOV     SUB+HIXBN,R0
 64 003171  110607                        ROR     R0                      ; ROTATE TO CORRECT BIT POSITION
 65 003172  110607                        ROR     R0
 66 003173  110607                        ROR     R0
 67 003174  110607                        ROR     R0
 68 003175  103207  170377                BIC     #HBLONB,R0              ;STRIP OFF UNUSED BITS
 69 003177  101207  120000                BIS     #HD.XBN,R0
 70 003201  104070  004755                MOV     R0,TSTBLK+1
 71 003203  104307  004753                MOV     SECTRK,R0               ;CHANGE BLOCK COUNT PER TRACK
 72 003205  104070  004762                MOV     R0,BLOCKT
 73 003207  104307  004765                MOV     SECGRP,R0               ;CHANGE BLOCK COUNT PER CYLINDER
 74 003211  104070  004761                MOV     R0,BLOCKG
 75 003213  003104                        BR      T7C                     ;NOW GO SEEK TO THIS XBN
 76 003214  106307  004747        T7E:    CMP     FDIACYL,R0              ;CHECK IF INCREMENTED TO DBN AREA
 77 003216  053222                        BNE     T7G
 78 003217  106301  004750                CMP     FDIACYL+1,R1
 79 003221  013237                        BEQ     T7X
 80 003222  104307  004754        T7G:    MOV     TSTBLK,R0               ;GET BLOCK NUMBER
 81 003224  105307  004761                ADD     BLOCKG,R0               ;ADD BLOCKS PER GROUP
 82 003226  104070  004754                MOV     R0,TSTBLK               ;SAVE NEW TEST BLOCK NUMBER
 83 003230  043236                        BCC     T7H
 84 003231  104307  004755                MOV     TSTBLK+1,R0
 85 003233  115407                        INC     R0
 86 003234  104070  004755                MOV     R0,TSTBLK+1
 87 003236  003104                T7H:    BR      T7C                     ;NOW TEST NEXT CYLINDER
 88 003237                        T7X:
```

```
                                        T8:
                                        ;
                                        ;       FORMAT A TRACK, THEN DO SEVERAL WRITES, READS AND DATA COMPARES
                                        ;       UNTIL ONE SECTOR PASSES.  DO THIS FOR EACH TRACK IN EACH GROUP
                                        ;       ON EACH CYLINDER IN THE DIAGNOSTIC AREA EXCEPT FOR THE LAST
                                        ;       (FACTORY FORMATTED) CYLINDER ON THE PACK
                                        ;
 1 003237                               
 2                                      
 3                                      
 4                                      
 5                                      
 6                                      
 7                                      
 8 003237  104201  004756               
 9 003241  024231                       
10 003242  114004                       
11 003243  104040  004754               
12 003245  104307  004711               
13 003247  110207                       
14 003250  110207                       
15 003251  110207                       
16 003252  110207                       
17 003253  103207  170377               
18 003255  101207  140000               
19 003257  104070  004755               
20 003261  104040  004772               T8A:    MOV     R4,CURTRK               ; SAVE TRACK NUMBER
21 003263  023444                               CALL    FORTRK                  ; FORMAT THE TRACK
22 003264  023563                               CALL    TRKTST                  ; TEST THE TRACK
23 003265  104307  004754                       MOV     TSTBLK,R0               ; GET LO TRACK'S STARTING BLOCK NUMBER
24 003267  105307  004753                       ADD     SECTRK,R0               ; ADD SECTORS/TRACK
25 003271  104070  004754                       MOV     R0,TSTBLK               ; SAVE
26 003273  043301                               BCC     T8B                     ; IF NO CARRY, BRANCH
27 003274  104307  004755                       MOV     TSTBLK+1,R0             ; GET HI ORDER
28 003276  115407                               INC     R0                      ; INCREMENT
29 003277  104070  004755                       MOV     R0,TSTBLK+1             ; SAVE
30 003301  104304  004772               T8B:    MOV     CURTRK,R4               ; GET TRACK NUMBER
31 003303  115404                               INC     R4                      ; INCREMENT TRACK NUMBER
32 003304  104303  004711                       MOV     SUB+TRKGRP,R3           ; GET NUMBER OF TRACKS/GROUP
33 003306  103203  177400                       BIC     #HIBYTE,R3              ; CLEAR UNUSED BITS
34 003310  106034                               CMP     R3,R4                   ; SEE IF ALL TRACKS READ
35 003311  053261                               BNE     T8A                     ; IF NOT, BRANCH
36 003312  114004                               CLR     R4                      ; ZERO R4
37 003313  115400  004760                       INC     TSTCYL+2                ; MOVE TO NEXT GROUP
38 003315  104301  004710                       MOV     SUB+GRPCYL,R1           ; GET GROUPS/CYLINDER
39 003317  103201  177400                       BIC     #HIBYTE,R1              ; CLEAR UNUSED BITS
40 003321  106010  004760                       CMP     R1,TSTCYL+2             ; COMPARE
41 003323  053333                               BNE     T8D                     ; IF ALL GROUPS NOT TESTED, BRANCH
42 003324  104040  004760                       MOV     R4,TSTCYL+2             ; START WITH GROUP 0 ON NEW CYLINDER
43 003326  115400  004756                       INC     TSTCYL                  ; INCREMENT CYLINDER
44 003330  043333                               BCC     T8D                     ; IF NO CARRY, BRANCH
45 003331  115400  004757                       INC     TSTCYL+1                ; INCREMENT HI CYLINDER
46 003333  106300  004756  004744       T8D:    CMP     TSTCYL,LDIACYL          ; SEE IF ON LAST CYLINDER (LO ORDER)
47 003336  053347                               BNE     1$                      ; IF NOT, BRANCH
48 003337  106300  004757  004745               CMP     TSTCYL+1,LDIACYL+1      ; SEE IF ON LAST CYLINDER (HI ORDER)
49 003342  053347                               BNE     1$                      ; IF NOT, BRANCH
50 003343  106300  004760  004746               CMP     TSTCYL+2,LDIACYL+2      ; SEE IF ON RESERVED GROUPS
51 003346  013353                               BEQ     T8X                     ; IF SO, END THIS PHASE OF TEST
52 003347  104201  004756               1$:     MOV     #TSTCYL,R1              ; POINT TO NEW CYLINDER OR GROUP
53 003351  024231                               CALL    SEEK                    ; ISSUE SEEK
54 003352  003261                               BR      T8A                     ; BRANCH
55 003353                              T8X:
```

INITIALIZE DRIVE AND LOOK AT DRIVE SIGNALS

```
1 003353  104203  004570          T9:     MOV     #CR.DIS,R3              ; POINT TO DISCONNECT COMMAND
2 003355                                  TALKX   TESTEX,MS52,MS53        ; SDI INTERCHANGE
  003355  021102                          CALL    TALKER                  ; INITIATE SDI INTERCHANGE
  003356  115003                          TST     R3                      ; SEE IF ERROR OCCURRED
  003357  013371                          BEQ     12$                     ; IF NOT, BRANCH
  003360  033365                          BPL     11$                     ; IF SO, BRANCH
  003361                                  ERRHRD  MS52;SEND COMMAND ERROR
  003361  021236                                                          CALL RERROR     ;ERROR # 36.
  003362  001044                                                          .WORD <PRMS*2000>+<2*400>+ERRN
  003363  003124                                                          .WORD MS52
  003364  003443                          BR      TESTEX
  003365                          11$:    ERRHRD  MS53;RECEIVE COMMAND ERROR
  003365  021236                                                          CALL RERROR     ;ERROR # 37.
  003366  001045                                                          .WORD <PRMS*2000>+<2*400>+ERRN
  003367  003156                                                          .WORD MS53
  003370  003443                          BR      TESTEX
  003371                          12$:
3 003371  106207  000176                  CMP     #COMPLT,R0              ; CHECK FOR SUCESSFUL RESPONSE
4 003373  013410                          BEQ     T9X                     ; IF SO, BRANCH
5 003374                                  ERRHRD  MS54,#COMPLT,R0         ; IF NOT, REPORT ERROR
  003374  100467                                                          MOV R0,-(SP)
  003375  104010  001316                                                  MOV R1,SAVREG
  003377  104201  000176                                                  MOV #COMPLT,R1
  003401  100461                                                          MOV R1,-(SP)
  003402  104301  001316                                                  MOV SAVREG,R1
  003404  021236                                                          CALL RERROR     ;ERROR # 38.
  003405  005046                                                          .WORD <PRMS*2000>+<2*400>+ERRN
  003406  003213                                                          .WORD MS54
6 003407  003443                          BR      TESTEX                  ; BRANCH TO END OF TEST
7 003410                          T9X:
```

```
      1 003410  114005                  T10:    CLR     R5              ; SET UP TIMEOUT COUNTER
      2 003411                          T10LOP: DSTAT   TESTEX,MS1,MS2  ; GET REAL TIME DRIVE STATE
        003411  020720                          CALL    RDSTAT          ; GET DRIVE STATUS
        003412  102201  010000                  BIT     #10000,R1       ; SEE IF ANY ERRORS
        003414  013430                          BEQ     2$              ; IF NO ERROR, BRANCH
        003415  102201  004000                  BIT     #4000,R1        ; SEE IF XMIT ERROR
        003417  013424                          BEQ     1$              ; IF SO, BRANCH
        003420                                  ERRHRD  MS1             ; REPORT INVALID STATUS ERROR
                                                                                CALL RERROR     ;ERROR # 39.
                                                                                .WORD <PRMS*2000>+<2*400>+ERRN
                                                                                .WORD MS1
        003420  021236
        003421  001047
        003422  000000
        003423  003443                          BR      TESTEX          ; BRANCH TO DONE
        003424                          1$:     ERRHRD  MS2             ; REPORT XMIT ERROR
                                                                                CALL RERROR     ;ERROR # 40.
                                                                                .WORD <PRMS*2000>+<2*400>+ERRN
                                                                                .WORD MS2
        003424  021236
        003425  001050
        003426  000045
        003427  003443                          BR      TESTEX          ; BRANCH TO DONE
        003430                          2$:
      3 003430  102201  000100                  BIT     #AVAIL,R1       ; SEE IF AVAILABLE IS ASSERTED
      4 003432  053443                          BNE     T10X            ; IF SO, BRANCH
      5 003433  117405                          DEC     R5              ; DECREMENT TIMEOUT COUNT
      6 003434  053411                          BNE     T10LOP          ; IF UNEXPIRED, BRANCH
      7 003435  103201  077674                  BIC     #077674,R1      ; CLEAR UNUSED BITS
      8 003437                                  ERRHRD  MS56,R1         ; REPORT ERROR
                                                                                MOV R1,-(SP)
        003437  100461                                                  CALL RERROR     ;ERROR # 41.
        003440  021236                                                  .WORD <PRMS*2000>+<2*400>+ERRN
        003441  003051                                                  .WORD MS56
        003442  003372
      9 003443                          T10X:
```

```
      1 003443  001764                    TESTEX: BR  TESTX
```

```
                003444                          FORTRK:
                                                ;
                                                ;       FORMAT TRACK GIVEN BY CURTRK, STARTING WITH DBN GIVEN IN TSTBLK
                                                ;       THE DATA WRITTEN IS UNPREDICTABLE
                                                ;
003444  104307  004754                          MOV     TSTBLK,R0               ; GET LO STARTING DBN
003446  104301  004755                          MOV     TSTBLK+1,R1             ; GET HI STARTING DBN
003450  104203  005433                          MOV     #OBUFF,R3               ; POINT TO OUTPUT BUFFER
003452  104304  004753                          MOV     SECTRK,R4               ; R4 CONTAINS NUMBER OF SECTORS TO FORMAT
003454  104205  006034                          MOV     #FCHAIN,R5              ; R5 POINTS TO FORMAT CHAIN
003456  100253                          FLOOP:  MOV     R3,(R5)+                ; MOVE POINTER TO BUFFER TO CHAIN
003457  100257                                  MOV     R0,(R5)+                ; MOVE LO DBN TO CHAIN
003460  100251                                  MOV     R1,(R5)+                ; MOVE HI DBN TO CHAIN
003461  115407                                  INC     R0                      ; INCREMENT LO DBN
003462  043464                                  BCC     FCARY                   ; IF NO CARRY, BRANCH
003463  115401                                  INC     R1                      ; INCREMENT HI DBN
003464  117404                          FCARY:  DEC     R4                      ; DECREMENT SECTOR COUNT
003465  053456                                  BNE     FLOOP                   ; BRANCH IF COUNT UNEXHAUSTED
003466  104207  100000                          MOV     #FSTOP,R0               ; GET FORMAT END-OF-CHAIN FLAG
003470  100157                                  MOV     R0,(R5)                 ; MOVE INTO CHAIN
003471                                          DSTAT   TESTEX,MS10,MS2         ; GET DRIVE STATE
003471  020720                                  CALL    RDSTAT                  ; GET DRIVE STATUS
003472  102201  010000                          BIT     #10000,R1               ; SEE IF ANY ERRORS
003474  013510                                  BEQ     2$                      ; IF NO ERROR, BRANCH
003475  102201  004000                          BIT     #4000,R1                ; SEE IF XMIT ERROR
003477  013504                                  BEQ     1$                      ; IF SO, BRANCH
003500                                          ERRHRD  MS10                    ; REPORT INVALID STATUS ERROR
003500  021236                                                                          CALL RERROR     ;ERROR # 42.
003501  001052                                                                          .WORD <PRMS*2000>+<2*400>+ERRN
003502  000454                                                                          .WORD MS10
003503  003443                                  BR      TESTEX                  ; BRANCH TO DONE
003504                          1$:             ERRHRD  MS2                     ; REPORT XMIT ERROR
003504  021236                                                                          CALL RERROR     ;ERROR # 43.
003505  001053                                                                          .WORD <PRMS*2000>+<2*400>+ERRN
003506  000045                                                                          .WORD MS2
003507  003443                                  BR      TESTEX                  ; BRANCH TO DONE
003510                          2$:
003510  102201  100000                          BIT     #RWRDY,R1               ; TEST R/W READY
003512  053517                                  BNE     FGO                     ; IF ASSERTED, BRANCH
003513                                          ERRHRD  MS35;READ/WRITE READY DROPPED BEFORE FORMAT
003513  021236                                                                          CALL RERROR     ;ERROR # 44.
003514  001054                                                                          .WORD <PRMS*2000>+<2*400>+ERRN
003515  002020                                                                          .WORD MS35
003516  003443                                  BR      TESTEX                  ; BRANCH TO EXIT
003517  104207  006034                  FGO:    MOV     #FCHAIN,R0              ; POINT TO FORMAT CHAIN (FOR XFC)
003521  104304  004713                          MOV     SUB+DATPRE,R4           ; GET DATA PREAMBLE LENGTHS
003523  103204  177400                          BIC     #HIBYTE,R4              ; CLEAR UNUSED BITS
003525  104303  004713                          MOV     SUB+HDRPRE,R3           ; GET HEADER PREAMBLE LENGTH
003527  110703                                  SWAB    R3                      ; SWAP BYTES
003530  103203  177400                          BIC     #HIBYTE,R3              ; CLEAR UNUSED BYTES
003532  104301  004772                          MOV     CURTRK,R1               ; TRACK NUMBER
003534  060001                                  XFC     FORMAT                  ; FORMAT THE TRACK (BUFFER CONTENTS ARE A DON'T CARE
003535  115001                                  TST     R1                      ; SEE IF ANY ERRORS OCCURRED
003536  013562                                  BEQ     FOREXT                  ; IF NOT, BRANCH
003537                                          ERRHRD  MS36,INS+1,INS+2,INS+3,CURTRK;TIMEOUT OF DRIVE OR READ/WRITE READY DROPPED D
003537  104010  001316                                                                  MOV R1,SAVREG
003541  104301  004772                                                                  MOV CURTRK,R1
```

```
        003543  100461                          MOV R1,-(SP)
        003544  104301  004661                  MOV INS+3,R1
        003546  10046[                          MOV R1,-(SP)
        003547  10430[  004660                  MOV INS+2,R1
        003551  100461                          MOV R1,-(SP)
        003552  104301  004657                  MOV INS+1,R1
        003554  100461                          MOV R1,-(SP)
        003555  104301  001316                  MOV SAVREG,R1
        003557  021236                          CALL RERROR     ;ERROR # 45.
        003560  011055                          .WORD <PRMS*2000>+<2*400>+ERRN
        003561  002052                          .WORD MS36
     37 003562  000000          FOREXT: RETURN
```

INITIALIZE DRIVE AND LOOK AT DRIVE SIGNALS

```
 1 003563                          TRKTST:
 2                                 ;
 3                                 ;       TEST THE ENTIRE TRACK UNTIL AT LEAST ONE BLOCK IS SUCCESSFULLY
 4                                 ;       WRITTEN AND READ WITH SEVERAL DATA PATTERNS
 5                                 ;
 6 003563  104307  004754                 MOV     TSTBLK,R0               ; MOVE LO STARTING DBN TO CURRENT DBN
 7 003565  104070  004763                 MOV     R0,CURBLK
 8 003567  104307  004755                 MOV     TSTBLK+1,R0             ; MOVE HI STARTING DBN TO CURRENT DBN
 9 003571  104070  004764                 MOV     R0,CURBLK+1
10 003573  104301  004753                 MOV     SECTRK,R1               ; GET SECTORS/TRACK
11 003575  117401                         DEC     R1                      ; ADJUST FOR END LOOP WHEN NEGATIVE
12 003576  104010  004774       TRKLOP:   MOV     R1,SECCNT               ; SAVE SECTORS LEFT IN SECTOR COUNT
13 003600  104201  000001                 MOV     #1,R1                   ; MOVE 1 TO CURRENT PATTERN
14 003602  023625                         CALL    WTRCMP                  ; WRITE THAN READ AND COMPARE THE SECTOR
15 003603  115007                         TST     R0                      ; SEE IF WRITES, READS AND COMPARES WERE GOOD
16 003604  013624                         BEQ     TRKEXT                  ; IF SO, BRANCH
17 003605  104307  004763                 MOV     CURBLK,R0               ; GET LO CURRENT BLOCK NUMBER
18 003607  115407                         INC     R0                      ; INCREMENT DBN NUMBER
19 003610  104070  004763                 MOV     R0,CURBLK               ; SAVE
20 003612  043620                         BCC     TRKBOT                  ; BRANCH IF NO CARRY
21 003613  104307  004764                 MOV     CURBLK+1,R0             ; GET HI DBN NUMBER
22 003615  115407                         INC     R0                      ; INCREMENT
23 003616  104070  004764                 MOV     R0,CURBLK+1             ; SAVE
24 003620  104301  004774       TRKBOT:   MOV     SECCNT,R1               ; GET SECTOR COUNT
25 003622  117401                         DEC     R1                      ; DECREMENT COUNT
26 003623  033576                         BPL     TRKLOP                  ; BRANCH IF COUNT POSITIVE
27 003624  000000             TRKEXT:   RETURN
```

INITIALIZE DRIVE AND LOOK AT DRIVE SIGNALS

```
 1 003625                          WTRCMP:
 2                                 ;
 3                                 ;       WRITE A DATA PATTERN TO A SECTOR, READ IT BACK, THEN DO A DATA
 4                                 ;       COMPARE ON THE DATA.  DO THIS AS MANY TIMES AS THERE IS PATTERNS.
 5                                 ;       CURRENT PATTERN NUMBER IN 'CURPAT'
 6                                 ;
 7 003625  104010  004773                MOV     R1,CURPAT               ; R1 IS CURRENT PATTERN NUMBER
 8 003627  105201  006321                ADD     #PATPTR,R1              ; R1 NOW POINTS TO PATTERN TO GENERATE
 9 003631  023652                        CALL    GENPAT                  ; GENERATE THE PATTERN IN THE OUTPUT BUFFER
10 003632  023673                        CALL    WRITEB                  ; WRITE THE PATTERN
11 003633  115007                        TST     R0                      ; SEE IF ANY ERROR OCCURRED
12 003634  053651                        BNE     WTREXT                  ; IF SO, BRANCH
13 003635  024027                        CALL    READB                   ; READ THE BLOCK BACK
14 003636  115007                        TST     R0                      ; SEE IF ANY ERRORS OCCURRED
15 003637  053651                        BNE     WTREXT                  ; IF SO, BRANCH
16 003640  024147                        CALL    CMPDAT                  ; COMPARE THE DATA
17 003641  115007                        TST     R0                      ; SEE IF ANY ERRORS OCCURRED
18 003642  053651                        BNE     WTREXT                  ; IF SO, BRANCH
19 003643  104301  004773                MOV     CURPAT,R1               ; GET CURRENT PATTERN
20 003645  115401                        INC     R1                      ; NEXT PATTERN
21 003646  106301  006321                CMP     PATPTR,R1               ; COMARE AGAINST MAXIMUM PATTERN NUMBER
22 003650  033625                        BPL     WTRCMP                  ; IF LESS OR EQUAL, LOOP
23 003651  000000              WTREXT: RETURN
```

INITIALIZE DRIVE AND LOOK AT DRIVE SIGNALS

```
                                GENPAT:
                                ;
                                ;       GENERATE THE DATA PATTERN IN THE OUTPUT BUFFER
                                ;
                                ;       R1 POINTS TO POINTER TO PATTERN BUFFER WHICH IS PATTERN LENGTH
                                ;       (1 WORD) FOLLOWED BY THAT MANY WORDS OF PATTERN
                                ;
 003652                         ;
 003652  100462                         PUSH    R2                      ; SAVE R2
                                                                                MOV R2,-(SP)
 003653  104117                         MOV     (R1),R0                 ; R0 POINTS TO START OF PATTERN BUFFER
 003654  104203  000400                 MOV     #256.,R3                ; R3 HOLDS PATTERN COUNT
 003656  104204  005433                 MOV     #OBUFF,R4               ; R4 POINTS TO OUTPUT BUFFER
 003660  104071                 GENOUT: MOV     R0,R1                   ; R1 POINTS TO START OF PATTERN BUFFER
 003661  104215                         MOV     (R1)+,R5                ; R5 CONTAINS LENGTH OF PATTERN
 003662  104212                 GENIN:  MOV     (R1)+,R2                ; GET WORD OF PATTERN
 003663  100242                         MOV     R2,(R4)+                ; MOVE TO BUFFER
 003664  117403                         DEC     R3                      ; DECREMENT BUFFER COUNT
 003665  013671                         BEQ     GENEXT                  ; IF BUFFER FULL, EXIT
 003666  117405                         DEC     R5                      ; DECREMENT PATTERN COUNT
 003667  053662                         BNE     GENIN                   ; IF NON-ZERO, BRANCH
 003670  003660                         BR      GENOUT                  ; START PATTERN OVER AGAIN
 003671                         GENEXT: POP     R2                      ; RESTORE R2
 003671  104262                                                                MOV (SP)+,R2
 003672  000000                         RETURN
```

```
 1 003673                                WRITEB:
 2                                       ;
 3                                       ;       WILL WRITE ONE SECTOR OF DATA TO THE DISK
 4                                       ;
 5 003673  104207  140000                        MOV     #WSTOP,R0
 6 003675  104070  005425                        MOV     R0,RW.STAT+CHAIN; MOVE END-OF-CHAIN TO CHAIN
 7 003677  104307  004763                        MOV     CURBLK,R0               ; MOVE LO DBN TO CHAIN
 8 003701  104070  005427                        MOV     R0,RW.LOW+CHAIN
 9 003703  104307  004764                        MOV     CURBLK+1,R0             ; MOVE HI DBN TO CHAIN
10 003705  104070  005430                        MOV     R0,RW.HI+CHAIN
11 003707  104307  004772                        MOV     CURTRK,R0               ; MOVE TRACK + RTC TO CHAIN
12 003711  101207  122400                        BIS     #WREAL,R0               ; SET REAL TIME COMMAND WRITTE
13 003713  104070  005431                        MOV     R0,RW.CMD+CHAIN
14 003715  104207  005433                        MOV     #OBUFF,R0               ; MOVE OUTPUT BUFFER TO CHAIN
15 003717  104070  005426                        MOV     R0,RW.BUF+CHAIN
16 003721                                       DSTAT   TESTEX,MS10,MS2         ; GET DRIVE STATE
   003721  020720                               CALL    RDSTAT                  ; GET DRIVE STATUS
   003722  102201  010000                       BIT     #10000,R1               ; SEE IF ANY ERRORS
   003724  013740                               BEQ     2$                      ; IF NO ERROR, BRANCH
   003725  102201  004000                       BIT     #4000,R1                ; SEE IF XMIT ERROR
   003727  013734                               BEQ     1$                      ; IF SO, BRANCH
   003730                                       ERRHRD  MS10                    ; REPORT INVALID STATUS ERROR
   003730  021236                                                                      CALL RERROR     ;ERROR # 46.
   003731  001056                                                                      .WORD <PRMS*2000>+<2*400>+ERRN
   003732  000454                                                                      .WORD MS10
   003733  003443                               BR      TESTEX                  ; BRANCH TO DONE
   003734                               1$:     ERRHRD  MS2                     ; REPORT XMIT ERROR
   003734  021236                                                                      CALL RERROR     ;ERROR # 47.
   003735  001057                                                                      .WORD <PRMS*2000>+<2*400>+ERRN
   003736  000045                                                                      .WORD MS2
   003737  003443                               BR      TESTEX                  ; BRANCH TO DONE
   003740                               2$:
17 003740  102201  100000                       BIT     #RWRDY,R1               ; SEE IF READ/WRITE READY IS ASSERTED
18 003742  053754                               BNE     WGO                     ; IF SO, BRANCH
19 003743  104307  004774                       MOV     SECCNT,R0               ; GET SECTOR COUNT
20 003745  054026                               BNE     WRTEXT                  ; IF NONZERO, DO NOT REPORT ERROR
21 003746                                       ERRHRD  MS39;READ/WRITE DROPPED READY BEFORE WRITE
   003746  021236                                                                      CALL RERROR     ;ERROR # 48.
   003747  001060                                                                      .WORD <PRMS*2000>+<2*400>+ERRN
   003750  002131                                                                      .WORD MS39
22 003751  104207  000001                       MOV     #1,R0                   ; FLAG ERROR
23 003753  004026                               BR      WRTEXT                  ; BRANCH
24 003754  060012                       WGO:    XFC     WAITSI                  ; WAIT FOR SECTOR OR INDEX PULSE
25 003755  104207  005425                       MOV     #CHAIN,R0               ; POINT TO WRITE CHAIN
26 003757  104304  004713                       MOV     SUB+DATPRE,R4           ; MOVE DATA PREAMBLE LENGTH TO R4
27 003761  103204  177400                       BIC     #HIBYTE,R4              ; CLEAR UNUSED BITS
28 003763  060003                               XFC     XWRITE                  ; WRITE THE BLOCK
29 003764  114007                               CLR     R0                      ; FLAG AS NO ERRORS
30 003765  115001                               TST     R1                      ; SEE IF AN ERROR OCCURRED
31 003766  014026                               BEQ     WRTEXT                  ; IF NOT, BRANCH
32 003767  104307  004774                       MOV     SECCNT,R0               ; SEE IF ERROR SHOULD BE REPORTED
33 003771  054026                               BNE     WRTEXT                  ; IF NOT, BRANCH
34 003772                                       ERRHRD  MS40,R1,CURBLK,CURBLK+1,INS+1,INS+2,INS+3,CURTRK;ERROR DURING WRITE
   003772  104010. 001316                                                              MOV R1,SAVREG
   003774  104301  004772                                                              MOV CURTRK,R1
   003776  100461                                                                      MOV R1,-(SP)
   003777  104301  004661                                                              MOV INS+3,R1
```

```
004001  100461                                          MOV R1,-(SP)
004002  104301  004660                                  MOV INS+2,R1
004004  100461                                          MOV R1,-(SP)
004005  104301  004657                                  MOV INS+1,R1
004007  100461                                          MOV R1,-(SP)
004010  104301  004764                                  MOV CURBLK+1,R1
004012  100461                                          MOV R1,-(SP)
004013  104301  004763                                  MOV CURBLK,R1
004015  100461                                          MOV R1,-(SP)
004016  104301  001316                                  MOV SAVREG,R1
004020  100461                                          MOV R1,-(SP)
004021  021236                                          CALL RERROR     ;ERROR # 49.
004022  017061                                          .WORD <PRMS*2000>+<2*400>+ERRN
004023  002163                                          .WORD MS40
 35 004024  104207  000001          MOV     #1,R0      ; MAKE R0 NONZERO TO REPORT ERROR
 36 004026  0000C0          WRTEXT: RETURN
```

```
                                READB:
                                ;
                                ;       READ ONE SECTOR FROM DISK
                                ;
004027                          
                                
                                
                                
004027  104207  100000         MOV     #RSTOP,R0               ; MOVE END-OF-CHAIN TO CHAIN
004031  104070  005425         MOV     R0,RW.STAT+CHAIN
004033  104307  004772         MOV     CURTRK,R0               ; MOVE TRACK NUMBER AND RTC TO CHAIN
004035  101207  013400         BIS     #RREAL,R0               ; SET REAL TIME READ COMMAND
004037  104070  005431         MOV     R0,RW.CMD+CHAIN
004041  104207  005003         MOV     #RBUFD,R0               ; MOVE POINTER TO INPUT BUFFER INTO CHAIN
004043  104070  005426         MOV     R0,RW.BUF+CHAIN
004045                          DSTAT   TESTEX,MS10,MS2         ; GET DRIVE STATE
004045  020720                 CALL    RDSTAT                  ; GET DRIVE STATUS
004046  102201  010000         BIT     #10000,R1               ; SEE IF ANY ERRORS
004050  014064                 BEQ     2$                      ; IF NO ERROR, BRANCH
004051  102201  004000         BIT     #4000,R1                ; SEE IF XMIT ERROR
004053  014060                 BEQ     1$                      ; IF SO, BRANCH
004054                          ERRHRD  MS10                    ; REPORT INVALID STATUS ERROR
004054  021236                                                          CALL RERROR     ;ERROR # 50.
004055  001062                                                          .WORD <PRMS*2000>+<2*400>+ERRN
004056  000454                                                          .WORD MS10
004057  003443                 BR      TESTEX                  ; BRANCH TO DONE
004060                  1$:     ERRHRD  MS2                     ; REPORT XMIT ERROR
004060  021236                                                          CALL RERROR     ;ERROR # 51.
004061  001063                                                          .WORD <PRMS*2000>+<2*400>+ERRN
004062  000045                                                          .WORD MS2
004063  003443                 BR      TESTEX                  ; BRANCH TO DONE
004064                  2$:
004064  102201  100000         BIT     #RWRDY,R1               ; SEE IF READ/WRITE READY IS ASSERTED
004066  054100                 BNE     RGO                     ; IF SO, BRANCH
004067  104307  004774         MOV     SECCNT,R0               ; GET SECTOR COUNT
004071  054146                 BNE     REDEXT                  ; IF NONZERO, DO NOT REPORT ERROR
004072                          ERRHRD  MS43;READ/WRITE DROPPED READY BEFORE READ
004072  021236                                                          CALL RERROR     ;ERROR # 52.
004073  001064                                                          .WORD <PRMS*2000>+<2*400>+ERRN
004074  002320                                                          .WORD MS43
004075  104207  000001         MOV     #1,R0                   ; FLAG ERROR
004077  004146                 BR      REDEXT                  ; BRANCH
004100  060012          RGO:   XFC     WAITSI                  ; WAIT FOR SECTOR OR INDEX PULSE
004101  104207  005425         MOV     #CHAIN,R0               ; POINT TO READ CHAIN
004103  060002                 XFC     XREAD                   ; READ THE SECTOR
004104  114007                 CLR     R0                      ; FLAG AS NO ERRORS
004105  115001                 TST     R1                      ; SEE IF ERROR OCCURRED
004106  014146                 BEQ     REDEXT                  ; IF NOT, BRANCH
004107  104307  004774         MOV     SECCNT,R0               ; SEE IF ERROR SHOULD BE REPORTED
004111  054146                 BNE     REDEXT                  ; IF NOT, BRANCH
004112                          ERRHRD  MS44,R1,CURBLK,CURBLK+1,INS+1,INS+2,INS+3,CURTRK;ERROR DURING READ
004112  104010  001316                                                  MOV R1,SAVREG
004114  104301  004772                                                  MOV CURTRK,R1
004116  100461                                                          MOV R1,-(SP)
004117  104301  004661                                                  MOV INS+3,R1
004121  100461                                                          MOV R1,-(SP)
004122  104301  004660                                                  MOV INS+2,R1
004124  100461                                                          MOV R1,-(SP)
004125  104301  004657                                                  MOV INS+1,R1
004127  100461                                                          MOV R1,-(SP)
004130  104301  004764                                                  MOV CURBLK+1,R1
```

INITIALIZE DRIVE AND LOOK AT DRIVE SIGNALS

```
        004132  100461                                                  MOV R1,-(SP)
        004133  104301  004763                                         MOV CURBLK,R1
        004135  100461                                                  MOV R1,-(SP)
        004136  104301  001316                                         MOV SAVREG,R1
        004140  100461                                                  MOV R1,-(SP)
        004141  021236                                                 CALL RERROR     ;ERROR # 53.
        004142  017065                                                 .WORD <PRMS*2000>+<2*400>+ERRN
        004143  002351                                                 .WORD MS44
     29 004144  104207  000001                MOV     #1,R0            ; FLAG ERROR
     30 004146  000000                REDEXT: RETURN
```

```
;INITIALIZE DRIVE AND LOOK AT DRIVE SIGNALS

  1 004147                                CMPDAT:
  2                                       ;
  3                                       ;       COMPARE THE DATA IN 'OBUFF' (OUTPUT BUFFER) WITH 'RBUFD'
  4                                       ;       (INPUT BUFFER)
  5                                       ;
  6 004147  104207  005433                        MOV     #OBUFF,R0               ; R0 POINTS AT OUTPUT BUFFER
  7 004151  104201  005003                        MOV     #RBUFD,R1               ; R1 POINTS AT INPUT BUFFER
  8 004153  114003                                CLR     R3                      ; COUNTER
  9 004154  104274                        CMPLOP: MOV     (R0)+,R4                ; GET OUTPUT BUFFER WORD
 10 004155  106214                                CMP     (R1)+,R4                ; COMPARE AGAINST INPUT BUFFER
 11 004156  014223                                BEQ     NOERR                   ; IF NO ERROR, BRANCH
 12 004157  104305  004774                        MOV     SECCNT,R5               ; SEE IF ERROR IS TO BE REPORTED
 13 004161  054230                                BNE     CMPEXT                  ; IF NOT, BRANCH
 14 004162                                        ERRHRD  MS45,R3,-(R0),-(R1),INS+1,INS+2,INS+3,CURTRK;DATA COMPARE FAILURE
    004162  104010  001316                                                                  MOV R1,SAVREG
    004164  10430C  004772                                                                  MOV CURTRK,R1
    004166  100461                                                                         MOV R1,-(SP)
    004167  104301  004661                                                                  MOV INS+3,R1
    004171  100461                                                                         MOV R1,-(SP)
    004172  104301  004660                                                                  MOV INS+2,R1
    004174  100461                                                                         MOV R1,-(SP)
    004175  104301  004657                                                                  MOV INS+1,R1
    004177  100461                                                                         MOV R1,-(SP)
    004200  10430:  001316                                                                  MOV SAVREG,R1
    004202  104020  001316                                                                  MOV R2,SAVREG
    004204  104412                                                                         MOV -(R1),R2
    004205  100462                                                                         MOV R2,-(SP)
    004206  104302  001316                                                                  MOV SAVREG,R2
    004210  104010  001316                                                                  MOV R1,SAVREG
    004212  104471                                                                         MOV -(R0),R1
    004213  100461                                                                         MOV R1,-(SP)
    004214  100463                                                                         MOV R3,-(SP)
    004215  104301  001316                                                                  MOV SAVREG,R1
    004217  021236                                                                         CALL RERROR     ;ERROR # 54.
    004220  017066                                                                         .WORD <PRMS*2000>+<2*400>+ERRN
    004221  002505                                                                         .WORD MS45
 15 004222  004230                                BR      CMPEXT                  ; EXIT
 16 004223  115403                        NOERR:  INC     R3                      ; INCREMENT COUNT
 17 004224  106203  000400                        CMP     #256.,R3                ; SEE IF COMPARE COMPLETE
 18 004226  054154                                BNE     CMPLOP                  ; IF NOT, BRANCH
 19 004227  114007                                CLR     R0                      ; FLAG AS NO ERRORS
 20 004230  000000                        CMPEXT: RETURN
```

```
                                        ;SEEK
                                        ;
                                        ;SEEK TO CYLINDER POINTED TO BY CONTENTS OF R1
                                        ;
                                        ;INPUTS:
                                        ;       R1 - POINTER TO CYLINDER NUMBER
                                        ;       R2 - SDI INTERCONNECT
                                        ;
004231                                  SEEK:   PUSH <R0,R1,R3>
004231  100467                                                                  MOV R0,-(SP)
004232  100461                                                                  MOV R1,-(SP)
004233  100463                                                                  MOV R3,-(SP)
004234  104217                                  MOV (R1)+,R0                    ;PUT CYLINDER INTO COMMAND
004235  104070  004657                         MOV R0,INS+1
004237  104217                                  MOV (R1)+,R0
004240  104070  004660                         MOV R0,INS+2
004242  104117                                  MOV (R1),R0                     ;PUT GROUP INTO COMMAND
004243  104070  004661                         MOV R0,INS+3
004245  104203  004621                 SEEKA:  MOV #CR.SEK,R3                  ;POINT TO COMMAND
                                                TALKX   TESTEX,MS46,MS47        ; SDI INTERCHANGE
                                                CALL    TALKER                  ; INITIATE SDI INTERCHANGE
                                                TST     R3                      ; SEE IF ERROR OCCURRED
                                                BEQ     12$                     ; IF NOT, BRANCH
                                                BPL     11$                     ; IF SO, BRANCH
                                                ERRHRD  MS46;SEND COMMAND ERROR
004247                                                                                  CALL RERROR     ;ERROR # 55.
004247  021102                                                                          .WORD <PRMS*2000>+<2*400>+ERRN
004250  115003                                                                          .WORD MS46
004251  014263
004252  034257
004253
004253  021236
004254  001067
004255  002645
004256  003443                                  BR      TESTEX
004257                                  11$:    ERRHRD  MS47;RECEIVE COMMAND ERROR
004257  021236                                                                          CALL RERROR     ;ERROR # 56.
004260  001070                                                                          .WORD <PRMS*2000>+<2*400>+ERRN
004261  002674                                                                          .WORD MS47
004262  003443                                  BR      TESTEX
004263                                  12$:
004263  106207  000176                         CMP #COMPLT,R0                  ;CHECK RESPONSE
004265  014313                                 BEQ SEEK1
004266                                         ERRHRD MS48,#COMPLT,R0,INS+1,INS+2,INS+3;SEEK COMMAND REJECTED
004266  104010  001316                                                                  MOV R1,SAVREG
004270  104301  004661                                                                  MOV INS+3,R1
004272  100461                                                                          MOV R1,-(SP)
004273  104301  004660                                                                  MOV INS+2,R1
004275  100461                                                                          MOV R1,-(SP)
004276  104301  004657                                                                  MOV INS+1,R1
004300  100461                                                                          MOV R1,-(SP)
004301  100467                                                                          MOV R0,-(SP)
004302  104201  000176                                                                  MOV #COMPLT,R1
004304  100461                                                                          MOV R1,-(SP)
004305  104301  001316                                                                  MOV SAVREG,R1
004307  021236                                                                          CALL RERROR     ;ERROR # 57.
004310  013071                                                                          .WORD <PRMS*2000>+<2*400>+ERRN
004311  002726                                                                          .WORD MS48
004312  004360                                 BR SEEK3
004313  114003                         SEEK1:  CLR R3                          ;SET UP WORST CASE SEEK TIME
004314                                 SEEK2:  DSTAT   TESTEX,MS10,MS2         ;GET DRIVE SIGNALS
004314  020720                                 CALL    RDSTAT                  ; GET DRIVE STATUS
004315  102201  010000                         BIT     #10000,R1               ; SEE IF ANY ERRORS
```

INITIALIZE DRIVE AND LOOK AT DRIVE SIGNALS

```
004317  014333                          BEQ     2$                      ; IF NO ERROR, BRANCH
004320  102201  004000                  BIT     #4000,R1                ; SEE IF XMIT ERROR
004322  014327                          BEQ     1$                      ; IF SO, BRANCH
004323                                  ERRHRD  MS10                    ; REPORT INVALID STATUS ERROR
004323  021236                                                                  CALL RERROR     ;ERROR # 58.
004324  001072                                                                  .WORD <PRMS*2000>+<2*400>+ERRN
004325  000454                                                                  .WORD MS10
004326  003443                          BR      TESTEX                  ; BRANCH TO DONE
004327                          1$:     ERRHRD  MS2                     ; REPORT XMIT ERROR
004327  021236                                                                  CALL RERROR     ;ERROR # 59.
004330  001073                                                                  .WORD <PRMS*2000>+<2*400>+ERRN
004331  000045                                                                  .WORD MS2
004332  003443                          BR      TESTEX                  ; BRANCH TO DONE
004333                          2$:
24 004333  102201  100000              BIT #RWRDY,R1                   ;IS R/W READY SET?
25 004335  054360                      BNE SEEK3                       ;  YES
26 004336  115403                      INC R3                          ;BUMP COUNT
27 004337  054314                      BNE SEEK2                       ;KEEP WAITING
28 004340                               ERRHRD MS51,INS+1,INS+2,INS+3   ;SEEK COMPLETE TIME OUT
004340  104010  001316                                                          MOV R1,SAVREG
004342  104301  004661                                                          MOV INS+3,R1
004344  100461                                                                  MOV R1,-(SP)
004345  104301  004660                                                          MOV INS+2,R1
004347  100461                                                                  MOV R1,-(SP)
004350  104301  004657                                                          MOV INS+1,R1
004352  100461                                                                  MOV R1,-(SP)
004353  104301  001316                                                          MOV SAVREG,R1
004355  021236                                                                  CALL RERROR     ;ERROR # 60.
004356  007074                                                                  .WORD <PRMS*2000>+<2*400>+ERRN
004357  003041                                                                  .WORD MS51
29 004360                       SEEK3:  POP <R3,R1,R0>
004360  104263                                                                  MOV (SP)+,R3
004361  104261                                                                  MOV (SP)+,R1
004362  104267                                                                  MOV (SP)+,R0
30 004363  000000                          RETURN
```

```
                                ;READ1
                                ;
                                ;READ SECTORS ON THE SELECTED TRACK UNTIL ONE IS READ CORRECTLY
                                ;OR THE MAXIMUM NUMBER OF SECTORS IS READ
                                ;
                                ;INPUTS:
                                ;       R2 - SDI INTERCONNECT
                                ;       R3 - MAXIMUM NUMBER OF SECTORS TO READ
                                ;       R4 - TRACK NUMBER
                                ;       R5 - POINTER TO BLOCK NUMBER OF FIRST SECTOR
                                ;OUTPUTS:
                                ;       NONE
                                ;
004364                          READ1:  PUSH <R0,R1,R3,R4,R5>
004364  100467                                                                 MOV R0,-(SP)
004365  100461                                                                 MOV R1,-(SP)
004366  100463                                                                 MOV R3,-(SP)
004367  100464                                                                 MOV R4,-(SP)
004370  100465                                                                 MOV R5,-(SP)
004371  104207  004777                 MOV #RBUF0+RW.LOW,R0
004373  104251                          MOV (R5)+,R1            ;PUT BLOCK NUMBER IN
004374  100271                          MOV R1,(R0)+            ;  READ BUFFER
004375  104151                          MOV (R5),R1
004376  100271                          MOV R1,(R0)+
004377  104171                          MOV (R0),R1             ;PUT IN TRACK NUMBER
004400  103201  000377                 BIC #377,R1             ;(MERGE WITH REAL-TIME COMMAND)
004402  101041                          BIS R4,R1
004403  100171                          MOV R1,(R0)
004404  104207  004775          READ1A: MOV #RBUF0,R0           ;POINT TO READ BUFFER
004406                          PUSH R2                         ;SAVE SDI INTERCONNECT
004406  100462                                                                 MOV R2,-(SP)
004407                          DSTAT TESTEX,MS10,MS2           ; GET DRIVE REAL TIME STATE
004407  020720                  CALL    RDSTAT                  ; GET DRIVE STATUS
004410  102201  010000          BIT     #10000,R1               ; SEE IF ANY ERRORS
004412  014426                  BEQ     2$                      ; IF NO ERROR, BRANCH
004413  102201  004000          BIT     #4000,R1                ; SEE IF XMIT ERROR
004415  014422                  BEQ     1$                      ; IF SO, BRANCH
004416                          ERRHRD  MS10                    ; REPORT INVALID STATUS ERROR
004416  021236                                                                 CALL RERROR     ;ERROR # 61.
004417  001075                                                                 .WORD <PRMS*2000>+<2*400>+ERRN
004420  000454                                                                 .WORD MS10
004421  003443                  BR      TESTEX                  ; BRANCH TO DONE
004422                          1$:     ERRHRD  MS2             ; REPORT XMIT ERROR
004422  021236                                                                 CALL RERROR     ;ERROR # 62.
004423  001076                                                                 .WORD <PRMS*2000>+<2*400>+ERRN
004424  000045                                                                 .WORD MS2
004425  003443                  BR      TESTEX                  ; BRANCH TO DONE
004426                          2$:
004426  102201  100000          BIT #RWRDY,R1                   ; SEE IF READ WRITE READY IS STILL HIGH
004430  054435                  BNE READ1C                      ; IF SO, BRANCH
004431                          ERRHRD MS43;READ/WRITE READY DROPPED BEFORE READ
004431  021236                                                                 CALL RERROR     ;ERROR # 63.
004432  001077                                                                 .WORD <PRMS*2000>+<2*400>+ERRN
004433  002320                                                                 .WORD MS43
004434  003443                  BR TESTEX                       ; BRANCH
004435  060012          READ1C: XFC WAITSI                      ;WAIT FOR SECTOR OR INDEX PULSE B4 READ
004436  060002                  XFC XREAD                       ;READ THE SECTOR
```

```
     33 004437                                  POP R2
        004437  104262                                                    MOV (SP)+,R2
     34 004440  115001                          TST R1                   ;CHECK FOR ERROR
     35 004441  01-521                          BEQ READ1X               ;END ROUTINE IF READ OK
     36 004442  117403                          DEC R3                   ;COUNT MAX SECTORS TO READ
     37 004443  014456                          BEQ READ1E               ;REPORT ERROR IF ALL READ
     38 004444  104207  004777                  MOV #RBUF0+RW.LOW,R0
     39 004446  104171                          MOV (R0),R1              ;INCREMENT BLOCK NUMBER
     40 004447  115401                          INC R1
     41 004450  100271                          MOV R1,(R0)+
     42 004451  044455                          BCC READ1B
     43 004452  104171                          MOV (R0),R1
     44 004453  115401                          INC R1
     45 004454  100171                          MOV R1,(R0)
     46 004455  004404              READ1B:     BR READ1A                ;GO READ NEXT SECTOR
     47 004456  104201  000114      READ1E:     MOV #'L,R1               ;PREPARE TO PRINT LBN
     48 004460  104307  005000                  MOV RBUF0+RW.HI,R0       ;GET BLOCK NUMBER
     49 004462  102207  170000                  BIT #170000,R0           ;CHECK FOR ZERO HEADER BITS
     50 004464  014467                          BEQ 1$
     51 004465  104201  000130                  MOV #'X,R1               ; IF NOT ZERO, CHANGE TO XBN
     52                                 ;       NO SECTOR ON TRACK CAN BE RFAD
     53 004467                          1$:     ERRHRD MS55,R1,RBUF0+RW.LOW,RBUF0+RW.HI,INS+1,INS+2,INS+3,RBUF0+RW.CMD
        004467  104010  001316                                                    MOV R1,SAVREG
        004471  104301  005001                                                    MOV RBUF0+RW.CMD,R1
        004473  100461                                                           MOV R1,-(SP)
        004474  104301  004661                                                    MOV INS+3,R1
        004476  100461                                                           MOV R1,-(SP)
        004477  104301  004660                                                    MOV INS+2,R1
        004501  100461                                                           MOV R1,-(SP)
        004502  104301  004657                                                    MOV INS+1,R1
        004504  100461                                                           MOV R1,-(SP)
        004505  104301  005000                                                    MOV RBUF0+RW.HI,R1
        004507  100461                                                           MOV R1,-(SP)
        004510  104301  004777                                                    MOV RBUF0+RW.LOW,R1
        004512  100461                                                           MOV R1,-(SP)
        004513  104301  001316                                                    MOV SAVREG,R1
        004515  100461                                                           MOV R1,-(SP)
        004516  021236                                                          CALL RERROR     ;ERROR # 64.
        004517  017100                                                          .WORD <PRMS*2000>+<2*400>+ERRN
        004520  003301                                                          .WORD MS55
     54 004521                          READ1X: POP <R5,R4,R3,R1,R0>
        004521  104265                                                           MOV (SP)+,R5
        004522  104264                                                           MOV (SP)+,R4
        004523  104263                                                           MOV (SP)+,R3
        004524  104261                                                           MOV (SP)+,R1
        004525  104267                                                           MOV (SP)+,R0
     55 004526  000000                          RETURN
```

```
                                ;PROGRAM VARIABLES

                                ;UNIT NUMBER STORAGE FOR DISK DRIVES TO TEST

004527  000020          UNITS:  .REPT   16.             ; 16 SUBUNITS (8 MAX) 4 ON EACH UNIT
                                .WORD   177777          ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
                                .ENDR
004527  177777                  .WORD   177777          ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
004530  177777                  .WORD   177777          ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
004531  177777                  .WORD   177777          ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
004532  177777                  .WORD   177777          ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
004533  177777                  .WORD   177777          ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
004534  177777                  .WORD   177777          ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
004535  177777                  .WORD   177777          ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
004536  177777                  .WORD   177777          ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
004537  177777                  .WORD   177777          ; MARK AL' UNITS AS NON-EXISTANT TO START WITH
004540  177777                  .WORD   177777          ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
004541  177777                  .WORD   177777          ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
004542  177777                  .WORD   177777          ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
004543  177777                  .WORD   177777          ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
004544  177777                  .WORD   177777          ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
004545  177777                  .WORD   177777          ; MARK ALL UNITS AS NON-EXISTANT TO START WITH
004546  177777                  .WORD   177777          ; MARK ALL UNITS AS NON-EXISTANT TO START WITH

004547  000000          UNITNB: .WORD 0                 ;NUMBER OF UNIT CURRENTLY UNDER TEST
                                                        ;POINTER TO TABLE ABOVE

004550  000000          SDI:    .WORD 0                 ;SDI INTERCONNECT CODE FOR XFC CALLS

                        ;       MESSAGE TABLES

004551                  CR.ONL: MSG ONL,2,ST,7                  ; DRIVE ONLINE
004551  004640                  .WORD ONL                       ;ADDRESS OF COMMAND
004552  000002                  .WORD 2                 ;SIZE OF COMMAND IN BYTES
004553  004664                  .WORD ST                        ;ADDRESS OF REPLY
004554  000007                  .WORD 7                 ;SIZE OF REPLY IN WORDS
004555  000000                  .WORD                   ; SUCCESSFUL COMPLETION CODE
004556                  CR.CLR: MSG DRC,2,ST,7                  ; DRIVE CLEAR
004556  004646                  .WORD DRC                       ;ADDRESS OF COMMAND
004557  000002                  .WORD 2                 ;SIZE OF COMMAND IN BYTES
004560  004664                  .WORD ST                        ;ADDRESS OF REPLY
004561  000007                  .WORD 7                 ;SIZE OF REPLY IN WORDS
004562  000000                  .WORD                   ; SUCCESSFUL COMPLETION CODE
004563                  CR.DIA: MSG DIA,3,ST,7                  ; DIAGNOSE
004563  004642                  .WORD DIA                       ;ADDRESS OF COMMAND
004564  000003                  .WORD 3                 ;SIZE OF COMMAND IN BYTES
004565  004664                  .WORD ST                        ;ADDRESS OF REPLY
004566  000007                  .WORD 7                 ;SIZE OF REPLY IN WORDS
004567  000000                  .WORD                   ; SUCCESSFUL COMPLETION CODE
004570                  CR.DIS: MSG DIS,,,ST,7                  ; DISCONNECT
004570  004644                  .WORD DIS                       ;ADDRESS OF COMMAND
004571  000002                  .WORD 2                 ;SIZE OF COMMAND IN BYTES
004572  004664                  .WORD ST                        ;ADDRESS OF REPLY
004573  000007                  .WORD 7                 ;SIZE OF REPLY IN WORDS
004574  000000                  .WORD                   ; SUCCESSFUL COMPLETION CODE
004575                  CR.GCR: MSG GCR,1,CR,12.                ; GET CHARACTERISTICS
004575  004650                  .WORD GCR                       ;ADDRESS OF COMMAND
```

```
                004576  000001                          .WORD 1                 ;SIZE OF COMMAND IN BYTES
                004577  004673                          .WORD CR                        ;ADDRESS OF REPLY
                004600  000014                          .WORD 12.                       ;SIZE OF REPLY IN WORDS
                004601  000000                          .WORD                   ; SUCCESSFUL COMPLETION CODE
             21 004602                          CR.SCR: MSG SCR,2,SUB,21.               ; GET SUBUNIT CHARACTERISTICS
                004602  004651                          .WORD SCR                       ;ADDRESS OF COMMAND
                004603  000002                          .WORD 2                 ;SIZE OF COMMAND IN BYTES
                004604  004706                          .WORD SUB                       ;ADDRESS OF REPLY
                004605  000025                          .WORD 21.                       ;SIZE OF REPLY IN WORDS
                004606  000000                          .WORD                   ; SUCCESSFUL COMPLETION CODE
             22 004607                          CR.GST: MSG GST,1,ST,7                  ; GET STATUS
                004607  004653                          .WORD GST                       ;ADDRESS OF COMMAND
                004610  000001                          .WORD 1                 ;SIZE OF COMMAND IN BYTES
                004611  004664                          .WORD ST                        ;ADDRESS OF REPLY
                004612  000007                          .WORD 7                 ;SIZE OF REPLY IN WORDS
                004613  0000C0                          .WORD                   ; SUCCESSFUL COMPLETION CODE
             23 004614                          CR.MOD: MSG MOD,3,ST,7                  ; MODE
                004614  004662                          .WORD MOD                       ;ADDRFSS OF COMMAND
                004615  000003                          .WORD 3                 ;SIZE OF COMMAND IN BYTES
                004616  004664                          .WORD ST                        ;ADDRESS OF REPLY
                004617  000007                          .WORD 7                 ;SIZE OF REPLY IN WORDS
                004620  000000                          .WORD                   ; SUCCESSFUL COMPLETION CODE
             24 004621                          CR.SEK: MSG INS,6,ST,7                  ; INITIATE SEEK
                004621  004656                          .WORD INS                       ;ADDRESS OF COMMAND
                004622  000006                          .WORD 6                 ;SIZE OF COMMAND IN BYTES
                004623  004664                          .WORD ST                        ;ADDRESS OF REPLY
                004624  000007                          .WORD 7                 ;SIZE OF REPLY IN WORDS
                004625  000000                          .WORD                   ; SUCCESSFUL COMPLETION CODE
             25 004626                          LONG:                                   ; LONG TIMEOUT COMMANDS FOLLOW
             26 004626                          CR.INR: MSG INR,1,ST,7                  ; INITIATE RECALIBRATE
                004626  004655                          .WORD INR                       ;ADDRESS OF COMMAND
                004627  000001                          .WORD 1                 ;SIZE OF COMMAND IN BYTES
                004630  004664                          .WORD ST                        ;ADDRESS OF REPLY
                004631  000007                          .WORD 7                 ;SIZE OF REPLY IN WORDS
                004632  000000                          .WORD                   ; SUCCESSFUL COMPLETION CODE
             27 004633                          CR.RUN: MSG RUN,1,ST,7                  ; RUN
                004633  004654                          .WORD RUN                       ;ADDRESS OF COMMAND
                004634  000001                          .WORD 1                 ;SIZE OF COMMAND IN BYTES
                004635  004664                          .WORD ST                        ;ADDRESS OF REPLY
                004636  000007                          .WORD 7                 ;SIZE OF REPLY IN WORDS
                004637  000000                          .WORD                   ; SUCCESSFUL COMPLETION CODE
             28
             29                                 ;       LEVEL 2 COMMAND MESSAGE DATA STRUCTURES
             30
             31 004640      000      213        ONL:    .BYTE 0,213                     ; BRING DRIVE ONLINE
             32 004641  000077                          .WORD 77
             33 004642      000      003        DIA:    .BYTE 0,3                       ; DIAGNOSE
             34 004643  000000                          .WORD 0
             35 004644      000      204        DIS:    .BYTE 0,204                     ; DISCONNECT
             36 004645  000000                          .WORD 0
             37 004646      000      005        DRC:    .BYTE 0,DRVCLR                  ; DRIVE CLEAR
             38 004647  000000                  ERRORS: .WORD 0                         ; ERROR FLAGS
             39 004650      000      207        GCR:    .BYTE 0,GETCHR                  ; GET CHARACTERISTICS
             40 004651      000      210        SCR:    .BYTE 0,GETSUB                  ; GET SUBUNIT CHARACTERISTICS
             41 004652  000000                  SUBUNT: .WORD 0                         ; SUBUNIT SELECTION IN LOW ORDER BYTE
             42 004653      000      011        GST:    .BYTE 0,GETSTA                  ; GET STATUS
             43 004654      000      014        RUN:    .BYTE 0,14                      ; SPIN UP DRIVE
```

```
     44 004655     000     216          INR:    .BYTE 0,IRECLB                  ; INITIATE RECALIBRATE
     45 004656     000     012          INS:    .BYTE 0,INSEEK                  ; INITIATE SEEK
     46 004657  000000                          .WORD 0                         ; INS CYLINDER/HEAD ARGUMENTS
     47 004660  000000                          .WORD 0
     48 004661  000000                          .WORD 0
     49 004662     000     201          MOD:    .BYTE 0,201                     ; CHANGE MODE
     50 004663     006     037                  .BYTE 6,37
     51
     52                                 ;       RESPONSE  MESSAGE DATA BUFFERS
     53
     54 004664                          ST:     .BLKW 7                         ;STATUS MESSAGE BUFFER
     55 004673                          CR:     .BLKW 31.                       ;CHARACTERISTICS MESSAGE BUFF
     56
     57 004732  000000                  DMSDI:  .WORD 0                         ; DUMMY SDI CONTROL BLK
     58 004733  000000                  NSCSL:  .WORD 0                         ;# OF SECTORS IN HEADER SEARCH
     59 004734  004701                          .WORD SUB-5                     ;POINTER TO SUBUNIT CHAR.
     60 004735                                  .BLKW 5                         ;WORD DMSDI+7 IS CLOBBERED BY UDA
     61                                                                         ;SO SET ASIDE SPACE
     62
     63                                 ;       DISK LOCATION POINTERS
     64
     65 004742                          LCDBN:  .BLKW 2                         ;FIRST FACTORY FORMATTED DBN
     66 004744                          LDIACYL:.BLKW 3                         ;FACTORY FORMATTED CYLINDER
     67 004747                          FDIACYL:.BLKW 2                         ;FIRST DIAGNOSTIC CYLINDER
     68 004751                          FXBNCYL:.BLKW 2                         ;FIRST XBN CYLINDER
     69 004753                          SECTRK: .BLKW 1                         ;SECTORS PER TRACK
     70
     71 004754                          TSTBLK: .BLKW 2                         ;TEST BLOCK NUMBER
     72 004756                          TSTCYL: .BLKW 3                         ;TEST CYLINDER NUMBER
     73 004761                          BLOCKG: .BLKW 1                         ;BLOCKS ON CURRENT GROUP
     74 004762                          BLOCKT: .BLKW 1                         ;BLOCKS ON CURRENT TRACK
     75 004763                          CURBLK: .BLKW 2                         ; CURRENT BLOCK NUMBER
     76 004765                          SECGRP: .BLKW 1                         ; NUMBER OF SECTORS PER GROUP
     77 004766                          CURGRP: .BLKW 1                         ; CURRENT GROUP NUMBER
     78 004767                          CURCYL: .BLKW 3                         ; CURRENT CYLINDER AND GROUP
     79 004772  000000                  CURTRK: .WORD 0                         ; CURRENT TRACK
     80 004773  000000                  CURPAT: .WORD 0                         ; CURRENT PATTERN
     81 004774  000000                  SECCNT: .WORD 0                         ; SECTOR COUNT
     82
     83
     84
     85                                 ;       READ DATA BUFFERS
     86
     87 004775                          RBUF0:                                  ; FIRST BUFFER
     88 004775  100000                          .WORD RSTOP                     ; STATUS AND LINK POINTER
     89 004776  005003                          .WORD RBUFD                     ; POINTER TO DATA
     90 004777                                  .BLKW 2
     91 005001  013400                          .WORD RREAL                     ; LEVEL 1 SDI COMMAND
     92 005002  004732                          .WORD DMSDI                     ; POINTER TO DUMMY SDI CONTROL BLOCK
     93
     94
     95 005003                          RBUFD:  .BLKW 274.                      ;READ DATA BUFFER
     96                                 ;       WRITE DATA BUFFERS
     97
     98 005425                          CHAIN:                                  ; READ AND WRITE CHAIN AREA
     99 005425  000000                          .WORD 0                         ; STATUS AREA
    100 005426  000000                          .WORD 0                         ; BUFFER POINTER
```

```
                101 005427  00000C                          .WORD 0                 ; LO ORDER HEADER
                102 005430  000000                          .WORD 0                 ; HI ORDER HEADER
                103 005431  000000                          .WORD 0                 ; REAL TIME COMMAND AND TRACK NUMBER
                104 005432  004732                          .WORD DMSDI             ; POINTER TO DUMMY SDI CONTROL BLOCK
                105
                106 005433                          OBUFF:  .BLKW 257.              ;WRITE DATA BUFFER
                107
                108
                109                                 ;       FORMAT CHAIN
                110
                111 006034  000074                  FCHAIN: .REPT 60.               ; UP TO 60 DBNS/TRK MAXIMUM
                112                                         .WORD   0               ; BUFFER POINTER
                113                                         .WORD   0               ; LO ORDER DBN
                114                                         .WORD   0               ; HI ORDER DBN
                115                                         .ENDR
                    006034  000000                          .WORD   0               ; BUFFER POINTER
                    006035  000000                          .WORD   0               ; LO ORDER DBN
                    006036  000000                          .WORD   0               ; HI ORDER DBN
                    006037  000000                          .WORD   0               ; BUFFER POINTER
                    006040  000000                          .WORD   0               ; LO ORDER DBN
                    006041  000000                          .WORD   0               ; HI ORDER DBN
                    006042  000000                          .WORD   0               ; BUFFER POINTER
                    006043  000000                          .WORD   0               ; LO ORDER DBN
                    006044  000000                          .WORD   0               ; HI ORDER DBN
                    006045  000000                          .WORD   0               ; BUFFER POINTER
                    006046  000000                          .WORD   0               ; LO ORDER DBN
                    006047  000000                          .WORD   0               ; HI ORDER DBN
                    006050  000000                          .WORD   0               ; BUFFER POINTER
                    006051  000000                          .WORD   0               ; LO ORDER DBN
                    006052  000000                          .WORD   0               ; HI ORDER DBN
                    006053  000000                          .WORD   0               ; BUFFER POINTER
                    006054  000000                          .WORD   0               ; LO ORDER DBN
                    006055  000000                          .WORD   0               ; HI ORDER DBN
                    006056  000000                          .WORD   0               ; BUFFER POINTER
                    006057  000000                          .WORD   0               ; LO ORDER DBN
                    006060  000000                          .WORD   0               ; HI ORDER DBN
                    006061  000000                          .WORD   0               ; BUFFER POINTER
                    006062  000000                          .WORD   0               ; LO ORDER DBN
                    006063  000000                          .WORD   0               ; HI ORDER DBN
                    006064  000000                          .WORD   0               ; BUFFER POINTER
                    006065  000000                          .WORD   0               ; LO ORDER DBN
                    006066  000000                          .WORD   0               ; HI ORDER DBN
                    006067  000000                          .WORD   0               ; BUFFER POINTER
                    006070  000000                          .WORD   0               ; LO ORDER DBN
                    006071  000000                          .WORD   0               ; HI ORDER DBN
                    006072  000000                          .WORD   0               ; BUFFER POINTER
                    006073  000000                          .WORD   0               ; LO ORDER DBN
                    006074  000000                          .WORD   0               ; HI ORDER DBN
                    006075  000000                          .WORD   0               ; BUFFER POINTER
                    006076  000000                          .WORD   0               ; LO ORDER DBN
                    006077  000000                          .WORD   0               ; HI ORDER DBN
                    006100  000000                          .WORD   0               ; BUFFER POINTER
                    006101  000000                          .WORD   0               ; LO ORDER DBN
                    006102  000000                          .WORD   0               ; HI ORDER DBN
                    006103  000000                          .WORD   0               ; BUFFER POINTER
                    006104  000000                          .WORD   0               ; LO ORDER DBN
                    006105  000000                          .WORD   0               ; HI ORDER DBN
```

```
006106  000000          .WORD   0               ; BUFFER POINTER
006107  000000          .WORD   0               ; LO ORDER DBN
006110  000000          .WORD   0               ; HI ORDER DBN
006111  000000          .WORD   0               ; BUFFER POINTER
006112  000000          .WORD   0               ; LO ORDER DBN
006113  000000          .WORD   0               ; HI ORDER DBN
006114  000000          .WORD   0               ; BUFFER POINTER
006115  000000          .WORD   0               ; LO ORDER DBN
006116  000000          .WORD   0               ; HI ORDER DBN
006117  000000          .WORD   0               ; BUFFER POINTER
006120  000000          .WORD   0               ; LO ORDER DBN
006121  000000          .WORD   0               ; HI ORDER DBN
006122  000000          .WORD   0               ; BUFFER POINTER
006123  000000          .WORD   0               ; LO ORDER DBN
006124  000000          .WORD   0               ; HI ORDER DBN
006125  000000          .WORD   0               ; BUFFER POINTER
006126  000000          .WORD   0               ; LO ORDER DBN
006127  000000          .WORD   0               ; HI ORDER DBN
006130  000000          .WORD   0               ; BUFFER POINTER
006131  000000          .WORD   0               ; LO ORDER DBN
006132  000000          .WORD   0               ; HI ORDER DBN
006133  000000          .WORD   0               ; BUFFER POINTER
006134  000000          .WORD   0               ; LO ORDER DBN
006135  000000          .WORD   0               ; HI ORDER DBN
006136  000000          .WORD   0               ; BUFFER POINTER
006137  000000          .WORD   0               ; LO ORDER DBN
006140  000000          .WORD   0               ; HI ORDER DBN
006141  000000          .WORD   0               ; BUFFER POINTER
006142  000000          .WORD   0               ; LO ORDER DBN
006143  000000          .WORD   0               ; HI ORDER DBN
006144  000000          .WORD   0               ; BUFFER POINTER
006145  000000          .WORD   0               ; LO ORDER DBN
006146  000000          .WORD   0               ; HI ORDER DBN
006147  000000          .WORD   0               ; BUFFER POINTER
006150  000000          .WORD   0               ; LO ORDER DBN
006151  000000          .WORD   0               ; HI ORDER DBN
006152  000000          .WORD   0               ; BUFFER POINTER
006153  000000          .WORD   0               ; LO ORDER DBN
006154  000000          .WORD   0               ; HI ORDER DBN
006155  000000          .WORD   0               ; BUFFER POINTER
006156  000000          .WORD   0               ; LO ORDER DBN
006157  000000          .WORD   0               ; HI ORDER DBN
006160  000000          .WORD   0               ; BUFFER POINTER
006161  000000          .WORD   0               ; LO ORDER DBN
006162  000000          .WORD   0               ; HI ORDER DBN
006163  000000          .WORD   0               ; BUFFER POINTER
006164  000000          .WORD   0               ; LO ORDER DBN
006165  000000          .WORD   0               ; HI ORDER DBN
006166  000000          .WORD   0               ; BUFFER POINTER
006167  000000          .WORD   0               ; LO ORDER DBN
006170  000000          .WORD   0               ; HI ORDER DBN
006171  000000          .WORD   0               ; BUFFER POINTER
006172  000000          .WORD   0               ; LO ORDER DBN
006173  000000          .WORD   0               ; HI ORDER DBN
006174  000000          .WORD   0               ; BUFFER POINTER
006175  000000          .WORD   0               ; LO ORDER DBN
006176  000000          .WORD   0               ; HI ORDER DBN
```

```
006177  000000C                 .WORD   0               ; BUFFER POINTER
006200  000000                  .WORD   0               ; LO ORDER DBN
006201  000000                  .WORD   0               ; HI ORDER DBN
006202  000000                  .WORD   0               ; BUFFER POINTER
006203  000000                  .WORD   0               ; LO ORDER DBN
006204  000000                  .WORD   0               ; HI ORDER DBN
006205  000000                  .WORD   0               ; BUFFER POINTER
006206  000000                  .WORD   0               ; LO ORDER DBN
006207  000000                  .WORD   0               ; HI ORDER DBN
006210  000000                  .WORD   0               ; BUFFER POINTER
006211  000000                  .WORD   0               ; LO ORDER DBN
006212  000000                  .WORD   0               ; HI ORDER DBN
006213  000000                  .WORD   0               ; BUFFER POINTER
006214  000000                  .WORD   0               ; LO ORDER DBN
006215  000000                  .WORD   0               ; HI ORDER DBN
006216  000000                  .WORD   0               ; BUFFER POINTER
006217  000000                  .WORD   0               ; LO ORDER DBN
006220  000000                  .WORD   0               ; HI ORDER DBN
006221  000000                  .WORD   0               ; BUFFER POINTER
006222  000000                  .WORD   0               ; LO ORDER DBN
006223  000000                  .WORD   0               ; HI ORDER DBN
006224  000000                  .WORD   0               ; BUFFER POINTER
006225  000000                  .WORD   0               ; LO ORDER DBN
006226  000000                  .WORD   0               ; HI ORDER DBN
006227  000000                  .WORD   0               ; BUFFER POINTER
006230  000000                  .WORD   0               ; LO ORDER DBN
006231  000000                  .WORD   0               ; HI ORDER DBN
006232  000000                  .WORD   0               ; BUFFER POINTER
006233  000000                  .WORD   0               ; LO ORDER DBN
006234  000000                  .WORD   0               ; HI ORDER DBN
006235  000000                  .WORD   0               ; BUFFER POINTER
006236  000000                  .WORD   0               ; LO ORDER DBN
006237  000000                  .WORD   0               ; HI ORDER DBN
006240  000000                  .WORD   0               ; BUFFER POINTER
006241  000000                  .WORD   0               ; LO ORDER DBN
006242  000000                  .WORD   0               ; HI ORDER DBN
006243  000000                  .WORD   0               ; BUFFER POINTER
006244  000000                  .WORD   0               ; LO ORDER DBN
006245  000000                  .WORD   0               ; HI ORDER DBN
006246  000000                  .WORD   0               ; BUFFER POINTER
006247  000000                  .WORD   0               ; LO ORDER DBN
006250  000000                  .WORD   0               ; HI ORDER DBN
006251  000000                  .WORD   0               ; BUFFER POINTER
006252  000000                  .WORD   0               ; LO ORDER DBN
006253  000000                  .WORD   0               ; HI ORDER DBN
006254  000000                  .WORD   0               ; BUFFER POINTER
006255  000000                  .WORD   0               ; LO ORDER DBN
006256  000000                  .WORD   0               ; HI ORDER DBN
006257  000000                  .WORD   0               ; BUFFER POINTER
006260  000000                  .WORD   0               ; LO ORDER DBN
006261  000000                  .WORD   0               ; HI ORDER DBN
006262  000000                  .WORD   0               ; BUFFER POINTER
006263  000000                  .WORD   0               ; LO ORDER DBN
006264  000000                  .WORD   0               ; HI ORDER DBN
006265  000000                  .WORD   0               ; BUFFER POINTER
006266  000000                  .WORD   0               ; LO ORDER DBN
006267  000000                  .WORD   0               ; HI ORDER DBN
```

```
        006270  000000                  .WORD   0               ; BUFFER POINTER
        006271  000000                  .WORD   0               ; LO ORDER DBN
        006272  000000                  .WORD   0               ; HI ORDER DBN
        006273  000000                  .WORD   0               ; BUFFER POINTER
        006274  000000                  .WORD   0               ; LO ORDER DBN
        006275  000000                  .WORD   0               ; HI ORDER DBN
        006276  000000                  .WORD   0               ; BUFFER POINTER
        006277  000000                  .WORD   0               ; LO ORDER DBN
        006300  000000                  .WORD   0               ; HI ORDER DBN
        006301  000000                  .WORD   0               ; BUFFER POINTER
        006302  000000                  .WORD   0               ; LO ORDER DBN
        006303  000000                  .WORD   0               ; HI ORDER DBN
        006304  000000                  .WORD   0               ; BUFFER POINTER
        006305  000000                  .WORD   0               ; LO ORDER DBN
        006306  000000                  .WORD   0               ; HI ORDER DBN
        006307  000000                  .WORD   0               ; BUFFER POINTER
        006310  000000                  .WORD   0               ; LO ORDER DBN
        006311  000000                  .WORD   0               ; HI ORDER DBN
        006312  000000                  .WORD   0               ; BUFFER POINTER
        006313  000000                  .WORD   0               ; LO ORDER DBN
        006314  000000                  .WORD   0               ; HI ORDER DBN
        006315  000000                  .WORD   0               ; BUFFER POINTER
        006316  000000                  .WORD   0               ; LO ORDER DBN
        006317  000000                  .WORD   0               ; HI ORDER DBN
    116 006320  000000                  .WORD   0               ; EXTRA WORD FOR END-OF-CHAIN FLAG
```

```
                                PATPTR: .WORD   4                       ; FOUR PATTERNS WILL BE TESTED
                                        .WORD   PAT4
                                        .WORD   PAT5
                                        .WORD   PAT6
                                        .WORD   PAT8
                                ;
                                ;       THE DATA PATTERNS (LENGTH OF PATTERN IN WORDS, FOLLOWED BY PATTERN)
                                ;
                                PAT1:   .WORD   1
                                        .WORD   105613
                                PAT2:   .WORD   1
                                        .WORD   031463
                                PAT3:   .WORD   1
                                        .WORD   030221
                                PAT4:   .WORD   16                      ; SHIFTING ONES
                                        .WORD   000001
                                        .WORD   000003
                                        .WORD   000007
                                        .WORD   000017
                                        .WORD   000037
                                        .WORD   000077
                                        .WORD   000177
                                        .WORD   000377
                                        .WORD   000777
                                        .WORD   001777
                                        .WORD   003777
                                        .WORD   007777
                                        .WORD   017777
                                        .WORD   037777
                                        .WORD   077777
                                        .WORD   177777
                                PAT5:   .WORD   16                      ; SHIFTING ZEROS
                                        .WORD   177776
                                        .WORD   177774
                                        .WORD   177770
                                        .WORD   177760
                                        .WORD   177740
                                        .WORD   177700
                                        .WORD   177600
                                        .WORD   177400
                                        .WORD   177000
                                        .WORD   176000
                                        .WORD   174000
                                        .WORD   170000
                                        .WORD   160000
                                        .WORD   140000
                                        .WORD   100000
                                        .WORD   000000
                                PAT6:   .WORD   16                      ; ALTERNATING ZERO WORD AND ONE WORD IN
                                        .WORD   000000                  ; 3-2-1-1-1 SEQUENCE
                                        .WORD   000000
                                        .WORD   000000
                                        .WORD   177777
                                        .WORD   177777
                                        .WORD   177777
                                        .WORD   000000
                                        .WORD   000000
```

```
 58 006407  177777                        .WORD   177777
 59 006410  177777                        .WORD   177777
 60 006411  000000                        .WORD   000000
 61 006412  177777                        .WORD   177777
 62 006413  000000                        .WORD   000000
 63 006414  177777                        .WORD   177777
 64 006415  000000                        .WORD   000000
 65 006416  177777                        .WORD   177777
 66 006417  000001              PAT7:     .WORD   1               ; B'1011011011011001'
 67 006420  133331                        .WORD   133331
 68 006421  000016              PAT8:     .WORD   16              ; B'0101010101010101' AND
 69 006422  052525                        .WORD   052525          ; B'1010101010101010'
 70 006423  052525                        .WORD   052525          ; IN 3-2-1-1-1 SEQURNCE
 71 006424  052525                        .WORD   052525
 72 006425  125252                        .WORD   125252
 73 006426  125252                        .WORD   125252
 74 006427  125252                        .WORD   125252
 75 006430  052525                        .WORD   052525
 76 006431  052525                        .WORD   052525
 77 006432  125252                        .WORD   125252
 78 006433  125252                        .WORD   125252
 79 006434  052525                        .WORD   052525
 80 006435  125252                        .WORD   125252
 81 006436  052525                        .WORD   052525
 82 006437  125252                        .WORD   125252
 83 006440  052525                        .WORD   052525
 84 006441  125252                        .WORD   125252
 85 006442  000001              PAT9:     .WORD   1               ; B'1101101101101100'
 86 006443  155554                        .WORD   155554
 87 006444  000016              PAT10:    .WORD   16              ; B'0010110100101101' AND
 88 006445  026455                        .WORD   026455          ; B'1101001011010010'
 89 006446  026455                        .WORD   026455          ; IN 3-2-1-1-1 SEQUENCE
 90 006447  026455                        .WORD   026455
 91 006450  151322                        .WORD   151322
 92 006451  151322                        .WORD   151322
 93 006452  151322                        .WORD   151322
 94 006453  026455                        .WORD   026455
 95 006454  026455                        .WORD   026455
 96 006455  151322                        .WORD   151322
 97 006456  151322                        .WORD   151322
 98 006457  026455                        .WORD   026455
 99 006460  151322                        .WORD   151322
100 006461  026455                        .WORD   026455
101 006462  151322                        .WORD   151322
102 006463  026455                        .WORD   026455
103 006464  151322                        .WORD   151322
104 006465  000001              PAT11:    .WORD   1               ; B'0110110110110110'
105 006466  066666                        .WORD   066666
106 006467  000016              PAT12:    .WORD   16              ; RIPLE ONE
107 006470  000001                        .WORD   000001
108 006471  000002                        .WORD   000002
109 006472  000004                        .WORD   000004
110 006473  000010                        .WORD   000010
111 006474  000020                        .WORD   000020
112 006475  000040                        .WORD   000040
113 006476  000100                        .WORD   000100
114 006477  000200                        .WORD   000200
```

```
115 006500  000400                      .WORD   000400
116 006501  001000                      .WORD   001000
117 006502  002000                      .WORD   002000
118 006503  004000                      .WORD   004000
119 006504  010000                      .WORD   010000
120 006505  020000                      .WORD   020000
121 006506  040000                      .WORD   040000
122 006507  100000                      .WORD   100000
123 006510  000016              PAT13:  .WORD   16              ; RIPLE ZERO
124 006511  177776                      .WORD   177776
125 006512  177775                      .WORD   177775
126 006513  177773                      .WORD   177773
127 006514  177767                      .WORD   177767
128 006515  177757                      .WORD   177757
129 006516  177737                      .WORD   177737
130 006517  177677                      .WORD   177677
131 006520  177577                      .WORD   177577
132 006521  177377                      .WORD   177377
133 006522  176777                      .WORD   176777
134 006523  175777                      .WORD   175777
135 006524  173777                      .WORD   173777
136 006525  167777                      .WORD   167777
137 006526  157777                      .WORD   157777
138 006527  137777                      .WORD   137777
139 006530  077777                      .WORD   077777
140 006531  000002              PAT14:  .WORD   2               ; B'1111010110110110'
141 006532  172666                      .WORD   172666          ; B'1101101101101101'
142 006533  155555                      .WORD   155555
143 006534  000016              PAT15:  .WORD   16              ; THIS DATA PATTERN MAY BE DEFINED BY THE
144 006535  000000                      .WORD   0               ; USER OR LEFT IN ITS DEFAULT STATE
145 006536  000000                      .WORD   0
146 006537  000000                      .WORD   0
147 006540  000000                      .WORD   0
148 006541  000000                      .WORD   0
149 006542  000000                      .WORD   0
150 006543  000000                      .WORD   0
151 006544  000000                      .WORD   0
152 006545  000000                      .WORD   0
153 006546  000000                      .WORD   0
154 006547  000000                      .WORD   0
155 006550  000000                      .WORD   0
156 006551  000000                      .WORD   0
157 006552  000000                      .WORD   0
158 006553  000000                      .WORD   0
159 006554  000000                      .WORD   0
```

```
                                ;MESSAGE STORAGE OVERLAY


006555                                  DMOVLY MS,0
006555  050630                          .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                        .NLIST BEX

000000      042     103     101  MS1:   .ASCII\'CANNOT RECEIVE DRIVE STATE FROM DRIVE'N\
000024      042     103     110        .ASCII\'CHECK IF DRIVE IS POWERED ON.'N\
000044      000                        .BYTE 0
000045      042     104     122  MS2:   .ASCII\'DRIVE STATE RECEIVED HAS BAD PARITY'N\
000070      000                        .BYTE 0
000071      042     104     122  MS3:   .ASCII\'DRIVE NOT ASSERTING RECEIVER READY IN DRIVE STATE'N\
000123      000                        .BYTE 0
000124      042     124     111  MS4:   .ASCII\'TIME-OUT ON SEND OF ONLINE COMMAND TO DRIVE'N\
000153      000                        .BYTE 0
000154      042     124     111  MS5:   .ASCII\'TIME-OUT ON RECEIVE OF ONLINE RESPONSE FROM DRIVE'N\
000206      000                        .BYTE 0
000207      042     117     116  MS6:   .ASCII\'ONLINE COMMAND DID NOT RETURN EXPECTED RESPONSE CODE'N\
000242      042     040     040        .ASCII\'  EXPECTED RESPONSE  '08N\
000257      042     040     040        .ASCII\'  ACTUAL RESPONSE    '08N\
000274      000                        .BYTE 0
000275      042     124     111  MS7:   .ASCII\'TIME-OUT ON SEND OF GET STATUS COMMAND TO DRIVE'N\
000326      000                        .BYTE 0
000327      042     124     111  MS8:   .ASCII\'TIME-OUT ON RECEIVE OF GET STATUS RESPONSE FROM DRIVE'N\
000363      000                        .BYTE 0
000364      042     107     105  MS9:   .ASCII\'GET STATUS COMMAND DID NOT RETURN EXPECTED RESPONSE CODE'N\
000421      042     040     040        .ASCII\'  EXPECTED RESPONSE  '08N\
000436      042     040     040        .ASCII\'  ACTUAL RESPONSE    '08N\
000453      000                        .BYTE 0
000454      042     106     101  MS10:  .ASCII\'FAILED TO RECEIVE VALID DRIVE STATE'N\
000477      000                        .BYTE 0
000500      042     101     124  MS12:  .ASCII\'ATTENTION DID NOT DROP STATE AFTER GET STATUS COMMAND'N\
000534      000                        .BYTE 0
000535      042     124     111  MS13:  .ASCII\'TIME-OUT ON SEND OF DRIVE CLEAR COMMAND TO DRIVE'N\
000566      000                        .BYTE 0
000567      042     124     111  MS14:  .ASCII\'TIME-OUT ON RECEIVE OF DRIVE CLEAR RESPONSE FROM DRIVE'N\
000623      000                        .BYTE 0
000624      042     104     122  MS15:  .ASCII\'DRIVE CLEAR COMMAND DID NOT RETURN EXPECTED RESPONSE CODE'N\
000662      042     040     040        .ASCII\'  EXPECTED RESPONSE  '08N\
000677      042     040     040        .ASCII\'  ACTUAL RESPONSE    '08N\
000714      000                        .BYTE 0
000715      042     124     111  MS16:  .ASCII\'TIME-OUT ON SEND OF CHANGE MODE COMMAND TO DRIVE'N\
000746      000                        .BYTE 0
000747      042     124     111  MS17:  .ASCII\'TIME-OUT ON RECEIVE OF CHANGE MODE RESPONSE FROM DRIVE'N\
001003      000                        .BYTE 0
001004      042     103     110  MS18:  .ASCII\'CHANGE MODE COMMAND DID NOT RETURN EXPECTED RESPONSE CODE'N\
001042      042     040     040        .ASCII\'  EXPECTED RESPONSE  '08N\
001057      042     040     040        .ASCII\'  ACTUAL RESPONSE    '08N\
001074      000                        .BYTE 0
001075      042     124     111  MS19:  .ASCII\'TIME-OUT ON SEND OF RECALIBRATE COMMAND TO DRIVE'N\
001126      000                        .BYTE 0
001127      042     124     111  MS20:  .ASCII\'TIME-OUT ON RECEIVE OF RECALIBRATE RESPONSE FROM DRIVE'N\
001163      000                        .BYTE 0
001164      042     122     105  MS21:  .ASCII\'RECALIBRATE COMMAND DID NOT RETURN EXPECTED RESPONSE CODE'N\
001222      042     040     040        .ASCII\'  EXPECTED RESPONSE  '08N\
001237      042     040     040        .ASCII\'  ACTUAL RESPONSE    '08N\
```

```
 57 001254  000                                .BYTE 0
 58 001255  042  101  124  MS24:   .ASCII\'ATTENTION SET AFTER RECALIBRATE COMMAND'N\
 59 001302  000                                .BYTE 0
 60 001303  042  116  10_  MS25:   .ASCII\'NEITHER R/W READY NOR ATTENTION SET AFTER RECALIBRATE COMMAND'N\
 61 001343  000                                .BYTE 0
 62 001344  042  124  111  M`26:   .ASCII\'TIME-OUT ON SEND OF GET UNIT CHARACTERISTICS COMMAND TO DRIVE'N\
 63 001404  000                                .BYTE 0
 64 001405  042  124  111  MS27:   .ASCII\'TIME-OUT ON RECEIVE OF GET UNIT CHARACTERISTICS RESPONSE FROM DRIVE'N\
 65 001450  000                                .BYTE 0
 66 001451  042  107  105  MS28:   .ASCII\'GET UNIT CHARACTERISTICS COMMAND DID NOT RETURN EXPECTED RESPONSE CODE'N\
 67 001515  042  040  040          .ASCII\'  EXPECTED RESPONSE  '\D8N\
 68 001532  042  040  040          .ASCII\'  ACTUAL RESPONSE    '\D8N\
 69 001547  000                                .BYTE 0
 70 001550  042  124  111  MS29:   .ASCII\'TIME-OUT ON SEND OF GET SUBUNIT CHARACTERISTICS COMMAND TO DRIVE'N\
 71 001611  000                                .BYTE 0
 72 001612  042  124  111  MS30:   .ASCII\'TIME-OUT ON RECEIVE OF GET SUBUNIT CHARACTERISTICS RESPONSE FROM DRIVE'N\
 73 001656  000                                .BYTE 0
 74 001657  042  107  105  MS31:   .ASCII\'GET SUBUNIT CHARACTERISTICS COMMAND DID NOT RETURN EXPECTED RESPONSE CODE'N\
 75 001725  042  040  040          .ASCII\'  EXPECTED RESPONSE  '\D8N\
 76 001742  042  040  040          .ASCII\'  ACTUAL RESPONSE    '\D8N\
 77 001757  000                                .BYTE 0
 78 001760  042  123  125  MS32:   .ASCII\'SUBUNIT CHARACTERISTICS SAY LESS THAN 1 DIAGNOSTIC CYLINDER'N\
 79 002017  000                                .BYTE 0
 80 002020  042  122  105  MS35:   .ASCII\'READ/WRITE READY DROPPED BEFORE FORMAT OPERATION'N\
 81 002051  000                                .BYTE 0
 82 002052  042  106  117  MS36:   .ASCII\'FORMAT OPERATION REPORTED TIME-OUT FAILURE'N\
 83 002100  042  040  040          .ASCII\'  CYLINDER '\D32''.   GROUP  '\D8''.  TRACK '\D8''.'N\
 84 002130  000                                .BYTE 0
 85 002131  042  122  105  MS39:   .ASCII\'READ/WRITE READY DROPPED BEFORE WRITE OPERATION'N\
 86 002162  000                                .BYTE 0
 87 002163  042  103  117  MS40:   .ASCII\'COULD NOT WRITE AND READ ANY BLOCK ON THIS TRACK. ON LAST BLOCK:'N\
 88 002224  042  127  122          .ASCII\'WRITE OPERATION REPORTED FAILURE -- ERROR CODE '\D8'' OCTAL.'N\
 89 002263  042  104  102          .ASCII\'DBN '\D28''.  CYLINDER '\D32''.  GROUP '\D8''.  TRACK '\D8''.'N\
 90 002317  000                                .BYTE 0
 91 002320  042  122  105  MS43:   .ASCII\'READ/WRITE READY DROPPED BEFORE READ OPERATION'N\
 92 002350  000                                .BYTE 0
 93 002351  042  103  117  MS44:   .ASCII\'COULD NOT WRITE AND READ ANY BLOCK ON THIS TRACK. ON LAST BLOCK:'N\
 94 002412  042  122  105          .ASCII\'READ OPERATION REPORTED FAILURE -- ERROR CODE '\D8'' OCTAL.'N\
 95 002450  042  104  102          .ASCII\'DBN '\D28''.  CYLINDER '\D32''.  GROUP '\D8''.  TRACK '\D8''.'N\
 96 002504  000                                .BYTE 0
 97 002505  042  103  117  MS45:   .ASCII\'COULD NOT WRITE AND READ ANY BLOCK ON THIS TRACK. ON LAST BLOCK:'N\
 98 002546  042  104  101          .ASCII\'DATA COMPARE FAILURE ON WORD '\D16''.'N\
 99 002571  042  105  130          .ASCII\'EXPECTED DATA  '\D16N\
100 002604  042  101  103          .ASCII\'ACTUAL DATA    '\D16N\
101 002616  042  103  131          .ASCII\'CYLINDER '\D32''.  GROUP '\D8''.  TRACK '\D8''.'N\
102 002644  000                                .BYTE 0
103 002645  042  124  111  MS46:   .ASCII\'TIME-OUT ON SEND OF SEEK COMMAND TO DRIVE'N\
104 002673  000                                .BYTE 0
105 002674  042  124  111  MS47:   .ASCII\'TIME-OUT ON RECEIVE OF SEEK RESPONSE FROM DRIVE'N\
106 002725  000                                .BYTE 0
107 002726  042  123  105  MS48:   .ASCII\'SEEK COMMAND DID NOT RETURN EXPECTED RESPONSE CODE'N\
108 002760  042  040  040          .ASCII\'  EXPECTED RESPONSE  '\D8N\
109 002775  042  040  040          .ASCII\'  ACTUAL RESPONSE    '\D8N\
110 003012  042  123  105          .ASCII\'SEEK WAS TO CYLINDER '\D32''.  GROUP '\D8''.'N\
111 003040  000                                .BYTE 0
112 003041  042  123  10_  MS51:   .ASCII\'SEEK COMPLETE TIME-OUT -- READ/WRITE READY DID NOT SET.'N\
113 003076  042  123  `05          .ASCII\'SEEK WAS TO CYLINDER '\D32''.  GROUP '\D8''.'N\
```

```
003123  000                            .BYTE 0
003124  042     124     111     MS52:  .ASCII  \"TIME-OUT ON SEND OF DISCONNECT COMMAND TO DRIVE"N\
003155  000                            .BYTE   0
003156  042     124     111     MS53:  .ASCII  \"TIME-OUT ON RECEIVE OF DISCONNECT RESPONSE FROM DRIVE"N\
003212  000                            .BYTE   0
003213  042     104     111     MS54:  .ASCII  \"DISCONNECT COMMAND DID NOT RETURN EXPECTED RESPONSE"N\
003246  042     040     040            .ASCII  \"  EXPECTED RESPONSE  "D8N\
003263  042     040     040            .ASCII  \"  ACTUAL RESPONSE    "D8N\
003300  000                            .BYTE   0
003301  042     116     117     MS55:  .ASCII\"NO BLOCK ON THIS TRACK CAN BE READ. LAST BLOCK TRIED:"N\
003335  101     042     102            .ASCII\A"BN "D28".  CYLINDER "D32".  GROUP "D8".  TRACK "D8"."N\
003371  000                            .BYTE 0
003372  042     101     126     MS56:  .ASCII  \"AVAILABLE WAS NOT ASSERTED AFTER DISCONNECT"N\
003421  042     040     040            .ASCII  \"  STATE RECEIVED  "O16N\
003435  000                            .BYTE   0
003436  042     124     111     MS57:  .ASCII  \"TIME-OUT ON SEND OF RUN COMMAND TO DRIVE"N\
003463  000                            .BYTE   0
003464  042     124     111     MS58:  .ASCII  \"TIME-OUT ON RECEIVE OF RUN RESPONSE FROM DRIVE"N\
003514  000                            .BYTE   0
003515  042     122     125     MS59:  .ASCII  \"RUN COMMAND DID NOT RETURN EXPECTED RESPONSE"N\
003544  042     040     040            .ASCII  \"  EXPECTED RESPONSE  "D8N\
003561  042     040     040            .ASCII  \"  ACTUAL RESPONSE    "D8N\
003576  000                            .BYTE   0
003577  042     125     116     ER80:  .ASCII  \"UNABLE TO FIND ALL UNITS REQUESTED FOR TESTING"N\
003627  042     124     110            .ASCII  \"THE FOLLOWING IS VISIBLE ON THE 4 UDA PORTS"N\
003656  116     116     116            .ASCII  \NNN"UDA PORT 0 --"NN\
003670  122     061                    .ASCII  \R1\
003671  116     116     042            .ASCII  \NN"UDA PORT 1 --"NN\
003703  122     061                    .ASCII  \R1\
003704  116     116     042            .ASCII  \NN"UDA PORT 2 --"NN\
003715  122     061                    .ASCII  \R1\
003716  116     116     042            .ASCII  \NN"UDA PORT 3 --"NN\
003730  122     061                    .ASCII  \R1\
003731  000                            .BYTE   0
003732  042     116     117     SER10: .ASCII  \"NO VALID STATUS CAN BE RECEIVED ON THIS PORT"N\
003761  042     103     110            .ASCII  \"CHECK SDI CABLE AND POWER TO DRIVE"N\
004004  000                            .BYTE   0
004005  042     122     105     SER11: .ASCII  \"RECEIVER READY WAS NEVER ASSERTED AFTER INIT"N\
004034  042     122     105            .ASCII  \"REAL TIME DRIVE STATE IS: "H16N\
004054  000                            .BYTE   0
004055  042     125     104     SER12: .ASCII  \"UDA TIMED OUT ON SEND OF SDI COMMAND TO DRIVE"N\
004105  000                            .BYTE   0
004106  042     125     104     SER13: .ASCII  \"UDA TIMED OUT ON RECEIVE OF SDI COMMAND FROM DRIVE"N\
004140  000                            .BYTE   0
004141  042     125     104     SER14: .ASCII  \"UDA DID NOT SEE START FRAME ON SDI RESPONSE FROM DRIVE"N\
004175  000                            .BYTE   0
004176  042     125     104     SER15: .ASCII  \"UDA SAW AN SDI LEVEL 0 FRAMING ERROR ON SDI RESPONSE"N\
004231  042     106     122            .ASCII  \"FROM DRIVE"N\
004240  000                            .BYTE   0
004241  042     125     104     SER16: .ASCII  \"UDA SAW AN SDI LEVEL 0 CHECKSUM ERROR ON SDI RESPONSE"N\
004275  042     106     122            .ASCII  \"FROM DRIVE"N\
004303  000                            .BYTE   0
004304  042     124     110     SER17: .ASCII  \"THE UDA SAYS THE BUFFER SIZE FOR THE RESPONSE FROM THE"N\
004340  042     107     105            .ASCII  \"GET STATUS COMMAND IS INSUFFICIENT FOR THE RESPONSE"N\
004373  000                            .BYTE   0
004374  042     101     040     SER18: .ASCII  \"A UNIT WITH THE FOLLOWING SUBUNITS ARE VISIBLE ON THIS PORT:"N\
004433  122     061                    .ASCII  \R1\
```

```
    171 004434      000                             .BYTE   0
    172 004435      123     061     060     SER18A: .ASCII  \S10D6N\
    173 004440      123     061     060     SER18B: .ASCII  \S10D6N\
    174 004443      123     061     060     SER18C: .ASCII  \S10D6N\
    175 004446      123     061     060     SER18D: .ASCII  \S10D6N\
    176 004451      000                             .BYTE   0
    177
    178 004451                                      DMEND
        004452  056137                              .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
    179         000001                              .END
```

SYMBOL TABLE

```
ARGS$ = 000010        DRVCLR= 000005        HBLONB= 170377        LBHINB= 177417        MS54    003213
ATTN  = 000002        DRVID = 000004        HDRPRE= 000005        LBLONB= 177760        MS55    003301
AVAIL = 000100        DRVONL= 000213        HD.BAD= 110000        LBNCYL= 000000        MS56    003372
BEUSED= 000040        DRVRUN= 000014        HD.DBN= 140000        LBNHST= 000012        MS57    003436
BF.DAT= 000000        D.LIMT= 000001        HD.LBN= 000000        LBNTRK= 000011        MS58    003464
BF.ECC= 000401        D.SCHR= 000002        HD.PRV= 050000        LCDBN   004742        MS59    003515
BF.EDC= 000400        ECC   = 000015        HD.RBN= 060000        LDIACY  004744        MS6     000207
BLOCKG  004761        ECCCHK= 010000        HD.REV= 030000        LINKLN= 000007        MS7     000275
BLOCKT  004762        ECCFLG= 010000        HD.XBN= 120000        LOBYTE= 000377        MS8     000327
BREAK = 000000        ECCRSH= 000002        HEADER= 000002        LONG    004626        MS9     000364
BUFFLG= 040000        ECHO  = 000010        HIBYTE= 177400        LONGTO= 000001        MWR   = 000017
BUFSIZ= 000036        ECHOC = 000350        HICYL = 000001        LOW   = 000002        MWRITE  001170
CHAIN   005425        EOC   = 100000        HIDBN = 000003        LUNIT   001315        NEWSUB= 004000
CHECK = 000010        ERECOV= 000006        HILBN = 000002        MEDTYP= 000006        NOERR   004223
CHGMOD= 000201        ERHARD= 001000        HIMEM = 010000        MESSAG= 000015        NSCSL   004733
CHRRES= 000170        ERLEV = 000002        HIRBN = 000003        MICREV= 000003        NUMPTR= 000005
CLRBUF  000777        ERRMC = 000014        HIXBN = 000002        MOD     004662        OBUFF   005433
CMPDAT  004147        ERRMES= 000013        HOSTRQ  000751        MRD   = 000016        ONL     004640
CMPEXT  004230        ERRN  = 000101        HRDREV= 000003        MREAD   001152        ONLYCL= 000200
CMPLOP  004154        ERRORS  004647        ILOOP   002004        MS1     000000        OUT.RQ  001006
COMPAR= 000006        ERSOFT- 001400        INITW = 040000        MS10    000454        OUT.01  001007
COMPLT= 000176        ERTXT   001716        INR     004655        MS12    000500        OUT.02  001010
CR      004673        ER80    003577        INS     004656        MS13    000535        OUT.03  001011
CR.CLR  004556        EXIT  = 000021        INSEEK= 000012        MS14    000567        OUT.04  001012
CR.DIA  004563        FB.DAT= 000000        INTEDC= 000105        MS15    000624        OUT.05  001013
CR.DIS  004570        FB.EDC= 000400        IN.RQ   001044        MS16    000715        OUT.06  001014
CR.GCR  004575        FCARY   003464        IN.01   001045        MS17    000747        OUT.07  001015
CR.GST  004607        FCHAIN  006034        IN.02   001046        MS18    001004        OUT.08  001016
CR.INR  004626        FCTSIZ= 000010        IN.03   001047        MS19    001075        OUT.09  001017
CR.MOD  004614        FDIACY  004747        IN.04   001050        MS2     000045        OUT.10  001020
CR.ONL  004551        FGO     003517        IN.05   001051        MS20    001127        OUT.11  001021
CR.RUN  004633        FIRSTU= 007717        IN.06   001052        MS21    001164        OUT.12  001022
CR.SCR  004602        FLOOP   003456        IN.07   001053        MS24    001255        OUT.13  001023
CR.SEK  004621        FOREXT  003562        IN.08   001054        MS25    001303        OUT.14  001024
CURBLK  004763        FORMAT- 000001        IN.09   001055        MS26    001344        OUT.15  001025
CURCYL  004767        FORTRK  003444        IN.10   001056        MS27    001405        OUT.16  001026
CURGRP  004766        FRAME - 000004        IN.11   001057        MS28    001451        OUT.17  001027
CURPAT  004773        FSTOP = 100000        IN.12   001060        MS29    001550        OUT.18  001030
CURTRK  004772        FTIME = 001000        IN.13   001061        MS3     000071        OUT.19  001031
CVT   = 000020        FTLDEV= 000400        IN.14   001062        MS30    001612        OUT.20  001032
DATCMP- 000002        FTLSYS= 000000        IN.15   001063        MS31    001657        OUT.21  001033
DATERR= 000020        FT.BUF= 000000        IN.16   001064        MS32    001760        OUT.22  001034
DATPRE- 000005        FT.HI = 000001        IN.17   001065        MS35    002020        OUT.23  001035
DBNCYL= 000022        FT.LOW= 000002        IN.18   001066        MS36    002052        OUT.24  001036
DCMPAL- 000001        FXBNCY  004751        IN.19   001067        MS39    002131        OUT.25  001037
DCYLS - 020000        GCR     004650        IN.20   001070        MS4     000124        OUT.26  001040
DIA     004642        GENEXT  003671        IN.21   001071        MS40    002163        OUT.27  001041
DINIT = 000011        GENIN   003662        IN.22   001072        MS43    002320        OUT.28  001042
DIREC = 010000        GENOUT  003660        IN.23   001073        MS44    002351        OUT.29  001043
DIS     004644        GENPAT  003652        IN.24   001074        MS45    002505        OVERFL= 000002
DISCON= 000204        GETCHR= 000207        IN.25   001075        MS46    002645        OVE.MN= 000714
DMSDI   004732        GETSTA- 000011        IN.26   001076        MS47    002674        OVE.MS= 000000
DONE    000016        GETSUB= 000210        IN.27   001077        MS48    002726        OVL.MN= 005642 G
DONECD  001774        GRPCYL= 000002        IN.28   001100        MS5     000154        OVL.MS= 004453 G
DRC     004646        GRPOFF= 000011        IN.29   001101        MS51    003041        OVL...- 012315
DROP  - 100000        GST     004653        IRECLB- 000216        MS52    003124        OVSTRT= 007774
DRTYPE- 000007        HBHINB  007777        LARGE - 000001        MS53    003156        OVS.MN= 001040 G
```

SYMBOL TABLE

```
OVS.MS= 014544 G    REVS   = 000007    STATOK  000737    TRKLOP  003576    T8X     003353
OV...  = 012735     RGO      004100    STATUS= 000007    TRKTST  003563    T9      003353
PATPTR  006321      RM     = 000004    STSRES= 000366    TSTBLK  004754    T9X     003410
PAT1    006326      RONLY  = 004000    ST.C   = 000002   TSTCYL  004756    UDATA   001210
PAT10   006444      RREAL  = 013400    ST.CON= 000002    T0      002147    UNITNB  004547
PAT11   006465      RSTOP  = 100000    ST.DB  = 001000   T0A     002204    UNITS   004527
PAT12   006467      RTRIES= 001000     ST.DF  = 000020   T0B     002232    UNIT0  = 000001
PAT13   006510      RUN      004654    ST.DR  = 000040   T00     002074    UNIT1  = 000002
PAT14   006531      RWRDY  = 100000    ST.ERR= 000002    T03     002332    UNIT2  = 000004
PAT15   006534      RW.ANG= 000006     ST.FE  = 000200   T1      002232    UNIT3  = 000010
PAT2    006330      RW.BUF= 000001     ST.FO  = 002000   T1MSIZ= 000000    UNSSUC= 000175
PAT3    006332      RW.CMD= 000004     ST.MOD= 000001    T10     003410    UREAD  = 000013
PAT4    006334      RW.HI  = 000003    ST.MSK= 000000    T10LOP  003411    UTOTST= 000012
PAT5    006355      RW.LOW= 000002     ST.OA  = 000200   T10X    003443    UWRITE= 000014
PAT6    006376      RW.SDI= 000005     ST.PE  = 000040   T2      002275    U.CBN  = 000037
PAT7    006417      RW.STA= 000000     ST.PS  = 000002   T2CMD  = 000002   U.CCNT= 000012
PAT8    006421      SAVREG  001316     ST.RE  = 000100   T2DLL  = 000001   U.CCOP= 000044
PAT9    006442      SBCRES= 000167     ST.RR  = 000100   T3      002367    U.CCYL= 000053
PORT2   001723      SCR      004651    ST.RTY= 000003    T3A     002424    U.CGRP= 000055
PORT3   001753      SCTWRD= 000377     ST.RU  = 000001   T3B     002426    U.COPY= 000043
PORT4   001761      SDI      004550    ST.SR  = 000020   T3C     002430    U.CSEC= 000021
PORT5   001770      SDILTO  001151     ST.STA= 000001    T3D     002456    U.CTRK= 000015
PRMS   - 000007     SDISTO  001150     ST.S7  = 000400   T3STRT  001722    U.ECCT= 000027
PTYPE$= 000001      SDIVER= 000000     ST.UNT= 000000    T31     002543    U.ELEV= 000024
RBNBN  = 000200     SECCNT  004774     ST.WE  = 000010   T4A     002471    U.LCYL= 000056
RBNTRK= 000004      SECGRP  004765     SUB    = 004706   T4A1    002561    U.LGRP= 000060
RBUFD   005003      SECTRK  004753     SUBUNT  004652    T4B     002613    U.MASK= 000022
RBUFLN= 000415      SEEK     004231    S.BADP= 000010    T4BB1  = 000005   U.MBN  = 000035
RBUF0   004775      SEEKA    004245    S.BESS= 000011    T4BB2  = 000006   U.MLEV= 000026
RCLB   = 000001     SEEK1    004313    S.MCNT= 000011    T4C     002631    U.MSEC= 000017
RCON1  = 000000     SEEK2    004314    S.MEGR= 000006    T4D     002635    U.NEXT= 000000
RCTCPS= 000001      SEEK3    004360    S.MEGW= 000007    T4MPRM= 000003    U.NFUN= 000010
RCTCSZ= 000014      SEKINP= 002000     S.PARM= 000000    T4MXFR= 000011    U.NSEC= 000016
RCV    = 000005     SEND   = 000004    S.PAT  = 000003   T4SEEK= 000010    U.PARM= 000033
RCVERR= 000004      SEQSEK= 000100     S.SCHR= 000005    T4SOFT= 000007    U.PAT  = 000011
RCVRDY= 000001      SER10    003732    S.SDCL= 000001    T4UPRM= 000004    U.PCTG= 000014
RDSTAT  000720      SER11    004005    S.TGOF= 000013    T6      002657    U.RBN  = 000041
READB   004027      SER12    004055    S.TGSS= 000015    T6A     002666    U.RTRY= 000025
READ1   004364      SER13    004106    S.TRKL= 000004    T6C     002721    U.RVER= 000046
READ1A  004404      SER14    004141    T       002034    T6D     002725    U.RWER= 000045
READ1B  004455      SER15    004176    TALKER  001102    T6E     002762    U.RWTO= 000006
READ1C  004435      SER16    004241    TALK1A  001114    T6E1    003001    U.SDIL= 000031
READ1E  004456      SER17    004304    TALK1B  001124    T6F     003021    U.SDIS= 000030
READ1X  004521      SER18    004374    TALK2A  001144    T6X     003022    U.SEEK= 000007
REDEXT  004146      SER18A   004435    TALK2B  001145    T7      003035    U.SNUM= 000047
REDWRT= 000100      SER18B   004440    TEST    002000    T7A     003046    U.SUBP= 000001
REGS$  = 177777     SER18C   004443    TESTEX  003443    T7C     003104    U.SUBU= 000034
REGU$  = 000001     SER18D   004446    TESTX   001764    T7C1    003140    U.TIMO= 000005
RERRCA  001301      SHRTTO= 000000     TEST4  = 000000   T7D     003153    U.TSEC= 000020
RERROR  001236      SNDAGN  000756     TIMEOU= 000001    T7E     003214    U.WPRT= 000032
RERRPA  001275      SPADJU  001312     TLEN.U= 000061    T7G     003222    U.WRIT= 000023
RESEEK= 020000      SS     = 000001    TO      001221    T7H     003236    WAITSI= 000012
RETRY  = 000004     ST       004664    TOOBIG= 000001    T7X     003237    WBUFLN= 000401
RETS   = 000001     STACK    001357    TRACKS= 000020    T8      003237    WCHECK= 000010
REVEC  = 000400     START    001360    TRKBOT  003620    T8A     003261    WCHKAL= 000004
REVECT= 000004      STATEX   000746    TRKEXT  003624    T8B     003301    WCONT  = 040000
REVINP= 000040      STATLP   000724    TRKGRP= 000003    T8D     003333    WGO     003754
```

```
UDAT3 DISK FUNCTIONAL MACRO X04.00 9-JUL-81 01:13:55 PAGE 65-6                    SEQ 0262
SYMBOL TABLE

WONLY = 002000          WRONG = 000002          WTRCMP  003625          XFERRT= 000000          XREAD = 000002
WREAL = 122400          WRTEXT  004026          WTREXT  003651          XMTERR= 000400          XWRITE= 000003
WRITEB  003673          WSTOP = 140000          XBNCYL= 000021          XOR     001211

. ABS.  025672      000
        000000      001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  4924 WORDS  ( 20 PAGES)
DYNAMIC MEMORY AVAILABLE FOR  69 PAGES
A:UDAT3.BIN,SY:UDAT3/C=[1,33]DMACRO,UDAT3T,UDATP,UDATR,UDAT3M,UDAT3S
```

CROSS REFERENCE TABLE (CREF V01-05)

```
ARGS$   37-18  37-18  37-18#  37-18#  37-23  37-23#  38-5   38-5   38-5#  38-5#  38-8   38-8#  39-4   39-4
        39-4#  39-4#  39-7    39-7#   40-4   40-4    40-4#  40-4#  40-7   40-7#  40-9   40-9   40-9#  40-9#
        40-12  40-12#  41-7    41-7    41-7#  41-7#   41-10  41-10#  42-4   42-4   42-4#  42-4#  42-7   42-7#
        43-3   43-3   43-3#   43-3#   43-6   43-6#   44-4   44-4   44-4#  44-4#  44-7   44-7#  44-11  44-11
        44-11#  44-11#  44-14   44-14#  44-23  44-23#  45-12  45-12  45-12#  45-12#  45-15  45-15#  47-36  47-36#
        51-2   51-2   51-2#   51-2#   51-5   51-5#   52-2   52-2   52-2#  52-2#  52-8   52-8#  54-21  54-21
        54-21#  54-21#  54-24   54-24#  54-36  54-36#  58-16  58-16  58-16#  58-16#  58-21  58-21#  58-34  58-34#
        59-12  59-12  59-12#  59-12#  59-17  59-17#  59-28  59-28#  60-14  60-14#  61-17  61-17  61-17#  61-17#
        61-20  61-20#  61-23   61-23   61-23#  61-23#  61-28  61-28#  62-26  62-26  62-26#  62-26#  62-29  62-29#
        62-53  62-53#
ATTN     6-22#  40-10  44-12
AVAIL    6-24#  52-3
BEUSED   8-35#
BF.DAT   3-62#
BF.ECC   3-64#
BF.EDC   3-63#
BLOCKG  49-16*  49-74*  49-81   63-73#
BLOCKT  49-28*  49-31   49-72*  63-74#
BREAK    4-3#
BUFFLG   3-47#
BUFSIZ  22-15   22-20   23-68#
CHAIN   58-6*   58-8*   58-10*  58-13*  58-15*  58-25   59-6*  59-9*  59-11*  59-21  63-98#
CHECK    5-67#
CHGMOD   6-32#
CHRRES   6-43#  38-6    38-8    38-8
CLRBUF  22-25#  22-27
CMPDAT  56-16   60-1#
CMPEXT  60-13   60-15   60-20#
CMPLOP  60-9#   60-18
COMPAR   4-9#
COMPLT   6-41#  39-5    39-7    39-7    41-8    41-10   41-10  42-5   42-7   42-7   43-4   43-6   43-6   44-5
        44-7   44-7   51-3    51-5    51-5    61-18   61-20  61-20
CR      32-3   38-11   38-15   63-20   63-55#
CR.CLR  41-6   63-17#
CR.DIA  63-18#
CR.DIS  51-1   63-19#
CR.GCR  38-4   63-20#
CR.GST  32-31  40-3    63-22#
CR.INR  44-3   63-26#
CR.MOD  42-3   63-23#
CR.ONL  39-3   63-16#
CR.RUN  43-2   63-27#
CR.SCR  45-11  63-21#
CR.SEK  61-16  63-24#
CURBLK  48-43*  48-44*  48-48   48-57*  48-59*  55-7*   55-9*  55-17  55-19*  55-21  55-23*  58-7   58-9   58-34
        58-34  58-34   58-34   59-28   59-28   59-28   59-28  63-75#
CURCYL  46-4*  46-5*   46-8*   46-9    63-78#
CURGRP  48-47*  48-61*  48-63*  48-65   63-77#
CURPAT  56-7*  56-19   63-80#
CURTRK  50-20*  50-30   54-32   54-36   54-36   58-11   58-34  58-34  59-7   59-28  59-28  60-14  60-14  63-79#
CVT      4-19#
D.LIMT   8-4#
D.SCHR   8-5#
DATCMP   8-40#
DATERR   8-21#
DATPRE   5-33#  54-27   58-26
```

CROSS REFERENCE TABLE (CREF V01-05)

```
DBNCYL    5-42#   47-32    47-46    48-18    48-26    48-71
DCMPAL    8-42#
DCYLS     8-27#
DIA      63-18    63-33#
DINIT     4-12#   32-16    37-12
DIREC     8-13#
DIS      63-19    63-35#
DISCON    6-30#
DMSDI    63-57#   63-92    63-104
DONE      6-17#   35-45    36-40
DONECD   36-40#   36-42
DRC      63-17    63-37#
DROP      8-10#
DRTYPE    5-15#
DRVCLR    6-35#   63-37
DRVID     5-14#
DRVONL    6-33#
DRVRUN    6-34#
ECC       4-16#
ECCCHK    8-28#
ECCFLG    3-45#
ECCRSH    5-11#
ECHO      4-11#
ECHOC     6-46#
EOC       3-46#
ERBO     35-39    65-137#
ERECOV    6-31#
ERHARD    6-52#
ERLEV     5-10#
ERRMC     6-15#   35-39    35-43
ERRMES    6-14#   29-37
ERRN     32-4#    37-18    37-18    37-18    37-18    37-18    37-18    37-18#   37-18#   37-23    37-23    37-23    37-23#   38-5
         38-5     38-5     38-5     38-5     38-5     38-5#    38-5#    38-8     38-8     38-8     38-8#    39-4     39-4     39-4
         39-4     39-4     39-4     39-4#    39-4#    39-7     39-7     39-7     39-7#    40-4     40-4     40-4     40-4     40-4
         40-4     40-4#    40-4     40-7     40-7     40-7     40-7#    40-9     40-9     40-9     40-9     40-9     40-9     40-9#
         40-9#    40-12    40-12    40-12    40-12#   41-7     41-7     41-7     41-7     41-7     41-7     41-7#    41-7#    41-10
         41-10    41-10    41-10#   42-4     42-4     42-4     42-4     42-4     42-4     42-4#    42-4#    42-7     42-7     42-7
         42-7#    43-3     43-3     43-3     43-3     43-3     43-3     43-3#    43-3#    43-6     43-6     43-6     43-6#    44-4
         44-4     44-4     44-4     44-4     44-4     44-4#    44-4#    44-7     44-7     44-7     44-7#    44-11    44-11    44-11
         44-11    44-11    44-11    44-11#   44-11#   44-14    44-14    44-14    44-14#   44-23    44-23    44-23    44-23#   45-12
         45-12    45-12    45-12    45-12    45-12    45-12#   45-12#   45-15    45-15    45-15    45-15#   47-36    47-36    47-36
         47-36#   51-2     51-2     51-2     51-2     51-2     51-2     51-2#    51-2#    51-5     51-5     51-5     51-5#    52-2
         52-2     52-2     52-2     52-2     52-2     52-2#    52-2#    52-8     52-8     52-8     52-8#    54-21    54-21    54-21
         54-21    54-21    54-21    54-21#   54-21#   54-24    54-24    54-24    54-24#   54-36    54-36    54-36    54-36#   58-16
         58-16    58-16    58-16    58-16    58-16    58-16#   58-16#   58-21    58-21    58-21    58-21#   58-34    58-34    58-34
         58-34#   59-12    59-12    59-12    59-12    59-12    59-12    59-12#   59-12#   59-17    59-17    59-17    59-17#   59-28
         59-28    59-28    59-28#   60-14    60-14    60-14    60-14#   61-17    61-17    61-17    61-17    61-17    61-17    61-17#
         61-17#   61-20    61-20    61-20    61-20#   61-23    61-23    61-23    61-23    61-23    61-23    61-23#   61-23#   61-28
         61-28    61-28    61-28#   62-26    62-26    62-26    62-26    62-26    62-26    62-26#   62-26#   62-29    62-29    62-29
         62-29#   62-53    62-53    62-53    62-53#
ERRORS   41-5*    63-38#
ERSOFT    6-53#
ERTXT    35-27    35-54#
EXIT      4-20#   35-47
FB.DAT    3-22#
FB.EDC    3-23#
FCARY    54-15    54-17#
```

CROSS REFERENCE TABLE (CREF V01-05)

```
FCHAIN  54-10    54-26    63-111#
FCTSIZ   5-36#
FDIACY  47-27*   47-28*   49-76    49-78    63-67#
FGO     54-23    54-26#
FIRSTU   7-11#
FLOOP   54-11#   54-18
FOREXT  54-35    54-37#
FORMAT   4-4#    54-33
FORWRK  50-21    54-1#
FRAME    5-66#
FSTOP    3-42#   54-19
FT.BUF   3-15#
FT.HI    3-16#
FT.LOW   3-17#
FTIME    8-16#
FTLDEV   6-51#   35-39
FTLSYS   6-50#
FXBNCY  47-18*   47-20*   49-57    49-59    63-68#
GCR     63-20    63-39#
GENEXT  57-17    57-21#
GENIN   57-14#   57-19
GENOUT  57-12#   57-20
GENPAT  56-9     57-1#
GETCHR   6-36#   63-39
GETSTA   6-38#   63-42
GETSUB   6-37#   63-40
GRPCYL   5-25#   47-44    48-16    48-62    48-73    49-42    50-38
GRPOFF   5-38#
GST     63-22    63-42#
HBHINB   5-56#   33-10    34-11    46-7     49-19
HBLONB   5-57#   48-40    49-24    49-68    50-17
HD.BAD   3-55#
HD.DBN   3-58#   48-42    50-18
HD.LBN   3-52#
HD.PRV   3-56#
HD.RBN   3-53#
HD.REV   3-54#
HD.XBN   3-57#   49-69
HDRPRE   5-34#   54-29
HEADER   5-62#
HIBYTE   5-54#   41-4     47-7     47-34    47-45    47-47    48-8     48-17    48-20    48-27    48-54    48-72    48-74    49-10
        49-12    49-27    49-43    50-33    50-39    54-28    54-31    58-27
HICYL    5-24#   49-18
HIDBN    5-30#   48-35    50-12
HILBN    5-26#   49-23
HIMEM    3-9#     7-11
HIRBN    5-29#
HIXBN    5-27#   49-63
HOSTRQ  22-2#    29-38    34-5     35-44    35-46    36-41
HRDREV   5-13#
ILOOP   37-18#   37-22
IN.01   23-39#   34-6
IN.02   23-40#
IN.03   23-41#
IN.04   23-42#
IN.05   23-43#
IN.06   23-44#
```

CROSS REFERENCE TABLE (CREF V01-05)

```
IN.07   23-45#
IN.08   23-46#
IN.09   23-47#
IN.10   23-48#
IN.11   23-49#
IN.12   23-50#
IN.13   23-51#
IN.14   23-52#
IN.15   23-53#
IN.16   23-54#
IN.17   23-55#
IN.18   23-56#
IN.19   23-57#
IN.20   23-58#
IN.21   23-59#
IN.22   23-60#
IN.23   23-61#
IN.24   23-62#
IN.25   23-63#
IN.26   23-64#
IN.27   23-65#
IN.28   23-66#
IN.29   23-67#
IN.RQ   22-19    23-38#   23-68
INITW    8-11#
INR     63-26    63-44#
INS     54-36    54-36    54-36    54-36    54-36    54-36    58-34    58-34    58-34    58-34    58-34    58-34    59-28    59-28
        59-28    59-28    59-28    59-28    60-14    60-14    60-14    60-14    60-14    60-14    61-11*   61-13*   61-15*   61-20
        61-20    61-20    61-20    61-20    61-20    61-28    61-28    61-28    61-28    61-28    61-28    62-53    62-53    62-53
        62-53    62-53    62-53    63-24    63-45#
INSEEK   6-40#   63-45
INTEDC   7-9#
IRECLB   6-39#   63-44
LARGE    5-74#
LBHINB   5-58#   45-9
LBLONB   5-59#   33-17    36-25    38-12    38-16
LBNCYL   5-23#   46-6     47-16
LBNHST   5-39#
LBNTRK   5-37#   47-6     49-11    49-26
LCDBN   48-45*   48-46*   49-35    63-65#
LDIACY  47-42*   47-43*   47-49*   47-50    48-47    48-65*   48-66    48-76*   49-37    50-46*   50-48*   50-50*   63-66#
LINKLN   3-70#
LOBYTE   5-55#
LONG    24-23    63-25#
LONGTO   5-6#    38-15
LOW      5-70#
LUNIT   29-20    29-44#   36-21*
MEDTYP   5-35#
MESSAG   6-16#
MICREV   5-12#
MOD     63-23    63-49#
MRD      4-17#   22-16
MREAD   25-10#
MS1     37-18    52-2     65-7#
MS10    40-9     44-11    54-21    58-16    59-12    61-23    62-26    65-30#
MS12    40-12    65-32#
MS13    41-7     65-34#
```

CROSS REFERENCE TABLE (CREF V01-05)

```
MS14     41-7     65-36#
MS15     41-10    65-38#
MS16     42-4     65-42#
MS17     42-4     65-44#
MS18     42-7     65-46#
MS19     44-4     65-50#
MS2      37-18    40-9     44-11    52-2     54-21    58-16    59-12    61-23    62-26    65-10#
MS20     44-4     65-52#
MS21     44-7     65-54#
MS24     44-14    65-58#
MS25     44-23    65-60#
MS26     38-5     65-62#
MS27     38-5     65-64#
MS28     38-8     65-66#
MS29     45-12    65-70#
MS3      37-23    65-12#
MS30     45-12    65-72#
MS31     45-15    65-74#
MS32     47-36    65-78#
MS35     54-24    65-80#
MS36     54-36    65-82#
MS39     58-21    65-85#
MS4      39-4     65-14#
MS40     58-34    65-87#
MS43     59-17    62-29    65-91#
MS44     59-28    65-93#
MS45     60-14    65-97#
MS46     61-17    65-103#
MS47     61-17    65-105#
MS48     61-20    65-107#
MS5      39-4     65-16#
MS51     61-28    65-112#
MS52     51-2     65-115#
MS53     51-2     65-117#
MS54     51-5     65-119#
MS55     62-53    65-123#
MS56     52-8     65-126#
MS57     43-3     65-129#
MS58     43-3     65-131#
MS59     43-6     65-133#
MS6      39-7     65-18#
MS7      40-4     65-22#
MS8      40-4     65-24#
MS9      40-7     65-26#
MWR       4-18#   22-21
MWRITE   26-10#
NEWSUB    8-14#
NOERR    60-11    60-16#
NSCSL    47-11*   63-58#
NUMPTR   35-39#
OBUFF    54-8     57-11    58-14    60-6     63-106#
ONL      63-16    63-31#
ONLYCL    8-32#
OUT.01   22-22    22-23    23-6#    29-13    35-39*
OUT.02   23-7#    35-39*   35-39*
OUT.03   23-8#    35-42*
OUT.04   23-9#    35-39*
```

CROSS REFERENCE TABLE (CREF V01-05)

```
OUT.05   23-10#   35-6
OUT.06   23-11#
OUT.07   23-12#
OUT.08   23-13#
OUT.09   23-14#
OUT.10   23-15#
OUT.11   23-16#
OUT.12   23-17#
OUT.13   23-18#
OUT.14   23-19#
OUT.15   23-20#
OUT.16   23-21#
OUT.17   23-22#
OUT.18   23-23#
OUT.19   23-24#
OUT.20   23-25#
OUT.21   23-26#
OUT.22   23-27#
OUT.23   23-28#
OUT.24   23-29#
OUT.25   23-30#
OUT.26   23-31#
OUT.27   23-32#
OUT.28   23-33#
OUT.29   22-23    23-34#
OUT.RQ   22-13*   22-14    23-5#    35-39*
OV...     2-39     2-39     2-39#   65-4     65-4     65-4     65-4#    65-178   65-178#
OVE.MN    2-39#   65-4
OVE.MS   65-4#    65-178
OVERFL    5-75#
OVL...    2-39     2-39#   65-4     65-4#    65-178   65-178#
OVL.MN    2-39     2-39    65-4     65-4     65-4#
OVL.MS   65-178   65-178   65-178#
OVS.MN    2-39#
OVS.MS   65-4#
OVSTRT    3-10#
PAT1     64-9#
PAT10    64-87#
PAT11    64-104#
PAT12    64-106#
PAT13    64-123#
PAT14    64-140#
PAT15    64-143#
PAT2     64-11#
PAT3     64-13#
PAT4     64-2     64-15#
PAT5     64-3     64-32#
PAT6     64-4     64-49#
PAT7     64-66#
PAT8     64-5     64-68#
PAT9     64-85#
PATPTR   56-8     56-21    64-1#
PORT2    36-10#   36-39
PORT3    36-26#   36-30
PORT4    36-27    36-31#
PORT5    36-15    36-17    36-19    36-37#
PRMS     37-18    37-18    37-18    37-18    37-18    37-18    37-18    37-18    37-18    37-18    37-18    37-18    37-18    37-18    37-18
```

CROSS REFERENCE TABLE (CREF V01-05)

37-18	37-18	37-18	37-18	37-18	37-18	37-18#	37-18#	37-23	37-23	37-23	37-23	37-23	37-23
37-23	37-23	37-23	37-23	37-23#	38-5	38-5	38-5	38-5	38-5	38-5	38-5	38-5	38-5
38-5	38-5	38-5	38-5	38-5	38-5	38-5	38-5	38-5	38-5	38-5	38-5#	38-5#	38-8
38-8	38-8	38-8	38-8	38-8	38-8	38-8	38-8	38-8	38-8#	39-4	39-4	39-4	39-4
39-4	39-4	39-4	39-4	39-4	39-4	39-4	39-4	39-4	39-4	39-4	39-4	39-4	39-4
39-4	39-4	39-4#	39-4#	39-7	39-7	39-7	39-7	39-7	39-7	39-7	39-7	39-7	39-7
39-7#	40-4	40-4	40-4	40-4	40-4	40-4	40-4	40-4	40-4	40-4	40-4	40-4	40-4
40-4	40-4	40-4	40-4	40-4	40-4	40-4	40-4#	40-4#	40-7	40-7	40-7	40-7	40-7
40-7	40-7	40-7	40-7	40-7	40-7#	40-9	40-9	40-9	40-9	40-9	40-9	40-9	40-9
40-9	40-9	40-9	40-9	40-9	40-9	40-9	40-9	40-9	40-9	40-9	40-9	40-9#	40-9#
40-12	40-12	40-12	40-12	40-12	40-12	40-12	40-12	40-12	40-12	40-12#	41-7	41-7	41-7
41-7	41-7	41-7	41-7	41-7	41-7	41-7	41-7	41-7	41-7	41-7	41-7	41-7	41-7
41-7	41-7	41-7	41-7#	41-7#	41-10	41-10	41-10	41-10	41-10	41-10	41-10	41-10	41-10
41-10	41-10#	42-4	42-4	42-4	42-4	42-4	42-4	42-4	42-4	42-4	42-4	42-4	42-4
42-4	42-4	42-4	42-4	42-4	42-4	42-4	42-4	42-4#	42-4#	42-7	42-7	42-7	42-7
42-7	42-7	42-7	42-7	42-7	42-7	42-7#	43-3	43-3	43-3	43-3	43-3	43-3	43-3
43-3	43-3	43-3	43-3	43-3	43-3	43-3	43-3	43-3	43-3	43-3	43-3	43-3	43-3#
43-3#	43-6	43-6	43-6	43-6	43-6	43-6	43-6	43-6	43-6	43-6	43-6#	44-4	44-4
44-4	44-4	44-4	44-4	44-4	44-4	44-4	44-4	44-4	44-4	44-4	44-4	44-4	44-4
44-4	44-4	44-4	44-4	44-4#	44-4#	44-7	44-7	44-7	44-7	44-7	44-7	44-7	44-7
44-7	44-7	44-7#	44-11	44-11	44-11	44-11	44-11	44-11	44-11	44-11	44-11	44-11	44-11
44-11	44-11	44-11	44-11	44-11	44-11	44-11	44-11	44-11	44-11#	44-11#	44-14	44-14	44-14
44-14	44-14	44-14	44-14	44-14	44-14	44-14	44-14#	44-23	44-23	44-23	44-23	44-23	44-23
44-23	44-23	44-23	44-23	44-23#	45-12	45-12	45-12	45-12	45-12	45-12	45-12	45-12	45-12
45-12	45-12	45-12	45-12	45-12	45-12	45-12	45-12	45-12	45-12	45-12	45-12#	45-12#	45-15
45-15	45-15	45-15	45-15	45-15	45-15	45-15	45-15	45-15	45-15#	47-36	47-36	47-36	47-36
47-36	47-36	47-36	47-36	47-36	47-36	47-36#	51-2	51-2	51-2	51-2	51-2	51-2	51-2
51-2	51-2	51-2	51-2	51-2	51-2	51-2	51-2	51-2	51-2	51-2	51-2	51-2	51-2#
51-2#	51-5	51-5	51-5	51-5	51-5	51-5	51-5	51-5	51-5	51-5	51-5#	52-2	52-2
52-2	52-2	52-2	52-2	52-2	52-2	52-2	52-2	52-2	52-2	52-2	52-2	52-2	52-2
52-2	52-2	52-2	52-2	52-2#	52-2#	52-8	52-8	52-8	52-8	52-8	52-8	52-8	52-8
52-8	52-8	52-8#	54-21	54-21	54-21	54-21	54-21	54-21	54-21	54-21	54-21	54-21	54-21
54-21	54-21	54-21	54-21	54-21	54-21	54-21	54-21	54-21	54-21#	54-21#	54-24	54-24	54-24
54-24	54-24	54-24	54-24	54-24	54-24	54-24	54-24#	54-36	54-36	54-36	54-36	54-36	54-36
54-36	54-36	54-36	54-36	54-36#	58-16	58-16	58-16	58-16	58-16	58-16	58-16	58-16	58-16
58-16	58-16	58-16	58-16	58-16	58-16	58-16	58-16	58-16	58-16	58-16	58-16#	58-16#	58-21
58-21	58-21	58-21	58-21	58-21	58-21	58-21	58-21	58-21	58-21#	58-34	58-34	58-34	58-34
58-34	58-34	58-34	58-34	58-34	58-34	58-34#	59-12	59-12	59-12	59-12	59-12	59-12	59-12
59-12	59-12	59-12	59-12	59-12	59-12	59-12	59-12	59-12	59-12	59-12	59-12	59-12	59-12#
59-12#	59-17	59-17	59-17	59-17	59-17	59-17	59-17	59-17	59-17	59-17	59-17#	59-28	59-28
59-28	59-28	59-28	59-28	59-28	59-28	59-28	59-28	59-28#	60-14	60-14	60-14	60-14	60-14
60-14	60-14	60-14	60-14	60-14	60-14#	61-17	61-17	61-17	61-17	61-17	61-17	61-17	61-17
61-17	61-17	61-17	61-17	61-17	61-17	61-17	61-17	61-17	61-17	61-17	61-17	61-17#	61-17#
61-20	61-20	61-20	61-20	61-20	61-20	61-20	61-20	61-20	61-20	61-20#	61-23	61-23	61-23
61-23	61-23	61-23	61-23	61-23	61-23	61-23	61-23	61-23	61-23	61-23	61-23	61-23	61-23
61-23	61-23	61-23	61-23#	61-23#	61-28	61-28	61-28	61-28	61-28	61-28	61-28	61-28	61-28
61-28	61-28#	62-26	62-26	62-26	62-26	62-26	62-26	62-26	62-26	62-26	62-26	62-26	62-26
62-26	62-26	62-26	62-26	62-26	62-26	62-26	62-26	62-26#	62-26#	62-29	62-29	62-29	62-29
62-29	62-29	62-29	62-29	62-29	62-29	62-29#	62-53	62-53	62-53	62-53	62-53	62-53	62-53
62-53	62-53	62-53	62-53#										
PTYPE$ 38-8	38-8	38-8	38-8	38-8#	38-8#	39-7	39-7	39-7	39-7	39-7#	39-7#	40-7	40-7
40-7	40-7	40-7#	40-7#	41-10	41-10	41-10	41-10	41-10#	41-10#	42-7	42-7	42-7	42-7
42-7#	42-7#	43-6	43-6	43-6	43-6	43-6#	43-6#	44-7	44-7	44-7	44-7	44-7#	44-7#
45-15	45-15	45-15	45-15	45-15#	45-15#	51-5	51-5	51-5	51-5	51-5#	51-5#	52-8	52-8
52-8#	54-36	54-36	54-36	54-36	54-36	54-36	54-36	54-36	54-36#	54-36#	54-36#	54-36#	58-34
58-34	58-34	58-34	58-34	58-34	58-34	58-34	58-34	58-34	58-34	58-34	58-34	58-34	58-34#
58-34#	58-34#	58-34#	58-34#	58-34#	58-34#	59-28	59-28	59-28	59-28	59-28	59-28	59-28	59-28

CROSS REFERENCE TABLE (CREF V01-05)

	59-28	59-28	59-28	59-28	59-28	59-28	59-28#	59-28#	59-28#	59-28#	59-28#	59-28#	59-28#	60-14
	60-14	60-14	60-14	60-14	60-14	60-14	60-14	60-14	60-14	60-14	60-14	60-14	60-14	60-14#
	60-14#	60-14#	60-14#	60-14#	60-14#	60-14#	61-20	61-20	61-20	61-20	61-20	61-20	61-20	61-20
	61-20	61-20	61-20#	61-20#	61-20#	61-20#	61-20#	61-28	61-28	61-28	61-28	61-28	61-28	61-28#
	61-28#	61-28#	62-53	62-53	62-53	62-53	62-53	62-53	62-53	62-53	62-53	62-53	62-53	62-53
	62-53	62-53	62-53#	62-53#	62-53#	62-53#	62-53#	62-53#	62-53#					
RBNBN	8-18#													
RBNTRK	5-31#	47-4												
RBUF0	62-15	62-24	62-38	62-48	62-53	62-53	62-53	62-53	62-53	62-53	63-87#			
RBUFD	59-10	60-7	63-89	63-95#										
RBUFLN	3-69#													
RCLB	8-23#													
RCONT	3-41#													
RCTCPS	5-8#													
RCTCSZ	5-40#													
RCV	4-8#	24-30	32-45											
RCVERR	6-23#	21-17												
RCVRDY	6-21#	32-23	37-19											
RDSTAT	21-2#	32-18	37-18	40-9	44-11	52-2	54-21	58-16	59-12	61-23	62-26			
READ1	48-51	49-34	49-39	62-14#										
READ1A	62-24#	62-46												
READ1B	62-42	62-46#												
READ1C	62-28	62-31#												
READ1E	62-37	62-47#												
READ1X	62-35	62-54#												
READB	56-13	59-1#												
REDEXT	59-16	59-19	59-25	59-27	59-30#									
REDWRT	8-19#													
REGS$	37-18	37-18	37-18#	37-18#	37-23	37-23#	38-5	38-5	38-5#	38-5#	38-8	38-8	38-8	38-8
	38-8	38-8	38-8#	38-8#	38-8#	39-4	39-4	39-4#	39-4#	39-7	39-7	39-7	39-7	39-7
	39-7	39-7#	39-7#	39-7#	40-4	40-4	40-4#	40-4#	40-7	40-7	40-7	40-7	40-7	40-7
	40-7#	40-7#	40-7#	40-9	40-9	40-9#	40-9#	40-12	40-12#	41-7	41-7	41-7#	41-7#	41-10
	41-10	41-10	41-10	41-10	41-10	41-10#	41-10#	41-10#	42-4	42-4	42-4#	42-4#	42-7	42-7
	42-7	42-7	42-7	42-7	42-7#	42-7#	42-7#	43-3	43-3	43-3#	43-3#	43-6	43-6	43-6
	43-6	43-6	43-6	43-6#	43-6#	43-6#	44-4	44-4	44-4#	44-4#	44-7	44-7	44-7	44-7
	44-7	44-7	44-7#	44-7#	44-7#	44-11	44-11	44-11#	44-11#	44-14	44-14#	44-23	44-23#	45-12
	45-12	45-12#	45-12#	45-15	45-15	45-15	45-15	45-15	45-15	45-15#	45-15#	45-15#	47-36	47-36#
	51-2	51-2	51-2#	51-2#	51-5	51-5	51-5	51-5	51-5	51-5	51-5#	51-5#	51-5#	52-2
	52-2	52-2#	52-2#	52-8	52-8	52-8#	54-21	54-21	54-21#	54-21#	54-24	54-24#	54-36	54-36
	54-36	54-36	54-36	54-36	54-36	54-36	54-36	54-36	54-36	54-36#	54-36#	54-36#	58-16	58-16
	58-16#	58-16#	58-21	58-21#	58-34	58-34	58-34	58-34	58-34	58-34	58-34	58-34	58-34	58-34
	58-34	58-34	58-34	58-34	58-34	58-34	58-34#	58-34#	58-34#	59-12	59-12	59-12#	59-12#	59-17
	59-17#	59-28	59-28	59-28	59-28	59-28	59-28	59-28	59-28	59-28	59-28	59-28	59-28	59-28
	59-28	59-28	59-28	59-28#	59-28#	59-28#	60-14	60-14	60-14	60-14	60-14	60-14	60-14	60-14
	60-14	60-14	60-14	60-14	60-14	60-14	60-14	60-14	60-14	60-14	60-14	60-14	60-14#	60-14#
	60-14#	60-14#	60-14#	60-14#	60-14#	61-17	61-17	61-17#	61-17#	61-20	61-20	61-20	61-20	61-20
	61-20	61-20	61-20	61-20	61-20	61-20	61-20	61-20#	61-20#	61-20#	61-23	61-23	61-23#	61-23#
	61-28	61-28	61-28	61-28	61-28	61-28	61-28	61-28	61-28	61-28#	61-28#	61-28#	62-26	62-26
	62-26#	62-26#	62-29	62-29#	62-53	62-53	62-53	62-53	62-53	62-53	62-53	62-53	62-53	62-53
	62-53	62-53	62-53	62-53	62-53	62-53	62-53#	62-53#	62-53#					
REGU$	38-8	38-8	38-8#	39-7	39-7	39-7#	40-7	40-7	40-7#	41-10	41-10	41-10#	42-7	42-7
	42-7#	43-6	43-6	43-6#	44-7	44-7	44-7#	45-15	45-15	45-15#	51-5	51-5	51-5#	54-36
	54-36	54-36	54-36	54-36	54-36#	54-36#	54-36#	54-36#	58-34	58-34	58-34	58-34	58-34	58-34
	58-34	58-34#	58-34#	58-34#	58-34#	58-34#	58-34#	59-28	59-28	59-28	59-28	59-28	59-28	59-28
	59-28#	59-28#	59-28#	59-28#	59-28#	59-28#	60-14	60-14	60-14	60-14	60-14	60-14	60-14	60-14
	60-14	60-14#	60-14#	60-14#	60-14#	60-14#	60-14#	61-20	61-20	61-20	61-20	61-20	61-20#	61-20#
	61-20#	61-20#	61-28	61-28	61-28	61-28	61-28#	61-28#	61-28#	62-53	62-53	62-53	62-53	62-53

CROSS REFERENCE TABLE (CREF V01-05)

```
        62-53   62-53   62-53#  62-53#  62-53#  62-53#  62-53#  62-53#
RERRCA  29-31   29-36#
RERROR  29-7#   37-18   37-18   37-23   38-5    38-5    38-8    39-4    39-4    39-7    40-4    40-4    40-7    40-9
        40-9    40-12   41-7    41-7    41-10   42-4    42-4    42-7    43-3    43-3    43-6    44-4    44-4    44-7
        44-11   44-11   44-14   44-23   45-12   45-12   45-15   47-36   51-2    51-2    51-5    52-2    52-2    52-8
        54-21   54-21   54-24   54-36   58-16   58-16   58-21   58-34   59-12   59-12   59-17   59-28   60-14   61-17
        61-17   61-20   61-23   61-23   61-28   62-26   62-26   62-29   62-53
RFRRPA  29-32#  29-35
RESEEK   8-12#
RETRY    8-22#
RETS     5-7#
REVEC    8-17#
REVECT   5-63#
REVINP   8-20#
REVS     5-16#
RGO     59-14   59-20#
RM       5-32#
RONLY    8-29#
RREAL    3-44#  59-8    63-91
RSTOP    3-40#  59-5    63-88
RTRIES   8-31#
RUN     63-27   63-43#
RW.ANG   3-34#
RW.BUF   3-29#  58-15*  59-11*
RW.CMD   3-32#  58-13*  59-9*   62-53   62-53
RW.HI    3-31#  58-10*  62-48   62-53   62-53
RW.LOW   3-30#  58-8*   62-15   62-38   62-53   62-53
RW.SDI   3-33#
RW.STA   3-28#  58-6*   59-6*
RWRDY    6-26#  44-17   54-22   58-17   59-13   61-24   62-27
S.BADP   7-62#   7-63
S.BESS   7-63#   7-68
S.MCNT   7-68#   7-69
S.MEGR   7-60#   7-61
S.MEGW   7-61#   7-62
S.PARM   7-55#   7-56
S.PAT    7-57#   7-58
S.SCHR   7-59#   7-60
S.SDCL   7-56#   7-57
S.TGOF   7-69#   7-70
S.TGSS   7-70#
S.TRKL   7-58#   7-59
SAVREG  29-46#  38-8    38-8*   39-7    39-7*   40-7    40-7*   41-10   41-10*  42-7    42-7*   43-6    43-6*   44-7
        44-7*   45-15   45-15*  51-5    51-5*   54-36   54-36*  58-34   58-34*  59-28   59-28*  60-14   60-14   60-14
        60-14*  60-14*  60-14*  61-20   61-20*  61-28   61-28*  62-53   62-53*
SBCRES   6-44#  45-13   45-15   45-15
SCR     63-21   63-40#
SCTWRD   7-8#
SDI     36-31*  37-11   63-12#
SDILTO  24-27   24-46#  38-18*
SDISTO  24-25   24-45#  38-3*   38-14*
SDIVER   5-4#
SECCNT  55-12*  55-24   58-19   58-32   59-15   59-26   60-12   63-81#
SECGRP  48-13*  48-30   49-73   63-76#
SECTRK  47-9*   48-9    48-50   48-57   48-69   49-36   49-71   50-24   54-9    55-10   63-69#
SEEK    46-10   47-51   48-67   49-33   49-38   50-9    50-53   61-9#
SEEK1   61-19   61-22#
```

```
SEEK2   61-23#  61-27
SEEK3   61-21   61-25   61-29#
SEEKA   61-16#
SEKINP   8-15#
SEND     4-7#   24-18   32-34
SEQSEK   8-34#
SER10   32-19   65-148#
SER11   32-24   35-12   65-151#
SER12   32-36   65-154#
SER13   32-53   65-156#
SER14   32-59   65-158#
SER15   32-63   65-160#
SER16   32-67   65-163#
SER17   32-69   65-166#
SER18   35-18   65-169#
SER18A  35-57   65-172#
SER18B  35-56   65-173#
SER18C  35-55   65-174#
SER18D  35-54   65-175#
SHRTTO   5-3#   38-11
SNDAGN  22-14#  22-18
SPADJU  29-30*  29-40#
SS       5-9#
ST      33-8    41-3    63-16   63-17   63-18   63-19   63-22   63-23   63-24   63-26   63-27   63-54#
ST.C     4-30#
ST.CON   4-29#
ST.DB    4-47#
ST.DF    4-44#
ST.DR    4-37#
ST.ERR   4-28#  41-3
ST.FE    4-41#
ST.FO    4-46#
ST.MOD   4-27#
ST.MSK   4-25#
ST.OA    4-35#
ST.PE    4-43#
ST.PS    4-39#
ST.RE    4-42#
ST.RR    4-36#
ST.RTY   4-31#
ST.RU    4-40#
ST.S7    4-48#
ST.SR    4-38#
ST.STA   4-26#
ST.UNT   4-24#
ST.WE    4-45#
STACK   20-8    30-5#   36-34
START   20-9    32-5#
STATEX  21-16   21-18   21-22#
STATLP  21-9#   21-14   21-20
STATOK  21-12   21-17#
STATUS   4-10#  21-9
STSRES   6-45#  40-5    40-7    40-7
SUB      5-22#  32-3    32-3#   46-6    47-4    47-6    47-16   47-24   47-32   47-44   47-46   48-7    48-16   48-18
        48-26   48-35   48-53   48-62   48-71   48-73   49-9    49-11   49-18   49-23   49-26   49-42   49-63   50-12
        50-32   50-38   54-27   54-29   58-26   63-21   63-59
SUBUNT  45-10*  63-41#
```

CROSS REFERENCE TABLE (CREF V01-05)

CROSS REFERENCE TABLE (CREF V01-05)

```
TALK1A  24-20    24-23#
TALK1B  24-26    24-28#  24-38
TALK2A  24-32    24-41#
TALK2B  24-22    24-36   24-40   24-42#
TALKER  24-15#   38-5    39-4    40-4    41-7    42-4    43-3    44-4    45-12   51-2    61-17
TEST    36-32    37-11#
TEST4    2-37#   24-1
TESTEX  37-18    37-18   37-24   38-5    38-5    38-9    39-4    39-4    39-8    40-4    40-4    40-8    40-9    40-9
        40-13    41-7    41-7    41-11   42-4    42-4    42-8    43-3    43-3    43-7    44-4    44-4    44-8    44-11
        44-11    44-15   44-24   45-12   45-12   45-16   47-37   51-2    51-2    51-6    52-2    52-2    53-1#   54-21
        54-21    54-25   58-16   58-16   59-12   59-12   61-17   61-17   61-23   61-23   62-26   62-26   62-30
TESTX   36-34#   53-1
TIMEOU   5-61#
TLEN.U   7-10#    7-11
TO      28-1#    38-13   38-17
TOOBIG   5-69#
TRACKS   8-36#
TRKBOT  55-20    55-24#
TRKEXT  55-16    55-27#
TRKGRP   5-28#   48-7    48-53   49-9    50-32
TRKLOP  55-12#   55-26
TRKTST  50-22    55-1#
TSTBLK  49-22*   49-25*  49-30   49-62*  49-70*  49-80   49-82*  49-84   49-86*  50-11*  50-19*  50-23   50-25*  50-27
        50-29*   54-6    54-7    55-6    55-8    63-71#
TSTCYL  49-17*   49-20*  49-21*  49-32   49-40   49-46*  49-48   49-50*  49-51   49-54*  49-56*  50-8    50-37*  50-40*
        50-42*   50-43*  50-45*  50-46   50-48   50-50   50-52   63-72#
U.CBN    7-41#    7-42
U.CCNT   7-22#    7-23
U.CCOP   7-44#    7-45
U.CCYL   7-48#    7-49
U.CGRP   7-49#    7-50
U.COPY   7-43#    7-44
U.CSEC   7-28#    7-29
U.CTRK   7-24#    7-25
U.ECCT   7-34#    7-35
U.ELEV   7-31#    7-32
U.LCYL   7-50#    7-51
U.LGRP   7-10     7-51#
U.MASK   7-29#    7-30
U.MBN    7-40#    7-41
U.MLEV   7-33#    7-34
U.MSEC   7-26#    7-27
U.NEXT   7-15#    7-16
U.NFUN   7-20#    7-21
U.NSEC   7-25#    7-26
U.PARM   7-38#    7-39
U.PAT    7-21#    7-22
U.PCTG   7-23#    7-24
U.RBN    7-42#    7-43
U.RTRY   7-32#    7-33
U.RVER   7-46#    7-47
U.RWER   7-45#    7-46
U.RWTO   7-18#    7-19
U.SDIL   7-36#    7-37
U.SDIS   7-35#    7-36
U.SEEK   7-19#    7-20
U.SNUM   7-47#    7-48
```

CROSS REFERENCE TABLE (CREF V01-05)

```
U.SUBP    7-16#    7-17
U.SUBU    7-39#    7-40
U.TIMO    7-17#    7-18
U.TSEC    7-27#    7-28
U.WPRT    7-37#    7-38
U.WRIT    7-30#    7-31
UDATA    25-14     25-16    26-11*   26-15    26-20#
UNIT0     5-46#    36-22
UNIT1     5-47#
UNIT2     5-48#
UNIT3     5-49#
UNITNB   36-20*    36-36    45-3     63-9#
UNITS    32-13     34-7     34-19    34-26    35-7     35-37    36-10    63-5#
UNSSUC    6-42#
UREAD     4-14#    25-15
UTOTST    6-13#    34-4
UWRITE    4-15#    26-16
WAITSI    4-13#    58-24    59-20    62-31
WBUFLN    3-68#     3-69
WCHECK    8-37#
WCHKAL    8-39#
WCONT     3-39#
WGO      58-18     58-24#
WONLY     8-30#
WREAL     3-43#    58-12
WRITEB   56-10     58-1#
WRONG     5-65#
WRTEXT   58-20     58-23    58-31    58-33    58-36#
WSTOP     3-38#    58-5
WTRCMP   55-14     56-1#    56-22
WTREXT   56-12     56-15    56-18    56-23#
XBNCYL    5-41#    47-24
XFERRT    5-5#
XMTERR    6-25#    21-11
XOR      27-10#
XREAD     4-5#     59-22    62-32
XWRITE    4-6#     58-28
```

UDAT3 DISK FUNCTIONAL MACRO X04.00 9-JUL-81 01:13:55 PAGE M-1 SEQ 0276
CROSS REFERENCE TABLE (CREF V01-05)

```
BCS      11-1#
CERROR   16-65#
DEVFTL   16-37#  35-39
DIAG$$    9-5#
DMCODE    1-3#    2-39
DMEND     1-32#  65-178
DMOVLY    1-18#   2-39   65-4
DSTAT    17-3#   37-18   40-9   44-11   52-2   54-21   58-16   59-12   61-23   62-26
ENDERR   16-93#
ERRDF    13-33#
ERRHRD   13-39#  37-18   37-18   37-23   38-5   38-5   38-8   39-4   39-4   39-7   40-4   40-4   40-7   40-9
         40-9    40-12   41-7    41-7    41-10  42-4   42-4   42-7   43-3   43-3   43-6   44-4   44-4   44-7
         44-11   44-11   44-14   44-23   45-12  45-12  45-15  47-36  51-2   51-2   51-5   52-2   52-2   52-8
         54-21   54-21   54-24   54-36   58-16  58-16  58-21  58-34  59-12  59-12  59-17  59-28  60-14  61-17
         61-17   61-20   61-23   61-23   61-28  62-26  62-26  62-29  62-53
ERROR    16-49#  35-39
ERROR$   14-3#   37-18   37-18   37-23   38-5   38-5   38-8   39-4   39-4   39-7   40-4   40-4   40-7   40-9
         40-9    40-12   41-7    41-7    41-10  42-4   42-4   42-7   43-3   43-3   43-6   44-4   44-4   44-7
         44-11   44-11   44-14   44-23   45-12  45-12  45-15  47-36  51-2   51-2   51-5   52-2   52-2   52-8
         54-21   54-21   54-24   54-36   58-16  58-16  58-21  58-34  59-12  59-12  59-17  59-28  60-14  61-17
         61-17   61-20   61-23   61-23   61-28  62-26  62-26  62-29  62-53
ERRORC   16-75#
ERRSF    13-27#
ERRSFT   13-47#
GETP$    15-15#  38-8    39-7    40-7    41-10  42-7   43-6   44-7   45-15  51-5   54-36  54-36  54-36  54-36
         58-34   58-34   58-34   58-34   58-34  58-34  59-28  59-28  59-28  59-28  59-28  59-28  60-14  60-14
         60-14   60-14   60-14   60-14   61-20  61-20  61-20  61-20  61-28  61-28  61-28  62-53  62-53  62-53
         62-53   62-53   62-53
HARDER   16-31#
MOVMSG   16-84#  35-39
MSG      10-3#   63-16   63-17   63-18   63-19  63-20  63-21  63-22  63-23  63-24  63-26  63-27
MSSG     16-111#
MSSGE    16-133#
NDERR    16-101#
OVTERM    2-39    2-39#   2-39#  65-4    65-4#  65-178
PARGS.   14-33#  38-8    38-8    39-7    39-7   40-7   40-7   41-10  41-10  42-7   42-7   43-6   43-6   44-7
         44-7    45-15   45-15   51-5    51-5   52-8   54-36  54-36  54-36  54-36  58-34  58-34  58-34  58-34
         58-34   58-34   58-34   59-28   59-28  59-28  59-28  59-28  59-28  59-28  60-14  60-14  60-14  60-14
         60-14   60-14   60-14   61-20   61-20  61-20  61-20  61-20  61-28  61-28  61-28  62-53  62-53  62-53
         62-53   62-53   62-53   62-53
POP      12-11#  21-22   22-28   24-42   25-17  26-17  27-15  29-39  32-38  32-52  32-55  33-4   33-26  34-15
         34-24   35-29   35-35   35-40   48-25  57-21  61-29  62-33  62-54
PUSH     12-3#   21-7    22-12   24-15   25-10  26-10  27-10  29-7   29-9   32-14  32-35  32-41  34-10  35-5
         35-17   35-30   48-15   57-8    61-9   62-14  62-25
REPSFT   16-10#
RSTR$    15-8#   38-8    39-7    40-7    41-10  42-7   43-6   44-7   45-15  51-5   54-36  58-34  59-28  60-14
         60-14   60-14   61-20   61-28   62-53
RXOR     18-4#
SAVR$    15-1#   38-8    39-7    40-7    41-10  42-7   43-6   44-7   45-15  51-5   54-36  58-34  59-28  60-14
         60-14   60-14   61-20   61-28   62-53
SOFTER   16-4#
SYSFTL   16-43#
TALKX    19-3#   38-5    39-4    40-4    41-7   42-4   43-3   44-4   45-12  51-2   61-17
```

UDAT4 DISK EXERCISER MACRO X04.00 9-JUL-81 01:12:03
TABLE OF CONTENTS

```
20-   1       START OF TEST CODE
21-   1       RDSTAT - GET DRIVE'S REAL TIME DRIVE STATE
22-   1       HOSTRQ - HOST REQUEST - REPORT ERRORS, MEGABYTES TRANSFERRED, ETC.
33-   1       CMPEDC - EDC CALCULATION ROUTINE
34-   1       RTDS   - REAL TIME DRIVE STATE ROUTINE
35-   1       TALK   - SDI LEVEL 2 INTERCHANGE ROUTINE
36-   1       SDI PROTOCOL MESSAGE TABLES
37-   1       BLKCHK - SEE IF A BLOCK WITH ERROR IS KNOWN TO BE BAD
38-   1       CMP2   - 24 BIT COMPARE
39-   1       BUILDP - BUILD THE READ OR WRITE CHAIN LINKS
40-   1       LINK   - BUILD A LINK (NODE) IN THE READ/WRITE CHAIN
41-   1       FILLIN - FILL THE READ/WRITE CHAIN LINK (NODE) WITH REQUIRED INFORMATION
42-   1       ALLOCM - ALLOCATE MEMORY FOR THE READ/WRITE BUFFERS AND CHAIN
43-   1       BULDUM - BUILD THE DUMMY SDI CONTROL BLOCK FOR READS AND WRITES
44-   1       CALC   - CALCULATE THE CYL, GRP AND TRACK FOR THE GIVEN L/RBN
45-   1       STORAGE AREA PATTERNS, DUMMY SDI AREA, OVERLAY INFORMATION AND MORE
46-   1       ROOT   - MAIN DRIVING MODULE OF TEST 4
47-   1       SEQNCR - MODULE SEQUENCER FOR TEST 4
48-   1       OVRLAY - OVERLAY PROCESS FOR BRINGING IN OVERLAYS IF NEEDED
49-   1       ********** NON-OVERLAY MODULE AFINIT - USED AFTER DRIVE IS INITILIZED
50-   1       DRPTST - SEE THAT AT LEAST ONE SUBUNIT IS ACTIVE ON A UNIT
54-   1       CORECT - CORRECT THE ERRORS
55-   1       REPERR - REPORT THE ERROR TO THE HOST
56-   1       RECOVR - RECOVER FROM THE ERROR
57-   1       CHKSTA - CHECK STATUS FOR ANY ERRORS THAT NEED RECOVERY
58-   1       ********** NON-OVERLAY MODULE REDSET - READ/WRITE SETUP MODULE
59-   1       ********** NON-OVERLAY MODULE START - TEST 4 INITIALIZATION
61-   1       BLDUNT - BUILD THE UNIT PARAMETER BLOCK
62-   1       BLDSUS - BUILD ALL SUBUNIT PARAMETER BLOCKS ON THIS UNIT
63-   1       BLDSUB - BUILD A SUBUNIT PARAMETER BLOCK
64-   1       BLDBES - FIND HOW MANY WORDS NEEDED FOR THE BEGIN/END OR TRACK/GROUP SETS
65-   1       BLDBB  - FIND HOW MANY WORDS NEEDED FOR THE BAD BLOCKS
66-   1       GETMEM - ALLOCATES MEMORY FOR UNIT AND SUBUNIT BLOCK AND SUBUNIT PARAMETERS
67-   1       COPYSU - COPY ALL SUBUNIT PARAMETERS TO SUBUNIT BLOCK
68-   1       SORTBE - SORT THE BEGIN/END SETS IN ASCENDING ORDER
69-   1       SWAPBE - IF BEGIN/END SETS OUT OF ORDER, SWAP
70-   1       SORTBB - SORT THE BAD BLOCKS IN ASCENDING ORDER
71-   1       SWAPBB - IF BAD BLOCKS OUT OF ORDER, SWAP
72-   1       SORTTG - SORT THE TRACK/GROUPS IN ASCENDING ORDER
73-   1       TROOT  - TEMPORARY ROOT FOR SEQNCR DURING TEST 4 SETUP
74-   1       INSET  - SET UP UNIT PARAMETERS FOR SEQNCR ROUTINES
75-   1       COMCHR - GET COMMON CHARACTERISTICS AND STORE NECESSARY INFO IN UNIT PARAMETERS
76-   1       INSCHR - GET ALL SUBUNITS' PARAMETERS (CHARACTERISTICS), ERROR CHECK AND STORE
77-   1       CHKCYL - CHECK VALIDITY OF STARTING AND ENDING CYLS, CONVERT TO BEGIN/END SET
78-   1       CYLBN  - GIVEN A CYLINDER, CALCULATE STARTING BN ON THAT CYLINDER
79-   1       CHKBES - CHECK THE BEGIN/END SETS FOR ERRORS
80-   1       CHKBB  - CHECK THE BAD BLOCKS FOR ERRORS
81-   1       COMPSC - COMPUTE SECTORS/GROUP AND SECTORS/CYLINDER
82-   1       CLCMAX - CALCULATE THE MAXIMUM WRITEABLE DBN
83-   1       CHKTG  - CHECK THE TRACK/GROUPS FOR ERRORS, AND CONVERT TO BN'S
85-   1       COMPDP - CALCULATE SECTORS/TRACKS OR SECTORS/GROUPS
86-   1       SMASK  - CALCULATE THE SUBUNIT MASK
87-   1       INITD  - LAST ERROR CHECKING AND SETUP MODULE BEFORE TESTING STARTS
88-   1       GMPARM - GET MASTER PARAMETERS (PATTERN 16) AND SET UP OVERLAY ADDRESSES
```

TABLE OF CONTENTS

```
                                .TITLE  UDAT4   DISK EXERCISER


                        ;
                        ; COPYRIGHT (C)  1980
                        ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
                        ;
                        ; THIS  SOFTWARE  IS  FURNISHED  UNDER A LICENSE FOR USE ONLY ON A
                        ; SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION
                        ; OF  THE  ABOVE COPYRIGHT  NOTICE.  THIS SOFTWARE, OR  ANY  OTHER
                        ; COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE  MADE AVAILABLE
                        ; TO  ANY  OTHER PERSON  EXCEPT  FOR USE ON SUCH SYSTEM AND TO ONE
                        ; WHO AGREES TO THESE LICENSE TERMS.  TITLE  TO  AND  OWNERSHIP OF
                        ; THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DIGITAL.
                        ;
                        ; THE  INFORMATION  IN THIS DOCUMENT IS SUBJECT  TO CHANGE WITHOUT
                        ; NOTICE AND SHOULD NOT BE CONSTRUED AS A  COMMITMENT  BY  DIGITAL
                        ; EQUIPMENT CORPORATION.
                        ;
                        ; DIGITAL ASSUMES NO  RESPONSIBILITY FOR THE USE OR RELIABILITY OF
                        ; ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;
                        ;THIS PROGRAM SHALL BE ASSEMBLED WITH THE PROGRAM DMACRO
                        ;USING A COMMAND LINE SIMILAR TO:
                        ;
                        ;       UDAT4.BIN,UDAT4-UDAT4T,UDATP,UDAT4M,UDATR

        000001          TEST4   =       1                       ; THIS IS TEST4
000000                  .ENABL  ABS
000000                  DMCODE  UDAT4,0,714,3,0
```

```
                                .SBTTL  UDA DM PROGRAM PARAMETERS
                                
                                        .LIST MEB
                                ;
                                ;       EQUATES
                                ;
                                ;       HIGHEST USABLE LOCATION OF UDA MEMORY + 1
                                ;
010000                          HIMEM   =       10000                   ; HIGH MEMORY+1
007774                          OVSTRT  =       7774                    ; OVERLAY ADDRESS LOCATION
                                ;
                                ;
                                ;       OFFSETS FOR FORMAT TRACK TABLE
                                ;
000000                          FT.BUF  =       0.                      ;BUFFER POINTER OFFSET
000001                          FT.HI   =       1.                      ;HI ORDER HEADER OFFSET
000002                          FT.LOW  =       2.                      ;LOW ORDER HEADER OFFSET
                                ;
                                ;
                                ;       OFFSETS FOR FORMAT TRACK BUFFER
                                ;
000000                          FB.DAT  =         0.                    ;FIRST DATA WORD OFFSET
000400                          FB.EDC  =       256.                    ;EDC WORD OFFSET
                                ;
                                ;
                                ;       OFFSETS FOR READ/WRITE I/O CHAIN TABLES
                                ;
000000                          RW.STAT -       0.                      ;STATUS AND NEXT BUFFER POINTER OFFSET
000001                          RW.BUF  =       1.                      ;POINTER TO DATA BUFFER
000002                          RW.LOW  =       2.                      ;HI ORDER EXPECTED HEADER
000003                          RW.HI   =       3.                      ;LOW ORDER EXPECTED HEADER
000004                          RW.CMD  =       4.                      ;SDI COMMAND AND HEAD ADDRESS
000005                          RW.SDI  -       5.                      ;DUMMY SDI CONTROL BLOCK POINTER
000006                          RW.ANG  =       6.                      ;THETA FROM INDEX
                                ;
                                ;       CONSTANTS FOR READ AND WRITE XFC'S
                                ;
140000                          WSTOP   =       140000                  ; LAST ENTRY IN CHAIN FOR WRITE
040000                          WCONT   =       40000                   ; WRITE CONTINUE
100000                          RSTOP   =       100000                  ; LAST ENTRY IN CHAIN FOR READ
000000                          RCONT   -       0                       ; READ CONTINUE
100000                          FSTOP   -       100000                  ; LAST ENTRY IN CHAIN FOR FORMAT
122400                          WREAL   -       122400                  ; WRITE REAL TIME ECOMMAND
013400                          RREAL   -       13400                   ; READ REAL TIME COMMAND
010000                          ECCFLG  -       10000                   ; ECC ERROR IN BUFFER BIT
100000                          EOC             100000                  ; END OF CHAIN
040000                          BUFFLG  -       40000                   ; BUFFER FULL OR EMPTY FLAG
                                ;
                                ;
                                ;       HEADER CODES
                                ;
000000                          HD.LBN  =       000000                  ;GOOD LBN
060000                          HD.RBN  -       060000                  ;GOOD RBN, PERHAPS UNUSED
030000                          HD.REV  =       030000                  ;REVECTORED LBN
110000                          HD.BAD  =       110000                  ;BAD BLOCK
050000                          HD.PRV          050000                  ;PRIMARY REVECTORED BLOCK
120000                          HD.XBN  -       120000                  ;XBN BLOCK
```

UDA DM PROGRAM PARAMETERS

```
                140000          HD.DBN  =       140000                  ;DBN BLOCK
                                ;       OFFSETS FOR DATA BUFFERS
                000000          BF.DAT  =         0.                    ;DATA
                000400          BF.EDC  =       256.                    ;ERROR DETECTION CODE
                000401          BF.ECC  =       257.                    ;LAST 17 ECC RESIDUES
                                ;       BUFFER AND READ/WRITE CHAIN LINK SIZES
                000401          WBUFLN  =       257.                    ; WRITE BUFFER SIZE
                000415          RBUFLN  =       WBUFLN+12.              ; READ BUFFER SIZE
                000007          LINKLN  =       7.                      ; LINK SIZE
```

```
                                ;       XFC DEFINITION EQUATES
                                ;
000000                          BREAK   =       0.      ;BREAKPOINT XFC CODE
000001                          FORMAT  =       1.      ;FORMAT TRACK XFC CODE
000002                          XREAD   =       2.      ;READ N SECTORS XFC CODE
000003                          XWRITE  =       3.      ;WRITE N SECTORS XFC CODE
000004                          SEND    =       4.      ;SEND SDI COMMAND XFC CODE
000005                          RCV     =       5.      ;RECEIVE SDI MESSAGE XFC CODE
000006                          COMPARE =       6.      ;COMPARE DATA PATTERN TO BUFFER
000007                          STATUS  =       7.      ;RETURN DRIVE STATUS XFC CODE
000010                          ECHO    =       8.      ;ECHO DATA TO DRIVE XFC CODE
000011                          DINIT   =       9.      ;DRIVE INITIALIZE XFC CODE
000012                          WAITSI  =       10.     ;WAIT FOR SECTOR OR INDEX PULSE
000013                          UREAD   =       11.     ;READ UNIBUS MEMORY XFC CODE
000014                          UWRITE  =       12.     ;WRITE UNIBUS MEMORY XFC CODE
000015                          ECC     =       13.     ;DO ECC ON BUFFER XFC CODE
000016                          MRD     =       14.     ;SEND BUFFER TO MAINTENANCE READ COMMAND
000017                          MWR     =       15.     ;GET BUFFER FROM MAINTENANCE WRITE COMMAND
000020                          CVT     =       16.     ;CONVERT TO PHYSICAL ADDRESS XFC CODE
000021                          EXIT    =       17.     ;TERMINATE DM PROGRAM XFC CODE
                                ;
                                ;       GET STATUS OFFSETS
                                ;
000000                          ST.UNT  =       0.      ;UNIT NUMBER
000000                          ST.MSK  =       0.      ;SUBUNIT MASK
000001                          ST.STA  =       1.      ;STATUS BYTE
000001                          ST.MOD  =       1.      ;MODE BYTE
000002                          ST.ERR  =       2.      ;ERROR BYTE
000002                          ST.CON  =       2.      ;CONTROLLER BYTE
000002                          ST.C    =       2.      ;C BITS
000003                          ST.RTY  =       3.      ;RETRY COUNT/FAILURE CODE
                                ;
                                ;       STATUS BIT DEFINITIONS
                                ;
000200                          ST.OA   =       200     ; ONLINE TO ANOTHER (SET IF DRIVE UNAVAILABLE)
000100                          ST.RR   =       100     ; READJUSTMENT BIT (SET IF RECALIBRATION REQUIRED)
000040                          ST.DR   =       40      ; DIAGNOSTIC REQUEST (SET IF DIAGNOSTIC REQUESTED)
000020                          ST.SR   =       20      ; SPINDLE READY (SET IF SPINDLE READY)
000002                          ST.PS   =       2       ; PORT SWITCH (SET IF PORT SWITCH IN)
000001                          ST.RU   =       1       ; RUN/STOP SWITCH (SET IF RUN/STOP SWITCH IN)
000200                          ST.FE   =       200     ; FATAL ERROR (SET IF FATAL ERROR OCCURRED)
000100                          ST.RE   =       100     ; RETRIABLE ERROR (SET IF RETRIABLE ERROR OCCURRED)
000040                          ST.PE   =       40      ; PROTOCOL ERROR (SET IF PROTOCOL ERROR OCCURRED)
000020                          ST.DF   =       20      ; INITIALIZATION FAILURE (SET IF INIT FAILED)
000010                          ST.WE   =       10      ; WRITE ENABLE (SET IF WRT ATTEMPTED ON PROT DISK)
002000                          ST.FO   =       2000    ; FORMATTING (SET IF FORMATTING ENABLED)
001000                          ST.DB   =       1000    ; DIAGNOSTIC CYLS (SET IF DIAG CYL ACCESS ENABLED)
000400                          ST.S7   =       400     ; SECTOR SIZE (SET FOR 576 BYTE SECTORS)
```

```
                                ;       GET COMMON CHARACTERISTICS OFFSETS
                                ;
000000                          SHRTTO  =       0.              ;SHORT TIMEOUT <3:0>
000000                          SDIVER  =       0.              ;SDI VERSION <7:4>
000000                          XFERRT  =       0.              ;TRANSFER RATE <15:0>
000001                          LONGTO  =       1.              ;LONG TIMEOUT <3:0>
000001                          RETS    =       1.              ;RETRIES <7:4>
000001                          RCTCPS  =       1.              ;F/RCT COPIES <11:8>
000001                          SS      =       1.              ;SECTOR SIZE <15:15>
000002                          ERLEV   =       2.              ;ERROR RETRY LEVELS <7:0>
000002                          ECCRSH  =       2.              ;ECC THRESHOLD <15:8>
000003                          MICREV  =       3.              ;MICROCODE REVISION NUMBER <7:0>
000003                          HRDREV  =       3.              ;HARDWARE REVISION NUMBER <15:8>
000004                          DRVID   =       4.              ;UNIQUE DRIVE ID <47:0>
000007                          DRTYPE  =       7.              ;DRIVE TYPE IDENTIFIER <7:0>
000007                          REVS    =       7.              ;REVS/SECOND <15:8>
                                ;
                                ;       GET SUBUNIT CHARACTERISTICS OFFSETS
                                ;
                                ;THESE OFFSETS ARE CURRENTLY GIVEN AS FOLLOWING THE COMMON CHARACTERISTICS
                                ;
000013                          SUB     =       11.             ;OFFSET TO PUT SUBUNIT AFTER COMMON;
000000                          LBNCYL  -       0.              ;NUMBER OF CYLINDERS IN LBN AREA <31:0>
000001                          HICYL   -       1.              ;HI ORDER CYLINDER BITS <15:12>
000002                          GRPCYL  -       2.              ;GROUPS PER CYLINDER <7:0>
000002                          HILBN   -       2.              ;HI STARTING LBN <11:8>
000002                          HIXBN   =       2.              ;HI STARTING XBN <15:12>
000003                          TRKGRP  -       3.              ;TRACKS PER GROUP <7:0>
000003                          HIRBN   -       3.              ;HI STARTING RBN <11:8>
000003                          HIDBN   -       3.              ;HI STARTING DBN <15:12>
000004                          RBNTRK  -       4.              ;RBNS PER TRACK <6:0>
000004                          RM              4.              ;REMOVABLE MEDIA <7:7> 1=REMOVEABLE
000005                          DATPRE  -       5.              ;DATA PREAMBLE SIZE IN WORDS <7:0>
000005                          HDRPRE  =       5.              ;HEADER PREAMBLE SIZE IN WORDS <15:8>
000006                          MEDTYP  =       6.              ;MEDIA TYPE <31:0>
000010                          FCTSIZ  =       8.              ;FCT COPY SIZE <15:0>
000011                          LBNTRK  -       9.              ;LBNS PER TRACK <7:0>
000011                          GRPOFF  =       9.              ;GROUP OFFSET (SECTORS) <15:8>
000012                          LBNHST  -       10.             ;LBNS IN HOST AREA <31:0>
000014                          RCTCSZ  =       12.             ;RCT COPY SIZE <15:0>
000021                          XBNCYL  =       17.             ;CYLS IN XBN AREA <15:0>
000022                          DBNCYL  -       18.             ;CYLS IN DBN AREA <15:8>
                                ;
                                ;       UNIT CODES
                                ;
000001                          UNIT0   -       1.              ;UNIT ZERO CODE
000002                          UNIT1   -       2.              ;UNIT ONE CODE
000004                          UNIT2           4.              ;UNIT TWO CODE
000010                          UNIT3           8.              ;UNIT THREE CODE
                                ;
                                ;       BIT MASK DEFINITIONS
                                ;
                                ;
177400                          HIBYTE  =       177400          ;HIGH BYTE MASK
000377                          LOBYTE          000377          ;LOW BYTE MASK
007777                          HBHINB  -       7777            ;HI BYTE, HI NIBBLE MASK
170377                          HBLONB          170377          ;HI BYTE, LO NIBBLE MASK
```

UDA DM PROGRAM PARAMETERS

```
      177417          LBHINB  =       177417          ;LO BYTE, HI NIBBLE MASK
      177760          LBLONB  =       177760          ;LO BYTE, LO NIBBLE MASK
                      ;
      000001          TIMEOUT =       1.              ;DRIVE TIMEOUT CODE
      000002          HEADER  =       2.              ;HEADER COMPARE FAILURE CODE
      000004          REVECT  =       4.              ;REVECTOR NEEDED CODE
                      ;
      000002          WRONG   =       2.              ;FIRST WORD NOT START FRAME CODE
      000004          FRAME   =       4.              ;FRAMING ERROR CODE
      000010          CHECK   =       8.              ;CHECKSUM ERROR CODE
                      ;
      000001          TOOBIG  =       1.              ;NUMBER OF WORDS EXCEEDS 7064
      000002          LOW     =       2.              ;DM BUFFER ADDRESS IS LESS THAN 714
                      ;
                      ;       BIT MASK
                      ;
      000001          LARGE   =       1.              ;BLOCK NUMBER TOO LARGE
      000002          OVERFL  =       2.              ;SECTOR NUMBER LARGER THAN 16 BITS
```

```
                        ;MAINTANENCE READ/WRITE REQUEST NUMBERS
                        ;
000000  T1MSIZ  =       0.              ;GET FREE MEMORY PARAMETERS
000001  T2DLL   =       1.              ;DOWNLINE LOAD DRIVE DIAGNOSTIC
000002  T2CMD   =       2.              ;MANUAL INTERVENTION TEST 2 PROTOCOL
000003  T4MPRM  =       3.              ;GET MASTER PARAMETERS FROM SW QUESTIONS
000004  T4UPRM  =       4.              ;GET UNIT PARAMETERS FROM HW QUESTIONS
000005  T4BB1   =       5.              ;GET BAD BLOCKS (1 THRU 14)
000006  T4BB2   =       6.              ;GET REST OF BAD BLOCKS (15 AND 16)
000007  T4SOFT  =       7.              ;ADD TO SOFT ERROR AND ECC COUNT
000010  T4SEEK  =       8.              ;ADD 1 TO SEEK COUNT
000011  T4MXFR  =       9.              ;ADD TO MEGABITS READ AND WRITTEN
000012  UTOTST  =       10.             ;GET UNITS TO TEST
000013  ERRMES  =       11.             ;PRINT ERROR MESSAGE
000014  ERRMC   =       12.             ;TEST 4 ERROR REPORTING
000015  MESSAG  =       13.             ;INFORMATION MESSAGE
000016  DONE    =       14.             ;MARK DM PROGRAM AS NO LONGER RUNNING
                        ;
                        ;       OTHER BIT DEFINITIIONS
                        ;
000001  RCVRDY  =       1               ; RECIEVER READY 1 = READY
000002  ATTN    =       2               ; ATTENTION BIT FOR RETURN DRIVE SIGNALS XFC
000004  RCVERR  =       4               ; RECIEVER ERROR
000100  AVAIL           100             ; AVAILABLE 1 = AVAILABLE
000400  XMTERR  =       400             ; TRANSMIT ERROR
100000  RWRDY   =       100000          ; IF SET, UDA IS ABLE TO READ AND/OR WRITE TO DRIVE
                        ;
                        ;       SDI COMMANDS AND RESPONSES
                        ;
000204  DISCON  =       204             ; DISCONNECT DRIVE
000006  ERECOV  =       6               ; ERROR RECOVERY
000201  CHGMOD  =       201             ; CHANGE MODE
000213  DRVONL  =       213             ; DRIVE ONLINE
000014  DRVRUN  =       14              ; DRIVE RUN
000005  DRVCLR  =       5               ; DRIVE CLEAR OPCODE
000207  GETCHR  =       207             ; GET CHARACTERISTICS
000210  GETSUB  =       210             ; GET SUBUNIT CHARACTERISTICS
000011  GETSTA  =       11              ; GET STATUS
000216  IRECLB  =       216             ; RECALIBRATE
000012  INSEEK  =       12              ; INITIATE SEEK
000176  COMPLT  =       176             ; SUCCESSFUL COMPLETION
000175  UNSSUC  =       175             ; UNSUCCESSFUL COMPLETION
000170  CHRRES  =       170             ; GET CHARACTERISTICS RESPONSE
000167  SBCRES  =       167             ; GET SUBUNIT CHARACTERISTICS RESPONSE
000366  STSRES  =       366             ; GET STATUS RESPONSE
000350  ECHOC   =       350             ; DIAGNOSTIC ECHO COMMAND AND RESPONSE
                        ;
                        ;       ERROR CODES
                        ;
000000  FTLSYS  =       0               ; SYSTEM FATAL ERROR
000400  FTLDEV  =       400             ; DEVICE FATAL
001000  ERHARD  =       1000            ; HARD ERROR
001400  ERSOFT  =       1400            ; SOFT ERROR
```

```
TEST 4 SPECIFIC INFORMATION

                                .SBTTL  TEST 4 SPECIFIC INFORMATION
                                ;
                                ;       TEST 4 SPECIFIC INFORMATION
                                ;
                                ;
                                ;       CONSTANTS
                                ;
000377                          SCTWRD  =       255.            ; NUMBER OF WORDS IN SECTOR TO FILL
000105                          INTEDC  =       69.             ; INITIAL EDC VALUE
000061                          TLEN.U  =       U.LGRP+1        ; UNIT PARAMETER LENGTH
007717                          FIRSTU  =       HIMEM-TLEN.U    ; LOCATION OF FIRST UNIT PARAMETER BLOCK
                                ;
                                ;       UNIT PARAMETER OFFSETS
                                ;
000000                          U.NEXT  =       0.              ; POINTER TO NEXT UNIT (RING LINKED LIST)
000001                          U.SUBP  =       U.NEXT+1        ; 4 WORDS OF SUBUNIT PARAMETER POINTERS
000005                          U.TIMO  =       U.SUBP+4        ; AREA TO STORE VARIOUS TIMEOUT VALUES
000006                          U.RWTO  =       U.TIMO+1        ; READ/WRITE TIMEOUT AREA
000007                          U.SEEK  =       U.RWTO+1        ; NUMBER OF SEEKS ISSUED
000010                          U.NFUN  =       U.SEEK+1        ; NEXT FUNCTION ADDRESS (FOR DEFERRED CALLS)
000011                          U.PAT   =       U.NFUN+1        ; PATTERN NUMBER TO WRITE
000012                          U.CCNT  =       U.PAT+1         ; CURRENT COUNT OF T/G LOOPS
000014                          U.PCTG  =       U.CCNT+2        ; POINTER TO CURRENT TRACK OR GROUP
000015                          U.CTRK  =       U.PCTG+1        ; TRACK COUNT FOR GROUP OPERATIONS
000016                          U.NSEC  =       U.CTRK+1        ; NUMBER OF SECTORS R/W THIS TRY
000017                          U.MSEC  =       U.NSEC+1        ; NUMBER OF SECTORS TO BE R/W
000020                          U.TSEC  =       U.MSEC+1        ; NUMBER OF SECTORS TO BE R/W THIS OP
000021                          U.CSEC  =       U.TSEC+1        ; COUNT OF SECTORS R/W SO FAR
000022                          U.MASK  =       U.CSEC+1        ; UNIT MASK FOR XFC CALLS (0001 - 1000)
000023                          U.WRIT  =       U.MASK+1        ; WRITE PROTECTION STATUS
000024                          U.ELEV  =       U.WRIT+1        ; CURRENT ERROR RECOVERY LEVEL
000025                          U.RTRY  =       U.ELEV+1        ; MAXIMUM NUMBER OF READ RETRIES
000026                          U.MLEV  =       U.RTRY+1        ; MAXIMUM NUMBER OF ERROR RECOVERY LEVELS
000027                          U.ECCT  =       U.MLEV+1        ; ECC THRESHOLD
000030                          U.SDIS  =       U.ECCT+1        ; SDI SHORT TIMEOUT
000031                          U.SDIL  =       U.SDIS+1        ; SDI LONG TIMEOUT
000032                          U.WPRT  =       U.SDIL+1        ; MASK TO WRITE PROTECT READ-ONLY DRIVES
000033                          U.PARM  =       U.WPRT+1        ; UNIT PARAMETER WORD
000034                          U.SUBU  =       U.PARM+1        ; SUBUNIT OFFSET (0 - 3)
000035                          U.MBN   =       U.SUBU+1        ; MASTER L/DBN
000037                          U.CBN   =       U.MBN+2         ; CURRENT L/DBN FOR START OF CHAIN
000041                          U.RBN   =       U.CBN+2         ; RBN TO BE READ/WRITTEN IF LBN REVECTORED
000043                          U.COPY  =       U.RBN+2         ; NUMBER OF RCT COPIES ON EACH SUBUNIT
000044                          U.CCOP  =       U.COPY+1        ; CURRENT RCT COPY THAT WE'RE WORKING ON
000045                          U.RWER  =       U.CCOP+1        ; ERROR (IF ANY) ON THE LAST R/W
000046                          U.RVER  =       U.RWER+1        ; ERROR THAT OCCURRED BEFOR THE REVECTOR OPERATION
000047                          U.SNUM  =       U.RVER+1        ; 4 WORDS THAT HOLD THE SUBUNIT LOGICAL NUMBERS
000053                          U.CCYL  =       U.SNUM+4        ; CURRENT CYLINDER
000055                          U.CGRP  =       U.CCYL+2
000056                          U.LCYL  =       U.CGRP+1        ; LAST CYLINDER SEEKED TO
000060                          U.LGRP  =       U.LCYL+2        ; LAST GROUP SEEKED TO
                                ;
                                ;       SUBUNIT PARAMETER OFFSET
                                ;
000000                          S.PARM  =       0.              ; SUBUNIT PARAMETER WORD
000001                          S.SDCL  =       S.PARM+1        ; STARTING DIAGNOSTIC CYLINDER
000003                          S.PAT   =       S.SDCL+2        ; PATTERN TO USE FOR WRITES
```

TEST 4 SPECIFIC INFORMATION

```
58          000004                  S.TRKL  =       S.PAT+1                 ; NUMBER OF SECTORS IN ONE TRACK
59          000005                  S.SCHR  =       S.TRKL+1                ; POINTER TO SUBUNIT CHARACTERISTICS
60          000006                  S.MEGR  =       S.SCHR+1                ; SECTORS READ (UP TO 245)
61          000007                  S.MEGW  =       S.MEGR+1                ; SECTORS WRITTEN (UP TO 245)
62          000010                  S.BADP  =       S.MEGW+1                ; POINTER TO BAD BLOCK AREA
63          000011                  S.BESS  =       S.BADP+1                ; START OF BEGIN/END SETS
64                                  ;
65                                  ;       IF TRACK/GROUP LIMITS ARE GIVEN, THE SUBUNIT PARAMETERS HAVE THE
66                                  ;       FOLLOWING FIELDS ADDED TO THEM
67                                  ;
68          000011                  S.MCNT  =       S.BESS                  ; MAXIMUM TRACK/GROUP COUNT
69          000013                  S.TGOF  =       S.MCNT+2                ; ORIGINAL TRACK/GROUP OFFSET
70          000015                  S.TGSS  =       S.TGOF+2                ; START OF TRACK/GROUP SETS
```

TEST 4 SPECIFIC INFORMATION

```
                                ;
                                ;       DUMMY SDI CONTROL BLOCK OFFSETS
                                ;
000001                          D.LIMT  =       1                       ; DUMMY SDI SEARCH LIMIT
000002                          D.SCHR  =       2                       ; DUMMY POINTER TO SUBUNIT CHAR-5
                                ;
                                ;
                                ;       UNIT PARAMETER BITS (U.PARM(R5))
                                ;
100000                          DROP    =       100000                  ; DROP BIT (SET IF UNIT OR SUBUNIT DROPPED)
040000                          INITW   =       40000                   ; INITIAL WRITE (SET IF INITIAL WRITE IN PROG)
020000                          RESEEK  =       20000                   ; IF 1, INDICATES THAT A SEEK IS NECESSARY
010000                          DIREC   =       10000                   ; DIRECTION (SET IF SEQUENTIAL ACCESSES DECREASING)
004000                          NEWSUB  =       4000                    ; SET IF SEQUENTIAL SEEKS MOVED TO NEW SUBUNIT
002000                          SEKINP  =       2000                    ; SEEK IN PROGRESS - SET IF TRUE
001000                          FTIME   =       1000                    ; FIRST TIME FLAG - SET FOR INIT CODE
000400                          REVEC   =       400                     ; REVECTOR BIT (SET IF BLOCK REVECTORED)
000200                          RBNBN   =       200                     ; SEE IF WORKING ON RBN
000100                          REDWRT  =       100                     ; REDWRT (READ OR WRITE IN PROGRESS SET IF WRITE)
000040                          REVINP  =       40                      ; REVECTORING OPERATION IN PROGRESS
000020                          DATERR  =       20                      ; DATA ERROR IF SET
000004                          RETRY   =       4                       ; IF CLEAR, START RETRIES AT ZERO
000001                          RCLB    =       1                       ; RECALIBRATION BIT (SET IF RECALIBRATE JUST DONE)
                                ;
                                ;       SUBUNIT PARAMETER BITS (S.PARM(R4))
                                ;
020000                          DCYLS   =       20000                   ; DIAGNOSTIC CYLINDER FLAG (SET IF DBNS)
010000                          ECCCHK  =       10000                   ; 1 IF ECC CORRECTION ALLOWED   MASTER BITS
004000                          RONLY   =       4000                    ; READ ONLY (SET IF READ ONLY)
002000                          WONLY   =       2000                    ; WRITE ONLY (SET IF WRITE ONLY)
001000                          RTRIES  =       1000                    ; 1 IF RETRIES ALLOWED
000200                          ONLYCL  =       200                     ; SET IF ONLY CYLINDERS SPECIFIED
                                                                        ; USED DURING SETUP ONLY
000100                          SEQSEK  =       100                     ; SEQUENTIAL SEEK (START UP TESTING ONLY)
000040                          BEUSED  =       40                      ; BEGIN/END SETS (USED IF SET)
000020                          TRACKS  =       20                      ; TRACKS OR GROUPS (TRACK IF SET)
000010                          WCHECK  =       10                      ; WRITE CHECK BIT (IF SET, WRITE CHECK WILL BE DONE)
                                ;                               WCHECK IS ALSO USED FOR THE UNIT PARAMETERS
000004                          WCHKAL  =       4                       ; SET IF WRITE CHECK ALWAYS TO BE DONE
000002                          DATCMP  =       2                       ; DATA COMPARE (SET IF DATA COMPARE TO BE DONE)
                                ;                               DATCMP IS ALSO USED FOR THE UNIT PARAMETERS
000001                          DCMPAL  =       1                       ; SET IF DATA COMPARE ALWAYS TO BE DONE
```

MACRO DEFINITIONS

```
                        .SBTTL  MACRO DEFINITIONS
                        ;
                        ;       DIAGNOSTIC MACRO FOR TEST4 OVERLAYS
                        ;
                                .MACRO  DIAG$$
                                TST     $$DIAG+$DIAG$
                                BEQ     .+6
                                MOV     #60000,R0
                                MOV     R0,@$$DIAG+$DIAG$
                                BR      .+1
                        $DIAG$  =       $DIAG$ + 1
                                .ENDM
```

MACRO DEFINITIONS

```
                        ;       MESSAGE CONTROL TABLE MACRO
                        ;
                        .MACRO MSG CMDBUF,CMDSZ,RPLBUF,RPLSZ,SUCCOM
                        .WORD CMDBUF                    ;ADDRESS OF COMMAND
                        .WORD CMDSZ                     ;SIZE OF COMMAND IN BYTES
                        .WORD RPLBUF                    ;ADDRESS OF REPLY
                        .WORD RPLSZ                     ;SIZE OF REPLY IN WORDS
                        .IF NB NUMBER
                        .WORD SUCCOM                    ; SUCCESSFUL COMPLETION CODE
                        .ENDC
                        .ENDM
```

```
UDAT4 DISK EXERCISER MACRO X04.00 9-JUL-81 0*:12:03 PAGE 11                                    SEQ 0290
MACRO DEFINITIONS

      1                                .MACRO  BCS     LAB..
      2                                                                BCC     .+2
      3                                                                BR      LAB..
      4                                .ENDM
```

MACRO DEFINITIONS

```
                        ;       PUSH REGISTER MACRO
                        ;
                                .MACRO PUSH R9
                                .IRP X,<R9>
                                MOV X,-(SP)
                                .ENDR
                                .ENDM
                        ;
                        ;       POP REGISTER MACRO
                        ;
                                .MACRO POP R9
                                .IRP X,<R9>
                                MOV (SP)+,X
                                .ENDR
                                .ENDM
```

MACRO DEFINITIONS

```
                        ;ERROR MACROS
                        ;THESE MACROS ARE CALLED TO REPORT ERRORS TO THE HOST PROGRAM.
                        ;THE MACRO NAMES ARE : ERRSF, ERRDF, ERRHRD, ERRSFT. EACH RESULTS IN THE HOST
                        ;BEING REQUESTED TO REPORT THE ERROR.
                        ;ARGUMENTS:     1 (MS$) MESSAGE POINTER
                        ;               2 (P1$) PARAMETER #1
                        ;               3 (P2$) PARAMETER #2
                        ;               4 (P3$) PARAMETER #3
                        ;               5 (P4$) PARAMETER #4
                        ;               6 (P5$) PARAMETER #5
                        ;               7 (P6$) PARAMETER #6
                        ;               8 (P7$) PARAMETER #7
                        ;               9 (P8$) PARAMETER #8
                        ;
                        ;THE MESSAGE POINTER MUST POINT TO AN ADDRESS IN THE OVERLAY 'MS' IMMEDIATELY
                        ;FOLLOWING THE MAIN CODE. ANY ADDRESS MODE MAY BE USED (E.G. #MS1, @R2).
                        ;THE ADDRESS MUST CONTAIN AN ASCII FORMAT STRING TO DETERMINE THE MESSAGE
                        ;TO PRINT.
                        ;THE PARAMETER ARGUMENTS ARE OPTIONAL. THEY SHOULD BE SUPPLIED ONLY WHEN
                        ;THERE IS DATA TO BE PASSED TO THE HOST THAT WILL BE USED IN PRINTING THE
                        ;MESSAGE. THESE PARAMETER ARGUMENTS ARE THE ADDRESS OF DATA TO BE PASSED
                        ;USING ANY ADDRESSING MODE DESIRED.
                        ;ALL REGISTERS ARE RETURNED UNCHANGED. IT SHOULD BE NOTED THAT ARGUMENTS
                        ;CONTAINING SOMETHING OTHER THAN A REGISTER NAME (E.G. #100 OR MEMADR)
                        ;ASSEMBLE TO INSTRUCTIONS THAT SAVE AND RESTORE A REGISTER ON THE STACK.

                                .MACRO ERRSF MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ 0,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .ENDM

                                .MACRO ERRDF MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ 1,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .ENDM

                                .MACRO ERRHRD MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NLIST
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ 2,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .LIST
                                .ENDM

                                .MACRO ERRSFT MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$
                                .NARG ARGS$
                                .RADIX 10
                                ERROR$ 3,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,\ERRN
                                .ENDM
```

MACRO DEFINITIONS

```
                        ;THE FOLLOWING MACRO ACTUALLY PROCESSES THE ERROR CALL TO THE HOST PROGRAM

                .MACRO ERROR$ ET$,MS$,P1$,P2$,P3$,P4$,P5$,P6$,P7$,P8$,ERRN$
                  .RADIX 8
                  PRMS=ARGS$-1
                  .IF LT,<PRMS>
                    .ERROR;NOT ENOUGH ARGUMENTS IN ERROR CALL
                  .ENDC
                  REGS$=-1
                  .IIF GE,<PRMS-8.>,PARGS. P8$
                  .IIF GE,<PRMS-7.>,PARGS. P7$
                  .IIF GE,<PRMS-6.>,PARGS. P6$
                  .IIF GE,<PRMS-5.>,PARGS. P5$
                  .IIF GE,<PRMS-4.>,PARGS. P4$
                  .IIF GE,<PRMS-3.>,PARGS. P3$
                  .IIF GE,<PRMS-2.>,PARGS. P2$
                  .IIF GE,<PRMS-1.>,PARGS. P1$
                  .IF GE REGS$
                    RSTR$ \REGS$
                  .ENDC
                  .RADIX 10
                  .LIST
                                                                CALL RERROR     ;ERROR # ERRN$'.
                  .NLIST
                  .RADIX 8
                  .LIST
                                                                .WORD <PRMS*2000>+<ET$*400>+ERRN
                                                                .WORD MS$
                  .NLIST
                  ERRN=ERRN+1
                .ENDM

                .MACRO PARGS.,ADDR$
                  .NTYPE PTYPE$,ADDR$
                  .IF EQ,<PTYPE$&70>
                    .IIF EQ,<PTYPE$&7>-REGS$,RSTR$ \REGS$
                    .LIST
                                                                MOV ADDR$,-(SP)
                    .NLIST
                  .IFF
                    .IF EQ,<PTYPE$&7>-1         ;PICK A REGISTER TO USE
                      REGU$=2                   ;SELECT R2 IF R1 IS USED IN PARAMETER FETCH
                    .IFF
                      REGU$=1                   ;OTHERWISE USE R1
                    .ENDC
                    .IF NE,<REGU$-REGS$>        ;IF REGISTER NOT ALREADY SAVED
                      .IF GE,REGS$
                        RSTR$ \REGS$            ;RESTORE CURRENT SAVED REGISTER
                      .ENDC
                      SAVR$ \REGU$              ;THEN SAVE SELECTED REGISTER
                    .ENDC
                    GETP$ \REGS$,ADDR$
                  .ENDC
                .ENDM
```

MACRO DEFINITIONS

```
                        .MACRO SAVR$ REGN
                          .LIST
                                                        MOV R'REGN,SAVREG
                          .NLIST
                          REGS$=REGN
                        .ENDM

                        .MACRO RSTR$ REGN
                          .LIST
                                                        MOV SAVREG,R'REGN
                          .NLIST
                          REGS$=-1
                        .ENDM

                        .MACRO GETP$ REGN,ADDR$
                          .LIST
                                                        MOV ADDR$,R'REGN
                                                        MOV R'REGN,-(SP)
                          .NLIST
                        .ENDM
```

```
                        ;
                        ;       PRIMARY ERROR REPORTING (TEST 4)
                        ;
                        .MACRO  SOFTER  NUM,ARGS
                        ERROR   #ERSOFT,NUM,<ARGS>
                                                        MOV     #ERRMES,R2
                                                        MOV     R2,OUT.RQ
                        .ENDM

                        .MACRO  REPSFT  SFTFLG,ECCFG
                        .IF     NB,SFTFLG
                                                        MOV     #1,OUT.02
                        .IFF
                                                        CLR     OUT.02
                        .ENDC
                        .IF     NB,ECCFG
                                                        MOV     #1,OUT.03
                        .IFF
                                                        CLR     OUT.03
                        .ENDC
                                                        PUSH    R0
                                                        MOV     #U.SNUM,R0
                                                        ADD     R5,R0
                                                        ADD     U.SUBU(R5),R0
                                                        MOV     (R0),OUT.01
                                                        MOV     #T4SOFT,R0
                                                        CALL    HOSTRQ
                                                        POP     R0
                        .ENDM

                        .MACRO  HARDER  NUM,ARGS
                        ERROR   #ERHARD,NUM,<ARGS>
                                                        MOV     #ERRMC,R2
                                                        MOV     R2,OUT.RQ
                        .ENDM

                        .MACRO  DEVFTL  NUM,ARGS
                        ERROR   #FTLDEV,NUM,<ARGS>
                                                        MOV     #ERRMC,R2
                                                        MOV     R2,OUT.RQ
                        .ENDM

                        .MACRO  SYSFTL  NUM,ARGS
                        ERROR   #FTLSYS,NUM,<ARGS>
                                                        MOV     #ERRMC,R2
                                                        MOV     R2,OUT.RQ
                        .ENDM

                        .MACRO  ERROR   TYPE,NUM,ARGS
                        .RADIX  10
                        NUMPTR  =       5
                        MOVMSG  #ER'NUM,4
                        .IF     NB,<ARGS>
                        .IRP    X,<ARGS>
                        MOVMSG  X,\NUMPTR
                        NUMPTR  =       NUMPTR + 1
                        .ENDR
```

MACRO DEFINITIONS

```
58                                      .ENDC
59                                                              MOV     #NUM,OUT.02
60                                                              BIS     TYPE,OUT.02
61                                                              MOV     #.,OUT.01
62                                      .RADIX
63                                      .ENDM
64
65                                      .MACRO  CERROR  STNUM,ARGS
66                                      .RADIX  10
67                              NUMPTR  -       STNUM
68                                      .IRP    X,<ARGS>
69                                      MOVMSG  X,\NUMPTR
70                              NUMPTR  =       NUMPTR + 1
71                                      .ENDR
72                                      .RADIX
73                                      .ENDM
74
75                                      .MACRO  ERRORC  ARGS
76                                      .RADIX  10
77                                      .IRP    X,<ARGS>
78                                      MOVMSG  X,\NUMPTR
79                              NUMPTR  -       NUMPTR + 1
80                                      .ENDR
81                                      .RADIX
82                                      .ENDM
83
84                                      .MACRO  MOVMSG  ARG,INDX
85
86                                      .IF     LT,INDX-10
87                                                              MOV     ARG,OUT.0'INDX
88                                      .IFF
89                                                              MOV     ARG,OUT.'INDX
90                                      .ENDC
91                                      .ENDM
92
93                                      .MACRO  ENDERR  POS
94                                      .IF     NB,POS
95                              NDERR   POS
96                                      .IFF
97                              NDERR   \NUMPTR
98                                      .ENDC
99                                      .ENDM
100
101                                     .MACRO  NDERR   POS
102                                     .IF     NE,POS
103                                                             BIS     #POS,ERRPOS     ; SET THE POSITION
104                                     .IFF
105                                                             CLR     ERRPOS          ; CLEAR THE POSITION
106                                     .ENDC
107                                     .ENDM
108
109                             ;       MESSAGE REPORTING MACRO
110                             ;
111                                     .MACRO  MSSG    NUM,ARGS
112                                     .RADIX  10
113                             NUMPTR  =       3
114                                     MOVMSG  #MS'NUM,2
```

```
115                                     .IF     NB,<ARGS>
116                                     .IRP    X,<ARGS>
117                                     MOVMSG  X,\NUMPTR
118                                     NUMPTR  =       NUMPTR + 1
119                                     .ENDR
120                                     .ENDC
121                                                             PUSH    <R0,R1>
122                                                             MOV     #U.SNUM,R1
123                                                             ADD     R5,R1
124                                                             ADD     U.SUBU(R5),R1
125                                                             MOV     (R1),OUT.01
126                                                             MOV     #MESSAG,R0
127                                                             CALL    HOSTRQ
128                                                             POP     <R1,R0>
129                                     .RADIX
130                                     .ENDM
131                                     ;
132                                     ;
133                                     .MACRO  MSSGE   NUM,ARGS
134                                     .RADIX  10
135                                     NUMPTR  =       3
136                                     MOVMSG  #'NUM,2
137                                     .IF     NB,<ARGS>
138                                     .IRP    X,<ARGS>
139                                     MOVMSG  X,\NUMPTR
140                                     NUMPTR  =       NUMPTR + 1
141                                     .ENDR
142                                     .ENDC
143                                                             PUSH    <R0,R1>
144                                                             MOV     #MESSAG,R0
145                                                             CALL    HOSTRQ
146                                                             POP     <R1,R0>
147                                     .RADIX
148                                     .ENDM
```

MACRO DEFINITIONS

```
                        ;RETURN DRIVE STATUS MACRO WITH ERROR REPORTING

                                .MACRO  DSTAT,LAB$,E1,E2
                                .NLIST
                                .NLIST  MEB
                                .LIST   MF
                                .LIST
                                CALL    RDSTAT                  ; GET DRIVE STATUS
                                BIT     #10000,R1               ; SEE IF ANY ERRORS
                                BEQ     2$                      ; IF NO ERROR, BRANCH
                                BIT     #4000,R1                ; SEE IF XMIT ERROR
                                BEQ     1$                      ; IF SO, BRANCH
                                ERRHRD  E1                      ; REPORT INVALID STATUS ERROR
                                BR      LAB$                    ; BRANCH TO DONE
                        1$:     ERRHRD  E2                      ; REPORT XMIT ERROR
                                BR      LAB$                    ; BRANCH TO DONE
                        2$:
                                .NLIST
                                .NLIST  ME
                                .LIST   MEB
                                .LIST
                                .ENDM
```

MACRO DEFINITIONS

```
                        ;
                        ;       XOR THE CONTENTS OF TWO REGISTERS
                        ;
                        .MACRO  RXOR    REG1,REG2
                        MOV     REG2,-(SP)              ; SAVE REGISTER REG2
                        BIC     REG1,REG2               ; CLEAR COORESPONDING BITS IN REG2
                        BIC     (SP)+,REG1              ; CLEAR COORESPONDING BITS IN REG1
                        BIS     REG1,REG2               ; OR WHAT'S LEFT
                        .ENDM
```

MACRO DEFINITIONS

```
                        ;       SDI INTERCHANGE WITH DRIVE WITH ERROR REPORTING

                                .MACRO  TALKX   ERRLAB,E1,E2
                                .NLIST
                                .NLIST  MEB
                                .LIST   ME
                                .LIST
                                CALL    TALKER                  ; INITIATE SDI INTERCHANGE
                                TST     R3                      ; SEE IF ERROR OCCURRED
                                BEQ     12$                     ; IF NOT, BRANCH
                                BPL     11$                     ; IF SO, BRANCH
                                ERRHRD  E1;SEND COMMAND ERROR
                                BR      ERRLAB
                        11$:    ERRHRD  E2;RECEIVE COMMAND ERROR
                                BR      ERRLAB
                        12$:
                                .NLIST
                                .NLIST  ME
                                .LIST   MEB
                                .LIST
                                .ENDM
```

```
                                .SBTTL  START OF TEST CODE
                                ;THE FOLLOWING IS FOR DEBUG PURPOSES ONLY. CAN BE NOP OR BREAKPOINT.
                                
000714  114007                          CLR R0                          ;CHANGE TO BREAKPOINT FOR DEBUG
                                
                                ;INITIALIZE STACK
                                
000715  104206  001142                  MOV #STACK,SP                   ;SET UP STACK POINTER
000717  004555                          BR      START                   ; BRANCH OVER SUPPORT CODE
```

```
                                .SBTTL  RDSTAT - GET DRIVE'S REAL TIME DRIVE STATE
000720                          RDSTAT:
                                ;
                                ;RETURN DRIVE STATUS
                                ;STATUS RETURNED IN DM REGISTER 1
                                ;
000720                                  PUSH    <R3,R0>                 ; SAVE R3 AND R0
000720  100463                                                                  MOV R3,-(SP)
000721  100467                                                                  MOV R0,-(SP)
000722  104203  000003                 MOV     #3,R3                   ; ALLOW ONLY 3 ERRORS
000724  060007                  STATLP: XFC     STATUS                  ; GET DRIVE'S STATUS
000725  103201  014000                  BIC     #14000,R1               ; CLEAR ERROR PASSING BITS
000727  102201  000400                  BIT     #XMTERR,R1              ; CHECK XMIT ERRORS
000731  010737                          BEQ     STATOK                  ; IF NO ERRORS, BRANCH
000732  117403                          DEC     R3                      ; DECREMENT TRANSMIT ERROR COUNT
000733  050724                          BNE     STATLP                  ; IF ERROR COUNT INCOMPLETE, BRANCH
000734  104201  010000                  MOV     #10000,R1               ; FLAG AS TRANSMIT ERROR
000736  000746                          BR      STATEX                  ; BRANCH
000737  102201  000004          STATOK: BIT     #RCVERR,R1              ; RECIEVER ERRORS
000741  050746                          BNE     STATEX                  ; IF VALID, BRANCH
000742  117403                          DEC     R3                      ; DECREMENT ERROR COUNT
000743  050724                          BNE     STATLP                  ; IF ERROR COUNT NON-ZERO, BRANCH
000744  104201  014000                  MOV     #14000,R1               ; FLAG AS INVALID STATUS ERROR
000746                          STATEX: POP     <R0,R3>                 ; RESTORE R0, R3
000746  104267                                                                  MOV (SP)+,R0
000747  104263                                                                  MOV (SP)+,R3
000750  000000                          RETURN                          ; RETURN TO CALLING MODULE
```

```
                                .SBTTL  HOSTRQ - HOST REQUEST - REPORT ERRORS, MEGABYTES TRANSFERRED, ETC.
000751                          HOSTRQ:
                                ;
                                ;SEND REQUEST BUFFER TO HOST AND WAIT FOR RESPONSE.
                                ;CLEAR ARGUMENT AREA OF OUT BUFFER IN PREPARATION
                                ;FOR NEXT HOSTRQ CALL.
                                ;
                                ;INPUTS:
                                ;       R0 - HOST REQUEST NUMBER
                                ;       OUT BUFFER LOADED WITH DATA

000751                                  PUSH <R0,R1,R2>
000751  100467                                                                  MOV R0,-(SP)
000752  100461                                                                  MOV R1,-(SP)
000753  100462                                                                  MOV R2,-(SP)
000754  104070  001006                 MOV R0,OUT.RQ           ;STORE REQUEST NUMBER IN BUFFER
000756  104207  001006          SNDAGN: MOV #OUT.RQ,R0          ;SEND BUFFER TO HOST
000760  104201  000036                 MOV #BUFSIZ,R1
000762  060016                         XFC MRD
000763  115001                         TST R1                  ;CHECK FOR ERROR
000764  050756                         BNE SNDAGN              ;IF ERROR, TRY AGAIN
000765  104207  001044                 MOV #IN.RQ,R0           ;WAIT FOR RESPONSE FROM HOST
000767  104201  000036                 MOV #BUFSIZ,R1
000771  060017                         XFC MWR
000772  104207  001007                 MOV #OUT.01,R0          ;CLEAR ARGUMENT WORDS IN BUFFER
000774  104201  000034                 MOV #OUT.29-OUT.01,R1
000776  114002                         CLR R2
000777  100272          CLRBUF: MOV R2,(R0)+
001000  117401                         DEC R1
001001  030777                         BPL CLRBUF
001002                                  POP <R2,R1,R0>
001002  104262                                                                  MOV (SP)+,R2
001003  104261                                                                  MOV (SP)+,R1
001004  104267                                                                  MOV (SP)+,R0
001005  000000                         RETURN
```

```
                                ;STORAGE AREA FOR MAINTENANCE WRITE AND READ BUFFERS
                                ;
                                ;OUT BUFFER - DATA TO SEND TO HOST
                                ;
001006  000000  OUT.RQ: .WORD 0         ;HOST REQUEST CODE
001007  000000  OUT.01: .WORD 0         ;DATA ARGUMENT  1
001010  000000  OUT.02: .WORD 0         ;DATA ARGUMENT  2
001011  000000  OUT.03: .WORD 0         ;DATA ARGUMENT  3
001012  000000  OUT.04: .WORD 0         ;DATA ARGUMENT  4
001013  000000  OUT.05: .WORD 0         ;DATA ARGUMENT  5
001014  000000  OUT.06: .WORD 0         ;DATA ARGUMENT  6
001015  000000  OUT.07: .WORD 0         ;DATA ARGUMENT  7
001016  000000  OUT.08: .WORD 0         ;DATA ARGUMENT  8
001017  000000  OUT.09: .WORD 0         ;DATA ARGUMENT  9
001020  000000  OUT.10: .WORD 0         ;DATA ARGUMENT 10
001021  000000  OUT.11: .WORD 0         ;DATA ARGUMENT 11
001022  000000  OUT.12: .WORD 0         ;DATA ARGUMENT 12
001023  000000  OUT.13: .WORD 0         ;DATA ARGUMENT 13
001024  000000  OUT.14: .WORD 0         ;DATA ARGUMENT 14
001025  000000  OUT.15: .WORD 0         ;DATA ARGUMENT 15
001026  000000  OUT.16: .WORD 0         ;DATA ARGUMENT 16
001027  000000  OUT.17: .WORD 0         ;DATA ARGUMENT 17
001030  000000  OUT.18: .WORD 0         ;DATA ARGUMENT 18
001031  000000  OUT.19: .WORD 0         ;DATA ARGUMENT 19
001032  000000  OUT.20: .WORD 0         ;DATA ARGUMENT 20
001033  000000  OUT.21: .WORD 0         ;DATA ARGUMENT 21
001034  000000  OUT.22: .WORD 0         ;DATA ARGUMENT 22
001035  000000  OUT.23: .WORD 0         ;DATA ARGUMENT 23
001036  000000  OUT.24: .WORD 0         ;DATA ARGUMENT 24
001037  000000  OUT.25: .WORD 0         ;DATA ARGUMENT 25
001040  000000  OUT.26: .WORD 0         ;DATA ARGUMENT 26
001041  000000  OUT.27: .WORD 0         ;DATA ARGUMENT 27
001042  000000  OUT.28: .WORD 0         ;DATA ARGUMENT 28
001043  000000  OUT.29: .WORD 0         ;DATA ARGUMENT 29

                                ;IN BUFFER - DATA RECEIVED FROM HOST
                                ;
001044  000000  IN.RQ:  .WORD 0         ;HOST REQUEST CODE (ECHO)
001045  000000  IN.01:  .WORD 0         ;DATA ARGUMENT  1
001046  000000  IN.02:  .WORD 0         ;DATA ARGUMENT  2
001047  000000  IN.03:  .WORD 0         ;DATA ARGUMENT  3
001050  000000  IN.04:  .WORD 0         ;DATA ARGUMENT  4
001051  000000  IN.05:  .WORD 0         ;DATA ARGUMENT  5
001052  000000  IN.06:  .WORD 0         ;DATA ARGUMENT  6
001053  000000  IN.07:  .WORD 0         ;DATA ARGUMENT  7
001054  000000  IN.08:  .WORD 0         ;DATA ARGUMENT  8
001055  000000  IN.09:  .WORD 0         ;DATA ARGUMENT  9
001056  000000  IN.10:  .WORD 0         ;DATA ARGUMENT 10
001057  000000  IN.11:  .WORD 0         ;DATA ARGUMENT 11
001060  000000  IN.12:  .WORD 0         ;DATA ARGUMENT 12
001061  000000  IN.13:  .WORD 0         ;DATA ARGUMENT 13
001062  000000  IN.14:  .WORD 0         ;DATA ARGUMENT 14
001063  000000  IN.15:  .WORD 0         ;DATA ARGUMENT 15
001064  000000  IN.16:  .WORD 0         ;DATA ARGUMENT 16
001065  000000  IN.17:  .WORD 0         ;DATA ARGUMENT 17
001066  000000  IN.18:  .WORD 0         ;DATA ARGUMENT 18
001067  000000  IN.19:  .WORD 0         ;DATA ARGUMENT 19
```

```
     58 001070  000000          IN.20:  .WORD 0                 ;DATA ARGUMENT 20
     59 001071  000000          IN.21:  .WORD 0                 ;DATA ARGUMENT 21
     60 001072  000000          IN.22:  .WORD 0                 ;DATA ARGUMENT 22
     61 001073  000000          IN.23:  .WORD 0                 ;DATA ARGUMENT 23
     62 001074  000000          IN.24:  .WORD 0                 ;DATA ARGUMENT 24
     63 001075  000000          IN.25:  .WORD 0                 ;DATA ARGUMENT 25
     64 001076  000000          IN.26:  .WORD 0                 ;DATA ARGUMENT 26
     65 001077  000000          IN.27:  .WORD 0                 ;DATA ARGUMENT 27
     66 001100  000000          IN.28:  .WORD 0                 ;DATA ARGUMENT 28
     67 001101  000000          IN.29:  .WORD 0                 ;DATA ARGUMENT 29
     68         000036          BUFSIZ  =       . - IN.RQ       ;SIZE OF BUFFER
```

HOSTRQ - HOST REQUEST - REPORT ERRORS, MEGABYTES TRANSFERRED,

```
      1                                         .IF     NE,TEST4
```

HOSTRQ - HOST REQUEST - REPORT ERRORS, MEGABYTES TRANSFERRED,

```
     49                                         .ENDC
```

```
                                ;STACK AREA
                                
001102  123456                          .WORD 123456            ;END MARKER FOR STACK
001103                          .BLKW 31.               ;STACK
001142  123456                  STACK:  .WORD 123456            ;MARKER FOR STACK UNDERFLOW
```

```
                                .SBTTL  CMPEDC - EDC CALCULATION ROUTINE
001143                          CMPEDC:
                                ;
                                ;       THIS WILL COMPUTE THE EDC OF A SECTOR POINTED TO BY R0, RETURNING
                                ;       THE VALUE IN R2
                                ;
001143                                  PUSH    <R4,R1,R0>      ; SAVE R1 AND R0 AND R4
001143  100464                                                                  MOV R4,-(SP)
001144  100461                                                                  MOV R1,-(SP)
001145  100467                                                                  MOV R0,-(SP)
001146  104201  000400                 MOV     #256.,R1        ; COMPUTE OVER 256 WORDS
001150  104202  000105                 MOV     #INTEDC,R2      ; MOVE INITIAL EDC VALUE TO R2
001152  021157                         CALL    EDC             ; COMPUTE EDC
001153                                  POP     <R0,R1,R4>      ; RESTORE R1 AND R0 AND R4
001153  104267                                                                  MOV (SP)+,R0
001154  104261                                                                  MOV (SP)+,R1
001155  104264                                                                  MOV (SP)+,R4
001156  000000                         RETURN
001157                          EDC:
                                ;
                                ;       THIS WILL COMPUTE THE EDC VALUE FOR ANY NUMBER OF WORDS
                                ;
                                ;       R0 POINTS TO BUFFER, R1 HAS THE COUNT, AND R2 HAS THE INITIAL
                                ;       EDC VALUE
                                ;
001157  104274                 EDCLOP: MOV     (R0)+,R4        ; GET A WORD
001160                                  RXOR    R4,R2           ; EXCLUSIVE OR THE WORD WITH THE CURRENT EDC VALUE
001160  100462                         MOV     R2,-(SP)                ; SAVE REGISTER R2
001161  103042                         BIC     R4,R2           ; CLEAR COORESPONDING BITS IN R2
001162  103264                         BIC     (SP)+,R4                ; CLEAR COORESPONDING BITS IN R4
001163  101042                         BIS     R4,R2           ; OR WHAT'S LEFT
001164  105022                         ADD     R2,R2           ; SHIFT R2 LEFT BY 1
001165  041170                         BCC     ZEREDC          ; IF THE HIGH BIT WAS CLEAR, BRANCH
001166  101202  000001                 BIS     #1,R2           ; SET LO BIT TO - OLD HI BIT
001170  117401                 ZEREDC: DEC     R1              ; DECREMENT COUNT
001171  051157                         BNE     EDCLOP          ; IF COUNT UNEXPIRED, BRANCH
001172  000000                         RETURN
```

```
                        .SBTTL  RTDS    - REAL TIME DRIVE STATE ROUTINE
001173  104652  000022 RTDS:   MOV     U.MASK(R5),R2   ; MOVE MASK TO R2
001175  020720                CALL    RDSTAT          ; GET DRIVE STATUS
001176  114002                CLR     R2              ; FLAG AS NO ERRORS
001177  102201  010000        BIT     #10000,R1       ; TEST STATUS FOR ERROR
001201  011203                BEQ     RTDSEX          ; IF NO ERROR, BRANCH
001202  115402                INC     R2              ; MARK AS ERROR
001203  000000 RTDSEX: RETURN
```

```
                                .SBTTL  TALK   - SDI LEVEL 2 INTERCHANGE ROUTINE
001204                          TALK:
                                ;
                                ;       TALK SENDS THE COMMAND TO THE DRIVE.  IF AN ERROR OCCURRS, R2 IS
                                ;       RETURNED NONZERO
                                ;
001204                          PUSH    R4              ; SAVE POINTER TO SUBUNIT PARAMETERS
001204  100464                                                          MOV R4,-(SP)
001205  104652  000022          MOV     U.MASK(R5),R2   ; GET UNIT SDI SELECT MASK
001207  104237                  MOV     (R3)+,R0        ; POINTS TO SDI COMMAND BUFFER
001210  104231                  MOV     (R3)+,R1        ; LOAD BYTE COUNT
001211  060004                  XFC     SEND            ; SEND SDI COMMAND
001212  115001                  TST     R1              ; SEE IF SDI COMMAND SENT SUCESSFULLY
001213  011240                  BEQ     10$             ; IF SO, BRANCH
001214                          HARDER  70,             ; REPORT
001214  104200  007230  001012                                  MOV     #ER70,OUT.04
001217  104200  000106  001010                                  MOV     #70,OUT.02
001222  101200  001000  001010                                  BIS     #ERHARD,OUT.02
001225  104200  001225  001007                                  MOV     #.,OUT.01
001230  104202  000014                                          MOV     #ERRMC,R2
001232  104020  001006                                          MOV     R2,OUT.RQ
001234                          ENDERR  6
001234  101200  000006  002357                                  BIS     #6,ERRPOS       ; SET THE POSITION
001237  001453                  BR      SNDEXT          ; BRANCH
001240  106203  001532  10$:    CMP     #LONG,R3        ; SEE IF LONG TIMEOUT TO BE USED
001242  071246                  BMI     11$             ; IF SO, BRANCH
001243  104654  000030          MOV     U.SDIS(R5),R4   ; R4 HAS SHORT TIMEOUT
001245  001250                  BR      1$              ; BRANCH
001246  104654  000031  11$:    MOV     U.SDIL(R5),R4   ; R4 HAS LONG TIMEOUT
001250  104137  1$:    MOV     (R3),R0         ; POINT TO RECEIVE BUFFER
001251  104631  000001          MOV     1(R3),R1        ; NUMBER OF WORDS IN RESPONSE
001253  060005                  XFC     RCV             ; RECEIVE SDI RESPONSE
001254  115001                  TST     R1              ; SEE IF SDI RESPONSE RECEIVED SUCESSFULLY
001255  011422                  BEQ     2$              ; IF SO, BRANCH
001256  106201  000001          CMP     #1,R1           ; SEE IF TIMEOUT
001260  051304                  BNE     3$              ; IF NOT, BRANCH
001261  117404                  DEC     R4              ; DECREMENT TIMEOUT VALUE
001262  051250                  BNE     1$              ; IF TIMEOUT UNEXPIRED, BRANCH
001263                          HARDER  71,             ; FLAG AS ERROR
001263  104200  007311  001012                                  MOV     #ER71,OUT.04
001266  104200  000107  001010                                  MOV     #71,OUT.02
001271  101200  001000  001010                                  BIS     #ERHARD,OUT.02
001274  104200  001274  001007                                  MOV     #.,OUT.01
001277  104202  000014                                          MOV     #ERRMC,R2
001301  104020  001006                                          MOV     R2,OUT.RQ
001303  001416                  BR      7$              ; BRANCH
001304  110601  3$:    ROR     R1              ; ROTATE BITS INTO POSITION TO TEST
001305  110601                  ROR     R1              ; SEE IF FIRST WORD NOT START FRAME
001306  041330                  BCC     4$              ; IF NOT, BRANCH
001307                          HARDER  72,             ; REPORT FIRST WORD NOT START FRAME
001307  104200  007373  001012                                  MOV     #ER72,OUT.04
001312  104200  000110  001010                                  MOV     #72,OUT.02
001315  101200  001000  001010                                  BIS     #ERHARD,OUT.02
001320  104200  001320  001007                                  MOV     #.,OUT.01
001323  104202  000014                                          MOV     #ERRMC,R2
001325  104020  001006                                          MOV     R2,OUT.RQ
001327  001416                  BR      7$              ; BRANCH
```

```
     38 001330  110601                    4$:   ROR     R1          ; SEE IF FRAMING ERROR
     39 001331  041353                          BCC     5$          ; IF NOT, BRANCH
     40 001332                                  HARDFR  73,         ; REPORT FRAMING ERROR
        001332  104200  007472  001012                                      MOV     #ER73,OUT.04
        001335  104200  000111  001010                                      MOV     #73,OUT.02
        001340  101200  001000  001010                                      BIS     #ERHARD,OUT.02
        001343  104200  001343  001007                                      MOV     #.,OUT.01
        001346  104202  000014                                              MOV     #ERRMC,R2
        001350  104020  001006                                              MOV     R2,OUT.RQ
     41 001352  001416                          BR      7$          ; BRANCH
     42 001353  110601                    5$:   ROR     R1          ; SEE IF CHECKSUM ERROR
     43 001354  041376                          BCC     6$          ; IF NOT (BUFFER TOO SMALL), BRANCH
     44 001355                                  HARDER  74,         ; REPORT CHECKSUM ERROR
        001355  104200  007562  001012                                      MOV     #ER74,OUT.04
        001360  104200  000112  001010                                      MOV     #74,OUT.02
        001363  101200  001000  001010                                      BIS     #ERHARD,OUT.02
        001366  104200  001366  001007                                      MOV     #.,OUT.01
        001371  104202  000014                                              MOV     #ERRMC,R2
        001373  104020  001006                                              MOV     R2,OUT.RQ
     45 001375  001416                          BR      7$          ; BRANCH
     46 001376                            6$:   HARDER  75,         ; REPORT BUFFER TO SMALL
        001376  104200  007652  001012                                      MOV     #ER75,OUT.04
        001401  104200  000113  001010                                      MOV     #75,OUT.02
        001404  101200  001000  001010                                      BIS     #ERHARD,OUT.02
        001407  104200  001407  001007                                      MOV     #.,OUT.01
        001412  104202  000014                                              MOV     #ERRMC,R2
        001414  104020  001006                                              MOV     R2,OUT.RQ
     47 001416                            7$:   ENDERR  6           ; FLAG END OF REPORTING BUFFER
        001416  101200  000006  002357                                      BIS     #6,ERRPOS       ; SET THE POSITION
     48 001421  001453                          BR      SNDEXT
     49 001422  114002                    2$:   CLR     R2          ; FLAG AS NO ERROR OCCURED
     50 001423  106637  000002                  CMP     2(R3),R0    ; SEE IF COMMAND ACCEPTED
     51 001425  011453                          BEQ     SNDEXT      ; IF SO, BRANCH
     52 001426                                  HARDER  76,         ; REPORT
        001426  104200  007746  001012                                      MOV     #ER76,OUT.04
        001431  104200  000114  001010                                      MOV     #76,OUT.02
        001434  101200  001000  001010                                      BIS     #ERHARD,OUT.02
        001437  104200  001437  001007                                      MOV     #.,OUT.01
        001442  104202  000014                                              MOV     #ERRMC,R2
        001444  104020  001006                                              MOV     R2,OUT.RQ
     53 001446                                  CERROR  6,R0        ; REPORT FURTHER ERRORS
        001446  104070  001014                                              MOV     R0,OUT.06
     54 001450  104200  100007  002357          MOV     #100007,ERRPOS ; FLAG AS STATUS GOOD
     55 001453                            SNDEXT: POP     R4          ; RESTORE R4
        001453  104264                                                      MOV (SP)+,R4
     56 001454  000000                          RETURN
```

```
                                .SBTTL  SDI PROTOCOL MESSAGE TABLES
                                ;
                                ;       MESSAGE TABLES
                                ;
001455                          CR.CLR: MSG DRC,2,ST,7,COMPLT   ; DRIVE CLEAR
001455  001550                          .WORD DRC                       ;ADDRESS OF COMMAND
001456  000002                          .WORD 2                 ;SIZE OF COMMAND IN BYTES
001457  001570                          .WORD ST                        ;ADDRESS OF REPLY
001460  000007                          .WORD 7                 ;SIZE OF REPLY IN WORDS
001461  000176                          .WORD COMPLT                    ; SUCCESSFUL COMPLETION CODE
001462                          CR.MOD: MSG MOD,3,ST,7,COMPLT   ; CHANGE MODE
001462  001546                          .WORD MOD                       ;ADDRESS OF COMMAND
001463  000003                          .WORD 3                 ;SIZE OF COMMAND IN BYTES
001464  001570                          .WORD ST                        ;ADDRESS OF REPLY
001465  000007                          .WORD 7                 ;SIZE OF REPLY IN WORDS
001466  000176                          .WORD COMPLT                    ; SUCCESSFUL COMPLETION CODE
001467                          CR.GCR: MSG GCR,1,ST,11.,CHRRES ; GET CHARACTERISTICS
001467  001552                          .WORD GCR                       ;ADDRESS OF COMMAND
001470  000001                          .WORD 1                 ;SIZE OF COMMAND IN BYTES
001471  001570                          .WORD ST                        ;ADDRESS OF REPLY
001472  000013                          .WORD 11.                       ;SIZE OF REPLY IN WORDS
001473  000170                          .WORD CHRRES                    ; SUCCESSFUL COMPLETION CODE
001474                          CR.SCR: MSG SCR,2,ST,19.,SBCRES ; GET SUBUNIT CHARACTERISTICS
001474  001553                          .WORD SCR                       ;ADDRESS OF COMMAND
001475  000002                          .WORD 2                 ;SIZE OF COMMAND IN BYTES
001476  001570                          .WORD ST                        ;ADDRESS OF REPLY
001477  000023                          .WORD 19.                       ;SIZE OF REPLY IN WORDS
001500  000167                          .WORD SBCRES                    ; SUCCESSFUL COMPLETION CODE
001501                          CR.GST: MSG GST,1,ST,7,STSRES   ; GET STATUS
001501  001555                          .WORD GST                       ;ADDRESS OF COMMAND
001502  000001                          .WORD 1                 ;SIZE OF COMMAND IN BYTES
001503  001570                          .WORD ST                        ;ADDRESS OF REPLY
001504  000007                          .WORD 7                 ;SIZE OF REPLY IN WORDS
001505  000366                          .WORD STSRES                    ; SUCCESSFUL COMPLETION CODE
001506                          CR.SEK: MSG INS,6,ST,7,COMPLT   ; INITIATE SEEK
001506  001560                          .WORD INS                       ;ADDRESS OF COMMAND
001507  000006                          .WORD 6                 ;SIZE OF COMMAND IN BYTES
001510  001570                          .WORD ST                        ;ADDRESS OF REPLY
001511  000007                          .WORD 7                 ;SIZE OF REPLY IN WORDS
001512  000176                          .WORD COMPLT                    ; SUCCESSFUL COMPLETION CODE
001513                          CR.ONL: MSG ONL,2,ST,7,COMPLT   ; BRING DRIVE ONLINE
001513  001566                          .WORD ONL                       ;ADDRESS OF COMMAND
001514  000002                          .WORD 2                 ;SIZE OF COMMAND IN BYTES
001515  001570                          .WORD ST                        ;ADDRESS OF REPLY
001516  000007                          .WORD 7                 ;SIZE OF REPLY IN WORDS
001517  000176                          .WORD COMPLT                    ; SUCCESSFUL COMPLETION CODE
001520                          CR.ERR: MSG ERR,2,ST,7,COMPLT   ; ERROR RECOVERY
001520  001564                          .WORD ERR                       ;ADDRESS OF COMMAND
001521  000002                          .WORD 2                 ;SIZE OF COMMAND IN BYTES
001522  001570                          .WORD ST                        ;ADDRESS OF REPLY
001523  000007                          .WORD 7                 ;SIZE OF REPLY IN WORDS
001524  000176                          .WORD COMPLT                    ; SUCCESSFUL COMPLETION CODE
001525                          CR.DIS: MSG DIS,2,ST,7,COMPLT   ; DISCONNECT DRIVE
001525  001544                          .WORD DIS                       ;ADDRESS OF COMMAND
001526  000002                          .WORD 2                 ;SIZE OF COMMAND IN BYTES
001527  001570                          .WORD ST                        ;ADDRESS OF REPLY
001530  000007                          .WORD 7                 ;SIZE OF REPLY IN WORDS
```

```
                001531  000176                          .WORD COMPLT                    ; SUCCESSFUL COMPLETION CODE
     14         001532                          LONG:                   ; ALL COMMANDS BEYOND THIS POINT ARE LONG TIMEOUT
     15         001532                          CR.INR: MSG INR,1,ST,7,COMPLT   ; INITIATE RECALIBRATE
                001532  001557                          .WORD INR                       ;ADDRESS OF COMMAND
                001533  000001                          .WORD 1                 ;SIZE OF COMMAND IN BYTES
                001534  001570                          .WORD ST                        ;ADDRESS OF REPLY
                001535  000007                          .WORD 7                 ;SIZE OF REPLY IN WORDS
                001536  000176                          .WORD COMPLT                    ; SUCCESSFUL COMPLETION CODE
     16         001537                          CR.RUN: MSG RUN,1,ST,7,COMPLT   ; INITIATE LOAD
                001537  001556                          .WORD RUN                       ;ADDRESS OF COMMAND
                001540  000001                          .WORD 1                 ;SIZE OF COMMAND IN BYTES
                001541  001570                          .WORD ST                        ;ADDRESS OF REPLY
                001542  000007                          .WORD 7                 ;SIZE OF REPLY IN WORDS
                001543  000176                          .WORD COMPLT                    ; SUCCESSFUL COMPLETION CODE
     17                                         ;
     18                                         ;       LEVEL 2 COMMAND MESSAGE DATA STRUCTURES
     19                                         ;
     20         001544     000      204        DIS:    .BYTE   0,DISCON        ; DISCONNECT DRIVE
     21         001545  000000                  WRITBT: .WORD   0               ; DO NOT SPIN DOWN DRIVE *****
     22         001546     000      201        MOD:    .BYTE   0,CHGMOD        ; CHANGE MODE
     23         001547  000000                  WRITBT: .WORD   0               ; MODE
     24         001550     000      005        DRC:    .BYTE   0,DRVCLR        ; DRIVE CLEAR
     25         001551  000000                  DCLR:   .WORD   0
     26         001552     000      207        GCR:    .BYTE   0,GETCHR        ; GET CHARACTERISTICS WITH A
     27         001553     000      210        SCR:    .BYTE   0,GETSUB        ; GET SUBUNIT CHARACTERISTICS
     28         001554  000000                  SUBUNT: .WORD   0               ; SUBUNIT SELECTION IN LOW ORDER BYTE
     29         001555     000      011        GST:    .BYTE   0,GETSTA        ; GET STATUS
     30         001556     000      014        RUN:    .BYTE   0,DRVRUN        ; DRIVE RUN
     31         001557     000      216        INR:    .BYTE   0,IRECLB        ; INITIATE RECALIBRATE
     32         001560     000      012        INS:    .BYTE   0,INSEEK        ; INITIATE SEEK
     33         001561  000000                  LOCYL:  .WORD   0               ; INS CYLINDER/HEAD ARGUMENTS
     34         001562  000000                          .WORD   0               ; HI CYLINDER
     35         001563  000000                          .WORD   0               ; GROUP
     36         001564     000      006        ERR:    .BYTE   0,ERECOV        ; ERROR RECOVERY
     37         001565  000000                  ERRLEV: .WORD   0
     38         001566     000      213        ONL:    .BYTE   0,DRVONL        ; ONLINE COMMAND
     39         001567  000040                          .WORD   40              ; MAXIMUM TIMEOUT VALUE
     40         001570                          ST:     .BLKW   19.
```

BLKCHK - SEE IF A BLOCK WITH ERROR IS KNOWN TO BE BAD

```
                                .SBTTL  BLKCHK - SEE IF A BLOCK WITH ERROR IS KNOWN TO BE BAD
001613                          BLKCHK:
                                ;
                                ;       BLKCHK CHECKS THE BLOCK IN U.LBN TO ASSURE THAT IT IS NOT A BAD
                                ;       BLOCK.  IF SO, R2 IS RETURNED NONZERO
                                ;
001613                          PUSH    <R2,R1,R0>      ; SAVE ALL REG'S
001613  100462                                                          MOV R2,-(SP)
001614  100461                                                          MOV R1,-(SP)
001615  100467                                                          MOV R0,-(SP)
001616  104641  000010          MOV     S.BADP(R4),R1   ; GET BAD BLOCK POINTER
001620  011634                  BEQ     1$              ; IF NO BAD BLOCKS, BRANCH
001621  104202  000002          MOV     #RW.LOW,R2      ; POINT TO LBN TO BE TESTED
001623  105072                  ADD     R0,R2           ; POINT TO LBN TO BE TESTED
001624  021645          3$:     CALL    CMP2            ; COMPARE BAD BLOCK TO LBN
001625  011637                  BEQ     2$              ; IF EQUAL, BRANCH
001626  071634                  BMI     1$              ; IF LIST GREATER THAN BLOCK, IT'S OK
001627  105201  000002          ADD     #2,R1           ; POINT TO NEXT BAD BLOCK
001631  104617  177777          MOV     -1(R1),R0       ; SEE IF EOL
001633  031624                  BPL     3$              ; IF NOT, BRANCH
001634  104207  000001  1$:     MOV     #1,R0           ; SET UP FOR CARRY TO BE SET (UNKNOWN BAD BLOCK)
001636  001640                  BR      4$
001637  114007          2$:     CLR     R0              ; SET UP FOR THE CARRY TO BE CLEAR (BAD BLOCK KNOWN)
001640  110607          4$:     ROR     R0              ; SET CARRY TO REFLECT KNOWLEDGE OF BAD BLOCK
001641                          POP     <R0,R1,R2>      ; RESTORE
001641  104267                                                          MOV (SP)+,R0
001642  104261                                                          MOV (SP)+,R1
001643  104262                                                          MOV (SP)+,R2
001644  000000                  RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                .SBTTL  CMP2    - 24 BIT COMPARE
001645                          CMP2:
                                ;
                                ;       CMP2 COMPARES A 24 BIT NUMBER POINTED TO BY R2 TO A 24 BIT NUMBER
                                ;       POINTED TO BY R1.  BOTH NUMBERS ARE LOW ORDER WORD FOLLOWED BY HIGH
                                ;       WORD (POINTER TO LOW WORD) AND THE HIGH 8 BITS <31:24> OF THE
                                ;       WORD POINTED TO BY R1 ARE STRIPPED OFF (THIS IS TO ELIMINATE THE
                                ;       END-OF-LIST FLAG ON THE B/E SETS AND BAD BLOCKS)
                                ;
001645  104617  000001                  MOV     1(R1),R0        ; MOVE HIGH ORDER WORD TO R0
001647  103207  170000                  BIC     #170000,R0      ; CLEAR UNUSED BITS
001651  106627  000001                  CMP     1(R2),R0        ; COMPARE HIGH ORDER WORRD TO R0
001653  051656                          BNE     CMPEXT          ; IF UNEQUAL, BRANCH
001654  104117                          MOV     (R1),R0         ; MOVE LOW ORDER WORD TO R0
001655  106127                          CMP     (R2),R0         ; COMPARE LOW ORDER WORD TO R0
001656  000200                  CMPEXT: RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                        .SBTTL  BUILDP - BUILD THE READ OR WRITE CHAIN LINKS
001657                                  BUILDP:
                                        ;
                                        ;       BUILD THE WHOLE READ/WRITE CHAIN
                                        ;
001657  104657  000033                  MOV     U.PARM(R5),R0   ; R0 HAS UNIT PARAMETERS
001661  104650  000017  002355          MOV     U.MSEC(R5),MAXSEC       ; GET NUMBER OF SECTORS TO READ/WRITE
001664  102207  000200                  BIT     #RBNBN,R0       ; SEE IF RBN TO BE READ/WRITTEN
001666  011676                          BEQ     1$              ; IF NOT, BRANCH
001667  104650  000041  002340          MOV     U.RBN(R5),CURBN ; MOVE LO ORDER TO CURRENT BN
001672  104650  000042  002341          MOV     U.RBN+1(R5),CURBN+1     ; MOVE HI ORDER TO CURRENT BN
001675  001704                          BR      4$              ; BRANCH
001676  104650  000037  002340  1$:     MOV     U.CBN(R5),CURBN         ; MOVE LO ORDER TO CURRENT BN
001701  104650  000040  002341          MOV     U.CBN+1(R5),CURBN+1     ; MOVE LO ORDER TO CURRENT BN
001704  104300  002473  002474  4$:     MOV     MEMPOL,TMEMPL   ; START OF BUILD (INITILIZE MEM POOL)
001707  114007                          CLR     R0              ; FORCE THE CALCULATION TABLE TO BE FILLED
001710  104070  002352                  MOV     R0,CHAINS       ; MARK CHAIN AS EMPTY
001712  022141                  5$:     CALL    CALC            ; CALCULATE CYL, TRK AND GRP
001713  115002                          TST     R2              ; SEE IF AN ERROR OCCURRED
001714  051746                          BNE     8$              ; EXIT
001715  106650  000053  002344          CMP     U.CCYL(R5),CYL  ; SEE IF LO CYL MATCHES
001720  051745                          BNE     7$              ; IF NOT, BRANCH
001721  106650  000054  002345          CMP     U.CCYL+1(R5),CYL+1      ; SEE IF HI CYL MATCHES
001724  051745                          BNE     7$              ; IF NOT, BRANCH
001725  106650  000055  002346          CMP     U.CGRP(R5),GROUP        ; SEE IF GROUP MATCHES
001730  051745                          BNE     7$              ; IF NOT, BRANCH
001731  021747                          CALL    LINK            ; LINK THIS SECTOR INTO THE CHAIN
001732  107200  000001  002355          SUB     #1,MAXSEC       ; DECREMENT SECTOR COUNT
001735  011745                          BEQ     7$              ; IF COUNT COMPLETE, BRANCH
001736  105200  000001  002340          ADD     #1,CURBN        ; ADD ONE FROM LO CURRENT SECTOR
001741  041712                          BCC     5$              ; IF NO CARRY, BRANCH
001742  115400  002341                  INC     CURBN+1         ; PROPOGATE CARRY
001744  001712                          BR      5$              ; BRANCH
001745  114002                  7$:     CLR     R2              ; NO ERRORS
001746  000000                  8$:     RETURN                  ; RETURN TO CALLING PROGRAM
```

LINK - BUILD A LINK (NODE) IN THE READ/WRITE CHAIN

```
                                .SBTTL  LINK    - BUILD A LINK (NODE) IN THE READ/WRITE CHAIN
002000'                         LINK:
                                ;
                                ;       LINK WILL CREATE THE LINKS FOR THE CHAIN
                                ;
001747  104653  000033                  MOV     U.PARM(R5),R3   ; R3 HAS UNIT PARAMETERS
001751  104307  002352                  MOV     CHAINS,R0       ; R0 POINTS TO START OF CHAIN
001753  051772                          BNE     2$              ; IF THE CHAIN IS ALLREADY STARTED, BRANCH
001754  102203  000100                  BIT     #REDWRT,R3      ; SEE IF READ IN PROGRESS
001756  011764                          BEQ     1$              ; IF SO, BRANCH
001757  104207  000401                  MOV     #WBUFLN,R0      ; R0 HAS NUMBER OF WORDS IN WRITE BUFFER
001761  022112                          CALL    ALLOCM          ; ALLOCATE THE MEMORY
001762  104070  002356                  MOV     R0,SECPTR       ; SECPTR POINTS TO WRITE BUFFER
001764  104207  000007          1$:     MOV     #LINKLN,R0      ; R0 HAS LENGTH OF CHAIN LINK
001766  022112                          CALL    ALLOCM          ; ALLOCATE THE MEMORY
001767  104070  002352                  MOV     R0,CHAINS       ; CHAINS POINTS TO FIRST LINK
001771  002012                          BR      4$              ; BRANCH
001772  104171                  2$:     MOV     (R0),R1         ; R1 HAS NEXT LINK POINTER
001773  102201  100000                  BIT     #EOC,R1         ; SEE IF THIS LINK IS THE LAST
001775  052002                          BNE     3$              ; IF SO, BRANCH
001776  103201  170000                  BIC     #^CHBHINB,R1    ; CLEAR UNUSED BITS
002000  104017                          MOV     R1,R0           ; R0 POINTS TO NEXT LINK
002001  001772                          BR      2$              ; LOOP
002002  104072                  3$:     MOV     R0,R2           ; SAVE R0
002003  104207  000007                  MOV     #LINKLN,R0      ; R0 HAS LENGTH OF LINK
002005  022112                          CALL    ALLOCM          ; ALLOCATE MEMORY FOR LINK
002006  103201  100000                  BIC     #EOC,R1         ; CLEAR THE END-OF-CHAIN FLAG
002010  101071                          BIS     R0,R1           ; PUT IN POINTER TO NEXT LINK
002011  100121                          MOV     R1,(R2)         ; PUT POINTER BACK IN LAST LINK
002012  022014                  4$:     CALL    FILLIN          ; PUT ALL VALUES IN NEW LINK
002013  000000                          RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                .SBTTL  FILLIN - FILL THE READ/WRITE CHAIN LINK (NODE) WITH REQUIRED INFORMATION
002014                          FILLIN:
                                ;
                                ;       FILLIN BUILDS THE PARAMETERS THE UDA REQUIRES TO WRITE OR READ A BLOCK
                                ;
002014  104302  002347                 MOV     TRACK,R2        ; GET TRACK (HEAD) NUMBER
002016  102203  000100                 BIT     #REDWRT,R3      ; SEE IF READ IN PROGRESS
002020  012032                         BEQ     1$              ; IF SO, BRANCH
002021  101202  122400                 BIS     #WREAL,R2       ; BUILD WRITE REAL TIME COMMAND
002023  100672  000004                 MOV     R2,RW.CMD(R0)   ; MOVE TO SDI REAL TIME COMMAND AREA
002025  104202  140000                 MOV     #WSTOP,R2       ; MOVE LAST BLOCK FLAG TO R2
002027  104301  002356                 MOV     SECPTR,R1       ; R1 POINTS TO WRITE BUFFER
002031  002046                         BR      2$              ; BRANCH
002032  101202  013400         1$:     BIS     #RREAL,R2       ; BUILD READ REAL TIME COMMAND
002034  100672  000004                 MOV     R2,RW.CMD(R0)   ; MOVE TO SDI REAL TIME COMMAND AREA
002036  104202  100000                 MOV     #RSTOP,R2       ; MOVE LAST BLOCK FLAG TO R2
002040                                  PUSH    R0              ; SAVE R0
002040  100467                                                                  MOV R0,-(SP)
002041  104207  000415                 MOV     #RBUFLN,R0      ; SIZE OF READ BUFFER
002043  022112                         CALL    ALLOCM          ; ALLOCATE BUFFER
002044  104071                         MOV     R0,R1           ; R1 POINTS TO BUFFER
002045                                  POP     R0              ; RESTORE R0
002045  104267                                                                  MOV (SP)+,R0
002046  100672  000000         2$:     MOV     R2,RW.STAT(R0)  ; MOVE LAST BLOCK FLAG TO NEXT BLOCK POINTER
002050  100671  000001                 MOV     R1,RW.BUF(R0)   ; MOVE POINTER TO SECTOR TO POINTER AREA
002052  104202  002447                 MOV     #DUMSDI,R2      ; R2 POINTS TO DUMMY SDI AREA
002054  100672  000005                 MOV     R2,RW.SDI(R0)   ; MOVE R2 TO POINTER TO DUMMY SDI AREA
002056  104302  002350                 MOV     SECTOR,R2       ; R2 CONTAINS SECTOR TO BE WRITTEN
002060  100672  000006                 MOV     R2,RW.ANG(R0)   ; MOVE TO ANGLE FROM INDEX
002062  104302  002340                 MOV     CURBN,R2        ; MOVE LOW ORDER BN TO R2
002064  100672  000002                 MOV     R2,RW.LOW(R0)   ; MOVE LOW ORDER BN TO LOW EXPECTED HEADER
002066  104302  002341                 MOV     CURBN+1,R2      ; MOVE HIGH ORDER BN TO R2
002070  102203  000200                 BIT     #RBNBN,R3       ; SEE IF BN IS AN RBN
002072  052103                         BNE     5$              ; IF SO, BRANCH
002073  104643  000000                 MOV     S.PARM(R4),R3   ; GET SUBUNIT PARAMETERS
002075  102203  020000                 BIT     #DCYLS,R3       ; SEE IF USING DIAGNOSTIC CYLINDERS
002077  012105                         BEQ     6$              ; IF NOT, BRANCH
002100  101202  140000                 BIS     #HD.DBN,R2      ; DIAGNOSTIC HEADER
002102  002105                         BR      6$              ; BRANCH
002103  101202  060000         5$:     BIS     #HD.RBN,R2      ; REVECTORED HEADER
002105  101302  002335         6$:     BIS     HIBN,R2         ; ADD HI BN BITS
002107  100672  000003                 MOV     R2,RW.HI(R0)    ; MOVE R2 TO HI WORD EXPECTED HEADER
002111  000000                         RETURN                  ; RETURN TO CALLING PROGRAM
```

ALLOCM - ALLOCATE MEMORY FOR THE READ/WRITE BUFFERS AND CHAIN

```
                                        .SBTTL  ALLOCM - ALLOCATE MEMORY FOR THE READ/WRITE BUFFERS AND CHAIN
002112                                  ALLOCM:
                                        ;
                                        ;       ALLOCATE MEMORY FOR READ/WRITE BUFFERS
                                        ;
002112  107070  002474                          SUB     R0,TMEMPL       ; SUBTRACT LENGTH FROM MEMORY POOL
002114  104307  002474                          MOV     TMEMPL,R0       ; R0 POINTS TO MEMORY ALLOCATED
002116  000000                                  RETURN                  ; RETURN TO CALLING MODULE
```

```
                                        .SBTTL  BULDUM - BUILD THE DUMMY SDI CONTROL BLOCK FOR READS AND WRITES
002117                                  BULDUM:
                                        ;
                                        ;       BULDUM WILL BUILD THE DUMMY SDI CONTROL BLOCK
                                        ;
002117  104643  000005                          MOV     S.SCHR(R4),R3   ; R3 POINTS TO SUBUNIT CHARACTERISTICS
002121  104632  000004                          MOV     RBNTRK(R3),R2   ; GET RBNS/TRACK
002123  103202  177600                          BIC     #177600,R2      ; CLEAR UNUSED BITS
002125  105632  000011                          ADD     LBNTRK(R3),R2   ; ADD LBNS/TRACK GIVING SECTORS/TRACK
002127  105022                                  ADD     R2,R2           ; DOUBLE SECTORS/TRACK FOR HEADER COMPARE LIMIT
002130  103202  177001                          BIC     #177001,R2      ; CLEAR UNUSED BITS
002132  104020  002450                          MOV     R2,DUMSDI+D.LIMT        ; MOVE TO DUMMY SEARCH LIMIT
002134  107203  000005                          SUB     #5,R3           ; R3 POINTS TO SUB CHAR - 5
002136  104030  002451                          MOV     R3,DUMSDI+D.SCHR        ; MOVE TO DUMMY SUB CHAR POINTER
002140  000000                                  RETURN                  ; RETURN TO CALLING PROGRAM
```

CALC - CALCULATE THE CYL, GRP AND TRACK FOR THE GIVEN L/RBN

```
                                        .SBTTL  CALC   - CALCULATE THE CYL, GRP AND TRACK FOR THE GIVEN L/RBN
002141                                  CALC:
                                        ;
                                        ;       CALC CALCULATES THE CYLINDER, GROUP, TRACK AND SECTOR OF THE BLOCK
                                        ;
002141  115007                                 TST     R0              ; SEE IF CALCULATION AREA MUST BE SET UP
002142  052261                                 BNE     MOVOUT          ; IF NOT, BRANCH
002143  104070  002343                         MOV     R0,RBNFLG       ; SET RBN FLAG AS ZERO
002145  104643  000005                         MOV     S.SCHR(R4),R3   ; R3 POINTS TO SUBUNIT'S CHARACTERISTICS
002147  104647  000000                         MOV     S.PARM(R4),R0   ; GET SUBUNIT PARAMETERS
002151  102207  020000                         BIT     #DCYLS,R0       ; SEE IF USING THE DIAGNOSTIC CYLINDERS
002153  052224                                 BNE     MOVDBN          ; IF SO, BRANCH
002154  104637  000011                         MOV     LBNTRK(R3),R0   ; MOVE LBN'S PER TRACK TO R0
002156  103207  177400                         BIC     #HIBYTE,R0      ; CLEAR UNUSED BITS
002160  104651  000C33                         MOV     U.PARM(R5),R1   ; MOVE UNIT PARAMETERS TO R0
002162  102201  000200                         BIT     #RBNBN,R1       ; SEE IF BLOCK REVECTORED
002164  012202                                 BEQ     LNCYL           ; IF NOT, BRANCH
002165  104070  002343                         MOV     R0,RBNFLG       ; STORE LBN'S PER TRACK IN RBN FLAG AREA
002167  104637  000004                         MOV     RBNTRK(R3),R0   ; MOVE RBN'S PER TRACK TO R0
002171  103207  177600                         BIC     #177600,R0      ; CLEAR THE HIGHER BITS, LEAVING RBN'S/TRACK
002173  104070  002342                         MOV     R0,LRDTRK       ; MOVE RBN'S PER TRACK TO CALCULATION AREA
002175  104637  000003                         MOV     HIRBN(R3),R0    ; GET HI RBN
002177  103207  170377                         BIC     #HBLONB,R0      ; CLEAR UNUSED BITS
002201  002210                                 BR      LNCONT          ; BRANCH
002202  104070  002342          LNCYL:         MOV     R0,LRDTRK       ; MOVE LBN'S PER TRACK TO CALCULATION AREA
002204  104637  000002                         MOV     HILBN(R3),R0    ; GET HI LBN
002206  103207  170377                         BIC     #HBLONB,R0      ; CLEAR UNUSED BITS
002210  104070  002335          LNCONT:        MOV     R0,HIBN         ; SAVE
002212  114007                                 CLR     R0              ; LOW ORDER CYLINDER IS ZERO
002213  104070  002336                         MOV     R0,STACYL       ; SAVE LO ORDER STARTING CYLINDER
002215  104637  000001                         MOV     HICYL(R3),R0    ; R0 CONTAINS HI CYLINDER BITS
002217  103207  007777                         BIC     #HBHINB,R0      ; CLEAR UNUSED BITS
002221  104070  002337                         MOV     R0,STACYL+1     ; MOVE HI STARTING CYLINDER TO UNIT PARAMETERS
002223  002261                                 BR      MOVOUT          ; BRANCH
002224  104637  000004          MOVDBN:        MOV     RBNTRK(R3),R0   ; MOVE NUMBER OF RBN'S PER TRACK TO R0
002226  103207  177600                         BIC     #177600,R0      ; CLEAR THE HIGHER BITS, LEAVING RBN'S/TRACK
002230  104631  000011                         MOV     LBNTRK(R3),R1   ; ADD LBN'S/TRK, GIVING DBN'S/TRK
002232  103201  177400                         BIC     #HIBYTE,R1      ; CLEAR UNUSED BITS
002234  105017                                 ADD     R1,R0           ; ADD LBNS/TRK TO RBNS/TRK TO GIVE DBNS/TRK
002235  104070  002342                         MOV     R0,LRDTRK       ; SAVE DBNS/TRK IN COMPUTATIONAL AREA
002237  104647  000001                         MOV     S.SDCL(R4),R0   ; GET LO ORDER STARTING DIAGNOSTIC CYLINDER
002241  104070  002336                         MOV     R0,STACYL       ; SAVE LO ORDER CYL
002243  104647  000002                         MOV     S.SDCL+1(R4),R0 ; GET HI ORDER STARTING DIAGNOSTIC CYLINDER
002245  104070  002337                         MOV     R0,STACYL+1     ; SAVE HI ORDER CYL
002247  104637  000003                         MOV     HIDBN(R3),R0    ; GET HI DBN
002251  110607                                 ROR     R0              ; ROTATE TO CORRECT POSITION
002252  110607                                 ROR     R0              ; ROTATE TO CORRECT POSITION
002253  110607                                 ROR     R0              ; ROTATE TO CORRECT POSITION
002254  110607                                 ROR     R0              ; ROTATE TO CORRECT POSITION
002255  103207  170377                         BIC     #HBLONB,R0      ; CLEAR UNUSED BITS
002257  104070  002335                         MOV     R0,HIBN         ; SAVE
002261  104207  002336          MOVOUT:        MOV     #CALCSC,R0      ; POINT TO CALCULATION AREA
002263  104641  000005                         MOV     S.SCHR(R4),R1   ; POINT TO SUBUNIT CHAR TABLE
002265  060020                                 XFC     CVT             ; CALCULATE
002266  104012                                 MOV     R1,R2           ; SEE IF CORRECTLY CALCULATED
002267  012334                                 BEQ     CLXEXT          ; IF CORRECT, BRANCH
002270                                         SOFTER  2,<CURBN,CURBN+1,LRDTRK,RBNFLG>
```

CALC - CALCULATE THE CYL, GRP AND TRACK FOR THE GIVEN L/RBN

```
        002270  104200  000225  001012                                          MOV     #ER2,OUT.04
        002273  104300  002340  001013                                          MOV     CURBN,OUT.05
        002276  104300  002341  001014                                          MOV     CURBN+1,OUT.06
        002301  104300  002342  001015                                          MOV     LRDTRK,OUT.07
        002304  104300  002343  001016                                          MOV     RBNFLG,OUT.08
        002307  104200  000002  001010                                          MOV     #2,OUT.02
        002312  101200  001400  001010                                          BIS     #ERSOFT,OUT.02
        002315  104200  002315  001007                                          MOV     #.,OUT.01
        002320  104202  000013                                                  MOV     #ERRMES,R2
        002322  104020  001006                                                  MOV     R2,OUT.RQ
     58 002324                          ERRORC  <STACYL,STACYL+1>       ; REPORT ERROR
        002324  104300  002336  001017                                          MOV     STACYL,OUT.09
        002327  104300  002337  001020                                          MOV     STACYL+1,OUT.10
     59 002332                          FNDERR  0
        002332  114000  002357                                                  CLR     ERRPOS          ; CLEAR THE POSITION
     60 002334  000000              CLXEXT: RETURN                  ; RETURN TO CALLING PROGRAM
```

STORAGE AREA PATTERNS, DUMMY SDI AREA, OVERLAY INFORMATION AND

```
                                .SBTTL  STORAGE AREA PATTERNS, DUMMY SDI AREA, OVERLAY INFORMATION AND MORE
002335                          HIBN:   .BLKW   1                       ; HI BN BITS <27:24>

002336                          CALCSC:                                 ; CALCULATION AREA FOR CYL, GRP, TRK
002336                          STACYL: .BLKW   2                       ; STARTING CYLINDER NUMBER
002340                          CURBN:  .BLKW   2                       ; L/R/DBN
002342                          LRDTRK: .BLKW   1                       ; SECTORS PER TRACK
002343                          RBNFLG: .BLKW   1                       ; RBN FLAG
002344                          CYL:    .BLKW   2                       ; CYLINDER
002346                          GROUP:  .BLKW   1                       ; GROUP
002347                          TRACK:  .BLKW   1                       ; TRACK
002350                          SECTOR: .BLKW   1                       ; SECTOR
002351                          INDEX:  .BLKW   1                       ; INDEX

002352                          CHAINS: .BLKW   1                       ; READ/WRITE CHAIN HEADER
002353                          SECMAX: .BLKW   1                       ; MAXIMUM NUMBER OF BUFFERS THAT ACN BE READ
002354                          CHNMAX: .BLKW   1                       ; MAXIMUM NUMBER OF BUFFERS THAT CAN BE WRITTEN
002355                          MAXSEC: .BLKW   1                       ; MAXIMUM NUMBER OF SECTORS THIS PASS
002356                          SECPTR: .BLKW   1                       ; SECTOR POINTER FOR WRITE

002357  000000                  ERRPOS: .WORD   0                       ; RTDS AND STATUS POSITION
                                ;
002360  000000                  OTABLE: .WORD   0                       ; OVERLAY TABLE
002361  000000                          .WORD   0
002362  000000                          .WORD   0
002363  043220                          .WORD   OVL.MS*4
                                ;       .WORD   3                       ; SETUP MODULE OVERLAY INFORMATION
                                ;       .WORD   0
                                ;       .WORD   0
                                ;       .WORD   OVL.ST*4
002364  000000                          .WORD   0                       ; OPERATION MODULE OVERLAY INFORMATION
002365  000000                          .WORD   0
002366  000000                          .WORD   0
                                ;       .WORD   OVL.OP*4
002367  010214                          .WORD   OVL.ST*4
002370  000001                          .WORD   1                       ; SEEK MODULE OVERLAY INFORMATION
002371  000000                          .WORD   0
002372  000000                          .WORD   0
002373  001224                          .WORD   OVL.SK*4
002374  000002                          .WORD   2                       ; SEEK COMPLETE MODULE OVERLAY INFORMATION
002375  000000                          .WORD   0
002376  000000                          .WORD   0
002377  001254                          .WORD   OVL.TS*4
002400  000000                          .WORD   0                       ; READ/WRITE SETUP OVERLAY INFORMATION
002401  000000                          .WORD   0
002402  000000                          .WORD   0
002403  000270                          .WORD   OVL.RW*4
002404  000000                          .WORD   0                       ; WRITE OVERLAY INFORMATION
002405  000003                          .WORD   MAXDST
002406  000000                          .WORD   0
002407  004254                          .WORD   OVL.W*4
002410  000000                          .WORD   0                       ; READ OVERLAY INFORMATION
002411  000003                          .WORD   MAXDST
002412  000000                          .WORD   0
002413  003224                          .WORD   OVL.R*4
002414  000000                          .WORD   0                       ; DATA CHECK MODULE OVERLAY INFORMATION
002415  000003                          .WORD   MAXDST
```

```
 58 00241E  000000                      .WORD   0
 59 002417  005460                      .WORD   OVL.CK*4
 60 002420  000000                      .WORD   0                       ; REVECTOR MODULE OVERLAY INFORMATION
 61 002421  000003                      .WORD   MAXDST
 62 002422  000000                      .WORD   0
 63 002423  003110                      .WORD   OVL.RV*4
 64 002424  000000                      .WORD   0
 65 002425  000003                      .WORD   MAXDST                  ; RECALIBRATION MODULE INFORMATION
 66 002426  000000                      .WORD   0
 67 002427  000260                      .WORD   OVL.RB*4
 68         000012              NUMOVL  =       <. - OTABLE> / 4
 69 002430  177777              PTABLE: .WORD   -1                      ; PAGE TABLE
 70 002431  004515                      .WORD   AREA0
 71 002432  177777                      .WORD   -1
 72 002433  006560                      .WORD   AREA1
 73 002434  177777                      .WORD   -1
 74 002435  007025                      .WORD   AREA2
 75 002436  177777                      .WORD   -1
 76 002437  007300                      .WORD   AREA3
 77         000003              MAXDST  =       <. - PTABLE> / 2 - 1
 78 002440  000000              PNUM:   .WORD   0                       ; PATTERN NUMBER STORAGE AREA FOR ERRORS
 79 002441  000000              LBNHLD: .WORD   0                       ; TEMPORARY STORAGE FOR LBN
 80 002442  000000                      .WORD
 81 002443  000000              ERRCNT: .WORD   0                       ; ERROR COUNT LOCATION
 82 002444  000000              M.PARM: .WORD   0                       ; MASTER PARAMETERS
 83 002445  000000              LOSEED: .WORD   0                       ; LO ORDER RANDOM NUMBER
 84 002446  000000              HISEED: .WORD   0                       ; HI ORDER RANDOM NUMBER
 85 002447                      DUMSDI: .BLKW   18.                     ; DUMMY SDI AREA
 86 002471  002443                      .WORD   DUMSDI-4                ; POINTER FOR LBN REVECTOR INFORMATION
 87                                     ;       BELEIVE IT OR NOT, THIS WILL WRITE THE REVECTORED LBN
 88                                     ;       IN LOCATIONS DUMSDI+8 AND DUMSDI+9
 89                                     ;       EG. DUMSDI+18. POINTS TO 12 SHORT OF WHERE TO PUT THE
 90                                     ;             REVECTOR INFORMATION
 91 002472  000000              DROPSU: .WORD   0                       ; DROP SUBUNIT FLAG
 92
 93 002473  010000              MEMPOL: .WORD   HIMEM                   ; MEMORY POOL FOR UNIT DATASTRUCTURES
 94 002474                      TMEMPL: .BLKW   1                       ; TEMPORARY MEMORY POOL FOR READ/WRITE
 95 002475  002515              PATPTR: .WORD   PAT0                    ; POINTERS TO DATA PATTERNS
 96 002476  002536                      .WORD   PAT1
 97 002477  002540                      .WORD   PAT2
 98 002500  002542                      .WORD   PAT3
 99 002501  002544                      .WORD   PAT4
100 002502  002565                      .WORD   PAT5
101 002503  002606                      .WORD   PAT6
102 002504  002627                      .WORD   PAT7
103 002505  002631                      .WORD   PAT8
104 002506  002652                      .WORD   PAT9
105 002507  002654                      .WORD   PAT10
106 002510  002675                      .WORD   PAT11
107 002511  002677                      .WORD   PAT12
108 002512  002720                      .WORD   PAT13
109 002513  002741                      .WORD   PAT14
110 002514  002745                      .WORD   PAT15
111                                    ;
112                                    ;       THE DATA PATTERNS (LENGTH OF PATTERN IN WORDS, FOLLOWED BY PATTERN)
113                                    ;
114 002515  000001              PAT0:   .WORD   1               ; USER DEFINEABLE PATTERN
```

```
115 002516  000000                        .WORD   0
116 002517  000000                        .WORD   0
117 002520  000000                        .WORD   0
118 002521  000000                        .WORD   0
119 002522  000000                        .WORD   0
120 002523  000000                        .WORD   0
121 002524  000000                        .WORD   0
122 002525  000000                        .WORD   0
123 002526  000000                        .WORD   0
124 002527  000000                        .WORD   0
125 002530  000000                        .WORD   0
126 002531  000000                        .WORD   0
127 002532  000000                        .WORD   0
128 002533  000000                        .WORD   0
129 002534  000000                        .WORD   0
130 002535  000000                        .WORD   0
131 002536  000001              PAT1:     .WORD   1               ; B'1000101110001011'
132 002537  105613                        .WORD   105613
133 002540  000001              PAT2:     .WORD   1
134 002541  031463                        .WORD   031463          ; B'0011001100110011'
135 002542  000001              PAT3:     .WORD   1
136 002543  030221                        .WORD   030221
137 002544  000020              PAT4:     .WORD   16.             ; SHIFTING ONES
138 002545  000001                        .WORD   000001
139 002546  000003                        .WORD   000003
140 002547  000007                        .WORD   000007
141 002550  000017                        .WORD   000017
142 002551  000037                        .WORD   000037
143 002552  000077                        .WORD   000077
144 002553  000177                        .WORD   000177
145 002554  000377                        .WORD   000377
146 002555  000777                        .WORD   000777
147 002556  001777                        .WORD   001777
148 002557  003777                        .WORD   003777
149 002560  007777                        .WORD   007777
150 002561  017777                        .WORD   017777
151 002562  037777                        .WORD   037777
152 002563  077777                        .WORD   077777
153 002564  177777                        .WORD   177777
154 002565  000020              PAT5:     .WORD   16.             ; SHIFTING ZEROS
155 002566  177776                        .WORD   177776
156 002567  177774                        .WORD   177774
157 002570  177770                        .WORD   177770
158 002571  177760                        .WORD   177760
159 002572  177740                        .WORD   177740
160 002573  177700                        .WORD   177700
161 002574  177600                        .WORD   177600
162 002575  177400                        .WORD   177400
163 002576  177000                        .WORD   177000
164 002577  176000                        .WORD   176000
165 002600  174000                        .WORD   174000
166 002601  170000                        .WORD   170000
167 002602  160000                        .WORD   160000
168 002603  140000                        .WORD   140000
169 002604  100000                        .WORD   100000
170 002605  000000                        .WORD   000000
171 002606  000020              PAT6:     .WORD   16.             ; ALTERNATING ZERO WORD AND ONE WORD IN
```

```
172 002607  000000                        .WORD   000000          ; 3-2-1-1-1 SEQUENCE
173 002610  000000                        .WORD   000000
174 002611  000000                        .WORD   000000
175 002612  177777                        .WORD   177777
176 002613  177777                        .WORD   177777
177 002614  177777                        .WORD   177777
178 002615  000000                        .WORD   000000
179 002616  000000                        .WORD   000000
180 002617  177777                        .WORD   177777
181 002620  177777                        .WORD   177777
182 002621  000000                        .WORD   000000
183 002622  177777                        .WORD   177777
184 002623  000000                        .WORD   000000
185 002624  177777                        .WORD   177777
186 002625  000000                        .WORD   000000
187 002626  177777                        .WORD   177777
188 002627  000001              PAT7:     .WORD   1               ; B'1011011011011001'
189 002630  133331                        .WORD   133331
190 002631  000020              PAT8:     .WORD   16.             ; B'0101010101010101' AND
191 002632  052525                        .WORD   052525          ; B'1010101010101010'
192 002633  052525                        .WORD   052525          ; IN 3-2-1-1-1 SEQURNCE
193 002634  052525                        .WORD   052525
194 002635  125252                        .WORD   125252
195 002636  125252                        .WORD   125252
196 002637  125252                        .WORD   125252
197 002640  052525                        .WORD   052525
198 002641  052525                        .WORD   052525
199 002642  125252                        .WORD   125252
200 002643  125252                        .WORD   125252
201 002644  052525                        .WORD   052525
202 002645  125252                        .WORD   125252
203 002646  052525                        .WORD   052525
204 002647  125252                        .WORD   125252
205 002650  052525                        .WORD   052525
206 002651  125252                        .WORD   125252
207 002652  000001              PAT9:     .WORD   1               ; B'1101101101101100'
208 002653  155554                        .WORD   155554
209 002654  000020              PAT10:    .WORD   16.             ; B'0010110100101101' AND
210 002655  026455                        .WORD   026455          ; B'1101001011010010'
211 002656  026455                        .WORD   026455          ; IN 3-2-1-1-1 SEQUENCE
212 002657  026455                        .WORD   026455
213 002660  151322                        .WORD   151322
214 002661  151322                        .WORD   151322
215 002662  151322                        .WORD   151322
216 002663  026455                        .WORD   026455
217 002664  026455                        .WORD   026455
218 002665  151322                        .WORD   151322
219 002666  151322                        .WORD   151322
220 002667  026455                        .WORD   026455
221 002670  151322                        .WORD   151322
222 002671  026455                        .WORD   026455
223 002672  151322                        .WORD   151322
224 002673  026455                        .WORD   026455
225 002674  151322                        .WORD   151322
226 002675  000001              PAT11:    .WORD   1               ; B'0110110110110110'
227 002676  066666                        .WORD   066666
228 002677  000020              PAT12:    .WORD   16.             ; RIPLE ONE
```

```
229 002700  000001                      .WORD   000001
230 002701  000002                      .WORD   000002
231 002702  000004                      .WORD   000004
232 002703  000010                      .WORD   000010
233 002704  000020                      .WORD   000020
234 002705  000040                      .WORD   000040
235 002706  000100                      .WORD   000100
236 002707  000200                      .WORD   000200
237 002710  000400                      .WORD   000400
238 002711  001000                      .WORD   001000
239 002712  002000                      .WORD   002000
240 002713  004000                      .WORD   004000
241 002714  010000                      .WORD   010000
242 002715  020000                      .WORD   020000
243 002716  040000                      .WORD   040000
244 002717  100000                      .WORD   100000
245 002720  000020              PAT13:  .WORD   16.             ; RIPLE ZERO
246 002721  177776                      .WORD   177776
247 002722  177775                      .WORD   177775
248 002723  177773                      .WORD   177773
249 002724  177767                      .WORD   177767
250 002725  177757                      .WORD   177757
251 002726  177737                      .WORD   177737
252 002727  177677                      .WORD   177677
253 002730  177577                      .WORD   177577
254 002731  177377                      .WORD   177377
255 002732  176777                      .WORD   176777
256 002733  175777                      .WORD   175777
257 002734  173777                      .WORD   173777
258 002735  167777                      .WORD   167777
259 002736  157777                      .WORD   157777
260 002737  137777                      .WORD   137777
261 002740  077777                      .WORD   077777
262 002741  000003              PAT14:  .WORD   3               ; B'1101101101101101'
263 002742  155555                      .WORD   155555          ; B'1011011011011011'
264 002743  133333                      .WORD   133333          ; B'1101101101101101'
265 002744  155555                      .WORD   155555          ; REPEATED
266 002745  000020              PAT15:  .WORD   16.             ; B'1011011011011001'
267 002746  133331                      .WORD   133331          ; B'1101101101101100'
268 002747  133331                      .WORD   133331          ; IN 3-2-1-1-1 SEQUENCE
269 002750  133331                      .WORD   133331
270 002751  155554                      .WORD   155554
271 002752  155554                      .WORD   155554
272 002753  155554                      .WORD   155554
273 002754  133331                      .WORD   133331
274 002755  133331                      .WORD   133331
275 002756  155554                      .WORD   155554
276 002757  155554                      .WORD   155554
277 002760  133331                      .WORD   133331
278 002761  155554                      .WORD   155554
279 002762  133331                      .WORD   133331
280 002763  155554                      .WORD   155554
281 002764  133331                      .WORD   133331
282 002765  155554                      .WORD   155554
```

```
                                        .SBTTL  ROOT    - MAIN DRIVING MODULE OF TEST 4
000000                          $DIAG$  =       0       ; SET UP DIAG MODULE
000000                          $$DIAG: .WORD   0       ; SETUP MODULE BREAKPOINT
000000                                  .WORD   0       ; SEEK MODULE
000000                                  .WORD   0       ; SEEK TEST
000000                                  .WORD   0       ; READ/WRITE SETUP
000000                                  .WORD   0       ; WRITE
000000                                  .WORD   0       ; READ
000000                                  .WORD   0       ; CHECK ECC AND DATA
000000                                  .WORD   0       ; REVECTOR
000000                                  .WORD   0       ; RECALIBRATE


104205  007717          ROOT:   MOV     #FIRSTU,R5              ; R5 POINTS TO FIRST UNIT TO BE TESTED
114007                          CLR     R0                      ; COUNT OF DROPPED UNITS IS SET TO ZERO
104655  000000          ROOTLP: MOV     U.NEXT(R5),R5           ; R5 POINTS TO NEXT UNIT IN CHAIN
104651  000033                  MOV     U.PARM(R5),R1           ; GET UNIT PARAMETERS
073020                          BMI     NOUNIT                  ; IF UNIT DROPPED, BRANCH
115000  002444                  TST     M.PARM                  ; SEE IF IN PROCESS OF INITIAL WRITE
033015                          BPL     1$                      ; IF NOT, BRANCH
102201  040000                  BIT     #INITW,R1               ; SEE IF THIS UNIT IS BEING INITALLY WRITTEN
013020                          BEQ     NOUNIT                  ; IF NOT, BRANCH
023062          1$:             CALL    SEQNCR                  ; CALL SEQUENCER
114007                          CLR     R0                      ; COUNT OF DROPPED UNITS IS SET TO ZERO
003061                          BR      ROTEXT                  ; BRANCH TO BOTTOM OF ROOT
115407          NOUNIT: INC     R0                              ; INCREMENT COUNT OF DROPPED UNITS
106207  000004                  CMP     #4,R0                   ; SEE IF FOUR (ALL) UNITS HAVE BEEN DROPPED
053061                          BNE     ROTEXT                  ; IF NOT, BRANCH
115000  002444                  TST     M.PARM                  ; SEE IF INITIAL WRITE WAS IN PROGRESS
033055                          BPL     1$                      ; IF NOT, BRANCH
103200  100000  002444          BIC     #100000,M.PARM          ; CLEAR INITIAL WRITE BIT
                                MSSG    2                       ; REPORT ALL INITIAL WRITES COMPLETE
104200  010611  001010                  MOV     #MS2,OUT.02
100467                                          MOV R0,-(SP)
100461                                          MOV R1,-(SP)
104201  000047                  MOV     #U.SNUM,R1
105051                          ADD     R5,R1
105651  000034                  ADD     U.SUBU(R5),R1
104110  001007                  MOV     (R1),OUT.01
104207  000015                  MOV     #MESSAG,R0
020751                          CALL    HOSTRQ
104261                                          MOV (SP)+,R1
104267                                          MOV (SP)+,R0
114007                          CLR     R0                      ; START UNIT COUNT AT ZERO
003061                          BR      ROTEXT                  ; NOW START TESTING
104207  000016          1$:     MOV     #DONE,R0                ; MOVE DONE OP CODE TO HOST COMMUNICATION AREA
020751                          CALL    HOSTRQ                  ; SENT DONE MESSAGE TO HOST
060021                          XFC     EXIT                    ; EXIT DIAGNOSTIC MACHINE MODE
003002          ROTEXT: BR      ROOTLP                          ; BRANCH TO TOP OF ROOT
```

```
                                .SBTTL  SEQNCR - MODULE SEQUENCER FOR TEST 4
003062                          SEQNCR:
                                ;
                                ;       SEQUENCER CALLS THE VARIOUS MODULES, USING THE ADDRESS RETURNED
                                ;       IN R0 BY THE PREVIOUS MODULE.  R0 HOLDS THE NEXT ADDRESS TO VECTOR
                                ;       TO. IF R1 IS ZERO, AN IMMIDATE CALL IS MADE TO THE NEXT ROUTINE.
                                ;       IF R2 IS NONZERO, AN ERROR WAS ENCOUNTERED, AND THE ERROR COUNT IS
                                ;       INCREMENTED.  THE ERROR IS THEN REPORTED TO THE HOST.
                                ;
003062  104657  000010                MOV     U.NFUN(R5),R0   ; LOAD R0 WITH NEXT FUNCTION ADDRESS
003064  023362          SEQLP:  CALL    DRPTST          ; MAKE SURE AT LEAST ONE ACTIVE SUBUNIT EXISTS
003065  115003                  TST     R3              ; SEE IF ACTIVE SUBUNIT EXISTS
003066  053151                  BNE     NOSUB           ; IF NOT, BRANCH
003067  106207  000100          CMP     #100,R0         ; SEE IF THE ROUTINE IS ON AN OVERLAY
003071  073125                  BMI     GO4IT           ; IF NOT, BRANCH
003072  104071                  MOV     R0,R1           ; SAVE THE OVERLAY NUMBER
003073  105077                  ADD     R0,R0           ; MULTIPLY THE OVERLAY NUMBER BY 4
003074  105077                  ADD     R0,R0
003075  104672  002360          MOV     OTABLE(R0),R2   ; GET THE OVERLAY AREA NUMBER
003077  105022                  ADD     R2,R2           ; MULTIPLY THE AREA NUMBER BY 2
003100  106621  002430          CMP     PTABLE(R2),R1   ; SEE IF THE OVERLAY REQUESTED IS IN MEMORY
003102  013123                  BEQ     LOADED          ; IF SO, BRANCH
003103  023152                  CALL    OVRLAY          ; IF NOT, GET THE OVERLAY
003104  100621  002430          MOV     R1,PTABLE(R2)   ; MARK THE OVERLAY AS IN MEMORY
003106  104677  002361          MOV     OTABLE+1(R0),R0 ; GET THE NUMBER OF OVERLAY AREAS DESTROYED
003110  104201  177777          MOV     #-1,R1          ; R1 CONTAINS THE 'EMPTY AREA' FLAG
003112                          PUSH    R2              ; SAVE R2
003112  100462                                                          MOV R2,-(SP)
003113  117407          GONE:   DEC     R0              ; DECREMENT THE DESTROYED COUNT
003114  073122                  BMI     PLOAD           ; IF ALL NECESSARY AREAS MARKED DESTROYED, BRANCH
003115  105202  000002          ADD     #2,R2           ; POINT TO NEXT OVERLAY AREA
003117  100621  002430          MOV     R1,PTABLE(R2)   ; MARK AREA AS DESTROYED
003121  003113                  BR      GONE            ; GO TO TOP OF LOOP
003122                  PLOAD:  POP     R2              ; RESTORE R2
003122  104262                                                          MOV (SP)+,R2
003123  104627  002431  LOADED: MOV     PTABLE+1(R2),R0 ; GET OVERLAY ADDRESS
003125                  GO4IT:  PUSH    R0              ; PUSH THE BRANCH ADDRESS ONTO THE STACK
003125  100467                                                          MOV R0,-(SP)
003126  000000                  RETURN                  ; VECTOR TO MODULE
003127  114000  002472  JMPRET: CLR     DROPSU          ; FLAG SUBUNIT AS NOT TO BE DROPPED
003131  023724                  CALL    CORECT          ; CORRECT THE ERROR
003132  104303  002472          MOV     DROPSU,R3       ; SEE IF SUBUNIT TO BE DROPPED
003134  013145                  BEQ     NOERR           ; IF NOT, BRANCH
003135  104641  000000          MOV     S.PARM(R4),R1   ; GET SUBUNIT PARAMETERS
003137  101201  100000          BIS     #DROP,R1        ; SET DROP BIT
003141  100641  000000          MOV     R1,S.PARM(R4)   ; SAVE SUBUNIT PARAMETERS
003143  104207  000001          MOV     #SETUP,R0       ; SETUP WILL BE NEXT MODULE CALLED
003145  115001          NOERR:  TST     R1              ; SEE IF IMMIDATE OR DELAYED CALL
003146  013064                  BEQ     SEQLP           ; IF ZERO (IMMIDATE) LOOP
003147  100657  000010          MOV     R0,U.NFUN(R5)   ; SAVE ROUTINE ADDRESS
003151  000000          NOSUB:  RETURN                  ; RETURN TO DRIVER
```

```
                        .SBTTL  OVRLAY - OVERLAY PROCESS FOR BRINGING IN OVERLAYS IF NEEDED
003152                  OVRLAY:
                        ;
                        ;       BRING IN THE OVERLAY
                        ;
003152                          PUSH    <R5,R0,R1,R2,R3>        ; SAVE REGISTERS
003152  100465                                                          MOV R5,-(SP)
003153  100467                                                          MOV R0,-(SP)
003154  100461                                                          MOV R1,-(SP)
003155  100462                                                          MOV R2,-(SP)
003156  100463                                                          MOV R3,-(SP)
003157  104205  000003          MOV     #3,R5           ; TRY TO READ 3 TIMES
003161  104671  002363  1$:     MOV     OTABLE+3(R0),R1 ; GET THE HI ORDER BITS OF UNIBUS ADDRESS
003163  103201  177774          BIC     #177774,R1      ; CLEAR UNUSED BITS
003165  104623  002431          MOV     PTABLE+1(R2),R3 ; POINT TO WHERE TO LOAD OVERLAY
003167  104672  002363          MOV     OTABLE+3(R0),R2 ; GET NUMBER OF WORDS TO TRANSFER
003171  110602                  ROR     R2              ; ROTATE BITS TO CORRECT POSITION
003172  110602                  ROR     R2
003173  103202  140000          BIC     #140000,R2      ; CLEAR UNUSED BITS
003175  104677  002362          MOV     OTABLE+2(R0),R0 ; GET LO ORDER UNIBUS ADDRESS
003177  060013                  XFC     UREAD           ; READ THE OVERLAY INTO UDA MEMORY
003200  101071                  BIS     R0,R1           ; ADD THE ERROR REGISTERS TOGETHER
003201  013246                  BEQ     2$              ; IF NO ERRORS, BRANCH
003202                          POP     <R3,R2,R1,R0>   ; RESTORE THE REGISTERS
003202  104263                                                          MOV (SP)+,R3
003203  104262                                                          MOV (SP)+,R2
003204  104261                                                          MOV (SP)+,R1
003205  104267                                                          MOV (SP)+,R0
003206                          PUSH    <R0,R1,R2,R3>   ; SAVE THE REGISTERS
003206  100467                                                          MOV R0,-(SP)
003207  100461                                                          MOV R1,-(SP)
003210  100462                                                          MOV R2,-(SP)
003211  100463                                                          MOV R3,-(SP)
003212  117405                  DEC     R5              ; DECREMENT COUNT
003213  053161                  BNE     1$              ; IF INCOMPLETE, BRANCH
003214                          DEVFTL  33              ; UNABLE TO READ FROM HOST MEMORY
003214  104200  004546  001012                                  MOV     #ER33,OUT.04
003217  104200  000041  001010                                  MOV     #33,OUT.02
003222  101200  000400  001010                                  BIS     #FTLDEV,OUT.02
003225  104200  003225  001007                                  MOV     #.,OUT.01
003230  104202  000014                                          MOV     #ERRMC,R2
003232  104020  001006                                          MOV     R2,OUT.RQ
003234  104200  177777  001011          MOV     #177777,OUT.03  ; FLAG AS NOT ASSOCIATED WITH ANY UNIT
003237  104207  000013                  MOV     #ERRMES,R0      ; SET UP TO REPORT ERROR
003241  020751                  CALL    HOSTRQ          ; REPORT
003242  104207  000016          MOV     #DONE,R0        ; REPORT THAT PROGRAM IS FINISHED
003244  020751                  CALL    HOSTRQ          ; REPORT
003245  060021                  XFC     EXIT            ; EXIT DM
003246                  2$:     POP     <R3,R2,R1,R0,R5>        ; RESTORE THE REGISTERS
003246  104263                                                          MOV (SP)+,R3
003247  104262                                                          MOV (SP)+,R2
003250  104261                                                          MOV (SP)+,R1
003251  104267                                                          MOV (SP)+,R0
003252  104265                                                          MOV (SP)+,R5
003253  000000                  RETURN                  ; RETURN TO PAGE
```

```
********** NON-OVERLAY MODULE AFINIT - USED AFTER DRIVE IS INI

                                        .SBTTL  ********** NON-OVERLAY MODULE AFINIT - USED AFTER DRIVE IS INITILIZED
003254                                  AFINIT:
                                        ;
                                        ;       AFTER THE DRIVE IS INITIALIZED, WAIT FOR RECEIVER READY (HOPEFULLY)
                                        ;       TO BE ASSERTED.  IF IT IS NOT ASSERTED, DROP THE ENTIRE UNIT
                                        ;
                                        ;       NOTE:
                                        ;               THIS IS A VERY UNSTRUCTURED PIECE OF CODE -- IT HAS SEVERAL
                                        ;               JUMPS DIRECTLY INTO SEQNCR AT SEVERAL DIFFERENT PLACES.
                                        ;
003254  021173                                 CALL    RTDS            ; GET REAL TIME DRIVE STATE
003255  115002                                 TST     R2              ; SEE IF ERROR DURING STATE INTERROGATION
003256  053267                                 BNE     NORTDS          ; IF ERROR, BRANCH
003257  102201  000001                         BIT     #RCVRDY,R1      ; SEE IF RECIEVER READY IS ASSERTED
003261  013267                                 BEQ     NORTDS          ; IF SO, BRANCH
003262  104207  000001                         MOV     #SETUP,R0       ; SETUP IS NEXT MODULE CALLED
003264  114001                                 CLR     R1              ; IMMIDATE CALL TO SETUP
003265  114002                                 CLR     R2              ; NO ERRORS
003266  003361                                 BR      ATNEXT          ; BRANCH TO EXIT
003267  104207  003254        NORTDS:          MOV     #AFINIT,R0      ; AFINIT IS NEXT ROUTINE CALLED
003271  104013                                 MOV     R1,R3           ; SAVE REAL TIME DRIVE STATE
003272  104651  000005                         MOV     U.TIMO(R5),R1   ; GET TIMEOUT VALUE
003274  117401                                 DEC     R1              ; DECREMENT TIMEOUT VALUE
003275  100651  000005                         MOV     R1,U.TIMO(R5)   ; SAVE TIMEOUT VALUE
003277  053361                                 BNE     ATNEXT          ; BRANCH TO SEQUENCER LOOP
003300  115002                                 TST     R2              ; SEE IF NO REAL TIME STATE
003301  053327                                 BNE     NRTDS           ; IF SO, BRANCH
003302  103203  077674                         BIC     #07674,R3       ; CLEAR UNUSED BITS
003304                                         DEVFTL  28,R3           ; RECIEVER READY WAS NEVER ASSERTED
003304  104200  003746  001012                                                 MOV     #ER28,OUT.04
003307  104030  001013                                                         MOV     R3,OUT.05
003311  104200  000034  001010                                                 MOV     #28,OUT.02
003314  101200  000400  001010                                                 BIS     #FTLDEV,OUT.02
003317  104200  003317  001007                                                 MOV     #.,OUT.01
003322  104202  000014                                                         MOV     #ERRMC,R2
003324  104020  001006                                                         MOV     R2,OUT.RQ
003326  003347                                 BR      IERR            ; BRANCH
003327                                 NRTDS:  DEVFTL  29              ; REPORT UNABLE TO GET REAL TIME DRIVE STATUS
003327  104200  004012  001012                                                 MOV     #ER29,OUT.04
003332  104200  000035  001010                                                 MOV     #29,OUT.02
003335  101200  000400  001010                                                 BIS     #FTLDEV,OUT.02
003340  104200  003340  001007                                                 MOV     #.,OUT.01
003343  104202  000014                                                         MOV     #ERRMC,R2
003345  104020  001006                                                         MOV     R2,OUT.RQ
003347                                 IERR:   ENDERR  0               ; DON'T REPORT STATUS OR STATE
003347  114000  002357                                                         CLR     ERRPOS          ; CLEAR THE POSITION
003351  104651  000033                         MOV     U.PARM(R5),R1   ; GET DRIVE PARAMETERS
003353  101201  100000                         BIS     #DROP,R1        ; FLAG DRIVE AS DROPPED
003355  100651  000033                         MOV     R1,U.PARM(R5)   ; SAVE DRIVE STATE
003357  024030                                 CALL    REPERR          ; REPORT ERROR
003360  114007                                 CLR     R0              ; R0 IS NOT 'AFINIT'S ADDRESS FOR 'TROOT'
003361  003151                         ATNEXT: BR      NOSUB           ; BRANCH TO SEQUENCER LOOP
```

DRPTST - SEE THAT AT LEAST ONE SUBUNIT IS ACTIVE ON A UNIT

```
                                .SBTTL  DRPTST - SEE THAT AT LEAST ONE SUBUNIT IS ACTIVE ON A UNIT
003362                          DRPTST:
                                ;
                                ;       DROP TEST MAKES SURE AT LEAST ONE ACTIVE SUBUNIT IS ACTIVE ON THE UNIT
                                ;       IF SO, R4 POINTS TO SUBUNIT PARAMETER BLOCK, U.SUBU IS UPDATED TO
                                ;       THE SUBUNIT OFFSET, AND R3 IS RETURNED POSITIVE.  IF NOT, R3 IS
                                ;       RETURNED NEGATIVE AND THE ENTIRE UNIT IS DROPPED
                                ;
003362                                  PUSH    R0              ; SAVE NEXT MODULE ADDRESS
003362  100467                                                          MOV R0,-(SP)
003363  104657  000033                 MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
003365  102207  001000                 BIT     #FTIME,R0       ; SEE IF FIRST TIME THROUGH
003367  013442                         BEQ     1$              ; IF NOT, BRANCH
003370  101207  024000                 BIS     #NEWSUB!RESEEK,R0       ; IF FIRST TIME, FORCE A SEEK, NEW SUBUNIT
003372  103207  011010                 BIC     #FTIME!DIREC.WCHECK,R0  ; NO LONGER FIRST TIME, SET DIRECTION UP
003374  100657  000033                 MOV     R0,U.PARM(R5)   ; SAVE
003376  104207  000001                 MOV     #U.SUBP,R0      ; R0 WILL POINT AT SUBUNIT POINTERS
003400  105057                         ADD     R5,R0           ; R0 POINTS AT SUBUNIT POINTERS
003401  114002                         CLR     R2              ; UP TO 4 SUBUNITS ON THIS UNIT
003402  100652  000021                 MOV     R2,U.CSEC(R5)   ; ZERO ALL TRACK COUNTS SO THE TEST AT THE
003404  100652  000016                 MOV     R2,U.NSEC(R5)   ; BEGINNING OF SETUP IS SATISIFIED SO CONTROL
003406  100652  000020                 MOV     R2,U.TSEC(R5)   ; DROPS THROUGH TO SETUP THE NEXT OPERATION
003410  104273                 1$:     MOV     (R0)+,R3        ; GET SUBUNIT POINTER
003411  073423                         BMI     12$             ; IF NO SUBUNIT, BRANCH
003412  104633  000000                 MOV     S.PARM(R3),R3   ; GET UNIT PARAMETERS
003414  073423                         BMI     12$             ; IF DROPPED, BRANCH
003415  115000  002444                 TST     M.PARM          ; SEE IF INTIIAL WRITE IN PROGRES
003417  033430                         BPL     13$             ; IF NOT, BRANCH
003420  102203  040000                 BIT     #INITW,R3       ; SEE IF THIS SUBUNIT IS TO BE INITIALLY WRITTEN
003422  053430                         BNE     13$             ; IF SO, BRANCH
003423  115402                 12$:    INC     R2              ; INCREMENT COUNT
003424  106202  000003                 CMP     #3,R2           ; SEE IF COUNT EXHAUSTED
003426  033410                         BPL     11$             ; IF NOT, BRANCH
003427  003553                         BR      4$              ; IF SO, DROP ENTIRE UNIT
003430  100652  000034         13$:    MOV     R2,U.SUBU(R5)   ; SAVE SUBUNIT OFFSET
003432  104024                         MOV     R2,R4           ; MOVE TO SUBUNIT POINTER REGISTER
003433  105054                         ADD     R5,R4           ; ADD UNIT POINTER
003434  105204  000001                 ADD     #U.SUBP,R4      ; ADD SUBUNIT POINTER OFFSET
003436  104144                         MOV     (R4),R4         ; R4 NOW POINTS TO SUBUNIT
003437  024161                         CALL    RECOVR          ; SET WRITE PROTECT IF REQUESTED
003440  114003                         CLR     R3              ; PROCESS THIS UNIT
003441  003561                         BR      5$              ; EXIT
003442  104204  000001         1$:     MOV     #U.SUBP,R4      ; R4 WILL POINT AT SUBUNIT POINTERS
003444  105054                         ADD     R5,R4           ; R4 POINTS TO SUBUNIT POINTERS
003445  105654  000034                 ADD     U.SUBU(R5),R4   ; R4 POINTS AT SPECIFIC SUBUNIT POINTER
003447  104144                         MOV     (R4),R4         ; R4 POINTS AT SUBUNIT PARAMETERS
003450  104641  000000                 MOV     S.PARM(R4),R1   ; GET UNIT PARAMETERS
003452  102207  004000                 BIT     #NEWSUB,R0      ; SEE IF SUPPOSED TO GO TO NEXT SUBUNIT
003454  013520                         BEQ     6$              ; IF NOT, BRANCH
003455  114002                         CLR     R2              ; FOR CEARING FOLLOWING WORDS
003456  100652  000021                 MOV     R2,U.CSEC(R5)   ; ZERO ALL TRACK COUNTS SO THE TEST AT THE
003460  100652  000016                 MOV     R2,U.NSEC(R5)   ; BEGINNING OF SETUP IS SATISIFIED SO CONTROL
003462  100652  000020                 MOV     R2,U.TSEC(R5)   ; DROPS THROUGH TO SETUP THE NEXT OPERATION
003464  115000  002444                 TST     M.PARM          ; SEE IF INITIAL WRITE IS IN PROGRESS
003466  033524                         BPL     2$              ; IF NOT, BRANCH
003467  103201  040000                 BIC     #INITW,R1       ; CLEAR INITIAL WRITE BIT
003471  100641  000000                 MOV     R1,S.PARM(R4)   ; SAVE
```

DRPTST - SEE THAT AT LEAST ONE SUBUNIT IS ACTIVE ON A UNIT

```
     57 003473                                          MSSG    1               ; REPORT INITIAL WRITE COMPLETE
        003473  104200  010573  001010                                                  MOV     #MS1,OUT.02
        003476  100467                                                                          MOV R0,-(SP)
        003477  100461                                                                          MOV R1,-(SP)
        003500  104201  000047                                                          MOV     #U.SNUM,R1
        003502  105051                                                                  ADD     R5,R1
        003503  105651  000034                                                          ADD     U.SUBU(R5),R1
        003505  104110  001007                                                          MOV     (R1),OUT.01
        003507  104207  000015                                                          MOV     #MESSAG,R0
        003511  020751                                                                  CALL    HOSTRQ
        003512  104261                                                                          MOV (SP)+,R1
        003513  104267                                                                          MOV (SP)+,R0
     58 003514  104652  000034                  MOV     U.SUBU(R5),R2   ; GET ACTIVE SUBUNIT POINTER
     59 003516  023614                          CALL    CHKUP           ; SEE IF ANY MORE UNITS TO INITIAL WRITE
     60 003517  003561                          BR      5$              ; BRANCH
     61 003520  114003              6$:         CLR     R3              ; IF ACTIVE, FLAG TO PROCESS
     62 003521  102201  100000                  BIT     #DROP,R1        ; SEE IF UNIT ACTIVE
     63 003523  013561                          BEQ     5$              ; IF ACTIVE, BRANCH
     64 003524  102201  040100              2$: BIT     #SEQSEK!INITW,R1        ; SEE IF SEQUENTIAL SEEKS
     65 003526  053531                          BNE     10$             ; IF SO, BRANCH
     66 003527  023664                          CALL    RNDSUB          ; GET RANDOM SUBUNIT
     67 003530  003561                          BR      5$              ; EXIT
     68 003531  104652  000034              10$: MOV     U.SUBU(R5),R2   ; R2 IS SUBUNIT NUMBER (0-3)
     69 003533  102207  010000                  BIT     #DIREC,R0       ; SEE IF GOING UP OR DOWN
     70 003535  053545                          BNE     3$              ; IF DOWN, BRANCH
     71 003536  023614                          CALL    CHKUP           ; FIND NEXT UPPER SUBUNIT
     72 003537  115007                          TST     R0              ; FIND ONE?
     73 003540  013561                          BEQ     5$              ; IF SO, BRANCH
     74 003541  023563                          CALL    CHKDN           ; SEARCH DOWN FOR ONE
     75 003542  115007                          TST     R0              ; FIND ONE?
     76 003543  013561                          BEQ     5$              ; IF SO, BRANCH
     77 003544  003553                          BR      4$              ; DROP UNIT
     78 003545  023563              3$:         CALL    CHKDN           ; LOOK DOWN FOR SUBUNIT
     79 003546  115007                          TST     R0              ; FIND ONE?
     80 003547  013561                          BEQ     5$              ; IF SO, BRANCH
     81 003550  023614                          CALL    CHKUP           ; LOOK UP FOR ONE
     82 003551  115007                          TST     R0              ; FIND ONE?
     83 003552  013561                          BEQ     5$              ; IF SO, BRANCH
     84 003553  104653  000033              4$: MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
     85 003555  101203  100000                  BIS     #DROP,R3        ; DROP UNIT
     86 003557  100653  000033                  MOV     R3,U.PARM(R5)   ; SAVE
     87 003561                              5$: POP     R0              ; RESTORE R0
        003561  104267                                                                  MOV (SP)+,R0
     88 003562  000000                          RETURN                  ; RETURN
```

DRPTST - SEE THAT AT LEAST ONE SUBUNIT IS ACTIVE ON A UNIT

```
                                CHKDN:
                                ;
                                ;       SEARCH THE SUBUNIT LIST DOWN AND TRY TO FIND AN ACTIVE SUBUNIT
                                ;
003563  104203  000001                MOV     #U.SUBP,R3      ; R3 WILLPOINT TO SUBUNIT POINTERS
003565  105053                        ADD     R5,R3           ; R3 POINTS TO SUBUNIT POINTERS
003566  117402                1$:     DEC     R2              ; DECREMENT OLD SUBUNIT
003567  073604                        BMI     2$              ; IF ALL SUBUNITS CHECKED, BRANCH
003570  104034                        MOV     R3,R4           ; MOVE SUBUNIT LIST POINTER TO R4
003571  105024                        ADD     R2,R4           ; POINT TO SUBUNIT
003572  104144                        MOV     (R4),R4         ; R4 POINTS TO SUBUNIT
003573  073566                        BMI     1$              ; IF NO UNIT, BRANCH
003574  104647  000000                MOV     S.PARM(R4),R0   ; R0 HAS SUBUNIT PARAMETERS
003576  073566                        BMI     1$              ; IF THIS SUBUNIT DROPPED, BRANCH
003577  100652  000034                MOV     R2,U.SUBU(R5)   ; SAVE SUBUNIT NUMBER
003601  114007                        CLR     R0              ; FOUND A SUBUNIT
003602  114003                        CLR     R3              ; PROCESS THIS SUBUNIT
003603  003613                        BR      3$              ; EXIT
003604  104657  000033        2$:     MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
003606  103207  010000                BIC     #DIREC,R0       ; SET DIRECTION UP
003610  100657  000033                MOV     R0,U.PARM(R5)   ; SAVE
003612  104057                        MOV     R5,R0           ; DIDN'T FIND A SUBUNIT
003613  000000        3$:     RETURN                          ; RETURN TO CALLING PROGRAM
```

```
                                CHKUP:
                                ;
                                ;       SEARCH THE SUBUNIT LIST UP AND TRY TO FIND AN ACTIVE SUBUNIT
                                ;
003614  104203  000001                MOV     #U.SUBP,R3      ; R3 WILLPOINT TO SUBUNIT POINTERS
003616  105053                        ADD     R5,R3           ; R3 POINTS TO SUBUNIT POINTERS
003617  115402                1$:     INC     R2              ; INCREMENT OLD SUBUNIT
003620  106202  000003                CMP     #3,R2           ; SEE IF ALL SUBUNITS CHECKED
003622  073645                        BMI     3$              ; IF ALL SUBUNITS CHECKED, BRANCH
003623  104034                        MOV     R3,R4           ; MOVE SUBUNIT LIST POINTER TO R4
003624  105024                        ADD     R2,P4           ; POINT TO SUBUNIT
003625  104144                        MOV     (R4),R4         ; R4 POINTS TO SUBUNIT
003626  073617                        BMI     1$              ; IF NO UNIT, BRANCH
003627  104647  000000                MOV     S.PARM(R4),R0   ; R0 HAS SUBUNIT PARAMETERS
003631  073617                        BMI     1$              ; IF THIS SUBUNIT DROPPED, BRANC
003632  115000  002444                TST     M.PARM          ; SEE IF INITIAL WRITE IN PROGRESS
003634  033640                        BPL     2$              ; IF NOT, BRANCH
003635  102207  040000                BIT     #INITW,R0       ; SEE IF INITIAL WRITE TO BE DONE ON SUBUNIT
003637  013617                        BEQ     1$              ; IF NOT, BRANCH
003640  100652  000034        2$:     MOV     R2,U.SUBU(R5)   ; SAVE SUBUNIT NUMBER
003642  114007                        CLR     R0              ; FOUND ONE
003643  114003                        CLR     R3              ; PROCESS THIS ONE
003644  003663                        BR      5$              ; EXIT
003645  104651  000033        3$:     MOV     U.PARM(R5),R1   ; GET UNIT PARAMETERS
003647  115000  002444                TST     M.PARM          ; SEE IF DOING AN INITIAL WRITE
003651  033657                        BPL     4$              ; IF NOT, BRANCH
003652  103201  040000                BIC     #INITW,R1       ; FLAG INITIAL WRITE AS COMPLETE ON THIS UNIT
003654  101201  001000                BIS     #FTIME,R1       ; START TESTING UNIT
003656  003661                        BR      9$              ; EXIT
003657  101201  010000        4$:     BIS     #DIREC,R1       ; SET DIRECTON DOWN
003661  100651  000033        9$:     MOV     R1,U.PARM(R5)   ; SAVE
003663  000000                5$:     RETURN                  ; RETURN TO CALLING PROGRAM
```

DRPTST - SEE THAT AT LEAST ONE SUBUNIT IS ACTIVE ON A UNIT

```
      1 003664                          RNDSUB:
      2                                 ;
      3                                 ;       GET NEXT SUBUNIT TO CHECK RANDOMLY
      4                                 ;
      5 003664  104207  000004                   MOV     #4,R0           ; SUBUNIT COUNT
      6 003666  104203  000001                   MOV     #U.SUBP,R3      ; R3 WILL POINT TO SUBUNIT POINTERS
      7 003670  105053                           ADD     R5,R3           ; R3 POINTS TO SUBUNIT POINTERS
      8 003671  104652  000034                   MOV     U.SUBU(R5),R2   ; R2 IS SUBUNIT NUMBER
      9 003673  115402                  1$:      INC     R2              ; GO TO NEXT UNIT
     10 003674  106202  000003                   CMP     #3,R2           ; SEE IF GREATER THAN 3
     11 003676  033700                           BPL     2$              ; IF NOT, BRANCH
     12 003677  114002                           CLR     R2              ; NOW ON SUBUNIT 0
     13 003700  104034                  2$:      MOV     R3,R4           ; R4 POINTS TO SUBUNIT POINTERS
     14 003701  105024                           ADD     R2,R4           ; R4 NOW POINTS TO A SPECIFIC SUBUNIT POINTER
     15 003702  104144                           MOV     (R4),R4         ; R4 NOW POINTS TO SUBUNIT PARAMETERS
     16 003703  073707                           BMI     3$              ; IF NO SUBUNIT, BRANCH
     17 003704  104641  000000                   MOV     S.PARM(R4),R1   ; GET SUBUNIT PARAMETERS
     18 003706  033720                           BPL     4$              ; IF ACTIVE, BRANCH
     19 003707  117407                  3$:      DEC     R0              ; DECREMENT COUNT
     20 003710  053673                           BNE     1$              ; IF INCOMPLETE, BRANCH
     21 003711  104653  000033                   MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
     22 003713  101203  100000                   BIS     #DROP,R3        ; SET DROP BIT
     23 003715  100653  000033                   MOV     R3,U.PARM(R5)   ; SAVE
     24 003717  003723                           BR      5$              ; EXIT
     25 003720  100652  000034          4$:      MOV     R2,U.SUBU(R5)   ; SAVE SUBUNIT NUMBER
     26 003722  114003                           CLR     R3              ; TEST THIS SUBUNIT IMMIDATELY
     27 003723  000000                  5$:      RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                        .SBTTL  CORECT - CORRECT THE ERRORS
003724                                  CORECT:
                                        ;
                                        ;       CORRECT THE ERRORS AND RECOVER
                                        ;
003724                                          PUSH    <R0,R1>         ; SAVE R0 AND R1
003724  100467                                                                  MOV R0,-(SP)
003725  100461                                                                  MOV R1,-(SP)
003726  104200  000005  002443                 MOV     #5,ERRCNT       ; ALLOW ONLY 5 TRIES AT ERROR RECOVERY
003731  115002                                  TST     R2              ; SEE IF REPORTING AN ERROR
003732  053771                                  BNE     REPORT          ; IF SO, BRANCH
003733  021173                                  CALL    RTDS            ; GET REAL TIME DRIVE STATE
003734  115002                                  TST     R2              ; SEE IF AN ERROR OCCURRED
003735  053747                                  BNE     NSTATE          ; IF SO, BRANCH
003736  102201  000002                         BIT     #ATTN,R1        ; SEE IF ATTENTION IS ASSERTED
003740  014025                                  BEQ     COREXT          ; IF NOT, BRANCH
003741  104653  000033                         MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
003743  102203  002000                         BIT     #SEEKIP,R3      ; SEE IF SEEK IN PROGRESS
003745  054025                                  BNE     COREXT          ; IF SO, LET THE SEEK ROUTINE REPORT ATTENTION
003746  004021                                  BR      RECOVE          ; OTHERWISE, BRANCH
003747                                  NSTATE: HARDER  17              ; REPORT ERROR
003747  104200  002442  001012                                                  MOV     #ER17,OUT.04
003752  104200  000021  001010                                                  MOV     #17,OUT.02
003755  101200  001000  001010                                                  BIS     #ERHARD,OUT.02
003760  104200  003760  001007                                                  MOV     #.,OUT.01
003763  104202  000014                                                          MOV     #ERRMC,R2
003765  104020  000006                                                          MOV     R2,OUT.RQ
003767                                          ENDERR  0               ; DON'T TRY TO REPORT REAL TIME DRIVE STATE
003767  114000  002357                                                          CLR     ERRPOS          ; CLEAR THE POSITION
003771  024030                          REPORT: CALL    REPERR          ; REPORT THE ERROR TO THE HOST
003772  114000  002357                         CLR     ERRPOS          ; CLEAR ERROR POSITION FLAG
003774  104303  002472                         MOV     DROPSU,R3       ; SEE IF MEGABIT REPORT OR UNIT IS TO BE DROPPED
003776  074025                                  BMI     COREXT          ; UNIT DROPPED - SKIP ERROR RECOVERY
003777  117400  002443                         DEC     ERRCNT          ; DECREMENT THE ERROR COUNT
004001  054021                                  BNE     RECOVE          ; BRANCH IF ERROR COUNT NON-ZERO
004002  104652  000022                         MOV     U.MASK(R5),R2   ; GET UNIT MASK FOR INIT
004004  060011                                  XFC     DINIT           ; INITILIZE THE DRIVE
004005  104202  000020                         MOV     #20,R2          ; TIMEOUT VALUE FOR INIT
004007  100652  000005                         MOV     R2,U.TIMO(R5)   ; SAVE TIMEOUT
004011                                          POP     <R1,R0>         ; DISCARD THE TOP 2 VALUES ON THE STACK
004011  104261                                                                  MOV (SP)+,R1
004012  104267                                                                  MOV (SP)+,R0
004013  104021                                  MOV     R2,R1           ; MAKE R1 NON-ZERO (DEFERRED CALL)
004014  104207  003254                         MOV     #AFINIT,R0      ; NEXT MODULE CALLED IS AFINIT
004016  114000  002472                         CLR     DROPSU          ; SUBUNIT IS NOT TO BE DROPPED
004020  004027                                  BR      HNDEX           ; BRANCH TO EXIT
004021  024161                          RECOVE: CALL    RECOVR          ; RECOVER FROM ERRORS
004022  115003                                  TST     R3              ; SEE IF ERROR RECOVERY COMPLETED CORRECTLY
004023  014025                                  BEQ     COREXT          ; IF NO ERRORS, BRANCH
004024  033771                                  BPL     REPORT          ; IF NOT, BRANCH
004025                                  COREXT: POP     <R1,R0>         ; RESTORE R1, R0
004025  104261                                                                  MOV (SP)+,R1
004026  104267                                                                  MOV (SP)+,R0
004027  000000                          HNDEX:  RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                        .SBTTL  REPERR - REPORT THE ERROR TO THE HOST
004030                                  REPERR:
                                        ;
                                        ;       REPORT THE ERROR OR MEGABIT TRANSFER REPORT
                                        ;
004030  104201  000047                  MOV     #U'.SNUM,R1     ; POINT TO SUBUNIT NUMBERS
004032  105651  000034                  ADD     U.SUBU(R5),R1   ; ADD OFFSET TO SUBUNIT POINTERS
004034  105051                          ADD     R5,R1           ; POINT TO ACTUAL SUBUNIT IN USE
004035  106200  000011  001043          CMP     #T4MXFR,OUT.29  ; SEE IF MEGABIT TRANSFER REPORT
004040  014150                          BEQ     MBXFER          ; IF SO, BRANCH
004041  104110  001011                  MOV     (R1),OUT.03     ; SAVE IN OUTPUT AREA
004043  104307  002357                  MOV     ERRPOS,R0       ; GET ERROR POSITION
004045  014145                          BEQ     7$              ; IF NO REPORTING, BRANCH
004046  103207  100000                  BIC     #100000,R0      ; CLEAR 'ALLREADY VALID' BIT
004050  105207  001006                  ADD     #OUT.RQ,R0      ; POINT TO OUTPUT BUFFER
004052  021173                          CALL    RTDS            ; GET REAL TIME DRIVE STATE
004053  115002                          TST     R2              ; SEE IF ERROR OCCURRED
004054  054060                          BNE     1$              ; IF SO, BRANCH
004055  103201  077674                  BIC     #077674,R1      ; CLEAR UNUSED BITS
004057  004062                          BR      2$              ; BRANCH
004060  104201  177777          1$:     MOV     #-1,R1          ; FLAG AS ERROR
004062  100271          2$:     MOV     R1,(R0)+        ; MOVE TO OUTPUT BUFFER
004063  115000  002357                  TST     ERRPOS          ; SEE IF STATUS VALID
004065  074135                          BMI     4$              ; IF SO, BRANCH
004066                                  PUSH    <R0,R4>         ; SAVE R0
004066  100467                                                                  MOV R0,-(SP)
004067  100464                                                                  MOV R4,-(SP)
004070  104203  001501                  MOV     #CR.GST,R3      ; POINT TO GET STATUS COMMAND
004072  104652  000022                  MOV     U.MASK(R5),R2   ; GET UNIT SDI SELECT MASK
004074  104237                          MOV     (R3)+,R0        ; POINTS TO SDI COMMAND BUFFER
004075  104231                          MOV     (R3)+,R1        ; LOAD BYTE COUNT
004076   60004                          XFC     SEND            ; SEND SDI COMMAND
004077   15001                          TST     R1              ; SEE IF SDI COMMAND SENT SUCESSFULLY
004100  054121                          BNE     8$              ; IF NOT, BRANCH
004101  106203  001532                  CMP     #LONG,R3        ; SEE IF LONG TIMEOUT TO BE USED
004103  074107                          BMI     21$             ; IF SO, BRANCH
004104  104654  000030                  MOV     U.SDIS(R5),R4   ; R4 HAS SHORT TIMEOUT
004106  004111                          BR      22$             ; BRANCH
004107  104654  000031          21$:    MOV     U.SDIL(R5),R4   ; R4 HAS LONG TIMEOUT
004111  104137          22$:    MOV     (R3),R0         ; POINT TO RECEIVE BUFFER
004112  104631  000001                  MOV     1(R3),R1        ; NUMBER OF WORDS IN RESPONSE
004114  060005                          XFC     RCV             ; RECEIVE SDI RESPONSE
004115  115001                          TST     R1              ; SEE IF SDI RESPONSE RECEIVED SUCESSFULLY
004116  014133                          BEQ     3$              ; IF SO, BRANCH
004117  177404                          DEC     R4              ; DECREMENT TIMEOUT VALUE
004120  054111                          BNE     22$             ; IF TIMEOUT UNEXPIRED, BRANCH
004121                          8$:     POP     <R4,R0>         ; RESTORE R4
004121  104264                                                                  MOV (SP)+,R4
004122  104267                                                                  MOV (SP)+,R0
004123  104203  177777                  MOV     #-1,R3          ; R3 CONTAINS 'UNABLE TO GET STATUS'
004125  104202  000007                  MOV     #7,R2           ; R2 CONTAINS COUNT
004127  100273          6$:     MOV     R3,(R0)+        ; MOVE TO OUTPUT BUFFER
004130  117402                          DEC     R2              ; DECREMENT COUNT
004131  054127                          BNE     6$              ; IF COUNT INCOMPLETE, BRANCH
004132  004145                          BR      7$              ; BRANCH
004133                          3$:     POP     <R4,R0>         ; RESTORE R4
004133  104264                                                                  MOV (SP)+,R4
```

REPERR - REPORT THE ERROR TO THE HOST

```
                004134  104267                                                          MOV (SP)+,R0
     53 004135  104201  001577          4$:     MOV     #ST+7,R1        ; R1 POINTS TO AFTER STATUS BUFFER
     54 004137  104202  000007                  MOV     #7,R2           ; R2 HAS COUNT
     55 004141  104413                  5$:     MOV     -(R1),R3        ; MOVE STATUS WORD TO R3
     56 004142  100273                          MOV     R3,(R0)+        ; MOVE TO OUTPUT BUFFER
     57 004143  117402                          DEC     R2              ; DECREMENT COUNT
     58 004144  054141                          BNE     5$              ; BRANCH IF COUNT INCOMPLETE
     59 004145  104307  001006          7$:     MOV     OUT.RQ,R0       ; MOVE REQUEST TO R0
     60 004147  004154                          BR      HRQ             ; BRANCH
     61 004150  104207  000011          MBXFER: MOV     #T4MXFR,R0      ; MOVE MEGABIT REQUEST NUMBER TO R0
     62 004152  104110  001007          BHRQ:   MOV     (R1),OUT.01     ; MOVE SUBUNIT NUMBER TO OUTPUT BUFFER
     63 004154  020751                  HRQ:    CALL    HOSTRQ          ; INITIATE HOST INTERCHANGE
     64 004155  104300  001045  002472          MOV     IN.01,DROPSU    ; SEE IF SUBUNIT TO BE DROPPED
     65 004160  000000                          RETURN
```

```
                                .SBTTL  RECOVR - RECOVER FROM THE ERROR
004161                          RECOVR:
                                ;
                                ;       SET UP DRIVE AS IT SHOULD BE
                                ;
004161                          PUSH    R0              ; SAVE R0
004161  100467                                                  MOV R0,-(SP)
004162  021173                  CALL    RTDS            ; GET REAL TIME DRIVE STATE
004163  115002                  TST     R2              ; SEE IF ERROR OCCURRED
004164  054265                  BNE     BONL            ; IF SO, CONTINUE (WHAT CAN I DO?)
004165  102201  000001          BIT     #RCVRDY,R1      ; SEE IF RECEIVER READY ASSERTED
004167  054265                  BNE     BONL            ; IF SO, BRANCH
                                ;       SEE IF OUTSTANDING RECEIVE
004170  104652  000022          MOV     U.MASK(R5),R2   ; GET UNIT MASK
004172  104207  001570          MOV     #ST,R0          ; POINT TO RECEIVE BUFFER
004174  104201  000007          MOV     #7,R1           ; CLEAR 7 WORDS
004176  114003                  CLR     R3              ; SET UP FOR CLEAR
004177  100273                1$:MOV    R3,(R0)+        ; CLEAR WORD
004200  117401                  DEC     R1              ; DECREMENT COUNT
004201  054177                  BNE     1$              ; IF INCOMPLETE, BRANCH
004202  104207  001570          MOV     #ST,R0          ; POINT TO RECEIVE BUFFER
004204  104201  000023          MOV     #19.,R1         ; RECEIVE A MAXIMUM OF 19 WORDS (MAX RESPONSE)
004206  060005                  XFC     RCV             ; SEE IF OUTSTANDING RESPONSE
004207  115001                  TST     R1              ; SEE IF ERROR OCCURRED
004210  054265                  BNE     BONL            ; IF SO, IT'S OK, BRANCH
004211                          HARDER  1,<R0,ST,ST+1,ST+2,ST+3,ST+4,ST+5,ST+6>
004211  104200  000062  001012                                  MOV     #ER1,OUT.04
004214  104070  001013                                          MOV     R0,OUT.05
004216  104300  001570  001014                                  MOV     ST,OUT.06
004221  104300  001571  001015                                  MOV     ST+1,OUT.07
004224  104300  001572  001016                                  MOV     ST+2,OUT.08
004227  104300  001573  001017                                  MOV     ST+3,OUT.09
004232  104300  001574  001020                                  MOV     ST+4,OUT.10
004235  104300  001575  001021                                  MOV     ST+5,OUT.11
004240  104300  001576  001022                                  MOV     ST+6,OUT.12
004243  104200  000001  001010                                  MOV     #1,OUT.02
004246  101200  001000  001010                                  BIS     #ERHARD,OUT.02
004251  104200  004251  001007                                  MOV     #.,OUT.01
004254  104202  000014                                          MOV     #ERRMC,R2
004256  104020  001006                                          MOV     R2,OUT.RQ
004260                          ENDERR
004260  101200  000015  002357                                  BIS     #15,ERRPOS      ; SET THE POSITION
004263  104053                  MOV     R5,R3           ; MAKE SURE ERROR IS FLAGGED
004264  004365                  BR      MODOK0          ; EXIT
004265  104203  001513  BONL:   MOV     #CR.ONL,R3      ; POINT TO ONLINE COMMAND
004267  021204                  CALL    TALK            ; INITIATE SDI INTERCHANGE
004270  115002                  TST     R2              ; SEE IF ERROR OCCURRED
004271  014276                  BEQ     ONLOK           ; IF NOT, BRANCH
004272                          CERROR  5,#SER2         ; REPORT SECONDARY ERROR
004272  104200  010124  001013                                  MOV     #SER2,OUT.05
004275  004365                  BR      MODOK0          ; BRANCH TO ERROR LOOP
004276  104203  001501  ONLOK:  MOV     #CR.GST,R3      ; POINT TO GET STATUS
004300  021204                  CALL    TALK            ; SEND SDI INTERCHANGE
004301  115002                  TST     R2              ; SEE IF AN ERROR OCCURRED
004302  014307                  BEQ     GSTOK           ; BRANCH IF NO ERROR
004303                          CERROR  5,#SER0         ; REPORT SECONDARY ERROR
004303  104200  010054  001013                                  MOV     #SER0,OUT.05
```

```
 40 004306  004365                          BR      MODOK0          ; LOOP AND TRY AGAIN
 41 004307  024370                GSTOK:     CALL    CHKSTA          ; UPDATE ALL STATUS BITS
 42 004310  115002                          TST     R2              ; SEE IF ERROR OCCURRED
 43 004311  054365                          BNE     MODOK0          ; IF SO, BRANCH
 44 004312  104301  001572                  MOV     ST+ST.ERR,R1    ; GET ERROR BITS
 45 004314  103201  177400                  BIC     #HIBYTE,R1      ; CLEAR UNUSED BITS
 46 004316  104010  001551                  MOV     R1,DCLR         ; MOVE ERROR BITS TO CLEAR DRIVE COMMAND
 47 004320  104203  001455                  MOV     #CR.CLR,R3      ; POINT TO SDI COMMAND
 48 004322  021204                          CALL    TALK            ; INITIATE SDI INTERCHANGE
 49 004323  115002                          TST     R2              ; SEE IF ANY ERRORS OCCURRED
 50 004324  014331                          BEQ     CLROK           ; IF NOT, BRANCH
 51 004325                                  CERROR  5,#SER1         ; REPORT SECONDARY ERROR
    004325  104200  010100  001013                                                  MOV     #SER1,OUT.05
 52 004330  004365                          BR      MODOK0          ; BRANCH TO ERROR LOOP
 53 004331  104653  000023        CLROK:    MOV     U.WRIT(R5),R3   ; GET WRITE PROTECTION BUTTON STATUS
 54 004333  101203  177406                  BIS     #177406,R3      ; SET OTHER CHANGE MODE BITS
 55 004335  104030  001547                  MOV     R3,WRITBT       ; MOVE TO CHANGE MODE COMMAND
 56 004337  104203  001462                  MOV     #CR.MOD,R3      ; POINT TO MODE COMMAND
 57 004341  021204                          CALL    TALK            ; INITIATE SDI INTERCHANGE
 58 004342  115002                          TST     R2              ; SEE IF ANY ERRORS OCCURRED
 59 004343  014350                          BEQ     TSTRCB          ; IF NOT, BRANCH
 60 004344                                  CERROR  5,#SER3         ; REPORT SECONDARY ERROR
    004344  104200  010146  001013                                                  MOV     #SER3,OUT.05
 61 004347  004365                          BR      MODOK0          ; BRANCH TO ERROR LOOP
 62 004350  104303  004367        TSTRCB:   MOV     RCBREQ,R3       ; SEE IF RECALIBRATE REQUESTED
 63 004352  014364                          BEQ     MODOK           ; IF NOT, BRANCH
 64 004354  104203  001532                  MOV     #CR.INR,R3      ; POINT TO INITIATE RECALIBRATE COMMAND
 65 004355  021204                          CALL    TALK            ; SEND INITIATE RECALIBRATE COMMAND
 66 004356  115002                          TST     R2              ; SEE IF ERROR OCCURRED
 67 004357  014364                          BEQ     MODOK           ; IF NOT, BRANCH
 68 004360                                  CERROR  5,#SER7         ; NOTE SECONDARY ERROR
    004360  104200  010275  001013                                                  MOV     #SER7,OUT.05
 69 004363  004500                          BR      CHKEXT          ; BRANCH
 70 004364  114003                MODOK:    CLR     R3              ; FLAG AS RECOVERED
 71 004365                        MODOK0:   POP     R0              ; RESTORE R1
    004365  104267                                                                  MOV (SP)+,R0
 72 004366  000000                          RETURN                  ; RETURN TO CALLING PROGRAM
 73
 74 004367  000000                RCBREQ:   .WORD   0               ; RECALIBRATE REQUESTED FLAG
```

```
                                .SBTTL  CHKSTA - CHECK STATUS FOR ANY ERRORS THAT NEED RECOVERY
004370                          CHKSTA:
                                ;
                                ;       CHECK THE STATUS OF THE DRIVE AND TAKE THE NECESSARY ACTION
                                ;       (IF THE PORT OR RUN SWITCH IS OUT, OR THE SPINDLE IS NOT READY,
                                ;       DROP THE DRIVE AND DISCONNECT)  IF RR IS SET, RECALIBRATE THE DRIVE,
                                ;       STORE THE WRITE PROTECT SWITCH SETTINGS IN U.WRIT(R5) SO CHANGE MODE
                                ;       CAN WRITE PROTECT THE SUBUNIT(S) IF REQUESTED
                                ;
004370  104207  177777                MOV     #-1,R0          ; SET UP FOR COMPLEMENT OF STATUS
004372  103307  001571                BIC     ST+ST.STA,R0    ; R0 HAS STATUS
004374  102207  000023                BIT     #ST.PS.ST.SR!ST.RU,R0   ; SEE IF ANY FATAL STATES EXIST
004376  014446                        BEQ     EVRYOK          ; IF NOT, BRANCH
004377  104207  000014                MOV     #ERRMC,R0       ; MOVE HOST REQUEST NUMBER TO R0
004401                                DEVFTL  100             ; REPORT SWITCH OUT OR SPINDLE NOT READY
004401  1042C0  010455  001012                                        MOV     #ER100,OUT.04
004404  104200  000144  001010                                        MOV     #100,OUT.02
004407  101200  000400  001010                                        BIS     #FTLDEV,OUT.02
004412  104200  004412  001007                                        MOV     #.,OUT.01
004415  104202  000014                                                MOV     #ERRMC,R2
004417  104020  001006                                                MOV     R2,OUT.R0
004421  104200  100006  002357        MOV     #100006,ERRPOS  ; STATUS VALID, START AT OUT.06
004424  024030                        CALL    REPERR          ; REPORT ERROR
004425  104203  001525                MOV     #CR.DIS,R3      ; POINT TO DISCONNECT COMMAND
004427  021204                        CALL    TALK            ; SEND DISCONNECT COMMAND
004430  115002                        TST     R2              ; SEE IF ANY ERRORS OCCURRED
004431  014436                        BEQ     OK2DRP          ; IF NOT, BRANCH
004432                                CERROR  5,#SER6         ; NOTE SECONDARY ERROR
004432  104200  010254  001013                                        MOV     #SER6,OUT.05
004435  004500                        BR      CHKEXT          ; BRANCH
004436  104653  000033        OK2DRP: MOV     U.PARM(R5),R3   ; GET NEXT UNIT POINTER
004440  101203  100000                BIS     #DROP,R3        ; FLAG ENTIRE UNIT AS DROPPED
004442  100653  000033                MOV     R3,U.PARM(R5)   ; SAVE DROP FLAG
004444  104032                        MOV     R3,R2           ; FLAG AS ERROR (NOT REALLY)
004445  004500                        BR      CHKEXT          ; BRANCH
004446  114003        EVRYOK: CLR     R3              ; FLAG RECALIBRATE AS NOT REQUIRED
004447  102207  000100                BIT     #ST.RR,R0       ; SEE IF DRIVE IS TO BE RECALIBRATED
004451  054454                        BNE     NORCLB          ; IF NOT, BRANCH
004452  104203  177777                MOV     #-1,R3          ; FLAG RECALIBRATE AS REQUESTED
004454  104030  004367        NORCLB: MOV     R3,RCBREQ       ; SAVE RECALIBRATE FLAG
004456  104307  001571                MOV     ST+ST.MOD,R0    ; GET MODE BITS
004460  110707                        SWAB    R0              ; MOVE WRITE PROTECT BUTTON STATUS TO LO BYTE
004461  103207  177417                BIC     #LBHINB,R0      ; CLEAR UNUSED BITS
004463  104642  000000                MOV     S.PARM(R4),R2   ; GET SUBUNIT PARAMETER BITS
004465  102202  020000                BIT     #DCYLS,R2       ; SEE IF IN DIAGNOSTIC AREA
004467  014473                        BEQ     1$              ; IF NOT, BRANCH
004470  115000  002444                TST     M.PARM          ; SEE IF INITIAL WRITE IN PROGRESS
004472  074475                        BMI     2$              ; IF SO, BRANCH
004473  101657  000032        1$:     BIS     U.WPRT(R5),R0   ; SET WRITE PROT BITS FOR READ ONLY DRIVES
004475  100657  000023        2$:     MOV     R0,U.WRIT(R5)   ; SAVE WRITE PROTECT STATUS FOR CHANGE MODE
004477  114002                        CLR     R2              ; FLAG AS NO ERRORS
004500  000000        CHKEXT: RETURN                          ; RETURN TO CALLING PROGRAM
```

```
********** NON-OVERLAY MODULE REDSET - READ/WRITE SETUP MODULE

                                        .SBTTL  ********** NON-OVERLAY MODULE REDSET - READ/WRITE SETUP MODULE
004501                                  REDSET:
                                        ;
                                        ;       SETUP FOR THE READ ERROR RECOVERY LEVELS
                                        ;
004501  104652  000026                  MOV     U.MLEV(R5),R2   ; GET MAXIMUM ERROR REVCOVERY LEVEL
004503  100652  000024                  MOV     R2,U.ELEV(R5)   ; STORE IN CURRENT LEVEL
004505  104207  000004                  MOV     #RDWRT,R0       ; RDWRT NEXT MODULE CALLED
004507  114001                          CLR     R1              ; CALL IMIDIATELY
004510  100651  000016                  MOV     R1,U.NSEC(R5)   ; NO SECTORES SUCCESSFULLY READ/WRITTEN
004512  114002                          CLR     R2              ; NO ERRORS
004513  003127                          BR      JMPRET          ; RETURN TO SEQUENCER
        004515                  AREA0           .+1
```

```
                                .SBTTL  ********** NON-OVERLAY MODULE START - TEST 4 INITIALIZATION
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;
                                ;       THIS 'OVERLAY' IS LOADED ALONG WITH THE INITIAL DOWNLINE LOAD OF
                                ;       TEST 4.  ONCE EXECUTED, THIS OVERLAY IS NEVER LOADED AGAIN UNLESS
                                ;       THE ENTIRE TEST IS STARTED (OR RESTARTED) AGAIN.  ALL OTHER
                                ;       OVERLAYS ARE LOADED INTO THE SPACE THAT THIS ONE OCCUPIES INITIALLY,
                                ;       THUS DESTROYING THIS OVERLAY.
                                ;
004514  000020          UNITS:  .REPT   16.
                                .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
                                .ENDR
004514  177777                 .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
004515  177777                 .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
004516  177777                 .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
004517  177777                 .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
004520  177777                 .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
004521  177777                 .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
004522  177777                 .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
004523  177777                 .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
004524  177777                 .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
004525  177777                 .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
004526  177777                 .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
004527  177777                 .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
004530  177777                 .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
004531  177777                 .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
004532  177777                 .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
004533  177777                 .WORD   177777          ; UNITS AND THEIR SURPERVISOR NUM
004534  000000          UCURSR: .WORD   0
004535  007717          LASTU:  .WORD   FIRSTU
004536  000000          UMASK:  .WORD   0
004537  000000          NUMBB:  .WORD   0
004540                  MAXDBN: .BLKW   2
004542  000000          USED:   .WORD   0
004543  000000          SECTRK: .WORD   0
004544  000000          SECGRP: .WORD   0
004545  000000          SECCYL: .WORD   0
004546                  TGS:    .BLKW   7
```

********** NON-OVERLAY MODULE START - TEST 4 INITIALIZATION

```
 1 004555  027467                    START:  CALL    GMPARM          ; GET MASTER PARAMETERS
 2 004556  027543                            CALL    GETU            ; GET ALL UNITS TO TEST
 3 004557  024644                            CALL    BLDUNT          ; BUILD UNIT AND SUBUNIT PARAMETERS
 4 004560  025527                            CALL    TROOT           ; FILL OUT UNIT AND SUBUNIT PARM'S, GET SUBUNIT CHAR
 5 004561  104307  002473                    MOV     MEMPOL,R0       ; R0 HAS POINTER TO FREE MEMORY
 6 004563  107207  006031                    SUB     #RUFARA,R0      ; SUBTRACT END OF CODE FROM FREE MEMORY
 7 004565  104073                            MOV     R0,R3           ; SAVE AMOUNT OF FREE MEMORY
 8 004566  114001                            CLR     R1              ; CLEAR COUNT
 9 004567  107207  000424            1$:     SUB     #RBUFLN+LINKLN,R0       ; SUBTRACT NEEDED MEM FOR READ
10 004571  074574                            BMI     2$              ; IF NO MORE MEMORY, BRANCH
11 004572  115401                            INC     R1              ; INCREMENT COUNT
12 004573  004567                            BR      1$              ; LOOP
13 004574  104010  002353            2$:     MOV     R1,SECMAX       ; SAVE IN SECTOR MAXIMUM
14 004576  054631                            BNE     3$              ; IF NO ERROR, BRANCH
15 004577                                    DEVFTL  32              ; INSUFFICENT MEMORY FOR READ/WRITE
   004577  104200  004444  001012                                                    MOV     #ER32,OUT.04
   004602  104200  000040  001010                                                    MOV     #32,OUT.02
   004605  101200  000400  001010                                                    BIS     #FTLDEV,OUT.02
   004610  104200  004610  001007                                                    MOV     #.,OUT.01
   004613  104202  000014                                                            MOV     #ERRMC,R2
   004615  104020  001006                                                            MOV     R2,OUT.RQ
16 004617  104200  177777  001011            MOV     #177777,OUT.03  ; NOT ASSOCIATED WITH ANY UNIT
17 004622  104207  000014                    MOV     #ERRMC,R0       ; SET UP FOR REPORT
18 004624  020751                            CALL    HOSTRQ          ; REPORT ERROR
19 004625  104207  000016                    MOV     #DONE,R0        ; END TEST
20 004627  020751                            CALL    HOSTRQ          ; SEND END-OF-TEST TO HOST
21 004630  060021                            XFC     EXIT            ; TERMINATE DM
22 004631  114001                    3$:     CLR     R1              ; CLEAR COUNT
23 004632  107203  000410                    SUB     #WBUFLN+LINKLN,R3       ; SUBTRACT NEEDED MEM FOR READ
24 004634  115401                    4$:     INC     R1              ; INCREMENT COUNT
25 004635  107203  000007                    SUB     #LINKLN,R3      ; SUBTRACT LINK LENGTH FROM MEMORY
26 004637  074641                            BMI     5$              ; IF MEMORY EXHAUSTED, BRANCH
27 004640  004634                            BR      4$              ; LOOP
28 004641  104010  002354            5$:     MOV     R1,CHNMAX       ; SAVE IN SECTOR MAXIMUM
29 004643  002777                            BR      ROOT            ; START EXERCISE
```

BLDUNT - BUILD THE UNIT PARAMETER BLOCK

```
                                .SBTTL  BLDUNT - BUILD THE UNIT PARAMETER BLOCK
004644                          BLDUNT:
                                ;
                                ;       BLDUNT WILL BUILD THE UNIT AND SUBUNIT PARAMETERS
                                ;
004644  104202  000001                  MOV     #1,R2           ; PORT 1 MASK
004646  104020  004536                  MOV     R2,UMASK        ; SAVE MASK
004650  104204  004514                  MOV     #UNITS,R4       ; POINT TO UNITS TO BE TESTED TABLE
004652  104147                  ULOP:   MOV     (R4),R0         ; GET UNIT TO BE TESTED
004653  074730                          BMI     NOU             ; IF FIRST UNIT NOT TO BE TESTED, BRANCH
004654  102207  040000                  BIT     #40000,R0       ; SEE IF FIRST SUBUNIT TO BE TESTED
004656  014676                          BEQ     3$              ; IF SO, BRANCH
004657  104647  000001                  MOV     1(R4),R0        ; GET SECOND SUBUNIT INFORMATION
004661  102207  040000                  BIT     #40000,R0       ; SEE IF THIS SUBUNIT IS USED
004663  014676                          BEQ     3$              ; IF SO, BRANCH
004664  104647  000002                  MOV     2(R4),R0        ; SEE IF THIRD SUBUNIT TO BE TESTED
004666  102207  040000                  BIT     #40000,R0       ; CHECK TESTING BIT
004670  014676                          BEQ     3$              ; IF IT IS TO BE TESTED, BRANCH
004671  104647  000003                  MOV     3(R4),R0        ; SEE IF FOURTH SUBUNIT TO BE TESTED
004673  102207  040000                  BIT     #40000,R0       ; TEST TESTING BIT
004675  054730                          BNE     NOU             ; IF UNIT (THE WHOLE THING) NOT TO BE TESTED, BRANCH
004676  104207  000061          3$:     MOV     #TLEN.U,R0      ; R0 HAS UNIT DATA STRUCTURE LENGTH
004700  025072                          CALL    GETMEM          ; GET MEMORY
004701  100707  177632                  MOV     R0,@LASTU       ; LINK LAST POINTER TO THIS ONE
004703  104070  004535                  MOV     R0,LASTU        ; SAVE POINTER TO THIS ONE
004705  100672  000022                  MOV     R2,U.MASK(R0)   ; SAVE PORT SELECT MASK
004707  104203  000004                  MOV     #4,R3           ; MAXIMUM OF FOUR SUBUNITS
004711                                  PUSH    R0              ; SAVE POINTER TO UNIT INFORMATION
004711  100467                                                                  MOV R0,-(SP)
004712  105207  000047                  ADD     #U.SNUM,R0      ; R0 WILL POINT TO SUBUNIT NUMBERS
004714  104245                  1$:     MOV     (R4)+,R5        ; MOVE SUBUNIT NUMBER TO UNIT PARAMETERS
004715  102205  040000                  BIT     #40000,R5       ; SEE IF SUBUNIT TO BE TESTED
004717  014722                          BEQ     2$              ; IF SO, BRANCH
004720  104205  177777                  MOV     #-1,R5          ; FLAG SUBUNIT AS NOT TESTED
004722  100275                  2$:     MOV     R5,(R0)+        ; MOVE TO SUBUNIT NUMBERS
004723  117403                          DEC     R3              ; DECREMENT COUNT
004724  054714                          BNE     1$              ; IF ALL SUBUNITS ARE FILLED IN, BRANCH
004725                                  POP     R0              ; RESTORE R0
004725  104267                                                                  MOV (SP)+,R0
004726  107204  000004                  SUB     #4,R4           ; R4 NOW POINTS TO BEGINNING OF UNIT HANDLED
004730  104302  004536          NOU:    MOV     UMASK,R2        ; GET UNIT PORT MASK
004732  110202                          ROL     R2              ; ROTATE TO NEXT PORT
004733  104020  004536                  MOV     R2,UMASK        ; SAVE MASK
004735  105204  000004                  ADD     #4,R4           ; R4 POINTS TO NEXT UNIT INFORMATION
004737  106204  004534                  CMP     #UNITS+16.,R4   ; SEE IF ALL UNITS SET UP
004741  054652                          BNE     ULOP            ; IF ALL UNITS NOT SETUP, BRANCH
004742  104203  007717                  MOV     #FIRSTU,R3      ; COMPLETE RING
004744  100703  177567                  MOV     R3,@LASTU       ; LAST UNIT NOW POINTS TO FIRST
004746  104207  007717                  MOV     #FIRSTU,R0      ; R0 POINTS TO FIRST UNIT PARAMETERS
004750                          SUSETL: PUSH    R0              ; SAVE POINTER TO UNIT PARAMETERS
004750  100467                                                                  MOV R0,-(SP)
004751  024763                          CALL    BLDSUS          ; BUILD SUBUNIT PARAMETERS
004752                                  POP     R0              ; RESTORE UNIT POINTER
004752  104267                                                                  MOV (SP)+,R0
004753  104677  000000                  MOV     U.NEXT(R0),R0   ; GO TO NEXT UNIT
004755  103207  170000                  BIC     #^C<HBHINB>,R0  ; CLEAR UNUSED BITS
004757  106207  007717                  CMP     #FIRSTU,R0      ; SEE IF THE ENTIRE RING IS SET UP
```

```
     54 004761  05475C                       BNE     SUSETL          ; IF NOT, BRANCH
     55 004762  000000                       RETURN
```

```
                                        .SBTTL  BLDSUS - BUILD ALL SUBUNIT PARAMETER BLOCKS ON THIS UNIT
004763                                  BLDSUS:
                                        ;
                                        ;       BUILD ALL SUBUNIT PARAMETERS
                                        ;
004763  104204  000001                          MOV     #U.SUBP,R4      ; R4 WILL POINT TO SUBUNIT POINTERS
004765  105074                                  ADD     R0,R4           ; R4 POINTS TO SUBUNIT POINTERS
004766  104203  000047                          MOV     #U.SNUM,R3      ; R3 WILL POINT TO SUBUNIT NUMBERS
004770  105073                                  ADD     R0,R3           ; R3 POINTS TO SUBUNIT NUMBERS
004771  104205  000004                          MOV     #4,R5           ; MAXIMUM NUMBER OF SUBUNITS
004773  104237                  SLOP:   MOV     (R3)+,R0        ; R0 GETS SUBUNIT NUMBER
004774  075007                          BMI     NOSUN           ; IF SUBUNIT INACTIVE, BRANCH
004776  104070  004534                  MOV     R0,UCURSR       ; UCURSR GETS SUBUNIT NUMBER
004777                                  PUSH    <R3,R4,R5>      ; SAVE R3 - R5
                                                                                        MOV R3,-(SP)
                                                                                        MOV R4,-(SP)
                                                                                        MOV R5,-(SP)
005002  025015                          CALL    BLDSUB          ; BUILD THE SUBUNIT PARAMETERS
005003                                  POP     <R5,R4,R3>      ; RESTORE R3 - R5
                                                                                        MOV (SP)+,R5
                                                                                        MOV (SP)+,R4
                                                                                        MOV (SP)+,R3
005006  005011                          BR      BSUSEX          ; BRANCH
005007  104207  177777          NOSUN:  MOV     #-1,R0          ; TO FLAG SUBUNIT AS INACTIVE
005011  100247                  BSUSEX: MOV     R0,(R4)+        ; MOVE POINTER TO SUB PARAM TO UNIT PARAMS
005012  117405                          DEC     R5              ; DECREMENT COUNT
005013  054773                          BNE     SLOP            ; IF UNEXPIRED, BRANCH
005014  000000                          RETURN                  ; RETURN TO CALLING PROGRAM
```

BLDSUB - BUILD A SUBUNIT PARAMETER BLOCK

```
      1                                 .SBTTL  BLDSUB - BUILD A SUBUNIT PARAMETER BLOCK
      2 005015                          BLDSUB:
      3                                 ;
      4                                 ;       BLDSUB WILL BUILD THE SUBUNIT PARAMETER BLOCK
      5                                 ;
      6 005015  025022                          CALL    BLDBES          ; SET UP BEGIN/END SETS
      7 005016  025053                          CALL    BLDBB           ; SET UP BAD BLOCKS
      8 005017  025072                          CALL    GETMEM          ; GET MEMORY FOR SUBUNIT PARAMETERS
      9 005020  025077                          CALL    COPYSU          ; COPY ALL SUBUNIT PARAMETERS
     10 005021  000000                          RETURN
```

```
                                        .SBTTL  BLDBES - FIND HOW MANY WORDS NEEDED FOR THE BEGIN/END OR TRACK/GROUP SETS
005022                                  BLDBES:
                                        ;
                                        ;       COMPUTE THE BEGIN/END SET AREA NEEDED
                                        ;
005022  104300  004534  001007                 MOV     UCURSR,OUT.01   ; MOVE SUBUNIT NUMBER TO COMMUNICATION AREA
005025  104207  000004                        MOV     #T4UPRM,R0      ; MOVE REQUEST NUMBER TO R0
005027  020751                               CALL    HOSTRQ          ; REQUEST INFORMATION FROM HOST
005030  102200  000040  001045               BIT     #BEUSED,IN.01   ; SEE IF BEGIN/END SETS ARE USED
005033  055041                               BNE     2$              ; IF SO, BRANCH
005034  104207  000016                        MOV     #S.TGSS+1,R0    ; MONIMUM CONFIGURATION LENGTH
005036  105307  001053                        ADD     IN.07,R0        ; ADD NUMBER OF T/G SETS
005040  005052                               BR      NOBES           ; BRANCH
005041  104207  000011          2$:           MOV     #S.BESS,R0      ; MINIMUM CONFIGURATION WORD COUNT
005043  104301  001047                        MOV     IN.03,R1        ; GET NUMBER OF BEGIN/END SETS
005045  055047                               BNE     1$              ; IF NON-ZERO, BRANCH
005046  115401                               INC     R1              ; MAKE R1 1
005047  105011          1$:           ADD     R1,R1           ; MAKE B/E SETS * 2
005050  105011                               ADD     R1,R1           ; MAKE B/E SETS * 4
005051  105017                               ADD     R1,R0           ; ADD TO LENGTH OF SUBUNIT PARAMETERS
005052  000000          NOBES:        RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                        .SBTTL  BLDBB  - FIND HOW MANY WORDS NEEDED FOR THE BAD BLOCKS
005053                                  BLDBB:
                                        ;
                                        ;       COUNT THE BAD BLOCK PARAMETERS
                                        ;
005053                                          PUSH    R0              ; SAVE SUBUNIT LENGTH
005053  100467                                                                  MOV R0,-(SP)
005054  104300  004534  001007                    MOV     UCURSR,OUT.01   ; MOVE SUBUNIT NUMBER TO COMMUNICATION AREA
005057  104207  000005                          MOV     #T4BB1,R0       ; MOVE REQUEST NUMBER TO R0
005061  020751                                  CALL    HOSTRQ          ; REQUEST INFORMATION FROM HOST
005062                                          POP     R0              ; RESTORE SUBUNIT LENGTH
005062  104267                                                                  MOV (SP)+,R0
005063  104301  001045                          MOV     IN.01,R1        ; GET NUMBER OF BAD BLOCKS
005065  104010  004537                          MOV     R1,NUMBB        ; SAVE NUMBER OF BAD BLOCKS
005067  105011                                  ADD     R1,R1           ; MAKE BAD BLOCKS * 2
005070  105017                                  ADD     R1,R0           ; ADD TO SUBUNIT DATA STRUCTURE LENGTH
005071  000000                        NOBBS:    RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                        .SBTTL  GETMEM - ALLOCATES MEMORY FOR UNIT AND SUBUNIT BLOCK AND SUBUNIT PARAMETERS
005072                                  GETMEM:
                                        ;
                                        ;       RETURN A BLOCK OF MEMORY
                                        ;
005072  107070  002473                  SUB     R0,MEMPOL       ; SUBTRACT REQUESTED LENGTH FROM MEMOY POOL
005074  104307  002473                  MOV     MEMPOL,R0       ; LOAD R0 WITH POINTER TO REQUESTED MEMORY
005076  000000                          RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                        .SBTTL  COPYSU - COPY ALL SUBUNIT PARAMETERS TO SUBUNIT BLOCK
005077                                  COPYSU:
                                        ;
                                        ;       COPY SUBUNIT PARAMETRS TO SUBUNIT BLOCK
                                        ;
                                        ;
005077                                          PUSH    R0              ; SAVE R0
005077  100467                                                          MOV R0,-(SP)
005100  104300  004534  001007                  MOV     UCURSR,OUT.01   ; MOVE SUBUNIT NUMBER TO COMMUNICATION AREA
005103  104207  000004                          MOV     #T4UPRM,R0      ; MOVE REQUEST NUMBER TO R0
005105  020751                                  CALL    HOSTRQ          ; REQUEST INFORMATION FROM HOST
005106                                          POP     R0              ; RESTORE R0
005106  104267                                                          MOV (SP)+,R0
005107  104301  001046                          MOV     IN.02,R1        ; GET PATTREN NUMBER
005111  100671  000003                          MOV     R1,S.PAT(R0)    ; SAVE
005113  104305  001045                          MOV     IN.01,R5        ; GET UNIT PARAMETERS
005115  102205  000040                          BIT     #BEUSED,R5      ; SEE IF BEGIN/END SETS USED
005117  055165                                  BNE     3$              ; IF SO, BRANCH
005120  100675  000000                          MOV     R5,S.PARM(R0)   ; SAVE IN SUBUNIT AREA
005122  104201  000011                          MOV     #S.BESS,R1      ; R1 WILL POINT TO BEGIN/END SET AREA
005124  105071                                  ADD     R0,R1           ; R1 POINTS TO TRACK/GROUP AREA
005125  104202  001047                          MOV     #IN.03,R2       ; R2 POINTS TO START/END CYL
005127  104225                                  MOV     (R2)+,R5        ; MOVE LIMITING CYLINDER TO SUBUNIT PARAMETERS
005130  100615  000002                          MOV     R5,2(R1)        ; MOVE LIMITING CYLINDER TO SUBUNIT PARAMETERS
005132  104225                                  MOV     (R2)+,R5        ; MOVE LIMITING CYLINDER TO SUBUNIT PARAMETERS
005133  100615  000003                          MOV     R5,3(R1)        ; MOVE LIMITING CYLINDER TO SUBUNIT PARAMETERS
005135  104225                                  MOV     (R2)+,R5        ; MOVE LIMITING CYLINDER TO SUBUNIT PARAMETERS
005136  100115                                  MOV     R5,(R1)         ; MOVE LIMITING CYLINDER TO SUBUNIT PARAMETERS
005137  104225                                  MOV     (R2)+,R5        ; MOVE LIMITING CYLINDER TO SUBUNIT PARAMETERS
005140  100615  000001                          MOV     R5,1(R1)        ; MOVE LIMITING CYLINDER TO SUBUNIT PARAMETERS
005142  105201  000004                          ADD     #4,R1           ; R1 POINTS AFTER LIMITING CYLINDERS
005144  104224                                  MOV     (R2)+,R4        ; GET TRACK/GROUP COUNT
005145  104225                          5$:     MOV     (R2)+,R5        ; GET WORD
005146  100215                                  MOV     R5,(R1)+        ; SAVE
005147  117404                                  DEC     R4              ; DECREMENT COUNT
005150  055145                                  BNE     5$              ; IF INCOMPLETE, BRANCH
005151  104415                                  MOV     -(R1),R5        ; GET LAST T/G
005152  101205  100000                          BIS     #100000,R5      ; MARK AS END OF LIST
005154  100115                                  MOV     R5,(R1)         ; SAVE
005155  104201  000015                          MOV     #S.TGSS,R1      ; R1 WILL POINT TO START OF T/G LIST
005157  105071                                  ADD     R0,R1           ; R1 POINTS TO START OF T/G LIST
005160  025502                                  CALL    SORTTG          ; SORT THE TRACK/GROUPS
005161  105301  001053                          ADD     IN.07,R1        ; ADD NUMBER OF T/G'S
005163  115401                                  INC     R1              ; R1 NOW POINTS TO AFTER T/G'S (FOR BAD BLOCKS)
005164  005241                                  BR      COPBB           ; BRANCH
005165  104201  000011                  3$:     MOV     #S.BESS,R1      ; R1 WILL POINT TO BEGIN/END SETS
005167  105071                                  ADD     R0,R1           ; R1 POINTS TO BEGIN/END SETS
005170  104202  001047                          MOV     #IN.03,R2       ; POINT TO BEGIN/END SET COUNT
005172  115000  001047                          TST     IN.03           ; SEE IF ANY BEGIN/END SETS SUPPLIED
005174  055220                                  BNE     2$              ; IF SO, BRANCH
005175  115402                                  INC     R2              ; SKIP COUNT
005176  101205  000200                          BIS     #ONLYCL,R5      ; FLAG AS BEGIN/END CYLINDER SUPPLIED
005200  100675  000000                          MOV     R5,S.PARM(R0)   ; SAVE IN SUBUNIT AREA
005202  104224                                  MOV     (R2)+,R4        ; GET LO STARTING CYL
005203  100614  000002                          MOV     R4,2(R1)        ; SAVE
005205  104224                                  MOV     (R2)+,R4        ; GET HI STARTING CYL
005206  100614  000003                          MOV     R4,3(R1)        ; SAVE
005210  104224                                  MOV     (R2)+,R4        ; GET LO ENDING CYL
```

```
 56 005211  100114                        MOV     R4,(R1)          ; SAVE
 57 005212  104224                        MOV     (R2)+,R4         ; GET HI ORDER ENDING CYL
 58 005213  100614  000001                MOV     R4,1(R1)         ; SAVE
 59 005215  105201  000004                ADD     #4,R1            ; R1 POINTS AFTER BEGIN/END CYLINDERS
 60 005217  005241                        BR      COPBB            ; CONTINUE
 61 005220  100675  000000        2$:     MOV     R5,S.PARM(R0)    ; SAVE IN SUBUNIT AREA
 62 005222  104224                        MOV     (R2)+,R4         ; GET BEGIN/END SET COUNT
 63 005223  105044                        ADD     R4,R4            ; MULTIPLY BEGIN/END SETS BY 4 WORDS
 64 005224  105044                        ADD     R4,R4
 65 005225  055230                        BNE     1$               ; IF BEGIN/END SET COUNT NON-ZERO, BRANCH
 66 005226  104204  000004                MOV     #4,R4            ; MAKE ONE BEGIN END SET
 67 005230  104225                1$:     MOV     (R2)+,R5         ; MOVE BEGIN/END SET TO PARAMETERS
 68 005231  100215                        MOV     R5,(R1)+         ; MOVE BEGIN/END SET TO PARAMETERS
 69 005232  117404                        DEC     R4               ; DECREMENT COUNT
 70 005233  055230                        BNE     1$               ; IF UNEXPIRED, BRANCH
 71 005234  104415                        MOV     -(R1),R5         ; MOVE EOL TO BEGIN/END LIST
 72 005235  101205  100000                BIS     #100000,R5       ; SET HI BIT TO FLAG EOL
 73 005237  100215                        MOV     R5,(R1)+         ; SAVE
 74 005240  025322                        CALL    SORTBE           ; SORT THE BEGIN/END SETS IN ASCENDING ORDER
 75 005241                        COPBB:  PUSH    R0               ; SAVE R0
    005241  100467                                                                    MOV R0,-(SP)
 76 005242  104300  004534  001007        MOV     UCURSR,OUT.01    ; MOVE TO COMMUNICATION AREA
 77 005245  104207  000005                MOV     #T4BB1,R0        ; MOVE REQUEST NUMBER TO R0
 78 005247  020751                        CALL    HOSTRQ           ; REQUEST INFORMATION FROM HOST
 79 005250                                POP     R0               ; RESTORE R0
    005250  104267                                                                    MOV (SP)+,R0
 80 005251  104202  001046                MOV     #IN.02,R2        ; R2 POINTS AT BAD BLOCKS
 81 005253  104304  004537                MOV     NUMBB,R4         ; R4 IS NUMBER OF BAD BLOCKS
 82 005255  015314                        BEQ     NOBB             ; IF NO BAD BLOCKS, BRANCH
 83 005256  100671  000010                MOV     R1,S.BADP(R0)    ; MOVE POINTER TO BAD BLOCKS TO SUB PARAM
 84 005260  114004                        CLR     R4               ; START LOOP AT ZERO
 85 005261  106204  000016        BBLOP:  CMP     #14.,R4          ; SEE IF SECOND BLOCK NEEDED
 86 005263  055276                        BNE     NOEXTR           ; IF NOT, BRANCH
 87 005264                                PUSH    R0               ; SAVE R0
    005264  100467                                                                    MOV R0,-(SP)
 88 005265  104300  004534  00*007        MOV     UCURSR,OUT.01    ; SAVE IN COMMUNICATION AREA
 89 005270  104207  000006                MOV     #T4BB2,R0        ; LOAD REQUEST NUMBER
 90 005272  020751                        CALL    HOSTRQ           ; REQUEST FURTHER BLOCKS FROM HOST
 91 005273                                POP     R0               ; RESTORE R0
    005273  104267                                                                    MOV (SP)+,R0
 92 005274  104202  001045                MOV     #IN.01,R2        ; POINT R2 TO BAD BLOCKS
 93 005276  104225                NOEXTR: MOV     (R2)+,R5         ; GET BAD BLOCK
 94 005277  100215                        MOV     R5,(R1)+         ; SAVE BAD BLOCK
 95 005300  104225                        MOV     (R2)+,R5         ; GET BAD BLOCK
 96 005301  100215                        MOV     R5,(R1)+         ; SAVE BAD BLOCK
 97 005302  115404                        INC     R4               ; INCREMENT R4
 98 005303  106304  004537                CMP     NUMBB,R4         ; SEE IF ALL BLOCKS COPIED
 99 005305  055261                        BNE     BBLOP            ; IF NOT, BRANCH
100 005306  104415                        MOV     -(R1),R5         ; GET LAST BAD BLOCK
101 005307  101205  100000                BIS     #100000,R5       ; FLAG AS EOL
102 005311  100215                        MOV     R5,(R1)+         ; SAVE EOL
103 005312  025414                        CALL    SORTBB           ; SORT THE BAD BLOCKS IN ASCENDING ORDER
104 005313  005316                        BR      CPYBEX           ; BRANCH
105 005314  100674  000010        NOBB:   MOV     R4,S.BADP(R0)    ; FLAG AS NO BAD BLOCKS
106 005316  114004                CPYBEX: CLR     R4               ; CLEAR R4
107 005317  100674  000005                MOV     R4,S.SCHR(R0)    ; FLAG CHARACTERISTICS AS NOT YET DEFINED
108 005321  000000                        RETURN                   ; RETURN TO CALLING PROGARM
```

```
                                        .SBTTL  SORTBE - SORT THE BEGIN/END SETS IN ASCENDING ORDER
005322                                  SORTBE:
                                        ;
                                        ;       SORT THE BEGIN/END SETS IN ASCENDING ORDER
                                        ;
005322  104672  000014                          MOV     S.BESS+3(R0),R2 ; SEE IF ONLY ONE BEGIN END SET
005324  075346                                  BMI     4$              ; IF SO, EXIT (ALLREADY SORTED)
005325  104202  000011                          MOV     #S.BESS,R2      ; R2 WILL POINT TO START OF BEGIN/END SETS
005327  105072                                  ADD     R0,R2           ; R2 POINTS TO BEGIN/END SETS
005330  104023                  1$:     MOV     R2,R3           ; R3 WILL POINT TO NEXT BEGIN/END SET
005331  105203  000004          2$:     ADD     #4,R3           ; R3 POINTS TO BEGIN/END SETS
005333  025347                                  CALL    CBES2           ; COMPARE THE TWO BEGIN/END SETS
005334  045336                                  BCC     3$              ; IF R2->B/E <= R3->B/E THEN ALLREADY IN ORDER
005335  025364                                  CALL    SWAPBE          ; SWAP TE BEGIN/END SETS
005336  104635  000003          3$:     MOV     3(R3),R5        ; SEE IF R3->END-OF-LIST
005340  035331                                  BPL     2$              ; IF NOT, BRANCH
005341  105202  000004                          ADD     #4,R2           ; R2->NEXT BEGIN/END SET
005343  104625  000003                          MOV     3(R2),R5        ; SEE IF R2->END-OF-LIST
005345  035330                                  BPL     1$              ; IF NOT, BRANCH
005346  000000                  4$:     RETURN                  ; RETURN TO COPYSU

005347                                  CBES2:
                                        ;
                                        ;       24 BIT COMPARE FOR BEGIN/END SET SORTING
                                        ;
005347  104634  000003                          MOV     3(R3),R4        ; GET WORD THAT R3 POINTS TO
005351  103204  177400                          BIC     #HIBYTE,R4      ; STRIP OFF UNUSED BITS
005353  104625  000003                          MOV     3(R2),R5        ; GET OTHER WORD TO COMPARE
005355  106045                                  CMP     R4,R5           ; COMPARE
005356  055363                                  BNE     1$              ; IF DIFFERENCE IS FOUND, BRANCH
005357  104625  000002                          MOV     2(R2),R5        ; GET OTHER WORD TO COMPARE
005361  106635  000002                          CMP     2(R3),R5        ; COMPARE
005363  000000                  1$:     RETURN                  ; RETURN TO SORTBE
```

```
                                .SBTTL  SWAPBE - IF BEGIN/END SETS OUT OF ORDER, SWAP
005364                          SWAPBE:
                                ;
                                ;       SWAP THE BEGIN/END SET POINTED TO BY R2 WITH THE BEGIN/END SET
                                ;       POINTED TO BY R3, KEEPING THE END-OF-LIST FLAG INTACT
                                ;
005364                                  PUSH    <R1,R2,R3>      ; SAVE POINTERS
005364  100461                                                                  MOV R1,-(SP)
005365  100462                                                                  MOV R2,-(SP)
005366  100463                                                                  MOV R3,-(SP)
005367  104201  000003                 MOV     #3,R1           ; SET UP LOOP COUNT
005371  104124                  1$:     MOV     (R2),R4         ; GET WORD FROM SET
005372  104135                          MOV     (R3),R5         ; GET WORD FROM OTHER SET
005373  100234                          MOV     R4,(R3)+        ; SWAP WORD
005374  100225                          MOV     R5,(R2)+        ; SWAP WORD
005375  117401                          DEC     R1              ; DECREMENT COUNT
005376  055371                          BNE     1$              ; LOOP IF INCOMPLETE
005377  104124                          MOV     (R2),R4         ; GET WORD FROM SET
005400  104135                          MOV     (R3),R5         ; GET WORD FROM OTHER SET
005401  104051                          MOV     R5,R1           ; MOVE TO TEMP STORAGE
005402  103205  177400                 BIC     #HIBYTE,R5      ; STRIP OFF END-OF-LIST FLAG, IF ANY
005404  107051                          SUB     R5,R1           ; R1 CONTAINS END-OF-LIST FLAG, IF ANY
005405  101014                          BIS     R1,R4           ; R4 NOW HAS END-OF-LIST FLAG, IF ANY
005406  100134                          MOV     R4,(R3)         ; SWAP WORD (AND EOL FLAG, IF ANY)
005407  100125                          MOV     R5,(R2)         ; SWAP WORD
005410                                  POP     <R3,R2,R1>      ; RESTORE THE REGISTERS
005410  104263                                                                  MOV (SP)+,R3
005411  104262                                                                  MOV (SP)+,R2
005412  104261                                                                  MOV (SP)+,R1
005413  000000                          RETURN                  ; RETURN TO SORTBE
```

SORTBB - SORT THE BAD BLOCKS IN ASCENDING ORDER

```
                                        .SBTTL  SORTBB - SORT THE BAD BLOCKS IN ASCENDING ORDER
005414                                  SORTBB:
                                        ;
                                        ;       SORT THE BAD BLOCKS IN ASCENDING ORDER
                                        ;
005414  104672  000010                          MOV     S.BADP(R0),R2   ; GET THE BAD BLOCK POINTER
005416  015440                                  BEQ     4$              ; IF NO BAD BLOCKS, BRANCH
005417  104625  000001                          MOV     1(R2),R5        ; SEE IF ONLY ONE BAD BLOCK
005421  075440                                  BMI     4$              ; IF SO BRANCH (ALLREADY SORTED)
005422  104023                          1$:     MOV     R2,R3           ; R3 WILL POINT AT NEXT BAD BLOCK
005423  105203  000002                  2$:     ADD     #2,R3           ; R3 POINTS TO NEXT BAD BLOCK
005425  025441                                  CALL    CBB2            ; COMPARE THE TWO BLOCKS
005426  045430                                  BCC     3$              ; IF IN ASCENDING ORDER, BRANCH
005427  025456                                  CALL    SWAPBB          ; SWAP THE BAD BLOCKS
005430  104635  000001                  3$:     MOV     1(R3),R5        ; SEE IF END-OF-LIST
005432  035423                                  BPL     2$              ; IF NOT, BRANCH
005433  105202  000002                          ADD     #2,R2           ; R2 POINTS TO NEXT BAD BLOCK
005435  104625  000001                          MOV     1(R2),R5        ; SEE IF END-OF-LIST
005437  035422                                  BPL     1$              ; IF NOT, BRANCH
005440  000000                          4$:     RETURN                  ; RETURN TO COPYSU

005441                                  CBB2:
                                        ;
                                        ;       24 BIT COMPARE FOR BAD BLOCKS
                                        ;
005441  104634  000001                          MOV     1(R3),R4        ; GET WORD THAT R3 POINTS TO
005443  103204  177400                          BIC     #HIBYTE,R4      ; STRIP OFF UNUSED BITS
005445  104625  000001                          MOV     1(R2),R5        ; GET OTHER WORD TO COMPARE
005447  103205  177400                          BIC     #HIBYTE,R5      ; STRIP OFF THE UNUSED BITS
005451  106045                                  CMP     R4,R5           ; COMPARE
005452  055455                                  BNE     1$              ; IF DIFFERENCE IS FOUND, BRANCH
005453  104125                                  MOV     (R2),R5         ; GET OTHER WORD TO COMPARE
005454  106135                                  CMP     (R3),R5         ; COMPARE
005455  000000                          1$:     RETURN                  ; RETURN TO SORTBB
```

```
                                .SBTTL  SWAPBB - IF BAD BLOCKS OUT OF ORDER, SWAP
005456                          SWAPBB:
                                ;
                                ;       SWAP THE BAD BLOCKS, RETAINING THE END-OF-LIST POINTER AT THE END
                                ;
005456                          PUSH    R1              ; SAVE R1
005456  100461                                                          MOV R1,-(SP)
005457  104124                  MOV     (R2),R4         ; GET LO ORDER BAD BLOCK OF 1ST SET
005460  104135                  MOV     (R3),R5         ; GET LO ORDER BAD BLOCK OF 2ND SET
005461  100134                  MOV     R4,(R3)         ; SWAP
005462  100125                  MOV     R5,(R2)         ; SWAP
005463  104624  000001          MOV     1(R2),R4        ; GET HI ORDER BAD BLOCK OF 1ST SET
005465  104635  000001          MOV     1(R3),R5        ; GET HI ORDER BAD BLOCK OF 2ND SET
005467  104051                  MOV     R5,R1           ; MOVE TO R1
005470  103205  177400          BIC     #HIBYTE,R5      ; STRIP OFF END-OF-LIST FLAG, IF ANY
005472  107051                  SUB     R5,R1           ; R1 CONTAINS END-OF-LIST FLAG, IF ANY
005473  101014                  BIS     R1,R4           ; PUT END-OF-LIST FLAG IN R4, IF ANY
005474  100634  000001          MOV     R4,1(R3)        ; SWAP
005476  100625  000001          MOV     R5,1(R2)        ; SWAP
005500                          POP     R1              ; RESTORE R1
005500  104261                                                          MOV (SP)+,R1
005501  000000                  RETURN                  ; RETURN TO SORTBB
```

```
      1                                 .SBTTL  SORTTG - SORT THE TRACK/GROUPS IN ASCENDING ORDER
      2 005502                          SORTTG:
      3                                 ;
      4                                 ;       SORTTG WILL SORT THE TRACKS OR GROUPS IN ASCENDING ORDER
      5                                 ;
      6 005502                                  PUSH    R1              ; SAVE R1
        005502  100461                                                              MOV R1,-(SP)
      7 005503  104115                  1$:     MOV     (R1),R5         ; GET START OF LIST
      8 005504  075525                          BMI     4$              ; IF NEGATIVE, WHOLE LIST IS SORTED, EXIT
      9 005505  104012                          MOV     R1,R2           ; R2 WILL POINT TO NEXT LIST MEMBER
     10 005506  115402                  2$:     INC     R2              ; R2 POINTS TO NEXT MEMBER
     11 005507  104125                          MOV     (R2),R5         ; GET NEXT MEMBER
     12 005510  104054                          MOV     R5,R4           ; COPY TO R4
     13 005511  103205  177400                  BIC     #HIBYTE,R5      ; CLEAR END-OF-LIST FLAG (IF ANY)
     14 005513  107054                          SUB     R5,R4           ; SAVE END-OF-LIST FLAG (IF ANY)
     15 005514  106115                          CMP     (R1),R5         ; SEE IF THE MEMBERS ARE IN ORDER
     16 005515  075521                          BMI     3$              ; IF SO, BRANCH
     17 005516  101114                          BIS     (R1),R4         ; COPY START OF LIST TO R4, RETAINING EOL FLAG
     18 005517  100115                          MOV     R5,(R1)         ; SWAP
     19 005520  100124                          MOV     R4,(R2)         ; SWAP
     20 005521  104125                  3$:     MOV     (R2),R5         ; GET MEMBER THAT SORT POINTER POINTS TO
     21 005522  035506                          BPL     2$              ; IF NOT END-OF-LIST, BRANCH
     22 005523  115401                          INC     R1              ; POINT TO NEXT START OF LIST
     23 005524  005503                          BR      1$              ; LOOP
     24 005525                          4$:     POP     R1              ; RESTORE R1
        005525  104261                                                              MOV (SP)+,R1
     25 005526  000000                          RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                .SBTTL  TROOT  - TEMPORARY ROOT FOR SEQNCR DURING TEST 4 SETUP
005527                          TROOT:
                                ;
                                ;       TROOT WILL FILL OUT THE AS YET UNDEFINED FIELDS IN BOTH UNIT
                                ;       AND SUBUNIT PARAMETERS, AS WELL AS GETTING THE SUBUNIT CAHRACTERISTICS
                                ;
005527  104205  007717                MOV     #FIRSTU,R5      ; R5 POINTS TO FIRST SUBUNIT
005531  114001                MKLOOP: CLR     R1              ; START WITH SUBUNIT ZERO (WEITHER USED OR NOT)
005532  104207  001000                MOV     #FTIME,R0       ; MARK AS FIRST TIME
005534  100657  000033                MOV     R0,U.PARM(R5)   ; SAVE
005536  104201  005562                MOV     #INSET,R1       ; FIRST MODULE IS INSET
005540  100651  000010                MOV     R1,U.NFUN(R5)   ; SAVE IN UNIT PARAMETERS
005542  023062                1$:     CALL    SEQNCR          ; RUN SEQUENCER
005543  106207  003254                CMP     #AFINIT,R0      ; SEE IF THE DRIVE HAS BEEN INITILIZED
005545  015542                        BEQ     1$              ; IF SO, BRANCH
005546  104657  000033                MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
005550  101207  001000                BIS     #FTIME,R0       ; SET FIRST TIME
005552  100657  000033                MOV     R0,U.PARM(R5)   ; SAVE
005554  104655  000000                MOV     U.NEXT(R5),R5   ; TRAVERSE TO NEXT UNIT
005556  106205  007717                CMP     #FIRSTU,R5      ; SEE IF TRAVERSED ENTIRE RING
005560  055531                        BNE     MKLOOP          ; IF NOT, BRANCH
005561  000000                        RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                        .SBTTL  INSET  - SET UP UNIT PARAMETERS FOR SEQNCR ROUTINES
005562                                  INSET:
                                        ;
                                        ;       INSET WILL SET UP EACH UNIT BEFORE IT STARTS RUNNING
                                        ;
005562  104657  000033                          MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
005564  103207  004000                          BIC     #NEWSUB,R0      ; NO LONGER NEW SUBUNITS
005566  100657  000033                          MOV     R0,U.PARM(R5)   ; SAVE
005570  104207  001750                          MOV     #1000.,R0       ; INITILIZE SEEK COUNTER
005572  100657  000007                          MOV     R0,U.SEEK(R5)   ; SAVE SEEK COUNTER
005574  104201  000012                          MOV     #10.,R1         ; SDI TIMEOUTS (LONG/SHORT) SET TO 90 SEC
005576  100651  000030                          MOV     R1,U.SDIS(R5)   ; MOVE TO PARAMETERS
005600  100651  000031                          MOV     R1,U.SDIL(R5)   ; MOVE TO PARAMETERS
005602  104207  005615                          MOV     #COMCHR,R0      ; COMMON CHARACTERISTICS NEXT MODULE CALLED
005604  114002                                  CLR     R2              ; NO ERRORS
005605  100652  000053                          MOV     R2,U.CCYL(R5)   ; CLEAR CYL+GROUP
005607  100652  000054                          MOV     R2,U.CCYL+1(R5)
005611  100652  000055                          MOV     R2,U.CGRP(R5)
005613  114001                                  CLR     R1              ; IMMIDATE CALL
005614  003127                                  BR      JMPRET          ; RETURN TO SEQUENCER
```

```
                                        .SBTTL  COMCHR - GET COMMON CHARACTERISTICS AND STORE NECESSARY INFO IN UNIT PARAMETERS
005615                                  COMCHR:
                                        ;
                                        ;       GET COMMON CHARACTERISTICS AND SET UP THE ERROR RECOVERY LEVEL,
                                        ;       RETRIES, LONG TIMEOUT, AND SHORT TIMEOUT
                                        ;
005615  104203  001467                          MOV     #CR.GCR,R3      ; POINT TO GET CHARACTERISTICS DATA BLOCK
005617  021204                                  CALL    TALK            ; GET CHARACTERISTICS
005620  115002                                  TST     R2              ; SEE IF ANY ERRORS OCCURRED
005621  015630                                  BEQ     SETRLT          ; IF NOT, BRANCH
005622                                          CERROR  5,#SER8         ; REPORT SECONDARY ERROR
005622  104200  010317  001013                                                  MOV     #SER8,OUT.05
005625  104207  005615                          MOV     #COMCHR,R0      ; GET COMMON CHARACTERISTICS IS NEXT MODULE
005627  005710                                  BR      COMEXT          ; BRANCH TO EXIT
005630  104307  001571          SETRLT: MOV     ST+RETS,R0      ; GET NUMBER OF RETRIES ALLOWED
005632  110607                                  ROR     R0              ; ROTATE TO CORRECT POSITION
005633  110607                                  ROR     R0
005634  110607                                  ROR     R0
005635  110607                                  ROR     R0
005636  103207  177760                          BIC     #LBLONB,R0      ; CLEAR UNUSED BITS
005640  100657  000025                          MOV     R0,U.RTRY(R5)   ; SAVE IN UNIT PARAMETERS
005642  104307  001572                          MOV     ST+ERLEV,R0     ; GET ERROR RECOVERY LEVELS
005644  103207  177400                          BIC     #HIBYTE,R0      ; CLEAR UNUSED BITS
005646  100657  000026                          MOV     R0,U.MLEV(R5)   ; SAVE IN UNIT PARAMETERS
005650  104307  001572                          MOV     ST+ECCRSH,R0    ; GET ECC THRESHOLD
005652  110707                                  SWAB    R0              ; MOVE ECC THRESHOLD TO LOWER BYTE
005653  103207  177400                          BIC     #HIBYTE,R0      ; CLEAR UNUSED BITS
005655  100657  000027                          MOV     R0,U.ECCT(R5)   ; SAVE
005657  104307  001570                          MOV     ST+SHRTTO,R0    ; GET SHORT TIMEOUT
005661  103207  177760                          BIC     #LBLONB,R0      ; CLEAR UNUSED BITS
005663  025712                                  CALL    TO              ; CALCULATE TIMEOUT
005664  100657  000030                          MOV     R0,U.SDIS(R5)   ; SAVE SHORT TIMEOUT IN UNIT PARAMETERS
005666  104307  001571                          MOV     ST+LONGTO,R0    ; GET LONG TIMEOUT
005670  103207  177760                          BIC     #LBLONB,R0      ; CLEAR UNUSED BITS
005672  025712                                  CALL    TO              ; CALCULATE TIMEOUT
005673  100657  000031                          MOV     R0,U.SDIL(R5)   ; SAVE IN UNIT PARAMETERS
005675  104307  001571                          MOV     ST+RCTCPS,R0    ; GET NUMBER OF RCT COPIES
005677  110707                                  SWAB    R0              ; MOVE TO LOW WORD
005700  103207  177760                          BIC     #LBLONB,R0      ; CLEAR UNUSED BITS
005702  117407                                  DEC     R0              ; ADJUST FOR TEST 4 INTERNALS
005703  100657  000043                          MOV     R0,U.COPY(R5)   ; SAVE
005705  104207  005724                          MOV     #INSCHR,R0      ; INSERT SUBUNIT CHARACTERISTICS NEXT MODULE
005707  114002                                  CLR     R2              ; NO ERRORS
005710  114001                  COMEXT: CLR     R1              ; IMMIDATE CALL TO NEXT MODULE
005711  003127                                  BR      JMPRET          ; RETURN TO SEQUNCER

005712                                  TO:
                                        ;
                                        ;       CALCULATE THE TIMEOUT IN 9SEC INTERVALS (UDA SDI RECEIVE XFC TAKES
                                        ;       9 SEC)
                                        ;
005712  104201  000001                          MOV     #1,R1           ; SET UP LOG2 SHIFTER
005714  105011                  1$:     ADD     R1,R1           ; DOUBLE THE TIMEOUT VALUE
005715  117407                          DEC     R0              ; DECREMENT COUNT
005716  055714                          BNE     1$              ; IF COUNT INCOMPLETE, BRANCH
005717  115407                  2$:     INC     R0              ; INCREMENT 9 SEC COUNT
005720  107201  000011                  SUB     #9.,R1          ; SUBTRACT 9 SEC FROM TIMEOUT
```

```
     57 005722  035717                    BPL     2$              ; IF MORE TIME TO GO, BRANCH
     58 005723  000000                    RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                .SBTTL  INSCHR - GET ALL SUBUNITS' PARAMETERS (CHARACTERISTICS), ERROR CHECK AND STORE
005724                          INSCHR:
                                ;
                                ;       INSCHR WILL GET THE SUBUNIT CHARACTERISTICS AND INITILIZE THE
                                ;       SUBUNIT PARAMETERS
                                ;
005724                                  PUSH    R4              ; SAVE R4 (SUBUNIT) POINTER
005724  100464                                                                    MOV R4,-(SP)
005725  104204  000001                 MOV     #U.SUBP,R4      ; R4 POINTS TO SUBUNIT POINTERS
005727  105054                         ADD     R5,R4           ; R4 POINTS TO SUBUNIT POINTERS
005730  114007                         CLR     R0              ; R2 CONTAINS SUBUNIT POINTER INDEX
005731  114000  004542                 CLR     USED            ; FLAG AS ALL SUBUNITS UNUSED
005733  104241                 7$:     MOV     (R4)+,R1        ; R1 POINTS TO SUBUNIT
005734  076076                         BMI     11$             ; IF SUBUNIT NOT TO BE TESTED, BRANCH
005735  104200  177777  004542         MOV     #-1,USED        ; FLAG AS SUBUNIT USED
005740  104612  000005                 MOV     S.SCHR(R1),R2   ; GET CHARACTERISTICS POINTER
005742  056076                         BNE     11$             ; IF ALLREADY DEFINED, BRANCH
005743  114002                         CLR     R2              ; ZERO R0
005744  100612  000007                 MOV     R2,S.MEGW(R1)   ; ZERO MEGABIT COUNT
005746  100612  000006                 MOV     R2,S.MEGR(R1)   ; ZERO MEGABIT COUNT
005750                                  PUSH    R0              ; SAVE R0
005750  100467                                                                    MOV R0,-(SP)
005751  027321                         CALL    SMASK           ; COMPUTE MASK FOR SUBUNIT CHAR REQ
005752  104070  001554                 MOV     R0,SUBUNT       ; MOVE TO SDI SEND BUFFER
005754  104203  001474                 MOV     #CR.SCR,R3      ; POINT TO SDI PARAMETER BUFFER
005756                                  PUSH    R1              ; SAVE R1
005756  100461                                                                    MOV R1,-(SP)
005757  021204                         CALL    TALK            ; SEND AND RECEIVE SDI INTERCHANGE
005760                                  POP     R1              ; RESTORE R1
005760  104261                                                                    MOV (SP)+,R1
005761  115002                         TST     R2              ; SEE IF AN ERROR OCCURRED
005762  056120                         BNE     12$             ; IF SO, BRANCH
005763  104617  000000                 MOV     S.PARM(R1),R0   ; GET SUBUNIT PARAMETERS
005765  114003                         CLR     R3              ; NO RBN'S PER TRACK (FOR LBN AREA)
005766  102207  020000                 BIT     #DCYLS,R0       ; SEE IF DIAGNOSTIC CYLINDERS ARE USED
005770  015775                         BEQ     5$              ; IF NOT, BRANCH
005771  104303  001574                 MOV     ST+RBNTRK,R3    ; GET RBN'S PER TRACK
005773  103203  177600                 BIC     #HIBYTE!200,R3  ; CLEAR UNUSED BITS
005775  105303  001601         5$:     ADD     ST+LBNTRK,R3    ; ADD LBNS PER TRACK (TO ZERO IF LBN AREA)
005777  103203  177400                 BIC     #HIBYTE,R3      ; CLEAR UNUSED BITS
006001  104030  004543                 MOV     R3,SECTRK       ; SAVE IN SECTORS PER TRACK
006003  100613  000004                 MOV     R3,S.TRKL(R1)   ; SAVE IN TRACK LENGTH
006005  026733                         CALL    COMPSC          ; COMPUTE SECTORS/GROUP AND SECTORS/CYL
006006  026766                         CALL    CLCMAX          ; CALCULATE MAXIMUM DBN NUMBER (IN CASE NEEDED)
006007  104617  000000                 MOV     S.PARM(R1),R0   ; GET SUBUNIT PARAMETERS
006011  102207  000040                 BIT     #BEUSED,R0      ; SEE IF BEGIN/END SETS ARE USED
006013  016017                         BEQ     21$             ; IF NOT, BRANCH
006014  102207  000200                 BIT     #ONLYCL,R0      ; SEE IF WE ALLREADY HAVE BEGIN/END SETS
006016  016022                         BEQ     8$              ; IF SO, BRANCH
006017  026131                 21$:    CALL    CHKCYL          ; CONVERT THE CYLS TO BN'S
006020  115002                         TST     R2              ; SEE IF AN ERROR OCCURRED
006021  056123                         BNE     3$              ; IF SO, BRANCH
006022  026311                 8$:     CALL    CHKBES          ; CHECK THE BEGIN/END SETS FOR ERRORS
006023  115002                         TST     R2              ; SEE IF AN ERROR OCCURRED
006024  056123                         BNE     3$              ; IF SO, BRANCH
006025  026571                         CALL    CHKBB           ; CHECK THE BAD BLOCKS FOR ERRORS
006026  115002                         TST     R2              ; SEE IF AN ERROR WAS DETECTED
```

```
                                        BNE     3$              ; IF SO, BRANCH
                                        MOV     S.PARM(R1),R0   ; GET SUBUNIT PARAMETERS
                                        BIT     #BEUSED,R0      ; SEE IF BEGIN/END SETS ARE USED
                                        BNE     22$             ; IF SO, BRANCH
                                        CALL    CHKTG           ; DO ALL TRACK/GROUP CALCULATIONS
                                        TST     R2              ; SEE IF AN ERROR OCCURRED
                                        BNE     3$              ; IF SO, BRANCH
                                22$:    MOV     #ST,R3          ; R3 POINTS TO SUBUNIT CHARACTERISTCS
                                        MOV     #19.,R0         ; MOVE WORD COUNT TO R0
                                        CALL    GETMEM          ;   T SOME MEMORY
                                        MOV     R0,S.SCHR(R1)   ; SIORE POINTER TO IN SUBUNIT PARAMETERS
                                        PUSH    R1              ; SAVE R1
006047  100461                                                                  MOV R1,-(SP)
                                        MOV     #19.,R1         ; MOVE WORD COUNT TO R1
                                9$:     MOV     (R3)+,R2        ; MOVE ONE WORD OF S CHAR TO R0
                                        MOV     R2,(R0)+        ; MOVE TO SUBUNIT CHAR AREA
                                        DEC     R1              ; DECREMENT WORD COUNT
                                        BNE     9$              ; IF COUNT UNEXPIRED, BRANCH
                                        POP     R               ; RESTORE R1
006056  104261                                                                  MOV (SP)+,R1
                                        MOV     S.SCHR(R1),R0   ; R0 POINTS TO SUBUNIT CHAR
                                        MOV     LBNCYL(R0),R2   ; R2 CONTAINS LO LBN CYLS
                                        ADD     XBNCYL(R0),R2   ; ADD LO XBN CYLS TO R2
                                        MOV     R2,S.SDCL(R1)   ; MOVE TO LO STARTING DIAG CYL
                                        MOV     LBNCYL+1(R0),R2 ; GET HI ORDER LBN CYLS
                                        BCC     10$             ; IF NO CARRY, BRANCH
                                        INC     R2              ; PROPOGATE CARRY
                                10$:    MOV     R2,S.SDCL+1(R1) ; STORE HI STARTING DIAG CYL
                                        POP     R0              ; RESTORE R0
006075  104267                                                                  MOV (SP)+,R0
                                11$:    INC     R0              ; INCREMENT SUBUNIT COUNT
                                        CMP     #3,R0           ; SEE IF ALL SUBUNITS HANDLED
                                        BPL     7$              ; IF NOT, BRANCH
                                        MOV     #INITD,R0       ; INITILIZE ALL DRIVE PARAMETERS NEXT
                                        CLR     R1              ; MAKE R1 ZERO (IMMIDATE CALL)
                                        CLR     R2              ; FLAG AS NO ERRORS
                                        TST     USED            ; SEE IF ANY SUBUNITS USED
                                        BNE     6$              ; IF SO, BRANCH
                                        MOV     U.PARM(R5),R1   ; GET UNIT PARAMETERS
                                        BIS     #DROP,R1        ; FLAG ENTIRE UNIT AS DROPPED
                                        MOV     R1,U.PARM(R5)   ; SAVE PARAMETERS
                                        BR      6$              ; BRANCH
                                12$:    CERROR  5,#SER5         ; FLAG SECONDARY ERROR
006120  104200  010220  001013                                                  MOV     #SER5,OUT.05
                                3$:     POP     R0              ; ADJUST STACK
006123  104267                                                                  MOV (SP)+,R0
                                4$:     MOV     #INSCHR,R0      ; INSCHR NEXT ROUTINE CALLED
                                        CLR     R1              ; IMMIDATE CALL
                                6$:     POP     R4              ; RESTORE R4
006127  104264                                                                  MOV (SP)+,R4
                                        BR      JMPRET          ; RETURN TO SEQUENCER
```

```
                                .SBTTL  CHKCYL - CHECK VALIDITY OF STARTING AND ENDING CYLS, CONVERT TO BEGIN/END SET
006131                          CHKCYL:
                                ;
                                ;       CALCULATE THE STARTING AND ENDING BN'S FROM THE GIVEN STARTING
                                ;       AND ENDING CYLS
                                ;
006131                          PUSH    <R4,R5>         ; SAVE R4, R5
006131  100464                                                          MOV R4,-(SP)
006132  100465                                                          MOV R5,-(SP)
006133  104614  000013         MOV     S.BESS+2(R1),R4 ; R4 HAS LO STARTING CYL
006135  104615  000014         MOV     S.BESS+3(R1),R5 ; R5 HAS HI STARTING CYL
006137  103205  100000         BIC     #100000,R5      ; CLEAR EOL FLAG (IF ANY)
006141  026237                 CALL    CYLBN           ; CALCULATE STARTING BN ON THAT CYL
006142  115002                 TST     R2              ; SEE IF ANY ERROR OCCURRED
006143  056234                 BNE     10$             ; IF SO, BRANCH
006144  100617  000013         MOV     R0,S.BESS+2(R1) ; SAVE LO ORDER STARTING BN
006146  101203  100000         BIS     #100000,R3      ; SET END-OF-LIST FLAG
006150  100613  000014         MOV     R3,S.BESS+3(R1) ; SAVE HI ORDER STARTING BN
006152  104615  000012         MOV     S.BESS+1(R1),R5 ; GET HI ORDER ENDING CYL
006154  106205  177777         CMP     #-1,R5          ; SEE IF SHOULD DEFAULT TO HIGHEST LBN
006156  016201                 BEQ     4$              ; IF SO, BRANCH
006157  104614  000011         MOV     S.BESS(R1),R4   ; GET LO ORDER ENDING CYL
006161  105204  000001         ADD     #1,R4           ; FIND STARTING BN OF NEXT CYL
006163  046165                 BCC     2$              ; IF NO CARRY, BRANCH
006164  115405                 INC     R5              ; PROPOGATE CARRY
006165  026237                 2$:     CALL    CYLBN   ; CACULATE STARTING BN OF NEXT CYL
006166  115002                 TST     R2              ; SEE IF ANY ERRORS
006167  056234                 BNE     10$             ; IF SO, BRANCH
006170  107207  000001         SUB     #1,R0           ; GET LAST BN OF PREVIOUS CYL
006172  046174                 BCC     3$              ; IF NO CARRY, BRANCH
006173  117403                 DEC     R3              ; PROPOGATE CARRY
006174  100617  000011         3$:     MOV     R0,S.BESS(R1)   ; SAVE LO ORDER ENDING BN
006176  100613  000012         MOV     R3,S.BESS+1(R1) ; SAVE HI ORDER ENDING BN
006200  006233                 BR      9$              ; EXIT
006201  104617  000000         4$:     MOV     S.PARM(R1),R0   ; GET SUBUNIT PARAMETERS
006203  102207  020000         BIT     #DCYLS,R0       ; SEE IF DIAGNOSTIC CYLINDERS USED
006205  056223                 BNE     6$              ; IF SO, BRANCH
006206  104307  001602         MOV     ST+LBNHST,R0    ; GET LO ORDER ENDING LBN
006210  107207  000001         SUB     #1,R0           ; LAST LBN IN LBN AREA
006212  100617  000011         MOV     R0,S.BESS(R1)   ; SAVE LO ORDER
006214  104307  001603         MOV     ST+LBNHST+1,R0  ; GET HI ORDER
006216  046220                 BCC     5$              ; IF NO CARRY, BRANCH
006217  117407                 DEC     R0              ; PROPOGATE CARRY
006220  100617  000012         5$:     MOV     R0,S.BESS+1(R1) ; SAVE HI ORDER
006222  006233                 BR      9$              ; EXIT
006223  104307  004540         6$:     MOV     MAXDBN,R0       ; GET LO ORDER MAX DBN
006225  100617  000011         MOV     R0,S.BESS(R1)   ; SAVE
006227  104307  004541         MOV     MAXDBN+1,R0     ; GET HI ORDER
006231  100617  000012         MOV     R0,S.BESS+1(R1) ; SAVE
006233  114002                 9$:     CLR     R2      ; NO ERRORS
006234                         10$:    POP     <R5,R4> ; RESTORE
006234  104265                                                          MOV (SP)+,R5
006235  104264                                                          MOV (SP)+,R4
006236  000000                 RETURN
```

```
                                        .SBTTL  CYLBN  - GIVEN A CYLINDER, CALCULATE STARTING BN ON THAT CYLINDER
006237                                  CYLBN:
                                        ;
                                        ;       TAKING THE CYL GIVEN IN R4,R5 CALCULATE THE FIRST BN ON THAT
                                        ;       CYL AND RETURN IN R2,R3.  IF BN EXCEEDS 28 BITS, REPORT
                                        ;       ERROR
                                        ;
006237  114007                          1$:     CLR     R0              ; CLEAR THE PRODUCT AREA
006240  114003                                  CLR     R3              ; CLEAR THE PRODUCT AREA
006241  115004                                  TST     R4              ; SEE IF LO ORDER IS ZERO
006242  056245                                  BNE     1$              ; IF NOT, BRANCH
006243  115005                                  TST     R5              ; SEE IF HI ORDER IS ZERO
006244  016307                                  BEQ     5$              ; IF SO, BRANCH AND EXIT
006245  107204  000001                  2$:     SUB     #1,R4           ; ADJUST COUNT
006247  046251                                  BCC     2$              ; IF NO BORROW, BRANCH
006250  117405                                  DEC     R5              ; PROPOGATE BORROW
006251  105307  004545                  3$:     ADD     SECCYL,R0       ; ADD LO SECTORS/CYL TO LO ORDER PROD
006253  046255                                  BCC     3$              ; IF NO CARRY, BRANCH
006254  115403                                  INC     R3              ; PROPOGATE CARRY
006255  106203  000377                  4$:     CMP     #377,R3         ; SEE IF HI ORDER TOO BIG
006257  036301                                  BPL     4$              ; IF NOT, BRANCH
006260                                          DEVFTL  57              ; REPORT LBN OVERFLOW
006260  104200  006330  001012                                                  MOV     #ER57,OUT.04
006263  104200  000071  001010                                                  MOV     #57,OUT.02
006266  101200  000400  001010                                                  BIS     #FTLDEV,OUT.02
006271  104200  006271  001007                                                  MOV     #.,OUT.01
006274  104202  000014                                                          MOV     #ERRMC,R2
006276  104020  001006                                                          MOV     R2,OUT.R0
006300  006310                                  BR      6$              ; EXIT
006301  107204  000001                  5$:     SUB     #1,R4           ; DECREMENT LO ORDER COUNT
006303  046251                                  BCC     2$              ; IF INCOMPLETE, BRANCH
006304  107205  000001                          SUB     #1,R5           ; DECREMENT HI ORDER COUNT
006306  046251                                  BCC     2$              ; IF INCOMPLETE, BRANCH
006307  114002                          6$:     CLR     R2              ; NO ERRORS
006310  000000                          7$:     RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                .SBTTL  CHKBES - CHECK THE BEGIN/END SETS FOR ERRORS
006311                          CHKBES:
                                ;
                                ;       CHECK THAT THE BEGIN BLOCK <= END BLOCK IN EACH BEGIN/END SET,
                                ;       THAT THE END BLOCK < THE BEGIN BLOCK OF THE NEXT BEGIN/END SET,
                                ;       AND THAT THE LAST END BLOCK IS A VALID LBN OR DBN FOR THAT DRIVE.
                                ;
006311                          ;       PUSH    <R4,R5>
006311  100464                                                                  MOV R4,-(SP)
006312  100465                                                                  MOV R5,-(SP)
006313  104203  000011          ;       MOV     #S.BESS,R3      ; R3 WILL POINT TO ENDING BLOCK NUMBER
006315  105013                  ;       ADD     R1,R3           ; R3 POINTS TO ENDING BN
006316  104202  000013          ;       MOV     #S.BESS+2,R2    ; R2 WILL POINT TO STARTING BN
006320  105012                  ;       ADD     R1,R2           ; R2 POINTS TO STARTING BN
006321  025441          1$:     ;       CALL    CBB2            ; COMPARE
006322  046344                  ;       BCC     2$              ; IF NO ERROR, BRANCH
006323                          ;       DEVFTL  50              ; REPORT BEGIN >= ENDING BN
006323  104200  005443  001012                                                  MOV     #ER50,OUT.04
006326  104200  000062  001010                                                  MOV     #50,OUT.02
006331  101200  000400  001010                                                  BIS     #FTLDEV,OUT.02
006334  104200  006334  001007                                                  MOV     #.,OUT.01
006337  104202  000014                                                          MOV     #ERRMC,R2
006341  104020  001006                                                          MOV     R2,OUT.RQ
006343  006566                  ;       BR      7$              ; BRANCH TO ERROR EXIT
006344  104625  000001  2$:     ;       MOV     1(R2),R5        ; SEE IF THE END-OF-LIST HAS BEEN FOUND
006346  076400                  ;       BMI     4$              ; IF SO, BRANCH
006347  105202  000004          ;       ADD     #4,R2           ; POINT TO NEXT BEGIN SET
006351  025441                  ;       CALL    CBB2            ; COMPARE
006352                          ;       BCS     3$              ; BRANCH
006352  046354                                                                  BCC     .+2
006353  006375                                                                  BR      3$
006354                          ;       DEVFTL  51              ; STARTING BN <= PREVIOUS ENDING BN
006354  104200  005510  001012                                                  MOV     #ER51,OUT.04
006357  104200  000063  001010                                                  MOV     #51,OUT.02
006362  101200  000400  001010                                                  BIS     #FTLDEV,OUT.02
006365  104200  006365  001007                                                  MOV     #.,OUT.01
006370  104202  000014                                                          MOV     #ERRMC,R2
006372  104020  001006                                                          MOV     R2,OUT.RQ
006374  006566                  ;       BR      7$              ; ERROR EXIT
006375  105203  000004  3$:     ;       ADD     #4,R3           ; R3 NOW POINTS TO NEXT END SET
006377  006321                  ;       BR      1$              ; LOOP
006400  104612  000000  4$:     ;       MOV     S.PARM(R1),R2   ; GET SUBUNIT CHARACTERISTICS
006402  102202  020000          ;       BIT     #DCYLS,R2       ; SEE IF DIAGNOSTIC CYLINDERS ARE IN USE
006404  056500                  ;       BNE     5$              ; IF SO, BRANCH
006405  104202  001602          ;       MOV     #ST+LBNHST,R2   ; R2 POINTS TO LBNS IN HOST AREA
006407  025441                  ;       CALL    CBB2            ; COMPARE
006410                          ;       BCS     6$              ; EXIT
006410  046412                                                                  BCC     .+2
006411  006565                                                                  BR      6$
006412  104612  000000                  ;       MOV     S.PARM(R1),R2   ; GET SUBUNIT CHARACTERISTICS
006414  102202  000040                  ;       BIT     #BEUSED,R2      ; SEE IF TRACKS/GROUPS IN USE
006416  016422                          ;       BEQ     10$             ; IF SO, BRANCH
006417  102202  000200                  ;       BIT     #ONLYCL,R2      ; SEE IF B/E SET COMPUTED BY CYLINDERS
006421  016443                          ;       BEQ     11$             ; IF NOT, BRANCH
006422                          10$:    ;       DEVFTL  61              ; REPORT ERROR
006422  104200  006540  001012                                                  MOV     #ER61,OUT.04
006425  104200  000075  001010                                                  MOV     #61,OUT.02
```

```
                006430  101200  000400  001010                                          BIS     #FTLDEV,OUT.02
                006433  104200  006433  001007                                          MOV     #.,OUT.01
                006436  104202  000014                                                  MOV     #ERRMC,R2
                006440  104020  001006                                                  MOV     R2,OUT.RQ
     38         006442  006566                  BR      7$              ; EXIT
     39         006443  107200  000001  001602 11$:    SUB     #1,ST+LBNHST    ; DECREMENT LO LBN COUNT BY 1
     40         006446  046451                  BCC     9$              ; IF NO BORROW, BRANCH
     41         006447  117400  001603          DEC     ST+LBNHST+1     ; PROPOGATE BORROW
     42         006451                  9$:     DEVFTL  52,<ST+LBNHST,ST+LBNHST+1>      ; LAST ENDING BN > MAX
                006451  104200  005575  001012                                          MOV     #ER52,OUT.04
                006454  104300  001602  001013                                          MOV     ST+LBNHST,OUT.05
                006457  104300  001603  001014                                          MOV     ST+LBNHST+1,OUT.06
                006462  104200  000064  001010                                          MOV     #52,OUT.02
                006465  101200  000400  001010                                          BIS     #FTLDEV,OUT.02
                006470  104200  006470  001007                                          MOV     #.,OUT.01
                006473  104202  000014                                                  MOV     #ERRMC,R2
                006475  104020  001006                                                  MOV     R2,OUT.RQ
     43         006477  006566                  BR      7$              ; ERROR EXIT
     44         006500  104032          5$:     MOV     R3,R2           ; R2 POINTS TO LAST END SET
     45         006501  104203  004540          MOV     #MAXDBN,R3      ; R3 POINTS TO MAXIMUM DBN
     46         006503  025441                  CALL    CBB2            ; COMPARE
     47         006504  046565                  BCC     6$              ; EXIT
     48         006505  104612  000000          MOV     S.PARM(R1),R2   ; GET SUBUNIT CHARACTERISTICS
     49         006507  102202  000040          BIT     #BEUSED,R2      ; SEE IF TRACKS/GROUPS IN USE
     50         006511  016515                  BEQ     12$             ; IF SO, BRANCH
     51         006512  102202  000200          BIT     #ONLYCL,R2      ; SEE IF B/E SET COMPUTED BY CYLINDERS
     52         006514  016536                  BEQ     13$             ; IF NOT, BRANCH
     53         006515                  12$:    DEVFTL  62              ; REPORT ERROR
                006515  104200  006766  001012                                          MOV     #ER62,OUT.04
                006520  104200  000076  001010                                          MOV     #62,OUT.02
                006523  101200  000400  001010                                          BIS     #FTLDEV,OUT.02
                006526  104200  006526  001007                                          MOV     #.,OUT.01
                006531  104202  000014                                                  MOV     #ERRMC,R2
                006533  10402C  001006                                                  MOV     R2,OUT.RQ
     54         006535  006566                  BR      7$              ; EXIT
     55         006536                  13$:    DEVFTL  52,<MAXDBN,MAXDBN+1>    ; LAST ENDING BN > MAX
                006536  104200  005575  001012                                          MOV     #ER52,OUT.04
                006541  104300  004540  001013                                          MOV     MAXDBN,OUT.05
                006544  104300  004541  001014                                          MOV     MAXDBN+1,OUT.06
                006547  104200  000064  001010                                          MOV     #52,OUT.02
                006552  101200  000400  001010                                          BIS     #FTLDEV,OUT.02
                006555  104200  006555  001007                                          MOV     #.,OUT.01
                006560  104202  000014                                                  MOV     #ERRMC,R2
                006562  104020  001006                                                  MOV     R2,OUT.RQ
     56         006564  006566                  BR      7$              ; ERROR EXIT
     57         006565  114002          6$:     CLR     R2              ; FLAG AS NO ERROR
     58         006566          7$:     POP     <R5,R4>         ; RESTORE REGISTERS
                006566  104265                                                                  MOV (SP)+,R5
                006567  104264                                                                  MOV (SP)+,R4
     59         006570  000000                  RETURN                  ; RETURN TO INSCHR
```

```
                                        .SBTTL  CHKBB  - CHECK THE BAD BLOCKS FOR ERRORS
006571                                  CHKBB:
                                        ;
                                        ;       CHECK THE BAD BLOCKS TO ASSURE THEY DON'T EXCEED THE MAXIMUM BLOCK
                                        ;       NUMBER
                                        ;
006571                                          PUSH    <R4,R5>         ; SAVE THE REGISTERS
006571  100464                                                                          MOV R4,-(SP)
006572  100465                                                                          MOV R5,-(SP)
006573  104613  000010                          MOV     S.BADP(R5),R3   ; GET POINTER TO BAD BLOCKS
006575  016727                                  BEQ     5$              ; IF NO BAD BLOCKS, BRANCH
006576  104632  000001          1$:             MOV     1(R3),R2        ; IS THIS THE LAST BAD BLOCK?
006600  076632                                  BMI     2$              ; IF SO, BRANCH
006601  104032                                  MOV     R3,R2           ; R2 WILL POINT TO NEXT BAD BLOCK
006602  105202  000002                          ADD     #2,R2           ; R2 POINTS TO NEXT BAD BLOCK
006604  025441                                  CALL    CBB2            ; COMPARE
006605  046611                                  BCC     7$              ; IF ERROR, BRANCH
006606  105203  000002                          ADD     #2,R3           ; POINT TO NEXT BAD BLOCK
006610  006576                                  BR      1$              ; LOOP
006611                          7$:             DEVFTL  53              ; DUPLICATE BAD BLOCKS
006611  104200  005660  001012                                                  MOV     #ER53,OUT.04
006614  104200  000065  001010                                                  MOV     #53,OUT.02
006617  101200  000400  001010                                                  BIS     #FTLDEV,OUT.02
006622  104200  006622  001007                                                  MOV     #.,OUT.01
006625  104202  000014                                                          MOV     #ERRMC,R2
006627  104020  001006                                                          MOV     R2,OUT.RQ
006631  006730                                  BR      6$              ; BRANCH
006632  104617  000000          2$:             MOV     S.PARM(R1),R0   ; GET SUBUNIT PARAMETERS
006634  102207  020000                          BIT     #DCYLS,R0       ; SEE IF DIAGNOSTIC CYLINDERS IN USE
006636  056673                                  BNE     3$              ; IF SO, BRANCH
006637  104202  001602                          MOV     #ST+LBNHST,R2   ; R2 POINTS TO NUMBER OF LBNS IN HOST AREA
006641  025441                                  CALL    CBB2            ; COMPARE
006642  046644                                  BCC     8$              ; IF ERROR, BRANCH
006643  006727                                  BR      5$              ; EXIT WITH NO ERROR
006644                          8$:             DEVFTL  54,<ST+LBNHST,ST+LBNHST+1>      ; BAD BLOCK > MAXIMUM
006644  104200  005674  001012                                                  MOV     #ER54,OUT.04
006647  104300  001602  001013                                                  MOV     ST+LBNHST,OUT.05
006652  104300  001603  001014                                                  MOV     ST+LBNHST+1,OUT.06
006655  104200  000066  001010                                                  MOV     #54,OUT.02
006660  101200  000400  001010                                                  BIS     #FTLDEV,OUT.02
006663  104200  006663  001007                                                  MOV     #.,OUT.01
006666  104202  000014                                                          MOV     #ERRMC,R2
006670  104020  001006                                                          MOV     R2,OUT.RQ
006672  006730                                  BR      6$              ; ERROR EXIT
006673  104032                          3$:     MOV     R3,R2           ; R2 POINTS TO LAS BAD BLOCK
006674  104203  004540                          MOV     #MAXDBN,R3      ; R3 POINTS TO MAXIMUM DBN
006676  025441                                  CALL    CBB2            ; COMPARE
006677  046727                                  BCC     5$              ; IF NO ERROR, BRANCH
006700                          4$:             DEVFTL  54,<MAXDBN,MAXDBN+1>    ; BAD BLOCK > MAXIMUM
006700  104200  005674  001012                                                  MOV     #ER54,OUT.04
006703  104300  004540  001013                                                  MOV     MAXDBN,OUT.05
006706  104300  004541  001014                                                  MOV     MAXDBN+1,OUT.06
006711  104200  000066  001010                                                  MOV     #54,OUT.02
006714  101200  000400  001010                                                  BIS     #FTLDEV,OUT.02
006717  104200  006717  001007                                                  MOV     #.,OUT.01
006722  104202  000014                                                          MOV     #ERRMC,R2
006724  104020  001006                                                          MOV     R2,OUT.RQ
```

```
     34 006726  006730                      BR      6$              ; BRANCH TO EXIT
     35 006727  114002            5$:       CLR     R2              ; FLAG AS NO ERRORS
     36 006730                    6$:       POP     <R5,R4>         ; RESTORE THE REGISTERES
        006730  104265                                                              MOV (SP)+,R5
        006731  104264                                                              MOV (SP)+,R4
     37 006732  000000                      RETURN                  ; RETURN TO INSCHR
```

```
                                .SBTTL  COMPSC - COMPUTE SECTORS/GROUP AND SECTORS/CYLINDER
006733                          COMPSC:
                                ;
                                ;       COMPUTE HOW MANY SECTORS/GROUP AND SECTORS/CYLINDER
                                ;       GIVEN HOW MANY SECTORS/TRACK IN SECTRK
                                ;
006733                          PUSH    <R0,R2>         ; SAVE REGISTERS
006733  100467                                                          MOV R0,-(SP)
006734  100462                                                          MOV R2,-(SP)
006735  114007                  CLR     R0              ; CLEAR RUNNING TOTAL
006736  104302  001573          MOV     ST+TRKGRP,R2    ; GET NUMBER OF TRACKS PER GROUP
006740  103202  177400          BIC     #HIBYTE,R2      ; CLEAR UNUSED BITS
006742  105307  004543  1$:     ADD     SECTRK,R0       ; ADD NUMBER OF SECTORS/TRACK TO RUNNING TOTAL
006744  117402                  DEC     R2              ; DECREMENT COUNT
006745  056742                  BNE     1$              ; IF COUNT INCOMPLETE, BRANCH
006746  104070  004544          MOV     R0,SECGRP       ; SAVE SECTORS/GROUP
006750  114007                  CLR     R0              ; CLEAR RUNNING TOTAL
006751  104302  001572          MOV     ST+GRPCYL,R2    ; GET GROUPS/CYLINDER
006753  103202  177400          BIC     #HIBYTE,R2      ; CLEAR UNUSED BITS
006755  105307  004544  2$:     ADD     SECGRP,R0       ; ADD SECTORS/GROUP TO RUNNING TOTAL
006757  117402                  DEC     R2              ; DECREMENT COUNT
006760  056755                  BNE     2$              ; IF COUNT INCOMPLETE, BRANCH
006761  104070  004545          MOV     R0,SECCYL       ; STORE SECTORS/CYLINDER
006763                          POP     <R2,R0>         ; RESTORE REGISTERS
006763  104262                                                          MOV (SP)+,R2
006764  104267                                                          MOV (SP)+,R0
006765  000000                  RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                .SBTTL  CLCMAX - CALCULATE THE MAXIMUM WRITEABLE DBN
006766                          CLCMAX:
                                ;
                                ;       CALCULATE THE MAXIMUM DBN NUMBER ALLOWED, AND PLACE IT IN MAXDBN
                                ;       FOR BEGIN/END SET GENERATION AND TESTING (BAD BLOCKS TOO)
                                ;
006766                          PUSH    <R0,R1,R2>      ; SAVE REGISTERS
006766  100467                                                          MOV R0,-(SP)
006767  100461                                                          MOV R1,-(SP)
006770  100462                                                          MOV R2,-(SP)
006771  114007                  CLR     R0              ; CLEAR PRODUCT AREA
006772  114001                  CLR     R1              ; CLEAR PRODUCT AREA
006773  104302  001612          MOV     ST+DBNCYL,R2    ; GET DIAG CYL COUNT
006775  110702                  SWAB    R2              ; MOVE TO LO ORDER BYTE
006776  103202  177400          BIC     #HIBYTE,R2      ; CLEAR UNUSED BITS
007000  105307  004545  1$:     ADD     SECCYL,R0       ; ADD SECTORS/CYL TO TOTAL
007002  047004                  BCC     2$              ; IF NO CARRY, BRANCH
007003  115401                  INC     R1              ; INCREMENT HI WORD
007004  117402  2$:     DEC     R2              ; DECREMENT COUNT
007005  057000                  BNE     1$              ; IF NO CARRY, BRANCH
007006  104302  001612          MOV     ST+DBNCYL,R2    ; GET NUMBER OF READ ONLY GROUPS
007010  103202  177400          BIC     #HIBYTE,R2      ; CLEAR UNUSED BITS
007012  107307  004544  3$:     SUB     SECGRP,R0       ; SUBTRACT NUMBER OF SECTORS/GROUP
007014  047016                  BCC     4$              ; IF NO BORROW, BRANCH
007015  117401                  DEC     R1              ; PROPOGATE BORROW
007016  117402  4$:     DEC     R2              ; DECREMENT COUNT
007017  057012                  BNE     3$              ; IF INCOMPLETE, BRANCH
007020  107207  000001          SUB     #1,R0           ; ADJUST FOR DBNS STARTING AT ZERO
007022  047024                  BCC     5$              ; IF NO BORROW, BRANCH
007023  117401                  DEC     R1              ; PROPOGATE BORROW
007024  104070  004540  5$:     MOV     R0,MAXDBN       ; SAVE LO ORDER WORD
007026  104010  004541          MOV     R1,MAXDBN+1     ; SAVE HI ORDER WORD
007030                          POP     <R2,R1,R0>      ; RESTORE REGISTERS
007030  104262                                                          MOV (SP)+,R2
007031  104261                                                          MOV (SP)+,R1
007032  104267                                                          MOV (SP)+,R0
007033  000000                  RETURN                  ; RETURN TO INSCHR
```

```
                                        .SBTTL  CHKTG  - CHECK THE TRACK/GROUPS FOR ERRORS, AND CONVERT TO BN'S
007034                                  CHKTG:
                                        ;
                                        ;       CONVERT THE TRACKS/GROUPS TO SECTOR COUNTS, AND FIND THE MAXIMUM
                                        ;       NUMBER OF 'LOOPS' THAT SETUP CAN RUN THROUGH TO TEST ALL
                                        ;       TRACKS ON THE TESTED CYLINDERS ONLY
                                        ;
                                        ;
                                        ;       FIRST COPY THE TRACKS/GROUPS TO A TEMP LOCATION
                                        ;
007034                                  PUSH    <R4,R5>                 ; SAVE REGISTERS
007034  100464                                                                  MOV R4,-(SP)
007035  100465                                                                  MOV R5,-(SP)
007036  104207  004546                 MOV     #TGS,R0                 ; POINT TO TEMPORARY STORAGE LOCATION
007040  104202  000015                 MOV     #S.TGSS,R2              ; R2 WILL POINT TO T/G LIST
007042  105012                         ADD     R1,R2                   ; R2 POINTS TO T/G LIST
007043  104225                 1$:     MOV     (R2)+,R5                ; GET WORD
007044  100275                         MOV     R5,(R0)+                ; SAVE IN TEMP LOCATION
007045  037043                         BPL     1$                      ; COPY ENTIRE LIST
                                        ;
                                        ;       NOW MOVE THE STARTING CYLINDER TO THE INITIAL OFFSET AREA
                                        ;
007046  104612  000014                 MOV     S.TGOF+1(R1),R2         ; GET HI ORDER STARTING CYL
007050  103202  170000                 BIC     #^CHBHINB,R2            ; CLEAR UNUSED BITS
007052  100612  000014                 MOV     R2,S.TGOF+1(R1)         ; SAVE
                                        ;
                                        ;       CHECK THAT THE MAXIMUM TRACK/GROUP IS VALID
                                        ;
007054  104617  000000                 MOV     S.PARM(R1),R0           ; GET SUBUNIT PARAMETERS
007056  102207  000020                 BIT     #TRACKS,R0              ; SEE IF PROCESSING GROUPS OR TRACKS
007060  057064                         BNE     2$                      ; IF TRACKS, BRANCH
007061  104307  001572                 MOV     ST+GRPCYL,R0            ; GET GROUPS PER CYLINDER
007063  007066                         BR      3$                      ; BRANCH
007064  104307  001573         2$:     MOV     ST+TRKGRP,R0            ; GET TRACKS PER GROUP
007066  103207  177400         3$:     BIC     #HIBYTE,R0              ; CLEAR UNUSED BITS
007070  104202  004546                 MOV     #TGS,R2                 ; POINT TO TRACK/GROUP LIST
007072  104223                 4$:     MOV     (R2)+,R3                ; SEE IF THIS WORD IS END-OF-LIST
007073  037072                         BPL     4$                      ; IF NOT, BRANCH
007074  103203  177400                 BIC     #HIBYTE,R3              ; CLEAR EOL FLAG
007076  106037                         CMP     R3,R0                   ; SEE IF WITHIN LIMITS
007077  077126                         BMI     5$                      ; IF SO, BRANCH
007100  117407                         DEC     R0                      ; ADJUST T/G MAX FOR 0-N
007101                                 DEVFTL  58,<R3,R0>              ; REPORT ERROR
007101  104200  006423  001012                 MOV     #ER58,OUT.04
007104  104030  001013                         MOV     R3,OUT.05
007106  104070  001014                         MOV     R0,OUT.06
007110  104200  000072  001010                 MOV     #58,OUT.02
007113  101200  000400  001010                 BIS     #FTLDEV,OUT.02
007116  104200  007116  001007                 MOV     #.,OUT.01
007121  104202  000014                         MOV     #ERRMC,R2
007123  104020  001006                         MOV     R2,OUT.RQ
007125  007277                         BR      17$                     ; EXIT
```

```
                                        ;
                                        ;       NOW COMPUTE THE NUMBER OF SECTORS FROM THE START OF THE CYLINDER/GROUP
                                        ;       TO THE FIRST TRACK/GROUP TO TEST
                                        ;
007126  104207  004546         5$:     MOV     #TGS,R0                 ; R0 POINTS TO TRACK/GROUP LIST
007130  104203  000013                  MOV     #S.TGOF,R3              ; R3 WILL POINT TO T/G INITIAL OFFSET
007132  105013                          ADD     R1,R3                   ; R3 POINTS TO T/G INITIAL OFFSET
007133  104172                          MOV     (R0),R2                 ; GET FIRST T/G
007134  103202  177400                  BIC     #HIBYTE,R2              ; CLEAR UNUSED BITS
007136  027302                          CALL    COMPDP                  ; COMPUTE SECTORS FROM CYL TO INIT T/G
007137  105135                          ADD     (R3),R5                 ; ADD TO INITIAL OFFSET
007140  100235                          MOV     R5,(R3)+                ; MOVE BACK
007141  047145                          BCC     6$                      ; IF NO CARRY, BRANCH
007142  104135                          MOV     (R3),R5                 ; GET HI ORDER WORD
007143  105405                          INC     R5                      ; PROPOGATE CARRY
007144  100135                          MOV     R5,(R3)                 ; SAVE
007145  115403                  6$:     INC     R3                      ; POINT TO NEXT AREA
                                        ;
                                        ;       NOW COMPUE HOW MANY SECTORS FROM THE LAST TRACK/GROUP TO TEST TO THE
                                        ;       NEXT TRACK/GROUP TO TEST
                                        ;
007146  104174                  7$:     MOV     (R0),R4                 ; GET TRACK/GROUP
007147  077203                          BMI     8$                      ; IF EOL, BRANCH
007150  104672  000001                  MOV     1(R0),R2                ; GET NEXT TRACK
007152  103202  177400                  BIC     #HIBYTE,R2              ; CLEAR EOL FLAG, IF ANY
007154  107272                          SUB     (R0)+,R2                ; GET HOW MANY T/G'S BETWEEN LAST/NEXT
007155  017157                          BEQ     20$                     ; IF ZERO, ERROR
007156  037200                          BPL     18$                     ; SHOULD BE AT LEAST 1 T/G BETWEEN
007157                          20$:    DEVFTL  59                      ; REPORT ERROR
                                                        MOV     #ER59,OUT.04
                                                        MOV     #59,OUT.02
                                                        BIS     #FTLDEV,OUT.02
                                                        MOV     #.,OUT.01
                                                        MOV     #ERRMC,R2
                                                        MOV     R2,OUT.RQ
007157  104200  006515  001012
007162  104200  000073  001010
007165  101200  000400  001010
007170  104200  007170  001007
007173  104202  000014
007175  104020  001006
007177  007277                          BR      17$                     ; EXIT
007200  027302                  18$:    CALL    COMPDP                  ; COMPUTE HOW MANY SECTORS TO NEXT T/G
007201  100235                          MOV     R5,(R3)+                ; STORE THE OFFSET
007202  007146                          BR      7$                      ; LOOP
                                        ;
                                        ;       NOW COMPUTE HOW MANY SECTORS FROM THE LAST T/G TO TEST TO THE FIRST
                                        ;
007203  104615  000000          8$:     MOV     S.PARM(R1),R5           ; GET SUBUNIT PARAMETERS
007205  102205  000020                  BIT     #TRACKS,R5              ; SEE IF USING GROUPS OR TRACKS
007207  057213                          BNE     9$                      ; IF TRACKS, BRANCH
007210  104302  001572                  MOV     ST+GRPCYL,R2            ; GET GROUPS/CYL
007212  007215                          BR      10$                     ; BRANCH
007213  104302  001573          9$:     MOV     ST+TRKGRP,R2            ; GET TRACKS/GROUP
007215  105302  004546          10$:    ADD     TGS,R2                  ; ADD STARTING T/G
007217  103202  177400                  BIC     #HIBYTE,R2              ; CLEAR UNUSED BITS
007221  103204  177400                  BIC     #HIBYTE,R4              ; CLEAR UNUSED BITS
007223  107042                          SUB     R4,R2                   ; FIND HOW MANY T/G'S TO SKIP
007224  027302                          CALL    COMPDP                  ; COMPUTE HOW MANY SECTORS
007225  100235                          MOV     R5,(R3)+                ; STORE IN LIST
007226  114005                          CLR     R5                      ; SET UP FOR MARKING EOL
007227  100235                          MOV     R5,(R3)+                ; STORE EOL
                                        ;
```

```
                                ;       NOW SEE HOW MANY TIMES THE SETUP ROUTINE CAN GO THROUGH THE LIST
                                ;       WITHOUT EXCEEDING THE MAXIMUM SECTOR NUMBER
                                ;
007230  104207  000011          MOV     #S.BESS,R0              ; R0 WILL POINT TO THE MAX SEC NUMBER
007232  105017                  ADD     R1,R0                   ; R0 POINTS TO THE MAX SECTOR NUMBER
007233  104674  000002          MOV     S.TGOF-S.BESS(R0),R4    ; R4 HAS LO ORDER INITIAL OFFSET
007235  104675  000003          MOV     S.TGOF-S.BESS+1(R0),R5  ; R5 HAS HI ORDER INITIAL OFFSET
007237                          PUSH    R1                      ; SAVE R1
007237  100461                                                          MOV R1,-(SP)
007240  114002                  CLR     R2                      ; CLEAR LO ORDER MAX COUNT
007241  114003                  CLR     R3                      ; CLEAR HI ORDER MAX COUNT
007242  104201  000004          MOV     #S.TGSS-S.BESS,R1       ; R1 WILL POINT TO SECTOR OFFSET AREA
007244  105071                  ADD     R0,R1                   ; R1 POINTS TO SECTOR OFFSET AREA
007245  100464          13$:    MOV     R4,-(SP)                ; SAVE LO ORDER TOTAL ON STACK
007246  104214                  MOV     (R1)+,R4                ; GET LO ORDER SECTOR OFFSET
007247  057254                  BNE     14$                     ; IF NOT EOL, BRANCH
007250  104071                  MOV     R0,R1                   ; R1 WILL POINT TO START OF OFFSET LIST
007251  105201  000004          ADD     #S.TGSS-S.BESS,R1       ; R1 POINTS TO START OF OFFSET LIST
007253  104214                  MOV     (R1)+,R4                ; GET NEW OFFSET
007254  105264          14$:    ADD     (SP)+,R4                ; ADD TO LOW ORDER SECTOR TOTAL
007255  047257                  BCC     15$                     ; IF NO CARRY, BRANCH
007256  115405                  INC     R5                      ; PROPOGATE CARRY
007257  106675  000001          CMP     1(R0),R5                ; SEE IF MAX EXCEEDED
007261  077272                  BMI     16$                     ; IF SO, BRANCH
007262  057265                  BNE     11$                     ; IF YOUR SURE IT'S OK, BRANCH
007263  106174                  CMP     (R0),R4                 ; SEE IF MAX EXCEEDED
007264  077272                  BMI     16$                     ; IF NOT, BRANCH
007265  105202  000001          ADD     #1,R2                   ; ADD 1 TO COUNT
007267  047245                  BCC     13$                     ; IF NO CARRY, BRANCH
007270  115403                  INC     R3                      ; PROPOGATE CARRY
007271  007245                  BR      13$                     ; BRANCH
007272                  16$:    POP     R1                      ; RESTORE R1
007272  104261                                                          MOV (SP)+,R1
007273  100172                  MOV     R2,(R0)                 ; STORE LOW ORDER MAXIMUM COUNT
007274  100673  000001          MOV     R3,1(R0)                ; STORE HI MAXIMUM COUNT
007276  114002                  CLR     R2                      ; CLEAR R2
007277                  17$:    POP     <R5,R4>                 ; RESTORE
007277  104265                                                          MOV (SP)+,R5
007300  104264                                                          MOV (SP)+,R4
007301  000000  ,               RETURN
```

COMPDP - CALCULATE SECTORS/TRACKS OR SECTORS/GROUPS

```
                                        .SBTTL  COMPDP - CALCULATE SECTORS/TRACKS OR SECTORS/GROUPS
007302                                  COMPDP:
                                        ;
                                        ;       COMPUTE HOW MANY SECTORS ARE IN THE NUMBER OF TRACKS OR GROUPS
                                        ;       PASSED IN R2
                                        ;
007302  114005                                  CLR     R5                      ; CLEAR RUNNING TOTAL
007303  104614  000000                          MOV     S.PARM(R1),R4           ; GET SUBUNIT PARAMETERS
007305  007316                                  BR      3$                      ; SEE IF IMMIDATE EXIT
007306  102204  000020                  1$:     BIT     #TRACKS,R4              ; SEE IF USING GROUPS OR TRACKS
007310  057314                                  BNE     2$                      ; IF TRACKS, BRANCH
007311  105305  004544                          ADD     SECGRP,R5               ; ADD SECTORS/GROUP TO RUNNING TOTAL
007313  007316                                  BR      3$                      ; BRANCH
007314  105305  004543                  2$:     ADD     SECTRK,R5               ; ADD SECTORS/TRACK TO RUNNING TOTAL
007316  117402                          3$:     DEC     R2                      ; DECREMENT COUNT
007317  037306                                  BPL     1$                      ; IF INCOMPLETE, BRANCH
007320  000000                                  RETURN                          ; RETURN TO CALLING PROGRAM
```

SMASK - CALCULATE THE SUBUNIT MASK

```
                                .SBTTL  SMASK  - CALCULATE THE SUBUNIT MASK
007321                          SMASK:
                                ;
                                ;       SMASK TAKES THE UNIT OFFSET (0 - 3) IN R0 AND CHANGES IT TO THE
                                ;       SUBUNIT MASK (0001 - 1000)
                                ;
007321                                  PUSH    R1              ; SAVE R1
007321  100461                                                          MOV R1,-(SP)
007322  104071                          MOV     R0,R1           ; MOVE R0 TO R1
007323  104207  000020                  MOV     #20,R0          ; SUBUNIT 0 MASK
007325  115001                          TST     R1              ; SEE IF SUBUNIT MASK SHIFTED TO CORRECT POSITION
007326  017332                SMASKL:   BEQ     SMASKX          ; IF SO, BRANCH
007327  110207                          ROL     R0              ; SHIFT MASK
007330  117401                          DEC     R1              ; DECREMENT COUNT
007331  007326                          BR      SMASKL          ; BRANCH
007332                          SMASKX: POP     R1              ; RESTORE R1
007332  104261                                                          MOV (SP)+,R1
007333  000000                          RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                        .SBTTL  INITD  - LAST ERROR CHECKING AND SETUP MODULE BEFORE TESTING STARTS
007334                                  INITD:
                                        ;
                                        ;       BRING DRIVE ONLINE, CLEAR ALL ERRORS (IF ANY) AND CHANGE THE
                                        ;       MODE TO THE REQUIRED VALUES
                                        ;
007334  104651  000033                  1$:     MOV     U.PARM(R5),R1   ; GET UNIT PARAMETERS
007336  077466                          2$:     BMI     DRVEXT          ; IF DRIVE DROPPED, BRANCH
007337  104207  000004                  3$:     MOV     #4,R0           ; MAXIMUM OF 4 SUBUNITS
007341  104201  000001                  4$:     MOV     #U.SUBP,R1      ; R1 WILL POINT TO SUBUNIT POINTERS
007343  105051                          5$:     ADD     R5,R1           ; R1 POINTS TO SUBUNIT POINTERS
007344  104212                          1$:     MOV     (R1)+,R2        ; GET POINTER TO SUBUNIT INFO
007345  077353                          2$:     BMI     2$              ; IF UNIT NOT TESTED, BRANCH
007346  104623  000000                  3$:     MOV     S.PARM(R2),R3   ; GET SUBUNIT PARAMETERS
007350  102203  040000                  4$:     BIT     #INITW,R3       ; SEE IF SUBUNIT WILL BE INITALLY WRITTEN
007352  057356                          5$:     BNE     3$              ; IF SO, BRANCH
007353  117407                          2$:     DEC     R0              ; DECREMENT COUNT
007354  057344                          3$:     BNE     1$              ; IF NOT ALL SUBUNITS CHECKED, BRANCH
007355  007367                          4$:     BR      4$              ; BRANCH
007356  104657  000033                  3$:     MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
007360  101207  040000                  4$:     BIS     #INITW,R0       ; FLAG UNIT AS SUBUNITS GET INITIALLY WRITTEN
007362  100657  000033                  5$:     MOV     R0,U.PARM(R5)   ; SAVE
007364  101200  100000  002444          4$:     BIS     #100000,M.PARM  ; FLAG MASTER PARAMETERS AS INITIAL WRITE TO BE DONE
007367  104207  000004                  4$:     MOV     #4,R0           ; MAXIMUM OF FOUR SUBUNITS
007371  104201  000005                  5$:     MOV     #U.SUBP+4,R1    ; R1 WILL POINT TO AFTERS SUBUNIT POINTERS
007373  105051                          5$:     ADD     R5,R1           ; R1 POINTS TO AFTERS SUBUNIT POINTERS
007374  114003                          5$:     CLR     R3              ; INITILIZE SUBUNIT PROTECTION FLAGS
007375  110203                          5$:     ROL     R3              ; ROTATE R3
007376  103203  000001                  5$:     BIC     #1,R3           ; CLEAR LO BIT
007400  104412                          6$:     MOV     -(R1),R2        ; GET SUBUNIT POINTER
007401  077411                          6$:     BMI     6$              ; IF NO SUBUNIT, BRANCH
007402  104622  000000                  6$:     MOV     S.PARM(R2),R2   ; GET SUBUNIT STATUS
007404  102202  004000                  6$:     BIT     #RONLY,R2       ; SEE IF READ ONLY DRIVE
007406  017411                          6$:     BEQ     6$              ; IF NOT READ ONLY, BRANCH
007407  101203  000001                  6$:     BIS     #1,R3           ; SET READ ONLY BIT
007411  117407                          6$:     DEC     R0              ; DECREMENT COUNT
007412  057375                          6$:     BNE     5$              ; IF COUNT UNEXPIRED, BRANCH
007413  110203                          6$:     ROL     R3              ; ROTATE MASK TO CORRECT POSITION
007414  110203                          6$:     ROL     R3              ; ROTATE MASK TO CORRECT POSITION
007415  110203                          6$:     ROL     R3              ; ROTATE MASK TO CORRECT POSITION
007416  110203                          6$:     ROL     R3              ; ROTATE MASK TO CORRECT POSITION
007417  103203  177417                  6$:     BIC     #LBHINB,R3      ; CLEAR UNUSED BITS
007421  100653  000032                  6$:     MOV     R3,U.WPRT(R5)   ; SAVE
007423  024161                          6$:     CALL    RECOVR          ; 'RECOVER' FROM ALL ERRORS
007424  115003                          6$:     TST     R3              ; SEE IF IT EXECUTED CORRECTLY
007425  057440                          6$:     BNE     ONLERR          ; IF NOT, BRANCH
007426  104657  000033                  6$:     MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
007430  103207  002000                  6$:     BIC     #SEKINP,R0      ; CLEAR SEEK IN PROGRESS BIT
007432  100657  000033                  6$:     MOV     R0,U.PARM(R5)   ; SAVE UNIT PARAMETERS
007434  104207  000001                  6$:     MOV     #SETUP,R0       ; SETUP NEXT ROUTINE CALLED
007436  104051                          6$:     MOV     R5,R1           ; MAKE R1 NON-ZERO (DEFERRED CALL)
007437  007466                          6$:     BR      DRVEXT          ; BRANCH
007440                                  ONLERR: SOFTER  21              ; REPORT ERROR
007440  104200  003170  001012                          MOV     #ER21,OUT.04
007443  104200  000025  001010                          MOV     #21,OUT.02
007446  101200  001400  001010                          BIS     #ERSOFT,OUT.02
007451  104200  007451  001007                          MOV     #..,OUT.01
```

```
        007454  104202  000013                                          MOV     #ERRMES,R2
        007456  104020  001006                                          MOV     R2,OUT.RQ
     54 007460                          ENDERR
        007460  101200  000005  002357                                  BIS     #5,ERRPOS       ; SET THE POSITION
     55 007463  104207  007334          MOV     #INITD,R0       ; RE-EXECUTE DRIVE ONLINE ROUTINE
     56 007465  114001                  CLR     R1              ; IMMIDATE CALL TO DRIVE ONLINE
     57 007466  003127          DRVEXT: BR      JMPRET          ; RETURN TO SEQUENCER
```

```
                                        .SBTTL  GMPARM - GET MASTER PARAMETERS (PATTERN 16) AND SET UP OVERLAY ADDRESSES
007467                                  GMPARM:
                                        ;
                                        ;       GET MASTER PARAMETERS
                                        ;
007467  104302  007774                          MOV     OVSTRT,R2       ; GET STARTING ADDRESS OF OVERLAY (LO)
007471  104303  007775                          MOV     OVSTRT+1,R3     ; GET STARTING ADDRESS OF OVERLAY (HI)
007473  104020  002362                          MOV     R2,OTABLE+2     ; MOVE STARTING ADDRESS INTO OVERLAY TABLE
007475  104207  000011                          MOV     #NUMOVL-1,R0    ; R0 HAS NUMBER OF OVERLAYS
007477  104205  002360                          MOV     #OTABLE,R5      ; R5 POINTS TO OVERLAY TABLE
007501  104651  000003                  MORE:   MOV     3(R5),R1        ; R1 HAS OVERLAY LENGTH
007503  110601                                  ROR     R1              ; SHIFT TO MAKE BYTES
007504  103201  100001                          BIC     #100001,R1      ; CLEAR UNUSED BITS
007506  105012                                  ADD     R1,R2           ; FIND ADDRESS OF NEXT OVERLAY
007507  047511                                  BCC     1$              ; IF NO CARRY, BRANCH
007510  115403                                  INC     R3              ; PROPOGATE CARRY
007511  100652  000006                  1$:     MOV     R2,6(R5)        ; STORE STARTING ADDRESS OF NEXT OVERLAY
007513  104651  000007                          MOV     7(R5),R1        ; GET OVERLAY LENGTH (<<2)
007515  101031                                  BIS     R3,R1           ; SET HI ORDER OVERLAY ADDRESS
007516  100651  000007                          MOV     R1,7(R5)        ; SAVE
007520  105205  000004                          ADD     #4,R5           ; POINT TO NEXT OVERLAY AREA
007522  117407                                  DEC     R0              ; DECREMENT COUNT
007523  057501                                  BNE     MORE            ; IF ALL OVERLAYS NOT SET UP, BRANCH
007524  104207  000003                          MOV     #T4MPRM,R0      ; R0 CONTAINS HOST REQUEST
007526  020751                                  CALL    HOSTRQ          ; INITIATE HOST REQUEST
007527  104207  001045                          MOV     #IN.01,R0       ; R0 POINTS TO PATTERN INFORMATION
007531  104272                                  MOV     (R0)+,R2        ; R2 HAS LENGTH OF PATTERN
007532  017542                                  BEQ     3$              ; IF NO PATTERN, BRANCH
007533  104201  002515                          MOV     #PAT0,R1        ; R1 POINTS TO PATTERN 16 AREA
007535  100212                                  MOV     R2,(R1)+        ; MOVE PATTERN LENGTH TO PAT0 AREA
007536  104273                  2$:     MOV     (R0)+,R3        ; GET WORD OF PATTERN
007537  100213                                  MOV     R3,(R1)+        ; MOVE TO PATTERN 16 AREA
007540  117402                                  DEC     R2              ; DECREMENT COUNT
007541  057536                                  BNE     2$              ; IF COUNT UNEXHAUSTED, BRANCH
007542  000000                  3$:     RETURN                          ; RETURN TO CALLING PROGRAM
```

GETU - POLL ALL PORTS, THEN GET UNITS TO TEST

```
                                .SBTTL  GETU   - POLL ALL PORTS, THEN GET UNITS TO TEST
007543                          GETU:
                                ;
                                ;       POLL ALL PORTS AND FILL IN A UDA PORT INFORMATION TABLE (UNITS)
                                ;
007543  104205  000001                  MOV     #1,R5           ; MOVE INITIAL MASK TO R5
007545  104204  004514                  MOV     #UNITS,R4       ; R4 POINTS TO UNIT TABLE
007547                          1$:     PUSH    R4              ; SAVE R4
007547  100464                                                                  MOV R4,-(SP)
007550  104052                  MOV     R5,R2           ; MOVE MASK TO R2
007551  060011                  XFC     DINIT           ; INITILIZE THE DRIVE
007552  104207  040000          MOV     #40000,R0       ; MOVE TIMEOUT TO R0
007554  020720                  14$:    CALL    RDSTAT          ; GET STATUS
007555  102201  010000          BIT     #10000,R1       ; SEE IF STATUS IS VALID
007557  057563                  BNE     3$              ; IF NOT, BRANCH
007560  102201  000001          BIT     #RCVRDY,R1      ; SEE IF RECEIVER READY ASSERTED
007562  057566                  BNE     5$              ; IF SO, BRANCH
007563  117407                  3$:     DEC     R0              ; DECREMENT COUNT
007564  057554                  BNE     14$             ; IF INCOMPLETE, BRANCH
007565  007641                  BR      6$              ; NO VALID STATE
007566  104203  001501          5$:     MOV     #CR.GST,R3      ; R3 POINTS TO GET STATUS COMMAND
007570  104237                  MOV     (R3)+,R0        ; SET ADR OF SDI COMMAND BUFFER
007571  104231                  MOV     (R3)+,R1        ; SET BUFFER LENGTH
007572  060004                  XFC     SEND            ; SEND COMMAND
007573  115001                  TST     R1              ; DID UNIT ACCEPT COMMAND
007574  057641                  BNE     6$              ; IF SO, BRANCH
007575                          PUSH    R4              ; SAVE R4
007575  100464                                                                  MOV R4,-(SP)
007576  104204  000012          MOV     #10.,R4         ; SET UP SHORT TIMEOUT
007600  104137                  11$:    MOV     (R3),R0         ; SET DATA BUFFER ADDRESS
007601  104631  000001          MOV     1(R3),R1        ; SET BUFFER LENGTH
007603  060005                  XFC     RCV             ; SEND RECEIVE SDI COMMAND
007604  115001                  TST     R1              ; DID ERROR OCCUR
007605  017615                  BEQ     13$             ; IF NOT, BRANCH
007606  106201  000001          CMP     #1,R1           ; SEE IF TIMEOUT
007610  057613                  BNE     12$             ; IF NOT, BRANCH
007611  117404                  DEC     R4              ; DECREMENT TIMEOUT VALUE
007612  057600                  BNE     11$             ; IF NOT TIMEOUT, BRANCH
007613                          12$:    POP     R4              ; RESTORE R4
007613  104264                                                                  MOV (SP)+,R4
007614  007641                  BR      6$              ; BRANCH TO EXIT
```

GETU - POLL ALL PORTS, THEN GET UNITS TO TEST

```
                                        ;
                                        ;       VALID UNIT FOUND, FILL IN TABLE
                                        ;
007615                          13$:    POP     R4              ; RESTORE R4
007615  104264                                                                  MOV (SP)+,R4
007616  104307  001570                  MOV     ST,R0           ; R0 HAS UNIT NUMBER
007620  104072                          MOV     R0,R2           ; COPY R0 TO R2
007621  103207  170000                  BIC     #^CHBHINB,R0    ; R0 HAS UNIT NUMBER
007623  101207  040000                  BIS     #40000,R0       ; FLAG UNIT AS NOT TESTED
007625  110702                          SWAB    R2              ; SWAP R2'S BYTES
007626  110602                          ROR     R2              ; MOVE SUBUNIT MASK TO LO NIBBLE
007627  110602                          ROR     R2              ; MOVE SUBUNIT MASK TO LO NIBBLE
007630  110602                          ROR     R2              ; MOVE SUBUNIT MASK TO LO NIBBLE
007631  110602                          ROR     R2              ; MOVE SUBUNIT MASK TO LO NIBBLE
007632  103202  177760                  BIC     #LBLONB,R2      ; CLEAR ALL BUT SUBUNIT BITS
007634  110602                  4$:     ROR     R2              ; MOVE SUBUNIT BIT TO CARRY
007635  047641                          BCC     6$              ; IF NO MORE SUBUNITS, BRANCH
007636  100247                          MOV     R0,(R4)+        ; MOVE SUBUNIT NUMBER TO TABLE
007637  115407                          INC     R0              ; INCREMENT SUBUNIT NUMBER
007640  007634                          BR      4$              ; BRANCH
007641                          6$:     POP     R4              ; R4 POINTS TO START OF UNIT JUST HANDLED
007641  104264                                                                  MOV (SP)+,R4
007642  105204  000004                  ADD     #4,R4           ; R4 WILL POINT TO NEXT UNIT (CLEAR CARRY FOR ROL)
007644  110205                          ROL     R5              ; R5 HAS NEXT UNIT PORT MASK
007645  106205  000020                  CMP     #20,R5          ; SEE IF ALL PORTS TESTED
007647  057547                          BNE     1$              ; IF NOT, BRANCH
```

GETU - POLL ALL PORTS, THEN GET UNITS TO TEST

```
                                                ;
                                                ;       NOW GET THE PLUG NUMBERS TO TEST AND FIND THEM IN THE TABLE
                                                ;
 007650  104207  000012                         MOV     #UTOTST,R0      ; GET WHAT SUBUNIT NUMBERS TO TEST REQUEST
 007652  020751                                 CALL    HOSTRQ          ; GET THE PLUG NUMBERS
 007653  104207  001045                         MOV     #IN.01,R0       ; R0 POINTS TO UNIT NUMBERS TO TEST
 007655  104201  004514                 7$:     MOV     #UNITS,R1       ; R1 POINTS TO UDA PORT INFORMATION
 007657  104112                         8$:     MOV     (R1),R2         ; R2 HAS UNIT
 007660  077667                                 BMI     9$              ; IF NONE, BRANCH
 007661  103202  170000                         BIC     #^CHBHINB,R2    ; CLEAR THE 'NOT TESTED BIT' FOR COMPARISON
 007663  106172                                 CMP     (R0),R2         ; SEE IF IT IS A UNIT TO TEST
 007664  057667                                 BNE     9$              ; NO MATCH
 007665  100112                                 MOV     R2,(R1)         ; SAVE UNIT AS ONE TO TEST
 007666  007726                                 BR      10$             ; LOOK FOR THE NEXT ONE
 007667  115401                         9$:     INC     R1              ; LOOK AT NEXT UNIT
 007670  1062C1  004534                         CMP     #UNITS+16.,R1   ; SEE IF ENTIRE TABLE SEARCHED
 007672  057657                                 BNE     8$              ; IF NOT, BRANCH
 007673                                         DEVFTL  80.             ; REPORT FATAL ERROR
 007673  104200  010365  001012                                                 MOV     #ER80,OUT.04
 007676  104200  000120  001010                                                 MOV     #80,OUT.02
 007701  101200  000400  001010                                                 BIS     #FTLDEV,OUT.02
 007704  104200  007704  001007                                                 MOV     #.,OUT.01
 007707  104202  000014                                                         MOV     #ERRMC,R2
 007711  104020  001006                                                         MOV     R2,OUT.RQ
 007713  104172                                 MOV     (R0),R2         ; GET UNIT NUMBER
 007714  104020  001011                         MOV     R2,OUT.03       ; SAVE IN OUTPUT
 007716  104207  000014                         MOV     #ERRMC,R0       ; SET UP FOR REPORT
 007720  020751                                 CALL    HOSTRQ          ; REPORT ERROR
 007721  104207  000016                         MOV     #DONE,R0        ; END TEST
 007723  020751                                 CALL    HOSTRQ          ; SEND END-OF-TEST TO HOST
 007724  060021                                 XFC     EXIT            ; TERMINATE DM
 007725  060000                                 XFC     BREAK           ; ERROR ********
 007726  115407                 10$:    INC     R0              ; POINT TO NEXT "     TO TEST
 007727  104172                                 MOV     (R0),R2         ; CHECK NEXT UNI
 007730  037655                                 BPL     7$              ; FIND IN UDA PORT INFORMATION
 007731  000000                                 RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                        .SBTTL  ********** NON-LOADED OVERLAY, ERROR MESSAGES
                                        ;*****************************************************************************
                                        ;*****************************************************************************
                                        ;*****************************************************************************
                                        ;*****************************************************************************
                                        ;
                                        ;       ERROR MESSAGE FORMAT BUFFERS (NEVER LOADED INTO UDA)
                                        ;
 007732                                 DMOVLY  MS,0                    ; ERROR MESSAGE OVERLAY
 007732  025207                         .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                        .NLIST  BEX
 000000  042  124  110                  .ASCII  \'THIS ERROR SHOULD NEVER APPEAR, IF IT DOES, CONTACT'N\
 000033  042  104  111                  .ASCII  \'DIAGNOSTIC ENGINERRING  <<SAVE PRINTOUT>>'N\
 000061  000                            .BYTE   0
 000062  042  117  125  ER1:           .ASCII  \'OUTSTANDING RESPONSE FROM DRIVE'\
 000102  042  122  105                  .ASCII  \'RESPONSE FOUND: 'D8N\
 000115  042  106  111                  .ASCII  \'FIRST SEVEN WORDS OF RESPONSE:'N\
 000135  042  040  042                  .ASCII  \'' 'D16S2016S2016S2016S2016S2016S2016N\
 000160  042  122  105                  .ASCII  \'REAL TIME DRIVE STATE 'H16N\
 000176  042  123  124                  .ASCII  \'STATUS: 'H16S2H16S2H16S2H16S2H16S2H16S2H16N\
 000224  000                            .BYTE   0
 000225  042  105  122  ER2:           .ASCII  \'ERROR IN DBN/RBN/LBN TO CYLINDER CONVRSION'N\
 000253  042  040  040                  .ASCII  \''     L/R/DBN BEGIN COMPUTED  'D24N\
 000275  042  040  040                  .ASCII  \''     L/R/DBN PER TRACK       'D8N\
 000317  042  040  040                  .ASCII  \''     LBNS PER TRACK IF RBN   'D8N\
 000340  042  040  040                  .ASCII  \''     STARTING CYLINDER       'D32N\
 000362  000                            .BYTE   0
 000363  042  123  105  ER3:           .ASCII  \'SEEK DID NOT COMPLETE, NEITHER ATTENTION OR'N\
 000412  042  122  105                  .ASCII  \'READ/WRITE READY WAS ASSERTED BEFORE TIMEOUT'N\
 000441  042  123  105                  .ASCII  \'SEEK FROM CYL 'D28''  GROUP 'D8N\
 000462  042  040  040                  .ASCII  \''     TO   CYL 'D28''  GROUP 'D8N\
 000502  042  122  105                  .ASCII  \'REAL TIME DRIVE STATE 'H16N\
 000520  042  123  124                  .ASCII  \'STATUS: 'H16S2H16S2H16S2H16S2H16S2H16S2H16N\
 000546  000                            .BYTE   0
 000547  042  101  124  ER4:           .ASCII  \'ATTENTION ASSERTED AFTER SEEK INITIATED'N\
 000574  042  123  105                  .ASCII  \'SEEK FROM CYL 'D28''  GROUP 'D8N\
 000614  042  040  040                  .ASCII  \''     TO   CYL 'D28''  GROUP 'D8N\
 000635  042  111  115                  .ASCII  \'IMMIDATE DRIVE STATE  'D16N\
 000653  042  122  105                  .ASCII  \'REAL TIME DRIVE STATE 'H16N\
 000671  042  123  124                  .ASCII  \'STATUS: 'H16S2H16S2H16S2H16S2H16S2H16S2H16N\
 000717  000                            .BYTE   0
 000720  042  110  105  ER5:           .ASCII  \'HEADER COMPARE FAILURE DURING WRITE'N\
 000743  042  127  122                  .ASCII  \'WRITE ATTEMPT RETRIES:  'D3N\
 000761  042  114  057                  .ASCII  \'L/DBN NUMBER    'D24N\
 000774  042  101  103                  .ASCII  \'ACTUAL L/R/DBN  'D24N\
 001007  042  124  122                  .ASCII  \'TRK 'D8''   GRP 'D8''   CYL 'D28N\
 001027  042  117  122                  .ASCII  \'ORIGIN OF LAST SEEK WAS CYL 'D28''  GROUP 'D8N\
 001056  042  122  105                  .ASCII  \'REAL TIME DRIVE STATE 'H16N\
 001074  042  123  124                  .ASCII  \'STATUS: 'H16S2H16S2H16S2H16S2H16S2H16S2H16N\
 001122  000                            .BYTE   0
 001123  042  124  111  ER6:           .ASCII  \'TIMEOUT OF DRIVE DURING WRITE ATTEMPT'N\
 001147  042  127  122                  .ASCII  \'WRITE ATTEMPT RETRIES:  'D3N\
 001165  042  114  057                  .ASCII  \'L/DBN NUMBER    'D24N\
 001200  042  101  103                  .ASCII  \'ACTUAL L/R/DBN  'D24N\
 001213  042  124  122                  .ASCII  \'TRK 'D8''   GRP 'D8''   CYL 'D28N\
 001233  042  117  122                  .ASCII  \'ORIGIN OF LAST SEEK WAS CYL 'D28''  GROUP 'D8N\
 001262  042  122  105                  .ASCII  \'REAL TIME DRIVE STATE 'H16N\
```

```
001300   042   123   124          .ASCII  \''STATUS: 'H16S2H16S2H16S2H16S2H16S2H16N\
001326   000                      .BYTE   0
001327   042   105   103   ER7:   .ASCII  \''ECC DETECTED ERROR''N\
001341   042   122   105          .ASCII  \''RETRY  'D4N\
001347   042   114   057          .ASCII  \''L/DBN NUMBER     'D24N\
001362   042   101   103          .ASCII  \''ACTUAL L/R/DBN   'D24N\
001375   042   124   122          .ASCII  \''TRK 'D8''   GRP 'D8''    CYL 'D28N\
001415   042   105   104          .ASCII  \''EDC COMPUTED     'O16N\
001430   042   105   104          .ASCII  \''EDC READ         'O16N\
001443   000                      .BYTE   0
001444   042   105   103   ER8:   .ASCII  \''ECC DETECTED ERROR, BUT CORRECTION FAILED''N\
001472   042   122   105          .ASCII  \''RETRY  'D4N\
001500   042   114   057          .ASCII  \''L/DBN NUMBER     'D24N\
001513   042   101   103          .ASCII  \''ACTUAL L/R/DBN   'D24N\
001526   042   124   122          .ASCII  \''TRK 'D8''   GRP 'D8''    CYL 'D28N\
001546   000                      .BYTE   0
001547   042   105   103   ER9:   .ASCII  \''ECC DETECTED ERROR, BUT CORRECTIONS EXCEED UNIT THRESHOLD''N\
001605   042   122   105          .ASCII  \''RETRY  'D4N\
001613   042   114   057          .ASCII  \''L/DBN NUMBER     'D24N\
001626   042   101   103          .ASCII  \''ACTUAL L/R/DBN   'D24N\
001641   042   124   122          .ASCII  \''TRK 'D8''   GRP 'D8''    CYL 'D28N\
001661   000                      .BYTE   0
001662   042   105   103   ER10:  .ASCII  \''ECC DETECTED ERROR, CORRECTED IT, BUT EDC STILL DETECTS ERROR''N\
001722   042   122   105          .ASCII  \''RETRY  'D4N\
001730   042   114   057          .ASCII  \''L/DBN NUMBER     'D24N\
001743   042   101   103          .ASCII  \''ACTUAL L/R/DBN   'D24N\
001756   042   124   122          .ASCII  \''TRK 'D8''   GRP 'D8''    CYL 'D28N\
001776   042   105   104          .ASCII  \''EDC COMPUTED     'O16N\
002011   042   105   104          .ASCII  \''EDC READ         'O16N\
002024   000                      .BYTE   0
002025   042   105   122   ER11:  .ASCII  \''ERROR RECOVERY TRIED ALL LEVELS WITHOUT SUCCESS''N\
002056   000                      .BYTE   0
002057   042   104   101   ER12:  .ASCII  \''DATA COMPARISON AGAINST KNOWN PATTERN FAILED''N\
002106   042   114   057          .ASCII  \''L/DBN NUMBER     'D24N\
002121   042   101   103          .ASCII  \''ACTUAL L/R/DBN   'D24N\
002134   042   124   122          .ASCII  \''TRK 'D8''   GRP 'D8''    CYL 'D28N\
002154   042   120   101          .ASCII  \''PATTERN NUMBER   'D4N\
002167   042   117   106          .ASCII  \''OFFSET OF ERROR WITHIN BUFFER 'D8N\
002210   042   117   106          .ASCII  \''OFFSET OF ERROR WITHIN DISPLAYED LIST 'D8N\
002236   123   064   117          .ASCII  \S4016S4016S4016S4016S4016S4016S4016\
002257   123   064   117          .ASCII  \S4016S4016S4016S4016S4016S4016S4016\
002301   000                      .BYTE   0
002302   042   125   116   ER16:  .ASCII  \''UNABLE TO RECEIVE REAL TIME DRIVE STATE AFTER SEEK''N\
002334   042   123   105          .ASCII  \''SEEK FROM CYL 'D28''   GROUP 'D8N\
002355   042   040   040          .ASCII  \''       TO   CYL 'D28''   GROUP 'D8N\
002375   042   122   105          .ASCII  \''REAL TIME DRIVE STATE 'H16N\
002413   042   123   124          .ASCII  \''STATUS: 'H16S2H16S2H16S2H16S2H16S2H16N\
002441   000                      .BYTE   0
002442   042   125   116   ER17:  .ASCII  \''UNABLE TO RECEIVE REAL TIME DRIVE STATE''N\
002467   000                      .BYTE   0
002470   042   101   124   ER18:  .ASCII  \''ATTEMPT TO WRITE ON WRITE PROTECTED DRIVE''N\
002516   042   122   105          .ASCII  \''REAL TIME DRIVE STATE 'H16N\
002534   042   123   124          .ASCII  \''STATUS: 'H16S2H16S2H16S2H16S2H16S2H16N\
002562   000                      .BYTE   0
002563   042   110   105   ER19:  .ASCII  \''HEADER COMPARE FAILURE DURING READ''N\
002605   042   122   105          .ASCII  \''READ ATTEMPT RETRIES:  'D3N\
002623   042   114   057          .ASCII  \''L/DBN NUMBER     'D24N\
```

```
114 002636  042  101  103          .ASCII  \'ACTUAL L/R/DBN  'D24N\
115 002651  042  124  122          .ASCII  \'TRK 'D8''  GRP 'D8''  CYL 'D28N\
116 002671  042  117  122          .ASCII  \'ORIGIN OF LAST SEEK WAS CYL 'D28''  GROUP 'D8N\
117 002720  042  122  105          .ASCII  \'REAL TIME DRIVE STATE 'H16N\
118 002736  042  123  124          .ASCII  \'STATUS: 'H16S2H16S2H16S2H16S2H16S2H16S2H16N\
119 002764  000                    .BYTE   0
120 002765  042  124  111  ER20:   .ASCII  \'TIMEOUT OF DRIVE DURING READ ATTEMPT'N\
121 003010  042  122  105          .ASCII  \'READ ATTEMPT RETRIES:  'D3N\
122 003026  042  114  057          .ASCII  \'L/DBN NUMBER    'D24N\
123 003041  042  101  103          .ASCII  \'ACTUAL L/R/DBN  'D24N\
124 003054  042  124  122          .ASCII  \'TRK 'D8''  GRP 'D8''  CYL 'D28N\
125 003074  042  117  122          .ASCII  \'ORIGIN OF LAST SEEK WAS CYL 'D28''  GROUP 'D8N\
126 003123  042  122  105          .ASCII  \'REAL TIME DRIVE STATE 'H16N\
127 003141  042  123  124          .ASCII  \'STATUS: 'H16S2H16S2H'6S2H16S2H16S2H16S2H16N\
128 003167  000                    .BYTE   0
129 003170  042  125  116  ER21:   .ASCII  \'UNABLE TO BRING DRIVE ONLINE/CHANGE MODE/CLEAR ERRORS FOR TEST'N\
130 003230  000                    .BYTE   0
131 003231  042  105  103  ER23:   .ASCII  \'ECC DETECTED ERROR IN BUFFER AND CORRECTED IT'N\
132 003261  042  122  105          .ASCII  \'RETRY  'D4N\
133 003267  042  114  057          .ASCII  \'L/DBN NUMBER    'D24N\
134 003302  042  101  103          .ASCII  \'ACTUAL L/R/DBN  'D24N\
135 003315  042  124  122          .ASCII  \'TRK 'D8''  GRP 'D8''  CYL 'D28N\
136 003335  000                    .BYTE   0
137 003336  042  105  104  ER24:   .ASCII  \'EDC DETECTED ERROR IN BUFFER BUT ECC DID NOT'N\
138 003365  042  122  105          .ASCII  \'RETRY  'D4N\
139 003373  042  114  057          .ASCII  \'L/DBN NUMBER    'D24N\
140 003406  042  101  103          .ASCII  \'ACTUAL L/R/DBN  'D24N\
141 003421  042  124  122          .ASCII  \'TRK 'D8''  GRP 'D8''  CYL 'D28N\
142 003441  042  105  104          .ASCII  \'EDC COMPUTED    'D16N\
143 003454  042  105  104          .ASCII  \'EDC READ        'D16N\
144 003467  000                    .BYTE   0
145 003470  042  101  124  ER25:   .ASCII  \'ATTEMPTED WRITE MAXIMUM NUMBER OF TIMES'N\
146 003515  042  114  057          .ASCII  \'L/DBN NUMBER    'D24N\
147 003530  042  101  103          .ASCII  \'ACTUAL L/R/DBN  'D24N\
148 003543  000                    .BYTE   0
149 003544  042  101  124  ER26:   .ASCII  \'ATTEMPTED READ MAXIMUM NUMBER OF TIMES'N\
150 003570  042  114  057          .ASCII  \'L/DBN NUMBER    'D24N\
151 003603  042  101  103          .ASCII  \'ACTUAL L/R/DBN  'D24N\
152 003616  000                    .BYTE   0
153 003617  042  123  105  ER27:   .ASCII  \'SEEK COMMAND RECIEVED UNSUCCESSFUL RESPONSE'N\
154 003646  042  101  103          .ASCII  \'ACTUAL L/R/DBN  'D24N\
155 003661  042  124  122          .ASCII  \'TRK 'D8''  GRP 'D8''  CYL 'D28N\
156 003701  042  122  105          .ASCII  \'REAL TIME DRIVE STATE 'H16N\
157 003717  042  123  124          .ASCII  \'STATUS: 'H16S2H16S2H16S2H16S2H16S2H16S2H16N\
158 003745  000                    .BYTE   0
159 003746  042  122  105  ER28:   .ASCII  \'RECEIVER READY NEVER ASSERTED AFTER INIT'N\
160 003773  042  122  105          .ASCII  \'REAL TIME DRIVE STATE 'D16N\
161 004011  000                    .BYTE   0
162 004012  042  125  116  ER29:   .ASCII  \'UNABLE TO GET REAL TIME DRIVE STATE AFTER DRIVE INITIALIZE'N\
163 004050  000                    .BYTE   0
164 004051  042  122  105  ER30:   .ASCII  \'READ/WRITE READY DROPPED BEFORE WRITE'N\
165 004075  042  127  122          .ASCII  \'WRITE ATTEMPT RETRIES:  'D3N\
166 004113  042  114  057          .ASCII  \'L/DBN NUMBER    'D24N\
167 004126  042  107  122          .ASCII  \'GRP 'D8''  CYL 'D28N\
168 004141  042  117  122          .ASCII  \'ORIGIN OF LAST SEEK WAS CYL 'D28''  GROUP 'D8N\
169 004170  042  111  115          .ASCII  \'IMMIDATE STATE        'D16N\
170 004202  042  122  105          .ASCII  \'REAL TIME DRIVE STATE 'H16N\
```

```
171 004220  042  123  124        .ASCII  \'STATUS: 'H16S2H16S2H16S2H16S2H16S2H16S2H16N\
172 004246  000                  .BYTE   0
173 004247  042  122  105  ER31: .ASCII  \'READ/WRITE READY DROPPED BEFORE READ'N\
174 004272  042  122  105        .ASCII  \'READ ATTEMPT RETRIES:  'D3N\
175 004310  042  114  057        .ASCII  \'L/DBN NUMBER    'D24N\
176 004323  042  107  122        .ASCII  \'GRP 'D8''   CYL 'D28N\
177 004336  042  117  122        .ASCII  \'ORIGIN OF LAST SEEK WAS CYL 'D28''  GROUP 'D8N\
178 004365  042  111  115        .ASCII  \'IMMIDATE STATE        'O16N\
179 004377  042  122  105        .ASCII  \'REAL TIME DRIVE STATE 'H16N\
180 004415  042  123  124        .ASCII  \'STATUS: 'H16S2H16S2H16S2H16S2H16S2H16S2H16N\
181 004443  000                  .BYTE   0
182 004444  042  111  116  ER32: .ASCII  \'INSUFFICIENT BUFFER MEMORY FOR READ/WRITE BUFFERS'\
183 004475  042  124  110        .ASCII  \'THIS ERROR SHOULD NEVER APPEAR, CONTACT DIAGNOSTIC'\
184 004527  042  123  125        .ASCII  \'SUPPORT  <<SAVE PRINTOUT>>'\
185 004545  000                  .BYTE   0
186 004546  042  103  110  ER33: .ASCII  \'CHECKSUM ERRORS WHEN READING OVERLAYS FROM HOST MEMORY'N\
187 004602  042  122  125        .ASCII  \'RUN TEST 1 TO INVESTIGATE PROBLEM'N\
188 004624  000                  .BYTE   0
189 004625  042  101  114  ER40: .ASCII  \'ALL COPIES OF RCT READ WITH ERROR,  UNABLE TO FIND RBN TO'N\
190 004663  042  122  105        .ASCII  \'REPLACE REVECTORED LBN'N\
191 004677  042  114  101        .ASCII  \'LAST RCT LBN SEARCHED: 'D24N\
192 004716  042  123  105        .ASCII  \'SEARCING FOR LBN:      'D24N\
193 004734  000                  .BYTE   0
194 004735  042  101  114  ER41: .ASCII  \'ALL COPIES OF RCT READ WITH ERROR,  UNABLE TO FIND RBN TO'N\
195 004773  042  122  105        .ASCII  \'REPLACE LBN WITH HEADER COMPARE ERROR'N\
196 005017  042  114  101        .ASCII  \'LAST RCT LBN SEARCHED: 'D24N\
197 005035  042  123  105        .ASCII  \'SEARCING FOR LBN:      'D24N\
198 005054  000                  .BYTE   0
199 005055  042  101  114  ER42: .ASCII  \'ALL COPIES OF RCT READ WITH ERROR,  UNABLE TO FIND RBN TO'N\
200 005113  042  122  105        .ASCII  \'REPLACE LBN WITH TIMEOUT ERROR'N\
201 005133  042  114  101        .ASCII  \'LAST RCT LBN SEARCHED: 'D24N\
202 005152  042  123  105        .ASCII  \'SEARCING FOR LBN:      'D24N\
203 005170  000                  .BYTE   0
204 005171  042  105  116  ER43: .ASCII  \'ENTIRE RCT AREA SEARCHED, COULD NOT FIND RBN TO REPLACE'N\
205 005226  042  122  105        .ASCII  \'REVECTORED LBN'N\
206 005236  042  123  105        .ASCII  \'SEARCING FOR LBN:      'D24N\
207 005255  000                  .BYTE   0
208 005256  042  105  116  ER44: .ASCII  \'ENTIRE RCT AREA SEARCHED, COULD NOT FIND RBN TO REPLACE'N\
209 005313  042  114  102        .ASCII  \'LBN WITH HEADER COMPARE ERROR'N\
210 005333  042  123  105        .ASCII  \'SEARCING FOR LBN:      'D24N\
211 005351  000                  .BYTE   0
212 005352  042  105  116  ER45: .ASCII  \'ENTIRE RCT AREA SEARCHED, COULD NOT FIND RBN TO REPLACE'N\
213 005407  042  114  102        .ASCII  \'LBN WITH TIMEOUT ERROR'N\
214 005423  042  123  105        .ASCII  \'SEARCING FOR LBN:      ' 24N\
215 005442  000                  .BYTE   0
216 005443  042  102  105  ER50: .ASCII  \'BEGIN/END SET STA TING BLOCK NUMBER GREATER THAN'N\
217 005474  042  105  116        .ASCII  \'ENDING BLOCK NUMBER'N\
218 005507  000                  .BYTE   0
219 005510  042  102  105  ER51: .ASCII  \'BEGIN/END SET STARTING BLOCK NUMBER LESS THAN OR'N\
220 005541  042  105  121        .ASCII  \'EQUAL TO THE PREVIOUS ENDING BLOCK NUMBER (OVERLAP)'N\
221 005574  000                  .BYTE   0
222 005575  042  114  101  ER52: .ASCII  \'LAST BEGIN/END SET ENDING BLOCK NUMBER GREATER THAN'N\
223 005630  042  115  101        .ASCII  \'MAXIMUM'N\
224 005635  042  115  101        .ASCII  \'MAXIMUM POSSIBLE BLOCK NUMBER: 'D24N\
225 005657  000                  .BYTE   0
226 005660  042  104  125  ER53: .ASCII  \'DUPLICATE BAD BLOCKS'N\
227 005673  000                  .BYTE   0
```

```
 228 005674  042  114  101  ER54:   .ASCII  \'LAST BAD BLOCK NUMBER EXCEEDS MAXIMUM'N\
 229 005720  042  115  101          .ASCII  \'MAXIMUM POSSIBLE BLOCK NUMBER: 'D24N\
 230 005742  000                    .BYTE   0
 231 005743  042  125  116  ER55:   .ASCII  \'UNABLE TO RECEIVE REAL TIME DRIVE STATE BEFORE WRITE'N\
 232 005776  042  127  122          .ASCII  \'WRITE ATTEMPT RETRIES:  'D3N\
 233 006015  042  114  057          .ASCII  \'L/DBN NUMBER    'D24N\
 234 006030  042  107  122          .ASCII  \'GRP 'D8''   CYL 'D28N,
 235 006042  042  117  122          .ASCII  \'ORIGIN OF LAST SEEK WAS CYL 'D28''  GROUP 'D8N\
 236 006071  042  122  105          .ASCII  \'REAL TIME DRIVE STATE 'H16N\
 237 006107  042  123  124          .ASCII  \'STATUS: 'H16S2H16S2H16S2H16S2H16S2H16S2H16N\
 238 006135  000                    .BYTE   0
 239 006136  042  125  116  ER56:   .ASCII  \'UNABLE TO RECEIVE REAL TIME DRIVE STATE BEFORE READ'N\
 240 006171  042  122  105          .ASCII  \'READ ATTEMPT RETRIES:  'D3N\
 241 006207  042  114  057          .ASCII  \'L/DBN NUMBER    'D24N\
 242 006222  042  107  122          .ASCII  \'GRP 'D8''   CYL 'D28N\
 243 006234  042  117  122          .ASCII  \'ORIGIN OF LAST SEEK WAS CYL 'D28''  GROUP 'D8N\
 244 006263  042  122  105          .ASCII  \'REAL TIME DRIVE STATE 'H16N\
 245 006301  042  123  124          .ASCII  \'STATUS: 'H16S2H16S2H16S2H16S2H16S2H16S2H16N\
 246 006327  000                    .BYTE   0
 247 006330  042  117  126  ER57:   .ASCII  \'OVERFLOW WHEN CALCULATING THE STARTING/ENDING L/DBN'N\
 248 006363  042  106  122          .ASCII  \'FROM THE GIVEN STARTING/ENDING CYLINDER'N\
 249 006410  042  103  131          .ASCII  \'CYLINDER TOO LARGE'N\
 250 006422  000                    .BYTE   0
 251 006423  042  115  101  ER58:   .ASCII  \'MAXIMUM TRACK/GROUP GIVEN EXCEEDS MAXIMUM FOR DEVICE'N\
 252 006456  042  124  122          .ASCII  \'TRACK/GROUP GIVEN: 'D8N\
 253 006472  042  115  101          .ASCII  \'MAXIMUM TRACK/GROUP ON DEVICE: 'D8N\
 254 006514  000                    .BYTE   0
 255 006515  042  124  127  ER59:   .ASCII  \'TWO IDENTICAL TRACKS/GROUPS GIVEN'N\
 256 006537  000                    .BYTE   0
 257 006540  042  105  116  ER61:   .ASCII  \'ENDING LBN COMPUTED FROM ENDING CYLINDER GIVEN EXCEEDS'N\
 258 006574  042  115  101          .ASCII  \'MAXIMUM LBN NUMBER ON DEVICE.  IF YOU SPECIFIED THE LAST'N\
 259 006632  042  103  131          .ASCII  \'CYLINDER ON THE DEVICE, THE REVECTOR TABLE STARTS WITHIN'N\
 260 006667  042  124  110          .ASCII  \'THE CYLINDER SPECIFIED.  IF SO, THE MAXIMUM CYLINDER YOU CAN'N\
 261 006727  042  124  105          .ASCII  \'TEST IS ONE LESS THAN THE MAXIMUM, OR TEST THE ENTIRE AREA'N\
 262 006765  000                    .BYTE   0
 263 006766  042  105  116  ER62:   .ASCII  \'ENDING DBN COMPUTED FROM ENDING CYLINDER GIVEN EXCEEDS'N\
 264 007022  042  115  101          .ASCII  \'MAXIMUM DBN NUMBER ON DEVICE.  IF YOU SPECIFIED THE LAST'N\
 265 007060  042  103  131          .ASCII  \'CYLINDER IN THE DIAGNOSTIC AREA, THE READ ONLY GROUPS START'N\
 266 007117  042  127  111          .ASCII  \'WITHIN THE CYLINDER SPECIFIED.  IF SO, THE MAXIMUM CYLINDER'N\
 267 007156  042  131  117          .ASCII  \'YOU CAN TEST IS ONE LESS THAN THE MAXIMUM, OR TEST THE'N\
 268 007212  042  105  116          .ASCII  \'ENTIRE DIAGNOSTIC AREA'N\
 269 007227  000                    .BYTE   0
 270 007230  042  124  111  ER70:   .ASCII  \'TIMEOUT OF UDA SEND'N\
 271 007243  122  061               .ASCII  \R1\
 272 007244  042  122  105          .ASCII  \'REAL TIME DRIVE STATE 'H16N\
 273 007262  042  123  124          .ASCII  \'STATUS: 'H16S2H16S2H16S2H16S2H16S2H16S2H16N\
 274 007310  000                    .BYTE   0
 275 007311  042  124  111  ER71:   .ASCII  \'TIMEOUT OF UDA RECEIVE'N\
 276 007325  122  061               .ASCII  \R1\
 277 007326  042  122  105          .ASCII  \'REAL TIME DRIVE STATE 'H16N\
 278 007344  042  123  124          .ASCII  \'STATUS: 'H16S2H16S2H16S2H16S2H16S2H16S2H16N\
 279 007372  000                    .BYTE   0
 280 007373  042  106  111  ER72:   .ASCII  \'FIRST WORD RECEIVED BY UDA WAS NOT A START FRAME'N\
 281 007424  122  061               .ASCII  \R1\
 282 007425  042  122  105          .ASCII  \'REAL TIME DRIVE STATE 'H16N\
 283 007443  042  123  124          .ASCII  \'STATUS: 'H16S2H16S2H16S2H16S2H16S2H16S2H16N\
 284 007471  000                    .BYTE   0
```

```
285 007472  042 106 122 ER73:  .ASCII  \"FRAMING ERROR ON SDI LEVEL 0 READ"N\
286 007514  122 061           .ASCII  \R1\
287 007515  042 122 105       .ASCII  \"REAL TIME DRIVE STATE "H16N\
288 007533  042 123 124       .ASCII  \"STATUS: "H16S2H16S2H16S2H16S2H16S2H16S2H16N\
289 007561  000               .BYTE   0
290 007562  042 103 110 ER74:  .ASCII  \"CHECKSUM ERROR ON SDI LEVEL 0 READ"N\
291 007604  122 061           .ASCII  \R1\
292 007605  042 122 105       .ASCII  \"REAL TIME DRIVE STATE "H16N\
293 007623  042 123 124       .ASCII  \"STATUS: "H16S2H16S2H16S2H16S2H16S2H16S2H16N\
294 007651  000               .BYTE   0
295 007652  042 122 105 ER75:  .ASCII  \"RECEIVE BUFFER SIZE SMALLER THAN RESPONSE"N\
296 007700  122 061           .ASCII  \R1\
297 007701  042 122 105       .ASCII  \"REAL TIME DRIVE STATE "H16N\
298 007717  042 123 124       .ASCII  \"STATUS: "H16S2H16S2H16S2H16S2H16S2H16S2H16N\
299 007745  000               .BYTE   0
300 007746  042 122 105 ER76:  .ASCII  \"RESPONSE OF SDI COMMAND NOT AS EXPECTED"N\
301 007773  122 061           .ASCII  \R1\
302 007774  042 122 105       .ASCII  \"RESPONSE RECEIVED     "D8N\
303 010007  042 122 105       .ASCII  \"REAL TIME DRIVE STATE "H16N\
304 010025  042 123 124       .ASCII  \"STATUS: "H16S2H16S2H16S2H16S2H16S2H16S2H16N\
305 010053  000               .BYTE   0
306 010054  042 101 124 SER0:  .ASCII  \"ATTEMPTING TO DO GET STATUS COMMAND"N\
307 010077  000               .BYTE   0
308 010100  042 101 124 SER1:  .ASCII  \"ATTEMPTING TO DO DRIVE CLEAR COMMAND"N\
309 010123  000               .BYTE   0
310 010124  042 101 124 SER2:  .ASCII  \"ATTEMPTING TO DO ONLINE COMMAND"N\
311 010145  000               .BYTE   0
312 010146  042 101 124 SER3:  .ASCII  \"ATTEMPTING TO DO CHANGE MODE COMMAND"N\
313 010171  000               .BYTE   0
314 010172  042 101 124 SER4:  .ASCII  \"ATTEMPTING TO DO ERROR RECOVERY COMMAND"N\
315 010217  000               .BYTE   0
316 010220  042 101 124 SER5:  .ASCII  \"ATTEMPTING TO DO GET SUBUNIT CHARACTERISTICS COMMAND"N\
317 010253  000               .BYTE   0
318 010254  042 101 124 SER6:  .ASCII  \"ATTEMPTING TO DISCONNECT DRIVE"N\
319 010274  000               .BYTE   0
320 010275  042 101 124 SER7:  .ASCII  \"ATTEMPTING TO RECALIBRATE DRIVE"N\
321 010316  000               .BYTE   0
322 010317  042 101 124 SER8:  .ASCII  \"ATTEMPTING TO GET COMMON CHARACTERISTICS"N\
323 010344  000               .BYTE   0
324 010345  042 101 124 SER9:  .ASCII  \"ATTEMPTING TO INITIATE SEEK"N\
325 010364  000               .BYTE   0
326 010365  042 125 116 ER80:  .ASCII  \"UNABLE TO FIND ALL UNITS REQUESTED FOR TESTING.  RUN"N\
327 010420  042 124 105       .ASCII  \"TEST 3 TO FIND WHAT IS VISIBLE ON THE FOUR UDA PORTS"N\
328 010454  000               .BYTE   0
329 010455  042 123 125 ER100: .ASCII  \"SUBUNIT DROPPED FROM TESTING"N\
330 010474  042 122 125       .ASCII  \"RUN OR PORT SWITCH OUT, OR SPINDLE DROPPED READY"N\
331 010526  042 122 105       .ASCII  \"REAL TIME DRIVE STATE "O16N\
332 010544  042 123 124       .ASCII  \"STATUS: "O16S2O16S2O16S2O16S2O16S2O16S2O16N\
333 010572  000               .BYTE   0
```

INFORMATIONAL MESSGES

```
                                        .SBTTL  INFORMATIONAL MESSGES
                                        ;
                                        ;       MESSAGES
                                        ;
010573  116     042     111     MS1:    .ASCII  \N''INITIAL WRITE COMPLETE'N\
010610  000                             .BYTE   0
010611  116     042     101     MS2:    .ASCII  \N''ALL INITIAL WRITES COMPLETE.  TESTING HAS BEGUN'N\
010642  000                             .BYTE   0
```

INFORMATIONAL MESSGES

```
                                        DMOVLY  ST,AREA0
                                        .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                .SBTTL  ********** OVERLAY 1 - SETUP OPERATION TO BE DONE THIS PASS
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;********************************** *******************************************
                                ;
                                ;
 000001                         SETUP   -       1.              ; SETUP OVERLAY
                                ;
                                ;       SETUP CLEARS ALL ERROR BITS, THE ERROR COUNT, FINDS THE NEXT
                                ;       LBN TO BE READ OR WRITTEN, DESIDES TO READ OR WRITE, CALCULATES
                                ;       THE CYLINDER, TRACK AND GROUP THE LBN RESIDES ON; THEN CALLS
                                ;       SEEK.
                                ;
                                        DIAG$$                          ; ************
 004515 115000  002766                  TST     $$DIAG+$DIAG$
 004517 014525                          BEQ     .+6
 004520 104207  060000                  MOV     #60000,R0
 004522 100707  176242                  MOV     R0,@$$DIAG+$DIAG$
 004524 004525                          BR      .+1
 004525 104657  000021                  MOV     U.CSEC(R5),R0   ; GET TOTAL NUMBER OF SECTORS ALLREADY R/W
 004527 105657  000016                  ADD     U.NSEC(R5),R0   ; ADD HOW MANY R/W THIS PASS
 004531 106657  000020                  CMP     U.TSEC(R5),R0   ; SEE IF ALL DONE
 004533 014565                          BEQ     2$              ; IF SO, BRANCH (NEW OPERATION)
 004534 100657  000021                  MOV     R0,U.CSEC(R5)   ; SAVE CURRENT COUNT
 004536 104657  000037                  MOV     U.CBN(R5),R0    ; GET LBN
 004540 105657  000016                  ADD     U.NSEC(R5),R0   ; ADD NUMBER OF SECTORS R/W THIS PASS
 004542 100657  000037                  MOV     R0,U.CBN(R5)    ; SAVE
 004544 044552                          BCC     1$              ; IF NO CARRY, BRANCH
 004545 104657  000040                  MOV     U.CBN+1(R5),R0  ; GET HI LBN
 004547 115407                          INC     R0              ; PROPOGATE CARRY
 004550 100657  000040                  MOV     R0,U.CBN+1(R5)  ; SAVE
 004552 114002                  1$:     CLR     R2              ; NO ERRORS
 004553 104651  000033                  MOV     U.PARM(R5),R1   ; GET UNIT PARAMETERS
 004555 102201  000224                  BIT     #RBNBN!DATERR!RETRY,R1  ; SEE IF REVECTORED BLOCK OR DATA ERROR
 004557 014650                          BEQ     15$             ; IF NOT, BRANCH
 004560 104207  000001                  MOV     #1,R0           ; ONLY READ/WRITE ONE SECTOR
 004562 100657  000017                  MOV     R0,U.MSEC(R5)   ; SAVE
 004564 004702                          BR      3$              ; EXIT -- SEEK
 004565 104657  000033          2$:     MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
 004567 102207  000010                  BIT     #WCHECK,R0      ; SEE IF WRITE CHECK IS TO BE DONE
 004571 014577                          BEQ     5$              ; IF NOT, BRANCH
 004572 103207  000114                  BIC     #WCHECK!REDWRT!RETRY,R0 ; CLEAR WRITE CHECK, MAKE OPERATION READ
 004574 100657  000033                  MOV     R0,U.PARM(R5)   ; SAVE
 004576 004635                          BR      6$              ; BRANCH
 004577 104641  000000          5$:     MOV     S.PARM(R4),R1   ; GET SUBUNIT PARAMETERS
 004601 102201  040100                  BIT     #SEQSEK!INITW,R1        ; SEE IF SEQUENTIAL SEEKS
 004603 054615                          BNE     7$              ; IF SO, BRANCH
 004604 102207  004000                  BIT     #NEWSUB,R0      ; SEE IF NEW SUBUNIT
 004606 054615                          BNE     7$              ; IF SO, BRANCH
 004607 101207  004000                  BIS     #NEWSUB,R0      ; FLAG AS NEW SUBUNIT
 004611 100657  000033                  MOV     R0,U.PARM(R5)   ; SAVE
 004613 114002                          CLR     R2              ; NO ERRORS
 004614 004625                          BR      11$             ; EXIT
```

```
 52 004615  024706                7$:     CALL    NXTSEC          ; GET NEXT LBN
 53 004616  115002                        TST     R2              ; SEE IF AN ERROR OCCURRED
 54 004617  054625                        BNE     11$             ; IF SO, BRANCH
 55 004620  104651  000033                MOV     U.PARM(R5),R1   ; GET UNIT PARAMETERS
 56 004622  102201  004000                BIT     #NEWSUB,R1      ; SEE IF NEW SUBUNIT
 57 004624  014630                        BEQ     10$             ; IF NOT, BRANCH
 58 004625  104207  000C01          11$:  MOV     #SETUP,R0       ; SETUP IS NEXT MODULE CALLED
 59 004627  004704                        BR      4$              ; EXIT
 60 004630  026343                10$:  CALL    CLRUP           ; CLEAR ERROR COUNT AND VARIOUS PARM BITS
 61 004631  026352                        CALL    RORW            ; SEE IF READ OR WRITE
 62 004632  026374                        CALL    PATTRN          ; FIND PATTERN TO WRITE
 63 004633  026416                        CALL    WCHK            ; SEE IF WRITE CHECK WILL BE DONE
 64 004634  026446                        CALL    DCOMP           ; SEE IF DATA COMPARE WILL BE DONE
 65 004635  104657  000035          6$:   MOV     U.MBN(R5),R0    ; GET LO LBN
 66 004637  100657  000037                MOV     R0,U.CBN(R5)    ; SAVE
 67 004641  104657  000036                MOV     U.MBN+1(R5),R0  ; GET HI LBN
 68 004643  100657  000040                MOV     R0,U.CBN+1(R5)  ; SAVE
 69 004645  114002                        CLR     R2              ; NO ERRORS
 70 004646  100652  000021                MOV     R2,U.CSEC(R5)   ; SECTORS R/W IS ZERO
 71 004650  100652  000041          15$:  MOV     R2,U.RBN(R5)    ; ZERO RBN NUMBER
 72 004652  100652  000042                MOV     R2,U.RBN+1(R5)  ; ********* TAKE OUT AFTER ROUTINE 3 IS IMP.
 73 004654  104657  000020                MOV     U.TSEC(R5),R0   ; GET TOTAL NUMBER OF SECTORS TO R/W
 74 004656  107657  000021                SUB     U.CSEC(R5),R0   ; SUBTRACT HOW MANY DONE
 75 004660  104651  000033                MOV     U.PARM(R5),R1   ; GET UNIT PARAMETERS
 76 004662  102201  000100                BIT     #REDWRT,R1      ; SEE IF WRITE IS IN PROGRESS
 77 004664  054673                        BNE     13$             ; IF SO, BRANCH
 78 004665  106307  002353                CMP     SECMAX,R0       ; SEE IF TRYING TO READ MORE THAN POSSIBLE
 79 004667  034700                        BPL     14$             ; IF NOT, BRANCH
 80 004670  104307  002353                MOV     SECMAX,R0       ; SET R0 TO MAXIMUM POSSIBLE SECTORS TO READ
 81 004672  004700                        BR      14$             ; BRANCH
 82 004673  106307  002354          13$:  CMP     CHNMAX,R0       ; SEE IF TRYING TO WRITE MORE THAN POSSIBLE
 83 004675  034700                        BPL     14$             ; IF NOT, BRANCH
 84 004676  104307  002354                MOV     CHNMAX,R0       ; SET R0 TO MAXIMUM POSSIBLE SECTORS TO WRITE
 85 004700  100657  000017          14$:  MOV     R0,U.MSEC(R5)   ; SAVE IN NUMBER OF SECTORS TO R/W THIS PASS
 86 004702  104207  000002          3$:   MOV     #SEEK,R0        ; POINT TO SEEK AS NEXT OPERATION
 87 004704  114001                  4$:   CLR     R1              ; IMMIDATE CALL
 88 004705  003127                  12$:  BR      JMPRET          ; RETURN TO SEQUENCER
```

```
                                .SBTTL  NXTSEC - FIND THE NEXT STARTING SECTOR TO READ/WRITE
004706                          NXTSEC:
                                ;
                                ;       NXTSEC WILL SEE IF THE SECTORS ARE COMPUTED LIMITED BY BEGIN/END
                                ;       SETS OR TRACK/GROUP SETS, AND CALL THE CORRECT MODULE TO COMPUTE THE
                                ;       NEXT SECTOR
                                ;
004706  104647  000000                  MOV     S.PARM(R4),R0   ; GET SUBUNIT PARAMETERS
004710  102207  040100                  BIT     #SEQSEK.INITW,R0        ; SEE IF BN'S ACCESSED SREQUENTIALLY
004712  054730                          BNE     2$              ; IF SO, BRANCH
004713  104651  000033                  MOV     U.PARM(R5),R1   ; GET UNIT PARAMETERS
004715  103201  004000                  BIC     #NEWSUB,R1      ; CLEAR NEW SUBUNIT FLAG
004717  100651  000033                  MOV     R1,U.PARM(R5)   ; SAVE
004721  102207  000040                  BIT     #BEUSED,R0      ; SEE IF BEGIN/END SETS USED
004723  014726                          BEQ     1$              ; IF NOT, BRANCH
004724  024741                          CALL    RNDBE           ; GET A RANDOM BLOCK NUMBER (BEGIN/END SETS)
004725  004737                          BR      5$              ; EXIT
004726  025475                  1$:     CALL    RNDTG           ; GET A RANDOM BLOCK NUMBER (TRACKS/GROUPS)
004727  004737                          BR      5$              ; BRANCH
004730  102207  000040          2$:     BIT     #BEUSED,R0      ; SEE IF BEGIN/END SETS USED
004732  014736                          BEQ     3$              ; IF NOT, BRANCH
004733  025117                          CALL    SEQBE           ; GET A SEQUENTIAL BLOCK NUMBER (BEGIN/END SETS)
004734  114002                          CLR     R2              ; NO ERRORS ********
004735  004740                          BR      4$              ; EXIT (SHOULD BRANCH TO 7$) ********
004736  025641                  3$:     CALL    SEQTG           ; GET A SEQUENTIAL BLOCK NUMBER (TRACKS/GROUPS)
004737  114002                  5$:     CLR     R2              ; NO ERRORS
004740  000000                  4$:     RETURN                  ; RETURN TO CALLING PROGRAM
```

RNDBE - FIND A RANDOM STARTING SECTOR USING BEGIN/END SETS

```
                                        .SBTTL  RNDBE  - FIND A RANDOM STARTING SECTOR USING BEGIN/END SETS
004741                                  RNDBE:
                                        ;
                                        ;       RNDBE GETS A RANDOM LBN NUMBER
                                        ;
                                        ;       GET A RANDOM BEGIN/END SET FROM THE SUBUNIT'S B/E SETS
                                        ;
004741  114007                                  CLR     R0              ; INITILIZE COUNTER
004742  104203  000014                          MOV     #S.BESS+3,R3    ; R3 POINTS TO FIRST B/E SET
004744  105043                                  ADD     R4,R3           ; R3 POINTS TO FIRST B/E SET
004745  104132                          CNTLOP: MOV     (R3),R2         ; MOVE EOL FLAG WORD TO R2
004746  074753                                  BMI     COUNTD          ; IF END-OF-LIST, BRANCH
004747  105203  000004                          ADD     #4,R3           ; POINT TO NEXT EOL WORD
004751  115407                                  INC     R0              ; INCREMENT COUNT
004752  004745                                  BR      CNTLOP          ; IF NOT, TRY AGAIN
004753  115007                          COUNTD: TST     R0              ; SEE IF COUNT = 0
004754  014775                                  BEQ     GOTOFS          ; IF SO, BRANCH
004755  026475                          TRYAGN: CALL    RANDOM          ; GET A RANDOM NUMBER
004756  103201  177774                          BIC     #177774,R1      ; MASK OUT ALL BUT 2 LOWEST BITS
004760  106207  000002                          CMP     #2,R0           ; TEST HOW COUNT RELATES TO 2
004762  074772                                  BMI     GOTOBE          ; IF COUNT IS 3, BRANCH
004763  014767                                  BEQ     THREBE          ; IF COUNT IS 2, BRANCH
004764  103201  177776                          BIC     #177776,R1      ; MASK OUT ALL BUT LOWEST BIT
004766  004772                                  BR      GOTOBE          ; BRANCH
004767  106201  000002                  THREBE: CMP     #2,R1           ; SEE IF RANDOM NUMBER OVER 2
004771  074755                                  BMI     TRYAGN          ; IF SO, GET ANOTHER NUMBER
004772  104017                          GOTOBE: MOV     R1,R0           ; MOVE RANDOM NUMBER TO R0
004773  105077                                  ADD     R0,R0           ; R0 X 2
004774  105077                                  ADD     R0,R0           ; R0 X 2
004775  105207  000011                  GOTOFS: ADD     #S.BESS,R0      ; POINT TO RANDOM BEGIN/END SET
004777  105047                                  ADD     R4,R0           ; POINT TO RANDOM BEGIN/END SET
```

```
                                ;
                                ;       ONCE A RANDOM B/E SET HAS BEEN FOUND, FIND A RANDOM LBN
                                ;       WITHIN THAT B/E SET
                                ;
                                        PUSH    <R4,R5>         ; SAVE R4 AND R5
                                                                                MOV R4,-(SP)
                                                                                MOV R5,-(SP)
                                        MOV     (R0)+,R4        ; R4 GETS LO ORDER ENDING LBN
                                        MOV     (R0)+,R5        ; R5 GETS HI ORDER ENDING LBN
                                        SUB     (R0)+,R4        ; SUBTRACT LO ORDER STARTING LBN
                                        BCC     NOBOR0          ; IF NO BORROW, BRANCH
                                        DEC     R5              ; ADJUST FOR BORROW
                                NOBOR0: MOV     (R0),R1         ; GET HI WORD
                                        BIC     #170000,R1      ; STRIP OFF EOL FLAG (IF ANY)
                                        SUB     R1,R5           ; SUBTRACT HI ORDER STARTING LBN
                                        DEC     R0              ; POINT TO LO ORDER STARTING LBN
                                        PUSH    R0              ; SAVE R0
                                                                                MOV R0,-(SP)
                                        TST     R5              ; SEE IF R5 IS ZERO
                                        BEQ     ONEREG          ; IF SO, BRANCH
                                        MOV     R5,R0           ; MOVE R5 TO R0 TO COMPUTE MASK
                                        CALL    MASK            ; COMPUTE RANDOM NUMBER MASK
                                RNDLP0: CALL    RANDOM          ; GET RANDOM NUMBER
                                        MOV     R1,R3           ; MOVE RANDOM NUMBER TO R3
                                        BIC     R0,R3           ; MASK OUT HIGH BITS
                                        CMP     R5,R3           ; COMPARE MAXIMUM RANDOM NUM TO RANDOM NUMBER
                                        BCS     RNDLP0          ; IF MAX RND NUM < RANDOM NUMBER, BRANCH
                                                                                BCC     .+2
                                                                                BR      RNDLP0
                                        BEQ     MASKLO          ; IF -, BRANCH
                                        CALL    RANDOM          ; GET ANOTHER RANDOM NUMBER
                                        BR      CLCLBN          ; BRANCH
                                ONEREG: CLR     R3              ; ZERO R3
                                MASKLO: MOV     R4,R0           ; MOVE R4 TO R0 TO COMPUTE MASK
                                        CALL    MASK            ; COMPUTE MASK
                                RNDLP1: CALL    RANDOM          ; GET A RANDOM NUMBER
                                        BIC     R0,R1           ; MASK OUT HIGH BITS
                                        CMP     R4,R1           ; SEE IF MAX RND NUM < RANDOM NUMBER
                                        BCS     RNDLP1          ; IF MAX RND NUM < RANDOM NUMBER, BRANCH
                                                                                BCC     .+2
                                                                                BR      RNDLP1
                                CLCLBN: MOV     R1,R2           ; MOVE RANDOM NUMBER TO R2
                                        POP     R0              ; RESTORE R0
                                                                                MOV (SP)+,R0
                                        ADD     (R0)+,R2        ; ADD LO ORDER STARTING BLOCK TO RANDOM NUM
                                        BCC     NOCARY          ; IF NO CARRY, BRANCH
                                        INC     R3              ; ADJUST HIGH ORDER FOR CARRY
                                NOCARY: MOV     (R0),R1         ; MOVE HI ORDER LBN TO R1
                                        BIC     #170000,R1      ; STRIP OFF EOL FLAG IF ANY
                                        ADD     R1,R3           ; ADD HI ORDER LBN
                                        POP     <R5,R4>         ; RESTORE R5, R4
                                                                                MOV (SP)+,R5
                                                                                MOV (SP)+,R4
                                        SUB     #3,R0           ; R0 POINTS TO ENDING LBN
                                        MOV     (R0)+,R1        ; R1 HAS LO ORDER ENDING LBN
                                        MOV     (R0),R0         ; R0 HAS HI ORDER ENDING LBN
                                        SUB     R2,R1           ; SUBTRACT LO LBN FROM LO ENDING LBN
```

RNDBE - FIND A RANDOM STARTING SECTOR USING BEGIN/END SETS

```
 48 005062  075067                        BMI     4$              ; IF RESULT IS NEGATIVE, BRANCH
 49 005063  045065                        BCC     1$              ; IF NO CARRY, BRANCH
 50 005064  117407                        DEC     R0              ; DECREMENT HI ENDING LBN
 51 005065  107037                1$:     SUB     R3,R0           ; SUBTRACT HI LBN FROM HI ENDING LBN
 52 005066  015071                        BEQ     2$              ; IF HI WORD IS ZERO, BRANCH
 53 005067  104201  077776        4$:     MOV     #77776,R1       ; MOVE MAXIMUM COUNT TO R1
 54 005071  105401                2$:     INC     R1              ; INCREMENT MAXIMUM SECTOR COUNT
 55 005072  104017                        MOV     R1,R0           ; COPY TO R0 FOR MAXMUM
 56 005073  026534                        CALL    MAXMUM          ; SET MAXIMUM COUNT (U.TSEC AND U.MSEC)
 57 005074  104207  000035                MOV     #U.MBN,R0       ; R0 POINTS TO LBN AREA
 58 005076  105057                        ADD     R5,R0           ; R0 POINTS TO LBN AREA
 59 005077  100272                        MOV     R2,(R0)+        ; MOVE LO ORDER TO LBN AREA
 60 005100  100173                        MOV     R3,(R0)         ; MOVE HI ORDER TO LBN AREA
 61 005101  000000                        RETURN                  ; RETURN TO RNDBLK
```

```
                                .SBTTL  MASK    - FIND MASK FOR RANDOM NUMBER
005102                          MASK:
                                ;
                                ;       MASK MASKS OUT HIGHER BITS OF A RANDOM NUMBER, SO THE PROB. OF THE
                                ;       RANDOM NUMBER EXCEEDING THE MAXIMUM DESIRED IS LESSENED
                                ;
005102                          PUSH    R1              ; SAVE R1
005102  100461                                                          MOV R1,-(SP)
005103  114001                  CLR     R1              ; ZERO R1
005104  105011          MASKLP: ADD     R1,R1           ; ROTATE R1 ONE BIT LEFT
005105  101201  000001          BIS     #1,R1           ; TURN ON BIT 0
005107  106017                  CMP     R1,R0           ; SEE IF R1 IS GREATER THAN R0
005110                          BCS     MASKLP          ; IF NOT, BRANCH
005110  045112                                                          BCC     .+2
005111  005104                                                          BR      MASKLP
005112  104207  177777          MOV     #-1,R0          ; SET UP FOR COMPLEMENT
005114  103017                  BIC     R1,R0           ; COMPLEMENT R1
005115                          POP     R1              ; RESTORE R1
005115  104261                                                          MOV (SP)+,R1
005116  000000                  RETURN                  ; RETURN TO RNDLBN
```

```
                                        .SBTTL  SEQBE  - FIND NEXT SEQUENTIAL STARTING SECTOR USING BEGIN/END SETS
005117                                  SEQBE:
                                        ;
                                        ;       SEQBE FINDS THE NEXT SEQUENTIAL LBN
                                        ;
005117  104657  000033                  MOV     U.PARM(R5),R0   ; MOVE UNIT PARAMETERS TO R0
005121  102207  010000                  BIT     #DIREC,R0       ; TEST DIRECTION
005123  015223                          BEQ     UP              ; IF SEQUENTIAL UP, BRANCH
005124  102207  004000                  BIT     #NEWSUB,R0      ; SEE IF FIRST TIME ON THIS SUBUNIT
005126  015155                          BEQ     5$              ; IF NOT, BRANCH
005127  103207  004000                  BIC     #NEWSUB,R0      ; FLAG AS NEW SUBUNIT UNNEDED
005131  100657  000033                  MOV     R0,U.PARM(R5)   ; SAVE
005133  104201  000011                  MOV     #S.BESS,R1      ; R0 WILL POINT TO BEGIN/END SETS
005135  105041                          ADD     R4,R1           ; R0 POINTS TO BEGIN/END SETS
005136  104617  000003          6$:     MOV     3(R1),R0        ; SEE IF END-OF-LIST
005140  075144                          BMI     8$              ; IF SO, BRANCH
005141  105201  000004                  ADD     #4,R1           ; POINT TO NEXT BEGIN/END SET
005143  005136                          BR      6$              ; LOOP
005144  104202  000035          8$:     MOV     #U.MBN,R2       ; R2 WILL POINT TO MASTER BN
005146  105052                          ADD     R5,R2           ; R2 POINTS TO MASTER BN
005147  104217                          MOV     (R1)+,R0        ; GET LO ORDER ENDING SET
005150  100127                          MOV     R0,(R2)         ; SAVE
005151  104217                          MOV     (R1)+,R0        ; GET HI ORDER ENDING SET
005152  100627  000001                  MOV     R0,1(R2)        ; SAVE
005154  005163                          BR      7$              ; BRANCH
005155  025275                  5$:     CALL    FNDBES          ; FIND BEGIN/END SET THAT LBN IS IN
005156  105201  000002                  ADD     #2,R1           ; POINT TO BEGIN SET
005160  021645                          CALL    CMP2            ; SEE IF LBN = BEGINNING LBN
005161  015203                          BEQ     BESDWN          ; IF SO, BRANCH
005162  025445                          CALL    DEC2            ; POINT TO NEXT LBN TO BE READ/WRITTEN
005163  104227                  7$:     MOV     (R2)+,R0        ; MOVE LO ORDER BN TO R0
005164  104223                          MOV     (R2)+,R3        ; MOVE HI ORDER BN TO R3
005165  107217                          SUB     (R1)+,R0        ; SUBTRACT BEGIN BN FROM BN
005166  075176                          BMI     4$              ; IF RESULT IS NEGATIVE, BRANCH
005167  045171                          BCC     1$              ; IF NO BORROW, BRANCH
005170  117403                          DEC     R3              ; PROPOGATE BORROW
005171  104111                  1$:     MOV     (R1),R1         ; GET BEGINNING BN
005172  103201  170000                  BIC     #^CHBHINB,R1    ; CLEAR EOL FLAG, IF ANY
005174  107013                          SUB     R1,R3           ; SUBTRACT HI ORDER BN FROM HI BN
005175  015200                          BEQ     2$              ; IF EQUAL, BRANCH
005176  104207  077776          4$:     MOV     #77776,R0       ; MOVE MAXIMUM COUNT TO R0
005200  115407                  2$:     INC     R0              ; MAKE SUBTRACTION INCLUSIVE
005201  026534                  3$:     CALL    MAXMUM          ; SET MAXIMUM COUNT (U.TSEC AND J.MSEC)
005202  005204                          BR      NLBEXT          ; BRANCH
005203  025314          BESDWN: CALL    PREVBE          ; FIND PREVIOUS B/E SET OR UNIT
005204  104651  000020      NLBEXT: MOV     U.TSEC(R5),R1   ; GET TOTAL NUMBER OF SECTORES TO READ/WRITE
005206  117401                          DEC     R1              ; DECREMENT SECTOR COUNT
005207  104653  000035                  MOV     U.MBN(R5),R3    ; GET LO ORDER MASTER BN
005211  107013                          SUB     R1,R3           ; SUBTRACT SECTORS TO READ/WRITE
005212  100653  000035                  MOV     R3,U.MBN(R5)    ; SAVE
005214  045222                          BCC     1$              ; IF NO CARRY, BRANCH
005215  104653  000036                  MOV     U.MBN+1(R5),R3  ; GET HI ORDER MASTER BN
005217  117403                          DEC     R3              ; PROPOGATE CARRY
005220  100653  000036                  MOV     R3,U.MBN+1(R5)  ; SAVE
005222  005274                  1$:     BR      BEEXT           ; BRANCH
005223  102207  004000          UP:     BIT     #NEWSUB,R0      ; SEE IF FIRST TIME ON THIS SUBUNIT
005225  015252                          BEQ     5$              ; IF NOT, BRANCH
```

```
005226  103207  004000            BIC     #NEWSUB,R0        ; FLAG AS NEW SUBUNIT UNNEDED
005230  100657  000033            MOV     R0,U.PARM(R5)     ; SAVE
005232  104201  000011            MOV     #S.BESS,R1        ; R0 WILL POINT TO BEGIN/END SETS
005234  105041                    ADD     R4,R1             ; R0 POINTS TO BEGIN/END SETS
005235  104202  000035            MOV     #U.MBN,R2         ; R2 WILL POINT TO MASTER BN
005237  105052                    ADD     R5,R2             ; R2 POINTS TO MASTER BN
005240  104617  000002            MOV     2(R1),R0          ; GET LO ORDER STARTING SET
005242  100127                    MOV     R0,(R2)           ; SAVE
005243  104617  000003            MOV     3(R1),R0          ; GET HI ORDER STARTING SET
005245  103207  170000            BIC     #^CHBHINB,R0      ; CLEAR EOL FLAG IF ANY
005247  100627  000001            MOV     R0,1(R2)          ; SAVE
005251  005256                    BR      7$                ; BRANCH
005252  025407            5$:     CALL    UPLBN             ; INCREMENT LBN
005253  025275                    CALL    FNDBES            ; FIND BEGIN/END SET THAT LBN IS IN
005254  015273                    BEQ     BESUP             ; IF LBN = ENDING LBN, BRANCH
005255  025430                    CALL    INC2              ; POINT TO NEXT LBN TO BE READ/WRITTEN
005256  104217            7$:     MOV     (R1)+,R0          ; R0 HAS LO ENDING LBN
005257  104113                    MOV     (R1),R3           ; R3 HAS HI ENDING LBN
005260  107227                    SUB     (R2)+,R0          ; SUBTRACT LO LBN FROM LO ENDING LBN
005261  075266                    BMI     4$                ; IF RESULT IS NEGATIVE, BRANCH
005262  045264                    BCC     1$                ; IF NO CARRY, BRANCH
005263  117403                    DEC     R3                ; PROPOGATE CARRY
005264  107123            1$:     SUB     (R2),R3           ; SUBTRACT HI LBN FROM ENDING LBN
005265  015270                    BEQ     2$                ; IF ZERO, BRANCH
005266  104207  077776    4$:     MOV     #77776,R0         ; MOVE MAXIMUM COUNT TO R0
005270  115407            2$:     INC     R0                ; MAKE SUBTRACTION INCLUSIVE
005271  026534            3$:     CALL    MAXMUM            ; SET MAXIMUM COUNT (U.TSEC AND U.MSEC)
005272  005274                    BR      BEEXT             ; BRANCH
005273  025350            BESUP:  CALL    NEXTBE            ; FIND NEXT B/E SET OR UNIT
005274  000000            BEEXT:  RETURN
```

FNDBES - FIND THE BEGIN/END SET CURRENT BN RESIDES IN

```
                                        .SBTTL  FNDBES - FIND THE BEGIN/END SET CURRENT BN RESIDES IN
005275                                  FNDBES:
                                        ;
                                        ;       FNDBES FINDS THE BEGIN/END SET THAT THE CURRENT LBN RESIDES IN
                                        ;
005275  104203  177777                          MOV     #-1,R3          ; START COUNTER (ADJUSTED)
005277  104201  000005                          MOV     #S.BESS-4,R1    ; POINT TO B/E SETS
005301  105041                                  ADD     R4,R1           ; POINT TO B/E SETS
005302  104202  000035                          MOV     #U.MBN,R2       ; POINT TO LBN
005304  105052                                  ADD     R5,R2           ; POINT TO LBN
005305  105201  000004          FNDLOP: ADD     #4,R1           ; INCREMENT B/E SET POINTER
005307  115403                          INC     R3              ; INCREMENT COUNT
005310  021645                          CALL    CMP2            ; COMPARE LBN WITH ENDING LBN
005311  015313                          BEQ     OUT0            ; IF = TO, BRANCH
005312  045305                          BCC     FNDLOP          ; IF LBN > ENDING LBN, BRANCH
005313  0000C0                  OUT0:   RETURN                  ; RETURN TO CALLING PROGRAM
```

PREVBE - MOVE TO PREVIOUS BEGIN/END SET

```
                                        .SBTTL  PREVBE - MOVE TO PREVIOUS BEGIN/END SET
005314                                  PREVBE:
                                        ;
                                        ;       PREVBE IS CALLED IF THE LBN'S ARE DECREASING, AND THE LAST LBN
                                        ;       WAS THE FIRST LBN WITHIN A B/E SET.  PREVBE FINDS THE PREVIOUS
                                        ;       B/E SET, OR, IF IT IS THE LAST ON THE SUBUNIT, FINDS THE LAST B/E
                                        ;       SET OF THE PREVIOUS UNIT.  IF IT IS THE FIRST BE SET OF THE FIRST
                                        ;       UNIT, THE DIRECTION OF THE LBN'S ARE REVERSED (THEY BEGIN GOING UP
                                        ;       THE UNIT)
                                        ;
005314  115003                          TST     R3              ; SEE IF ALLREADY ON 1ST B/E SET
005315  055325                          BNE     5$              ; IF NOT, BRANCH
005316  104657  000033                  MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
005320  101207  004000                  BIS     #NEWSUB,R0      ; MARK AS GOING ON TO THE NEXT UNIT
005322  100657  000033                  MOV     R0,U.PARM(R5)   ; SAVE
005324  005347                          BR      6$              ; EXIT
005325  107201  000006          5$:     SUB     #6,R1           ; POINT TO PREVIOUS ENDING SET
005327  025462                          CALL    MOV2            ; MOVE ENDING SET TO LBN
005330  104217                          MOV     (R1)+,R0        ; R0 HAS LO ENDING LBN
005331  104212                          MOV     (R1)+,R2        ; R2 HAS HI ENDING LBN
005332  107217                          SUB     (R1)+,R0        ; SUBTRACT LO BEGINNING LBN FROM LO ENDING
005333  075343                          BMI     4$              ; IF RESULT IS NEGATIVE, BRANCH
005334  045336                          BCC     1$              ; IF NO CARRY, BRANCH
005335  117402                          DEC     R2              ; PROPOGATE CARRY
005336  104111          1$:             MOV     (R1),R1         ; GET BEGINNING BN
005337  103201  170000                  BIC     #^CHBHINB,R1    ; CLEAR EOL FLAG, IF ANY
005341  107012                          SUB     R1,R2           ; SUBTRACT HI ORDER BN FROM HI BN
005342  015345                          BEQ     2$              ; IF ZERO, BRANCH
005343  104207  077776          4$:     MOV     #77776,R0       ; MOVE MAXIMUM COUNT TO R0
005345  115407                  2$:     INC     R0              ; TO MAKE IT AN INCLUSIVE SUBTRACT
005346  026534                  3$:     CALL    MAXMUM          ; SET MAXIMUM COUNT (U.TSEC AND U.MSEC)
005347  000000                  6$:     RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                        .SBTTL  NEXTBE - MOVE TO NEXT BEGIN/END SET
005350                                  NEXTBE:
                                        ;
                                        ;       NEXTBE IS CALLED WHEN THE CURRENT LBN IS THE LAST OF ITS B/E SET.
                                        ;       NEXTBE GOES TO THE NEXT B/E SET FOR THE NEXT LBN.  IF THE LBN IS
                                        ;       THE LAST ON THE SUBUNIT, IT GOES TO THE FIRST B/E SET OF THE NEXT
                                        ;       UNIT.  IF IT IS THE LAST B/E SET ON THE LAST UNIT, THE DIRECTION IS
                                        ;       CHANGED (DOWN), AND THE INITIAL WRITE BIT IS CLEARED IN CASE AN
                                        ;       INITIAL WRITE IS IN  PROGRESS
                                        ;
005350  104617  000003                          MOV     3(R1),R0        ; SEE IF THIS IS LAST B/E SET ON THIS UNIT
005352  035362                                  BPL     1$              ; IF SO, BRANCH
005353  104657  000033                          MOV     U.PARM(R5),R0   ; GET SUBUNIT PARAMETERS
005355  101207  004000                          BIS     #NEWSUB,R0      ; MARK AS GOING ON TO A NEW SUBUNIT
005357  100657  000033                          MOV     R0,U.PARM(R5)   ; SAVE
005361  005406                                  BR      7$              ; EXIT
005362  105201  000006                  1$:     ADD     #6,R1           ; POINT TO NEXT BEGINNING LBN
005364  025462                                  CALL    MOV2            ; MOVE BEGINNING LBN TO LBN
005365  107201  000002                          SUB     #2,R1           ; R1 POINTS TO END SET
005367  104217                                  MOV     (R1)+,R0        ; R0 HAS LO ENDING LBN
005370  104213                                  MOV     (R1)+,R3        ; R3 HAS HI ENDING SET
005371  107217                                  SUB     (R1)+,R0        ; SUBTRACT BEGINNING LBN
005372  075402                                  BMI     5$              ; IF RESULT IS NEGATIVE, BRANCH
005373  045375                                  BCC     2$              ; IF NO CARRY, BRANCH
005374  117403                                  DEC     R3              ; PROPOGATE CARRY
005375  104111                  2$:     MOV     (R1),R1         ; GET BEGINNING BN
005376  103201  170000                          BIC     #^CHBHINB,R1    ; CLEAR EOL FLAG, IF ANY
005400  107013                                  SUB     R1,R3           ; SUBTRACT HI ORDER BN FROM HI BN
005401  015404                                  BEQ     3$              ; IF ZERO, BRANCH
005402  104207  077776                  5$:     MOV     #77776,R0       ; MOVE MAXIMUM COUNT TO R0
005404  115407                          3$:     INC     R0              ; INCREMENT R0 (INCLUSIVE SUBTRACT)
005405  026534                          4$:     CALL    MAXMUM          ; SET MAXIMUM COUNT (U.TSEC AND U.MSEC)
005406  000000                          7$:     RETURN                  ; RETURN TO CALLING PROGRAM
```

UPLBN - IF GOING UP, UPDATE CURRENT BN TO LAST BN READ/WRITTE

```
                                        .SBTTL  UPLBN  - IF GOING UP, UPDATE CURRENT BN TO LAST BN READ/WRITTEN
005407                                  UPLBN:
                                        ;
                                        ;       UPLBN UPDATES U.MBN TO POINT TO THE LAST LBN READ/WRITTEN
                                        ;       (INCREASING LBN)
                                        ;
005407                                          PUSH    <R0,R2>         ; SAVE R0,R2
005407  100467                                                                  MOV R0,-(SP)
005410  100462                                                                  MOV R2,-(SP)
005411  104202  000035                         MOV     #U.MBN,R2       ; R2 POINTS TO LAST LBN
005413  105052                                 ADD     R5,R2           ; R2 POINTS TO LAST LBN
005414  104657  000020                         MOV     U.TSEC(R5),R0   ; GET NUMBER OF SECTORS WRITTEN
005416  117407                                 DEC     R0              ; SUBTRACT ONE
005417  105127                                 ADD     (R2),R0         ; ADD LOW ORDER LBN TO R0
005420  100227                                 MOV     R0,(R2)+        ; SAVE LOW ORDER WORD, POINT TO HIGH ORDER
005421  045425                                 BCC     UPEXT           ; IF NO CARRY, BRANCH
005422  104127                                 MOV     (R2),R0         ; GET HIGH ORDER WORD
005423  115407                                 INC     R0              ; INCREMENT
005424  100127                                 MOV     R0,(R2)         ; SAVE HIGH ORDER WORD
005425                                  UPEXT:  POP     <R2,R0>         ; RESTORE R0,R2
005425  104262                                                                  MOV (SP)+,R2
005426  104267                                                                  MOV (SP)+,R0
005427  000000                                 RETURN                  ; RETURN TO CALLING PROGRAM
```

INC2 - 24 BIT INCREMENT

```
                                .SBTTL  INC2    - 24 BIT INCREMENT
005430                          INC2:
                                ;
                                ;       INC2 INCREMENTS A TWO WORD NUMBER (SAME FMT AS CMP2) BY ONE
                                ;
005430                                  PUSH    R0              ; SAVE R0
005430  100467                                                                  MOV R0,-(SP)
005431  104127                          MOV     (R2),R0         ; MOVE LOW ORDER WORD TO R0
005432  105207  000001                  ADD     #1,R0           ; INCREMENT R0
005434  100127                          MOV     R0,(R2)         ; SAVE R0
005435  045443                          BCC     CARYNO          ; IF NO CARRY, BRANCH
005436  104627  000001                  MOV     1(R2),R0        ; MOVE HIGH ORDER WORD TO R0
005440  115407                          INC     R0              ; INCREMENT R0
005441  100627  000001                  MOV     R0,1(R2)        ; SAVE R0
005443                          CARYNO: POP     R0              ; RESTORE R0
005443  104267                                                                  MOV (SP)+,R0
005444  000000                          RETURN
```

```
                                .SBTTL  DEC2    - 24 BIT DECREMENT
005445                          DEC2:
                                ;
                                ;       DEC2 DECREMENTS A TWO WORD NUMBER (SAME FMT AS CMP2) BY ONE
                                ;
005445                                  PUSH    R0              ; SAVE R0
005445  100467                                                                  MOV R0,-(SP)
005446  104127                          MOV     (R2),R0         ; MOVE LOW ORDER WORD TO R0
005447  107207  000001                  SUB     #1,R0           ; DECREMENT R0
005451  100127                          MOV     R0,(R2)         ; SAVE R0
005452  045460                          BCC     BORONO          ; IF NO BORROW, BRANCH
005453  104627  000001                  MOV     1(R2),R0        ; MOVE HIGH ORDER WORD TO R0
005455  117407                          DEC     R0              ; DECREMENT R0
005456  100627  000001                  MOV     R0,1(R2)        ; SAVE R0
005460                          BORONO: POP     R0              ; RESTORE R0
005460  104267                                                                  MOV (SP)+,R0
005461  000000                          RETURN                  ; RETURN TO CALLING PROGRAM
```

```
MOV2   - 24 BIT MOVE

                                .SBTTL  MOV2    - 24 BIT MOVE
005462                          MOV2:
                                ;
                                ;       MOV2 COPIES A 24 BIT NUMBER POINTED TO BY R1 TO TWO WORDS POINTED
                                ;       TO BY R2.  BITS 31-24 ARE CLEARED BEFORE COPYING
                                ;
005462                          PUSH    R0              ; SAVE R0
005462  100467                                                          MOV R0,-(SP)
005463  104117                  MOV     (R1),R0         ; MOVE SOURCE LOW ORDER WORD TO R0
005464  100127                  MOV     R0,(R2)         ; MOVE R0 TO DESTINATION LOW ORDER WORD
005465  104617  000001          MOV     1(R1),R0        ; MOVE SOURCE HIGH ORDER WORD TO R0
005467  103207  177400          BIC     #HIBYTE,R0      ; STRIP OFF UNUSED BITS
005471  100627  000001          MOV     R0,1(R2)        ; MOVE R0 TO DESTINATION HIGH ORDER WORD
005473                          POP     R0              ; RESTORE R0
005473  104267                                                          MOV (SP)+,R0
005474  0000C0                  RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                .SBTTL  RNDTG  - FIND A RANDOM STARTING SECTOR USING TRACK/GROUP LIMITS
005475                          RNDTG:
                                ;
                                ;       COME UP WITH A RANDOM BN TO TEST USING THE TRACK/GROUP SETS
                                ;
                                ;       FIRST COME UP WITH A RANDOM COUNT <= MAXIMUM COUNT
                                ;
005475  104647  000012                  MOV     S.MCNT+1(R4),R0 ; GET HI ORDER MAX COUNT
005477  015513                          BEQ     2$              ; IF HI ORDER ZERO, BRANCH
005500  025102                          CALL    MASK            ; CREATE MASK FOR RANDOM NUMBER
005501  026475                  1$:     CALL    RANDOM          ; GET RANDOM NUMBER
005502  103071                          BIC     R0,R1           ; STRIP OFF UNUSED BITS
005503  104012                          MOV     R1,R2           ; SAVE IN R2 IF NEEDED
005504  106641  000012                  CMP     S.MCNT+1(R4),R1 ; SEE IF THE MAX IS EXCEEDED
005506                                  BCS     1$              ; IF SO, BRANCH
005506  045510                                                                  BCC     .+2
005507  005501                                                                  BR      1$
005510  015514                          BEQ     3$              ; IF EQUAL, WE HAVE TO FIND THE LO ORDER WORD
005511  026475                          CALL    RANDOM          ; IF NOT EQUAL, ANY LO ORDER WORD WILL DO
005512  005525                          BR      5$              ; BRANCH
005513  114002                  2$:     CLR     R2              ; HI ORDER WORD IS ZERO
005514  104647  000011          3$:     MOV     S.MCNT(R4),R0   ; GET LO ORDER MAXIMUM
005516  025102                          CALL    MASK            ; CREATE MASK
005517  026475                  4$:     CALL    RANDOM          ; GET RANDOM NUMBER
005520  103071                          BIC     R0,R1           ; STRIP OFF UNUSED BITS
005521  106641  000011                  CMP     S.MCNT(R4),R1   ; SEE IF THE MAX EXCEEDED
005523                                  BCS     4$              ; IF SO, BRANCH
005523  045525                                                                  BCC     .+2
005524  005517                                                                  BR      4$
                                ;
                                ;       NOW FIND THE BN ASSOCIATED WITH THAT COUNT
                                ;
005525                          5$:     PUSH    <R5,R4>         ; PUSH THE UNIT AND SUBUNIT POINTERS
005525  100465                                                                  MOV R5,-(SP)
005526  100464                                                                  MOV R4,-(SP)
005527  104205  000013                  MOV     #S.TGOF,R5      ; R5 WILL POINT TO THE TRACK/GROUP OFFSETS
005531  105045                          ADD     R4,R5           ; R5 POINTS TO THE TRACK/GROUP ORIGINAL OFFSET
005532  104257                          MOV     (R5)+,R0        ; GET LO ORDER OFFSET, MOVE TO RUNNING TOTAL
005533  104253                          MOV     (R5)+,R3        ; GET HI ORDER OFFSET, MOVE TO RUNNING TOTAL
005534  107201  000001          6$:     SUB     #1,R1           ; DECREMENT LO COUNT BY ONE
005536  045543                          BCC     7$              ; IF NO BORROW, BRANCH
005537  107202  000001                  SUB     #1,R2           ; DECREMENT HI COUNT BY ONE
005541                                  BCS     9$              ; IF COUNT EXHAUSTED, BRANCH
005541  045543                                                                  BCC     .+2
005542  005555                                                                  BR      9$
005543  104254                  7$:     MOV     (R5)+,R4        ; GET T/G OFFSET
005544  055551                          BNE     8$              ; IF NOT END OF LIST, BRANCH
005545  104205  000015                  MOV     #S.TGSS,R5      ; R5 WILL POINT TO START OF T/G LIST
005547  105165                          ADD     (SP),R5         ; R5 POINTS TO START OF T/G LIST
005550  104254                          MOV     (R5)+,R4        ; GET T/G OFFSET
005551  105047                  8$:     ADD     R4,R0           ; ADD TO RUNNING TOTAL
005552  045534                          BCC     6$              ; IF NO CARRY, BRANCH
005553  115403                          INC     R3              ; PROPOGATE CARRY
005554  005534                          BR      6$              ; BRANCH
005555                          9$:     POP     <R4,R5>         ; RESTORE
005555  104264                                                                  MOV (SP)+,R4
005556  104265                                                                  MOV (SP)+,R5
```

```
                                        ;
                                        ;       NOW FIND A RANDOM OFFSET INTO THE RANDOM TRACK OR GROUP
                                        ;
005557                                  PUSH    R0              ; STORE LO ORDER TOTAL ON STACK
005557  100467                                                                 MOV R0,-(SP)
005560  104642  000004                  MOV     S.TRKL(R4),R2   ; R2 HAS TRACK LENGTH
005562  104647  000000                  MOV     S.PARM(R4),R0   ; GET SUBUNIT PARAMETERS
005564  102207  000020                  BIT     #TRACKS,R0      ; SEE IF USING TRACKS
005566  055602                          BNE     11$             ; IF SO, BRANCH
005567  104641  000005                  MOV     S.SCHR(R4),R1   ; R1 POINTS TO SUBUNIT PARAMETERS
005571  104611  000003                  MOV     TRKGRP(R1),R1   ; R1 HAS NUMBER OF TRACKS/GROUP
005573  103201  177400                  BIC     #HIBYTE,R1      ; CLEAR UNUSED BITS
005575  114007                          CLR     R0              ; CLEAR RUNNING TOTAL
005576  105027                  10$:    ADD     R2,R0           ; FIND SECTORS/GROUP
005577  117401                          DEC     R1              ; DECREMENT COUNT
005600  055576                          BNE     10$             ; IF UNEXHAUSTED, BRANCH
005601  104072                          MOV     R0,R2           ; MOVE COUNT TO R2 (MATCH WITH TRACKS ABOVE)
005602                          11$:    PUSH    R2              ; SAVE ON STACK
005602  100462                                                                 MOV R2,-(SP)
005603  117402                          DEC     R2              ; ADJUST COUNT
005604  104027                          MOV     R2,R0           ; MOVE TO MASKING REGISTER
005605  025102                          CALL    MASK            ; FIND MASK
005606  026475                  12$:    CALL    RANDOM          ; GET RANDOM NUMBER
005607  103071                          BIC     R0,R1           ; STRIP OFF UNUSED BITS
005610  106021                          CMP     R2,R1           ; SEE IF RANDOM NUMBER SMALL ENOUGH
005611                                  BCS     12$             ; IF NOT, BRANCH
005611  045613                                                                 BCC     .+2
005612  005606                                                                 BR      12$
005613                                  POP     R2              ; GET NUMBER OF SECTORS
005613  104262                                                                 MOV (SP)+,R2
005614  107012                          SUB     R1,R2           ; R2 IS NOW SECTORS REMAINING IN T/G
005615  105261                                                                 ADD     (SP)+,R1
005616  100651  000035                  MOV     R1,U.MBN(R5)    ; SAVE
005620  045622                          BCC     13$             ; IF NO CARRY, BRANCH
005621  115403                          INC     R3              ; PROPOGATE CARRY
005622  100653  000036          13$:    MOV     R3,U.MBN+1(R5)  ; SAVE HI ORDER STARTING BN
                                        ;
                                        ;       NOW COME UP WITH A RANDOM NUMBER OF SECTORS TO R/W WITHOUT GOING
                                        ;       OFF THE TRACK/GROUP THAT WE'RE ON
                                        ;
005624  104027                          MOV     R2,R0           ; R0 IS SECTORS LEFT ON THIS T/G
005625  025102                          CALL    MASK            ; GET MASK
005626  101207  177400                  BIS     #HIBYTE,R0      ; ALLOW MAXIMUM OF 255 SECTORS READ/WRITTEN
005630  026475                  14$:    CALL    RANDOM          ; GET RANDOM NUMBER
005631  103071                          BIC     R0,R1           ; STRIP OFF UNUSED BITS
005632  015630                          BEQ     14$             ; MUST BE GREATER THAN ZERO
005633  106021                          CMP     R2,R1           ; SEE IF SMALL ENOUGH
005634                                  BCS     14$             ; IF NOT, BRANCH
005634  045636                                                                 BCC     .+2
005635  005630                                                                 BR      14$
005636  100651  000020                  MOV     R1,U.TSEC(R5)   ; SAVE SECTORS TO WRITE
005640  000000                          RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                .SBTTL  SEQTG  - FIND NEXT SEQUENTIAL STARTING SECTOR USING TRACK/GROUP LIMITS
005641                          SEQTG:
                                ;
                                ;       FIND THE NEXT TRACK/GROUP TO TEST SEQUENTIALLY
                                ;
005641  104657  000033                  MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
005643  104641  000000                  MOV     S.PARM(R4),R1   ; GET SUBUNIT PARAMETERS
005645  102207  010000                  BIT     #DIREC,R0       ; SEE IF GOING UP (INITIAL WRITE) OR DOWN
005647  015657                          BEQ     3$              ; IF GOING UP OR INITIAL WRITE, BRANCH
005650  102201  000020                  BIT     #TRACKS,R1      ; SEE IF TRACKS OR GROUPS
005652  015655                          BEQ     2$              ; IF GROUPS, BRANCH
005653  025774                          CALL    DOWNT           ; GO DOWN A TRACK
005654  005665                          BR      5$              ; CALCULATE NUMBER OF SECTORS TO READ/WRITE
005655  026136                  2$:     CALL    DOWNG           ; GO DOWN A GROUP
005656  005665                          BR      5$              ; CALCULATE NUMBER OF SECTORS TO READ/WRITE
005657  102201  000020          3$:     BIT     #TRACKS,R1      ; SEE IF TRACKS OR GROUPS
005661  015664                          BEQ     4$              ; IF GROUPS, BRANCH
005662  025715                          CALL    UPT             ; GO UP A TRACK
005663  005665                          BR      5$              ; CALCULATE NUMBER OF SECTORS TO READ/WRITE
005664  026056                  4$:     CALL    UPG             ; GO UP A GROUP
005665  104642  000004          5$:     MOV     S.TRKL(R4),R2   ; GET SECTORS/TRACK
005667  104641  000000                  MOV     S.PARM(R4),R1   ; GET SUBUNIT PARAMETERS
005671  115000  002444                  TST     M.PARM          ; SEE IF INITIAL WRITE IN PROGRESS
005673  035712                          BPL     7$              ; IF NOT, BRANCH
005674  102201  000020                  BIT     #TRACKS,R1      ; SEE IF TRACKS IN USE
005676  055712                          BNE     7$              ; IF SO, BRANCH
005677  104027                          MOV     R2,R0           ; COPY SECTORS/TRACK TO R0
005700  104641  000005                  MOV     S.SCHR(R4),R1   ; GET POINTER TO SUBUNIT CHARACTERISTICS
005702  104611  000003                  MOV     TRKGRP(R1),R1   ; GET TRACKS/GROUP
005704  103201  177400                  BIC     #HIBYTE,R1      ; CLEAR UNUSED BITS
005706  114002                          CLR     R2              ; CLEAR RUNNING TOTAL
005707  105072                  6$:     ADD     R0,R2           ; ADD TO RUNNING TOTAL
005710  117401                          DEC     R1              ; DECREMENT COUNT
005711  055707                          BNE     6$              ; IF INCOMPLETE, BRANCH
005712  100652  000020          7$:     MOV     R2,U.TSEC(R5)   ; SAVE NUMBER OF SECTORS TO PROCESS
005714  000000                  8$:     RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                .SBTTL  UPT     - MOVE UP ONE TRACK
005715                          UPT:
                                ;
                                ;       POINT U.MBN TO THE NEXT TRACK TO TEST (IT DOESN'T HAVE TO BE A
                                ;       TRACK, BUT IF GROUPS, U.MBN MUST BE ADJUSTED BACK TO THE FIRST
                                ;       BN IN THE GROUP
                                ;
005715  102207  004000                BIT     #NEWSUB,R0      ; SEE IF THIS IS A NEW SUBUNIT
005717  015722                        BEQ     10$             ; IF NOT, BRANCH
005720  026172                        CALL    NEWUPT          ; SETUP NEW SUBUNIT
005721  005773                        BR      4$              ; EXIT
005722  104657  000012        10$:    MOV     U.CCNT(R5),R0   ; GET LO ORDER COUNT
005724  107207  000001                SUB     #1,R0           ; DECREMENT
005726  100657  000012                MOV     R0,U.CCNT(R5)   ; SAVE
005730  045747                        BCC     2$              ; IF NO BORROW, BRANCH
005731  104657  000013                MOV     U.CCNT+1(R5),R0 ; GET HI ORDER COUNT
005733  107207  000001                SUB     #1,R0           ; PROPOGATE BORROW
005735  045745                        BCC     1$              ; IF NO BORROW, BRANCH
005736  104657  000033                MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
005740  101207  004000                BIS     #NEWSUB,R0      ; FLAG AS TO GO ONTO NEXT SUBUNIT
005742  100657  000033                MOV     R0,U.PARM(R5)   ; SAVE
005744  005773                        BR      4$              ; EXIT
005745  100657  000013        1$:     MOV     R0,U.CCNT+1(R5) ; SAVE
005747  104657  000014        2$:     MOV     U.PCTG(R5),R0   ; GET POINTER TO CURRENT TRACK/GROUP
005751  104271                        MOV     (R0)+,R1        ; GET OFFSET TO NEXT TRACK
005752  055757                        BNE     3$              ; IF OFFSET, BRANCH
005753  104207  000015                MOV     #S.TGSS,R0      ; R0 WILL POINT TO START OF OFFSETS
005755  105047                        ADD     R4,R0           ; R0 POINTS TO OFFSETS
005756  104271                        MOV     (R0)+,R1        ; R1 HAS OFFSET
005757  100657  000014        3$:     MOV     R0,U.PCTG(R5)   ; SAVE POINTER TO CURRENT TRACK/GROUP
005761  105651  000035                ADD     U.MBN(R5),R1    ; ADD OFFSET TO CURRENT BN
005763  100651  000035                MOV     R1,U.MBN(R5)    ; SAVE
005765  045773                        BCC     4$              ; IF NO CARRY, EXIT
005766  104651  000036                MOV     U.MBN+1(R5),R1  ; GET HI ORDER BN
005770  115401                        INC     R1              ; PROPOGATE CARRY
005771  100651  000036                MOV     R1,U.MBN+1(R5)  ; SAVE
005773  000000        4$:     RETURN                          ; RETURN TO CALLING PROGRAM
```

```
                                        .SBTTL  DOWNT  - MOVE DOWN ONE TRACK
005774                                  DOWNT:
                                        ;
                                        ;       MOVE THE MASTER BN DOWN TO TEST THE PROCEEDING TRACK/GROUP
                                        ;
005774  102207  004000                  10$:    BIT     #NEWSUB,R0      ; SEE IF THIS IS A NEW SUBUNIT
005776  016001                          BEQ     10$                     ; IF NOT, BRANCH
005777  026226                          CALL    NEWDNT                  ; INITIALIZE THIS SUBUNIT
006000  006055                          BR      5$                      ; EXIT
006001  104657  000012                  MOV     U.CCNT(R5),R0           ; GET LO ORDER COUNT
006003  107207  000001                  SUB     #1,R0                   ; DECREMENT COUNT
006005  100657  000012                  MOV     R0,U.CCNT(R5)           ; SAVE
006007  046026                          BCC     2$                      ; IF NO BORROW, BRANCH
006010  104657  000013                  MOV     U.CCNT+1(R5),R0         ; GET HI ORDER COUNT
006012  107207  000001                  SUB     #1,R0                   ; PROPOGATE BORROW
006014  046024                          BCC     1$                      ; IF NO CARRY, BRANCH
006015  104657  000033                  MOV     U.PARM(R5),R0           ; GET UNIT PARAMETERS
006017  101207  004000                  BIS     #NEWSUB,R0              ; FLAG AS TO GO ONTO NEXT SUBUNIT
006021  100657  000033                  MOV     R0,U.PARM(R5)           ; SAVE
006023  006055                          BR      5$                      ; EXIT
006024  100657  000013                  1$:     MOV     R0,U.CCNT+1(R5) ; SAVE
006026  104201  000015                  2$:     MOV     #S.TGSS,R1      ; R0 WILL POINT TO START OF T/G OFFSETS
006030  105041                          ADD     R4,R1                   ; R0 POINTS TO START OF T/G OFFSETS
006031  104657  000014                  MOV     U.PCTG(R5),R0           ; GET POINTER INTO LIST
006033  106071                          CMP     R0,R1                   ; SEE IF AT START OF LIST
006034  056040                          BNE     4$                      ; IF NOT, BRANCH
006035  115407                          3$:     INC     R0              ; R0 POINTS TO NEXT WORD
006036  104171                          MOV     (R0),R1                 ; SEE IF END OF LIST
006037  056035                          BNE     3$                      ; IF NOT, BRANCH
006040  104651  000035                  4$:     MOV     U.MBN(R5),R1    ; GET LO ORDER MASTER BN
006042  107471                          SUB     -(R0),R1                ; SUBTRACT OFFSET
006043  100651  000035                  MOV     R1,U.MBN(R5)            ; SAVE
006045  100657  000014                  MOV     R0,U.PCTG(R5)           ; SAVE POINTER
006047  046055                          BCC     5$                      ; IF NO BORROW, EXIT
006050  104657  000036                  MOV     U.MBN+1(R5),R0          ; GET HI ORDER WORD
006052  117407                          DEC     R0                      ; PROPOGATE BORROW
006053  100657  000036                  MOV     R0,U.MBN+1(R5)          ; SAVE
006055  000000                          5$:     RETURN                  ; RETURN TO CALLING PROGRAM
```

UPG - MOVE UP ONE GROUP

```
                                .SBTTL  UPG     - MOVE UP ONE GROUP
006056                          UPG:
                                ;
                                ;       MOVE MASTER BN UP TO THE NEXT GROUP
                                ;
006056  102207  004000          10$:    BIT     #NEWSUB,R0      ; SEE IF THIS IS A NEW SUBUNIT
006060  016064                  BEQ     10$             ; IF NOT, BRANCH
006061  026172                  CALL    NEWUPT          ; INITILIZE THE SUBUNIT FOR TRACKS
006062  026310                  CALL    SETSEC          ; INITILIZE THE SUBUNIT FOR GROUPS
006063  006135                  BR      5$              ; EXIT
006064  102201  040000          BIT     #INITW,R1       ; SEE IF AN INITIAL WRITE IS BEING DONE
006066  056133                  BNE     4$              ; IF SO, BRANCH
006067  104641  000004          MOV     S.TRKL(R4),R1   ; R1 HAS TRACK LENGTH
006071  104657  000015          MOV     U.CTRK(R5),R0   ; GET TRACK COUNT
006073  117407                  DEC     R0              ; DECREMENT COUNT
006074  016112                  BEQ     1$              ; IF COUNT EXHAUSTED, BRANCH
006075  100657  000015          MOV     R0,U.CTRK(R5)   ; SAVE
006077  105651  000035          ADD     U.MBN(R5),R1    ; MOVE BN TO NEXT TRACK
006101  100651  000035          MOV     R1,U.MBN(R5)    ; SAVE
006103  046135                  BCC     5$              ; EXIT
006104  104657  000036          MOV     U.MBN+1(R5),R0  ; GET HI ORDER BN
006106  115407                  INC     R0              ; PROPOGATE CARRY
006107  100657  000036          MOV     R0,U.MBN+1(R5)  ; SAVE
006111  006135                  BR      5$              ; EXIT
006112  026310                  1$:     CALL    SETSEC          ; SETUP NEXT TRACK/GROUP COUNT
006113  114002                  CLR     R2              ; CLEAR RUNNING TOTAL
006114  117407                  2$:     DEC     R0              ; DECREMENT COUNT
006115  016120                  BEQ     3$              ; IF COUNT EXPIRED, BRANCH
006116  105012                  ADD     R1,R2           ; ADD SECTORS/TRACK TO RUNNING TOTAL
006117  006114                  BR      2$              ; LOOP
006120  104657  000035          3$:     MOV     U.MBN(R5),R0    ; GET LO ORDER MASTER BN
006122  107027                  SUB     R2,R0           ; ADJUST BACK TO START OF GROUP
006123  100657  000035          MOV     R0,U.MBN(R5)    ; SAVE
006125  046133                  BCC     4$              ; IF NO BORROW, BRANCH
006126  104657  000036          MOV     U.MBN+1(R5),R0  ; GET HI ORDER MASTER BN
006130  117407                  DEC     R0              ; PROPOGATE BORROW
006131  100657  000036          MOV     R0,U.MBN+1(R5)  ; SAVE
006133  114007                  4$:     CLR     R0              ; CLEAR R0 SO UPT DOESN'T DETECT A NEW SUBUNIT
006134  025715                  CALL    UPT             ; GO TO NEXT GROUP
006135  000000                  5$:     RETURN          ; RETURN TO CALLING PROGRAM
```

```
                                .SBTTL  DOWNG  - MOVE DOWN ONE GROUP
006136                          DOWNG:
                                ;
                                ;       MOVE MASTER BN TO THE NEXT LOWER GROUP
                                ;
006136  102207  004000                  BIT     #NEWSUB,R0      ; SEE IF THIS IS A NEW SUBIUNIT
006140  016144                          BEQ     10$             ; IF NOT, BRANCH
006141  026226                          CALL    NEWDNT          ; SETUP THIS SUBUNIT FOR TRACKS
006142  026321                          CALL    LSTTRK          ; SETUP THIS SUBUNIT FOR GROUPS
006143  006171                          BR      4$              ; EXIT
006144  104657  000015          10$:    MOV     U.CTRK(R5),R0   ; GET TRACK COUNT
006146  117407                          DEC     R0              ; DECREMENT COUNT
006147  016167                          BEQ     1$              ; IF EXPIRED, BRANCH
006150  100657  000015                  MOV     R0,U.CTRK(R5)   ; SAVE
006152  104657  000035                  MOV     U.MBN(R5),R0    ; GET LO ORDER MASTER BN
006154  107647  000004                  SUB     S.TRKL(R4),R0   ; SUBTRACT TRACK LENGTH
006156  100657  000035                  MOV     R0,U.MBN(R5)    ; SAVE
006160  046171                          BCC     4$              ; EXIT
006161  104657  000036                  MOV     U.MBN+1(R5),R0  ; GET HI MASTER BN
006163  117407                          DEC     R0              ; PROPOGATE BORROW
006164  100657  000036                  MOV     R0,U.MBN+1(R5)  ; SAVE
006166  006171                          BR      4$              ; EXIT
006167  025774                  1$:     CALL    DOWNT           ; GO DOWN A GROUP
006170  026321                          CALL    LSTTRK          ; SET MASTER BN ON LAST TRACK OF GROUP
006171  000000                  4$:     RETURN                  ; RETURN TO CALLING PROGRM
```

```
                                        .SBTTL  NEWUPT - INITILIZE PARAMETERS FOR SEQUENTIALLY UP BY TRACKS
006172                                  NEWUPT:
                                        ;
                                        ;       INITIALIZE THIS SUBUNIT FOR SEQUENTIALLY ACCESSING TRACKS IN
                                        ;       ASCENDING ORDER
                                        ;
006172  104647  000011                          MOV     S.MCNT(R4),R0   ; GET MAXIMUM COUNT
006174  100657  000012                          MOV     R0,U.CCNT(R5)   ; SAVE IN CURRENT COUNT
006176  104647  000012                          MOV     S.MCNT+1(R4),R0 ; GET HI ORDER MAXIMUM COUNT
006200  100657  000013                          MOV     R0,U.CCNT+1(R5) ; SAVE IN HI ORDER CURRENT COUNT
006202  104647  000013                          MOV     S.TGOF(R4),R0   ; GET STARTING OFFSET
006204  100657  000035                          MOV     R0,U.MBN(R5)    ; SAVE IN MASTER BN
006206  104647  000014                          MOV     S.TGOF+1(R4),R0 ; GET HI ORDER STARTING OFFSET
006210  100657  000036                          MOV     R0,U.MBN+1(R5)  ; SAVE IN HI ORDER MASTER BN
006212  104207  000015                          MOV     #S.TGSS,R0      ; R0 WILL POINT TO START OF OFFSETS
006214  105047                                  ADD     R4,R0           ; R0 POINTS TO START OF OFFSETS
006215  100657  000014                          MOV     R0,U.PCTG(R5)   ; SAVE IN POINTER TO CURREN' TRACK/GROUP
006217  104657  000033                          MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
006221  103207  004000                          BIC     #NEWSUB,R0      ; NOW ON NEW UNIT
006223  100657  000033                          MOV     R0,U.PARM(R5)   ; SAVE PARAMETERS
006225  000000                                  RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                .SBTTL  NEWDNT - INITILIZE PARAMETERS FOR SEQUENTIALLY DOWN BY TRACKS
006226                          NEWDNT:
                                ;
                                ;       INITIALIZE THIS SUBUNIT FOR SEQUENTIALLY ACCESSING TRACKS IN
                                ;       DESCENDING ORDER
                                ;
006226                          PUSH    <R5,R4>         ; SAVE UNIT AND SUBUNIT POINTERS
006226  100465                                                          MOV R5,-(SP)
006227  100464                                                          MOV R4,-(SP)
006230  104642  000013          MOV     S.TGOF(R4),R2   ; R2 HAS LO ORDER STARTING OFFSET
006232  104643  000014          MOV     S.TGOF+1(R4),R3 ; R3 HAS HI ORDER STARTING OFFSET
006234  104647  000011          MOV     S.MCNT(R4),R0   ; R0 HAS LO ORDER MAXIMUM COUNT
006236  100657  000012          MOV     R0,U.CCNT(R5)   ; SAVE LO ORDER MAXIMUM COUNT
006240  104641  000012          MOV     S.MCNT+1(R4),R1 ; R1 HAS HI ORDER MAXIMUM COUNT
006242  100651  000013          MOV     R1,U.CCNT+1(R5) ; SAVE HI ORDER MAXIMUM COUNT
006244  105204  000015          ADD     #S.TGSS,R4      ; R4 POINTS TO START OF OFFSETS
006246  107207  000001  1$:     SUB     #1,R0           ; DECREMENT COUNT
006250  046255                  BCC     2$              ; IF NO CARRY, BRANCH
006251  107201  000001          SUB     #1,R1           ; PROPOGATE BORROW
006253                          BCS     4$              ; IF COUNT EXHAUSTED, BRANCH
006253  046255                                                          BCC     .+2
006254  006267                                                          BR      4$
006255  104245  2$:     MOV     (R4)+,R5        ; GET OFFSET
006256  056263          BNE     3$              ; IF NOT END-OF-LIST, BRANCH
006257  104204  000015          MOV     #S.TGSS,R4      ; R4 WILL POINT TO START OF OFFSETS
006261  105164                  ADD     (SP),R4         ; R4 POINTS TO START OF OFFSETS
006262  104245                  MOV     (R4)+,R5        ; GET OFFSET
006263  105052  3$:     ADD     R5,R2           ; ADD OFFSET TO BN
006264  046246                  BCC     1$              ; IF NO CARRY, LOOP
006265  115403                  INC     R3              ; PROPOGATE CARRY
006266  006246                  BR      1$              ; LOOP
006267  104665  000001  4$:     MOV     1(SP),R5        ; GET UNIT POINTER
006271  100654  000014          MOV     R4,U.PCTG(R5)   ; SAVE POINTER TO CURRENT TRACK/GROUP
006273                          POP     <R4,R5>         ; RESTORE
006273  104264                                                          MOV (SP)+,R4
006274  104265                                                          MOV (SP)+,R5
006275  100652  000035          MOV     R2,U.MBN(R5)    ; SAVE LO ORDER STARTING BN
006277  100653  000036          MOV     R3,U.MBN+1(R5)  ; SAVE HI ORDER
006301  104653  000033          MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
006303  103203  004000          BIC     #NEWSUB,R3      ; NOW ON NEW UNIT
006305  100653  000033          MOV     R3,U.PARM(R5)   ; SAVE PARAMETERS
006307  000000                  RETURN                  ; RETURN TO CALLING PROGRAM
```

```
      1                                 .SBTTL  SETSEC - SETUP TRACK/GROUP COUNT FOR SELECTED GROUPS
      2 006310                          SETSEC:
      3                                 ;
      4                                 ;       MOVE TRACKS/GROUP TO U.CTRK
      5                                 ;
      6 006310  104647  000005                 MOV     S.SCHR(R4),R0   ; R0 POINTS TO SUBUNIT CHARACTERISTICS
      7 006312  104677  000003                 MOV     TRKGRP(R0),R0   ; R0 HAS TRACKS/GROUPS
      8 006314  103207  177400                 BIC     #HIBYTE,R0      ; CLEAR UNUSED BITS
      9 006316  100657  000015                 MOV     R0,U.CTRK(R5)   ; SAVE AS TRACK COUNT
     10 006320  000000                         RETURN                  ; RETURN TO CALLING MODULE
```

LSTTRK - SET MASTER BN TO POINT TO LAST TRACK IN THE GROUP

```
                                        .SBTTL  LSTTRK - SET MASTER BN TO POINT TO LAST TRACK IN THE GROUP
006321                                  LSTTRK:
                                        ;
                                        ;       SET MBN TO LAST TRACK OF GROUP
                                        ;
006321  026310                          CALL    SETSEC          ; SETUP TRACK/GROUP COUNT
006322  114002                          CLR     R2              ; CLEAR RUNNING TOTAL
006323  117407                  2$:     DEC     R0              ; DECREMENT COUNT
006324  016330                          BEQ     3$              ; IF COUNT EXPIRED, BRANCH
006325  105642  000004                  ADD     S.TRKL(R4),R2   ; ADD SECTORS/TRACK TO RUNNING LENGTH
006327  006323                          BR      2$              ; LOOP
006330  105652  000035          3$:     ADD     U.MBN(R5),R2    ; MBN WILL POINT TO LAST TRACK IN GROUP
006332  100652  000035                  MOV     R2,U.MBN(R5)    ; SAVE
006334  046340                          BCC     4$              ; IF NO CARRY, EXIT
006335  104657  000036                  MOV     U.MBN+1(R5),R0  ; GET HI ORDER BN
006337  115407                          INC     R0              ; PROPOGATE CARRY
006340  100657  000036          4$:     MOV     R0,U.MBN+1(R5)  ; SAVE
006342  000000                          RETURN                  ; RETURN TO CALLING MODULE
```

```
                                        .SBTTL  CLRUP  - CLEAR ALL PARAMETER BITS
006343                                  CLRUP:
                                        ;
                                        ;       CLRUP CLEARS ALL NECESSARY FLAG BITS
                                        ;
006343  104657  000033                          MOV     U.PARM(R5),R0   ; MOVE UNIT PARAMETERS TO R0
                                        ;       CLEAR ALL FLAGS BEFORE THE NEXT OPERATION
006345  103207  004776                          BIC     #DATERR.RBNBN!REVEC.REVINP.WCHECK!DATCMP'REDWRT.RETRY.NEWSUB,R0
006347  100657  000033                          MOV     R0,U.PARM(R5)   ; SAVE UNIT PARAMETERS
006351  000000                                  RETURN
```

RORW - DETERMINE IF A READ OR A WRITE IS TO BE DONE

```
                                        .SBTTL  RORW   - DETERMINE IF A READ OR A WRITE IS TO BE DONE
006352                                  RORW:
                                        ;
                                        ;       RORW DETERMINES IF THE BLOCK SELECTED WILL BE READ OR WRITTEN
                                        ;
006352  104642  000000                          MOV     S.PARM(R4),R2   ; GET SUBUNIT PARAMETERS
006354  104657  000033                          MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
006356  102202  042000                          BIT     #INITW.WONLY,R2 ; SEE IF INITIAL WRITE OR WRITE ONLY
006360  056367                                  BNE     1$              ; IF SO, BRANCH
006361  102202  004000                          BIT     #RONLY,R2       ; SEE IF READ ONLY
006363  056373                                  BNE     2$              ; IF SO, BRANCH
006364  026475                                  CALL    RANDOM          ; GET RANDOM NUMBER
006365  110601                                  ROR     R1              ; ROTATE LOW BIT INTO CARRY
006366  046373                                  BCC     2$              ; IF LOW BIT ZERO, BRANCH
006367  101207  000100                  1$:     BIS     #REDWRT,R0      ; SET READ OR WRITE BIT (WRITE)
006371  100657  000033                          MOV     R0,U.PARM(R5)   ; SAVE UNIT PARAMETERS
006373  000000                          2$:     RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                        .SBTTL  PATTRN - IF WRITE, DETERMINE WHAT PATTERN IS TO BE WRITTEN
006374                                  PATTRN:
                                        ;
                                        ;       FIND PATTERN TO USE (ONLY IF WRITE TO BE DONE)
                                        ;
006374  104653  000033                          MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
006376  102203  000100                          BIT     #REDWRT,R3      ; SEE IF WRITE TO BE DONE
006400  016415                                  BEQ     PATEXT          ; IF NOT, BRANCH
006401  104641  000003                          MOV     S.PAT(R4),R1    ; GET PATTERN NUMBER
006403  016407                                  BEQ     RNDO            ; IF PATTERN NOT SELECTED, BRANCH
006404  103201  177760                          BIC     #LBLONB,R1      ; SEE IF SPECIAL PATTERN USED
006406  006413                                  BR      FIXPAT          ; IF PATTERN SELECTED, USE IT
006407  026475                          RNDO:   CALL    RANDOM          ; GET RANDOM PATTERN NUMBER
006410  103201  177760                          BIC     #LBLONB,R1      ; CLEAR UNUSED BITS
006412  016407                                  BEQ     RNDO            ; NO SUCH THING AS PATTERN 0, TRY AGAIN
006413  100651  000011                  FIXPAT: MOV     R1,U.PAT(R5)    ; SAVE THE PATTERN NUMBER
006415  000000                          PATEXT: RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                .SBTTL  WCHK   - IF WRITE, SEE IF A WRITE CHECK IS TO BE DONE
006416                          WCHK:
                                ;
                                ;       WCHK SETS THE WRITE CHECK BIT IF A WRITE CHECK IS TO BE DONE
                                ;
006416  104653  000033                  MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
006420  104642  000000                  MOV     S.PARM(R4),R2   ; GET SUBUNIT PARAMETERS
006422  102203  000100                  BIT     #REDWRT,R3      ; SEE IF WRITE IS TO BE DONE
006424  016445                          BEQ     2$              ; IF NOT, BRANCH
006425  102202  000010                  BIT     #WCHECK,R2      ; SEE IF WRITE CHECK IS TO BE DONE
006427  016445                          BEQ     2$              ; IF NOT, BRANCH
006430  102202  040000                  BIT     #INITW,R2       ; SEE IF INITIAL WRITE IS IN PROGRESS
006432  056445                          BNE     2$              ; IF SO, NO WRITE CHECK, SO BRANCH
006433  102202  000004                  BIT     #WCHKAL,R2      ; SEE IF WRITE CHECK IS ALWAYS TO BE DONE
006435  056441                          BNE     1$              ; IF SO, BRANCH
006436  026475                          CALL    RANDOM          ; GET RANDOM NUMBER
006437  110601                          ROR     R1              ; 1/2 OF THE TIME WRITE CHECK
006440  046445                          BCC     2$              ; BRANCH (NO WRITE CHECK) IF LOWEST BIT CLEAR
006441  101203  000010          1$:     BIS     #WCHECK,R3      ; SET WRITE CHECK BIT
006443  100653  000033                  MOV     R3,U.PARM(R5)   ; SAVE UNIT PARAMETERS
006445  000000                  2$:     RETURN                  ; RETURN TO CALLING PROGRAM
```

DCOMP - IF READ, SEE IF DATA COMPARE IS TO BE DONE

```
      1                                    .SBTT   DCOMP  - IF READ, SEE IF DATA COMPARE IS TO BE DONE
      2 006446                             DCOMP:
      3                                    ;
      4                                    ;       DCOMP SETS THE DATCMP BIT IF A DATA COMPARE IS TO BE DONE
      5                                    ;
      6 006446  104642  000000                     MOV     S.PARM(R4),R2   ; GET SUBUNIT PARAMETRS
      7 006450  102202  000002                     BIT     #DATCMP,R2      ; SEE IF DATA COMPARE REQUESTED
      8 006452  016474                             BEQ     DCEXT           ; IF NOT, BRANCH
      9 006453  104653  000033                     MOV     U.PARM(R5),R3   ; GET UNIIT PARAMETERS
     10 006455  102203  000100                     BIT     #REDWRT,R3      ; SEE IF READ IS TO BE DONE
     11 006457  016463                             BEQ     DCREAD          ; IF SO, BRANCH
     12 006460  102203  000010                     BIT     #WCHECK,R3      ; SEE IF WRITE CHECK IS TO BE DONE
     13 006462  016474                             BEQ     DCEXT           ; IF NOT, BRANCH
     14 006463  102202  000001           DCREAD:   BIT     #DCMPAL,R2      ; SEE IF DATA COMPARE ALWAYS DONE
     15 006465  056471                             BNE     DCMPYS          ; IF SO, BRANCH
     16 006466  026475                             CALL    RANDOM          ; GET RANDOM NUMBER
     17 006467  103201  177775                     BIC     #^CDATCMP,R1    ; CLEAR ALL BUT THE (POSSIBLY) DATA COMPARE FLAG
     18 006471  101013                    DCMPYS:  BIS     R1,R3           ; SET DATA COMPARE BIT
     19 006472  100653  000033            DCOUT:   MOV     R3,U.PARM(R5)   ; SAVE UNIT PARAMETERS
     20 006474  000000                    DCEXT:   RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                .SBTTL  RANDOM - RANDOM NUMBER ROUTINE
006475                          RANDOM:
                                ;
                                ;       RANDOM CALCULATES A RANDOM NUMBER AND RETURNS IT IN R1
                                ;       RANGE 0 - 2^16-1
                                ;
006475                                  PUSH    <R0,R2>         ; SAVE R0 AND R2
006475  100467                                                                  MOV R0,-(SP)
006476  100462                                                                  MOV R2,-(SP)
006477  104307  002445                 MOV     LOSEED,R0       ; MOVE LO ORDER SEED TO R0
006501  104301  002446                 MOV     HISEED,R1       ; MOVE HI SEED TO R1
006503  104202  000007                 MOV     #7,R2           ; MOVE LOOP COUNT TO R2
006505  110207                  1$:     ROL     R0              ; ROTATE LO ORDER NUMBER BY 1
006506  110201                          ROL     R1              ; ROTATE HI OREDR NUMBER BY 1 (PROPOGATE CARRY)
006507  103207  000001                 BIC     #1,R0           ; CLEAR LOWER BIT
006511  117402                         DEC     R2              ; DECREMENT COUNT
006512  056505                         BNE     1$              ; IF COUNT INCOMPLETE, BRANCH
006513  105307  002445                 ADD     LOSEED,R0       ; ADD ORIGINAL SEED (X129)
006515  046517                         BCC     2$              ; IF NO CARRY, BRANCH
006516  115401                         INC     R1              ; PROPOGATE CARRY
006517  105207  001057          2$:    ADD     #1057,R0        ; ADD LO CONSTANT
006521  046523                         BCC     3$              ; IF NO CARRY, BRANCH
006522  115401                         INC     R1              ; PROPOGATE CARRY
006523  105201  047401          3$:    ADD     #47401,R1       ; ADD HI CONSTANT
006525  104070  002445                 MOV     R0,LOSEED       ; SAVE LO ORDER SEED
006527  104010  002446                 MOV     R1,HISEED       ; SAVE HI ORDER SEED
006531                                  POP     <R2,R0>         ; RESTORE R2 AND R0
006531  104262                                                                  MOV (SP)+,R2
006532  104267                                                                  MOV (SP)+,R0
006533  000000                          RETURN
```

```
                                        .SBTTL  MAXMUM - FOR SEQUENTIAL B/E SETS, DETERMINE HOW MANY SECTORS TO READ/WRITE
006534                                  MAXMUM:
                                        ;
                                        ;       MAXMUM WILL COME UP WITH A MAXIMUM NUMBER OF SECTORS TO READ/WRITE
                                        ;       AND INITILIZE BOTH U.TSEC AND U.MSEC TO THIS VALUE.  IF AN INITIAL
                                        ;       WRITE IS IN PROGRESS, AS MANY SECTORS WILL BE READ/WRITTEN AS
                                        ;       POSSIBLE  R0 HAS MAXIMUM POSSIBLE SECTORS BY LIMITATION OF B/E SETS
                                        ;
006534                                          PUSH    <R1,R3>         ; SAVE R1, R3
006534  100461                                                                  MOV R1,-(SP)
006535  100463                                                                  MOV R3,-(SP)
006536  104643  000000                         MOV     S.PARM(R4),R3   ; GET SUBUNIT PARAMETERS
006540  102203  040100                         BIT     #INITW.SEQSEK,R3        ; SEE IF INITIAL WRITE IS IN PROGRESS
006542  056552                                 BNE     2$              ; IF SO, WRITE THE MAXIMUM POSSIBLE SECTORS
006543  026475                          1$:    CALL    RANDOM          ; GET A RANDOM NUMBER OF SECTORS TO READ/WRITE
006544  103201  177400                         BIC     #HIBYTE,R1      ; READ/WRITE A MAXIMUM OF 255 SECTORS
006546  016543                                 BEQ     1$              ; IF ZERO, GET ANOTHER RANDOM NUMBER
006547  106071                                 CMP     R0,R1           ; SEE IF RANDOM NUMBER EXCEEDS POSSIBLE
006550  076552                                 BMI     2$              ; IF SO, BRANCH
006551  104017                                 MOV     R1,R0           ; ONLY READ/WRITE THE RANDOM NUMBER OF SECTORS
006552  100657  000020                  2$:    MOV     R0,U.TSEC(R5)   ; SAVE IN TSEC
006554                                         POP     <R3,R1>         ; RESTORE R3, R1
006554  104263                                                                  MOV (SP)+,R3
006555  104261                                                                  MOV (SP)+,R1
006556  000000                                 RETURN                  ; RETURN TO CALLING PROGRAM
006560                                  AREA1   =       .+1
```

```
                                .SBTTL  ********** OVERLAY 2 - IF NECESSARY, ISSUE SEEK
                                        DMOVLY  SK,AREA1
006557  034666                          .WREDC          ;OUTPUT EDC FOR TH'S OVERLAY
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;************************************* ******* *********************************
                                ;
                                ;
000002                          SEEK    =       2.              ; SEEK OVERLAY
                                ;
                                ;       SEEK ISSUES A SEEK TO THE CYLINDER SPECIFIED IN U.CCYL(R5) (2 WORDS)
                                ;       AND GROUP GIVEN IN U.CGRP(R5) (1 WORD).
                                ;       IT THEN RETURNS WITH A DEFERED CALL TO SEKTST (TEST SEEK) TO SEE IF
                                ;       THE SEEK COMPLETED SUCCESSFULLY.  IF SO, THE READ WRITE MODULE IS
                                ;       THEN CALLED
                                ;
006560                          DIAG$$                          ; ************
006560  115000  002767                  TST     $$DIAG+$DIAG$
006562  016570                          BEQ     .+6
006563  104207  060000                  MOV     #60000,R0
006565  100707  174200                  MOV     R0,@$$DIAG+$DIAG$
006567  006570                          BR      .+1
006570  026625                          CALL    TSTNEC          ; SEE IF SEEK IS NECESSARY
                                ;
                                ;       IT SEEMS TO ME THAT IF THE PROGRAM IS CORRECT, I'LL NEVER GET
                                ;       THIS ERROR.  TO REDUCE CONFUSION FOR THE TIME BEING, I'LL ASSUME
                                ;       THAT THIS ERROR DOESN'T OCCUR
                                ;
                                ;       MOV     #SETUP,R0       ; IF ERROR, SETUP IS NEXT MODULE *******
                                ;       MOV     R0,R1           ; MAKE R1 NON-ZERO FOR DEFERRED CALL IF ERROR
                                ;       TST     R2              ; SEE IF ERROR OCCURRED
                                ;       BNE     SEKEXT          ; IF SO, BRANCH
                                ;
006571  114002                          CLR     R2              ; FAKE AS NO ERROR (I THINK IT'S REDUNDANT)
006572  115003                          TST     R3              ; SEE IF SEEK IS NECESSARY
006573  016616                          BEQ     NOSEEK          ; IF ZERO (NO SEEK NECESSARY), BRANCH
006574  026677                          CALL    ISUSEK          ; ISSUE SEEK
006575  115002                          TST     R2              ; SEE IF ERROR OCCURRED
006576  056621                          BNE     AFTERR          ; IF ERROR, BRANCH
006577  104657  000033                  MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
006601  101207  002000                  BIS     #SEKINP,R0      ; FLAG AS SEEK IN PROGRESS
006603  100657  000033                  MOV     R0,U.PARM(R5)   ; SAVE
006605  104207  003720                  MOV     #2000.,R0       ; MOVE TIMEOUT TO PARAMETERS
006607  100657  000005                  MOV     R0,U.TIMO(R5)   ; INITILIZE TIMEOUT VALUE
006611  104201  000001                  MOV     #1,R1           ; DEFERRED CALL TO SEKTST
006613  104207  000003                  MOV     #SEKTST,R0      ; SEKTST NEXT TO CALL
006615  006624                          BR      SEKEXT          ; BRANCH
006616  104207  004501          NOSEEK: MOV     #REDSET,R0      ; READ/WRITE ROUTINE NEXT
006620  006623                          BR      SEKOUT          ; BRANCH
006621  104207  000002          AFTERR: MOV     #SEEK,R0        ; SEEK IS NEXT MODULE CALLED ******
006623  114001                  SEKOUT: CLR     R1              ; IMMIDATE CALL
006624  003127                  SEKEXT: BR      JMPRET          ; RETURN TO CALLING PROGRAM
```

```
                                        .SBTTL  TSTNEC - TEST TO SEE IF SEEK IS NECESSARY
006625                                  TSTNEC:
                                        ;
                                        ;       TSTNEC TESTS TO SEE IF A SEEK IS NECESSARY.  IF NOT, R3 IS RETURNED
                                        ;       AS ZERO, NONZERO OTHERWISE.
                                        ;
006625  104657  000033                          MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
006627  102207  000200                          BIT     #RBNBN,R0       ; SEE IF BLOCK REVECTORED
006631  016641                                  BEQ     2$              ; IF NOT, BRANCH
006632  104650  000041  002340                  MOV     U.RBN(R5),CURBN ; MOVE RBN TO CALCULATION AREA
006635  104650  000042  002341                  MOV     U.RBN+1(R5),CURBN+1     ; MOVE RBN TO CALCULATION AREA
006640  006647                                  BR      3$              ; BRANCH
006641  104650  000037  002340  2$:             MOV     U.CBN(R5),CURBN ; MOVE LBN TO CALCULATION AREA
006644  104650  000040  002341                  MOV     U.CBN+1(R5),CURBN+1     ; MOVE LBN TO CALCULATION AREA
006647  114007                  3$:             CLR     R0              ; TELL CALC ROUTINE TO SET UP PARAMETERS
006650  022141                                  CALL    CALC            ; CALCULATE CYL AND GROUP
006651  115002                                  TST     R2              ; SEE IF AN ERROR OCCURRED
006652  056676                                  BNE     10$             ; IF SO, BRANCH
006653  104203  002344                          MOV     #CYL,R3         ; R3 POINTS TO CALCULATED CYLINDER
006655  104657  000033                          MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
006657  102207  020000                          BIT     #RESEEK,R0      ; SEE IF SEEK MANDATORY
006661  056675                                  BNE     9$              ; IF SO, BRANCH
006662  104202  000053                          MOV     #U.CCYL,R2      ; R2 WILL POINT TO CURRENT CYL
006664  105052                                  ADD     R5,R2           ; R2 POINTS TO CURRENT CYLINDER
006665  104201  000003                          MOV     #3,R1           ; GET COUNT
006667  104237                  4$:             MOV     (R3)+,R0        ; GET WORD
006670  106227                                  CMP     (R2)+,R0        ; SEE IF CYL AND GROUP THE SAME
006671  056675                                  BNE     9$              ; IF NOT, BRANCH
006672  117401                                  DEC     R1              ; DECREMENT COUNT
006673  056667                                  BNE     4$              ; IF COUNT INCOMPLETE, BRANCH
006674  114003                                  CLR     R3              ; FLAG AS SEEK NOT NECESSARY
006675  114002                  9$:             CLR     R2              ; FLAG AS NO ERRORS
006676  000000                  10$:            RETURN                  ; RETURN TO SEEK
```

```
                                .SBTTL  ISUSEK - ISSUE SEEK COMMAND
006677                          ISUSEK:
                                ;
                                ;       ISUSEK ISSUES THE SEEK TO THE DRIVE
                                ;
006677  104300  002344  001561          MOV     CYL,INS+1       ; MOVE LOW CYL TO SEEK COMMAND
006702  104300  002345  001562          MOV     CYL+1,INS+2     ; SET LOWER BITS
006705  104300  002346  001563          MOV     GROUP,INS+3     ; MOVE GROUP TO SEEK COMMAND
006710  104203  001506                  MOV     #CR.SEK,R3      ; SET UP FOR TALK
006712  021204                          CALL    TALK            ; SEND SEEK
006713  115002                          TST     R2              ; SEE IF ERROR OCCURRED
006714  056744                          BNE     10$             ; IF SO, BRANCH
006715  104207  000053                  MOV     #U.CCYL,R0      ; R0 WILL POINT TO CURRENT CYLINDER
006717  105057                          ADD     R5,R0           ; R0 POINTS TO CURRENT CYLINDER
006720  104201  000003                  MOV     #3,R1           ; MOVE THREE WORDS
006722                                  PUSH    <R0,R1>         ; SAVE POINTER AND COUNT
006722  100467                                                                  MOV R0,-(SP)
006723  100461                                                                  MOV R1,-(SP)
006724  104202  000056                  MOV     #U.LCYL,R2      ; R2 WILL POINT TO LAST CYLINDER
006726  105052                          ADD     R5,R2           ; R2 POINTS TO LAST CYLINDER
006727  104273                  1$:     MOV     (R0)+,R3        ; GET WORD
006730  100223                          MOV     R3,(R2)+        ; SAVE
006731  117401                          DEC     R1              ; DECREMENT COUNT
006732  056727                          BNE     1$              ; IF COUNT INCOMPLETE, BRANCH
006733                                  POP     <R2,R0>         ; RESTORE
006733  104262                                                                  MOV (SP)+,R2
006734  104267                                                                  MOV (SP)+,R0
006735  104201  002344          4$:     MOV     #CYL,R1         ; R2 POINTS TO NEW CYL
006737  104213                  2$:     MOV     (R1)+,R3        ; GET WORD
006740  100273                          MOV     R3,(R0)+        ; SAVE
006741  117402                          DEC     R2              ; DECREMENT COUNT
006742  056737                          BNE     2$              ; IF COUNT INCOMPLETE, BRANCH
006743  007023                          BR      ISUEXT          ; BRANCH (NOTE THAT R2 IS ZERO - NO ERRORS)
006744  106207  000175          10$:    CMP     #175,R0         ; SEE IF UNSUCCESSFUL RESPONSE WAS RECIEVED
006746  057020                          BNE     TMPERR          ; IF NOT, BRANCH
006747                                  SOFTER  27,<CURBN,CURBN+1,TRACK,GROUP,CYL,CYL+1>
006747  104200  003617  001012                                          MOV     #ER27,OUT.04
006752  104300  002340  001013                                          MOV     CURBN,OUT.05
006755  104300  002341  001014                                          MOV     CURBN+1,OUT.06
006760  104300  002347  001015                                          MOV     TRACK,OUT.07
006763  104300  002346  001016                                          MOV     GROUP,OUT.08
006766  104300  002344  001017                                          MOV     CYL,OUT.09
006771  104300  002345  001020                                          MOV     CYL+1,OUT.10
006774  104200  000033  001010                                          MOV     #27,OUT.02
006777  101200  001400  001010                                          BIS     #ERSOFT,OUT.02
007002  104200  007002  001007                                          MOV     #.,OUT.01
007005  104202  000013                                                  MOV     #ERRMES,R2
007007  104020  001006                                                  MOV     R2,OUT.RQ
007011  103200  077777  002357          BIC     #77777,ERRPOS   ; CLEAR PREVIOUS ERRPOS SETTING
007014                                  ENDERR
007014  101200  000013  002357                                          BIS     #13,ERRPOS      ; SET THE POSITION
007017  007023                          BR      ISUEXT          ; BRANCH
007020                          TMPERR: CERROR  5,#SER9         ; REPORT SECONDARY ERROR
007020  104200  010345  001013                                          MOV     #SER9,OUT.05
007023  000000                  ISUEXT: RETURN                  ; RETURN TO CALLING PROGRAM
        007025                  AREA2   =       .+1
```

```
********** OVERLAY 3 - SEE IF THE SEEK IS COMPLETE

                                .SBTTL  ********** OVERLAY 3 - SEE IF THE SEEK IS COMPLETE
007024                                  DMOVLY  TS,AREA2
007024  123711                          .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*****************************************************************************
                                ;*************************************** *************************************
                                ;
                                ;
                                ;
                        000003  SEKTST  -       3.              ; SEEK COMPLETE TEST OVERLAY
                                ;
                                ;       SEKTST TESTS TO SEE IF READ/WRITE READY OR ATTENTION IS HIGH
                                ;       R2 RETURNED AS ZERO IF SEEK WAS SUCESSFUL, -1 IF SEEK INCOMPLETE
                                ;       POSITIVE NON-ZERO IF AN ERROR WAS DETECTED
                                ;
007025                                  DIAG$$                  ; ************
007025  115000  002770                  TST     $$DIAG+$DIAG$
007027  017035                          BEQ     .+6
007030  104207  060000                  MOV     #60000,R0
007032  100707  173734                  MOV     R0,@$$DIAG+$DIAG$
007034  007035                          BR      .+1
007035  021173                          CALL    RTDS            ; GET REAL TIME  DRIVE STATE
007036  115002                          TST     R2              ; SEE IF ERROR OCCURRED
007037  057160                          BNE     RTDSE           ; IF SO, BRANCH
007040  102201  000002                  BIT     #ATTN,R1        ; SEE IF ATTENTION IS HIGH
007042  057226                          BNE     ATNHI           ; IF ATTENTION HIGH, BRANCH
007043  102201  100000                  BIT     #RWRDY,R1       ; SEE IF READ/WRITE READY
007045  017102                          BEQ     STSERR          ; IF NOT, BRANCH
007046  104657  000033                  MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
007050  103207  022000                  BIC     #RESEEK!SEKINP,R0       ; CLEAR RESEEK BIT
007052  100657  000033                  MOV     R0,U.PARM(R5)   ; SAVE PARAMETERS
007054  104657  000007                  MOV     U.SEEK(R5),R0   ; GET NUMBER OF SEEKS ISSUED
007056  117407                          DEC     R0              ; DECREMENT SEEK COUNT
007057  057074                          BNE     SEKCNT          ; IF NO REPORT NEEDED, BRANCH
007060  104201  000047                  MOV     #U.SNUM,R1      ; R1 WILL POINT TO SUBUNIT NUMBERS
007062  105051                          ADD     R5,R1           ; R1 POINTS TO SUBUNIT NUMBERS
007063  105651  000034                  ADD     U.SUBU(R5),R1   ; R1 POINTS TO SUBUNIT IN USE NUMBER
007065  104110  001007                  MOV     (R1),OUT.01     ; MOVE TO OUTPUT
007067  104207  000010                  MOV     #T4SEEK,R0      ; MOVE SEEK REPORT NUMBER TO R0
007071  020751                          CALL    HOSTRQ          ; REPORT TO HOST
007072  104207  001750                  MOV     #1000.,R0       ; SET UP FOR NEW COUNT
007074  100657  000007          SEKCNT: MOV     R0,U.SEEK(R5)   ; SAVE COUNT
007076  104207  004501                  MOV     #REDSET,R0      ; READ/WRITE SETUP IS NEXT MODULE
007100  114001                          CLR     R1              ; IMMIDATE CALL TO NEXT MODULE
007101  007276                          BR      STSEXT          ; BRANCH
007102  104651  000005          STSERR: MOV     U.TIMO(R5),R1   ; GET TIMEOUT VALUE
007104  117401                          DEC     R1              ; DECREMENT TIMEOUT VALUE
007105  104207  000003                  MOV     #SEKTST,R0      ; SEEK TEST IS NEXT MODULE
007107  100651  000005                  MOV     R1,U.TIMO(R5)   ; SAVE TIMEOUT VALUE
007111  057276                          BNE     STSEXT          ; IF NON-ZERO BRANCH
007112                                  HARDER  3,<U.LCYL(R5),U.LCYL+1(R5),U.LGRP(R5)>
007112  104200  000363  001012                                                  MOV     #ER3,OUT.04
007115  104650  000056  001013                                                  MOV     U.LCYL(R5),OUT.05
007120  104650  000057  001014                                                  MOV     U.LCYL+1(R5),OUT.06
007123  104650  000060  001015                                                  MOV     U.LGRP(R5),OUT.07
007126  104200  000003  001010                                                  MOV     #3,OUT.02
```

********** OVERLAY 3 - SEE IF THE SEEK IS COMPLETE

```
        007131  101200  001000  001010                                          BIS     #ERHARD,OUT.02
        007134  104200  007134  001007                                          MOV     #.,OUT.01
        007137  104202  000014                                                  MOV     #ERRMC,R2
        007141  104020  001006                                                  MOV     R2,OUT.RQ
    47  007143                          ERRORC  <U.CCYL(R5),U.CCYL+1(R5),U.CGRP(R5)>
        007143  104650  000053  001016                                          MOV     U.CCYL(R5),OUT.08
        007146  104650  000054  001017                                          MOV     U.CCYL+1(R5),OUT.09
        007151  104650  000055  001020                                          MOV     U.CGRP(R5),OUT.10
    48  007154                          ENDERR
        007154  101200  000013  002357                                          BIS     #13,ERRPOS      ; SET THE POSITION
    49  007157  007273                  BR      STSERX          ; BRANCH
    50  007160                  RTDSE:  HARDER  16,<U.LCYL(R5),U.LCYL+1(R5),U.LGRP(R5)>
        007160  104200  002302  001012                                          MOV     #ER16,OUT.04
        007163  104650  000056  001013                                          MOV     U.LCYL(R5),OUT.05
        007166  104650  000057  001014                                          MOV     U.LCYL+1(R5),OUT.06
        007171  104650  000060  001015                                          MOV     U.LGRP(R5),OUT.07
        007174  104200  000020  001010                                          MOV     #16,OUT.02
        007177  101200  001000  001010                                          BIS     #ERHARD,OUT.02
        007202  104200  007202  001007                                          MOV     #.,OUT.01
        007205  104202  000014                                                  MOV     #ERRMC,R2
        007207  104020  001006                                                  MOV     R2,OUT.RQ
    51  007211                          ERRORC  <U.CCYL(R5),U.CCYL+1(R5),U.CGRP(R5)>
        007211  104650  000053  001016                                          MOV     U.CCYL(R5),OUT.08
        007214  104650  000054  001017                                          MOV     U.CCYL+1(R5),OUT.09
        007217  104650  000055  001020                                          MOV     U.CGRP(R5),OUT.10
    52  007222                          ENDERR
        007222  101200  000013  002357                                          BIS     #13,ERRPOS      ; SET THE POSITION
    53  007225  007273                  BR      STSERX          ; BRANCH TO EXIT
    54  007226                  ATNHI:  SOFTER  4,<U.LCYL(R5),U.LCYL+1(R5),U.LGRP(R5)>
        007226  104200  000547  001012                                          MOV     #ER4,OUT.04
        007231  104650  000056  001013                                          MOV     U.LCYL(R5),OUT.05
        007234  104650  000057  001014                                          MOV     U.LCYL+1(R5),OUT.06
        007237  104650  000060  001015                                          MOV     U.LGRP(R5),OUT.07
        007242  104200  000004  001010                                          MOV     #4,OUT.02
        007245  101200  001400  001010                                          BIS     #ERSOFT,OUT.02
        007250  104200  007250  001007                                          MOV     #.,OUT.01
        007253  104202  000013                                                  MOV     #ERRMES,R2
        007255  104020  001006                                                  MOV     R2,OUT.RQ
    55  007257                          ERRORC  <U.CCYL(R5),U.CCYL+1(R5),U.CGRP(R5)>
        007257  104650  000053  001016                                          MOV     U.CCYL(R5),OUT.08
        007262  104650  000054  001017                                          MOV     U.CCYL+1(R5),OUT.09
        007265  104650  000055  001020                                          MOV     U.CGRP(R5),OUT.10
    56  007270                          ENDERR
        007270  101200  000013  002357                                          BIS     #13,ERRPOS      ; SET THE POSITION
    57  007273  104207  000002          STSERX: MOV     #SEEK,R0        ; SEEK IS NEXT MODULE CALLED
    58  007275  114001                          CLR     R1              ; IMMIDIATE CALL TO SEEK
    59  007276  003127                  STSEXT: BR      JMPRET          ; RETURN TO SEQUENCER
    60          007300                  AREA3           .+1
```

```
                                .SBTTL  ********** OVERLAY 4 - SET UP FOR READ OR WRITE OPERATION
007277                                  DMOVLY  RW,AREA0
007277  127343                          .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;************************************** *****************************************
                                ;
                                ;
                                ;
000004                          RDWRT   =       4.              ; READ/WRITE OVERLAY
                                ;
                                ;       RDWRT DESIDES WEITHER TO READ OR WRITE THE LBN SET UP, DETERMINED
                                ;       BY THE PARAMETERS IN U.PARM(R5).  THESE PARAMETERS WERE SET UP
                                ;       IN THE SETUP MODULE
                                ;
004515                                  DIAG$$                          ; ************
004515  115000  002771                  TST     $$DIAG+$DIAG$
004517  014525                         BEQ     .+6
004520  104207  060000                  MOV     #60000,R0
004522  100707  176245                  MCV     R0,@$$DIAG+$DIAG$
004524  004525                         BR      .+1
004525  104653  000033                  MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
004527  102203  000001                  BIT     #RCLB,R3        ; SEE IF RECALIBRATE JUST DONE
004531  054543                         BNE     3$              ; IF SO, BRANCH (LEAVE RETRIES AS THEY ARE)
004532  102203  000004                  BIT     #RETRY,R3       ; IF THIS IS A RETRY, START RETRIES AT 1
004534  014537                         BEQ     1$              ; IF NOT, BRANCH
004535  114001                         CLR     R1              ; START RETRIES AT 1
004536  004541                         BR      2$              ; BRANCH
004537  104201  177777        1$:     MOV     #-1,R1          ; START RETRIES AT 0
004541  100651  000006        2$:     MOV     R1,U.RWTO(R5)   ; SAVE IN UNIT PARAMETER AREA
004543  114001                3$:     CLR     R1              ; IMMIDATE CALL TO NEXT MODULE
004544  114002                         CLR     R2              ; NO ERRORS
004545  102203  000100                  BIT     #REDWRT,R3      ; SEE IF READ IS TO BE DONE
004547  014556                         BEQ     RDMOD           ; IF SO, BRANCH
004550  102203  000040                  BIT     #REVINP,R3      ; SEE IF REVECTOR IN PROGRESS
004552  054556                         BNE     RDMOD           ; IF SO, BRANCH
004553  104207  000005                  MOV     #WRITE,R0       ; WRITE ROUTINE IS NEXT
004555  004565                         BR      RWEXT           ; BRANCH
004556  104207  000006        RDMOD:  MOV     #READ,R0        ; READ ROUTINE IS NEXT
004560  102203  000020                  BIT     #DATERR,R3      ; SEE IF DATA ERROR
004562  054565                         BNE     RWEXT           ; IF SO, BRANCH
004563  100652  000005                  MOV     R2,U.TIMO(R5)   ; ZERO READ TIMEOUT VALUE
004565  103203  000025        RWEXT:  BIC     #RETRY!DATERR.RCLB,R3   ; CLEAR BITS
004567  100653  000033                  MOV     R3,U.PARM(R5)   ; SAVE
004571  003127                         BR      .MPRET          ; RETURN TO SEQUENCER
```

```
                                .SBTTL  ********** OVERLAY 5 - WRITE
                                        DMOVLY  W,AREA0
004572                                  .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
004572  013276
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;************************************** *****************************************
                                ;
                                ;
                                ;
000005                          WRITE   =       5.
                                ;
                                ;       WRITE GENERATES THE PATTERNS AND WRITES THEM TO THE DRIVE
                                ;
                                ;
                                        .ENABL  LSB
004515                                  DIAG$$                  ; ************
004515  115000  002772                  ~ST     $$DIAG+$DIAG$
004517  014525                          BEQ     .+6
004520  104207  060000                  MOV     #60000,R0
004522  100707  176246                  MOV     R0,@$$DIAG+$DIAG$
004524  004525                          BR      .+1
004525  104657  000006                  MOV     U.RWTO(R5),R0   ; MOVE READ/WRITE TIMEOUT VALUE TO R0
004527  115407                          INC     R0              ; DECREMENT COUNT
004530  104072                          MOV     R0,R2           ; COPY COUNT TO R2
004531  103202  177400                  BIC     #HIBYTE,R2      ; CLEAR HIBYTE IF NECESSARY
004533  100652  000006                  MOV     R2,U.RWTO(R5)   ; SAVE READ/WRITE TIMEOUT
004535  014645                          BEQ     8$              ; IF ZERO, WE KNOW THAT THE TIMEOUT IS UNEX
004536  104641  000000                  MOV     S.PARM(R4),R1   ; GET SUBUNIT PARAMETERS
004540  102201  001000                  BIT     #RTRIES,R1      ; SEE IF RETRIES ARE ENABLED
004542  014556                          BEQ     9$              ; IF NOT, BRANCH
004543  106207  000002                  CMP     #2,R0           ; SEE IF RECALIBRATION REQUIRED
004545  054553                          BNE     10$             ; IF NOT, BRANCH
004546  104207  000011                  MOV     #RECALB,R0      ; RECALIBRATION IS NEXT MODULE
004550  114001                          CLR     R1              ; IMMIDATE CALL
004551  114002                          CLR     R2              ; NO ERRORS
004552  004773                          BR      WRTEXT          ; EXIT
004553  106202  000003          10$:    CMP     #3,R2           ; SEE IF TIMEOUT EXPIRED
004555  054645                          BNE     8$              ; IF TIMEOUT UNEXPIRED, BRANCH
004556  102201  020000          9$:     BIT     #DCYLS,R1       ; SEE IF IN THE DIAGNOSTIC AREA
004560  054571                          BNE     11$             ; IF SO, NO REVECTORS, BRANCH
004561  104657  000033                  MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
004563  102207  000200                  BIT     #RBNBN,R0       ; SEE IF HANDLING A REVECTORED BLOCK
004565  054571                          BNE     11$             ; IF SO, BRANCH
004566  104653  000045                  MOV     U.RWER(R5),R3   ; GET LAST WRITE ERROR
004570  054663                          BNE     7$              ; IF IT IS A WRITE SPECIFIC ERRO, TRY REVECTORING
004571                          11$:    HARDER  25,<U.CBN(R5),U.CBN+1(R5),U.RBN(R5),U.RBN+1(R5)>
004571  104200  003470  001012                                                  MOV     #ER25,OUT.04
004574  104650  000037  001013                                                  MOV     U.CBN(R5),OUT.05
004577  104650  000040  001014                                                  MOV     U.CBN+1(R5),OUT.06
004602  104650  000041  001015                                                  MOV     U.RBN(R5),OUT.07
004605  104650  000042  001016                                                  MOV     U.RBN+1(R5),OUT.08
004610  104200  000031  001010                                                  MOV     #25,OUT.02
004613  101200  001000  001010                                                  BIS     #ERHARD,OUT.02
004616  104200  004616  001007                                                  MOV     #.,OUT.01
004621  104202  000014                                                          MOV     #ERRMC,R2
004623  104020  001006                                                          MOV     R2,OUT.RQ
004625                                  ENDERR  0
```

```
                                                                CLR     ERRPOS          ; CLEAR THE POSITION
    004625  114000  002357
 42 004627  104653  000033             MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
 43 004631  103203  000200             BIC     #RBNBN,R3       ; IF HANDLING AN RBN, CLEAR IT
 44 004633  100653  000033             MOV     R3,U.PARM(R5)   ; SAVE
 45 004635  104207  000001             MOV     #1,R0           ; ONLY 1 SECTOR HANDLED
 46 004637  100657  000016             MOV     R0,U.NSEC(R5)   ; STORE
 47 004641  104207  000001             MOV     #SETUP,R0       ; SETUP IS NEXT MODULE CALLED
 48 004643  104051                     MOV     R5,R1           ; DELAYED CALL TO SETUP
 49 004644  004773                     BR      WRTEXT          ; BRANCH
 50 004645  114002              8$:    CLR     R2              ; TO CLEAR READ ERRORS
 51 004646  100652  000045             MOV     R2,U.RWER(R5)   ; FLAG AS NO ERRORS
 52 004650  021657                     CALL    BUILDP          ; BUILD THE PARAMETERS
 53 004651  115002                     TST     R2              ; SEE IF AN ERROR OCCURRED
 54 004652  054674                     BNE     1$              ; IF SO, BRANCH
 55 004653  022117                     CALL    BULDUM          ; BUILD DUMMY SDI CONTROL BLOCK
 56 004654  024774                     CALL    BULDSC          ; BUILD THE SECTOR
 57 004655  025057                     CALL    WBLOCK          ; WRITE THE SECTOR
 58 004656  104657  000033             MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
 59 004660  115002                     TST     R2              ; SEE IF ERROR OCCURRED
 60 004661  014714                     BEQ     2$              ; IF NOT, BRANCH
 61 004662  034674                     BPL     1$              ; IF NOT REVECTORED, BRANCH
 62 004663  101207  000400      7$:    BIS     #REVEC,R0       ; FLAG BLOCK AS REVECTORED
 63 004665  100657  000033             MOV     R0,U.PARM(R5)   ; SAVE
 64 004667  104207  000010             MOV     #REVCT,R0       ; REVECTOR NEXT MODULE CALLED
 65 004671  114002                     CLR     R2              ; NO ERRORS (REVECTORED SECTOR)
 66 004672  114001                     CLR     R1              ; IMMIDATE CALL TO REVECTOR ROUTINE
 67 004673  004773                     BR      WRTEXT          ; BRANCH
 68 004674  104651  000016      1$:    MOV     U.NSEC(R5),R1   ; GET NUMBER OF SECTORS WRITTEN
 69 004676  054705                     BNE     3$              ; IF ANY WERE WRITTEN, BRANCH
 70 004677  115401                     INC     R1              ; R0 IS NOW 1
 71 004700  100651  000017             MOV     R1,U.MSEC(R5)   ; JUST TRY TO WRITE ONE SECTOR
 72 004702  104207  000005             MOV     #WRITE,R0       ; WRITE IS NEXT MODULE CALLED
 73 004704  004772                     BR      6$              ; BRANCH
 74 004705  101207  000004      3$:    BIS     #RETRY,R0       ; FLAG THAT ATTEMPTING RETRIES
 75 004707  100657  000033             MOV     R0,U.PARM(R5)   ; SAVE
 76 004711  104207  000001             MOV     #SETUP,R0       ; SETUP IS NEXT MODULE CALLED
 77 004713  004772                     BR      6$              ; BRANCH
 78 004714  104653  000033      2$:    MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
 79 004716  103203  000200             BIC     #RBNBN,R3       ; IF HANDLING AN RBN, CLEAR IT
 80 004720  100653  000033             MOV     R3,U.PARM(R5)   ; SAVE
 81 004722  104657  000006             MOV     U.RWTO(R5),R0   ; GET READ/WRITE TIMEOUT
 82 004724  014746                     BEQ     5$              ; IF SUCCEDED ON FIRST TRY, BRANCH
 83 004725                             REPSFT  SOFT            ; REPORT ONE SOFT ERROR (NO ECC CORRECTION)
    004725  104200  000001  001010                                     MOV     #1,OUT.02
    004730  114000  001011                                             CLR     OUT.03
    004732  100467                                                             MOV R0,-(SP)
    004733  104207  000047                                             MOV     #U.SNUM,R0
    004735  105057                                                     ADD     R5,R0
    004736  105657  000034                                             ADD     U.SUBU(R5),R0
    004740  104170  001007                                             MOV     (R0),OUT.01
    004742  104207  000007                                             MOV     #T4SOFT,R0
    004744  020751                                                     CALL    HOSTRQ
    004745  104267                                                             MOV (SP)+,R0
 84 004746  104207  000001      5$:    MOV     #SETUP,R0       ; SETUP NEXT MODULE CALLED
 85 004750  104643  000007             MOV     S.MEGW(R4),R3   ; GET MEGABYTE COUNT
 86 004752  106203  003642             CMP     #1954.,R3       ; SEE IF ONE MEGABYTE TRANSFERED
 87 004754  034772                     BPL     6$              ; BRANCH IF NOT
```

********** OVERLAY 5 - WRITE

```
     88 004755  107203  003642                    SUB     #1954.,R3       ; CLEAR MEGABYTE COUNT
     89 004757  100643  000007                    MOV     R3,S.MEGW(R4)   ; SAVE NEW COUNT
     90 004761  114000  001010                    CLR     OUT.02          ; NOT REPORTING READS
     91 004763  104200  000001  001011            MOV     #1,OUT.03       ; REPORT 1 MEGABIT WRITTEN
     92 004766  104202  000011                    MOV     #T4MXFR,R2      ; SET UP FOR MEGABIT REPORT
     93 004770  104020  001043                    MOV     R2,OUT.29       ; FLAG AS NON-ERROR
     94 004772  104051                  6$:      MOV     R5,R1           ; DELAYED CALL TO NEXT ROUTINE
     95 004773  003127                  WRTEXT:  BR      JMPRET          ; RETURN TO RDWRT
     96                                           .DSABL  LSB
```

```
                                        .SBTTL  BULDSC - BUILD THE SECTOR TO WRITE (FILL WITH PATTERN AND CALC EDC)
004774                                  BULDSC:
                                        ;
                                        ;       BULDSC BUILDS THE INFORMATION AND DATA PATTERN TO BE WRITTEN ON THE
                                        ;       SECTOR
                                        ;
004774  104651  000011                          MOV     U.PAT(R5),R1    ; GET PATTERN NUMBER TO WRITE
004776  104013                                  MOV     R1,R3           ; COPY PATTERN NUMBER TO R3
004777  110701                                  SWAB    R1              ; SWAP THE BYTES
005000  101013                                  BIS     R1,R3           ; REPLICATE THE PATTERN NUMBER
005001  104031                                  MOV     R3,R1           ; COPY TO R1
005002  110201                                  ROL     R1              ; ROTATE TO NEXT POSITION
005003  110201                                  ROL     R1
005004  110201                                  ROL     R1
005005  110201                                  ROL     R1
005006  103201  007417                          BIC     #7417,R1        ; CLEAR UNUSED BITS
005010  101013                                  BIS     R1,R3           ; REPLICATE THE PATTERN
005011  104307  002356                          MOV     SECPTR,R0       ; R0 POINTS TO BUFFER AREA
005013  100273                                  MOV     R3,(R0)+        ; MOVE PATTERN NUMBERS TO SECTOR
005014  025022                                  CALL    COPPAT          ; GENERATE PATTERN IN SECTOR
005015  117407                                  DEC     R0              ; POINT TO BEGINNING OF SECTOR
005016  021143                                  CALL    CMPEDC          ; COMPUTE THE EDC OVER THE SECTOR
005017  100672  000400                          MOV     R2,BF.EDC(R0)   ; STORE EDC
005021  000000                                  RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                .SBTTL  COPPAT - COPY THE DATA PATTERN TO BUFFER
005022                          COPPAT:
                                ;
                                ;       COPPAT COPIES THE PATTERN TO THE SECTOR TO BE WRITTEN
                                ;
005022                                  PUSH    <R0,R4,R5>      ; SAVE R0,R4,R5
005022  100467                                                                  MOV R0,-(SP)
005023  100464                                                                  MOV R4,-(SP)
005024  100465                                                                  MOV R5,-(SP)
005025  104652  000011                 MOV     U.PAT(R5),R2    ; R2 HAS PATTERN NUMBER
005027  104622  002475                 MOV     PATPTR(R2),R2   ; POINT TO PATTERN
005031  104201  000377                 MOV     #SCTWRD,R1      ; R1 HAS NUMBER OF WORDS TO FILL WITH PATTERN
005033  104024          COPLP0: MOV     R2,R4           ; R4 POINTS TO LENGTH OF PATTERN
005034  104243                  MOV     (R4)+,R3        ; R3 CONTAINS LENGTH OF PATTERN
005035  106203  000001                 CMP     #1,R3           ; SEE IF PATTERN IS 1 WORD LONG
005037  015047                         BEQ     ONEPAT          ; IF SO, BRANCH
005040  104245          COPLP1: MOV     (R4)+,R5        ; R5 GETS 1 WORD OF THE DATA PATTERN
005041  100275                  MOV     R5,(R0)+        ; MOVE PATTERN WORD TO SECTOR AREA
005042  117401                  DEC     R1              ; DECREMENT NUMBER OF WORDS TO WRITE IN SECTOR
005043  015053                  BEQ     COPFIN          ; IF ALL WORDS WRITTEN, BRANCH
005044  117403                  DEC     R3              ; DECREMENT COUNT OF WORDS IN PATTERN
005045  055040                  BNE     COPLP1          ; IF DATA PATTERN UNFINISHED, BRANCH
005046  005033                  BR      COPLP0          ; BRANCH
005047  104145          ONEPAT: MOV     (R4),R5         ; GET 1 WORD OF DATA PATTERN
005050  100275          COPLP2: MOV     R5,(R0)+        ; MOVE PATTERN TO SECTOR AREA
005051  117401                  DEC     R1              ; DECREMENT NUMBER OF WORDS TO MOVE TO SECTOR
005052  055050                  BNE     COPLP2          ; IF INCOMPLETE, BRANCH
005053          COPFIN: POP     <R5,R4,R0>      ; RESTORE R5,R4,R0
005053  104265                                                                  MOV (SP)+,R5
005054  104264                                                                  MOV (SP)+,R4
005055  104267                                                                  MOV (SP)+,R0
005056  000000                  RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                        .SBTTL  WBLOCK - WRITE THE SECTOR(S) AND THEN CHECK FOR ERRORS
005057                                  WBLOCK:
                                        ;
                                        ;       WBLOCK WRITES THE SECTOR(S) ONTO THE DRIVE
                                        ;
005057  021173                          CALL    RTDS            ; GET REAL TIME STATE
005060  115002                          TST     R2              ; SEE IF ANY ERRORS
005061  055150                          BNE     .$              ; IF SO, BRANCH
005062  102201  100000                  BIT     #RWRDY,R1       ; SEE IF READ/WRITE ASSERTED
005064  055227                          BNE     2$              ; IF SO, BRANCH
005065  103201  077674                  BIC     #77674,R1       ; CLEAR UNUSED BITS
005067                                  SOFTER  30,<U.RWTO(R5),U.CBN(R5),U.CBN+1(R5)>
005067  104200  004051  001012                                          MOV     #ER30,OUT.04
005072  104650  000006  001013                                          MOV     U.RWTO(R5),OUT.05
005075  104650  000037  001014                                          MOV     U.CBN(R5),OUT.06
005100  104650  000040  001015                                          MOV     U.CBN+1(R5),OUT.07
005103  104200  000036  001010                                          MOV     #30,OUT.02
005106  101200  001400  001010                                          BIS     #ERSOFT,OUT.02
005111  104200  005111  001007                                          MOV     #.,OUT.01
005114  104202  000013                                                  MOV     #ERRMES,R2
005116  104020  001006                                                  MOV     R2,OUT.RQ
005120                                  ERRORC  <U.CGRP(R5),U.CCYL(R5),U.CCYL+1(R5)>
005120  104650  000055  001016                                          MOV     U.CGRP(R5),OUT.08
005123  104650  000053  001017                                          MOV     U.CCYL(R5),OUT.09
005126  104650  000054  001020                                          MOV     U.CCYL+1(R5),OUT.10
005131                                  ERRORC  <U.LCYL(R5),U.LCYL+1(R5),U.LGRP(R5),R1>
005131  104650  000056  001021                                          MOV     U.LCYL(R5),OUT.11
005134  104650  000057  001022                                          MOV     U.LCYL+1(R5),OUT.12
005137  104650  000060  001023                                          MOV     U.LGRP(R5),OUT.13
005142  104010  001024                                                  MOV     R1,OUT.14
005144                                  ENDERR
005144  101200  000017  002357                                          BIS     #17,ERRPOS      ; SET THE POSITION
005147  005525                          BR      WOUT            ; BRANCH
005150                          1$:     SOFTER  55,<U.RWTO(R5),U.CBN(R5),U.CBN+1(R5)>
005150  104200  005743  001012                                          MOV     #ER55,OUT.04
005153  104650  000006  001013                                          MOV     U.RWTO(R5),OUT.05
005156  104650  000037  001014                                          MOV     U.CBN(R5),OUT.06
005161  104650  000040  001015                                          MOV     U.CBN+1(R5),OUT.07
005164  104200  000067  001010                                          MOV     #55,OUT.02
005167  101200  001400  001010                                          BIS     #ERSOFT,OUT.02
005172  104200  005172  001007                                          MOV     #.,OUT.01
005175  104202  000013                                                  MOV     #ERRMES,R2
005177  104020  001006                                                  MOV     R2,OUT.RQ
005201                                  ERRORC  <U.CGRP(R5),U.CCYL(R5),U.CCYL+1(R5)>
005201  104650  000055  001016                                          MOV     U.CGRP(R5),OUT.08
005204  104650  000053  001017                                          MOV     U.CCYL(R5),OUT.09
005207  104650  000054  001020                                          MOV     U.CCYL+1(R5),OUT.10
005212                                  ERRORC  <U.LCYL(R5),U.LCYL+1(R5),U.LGRP(R5)>
005212  104650  000056  001021                                          MOV     U.LCYL(R5),OUT.11
005215  104650  000057  001022                                          MOV     U.LCYL+1(R5),OUT.12
005220  104650  000060  001023                                          MOV     U.LGRP(R5),OUT.13
005223                                  ENDERR
005223  101200  000016  002357                                          BIS     #16,ERRPOS      ; SET THE POSITION
005226  005525                          BR      WOUT            ; BRANCH
005227                          2$:     PUSH    R4              ; SAVE R4
005227  100464                                                                  MOV R4,-(SP)
005230  104307  002352                  MOV     CHAINS,R0       ; POINT TO START OF CHAIN
```

```
                                        MOV     U.MASK(R5),R2   ; R2 HAS SDI INTERCONNECT
                                        MOV     S.SCHR(R4),R4   ; R4 POINTS TO SUBUNIT CHARACTERISTICS
                                        MOV     DATPRE(R4),R4   ; R4 CONTAINS DATA PREAMBLE LENGTH
                                        BIC     #HIBYTE,R4      ; STRIP OFF UNUSED BITS
                                        XFC     WAITSI          ; WAIT FOR SECTOR OR INDEX PULSE
                                        XFC     XWRITE          ; WRITE THE SECTOR(S)
                                        POP     R4              ; RESTORE R4
005244  104264                                                                  MOV (SP)+,R4
                                        CALL    FNDWER          ; FIND BUFFER THAT CAUSED ERROR
                                        MOV     R2,U.NSEC(R5)   ; SAVE SECTORS WRITTEN
                                        ADD     S.MEGW(R4),R2   ; ADD TO MEGABYTES WRITTEN
                                        MOV     R2,S.MEGW(R4)   ; SAVE
                                        TST     R1              ; SEE IF ERROR OCCURRED
                                        BEQ     WEXT            ; IF NO ERROR, BRANCH
                                        MOV     R1,U.RWER(R5)   ; SAVE WRITE ERROR
                                        CALL    BLKCHK          ; SEE IF THIS IS A KNOWN BAD BLOCK
                                        BCS     3$              ; IF NOT, BRANCH
005261  045263                                                                  BCC     .+2
005262  005271                                                                  BR      3$
                                        MOV     U.NSEC(R5),R2   ; GET NUMBER OF SECTORS HANDLED
                                        INC     R2              ; INCREMENT COUNT
                                        MOV     R2,U.NSEC(R5)   ; SAVE
                                        BR      WEXT            ; EXIT AS NO ERRORS
                        3$:             CMP     #3,R1           ; SEE IF THIS BLOCK IS REVECTORED
                                        BEQ     NOHCFL          ; IF SO, BRANCH
                                        BIT     #6,R1           ; SEE IF HEADER COMPARE FAILURE
                                        BEQ     NOTOUT          ; IF NOT, BRANCH
                                        SOFTER  5,<U.RWTO(R5),RW.LOW(R0),RW.HI(R0),U.RBN(R5)>
005277  104200  000720  001012                                                  MOV     #ER5,OUT.04
005302  104650  000006  001013                                                  MOV     U.RWTO(R5),OUT.05
005305  104670  000002  001014                                                  MOV     RW.LOW(R0),OUT.06
005310  104670  000003  001015                                                  MOV     RW.HI(R0),OUT.07
005313  104650  000041  001016                                                  MOV     U.RBN(R5),OUT.08
005316  104200  000005  001010                                                  MOV     #5,OUT.02
005321  101200  001400  001010                                                  BIS     #ERSOFT,OUT.02
005324  104200  005324  001007                                                  MOV     #.,OUT.01
005327  104202  000013                                                          MOV     #ERRMES,R2
005331  104020  001006                                                          MOV     R2,OUT.RQ
                                        ERRORC  <U.RBN+1(R5),RW.CMD(R0),U.CGRP(R5),U.CCYL(R5),U.CCYL+1(R5)>
005333  104650  000042  001017                                                  MOV     U.RBN+1(R5),OUT.09
005336  104670  000004  001020                                                  MOV     RW.CMD(R0),OUT.10
005341  104650  000055  001021                                                  MOV     U.CGRP(R5),OUT.11
005344  104650  000053  001022                                                  MOV     U.CCYL(R5),OUT.12
005347  104650  000054  001023                                                  MOV     U.CCYL+1(R5),OUT.13
                                        ERRORC  <U.LCYL(R5),U.LCYL+1(R5),U.LGRP(R5)>
005352  104650  000056  001024                                                  MOV     U.LCYL(R5),OUT.14
005355  104650  000057  001025                                                  MOV     U.LCYL+1(R5),OUT.15
005360  104650  000060  001026                                                  MOV     U.LGRP(R5),OUT.16
                                        ENDERR
005363  101200  000021  002357                                                  BIS     #21,ERRPOS      ; SET THE POSITION
                                        BR      WOUT            ; BRANCH
                        NOTOUT: MOV     U.WRIT(R5),R1   ; GET WRITE PROTECTION STATUS
                                MOV     U.SUBU(R5),R2   ; GET SUBUNIT IN USE
                                ADD     #3,R2           ; ADD 3 TO SHIFT WRITE PROT STAT TO LO NIBBLE
                        FNDWRT: ROR     R1              ; ROTATE PROTECTION STATUS (LO BIT TO CARRY)
                                DEC     R2              ; DECREMENT SUBUNIT IN USE
                                BPL     FNDWRT          ; IF >= 0, BRANCH
```

```
     59 005400  110601                          ROR     R1              ; MOVE THE BIT TO TEST TO CARRY
     60 005401  045426                          BCC     WTIMOT          ; IF CARRY CLEAR (NOT WRITE PROTECTED) BRANCH
     61 005402                                  DEVFTL  18              ; REPORT ERROR
        005402  104200  002470  001012                                          MOV     #ER18,OUT.04
        005405  104200  000022  001010                                          MOV     #18,OUT.02
        005410  101200  000400  001010                                          BIS     #FTLDEV,OUT.02
        005413  104200  005413  001007                                          MOV     #.,OUT.01
        005416  104202  000014                                                  MOV     #ERRMC,R2
        005420  104020  001006                                                  MOV     R2,OUT.RQ
     62 005422                                  ENDERR
        005422  101200  000005  002357                                          BIS     #5,ERRPOS       ; SET THE POSITION
     63 005425  005525                          BR      WOUT            ; BRANCH
     64 005426                          WTIMOT: SOFTER  6,<U.RWTO(R5),RW.LOW(R0),RW.HI(R0),U.RBN(R5)>
        005426  104200  001123  001012                                          MOV     #ER6,OUT.04
        005431  104650  000006  001013                                          MOV     U.RWTO(R5),OUT.05
        005434  104670  000002  001014                                          MOV     RW.LOW(R0),OUT.06
        005437  104670  000003  001015                                          MOV     RW.HI(R0),OUT.07
        005442  104650  000041  001016                                          MOV     U.RBN(R5),OUT.08
        005445  104200  000006  001010                                          MOV     #6,OUT.02
        005450  101200  001400  001010                                          BIS     #ERSOFT,OUT.02
        005453  104200  005453  001007                                          MOV     #.,OUT.01
        005456  104202  000013                                                  MOV     #ERRMES,R2
        005460  104020  001006                                                  MOV     R2,OUT.RQ
     65 005462                                  ERRORC  <U.RBN+1(R5),RW.CMD(R0),U.CGRP(R5),U.CCYL(R5),U.CCYL+1(R5)>
        005462  104650  000042  001017                                          MOV     U.RBN+1(R5),OUT.09
        005465  104670  000004  001020                                          MOV     RW.CMD(R0),OUT.10
        005470  104650  000055  001021                                          MOV     U.CGRP(R5),OUT.11
        005473  104650  000053  001022                                          MOV     U.CCYL(R5),OUT.12
        005476  104650  000054  001023                                          MOV     U.CCYL+1(R5),OUT.13
     66 005501                                  ERRORC  <U.LCYL(R5),U.LCYL+1(R5),U.LGRP(R5)>
        005501  104650  000056  001024                                          MOV     U.LCYL(R5),OUT.14
        005504  104650  000057  001025                                          MOV     U.LCYL+1(R5),OUT.15
        005507  104650  000060  001026                                          MOV     U.LGRP(R5),OUT.16
     67 005512                                  ENDERR
        005512  101200  000021  002357                                          BIS     #21,ERRPOS      ; SET THE POSITION
     68 005515  005525                          BR      WOUT            ; BRANCH
     69 005516  104652  000016          NOHCFL: MOV     U.NSEC(R5),R2   ; SEE IF THE FIRST BUFFER HAS BEEN WRITTEN
     70 005520  055524                          BNE     WEXT            ; IF SO, BRANCH (DON'T REVECTOR NOW)
     71 005521  104202  177777                  MOV     #-1,R2          ; FLAG BLOCK AS REVECTORED
     72 005523  005525                          BR      WOUT            ; BRANCH
     73 005524  114002                  WEXT:   CLR     R2              ; NO ERRORS
     74 005525  000000                  WOUT:   RETURN                  ; RETURN TO CALLING PROGRAM;
```

```
                                .SBTTL  FNDWER - IF ERROR, FIND IT'S POSITION IN THE CHAIN
005526                          FNDWER:
                                ;
                                ;       FIND THE FIRST BUFFER THAT IS NOT WRITTEN
                                ;
                                        PUSH    <R1,R4,R5>      ; SAVE REGISTERS
005526  100461                                                                  MOV R1,-(SP)
005527  100464                                                                  MOV R4,-(SP)
005530  100465                                                                  MOV R5,-(SP)
005531  104654  000037                MOV     U.CBN(R5),R4    ; GET LO ORDER BN
005533  104655  000040                MOV     U.CBN+1(R5),R5  ; GET HI BN
005535  114002                        CLR     R2              ; CLEAR SECTOR COUNT
005536  104307  002352                MOV     CHAINS,R0       ; R0 POINTS TO FIRST BUFFER
005540  100674  000002        1$:     MOV     R4,RW.LOW(R0)   ; MOVE TO CHAIN
005542  100675  000003                MOV     R5,RW.HI(R0)    ; MOVE TO CHAIN
005544  104171                        MOV     (R0),R1         ; GET STATUS BITS
005545  102201  040000                BIT     #BUFFLG,R1      ; SEE IF THIS BUFFER HAS BEEN WRITTEN
005547  055563                        BNE     2$              ; IF NOT, BRANCH
005550  115402                        INC     R2              ; INCREMENT SECTOR COUNT
005551  115001                        TST     R1              ; SEE IF END-OF-LIST
005552  075563                        BMI     2$              ; IF LAST BUFFER, EXIT
005553  103201  170000                BIC     #^CHBHINB,R1    ; CLEAR UNUSED BITS
005555  104017                        MOV     R1,R0           ; MOVE TO R0
005556  105204  000001                ADD     #1,R4           ; ADD ONE TO LOW ORDER BN
005560  045540                        BCC     1$              ; IF NO CARRY, BRANCH
005561  115405                        INC     R5              ; PRPOGATE CARRY
005562  005540                        BR      1$              ; LOOP
005563                          2$:     POP     <R5,R4,R1>      ; RESTORE REGISTERS
005563  104265                                                                  MOV (SP)+,R5
005564  104264                                                                  MOV (SP)+,R4
005565  104261                                                                  MOV (SP)+,R1
005566  000000                        RETURN                  ; RETURN TO CALLING PROGRAM
005570                          BUFA1:          .+1             ; SETUP FOR SCRATCH AREA BUFFER
```

```
                                .SBTTL  ********** OVERLAY 6 - READ
                                        DMOVLY  R,AREA0
005567  145737                          .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                ;***************************************************************************
                                ;***************************************************************************
                                ;***************************************************************************
                                ;***************************************************************************
                                ;
                                ;
                                ;
000006                          READ    =       6.
                                ;
                                ;       READ READS A BLOCK FROM THE DEVICE
                                ;
                                        .ENABL  LSB
004515                                  DIAG$$                  ; ************
004515  115000  002773                  TST     $$DIAG+$DIAG$
004517  014525                          BEQ     .+6
004520  104207  060000                  MOV     #60000,R0
004522  100707  176247                  MOV     R0,@$$DIAG+$DIAG$
004524  004525                          BR      .+1
004525  104657  000006                  MOV     U.RWTO(R5),R0   ; MOVE READ/WRITE TIMEOUT VALUE TO R0
004527  115407                          INC     R0              ; INCREMENT COUNT
004530  104072                          MOV     R0,R2           ; COPY COUNT TO R2
004531  103202  177400                  BIC     #HIBYTE,R2      ; CLEAR UNUSED BITS
004533  100652  000006                  MOV     R2,U.RWTO(R5)   ; SAVE READ/WRITE TIMEOUT
004535  014645                          BEQ     5$              ; IF ZERO, FIRST TIME -- BRANCH
004536  104641  000000                  MOV     S.PARM(R4),R1   ; GET SUBUNIT PARAMETERS
004540  102201  001000                  BIT     #RTRIES,R1      ; SEE IF RETRIES ENABLED
004542  014556                          BEQ     9$              ; IF NOT, BRANCH
004543  106207  000002                  CMP     #2,R0           ; SEE IF RECALIBRATION REQUIRED
004545  054553                          BNE     10$             ; IF NOT, BRANCH
004546  104207  000011                  MOV     #RECALB,R0      ; RECALIBRATION IS NEXT MODULE
004550  114001                          CLR     R1              ; IMMIDATE CALL
004551  114002                          CLR     R2              ; NO ERRORS
004552  004736                          BR      8$              ; EXIT
004553  106202  000003          10$:    CMP     #3,R2           ; SEE IF TIMEOUT EXPIRED
004555  054645                          BNE     5$              ; IF TIMEOUT UNEXPIRED, BRANCH
004556  102201  020000          9$:     BIT     #DCYLS,R1       ; SEE IF ON DIAGNOSTIC CYLINDERS
004560  054571                          BNE     11$             ; IF SO, NO REVECTORS, BRANCH
004561  104657  000033                  MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
004563  102207  000600                  BIT     #RBNBN.REVEC,R0 ; SEE IF HANDLING A REVECTORED BLOCK
004565  054571                          BNE     11$             ; IF SO, BRANCH
004566  104657  000045                  MOV     U.RWER(R5),R0   ; GET LAST READ ERROR
004570  054660                          BNE     7$              ; IF IT IS A READ SPECIFIC ERROR, TRY REVECTORING
004571                          11$:    HARDER  26,<U.CBN(R5),U.CBN+1(R5),U.RBN(R5),U.RBN+1(R5)>
004571  104200  003544  001012                                          MOV     #ER26,OUT.04
004574  104650  000037  001013                                          MOV     U.CBN(R5),OUT.05
004577  104650  000040  001014                                          MOV     U.CBN+1(R5),OUT.06
004602  104650  000041  001015                                          MOV     U.RBN(R5),OUT.07
004605  104650  000042  001016                                          MOV     U.RBN+1(R5),OUT.08
004610  104200  000032  001010                                          MOV     #26,OUT.02
004613  101200  001000  001010                                          BIS     #ERHARD,OUT.02
004616  104200  004616  001007                                          MOV     #.,OUT.01
004621  104202  000014                                                  MOV     #ERRMC,R2
004623  104020  001006                                                  MOV     R2,OUT.RQ
004625                                  ENDERR  0
```

```
                004625  114000  002357                                          CLR     ERRPOS          ; CLEAR THE POSITION
     42         004627  104653  000033                  MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
     43         004631  103203  000200                  BIC     #RBNBN,R3       ; IF HANDLING AN RBN, CLEAR IT
     44         004633  100653  000033                  MOV     R3,J.PARM(R5)   ; SAVE
     45         004635  104207  000001                  MOV     #1,R0           ; ONE SECTOR HANDLED
     46         004637  100657  000016                  MOV     R0,U.NSEC(R5)   ; SAVE
     47         004641  104207  000001                  MOV     #SETUP,R0       ; SETUP IS NEXT MODULE CALLED
     48         004643  104051                          MOV     R5,R1           ; DELAYED CALL TO SETUP
     49         004644  004727                          BR      REDEXT          ; BRANCH
     50         004645  114002              5$:         CLR     R2              ; TO CLEAR READ ERRORS
     51         004646  100652  000045                  MOV     R2,U.RWER(R5)   ; FLAG AS NO ERRORS
     52         004650  021657                          CALL    BUILDP          ; BUILD THE PARAMETERS
     53         004651  115002                          TST     R2              ; SEE IF AN ERROR OCCURRED
     54         004652  054670                          BNE     1$              ; IF SO, BRANCH
     55         004653  022117                          CALL    BULDUM          ; BUILD THE DUMMY SDI CONTROL BLOCK
     56         004654  024737                          CALL    RBLOCK          ; READ THE SECTOR
     57         004655  115002                          TST     R2              ; SEE IF AN ERROR OCCURRED
     58         004656  014700                          BEQ     2$              ; IF NOT, BRANCH
     59         004657  034670                          BPL     1$              ; IF NOT REVECTORED, BRANCH
     60         004660  104657  000033        7$:       MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
     61         004662  101207  000400                  BIS     #REVEC,R0       ; MARK BLOCK AS REVECTORED
     62         004664  100657  000033                  MOV     R0,U.PARM(R5)   ; SAVE
     63         004666  114002                          CLR     R2              ; NO ERRORS (REVECTORED SECTOR)
     64         004667  004726                          BR      4$              ; BRANCH
     65         004670  104207  000001        1$:       MOV     #1,R0           ; ONLY TRY TO READ ONE BUFFER
     66         004672  100657  000017                  MOV     R0,U.MSEC(R5)   ; SAVE
     67         004674  104207  000006                  MOV     #READ,R0        ; NEXT MODULE IS READ (TRY AGAIN)
     68         004676  104051                          MOV     R5,R1           ; DELAYED CALL TO NEXT ROUTINE
     69         004677  004736                          BR      8$              ; BRANCH
     70         004700  104657  000006        2$:       MOV     U.RWTO(R5),R0   ; SEE IF ANY SOFT ERRORS OCCURED
     71         004702  014724                          BEQ     3$              ; IF NOT, BRANCH
     72         004703                                  REPSFT  SOFT            ; REPORT SOFT ERROR
                004703  104200  000001  001010                                                  MOV     #1,OUT.02
                004706  114000  001011                                                          CLR     OUT.03
                004710  100467                                                                          MOV R0,-(SP)
                004711  104207  000047                                                          MOV     #U.SNUM,R0
                004713  105057                                                                  ADD     R5,R0
                004714  105657  000034                                                          ADD     U.SUBU(R5),R0
                004716  104170  001007                                                          MOV     (R0),OUT.01
                004720  104207  000007                                                          MOV     #T4SOFT,R0
                004722  020751                                                                  CALL    HOSTRQ
                004723  104267                                                                          MOV (SP)+,R0
     73         004724  104207  000007        3$:       MOV     #CHKECC,R0      ; CHKECC NEXT MODULE CALLED
     74         004726  114001                4$:       CLR     R1              ; IMMIDATE CALL TO NEXT MODULE
     75         004727  104653  000033        REDEXT:   MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
     76         004731  102203  000400                  BIT     #REVEC,R3       ; SEE IF HANDLING A REVECTORING OPERATION
     77         004733  014736                          BEQ     8$              ; IF NOT, BRANCH
     78         004734  104207  000010                  MOV     #REVCT,R0       ; IF SO, REVCT IS NEXT MODULE
     79         004736  003127                8$:       BR      JMPRET          ; RETURN TO RDWRT MODULE
     80                                                 .DSABL  LSB
```

RBLOCK - READ THE SECTORS

```
                                        .SBTTL  RBLOCK - READ THE SECTORS
004737                                  RBLOCK:
                                        ;
                                        ;       READ A SECTOR FROM THE DEVICE
                                        ;
004737  021173                          CALL    RTDS            ; GET REAL TIME STATE
004740  115002                          TST     R2              ; SEE IF ANY ERRORS
004741  055030                          BNE     1$              ; IF SO, BRANCH
004742  102201  100000                  BIT     #RWRDY,R1       ; SEE IF READ/WRITE ASSERTED
004744  055107                          BNE     2$              ; IF SO, BRANCH
004745  103201  077674                  BIC     #77674,R1       ; CLEAR UNUSED BITS
004747                                  SOFTER  31,<U.RWTO(R5),U.CBN(R5),U.CBN+1(R5)>
004747  104200  004247  001012                                          MOV     #ER31,OUT.04
004752  104650  000006  001013                                          MOV     U.RWTO(R5),OUT.05
004755  104650  000037  001014                                          MOV     U.CBN(R5),OUT.06
004760  104650  000040  001015                                          MOV     U.CBN+1(R5),OUT.07
004763  104200  000037  001010                                          MOV     #31,OUT.02
004766  101200  001400  001010                                          BIS     #ERSOFT,OUT.02
004771  104200  004771  001007                                          MOV     #.,OUT.01
004774  104202  000013                                                  MOV     #ERRMES,R2
004776  104020  001006                                                  MOV     R2,OUT.RQ
005000                                  ERRORC  <U.CGRP(R5),U.CCYL(R5),U.CCYL+1(R5)>
005000  104650  000055  001016                                          MOV     U.CGRP(R5),OUT.08
005003  104650  000053  001017                                          MOV     U.CCYL(R5),OUT.09
005006  104650  000054  001020                                          MOV     U.CCYL+1(R5),OUT.10
005011                                  ERRORC  <U.LCYL(R5),U.LCYL+1(R5),U.LGRP(R5),R1>
005011  104650  000056  001021                                          MOV     U.LCYL(R5),OUT.11
005014  104650  000057  001022                                          MOV     U.LCYL+1(R5),OUT.12
005017  104650  000060  001023                                          MOV     U.LGRP(R5),OUT.13
005022  104010  001024                                                  MOV     R1,OUT.14
005024                                  ENDERR
005024  101200  000017  002357                                          BIS     #17,ERRPOS      ; SET THE POSITION
005027  005322                          BR      ROUT            ; BRANCH
005030                          1$:     SOFTER  56,<U.RWTO(R5),U.CBN(R5),U.CBN+1(R5)>
005030  104200  006136  001012                                          MOV     #ER56,OUT.04
005033  104650  000006  001013                                          MOV     U.RWTO(R5),OUT.05
005036  104650  000037  001014                                          MOV     U.CBN(R5),OUT.06
005041  104650  000040  001015                                          MOV     U.CBN+1(R5),OUT.07
005044  104200  000070  001010                                          MOV     #56,OUT.02
005047  101200  001400  001010                                          BIS     #ERSOFT,OUT.02
005052  104200  005052  001007                                          MOV     #.,OUT.01
005055  104202  000013                                                  MOV     #ERRMES,R2
005057  104020  001006                                                  MOV     R2,OUT.RQ
005061                                  ERRORC  <U.CGRP(R5),U.CCYL(R5),U.CCYL+1(R5)>
005061  104650  000055  001016                                          MOV     U.CGRP(R5),OUT.08
005064  104650  000053  001017                                          MOV     U.CCYL(R5),OUT.09
005067  104650  000054  001020                                          MOV     U.CCYL+1(R5),OUT.10
005072                                  ERRORC  <U.LCYL(R5),U.LCYL+1(R5),U.LGRP(R5)>
005072  104650  000056  001021                                          MOV     U.LCYL(R5),OUT.11
005075  104650  000057  001022                                          MOV     U.LCYL+1(R5),OUT.12
005100  104650  000060  001023                                          MOV     U.LGRP(R5),OUT.13
005103                                  ENDERR
005103  101200  000016  002357                                          BIS     #16,ERRPOS      ; SET THE POSITION
005106  005322                          BR      ROUT            ; BRANCH
005107  104307  002352          2$:     MOV     CHAINS,R0       ; POINT TO START OF CHAIN
005111  104652  000022                  MOV     U.MASK(R5),R2   ; R2 HAS SDI INTERCONNECT
005113  060012                          XFC     WAITSI          ; WAIT FOR SECTOR OR INDEX PULSE
```

RBLOCK - READ THE SECTORS

```
                                XFC     XREAD           ; WRITE THE SECTOR(S)
                                TST     R1              ; SEE IF ERROR OCCURRED
                                BEQ     REXT            ; IF NO ERROR, BRANCH
                                MOV     R1,U.RWER(R5)   ; SAVE ERROR TYPE
                                MOV     @CHAINS,R0      ; GET FIRST STATUS
                                BIT     #BUFFLG,R0      ; SEE IF FIRST BUFFER FULL
                                BNE     REXT            ; IF SO, BRANCH (NO ERROR)
                                CALL    BLKCHK          ; SEE IF THIS IS A KNOWN BAD BLOCK
                                BCC     REXT            ; IF SO, BRANCH
                                CMP     #3,R1           ; SEE IF BLOCK IS REVECTORED
                                BEQ     NHCFL           ; IF SO, BRANCH
                                BIT     #6,R1           ; SEE IF HEADER COMPARE FAILURE
                                BEQ     NTOUT           ; IF NOT, BRANCH
                                SOFTER  19,<U.RWTO(R5),RW.LOW(R0),RW.HI(R0),U.RBN(R5)>
005136  104200  002563  001012                                  MOV     #ER19,OUT.04
005141  104650  000006  001013                                  MOV     U.RWTO(R5),OUT.05
005144  104670  000002  001014                                  MOV     RW.LOW(R0),OUT.06
005147  104670  000003  001015                                  MOV     RW.HI(R0),OUT.07
005152  104650  000041  001016                                  MOV     U.RBN(R5),OUT.08
005155  104200  000023  001010                                  MOV     #19,OUT.02
005160  101200  001400  001010                                  BIS     #ERSOFT,OUT.02
005163  104200  005163  001007                                  MOV     #.,OUT.01
005166  104202  000013                                          MOV     #ERRMES,R2
005170  104020  001006                                          MOV     R2,OUT.RQ
                                ERRORC  <U.RBN+1(R5),RW.CMD(R0),U.CGRP(R5),U.CCYL(R5),U.CCYL+1(R5)>
005172  104650  000042  001017                                  MOV     U.RBN+1(R5),OUT.09
005175  104670  000004  001020                                  MOV     RW.CMD(R0),OUT.10
005200  104650  000055  001021                                  MOV     U.CGRP(R5),OUT.11
005203  104650  000053  001022                                  MOV     U.CCYL(R5),OUT.12
005206  104650  000054  001023                                  MOV     U.CCYL+1(R5),OUT.13
                                ERRORC  <U.LCYL(R5),U.LCYL+1(R5),U.LGRP(R5)>
005211  104650  000056  001024                                  MOV     U.LCYL(R5),OUT.14
005214  104650  000057  001025                                  MOV     U.LCYL+1(R5),OUT.15
005217  104650  000060  001026                                  MOV     U.LGRP(R5),OUT.16
                                ENDERR
005222  101200  000021  002357                                  BIS     #21,ERRPOS      ; SET THE POSITION
005225  005322                  BR      ROUT            ; BRANCH
005226                  NTOUT:  SOFTER  20,<U.RWTO(R5),RW.LOW(R0),RW.HI(R0),U.RBN(R5)>
005226  104200  002765  001012                                  MOV     #ER20,OUT.04
005231  104650  000006  001013                                  MOV     U.RWTO(R5),OUT.05
005234  104670  000002  001014                                  MOV     RW.LOW(R0),OUT.06
005237  104670  000003  001015                                  MOV     RW.HI(R0),OUT.07
005242  104650  000041  001016                                  MOV     U.RBN(R5),OUT.08
005245  104200  000024  001010                                  MOV     #20,OUT.02
005250  101200  001400  001010                                  BIS     #ERSOFT,OUT.02
005253  104200  005253  001007                                  MOV     #.,OUT.01
005256  104202  000013                                          MOV     #ERRMES,R2
005260  104020  001006                                          MOV     R2,OUT.RQ
                                ERRORC  <U.RBN+1(R5),RW.CMD(R0),U.CGRP(R5),U.CCYL(R5),U.CCYL+1(R5)>
005262  104650  000042  001017                                  MOV     U.RBN+1(R5),OUT.09
005265  104670  000004  001020                                  MOV     RW.CMD(R0),OUT.10
005270  104650  000055  001021                                  MOV     U.CGRP(R5),OUT.11
005273  104650  000053  001022                                  MOV     U.CCYL(R5),OUT.12
005276  104650  000054  001023                                  MOV     U.CCYL+1(R5),OUT.13
                                ERRORC  <U.LCYL(R5),U.LCYL+1(R5),U.LGRP(R5)>
005301  104650  000056  001024                                  MOV     U.LCYL(R5),OUT.14
005304  104650  000057  001025                                  MOV     U.LCYL+1(R5),OUT.15
```

```
     005307  104650  000060  001026                          MOV     U.LGRP(R5),OUT.T6
  46 005312                                  ENDERR
     005312  101200  000021  002357                          BIS     #21,ERRPOS      ; SET THE POSITION
  47 005315  005322                          BR      ROUT    ; BRANCH
  48 005316  104202  177777          NHCFL:  MOV     #-1,R2  ; FLAG AS REVECTOR FOUND
  49 005320  005322                          BR      ROUT    ; BRANCH
  50 005321  114002                  REXT:   CLR     R2      ; NO ERRORS
  51 005322  000000                  ROUT:   RETURN          ; RETURN TO CALLING PROGRAM;
```

```
                                .SBTTL  FNDRER - FILL IN THE READ CHAIN WITH THE CORRECT HEADERS
                                ;
                                ;       IN CASE OF PRIMARY REVECTORING, FILL IN THE CHAIN WITH THE CORRECT
                                ;       HEADERS
                                ;
005323                          PUSH    <R1,R4,R5>      ; SAVE REGS
005323  100461                                                          MOV R1,-(SP)
005324  100464                                                          MOV R4,-(SP)
005325  100465                                                          MOV R5,-(SP)
005326  104654  000037          MOV     U.CBN(R5),R4    ; GET LO ORDER CBN
005330  104655  000040          MOV     U.CBN+1(R5),R5  ; GET HI CBN
005332  104307  002352          MOV     CHAINS,R0       ; R0 POINTS TO START OF LIST
005334  100674  000002  1$:     MOV     R4,RW.LOW(R0)   ; MOVE TO CHAIN
005336  100675  000003          MOV     R5,RW.HI(R0)    ; MOVE TO CHAIN
005340  104171                  MOV     (R0),R1         ; GET STATUS OF BUFFER
005341  075355                  BMI     2$              ; IF END OF LIST, BRANCH
005342  102201  040000          BIT     #BUFFLG,R1      ; SEE IF BUFFER FULL
005344  015355                  BEQ     2$              ; IF NOT, BRANCH
005345  103201  170000          BIC     #^CHBHINB,R1    ; CLEAR UNUSED BITS
005347  104017                  MOV     R1,R0           ; MOVE TO NEW POINTER
005350  105204  000001          ADD     #1,R4           ; INCREMENT BN
005352  045334                  BCC     1$              ; IF NO CARRY, BRANCH
005353  115405                  INC     R5              ; PROPOGATE CARRY
005354  005334                  BR      1$              ; LOOP
005355                  2$:     POP     <R5,R4,R1>      ; RESTORE R1
005355  104265                                                          MOV (SP)+,R5
005356  104264                                                          MOV (SP)+,R4
005357  104261                                                          MOV (SP)+,R1
005360  000000                  RETURN                  ; RETURN TO CALLING MODULE
```

```
                                .SBTTL  ********** OVERLAY 7 - EDC + ECC CHECKING, DATA COMPARISON, RETRIES
005361                                  DMOVLY  CK,AREA0
005361  022162                          .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;********************************************* *******************************
                                ;
                                ;
000007                          CHKECC  =       7.              ; DATA CHECK OVERLAY
                                ;
                                ;       SEE IF ECC ERROR WAS DETECTED
                                ;
004515                          DIAG$$                          ; ************
004515  115000  002774                  TST     $$DIAG+$DIAG$
004517  014525                          BEQ     .+6
004520  104207  060000                  MOV     #60000,R0
004522  100707  176250                  MOV     R0,@$$DIAG+$DIAG$
004524  004525                          BR      .+1
004525  104307  002352                  MOV     CHAINS,R0       ; R0 POINTS TO CHAIN
004527  104671  000000                  MOV     RW.STAT(R0),R1  ; R1 CONTAINS STATUS
004531  102201  040000                  BIT     #BUFFLG,R1      ; SEE IF BUFFER FULL
004533  015262                          BEQ     EDCDET          ; IF NOT, BRANCH
004534  102201  010000                  BIT     #ECCFLG,R1      ; SEE IF ECC ERROR DETECTED
004536  015161                          BEQ     ECCEDT          ; IF NOT, BRANCH
004537  021613                          CALL    BLKCHK          ; SEE IF THIS IS A KNOWN BED BLOCK
004540  045262                          BCC     EDCDET          ; IF SO, BRANCH
004541  104643  000000                  MOV     S.PARM(R4),R3   ; GET SUBUNIT PARAMETERS
004543  102203  010000                  BIT     #ECCCHK,R3      ; SEE IF ECC CORRECTION REQUESTED
004545  054641                          BNE     CORECC          ; IF SO, BRANCH
004546                                  SOFTER  7,<U.TIMO(R5),RW.LOW(R0),RW.HI(R0),U.RBN(R5)>
004546  104200  001327  001012                                          MOV     #ER7,OUT.04
004551  104650  000005  001013                                          MOV     U.TIMO(R5),OUT.05
004554  104670  000002  001014                                          MOV     RW.LOW(R0),OUT.06
004557  104670  000003  001015                                          MOV     RW.HI(R0),OUT.07
004562  104650  000041  001016                                          MOV     U.RBN(R5),OUT.08
004565  104200  000007  001010                                          MOV     #7,OUT.02
004570  101200  001400  001010                                          BIS     #ERSOFT,OUT.02
004573  104200  004573  001007                                          MOV     #.,OUT.01
004576  104202  000013                                                  MOV     #ERRMES,R2
004600  104020  001006                                                  MOV     R2,OUT.RQ
004602                                  ERRORC  <U.RBN+1(R5),RW.CMD(R0),U.CGRP(R5),U.CCYL(R5),U.CCYL+1(R5)>
004602  104650  000042  001017                                          MOV     U.RBN+1(R5),OUT.09
004605  104670  000004  001020                                          MOV     RW.CMD(R0),OUT.10
004610  104650  000055  001021                                          MOV     U.CGRP(R5),OUT.11
004613  104650  000053  001022                                          MOV     U.CCYL(R5),OUT.12
004616  104650  000054  001023                                          MOV     U.CCYL+1(R5),OUT.13
004621                                  ENDERR  0
004621  114000  002357                                                  CLR     ERRPOS          ; CLEAR THE POSITION
004623  102203  001000                  BIT     #RTRIES,R3      ; SEE IF RETRIES ARE ENABLED
004625  055257                          BNE     ECCERR          ; IF SO, BRANCH
004626  103200  000400  001010          BIC     #400,OUT.02     ; MAKE SOFT ERROR A HARD ERROR
004631  104200  000014  001006          MOV     #ERRMC,OUT.RQ   ; COUNT ERROR
004634  104207  177777                  MOV     #-1,R0          ; FLAG THAT HARD ERROR OCCURRED
004636  100657  000024                  MOV     R0,U.ELEV(R5)   ; SAVE
004640  005263                          BR      CMPNXT          ; BRANCH
```

```
     36 004641                                  CORECC: PUSH    R0              ; SAVE POINTER TO LINK
        004641  100467                                                          MOV R0,-(SP)
     37 004642  060015                                  XFC     ECC             ; APPLY ECC CORRECTION
     38 004643  115001                                  TST     R1              ; SEE IF CORRECTION WORKED
     39 004644  014741                                  BEQ     ECCCOK          ; IF SO, BRANCH
     40 004645                                          POP     R0              ; RESTORE POINTER TO LINK
        004645  104267                                                          MOV (SP)+,R0
     41 004646                                          SOFTER  8,<U.TIMO(R5),RW.LOW(R0),RW.HI(R0),U.RBN(R5)>
        004646  104200  001444  001012                                          MOV     #ER8,OUT.04
        004651  104650  000005  001013                                          MOV     U.TIMO(R5),OUT.05
        004654  104670  000002  001014                                          MOV     RW.LOW(R0),OUT.06
        004657  104670  000003  001015                                          MOV     RW.HI(R0),OUT.07
        004662  104650  000041  001016                                          MOV     U.RBN(R5),OUT.08
        004665  104200  000010  001010                                          MOV     #8,OUT.02
        004670  101200  001400  001010                                          BIS     #ERSOFT,OUT.02
        004673  104200  004673  001007                                          MOV     #.,OUT.01
        004676  104202  000013                                                  MOV     #ERRMES,R2
        004700  104020  001006                                                  MOV     R2,OUT.RQ
     42 004702                                          ERRORC  <U.RBN+1(R5),RW.CMD(R0),U.CGRP(R5),U.CCYL(R5),U.CCYL+1(R5)>
        004702  104650  000042  001017                                          MOV     U.RBN+1(R5),OUT.09
        004705  104670  000004  001020                                          MOV     RW.CMD(R0),OUT.10
        004710  104650  000055  001021                                          MOV     U.CGRP(R5),OUT.11
        004713  104650  000053  001022                                          MOV     U.CCYL(R5),OUT.12
        004716  104650  000054  001023                                          MOV     U.CCYL+1(R5),OUT.13
     43 004721                                          ENDERR  0
        004721  114000  002357                                                  CLR     ERRPOS          ; CLEAR THE POSITION
     44 004723  102203  001000                          BIT     #RTRIES,R3      ; SEE IF RETRIES ARE ENABLED
     45 004725  055257                                  BNE     ECCERR          ; IF SO, BRANCH
     46 004726  103200  000400  001010                  BIC     #400,OUT.02     ; MAKE SOFT ERROR A HARD ERROR
     47 004731  104200  000014  001006                  MOV     #ERRMC,OUT.RQ   ; COUNT ERROR
     48 004734  104207  177777                          MOV     #-1,R0          ; FLAG THAT HARD ERROR OCCURRED
     49 004736  100657  000024                          MOV     R0,U.ELEV(R5)   ; SAVE
     50 004740  005263                                  BR      CMPNXT          ; BRANCH
     51 004741  106657  000027          ECCCOK: CMP     U.ECCT(R5),R0   ; SEE IF CORRECTIONS EXCEED THRESHOLD
     52 004743  035040                                  BPL     ECCTOK          ; IF NOT, BRANCH
     53 004744                                          POP     R0              ; RESTORE POINTER TO LINK
        004744  104267                                                          MOV (SP)+,R0
     54 004745                                          SOFTER  9,<U.TIMO(R5),RW.LOW(R0),RW.HI(R0),U.RBN(R5)>
        004745  104200  001547  001012                                          MOV     #ER9,OUT.04
        004750  104650  000005  001013                                          MOV     U.TIMO(R5),OUT.05
        004753  104670  000002  001014                                          MOV     RW.LOW(R0),OUT.06
        004756  104670  000003  001015                                          MOV     RW.HI(R0),OUT.07
        004761  104650  000041  001016                                          MOV     U.RBN(R5),OUT.08
        004764  104200  000011  001010                                          MOV     #9,OUT.02
        004767  101200  001400  001010                                          BIS     #ERSOFT,OUT.02
        004772  104200  004772  001007                                          MOV     #.,OUT.01
        004775  104202  000013                                                  MOV     #ERRMES,R2
        004777  104020  001006                                                  MOV     R2,OUT.RQ
     55 005001                                          ERRORC  <U.RBN+1(R5),RW.CMD(R0),U.CGRP(R5),U.CCYL(R5),U.CCYL+1(R5)>
        005001  104650  000042  001017                                          MOV     U.RBN+1(R5),OUT.09
        005004  104670  000004  001020                                          MOV     RW.CMD(R0),OUT.10
        005007  104650  000055  001021                                          MOV     U.CGRP(R5),OUT.11
        005012  104650  000053  001022                                          MOV     U.CCYL(R5),OUT.12
        005015  104650  000054  001023                                          MOV     U.CCYL+1(R5),OUT.13
     56 005020                                          ENDERR  0
        005020  114000  002357                                                  CLR     ERRPOS          ; CLEAR THE POSITION
     57 005022  102203  001000                          BIT     #RTRIES,R3      ; SEE IF RETRIES ARE ENABLED
```

```
     58 005024  055257                          BNE     ECCERR          ; IF SO, BRANCH
     59 005025  103200  000400  001010          BIC     #400,OUT.02     ; MAKE SOFT ERROR A HARD ERROR
     60 005030  104200  000014  001006          MOV     #ERRMC,OUT.RQ   ; COUNT ERROR
     61 005033  104207  177777                  MOV     #-1,R0          ; FLAG THAT HARD ERROR OCCURRED
     62 005035  100657  000024                  MOV     R0,U.ELEV(R5)   ; SAVE
     63 005037  005263                          BR      CMPNXT          ; BRANCH
     64 005040                          ECCTOK: POP     R3              ; RESTORE POINTER TO BUFFER
        005040  104263                                                                  MOV (SP)+,R3
     65 005041  104637  000001                  MOV     RW.BUF(R3),R0   ; R0 POINTS TO BUFFER
     66 005043  021143                          CALL    CMPEDC          ; COMPUTE EDC VALUE
     67 005044  106672  000400                  CMP     BF.EDC(R0),R2   ; SEE IF EDC VALUE MATCHES
     68 005046  055071                          BNE     1$              ; IF NOT, BRANCH
     69 005047                                  REPSFT  ,ECC            ; REPORT ECC CORRECTION
        005047  114000  001010                                                  CLR     OUT.02
        005051  104200  000001  001011                                         MOV     #1,OUT.03
        005054  100467                                                                  MOV R0,-(SP)
        005055  104207  000047                                                 MOV     #U.SNUM,R0
        005057  105057                                                         ADD     R5,R0
        005060  105657  000034                                                 ADD     U.SUBU(R5),R0
        005062  104170  001007                                                 MOV     (R0),OUT.01
        005064  104207  000007                                                 MOV     #T4SOFT,R0
        005066  020751                                                         CALL    HOSTRQ
        005067  104267                                                                  MOV (SP)+,R0
     70 005070  005262                          BR      EDCDET          ; EXIT
     71 005071  104027                  1$:     MOV     R2,R0           ; SAVE THE COMPUTED EDC
     72 005072                                  HARDER  10,<U.TIMO(R5),RW.!OW(R3),RW.HI(R3),U.RBN(R5)>
        005072  104200  001662  001012                                         MOV     #ER10,OUT.04
        005075  104650  000005  001013                                         MOV     U.TIMO(R5),OUT.05
        005100  104630  000002  001014                                         MOV     RW.LOW(R3),OUT.06
        005103  104630  000003  001015                                         MOV     RW.HI(R3),OUT.07
        005106  104650  000041  001016                                         MOV     U.RBN(R5),OUT.08
        005111  104200  000012  001010                                         MOV     #10,OUT.02
        005114  101200  001000  001010                                         BIS     #ERHARD,OUT.02
        005117  104200  005117  001007                                         MOV     #.,OUT.01
        005122  104202  000014                                                 MOV     #ERRMC,R2
        005124  104020  001006                                                 MOV     R2,OUT.RQ
     73 005126                                  ERRORC  <U.RBN+1(R5),RW.CMD(R3),U.CGRP(R5),U.CCYL(R5),U.CCYL+1(R5)>
        005126  104650  000042  001017                                         MOV     U.RBN+1(R5),OUT.09
        005131  104630  000004  001020                                         MOV     RW.CMD(R3),OUT.10
        005134  104650  000055  001021                                         MOV     U.CGRP(R5),OUT.11
        005137  104650  000053  001022                                         MOV     U.CCYL(R5),OUT.12
        005142  104650  000054  001023                                         MOV     U.CCYL+1(R5),OUT.13
     74 005145                                  ERRORC  <R0,BF.EDC(R0)> ; REPORT ERROR
        005145  104070  001024                                                 MOV     R0,OUT.14
        005147  104670  000400  001025                                         MOV     BF.EDC(R0),OUT.15
     75 005152                                  ENDERR  0
        005152  114000  002357                                                 CLR     ERRPOS          ; CLEAR THE POSITION
     76 005154  104207  177777                  MOV     #-1,R0          ; FLAG THAT HARD ERROR OCCURRED
     77 005156  100657  000024                  MOV     R0,U.ELEV(R5)   ; SAVE
     78 005160  005263                          BR      CMPNXT          ; BRANCH
     79 005161  104073                  ECCEDT: MOV     R0,R3           ; R3 POINTS TO LINK
     80 005162  104677  000001                  MOV     RW.BUF(R0),R0   ; R0 POINTS TO SECTOR
     81 005164  021143                          CALL    CMPEDC          ; COMPUTE EDC
     82 005165  106672  000400                  CMP     BF.EDC(R0),R2   ; SEE IF EDC VALUES ARE EQUAL
     83 005167  015262                          BEQ     EDCDET          ; IF SO, BRANCH
     84 005170                                  HARDER  24,<U.TIMO(R5),RW.LOW(R3),RW.HI(R3),U.RBN(R5)>
        005170  104200  003336  001012                                         MOV     #ER24,OUT.04
```

```
        005173  104650  000005  001013                                  MOV     U.TIMO(R5),OUT.05
        005176  104630  000002  001014                                  MOV     RW.LOW(R3),OUT.06
        005201  104630  000003  001015                                  MOV     RW.HI(R3),OUT.07
        005204  104650  000041  001016                                  MOV     U.RBN(R5),OUT.08
        005207  104200  000030  001010                                  MOV     #24,OUT.02
        005212  101200  001000  001010                                  BIS     #ERHARD,OUT.02
        005215  104200  005215  001007                                  MOV     #.,OUT.01
        005220  104202  000014                                          MOV     #ERRMC,R2
        005222  104020  001006                                          MOV     R2,OUT.RQ
     85 005224                          ERRORC  <U.RBN+1(R5),RW.CMD(R3),U.CGRP(R5),U.CCYL(R5),U.CCYL+1(R5)>
        005224  104650  000042  001017                                  MOV     U.RBN+1(R5),OUT.09
        005227  104630  000004  001020                                  MOV     RW.CMD(R3),OUT.10
        005232  104650  000055  001021                                  MOV     U.CGRP(R5),OUT.11
        005235  104650  000053  001022                                  MOV     U.CCYL(R5),OUT.12
        005240  104650  000054  001023                                  MOV     U.CCYL+1(R5),OUT.13
     86 005243                          ERRORC  <R2,BF.EDC(R0)> ; REPORT ERROR
        005243  104020  001024                                          MOV     R2,OUT.14
        005245  104670  000400  001025                                  MOV     BF.EDC(R0),OUT.15
     87 005250                          ENDERR  0
        005250  114000  002357                                          CLR     ERRPOS          ; CLEAR THE POSITION
     88 005252  104207  177777          MOV     #-1,R0          ; FLAG THAT HARD ERROR OCCURRED
     89 005254  100657  000024          MOV     R0,U.ELEV(R5)   ; SAVE
     90 005256  005263                  BR      CMPNXT
     91 005257  104207  005267  ECCERR: MOV     #ERCOV,R0       ; ERROR RECOVERY IS NEXT MODULE
     92 005261  005265                  BR      EXTCHK          ; BRANCH
     93 005262  114002          EDCDET: CLR     R2              ; NO ERRORS
     94 005263  104207  005440  CMPNXT: MOV     #CMPDAT,R0      ; DATA COMPARE IS NEXT MODULE
     95 005265  114001          EXTCHK: CLR     R1              ; IMMIDATE CALL TO NEXT MODULE
     96 005266  003127                  BR      JMPRET          ; RETURN
```

```
                                .SBTTL  ERCOV  - RETRIES DUE TO DATA ERRORS
005267                          ERCOV:
                                ;
                                ;       ERROR RECOVERY AND RETRIES
                                ;
005267  104207  005440                MOV     #CMPDAT,R0      ; DATA COMPARE IS NEXT MODULE IF NO RETRIES
005271  104641  000000                MOV     S.PARM(R4),R1   ; GET SUBUNIT PARAMETERS
005273  102201  001000                BIT     #RTRIES,R1      ; SEE IF RETRIES ARE ENABLED
005275  015340                        BEQ     EROUT           ; IF NOT, BRANCH
005276  104657  000005                MOV     U.TIMO(R5),R0   ; GET NUMBER OF RETRIES ALLREADY ATTEMPTED
005300  106657  000025                CMP     U.RTRY(R5),R0   ; COMPARE MAXIMUM COUNT WITH RETRIES ATTEMPTED
005302  015336                        BEQ     NLEV            ; IF RETRIES EXHAUSTED, BRANCH
005303  115407                        INC     R0              ; INCREMENT RETRY COUNT
005304  100657  000005                MOV     R0,U.TIMO(R5)   ; SAVE RETRY COUNT
005306  104657  000016                MOV     U.NSEC(R5),R0   ; SEE IF THIS ERROR IS IN THE 1ST SECTOR
005310  015322                        BEQ     1$              ; IF SO, BRANCH
005311  104651  000033                MOV     U.PARM(R5),R1   ; GET UNIT PARAMETERS
005313  101201  000020                BIS     #DATERR,R1      ; FLAG AS DATA ERROR
005315  100651  000033                MOV     R1,U.PARM(R5)   ; SAVE
005317  104207  000001                MOV     #SETUP,R0       ; SETUP IS NEXT MODULE CALLED
005321  005341                        BR      EREXT           ; BRANCH
005322  104207  000001        1$:     MOV     #1,R0           ; ONLY ALLOW 1 SECTOR TO BE WORKED ON
005324  100657  000017                MOV     R0,U.MSEC(R5)   ; SAVE
005326  104207  177777                MOV     #-1,R0          ; START READ RETRIES AT ZERO
005330  100657  000006                MOV     R0,U.RWTO(R5)   ; SAVE
005332  104207  000006                MOV     #READ,R0        ; READ IS NEXT MODULE
005334  104051                        MOV     R5,R1           ; DELAYED CALL TO READ
005335  005341                        BR      EREXT           ; BRANCH
005336  104207  005343        NLEV:   MOV     #NEWLEV,R0      ; NEW LEVEL OF ERROR RECOVERY WILL BE TRIED
005340  114001                EROUT:  CLR     R1              ; IMMIDATE CALL TO NEXT MODULE
005341  114002                EREXT:  CLR     R2              ; NO ERRORS
005342  003127                        BR      JMPRET          ; RETURN TO SEQNCR
```

```
                                        .SBTTL  NEWLEV - SET ERROR RECOVERY LEVEL TO NEW VALUE, THEN RETRY AGAIN
005343                                  NEWLEV:
                                        ;
                                        ;       INITIATE A NEW LEVEL OF ERROR RECOVERY
                                        ;
005343  104657  000024                  MOV     U.ELEV(R5),R0   ; GET CURRENT ERROR RECOVERY LEVEL
005345  055377                          BNE     LEVNZR          ; IF NON-ZERO, BRANCH
005346                                  HARDER  11              ; ALL LEVELS TRIED WITHOUT SUCCESS
005346  104200  002025  001012                                          MOV     #ER11,OUT.04
005351  104200  000013  001010                                          MOV     #11,OUT.02
005354  101200  001000  001010                                          BIS     #ERHARD,OUT.02
005357  104200  005357  001007                                          MOV     #.,OUT.01
005362  104202  000014                                                  MOV     #ERRMC,R2
005364  104020  001006                                                  MOV     R2,OUT.RQ
005366                                  ENDERR  0
005366  114000  002357                                                  CLR     ERRPOS          ; CLEAR THE POSITION
005370  104207  177777                  MOV     #-1,R0          ; FLAG THAT HARD ERROR OCCURRED
005372  100657  000024                  MOV     R0,U.ELEV(R5)   ; SAVE
005374  104207  005440                  MOV     #CMPDAT,R0      ; DATA COMPARE NEXT MODULE CALLED
005376  005415                          BR      NEWEXT          ; BRANCH TO EXIT
005377  104070  001565          LEVNZR: MOV     R0,ERRLEV       ; MOVE ERROR LEVEL TO SDI COMMAND
005401  117407                          DEC     R0              ; DECREMENT ERROR LEVEL
005402  100657  000024                  MOV     R0,U.ELEV(R5)   ; SAVE ERROR LEVEL
005404  104657  000033                  MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
005406  101207  020000                  BIS     #RESEEK,R0      ; FORCE A RESEEK ON THE NEXT PASS
005410  100657  000033                  MOV     R0,U.PARM(R5)   ; SAVE UNIT PARAMETERS
005412  104207  005417                  MOV     #SNDLEV,R0      ; SNDLEV IS NEXT MODULE
005414  114002                          CLR     R2              ; NO ERRORS
005415  114001          NEWEXT: CLR     R1              ; IMMIDATE CALL TO NEXT MODULE
005416  003127                  BR      JMPRET          ; RETURN TO SEQUENCER
```

```
                                      .SBTTL  SNDLEV - SEND THE NEW LEVEL OF ERROR RECOVERY TO THE DRIVE
  005417                              SNDLEV:
                                      ;
                                      ;       SNDLEV WILL SEND THE ERROR RECOVERY LEVEL TO THE DRIVE
                                      ;
  005417  104203  001520                      MOV     #CR.ERR,R3      ; POINT TO ERROR RECOVERY COMMAND
  005421  021204                              CALL    TALK            ; SEND SDI INTERCHANGE
  005422  115002                              TST     R2              ; SEE IF AN ERROR OCCURRED
  005423  015433                              BEQ     TALKOK          ; IF NOT, BRANCH
  005424                                      CERROR  5,#SER4         ; REPORT ERROR
  005424  104200  010172  001013                                                      MOV     #SER4,OUT.05
  005427  104207  005417                      MOV     #SNDLEV,R0      ; SNDLEV WILL BE NEXT MODULE CALLED *******
  005431  114001                              CLR     R1              ; CALL IMIDATELY
  005432  005437                              BR      SNDEX           ; BRANCH
  005433  104207  000004              TALKOK: MOV     #RDWRT,R0       ; RDWRT IS NEXT MODULE CALLED
  005435  104051                              MOV     R5,R1           ; DELAYED CALL TO NEXT ROUTINE
  005436  114002                              CLR     R2              ; NO ERRORS
  005437  003127                      SNDEX:  BR      JMPRET          ; RETURN
```

```
                                .SBTTL  CMPDAT - DATA COMPARISON ON READ BUFFER(S)
005440                          CMPDAT:
                                ;
                                ;       CMPDAT COMPARES THE PATTERN WITH THAT READ FROM THE SECTOR
                                ;
005440                          PUSH    <R4,R5>         ; SAVE R4,R5
005440  100464                                                          MOV R4,-(SP)
005441  100465                                                          MOV R5,-(SP)
005442  104657  000033          MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
005444  102207  000002  2$:     BIT     #DATCMP,R0      ; SEE IF DATA COMPARE REQUESTED
005446  015536                  BEQ     CSCEXT          ; IF NOT, BRANCH
005447  104307  002352          MOV     CHAINS,R0       ; R0 POINTS TO LINK
005451  104172                  MOV     (R0),R2         ; GET BUFFER STATUS
005452  102202  040000          BIT     #BUFFLG,R2      ; SEE IF BUFFER FULL
005454  015536                  BEQ     CSCEXT          ; IF NOT, BRANCH
005455  104677  000001          MOV     RW.BUF(R0),R0   ; R0 POINTS TO BUFFER
005457  104172                  MOV     (R0),R2         ; R2 HAS PATTEPN NUMBER (IN EACH NIBBLE)
005460  103202  177760          BIC     #LBLONB,R2      ; CLEAR UNUSED BITS
005462  104020  002440          MOV     R2,PNUM         ; SAVE PATTERN NUMBER
005464  104021                  MOV     R2,R1           ; BUILD PATTERN NUMBER WORD IN R1
005465  110702                  SWAB    R2              ; MOVE PATTERN NUMBER TO HI BYTE
005466  101021                  BIS     R2,R1           ; SET HI BITS
005467  110202                  ROL     R2              ; ROTATE TO HI NIBBLE
005470  110202                  ROL     R2
005471  110202                  ROL     R2
005472  110202                  ROL     R2
005473  103202  007777          BIC     #HBHINB,R2      ; CLEAR UNUSED BITS
005475  101021                  BIS     R2,R1           ; SET BITS
005476  110702                  SWAB    R2              ; MOVE TO LO BYTE
005477  101021                  BIS     R2,R1           ; SET BITS
005500  106271                  CMP     (R0)+,R1        ; SEE IF REDUNDANT PATTERN OK
005501  055542                  BNE     FWRD            ; IF NOT, BRANCH
005502  104302  002440          MOV     PNUM,R2         ; RESTORE R2
005504  104622  002475          MOV     PATPTR(R2),R2   ; POINT TO PATTERN
005506  104201  000001          MOV     #1,R1           ; R1 HAS OFFSET INTO BUFFER
005510  104024          XOPLP0: MOV     R2,R4           ; R4 POINTS TO LENGTH OF PATTERN
005511  104243                  MOV     (R4)+,R3        ; R3 CONTAINS LENGTH OF PATTERN
005512  106203  000001          CMP     #1,R3           ; SEE IF PATTERN IS 1 WORD LONG
005514  015527                  BEQ     ONEPAX          ; IF SO, BRANCH
005515  104245          XOPLP1: MOV     (R4)+,R5        ; R5 GETS 1 WORD OF THE DATA PATTERN
005516  106275                  CMP     (R0)+,R5        ; COMPARE PATTERN WORD TO SECTOR AREA
005517  055543                  BNE     CMPERR          ; IF NOMATCH, BRANCH
005520  115401                  INC     R1              ; INCREMENT OFFSET
005521  106201  000400          CMP     #SCTWRD+1,R1    ; SEE IF ENTIRE BUFFER COMPARED
005523  015536                  BEQ     CSCEXT          ; IF ALL WORDS COMPARED, BRANCH
005524  117403                  DEC     R3              ; DECREMENT COUNT OF WORDS IN PATTERN
005525  055515                  BNE     XOPLP1          ; IF DATA PATTERN UNFINISHED, BRANCH
005526  005510                  BR      XOPLP0          ; BRANCH
005527  104145          ONEPAX: MOV     (R4),R5         ; GET 1 WORD OF DATA PATTERN
005530  106275          XOPLP2: CMP     (R0)+,R5        ; COMPARE PATTERN TO SECTOR READ
005531  055543                  BNE     CMPERR          ; IF COMPARE ERROR, BRANCH
005532  115401                  INC     R1              ; INCREMENT NUMBER OF WORDS TO COMPARE
005533  106201  000400          CMP     #SCTWRD+1,R1    ; SEE IF ALL WORDS COMPARED
005535  055530                  BNE     XOPLP2          ; IF INCOMPLETE, BRANCH
005536  114002          CSCEXT: CLR     R2              ; NO ERRORS
005537                          POP     <R5,R4>         ; RESTORE R5,R4
005537  104265                                                          MOV (SP)+,R5
```

```
                                                                                    MOV (SP)+,R4
 55 005541  005712                     BR      CMXEXT          ; BRANCH
 56 005542  114001           FWRD:      CLR     R1              ; ZERO OFFSET
 57 005543                  CMPERR:     POP     <R5,R4>         ; RESTORE R5,R4
    005543  104265                                                                 MOV (SP)+,R5
    005544  104264                                                                 MOV (SP)+,R4
 58 005545                              PUSH    R0              ; SAVE PO'NTER TO BUFFER
    005545  100467                                                                 MOV R0,-(SP)
 59 005546  104037                     MOV     R3,R0           ; MOVE POINTER TO READ CHAIN TO R0
 60 005547  021613                     CALL    BLKCHK          ; SEE IF THIS IS A KNOWN BAD BLOCK
 61 005550                              POP     R0              ; RESTORE R0
    005550  104267                                                                 MOV (SP)+,R0
 62 005551                              BCS     1$              ; IF NOT, BRANCH
    005551  045553                                                                 BCC     .+2
    005552  005554                                                                 BR      1$
 63 005553  005712                     BR      CMXEXT          ; EXIT
 64 005554  104303  002352      1$:     MOV     CHAINS,R3       ; R3 POINTS TO LINK
 65 005556                              HARDER  12,<RW.LOW(R3),RW.HI(R3),U.RBN(R5),U.RBN+1(R5)>
    005556  104200  002057  001012                                                 MOV     #ER12,OUT.04
    005561  104630  000002  001013                                                 MOV     RW.LOW(R3),OUT.05
    005564  104630  000003  001014                                                 MOV     RW.HI(R3),OUT.06
    005567  104650  000041  001015                                                 MOV     U.RBN(R5),OUT.07
    005572  104650  000042  001016                                                 MOV     U.RBN+1(R5),OUT.08
    005575  104200  000014  001010                                                 MOV     #12,OUT.02
    005600  101200  001000  001010                                                 BIS     #ERHARD,OUT.02
    005603  104200  005603  001007                                                 MOV     #.,OUT.01
    005606  104202  000014                                                         MOV     #ERRMC,R2
    005610  104020  001006                                                         MOV     R2,OUT.RQ
 66 005612                              ERRORC  <RW.CMD(R3),U.CGRP(R5),U.CCYL(R5),U.CCYL+1(R5)>
    005612  104630  000004  001017                                                 MOV     RW.CMD(R3),OUT.09
    005615  104650  000055  001020                                                 MOV     U.CGRP(R5),OUT.10
    005620  104650  000053  001021                                                 MOV     U.CCYL(R5),OUT.11
    005623  104650  000054  001022                                                 MOV     U.CCYL+1(R5),OUT.12
 67 005626  106201  000004             CMP     #4,R1           ; SEE IF IN FIRST 4 WORDS
 68 005630  045636                     BCC     SMLSUB          ; IF SO, BRANCH
 69 005631  107207  000004             SUB     #4,R0           ; LOOK BACK FOUR WORDS
 70 005633  104203  000003             MOV     #3,R3           ; OFFSET OF ERROR IN PRINTPOUT IS 3
 71 005635  005641                     BR      RCMPER          ; BRANCH
 72 005636  107017          SMLSUB:    SUB     R1,R0           ; BACK UP POINTER
 73 005637  104013                     MOV     R1,R3           ; R3 IS OFFSET OF ERROR IN PRINTOUT
 74 005640  117407                     DEC     R0              ; TO POINT TO START OF BUFFER
 75 005641                  RCMPER:     ERRORC  <PNUM,R1,R3,(R0)+,(R0)+,(R0)+,(R0)+,(R0)+,(R0)+,(R0)+>
    005641  104300  002440  001023                                                 MOV     PNUM,OUT.13
    005644  104010  001024                                                         MOV     R1,OUT.14
    005646  104030  001025                                                         MOV     R3,OUT.15
    005650  104270  001026                                                         MOV     (R0)+,OUT.16
    005652  104270  001027                                                         MOV     (R0)+,OUT.17
    005654  104270  001030                                                         MOV     (R0)+,OUT.18
    005656  104270  001031                                                         MOV     (R0)+,OUT.19
    005660  104270  001032                                                         MOV     (R0)+,OUT.20
    005662  104270  001033                                                         MOV     (R0)+,OUT.21
    005664  104270  001034                                                         MOV     (R0)+,OUT.22
    005666  104270  001035                                                         MOV     (R0)+,OUT.23
 76 005670                              ERRORC  <(R0)+,(R0)+,(R0)+,(R0)+,(R0)+,(R0)+>
    005670  104270  001036                                                         MOV     (R0)+,OUT.24
    005672  104270  001037                                                         MOV     (R0)+,OUT.25
    005674  104270  001040                                                         MOV     (R0)+,OUT.26
```

```
                005676  104270  001041                                          MOV     (R0)+,OUT.27
                005700  104270  001042                                          MOV     (R0)+,OUT.28
                005702  104270  001043                                          MOV     (R0)+,OUT.29
            77  005704                          ENDERR  0
                005704  114000  002357                                          CLR     ERRPOS          ; CLEAR THE POSITION
            78  005706  104207  177777          MOV     #-1,R0          ; FLAG THAT HARD ERROR OCCURRED
            79  005710  100657  000024          MOV     R0,U.ELEV(R5)   ; SAVE
            80  005712  104657  000024  CMXEXT: MOV     U.ELEV(R5),R0   ; GET ERROR RECOVERY LEVEL
            81  005714  075744                  BMI     5$              ; IF HAR ERROR OCCURRED, BRANCH
            82  005715  106652  000026          CMP     U.MLEV(R5),R2   ; SEE IF ANY LEVELS USED
            83  005717  055723                  BNE     6$              ; IF SO, BRANCH
            84  005720  104652  000005          MOV     U.TIMO(R5),R2   ; GET RETRIES
            85  005722  015744                  BEQ     5$              ; IF NO RETRIES, BRANCH (NO ERRORS)
            86  005723                  6$:     REPSFT  SOFT            ; REPORT SOFT ERROR
                005723  104200  000001  001010                                  MOV     #1,OUT.02
                005726  114000  001011                                          CLR     OUT.03
                005730  100467                                                          MOV R0,-(SP)
                005731  104207  000047                                          MOV     #U.SNUM,R0
                005733  105057                                                  ADD     R5,R0
                005734  105657  000034                                          ADD     U.SUBU(R5),R0
                005736  104170  001007                                          MOV     (R0),OUT.01
                005740  104207  000007                                          MOV     #T4SOFT,R0
                005742  020751                                                  CALL    HOSTRQ
                005743  104267                                                          MOV (SP)+,R0
            87  005744  104657  000016  5$:     MOV     U.NSEC(R5),R0   ; GET NUMBER OF SECTORS HANDLED SO FAR
            88  005746  115407                  INC     R0              ; INCREMENT COUNT
            89  005747  100657  000016          MOV     R0,U.NSEC(R5)   ; SAVE
            90  005751  104701  174377          MOV     @CHAINS,R1      ; R1 IS STATUS OF LINK JUST CHECKED
            91  005753  102201  100000          BIT     #EOC,R1         ; SEE IF END OF CHAIN
            92  005755  055772                  BNE     1$              ; IF SO, BRANCH
            93  005756  103201  170000          BIC     #^CHBHINB,R1    ; CLEAR UNUSED BITS
            94  005760  104010  002352          MOV     R1,CHAINS       ; CHAINS NOW POINTS TO NEXT LINK
            95  005762  104111                  MOV     (R1),R1         ; GET STATUS OF NEXT BUFFER
            96  005763  102201  040000          BIT     #BUFFLG,R1      ; SEE IF EMPTY BUFFER
            97  005765  015772                  BEQ     1$              ; IF SO, BRANCH *****
            98                          ;       IF THE ABOVE BRANCH IS TAKEN, ONE OF 3 CONDITIONS EXIST:
            99                          ;               1) HEADER COMPARE FAILURE
           100                          ;               2) TIMEOUT DURING READ
           101                          ;               3) REVECTORED BLOCK
           102                          ;                       BOTH 1 AND 2 ARE ERRORS THAT ARE 'MASKED' BY THE PROGRAM
           103  005766  104207  000007                  MOV     #CHKECC,R0      ; CHKECC NEXT MODULE
           104  005770  114001                          CLR     R1              ; IMMIDATE CALL TO NEXT MODULE
           105  005771  006027                          BR      4$              ; BRANCH
           106  005772  104653  000033          1$:     MOV     U.PARM(R5),R3   ; GET UNIT PARAMETERS
           107  005774  103203  000200                  BIC     #RBNBN,R3       ; IF HANDLING AN RBN, CLEAR IT
           108  005776  100653  000033                  MOV     R3,U.PARM(R5)   ; SAVE
           109  006000  105647  000006                  ADD     S.MEGR(R4),R0   ; GET MEGABYTE COUNT
           110  006002  115002                          TST     R2              ; SEE IF ERROR ALLREADY OUTSTANDING
           111  006003  056022                          BNE     3$              ; IF SO, BRANCH
           112  006004  106207  003642                  CMP     #1954.,R0       ; SEE IF ONE MEGABYTE TRANSFERED
           113  006006  036022                          BPL     3$              ; IF NOT, BRANCH
           114  006007  107207  003642                  SUB     #1954.,R0       ; ZERO COUNT
           115  006011  104200  000001  001010          MOV     #1,OUT.02       ; REPORT 1 MEGABYTE READ
           116  006014  114000  001011                  CLR     OUT.03          ; NO WRITE MEGABYTES REPORTED
           117  006016  104202  000011                  MOV     #T4MXFR,R2      ; SET UP FOR MEGABIT REPORT
           118  006020  104020  001043                  MOV     R2,OUT.29       ; FLAG AS NON-ERROR
           119  006022  100647  000006          3$:     MOV     R0,S.MEGR(R4)   ; SAVE COUNT
```

```
 120 006024  104207  000001                  MOV     #SETUP,R0       ; SETUP IS NEXT MODULE CALLED
 121 006026  104051                          MOV     R5,R1           ; DEFERRED CALL TO NEXT MODULE
 122 006027  003127                  4$:     BR      JMPRET          ; RETURN TO SEQUENCER
 123         006031                  BUFA2   =       .+1             ; SET UP FOR SCRATCH AREA
 124                                         .IF     GE,BUFA1-BUFA2  ; SEE WHICH MODULE IS LARGER
 125                                 BUFARA  =       BUFA1           ; IF IT IS READ/WRITE, SCRACH STARTS AFTER THAT
 126                                         .IFF
 127         006031                  BUFARA  =       BUFA2           ; IF IT IS CHKECC, SCRATCH STARTS AFTERS THAT
 128                                         .ENDC
```

```
                                .SBTTL  ********** OVERLAY 8 - REVECTORED SECTOR HANDLING
                                        DMOVLY  RV,AREA0
006030  041147                          .WREDC          ;OUTPUT EDC FOR THIS OVERLAY
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*******************************************************************************
                                ;*********************************** *******************************************
                                ;
                                ;
                                ;
                                        .ENABL  LSB
000010                          REVCT   =       8.              ; REVECTOR OVERLAY
004515                                  DIAG$$                  ; ************
004515  115000  002775                  TST     $$DIAG+$DIAG$
004517  014525                          BEQ     .+6
004520  104207  060000                  MOV     #60000,R0
004522  100707  176251                  MOV     R0,@$$DIAG+$DIAG$
004524  004525                          BR      .+1
004525  104657  000033                  MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
004527  102207  000040                  BIT     #REVINP,R0      ; SEE IF REVECTOR ALLREADY IN PROGRESS
004531  054534                          BNE     1$              ; IF SO, BRANCH
004532  024562                          CALL    REVSUP          ; SETUP REVECTOR OPERATION
004533  004561                          BR      4$              ; EXIT
004534  104651  000021          1$:     MOV     U.CSEC(R5),R1   ; GET NUMBER OF SECTORS R/W SO FAR
004536  105651  000035                  ADD     U.MBN(R5),R1    ; ADD STARTING SECTOR OF OPERATION
004540  104010  002340                  MOV     R1,CURBN        ; MOVE TO TEMP STORAGE
004542  104651  000036                  MOV     U.MBN+1(R5),R1  ; GET HI STARTING SECTOR
004544  044546                          BCC     2$              ; IF NO CARRY, BRANCH
004545  115401                          INC     R1              ; PROPOGATE CARRY
004546  104010  002341          2$:     MOV     R1,CURBN+1      ; SAVE
004550  024625                          CALL    REVSOK          ; SEE IF RCT SECTOR JUST READ OS OK
004551  115001                          TST     R1              ; SEE IF TESTED OK
004552  054561                          BNE     4$              ; IF NOT, EXIT
004553  025037                          CALL    SEARCH          ; SEARCH THE SECTOR TO FIND THE LBN
004554  115001                          TST     R1              ; SEE IF LBN FOUND
004555  014561                          BEQ     4$              ; LBN FOUND, BRANCH AND READ RBN
004556  115002                          TST     R2              ; SEE IF NULL FLAG FOUND (REVECTOR NOT FOUND)
004557  054561                          BNE     4$              ; IF NOT, BRANCH
004560  025302                          CALL    NXTRCT          ; READ NEXT RCT SECTOR
004561  003127                  4$:     BR      JMPRET          ; RETURN CONTROL TO SEQUENCER
                                        .DSABL  LSB
```

REVSUP - SETUP THE REVECTOR OPERATION (TO READ RCT)

```
                                        .SBTTL  REVSUP - SETUP THE REVECTOR OPERATION (TO READ RCT)
004562                                  REVSUP:
                                        ;
                                        ;       INITILIZE ALL PARAMETERS FOR READING THE RCT TO FING THE RBN THAT
                                        ;       A HEADER HAS BEEN REVECTORED TO
                                        ;
004562  101207  000040                          BIS     #REVINP,R0      ; FLAG AS REVECTOR IN PROGRESS
004564  100657  000033                          MOV     R0,U.PARM(R5)   ; SAVE PARAMETERS
004566  104657  000045                          MOV     U.RWER(R5),R0   ; GET READ/WRITE ERROR TYPE
004570  100657  000046                          MOV     R0,U.RVER(R5)   ; SAVE FOR REVECTOR INFORMATION
004572  104641  000005                          MOV     S.SCHR(R4),R1   ; R1 POINTS TO SUBUNIT CHARACTERISTICS
004574  114007                                  CLR     R0              ; USE R0 TO INITILIZE VALUES
004575  100657  000041                          MOV     R0,U.RBN(R5)    ; START WITH RBN ZERO
004577  100657  000042                          MOV     R0,U.RBN+1(R5)  ; START WITH RBN ZERO
004601  100657  000044                          MOV     R0,U.CCOP(R5)   ; ON ORIGINAL COPY OF RCT
004603  115407                                  INC     R0              ; R0 IS NOW 1
004604  100657  000017                          MOV     R0,U.MSEC(R5)   ; ONLY READ 1 SECTOR AT A TIME
004606  115407                                  INC     R0              ; R0 IS NOW 2
004607  105617  000012                          ADD     LBNHST(R1),R0   ; R0 POINTS TO 1ST REVECTOR INFORMATION SECTOR
004611  100657  000037                          MOV     R0,U.CBN(R5)    ; SAVE
004613  104617  000013                          MOV     LBNHST+1(R1),R0 ; R0 HAS HI FIRST REV INFO SECTOR
004615  044617                                  BCC     1$              ; IF NO CARY, BRANCH
004616  115407                                  INC     R0              ; PROPOGATE CARRY
004617  100657  000040                  1$:     MOV     R0,U.CBN+1(R5)  ; SAVE
004621  114002                                  CLR     R2              ; NO ERRORS
004622  104207  000002                          MOV     #SEEK,R0        ; SEEK IS NEXT MODULE
004624  000000                                  RETURN
```

```
                                .SBTTL  REVSOK - SEE IF THE REVECTOR INFO SECTOR JUST READ IS OK
004625                          REVSOK:
                                ;
                                ;       SEE IF THE SECTOR WAS READ.  IF SO, SEE IF ANY ECC ERRORS WERE
                                ;       FOUND.  IF SO, TRY TO CORRECT.  IF SUCCESSFUL, RETURN R1 AS ZERO.
                                ;       IF NOT, GO TO THE NEXT RCT COPY
                                ;
004625  104303  002352                  MOV     CHAINS,R3       ; R3 POINTS TO SECTOR LINK
004627  104131                          MOV     (R3),R1         ; GET STATUS
004630  102201  040000                  BIT     #BUFFLG,R1      ; SEE IF SECTOR FULL
004632  014646                          BEQ     2$              ; IF NOT, GO TO NEXT COPY
004633  102201  010000                  BIT     #ECCFLG,R1      ; SEE IF ECC ERROR
004635  014644                          BEQ     1$              ; IF NOT, SECTOR OK
004636  060015                          XFC     ECC             ; CORRECT THE SECTOR
004637  115001                          TST     R1              ; SEE IF ANY ERRORS OCCURRED
004640  054646                          BNE     2$              ; IF SO, BRANCH
004641  106657  000027                  CMP     U.ECCT(R5),R0   ; SEE IF CORRECTIONS EXCEED THRESHOLD
004643  074646                          BMI     2$              ; IF SO, BRANCH
004644  114001                  1$:     CLR     R1              ; THIS SECTOR IS OK
004645  005036                          BR      7$              ; EXIT
004646  104651  000044          2$:     MOV     U.CCOP(R5),R1   ; GET NUMBER OF COPIES TRIED SO FAR
004650  115401                          INC     R1              ; TRY ANOTHER COPY
004651  106651  000043                  CMP     U.COPY(R5),R1   ; CHECK AGAINST MAX
004653  035014                          BPL     5$              ; IF ALL COPIES UNTRIED, BRANCH
004654  104651  000046                  MOV     U.RVER(R5),R1   ; GET ORIGINAL ERROR TYPE
004656  106201  000003                  CMP     #3,R1           ; SEE IF RBN HEADER WAS ORIGINALLY FOUND
004660  054716                          BNE     4$
004661                                  HARDER  40,<U.CBN(R5),U.CBN+1(R5),CURBN,CURBN+1>
004661  104200  004625  001012                                          MOV     #ER40,OUT.04
004664  104650  000037  001013                                          MOV     U.CBN(R5),OUT.05
004667  104650  000040  001014                                          MOV     U.CBN+1(R5),OUT.06
004672  104300  002340  001015                                          MOV     CURBN,OUT.07
004675  104300  002341  001016                                          MOV     CURBN+1,OUT.08
004700  104200  000050  001010                                          MOV     #40,OUT.02
004703  101200  001000  001010                                          BIS     #ERHARD,OUT.02
004706  104200  004706  001007                                          MOV     #.,OUT.01
004711  104202  000014                                                  MOV     #ERRMC,R2
004713  104020  001006                                                  MOV     R2,OUT.RQ
004715  005011                          BR      8$              ; SKIP NEXT ERRORS
004716  110601                  4$:     ROR     R1              ; SEE IF TIMEOUT
004717  044755                          BCC     3$              ; IF NOT, BRANCH
004720                                  HARDER  42,<U.CBN(R5),U.CBN+1(R5),CURBN,CURBN+1>
004720  104200  005055  001012                                          MOV     #ER42,OUT.04
004723  104650  000037  001013                                          MOV     U.CBN(R5),OUT.05
004726  104650  000040  001014                                          MOV     U.CBN+1(R5),OUT.06
004731  104300  002340  001015                                          MOV     CURBN,OUT.07
004734  104300  002341  001016                                          MOV     CURBN+1,OUT.08
004737  104200  000052  001010                                          MOV     #42,OUT.02
004742  101200  001000  001010                                          BIS     #ERHARD,OUT.02
004745  104200  004745  001007                                          MOV     #.,OUT.01
004750  104202  000014                                                  MOV     #ERRMC,R2
004752  104020  001006                                                  MOV     R2,OUT.RQ
004754  005011                          BR      8$              ; SKIP NEXT ERROR
004755                          3$:     HARDER  41,<U.CBN(R5),U.CBN+1(R5),CURBN,CURBN+1>
004755  104200  004735  001012                                          MOV     #ER41,OUT.04
004760  104650  000037  001013                                          MOV     U.CBN(R5),OUT.05
004763  104650  000040  001014                                          MOV     U.CBN+1(R5),OUT.06
```

REVSOK - SEE IF THE REVECTOR INFO SECTOR JUST READ IS OK

```
004766  104300  002340  001015                          MOV     CURBN,OUT.07
004771  104300  002341  001016                          MOV     CURBN+1,OUT.08
004774  104200  000051  001010                          MOV     #41,OUT.02
004777  101200  001000  001010                          BIS     #ERHARD,OUT.02
005002  104200  005002  001007                          MOV     #.,OUT.01
005005  104202  000014                                  MOV     #ERRMC,R2
005007  104020  001006                                  MOV     R2,OUT.R0
 35 005011  025251                  8$:     CALL    RVFAIL          ; CLEAR ALL BITS AND DO A FAIL EXIT
 36 005012  104051                          MOV     R5,R1           ; FLAG AS ERROR
 37 005013  005036                          BR      7$              ; EXIT
 38 005014  100651  000044          5$:     MOV     R1,U.CCOP(R5)   ; SAVE COPY COUNT
 39 005016  104647  000005                  MOV     S.SCHR(R4),R0   ; R0 POINTS TO SUBUNIT PARAMETERS
 40 005020  104677  000014                  MOV     RCTCSZ(R0),R0   ; R0 IS RCT COPY SIZE
 41 005022  105657  000037                  ADD     U.CBN(R5),R0    ; ADD CURRENT SECTOR
 42 005024  100657  000037                  MOV     R0,U.CBN(R5)    ; SAVE
 43 005026  045034                          BCC     6$              ; IF NO CARRY, BRANCH
 44 005027  104657  000040                  MOV     U.CBN+1(R5),R0  ; GET HI CURRENT BLOCK NUMBER
 45 005031  115407                          INC     R0              ; INCREMENT COUNT
 46 005032  100657  000040                  MOV     R0,U.CBN+1(R5)  ; SAVE
 47 005034  104207  000002          6$:     MOV     #SEEK,R0        ; SEEK IS NEXT MODULE
 48 005036  000000                  7$:     RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                .SBTTL  SEARCH - TRY TO FIND THE LBN IN THE RCT SECTOR JUST READ
005037                          SEARCH:
                                ;
                                ;       SEARCH THE RCT SECTOR JUST READ TO FIND THE LBN OR THE NULL ENTRY
                                ;
005037  104307  002352                  MOV     CHAINS,R0       ; R0 POINTS TO LINK (NODE) IN READ CHAIN
005041  104677  000001                  MOV     RW.BUF(R0),R0   ; R0 NOW POINTS TO BUFFER
005043  114001                          CLR     R1              ; R1 IS RBN NUMBER OFFSET WITHIN THE SECTOR
005044  104672  000001          1$:     MOV     1(R0),R2        ; GET RCT CODE
005046  075133                          BMI     6$              ; IF NULL POINTER, ENTIRE TABLE EXHAUSTED, BRANCH
005047  102202  020000                  BIT     #020000,R2      ; SEE IF IT IS A USED RBN
005051  015111                          BEQ     5$              ; IF NOT, BRANCH
005052  103202  170000                  BIC     #^CHBHINB,R2    ; CLEAR CODE
005054  106302  002341                  CMP     CURBN+1,R2      ; SEE IF HI ORDER MATCHES
005056  055111                          BNE     5$              ; IF NOT, BRANCH
005057  106170  002340                  CMP     (R0),CURBN      ; SEE IF LO ORDER MATCHES
005061  055111                          BNE     5$              ; IF NOT, BRANCH
005062  105651  000041                  ADD     U.RBN(R5),R1    ; ADD RUNNING RBN TO OFFSET
005064  100651  000041                  MOV     R1,U.RBN(R5)    ; SAVE
005066  045074                          BCC     2$              ; IF NO CARRY, BRANCH
005067  104651  000042                  MOV     U.RBN+1(R5),R1  ; GET HI ORDER RBN
005071  115401                          INC     R1              ; PROPOGATE CARRY
005072  100651  000042                  MOV     R1,U.RBN+1(R5)  ; SAVE
005074  104657  000033          2$:     MOV     U.PARM(R5),R0   ; GET UNIT PARAMETERS
005076  101207  000200                  BIS     #RBNBN,R0       ; FLAG ALL ROUTINES THAT THIS IS AN RBN
005100  100657  000033                  MOV     R0,U.PARM(R5)   ; SAVE
005102  025251                          CALL    RVFAIL          ; FAIL EXIT DOES WHAT WE WANT, JUST CLEAR R1 AND R2
005103  114001                          CLR     R1              ; NO SECTORS R/W SO FAR
005104  100651  000016                  MOV     R1,U.NSEC(R5)   ; SAVE
005106  114002                          CLR     R2              ; NO ERRORS
005107  114001                          CLR     R1              ; FOUND IT
005110  005250                          BR      7$              ; EXIT
005111  105207  000002          5$:     ADD     #2,R0           ; POINT TO NEXT RBN RECORD
005113  115401                          INC     R1              ; INCREMENT RBN OFFSET
005114  106201  000200                  CMP     #128.,R1        ; SEE IF ALL RECORDS TRIED
005116  055044                          BNE     1$              ; IF NOT, BRANCH
005117  114002                          CLR     R2              ; TO SIGNAL CALLING ROUTINE TO READ NEXT SECTOR
005120  105651  000041                  ADD     U.RBN(R5),R1    ; ADD OLD RBN TO 128
005122  100651  000041                  MOV     R1,U.RBN(R5)    ; SAVE
005124  045250                          BCC     7$              ; IF NO CARRY, EXIT
005125  104651  000042                  MOV     U.RBN+1(R5),R1  ; GET HI ORDER RBN
005127  115401                          INC     R1              ; PROPOGATE CARRY
005130  100651  000042                  MOV     R1,U.RBN+1(R5)  ; SAVE
005132  005250                          BR      7$              ; EXIT
005133  104651  000046          6$:     MOV     U.RVER(R5),R1   ; GET ORIGINAL ERROR
005135  106201  000003                  CMP     #3,R1           ; SEE IF REVECTORED BLOCK ORIGINALLY FOUND
005137  055167                          BNE     8$              ; IF NOT, BRANCH
005140                                  HARDER  43,<CURBN,CURBN+1>      ; REPORT ERROR
                                                                        MOV     #ER43,OUT.04
005140  104200  005171  001012                                          MOV     CURBN,OUT.05
005143  104300  002340  001013                                          MOV     CURBN+1,OUT.06
005146  104300  002341  001014                                          MOV     #43,OUT.02
005151  104200  000053  001010                                          BIS     #ERHARD,OUT.02
005154  101200  001000  001010                                          MOV     #.,OUT.01
005157  104200  005157  001007                                          MOV     #ERRMC,R2
005162  104202  000014                                                  MOV     R2,OUT.RQ
005164  104020  001006
005166  005246                          BR      9$              ; SKIP NEXT ERRORS
```

SEARCH - TRY TO FIND THE LBN IN THE RCT SECTOR JUST READ

```
     50 005167  110601                    8$:     ROR     R1              ; SEE IF TIMEOUT
     51 005170  045220                            BCC     3$              ; IF NOT, BRANCH
     52 005171                                    HARDER  45,<CURBN,CURBN+1>      ; REPORT ERROR
        005171  104200  005352  001012                                            MOV     #ER45,OUT.04
        005174  104300  002340  001013                                            MOV     CURBN,OUT.05
        005177  104300  002341  001014                                            MOV     CURBN+1,OUT.06
        005202  104200  000055  001010                                            MOV     #45,OUT.02
        005205  101200  001000  001010                                            BIS     #ERHARD,OUT.02
        005210  104200  005210  001007                                            MOV     #.,OUT.01
        005213  104202  000014                                                    MOV     #ERRMC,R2
        005215  104020  001006                                                    MOV     R2,OUT.RQ
     53 005217  005246                            BR      9$              ; SKIP NEXT ERROR
     54 005220                            3$:     HARDER  44,<CURBN,CURBN+1>      ; REPORT ERROR
        005220  104200  005256  001012                                            MOV     #ER44,OUT.04
        005223  104300  002340  001013                                            MOV     CURBN,OUT.05
        005226  104300  002341  001014                                            MOV     CURBN+1,OUT.06
        005231  104200  000054  001010                                            MOV     #44,OUT.02
        005234  101200  001000  001010                                            BIS     #ERHARD,OUT.02
        005237  104200  005237  001007                                            MOV     #.,OUT.01
        005242  104202  000014                                                    MOV     #ERRMC,R2
        005244  104020  001006                                                    MOV     R2,OUT.RQ
     55 005246  025251                    9$:     CALL    RVFAIL          ; CLEAR ALL BITS, FAIL EXIT
     56 005247  104051                            MOV     R5,R1           ; FLAG AS ERROR
     57 005250  000000                    7$:     RETURN                  ; RETURN TO CALLING PROGRAM
```

```
                                .SBTTL  RVFAIL - CLEAR ALL REVECTOR BITS, AND CALL SETUP NEXT
005251                          RVFAIL:
                                ;
                                ;       CLEAR ALL REVECTOR BITS, AND RESTORE U.CBN TO WHAT IT WAS
                                ;
005251  104651  000033                MOV     U.PARM(R5),R1   ; GET UNIT PARAMETERS
005253  103201  000440                BIC     #REVEC!REVINP,R1        ; CLEAR ALL REVECTOR BITS
005255  100651  000033                MOV     R1,U.PARM(R5)   ; SAVE
005257  104201  000001                MOV     #1,R1           ; FLAG THAT A SECTOR HAS BEEN READ (SKIPPED)
005261  100651  000016                MOV     R1,U.NSEC(R5)   ; SAVE
005263  104651  000021                MOV     U.CSEC(R5),R1   ; GET NUMBER OF SECTORS
005265  105651  000035                ADD     U.MBN(R5),R1    ; ADD STARTING SECTOR NUMBER
005267  100651  000037                MOV     R1,U.CBN(R5)    ; RESTORE CBN
005271  104651  000036                MOV     U.MBN+1(R5),R1  ; GET HI MBN
005273  045275                       BCC     1$              ; IF NO CARRY, BRANCH
005274  115401                       INC     R1              ; PROPOGATE CARRY
005275  100651  000040        1$:     MOV     R1,U.CBN+1(R5)  ; SAVE HI IN CBN
005277  104207  000001                MOV     #SETUP,R0       ; SETUP NEXT ROUTINE
005301  000000                       RETURN                  ; RETURN TO CALLING PROGRAM
```

```
      1                                    .SBTTL  NXTRCT - GET THE NEXT RCT SECTOR TO SEARCH
      2 005302                             NXTRCT:
      3                                    ;
      4                                    ;       INCREMENT CBN BY 1 AFTER SUBTACTING THE NUMMBER OF COPIES*RCT SIZE
      5                                    ;       THAT WAS SEARCHED THROUGH ON THE LAST PASS
      6                                    ;
      7 005302  104652  000037                  MOV     U.CBN(R5),R2    ; R2 HAS LO ORDER RCT BN
      8 005304  104653  000040                  MOV     U.CBN+1(R5),R3  ; R3 HAS HI ORDER RCT BN
      9 005306  104657  000044                  MOV     U.CCOP(R5),R0   ; GET NUMBER OF COPIES THAT HAVE BEEN READ
     10 005310  015322                          BEQ     3$              ; IF NO COPIES, BRANCH
     11 005311  104641  000005                  MOV     S.SCHR(R4),R1   ; R1 POINTS TO SUBUNIT CHARACTERISTICS
     12 005313  104611  000014                  MOV     RCTCSZ(R1),R1   ; R1 HAS RCT SIZE
     13 005315  107012                    1$:   SUB     R1,R2           ; SUBTRACT FROM LO ORDER BN
     14 005316  045320                          BCC     2$              ; IF NO BORROW, BRANCH
     15 005317  117403                          DEC     R3              ; PROPOGATE BORROW
     16 005320  117407                    2$:   DEC     R0              ; DECREMENT COUNT
     17 005321  055315                          BNE     1$              ; IF NO CARRY, BRANCH
     18 005322  105202  000001            3$:   ADD     #1,R2           ; INCREMENT CBN BY 1
     19 005324  045326                          BCC     4$              ; IF NO CARRY, BRANCH
     20 005325  115403                          INC     R3              ; PROPOGATE CARRY
     21 005326  100652  000037            4$:   MOV     R2,U.CBN(R5)    ; SAVE LO ORDER
     22 005330  100653  000040                  MOV     R3,U.CBN+1(R5)  ; SAVE HI ORDER
     23 005332  114002                          CLR     R2              ; NO ERRORS
     24 005333  104207  000002                  MOV     #SEEK,R0        ; NEXT MODULE IS RDWRT
     25 005335  000000                          RETURN
```

```
                                  .SBTTL    ********** OVERLAY 9 - RECALIBRATION MODULE
005336                                      DMOVLY   RB,AREA0
005336  025600                              .WREDC            ;OUTPUT EDC FOR THIS OVERLAY
                                  ;*************************************************************************
                                  ;*************************************************************************
                                  ;*************************************************************************
                                  ;*********************************** *************************************
                                  ;
                                  ;
                                  ;
                                            .ENABL   LSB
        000011                    RECALB =           9.                ; RECALIBRATION MODULE
004515                                      DIAG$$                     ; *********
004515  115000  002776                      TST      $$DIAG+$DIAG$
004517  014525                              BEQ      .+6
004520  104207  060000                      MOV      #60000,R0
004522  100707  176252                      MOV      R0,@$$DIAG+$DIAG$
004524  004525                              BR       .+1
004525  104203  001532                      MOV      #CR.INR,R3        ; POINT TO THE INITIATE RECALIBRATION ROUTINE
004527  115002                              TST      R2                ; SEE IF ANY ERRORS OCCURRED
004530  014540                              BEQ      1$                ; IF NOT, BRANCH
004531                                      CERROR   5,#SER7           ; REPORT SECONDARY ERROR
004531  104200  010275  001013                                                   MOV     #SER7,OUT.05
004534  104051                              MOV      R5,R1             ; DEFERRED CALL
004535  104207  000011                      MOV      #RECALB,R0        ; TRY RECALB AGAIN
004537  004567                              BR       2$                ; EXIT
004540  104657  000033            1$:       MOV      U.PARM(R5),R0     ; GET UNIT PARAMETERS
004542  101207  020001                      BIS      #RESEEK!RCLB,R0   ; FORCE A SEEK, MARK RECALIBRATION JUST DONE
004544  100657  000033                      MOV      R0,U.PARM(R5)     ; SAVE
004546  104657  000006                      MOV      U.RWTO(R5),R0     ; GET READ/WRITE TIMEOUT VALUE
004550  117407                              DEC      R0                ; ADJUST BECAUSE IT DIDN'T R/W LAST TIME
004551  101207  100000                      BIS      #100000,R0        ; MAKE THE RECALIBRATION COMPARE FAIL
004553  100657  000006                      MOV      R0,U.RWTO(R5)     ; SAVE
004555  114002                              CLR      R2                ; NO ERRORS
004556  100652  000053                      MOV      R2,U.CCYL(R5)     ; ZERO CYLINDER
004560  100652  000054                      MOV      R2,U.CCYL+1(R5)
004562  100652  000055                      MOV      R2,U.CGRP(R5)
004564  114001                              CLR      R1                ; IMMIDATE CALL
004565  104207  000002                      MOV      #SEEK,R0          ; SEEK IS NEXT MODULE
004567  003127                    2$:       BR       JMPRET            ; RETURN TO SEQNCR
                                            .DSABL   LSB
004570                                      DMEND
004570  140241                              .WREDC            ;OUTPUT EDC FOR THIS OVERLAY
        000001                              .END
```

SYMBOL TABLE

```
AFINIT  003254   CLCLBN  005042   DCOUT   006472   ER17    002442   FIRSTU= 007717
AFTERR  006621   CLCMAX  006766   DCREAD  006463   ER18    002470   FIXPAT  006413
ALLOCM  002112   CLRBUF  000777   DCYLS = 020000   ER19    002563   FNDBES  005275
AREA0 = 004515   CLROK   004331   DEC2    005445   ER2     000225   FNDLOP  005305
AREA1 = 006560   CLRUP   006343   DINIT = 000011   ER20    002765   FNDWER  005526
AREA2 = 007025   CLXEXT  002334   DIREC = 010000   ER21    003170   FNDWRT  005375
AREA3 = 007300   CMPDAT  005440   DIS     001544   ER23    003231   FORMAT= 000001
ATNEXT  003361   CMPEDC  001143   DISCON= 000204   ER24    003336   FRAME = 000004
ATNHI   007226   CMPERR  005543   DONE  = 000016   ER25    003470   FSTOP = 100000
ATTN  = 000002   CMPEXT  001656   DOWNG   006136   ER26    003544   FTIME = 001000
AVAIL = 000100   CMPNXT  005263   DOWNT   005774   ER27    003617   FTLDEV= 000400
BBLOP   005261   CMP2    001645   DRC     001550   ER28    003746   FTLSYS= 000000
BEEXT   005274   CMXEXT  005712   DROP  = 100000   ER29    004012   FT.BUF= 000000
BESDWN  005203   CNTLOP  004745   DROPSU  002472   ER3     000363   FT.HI = 000001
BESUP   005273   COMCHR  005615   DRPTST  003362   ER30    004051   FT.LOW= 000002
BEUSED= 000040   COMEXT  005710   DRTYPE= 000007   ER31    004247   FWRD    005542
BF.DAT= 000000   COMPAR= 000006   DRVCLR= 000005   ER32    004444   GCR     001552
BF.ECC= 000401   COMPDP  007302   DRVEXT  007466   ER33    004546   GETCHR= 000207
BF.EDC= 000400   COMPLT= 000176   DRVID = 000004   ER4     000547   GETMEM  005072
BHRQ    004152   COMPSC  006733   DRVONL= 000213   ER40    004625   GETSTA= 000011
BLDBB   005053   COPBB   005241   DRVRUN= 000014   ER41    004735   GETSUB= 000210
BLDBES  005022   COPFIN  005053   DUMSDI  002447   ER42    005055   GETU    007543
BLDSUB  005015   COPLP0  005033   D.LIMT= 000001   ER43    005171   GMPARM  007467
BLDSUS  004763   COPLP1  005040   D.SCHR= 000002   ER44    005256   GONE    003113
BLDUNT  004644   COPLP2  005050   ECC   = 000015   ER45    005352   GOTOBE  004772
BLKCHK  001613   COPPAT  005022   ECCCHK= 010000   ER5     000720   GOTOFS  004775
BONL    004265   COPYSU  005077   ECCCOK  004741   ER50    005443   GO4IT   003125
BORONO  005460   CORECC  004641   ECCEDT  005161   ER51    005510   GROUP   002346
BREAK = 000000   CORECT  003724   ECCERR  005257   ER52    005575   GRPCYL= 000002
BSUSEX  005011   COREXT  004025   ECCFLG= 010000   ER53    005660   GRPOFF= 000011
BUFARA= 006031   COUNTD  004753   ECCRSH= 000002   ER54    005674   GST     001555
BUFA1 = 005570   CPYBEX  005316   ECCTOK  005040   ER55    005743   GSTOK   004307
BUFA2 = 006031   CR.CLR  001455   ECHO  = 000010   ER56    006136   HBHINB= 007777
BUFFLG= 040000   CR.DIS  001525   ECHOC = 000350   ER57    006330   HBLONB= 170377
BUFSIZ= 000036   CR.ERR  001520   EDC     001157   ER58    006423   HDRPRE= 000005
BUILDP  001657   CR.GCR  001467   EDCDET  005262   ER59    006515   HD.BAD= 110000
BULDSC  004774   CR.GST  001501   EDCLOP  001157   ER6     001123   HD.DBN= 140000
BULDUM  002117   CR.INR  001532   EOC   = 100000   ER61    006540   HD.LBN= 000000
CALC    002141   CR.MOD  001462   ERCOV   005267   ER62    006766   HD.PRV= 050000
CALCSC  002336   CR.ONL  001513   ERECOV= 000006   ER7     001327   HD.RBN= 060000
CARYNO  005443   CR.RUN  001537   EREXT   005341   ER70    007230   HD.REV= 030000
CBB2    005441   CR.SCR  001474   ERHARD= 001000   ER71    007311   HD.XBN= 120000
CBES2   005347   CR.SEK  001506   ERLEV = 000002   ER72    007373   HEADER= 000002
CHAINS  002352   CSCEXT  005536   EROUT   005340   ER73    007472   HIBN    002335
CHECK = 000010   CURBN   002340   ERR     001564   ER74    007562   HIBYTE= 177400
CHGMOD= 000201   CVT   = 000020   ERRCNT  002443   ER75    007652   HICYL = 000001
CHKBB   006571   CYL     002344   ERRLEV  001565   ER76    007746   HIDBN = 000003
CHKBES  006311   CYLBN   006237   ERRMC = 000014   ER8     001444   HILBN = 000002
CHKCYL  006131   DATCMP= 000002   ERRMES= 000013   ER80    010365   HIMEM = 010000
CHKDN   003563   DATERR= 000020   ERRPOS  002357   ER9     001547   HIRBN = 000003
CHKECC= 000007   DATPRE= 000005   ERSOFT= 001400   EVRYOK  004446   HISEED  002446
CHKEXT  004500   DBNCYL= 000022   ER1     000062   EXIT  = 000021   HIXBN = 000002
CHKSTA  004370   DCEXT   006474   ER10    001662   EXTCHK  005265   HNDEX   004027
CHKTG   007034   DCLR    001551   ER100   010455   FB.DAT= 000000   HOSTRQ  000751
CHKUP   003614   DCMPAL= 000001   ER11    002025   FB.EDC= 000400   HRDREV= 000003
CHNMAX  002354   DCMPYS  006471   ER12    002057   FCTSIZ= 000010   HRQ     004154
CHRRES  000170   DCOMP   006446   ER16    002302   FILLIN  002014   IERR    003347
```

SYMBOL TABLE

```
INC2    005430  LOADED  003123  NOU     004730  OVE.RW= 004515      RBNFLG  002343
INDEX   002351  LOBYTE= 000377  NOUNIT  003020  OVE.SK= 006560      RBNTRK= 000004
INITD   007334  LOCYL   001561  NRTDS   003327  OVE.ST= 004515      RBUFLN= 000415
INITW = 040000  LONG    001532  NSTATE  003747  OVE.TS= 007025      RCBREQ  004367
INR     001557  LONGTO= 000001  NTOUT   005226  OVE.W = 004515      RCLB  = 000001
INS     001560  LOSEED  002445  NUMBB   004537  OVL.CK= 001314 G    RCMPER  005641
INSCHR  005724  LOW   = 000002  NUMOVL= 000012  OVL.MN- 007017 G    RCONT = 000000
INSEEK= 000012  LRDTRK  002342  NUMPTR= 000006  OVL.MS= 010644 G    RCTCPS= 000001
INSET   005562  LSTTRK  006321  NXTRCT  005302  OVL.R = 000645 G    RCTCSZ= 000014
INTEDC= 000105  MASK    005102  NXTSEC  004706  OVL.RB= 000054 G    RCV   = 000005
IN.RQ   001044  MASKLO  005033  OK2DRP  004436  OVL.RV= 000622 G    RCVERR= 000004
IN.01   001045  MASKLP  005104  ONEPAT  005047  OVL.RW= 000056 G    RCVRDY= 000001
IN.02   001046  MAXDBN  004540  ONEPAX  005527  OVL.SK= 000245 G    RDMOD   004556
IN.03   001047  MAXDST= 000003  ONEREG  005032  OVL.ST= 002043 G    RDSTAT  000720
IN.04   001050  MAXMUM  006534  ONL     001566  OVL.TS= 000253 G    RDWRT = 000004
IN.05   001051  MAXSEC  002355  ONLERR  007440  OVL.W = 001053 G    READ  = 000006
IN.06   001052  MBXFER  004150  ONLOK   004276  OVL...= 026656      RECALB= 000011
IN.07   001053  MEDTYP= 000006  ONLYCL= 000200  OVRLAY  003152      RECOVE  004021
IN.08   001054  MEMPOL  002473  OTABLE  002360  OVSTRT= 007774      RECOVR  004161
IN.09   001055  MESSAG= 000015  OUT.RQ  001006  OVS.CK= 052150 G    REDEXT  004727
IN.10   001056  MICREV= 000003  OUT.01  001007  OVS.MN= 001040 G    REDSET  004501
IN.11   001057  MKLOOP  005531  OUT.02  001010  OVS.MS= 017076 G    REDWRT= 000100
IN.12   001060  MOD     001546  OUT.03  001011  OVS.R = 050436 G    REPERR  004030
IN.13   001061  MODOK   004364  OUT.04  001012  OVS.RB= 056444 G    REPORT  003771
IN.14   001062  MODOK0  004365  OUT.05  001013  OVS.RV= 055000 G    RESEEK= 020000
IN.15   001063  MORE    007501  OUT.06  001014  OVS.RW= 046154 G    RETRY = 000004
IN.16   001064  MOVDBN  002224  OUT.07  001015  OVS.SK= 044714 G    RETS  = 000001
IN.17   001065  MOVOUT  002261  OUT.08  001016  OVS.ST= 040606 G    REVCT = 000010
IN.18   001066  MOV2    005462  OUT.09  001017  OVS.TS= 045426 G    REVEC = 000400
IN.19   001067  MRD   = 000016  OUT.10  001020  OVS.W = 046310 G    REVECT= 000004
IN.20   001070  MS1     010573  OUT.11  001021  OV... = 027276      REVINP= 000040
IN.21   001071  MS2     010611  OUT.12  001022  PATEXT  006415      REVS  = 000007
IN.22   001072  MWR   = 000017  OUT.13  001023  PATPTR  002475      REVSOK  004625
IN.23   001073  M.PARM  002444  OUT.14  001024  PATTRN  006374      REVSUP  004562
IN.24   001074  NEWDNT  006226  OUT.15  001025  PAT0    002515      REXT    005321
IN.25   001075  NEWEXT  005415  OUT.16  001026  PAT1    002536      RM    = 000004
IN.26   001076  NEWLEV  005343  OUT.17  001027  PAT10   002654      RNDBE   004741
IN.27   001077  NEWSUB= 004000  OUT.18  001030  PAT11   002675      RNDLP0  005021
IN.28   001100  NEWUPT  006172  OUT.19  001031  PAT12   002677      RNDLP1  005035
IN.29   001101  NEXTBE  005350  OUT.20  001032  PAT13   002720      RNDSUB  003664
IRECLB= 000216  NHCFL   005316  OUT.21  001033  PAT14   002741      RNDTG   005475
ISUEXT  007023  NLBEXT  005204  OUT.22  001034  PAT15   002745      RNDO    006407
ISUSEK  006677  NLEV    005336  OUT.23  001035  PAT2    002540      RONLY = 004000
JMPRET  003127  NOBB    005314  OUT.24  001036  PAT3    002542      ROOT    002777
LARGE = 000001  NOBBS   005071  OUT.25  001037  PAT4    002544      ROOTLP  003002
LASTU   004535  NOBES   005052  OUT.26  001040  PAT5    002565      RORW    006352
LBHINB= 177417  NOBORO  005007  OUT.27  001041  PAT6    002606      ROTEXT  003061
LBLONB= 177760  NOCARY  005047  OUT.28  001042  PAT7    002627      ROUT    005322
LBNCYL= 000000  NOERR   003145  OUT.29  001043  PAT8    002631      RREAL = 013400
LBNHLD  002441  NOEXTR  005276  OUT0    005313  PAT9    002652      RSTOP = 100000
LBNHST= 000012  NOHCFL  005516  OVERFL- 000002  PLOAD   003122      RTDS    001173
LBNTRK= 000011  NORCLB  004454  OVE.CK= 004515  PNUM    002440      RTDSE   007160
LEVNZR  005377  NORTDS  003267  OVE.MN- 000714  PREVBE  005314      RTDSEX  001203
LINK    001747  NOSEEK  006616  OVE.MS= 000000  PTABLE  002430      RTRIES= 001000
LINKLN- 000007  NOSUB   003151  OVE.R = 004515  RANDOM  006475      RUN     001556
LNCONT  002210  NOSUN   005007  OVE.RB= 004515  RBLOCK  004737      RVFAIL  005251
LNCYL   002202  NOTOUT  005367  OVE.RV- 004515  RBNBN - 000200      RWEXT   004565
```

SYMBOL TABLE

```
RWRDY = 100000  SHRTTO= 000000  ST.S1A= 000001  T4MPRM= 000003  U.PCTG= 000014
RW.ANG= 000006  SLOP    004773  ST.S7 = 000400  T4MXFR= 000011  U.RBN = 000041
RW.BUF= 000001  SMASK   007321  ST.UNT= 000000  T4SEEK= 000010  U.RTRY= 000025
RW.CMD= 000004  SMASKL  007326  ST.WE = 000010  T4SOFT= 000007  U.RVER= 000046
RW.HI = 000003  SMASKX  007332  SUB   = 000013  T4UPRM= 000004  U.RWER= 000045
RW.LOW= 000002  SMLSUB  005636  SUBUNT  001554  UCURSR  004534  U.RWTO= 000006
RW.SDI= 000005  SNDAGN  000756  SUSETL  004750  ULOP    004652  U.SDIL= 000031
RW.STA= 000000  SNDEX   005437  SWAPBB  005456  UMASK   004536  U.SDIS= 000030
SBCRES= 000167  SNDEXT  001453  SWAPBE  005364  UNITS   004514  U.SEEK= 000007
SCR     001553  SNDLEV  005417  S.BADP= 000010  UNIT0 = 000001  U.SNUM= 000047
SCTWRD= 000377  SORTBB  005414  S.BESS= 000011  UNIT1 = 000002  U.SUBP= 000001
SDIVER= 000000  SORTBE  005322  S.MCNT= 000011  UNIT2 = 000004  U.SUBU= 000034
SEARCH  005037  SORTTG  005502  S.MEGR= 000006  UNIT3 = 000010  U.TIMO= 000005
SECCYL  004545  SS    = 000001  S.MEGW= 000007  UNSSUP= 000175  U.TSEC= 000020
SECGRP  004544  ST      001570  S.PARM= 000000  UP      005223  U.WPRT= 000032
SECMAX  002353  STACK   001142  S.PAT = 000003  UPEXT   005425  U.WRIT= 000023
SECPTR  002356  STACYL  002336  S.SCHR= 000005  UPG     006056  WAITSI= 000012
SECTOR  002350  START   004555  S.SDCL= 000001  UPLBN   005407  WBLOCK  005057
SECTRK  004543  STATEX  000746  S.TGOF= 000013  UPT     005715  WBUFLN= 000401
SEEK  = 000002  STATLP  000724  S.TGSS= 000015  UREAD = 000013  WCHECK= 000010
SEKCNT  007074  STATOK  000737  S.TRKL= 000004  USED    004542  WCHK    006416
SEKEXT  006624  STATUS= 000007  TALK    001204  UTOTST= 000012  WCHKAL= 000004
SEKINP= 002000  STSERR  007102  TALKOK  005433  UWRITE= 000014  WCONT = 040000
SEKOUT  006623  STSERX  007273  TEST4 = 000001  U.CBN = 000037  WEXT    005524
SEKTST= 000003  STSEXT  007276  TGS     004546  U.CCNT= 000012  WONLY = 002000
SEND  = 000004  STSRES= 000366  THREBE  004767  U.CCOP= 000044  WOUT    005525
SEQBE   005117  ST.C  = 000002  TIMEOU= 000001  U.CCYL= 000053  WREAL = 122400
SEQLP   003064  ST.CON= 000002  TLEN.U= 000061  U.CGRP= 000055  WRITBT  001547
SEQNCR  003062  ST.DB = 001000  TMEMPL  002474  U.COPY= 000043  WRITE = 000005
SEQSEK= 000100  ST.DF = 000020  TMPERR  007020  U.CSEC= 000021  WRONG = 000002
SEQTG   005641  ST.DR = 000040  TO      005712  U.CTRK= 000015  WRTEXT  004773
SER0    010054  ST.ERR= 000002  TOOBIG= 000001  U.ECCT= 000027  WSTOP = 140000
SER1    010100  ST.FE = 000200  TRACK   002347  U.ELEV= 000024  WTIMOT  005426
SER2    010124  ST.FO = 002000  TRACKS= 000020  U.LCYL= 000056  XBNCYL= 000021
SER3    010146  ST.MOD  000001  TRKGRP= 000003  U.LGRP= 000060  XFERRT= 000000
SER4    010172  ST.MSK= 000000  TROOT   005527  U.MASK= 000022  XMTERR= 000400
SER5    010220  ST.OA = 000200  TRYAGN  004755  U.MBN = 000035  XOPLP0  005510
SER6    010254  ST.PE = 000040  TSTNEC  006625  U.MLEV= 000026  XOPLP1  005515
SER7    010275  ST.PS = 000002  TSTRCB  004350  U.MSEC= 000017  XOPLP2  005530
SER8    010317  ST.RE = 000100  T1MSIZ= 000000  U.NEXT= 000000  XREAD = 000002
SER9    010345  ST.RR = 000100  T2CMD = 000002  U.NFUN= 000010  XWRITE= 000003
SETRLT  005630  ST.RTY= 000003  T2DLL = 000001  U.NSEC= 000016  ZEREDC  001170
SETSEC  006310  ST.RU = 000001  T4BB1 = 000005  U.PARM= 000033  $DIAG$= 000011
SETUP = 000001  ST.SR = 000020  T4BB2   000006  U.PAT = 000011  $$DIAG  002766

. ABS.  056574    000
        000000    001
ERRORS DETECTED:  0

VIRTUAL MEMORY USED:  6249 WORDS  ( 25 PAGES)
DYNAMIC MEMORY AVAILABLE FOR  70 PAGES
A:UDAT4.BIN,SY:UDAT4/C=[1,33]DMACRO,UDAT4T,UDATP,UDATR,UDAT4M
```

CROSS REFERENCE TABLE (CREF V01-05)

```
$$DIAG  46-3#    94-17*   94-17*   124-18*  124-18*  127-16*  127-16*  128-16*  128-16*  129-15*  129-15*  134-15*  134-15*  137-14*
        137-14*  142-12*  142-12*  148-12*  148-12*
$DIAG$  46-2#    94-17    94-17#   94-17*   94-17*   124-18   124-18#  124-18*  124-18*  127-16   127-16#  127-16*  127-16*  128-16
        128-16#  128-16*  128-16*  129-15   129-15#  129-15*  129-15*  134-15   134-15#  134-15*  134-15*  137-14   137-14#  137-14*
        137-14*  142-12   142-12#  142-12*  142-12*  148-12   148-12#  148-12*  148-12*
AFINIT  49-2#    49-20    54-33    73-14
AFTERR  124-35   124-46#
ALLOCM  40-12    40-15    40-26    41-19    42-2#
AREA0   45-70    58-13#   94-1     94-1     128-2    128-2    129-2    129-2    134-2    134-2    137-2    137-2    142-2    142-2
        148-2    148-2
AREA1   45-72    123-22#  124-2    124-2
AREA2   45-74    126-38#  127-2    127-2
AREA3   45-76    127-60#
ATNEXT  49-19    49-25    49-38#
ATNHI   127-21   127-54#
ATTN     6-22#   54-13    127-20
AVAIL    6-24#
BBLOP   67-85#   67-99
BEEXT   99-55    99-85    99-87#
BESDWN  99-29    99-45#
BESUP   99-72    99-86#
BEUSED   8-35#   64-9     67-14    76-42    76-56    79-33    79-49    95-14    95-20
BF.DAT   3-62#
BF.ECC   3-64#
BF.EDC   3-63#   130-23*  137-67   137-74   137-82   137-86
BHRQ    55-62#
BLDBB   63-7     65-2#
BLDBES  63-6     64-2#
BLDSUB  62-15    63-2#
BLDSUS  61-49    62-2#
BLDUNT  60-3     61-2#
BLKCHK  37-2#    132-38   135-32   137-21   141-60
BONL     56-9     56-11    56-24    56-29#
BORONO  105-10   105-14#
BREAK    4-3#    91-26
BSUSEX  62-17    62-19#
BUFA1   133-27#  141-124
BUFA2   141-123# 141-124  141-127
BUFARA  60-6     141-127#
BUFFLG   3-47#   133-14   135-30   136-14   137-17   141-12   141-96   144-10
BUFSIZ  22-15    22-20    23-68#
BUILDP  39-2#    129-52   134-52
BULDSC  129-56   130-2#
BULDUM  43-2#    129-55   134-55
CALC    39-18    44-2#    125-16
CALCSC  44-52    45-4#
CARYNO  104-10   104-14#
CBB2    70-12    70-23#   79-13    79-20    79-30    79-46    80-14    80-24    80-31
CBES2   68-12    68-23#
CHAINS  39-17*   40-7     40-16*   45-15#   132-23   133-10   135-22   135-29   136-9    137-15   141-10   141-64   141-90   141-94*
        144-8    145-6
CHECK    5-67#
CHGMOD   6-32#   36-22
CHKBB   76-52    80-2#
CHKBES  76-49    79-2#
CHKCYL  76-46    77-2#
```

CROSS REFERENCE TABLE (CREF V01-05)

```
CHKDN    50-74    50-78    51-1#
CHKECC  134-73   137-10#  141-103
CHKEXT   56-69    57-23    57-28    57-45#
CHKSTA   56-41    57-2#
CHKTG    76-58    83-2#
CHKUP    50-59    50-71    50-81    52-1#
CHNMAX   45-17#   60-28*   94-82    94-84
CHRRES    6-43#   36-7
CLCLBN   97-27    97-35#
CLCMAX   76-40    82-2#
CLRBUF   22-25#   22-27
CLROK    56-50    56-53#
CLRUP    94-60   117-2#
CLXEXT   44-56    44-60#
CMP2     37-12    38-2#    99-28   100-13
CMPDAT  137-94   138-6    139-12   141-2#
CMPEDC   33-2#   130-22   137-66   137-81
CMPERR  141-40   141-49   141-57#
CMPEXT   38-13    38-16#
CMPNXT  137-35   137-50   137-63   137-78   137-90   137-94#
CMXEXT  141-55   141-63   141-80#
CNTLOP   96-11#   96-15
COMCHR   74-14    75-2#    75-12
COMEXT   75-13    75-43#
COMPAR    4-9#
COMPDP   84-10    84-31    84-47    85-2#
COMPLT    6-41#   36-5     36-6     36-10    36-11    36-12    36-13    36-15    36-16
COMPSC   76-39    81-2#
COPBB    67-42    67-60    67-75#
COPFIN  131-17   131-25#
COPLP0  131-10#  131-20
COPLP1  131-14#  131-19
COPLP2  131-22#  131-24
COPPAT  130-20   131-2#
COPYSU   63-9     67-2#
CORECC  137-25   137-36#
CORECT   47-38    54-2#
COREXT   54-14    54-17    54-24    54-38    54-40#
COUNTD   96-12    96-16#
CPYBEX   67-104   67-106#
CR.CLR   36-5#    56-47
CR.DIS   36-13#   57-18
CR.ERR   36-12#  140-6
CR.GCR   36-7#    75-7
CR.GST   36-9#    55-26    56-35    89-20
CR.INR   36-15#   56-64   148-13
CR.MOD   36-6#    56-56
CR.ONL   36-11#   56-29
CR.RUN   36-16#
CR.SCR   36-8#    76-23
CR.SEK   36-10#  126-9
CSCEXT  141-9    141-13   141-43   141-53#
CURBN    39-10*   39-11*   39-13*   39-14*   39-30*   39-3.*   41-28    41-30    44-57    44-57    45-6#   125-10*  125-11*  125-13*
        125-14*  126-32   126-32   142-20*  142-24*  144-28   144-28   144-32   144-32   144-34   144-34   145-14   145-16*  145-48
        145-48   145-52   145-52   145-54   145-54
CVT       4-19#   44-54
CYL      39-21*   39-23*   45-9#   125-19   126-6    126-7    126-24   126-32   126-32
```

CROSS REFERENCE TABLE (CREF V01-05)

```
CYLBN   77-11    77-24    78-2#
D.LIMT   8-4#    43-12*
D.SCHR   8-5#    43-14*
DATCMP   8-40#  117-8    12'-7   121-17   141-8
DATERR   8-21#   94-32   117-8   128-35   128-38   138-18
DATPRE   5-33#  132-26
DBNCYL   5-42#   82-10    82-18
DCEXT  121-8    121-13   121-20#
DCLR    36-25#   56-46*
DCMPAL   8-42#  121-14
DCMPYS 121-15   121-18#
DCOMP   94-64   121-2#
DCOUT  121-19#
DCREAD 121-11   121-14#
DCYLS    8-27#   41-34    44-11    57-38    76-31    77-34    79-27    80-21   129-33   134-33
DEC2    99-30   105-2#
DINIT    4-12#   54-28    89-10
DIREC    8-13#   50-14    50-69    51-20    52-30    99-7    108-8
DIS     36-13    36-20#
DISCON   6-30#   36-20
DONE     6-17#   46-35    48-27    60-19    91-23
DOWNG  108-14   112-2#
DOWNT  108-12   110-2#   112-23
DRC      36-5    36-24#
DROP     8-10#   47-42    49-34    50-62    50-85    53-22    57-25    76-90
DROPSU  45-91#   47-37*   47-39    54-23    54-34*   55-64*
DRPTST  47-11    50-2#
DRTYPE   5-15#
DRVCLR   6-35#   36-24
DRVEXT  87-8     87-52    87-57#
DRVID    5-14#
DRVONL   6-33#   36-38
DRVRUN   6-34#   36-30
DUMSDI  41-24    43-12*   43-14*   45-85#   45-86
ECC      4-16#  137-37   144-1.
ECCCHK   8-28#  137-24
ECCCOK 137-39   137-51#
ECCEDT 137-20   137-79#
ECCERR 137-30   137-45   137-58   137-91#
ECCFLG   3-45#  137-19   144-12
ECCRSH   5-11#   75-24
ECCTOK 137-52   137-64#
ECHO     4-11#
ECHOC    6-46#
EDC     33-10    33-13#
EDCDET 137-18   137-22   137-70   137-83   137-93#
EDCLOP  33-20#   33-26
EOC      3-46#   40-19    40-27   141-91
ER1     56-25    92-14#
ER10    92-79#  137-72
ER100   57-15    92-329#
ER11    92-87#  139-8
ER12    92-89#  141-65
ER16    92-99#  127-50
ER17    54-19    92-105#
ER18    92-107# 132-61
ER19    92-111# 135-38
```

CROSS REFERENCE TABLE (CREF V01-05)

CROSS REFERENCE TABLE (CREF V01-05)

```
ERRMC     6-15#   35-14   35-31   35-36   35-40   35-44   35-46   35-52   48-23   49-29   49-31   54-19   56-25   57-14
         57-15    60-15   60-17   78-22   79-15   79-22   79-37   79-42   79-53   79-55   80-18   80-27   80-33   83-41
         84-29    91-18   91-21  127-46  127-50  129-40  132-61  134-40  137-32  137-47  137-60  137-72  137-84  139-8
        141-65   144-28  144-32  144-34  145-48  145-52  145-54
ERRMES    6-14#   44-57   48-25   87-53  126-32  127-54  132-12  132-17  132-48  132-64  135-12  135-17  135-38  135-43
        137-26   137-41  137-54
ERRPOS   35-15*   35-47*  35-54*  44-59*  45-21#  49-32*  54-20*  54-22*  55-12   55-23*  56-26*  57-16*  87-54*  126-33*
        126-34*  127-48*  127-52*  127-56*  129-41*  132-15*  132-20*  132-51*  132-62*  132-67*  134-41*  135-15*  135-20*  135-41*
        135-46*  137-28*  137-43*  137-56*  137-75*  137-87*  139-9*  141-77*
ERSOFT    6-53#   44-57   87-53  126-32  127-54  132-12  132-17  132-48  132-64  135-12  135-17  135-38  135-43  137-26
        137-41   137-54
EvRYOK   57-13   57-29#
EXIT      4-20#   46-37   48-29   60-21   91-25
EXTCHK  137-92   137-95#
FB.DAT    3-22#
FB.EDC    3-23#
FCTSIZ    5-36#
FILLIN   40-30   41-2#
FIRSTU    7-11#   46-14   59-19   61-45   61-47   61-53   73-7    73-20
FIXPAT  119-12   119-16#
FNDBES   99-26   99-71   100-2#
FNDLOP  100-11#  100-15
FNDWER  132-31   133-2#
FNDWRT  132-56#  132-58
FORMAT    4-4#
FRAME     5-66#
FSTOP     3-42#
FT.BUF    3-15#
FT.HI     3-16#
FT.LOW    3-17#
FTIME     8-16#   50-11   50-14   52-28   73-9    73-17
FTLDEV    6-51#   48-23   49-29   49-31   57-15   60-15   78-22   79-15   79-22   79-37   79-42   79-53   79-55   80-18
         80-27    80-33   83-41   84-29   91-18  132-61
FTLSYS    6-50#
FWRD    141-30   141-56#
GCR      36-7     36-26#
GETCHR    6-36#   36-26
GETMEM   61-23    63-8    66-2#   76-63
GETSTA    6-38#   36-29
GETSUB    6-37#   36-27
GETU     60-2     89-2#
GMPARM   60-1     88-2#
GO4IT    47-15    47-35#
GONE     47-28#   47-32
GOTOBE   96-21    96-24   96-27#
GOTOFS   96-17    96-30#
GROUP    39-25*   45-10#  126-8   126-32
GRPCYL    5-25#   81-16   83-30   84-40
GRPOFF    5-38#
GST      36-9     36-29#
GSTOK    56-38    56-41#
HBHINB    5-56#   40-21   44-32   61-52   83-22   90-7    91-10   99-38   99-67   101-26  102-27  133-19  136-16  141-25
        141-93   145-13
HBLONB    5-57#   44-23   44-27   44-50
HD.BAD    3-55#
HD.DBN    3-58#   41-36
HD.LBN    3-52#
```

CROSS REFERENCE TABLE (CREF V01-05)

```
HD.PRV    3-56#
HD.RBN    3-53#   41-38
HD.REV    3-54#
HD.XBN    3-57#
HDRPRE    5-34#
HEADER    5-62#
HIBN     41-39    44-28*  44-51*  45-2#
HIBYTE    5-54#   44-14   44-38   56-45   68-28   69-18   70-28   70-30   71-14   72-13   75-22   75-26   76-34   76-36
         81-10    81-17   82-12   82-19   83-33   83-37   84-9    84-25   84-44   84-45   106-11  107-58  107-85  108-30
        115-8    123-14  129-19  132-27  134-19
HICYL     5-24#   44-31
HIDBN     5-30#   44-45
HILBN     5-26#   44-26
HIMEM     3-9#     7-11   45-93
HIRBN     5-29#   44-22
HISEED   45-84#   122-9   122-24*
HIXBN     5-27#
HNDEX    54-35    54-41#
HOSTRQ   22-2#    46-32   46-36   48-26   48-28   50-57   55-63   60-18   60-20   64-8    65-9    67-9    67-78   67-90
         88-25    91-5    91-22   91-24   127-35  129-83  134-72  137-69  141-86
HRDREV    5-13#
HRQ      55-60    55-63#
IERR     49-30    49-32#
IN.01    23-39#   55-64   64-9*   65-11   67-13   67-92   88-26   91-6
IN.02    23-40#   67-11   67-80
IN.03    23-41#   64-15   67-19   67-45   67-46*
IN.04    23-42#
IN.05    23-43#
IN.06    23-44#
IN.07    23-45#   64-12   67-40
IN.08    23-46#
IN.09    23-47#
IN.10    23-48#
IN.11    23-49#
IN.12    23-50#
IN.13    23-51#
IN.14    23-52#
IN.15    23-53#
IN.16    23-54#
IN.17    23-55#
IN.18    23-56#
IN.19    23-57#
IN.20    23-58#
IN.21    23-59#
IN.22    23-60#
IN.23    23-61#
IN.24    23-62#
IN.25    23-63#
IN.26    23-64#
IN.27    23-65#
IN.28    23-66#
IN.29    23-67#
IN.RQ    22-19    23-38#  23-68
INC2     99-73   104-2#
INDEX    45-13#
INITD    76-84    87-2#   87-55
INITW     8-11#   46-21   50-28   50-55   50-64   52-18   52-27   87-15   87-21   94-44   95-9    111-11  118-8   120-12
```

CROSS REFERENCE TABLE (CREF V01-05)

```
        123-11
INR      36-15    36-31#
INS      36-10    36-32#  126-6*   126-7*   126-8*
INSCHR   75-41    76-2#    76-95
INSEEK    6-40#   36-32
INSET    73-11    74-2#
INTEDC    7-9#    33-9
IRECLB    6-39#   36-31
ISUEXT  126-29   126-35   126-37#
ISUSEK  124-33   126-2#
JMPRET   47-37#   58-12    74-20    75-44    76-98    87-57    94-88   124-48   127-59   128-40   129-95   134-79   137-96   138-32
        139-23   140-17   141-122  142-34   148-33
LARGE     5-74#
LASTU    59-19#   61-24*   61-25*   61-46*
LBHINB    5-58#   57-36    87-42
LBLONB    5-59#   75-19    75-29    75-33    75-38    90-14   119-11   119-14   141-16
LBNCYL    5-23#   76-73    76-76
LBNHLD   45-79#
LBNHST    5-39#   77-36    77-39    79-29    79-39*   79-41*   79-42    79-42    80-23    80-27    80-27   143-19   143-21
LBNTRK    5-37#   43-9     44-13    44-37    76-35
LEVNZR  139-7    139-14#
LINK     39-27    40-2#
LINKLN    3-70#   40-14    40-25    60-9     60-23    60-25
LNCONT   44-24    44-28#
LNCYL    44-17    44-25#
LOADED   47-22    47-34#
LOBYTE    5-55#
LOCYL    36-33#
LONG     35-17    36-14#   55-33
LONGTO    5-6#    75-32
LOSEED   45-83#  122-8    122-16   122-23*
LOW       5-70#
LRDTRK   44-21*   44-25*   44-40*   44-57    45-7#
LSTTRK  112-9    112-24   116-2#
M.PARM   45-82#   46-19*   46-23*   46-31*   50-26*   50-53*   52-16*   52-25*   57-40*   87-23*   108-23*
MASK     97-19    97-30    98-2#    107-10   107-21   107-67   107-84
MASKLO   97-25    97-29#
MASKLP   98-9#    98-12
MAXDBN   59-22#   77-44    77-46    79-45    79-55    79-55    80-30    80-33    80-33    82-28*   82-29*
MAXDST   45-49    45-53    45-57    45-61    45-65    45-77#
MAXMUM   97-56    99-43    99-84   101-31   102-32   123-2#
MAXSEC   39-7*    39-28*   45-18#
MBXFER   55-10    55-61#
MEDTYP    5-35#
MEMPOL   39-15    45-93#   60-5     66-6*    66-7
MESSAG    6-16#   46-32    50-57
MICREV    5-12#
MKLOOP   73-8#    73-21
MOD      36-6     36-22#
MODOK    56-63    56-67    56-70#
MODOK0   56-28    56-34    56-40    56-43    56-52    56-61    56-71#
MORE     88-11#   88-23
MOV2    101-18   102-18   106-2#
MOVDBN   44-12    44-35#
MOVOUT   44-7     44-34    44-52#
MRD       4-17#   22-16
MS1      50-57    93-5#
```

CROSS REFERENCE TABLE (CREF V01-05)

```
MS2      46-32    93-7#
MWR       4-18#   22-21
NEWDNT  110-8     112-8     114-2#
NEWEXT  139-13    139-22#
NEWLEV  138-29    139-2#
NEWSUB    8-14#    50-13     50-47     74-7      94-46     94-48     94-56     95-12     99-9      99-11     99-56     99-58     101-14    102-14
        109-8     109-20    110-6     110-18    111-6     112-6     113-19    114-34    117-8
NEWUPT  109-10    111-8     113-2#
NEXTBE   99-86    102-2#
NHCFL   135-35    135-48#
NLBEXT   99-44    99-46#
NLEV    138-12    138-29#
NOBB     67-82    67-105#
NOBBS    65-15#
NOBES    64-13    64-21#
NOBORO   97-9     97-11#
NOCARY   97-38    97-40#
NOERR    47-40    47-45#
NOEXTR   67-86    67-93#
NOHCFL  132-45    132-69#
NORCLB   57-31    57-33#
NORTDS   49-13    49-15     49-20#
NOSEEK  124-32    124-44#
NOSUB    47-13    47-48#    49-38
NOSUN    62-12    62-18#
NOTOUT  132-47    132-53#
NOU      61-10    61-21     61-39#
NOUNIT   46-18    46-22     46-26#
NRTDS    49-27    49-31#
NSTATE   54-12    54-19#
NTOUT   135-37    135-43#
NUMBB    59-21#   65-12*    67-81     67-98
NUMOVL   45-68#   88-9
NUMPTR   35-14#   35-31#    35-36#    35-40#    35-44#    35-46#    35-52#    35-53     35-53     35-53#    35-53#    44-57     44-57     44-57
         44-57    44-57     44-57     44-57     44-57     44-57#    44-57#    44-57#    44-57#    44-57#    44-58     44-58     44-58     44-58
         44-58#   44-58#    46-32#    48-23#    49-29     49-29     49-29#    49-29#    49-31#    50-57#    54-19#    56-25     56-25     56-25
         56-25    56-25     56-25     56-25     56-25     56-25     56-25     56-25     56-25     56-25     56-25     56-25     56-25     56-25#
         56-25#   56-25#    56-25#    56-25#    56-25#    56-25#    56-25#    56-25#    56-26     56-33     56-33     56-33#    56-33#    56-39
         56-39    56-39#    56-39#    56-51     56-51     56-51#    56-51#    56-60     56-60     56-60#    56-60#    56-68     56-68     56-68#
         56-68#   57-15#    57-22     57-22     57-22#    57-22#    60-15#    75-11     75-11     75-11#    75-11#    76-93     76-93     76-93#
         76-93#   78-22#    79-15#    79-22#    79-37#    79-42     79-42     79-42     79-42     79-42#    79-42#    79-42#    79-53#    79-55
         79-55    79-55     79-55     79-55#    79-55#    79-55#    80-18#    80-27     80-27     80-27     80-27     80-27#    80-27#    80-27#
         80-33    80-33     80-33     80-33     80-33#    80-33#    80-33#    83-41     83-41     83-41     83-41     83-41#    83-41#    83-41#
         84-29#   87-53#    87-54     91-18#    126-32    126-32    126-32    126-32    126-32    126-32    126-32    126-32    126-32    126-32
        126-32    126-32    126-32#   126-32#   126-32#   126-32#   126-32#   126-32#   126-32#   126-34    126-36    126-36    126-36#   126-36#
        127-46    127-46    127-46    127-46    127-46    127-46    127-46#   127-46#   127-46#   127-46#   127-47    127-47    127-47    127-47
        127-47    127-47    127-47#   127-47#   127-47#   127-48    127-50    127-50    127-50    127-50    127-50    127-50    127-50#   127-50#
        127-50#   127-50#   127-51    127-51    127-51    127-51    127-51    127-51    127-51#   127-51#   127-51#   127-52    127-54    127-54
        127-54    127-54    127-54    127-54    127-54#   127-54#   127-54#   127-54#   127-55    127-55    127-55    127-55    127-55    127-55
        127-55#   127-55#   127-55#   127-56    129-40    129-40    129-40    129-40    129-40    129-40    129-40    129-40    129-40#   129-40#
        129-40#   129-40#   129-40#   132-12    132-12    132-12    132-12    132-12    132-12    132-12#   132-12#   132-12#   132-12#   132-13
        132-13    132-13    132-13    132-13    132-13    132-13#   132-13#   132-13#   132-14    132-14    132-14    132-14    132-14    132-14
        132-14    132-14    132-14#   132-14#   132-14#   132-14#   132-15    132-17    132-17    132-17    132-17    132-17    132-17    132-17#
        132-17#   132-17#   132-17#   132-18    132-18    132-18    132-18    132-18    132-18    132-18#   132-18#   132-18#   132-19    132-19
        132-19    132-19    132-19    132-19    132-19#   132-19#   132-19#   132-20    132-48    132-48    132-48    132-48    132-48    132-48
        132-48    132-48    132-48#   132-48#   132-48#   132-48#   132-48#   132-49    132-49    132-49    132-49    132-49    132-49    132-49
        132-49    132-49    132-49    132-49#   132-49#   132-49#   132-49#   132-49#   132-50    132-50    132-50    132-50    132-50    132-50
```

CROSS REFERENCE TABLE (CREF V01-05)

```
        132-50#  132-50#  132-50#  132-51   132-61#  132-62   132-64   132-64   132-64   132-64   132-64   132-64   132-64   132-64
        132-64#  132-64#  132-64#  132-64#  132-64#  132-65   132-65   132-65   132-65   132-65   132-65   132-65   132-65   132-65
        132-65   132-65#  132-65#  132-65#  132-65#  132-65#  132-66   132-66   132-66   132-66   132-66   132-66   132-66#  132-66#
        132-66#  132-67   134-40   134-40   134-40   134-40   134-40   134-40   134-40   134-40   134-40#  134-40#  134-40#  134-40#
        134-40#  135-12   135-12   135-12   135-12   135-12   135-12   135-12#  135-12#  135-12#  135-12#  135-13   135-13   135-13
        135-13   135-13   135-13   135-13#  135-13#  135-13#  135-14   135-14   135-14   135-14   135-14   135-14   135-14   135-14
        135-14#  135-14#  135-14#  135-14#  135-15   135-17   135-17   135-17   135-17   135-17   135-17   135-17#  135-17#  135-17#
        135-17#  135-18   135-18   135-18   135-18   135-18   135-18   135-18#  135-18#  135-18#  135-19   135-19   135-19   135-19
        135-19   135-19   135-19#  135-19#  135-19#  135-20   135-38   135-38   135-38   135-38   135-38   135-38   135-38   135-38
        135-38#  135-38#  135-38#  135-38#  135-38#  135-39   135-39   135-39   135-39   135-39   135-39   135-39   135-39   135-39
        135-39   135-39#  135-39#  135-39#  135-39#  135-39#  135-40   135-40   135-40   135-40   135-40   135-40   135-40#  135-40#
        135-40#  135-41   135-43   135-43   135-43   135-43   135-43   135-43   135-43   135-43   135-43#  135-43#  135-43#  135-43#
        135-43#  135-44   135-44   135-44   135-44   135-44   135-44   135-44   135-44   135-44   135-44   135-44#  135-44#  135-44#
        135-44#  135-44#  135-45   135-45   135-45   135-45   135-45   135-45   135-45#  135-45#  135-45#  135-46   137-26   137-26
        137-26   137-26   137-26   137-26   137-26   137-26   137-26#  137-26#  137-26#  137-26#  137-26#  137-27   137-27   137-27
        137-27   137-27   137-27   137-27   137-27   137-27   137-27   137-27#  137-27#  137-27#  137-27#  137-27#  137-41   137-41
        137-41   137-41   137-41   137-41   137-41   137-41   137-41#  137-41#  137-41#  137-41#  137-41#  137-42   137-42   137-42
        137-42   137-42   137-42   137-42   137-42   137-42   137-42   137-42#  137-42#  137-42#  137-42#  137-42#  137-54   137-54
        137-54   137-54   137-54   137-54   137-54   137-54   137-54#  137-54#  137-54#  137-54#  137-54#  137-55   137-55   137-55
        137-55   137-55   137-55   137-55   137-55   137-55   137-55   137-55#  137-55#  137-55#  137-55#  137-55#  137-72   137-72
        137-72   137-72   137-72   137-72   137-72   137-72   137-72#  137-72#  137-72#  137-72#  137-72#  137-73   137-73   137-73
        137-73   137-73   137-73   137-73   137-73   137-73   137-73   137-73#  137-73#  137-73#  137-73#  137-73#  137-74   137-74
        137-74   137-74   137-74#  137-74#  137-84   137-84   137-84   137-84   137-84   137-84   137-84   137-84   137-84#  137-84#
        137-84#  137-84#  137-84#  137-85   137-85   137-85   137-85   137-85   137-85   137-85   137-85   137-85   137-85   137-85#
        137-85#  137-85#  137-85#  137-85#  137-86   137-86   137-86   137-86   137-86#  137-86#  139-8#   140-10   140-10   140-10#
        140-10#  141-65   141-65   141-65   141-65   141-65   141-65   141-65   141-65   141-65#  141-65#  141-65#  141-65#  141-65#
        141-66   141-66   141-66   141-66   141-66   141-66   141-66   141-66   141-66#  141-66#  141-66#  141-66#  141-75   141-75
        141-75   141-75   141-75   141-75   141-75   141-75   141-75   141-75   141-75   141-75   141-75   141-75   141-75   141-75
        141-75   141-75   141-75   141-75   141-75   141-75   141-75#  141-75#  141-75#  141-75#  141-75#  141-75#  141-75#  141-75#
        141-75#  141-75#  141-75#  141-76   141-76   141-76   141-76   141-76   141-76   141-76   141-76   141-76   141-76   141-76
        141-76   141-76#  141-76#  141-76#  141-76#  141-76#  141-76#  144-28   144-28   144-28   144-28   144-28   144-28   144-28
        144-28   144-28#  144-28#  144-28#  144-28#  144-28#  144-32   144-32   144-32   144-32   144-32   144-32   144-32   144-32
        144-32#  144-32#  144-32#  144-32#  144-32#  144-34   144-34   144-34   144-34   144-34   144-34   144-34   144-34   144-34#
        144-34#  144-34#  144-34#  144-34#  145-48   145-48   145-48   145-48   145-48#  145-48#  145-48#  145-52   145-52   145-52
        145-52   145-52#  145-52#  145-52#  145-54   145-54   145-54   145-54   145-54#  145-54#  145-54#  148-16   148-16   148-16#
        148-16#
NXTRCT  142-33   147-2#
NXTSEC   94-52    95-2#
OK2DRP   57-21    57-24#
ONEPAT  131-13   131-21#
ONEPAX  141-37   141-47#
ONEREG   97-17    97-28#
ONL      36-11    36-38#
ONLERR   87-46    87-53#
ONLOK    56-32    56-35#
ONLYCL    8-32#   67-49    76-44    79-35    79-51
OTABLE   45-23#   45-68    47-19    47-25    48-8     48-11    48-15    88-8*    88-10
OUT.01   22-22    22-23    23-6#    35-14*   35-31*   35-36*   35-40*   35-44*   35-46*   35-52*   44-57*   46-32*   48-23*   49-29*
         49-31*   50-57*   54-19*   55-62*   56-25*   57-15*   60-15*   64-6*    65-7*    67-7*    67-76*   67-88*   78-22*   79-15*
         79-22*   79-37*   79-42*   79-53*   79-55*   80-18*   80-27*   80-33*   83-41*   84-29*   87-53*   91-18*   126-32*  127-33*
        127-46*  127-50*  127-54*  129-40*  129-83*  132-12*  132-17*  132-48*  132-61*  132-64*  134-40*  134-72*  135-12*  135-17*
        135-38*  135-43*  137-26*  137-41*  137-54*  137-69*  137-72*  137-84*  139-8*   141-65*  141-86*  144-28*  144-32*  144-34*
        145-48*  145-52*  145-54*
OUT.02   23-7#    35-14*   35-14*   35-31*   35-31*   35-36*   35-36*   35-40*   35-40*   35-44*   35-44*   35-46*   35-46*   35-52*
         35-52*   44-57*   44-57*   46-32*   48-23*   48-23*   49-29*   49-29*   49-31*   49-31*   50-57*   54-19*   54-19*   56-25*
         56-25*   57-15*   57-15*   60-15*   60-15*   78-22*   78-22*   79-15*   79-15*   79-22*   79-22*   79-37*   79-37*   79-42*
         79-42*   79-53*   79-53*   79-55*   79-55*   80-18*   80-18*   80-27*   80-27*   80-33*   80-33*   83-41*   83-41*   84-29*
```

CROSS REFERENCE TABLE (CREF V01-05)

	84-29*	87-53*	87-53*	91-18*	91-18*	126-32*	126-32*	127-46*	127-46*	127-50*	127-50*	127-54*	127-54*	129-40*
	129-40*	129-83*	129-90*	132-12*	132-12*	132-17*	132-17*	132-48*	132-48*	132-61*	132-61*	132-64*	132-64*	134-40*
	134-40*	134-72*	135-12*	135-12*	135-17*	135-17*	135-38*	135-38*	135-43*	135-43*	137-26*	137-26*	137-31*	137-41*
	137-41*	137-46*	137-54*	137-54*	137-59*	137-69*	137-72*	137-72*	137-84*	137-84*	139-8*	139-8*	141-65*	141-65*
	141-86*	141-115*	144-28*	144-28*	144-32*	144-32*	144-34*	144-34*	145-48*	145-48*	145-52*	145-52*	145-54*	145-54*
OUT.03	23-8#	48-24*	55-11*	60-16*	91-20*	129-83*	129-91*	134-72*	137-69*	141-86*	141-116*			
OUT.04	23-9#	35-14*	35-31*	35-36*	35-40*	35-44*	35-46*	35-52*	44-57*	48-23*	49-29*	49-31*	54-19*	56-25*
	57-15*	60-15*	78-22*	79-15*	79-22*	79-37*	79-42*	79-53*	79-55*	80-18*	80-27*	80-33*	83-41*	34-29*
	87-53*	91-18*	126-32*	127-46*	127-50*	127-54*	129-40*	132-12*	132-17*	132-48*	132-61*	132-64*	134-40*	135-12*
	135-17*	135-38*	135-43*	137-26*	137-41*	137-54*	137-72*	137-84*	139-8*	141-65*	144-28*	144-32*	144-34*	145-48*
	145-52*	145-54*												
OUT.05	23-10#	44-57*	49-29*	56-25*	56-33*	56-39*	56-51*	56-60*	56-68*	57-22*	75-11*	76-93*	79-42*	79-55*
	80-27*	80-33*	83-41*	126-32*	126-36*	127-46*	127-50*	127-54*	129-40*	132-12*	132-17*	132-48*	132-64*	134-40*
	135-12*	135-17*	135-38*	135-43*	137-26*	137-41*	137-54*	137-72*	137-84*	140-10*	141-65*	144-28*	144-32*	144-34*
	145-48*	145-52*	145-54*	148-16*										
OUT.06	23-11#	35-53*	44-57*	56-25*	79-42*	79-55*	80-27*	80-33*	83-41*	126-32*	127-46*	127-50*	127-54*	129-40*
	132-12*	132-17*	132-48*	132-64*	134-40*	135-12*	135-17*	135-38*	135-43*	137-26*	137-41*	137-54*	137-72*	137-84*
	141-65*	144-28*	144-32*	144-34*	145-48*	145-52*	145-54*							
OUT.07	23-12#	44-57*	56-25*	126-32*	127-46*	127-50*	127-54*	129-40*	132-12*	132-17*	132-48*	132-64*	134-40*	135-12*
	135-17*	135-38*	135-43*	137-26*	137-41*	137-54*	137-72*	137-84*	141-65*	144-28*	144-32*	144-34*		
OUT.08	23-13#	44-57*	56-25*	126-32*	127-47*	127-51*	127-55*	129-40*	132-13*	132-18*	132-48*	132-64*	134-40*	135-13*
	135-18*	135-38*	135-43*	137-26*	137-41*	137-54*	137-72*	137-84*	141-65*	144-28*	144-32*	144-34*		
OUT.09	23-14#	44-58*	56-25*	126-32*	127-47*	127-51*	127-55*	132-13*	132-18*	132-49*	132-65*	135-13*	135-18*	135-39*
	135-44*	137-27*	137-42*	137-55*	137-73*	137-85*	141-66*							
OUT.10	23-15#	44-58*	56-25*	126-32*	127-47*	127-51*	127-55*	132-13*	132-18*	132-49*	132-65*	135-13*	135-18*	135-39*
	135-44*	137-27*	137-42*	137-55*	137-73*	137-85*	141-66*							
OUT.11	23-16#	56-25*	132-14*	132-19*	132-49*	132-65*	135-14*	135-19*	135-39*	135-44*	137-27*	137-42*	137-55*	137-73*
	137-85*	141-66*												
OUT.12	23-17#	56-25*	132-14*	132-19*	132-49*	132-65*	135-14*	135-19*	135-39*	135-44*	137-27*	137-42*	137-55*	137-73*
	137-85*	141-66*												
OUT.13	23-18#	132-14*	132-19*	132-49*	132-65*	135-14*	135-19*	135-39*	135-44*	137-27*	137-42*	137-55*	137-73*	137-85*
	141-75*													
OUT.14	23-19#	132-14*	132-50*	132-66*	135-14*	135-40*	135-45*	137-74*	137-86*	141-75*				
OUT.15	23-20#	132-50*	132-66*	135-40*	135-45*	137-74*	137-86*	141-75*						
OUT.16	23-21#	132-50*	132-66*	135-40*	135-45*	141-75*								
OUT.17	23-22#	141-75*												
OUT.18	23-23#	141-75*												
OUT.19	23-24#	141-75*												
OUT.20	23-25#	141-75*												
OUT.21	23-26#	141-75*												
OUT.22	23-27#	141-75*												
OUT.23	23-28#	141-75*												
OUT.24	23-29#	141-76*												
OUT.25	23-30#	141-76*												
OUT.26	23-31#	141-76*												
OUT.27	23-32#	141-76*												
OUT.28	23-33#	141-76*												
OUT.29	22-23	23-34#	55-9*	129-93*	141-76*	141-118*								
OUT.RQ	22-13*	22-14	23-5#	35-14*	35-31*	35-36*	35-40*	35-44*	35-46*	35-52*	44-57*	48-23*	49-29*	49-31*
	54-19*	55-15	55-59	56-25*	57-15*	60-15*	78-22*	79-15*	79-22*	79-37*	79-42*	79-53*	79-55*	80-18*
	80-27*	80-33*	83-41*	84-29*	87-53*	91-18*	126-32*	127-46*	127-50*	127-54*	129-40*	132-12*	132-17*	132-48*
	132-61*	132-64*	134-40*	135-12*	135-17*	135-38*	135-43*	137-26*	137-32*	137-41*	137-47*	137-54*	137-60*	137-72*
	137-84*	139-8*	141-65*	144-28*	144-32*	144-34*	145-48*	145-52*	145-54*					
OUTO	100-14	100-16#												
OV...	2-39	2-39	2-39#	92-9	92-9	92-9	92-9#	94-1	94-1	94-1	94-1#	124-2	124-2	124-2
	124-2#	127-2	127-2	127-2	127-2#	128-2	128-2	128-2	128-2#	129-2	129-2	129-2	129-2#	134-2
	134-2	134-2	134-2#	137-2	137-2	137-2	137-2#	142-2	142-2	142-2	142-2#	148-2	148-2	148-2

CROSS REFERENCE TABLE (CREF V01-05)

```
        148-2#   148-35   148-35#
OVE.CK  137-2#   142-2
OVE.MN    2-39#   92-9
OVE.MS   92-9#    94-1
OVE.R   134-2#   137-2
OVE.RB  148-2#   148-35
OVE.RV  142-2#   148-2
OVE.RW  128-2#   129-2
OVE.SK  124-2#   127-2
OVE.ST   94-1#   124-2
OVE.TS  127-2#   128-2
OVE.W   129-2#   134-2
OVERFL    5-75#
OVL...    2-39     2-39#   92-9     92-9#    94-1     94-1#   124-2    124-2#   127-2    127-2#   128-2    128-2#   129-2    129-2#
        134-2    134-2#   137-2    137-2#   142-2    142-2#   148-2    148-2#   148-35   148-35#
OVL.CK   45-59    142-2    142-2    142-2#
OVL.MN    2-39      2-39    92-9     92-9     92-9#
OVL.MS   45-26     94-1     94-1     94-1#
OVL.R    45-55    137-2    137-2    137-2#
OVL.RB   45-67    148-35   148-35   148-35#
OVL.RV   45-63    148-2    148-2    148-2#
OVL.RW   45-47    129-2    129-2    129-2#
OVL.SK   45-39    127-2    127-2    127-2#
OVL.ST   45-35    124-2    124-2    124-2#
OVL.TS   45-43    128-2    128-2    128-2#
OVL.W    45-51    134-2    134-2    134-2#
OVRLAY   47-23     48-2#
OVS.CK  137-2#
OVS.MN    2-39#
OVS.MS   92-9#
OVS.R   134-2#
OVS.RB  148-2#
OVS.RV  142-2#
OVS.RW  128-2#
OVS.SK  124-2#
OVS.ST   94-1#
OVS.TS  127-2#
OVS.W   129-2#
OVSTRT    3-10#    88-6     88-7
PAT0     45-95     45-114#  88-29
PAT1     45-96     45-131#
PAT10    45-105    45-209#
PAT11    45-106    45-226#
PAT12    45-107    45-228#
PAT13    45-108    45-245#
PAT14    45-109    45-262#
PAT15    45-110    45-266#
PAT2     45-97     45-133#
PAT3     45-98     45-135#
PAT4     45-99     45-137#
PAT5     45-100    45-154#
PAT6     45-101    45-171#
PAT7     45-102    45-188#
PAT8     45-103    45-190#
PAT9     45-104    45-207#
PATEXT  119-8     119-17#
PATPTR   45-95#   131-8     141-32
```

CROSS REFERENCE TABLE (CREF V01-05)

```
PATTRN  94-62   119-2#
P  AD   47-29    47-33#
PNUM    45-78#  141-17*  141-31   141-75
PREVBE  99-45   101-2#
PTABLE  45-69#   45-77    47-21    47-24*   47-31*   47-34    48-10
RANDOM  96-18    97-20    97-26    97-31   107-11   107-17   107-22   107-68   107-86   118-12   119-13   120-16   121-16   122-2#
       123-13
RBLOCK 134-56   135-2#
RBNBN    8-18#   39-8     41-31    44-16    94-32   117-8    125-8    129-36   129-43   129-79   134-36   134-43   141-107  145-25
RBNFLG  44-8*    44-18*   44-57    45-8#
RBNTRK   5-31#   43-7     44-19    44-35    76-33
RBUFLN   3-69#   41-18    60-9
RCBREQ  56-62    56-74#   57-33*
RCLB     8-23#  128-18   128-38   148-21
RCMPER 141-71   141-75#
RCONT    3-41#
RCTCPS   5-8#    75-36
RCTCSZ   5-40#  144-40   147-12
RCV      4-8#    35-24    55-40    56-22    89-30
RCVERR   6-23#   21-17
RCVRDY   6-21#   49-14    56-10    89-15
RDMOD  128-29   128-31   128-34#
RDSTAT  21-2#    34-3     89-12
RDWRT   58-8    128-10#  140-14
READ   128-34   134-10#  134-67   138-26
RECALB 129-27   134-27   148-11#  148-18
RECOVE  54-18    54-26    54-36#
RECOVR  50-39    54-36    56-2#    87-44
REDEXT 134-49   134-75#
REDSET  58-2#   124-44   127-38
REDWRT   8-19#   40-9     41-7     94-40    94-76   117-8    118-15   119-7    120-8    121-10   128-28
REPERR  49-36    54-21    55-2#    57-17
REPORT  54-9     54-21#   54-39
RESEEK   8-12#   50-13   125-21   127-25   139-18   148-21
RETRY    8-22#   94-32    94-4)   117-8    128-20   128-38   129-74
RETS     5-7#    75-14
REVCT  129-64   134-78   142-11#
REVEC    8-17#  117-8    129-62   134-36   134-61   134-76   146-7
REVECT   5-63#
REVINP   8-20#  117-8    128-30   142-14   143-7    146-7
REVS     5-16#
REVSOK 142-25   144-2#
REVSUP 142-16   143-2#
REXT   135-27   135-31   135-33   135-50#
RM       5-32#
RNDO   119-10   119-13#  119-15
RNDBE   95-16    96-2#
RNDLP0  97-20#   97-24
RNDLP1  97-31#   97-34
RNDSUB  50-66    53-1#
RNDTG   95-18   107-2#
RONLY    8-29#   87-33   118-10
ROOT    46-14#   60-29
ROOTLP  46-16#   46-38
RORW    94-61   118-2#
ROTEXT  46-25    46-28    46-34    46-38#
ROUT   135-16   135-21   135-42   135-47   135-49   135-51#
```

CROSS REFERENCE TABLE (CREF V01-05)

```
RREAL     3-44#   41-14
RSTOP     3-40#   41-16
RTDS     34-2#    49-11    54-10    55-16    56-7     127-17   132-6    135-6
RTDSE   127-19    127-50#
RTDSEX   34-6     34-8#
RTRIES    8-31#   129-23   134-23   137-29   137-44   137-57   138-8
RUN      36-16    36-30#
RVFAIL  144-35    145-27   145-55   146-2#
RW.ANG    3-34#   41-27*
RW.BUF    3-29#   41-23*   137-65   137-80   141-14   145-7
RW.CMD    3-32#   41-10*    41-15*  132-49   132-65   135-39   135-44   137-27   137-42   137-55   137-73   137-85   141-66
RW.HI     3-31#   41-40*   132-48   132-64   133-12*  135-38   135-43   136-11*  137-26   137-41   137-54   137-72   137-84   141-65
RW.LOW    3-30#   37-10     41-29*  132-48   132-64   133-11*  135-38   135-43   136-10*  137-26   137-41   137-54   137-72   137-84
        141-65
RW.SDI    3-33#   41-25*
RW.STA    3-28#   41-22*   137-16
RWEXT   128-33    128-36   128-38#
RWRDY     6-26#   127-22   132-9    135-9
S.BADP    7-62#    7-63     37-8     67-83*   67-105*  70-6     80-8
S.BESS    7-63#    7-68     64-14    67-17    67-43    68-6     68-8     77-8     77-9     77-14*   77-16*   77-17    77-20    77-30*
         77-31*   77-38*   77-42*   77-45*   77-47*   79-9     79-11    84-55    84-57    84-58    84-62    84-68    96-9     96-30
         99-13    99-60    100-7
S.MCNT    7-68#    7-69    107-8    107-14   107-20   107-24   113-7    113-9    114-10   114-12
S.MEGR    7-60#    7-61     76-19*  141-109  141-119*
S.MEGW    7-61#    7-62     76-18*  129-85   129-89*  132-33   132-34*
S.PARM    7-55#    7-56     41-33    44-10    47-41    47-43*   50-24    50-46    50-56*   51-13    52-14    53-17    57-37    67-16*
         67-50*   67-61*   76-29    76-41    76-55    77-33    79-26    79-32    79-48    80-20    83-27    84-37    85-8     87-14
         87-32    94-43    95-8     107-53   108-7    108-22   118-6    120-7    121-6    123-10   129-22   134-22   137-23   138-7
S.PAT     7-57#    7-58    67-12*   119-9
S.SCHR    7-59#    7-60     43-6     44-9     44-53    67-107*  76-15    76-64*   76-72    107-56   108-28   115-6    132-25   143-11
        144-39   147-11
S.SDCL    7-56#    7-57     44-41    44-43    76-75*   76-79*
S.TGOF    7-69#    7-70     83-21    83-23*   84-6     84-57    84-58    107-30   113-11   113-13   114-8    114-9
S.TGSS    7-70#   64-11     67-37    83-13    84-62    84-68    107-40   109-27   110-22   113-15   114-14   114-21
S.TRKL    7-58#    7-59     76-33*   107-52   108-21   111-13   112-16   116-10
SBCRES    6-44#   36-8
SCR      36-8     36-27#
SCTWRD    7-8#    131-9    141-42   141-51
SDIVER    5-4#
SEARCH  142-28    145-2#
SECCYL   59-26#   78-17    81-21*   82-13
SECGRP   59-25#   81-14*   81-18    82-20    85-12
SECMAX   45-16#   60-13*   94-78    94-80
SECPTR   40-13*   41-12    45-19#   130-18
SECTOR   41-26    45-12#
SECTRK   59-24#   76-37*   81-11    85-14
SEEK     94-86   124-10#  124-46   127-57   143-26   144-47   147-24   148-32
SEKCNT  127-29    127-37#
SEKEXT  124-43    124-48#
SEKINP    8-15#   54-16    87-48    124-37   127-25
SEKOUT  124-45    124-47#
SEKTST  124-42    127-10#  127-43
SEND      4-7#    35-11    55-30    89-23
SEQBE    95-22    99-2#
SEQLP    47-11#   47-46
SEQNCR   46-23    47-2#    73-13
SEQSEK    8-34#   50-64    94-44    95-9     123-11
```

CROSS REFERENCE TABLE (CREF V01-05)

```
SEQTG    95-25   108-2#
SER0     56-39    92-306#
SER1     56-51    92-308#
SER2     56-33    92-310#
SER3     56-60    92-312#
SER4     92-314# 140-10
SER5     76-93    92-316#
SER6     57-22    92-318#
SER7     56-68    92-320# 148-16
SER8     75-11    92-322#
SER9     92-324# 126-36
SETRLT   75-10    75-14#
SETSEC  111-9    111-25   115-2#   116-6
SETUP    47-44    49-16    87-50    94-10#   94-58   129-47   129-76   129-84   134-47   138-20   141-120  146-18
SHRTTO    5-3#    75-28
SLOP     62-11#   62-21
SMASK    76-21    86-2#
SMASKL   86-11#   86-14
SMASKX   86-11    86-15#
SMLSUB  141-68   141-72#
SNDAGN   22-14#   22-18
SNDEX   140-13   140-17#
SNDEXT   35-16    35-48    35-51    35-55#
SNDLEV  139-20   140-2#   140-11
SORTBB   67-103   70-2#
SORTBE   67-74    68-2#
SORTTG   67-39    72-2#
SS        5-9#
ST       36-5     36-6     36-7     36-8     36-9     36-10    36-11    36-12    36-13    36-15    36-16    36-40#   55-53    56-14
         56-20    56-25    56-25    56-25    56-25    56-25    56-25    56-25    56-44    57-11    57-34    75-14    75-21    75-24
         75-28    75-32    75-36    76-33    76-35    76-61    77-36    77-39    79-29    79-39*   79-41*   79-42    79-42    80-23
         80-27    80-27    81-9     81-16    82-10    82-18    83-30    83-32    84-40    84-42    90-5
ST.C      4-30#
ST.CON    4-29#
ST.DB     4-47#
ST.DF     4-44#
ST.DR     4-37#
ST.ERR    4-28#   56-44
ST.FE     4-41#
ST.FO     4-46#
ST.MOD    4-27#   57-34
ST.MSK    4-25#
ST.OA     4-35#
ST.PE     4-43#
ST.PS     4-39#   57-12
ST.RE     4-42#
ST.RR     4-36#   57-30
ST.RTY    4-31#
ST.RU     4-40#   57-12
ST.S7     4-48#
ST.SR     4-38#   57-12
ST.STA    4-26#   57-11
ST.UNT    4-24#
ST.WE     4-45#
STACK    20-8     30-5#
STACYL   44-30*   44-33*   44-42*   44-44*   44-58    44-58    45-5#
START    20-9     60-1#
```

CROSS REFERENCE TABLE (CREF V01-05)

```
STATEX  21-16    21-18    21-22#
STATLP  21-9#    21-14    21-20
STATOK  21-12    21-17#
STATUS   4-10#   21-9
STSERR 127-23   127-41#
STSERX 127-49   127-53   127-57#
STSEXT 127-40   127-45   127-59#
STSRES   6-45#   36-9
SUB      5-22#
SUBUNT  36-28#   76-22*
SUSETL  61-48#   61-54
SWAPBB  70-14    71-2#
SWAPBE  68-14    69-2#
T1MSIZ   6-3#
T2CMD    6-5#
T2DLL    6-4#
T4BB1    6-8#    65-8     67-77
T4BB2    6-9#    67-89
T4MPRM   6-6#    88-24
T4MXFR   6-12#   55-9     55-61   129-92   141-117
T4SEEK   6-11#  127-34
T4SOFT   6-10#  129-83   134-72   137-69   141-86
T4UPRM   6-7#    64-7     67-8
TALK    35-2#    56-30    56-36    56-48    56-57    56-65    57-19    75-8     76-25    126-10   140-7
TALKOK 140-9    140-14#
TEST4    2-37#   24-1
TGS     59-27#   83-12    83-34    84-5     84-43
THREBE  96-22    96-25#
TIMEOU   5-61#
TLEN.U   7-10#    7-11    61-22
TMEMPL  39-15*   42-6*    42-7     45-94#
TMPERR 126-31   126-36#
TO      75-30    75-34    75-46#
TOOBIG   5-69#
TRACK   41-6     45-11#  126-32
TRACKS   8-36#   83-28    84-38    85-10    107-54   108-10   108-16   108-25
TRKGRP   5-28#   81-9     83-32    84-42    107-57   108-29   115-7
TROOT   60-4     73-2#
TRYAGN  96-18#   96-26
TSTNEC 124-19   125-2#
TSTRCB  56-59    56-62#
U.CBN    7-41#    7-42    39-13    39-14    94-23    94-25*   94-27    94-29*   94-66*   94-68*   125-13   125-14   129-40   129-40
       132-12   132-12   132-17   132-17   133-7    133-8    134-40   134-40   135-12   135-12   135-17   135-17   136-7    136-8
       143-20*  143-24*  144-28   144-28   144-32   144-32   144-34   144-34   144-41   144-42*  144-44   144-46*  146-13*  146-17*
       147-7    147-8    147-21*  147-22*
U.CCNT   7-22#    7-23    109-12   109-14*  109-16   109-23*  110-10   110-12*  110-14   110-21*  113-8*   113-10*  114-11*  114-13*
U.CCOP   7-44#    7-45    143-15*  144-21   144-38*  147-9
U.CCYL   7-48#    7-49    39-21    39-23    74-16*   74-17*   125-23   126-13   127-47   127-47   127-51   127-51   127-55   127-55
       132-13   132-13   132-18   132-18   132-49   132-49   132-65   132-65   135-13   135-13   135-18   135-18   135-39   135-39
       135-44   135-44   137-27   137-27   137-42   137-42   137-55   137-55   137-73   137-73   137-85   137-85   141-66   141-66
       148-28*  148-29*
U.CGRP   7-49#    7-50    39-25    74-18*   127-47   127-51   127-55   132-13   132-18   132-49   132-65   135-13   135-18   135-39
       135-44   137-27   137-42   137-55   137-73   137-85   141-66   148-30*
U.COPY   7-43#    7-44    75-40*   144-23
U.CSEC   7-28#    7-29    50-19*   50-50*   94-18    94-22*   94-70*   94-74    142-18   146-11
U.CTRK   7-24#    7-25    111-14   111-17*  112-11   112-14*  115-9*
U.ECCT   7-34#    7-35    75-27*   137-51   144-17
```

CROSS REFERENCE TABLE (CREF V01-05)

```
U.ELEV    7-31#   7-32    58-7*   137-34*  137-49*  137-62*  137-77*  137-89*  139-6    139-11*  139-16*  141-79*  141-80
U.LCYL    7-50#   7-51    126-17  127-46   127-46   127-50   127-50   127-54   127-54   132-14   132-14   132-19   132-19   132-50
          132-50  132-66  132-66  135-14   135-14   135-19   135-19   135-40   135-40   135-45   135-45
U.LGRP    7-10    7-51#   127-46  127-50   127-54   132-14   132-19   132-50   132-66   135-14   135-19   135-40   135-45
U.MASK    7-29#   7-30    34-2    35-8     54-27    55-27    56-13    61-26*   132-24   135-23
U.MBN     7-40#   7-41    94-65   94-67    97-57    99-19    99-48    99-50*   99-52    99-54*   99-62    100-9    103-8    107-75*
          107-78* 109-31  109-32* 109-34   109-36*  110-30   110-32*  110-35   110-37*  111-18   111-19*  111-21   111-23*  111-31
          111-33* 111-35  111-37* 112-15   112-17*  112-19   112-21*  113-12*  113-14*  114-31*  114-32*  116-12   116-13*  116-15
          116-17* 142-19  142-21  146-12   146-14
U.MLEV    7-33#   7-34    58-6    75-23*   141-82
U.MSEC    7-26#   7-27    39-7    94-35*   94-85*   129-71*  134-66*  138-23*  143-17*
U.NEXT    7-15#   7-16    46-16   61-51    73-19
U.NFUN    7-20#   7-21    47-10   47-47*   73-12*
U.NSEC    7-25#   7-26    50-20*  50-51*   58-10*   94-19    94-24    129-46*  129-68   132-32*  132-40   132-42*  132-69   134-46*
          138-15  141-87  141-89* 145-29*  146-10*
U.PARM    7-38#   7-39    39-6    40-6     44-15    46-17    49-33    49-35*   50-10    50-15*   50-84    50-86*   51-19    51-21*
          52-24   52-31*  53-21   53-23*   54-15    57-24    57-26*   73-10*   73-16    73-18*   74-6     74-8*    76-89    76-91*
          87-7    87-20   87-22*  87-47    87-49*   94-31    94-37    94-41*   94-49*   94-55    94-75    95-11    95-13*   99-6
          99-12*  99-59*  101-13  101-15*  102-13   102-15*  108-6    109-19   109-21*  110-17   110-19*  113-18   113-20*  114-33
          114-35* 117-6   117-9*  118-7    118-16*  119-6    120-6    120-20*  121-9    121-19*  124-36   124-38*  125-7    125-20
          127-24  127-26* 128-17  128-39*  129-35   129-42   129-44*  129-58   129-63*  129-75*  129-78   129-80*  134-35   134-42
          134-44* 134-60  134-62* 134-75   138-17   138-19*  139-17   139-19*  141-7    141-106  141-108* 142-13   143-8*   145-24
          145-26* 146-6   146-8*  148-20   148-22*
U.PAT     7-21#   7-22    119-16* 130-7    131-7
U.PCTG    7-23#   7-24    109-24  109-30*  110-24   110-33*  113-17*  114-29*
U.RBN     7-42#   7-43    39-10   39-11    94-71*   94-72*   125-10   125-11   129-40   129-40   132-48   132-49   132-64   132-65
          134-40  134-40  135-38  135-39   135-43   135-44   137-26   137-27   137-41   137-42   137-54   137-55   137-72   137-73
          137-84  137-85  141-65  141-65   143-13*  143-14*  145-18   145-19*  145-21   145-23*  145-38   145-39*  145-41   145-43*
U.RTRY    7-32#   7-33    75-20*  138-11
U.RVER    7-46#   7-47    143-10* 144-25   145-45
U.RWER    7-45#   7-46    129-38  129-51*  132-37*  134-38   134-51*  135-28*  143-9
U.RWTO    7-18#   7-19    128-25* 129-16   129-20*  129-81   132-12   132-17   132-48   132-64   134-16   134-20*  134-70   135-12
          135-17  135-38  135-43  138-25*  148-23   148-26*
U.SDIL    7-36#   7-37    35-21   55-37    74-13*   75-35*
U.SDIS    7-35#   7-36    35-19   55-35    74-12*   75-31*
U.SEEK    7-19#   7-20    74-10*  127-27   127-37*
U.SNUM    7-47#   7-48    46-32   50-57    55-6     61-29    62-8     127-30   129-83   134-72   137-69   141-86
U.SUBP    7-16#   7-17    50-16   50-37    50-42    51-5     52-5     53-6     62-6     76-8     87-10    87-25
U.SUBU    7-39#   7-40    46-32   50-34*   50-44    50-57    50-58    50-68    51-15*   52-20*   53-8     53-25*   55-7     127-32
          129-83  132-54  134-72  137-69   141-86
U.TIMO    7-17#   7-18    49-22   49-24*   54-30*   124-40*  127-41   127-44*  128-37*  137-26   137-41   137-54   137-72   137-84
          138-10  138-14* 141-84
U.TSEC    7-27#   7-28    50-21*  50-52*   94-20    94-73    99-46    103-10   107-91*  108-35*  123-19*
U.WPRT    7-37#   7-38    57-42   87-43*
U.WRIT    7-30#   7-31    56-53   57-43*   132-53
UCURSR    59-18#  62-13*  64-6    65-7     67-7     67-76    67-88
ULOP      61-9#   61-44
UMASK     59-20#  61-7*   61-39   61-41*
UNIT0      5-46#
UNIT1      5-47#
UNIT2      5-48#
UNIT3      5-49#
UNITS     59-15#  61-8    61-43   89-7     91-7     91-16
UNSSUC     6-42#
UP         99-8    99-56#
UPEXT     103-14  103-18#
UPG       108-20  111-2#
```

CROSS REFERENCE TABLE (CREF V01-05)

```
UPLBN    99-70   103-2#
UPT     108-18   109-2#   111-39
UREAD     4-14#    48-16
USED     59-23#    76-11*   76-14*   76-87*
UTOTST    6-13#    91-4
UWRITE    4-15#
WAITSI    4-13#  132-28   135-24
WBLOCK  129-57   132-2#
WBUFLN    3-68#     3-69     40-11    60-23
WCHECK    8-37#    50-14     94-38    94-40    117-8    120-10   120-19   121-12
WCHK     94-63   120-2#
WCHKAL    8-39#  120-14
WCONT     3-39#
WEXT    132-36   132-43   132-70   132-73#
WONLY     8-30#  118-8
WOUT    132-16   132-21   132-52   132-63   132-68   132-72   132-74#
WREAL     3-43#    41-9
WRITBT   36-23#    56-55*
WRITE   128-32   129-10#  129-72
WRONG     5-65#
WRTEXT  129-30   129-49   129-67   129-95#
WSTOP     3-38#    41-11
WTIMOT  132-60   132-64#
XBNCYL    5-41#    76-74
XFERRT    5-5#
XMTERR    6-25#    21-11
XOPLP0  141-34#  141-46
XOPLP1  141-38#  141-45
XOPLP2  141-48#  141-52
XREAD     4-5#   135-25
XWRITE    4-6#   132-29
ZEREDC   33-23     33-25#
```

CROSS REFERENCE TABLE (CREF V01-05)

```
BCS      11-1#   79-21   79-31   97-24   97-34   98-12  107-15  107-25  107-37  107-71  107-90  114-18  132-39  141-62
CERROR   16-65#  35-53   56-33   56-39   56-51   56-60   56-68   57-22   75-11   76-93  126-36  140-10  148-16
DEVFTL   16-37#  48-23   49-29   49-31   57-15   60-15   78-22   79-15   79-22   79-37   79-42   79-53   79-55   80-18
         80-27   80-33   83-41   84-29   91-18  132-61
DIAG$$    9-5#   94-17  124-18  127-16  128-16  129-15  134-15  137-14  142-12  148-12
DMCODE    1-3#    2-39
DMEND     1-32#  148-35
DMOVLY    1-18#   2-39   92-9    94-1    124-2   127-2   128-2   129-2   134-2   137-2   142-2   148-2
DSTAT    17-3#
ENDERR   16-93#  35-15   35-47   44-59   49-32   54-20   56-26   87-54  126-34  127-48  127-52  127-56  129-41  132-15
        132-20  132-51  132-62  132-67  134-41  135-15  135-20  135-41  135-46  137-28  137-43  137-56  137-75  137-87
        139-9   141-77
ERRDF    13-33#
ERRHRD   13-39#
ERROR    16-49#  35-14   35-31   35-36   35-40   35-44   35-46   35-52   44-57   48-23   49-29   49-31   54-19   56-25
         57-15   60-15   78-22   79-15   79-22   79-37   79-42   79-53   79-55   80-18   80-27   80-33   83-41   84-29
         87-53   91-18  126-32  127-46  127-50  127-54  129-40  132-12  132-17  132-48  132-61  132-64  134-40  135-12
        135-17  135-38  135-43  137-26  137-41  137-54  137-72  137-84  139-8   141-65  144-28  144-32  144-34  145-48
        145-52  145-54
ERROR$   14-3#
ERRORC   16-75#  44-58  127-47  127-51  127-55  132-13  132-14  132-18  132-19  132-49  132-50  132-65  132-66  135-13
        135-14  135-18  135-19  135-39  135-40  135-44  135-45  137-27  137-42  137-55  137-73  137-74  137-85  137-86
        141-66  141-75  141-76
ERRSF    13-27#
ERRSFT   13-47#
GETP$    15-15#
HARDER   16-31#  35-14   35-31   35-36   35-40   35-44   35-46   35-52   54-19   56-25  127-46  127-50  129-40  134-40
        137-72  137-84  139-8   141-65  144-28  144-32  144-34  145-48  145-52  145-54
MOVMSG   16-84#  35-14   35-31   35-36   35-40   35-44   35-46   35-52   35-53   44-57   44-57   44-57   44-57   44-57
         44-58   44-58   46-32   48-23   49-29   49-29   49-31   50-57   54-19   56-25   56-25   56-25   56-25   56-25
         56-25   56-25   56-25   56-25   56-33   56-39   56-51   56-60   56-68   57-15   57-22   60-15   75-11   76-93
         78-22   79-15   79-22   79-37   79-42   79-42   79-42   79-53   79-55   79-55   79-55   80-18   80-27   80-27
         80-27   80-33   80-33   80-33   83-41   83-41   83-41   84-29   87-53   91-18  126-32  126-32  126-32  126-32
        126-32  126-32  126-32  126-36  127-46  127-46  127-46  127-46  127-47  127-47  127-47  127-50  127-50  127-50
        127-50  127-51  127-51  127-51  127-54  127-54  127-54  127-54  127-55  127-55  127-55  129-40  129-40  129-40
        129-40  129-40  132-12  132-12  132-12  132-12  132-13  132-13  132-13  132-14  132-14  132-14  132-14  132-17
        132-17  132-17  132-17  132-18  132-18  132-18  132-19  132-19  132-19  132-48  132-48  132-48  132-48  132-48
        132-49  132-49  132-49  132-49  132-49  132-50  132-50  132-50  132-61  132-64  132-64  132-64  132-64  132-64
        132-65  132-65  132-65  132-65  132-65  132-66  132-66  132-66  134-40  134-40  134-40  134-40  134-40  135-12
        135-12  135-12  135-12  135-13  135-13  135-13  135-14  135-14  135-14  135-14  135-17  135-17  135-17  135-17
        135-18  135-18  135-18  135-19  135-19  135-19  135-38  135-38  135-38  135-38  135-38  135-39  135-39  135-39
        135-39  135-39  135-40  135-40  135-40  135-43  135-43  135-43  135-43  135-43  135-44  135-44  135-44  135-44
        135-44  135-45  135-45  135-45  137-26  137-26  137-26  137-26  137-26  137-27  137-27  137-27  137-27  137-27
        137-41  137-41  137-41  137-41  137-41  137-42  137-42  137-42  137-42  137-42  137-54  137-54  137-54  137-54
        137-54  137-55  137-55  137-55  137-55  137-55  137-72  137-72  137-72  137-72  137-72  137-73  137-73  137-73
        137-73  137-73  137-74  137-74  137-84  137-84  137-84  137-84  137-84  137-85  137-85  137-85  137-85  137-85
        137-86  137-86  139-8   140-10  141-65  141-65  141-65  141-65  141-65  141-66  141-66  141-66  141-66  141-75
        141-75  141-75  141-75  141-75  141-75  141-75  141-75  141-75  141-75  141-75  141-76  141-76  141-76  141-76
        141-76  141-76  144-28  144-28  144-28  144-28  144-28  144-32  144-32  144-32  144-32  144-32  144-34  144-34
        144-34  144-34  144-34  145-48  145-48  145-48  145-52  145-52  145-52  145-54  145-54  145-54  148-16
MSG      10-3#   36-5    36-6    36-7    36-8    36-9    36-10   36-11   36-12   36-13   36-15   36-16
MSSG     16-111#  46-32   50-57
MSSGE    16-133#
NDERR    16-101#  35-15   35-47   44-59   49-32   54-20   56-26   87-54  126-34  127-48  127-52  127-56  129-41  132-15
        132-20  132-51  132-62  132-67  134-41  135-15  135-20  135-41  135-46  137-28  137-43  137-56  137-75  137-87
        139-9   141-77
```

CROSS REFERENCE TABLE (CREF V01-05)