RC25

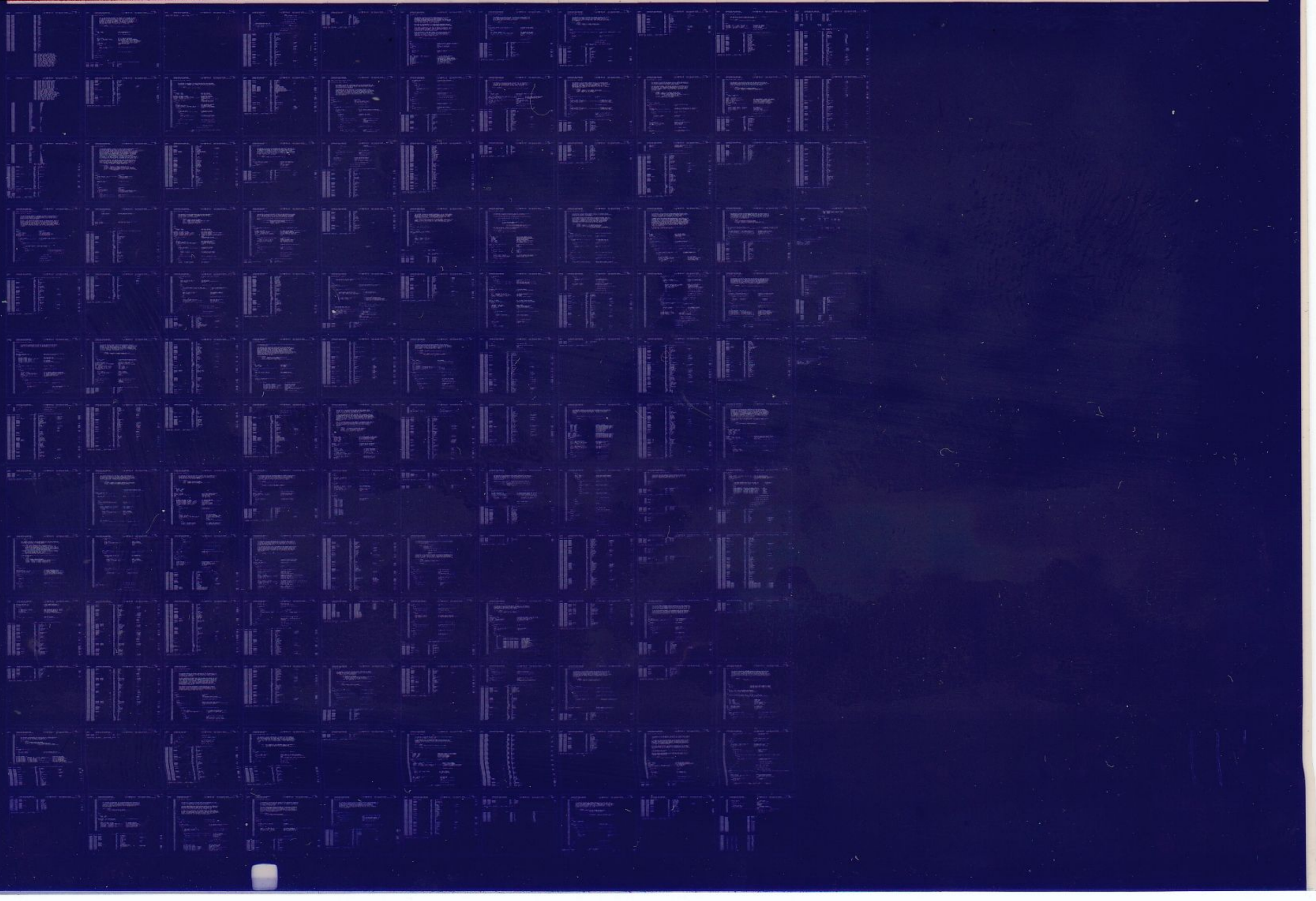
RC25 DISK EXERCISER
CZRCDA0

AH-T267A-MC
FICHE 1 OF 2

RC25

RC25 DISK EXERCISER
CZRCDA0

AH-T267A-MC
FICHE 2 OF 2



%(

IDENTIFICATION

PRODUCT CODE: AC-T266A-MC

PRODUCT NAME: CZRCDA0 RC25 DISK EXERCISER

PRODUCT DATE: JULY 13, 1983

MAINTAINER: SMALL STORAGE SYSTEMS DIAGNOSTICS

AUTHOR: JAMES S. DOUCETTE

TABLE OF CONTENTS

1.0 GENERAL INFORMATION

1.1 PROGRAM ABSTRACT

THE RC25 DISK EXERCISER IS DESIGNED TO VERIFY THE INTEGRITY OF THE DRIVE(S) UNDER TEST, AND TO DETECT FAULTS AT THE FUNCTIONAL LEVEL ONLY. UNDER THE MULTI-DRIVE SUBTEST, THE PROGRAM SHOULD GIVE A CLEAR INDICATION AS TO HOW AN RC25 DRIVE WILL FUNCTION EITHER ALONE OR WITH OTHER RC25 DRIVES UNDER STRESSFUL OPERATING SYSTEM CONDITIONS. THESE CONDITIONS ARE CREATED BY ISSUING A HEAVY LOAD OF MSCP I/O COMMANDS TO ALL ONLINE UNITS. THE USER CAN CONTROL MANY OF THE I/O COMMAND PARAMETERS THROUGH THE HARDWARE AND SOFTWARE QUESTIONS, INCLUDING FUNCTION (WRITE-ONLY, READ-ONLY, WRITES AND READS, WRITE-COMPARES, READ-COMPARES), DISK BLOCK SELECTION (RANDOM VS. SEQUENTIAL), TRACK NUMBER LIMITS, AND THE SELECTION OF DATA PATTERNS. IN ADDITION, THE USER CAN CHOOSE INSTEAD TO RUN THE DM EXERCISER SUBTEST. THIS SUBTEST CONSISTS OF THE ROM-RESIDENT FRONT PANEL EXERCISER RUNNING IN THE DM (DIAGNOSTIC MACHINE) UNDER HOST CONTROL. SEE SECTION 6.0 FOR MORE INFORMATION ON EACH EXERCISER SUBTEST.

THIS EXERCISER CAN TEST UP TO 4 CONTROLLERS (4 DIFFERENT IP ADDRESSES), EACH CONTROLLER HAVING UP TO 2 DRIVES, AND EACH DRIVE WITH 2 PLATTERS.

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

THE FOLLOWING HARDWARE IS REQUIRED TO RUN THE RC25 DISK EXERCISER:

* PDP-11 CPU
* 28K WORDS OF MEMORY (MEMORY ABOVE 28K, IF AVAILABLE, WILL BE USED FOR DATA TRANSFERS)
* FROM 1 TO 4 RC25 CONTROLLERS WITH UP TO 2 DRIVES PER CONTROLLER AND EXACTLY 2 PLATTERS PER DRIVE (I.E., A REMOVABLE CARTRIDGE MUST BE PRESENT IN EACH DRIVE, ALTHOUGH IT NEED NOT PARTICIPATE IN THE TEST)
* XXDP+ MEDIA DEVICE (I.E., RK05, RL02)
* KW11-L OR KW11-P CLOCK
* CONSOLE TERMINAL

1.3 RELATED DOCUMENTS AND STANDARDS

* CHQUS - XXDP+ USER'S MANUAL
* UNIBUS/Q-BUS STORAGE SYSTEMS PORT (UQSSP), V1.5
* MASS STORAGE COMMUNICATION PROTOCOL (MSCP), V1.2
* DIAGNOSTIC AND UTILITIES PROTOCOL (DUP), V0.5
* FRONT PANEL EXERCISER DIAGNOSTIC SPECIFICATION, REV 1.2
* RC25 AZTEC ENGINEERING SPECIFICATION, REV 5

1.4 DIAGNOSTIC HIERARCHY PREREQUISITES

ALL RC25 DRIVE-UNITS TO BE EXERCISED BY THIS PROGRAM MUST HAVE BEEN SUCCESSFULLY VERIFIED BY THE RC25 AZTEC FRONT-END / HOST DIAGNOSTIC. THE FRONT-END / HOST DIAGNOSTIC REQUIRES THAT THE BUS, HOST PROCESSOR, MEMORY, SYSTEM CLOCK AND CONSOLE TERMINAL ARE ALL FUNCTIONING PROPERLY.

1.5 ASSUMPTIONS

* REMOVABLE CARTRIDGE IS PRESENT IN ALL RC25 DRIVE-UNITS TO BE EXERCISED
* ALL RC25 DRIVE-UNITS HAVE BEEN SUCCESSFULLY SPUN-UP
* ALL RC25 CONTROLLERS WILL INTERRUPT THE HOST AT BR LEVEL 5 OR LESS.

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 (BY ^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".

FOR THE RC25 DISK EXERCISER, THE START, RESTART, AND CONTINUE COMMANDS PERFORM SIMILAR FUNCTIONS. THE DIFFERENCES ARE SUMMARIZED IN THE TABLE BELOW:

	START	RESTART	CONTINUE
ALLOW CHANGES TO HARDWARE CONFIGURATION	X		
CLEAR ELAPSED TIME OF EXERCISER TO 00:00:00	X		
SAVE STATISTICS ACCUMULATED DURING THE LAST INCOMPLETE PASS			X
RESET THE COMMAND REFERENCE NUMBER USED IN MSCP COMMANDS	X	X	
IF SEQUENTIAL LBN MODE, RESET THE STARTING BLOCK NUMBER USED IN I/O OPERATIONS	X	X	

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.

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					

FOR THE RC25 DISK EXERCISER, THE /TESTS SWITCH IS MEANINGLESS SINCE THERE IS ONLY ONE TEST.

2.3 FLAGS

FLAGS ARE USED TO SET UP CERTAIN OPERATIONAL PARAMETERS SUCH A 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 (GENERAL, BASIC, AND EXTENDED)
IBR*	INHIBIT BASIC AND EXTENDED ERROR REPORTS
IXR*	INHIBIT EXTENDED ERROR REPORTS
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
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
```

FOR THE RC25 DISK EXERCISER, THE FOLLOWING FLAGS HAVE NO EFFECT: ADR AND EVL.

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 (A) THE HARDWARE INFORMATION HAS BEEN "PRELOADED" USING THE SETUP UTILITY (SEE CHAPTER 6 OF THE XXDP+ USER'S MANUAL), OR (B) YOU WISH TO USE THE DEFAULT HARDWARE CONFIGURATION THAT IS SUPPLIED WITH THE PROGRAM (SEE BELOW). 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 HARDWARE QUESTIONS FOR EACH UNIT. UNDER THIS PROGRAM, A UNIT IS DEFINED TO BE ONE DISK PLATTER.

A WORD ABOUT SEMANTICS: IN ALL MESSAGES OUTPUT BY THE RC25 EXERCISER, AS WELL AS IN THE DOCUMENTATION, THE PHRASES "UNIT X" AND "UNIT NUMBER" REFER TO THE SEQUENTIALLY-ASSIGNED UNIT NUMBER GIVEN BY THE DIAGNOSTIC SUPERVISOR. THE PHRASES "PLATTER X" AND "PLATTER ADDRESS" REFER TO THE UNIT NUMBER ASSIGNED TO THE DISK THROUGH THE UNIT PLUG ON THE FRONT OF THE DRIVE. THESE TWO NUMBERS ARE NOT RELATED.

DEFAULT ANSWERS APPEAR WITH QUESTIONS FOR WHICH THE USER MAY SIMPLY TYPE A CARRIAGE RETURN.

IP ADDRESS (O) 172150 ?
ENTER THE ADDRESS OF THE IP REGISTER OF ONE RC25 AS ADDRESSED BY THE PROCESSOR WITH MEMORY MANAGEMENT TURNED OFF. THE PROGRAM EXPECTS AN EVEN 16-BIT ADDRESS IN THE RANGE OF 160000 TO 177774.

VECTOR (O) 154 ?
ANSWER WITH THE INTERRUPT VECTOR OF THE SAME RC25 IN THE ABOVE QUESTION. AN EVEN VECTOR ADDRESS IN THE RANGE OF 4 TO 774 WILL BE ACCEPTED. SINCE RC25 VECTOR ADDRESSES ARE PROGRAMMABLE AND NOT HARDWIRED, CARE MUST BE TAKEN NOT TO SPECIFIY A VECTOR WHICH IS USED FOR ANOTHER PERIPHERAL DEVICE.

BR LEVEL (O) 5 ?
ANSWER WITH THE BUS REQUEST INTERRUPT LEVEL USED BY THE ABOVE RC25. LEVELS 4 THORUGH 7 ARE ACCEPTABLE. (NOTE: THE PROGRAM EXPECTS THAT ALL RC25'S WILL BE AT BR LEVEL 5 OR LESS. IF THIS IS NOT THE CASE, THEN ONLY ONE RC25 CONTROLLER WITH UP TO FOUR UNITS MAY BE EXERCISED AT A TIME).

PLATTER ADDRESS (UNIT PLUG) (D) 0 ?
ENTER A NUMBER BETWEEN 0 AND 253 FOR ONE DISK WHICH CORRESPONDS TO THE UNIT PLUG ON THE FRONT OF THE DEVICE. EVEN NUMBERS DESIGNATE THE REMOVABLE PLATTER (UPPER DISK), ODD NUMBERS REFER TO THE FIXED PLATTER (LOWER DISK).

ALLOW WRITES TO CUSTOMER DATA AREA ON THIS PLATTER (L) ?
NO DEFAULT. IF THE MULTI-DRIVE SUBTEST IS BEING RUN, A "NO" ANSWER HERE WILL FORCE THE EXERCISER TO PERFORM READ-ONLY OPERATIONS TO THE CUSTOMER DATA AREA OF THIS UNIT. IF THE DM EXERCISER IS BEING RUN, THIS QUESTION HAS NO EFFECT, SINCE THE DM EXERCISER WRITES ON THE DIAGNOSTIC TRACKS ONLY. IN ANY CASE, IF YOU ANSWER "YES", THE FOLLOWING QUESTION WILL APPEAR.

** WARNING - CUSTOMER DATA AREA MAY BE OVERWRITTEN! ... CONFIRM (L) ?
NO DEFAULT. A "YES" ANSWER IS EXPECTED, BUT A "NO" ANSWER WILL PREVENT THE CUSTOMER DATA AREA FROM BEING OVERWRITTEN.

EXAMPLE OF HARDWARE DIALOG:

```
CHANGE HW (L)  ? Y<CR>

# UNITS (D)  ? 2<CR>

UNIT 0
IP ADDRESS (O)  172150 ? <CR>
VECTOR (O)  154 ? <CR>
BR LEVEL (O)  5 ? <CR>
PLATTER ADDRESS ... (D)  0 ? 252<CR>
ALLOW WRITES ... (L)  ? Y<CR>
** WARNING - ... CONFIRM (L)  ? N<CR>   ! USER REVERSES ANSWER

UNIT 1
IP ADDRESS (O)  172150 ? <CR>
VECTOR (O)  154 ? <CR>
BR LEVEL (O)  5 ? <CR>
PLATTER ADDRESS ... (D)  252 ? 253<CR>
ALLOW WRITES ... (L) ? Y<CR>
** WARNING - ... CONFIRM (L)  ? Y<CR>
```

AS MENTIONED ABOVE, THERE IS A DEFAULT HARDWARE CONFIGURATION THAT IS SUPPLIED WITH THE PROGRAM. IT CONSISTS OF TWO UNITS AS FOLLOWS:

```
UNIT 0
IP ADDRESS:      172150
VECTOR:          154
BR LEVEL:        5
PLATTER ADDRESS: 0
ALLOW WRITES?    N

UNIT 1
IP ADDRESS:      172150
VECTOR:          154
BR LEVEL:        5
PLATTER ADDRESS: 1
ALLOW WRITES?    N
```

THIS CONFIGURATION TAKES EFFECT IF THE USER DOES NOT USE THE SETUP UTILITY AND HAS BYPASSED THE HARDWARE QUESTIONS WHEN THE PROGRAM IS FIRST STARTED.

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 BELOW:

```
ERROR LIMIT (0 FOR NO LIMIT) (D)  32 ?
        ENTER THE NUMBER OF HARD ERRORS ALLOWED PER UNIT PER PASS
        BEFORE THE UNIT IS DROPPED FROM TESTING. A NUMBER IN THE
        RANGE OF 0 TO 65535 WILL BE ACCEPTED.
```

```
TRANSFER LIMIT IN MEGABYTES (0 FOR NO LIMIT) (D)  2 ?
        ENTER THE NUMBER OF MEGABYTES TO BE TRANSFERRED TO / FROM EACH
        UNIT BEFORE AN END-OF-PASS IS DECLARED. A NUMBER IN THE RANGE
        OF 0 TO 65535 WILL BE ACCEPTED. AS A REFERENCE, ON A SYSTEM
        CONFIGURED WITH ONE DRIVE (TWO UNITS) AND 128K OF MEMORY, THE
        MULTI-DRIVE SUBTEST WILL REACH A 10 MEGABYTE TRANSFER LIMIT IN
        APPROXIMATELY 8 MINUTES. FOR THE SAME SYSTEM, THE DM EXERCISER
        SUBTEST WILL REACH A 2 MEGABYTE LIMIT IN APPROXIMATELY 20
        MINUTES.
SUPPRESS PRINTING ERROR LOG MESSAGES (L) Y ?
        ERROR LOG MESSAGES USUALLY DESCRIBE MINOR ERRORS ENCOUNTERED
        DURING I/O OPERATIONS, SUCH AS ECC SYMBOL ERRORS. ANSWER "N"
        TO SEE ALL ERROR LOG MESSAGES.
RUN DM EXERCISER INSTEAD OF MULTI-DRIVE SUBTEST (L) N ?
        THIS QUESTION DETERMINES WHICH OF THE TWO EXERCISER SUBTESTS
        WILL BE RUN. THE MULTI-DRIVE SUBTEST IS HOST-DRIVEN WHILE THE
        DM EXERCISER IS THE FRONT PANEL TEST EXECUTING IN THE
        DIAGNOSTIC MACHINE AT THE DEVICE. SEE SECTION 6.0 FOR MORE
        INFORMATION. THE REMAINING QUESTIONS WILL ONLY BE ASKED IF THE
        MULTI-DRIVE SUBTEST IS BEING RUN.
RANDOM SEEK MODE (L) Y ?
        ANSWERING "Y" WILL CAUSE EACH I/O OPERATION TO SELECT A RANDOM
        STARTING DISK BLOCK NUMBER. ANSWER "N" TO HAVE STARTING BLOCK
        NUMBERS SELECTED SEQUENTIALLY ON EACH PLATTER.
STARTING TRACK (D)  0 ?
        ENTER THE STARTING TRACK NUMBER OF THE DISK AREA YOU WISH TO
        TEST. A NUMBER IN THE RANGE 0 TO 1641 WILL BE ACCEPTED.
ENDING TRACK (D)  1641 ?
        ENTER THE ENDING TRACK NUMBER OF THE DISK AREA YOU WISH TO
        TEST.
READ-COMPARES PERFORMED AT THE CONTROLLER (L) Y ?
        A "YES" ANSWER WILL CAUSE ALL READ OPERATIONS TO BE CHANGED TO
        READ-COMPARE. THIS ESSENTIALLY FORCES THE CONTROLLER TO PERFORM
        TWO READ OPERATIONS ON THE SAME DISK SECTORS AND TO COMPARE THE
        RESULTS. AFTER ANSWERING THIS QUESTION, THE FOLLOWING MESSAGE
        APPEARS:

THE REMAINING QUESTIONS ONLY APPLY TO UNPROTECTED PLATTERS.

WRITE ONLY (L) N ?
        ANSWER "Y" FOR WRITES ONLY; OTHERWISE THE PROGRAM WILL CHOOSE
        WRITES AND READS ON A RANDOM BASIS.
WRITE-COMPARES PERFORMED AT THE CONTROLLER (L) Y ?
        ANSWERING "YES" CAUSES ALL WRITE I/O REQUESTS TO BE CHANGED TO
        WRITE-COMPARE. IN THIS SITUATION, AFTER EACH WRITE, THE
        CONTROLLER WILL READ BACK THE DATA FROM THE MEDIA AND COMPARE
        IT TO THE ORIGINAL DATA RE-OBTAINED FROM THE HOST.
CHECK ALL WRITES AT HOST BY READING (L) N ?
        THIS QUESTION WILL ONLY BE ASKED IF THE PREVIOUS QUESTION WAS
        ANSWERED "NO". A "YES" ANSWER HERE CAUSES EACH HOST WRITE
        REQUEST TO BE FOLLOWED BY A READ REQUEST. WHEN BOTH OPERATIONS
        HAVE COMPLETED, THE HOST WILL COMPARE THE DATA STORED IN THE
        TWO I/O BUFFERS. THERE ARE SOME UNDESIRABLE IMPLICATIONS IN
        CHOOSING THIS OPTION, INCLUDING THE CONSUMPTION OF EXTRA CPU
        TIME, AND THE OVERRIDING OF THE CONTROLLER'S NORMAL
        OPTIMIZATION POLICIES (DUE TO THE "EXPRESS REQUEST" COMMAND
        MODIFIER WHICH MUST BE SET FOR EACH WRITE-READ PAIR).
```

```
USER-DEFINED DATA PATTERN (L) N ?
        AN ANSWER OF "YES" ALLOWS YOU TO DEFINE YOUR OWN DATA PATTERN
        (UP TO 16 WORDS) TO BE USED IN ALL WRITE OPERATIONS. A "NO"
        ANSWER ALLOWS THE USE OF OTHER PRE-DEFINED DATA PATTERNS WHICH
        MAY BE SELECTED IN THE NEXT QUESTION.
SELECT PRE-DEFINED DATA PATTERN (0 FOR SEQUENTIAL SELECTION) (D)  0 ?
        THIS QUESTION WILL ONLY BE ASKED IF THE PREVIOUS QUESTION WAS
        ANSWERED "NO". THE 21 PRE-DEFINED DATA PATTERNS ARE SHOWN IN
        APPENDIX A. AN ANSWER OF "0" CAUSES EACH OF THE 21 PATTERNS TO
        BE SELECTED SEQUENTIALLY ON EACH UNIT. NOTE THAT PATTERN 1
        CONSISTS ENTIRELY OF RANDOM NUMBERS. THIS COMPLETES THE SOFT-
        WARE QUESTIONING; THE FOLLOWING QUESTIONS ARE ASKED ONLY IF YOU
        CHOSE TO DEFINE YOUR OWN DATA PATTERN.
NUMBER OF WORDS IN DATA PATTERN (16 MAXIMUM) (D)  16 ?
PATTERN VALUE (0) ?
        THIS QUESTION REPEATED AS APPROPRIATE.
```

2.6 EXTENDED P-TABLE DIALOGUE

WHEN YOU ANSWER THE HARDWARE QUESTIONS, YOU ARE BUILDING ENTRIES IN A TABLE THAT DESCRIBES THE DEVICES UNDER TEST. THE SIMPLEST WAY TO BUILD THIS TABLE IS TO ANSWER ALL QUESTIONS FOR EACH UNIT TO BE TESTED. IF YOU HAVE A MULTIPLEXED DEVICE SUCH AS A MASS STORAGE CONTROLLER WITH SEVERAL DRIVES OR A COMMUNICATION DEVICE WITH SEVERAL LINES, THIS BECOMES TEDIOUS SINCE MOST OF THE ANSWERS ARE REPETITIOUS.

TO ILLUSTRATE A MORE EFFICIENT METHOD, SUPPOSE YOU ARE TESTING A FICTIONAL DEVICE, THE XY11. SUPPOSE THIS DEVICE CONSISTS OF A CONTROL MODULE WITH EIGHT UNITS (SUB-DEVICES) ATTACHED TO IT. THESE UNITS ARE DESCRIBED BY THE OCTAL NUMBERS 0 THROUGH 7. THERE IS ONE HARDWARE PARAMETER THAT CAN VARY AMONG UNITS CALLED THE Q-FACTOR. THIS Q-FACTOR MAY BE 0 OR 1. BELOW IS A SIMPLE WAY TO BUILD A TABLE FOR ONE XY11 WITH EIGHT UNITS.

```
# UNITS (D) ?  8<CR>

UNIT 1
CSR ADDRESS (0) ?  160000<CR>
SUB-DEVICE # (0) ?  0<CR>
Q-FACTOR (0) 0 ?  1<CR>

UNIT 2
CSR ADDRESS (0) ?  160000<CR>
SUB-DEVICE # (0) ?  1<CR>
Q-FACTOR (0) 1 ?  0<CR>

UNIT 3
CSR ADDRESS (0) ?  160000<CR>
SUB-DEVICE # (0) ?  2<CR>
Q-FACTOR (0) 0 ?  <CR>

UNIT 4
CSR ADDRESS (0) ?  160000<CR>
SUB-DEVICE # (0) ?  3<CR>
Q-FACTOR (0) 0 ?  <CR>
```

```
        UNIT 5
        CSR ADDRESS (O) ?  160000<CR>
        SUB-DEVICE # (O) ?  4<CR>
        Q-FACTOR (O) 0 ?  <CR>

        UNIT 6
        CSR ADDRESS (O) ?  160000<CR>
        SUB-DEVICE # (O) ?  5<CR>
        Q-FACTOR (O) 0 ?  <CR>

        UNIT 7
        CSR ADDRESS (O) ?  160000<CR>
        SUB-DEVICE # (O) ?  6<CR>
        Q-FACTOR (O) 0 ?  1<CR>

        UNIT 8
        CSR ADDRESS (O)  160000<CR>
        SUB-DEVICE # (O) ?  7<CR>
        Q-FACTOR (O) 1 ?  <CR>
```

NOTICE THAT THE DEFAULT VALUE FOR THE Q-FACTOR CHANGES WHEN A NON-DEFAULT RESPONSE IS GIVEN. BE CAREFUL WHEN SPECIFYING MULTIPLE UNITS!

AS YOU CAN SEE FROM THE ABOVE EXAMPLE, THE HARDWARE PARAMETERS DO NOT VARY SIGNIFICANTLY FROM UNIT TO UNIT. THE PROCEDURE SHOWN IS NOT VERY EFFICIENT.

THE RUNTIME SERVICES CAN TAKE MULTIPLE UNIT SPECIFICATIONS HOWEVER. LET'S BUILD THE SAME TABLE USING THE MULTIPLE SPECIFICATION FEATURE.

```
        # UNITS (D) ?  8<CR>

        UNIT 1
        CSR ADDRESS (O) ?  160000<CR>
        SUB-DEVICE # (O) ?  0,1<CR>
        Q-FACTOR (O) 0 ?  1,0<CR>

        UNIT 3
        CSR ADDRESS (O) ?  160000<CR>
        SUB-DEVICE # (O) ?  2-5<CR>
        Q-FACTOR (O) 0 ?  0<CR>

        UNIT 7
        CSR ADDRESS (O) ?  160000<CR>
        SUB-DEVICE # (O) ?  6,7<CR>
        Q-FACTOR (O) 0 ?  1<CR>
```

AS YOU CAN SEE IN THE ABOVE DIALOGUE, THE RUNTIME SERVICES WILL BUILD AS MANY ENTRIES AS IT CAN WITH THE INFORMATION GIVEN IN ANY ONE PASS THROUGH THE QUESTIONS. IN THE FIRST PASS, TWO ENTRIES ARE BUILT SINCE TWO SUB-DEVICES AND Q-FACTORS WERE SPECIFIED. THE SERVICES ASSUME THAT THE CSR ADDRESS IS 160000 FOR BOTH SINCE IT WAS SPECIFIED ONLY ONCE. IN THE SECOND PASS, FOUR ENTRIES WERE BUILT. THIS IS BECAUSE FOUR SUB-DEVICES WERE SPECIFIED. THE "-" CONSTRUCT TELLS THE RUNTIME SERVICES TO INCREMENT THE DATA FROM THE FIRST NUMBER TO THE SECOND. IN THIS CASE, SUB-DEVICES 2, 3, 4 AND 5 WERE SPECIFIED. (IF THE SUB-DEVICE WERE SPECIFIED BY ADDRESSES, THE INCREMENT WOULD BE BY 2 SINCE

ADDRESSES MUST BE ON AN EVEN BOUNDARY). THE CSR ADDRESSES AND Q-FACTORS FOR THE FOUR ENTRIES ARE ASSUMED TO BE 160000 AND 0 RESPECTIVELY SINCE THEY WERE ONLY SPECIFIED ONCE. THE LAST TWO UNITS ARE SPECIFIED IN THE THIRD PASS.

THE WHOLE PROCESS COULD HAVE BEEN ACCOMPLISHED IN ONE PASS AS SHOWN BELOW.

```
# UNITS (D) ?  8<CR>

UNIT 1
CSR ADDRESS (O) ?  160000<CR>
SUB-DEVICE # (O) ?  0-7<CR>
Q-FACTOR (O) 0 ?  0,1,0,,,,1,1<CR>
```

AS YOU CAN SEE FROM THIS EXAMPLE, NULL REPLIES (COMMAS ENCLOSING A NULL FIELD) TELL THE RUNTIME SERVICES TO REPEAT THE LAST REPLY.

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-1) (N IS THE NUMBER OF UNITS CONFIGURED)
        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. FOR THE RC25 DISK EXERCISER, ALL BASIC ERROR MESSAGES CONTAIN THE ELAPSED TIME (THE TIME SINCE THE LAST START COMMAND).

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.

3.2 SPECIFIC ERROR MESSAGES

THIS LIST OF RC25 EXERCISER ERROR MESSAGES IS ARRANGED BY ERROR NUMBER. THE LETTERS G, B, AND X REFER TO THE GENERAL, BASIC, AND EXTENDED TEXT OF THE ERROR MESSAGE, RESPECTIVELY. ALL NUMERIC VALUES ARE IN OCTAL EXCEPT FOR THE ELAPSED TIME OR WHEN THE NUMBER IS FOLLOWED BY A DECIMAL POINT.

3.2.1 SYSTEM-FATAL ERROR MESSAGES

THESE CONDITIONS WILL CAUSE THE DIAGNOSTIC TO BE ABORTED. THEY WILL APPEAR AFTER THE SOFTWARE DIALOG.

```
1  G  TOO MANY UNITS
   B  XX:XX:XX  MORE THAN 16. UNITS SPECIFIED

2  G  NEITHER P NOR L CLOCK WAS FOUND ON THE SYSTEM
```

3.2.2 CONFIGURATION ERROR MESSAGES

CONFIGURATION ERRORS INDICATE ILLEGAL HARDWARE CONFIGURATIONS. THESE SITUATIONS ARE DETECTED BEFORE ANY ACTUAL TESTING TAKES PLACE. AS SUCH, THEY ARE SPECIAL IN THAT THERE IS NO ERROR NUMBER ASSOCIATED WITH THEM, AND THEY ARE NOT INCLUDED IN THE SUMMARY REPORT NOR THE END-OF-PASS MESSAGE. ANY UNITS INVOLVED ARE IMMEDIATELY DROPPED.

IF YOU TRY TO CONFIGURE MORE THAN 4 UNITS AT THE SAME IP ADDRESS, THE FOLLOWING MESSAGE WILL APPEAR:

ALREADY 4 UNITS AT IP XXXXXX(O)

THE REMAINING UNITS (OVER 4) WILL BE DROPPED. IF YOU TRY TO CONFIGURE MORE THAN 4 CONTROLLERS, THE FOLLOWING MESSAGE WILL APPEAR:

MORE THAN 4 DIFFERENT IP ADDRESSES

ALL UNITS ASSOCIATED WITH THE 5TH (AND HIGHER) CONTROLLERS WILL BE DROPPED. FINALLY, IF YOU SPECIFY IDENTICAL PLATTER ADDRESSES UNDER ONE CONTROLLER (IP ADDRESS), THE FOLLOWING MESSAGE WILL APPEAR:

DUPLICATE PLATTER ADDRESS XXX. AT IP XXXXXX(O)

ALL UNITS (BEYOND THE FIRST) WITH IDENTICAL PLATTER ADDRESSES WILL BE DROPPED.

3.2.3 DEVICE-FATAL ERROR MESSAGES

IF THE ERROR PERTAINS TO A CONTROLLER, THEN ALL THE UNITS ATTACHED TO THE CONTROLLER WILL BE DROPPED. OTHERWISE, THE ERROR PERTAINS TO A SINGLE UNIT, AND ONLY THAT UNIT IS DROPPED.

```
10  G  REGISTER EXISTENCE TEST FAILED
    B  XX:XX:XX  NO RESPONSE AT ADDRESS XXXXXX(O)

11     ABOUT TO VERIFY VECTOR XXX(O) FOR DEVICE XXXXXX(O) ...
           (FOLLOWED BY EITHER NOTHING OR)
    G  VECTOR TEST FAILED

12  G  BR LEVEL TEST FAILED
    B  XX:XX:XX  INCORRECT BR LEVEL GIVEN FOR DEVICE XXXXXX(O)
```

ERROR 13 REFERS TO THE FOUR STEP HARD INITIALIZE SEQUENCE DESCRIBED IN THE UNIBUS / Q-BUS STORAGE SYSTEMS PORT (UQSSP) SPEC. THE SA REGISTER CODE IS PROVIDED TO ASSIST IN DIAGNOSIS.

```
13  G  INIT SEQUENCE FAILED
    B  XX:XX:XX  STEP X READ ERROR ON DEVICE XXXXXX(O)
    X       SA: XXXXXX(O)
```

ERROR 14 MEANS THAT THE ERROR BIT (BIT 15) OF A DEVICE'S SA REGISTER WAS FOUND SET TO 1. THE LOW-ORDER 11 BITS DESCRIBE THE ERROR.

```
14  G  FATAL PORT / CONTROLLER ERROR
    B  XX:XX:XX  ERROR CODE RECEIVED IN SA REGISTER OF DEVICE
                                XXXXXX(O)
    X       SA: XXXXXX(O)
```

```
ERROR 15 IS REPORTED WHEN THE RC25 FAILS TO RESPOND TO ONE OF THE
LISTED MSCP COMMANDS.

        15  G  MESSAGE RESPONSE TIMEOUT
            B  XX:XX:XX  FAILED TO RECEIVE END MESSAGE FROM DEVICE
                                 XXXXXX(O)
            X       COMMAND: SET CTLR CHAR (OR)
                             ONLINE (OR)
                             EXECUTE SUPPLIED PROGRAM (OR)
                             SEND DATA

ERROR 16 REFERS TO THE MSCP "ONLINE" MESSAGE SENT TO EACH UNIT DURING
THE INITIALIZATION SUBTEST. THE STATUS CODE AND SUB-CODE ARE TAKEN
FROM THE MSCP END MESSAGE (AS THEY ARE IN ALL ERROR MESSAGES).

        16  G  ONLINE FAILED
            B  XX:XX:XX  ERROR IN RESPONSE TO ONLINE COMMAND FOR
                                 PLATTER XXX.
            X       STATUS CODE: XX(O)
            X       SUB-CODE: XXXX(O)

ERROR 17 IMPLIES THAT (A) YOU HAVE ALLOWED WRITES TO THE CUSTOMER DATA
AREA THROUGH THE HARDWARE QUESTION, AND (B) THE WRITE-PROTECT
PUSHBUTTON FOR THE UNIT IS ENGAGED.

        17  G  WRITE-PROTECT CONFLICT
            B  XX:XX:XX  PLATTER XXX. IS SW WRITE-ENABLED BUT HW
                                 WRITE-PROTECTED

ERROR 18 REFERS TO THE MSCP "ACCESS" COMMANDS WHICH ARE ISSUED TO EACH
UNIT AS PART OF THE INITIALIZATION SUBTEST. IF EXTENDED ERROR MESSAGES
ARE ENABLED AND THE STATUS CODE AND SUB-CODE DO NOT APPEAR, THEN THE
ACCESS COMMAND HAS TIMED OUT (I.E., NO RESPONSE RECEIVED WITHIN THE
REQUIRED INTERVAL).

        18  G  ACCESS FAILED
            B  XX:XX:XX  ACCESS FAILED ON PLATTER XXX.
            X       STATUS CODE: XX(O)
            X       SUB-CODE: XXXX(O)

ERROR 19 OCCURS WHEN AN MSCP I/O COMMAND RESULTS IN AN END MESSAGE WITH
THE "UNIT OFFLINE" STATUS CODE.

        19  G  FATAL I/O ERROR
            B  XX:XX:XX  PLATTER XXX. WENT OFFLINE
            X       CMD REF NUM: XXXXXX(O)
            X       SUB-CODE: XXXX(O)
            X       COMMAND: READ (OR)
                             READ-COMPARE (OR)
                             WRITE (OR)
                             WRITE-COMPARE (OR)
                             XXX(O)
            X       BAD BLOCK REPORTED: XXXXX.          ! EITHER THIS
            X       LBN: XXXXX.                         !   OR THIS
            X       BYTE COUNT IN COMMAND: XXXXX.
            X       ACTUAL # BYTES TRANSFERRED: XXXXX.
            X       I/O BUFFER DESCRIPTOR: XXXXXX(O) XXXXXX(O)
```

```
        20  G  CONTROLLER TIMEOUT
            B  XX:XX:XX  DEVICE XXXXXX(O) NOT PROCESSING COMMAND
                         PACKETS
```

ERROR 21 ARISES IF YOU ARE RUNNING THE DM EXERCISER SUBTEST, AND ONE OF THE INITIAL DUP COMMANDS WAS REJECTED WITH A STATUS CODE OTHER THAN "SUCCESS". NOTE THAT THE STATUS CODE IS NOT FROM MSCP.

```
        21  G  DUP COMMAND FAILED
            B  XX:XX:XX  MESSAGE REJECTED BY DUP DERVER ON DEVICE
                                 XXXXXX(O)
            X       COMMAND: EXECUTE SUPPLIED PROGRAM (OR)
            X                SEND DATA
            X       DUP STATUS CODE: X.
```

ERROR 22 IS ALSO RELATED TO THE DM EXERCISER SUBTEST. IT INDICATES THAT THE FRONT PANEL TEST (EXECUTING UNDER HOST CONTROL IN THE DM) HAS FAILED TO PROVIDE THE HOST WITH A SET OF OPERATING STATISTICS WITHIN A PRE-DETERMINED TIME INTERVAL. THE FPT IS ASSUMED TO BE DEAD.

```
        22  G  DM EXERCISER TIMEOUT
            B  XX:XX:XX  NO RESPONSE FROM FRONT PANEL TEST EXECUTING IN
                                 DEVICE XXXXXX(O)
```

3.2.4 HARD ERROR MESSAGES

MOST HARD ERRORS ARE REPORTED TO THE HOST THROUGH THE END MESSAGES ASSOCIATED WITH MSCP READ AND WRITE COMMANDS. ERROR NUMBERS 31 - 41 ARE BASED ON MSCP STATUS CODES 1 - 11.

```
        30  G  I/O REQUEST FAILED
            B  XX:XX:XX  PLATTER XXX. - STATUS CODE: XX(O)
            X       CMD REF NUM: XXXXXX(O)
            X       SUB-CODE: XXXX(O)
            X       COMMAND: READ (OR)
                             READ-COMPARE (OR)
                             WRITE (OR)
                             WRITE-COMPARE (OR)
                             XXX(O)
            X       BAD BLOCK REPORTED: XXXXX.              ! EITHER THIS
            X       LBN: XXXXX.                             !   OR THIS
            X       BYTE COUNT IN COMMAND: XXXXX.
            X       ACTUAL # BYTES TRANSFERRED: XXXXX.
            X       I/O BUFFER DESCRIPTOR: XXXXXX(O) XXXXXX(O)

        31  G  I/O REQUEST FAILED
            B  XX:XX:XX  PLATTER XXX. - INVALID COMMAND
            X       CMD REF NUM: XXXXXX(O)
            X       SUB-CODE: XXXX(O)
            X       COMMAND: READ (OR)
                             READ-COMPARE (OR)
                             WRITE (OR)
                             WRITE-COMPARE (OR)
                             XXX(O)
            X       BAD BLOCK REPORTED: XXXXX.              ! EITHER THIS
            X       LBN: XXXXX.                             !   OR THIS
            X       BYTE COUNT IN COMMAND: XXXXX.
            X       ACTUAL # BYTES TRANSFERRED: XXXXX.
            X       I/O BUFFER DESCRIPTOR: XXXXXX(O) XXXXXX(O)
```

```
32  G  I/O REQUEST FAILED
    B  XX:XX:XX  PLATTER XXX. - COMMAND ABORTED
    X      CMD REF NUM: XXXXXX(O)
    X      SUB-CODE: XXXX(O)
    X      COMMAND: READ (OR)
                    READ-COMPARE (OR)
                    WRITE (OR)
                    WRITE-COMPARE (OR)
                    XXX(O)
    X      BAD BLOCK REPORTED: XXXXX.            ! EITHER THIS
    X      LBN: XXXXX.                           !   OR THIS
    X      BYTE COUNT IN COMMAND: XXXXX.
    X      ACTUAL # BYTES TRANSFERRED: XXXXX.
    X      I/O BUFFER DESCRIPTOR: XXXXXX(O) XXXXXX(O)

34  G  I/O REQUEST FAILED
    B  XX:XX:XX  PLATTER XXX. - UNIT-AVAILABLE
    X      CMD REF NUM: XXXXXX(O)
    X      SUB-CODE: XXXX(O)
    X      COMMAND: READ (OR)
                    READ-COMPARE (OR)
                    WRITE (OR)
                    WRITE-COMPARE (OR)
                    XXX(O)
    X      BAD BLOCK REPORTED: XXXXX.            ! EITHER THIS
    X      LBN: XXXXX.                           !   OR THIS
    X      BYTE COUNT IN COMMAND: XXXXX.
    X      ACTUAL # BYTES TRANSFERRED: XXXXX.
    X      I/O BUFFER DESCRIPTOR: XXXXXX(O) XXXXXX(O)

35  G  I/O REQUEST FAILED
    B  XX:XX:XX  PLATTER XXX. - MEDIA FORMAT ERROR
    X      CMD REF NUM: XXXXXX(O)
    X      SUB-CODE: XXXX(O)
    X      COMMAND: READ (OR)
                    READ-COMPARE (OR)
                    WRITE (OR)
                    WRITE-COMPARE (OR)
                    XXX(O)
    X      BAD BLOCK REPORTED: XXXXX.            ! EITHER THIS
    X      LBN: XXXXX.                           !   OR THIS
    X      BYTE COUNT IN COMMAND: XXXXX.
    X      ACTUAL # BYTES TRANSFERRED: XXXXX.
    X      I/O BUFFER DESCRIPTOR: XXXXXX(O) XXXXXX(O)

36  G  I/O REQUEST FAILED
    B  XX:XX:XX  PLATTER XXX. - WRITE-PROTECTED
    X      CMD REF NUM: XXXXXX(O)
    X      SUB-CODE: XXXX(O)
    X      COMMAND: READ (OR)
                    READ-COMPARE (OR)
                    WRITE (OR)
                    WRITE-COMPARE (OR)
                    XXX(O)
    X      BAD BLOCK REPORTED: XXXXX.            ! EITHER THIS
    X      LBN: XXXXX.                           !   OR THIS
    X      BYTE COUNT IN COMMAND: XXXXX.
    X      ACTUAL # BYTES TRANSFERRED: XXXXX.
    X      I/O BUFFER DESCRIPTOR: XXXXXX(O) XXXXXX(O)
```

```
37  G  I/O REQUEST FAILED
    B  XX:XX:XX  PLATTER XXX. - DEVICE COMPARE ERROR
    X      CMD REF NUM: XXXXXX(O)
    X      SUB-CODE: XXXX(O)
    X      COMMAND: READ (OR)
                    READ-COMPARE (OR)
                    WRITE (OR)
                    WRITE-COMPARE (OR)
                    XXX(O)
    X      BAD BLOCK REPORTED: XXXXX.            ! EITHER THIS
    X      LBN: XXXXX.                           !   OR THIS
    X      BYTE COUNT IN COMMAND: XXXXX.
    X      ACTUAL # BYTES TRANSFERRED: XXXXX.
    X      I/O BUFFER DESCRIPTOR: XXXXXX(O) XXXXXX(O)

38  G  I/O REQUEST FAILED
    B  XX:XX:XX  PLATTER XXX. - DATA ERROR
    X      CMD REF NUM: XXXXXX(O)
    X      SUB-CODE: XXXX(O)
    X      COMMAND: READ (OR)
                    READ-COMPARE (OR)
                    WRITE (OR)
                    WRITE-COMPARE (OR)
                    XXX(O)
    X      BAD BLOCK REPORTED: XXXXX.            ! EITHER THIS
    X      LBN: XXXXX.                           !   OR THIS
    X      BYTE COUNT IN COMMAND: XXXXX.
    X      ACTUAL # BYTES TRANSFERRED: XXXXX.
    X      I/O BUFFER DESCRIPTOR: XXXXXX(O) XXXXXX(O)

39  G  I/O REQUEST FAILED
    B  XX:XX:XX  PLATTER XXX. - HOST BUFFER ACCESS ERROR
    X      CMD REF NUM: XXXXXX(O)
    X      SUB-CODE: XXXX(O)
    X      COMMAND: READ (OR)
                    READ-COMPARE (OR)
                    WRITE (OR)
                    WRITE-COMPARE (OR)
                    XXX(O)
    X      BAD BLOCK REPORTED: XXXXX.            ! EITHER THIS
    X      LBN: XXXXX.                           !   OR THIS
    X      BYTE COUNT IN COMMAND: XXXXX.
    X      ACTUAL # BYTES TRANSFERRED: XXXXX.
    X      I/O BUFFER DESCRIPTOR: XXXXXX(O) XXXXXX(O)

40  G  I/O REQUEST FAILED
    B  XX:XX:XX  PLATTER XXX. - CONTROLLER ERROR
    X      CMD REF NUM: XXXXXX(O)
    X      SUB-CODE: XXXX(O)
    X      COMMAND: READ (OR)
                    READ-COMPARE (OR)
                    WRITE (OR)
                    WRITE-COMPARE (OR)
                    XXX(O)
    X      BAD BLOCK REPORTED: XXXXX.            ! EITHER THIS
    X      LBN: XXXXX.                           !   OR THIS
    X      BYTE COUNT IN COMMAND: XXXXX.
    X      ACTUAL # BYTES TRANSFERRED: XXXXX.
    X      I/O BUFFER DESCRIPTOR: XXXXXX(O) XXXXXX(O)
```

```
41  G  I/O REQUEST FAILED
    B  XX:XX:XX  PLATTER XXX. - DRIVE ERROR
    X      CMD REF NUM: XXXXXX(O)
    X      SUB-CODE: XXXX(O)
    X      COMMAND: READ (OR)
                    READ-COMPARE (OR)
                    WRITE (OR)
                    WRITE-COMPARE (OR)
                    XXX(O)
    X      BAD BLOCK REPORTED: XXXXX.          ; EITHER THIS
    X      LBN: XXXXX.                         !   OR THIS
    X      BYTE COUNT IN COMMAND: XXXXX.
    X      ACTUAL # BYTES TRANSFERRED: XXXXX.
    X      I/O BUFFER DESCRIPTOR: XXXXXX(O) XXXXXX(O)
```

ERROR 42 IS DETECTED AT THE HOST. IT CAN ARISE IF THE USER HAS CHOSEN THE OPTION TO CHECK ALL WRITES AT THE HOST BY FOLLOWING EACH WITH A READ TO THE SAME DISK ADDRESS.

```
42  G  I/O REQUEST FAILED
    B  XX:XX:XX  PLATTER XXX. - HOST-DETECTED WRITE-COMPARE
                                ERROR
    X      LBN: XXXXX.
    X      BYTE COUNT IN COMMAND: XXXXX.
    X      ACTUAL # BYTES TRANSFERRED: XXXXX.
```

ERROR 43 DESCRIBES A SINGLE I/O COMMAND FOR WHICH NO MSCP END MESSAGE WAS EVER RECEIVED.

```
43  G  I/O REQUEST FAILED
    B  XX:XX:XX  PLATTER XXX. - COMMAND TIMEOUT
    X      CMD REF NUM: XXXXXX(O)
    X      COMMAND: READ (OR)
                    READ-COMPARE (OR)
                    WRITE (OR)
                    WRITE-COMPARE (OR)
                    XXX(O)
    X      LBN: XXXXX.
    X      BYTE COUNT IN COMMAND: XXXXX.
    X      I/O BUFFER DESCRIPTOR: XXXXXX(O) XXXXXX(O)
```

3.2.5 ERROR LOG MESSAGES

ERROR LOG MESSAGES (DATAGRAMS) DESCRIBE MINOR ERRORS ENCOUNTERED BY THE CONTROLLER. THEIR OCCAISIONAL OCCURRENCE IS NORMAL. RELEVANT FIELDS OF A DATAGRAM ARE PRINTED (A) IF THE USER IS RUNNING THE PROGRAM IN ATTENDED MODE, AND (B) IF NOT SUPPRESSED BY THE USER DURING THE SW DIALOG. ONE OF THESE FIELDS, THE COMMAND REFERENCE NUMBER, CAN LINK THE DATAGRAM WITH A HARD ERROR.

ERROR LOG MESSAGES ARE NOT INCLUDED IN THE CUMULATIVE ERROR COUNT WHICH APPEARS WITH THE END-OF-PASS MESSAGE. HOWEVER, IF THE MULTI-DRIVE SUBTEST IS BEING RUN, THEN DATAGRAMS WHICH APPLY TO A PARTICULAR UNIT (AS OPPOSED TO THE CONTROLLER) ARE TALLIED FOR THE STATISTICAL REPORT REGARDLESS OF WHETHER THEIR PRINTING HAS BEEN SUPPRESSED. THE FORMAT IS AS FOLLOWS:

```
ERROR LOG MESSAGE RECEIVED:
    CMD REF NUM: XXXXXX(O)
    PLATTER: XXX.                    ! FOR SMALL DISK ERRORS ONLY
    FORMAT: CONTROLLER ERROR (OR)
            HOST MEMORY ACCESS ERROR (OR)
            SMALL DISK ERROR (OR)
            XXX(O)
    EVENT CODE: SUCCESS (OR)
                INVALID COMMAND (OR)
                COMMAND ABORTED (OR)
                UNIT-OFFLINE (OR)
                UNIT-AVAILABLE (OR)
                MEDIA FORMAT ERROR (OR)
                WRITE-PROTECTED (OR)
                DEVICE COMPARE ERROR (OR)
                DATA ERROR (OR)
                HOST BUFFER ACCESS ERROR (OR)
                CONTROLLER ERROR (OR)
                DRIVE ERROR (OR)
                XXX(O)
    SUB-CODE: XXXX(O)
    HOST MEM ADDR: XXXXXX(O) XXXXXX(O) ! HOST MEM ACC ERR ONLY
```

4.0 PERFORMANCE AND PROGRESS REPORTS

AT THE END OF EACH PASS, THE PASS COUNT IS GIVEN ALONG WITH THE TOTAL NUMBER OF ERRORS REPORTED SINCE THE DIAGNOSTIC WAS STARTED. THIS ERROR COUNT DOES NOT INCLUDE ERROR LOG MESSAGES (SECTION 3.2.5), CONFIGURATION ERRORS (SECTION 3.2.2), NOR ERRORS DETECTED BY THE FRONT PANEL TEST UNDER THE DM EXERCISER. THE "EOP" SWITCH CAN BE USED TO CONTROL HOW OFTEN THE END-OF-PASS MESSAGE IS PRINTED. SECTION 2.2 DESCRIBES SWITCHES.

PRIOR TO THE END-OF-PASS MESSAGE, THE STATISTICAL SUMMARY REPORT IS AUTOMATICALLY PRINTED BY THE EXERCISER. ITS FORMAT DEPENDS ON WHICH SUBTEST (MULTI-DRIVE OR DM EXERCISER) IS BEING RUN. AN ASTERISK (*) APPEARING NEXT TO A NUMBER IN THE FORMAT BELOW REPRESENTS EITHER A SPACE OR THE LETTER "K". THE "K" FORMAT WILL BE USED IF THE STATISTIC IS GREATER THAN 65,000. ALL VALUES ARE IN DECIMAL RADIX.

4.1 MULTI-DRIVE SUBTEST REPORTS

ELAPSED TIME: XX:XX:XX

UNIT NO.	NO. READS (K=1000)	NO. WRITES (K=1000)	MBYTES READ	MBYTES WRITTEN	NO. HARD ERRORS	NO. ERROR LOGS
XX	XXXXX*	XXXXX*	XXXXX	XXXXX	XXXXX	XXXXX
XX	XXXXX*	XXXXX*	XXXXX	XXXXX	XXXXX	XXXXX
XX	XXXXX*	XXXXX*	XXXXX	XXXXX	XXXXX	XXXXX
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.

ECC ERRORS ARE REPORTED THROUGH ERROR LOG MESSAGES. THEY ARE TALLIED REGARDLESS OF WHETHER THE MESSAGE PRINTING HAS BEEN SUPPRESSED.

N SYMBOL ECC ERRORS:

UNIT NO.	ECC FIELD ONLY	N = 1	N = 2	N = 3	N = 4	N = 5	N = 6	N = 7	N = 8
XX	XXXXX	XXXXX	XXXXX	XXXXX	XXXXX	XXXXX	XXXXX	XXXXX	XXXXX
XX	XXXXX	XXXXX	XXXXX	XXXXX	XXXXX	XXXXX	XXXXX	XXXXX	XXXXX
XX	XXXXX	XXXXX	XXXXX	XXXXX	XXXXX	XXXXX	XXXXX	XXXXX	XXXXX
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.

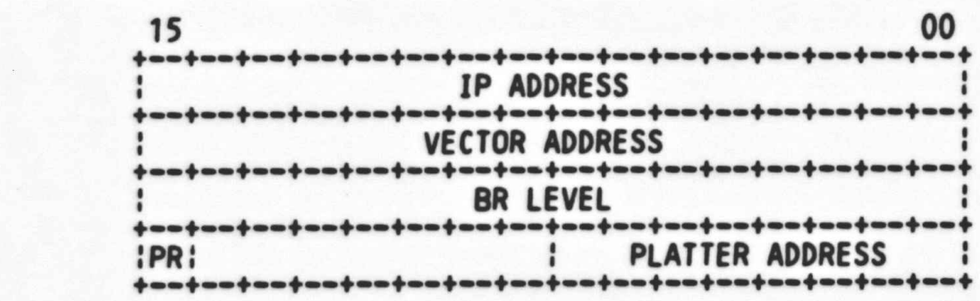
4.2 DM EXERCISER SUBTEST REPORT

ELAPSED TIME: XX:XX:XX

UNIT NO.	NO. READS (K=1000)	NO. WRITES (K=1000)	NO. SEEKS (K=1000)	MBYTES READ	MBYTES WRITTEN	NO. HARD ERRORS	NO. SOFT ERRORS
XX	XXXXX*	XXXXX*	XXXXX*	XXXXX	XXXXX	XXXXX	XXXXX
XX	XXXXX*	XXXXX*	XXXXX*	XXXXX	XXXXX	XXXXX	XXXXX
XX	XXXXX*	XXXXX*	XXXXX*	XXXXX	XXXXX	XXXXX	XXXXX
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.

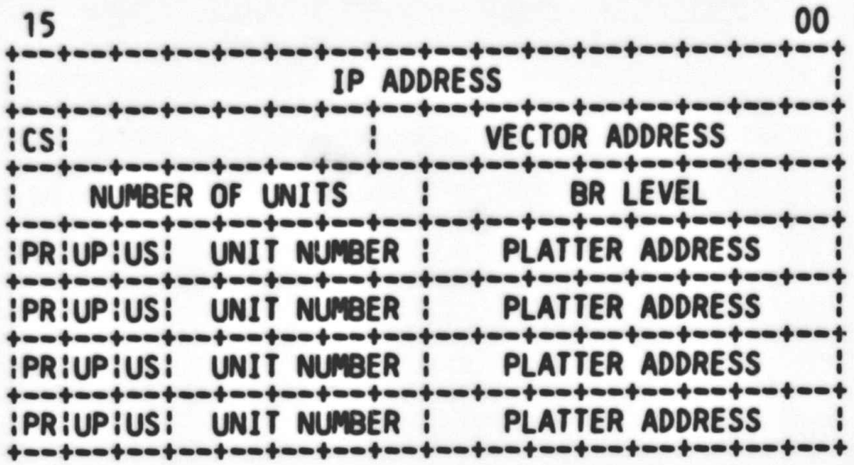
5.0 DEVICE INFORMATION TABLES

THE HARDWARE P-TABLE APPEARS AS FOLLOWS FOR EACH UNIT:



PR - UNIT PROTECTION BIT

DURING THE INITIALIZATION PROCESS AFTER A START, RESTART, OR CONTINUE COMMAND, THE RC25 DISK EXERCISER CREATES UP TO FOUR RUN-TIME TABLES (CALLED CONTROLLER STATUS TABLES, OR CST'S), AND FILLS THEM WITH INFORMATION DERIVED FROM THE HARDWARE P-TABLES OF ALL ACTIVE UNITS. THESE TABLES ARE NOT USER-ACCESSIBLE; THEIR FORMAT IS PRESENTED HERE ONLY TO SHOW THE VIEW OF THE WORLD AS SEEN BY THE EXERCISER. EACH CST IS A DYNAMIC 7-WORD BLOCK:



CS - CONTROLLER STATUS BIT (ONLINE / OFFLINE)
PR - UNIT PROTECTION BIT
UP - UNIT PRESENT BIT
US - UNIT STATUS BIT (ONLINE / OFFLINE)

6.0 SUBTEST SUMMARIES

THE RC25 DISK EXERCISER CONSISTS OF ONE TEST SUBDIVIDED INTO THREE SUBTESTS: INITIALIZATION, MULTI-DRIVE, AND DM EXERCISER. IN THE FIRST PASS FOLLOWING A START, RESTART, OR CONTINUE COMMAND, THE INITIALIZATION SUBTEST WILL ALWAYS PRECEDE THE EXECUTION OF ANY OTHER SUBTEST. SUBSEQUENT PASSES WILL NOT INCLUDE THE INITIALIZATION SUBTEST.

6.1 INITIALIZATION SUBTEST

THE PURPOSE OF THE INITIALIZATION SUBTEST IS TO VERIFY THE HARDWARE CONFIGURATION AS SPECIFIED BY THE OPERATOR, AND TO BRING EACH UNIT ONLINE. SPECIFICALLY, THE SUBTEST PERFORMS THE FOLLOWING ACTIONS FOR EACH CONTROLLER IN TURN:

1. A DEVICE REGISTER EXISTENCE TEST - TO VERIFY THAT THE CONTROLLER'S IP AND SA REGISTERS APPEAR ON THE UNIBUS,
2. A VECTOR INTERRUPT TEST - TO INSURE THAT THE DEVICE CAN INITIATE VECTORED INTERRUPTS TO THE PROCESSOR,
3. A BR LEVEL TEST - TO VERIFY THAT THE DEVICE INTERRUPT OCCURS AT THE BR LEVEL SPECIFIED BY THE OPERATOR,
4. A FOUR-STEP HARD INIT SEQUENCE, AS DEFINED IN THE UNIBUS / Q-BUS STORAGE SYSTEMS PORT SPECIFICATION,
5. THE SENDING OF THE "SET CONTROLLER CHARACTERISTICS" COMMAND (MSCP),
6. THE SENDING OF THE MSCP "ONLINE" COMMAND FOR EACH PLATTER SPINNING UNDER THE CONTROLLER,
7. THE SENDING OF ONE OR TWO MSCP "ACCESS" COMMANDS FOR EACH PLATTER UNDER THE CONTROLLER TO VERIFIY THAT EACH DISK CAN BE READ.

ANY CONTROLLER-FATAL ERRORS ENCOUNTERED DURING THE INITIALIZATION SUBTEST WILL CAUSE ALL ITS UNITS TO BE DROPPED FROM TESTING. ANY UNIT-FATAL ERRORS WILL CAUSE ONLY THE APPROPRIATE UNIT TO BE DROPPED. FOR ALL SURVIVING CONTROLLERS AND UNITS, THE PROGRAM WILL THEN BEGIN EXECUTION OF EITHER THE MULTI-DRIVE OR DM EXERCISER SUBTEST.

6.2 MULTI-DRIVE SUBTEST

THE PURPOSE OF THE MULTI-DRIVE SUBTEST IS TO EXERCISE THE DISK DRIVES IN A MANNER SIMILAR TO HEAVY USAGE IN AN OPERATING SYSTEM ENVIRONMENT. THE SUBTEST ISSUES MSCP I/O COMMANDS TO ALL UNITS FAST ENOUGH SO THAT EACH CONTROLLER HAS SEVERAL COMMANDS QUEUED AT ANY GIVEN TIME. THE I/O COMMAND PARAMETERS ARE GOVERNED BY USER RESPONSES DURING THE SOFTWARE QUESTIONING, EXCEPT THAT BYTE COUNTS ARE DETERMINED AT RANDOM. AS EACH I/O OPERATION COMPLETES, THE PROGRAM UPDATES THE STATISTICAL TALLY UNTIL THE SPECIFIED NUMBER OF MEGABYTES HAS BEEN TRANSFERRED TO / FROM EACH UNIT. WHEN THIS HAPPENS, END-OF-PASS IS DECLARED, AND THE SUMMARY REPORT IS PRINTED.

IF THE RC25 EXPERIENCES A READ / WRITE ERROR, IT WILL TRY A VARIABLE NUMBER OF TIMES TO CORRECT THE ERROR. IF ALL RETRIES FAIL, THEN A HARD ERROR WILL BE REPORTED TO THE HOST, AN ERROR MESSAGE WILL BE DISPLAYED ON THE CONSOLE TERMINAL, AND THE ERROR WILL BE TALLIED FOR THE SUMMARY REPORT. THE UNIT INVOLVED WILL BE DROPPED FROM TESTING IF ITS HARD ERROR COUNT HAS EXCEEDED THE USER-SPECIFIED LIMIT.

6.3 DM EXERCISER SUBTEST

THE DM EXERCISER CONSISTS OF THE FRONT PANEL TEST (ROM RESIDENT ON THE CONTROLLER BOARD) RUN IN A CONTINUOUS LOOP UNDER HOST SUPERVISION. IT IS USED TO EXERCISER A DISK BY (A) PERFORMING READS ON ALL TRACKS OF THE PLATTER, AND (B) FORCING WRITES TO THE DIAGNOSTIC TRACKS ONLY. ALL ERRORS DETECTED IN THE DIAGNOSTIC MACHINE, BOTH HARD AND SOFT, WILL BE REPORTED TO THE HOST IN STATISTICAL FORM FOR THE SUMMARY REPORT, BUT THEY WILL NOT APPEAR IN THE CUMULATIVE ERROR COUNT WITH THE END-OF-PASS MESSAGE. THE NUMBER OF SEEKS, READS, AND WRITES WILL ALSO BE REPORTED AND TALLIED. AS IN THE MULTI-DRIVE SUBTEST, END-OF-PASS WILL BE DECLARED WHEN ALL UNITS HAVE REACHED THE MEGABYTE TRANSFER LIMIT SPECIFIED BY THE USER. (NOTE THAT THIS SUBTEST TAKES MUCH LONGER TO ACCOMPLISH THE SAME FEAT; IT TAKES ABOUT 10 MINUTES PER UNIT TO REACH A 2 MEGABYTE TRANSFER LIMIT).

THE HOST INITIATES THE FRONT PANEL TEST (FPT) WITH A CROM PRIMER PROGRAM, WHICH IS DOWNLINE LOADED FROM THE HOST TO THE RC25 VIA THE DUP COMMAND "EXECUTE SUPPLIED PROGRAM". THE CROM PRIMER IS RESPONSIBLE FOR LOADING THE FRONT PANEL TEST INTO RC25 RAM SPACE, THEN PASSING CONTROL TO IT. IF THE "EXECUTE SUPPLIED PROGRAM" SUCCEEDS, THE HOST RESPONDS BY ISSUING THE DUP COMMAND "SEND DATA". THIS COMMAND TELLS THE FRONT PANEL TEST THE LOCATION OF THE HOST / FPT COMMUNICATION AREA, A BLOCK OF HOST MEMORY RESERVED FOR MUTUAL ACCESS.

THE FPT EXERCISES ONE PLATTER AT A TIME PER PASS, AND EACH FPT PASS TAKES APPROXIMATELY 5 MINUTES. AT THE COMPLETION OF EACH FPT PASS, A NEW SET OF STATISTICS IS LOADED INTO THE HOST COMMUNICATION AREA, AND THE FPT GOES ON TO EXERCISE THE NEXT UNIT. IF THE HOST FAILS TO DETECT A NEW SET OF STATS WITHIN A TIMED INTERVAL, THEN A DEVICE-FATAL ERROR WILL BE DECLARED, AND ALL UNITS ASSOCIATED WITH THE CONTROLLER WILL BE DROPPED. OTHERWISE, THE UNIT NUMBER AND THE NUMBER OF BLOCKS TRANSFERRED DURING THE FPT PASS WILL BE DISPLAYED TO THE USER (IF IN ATTENDED MODE).

APPENDIX A - DATA PATTERNS

THERE ARE 21 PRE-DEFINED DATA PATTERNS SUPPLIED WITH THE MULTI-DRIVE SUBTEST. THROUGH THE SOFTWARE QUESTIONS, YOU MAY CHOSE ANY ONE PATTERN TO BE USED FOR ALL WRITE OPERATIONS, OR YOU MAY HAVE ALL PATTERNS USED SEQUENTIALLY. THE "LBN" APPEARING IN PATTERNS 17 - 21 REPRESENTS THE STARTING BLOCK NUMBER OF THE CURRENT WRITE REQUEST.

	HEX	OCTAL	BINARY
PATTERN 1	R A N D O M	N U M B E R S	
PATTERN 2	0000	000000	0 000 000 000 000 000
PATTERN 3	FFFF	177777	1 111 111 111 111 111
PATTERN 4	8B8B	105613	1 000 101 110 001 011
PATTERN 5	3333	031463	0 011 001 100 110 011
PATTERN 6	3091	030221	0 011 000 010 010 001
PATTERN 7	0001	000001	0 000 000 000 000 001
	0003	000003	0 000 000 000 000 011
	0007	000007	0 000 000 000 000 111
	000F	000017	0 000 000 000 001 111
	001F	000037	0 000 000 000 011 111
	003F	000077	0 000 000 000 111 111
	007F	000177	0 000 000 001 111 111
	00FF	000377	0 000 000 011 111 111
	01FF	000777	0 000 000 111 111 111
	03FF	001777	0 000 001 111 111 111
	07FF	003777	0 000 011 111 111 111
	0FFF	007777	0 000 111 111 111 111
	1FFF	017777	0 001 111 111 111 111
	3FFF	037777	0 011 111 111 111 111
	7FFF	077777	0 111 111 111 111 111
	FFFF	177777	1 111 111 111 111 111
PATTERN 8	FFFE	177776	1 111 111 111 111 110
	FFFC	177774	1 111 111 111 111 100
	FFF8	177770	1 111 111 111 111 000
	FFF0	177760	1 111 111 111 110 000
	FFE0	177740	1 111 111 111 100 000
	FFC0	177700	1 111 111 111 000 000
	FF80	177600	1 111 111 110 000 000
	FF00	177400	1 111 111 100 000 000
	FE00	177000	1 111 111 000 000 000
	FC00	176000	1 111 110 000 000 000
	F800	174000	1 111 100 000 000 000
	F000	170000	1 111 000 000 000 000
	E000	160000	1 110 000 000 000 000
	C000	140000	1 100 000 000 000 000
	8000	100000	1 000 000 000 000 000
	0000	000000	0 000 000 000 000 000

PATTERN 9	0000	000000	0 000 000 000 000 000
	0000	000000	0 000 000 000 000 000
	0000	000000	0 000 000 000 000 000
	FFFF	177777	1 111 111 111 111 111
	FFFF	177777	1 111 111 111 111 111
	FFFF	177777	1 111 111 111 111 111
	0000	000000	0 000 000 000 000 000
	0000	000000	0 000 000 000 000 000
	FFFF	177777	1 111 111 111 111 111
	FFFF	177777	1 111 111 111 111 111
	0000	000000	0 000 000 000 000 000
	FFFF	177777	1 111 111 111 111 111
	0000	000000	0 000 000 000 000 000
	FFFF	177777	1 111 111 111 111 111
	0000	000000	0 000 000 000 000 000
	FFFF	177777	1 111 111 111 111 111
PATTERN 10	B6D9	133331	1 011 011 011 011 001
PATTERN 11	5555	052525	0 101 010 101 010 101
	5555	052525	0 101 010 101 010 101
	5555	052525	0 101 010 101 010 101
	AAAA	125252	1 010 101 010 101 010
	AAAA	125252	1 010 101 010 101 010
	AAAA	125252	1 010 101 010 101 010
	5555	052525	0 101 010 101 010 101
	5555	052525	0 101 010 101 010 101
	AAAA	125252	1 010 101 010 101 010
	AAAA	125252	1 010 101 010 101 010
	5555	052525	0 101 010 101 010 101
	AAAA	125252	1 010 101 010 101 010
	5555	052525	0 101 010 101 010 101
	AAAA	125252	1 010 101 010 101 010
	5555	052525	0 101 010 101 010 101
	AAAA	125252	1 010 101 010 101 010
PATTERN 12	2D2D	026455	0 010 110 100 101 101
	2D2D	026455	0 010 110 100 101 101
	2D2D	026455	0 010 110 100 101 101
	D2D2	151322	1 101 001 011 010 010
	D2D2	151322	1 101 001 011 010 010
	D2D2	151322	1 101 001 011 010 010
	2D2D	026455	0 010 110 100 101 101
	2D2D	026455	0 010 110 100 101 101
	D2D2	151322	1 101 001 011 010 010
	D2D2	151322	1 101 001 011 010 010
	2D2D	026455	0 010 110 100 101 101
	2D2D	026455	0 010 110 100 101 101
	D2D2	151322	1 101 001 011 010 010
	2D2D	026455	0 010 110 100 101 101
	D2D2	151322	1 101 001 011 010 010
	2D2D	026455	0 010 110 100 101 101
	D2D2	151322	1 101 001 011 010 010
	2D2D	026455	0 010 110 100 101 101
	D2D2	151322	1 101 001 011 010 010
	2D2D	026455	0 010 110 100 101 101

PATTERN 13	6DB6	066666	0 110 110 110 110 110
PATTERN 14	0001	000001	0 000 000 000 000 001
	0002	000002	0 000 000 000 000 010
	0004	000004	0 000 000 000 000 100
	0008	000010	0 000 000 000 001 000
	0010	000020	0 000 000 000 010 000
	0020	000040	0 000 000 000 100 000
	0040	000100	0 000 000 001 000 000
	0080	000200	0 000 000 010 000 000
	0100	000400	0 000 000 100 000 000
	0200	001000	0 000 001 000 000 000
	0400	002000	0 000 010 000 000 000
	0800	004000	0 000 100 000 000 000
	1000	010000	0 001 000 000 000 000
	2000	020000	0 010 000 000 000 000
	4000	040000	0 100 000 000 000 000
	8000	100000	1 000 000 000 000 000
PATTERN 15	FFFE	177776	1 111 111 111 111 110
	FFFD	177775	1 111 111 111 111 101
	FFFB	177773	1 111 111 111 111 011
	FFF7	177767	1 111 111 111 110 111
	FFEF	177757	1 111 111 111 101 111
	FFDF	177737	1 111 111 111 011 111
	FFBF	177677	1 111 111 110 111 111
	FF7F	177577	1 111 111 101 111 111
	FEFF	177377	1 111 111 011 111 111
	FDFF	176777	1 111 110 111 111 111
	FBFF	175777	1 111 101 111 111 111
	F7FF	173777	1 111 011 111 111 111
	EFFF	167777	1 110 111 111 111 111
	DFFF	157777	1 101 111 111 111 111
	BFFF	137777	1 011 111 111 111 111
	7FFF	077777	0 111 111 111 111 111
PATTERN 16	B6D9	133331	1 011 011 011 011 001
	B6D9	133331	1 011 011 011 011 001
	B6D9	133331	1 011 011 011 011 001
	DB6C	155554	1 101 101 101 101 100
	DB6C	155554	1 101 101 101 101 100
	DB6C	155554	1 101 101 101 101 100
	B6D9	133331	1 011 011 011 011 001
	B6D9	133331	1 011 011 011 011 001
	DB6C	155554	1 101 101 101 101 100
	DB6C	155554	1 101 101 101 101 100
	B6D9	133331	1 011 011 011 011 001
	DB6C	155554	1 101 101 101 101 100
	B6D9	133331	1 011 011 011 011 001
	DB6C	155554	1 101 101 101 101 100
	B6D9	133331	1 011 011 011 011 001
	DB6C	155554	1 101 101 101 101 100

PATTERN 17	LBN	LBN			LBN			
	8D36	106466	1	000	110	100	110	110
	8D36	106466	1	000	110	100	110	110
	72C9	071311	0	111	001	011	001	001
	72C9	071311	0	111	001	011	001	001
	72C9	071311	0	111	001	011	001	001
	8D36	106466	1	000	110	100	110	110
	8D36	106466	1	000	110	100	110	110
	8D36	106466	1	000	110	100	110	110
	8D36	106466	1	000	110	100	110	110
	72C9	071311	0	111	001	011	001	001
	72C9	071311	0	111	001	011	001	001
	72C9	071311	0	111	001	011	001	001
	72C9	071311	0	111	001	011	001	001
	72C9	071311	0	111	001	011	001	001
	8D36	106466	1	000	110	100	110	110
	8D36	106466	1	000	110	100	110	110
	8D36	106466	1	000	110	100	110	110
	8D36	106466	1	000	110	100	110	110
	8D36	106466	1	000	110	100	110	110
	8D36	106466	1	000	110	100	110	110
PATTERN 18	8D36	106466	1	000	110	100	110	110
	LBN	LBN				LBN		
	72C9	071311	0	111	001	011	001	001
	8D36	106466	1	000	110	100	110	110
	8D36	106466	1	000	110	100	110	110
	8D36	106466	1	000	110	100	110	110
	72C9	071311	0	111	001	011	001	001
	72C9	071311	0	111	001	011	001	001
	72C9	071311	0	111	001	011	001	001
	72C9	071311	0	111	001	011	001	001
	8D36	106466	1	000	110	100	110	110
	8D36	106466	1	000	110	100	110	110
	8D36	106466	1	000	110	100	110	110
	8D36	106466	1	000	110	100	110	110
	8D36	106466	1	000	110	100	110	110
	72C9	071311	0	111	001	011	001	001
	72C9	071311	0	111	001	011	001	001
	72C9	071311	0	111	001	011	001	001
	72C9	071311	0	111	001	011	001	001
	72C9	071311	0	111	001	011	001	001
	72C9	071311	0	111	001	011	001	001
PATTERN 19	LBN	LBN				LBN		
	B999	134631	1	011	100	110	011	001
	B999	134631	1	011	100	110	011	001
	4666	043146	0	100	011	001	100	110
	4666	043146	0	100	011	001	100	110
	4666	043146	0	100	011	001	100	110
	B999	134631	1	011	100	110	011	001
	B999	134631	1	011	100	110	011	001
	B999	134631	1	011	100	110	011	001
	B999	134631	1	011	100	110	011	001
	4666	043146	0	100	011	001	100	110
	4666	043146	0	100	011	001	100	110
	4666	043146	0	100	011	001	100	110
	4666	043146	0	100	011	001	100	110

	4666	043146	0 100 011 001 100 110
	B999	134631	1 011 100 110 011 001
	B999	134631	1 011 100 110 011 001
	B999	134631	1 011 100 110 011 001
	B999	134631	1 011 100 110 011 001
	B999	134631	1 011 100 110 011 001
	B999	134631	1 011 100 110 011 001
PATTERN 20	B999	134631	1 011 100 110 011 001
	LBN	LBN	LBN
	4666	043146	0 100 011 001 100 110
	B999	134631	1 011 100 110 011 001
	B999	134631	1 011 100 110 011 001
	B999	134631	1 011 100 110 011 001
	4666	043146	0 100 011 001 100 110
	4666	043146	0 100 011 001 100 110
	4666	043146	0 100 011 001 100 110
	4666	043146	0 100 011 001 100 110
	B999	134631	1 011 100 110 011 001
	B999	134631	1 011 100 110 011 001
	B999	134631	1 011 100 110 011 001
	B999	134631	1 011 100 110 011 001
	B999	134631	1 011 100 110 011 001
	4666	043146	0 100 011 001 100 110
	4666	043146	0 100 011 001 100 110
	4666	043146	0 100 011 001 100 110
	4666	043146	0 100 011 001 100 110
	4666	043146	0 100 011 001 100 110
	4666	043146	0 100 011 001 100 110
PATTERN 21	LBN	LBN	LBN

APPENDIX B - GLOSSARY

AZTEC - PROJECT NAME FOR THE RC25

MASS STORAGE COMMUNICATION PROTOCOL (MSCP) - THE METHOD OF COMMUNICATION USED BETWEEN THE HOST AND RC25 CONTROLLER.

DIAGNOSTIC MACHINE (DM) - AN INTERPRETER BUILT INTO THE RC25 FIRMWARE THAT IS USED TO EXECUTE RESIDENT OR DOWNLINE LOADED PROGRAMS.

INITIALIZATION SEQUENCE - THE SERIES OF COMMANDS EXECUTED TO BRING AN RC25 ONLINE TO THE HOST. THIS SEQUENCE IS A STRICTLY-DEFINED SET OF FOUR STEPS WHICH INFORMS THE DEVICE OF SUCH THINGS AS RING LOCATION AND SIZE.

UNIT - A UNIT AS SEEN BY THE DIAGNOSTIC SUPERVISOR; THAT IS, A SINGLE RC25 DISK PLATTER.

DRIVE-UNIT - A SINGLE RC25 DRIVE WITH A FIXED AND A REMOVABLE PLATTER ON ONE SPINDLE.

UNIT NUMBER - A NUMBER BETWEEN 0 AND 15 INCLUSIVE WHICH IDENTIFIES A PLATTER TO THE DIAGNOSTIC SUPERVISOR. UNIT NUMBERS ARE ASSIGNED SEQUENTIALLY FROM 0 BY THE DIAGNOSTIC SUPERVISOR DURING THE HARDWARE CONFIGURATION QUESTIONS. THERE IS NO RELATIONSHIP BETWEEN A DISK'S UNIT NUMBER AND ITS PLATTER ADDRESS.

PLATTER ADDRESS - A NUMBER BETWEEN 0 AND 253 THAT UNIQUELY IDENTIFIES ONE RC25 PLATTER. THIS NUMBER IS ASSIGNED THROUGH THE UNIT PLUG ON THE FRONT PANEL OF AN RC25. EVEN PLATTER ADDRESSES ALWAYS DESIGNATE THE REMOVABLE DISK AND ODD PLATTER ADDRESSES ALWAYS DESIGNATE THE FIXED DISK.

CUSTOMER DATA AREA (ALSO HOST APPLICATION AREA) - BLOCKS RESERVED ON AN RC25 PLATTER FOR FREE USE BY THE CUSTOMER. THIS AREA EXCLUDES THE DIAGNOSTIC TRACKS (DBN), THE FCT, THE RCT, AND RBN'S.

PROTECTED CUSTOMER DATA AREA - AN RC25 PLATTER WHOSE CUSTOMER TRACKS ARE WRITE-PROTECTED BY THE USER THROUGH THE HARDWARE CONFIGURATION SECTION.

LBN (LOGICAL BLOCK NUMBER) - GIVEN THAT N IS THE NUMBER OF BLOCKS IN THE HOST APPLICATION AREA, LBN IS A NUMBER BETWEEN 0 AND (N - 1) INCLUSIVE WHICH IDENTIFIES A SINGLE SUCH BLOCK.

)%

```
CZRCD.EXE   Memory allocation map  TKB M40.02      Page 1
                   11-JUL-83   08:48



Partition name : DUMMY
Identification : V01.0
Task  UIC      : [305,45]
Task attributes: -HD
Total address windows: 1.
Task  image  size  : 14720. words
Task address limits: 002000 073327
R-W disk blk limits: 000002 000073 000072 00058.

*** Root segment: CZRCD1


R/W mem  limits: 002000 073327 071330 29400.
Disk blk limits: 000002 000073 000072 00058.


Memory allocation synopsis:

Section                                          Title  Ident  File
-------

. BLK.:(RW,I,LCL,REL,CON)      002000 000000 00000.
$CODE$:(RO,I,LCL,REL,CON)      002000 040446 16678.
                               002000 006550 03432.  CZRCD1 V01.0  CZRCD1.OBJ;1
                               010550 011604 04996.  CZRCD2 V01.0  CZRCD2.OBJ;1
                               022354 017412 07946.  CZRCD3 V01.0  CZRCD3.OBJ;1
                               041766 000034 00028.  B16ABS 2.4    AZLIB.OLB;1
                               042022 000316 00206.  B16MUL 2.8    AZLIB.OLB;1
                               042340 000106 00070.  B16SAV 2.4    AZLIB.OLB;1
$FFF$ :(RO,I,LCL,REL,CON)      042446 023714 10188.
                               042446 023714 10188.  CZRCD1 V01.0  CZRCD1.OBJ;1
$GGG$ :(RO,I,LCL,REL,CON)      066362 002616 01422.
                               066362 002616 01422.  CZRCD3 V01.0  CZRCD3.OBJ;1
$PLIT$:(RO,D,LCL,REL,CON)      071200 002066 01078.
                               071200 001716 00974.  CZRCD2 V01.0  CZRCD2.OBJ;1
                               073116 000150 00104.  CZRCD3 V01.0  CZRCD3.OBJ;1
$XYZ$ :(RO,I,LCL,REL,CON)      073266 000040 00032.
                               073266 000040 00032.  CZRCD4 V01.0  CZRCD4.OBJ;1


Global symbols:

ADR     000020    BIT10   002000    BIT9    001000    CER.01  004140-R  CST     042776-R  DUM.IE  004450-R  EBD.18  006160-R
BIT0    000001    BIT11   004000    BL$ABS  042010-R  CER.02  004224-R  CST.AD  043066-R  DUM.UC  004376-R  EBD.19  006224-R
BIT00   000001    BIT12   010000    BL$DIV  042246-R  CER.03  004270-R  CTLR.C  066266-R  DUM.UE  004526-R  EBD.20  006264-R
BIT01   000002    BIT13   020000    BL$LAS  073266-R  CLK.CS  066330-R  CUOFF   066264-R  DUM.00  004344-R  EBD.21  006350-R
BIT02   000004    BIT14   040000    BL$MOD  042260-R  CLK.HE  066326-R  DCT     043070-R  DUR     066270-R  EBD.22  006436-R
BIT03   000010    BIT15   100000    BL$MUL  042022-R  CLK.IN  015020-R  DCT.AD  043200-R  D$PCNT  002122-R  EBH.31  006556-R
BIT04   000020    BIT2    000004    BL$SGN  041766-R  CLK.TY  066324-R  DFPTBL  010450-R  EBD.10  005400-R  EBH.32  006602-R
BIT05   000040    BIT3    000010    BL$SHF  042272-R  CLK.VE  066332-R  DMC.AD  043526-R  EBD.12  005444-R  EBH.34  006650-R
BIT06   000100    BIT4    000020    BOE     000400    COPY.B  015122-R  DM.COM  043206-R  EBD.13  005526-R  EBH.35  006674-R
BIT07   000200    BIT5    000040    BUFF.D  065506-R  CPLAT   066262-R  DROP.C  016324-R  EBD.14  005604-R  EBH.36  006724-R
BIT08   000400    BIT6    000100    BUFF.O  066106-R  CPT     042756-R  DRV.CT  016430-R  EBD.15  005676-R  EBH.37  006750-R
BIT09   001000    BIT7    000200    BUFF.S  066320-R  CRLF    010442-R  DUM.CE  004420-R  EBD.16  005766-R  EBH.38  007002-R
BIT1    000002    BIT8    000400    CCTLR   066260-R  CRN     066360-R  DUM.HE  004470-R  EBD.17  006062-R  EBH.39  007022-R
```

CZRCD.EXE Memory allocation map TKB M40.02 Page 2
CZRCD1 11-JUL-83 08:48

```
EBH.40 007060-R  EOP.FL 066254-R  GP$17  011044-R  L$CCP  002106-R  L$REV  002010-R  PRI04  000200  SWP.DP 010475-R
EBH.41 007106-R  ERRBLK 002134-R  GP$18  011054-R  L$CLEA 014314-R  L$RPT  012236-R  PRI05  000240  SWP.ER 010464-R
EBH.42 007126-R  ERRMSG 002132-R  GP$19  011064-R  L$CO   002032-R  L$SFTL 010714-R  PRI06  000300  SWP.ET 010472-R
EBH.43 007174-R  ERRNBR 002130-R  GP$2   010636-R  L$DEPO 002011-R  L$SOFT 010716-R  PRI07  000340  SWP.FL 010474-R
EBS.01 005332-R  ERRTYP 002126-R  GP$20  011102-R  L$DESC 010576-R  L$SPC  002056-R  PUTA.B 016136-R  SWP.ST 010470-R
EF.CON 000036    ETIME  010372-R  GP$21  011114-R  L$DESP 002076-R  L$SPCP 002020-R  PUTA.E 015600-R  SWP.UC 010476-R
EF.NEW 000035    EVL    000004    GP$3   010646-R  L$DEVP 002060-R  L$SPTP 002024-R  PUT.EN 015564-R  SWP.UD 010500-R
EF.PWR 000034    EX.BB  007654-R  GP$4   010660-R  L$DISP 002124-R  L$STA  002030-R  PUT.IO 016052-R  SWP.XF 010466-R
EF.RES 000037    EX.BC  010012-R  GP$5   010672-R  L$DLY  002116-R  L$SW   010464-R  PUT.RE 015746-R  SWQ1   002414-R
EF.STA 000040    EX.BD  010070-R  GP$6   010702-R  L$DTP  002040-R  L$SWLE 010462-R  QIO    066310-R  SWQ10  003024-R
EGD.10 004654-R  EX.CBC 007744-R  GP$7   010716-R  L$DTYP 002034-R  L$TEST 002114-R  RCINT0 037350-R  SWQ11  003100-R
EGD.11 004714-R  EX.CMD 007474-R  GP$8   010730-R  L$DU   014710-R  L$TIML 002014-R  RCINT1 037366-R  SWQ12  003144-R
EGD.12 004740-R  EX.CMP 007636-R  GP$9   010742-R  L$DUT  002072-R  L$UNIT 002012-R  RCINT2 037406-R  SWQ13  003176-R
EGD.13 004766-R  EX.CRN 007304-R  HARD.E 016544-R  L$DVTY 010550-R  MEM.MG 066255-R  RCINT3 037426-R  SWQ14  003274-R
EGD.14 005014-R  EX.DSC 007400-R  HDE    100000    L$EF   002052-R  MEM.SI 066314-R  RC25.A 043202-R  SWQ15  003352-R
EGD.15 005052-R  EX.EL  010150-R  HOURS  066334-R  L$ENVI 002044-R  MINUTE 066336-R  RETPKT 062442-R  SWQ2   002452-R
EGD.16 005104-R  EX.ESP 007546-R  HWPT.B 010454-R  L$ERRT 002126-R  MSCP.E 045502-R  RPS.X1 043570-R  SWQ3   002530-R
EGD.17 005122-R  EX.EVC 010262-R  HWPT.I 010450-R  L$ETP  002102-R  MSG.01 003466-R  RPS.X2 043572-R  SWQ4   002576-R
EGD.18 005152-R  EX.FMT 010242-R  HWPT.P 010456-R  L$EXP1 002046-R  MSG.02 003520-R  RP.ADD 065504-R  SWQ5   002656-R
EGD.19 005170-R  EX.HMA 010306-R  HWPT.V 010452-R  L$EXP4 002064-R  MSG.03 003616-R  RP.IND 065502-R  SWQ6   002700-R
EGD.20 005210-R  EX.LBN 007720-R  HWQ1   002136-R  L$EXP5 002066-R  MSG.04 003636-R  RP.SAV 043530-R  SWQ7   002720-R
EGD.21 005234-R  EX.ONL 007534-R  HWQ2   002152-R  L$HARD 010626-R  MSG.05 003670-R  RP.USE 065442-R  SWQ8   002736-R
EGD.22 005260-R  EX.O3  010356-R  HWQ3   002162-R  L$HIME 002120-R  MSG.06 003730-R  SA.REG 066354-R  SWQ9   003010-R
EGH.30 005306-R  EX.PA  010212-R  HWQ4   002174-R  L$HPCP 002016-R  MSG.07 003772-R  SB.COD 066346-R  TALLY  044100-R
EGS.01 004556-R  EX.RD  007616-R  HWQ5   002230-R  L$HPTP 002022-R  MSG.08 004044-R  SECOND 066340-R  TICKS  066342-R
EGS.02 004576-R  EX.SA  007256-R  HWQ6   002314-R  L$HRDL 010624-R  NEX    066356-R  SEND   017170-R  T$FREE 073322-R
EMS.01 020746-R  EX.SB  007440-R  IBE    010000    L$HW   010450-R  NEX.TR 015010-R  SET.CP 015162-R  T$PTHV 000002
EMS.10 021010-R  EX.SC  007342-R  IDU    000040    L$HWLE 010446-R  NULL   003464-R  SET.UP 015356-R  T.ADDR 045500-R
EMS.12 021056-R  EX.SCC 007514-R  IER    020000    L$ICP  002104-R  NUM.BU 066322-R  SFPTBL 010464-R  T.FLAG 066253-R
EMS.13 021120-R  EX.SND 007602-R  IIP.FL 066256-R  L$INIT 014072-R  OCL.X1 044074-R  STC.00 006542-R  T1     022460-R
EMS.14 021172-R  EX.WRT 007626-R  IN.IOD 016236-R  L$LADP 002026-R  OCL.X2 044076-R  STC.01 006556-R  UAM    000200
EMS.15 021240-R  FREE.M 066316-R  IODQ   066206-R  L$LAST 073272-R  OF.RC  066352-R  STC.02 006602-R  UPD.IO 016634-R
EMS.16 021306-R  GET.EN 015454-R  IODQ.I 066246-R  L$LOAD 002100-R  OUTC.L 043574-R  STC.03 006626-R  WAIT   017434-R
EMS.17 021360-R  GET.IO 015762-R  IODQ.O 066250-R  L$LUN  002074-R  OUTC.T 043674-R  STC.04 006650-R  XFR.CH 016776-R
EMS.18 021422-R  GET.RE 015640-R  IRC25. 043204-R  L$MREV 002050-R  OUT.IO 016200-R  STC.05 006674-R  $END.L 073324-R
EMS.19 021474-R  GP$DIS 011022-R  ISR    000100    L$NAME 002000-R  OVF.CH 016722-R  STC.06 006724-R  $SAVE2 042340-R
EMS.20 021542-R  GP$1   010626-R  IXE    004000    L$NDHR 010712-R  PATCH  042446-R  STC.07 006750-R  $SAVE3 042354-R
EMS.21 021604-R  GP$10  010750-R  LOE    040000    L$NDHW 010460-R  PLATT  010416-R  STC.08 007002-R  $SAVE4 042372-R
EMS.22 021672-R  GP$11  010760-R  LOT    000010    L$NDSF 011134-R  PNT    001000    STC.09 007022-R  $SAVE5 042412-R
EMS.30 021734-R  GP$12  010766-R  L$ACP  002110-R  L$NDSW 010540-R  PRI    002000    STC.10 007060-R
EMS.42 022104-R  GP$13  011000-R  L$APT  002036-R  L$PRIO 002042-R  PRI00  000000    STC.11 007106-R
EMS.43 022172-R  GP$14  011014-R  L$AU   015000-R  L$PROT 010542-R  PRI01  000040    STEP   066350-R
ENTRY. 066252-R  GP$15  011026-R  L$AUT  002070-R  L$PRT  002112-R  PRI02  000100    ST.COD 066344-R
ENV.US 062302-R  GP$16  011034-R  L$AUTO 014104-R  L$REPP 002062-R  PRI03  000140    SWM1   003370-R


*** Task builder statistics:

    Total work file references: 73128.
    Work  file  reads: 0.
    Work  file writes: 0.
    Size of core pool: 5486. words (21. pages)
    Size of work file: 3840. words (15. pages)
    Elapsed time:00:00:23
```

CZRCD CREATED BY TKB ON 11-JUL-83 AT 08:48 PAGE 1

GLOBAL CROSS REFERENCE CREF V01

SYMBOL	VALUE	REFERENCES...		
ADR	000020	# CZRCD1	# CZRCD2	# CZRCD3
BIT0	000001	# CZRCD1	# CZRCD2	# CZRCD3
BIT00	000001	# CZRCD1	# CZRCD2	# CZRCD3
BIT01	000002	# CZRCD1	# CZRCD2	# CZRCD3
BIT02	000004	# CZRCD1	# CZRCD2	# CZRCD3
BIT03	000010	# CZRCD1	# CZRCD2	# CZRCD3
BIT04	000020	# CZRCD1	# CZRCD2	# CZRCD3
BIT05	000040	# CZRCD1	# CZRCD2	# CZRCD3
BIT06	000100	# CZRCD1	# CZRCD2	# CZRCD3
BIT07	000200	# CZRCD1	# CZRCD2	# CZRCD3
BIT08	000400	# CZRCD1	# CZRCD2	# CZRCD3
BIT09	001000	# CZRCD1	# CZRCD2	# CZRCD3
BIT1	000002	# CZRCD1	# CZRCD2	# CZRCD3
BIT10	002000	# CZRCD1	# CZRCD2	# CZRCD3
BIT11	004000	# CZRCD1	# CZRCD2	# CZRCD3
BIT12	010000	# CZRCD1	# CZRCD2	# CZRCD3
BIT13	020000	# CZRCD1	# CZRCD2	# CZRCD3
BIT14	040000	# CZRCD1	# CZRCD2	# CZRCD3
BIT15	100000	# CZRCD1	# CZRCD2	# CZRCD3
BIT2	000004	# CZRCD1	# CZRCD2	# CZRCD3
BIT3	000010	# CZRCD1	# CZRCD2	# CZRCD3
BIT4	000020	# CZRCD1	# CZRCD2	# CZRCD3
BIT5	000040	# CZRCD1	# CZRCD2	# CZRCD3
BIT6	000100	# CZRCD1	# CZRCD2	# CZRCD3
BIT7	000200	# CZRCD1	# CZRCD2	# CZRCD3
BIT8	000400	# CZRCD1	# CZRCD2	# CZRCD3
BIT9	001000	# CZRCD1	# CZRCD2	# CZRCD3
BL$ABS	042010-R	# B16ABS	CZRCD3	
BL$DIV	042246-R	# B16MUL	CZRCD3	
BL$LAS	073266-R	# CZRCD4		
BL$MOD	042260-R	# B16MUL	CZRCD3	
BL$MUL	042022-R	# B16MUL	CZRCD2	CZRCD3
BL$SGN	041766-R	# B16ABS		
BL$SHF	042272-R	# B16MUL	CZRCD3	
BOE	000400	# CZRCD1	# CZRCD2	# CZRCD3
BUFF.D	065506-R	# CZRCD1	CZRCD2	CZRCD3
BUFF.O	066106-R	# CZRCD1	CZRCD2	CZRCD3
BUFF.S	066320-R	# CZRCD1	CZRCD2	CZRCD3
CCTLR	066260-R	# CZRCD1	CZRCD2	CZRCD3
CER.01	004140-R	# CZRCD1	CZRCD2	
CER.02	004224-R	# CZRCD1	CZRCD2	
CER.03	004270-R	# CZRCD1	CZRCD2	
CLK.CS	066330-R	# CZRCD1	CZRCD2	CZRCD3
CLK.HE	066326-R	# CZRCD1	CZRCD2	CZRCD3
CLK.IN	015020-R	# CZRCD2		
CLK.TY	066324-R	# CZRCD1	CZRCD2	CZRCD3
CLK.VE	066332-R	# CZRCD1	CZRCD2	CZRCD3
COPY.B	015122-R	# CZRCD2	CZRCD3	
CPLAT	066262-R	# CZRCD1	CZRCD2	CZRCD3
CPT	042756-R	# CZRCD1	CZRCD2	CZRCD3
CRLF	010442-R	# CZRCD1	CZRCD2	
CRN	066360-R	# CZRCD1	CZRCD2	CZRCD3

SYMBOL	VALUE	REFERENCES...		
CST	042776-R	# CZRCD1	CZRCD2	CZRCD3
CST.AD	043066-R	# CZRCD1	CZRCD2	CZRCD3
CTLR.C	066266-R	# CZRCD1	CZRCD2	CZRCD3
CUOFF	066264-R	# CZRCD1	CZRCD2	CZRCD3
DCT	043070-R	# CZRCD1	CZRCD2	CZRCD3
DCT.AD	043200-R	# CZRCD1	CZRCD2	CZRCD3
DFPTBL	010450-R	# CZRCD1		
DMC.AD	043526-R	# CZRCD1	CZRCD2	CZRCD3
DM.COM	043206-R	# CZRCD1	CZRCD2	CZRCD3
DROP.C	016324-R	# CZRCD2	CZRCD3	
DRV.CT	016430-R	# CZRCD2	CZRCD3	
DUM.CE	004420-R	# CZRCD1	CZRCD2	
DUM.HE	004470-R	# CZRCD1	CZRCD2	
DUM.IE	004450-R	# CZRCD1	CZRCD2	
DUM.UC	004376-R	# CZRCD1	CZRCD2	
DUM.UE	004526-R	# CZRCD1	CZRCD2	
DUM.00	004344-R	# CZRCD1	CZRCD2	
DUR	066270-R	# CZRCD1	CZRCD2	CZRCD3
D$PCNT	002122-R	# CZRCD1		
EBD.10	005400-R	# CZRCD1	CZRCD2	
EBD.12	005444-R	# CZRCD1	CZRCD2	
EBD.13	005526-R	# CZRCD1	CZRCD2	
EBD.14	005604-R	# CZRCD1	CZRCD2	
EBD.15	005676-R	# CZRCD1	CZRCD2	
EBD.16	005766-R	# CZRCD1	CZRCD2	
EBD.17	006062-R	# CZRCD1	CZRCD2	
EBD.18	006160-R	# CZRCD1	CZRCD2	
EBD.19	006224-R	# CZRCD1	CZRCD2	
EBD.20	006264-R	# CZRCD1	CZRCD2	
EBD.21	006350-R	# CZRCD1	CZRCD2	
EBD.22	006436-R	# CZRCD1	CZRCD2	
EBH.31	006556-R	# CZRCD1	CZRCD2	
EBH.32	006602-R	# CZRCD1	CZRCD2	
EBH.34	006650-R	# CZRCD1	CZRCD2	
EBH.35	006674-R	# CZRCD1	CZRCD2	
EBH.36	006724-R	# CZRCD1	CZRCD2	
EBH.37	006750-R	# CZRCD1	CZRCD2	
EBH.38	007002-R	# CZRCD1	CZRCD2	
EBH.39	007022-R	# CZRCD1	CZRCD2	
EBH.40	007060-R	# CZRCD1	CZRCD2	
EBH.41	007106-R	# CZRCD1	CZRCD2	
EBH.42	007126-R	# CZRCD1	CZRCD2	
EBH.43	007174-R	# CZRCD1	CZRCD2	
EBS.01	005332-R	# CZRCD1	CZRCD2	
EF.CON	000036	# CZRCD1	# CZRCD2	# CZRCD3
EF.NEW	000035	# CZRCD1	# CZRCD2	# CZRCD3
EF.PWR	000034	# CZRCD1	# CZRCD2	# CZRCD3
EF.RES	000037	# CZRCD1	# CZRCD2	# CZRCD3
EF.STA	000040	# CZRCD1	# CZRCD2	# CZRCD3
EGD.10	004654-R	# CZRCD1	CZRCD3	
EGD.11	004714-R	# CZRCD1	CZRCD3	
EGD.12	004740-R	# CZRCD1	CZRCD3	

SYMBOL	VALUE	REFERENCES...		
EGD.13	004766-R	# CZRCD1	CZRCD3	
EGD.14	005014-R	# CZRCD1	CZRCD3	
EGD.15	005052-R	# CZRCD1	CZRCD3	
EGD.16	005104-R	# CZRCD1	CZRCD3	
EGD.17	005122-R	# CZRCD1	CZRCD3	
EGD.18	005152-R	# CZRCD1	CZRCD3	
EGD.19	005170-R	# CZRCD1	CZRCD3	
EGD.20	005210-R	# CZRCD1	CZRCD3	
EGD.21	005234-R	# CZRCD1	CZRCD3	
EGD.22	005260-R	# CZRCD1	CZRCD3	
EGH.30	005306-R	# CZRCD1	CZRCD3	
EGS.01	004556-R	# CZRCD1	CZRCD2	
EGS.02	004576-R	# CZRCD1	CZRCD2	
EMS.01	020746-R	# CZRCD2		
EMS.10	021010-R	# CZRCD2	CZRCD3	
EMS.12	021056-R	# CZRCD2	CZRCD3	
EMS.13	021120-R	# CZRCD2	CZRCD3	
EMS.14	021172-R	# CZRCD2	CZRCD3	
EMS.15	021240-R	# CZRCD2	CZRCD3	
EMS.16	021306-R	# CZRCD2	CZRCD3	
EMS.17	021360-R	# CZRCD2	CZRCD3	
EMS.18	021422-R	# CZRCD2	CZRCD3	
EMS.19	021474-R	# CZRCD2	CZRCD3	
EMS.20	021542-R	# CZRCD2	CZRCD3	
EMS.21	021604-R	# CZRCD2	CZRCD3	
EMS.22	021672-R	# CZRCD2	CZRCD3	
EMS.30	021734-R	# CZRCD2	CZRCD3	
EMS.42	022104-R	# CZRCD2	CZRCD3	
EMS.43	022172-R	# CZRCD2	CZRCD3	
ENTRY.	066252-R	# CZRCD1	CZRCD2	CZRCD3
ENV.US	062302-R	# CZRCD1	CZRCD2	CZRCD3
EOP.FL	066254-R	# CZRCD1	CZRCD2	CZRCD3
ERRBLK	002134-R	# CZRCD1		
ERRMSG	002132-R	# CZRCD1		
ERRNBR	002130-R	# CZRCD1		
ERRTYP	002126-R	# CZRCD1		
ETIME	010372-R	# CZRCD1	CZRCD2	
EVL	000004	# CZRCD1	# CZRCD2	# CZRCD3
EX.BB	007654-R	# CZRCD1	CZRCD2	
EX.BC	010012-R	# CZRCD1	CZRCD2	
EX.BD	010070-R	# CZRCD1	CZRCD2	
EX.CBC	007744-R	# CZRCD1	CZRCD2	
EX.CMD	007474-R	# CZRCD1	CZRCD2	
EX.CMP	007636-R	# CZRCD1	CZRCD2	
EX.CRN	007304-R	# CZRCD1	CZRCD2	CZRCD3
EX.DSC	007400-R	# CZRCD1	CZRCD2	
EX.EL	010150-R	# CZRCD1	CZRCD3	
EX.ESP	007546-R	# CZRCD1	CZRCD2	
EX.EVC	010262-R	# CZRCD1	CZRCD3	
EX.FMT	010242-R	# CZRCD1	CZRCD3	
EX.HMA	010306-R	# CZRCD1	CZRCD3	
EX.LBN	007720-R	# CZRCD1	CZRCD2	

```
SYMBOL  VALUE      REFERENCES...

EX.ONL  007534-R  # CZRCD1    CZRCD2
EX.O3   010356-R  # CZRCD1    CZRCD2    CZRCD3
EX.PA   010212-R  # CZRCD1    CZRCD3
EX.RD   007616-R  # CZRCD1    CZRCD2
EX.SA   007256-R  # CZRCD1    CZRCD2
EX.SB   007440-R  # CZRCD1    CZRCD2    CZRCD3
EX.SC   007342-R  # CZRCD1    CZRCD2
EX.SCC  007514-R  # CZRCD1    CZRCD2
EX.SND  007602-R  # CZRCD1    CZRCD2
EX.WRT  007626-R  # CZRCD1    CZRCD2
FREE.M  066316-R  # CZRCD1    CZRCD2    CZRCD3
GET.EN  015454-R  # CZRCD2    CZRCD3
GET.IO  015762-R  # CZRCD2    CZRCD3
GET.RE  015640-R  # CZRCD2    CZRCD3
GP$DIS  011022-R  # CZRCD2
GP$1    010626-R  # CZRCD2
GP$10   010750-R  # CZRCD2
GP$11   010760-R  # CZRCD2
GP$12   010766-R  # CZRCD2
GP$13   011000-R  # CZRCD2
GP$14   011014-R  # CZRCD2
GP$15   011026-R  # CZRCD2
GP$16   011034-R  # CZRCD2
GP$17   011044-R  # CZRCD2
GP$18   011054-R  # CZRCD2
GP$19   011064-R  # CZRCD2
GP$2    010636-R  # CZRCD2
GP$20   011102-R  # CZRCD2
GP$21   011114-R  # CZRCD2
GP$3    010646-R  # CZRCD2
GP$4    010660-R  # CZRCD2
GP$5    010672-R  # CZRCD2
GP$6    010702-R  # CZRCD2
GP$7    010716-R  # CZRCD2
GP$8    010730-R  # CZRCD2
GP$9    010742-R  # CZRCD2
HARD.E  016544-R  # CZRCD2    CZRCD3
HOE     100000    # CZRCD1  # CZRCD2  # CZRCD3
HOURS   066334-R  # CZRCD1    CZRCD2    CZRCD3
HWPT.B  010454-R  # CZRCD1
HWPT.I  010450-R  # CZRCD1
HWPT.P  010456-R  # CZRCD1
HWPT.V  010452-R  # CZRCD1
HWQ1    002136-R  # CZRCD1    CZRCD2
HWQ2    002152-R  # CZRCD1    CZRCD2
HWQ3    002162-R  # CZRCD1    CZRCD2
HWQ4    002174-R  # CZRCD1    CZRCD2
HWQ5    002230-R  # CZRCD1    CZRCD2
HWQ6    002314-R  # CZRCD1    CZRCD2
IBE     010000    # CZRCD1  # CZRCD2  # CZRCD3
IDU     000040    # CZRCD1  # CZRCD2  # CZRCD3
IER     020000    # CZRCD1  # CZRCD2  # CZRCD3
```

SYMBOL	VALUE	REFERENCES...		
IIP.FL	066256-R	# CZRCD1	CZRCD2	CZRCD3
IN.IOD	016236-R	# CZRCD2	CZRCD3	
IODQ	066206-R	# CZRCD1	CZRCD2	CZRCD3
IODQ.I	066246-R	# CZRCD1	CZRCD2	CZRCD3
IODQ.O	066250-R	# CZRCD1	CZRCD2	CZRCD3
IRC25.	043204-R	# CZRCD1	CZRCD2	CZRCD3
ISR	000100	# CZRCD1	# CZRCD2	# CZRCD3
IXE	004000	# CZRCD1	# CZRCD2	# CZRCD3
LOE	040000	# CZRCD1	# CZRCD2	# CZRCD3
LOT	000010	# CZRCD1	# CZRCD2	# CZRCD3
L$ACP	002110-R	# CZRCD1		
L$APT	002036-R	# CZRCD1		
L$AU	015000-R	CZRCD1	# CZRCD2	
L$AUT	002070-R	# CZRCD1		
L$AUTO	014104-R	CZRCD1	# CZRCD2	
L$CCP	002106-R	# CZRCD1		
L$CLEA	014314-R	CZRCD1	# CZRCD2	
L$CO	002032-R	# CZRCD1		
L$DEPO	002011-R	# CZRCD1		
L$DESC	010576-R	CZRCD1	# CZRCD2	
L$DESP	002076-R	# CZRCD1		
L$DEVP	002060-R	# CZRCD1		
L$DISP	002124-R	# CZRCD1		
L$DLY	002116-R	# CZRCD1	CZRCD2	CZRCD3
L$DTP	002040-R	# CZRCD1		
L$DTYP	002034-R	# CZRCD1		
L$DU	014710-R	CZRCD1	# CZRCD2	
L$DUT	002072-R	# CZRCD1		
L$DVTY	010550-R	CZRCD1	# CZRCD2	
L$EF	002052-R	# CZRCD1		
L$ENVI	002044-R	# CZRCD1		
L$ERRT	002126-R	# CZRCD1		
L$ETP	002102-R	# CZRCD1		
L$EXP1	002046-R	# CZRCD1		
L$EXP4	002064-R	# CZRCD1		
L$EXP5	002066-R	# CZRCD1		
L$HARD	010626-R	CZRCD1	# CZRCD2	
L$HIME	002120-R	# CZRCD1	CZRCD3	
L$HPCP	002016-R	# CZRCD1		
L$HPTP	002022-R	# CZRCD1		
L$HRDL	010624-R	# CZRCD2		
L$HW	010450-R	# CZRCD1		
L$HWLE	010446-R	# CZRCD1		
L$ICP	002104-R	# CZRCD1		
L$INIT	014072-R	CZRCD1	# CZRCD2	
L$LADP	002026-R	# CZRCD1		
L$LAST	073272-R	CZRCD1	# CZRCD4	
L$LOAD	002100-R	# CZRCD1		
L$LUN	002074-R	# CZRCD1	CZRCD2	CZRCD3
L$MREV	002050-R	# CZRCD1		
L$NAME	002000-R	# CZRCD1		
L$NDHR	010712-R	# CZRCD2		

CZRCD CREATED BY TKB ON 11-JUL-83 AT 08:48 PAGE 6

GLOBAL CROSS REFERENCE CREF V01

SYMBOL	VALUE	REFERENCES...		
L$NDHW	010460-R	# CZRCD1		
L$NDSF	011134-R	# CZRCD2		
L$NDSW	010540-R	# CZRCD1		
L$PRIO	002042-R	# CZRCD1		
L$PROT	010542-R	# CZRCD1		
L$PRT	002112-R	# CZRCD1		
L$REPP	002062-R	# CZRCD1		
L$REV	002010-R	# CZRCD1		
L$RPT	012236-R	CZRCD1	# CZRCD2	
L$SFTL	010714-R	# CZRCD2		
L$SOFT	010716-R	CZRCD1	# CZRCD2	
L$SPC	002056-R	# CZRCD1		
L$SPCP	002020-R	# CZRCD1		
L$SPTP	002024-R	# CZRCD1		
L$STA	002030-R	# CZRCD1		
L$SW	010464-R	# CZRCD1		
L$SWLE	010462-R	# CZRCD1		
L$TEST	002114-R	# CZRCD1		
L$TIML	002014-R	# CZRCD1		
L$UNIT	002012-R	# CZRCD1	CZRCD2	CZRCD3
MEM.MG	066255-R	# CZRCD1	CZRCD2	CZRCD3
MEM.SI	066314-R	# CZRCD1	CZRCD2	CZRCD3
MINUTE	066336-R	# CZRCD1	CZRCD2	CZRCD3
MSCP.E	045502-R	# CZRCD1	CZRCD2	CZRCD3
MSG.01	003466-R	# CZRCD1	CZRCD2	
MSG.02	003520-R	# CZRCD1	CZRCD3	
MSG.03	003616-R	# CZRCD1	CZRCD3	
MSG.04	003636-R	# CZRCD1	CZRCD3	
MSG.05	003670-R	# CZRCD1	CZRCD3	
MSG.06	003730-R	# CZRCD1	CZRCD3	
MSG.07	003772-R	# CZRCD1	CZRCD2	
MSG.08	004044-R	# CZRCD1	CZRCD3	
NEX	066356-R	# CZRCD1	CZRCD2	CZRCD3
NEX.TR	015010-R	# CZRCD2	CZRCD3	
NULL	003464-R	# CZRCD1		
NUM.BU	066322-R	# CZRCD1	CZRCD2	CZRCD3
OCL.X1	044074-R	# CZRCD1	CZRCD2	CZRCD3
OCL.X2	044076-R	# CZRCD1	CZRCD2	CZRCD3
OF.RC	066352-R	# CZRCD1	CZRCD2	CZRCD3
OUTC.L	043574-R	# CZRCD1	CZRCD2	CZRCD3
OUTC.T	043674-R	# CZRCD1	CZRCD2	CZRCD3
OUT.IO	016200-R	# CZRCD2	CZRCD3	
OVF.CH	016722-R	# CZRCD2	CZRCD3	
PATCH	042446-R	# CZRCD1	CZRCD2	CZRCD3
PLATT	010416-R	# CZRCD1	CZRCD2	
PNT	001000	# CZRCD1	# CZRCD2	# CZRCD3
PRI	002000	# CZRCD1	# CZRCD2	# CZRCD3
PRI00	000000	# CZRCD1	# CZRCD2	# CZRCD3
PRI01	000040	# CZRCD1	# CZRCD2	# CZRCD3
PRI02	000100	# CZRCD1	# CZRCD2	# CZRCD3
PRI03	000140	# CZRCD1	# CZRCD2	# CZRCD3
PRI04	000200	# CZRCD1	# CZRCD2	# CZRCD3

SYMBOL	VALUE	REFERENCES...		
PRI05	000240	# CZRCD1	# CZRCD2	# CZRCD3
PRI06	000300	# CZRCD1	# CZRCD2	# CZRCD3
PRI07	000340	# CZRCD1	# CZRCD2	# CZRCD3
PUTA.B	016136-R	# CZRCD2	CZRCD3	
PUTA.E	015600-R	# CZRCD2		
PUT.EN	015564-R	# CZRCD2	CZRCD3	
PUT.IO	016052-R	# CZRCD2	CZRCD3	
PUT.RE	015746-R	# CZRCD2	CZRCD3	
QIO	066310-R	# CZRCD1	CZRCD2	CZRCD3
RCINT0	037350-R	# CZRCD3		
RCINT1	037366-R	# CZRCD3		
RCINT2	037406-R	# CZRCD3		
RCINT3	037426-R	# CZRCD3		
RC25.A	043202-R	# CZRCD1	CZRCD2	CZRCD3
RETPKT	062442-R	# CZRCD1	CZRCD2	CZRCD3
RPS.X1	043570-R	# CZRCD1	CZRCD2	CZRCD3
RPS.X2	043572-R	# CZRCD1	CZRCD2	CZRCD3
RP.ADD	065504-R	# CZRCD1	CZRCD2	CZRCD3
RP.IND	065502-R	# CZRCD1	CZRCD2	CZRCD3
RP.SAV	043530-R	# CZRCD1	CZRCD2	CZRCD3
RP.USE	065442-R	# CZRCD1	CZRCD2	CZRCD3
SA.REG	066354-R	# CZRCD1	CZRCD2	CZRCD3
SB.COD	066346-R	# CZRCD1	CZRCD2	CZRCD3
SECOND	066340-R	# CZRCD1	CZRCD2	CZRCD3
SEND	017170-R	# CZRCD2	CZRCD3	
SET.CP	015162-R	# CZRCD2	CZRCD3	
SET.UP	015356-R	# CZRCD2	CZRCD3	
SFPTBL	010464-R	# CZRCD1		
STC.00	006542-R	# CZRCD1	CZRCD3	
STC.01	006556-R	# CZRCD1	CZRCD3	
STC.02	006602-R	# CZRCD1	CZRCD3	
STC.03	006626-R	# CZRCD1	CZRCD3	
STC.04	006650-R	# CZRCD1	CZRCD3	
STC.05	006674-R	# CZRCD1	CZRCD3	
STC.06	006724-R	# CZRCD1	CZRCD3	
STC.07	006750-R	# CZRCD1	CZRCD3	
STC.08	007002-R	# CZRCD1	CZRCD3	
STC.09	007022-R	# CZRCD1	CZRCD3	
STC.10	007060-R	# CZRCD1	CZRCD3	
STC.11	007106-R	# CZRCD1	CZRCD3	
STEP	066350-R	# CZRCD1	CZRCD2	CZRCD3
ST.COD	066344-R	# CZRCD1	CZRCD2	CZRCD3
SWM1	003370-R	# CZRCD1	CZRCD2	
SWP.DP	010475-R	# CZRCD1	CZRCD3	
SWP.ER	010464-R	# CZRCD1	CZRCD2	
SWP.ET	010472-R	# CZRCD1	CZRCD2	CZRCD3
SWP.FL	010474-R	# CZRCD1	CZRCD2	CZRCD3
SWP.ST	010470-R	# CZRCD1	CZRCD2	CZRCD3
SWP.UC	010476-R	# CZRCD1	CZRCD3	
SWP.UD	010500-R	# CZRCD1	CZRCD3	
SWP.XF	010466-R	# CZRCD1	CZRCD2	
SWQ1	002414-R	# CZRCD1	CZRCD2	

SYMBOL	VALUE	REFERENCES...					
SWQ10	003024-R	# CZRCD1	CZRCD2				
SWQ11	003100-R	# CZRCD1	CZRCD2				
SWQ12	003144-R	# CZRCD1	CZRCD2				
SWQ13	003176-R	# CZRCD1	CZRCD2				
SWQ14	003274-R	# CZRCD1	CZRCD2				
SWQ15	003352-R	# CZRCD1	CZRCD2				
SWQ2	002452-R	# CZRCD1	CZRCD2				
SWQ3	002530-R	# CZRCD1	CZRCD2				
SWQ4	002576-R	# CZRCD1	CZRCD2				
SWQ5	002656-R	# CZRCD1	CZRCD2				
SWQ6	002700-R	# CZRCD1	CZRCD2				
SWQ7	002720-R	# CZRCD1	CZRCD2				
SWQ8	002736-R	# CZRCD1	CZRCD2				
SWQ9	003010-R	# CZRCD1	CZRCD2				
TALLY	044100-R	# CZRCD1	CZRCD2	CZRCD3			
TICKS	066342-R	# CZRCD1	CZRCD2	CZRCD3			
T$FREE	073322-R	# CZRCD4					
T$PTHV	000002	CZRCD1	# CZRCD4				
T.ADDR	045500-R	# CZRCD1	CZRCD2	CZRCD3			
T.FLAG	066253-R	# CZRCD1	CZRCD2	CZRCD3			
T1	022460-R	CZRCD1	# CZRCD3				
UAM	000200	# CZRCD1	# CZRCD2	# CZRCD3			
UPD.IO	016634-R	# CZRCD2	CZRCD3				
WAIT	017434-R	# CZRCD2	CZRCD3				
XFR.CH	016776-R	# CZRCD2	CZRCD3				
$END.L	073324-R	# CZRCD4					
$SAVE2	042340-R	B16MUL	# B16SAV	CZRCD2	CZRCD3		
$SAVE3	042354-R	# B16SAV	CZRCD2	CZRCD3			
$SAVE4	042372-R	# B16SAV	CZRCD2	CZRCD3			
$SAVE5	042412-R	B16MUL	# B16SAV	CZRCD2	CZRCD3		

```
!*****************************************************************************
!                                                                             *
!                    L I T E R A L S                                          *
!                                                                             *
!*****************************************************************************
!
LITERAL
!
!****** HARDWARE LIMITS AND PARAMETERS
!
        MAX_CTLR        = 4,                    ! MAXIMUM NUMBER OF RC25 CONTROLLERS ALLOWED
        MAX_UNITS       = MAX_CTLR * 4,         ! MAXIMUM NUMBER OF UNITS TO TEST
        MAX_TRACK       = 164T,                 ! LARGEST TRACK NUMBER IN HOST AREA
        MAX_SECT        = 30,                   ! LARGEST HOST SECTOR NUMBER IN EACH TRACK
        SEC_PER_TRK     = 31,                   ! NUMBER OF HOST SECTORS PER TRACK
        BLK_SIZE        = 512,                  ! NUMBER OF BYTES PER SECTOR
!
!****** U/Q PORT RING SIZES
!
        CR_LOG          = 3,                    ! LOG2 LENGTH OF COMMAND RING
        RR_LOG          = 3,                    ! LOG2 LENGTH OF RESPONSE RING
        CRING_LEN       = 1 ^ CR_LOG,           ! COMMAND RING LENGTH
        RRING_LEN       = 1 ^ RR_LOG,           ! RESPONSE RING LENGTH
!
!****** TABLE AND OTHER STRUCTURE SIZES
!
        HWPT_LEN        = 4,                    ! SIZE (WORDS) OF HW P-TABLE
        COMM_LEN        = (RRING_LEN * 2) + (CRING_LEN * 2) + 4, ! SIZE (WORDS) OF U/Q PORT COMM AREA FOR ONE CTLR
        CST_LEN         = 7,                    ! SIZE (WORDS) OF A CONTROLLER STATUS TABLE
        TALLY_LEN       = 24,                   ! SIZE (WORDS) OF A UNIT'S STATISTICS TABLE
        RP_LEN          = 24,                   ! SIZE (WORDS) OF A RETURN PACKET
        RPS_LEN         = CRING_LEN,            ! SIZE (BYTES) OF A CONTROLLER'S RETPKT SAVE AREA
        IODQ_LEN        = MAX_CTLR * RRING_LEN, ! NUMBER OF ENTRIES IN I/O DONE QUEUE (IODQ)
        MSG_LEN         = 30,                   ! SIZE (WORDS) OF AN MSCP MESSAGE (TEXT PORTION)
        ENV_LEN         = MSG_LEN + 4,          ! SIZE (WORDS) OF AN MSCP ENVELOPE
        DCT_LEN         = 9,                    ! SIZE (WORDS) OF A DRIVER CONTROLLER TABLE
        BST_LEN         = 2,                    ! SIZE (WORDS) OF A UNIT'S BLOCK SEQ TABLE ENTRY
        RDM_LEN         = 16,                   ! SIZE (WORDS) OF THE RANDOM NUMBER TABLE
        DESC_LEN        = 2,                    ! SIZE (WORDS) OF AN I/O BUFFER DESCRIPTOR
        DMC_LEN         = 26,                   ! SIZE (WORDS) OF DM_COMM AREA FOR ONE CONTROLLER
        RP_CNT          = MAX_CTLR * RRING_LEN, ! NUMBER OF RETURN PACKETS IN POOL
        MAX_UDP_CNT     = 16,                   ! MAX SIZE (WORDS) OF USER DATA PATTERN
        MAX_BUF_CNT     = (CRING_LEN * 2) * MAX_CTLR, ! MAX NO. OF I/O BUFFERS (SIZE OF BUFF_DESC AND BUFF_OWN)
        ENV_CNT         = ((CRING_LEN * 2) + RRING_LEN) * MAX_CTLR, ! NO. OF MSCP ENVELOPES IN POOL
        OUTC_CNT        = CRING_LEN * 2,        ! NUMBER OF ENTRIES IN A CONTROLLER'S OUTSTANDING CMD LIST
        DP_CNT          = 21,                   ! NUMBER OF PRE-DEFINED DATA PATTERNS
!
!****** OFFSETS
!
        OF_UN           = 3,                    ! WORD OFFSET FROM START OF CST TO FIRST UNIT
!
!****** SW P-TABLE FLAGS (SWP_FLAGS)
!
        SWF_SEL         = %O'1',                ! SUPPRESS PRINTING ERROR LOG MESSAGES
        SWF_DM          = %O'2',                ! RUN DM EXERCISER INSTEAD OF MULTI-DRIVE SUBTEST
        SWF_RDM         = %O'4',                ! RANDOM SEEK MODE
        SWF_CRC         = %O'10',               ! READ-COMPARES AT CONTROLLER
        SWF_WO          = %O'20',               ! WRITE ONLY
```

```
        SWF_CWC         = %O'40',               ! WRITE-COMPARES AT CONTROLLER
        SWF_HWC         = %O'100',              ! WRITE-COMPARES AT HOST
        SWF_UDP         = %O'200',              ! USER-DEFINED DATA PATTERN
        SWF_PHWC        = %O'6',                ! HOST WRITE COMPARES (BIT POSITION)
!
!****** ENTRY_REASON VALUES (HOW CURRENT PASS WAS INVOKED)
!
        START           = 1,                    ! START
        RESTART         = 2,                    ! RESTART
        CONT            = 3,                    ! CONTINUE
        PWR_FAIL        = 4,                    ! POWER FAIL
        NEW_PASS        = 5,                    ! NEW PASS
!
!****** DROP UNIT REASONS (LOADED INTO DUR VECTOR)
!
        DU_USER         = 0,                    ! USER COMMAND
        DU_CONF         = 1,                    ! CONFIGURATION ERROR
        DU_INIT         = 2,                    ! INITIALIZATION ERROR
        DU_HERR         = 3,                    ! HARD ERROR LIMIT REACHED
        DU_FATAL        = 4,                    ! UNRECOVERABLE DEVICE ERROR
!
!****** TIMEOUT VALUES (IN SECONDS)
!
        TO_INIT         = 120,                  ! INIT SUBTEST COMMANDS
        TO_DUP          = 30,                   ! DUP COMMANDS
        TO_IO           = 300,                  ! I/O TRANSFER COMMANDS
        TO_DM           = 360,                  ! ONE FPT PASS (DM EXERCISER SUBTEST)
!
!****** MISCELLANEOUS LITERALS
!
        INI_ATT         = 2,                    ! NUMBER OF HARD INIT ATTEMPTS BEFORE FAILURE IS ASSUMED
        WR_RING         = ((%O'200') OR (CR_LOG ^ 3) OR (RR_LOG)), ! USED IN HARD INIT SEQUENCE
        QIO_PER_CTLR    = CRING_LEN * 2,        ! MAXIMUM NUMBER OF OUTSTANDING QIOS PER CONTROLLER
!
!****** MSCP ENVELOPE DESCRIPTOR
!
        ED_OWN          = %O'100000',           ! OWNERSHIP BIT
!
!****** CONNECTION ID VALUES (MSCP_ENV, RETPKT)
!           (SERVE AS SOURCES AND DESTINATIONS OF MSCP MESSAGES)
!
        CID_DISK        = 0,                    ! DISK MSCP
!       CID_TAPE        = 1,                    ! TAPE MSCP
        CID_DUP         = 2,                    ! DIAGNOSTIC AND UTILITIES PROTOCOL
        CID_DRIVER      = 3,                    ! EXERCISER "DRIVER"
!
!****** MESSAGE TYPE VALUES
!
        MT_SEQ          = 0,                    ! SEQUENTIAL (FROM PORT)
        MT_DG           = 1,                    ! DATAGRAM (FROM PORT)
!       MT_CRD          = 2,                    ! CREDIT NOTIFICATION (FROM PORT)
        MT_FATAL        = 3,                    ! FATAL DEVICE ERROR (FROM "DRIVER")
        MT_TIMEOUT      = 4,                    ! COMMAND TIMEOUT (FROM "DRIVER")
!
!****** MSCP COMMAND PACKET OPCODES
!
        OP_SCC          = %O'4',                ! SET CONTROLLER CHARACTERISTICS COMMAND
        OP_ONL          = %O'11',               ! ONLINE COMMAND
```

```
        OP_ACC          = %O'20',               ! ACCESS COMMAND
        OP_RD           = %O'41',               ! READ COMMAND
        OP_WRT          = %O'42',               ! WRITE COMMAND
        OP_MSK          = %O'177',              ! OPCODE MASK
        OP_END          = %O'200',              ! ENDCODE DESIGNATOR

!****** DUP COMMAND OPCODES

        OP_ESP          = %O'2',                ! EXECUTE SUPPLIED PROGRAM
        OP_SND          = %O'4',                ! SEND DATA

!****** MSCP COMMAND MODIFIERS

        MD_EXP          = %O'100000',           ! EXPRESS REQUEST
        MD_CMP          = %O'40000',            ! COMPARE

!****** MSCP ERROR LOG MESSAGES - FORMAT FIELD CODES

        FM_CNT          = 0,                    ! CONTROLLER ERROR
        FM_BAD          = 1,                    ! HOST MEMORY ACCESS ERROR
        FM_SDE          = 4,                    ! SMALL DISK ERROR

!****** CONTROLLER FLAGS (IN SET CONTROLLER CHARACTERISTICS COMMAND)

        CF_MSC          = %O'100',              ! ENABLE MISCELLANEOUS ERROR LOG MESSAGES
        CF_THS          = %O'20',               ! ENABLE THIS HOST'S ERROR LOG MESSAGES

!****** UNIT FLAGS (IN ONLINE RESPONSE)

        UF_WPH          = %O'20000',            ! WRITE PROTECT (HARDWARE)

!****** STATUS / EVENT CODE DEFINITIONS

        ST_SUC          = %O'0',                ! SUCCESS
        ST_OFL          = %O'3',                ! UNIT OFFLINE
        ST_DAT          = %O'10',               ! DATA ERROR

!****** END MESSAGE FLAGS

        EF_BBR          = %O'200',              ! BAD BLOCK REPORTED

!****** RC25 LITERALS

        RCIP            = 0,                    ! IP REGISTER
        RCSA            = 1,                    ! SA REGISTER

!    SA REGISTER BIT DEFINITIONS

        SA_S1           = %O'004000',           ! STEP 1 STATUS BIT
        SA_S2           = %O'010000',           !    :  2
        SA_S3           = %O'020000',           !    :  3
        SA_S4           = %O'040000',           !    V  4
        SA_ERR          = %O'100000',           ! ERROR INDICATOR
        SA_INT          = %O'000200',           ! INTERRUPT ENABLE DURING INITIALIZATION
        SA_GO           = %O'000001',           ! GO BIT TO START FIRMWARE

!    INITIALIZATION SEQUENCE READ MASKS
```

```
        S1_MASK             = %O'176000',           ! STEP 1 READ BITS
        S2_MASK             = %O'174377',           !   :  2
        S3_MASK             = %O'174377',           !   :  3
        S4_MASK             = %O'174000',           !   V  4
!
!****** MEMORY MANAGEMENT (KT-11) REGISTER LOCATIONS
!
        KTPDR0              = %O'172300',           ! PAGE DESCRIPTOR REGISTERS
        KTPDR1              = %O'172302',
        KTPDR2              = %O'172304',
        KTPDR3              = %O'172306',
        KTPDR4              = %O'172310',
        KTPDR5              = %O'172312',
        KTPDR6              = %O'172314',
        KTPDR7              = %O'172316',
        KTPAR0              = %O'172340',           ! PAGE ADDRESS REGISTERS
        KTPAR1              = %O'172342',
        KTPAR2              = %O'172344',
        KTPAR3              = %O'172346',
        KTPAR4              = %O'172350',
        KTPAR5              = %O'172352',
        KTPAR6              = %O'172354',
        KTPAR7              = %O'172356',
        MMR0                = %O'177572',           ! MEMORY MANAGEMENT REGISTER 0
        MMR3                = %O'172516',           ! MEMORY MANAGEMENT REGISTER 3
!
!****** LITERALS FOR READABILITY
!
        YES                 = 1,
        NO                  = 0,
        TRUE                = 1,
        FALSE               = 0,
        SUCCESS             = 1,
        FAILURE             = 0,
        FOUND               = 1,
        NOT_FOUND           = 0,
        PRESENT             = 1,                    ! PLATTER IS PRESENT IN CONTROLLER
        NOT_PRESENT         = 0,                    ! PLATTER IS NOT PRESENT IN CONTROLLER
        UNPROTECTED         = 1,                    ! PLATTER HAS UNPROTECTED CUSTOMER LBN'S
        PROTECTED           = 0,                    ! PLATTER HAS PROTECTED CUSTOMER LBN'S
        ONLINE              = 1,
        OFFLINE             = 0,
        ALL_ONES            = %O'177777',
!
!****** CLOCK FLAGS FOR CLK_TYPE
!
        NO_CLOCK            = 0,
        L_CLOCK             = -1,
        P_CLOCK             = 1;
```

```
!*****************************************************************************
!                                                                             *
!                       F I E L D S                                           *
!                                                                             *
!*****************************************************************************
!
FIELD
!
!****** HARDWARE P-TABLE FIELDS
!
HWP_FIELDS =
        SET
        HWP_IP_ADDR     = [0, 0, 16, 0],        ! IP ADDRESS
        HWP_VECTOR      = [1, 0, 16, 0],        ! VECTOR ADDRESS
        HWP_BR_LEVEL    = [2, 0, 16, 0],        ! BUS REQUEST LEVEL
        HWP_PLAT        = [3, 0, 16, 0],        ! PLATTER (ALL FIELDS)
        HWP_PLAT_ADDR   = [3, 0, 8, 0],         ! PLATTER ADDRESS (MSCP UNIT NO.)
        HWP_PLAT_CP     = [3, 15, 1, 0]         ! PROTECT CUSTOMER DATA BIT
        TES;
!
!****** U/Q PORT COMMUNICATION AREA HEADER FIELDS
!
COM_FIELDS =
        SET
        ADAP_CH         = [1, 8, 8, 0],         ! ADAPTER CHANNEL NUMBER FOR PURGES
        CMD_INT         = [2, 0, 16, 0],        ! COMMAND RING INTERRUPT
        RSP_INT         = [3, 0, 16, 0]         ! RESPONSE RING INTERRUPT
        TES;
!
!****** CONTROLLER STATUS TABLE (CST) FIELDS
!
C_FIELDS =
        SET
        IP_ADDR         = [0, 0, 16, 0],        ! IP ADDRESS
        VEC_ADDR        = [1, 0, 9, 0],         ! VECTOR ADDRESS
        STATE           = [1, 15, 1, 0],        ! CONTROLLER STATUS
        BR_LEV          = [2, 0, 8, 0],         ! BUS REQUEST LEVEL
        U_CNT           = [2, 8, 8, 0],         ! NUMBER OF TESTABLE UNITS UNDER THIS CONTROLLER
!
        P1_ALL          = [3, 0, 16, 0],        ! 1ST PLATTER (ALL FIELDS)
        P1_ADDR         = [3, 0, 8, 0],         ! 1ST PLATTER ADDRESS (MSCP UNIT NO.)
        P1_UNIT         = [3, 8, 5, 0],         ! 1ST PLATTER UNIT NUMBER (DRS UNIT NO.)
        P1_STAT         = [3, 13, 1, 0],        ! 1ST PLATTER STATUS BIT
        P1_PRES         = [3, 14, 1, 0],        ! 1ST PLATTER PRESENT BIT
        P1_PROT         = [3, 15, 1, 0]         ! 1ST PLATTER PROTECT CUSTOMER DATA BIT
!
!       THE REMAINING C_FIELDS ARE NOT REFERENCED DIRECTLY, BUT RATHER WITH OFFSETS
!       AND MACROS. THE FIELDS ARE DEFINED HERE ONLY FOR DOCUMENTATION PURPOSES.
!
!       P2_ALL          = [4, 0, 16, 0],        ! 2ND PLATTER (ALL FIELDS)
!       P2_ADDR         = [4, 0, 8, 0],         ! 2ND PLATTER ADDRESS (MSCP UNIT NO.)
!       P2_UNIT         = [4, 8, 5, 0],         ! 2ND PLATTER UNIT NUMBER (DRS UNIT NO.)
!       P2_STAT         = [4, 13, 1, 0],        ! 2ND PLATTER STATUS BIT
!       P2_PRES         = [4, 14, 1, 0],        ! 2ND PLATTER PRESENT BIT
!       P2_PROT         = [4, 15, 1, 0],        ! 2ND PLATTER PROTECT CUSTOMER DATA BIT
!
!       P3_ALL          = [5, 0, 16, 0],        ! 3RD PLATTER (ALL FIELDS)
!       P3_ADDR         = [5, 0, 8, 0],         ! 3RD PLATTER ADDRESS (MSCP UNIT NO.)
```

```
        P3_UNIT         = [5, 8, 5, 0],         ! 3RD PLATTER UNIT NUMBER (DRS UNIT NO.)
        P3_STAT         = [5, 13, 1, 0],        ! 3RD PLATTER STATUS BIT
        P3_PRES         = [5, 14, 1, 0],        ! 3RD PLATTER PRESENT BIT
        P3_PROT         = [5, 15, 1, 0],        ! 3RD PLATTER PROTECT CUSTOMER DATA BIT

        P4_ALL          = [6, 0, 16, 0],        ! 4TH PLATTER (ALL FIELDS)
        P4_ADDR         = [6, 0, 8, 0],         ! 4TH PLATTER ADDRESS (MSCP UNIT NO.)
        P4_UNIT         = [6, 8, 5, 0],         ! 4TH PLATTER UNIT NUMBER (DRS UNIT NO.)
        P4_STAT         = [6, 13, 1, 0],        ! 4TH PLATTER STATUS BIT
        P4_PRES         = [6, 14, 1, 0],        ! 4TH PLATTER PRESENT BIT
        P4_PROT         = [6, 15, 1, 0]         ! 4TH PLATTER PROTECT CUSTOMER DATA BIT
        TES,

!****** MSCP ENVELOPE FIELDS
!               (NOTE: THE FIRST TWO WORDS OF AN MSCP ENVELOPE (ITS BASE
!               ADDRESS) CONTAIN THE ENVELOPE'S OWN DESCRIPTOR, RATHER THAN THE
!               MESSAGE BODY (TEXT + 0). THE MESSAGE BODY BEGINS AT WORD 4.

E_FIELDS =
        SET

!   HEADER FIELDS

        ENV_LO          = [0, 0, 16, 0],        ! ENVELOPE DESCRIPTOR (LO ORDER)
        ENV_HI          = [1, 0, 16, 0],        ! ENVELOPE DESCRIPTOR (HI ORDER - ALL FIELDS)
        ENV_U           = [1, 0, 2, 0],         ! ENVELOPE DESCRIPTOR (HI ORDER UNIBUS BITS)
        ENV_Q           = [1, 2, 4, 0],         ! ENVELOPE DESCRIPTOR (HI ORDER Q-BUS BITS)
        ENV_F           = [1, 14, 1, 0],        ! ENVELOPE DESCRIPTOR FLAG BIT
        ENV_O           = [1, 15, 1, 0],        ! ENVELOPE DESCRIPTOR OWNERSHIP BIT
        MSGLEN          = [2, 0, 16, 0],        ! MESSAGE LENGTH
        CREDITS         = [3, 0, 4, 0],         ! CREDITS
        MSGTYP          = [3, 4, 4, 0],         ! MESSAGE TYPE
        CONNID          = [3, 8, 8, 0],         ! CONNECTION ID

!   GENERIC COMMAND PACKET AND END PACKET FILEDS

        CRN_LO          = [4, 0, 16, 0],        ! COMMAND REF NUMBER (LO ORDER)
        CRN_HI          = [5, 0, 16, 0],        ! COMMAND REF NUMBER (HI ORDER)
        PL_ADDR         = [6, 0, 16, 0],        ! PLATTER ADDRESS (MSCP UNIT NUMBER)
        OPCODE          = [8, 0, 8, 0],         ! OPCODE AND ENDCODE
        MODIFY          = [9, 0, 16, 0],        ! COMMAND MODIFIERS

!   READ, WRITE, AND ACCESS COMMAND FIELDS (FOR COMMAND AND END PACKETS)

        BC_LO           = [10, 0, 16, 0],       ! BYTE COUNT (LO ORDER)
        BC_HI           = [11, 0, 16, 0],       ! BYTE COUNT (HI ORDER)
        BUF_0           = [12, 0, 16, 0],       ! I/O BUFFER DESCRIPTOR
        BUF_1           = [13, 0, 16, 0],       !
        BUF_2           = [14, 0, 16, 0],       !
        BUF_3           = [15, 0, 16, 0],       !
        BUF_4           = [16, 0, 16, 0],       !
        BUF_5           = [17, 0, 16, 0],       !
        LBN_L           = [18, 0, 16, 0],       ! LOGICAL BLOCK NUMBER (LO ORDER)
        LBN_H           = [19, 0, 16, 0],       ! LOGICAL BLOCK NUMBER (HI ORDER)

!   SET CONTROLLER CHARACTERISTICS COMMAND FIELDS

        C_FLAGS         = [11, 0, 16, 0],       ! CONTROLLER FLAGS
```

```
!
!   ONLINE COMMAND FIELDS
!
        DDPAR           = [18, 0, 16, 0],       ! DEVICE-DEPENDENT PARAMETERS
!
!   DUP COMMAND FIELDS
!
        DBC_LO          = [10, 0, 16, 0],       ! BYTE COUNT (LO ORDER)
        DBC_HI          = [11, 0, 16, 0],       ! BYTE COUNT (HI ORDER)
        DBUF_0          = [12, 0, 16, 0],       ! BUFFER DESCRIPTOR
        DBUF_1          = [13, 0, 16, 0],       !
        DBUF_2          = [14, 0, 16, 0],       !
        DBUF_3          = [15, 0, 16, 0],       !
        DBUF_4          = [16, 0, 16, 0],       !
        DBUF_5          = [17, 0, 16, 0],       !
        OBUF_0          = [18, 0, 16, 0],       ! OVERLAY BUFFER DESCRIPTOR
        OBUF_1          = [19, 0, 16, 0],       !
        OBUF_2          = [20, 0, 16, 0],       !
        OBUF_3          = [21, 0, 16, 0],       !
        OBUF_4          = [22, 0, 16, 0],       !
        OBUF_5          = [23, 0, 16, 0],       !
!
!   ERROR LOG MESSAGE FIELDS
!
        FORMAT          = [8, 0, 8, 0],         ! FORMAT
        EVENT           = [9, 0, 5, 0],         ! EVENT CODE
        SUBC            = [9, 5, 11, 0],        ! SUB-CODE
        MA_LO           = [16, 0, 16, 0],       ! HOST MEMORY ADDRESS (LO ORDER)
        MA_HI           = [17, 0, 16, 0]        ! HOST MEMORY ADDRESS (HI ORDER)
        TES,
!
!****** RETURN PACKET (RETPKT) FIELDS
!           (SIMILAR, BUT NOT IDENTICAL, TO MSCP ENVELOPE FIELDS)
!
RP_FIELDS =
        SET
!
!   COMMON TO ALL RETURN PACKETS FROM DISK MSCP
!
        MESLEN          = [0, 0, 16, 0],        ! MESSAGE LENGTH
        CTLR            = [1, 0, 4, 0],         ! CONTROLLER NUMBER (CREDITS OVERWRITTEN)
        MESTYP          = [1, 4, 4, 0],         ! MESSAGE TYPE
        CONID           = [1, 8, 8, 0],         ! CONNECTION ID
        CRF_LO          = [2, 0, 16, 0],        ! COMMAND REFERENCE NUMBER (LO ORDER)
        CRF_HI          = [3, 0, 16, 0],        ! COMMAND REFERENCE NUMBER (HI ORDER)
        PLAT            = [4, 0, 16, 0],        ! PLATTER ADDRESS (MSCP UNIT NUMBER)
        CMDMOD          = [5, 0, 16, 0],        ! COMMAND MODIFIERS
        ENDCOD          = [6, 0, 8, 0],         ! END CODE
        FLAGS           = [6, 8, 8, 0],         ! FLAGS
        STATUS          = [7, 0, 16, 0],        ! STATUS AND SUB-CODE
        STSCOD          = [7, 0, 5, 0],         ! STATUS CODE
        SUBCOD          = [7, 5, 11, 0],        ! SUB-CODE
!
!   READ, WRITE, AND ACCESS COMMAND RETURN PACKETS
!
        BCNT_LO         = [8, 0, 16, 0],        ! BYTE COUNT (LO ORDER)
        BCNT_HI         = [9, 0, 16, 0],        ! BYTE COUNT (HI ORDER)
        BUFF_0          = [10, 0, 16, 0],       ! I/O BUFFER DESCRIPTOR
```

```
        BUFF_1          = [11, 0, 16, 0],       !
!       BUFF_2          = [12, 0, 16, 0],       !
!       BUFF_3          = [13, 0, 16, 0],       !
!       BUFF_4          = [14, 0, 16, 0],       !
!       BUFF_5          = [15, 0, 16, 0],       !
        BBLK_LO         = [16, 0, 16, 0],       ! FIRST BAD BLOCK (LO ORDER)
        BBLK_HI         = [17, 0, 16, 0],       ! FIRST BAD BLOCK (HI ORDER)
        CBCNT_LO        = [18, 0, 16, 0],       ! BYTE COUNT FROM CMD PACKET (LO ORDER)
        CBCNT_HI        = [19, 0, 16, 0],       ! BYTE COUNT FROM CMD PACKET (HI ORDER)
        LBN_LO          = [20, 0, 16, 0],       ! LOGICAL BLOCK NUMBER (LO ORDER)
        LBN_HI          = [21, 0, 16, 0],       ! LOGICAL BLOCK NUMBER (HI ORDER)
!
!   UNIT ONLINE RETURN PACKET
!
        U_FLGS          = [9, 0, 16, 0],        ! UNIT FLAGS
        USIZ_LO         = [20, 0, 16, 0],       ! UNIT SIZE (LO ORDER)
        USIZ_HI         = [21, 0, 16, 0]        ! UNIT SIZE (HI ORDER)
        TES,
!
!****** STATISTICS TABLE (TALLY) FIELDS
!
T_FIELDS =
        SET
        READ_LO         = [0, 0, 16, 0],        ! NUMBER OF READS (LO ORDER)
        READ_HI         = [1, 0, 16, 0],        ! NUMBER OF READS (HI ORDER)
        WRIT_LO         = [2, 0, 16, 0],        ! NUMBER OF WRITES (LO ORDER)
        WRIT_HI         = [3, 0, 16, 0],        ! NUMBER OF WRITES (HI ORDER)
        SEEK_LO         = [4, 0, 16, 0],        ! NUMBER OF DM EXERCISER SEEKS (LO ORDER)
        SEEK_HI         = [5, 0, 16, 0],        ! NUMBER OF DM EXERCISER SEEKS (HI ORDER)
        BR_LO           = [6, 0, 16, 0],        ! NUMBER OF BYTES READ (LO ORDER 1000)
        BR_HI           = [7, 0, 16, 0],        ! NUMBER OF BYTES READ (HI ORDER 1000)
        MB_READ         = [8, 0, 16, 0],        ! MEGABYTES READ
        BW_LO           = [9, 0, 16, 0],        ! NUMBER OF BYTES WRITTEN (LO ORDER 1000)
        BW_HI           = [10, 0, 16, 0],       ! NUMBER OF BYTES WRITTEN (HI ORDER 1000)
        MB_WRIT         = [11, 0, 16, 0],       ! MEGABYTES WRITTEN
        ER_HRD          = [12, 0, 16, 0],       ! NUMBER OF HARD ERRORS
        ER_LOG          = [13, 0, 16, 0],       ! NUMBER OF ERROR LOG MESSAGES
        ER_SFT          = [14, 0, 16, 0],       ! NUMBER OF DM SOFT ERRORS
        ECC_1           = [15, 0, 16, 0],       ! NUMBER OF 1-SYMBOL ECC ERRORS
        ECC_2           = [16, 0, 16, 0],       ! NUMBER OF 2-SYMBOL ECC ERRORS
        ECC_3           = [17, 0, 16, 0],       ! NUMBER OF 3-SYMBOL ECC ERRORS
        ECC_4           = [18, 0, 16, 0],       ! NUMBER OF 4-SYMBOL ECC ERRORS
        ECC_5           = [19, 0, 16, 0],       ! NUMBER OF 5-SYMBOL ECC ERRORS
        ECC_6           = [20, 0, 16, 0],       ! NUMBER OF 6-SYMBOL ECC ERRORS
        ECC_7           = [21, 0, 16, 0],       ! NUMBER OF 7-SYMBOL ECC ERRORS
        ECC_8           = [22, 0, 16, 0],       ! NUMBER OF 8-SYMBOL ECC ERRORS
        ECC_ONLY        = [23, 0, 16, 0]        ! ECC-FIELD-ONLY ERRORS
        TES,
!
!****** DRIVER CONTROLLER TABLE (DCT) FIELDS
!
DC_FIELDS =
        SET
        WORD0           = [0, 0, 16, 0],        ! ALL FIELDS IN WORD 0
        CRING_CNT       = [0, 0, 8, 0],         ! NUMBER OF SLOTS IN CRING NOT YET RETURNED TO HOST
        IG_INT          = [0, 14, 1, 0],        ! IGNORE INTERRUPT BIT
        STAT            = [0, 15, 1, 0],        ! ONLINE / OFFLINE STATUS
        SA_SAVE         = [1, 0, 16, 0],        ! SA REGISTER SAVE WORD
```

```
        RR_BEG          = [2, 0, 16, 0],        ! FIXED ADDRESSES OF START AND
        RR_END          = [3, 0, 16, 0],        !       END OF EACH RING
        CR_BEG          = [4, 0, 16, 0],        !             :
        CR_END          = [5, 0, 16, 0],        !             V
        RR_POLL         = [6, 0, 16, 0],        ! ADDR OF NEXT RRING SLOT TO BE POLLED
        CR_POLL         = [7, 0, 16, 0],        ! ADDR OF NEXT CRING SLOT TO BE POLLED
        CR_NEXT         = [8, 0, 16, 0]         ! ADDR OF NEXT AVAIL CRING SLOT
        TES,
!
!****** DM EXERCISER COMMUNICATION AREA (DM_COMM) FIELDS
!
DMC_FIELDS =
        SET
        FPT_ACC         = [24, 0, 16, 0],       ! FRONT PANEL TEST ACCESS WORD
        HOST_ACC        = [25, 0, 16, 0]        ! HOST ACCESS WORD
!
!       THE REMAINING DMC_FIELDS ARE REFERENCED BY ADDRESS POINTERS. THE FIELDS
!       ARE DEFINED HERE ONLY FOR DOCUMENTATION PURPOSES.
!
        DM_P1           = [0, 0, 16, 0],        ! PLATTER ADDRESSES (UNIT PLUG
        DM_P2           = [1, 0, 16, 0],        !     NUMBERS) OF DISKS TO BE
        DM_P3           = [2, 0, 16, 0],        !     DM-EXERCISED
        DM_P4           = [3, 0, 16, 0],        !
!
        P1_SEEKS        = [4, 0, 16, 0],        ! 1ST PLATTER - NO. OF SEEKS
        P1_READS        = [5, 0, 16, 0],        ! 1ST PLATTER - NO. OF READS
        P1_WRITES       = [6, 0, 16, 0],        ! 1ST PLATTER - NO. OF WRITES
        P1_SOFT         = [7, 0, 16, 0],        ! 1ST PLATTER - NO. OF SOFT ERRORS
        P1_HARD         = [8, 0, 16, 0],        ! 1ST PLATTER - NO. OF HARD ERRORS
!
        P2_SEEKS        = [9, 0, 16, 0],        ! 2ND PLATTER - NO. OF SEEKS
        P2_READS        = [10, 0, 16, 0],       ! 2ND PLATTER - NO. OF READS
        P2_WRITES       = [11, 0, 16, 0],       ! 2ND PLATTER - NO. OF WRITES
        P2_SOFT         = [12, 0, 16, 0],       ! 2ND PLATTER - NO. OF SOFT ERRORS
        P2_HARD         = [13, 0, 16, 0],       ! 2ND PLATTER - NO. OF HARD ERRORS
!
        P3_SEEKS        = [14, 0, 16, 0],       ! 3RD PLATTER - NO. OF SEEKS
        P3_READS        = [15, 0, 16, 0],       ! 3RD PLATTER - NO. OF READS
        P3_WRITES       = [16, 0, 16, 0],       ! 3RD PLATTER - NO. OF WRITES
        P3_SOFT         = [17, 0, 16, 0],       ! 3RD PLATTER - NO. OF SOFT ERRORS
        P3_HARD         = [18, 0, 16, 0],       ! 3RD PLATTER - NO. OF HARD ERRORS
!
        P4_SEEKS        = [19, 0, 16, 0],       ! 4TH PLATTER - NO. OF SEEKS
        P4_READS        = [20, 0, 16, 0],       ! 4TH PLATTER - NO. OF READS
        P4_WRITES       = [21, 0, 16, 0],       ! 4TH PLATTER - NO. OF WRITES
        P4_SOFT         = [22, 0, 16, 0],       ! 4TH PLATTER - NO. OF SOFT ERRORS
        P4_HARD         = [23, 0, 16, 0]        ! 4TH PLATTER - NO. OF HARD ERRORS
        TES,
```

```
!
!****** BLOCK SEQUENCE TABLE (BST) FIELDS
!
B_FIELDS =
        SET
        SECTOR          = [0, 0, 16, 0],        ! SECTOR
        TRACK           = [1, 0, 16, 0]         ! TRACK
        TES,
!
!****** I/O BUFFER DESCRIPTOR FIELDS (BUFF_DESC)
!
BD_FIELDS =
        SET
        BD_LO           = [0, 0, 16, 0],        ! LOW-ORDER 16 BITS
        BD_HI           = [1, 0, 16, 0]         ! HIGH-ORDER U/Q BITS
        TES,
!
!****** RC25 REGISTER FIELDS
!
RC_REG =
        SET
        RC_ALL          = [0, 16, 0]            ! DEFINE ALL BITS
        TES;
```

```
!*****************************************************************************
!                                                                            *
!                       M A C R O S                                          *
!                                                                            *
!*****************************************************************************

MACRO

!****** ALL FIELDS OF A WORD

        ALLBIT          = 0, 16, 0%,            ! ALL FIELDS

!****** CST FIELDS (WORDS 3 - 6)

        P_ADDR          = 0, 8, 0%,             ! PLATTER ADDRESS (MSCP UNIT NO.)
        P_UNIT          = 8, 5, 0%,             ! PLATTER UNIT NUMBER (DRS UNIT NO.)
        P_STAT          = 13, 1, 0%,            ! PLATTER STATUS BIT
        P_PRES          = 14, 1, 0%,            ! PLATTER PRESENT BIT
        P_PROT          = 15, 1, 0%,            ! PLATTER PROTECTION BIT

!****** BIT TEST

BIT_TST (ADDR, EXPECTED) =
    (IF (.ADDR AND EXPECTED) EQLU EXPECTED
    THEN
        TRUE
    ELSE
        FALSE )%,

!****** RC25 WRITE

WRT_RC25 (O, FIELDNAM, IMAGE) =
    BEGIN
    LOCAL
        RC_REG;
    RC_REG <%FIELDEXPAND (FIELDNAM)> = IMAGE;
    (.RC25_ADDR + (%UPVAL * O)) = .RC_REG;
    END%;
```

```
!*****************************************************************************
!                                                                             *
!                        S T R U C T U R E S                                  *
!                                                                             *
!*****************************************************************************
!
!
!****** RC25 ACCESS ALGGRITHM
!
STRUCTURE
    RC25 [O, P, S, E] =
        BEGIN
        LOCAL
            RC_REG;
        RC_REG = .(RC25 + %UPVAL * O) <O, %BPVAL, O>;
        RC_REG
        END
        <P, S, E>;
```

```
0001  MODULE CZRCD1 (
0002          %TITLE 'CZRCDA0 RC25 DISK EXERCISER'
0003          IDENT = 'V01.0',
0004          ADDRESSING_MODE (ABSOLUTE)
0005          ) =
0006  BEGIN
0007
0043  %SBTTL 'PROGRAM HEADER'
0044
0045  LIBRARY 'CZRCDL';                       ! RC25 EXERCISER GLOBAL LIBRARY
0046  REQUIRE 'BLSMAC.REQ';                   ! DIAGNOSTIC SUPERVISOR LIBRARY
1535
1536  LITERAL
1537      DS$NBR_OF_TESTS = 1;                ! NUMBER OF TESTS IN THIS DIAGNOSTIC
1538
1539  EQUALS;
1540
1541  POINTER (ALL);
1542
1543  !+
1544  !       THE PROGRAM HEADER IS THE INTERFACE BETWEEN THE DIAGNOSTIC PROGRAM
1545  !       AND THE SUPERVISOR. THE ARGUMENTS FOR THE "HEADER" MACRO ARE: PROGRAM
1546  !       NAME, REV, PATCH, LONGEST TEST TIME, TYPE (WHERE 0 = SEQUENTIAL
1547  !       DIAGNOSTIC, 1 = EXERCISER), AND, OPTIONALLY, THE PROCESSOR PRIORITY TO
1548  !       BE SET WHEN STARTING THE DIAGNOSTIC.
1549  !-
1550
1551  HEADER (%ASCII' CZRCD', %ASCII'A', %ASCII'0', 65535, 1, PRI00);
1552
1553  !+
1554  !       THE DISPATCH TABLE CONTAINS THE STARTING ADDRESS OF EACH TEST. IT
1555  !       IS USED BY THE SUPERVISOR TO DISPATCH TO EACH TEST. THE LITERAL
1556  !       "DS$NBR_OF_TESTS" REPRESENTS THE NUMBER OF TESTS (1) IN THIS PROGRAM.
1557  !-
1558
1559  DISPATCH (DS$NBR_OF_TESTS);
```

```
;       1560  %SBTTL 'GLOBAL DATA SECTION'
;       1561
;       1562  !+
;       1563  !       THE GLOBAL DATA SECTION CONTAINS ALL DYNAMICALLY-MODIFIED DATA.
;       1564  !-
;       1565
;       1566  PSECT GLOBAL = $FFF$ (READ, NOWRITE, EXECUTE, LOCAL, CONCATENATE);
;       1567
;       1568  GLOBAL
;       1569      PATCH : VECTOR [100, WORD],                     ! PATCH AREA
;       1570      CPT : VECTOR [MAX_UNITS, BYTE],
;       1571  !       CURRENT PASS TESTING (YES / NO) PER UNIT
;       1572      CST : BLOCKVECTOR [MAX_CTLR, CST_LEN, WORD] FIELD (C_FIELDS),
;       1573  !       RUN-TIME CONTROLLER STATUS TABLES
;       1574      CST_ADDR : REF BLOCK [CST_LEN, WORD] FIELD (C_FIELDS),
;       1575  !       CONTROLLER STATUS TABLE ADDRESS OF "CURRENT" CONTROLLER
;       1576      DCT : BLOCKVECTOR [MAX_CTLR, DCT_LEN, WORD] FIELD (DC_FIELDS),
;       1577  !       DRIVER CONTROLLER TABLES
;       1578      DCT_ADDR : REF BLOCK [DCT_LEN, WORD] FIELD (DC_FIELDS),
;       1579  !       ADDRESS OF "CURRENT" DRIVER CONTROLLER TABLE
;       1580      RC25_ADDR : REF RC25 FIELD (RC_REG),
;       1581  !       DEVICE ADDRESS OF "CURRENT" CONTROLLER
;       1582      IRC25_ADDR : REF RC25 FIELD (RC_REG),
;       1583  !       DEVICE ADDRESS OF INTERRUPTING CONTROLLER
;       1584      DM_COMM : BLOCKVECTOR [MAX_CTLR, DMC_LEN, WORD] FIELD (DMC_FIELDS),
;       1585  !       DM EXERCISER COMMUNICATION AREA (LINK TO FRONT PANEL TEST)
;       1586      DMC_ADDR : REF BLOCK [DMC_LEN, WORD] FIELD (DMC_FIELDS),
;       1587  !       ADDRESS OF CURRENT CONTROLLER'S DM EXERCISER COMMUNICATION AREA
;       1588      RP_SAVE : VECTOR [MAX_CTLR * RPS_LEN, BYTE, SIGNED],
;       1589  !       RETURN PACKET SAVE AREA
;       1590      RPS_X1 : WORD,                                  ! STARTING INDEX OF CURRENT CONTROLLER'S RP_SAVE AREA
;       1591      RPS_X2 : WORD,                                  ! ENDING INDEX OF CURRENT CONTROLLER'S RP_SAVE AREA
;       1592      OUTC_LIST : VECTOR [MAX_CTLR * OUTC_CNT, BYTE, SIGNED],
;       1593  !       OUTSTANDING COMMAND LIST (CONTAINS MSCP ENVELOPE INDECES)
;       1594      OUTC_TIMR : VECTOR [MAX_CTLR * OUTC_CNT, WORD],
;       1595  !       OUTSTANDING COMMAND TIMERS
;       1596      OCL_X1 : WORD,
;       1597  !       STARTING INDEX OF CURRENT CONTROLLER'S OUTSTANDING COMMAND AREA
;       1598      OCL_X2 : WORD,
;       1599  !       ENDING INDEX OF CURRENT CONTROLLER'S OUTSTANDING COMMAND AREA
;       1600      TALLY : VECTOR [MAX_UNITS * TALLY_LEN, WORD] FIELD (T_FIELDS),
;       1601  !       STATISTICS TABLES
;       1602      T_ADDR : REF BLOCK [TALLY_LEN, WORD] FIELD (T_FIELDS),
;       1603  !       ADDRESS OF STATISTICS TABLE (TALLY) FOR CURRENT UNIT
;       1604      MSCP_ENV : BLOCKVECTOR [ENV_CNT, ENV_LEN, WORD] FIELD (E_FIELDS),
;       1605  !       MSCP ENVELOPE POOL
;       1606      ENV_USE : VECTOR [ENV_CNT, BYTE, SIGNED],
;       1607  !       MSCP ENVELOPE POOL ALLOCATION TABLE
;       1608      RETPKT : BLOCKVECTOR [RP_CNT, RP_LEN, WORD] FIELD (RP_FIELDS),
;       1609  !       RETURN PACKET POOL
;       1610      RP_USE : VECTOR [RP_CNT, BYTE, SIGNED],
;       1611  !       RETURN PACKET POOL ALLOCATION TABLE
;       1612      RP_INDX : WORD,                                 ! CURRENT RETURN PACKET INDEX
;       1613      RP_ADDR : REF BLOCK [RP_LEN, WORD] FIELD (RP_FIELDS),
;       1614  !       CURRENT RETURN PACKET ADDRESS
;       1615      BUFF_DESC : BLOCKVECTOR [MAX_BUF_CNT, DESC_LEN, WORD] FIELD (BD_FIELDS),
;       1616  !       TABLE OF I/O BUFFER DESCRIPTORS
```

```
;       1617      BUFF_OWN : VECTOR [MAX_BUF_CNT, BYTE, SIGNED],
;       1618      !       I/O BUFFER OWNERSHIP (CONTROLLER NUMBER)
;       1619      IODQ : VECTOR [IODQ_LEN, BYTE],
;       1620      !       I/O DONE QUEUE - CIRCULAR QUEUE OF RETPKT INDECES
;       1621      IODQ_IN : WORD,                         ! I/O DONE QUEUE IN POINTER
;       1622      IODQ_OUT : WORD,                        ! I/O DONE QUEUE OUT POINTER
;       1623      ENTRY_REASON : BYTE,                    ! HOW CURRENT PASS WAS INVOKED
;       1624      T_FLAG : BYTE,                          ! ONE SECOND TIMING FLAG
;       1625      EOP_FLAG : BYTE,                        ! END-OF-PASS FLAG
;       1626      MEM_MGMT : BYTE,                        ! MEMORY MANAGEMENT FLAG
;       1627      IIP_FLAG : BYTE,                        ! INITIALIZATION-IN-PROGRESS FLAG
;       1628      CCTLR : WORD,                           ! NUMBER OF "CURRENT" CONTROLLER
;       1629      CPLAT : WORD,                           ! CURRENT PLATTER ADDRESS (MSCP UNIT NUMBER)
;       1630      CUOFF : WORD,                           ! CST OFFSET FOR CURRENT UNIT
;       1631      CTLR_CNT : WORD,                        ! TOTAL NUMBER OF CONFIGURED CONTROLLERS
;       1632      DUR : VECTOR [MAX_UNITS, BYTE],         ! DROP UNIT REASON
;       1633      QIO : VECTOR [MAX_CTLR, BYTE],          ! NUMBER OF OUTSTANDING QIOS PER CONTROLLER
;       1634      MEM_SIZE : WORD,                        ! AVAILABLE MEMORY (IN WORDS) UP TO 28K
;       1635      FREE_MEM_ADDR,                          ! START OF FREE MEMORY BELOW 28K
;       1636      BUFF_SIZE : WORD,                       ! SIZE (BYTES) OF AN I/O BUFFER
;       1637      NUM_BUFF : WORD,                        ! NUMBER OF I/O BUFFERS
;       1638      CLK_TYPE : WORD,                        ! TYPE OF CLOCK ON SYSTEM
;       1639                                              !   (0 = NONE, -1 = L-CLOCK, 1 = P_CLOCK)
;       1640      CLK_HERTZ : WORD,                       ! CLOCK HERTZ RATE
;       1641      CLK_CSR,                                ! CLOCK CSR ADDRESS
;       1642      CLK_VECTOR,                             ! CLOCK VECTOR ADDRESS
;       1643      HOURS : WORD,                           ! ELAPSED TIME - HOURS,
;       1644      MINUTES : WORD,                         !       MINUTES,
;       1645      SECONDS : WORD,                         !       SECONDS,
;       1646      TICKS : WORD,                           !       TICKS
;       1647      ST_CODE : WORD,                         ! CURRENT STATUS CODE
;       1648      SB_CODE : WORD,                         ! CURRENT SUB-CODE
;       1649      STEP : WORD,                            ! CURRENT STEP IN HARD_INIT
;       1650      OF_RC : SIGNED WORD,                    ! OFFSET (0 OR 2) TO READ IP OR SA
;       1651      SA_REG : WORD,                          ! STORAGE FOR SA REGISTER READS AND WRITES
;       1652      NEX : WORD,                             ! NON-EXISTENT MEMORY TRAP INDICATOR
;       1653      CRN : WORD;                             ! COMMAND REF NUMBER OF LAST COMMAND SENT
;       1654
;       1655      !+
;       1656      !       THE ERRTBL MACRO IS REQUIRED WHETHER OR NOT THE PROGRAM USES THE
;       1657      !       "ERROR" MACRO.  THE ERRTBL MACRO EXPANDS INTO FOUR WORDS THAT ARE
;       1658      !       USED BY THE RUNTIME SERVICES DURING AN ERROR CALL: ERROR TYPE,
;       1659      !       ERROR NUMBER, ADDRESS OF ERROR MESSAGE AND ADDRESS OF MESSAGE
;       1660      !       BLOCK.  THERE MUST BE ONLY ONE ERRTBL IN ANY PROGRAM.  THIS SECTION
;       1661      !       IS NOT OPTIONAL.
;       1662      !-
;       1663
;       1664      ERRTBL;
```

```
;       1665  %SBTTL 'GLOBAL TEXT SECTION'
;       1666
;       1667  !+
;       1668  !       THE GLOBAL TEXT SECTION CONTAINS ALL MESSAGES OUTPUT TO THE OPERATOR
;       1669  !       DURING THE OPERATION OF THE EXERCISER. THIS INCLUDES HARDWARE AND
;       1670  !       SOFTWARE DIALOG PROMPTS, ERROR MESSAGES, AND DROP UNIT MESSAGES.
;       1671  !-
;       1672
;       1673  GLOBAL BIND
;       1674  !
;       1675  !****** HARDWARE DIALOG
;       1676  !
;       1677      HWQ1 = UPLIT (%ASCIZ'IP ADDRESS'),
;       1678      HWQ2 = UPLIT (%ASCIZ'VECTOR'),
;       1679      HWQ3 = UPLIT (%ASCIZ'BR LEVEL'),
;       1680      HWQ4 = UPLIT (%ASCIZ'PLATTER ADDRESS (UNIT PLUG)'),
;       1681      HWQ5 = UPLIT (%ASCIZ'ALLOW WRITES TO CUSTOMER DATA AREA ON THIS PLATTER'),
;       1682      HWQ6 = UPLIT (%ASCIZ'** WARNING - CUSTOMER DATA AREA MAY BE OVERWRITTEN! ... CONFIRM'),
;       1683  !
;       1684  !****** SOFTWARE DIALOG
;       1685  !
;       1686      SWQ1 = UPLIT (%ASCIZ'ERROR LIMIT (0 FOR NO LIMIT)'),
;       1687      SWQ2 = UPLIT (%ASCIZ'TRANSFER LIMIT IN MEGABYTES (0 FOR NO LIMIT)'),
;       1688      SWQ3 = UPLIT (%ASCIZ'SUPPRESS PRINTING ERROR LOG MESSAGES'),
;       1689      SWQ4 = UPLIT (%ASCIZ'RUN DM EXERCISER INSTEAD OF MULTI-DRIVE SUBTEST'),
;       1690      SWQ5 = UPLIT (%ASCIZ'RANDOM SEEK MODE'),
;       1691      SWQ6 = UPLIT (%ASCIZ'STARTING TRACK'),
;       1692      SWQ7 = UPLIT (%ASCIZ'ENDING TRACK'),
;       1693      SWQ8 = UPLIT (%ASCIZ'READ-COMPARES PERFORMED AT THE CONTROLLER'),
;       1694      SWQ9 = UPLIT (%ASCIZ'WRITE ONLY'),
;       1695      SWQ10 = UPLIT (%ASCIZ'WRITE-COMPARES PERFORMED AT THE CONTROLLER'),
;       1696      SWQ11 = UPLIT (%ASCIZ'CHECK ALL WRITES AT HOST BY READING'),
;       1697      SWQ12 = UPLIT (%ASCIZ'USER-DEFINED DATA PATTERN'),
;       1698      SWQ13 = UPLIT (%ASCIZ'SELECT PRE-DEFINED DATA PATTERN (0 FOR SEQUENTIAL SELECTION)'),
;       1699      SWQ14 = UPLIT (%ASCIZ'NUMBER OF WORDS IN DATA PATTERN (16 MAXIMUM)'),
;       1700      SWQ15 = UPLIT (%ASCIZ'PATTERN VALUE'),
;       1701      SWM1 = UPLIT (%ASCIZ'THE REMAINING QUESTIONS ONLY APPLY TO UNPROTECTED PLATTERS.'),
;       1702      NULL = UPLIT (%ASCIZ''),    ! ADDED TO COVER DRS BUG (NEXT MESSAGE ALSO PRINTED)
;       1703  !
;       1704  !****** INFORMATION MESSAGES (IF ATTENDED MODE, THEN PRINTF)
;       1705  !
;       1706      MSG_01 = UPLIT (%ASCIZ'%APOWER DELAY - WAITING%N'),
;       1707      MSG_02 = UPLIT (%ASCIZ'%N%AABOUT TO VERIFY VECTOR %O3%A(O) FOR DEVICE %O6%A(O) ... '),
;       1708      MSG_03 = UPLIT (%ASCIZ'%ACOMPLETED.%N'),
;       1709      MSG_04 = UPLIT (%ASCIZ'%N%AINIT SUBTEST START%N'),
;       1710      MSG_05 = UPLIT (%ASCIZ'%N%AMULTI-DRIVE SUBTEST START%N'),
;       1711      MSG_06 = UPLIT (%ASCIZ'%N%ADM EXERCISER SUBTEST START%N'),
;       1712      MSG_07 = UPLIT (%ASCIZ'%AUNIT %D2%A. - TRANSFER LIMIT REACHED%N'),
;       1713      MSG_08 = UPLIT (%ASCIZ'%D5%A. BLOCKS TRANSFERRED ON UNIT %D2%A. (PLATTER %D3%A.)%N'),
;       1714  !
;       1715  !****** CONFIGURATION ERROR MESSAGES (IF ATTENDED MODE, THEN PRINTF)
;       1716  !
;       1717      CER_01 = UPLIT (%ASCIZ'%N%ADUPLICATE PLATTER ADDRESS %D3%A. AT IP %O6%A(O)'),
;       1718      CER_02 = UPLIT (%ASCIZ'%N%AALREADY 4 UNITS AT IP %O6%A(O)'),
;       1719      CER_03 = UPLIT (%ASCIZ'%N%AMORE THAN %D1%A DIFFERENT IP ADDRESSSES'),
;       1720  !
;       1721  !****** DROP UNIT MESSAGES
```

```
;         1722  !
;         1723      DUM_00 = UPLIT (%ASCIZ'%AUNIT %D2%A. DROPPED - '),
;         1724      DUM_UC = UPLIT (%ASCIZ'%AUSER COMMAND%N'),
;         1725      DUM_CE = UPLIT (%ASCIZ'%ACONFIGURATION ERROR%N'),
;         1726      DUM_IE = UPLIT (%ASCIZ'%AINIT ERROR%N'),
;         1727      DUM_HE = UPLIT (%ASCIZ'%AHARD ERROR LIMIT REACHED%N'),
;         1728      DUM_UE = UPLIT (%ASCIZ'%AUNRECOVERABLE ERROR%N'),
;         1729  !
;         1730  !****** GENERAL ERROR MESSAGES
;         1731  !
;         1732  !       SYSTEM FATAL (ERRSF)
;         1733  !
;         1734      EGS_01 = UPLIT (%ASCIZ'TOO MANY UNITS'),
;         1735      EGS_02 = UPLIT (%ASCIZ'NEITHER P NOR L CLOCK WAS FOUND ON THE SYSTEM'),
;         1736  !
;         1737  !       DEVICE FATAL (ERRDF)
;         1738  !
;         1739      EGD_10 = UPLIT (%ASCIZ'REGISTER EXISTENCE TEST FAILED'),
;         1740      EGD_11 = UPLIT (%ASCIZ'VECTOR TEST FAILED'),
;         1741      EGD_12 = UPLIT (%ASCIZ'BR LEVEL TEST FAILED'),
;         1742      EGD_13 = UPLIT (%ASCIZ'INIT SEQUENCE FAILED'),
;         1743      EGD_14 = UPLIT (%ASCIZ'FATAL PORT / CONTROLLER ERROR'),
;         1744      EGD_15 = UPLIT (%ASCIZ'MESSAGE RESPONSE TIMEOUT'),
;         1745      EGD_16 = UPLIT (%ASCIZ'ONLINE FAILED'),
;         1746      EGD_17 = UPLIT (%ASCIZ'WRITE-PROTECT CONFLICT'),
;         1747      EGD_18 = UPLIT (%ASCIZ'ACCESS FAILED'),
;         1748      EGD_19 = UPLIT (%ASCIZ'FATAL I/O ERROR'),
;         1749      EGD_20 = UPLIT (%ASCIZ'CONTROLLER TIMEOUT'),
;         1750      EGD_21 = UPLIT (%ASCIZ'DUP COMMAND FAILED'),
;         1751      EGD_22 = UPLIT (%ASCIZ'DM EXERCISER TIMEOUT'),
;         1752  !
;         1753  !       HARD (ERRHRD)
;         1754  !
;         1755      EGH_30 = UPLIT (%ASCIZ'I/O REQUEST FAILED'),
;         1756  !
;         1757  !****** BASIC ERROR MESSAGES (PRINTB)
;         1758  !
;         1759  !       SYSTEM FATAL (ERRSF)
;         1760  !
;         1761      EBS_01 = UPLIT (%ASCIZ'%AMORE THAN %D2%A. UNITS SPECIFIED%N'),
;         1762  !
;         1763  !       DEVICE FATAL (ERRDF)
;         1764  !
;         1765      EBD_10 = UPLIT (%ASCIZ'%ANO RESPONSE AT ADDRESS %O6%A(0)%N'),
;         1766      EBD_12 = UPLIT (%ASCIZ'%AINCORRECT BR LEVEL GIVEN FOR DEVICE %O6%A(0)%N'),
;         1767      EBD_13 = UPLIT (%ASCIZ'%ASTEP %D1%A READ ERROR ON DEVICE %O6%A(0)%N'),
;         1768      EBD_14 = UPLIT (%ASCIZ'%AERROR CODE RECEIVED IN SA REGISTER OF DEVICE %O6%A(0)%N'),
;         1769      EBD_15 = UPLIT (%ASCIZ'%AFAILED TO RECEIVE END MESSAGE FROM DEVICE %O6%A(0)%N'),
;         1770      EBD_16 = UPLIT (%ASCIZ'%AERROR IN RESPONSE TO ONLINE COMMAND FOR PLATTER %D3%A.%N'),
;         1771      EBD_17 = UPLIT (%ASCIZ'%APLATTER %D3%A. IS SW WRITE-ENABLED BUT HW WRITE-PROTECTED%N'),
;         1772      EBD_18 = UPLIT (%ASCIZ'%AACCESS FAILED ON PLATTER %D3%A.%N'),
;         1773      EBD_19 = UPLIT (%ASCIZ'%APLATTER %D3%A. WENT OFFLINE%N'),
;         1774      EBD_20 = UPLIT (%ASCIZ'%ADEVICE %O6%A(0) NOT PROCESSING COMMAND PACKETS%N'),
;         1775      EBD_21 = UPLIT (%ASCIZ'%AMESSAGE REJECTED BY DUP SERVER ON DEVICE %O6%A(0)%N'),
;         1776      EBD_22 = UPLIT (%ASCIZ'%ANO RESPONSE FROM FRONT PANEL TEST EXECUTING IN DEVICE %O6%A(0)%N'),
;         1777  !
;         1778  !       HARD (ERRHRD) - MAINLY BASED ON MSCP STATUS CODES
```

```
;          1780      STC_00 = UPLIT (%ASCIZ'%ASUCCESS%N'),
;          1781      STC_01 = UPLIT (%ASCIZ'%AINVALID COMMAND%N'),
;          1782      STC_02 = UPLIT (%ASCIZ'%ACOMMAND ABORTED%N'),
;          1783      STC_03 = UPLIT (%ASCIZ'%AUNIT-OFFLINE%N'),
;          1784      STC_04 = UPLIT (%ASCIZ'%AUNIT-AVAILABLE%N'),
;          1785      STC_05 = UPLIT (%ASCIZ'%AMEDIA FORMAT ERROR%N'),
;          1786      STC_06 = UPLIT (%ASCIZ'%AWRITE-PROTECTED%N'),
;          1787      STC_07 = UPLIT (%ASCIZ'%ADEVICE COMPARE ERROR%N'),
;          1788      STC_08 = UPLIT (%ASCIZ'%ADATA ERROR%N'),
;          1789      STC_09 = UPLIT (%ASCIZ'%AHOST BUFFER ACCESS ERROR%N'),
;          1790      STC_10 = UPLIT (%ASCIZ'%ACONTROLLER ERROR%N'),
;          1791      STC_11 = UPLIT (%ASCIZ'%ADRIVE ERROR%N'),
;          1793      EBH_31 = STC_01,
;          1794      EBH_32 = STC_02,
;          1795      EBH_34 = STC_04,
;          1796      EBH_35 = STC_05,
;          1797      EBH_36 = STC_06,
;          1798      EBH_37 = STC_07,
;          1799      EBH_38 = STC_08,
;          1800      EBH_39 = STC_09,
;          1801      EBH_40 = STC_10,
;          1802      EBH_41 = STC_11,
;          1803      EBH_42 = UPLIT (%ASCIZ'%AHOST-DETECTED WRITE-COMPARE ERROR%N'),
;          1804      EBH_43 = UPLIT (%ASCIZ'%AFAILED TO RECEIVE END MESSAGE FOR I/O COMMAND%N'),
;          1805  !
;          1806  !****** EXTENDED ERROR (PRINTX) AND ERROR LOG (PRINTF) MESSAGES
;          1807  !
;          1808      EX_SA = UPLIT (%ASCIZ'%A    SA: %O6%A(O)%N'),
;          1809      EX_CRN = UPLIT (%ASCIZ'%A    CMD REF NUM: %O6%A(O)%N'),
;          1810      EX_SC = UPLIT (%ASCIZ'%A    STATUS CODE: %O2%A(O)%N'),
;          1811      EX_DSC = UPLIT (%ASCIZ'%A    DUP STATUS CODE: %D1%A.%N'),
;          1812      EX_SB = UPLIT (%ASCIZ'%A    SUB-CODE: %O4%A(O)%N'),
;          1813      EX_CMD = UPLIT (%ASCIZ'%A    COMMAND: '),
;          1814      EX_SCC = UPLIT (%ASCIZ'%ASET CTLR CHAR'),
;          1815      EX_ONL = UPLIT (%ASCIZ'%AONLINE'),
;          1816      EX_ESP = UPLIT (%ASCIZ'%AEXECUTE SUPPLIED PROGRAM'),
;          1817      EX_SND = UPLIT (%ASCIZ'%ASEND DATA'),
;          1818      EX_RD = UPLIT (%ASCIZ'%AREAD'),
;          1819      EX_WRT = UPLIT (%ASCIZ'%AWRITE'),
;          1820      EX_CMP = UPLIT (%ASCIZ'%A-COMPARE%N'),
;          1821      EX_BB = UPLIT (%ASCIZ'%A    BAD BLOCK REPORTED: %D5%A.%N'),
;          1822      EX_LBN = UPLIT (%ASCIZ'%A    LBN: %D5%A.%N'),
;          1823      EX_CBC = UPLIT (%ASCIZ'%A    BYTE COUNT IN COMMAND: %D5%A.%N'),
;          1824      EX_BC = UPLIT (%ASCIZ'%A    ACTUAL # OF BYTES TRANSFERRED: %D5%A.%N'),
;          1825      EX_BD = UPLIT (%ASCIZ'%A    I/O BUFFER DESCRIPTOR:%O7%A(O)%O7%A(O)%N'),
;          1826      EX_EL = UPLIT (%ASCIZ'%N%AERROR LOG MESSAGE RECEIVED:%N'),
;          1827      EX_PA = UPLIT (%ASCIZ'%A    PLATTER: %D3%A.%N'),
;          1828      EX_FMT = UPLIT (%ASCIZ'%A    FORMAT: '),
;          1829      EX_EVC = UPLIT (%ASCIZ'%A    EVENT CODE: '),
;          1830      EX_HMA = UPLIT (%ASCIZ'%A    HOST MEM ADDR:%O7%A(O)%O7%A(O)%N'),
;          1831      EX_03 = UPLIT (%ASCIZ'%O3%A(O)%N'),
;          1832  !
;          1833  !****** MISCELLANEOUS
;          1834  !
;          1835      ETIME = UPLIT (%ASCIZ'%D2%A:%D2%A:%D2%A  '),
;          1836      PLATT = UPLIT (%ASCIZ'%APLATTER %D3%A. - '),
;          1837      CRLF = UPLIT (%ASCIZ'%N');
```

```
;       1838  %SBTTL 'DEFAULT HARDWARE P-TABLE'
;       1839
;       1840  !+
;       1841  !       THE DEFAULT HARDWARE P-TABLE CONTAINS DEFAULT VALUES OF THE TEST-DEVICE
;       1842  !       PARAMETERS.  THE STRUCTURE OF THIS TABLE IS IDENTICAL TO THE STRUCTURE
;       1843  !       OF THE ACTUAL HARDWARE P-TABLES, WHICH RESIDE IN SUPERVISOR SPACE, AND
;       1844  !       IS USED AS A ''TEMPLATE'' FOR BUILDING THE P-TABLES.
;       1845  !-
;       1846
;       1847  BGNHW (DFPTBL);
;       1848
;       1849  GLOBAL
;       1850
;       1851      HWPT_IP_ADDR : WORD INITIAL (%O'172150'),    ! IP ADDRESS
;       1852      HWPT_VECTOR : WORD INITIAL (%O'154'),        ! VECTOR ADDRESS
;       1853      HWPT_BR_LEVEL : WORD INITIAL (5),            ! BR LEVEL
;       1854      HWPT_PLAT : WORD INITIAL (0);                ! PLATTER ADDR, PROTECTON BIT
;       1855
;       1856  ENDHW;
```

```
:          1857  %SBTTL 'SOFTWARE P-TABLE'
:          1858
:          1859  !+
:          1860  !       THE SOFTWARE P-TABLE CONTAINS VARIOUS DATA USED BY THE PROGRAM AS
:          1861  !       OPERATIONAL PARAMETERS.  THESE PARAMETERS ARE SET UP AT ASSEMBLY TIME
:          1862  !       AND MAY BE VARIED BY THE OPERATOR AT RUN TIME.
:          1863  !-
:          1864
:          1865  BGNSW (SFPTBL);
:          1866
:          1867  GLOBAL
:          1868
:          1869      SWP_ERROR : WORD INITIAL (32),              ! HARD ERROR LIMIT FOR DROPPING UNIT
:          1870      SWP_XFER : WORD INITIAL (2),                ! TRANSFER LIMIT PER UNIT PER PASS
:          1871      SWP_STRACK : WORD INITIAL (0),              ! STARTING TRACK
:          1872      SWP_ETRACK : WORD INITIAL (MAX_TRACK),      ! ENDING TRACK
:          1873      SWP_FLAGS : BYTE INITIAL (%O'55'),          ! FLAGS (SEE DOCUMENTATION)
:          1874      SWP_DPAT : BYTE INITIAL (0),                ! DATA PATTERN NUMBER
:          1875      SWP_UCNT : WORD INITIAL (MAX_UDP_CNT),      ! USER DATA PATTERN COUNT
:          1876      SWP_UDPAT : VECTOR [MAX_UDP_CNT, WORD];     ! USER DATA PATTERN
:          1877
:          1878  ENDSW;
```

```
;       1879  %SBTTL 'PROTECTION TABLE'
;       1880
;       1881  !+
;       1882  !       THIS TABLE IS USED BY THE RUNTIME SERVICES TO PROTECT THE LOAD MEDIA.
;       1883  !               1ST ARG = BYTE OFFSET INTO P-TABLE FOR CSR ADDRESS
;       1884  !               2ND ARG = BYTE OFFSET INTO P-TABLE FOR MASSBUS ADDRESS
;       1885  !               3RD ARG = BYTE OFFSET INTO P-TABLE FOR DRIVE NUMBER
;       1886  !       BYTE OFFSET REFERS TO THE NUMBER OF BYTES FROM THE BEGINNING OF A
;       1887  !       PTABLE ENTRY TO THE ITEM IN QUESTION. IF THE PARTICULAR ITEM DOES NOT
;       1888  !       APPLY, THEN ENTRY IS SET TO -1.  WHEN THE RUNTIME SERVICES EXECUTES A
;       1889  !       GPHARD, IT USES THESE OFFSETS (IF NOT SET TO -1) TO GET THE ITEMS AND
;       1890  !       COMPARE WITH THOSE SAVED IN THE XXDP+ MONITOR.  IF THE UNIT BEING
;       1891  !       REQUESTED MATCHES THE LOAD DEVICE, THEN THE RUNTIME SERVICES RETURN AN
;       1892  !       INCOMPLETE FLAG ON THE GPHARD.
;       1893  !-
;       1894
;       1895  BGNPROT (-1, -1, -1);
;       1896
;       1897  ENDPROT;
;       1898
;       1899  END
;       1900
;       1901  ELUDOM


                                        .TITLE  CZRCD1 CZRCDA0 RC25 DISK EXERCISER
                                        .IDENT  /V01.0/
                                        .ENABL  AMA


000000                                  .PSECT  $CODE$,  RO
000000    040   103   132     L$NAME::.ASCII  / CZ/
000003    122   103   104             .ASCII  /RCD/
000006    000                         .BYTE   0
000007    000                         .BYTE   0
000010                          L$REV::
000010    101                         .ASCII  /A/
000011    060                         .ASCII  /0/
000012  000000G                 L$UNIT::.WORD   T$PTHV
000014  177777                  L$TIML::.WORD   -1
000016  000000G                 L$HPCP::.WORD   L$HARD
000020  000000G                 L$SPCP::.WORD   L$SOFT
000022  006450'                 L$HPTP::.WORD   L$HW
000024  006464'                 L$SPTP::.WORD   L$SW
000026  000000G                 L$LADP::.WORD   L$LAST
000030  000000                  L$STA:: .WORD   0
000032  000000                  L$CO::  .WORD   0
000034  000001                  L$DTYP::.WORD   1
000036  000000                  L$APT:: .WORD   0
000040  000124'                 L$DTP:: .WORD   L$DISPATCH
000042  000000                  L$PRIO::.WORD   0
000044  000000                  L$ENVI::.WORD   0
000046  000000                  L$EXP1::.WORD   0
000050                          L$MREV::
000050    003                         .BYTE   3
000051    003                         .BYTE   3
000052  000000                  L$EF::  .WORD   0
```

```
000054  000000                          .WORD   0
000056  000000                  L$SPC:: .WORD   0
000060  000000G                 L$DEVP::.WORD   L$DVTYP
000062  000000G                 L$REPP::.WORD   L$RPT
000064  000000                  L$EXP4::.WORD   0
000066  000000                  L$EXP5::.WORD   0
000070  000000G                 L$AUT:: .WORD   L$AU
000072  000000G                 L$DUT:: .WORD   L$DU
000074  000000                  L$LUN:: .WORD   0
000076  000000G                 L$DESP::.WORD   L$DESC
000100  104035                  L$LOAD::.WORD   -73743
000102  000126'                 L$ETP:: .WORD   L$ERRTBL
000104  000000G                 L$ICP:: .WORD   L$INIT
000106  000000G                 L$CCP:: .WORD   L$CLEAN
000110  000000G                 L$ACP:: .WORD   L$AUTO
000112  006542'                 L$PRT:: .WORD   L$PROT
000114  000000                  L$TEST::.WORD   0
000116  000000                  L$DLY:: .WORD   0
000120  000000                  L$HIME::.WORD   0
000122  000001                  D$PCNT::.WORD   1
000124  000000G                 L$DISPATCH::
                                        .WORD   T1
000126                          ERRTYP::.BLKW   1
000130                          ERRNBR::.BLKW   1
000132                          ERRMSG::.BLKW   1
000134                          ERRBLK::.BLKW   1
000136  111     120     040     P.AAA:  .ASCII  /IF /
000141  101     104     104             .ASCII  /ADD/
000144  122     105     123             .ASCII  /RES/
000147  123     000     000             .ASCII  /S/<00><00>
000152  126     105     103     P.AAB:  .ASCII  /VEC/
000155  124     117     122             .ASCII  /TOR/
000160  000     000                     .ASCII  <00><00>
000162  102     122     040     P.AAC:  .ASCII  /BR /
000165  114     105     126             .ASCII  /LEV/
000170  105     114     000             .ASCII  /EL/<00>
000173  000                             .ASCII  <00>
000174  120     114     101     P.AAD:  .ASCII  /PLA/
000177  124     124     105             .ASCII  /TTE/
000202  122     040     101             .ASCII  /R A/
000205  104     104     122             .ASCII  /DDR/
000210  105     123     123             .ASCII  /ESS/
000213  040     050     125             .ASCII  / (U/
000216  116     111     124             .ASCII  /NIT/
000221  040     120     114             .ASCII  / PL/
000224  125     107     051             .ASCII  /UG)/
000227  000                             .ASCII  <00>
000230  101     114     114     P.AAE:  .ASCII  /ALL/
000233  117     127     040             .ASCII  /OW /
000236  127     122     111             .ASCII  /WRI/
000241  124     105     123             .ASCII  /TES/
000244  040     124     117             .ASCII  / TO/
000247  040     103     125             .ASCII  / CU/
000252  123     124     117             .ASCII  /STO/
000255  115     105     122             .ASCII  /MER/
000260  040     104     101             .ASCII  / DA/
000263  124     101     040             .ASCII  /TA /
```

```
000266  101  122  105              .ASCII  /ARE/
000271  101  040  117              .ASCII  /A O/
000274  116  040  124              .ASCII  /N T/
000277  110  111  123              .ASCII  /HIS/
000302  040  120  114              .ASCII  / PL/
000305  101  124  124              .ASCII  /ATT/
000310  105  122  000              .ASCII  /ER/<00>
000313  000                       .ASCII  <00>
000314  052  052  040  P.AAF:      .ASCII  /** /
000317  127  101  122              .ASCII  /WAR/
000322  116  111  116              .ASCII  /NIN/
000325  107  040  055              .ASCII  /G -/
000330  040  103  125              .ASCII  / CU/
000333  123  124  117              .ASCII  /STO/
000336  115  105  122              .ASCII  /MER/
000341  040  104  101              .ASCII  / DA/
000344  124  101  040              .ASCII  /TA /
000347  101  122  105              .ASCII  /ARE/
000352  101  040  115              .ASCII  /A M/
000355  101  131  040              .ASCII  /AY /
000360  102  105  040              .ASCII  /BE /
000363  117  126  105              .ASCII  /OVE/
000366  122  127  122              .ASCII  /RWR/
000371  111  124  124              .ASCII  /ITT/
000374  105  116  041              .ASCII  /EN!/
000377  040  056  056              .ASCII  / ../
000402  056  040  103              .ASCII  /. C/
000405  117  116  106              .ASCII  /ONF/
000410  111  122  115              .ASCII  /IRM/
000413  000                       .ASCII  <00>
000414  105  122  122  P.AAG:      .ASCII  /ERR/
000417  117  122  040              .ASCII  /OR /
000422  114  111  115              .ASCII  /LIM/
000425  111  124  040              .ASCII  /IT /
000430  050  060  040              .ASCII  /(0 /
000433  106  117  122              .ASCII  /FOR/
000436  040  116  117              .ASCII  / NO/
000441  040  114  111              .ASCII  / LI/
000444  115  111  124              .ASCII  /MIT/
000447  051  000  000              .ASCII  /)/<00><00>
000452  124  122  101  P.AAH:      .ASCII  /TRA/
000455  116  123  106              .ASCII  /NSF/
000460  105  122  040              .ASCII  /ER /
000463  114  111  115              .ASCII  /LIM/
000466  111  124  040              .ASCII  /IT /
000471  111  116  040              .ASCII  /IN /
000474  115  105  107              .ASCII  /MEG/
000477  101  102  131              .ASCII  /ABY/
000502  124  105  123              .ASCII  /TES/
000505  040  050  060              .ASCII  / (0/
000510  040  106  117              .ASCII  / FO/
000513  122  040  116              .ASCII  /R N/
000516  117  040  114              .ASCII  /O L/
000521  111  115  111              .ASCII  /IMI/
000524  124  051  000              .ASCII  /T)/<00>
000527  000                       .ASCII  <00>
000530  123  125  120  P.AAI:      .ASCII  /SUP/
```

```
000533  120  122  105                  .ASCII  /PRE/
000536  123  123  040                  .ASCII  /SS /
000541  120  122  111                  .ASCII  /PRI/
000544  116  124  111                  .ASCII  /NTI/
000547  116  107  040                  .ASCII  /NG /
000552  105  122  122                  .ASCII  /ERR/
000555  117  122  040                  .ASCII  /OR /
000560  114  117  107                  .ASCII  /LOG/
000563  040  115  105                  .ASCII  / ME/
000566  123  123  101                  .ASCII  /SSA/
000571  107  105  123                  .ASCII  /GES/
000574  000  000                       .ASCII  <00><00>
000576  122  125  116        P.AAJ:    .ASCII  /RUN/
000601  040  104  115                  .ASCII  / DM/
000604  040  105  130                  .ASCII  / EX/
000607  105  122  103                  .ASCII  /ERC/
000612  111  123  105                  .ASCII  /ISE/
000615  122  040  111                  .ASCII  /R I/
000620  116  123  124                  .ASCII  /NST/
000623  105  101  104                  .ASCII  /EAD/
000626  040  117  106                  .ASCII  / OF/
000631  040  115  125                  .ASCII  / MU/
000634  114  124  111                  .ASCII  /LTI/
000637  055  104  122                  .ASCII  /-DR/
000642  111  126  105                  .ASCII  /IVE/
000645  040  123  125                  .ASCII  / SU/
000650  102  124  105                  .ASCII  /BTE/
000653  123  124  000                  .ASCII  /ST/<00>
000656  122  101  116        P.AAK:    .ASCII  /RAN/
000661  104  117  115                  .ASCII  /DOM/
000664  040  123  105                  .ASCII  / SE/
000667  105  113  040                  .ASCII  /EK /
000672  115  117  104                  .ASCII  /MOD/
000675  105  000  000                  .ASCII  /E/<00><00>
000700  123  124  101        P.AAL:    .ASCII  /STA/
000703  122  124  111                  .ASCII  /RTI/
000706  116  107  040                  .ASCII  /NG /
000711  124  122  101                  .ASCII  /TRA/
000714  103  113  000                  .ASCII  /CK/<00>
000717  000                           .ASCII  <00>
000720  105  116  104        P.AAM:    .ASCII  /END/
000723  111  116  107                  .ASCII  /ING/
000726  040  124  122                  .ASCII  / TR/
000731  101  103  113                  .ASCII  /ACK/
000734  000  000                       .ASCII  <00><00>
000736  122  105  101        P.AAN:    .ASCII  /REA/
000741  104  055  103                  .ASCII  /D-C/
000744  117  115  120                  .ASCII  /OMP/
000747  101  122  105                  .ASCII  /ARE/
000752  123  040  120                  .ASCII  /S P/
000755  105  122  106                  .ASCII  /ERF/
000760  117  122  115                  .ASCII  /ORM/
000763  105  104  040                  .ASCII  /ED /
000766  101  124  040                  .ASCII  /AT /
000771  124  110  105                  .ASCII  /THE/
000774  040  103  117                  .ASCII  / CO/
000777  116  124  122                  .ASCII  /NTR/
```

```
001002  117  114  114                .ASCII  /OLL/
001005  105  122  000                .ASCII  /ER/<00>
001010  127  122  111        P.AAO:  .ASCII  /WRI/
001013  124  105  040                .ASCII  /TE /
001016  117  116  114                .ASCII  /ONL/
001021  131  000  000                .ASCII  /Y/<00><00>
001024  127  122  111        P.AAP:  .ASCII  /WRI/
001027  124  105  055                .ASCII  /TE-/
001032  103  117  115                .ASCII  /COM/
001035  120  101  122                .ASCII  /PAR/
001040  105  123  040                .ASCII  /ES /
001043  120  105  122                .ASCII  /PER/
001046  106  117  122                .ASCII  /FOR/
001051  115  105  104                .ASCII  /MED/
001054  040  101  124                .ASCII  / AT/
001057  040  124  110                .ASCII  / TH/
001062  105  040  103                .ASCII  /E C/
001065  117  116  124                .ASCII  /ONT/
001070  122  117  114                .ASCII  /ROL/
001073  114  105  122                .ASCII  /LER/
001076  000  000                    .ASCII  <00><00>
001100  103  110  105        P.AAQ:  .ASCII  /CHE/
001103  103  113  040                .ASCII  /CK /
001106  101  114  114                .ASCII  /ALL/
001111  040  127  122                .ASCII  / WR/
001114  111  124  105                .ASCII  /ITE/
001117  123  040  101                .ASCII  /S A/
001122  124  040  110                .ASCII  /T H/
001125  117  123  124                .ASCII  /OST/
001130  040  102  131                .ASCII  / BY/
001133  040  122  105                .ASCII  / RE/
001136  101  104  111                .ASCII  /ADI/
001141  116  107  000                .ASCII  /NG/<00>
001144  125  123  105        P.AAR:  .ASCII  /USE/
001147  122  055  104                .ASCII  /R-D/
001152  105  106  111                .ASCII  /EFI/
001155  116  105  104                .ASCII  /NED/
001160  040  104  101                .ASCII  / DA/
001163  124  101  040                .ASCII  /TA /
001166  120  101  124                .ASCII  /PAT/
001171  124  105  122                .ASCII  /TER/
001174  116  000                    .ASCII  /N/<00>
001176  123  105  114        P.AAS:  .ASCII  /SEL/
001201  105  103  124                .ASCII  /ECT/
001204  040  120  122                .ASCII  / PR/
001207  105  055  104                .ASCII  /E-D/
001212  105  106  111                .ASCII  /EFI/
001215  116  105  104                .ASCII  /NED/
001220  040  104  101                .ASCII  / DA/
001223  124  101  040                .ASCII  /TA /
001226  120  101  124                .ASCII  /PAT/
001231  124  105  122                .ASCII  /TER/
001234  116  040  050                .ASCII  /N (/
001237  060  040  106                .ASCII  /0 F/
001242  117  122  040                .ASCII  /OR /
001245  123  105  121                .ASCII  /SEQ/
001250  125  105  116                .ASCII  /UEN/
```

```
001253  124  111  101                .ASCII  /TIA/
001256  114  040  123                .ASCII  /L S/
001261  105  114  105                .ASCII  /ELE/
001264  103  124  111                .ASCII  /CTI/
001267  117  116  051                .ASCII  /ON)/
001272  000  000                     .ASCII  <00><00>
001274  116  125  115       P.AAT:   .ASCII  /NUM/
001277  102  105  122                .ASCII  /BER/
001302  040  117  106                .ASCII  / OF/
001305  040  127  117                .ASCII  / WO/
001310  122  104  123                .ASCII  /RDS/
001313  040  111  116                .ASCII  / IN/
001316  040  104  101                .ASCII  / DA/
001321  124  101  040                .ASCII  /TA /
001324  120  101  124                .ASCII  /PAT/
001327  124  105  122                .ASCII  /TER/
001332  116  040  050                .ASCII  /N (/
001335  061  066  040                .ASCII  /16 /
001340  115  101  130                .ASCII  /MAX/
001343  111  115  125                .ASCII  /IMU/
001346  115  051  000                .ASCII  /M)/<00>
001351  000                          .ASCII  <00>
001352  120  101  124       P.AAU:   .ASCII  /PAT/
001355  124  105  122                .ASCII  /TER/
001360  116  040  126                .ASCII  /N V/
001363  101  114  125                .ASCII  /ALU/
001366  105  000                     .ASCII  /E/<00>
001370  124  110  105       P.AAV:   .ASCII  /THE/
001373  040  122  105                .ASCII  / RE/
001376  115  101  111                .ASCII  /MAI/
001401  116  111  116                .ASCII  /NIN/
001404  107  040  121                .ASCII  /G Q/
001407  125  105  123                .ASCII  /UES/
001412  124  111  117                .ASCII  /TIO/
001415  116  123  040                .ASCII  /NS /
001420  117  116  114                .ASCII  /ONL/
001423  131  040  101                .ASCII  /Y A/
001426  120  120  114                .ASCII  /PPL/
001431  131  040  124                .ASCII  /Y T/
001434  117  040  125                .ASCII  /O U/
001437  116  120  122                .ASCII  /NPR/
001442  117  124  105                .ASCII  /OTE/
001445  103  124  105                .ASCII  /CTE/
001450  104  040  120                .ASCII  /D P/
001453  114  101  124                .ASCII  /LAT/
001456  124  105  122                .ASCII  /TER/
001461  123  056  000                .ASCII  /S./<00>
001464  000  000       P.AAW:   .ASCII  <00><00>
001466  045  101  120       P.AAX:   .ASCII  /%AP/
001471  117  127  105                .ASCII  /OWE/
001474  122  040  104                .ASCII  /R D/
001477  105  114  101                .ASCII  /ELA/
001502  131  040  055                .ASCII  /Y -/
001505  040  127  101                .ASCII  / WA/
001510  111  124  111                .ASCII  /ITI/
001513  116  107  045                .ASCII  /NG%/
001516  116  000                     .ASCII  /N/<00>
```

```
001520  045  116  045        P.AAY:  .ASCII  /%N%/
001523  101  101  102               .ASCII  /AAB/
001526  117  125  124               .ASCII  /OUT/
001531  040  124  117               .ASCII  / TO/
001534  040  126  105               .ASCII  / VE/
001537  122  111  106               .ASCII  /RIF/
001542  131  040  126               .ASCII  /Y V/
001545  105  103  124               .ASCII  /ECT/
001550  117  122  040               .ASCII  /OR /
001553  045  117  063               .ASCII  /%O3/
001556  045  101  050               .ASCII  /%A(/
001561  117  051  040               .ASCII  /O) /
001564  106  117  122               .ASCII  /FOR/
001567  040  104  105               .ASCII  / DE/
001572  126  111  103               .ASCII  /VIC/
001575  105  040  045               .ASCII  /E %/
001600  117  066  045               .ASCII  /O6%/
001603  101  050  117               .ASCII  /A(O/
001606  051  040  056               .ASCII  /) ./
001611  056  056  040               .ASCII  /.. /
001614  000  000                    .ASCII  <00><00>
001616  045  101  103        P.AAZ:  .ASCII  /%AC/
001621  117  115  120               .ASCII  /OMP/
001624  114  105  124               .ASCII  /LET/
001627  105  104  056               .ASCII  /ED./
001632  045  116  000               .ASCII  /%N/<00>
001635  000                        .ASCII  <00>
001636  045  116  045        P.ABA:  .ASCII  /%N%/
001641  101  111  116               .ASCII  /AIN/
001644  111  124  040               .ASCII  /IT /
001647  123  125  102               .ASCII  /SUB/
001652  124  105  123               .ASCII  /TES/
001655  124  040  123               .ASCII  /T S/
001660  124  101  122               .ASCII  /TAR/
001663  124  045  116               .ASCII  /T%N/
001666  000  000                    .ASCII  <00><00>
001670  045  116  045        P.ABB:  .ASCII  /%N%/
001673  101  115  125               .ASCII  /AMU/
001676  114  124  111               .ASCII  /LTI/
001701  055  104  122               .ASCII  /-DR/
001704  111  126  105               .ASCII  /IVE/
001707  040  123  125               .ASCII  / SU/
001712  102  124  105               .ASCII  /BTE/
001715  123  124  040               .ASCII  /ST /
001720  123  124  101               .ASCII  /STA/
001723  122  124  045               .ASCII  /RT%/
001726  116  000                    .ASCII  /N/<00>
001730  045  116  045        P.ABC:  .ASCII  /%N%/
001733  101  104  115               .ASCII  /ADM/
001736  040  105  130               .ASCII  / EX/
001741  105  122  103               .ASCII  /ERC/
001744  111  123  105               .ASCII  /ISE/
001747  122  040  123               .ASCII  /R S/
001752  125  102  124               .ASCII  /UBT/
001755  105  123  124               .ASCII  /EST/
001760  040  123  124               .ASCII  / ST/
001763  101  122  124               .ASCII  /ART/
```

```
001766  045  116  000                    .ASCII  /%N/<00>
001771  000                              .ASCII  <00>
001772  045  101  125          P.ABD:    .ASCII  /%AU/
001775  116  111  124                    .ASCII  /NIT/
002000  040  045  104                    .ASCII  / %D/
002003  062  045  101                    .ASCII  /2%A/
002006  056  040  055                    .ASCII  /. -/
002011  040  124  122                    .ASCII  / TR/
002014  101  116  123                    .ASCII  /ANS/
002017  106  105  122                    .ASCII  /FER/
002022  040  114  111                    .ASCII  / LI/
002025  115  111  124                    .ASCII  /MIT/
002030  040  122  105                    .ASCII  / RE/
002033  101  103  110                    .ASCII  /ACH/
002036  105  104  045                    .ASCII  /ED%/
002041  116  000  000                    .ASCII  /N/<00><00>
002044  045  104  065          P.ABE:    .ASCII  /%D5/
002047  045  101  056                    .ASCII  /%A./
002052  040  102  114                    .ASCII  / BL/
002055  117  103  113                    .ASCII  /OCK/
002060  123  040  124                    .ASCII  /S T/
002063  122  101  116                    .ASCII  /RAN/
002066  123  106  105                    .ASCII  /SFE/
002071  122  122  105                    .ASCII  /RRE/
002074  104  040  117                    .ASCII  /D O/
002077  116  040  125                    .ASCII  /N U/
002102  116  111  124                    .ASCII  /NIT/
002105  040  045  104                    .ASCII  / %D/
002110  062  045  101                    .ASCII  /2%A/
002113  056  040  050                    .ASCII  /. (/
002116  120  114  101                    .ASCII  /PLA/
002121  124  124  105                    .ASCII  /TTE/
002124  122  040  045                    .ASCII  /R %/
002127  104  063  045                    .ASCII  /D3%/
002132  101  056  051                    .ASCII  /A.)/
002135  045  116  000                    .ASCII  /%N/<00>
002140  045  116  045          P.ABF:    .ASCII  /%N%/
002143  101  104  125                    .ASCII  /ADU/
002146  120  114  111                    .ASCII  /PLI/
002151  103  101  124                    .ASCII  /CAT/
002154  105  040  120                    .ASCII  /E P/
002157  114  101  124                    .ASCII  /LAT/
002162  124  105  122                    .ASCII  /TER/
002165  040  101  104                    .ASCII  / AD/
002170  104  122  105                    .ASCII  /DRE/
002173  123  123  040                    .ASCII  /SS /
002176  045  104  063                    .ASCII  /%D3/
002201  045  101  056                    .ASCII  /%A./
002204  040  101  124                    .ASCII  / AT/
002207  040  111  120                    .ASCII  / IP/
002212  040  045  117                    .ASCII  / %O/
002215  066  045  101                    .ASCII  /6%A/
002220  050  117  051                    .ASCII  /(O)/
002223  000                              .ASCII  <00>
002224  045  116  045          P.ABG:    .ASCII  /%N%/
002227  101  101  114                    .ASCII  /AAL/
002232  122  105  101                    .ASCII  /REA/
```

```
002235  104  131  040                    .ASCII  /DY /
002240  064  040  125                    .ASCII  /4 U/
002243  116  111  124                    .ASCII  /NIT/
002246  123  040  101                    .ASCII  /S A/
002251  124  040  111                    .ASCII  /T I/
002254  120  040  045                    .ASCII  /P %/
002257  117  066  045                    .ASCII  /O6%/
002262  101  050  117                    .ASCII  /A(O/
002265  051  000  000                    .ASCII  /)/<00><00>
002270  045  116  045            P.ABH:  .ASCII  /%N%/
002273  101  115  117                    .ASCII  /AMO/
002276  122  105  040                    .ASCII  /RE /
002301  124  110  101                    .ASCII  /THA/
002304  116  040  045                    .ASCII  /N %/
002307  104  061  045                    .ASCII  /D1%/
002312  101  040  104                    .ASCII  /A D/
002315  111  106  106                    .ASCII  /IFF/
002320  105  122  105                    .ASCII  /ERE/
002323  116  124  040                    .ASCII  /NT /
002326  111  120  040                    .ASCII  /IP /
002331  101  104  104                    .ASCII  /ADD/
002334  122  105  123                    .ASCII  /RES/
002337  123  123  105                    .ASCII  /SSE/
002342  123  000                        .ASCII  /S/<00>
002344  045  101  125            P.ABI:  .ASCII  /%AU/
002347  116  111  124                    .ASCII  /NIT/
002352  040  045  104                    .ASCII  / %D/
002355  062  045  101                    .ASCII  /2%A/
002360  056  040  104                    .ASCII  /. D/
002363  122  117  120                    .ASCII  /ROP/
002366  120  105  104                    .ASCII  /PED/
002371  040  055  040                    .ASCII  / - /
002374  000  000                        .ASCII  <00><00>
002376  045  101  125            P.ABJ:  .ASCII  /%AU/
002401  123  105  122                    .ASCII  /SER/
002404  040  103  117                    .ASCII  / CO/
002407  115  115  101                    .ASCII  /MMA/
002412  116  104  045                    .ASCII  /ND%/
002415  116  000  000                    .ASCII  /N/<00><00>
002420  045  101  103            P.ABK:  .ASCII  /%AC/
002423  117  116  106                    .ASCII  /ONF/
002426  111  107  125                    .ASCII  /IGU/
002431  122  101  124                    .ASCII  /RAT/
002434  111  117  116                    .ASCII  /ION/
002437  040  105  122                    .ASCII  / ER/
002442  122  117  122                    .ASCII  /ROR/
002445  045  116  000                    .ASCII  /%N/<00>
002450  045  101  111            P.ABL:  .ASCII  /%AI/
002453  116  111  124                    .ASCII  /NIT/
002456  040  105  122                    .ASCII  / ER/
002461  122  117  122                    .ASCII  /ROR/
002464  045  116  000                    .ASCII  /%N/<00>
002467  000                            .ASCII  <00>
002470  045  101  110            P.ABM:  .ASCII  /%AH/
002473  101  122  104                    .ASCII  /ARD/
002476  040  105  122                    .ASCII  / ER/
002501  122  117  122                    .ASCII  /ROR/
```

```
002504  040  114  111                    .ASCII  / LI/
002507  115  111  124                    .ASCII  /MIT/
002512  040  122  105                    .ASCII  / RE/
002515  101  103  110                    .ASCII  /ACH/
002520  105  104  045                    .ASCII  /ED%/
002523  116  000  000                    .ASCII  /N/<00><00>
002526  045  101  125          P.ABN:    .ASCII  /%AU/
002531  116  122  105                    .ASCII  /NRE/
002534  103  117  126                    .ASCII  /COV/
002537  105  122  101                    .ASCII  /ERA/
002542  102  114  105                    .ASCII  /BLE/
002545  040  105  122                    .ASCII  / ER/
002550  122  117  122                    .ASCII  /ROR/
002553  045  116  000                    .ASCII  /%N/<00>
002556  124  117  117          P.ABO:    .ASCII  /TOO/
002561  040  115  101                    .ASCII  / MA/
002564  116  131  040                    .ASCII  /NY /
002567  125  116  111                    .ASCII  /UNI/
002572  124  123  000                    .ASCII  /TS/<00>
002575  000                             .ASCII  <00>
002576  116  105  111          P.ABP:    .ASCII  /NEI/
002601  124  110  105                    .ASCII  /THE/
002604  122  040  120                    .ASCII  /R P/
002607  040  116  117                    .ASCII  / NO/
002612  122  040  114                    .ASCII  /R L/
002615  040  103  114                    .ASCII  / CL/
002620  117  103  113                    .ASCII  /OCK/
002623  040  127  101                    .ASCII  / WA/
002626  123  040  106                    .ASCII  /S F/
002631  117  125  116                    .ASCII  /OUN/
002634  104  040  117                    .ASCII  /D O/
002637  116  040  124                    .ASCII  /N T/
002642  110  105  040                    .ASCII  /HE /
002645  123  131  123                    .ASCII  /SYS/
002650  124  105  115                    .ASCII  /TEM/
002653  000                             .ASCII  <00>
002654  122  105  107          P.ABQ:    .ASCII  /REG/
002657  111  123  124                    .ASCII  /IST/
002662  105  122  040                    .ASCII  /ER /
002665  105  130  111                    .ASCII  /EXI/
002670  123  124  105                    .ASCII  /STE/
002673  116  103  105                    .ASCII  /NCE/
002676  040  124  105                    .ASCII  / TE/
002701  123  124  040                    .ASCII  /ST /
002704  106  101  111                    .ASCII  /FAI/
002707  114  105  104                    .ASCII  /LED/
002712  000  000                        .ASCII  <00><00>
002714  126  105  103          P.ABR:    .ASCII  /VEC/
002717  124  117  122                    .ASCII  /TOR/
002722  040  124  105                    .ASCII  / TE/
002725  123  124  040                    .ASCII  /ST /
002730  106  101  111                    .ASCII  /FAI/
002733  114  105  104                    .ASCII  /LED/
002736  000  000                        .ASCII  <00><00>
002740  102  122  040          P.ABS:    .ASCII  /BR /
002743  114  105  126                    .ASCII  /LEV/
002746  105  114  040                    .ASCII  /EL /
```

```
002751  124  105  123                  .ASCII  /TES/
002754  124  040  106                  .ASCII  /T F/
002757  101  111  114                  .ASCII  /AIL/
002762  105  104  000                  .ASCII  /ED/<00>
002765  000                           .ASCII  <00>
002766  111  116  111          P.ABT:  .ASCII  /INI/
002771  124  040  123                  .ASCII  /T S/
002774  105  121  125                  .ASCII  /EQU/
002777  105  116  103                  .ASCII  /ENC/
003002  105  040  106                  .ASCII  /E F/
003005  101  111  114                  .ASCII  /AIL/
003010  105  104  000                  .ASCII  /ED/<00>
003013  000                           .ASCII  <00>
003014  106  101  124          P.ABU:  .ASCII  /FAT/
003017  101  114  040                  .ASCII  /AL /
003022  120  117  122                  .ASCII  /POR/
003025  124  040  057                  .ASCII  /T /<57>
003030  040  103  117                  .ASCII  / CO/
003033  116  124  122                  .ASCII  /NTR/
003036  117  114  114                  .ASCII  /OLL/
003041  105  122  040                  .ASCII  /ER /
003044  105  122  122                  .ASCII  /ERR/
003047  117  122  000                  .ASCII  /OR/<00>
003052  115  105  123          P.ABV:  .ASCII  /MES/
003055  123  101  107                  .ASCII  /SAG/
003060  105  040  122                  .ASCII  /E R/
003063  105  123  120                  .ASCII  /ESP/
003066  117  116  123                  .ASCII  /ONS/
003071  105  040  124                  .ASCII  /E T/
003074  111  115  105                  .ASCII  /IME/
003077  117  125  124                  .ASCII  /OUT/
003102  000  000                       .ASCII  <00><00>
003104  117  116  114          P.ABW:  .ASCII  /ONL/
003107  111  116  105                  .ASCII  /INE/
003112  040  106  101                  .ASCII  / FA/
003115  111  114  105                  .ASCII  /ILE/
003120  104  000                       .ASCII  /D/<00>
003122  127  122  111          P.ABX:  .ASCII  /WRI/
003125  124  105  055                  .ASCII  /TE-/
003130  120  122  117                  .ASCII  /PRO/
003133  124  105  103                  .ASCII  /TEC/
003136  124  040  103                  .ASCII  /T C/
003141  117  116  106                  .ASCII  /ONF/
003144  114  111  103                  .ASCII  /LIC/
003147  124  000  000                  .ASCII  /T/<00><00>
003152  101  103  103          P.ABY:  .ASCII  /ACC/
003155  105  123  123                  .ASCII  /ESS/
003160  040  106  101                  .ASCII  / FA/
003163  111  114  105                  .ASCII  /ILE/
003166  104  000                       .ASCII  /D/<00>
003170  106  101  124          P.ABZ:  .ASCII  /FAT/
003173  101  114  040                  .ASCII  /AL /
003176  111  057  117                  .ASCII  /I/<57>/O/
003201  040  105  122                  .ASCII  / ER/
003204  122  117  122                  .ASCII  /ROR/
003207  000                           .ASCII  <00>
003210  103  117  116          P.ACA:  .ASCII  /CON/
```

```
003213  124  122  117                .ASCII  /TRO/
003216  114  114  105                .ASCII  /LLE/
003221  122  040  124                .ASCII  /R T/
003224  111  115  105                .ASCII  /IME/
003227  117  125  124                .ASCII  /OUT/
003232  000  000                     .ASCII  <00><00>
003234  104  125  120  P.ACB:        .ASCII  /DUP/
003237  040  103  117                .ASCII  / CO/
003242  115  115  101                .ASCII  /MMA/
003245  116  104  040                .ASCII  /ND /
003250  106  101  111                .ASCII  /FAI/
003253  114  105  104                .ASCII  /LED/
003256  000  000                     .ASCII  <00><00>
003260  104  115  040  P.ACC:        .ASCII  /DM /
003263  105  130  105                .ASCII  /EXE/
003266  122  103  111                .ASCII  /RCI/
003271  123  105  122                .ASCII  /SER/
003274  040  124  111                .ASCII  / TI/
003277  115  105  117                .ASCII  /MEO/
003302  125  124  000                .ASCII  /UT/<00>
003305  000                         .ASCII  <00>
003306  111  057  117  P.ACD:        .ASCII  /I/<57>/O/
003311  040  122  105                .ASCII  / RE/
003314  121  125  105                .ASCII  /QUE/
003317  123  124  040                .ASCII  /ST /
003322  106  101  111                .ASCII  /FAI/
003325  114  105  104                .ASCII  /LED/
003330  000  000                     .ASCII  <00><00>
003332  045  101  115  P.ACE:        .ASCII  /%AM/
003335  117  122  105                .ASCII  /ORE/
003340  040  124  110                .ASCII  / TH/
003343  101  116  040                .ASCII  /AN /
003346  045  104  062                .ASCII  /%D2/
003351  045  101  056                .ASCII  /%A./
003354  040  125  116                .ASCII  / UN/
003357  111  124  123                .ASCII  /ITS/
003362  040  123  120                .ASCII  / SP/
003365  105  103  111                .ASCII  /ECI/
003370  106  111  105                .ASCII  /FIE/
003373  104  045  116                .ASCII  /D%N/
003376  000  000                     .ASCII  <00><00>
003400  045  101  116  P.ACF:        .ASCII  /%AN/
003403  117  040  122                .ASCII  /O R/
003406  105  123  120                .ASCII  /ESP/
003411  117  116  123                .ASCII  /ONS/
003414  105  040  101                .ASCII  /E A/
003417  124  040  101                .ASCII  /T A/
003422  104  104  122                .ASCII  /DDR/
003425  105  123  123                .ASCII  /ESS/
003430  040  045  117                .ASCII  / %O/
003433  066  045  101                .ASCII  /6%A/
003436  050  117  051                .ASCII  /(O)/
003441  045  116  000                .ASCII  /%N/<00>
003444  045  101  111  P.ACG:        .ASCII  /%AI/
003447  116  103  117                .ASCII  /NCO/
003452  122  122  105                .ASCII  /RRE/
003455  103  124  040                .ASCII  /CT /
```

```
003460  102  122  040              .ASCII  /BR /
003463  114  105  126              .ASCII  /LEV/
003466  105  114  040              .ASCII  /EL /
003471  107  111  126              .ASCII  /GIV/
003474  105  116  040              .ASCII  /EN /
003477  106  117  122              .ASCII  /FOR/
003502  040  104  105              .ASCII  / DE/
003505  126  111  103              .ASCII  /VIC/
003510  105  040  045              .ASCII  /E %/
003513  117  066  045              .ASCII  /O6%/
003516  101  050  117              .ASCII  /A(O/
003521  051  045  116              .ASCII  /)%N/
003524  000  000                   .ASCII  <00><00>
003526  045  101  123  P.ACH:      .ASCII  /%AS/
003531  124  105  120              .ASCII  /TEP/
003534  040  045  104              .ASCII  / %D/
003537  061  045  101              .ASCII  /1%A/
003542  040  122  105              .ASCII  / RE/
003545  101  104  040              .ASCII  /AD /
003550  105  122  122              .ASCII  /ERR/
003553  117  122  040              .ASCII  /OR /
003556  117  116  040              .ASCII  /ON /
003561  104  105  126              .ASCII  /DEV/
003564  111  103  105              .ASCII  /ICE/
003567  040  045  117              .ASCII  / %O/
003572  066  045  101              .ASCII  /6%A/
003575  050  117  051              .ASCII  /(O)/
003600  045  116  000              .ASCII  /%N/<00>
003603  000                       .ASCII  <00>
003604  045  101  105  P.ACI:      .ASCII  /%AE/
003607  122  122  117              .ASCII  /RRO/
003612  122  040  103              .ASCII  /R C/
003615  117  104  105              .ASCII  /ODE/
003620  040  122  105              .ASCII  / RE/
003623  103  105  111              .ASCII  /CEI/
003626  126  105  104              .ASCII  /VED/
003631  040  111  116              .ASCII  / IN/
003634  040  123  101              .ASCII  / SA/
003637  040  122  105              .ASCII  / RE/
003642  107  111  123              .ASCII  /GIS/
003645  124  105  122              .ASCII  /TER/
003650  040  117  106              .ASCII  / OF/
003653  040  104  105              .ASCII  / DE/
003656  126  111  103              .ASCII  /VIC/
003661  105  040  045              .ASCII  /E %/
003664  117  066  045              .ASCII  /O6%/
003667  101  050  117              .ASCII  /A(O/
003672  051  045  116              .ASCII  /)%N/
003675  000                       .ASCII  <00>
003676  045  101  106  P.ACJ:      .ASCII  /%AF/
003701  101  111  114              .ASCII  /AIL/
003704  105  104  040              .ASCII  /ED /
003707  124  117  040              .ASCII  /TO /
003712  122  105  103              .ASCII  /REC/
003715  105  111  126              .ASCII  /EIV/
003720  105  040  105              .ASCII  /E E/
003723  116  104  040              .ASCII  /ND /
```

```
003726  115  105  123                 .ASCII  /MES/
003731  123  101  107                 .ASCII  /SAG/
003734  105  040  106                 .ASCII  /E F/
003737  122  117  115                 .ASCII  /ROM/
003742  040  104  105                 .ASCII  / DE/
003745  126  111  103                 .ASCII  /VIC/
003750  105  040  045                 .ASCII  /E %/
003753  117  066  045                 .ASCII  /O6%/
003756  101  050  117                 .ASCII  /A(O/
003761  051  045  116                 .ASCII  /)%N/
003764  000  000                      .ASCII  <00><00>
003766  045  101  105       P.ACK:    .ASCII  /%AE/
003771  122  122  117                 .ASCII  /RRO/
003774  122  040  111                 .ASCII  /R I/
003777  116  040  122                 .ASCII  /N R/
004002  105  123  120                 .ASCII  /ESP/
004005  117  116  123                 .ASCII  /ONS/
004010  105  040  124                 .ASCII  /E T/
004013  117  040  117                 .ASCII  /O O/
004016  116  114  111                 .ASCII  /NLI/
004021  116  105  040                 .ASCII  /NE /
004024  103  117  115                 .ASCII  /COM/
004027  115  101  116                 .ASCII  /MAN/
004032  104  040  106                 .ASCII  /D F/
004035  117  122  040                 .ASCII  /OR /
004040  120  114  101                 .ASCII  /PLA/
004043  124  124  105                 .ASCII  /TTE/
004046  122  040  045                 .ASCII  /R %/
004051  104  063  045                 .ASCII  /D3%/
004054  101  056  045                 .ASCII  /A.%/
004057  116  000  000                 .ASCII  /N/<00><00>
004062  045  101  120       P.ACL:    .ASCII  /%AP/
004065  114  101  124                 .ASCII  /LAT/
004070  124  105  122                 .ASCII  /TER/
004073  040  045  104                 .ASCII  / %D/
004076  063  045  101                 .ASCII  /3%A/
004101  056  040  111                 .ASCII  /. I/
004104  123  040  123                 .ASCII  /S S/
004107  127  040  127                 .ASCII  /W W/
004112  122  111  124                 .ASCII  /RIT/
004115  105  055  105                 .ASCII  /E-E/
004120  116  101  102                 .ASCII  /NAB/
004123  114  105  104                 .ASCII  /LED/
004126  040  102  125                 .ASCII  / BU/
004131  124  040  110                 .ASCII  /T H/
004134  127  040  127                 .ASCII  /W W/
004137  122  111  124                 .ASCII  /RIT/
004142  105  055  120                 .ASCII  /E-P/
004145  122  117  124                 .ASCII  /ROT/
004150  105  103  124                 .ASCII  /ECT/
004153  105  104  045                 .ASCII  /ED%/
004156  116  000                      .ASCII  /N/<00>
004160  045  101  101       P.ACM:    .ASCII  /%AA/
004163  103  103  105                 .ASCII  /CCE/
004166  123  123  040                 .ASCII  /SS /
004171  106  101  111                 .ASCII  /FAI/
004174  114  105  104                 .ASCII  /LED/
```

```
004177  040  117  116                .ASCII  / ON/
004202  040  120  114                .ASCII  / PL/
004205  101  124  124                .ASCII  /ATT/
004210  105  122  040                .ASCII  /ER /
004213  045  104  063                .ASCII  /%D3/
004216  045  101  056                .ASCII  /%A./
004221  045  116  000                .ASCII  /%N/<00>
004224  045  101  120      P.ACN:    .ASCII  /%AP/
004227  114  101  124                .ASCII  /LAT/
004232  124  105  122                .ASCII  /TER/
004235  040  045  104                .ASCII  / %D/
004240  063  045  101                .ASCII  /3%A/
004243  056  040  127                .ASCII  /. W/
004246  105  116  124                .ASCII  /ENT/
004251  040  117  106                .ASCII  / OF/
004254  106  114  111                .ASCII  /FLI/
004257  116  105  045                .ASCII  /NE%/
004262  116  000                    .ASCII  /N/<00>
004264  045  101  104      P.ACO:    .ASCII  /%AD/
004267  105  126  111                .ASCII  /EVI/
004272  103  105  040                .ASCII  /CE /
004275  045  117  066                .ASCII  /%O6/
004300  045  101  050                .ASCII  /%A(/
004303  117  051  040                .ASCII  /O) /
004306  116  117  124                .ASCII  /NOT/
004311  040  120  122                .ASCII  / PR/
004314  117  103  105                .ASCII  /OCE/
004317  123  123  111                .ASCII  /SSI/
004322  116  107  040                .ASCII  /NG /
004325  103  117  115                .ASCII  /COM/
004330  115  101  116                .ASCII  /MAN/
004333  104  040  120                .ASCII  /D P/
004336  101  103  113                .ASCII  /ACK/
004341  105  124  123                .ASCII  /ETS/
004344  045  116  000                .ASCII  /%N/<00>
004347  000                         .ASCII  <00>
004350  045  101  115      P.ACP:    .ASCII  /%AM/
004353  105  123  123                .ASCII  /ESS/
004356  101  107  105                .ASCII  /AGE/
004361  040  122  105                .ASCII  / RE/
004364  112  105  103                .ASCII  /JEC/
004367  124  105  104                .ASCII  /TED/
004372  040  102  131                .ASCII  / BY/
004375  040  104  125                .ASCII  / DU/
004400  120  040  123                .ASCII  /P S/
004403  105  122  126                .ASCII  /ERV/
004406  105  122  040                .ASCII  /ER /
004411  117  116  040                .ASCII  /ON /
004414  104  105  126                .ASCII  /DEV/
004417  111  103  105                .ASCII  /ICE/
004422  040  045  117                .ASCII  / %O/
004425  066  045  101                .ASCII  /6%A/
004430  050  117  051                .ASCII  /(O)/
004433  045  116  000                .ASCII  /%N/<00>
004436  045  101  116      P.ACQ:    .ASCII  /%AN/
004441  117  040  122                .ASCII  /O R/
004444  105  123  120                .ASCII  /ESP/
```

```
004447  117     116     123             .ASCII  /ONS/
004452  105     040     106             .ASCII  /E F/
004455  122     117     115             .ASCII  /ROM/
004460  040     106     122             .ASCII  / FR/
004463  117     116     124             .ASCII  /ONT/
004466  040     120     101             .ASCII  / PA/
004471  116     105     114             .ASCII  /NEL/
004474  040     124     105             .ASCII  / TE/
004477  123     124     040             .ASCII  /ST /
004502  105     130     105             .ASCII  /EXE/
004505  103     125     124             .ASCII  /CUT/
004510  111     116     107             .ASCII  /ING/
004513  040     111     116             .ASCII  / IN/
004516  040     104     105             .ASCII  / DE/
004521  126     111     103             .ASCII  /VIC/
004524  105     040     045             .ASCII  /E %/
004527  117     066     045             .ASCII  /O6%/
004532  101     050     117             .ASCII  /A(O/
004535  051     045     116             .ASCII  /)%N/
004540  000     000                     .ASCII  <00><00>
004542  045     101     123     P.ACR:  .ASCII  /%AS/
004545  125     103     103             .ASCII  /UCC/
004550  105     123     123             .ASCII  /ESS/
004553  045     116     000             .ASCII  /%N/<00>
004556  045     101     111     P.ACS:  .ASCII  /%AI/
004561  116     126     101             .ASCII  /NVA/
004564  114     111     104             .ASCII  /LID/
004567  040     103     117             .ASCII  / CO/
004572  115     115     101             .ASCII  /MMA/
004575  116     104     045             .ASCII  /ND%/
004600  116     000                     .ASCII  /N/<00>
004602  045     101     103     P.ACT:  .ASCII  /%AC/
004605  117     115     115             .ASCII  /OMM/
004610  101     116     104             .ASCII  /AND/
004613  040     101     102             .ASCII  / AB/
004616  117     122     124             .ASCII  /ORT/
004621  105     104     045             .ASCII  /ED%/
004624  116     000                     .ASCII  /N/<00>
004626  045     101     125     P.ACU:  .ASCII  /%AU/
004631  116     111     124             .ASCII  /NIT/
004634  055     117     106             .ASCII  /-OF/
004637  106     114     111             .ASCII  /FLI/
004642  116     105     045             .ASCII  /NE%/
004645  116     000     000             .ASCII  /N/<00><00>
004650  045     101     125     P.ACV:  .ASCII  /%AU/
004653  116     111     124             .ASCII  /NIT/
004656  055     101     126             .ASCII  /-AV/
004661  101     111     114             .ASCII  /AIL/
004664  101     102     114             .ASCII  /ABL/
004667  105     045     116             .ASCII  /E%N/
004672  000     000                     .ASCII  <00><00>
004674  045     101     115     P.ACW:  .ASCII  /%AM/
004677  105     104     111             .ASCII  /EDI/
004702  101     040     106             .ASCII  /A F/
004705  117     122     115             .ASCII  /ORM/
004710  101     124     040             .ASCII  /AT /
004713  105     122     122             .ASCII  /ERR/
```

```
004716  117  122  045                .ASCII  /OR%/
004721  116  000  000                .ASCII  /N/<00><00>
004724  045  101  127  P.ACX:       .ASCII  /%AW/
004727  122  111  124                .ASCII  /RIT/
004732  105  055  120                .ASCII  /E-P/
004735  122  117  124                .ASCII  /ROT/
004740  105  103  124                .ASCII  /ECT/
004743  105  104  045                .ASCII  /ED%/
004746  116  000                    .ASCII  /N/<00>
004750  045  101  104  P.ACY:       .ASCII  /%AD/
004753  105  126  111                .ASCII  /EVI/
004756  103  105  040                .ASCII  /CE /
004761  103  117  115                .ASCII  /COM/
004764  120  101  122                .ASCII  /PAR/
004767  105  040  105                .ASCII  /E E/
004772  122  122  117                .ASCII  /RRO/
004775  122  045  116                .ASCII  /R%N/
005000  000  000                    .ASCII  <00><00>
005002  045  101  104  P.ACZ:       .ASCII  /%AD/
005005  101  124  101                .ASCII  /ATA/
005010  040  105  122                .ASCII  / ER/
005013  122  117  122                .ASCII  /ROR/
005016  045  116  000                .ASCII  /%N/<00>
005021  000                        .ASCII  <00>
005022  045  101  110  P.ADA:       .ASCII  /%AH/
005025  117  123  124                .ASCII  /OST/
005030  040  102  125                .ASCII  / BU/
005033  106  106  105                .ASCII  /FFE/
005036  122  040  101                .ASCII  /R A/
005041  103  103  105                .ASCII  /CCE/
005044  123  123  040                .ASCII  /SS /
005047  105  122  122                .ASCII  /ERR/
005052  117  122  045                .ASCII  /OR%/
005055  116  000  000                .ASCII  /N/<00><00>
005060  045  101  103  P.ADB:       .ASCII  /%AC/
005063  117  116  124                .ASCII  /ONT/
005066  122  117  114                .ASCII  /ROL/
005071  114  105  122                .ASCII  /LER/
005074  040  105  122                .ASCII  / ER/
005077  122  117  122                .ASCII  /ROR/
005102  045  116  000                .ASCII  /%N/<00>
005105  000                        .ASCII  <00>
005106  045  101  104  P.ADC:       .ASCII  /%AD/
005111  122  111  126                .ASCII  /RIV/
005114  105  040  105                .ASCII  /E E/
005117  122  122  117                .ASCII  /RRO/
005122  122  045  116                .ASCII  /R%N/
005125  000                        .ASCII  <00>
005126  045  101  110  P.ADD:       .ASCII  /%AH/
005131  117  123  124                .ASCII  /OST/
005134  055  104  105                .ASCII  /-DE/
005137  124  105  103                .ASCII  /TEC/
005142  124  105  104                .ASCII  /TED/
005145  040  127  122                .ASCII  / WR/
005150  111  124  105                .ASCII  /ITE/
005153  055  103  117                .ASCII  /-CO/
005156  115  120  101                .ASCII  /MPA/
```

```
005161  122  105  040                    .ASCII  /RE /
005164  105  122  122                    .ASCII  /ERR/
005167  117  122  045                    .ASCII  /OR%/
005172  116  000                         .ASCII  /N/<00>
005174  045  101  106          P.ADE:    .ASCII  /%AF/
005177  101  111  114                    .ASCII  /AIL/
005202  105  104  040                    .ASCII  /ED /
005205  124  117  040                    .ASCII  /TO /
005210  122  105  103                    .ASCII  /REC/
005213  105  111  126                    .ASCII  /EIV/
005216  105  040  105                    .ASCII  /E E/
005221  116  104  040                    .ASCII  /ND /
005224  115  105  123                    .ASCII  /MES/
005227  123  101  107                    .ASCII  /SAG/
005232  105  040  106                    .ASCII  /E F/
005235  117  122  040                    .ASCII  /OR /
005240  111  057  117                    .ASCII  /I/<57>/O/
005243  040  103  117                    .ASCII  / CO/
005246  115  115  101                    .ASCII  /MMA/
005251  116  104  045                    .ASCII  /ND%/
005254  116  000                         .ASCII  /N/<00>
005256  045  101  040          P.ADF:    .ASCII  /%A /
005261  040  040  040                    .ASCII  /   /
005264  123  101  072                    .ASCII  /SA:/
005267  040  045  117                    .ASCII  / %O/
005272  066  045  101                    .ASCII  /6%A/
005275  050  117  051                    .ASCII  /(O)/
005300  045  116  000                    .ASCII  /%N/<00>
005303  000                            .ASCII  <00>
005304  045  101  040          P.ADG:    .ASCII  /%A /
005307  040  040  040                    .ASCII  /   /
005312  103  115  104                    .ASCII  /CMD/
005315  040  122  105                    .ASCII  / RE/
005320  106  040  116                    .ASCII  /F N/
005323  125  115  072                    .ASCII  /UM:/
005326  040  045  117                    .ASCII  / %O/
005331  066  045  101                    .ASCII  /6%A/
005334  050  117  051                    .ASCII  /(O)/
005337  045  116  000                    .ASCII  /%N/<00>
005342  045  101  040          P.ADH:    .ASCII  /%A /
005345  040  040  040                    .ASCII  /   /
005350  123  124  101                    .ASCII  /STA/
005353  124  125  123                    .ASCII  /TUS/
005356  040  103  117                    .ASCII  / CO/
005361  104  105  072                    .ASCII  /DE:/
005364  040  045  117                    .ASCII  / %O/
005367  062  045  101                    .ASCII  /2%A/
005372  050  117  051                    .ASCII  /(O)/
005375  045  116  000                    .ASCII  /%N/<00>
005400  045  101  040          P.ADI:    .ASCII  /%A /
005403  040  040  040                    .ASCII  /   /
005406  104  125  120                    .ASCII  /DUP/
005411  040  123  124                    .ASCII  / ST/
005414  101  124  125                    .ASCII  /ATU/
005417  123  040  103                    .ASCII  /S C/
005422  117  104  105                    .ASCII  /ODE/
005425  072  040  045                    .ASCII  /: %/
```

```
005430  104  061  045            .ASCII  /D1%/
005433  101  056  045            .ASCII  /A.%/
005436  116  000                 .ASCII  /N/<00>
005440  045  101  040  P.ADJ:     .ASCII  /%A /
005443  040  040  040            .ASCII  /   /
005446  123  125  102            .ASCII  /SUB/
005451  055  103  117            .ASCII  /-CO/
005454  104  105  072            .ASCII  /DE:/
005457  040  045  117            .ASCII  / %O/
005462  064  045  101            .ASCII  /4%A/
005465  050  117  051            .ASCII  /(O)/
005470  045  116  000            .ASCII  /%N/<00>
005473  000                      .ASCII  <00>
005474  045  101  040  P.ADK:     .ASCII  /%A /
005477  040  040  040            .ASCII  /   /
005502  103  117  115            .ASCII  /COM/
005505  115  101  116            .ASCII  /MAN/
005510  104  072  040            .ASCII  /D: /
005513  000                      .ASCII  <00>
005514  045  101  123  P.ADL:     .ASCII  /%AS/
005517  105  124  040            .ASCII  /ET /
005522  103  124  114            .ASCII  /CTL/
005525  122  040  103            .ASCII  /R C/
005530  110  101  122            .ASCII  /HAR/
005533  000                      .ASCII  <00>
005534  045  101  117  P.ADM:     .ASCII  /%AO/
005537  116  114  111            .ASCII  /NLI/
005542  116  105  000            .ASCII  /NE/<00>
005545  000                      .ASCII  <00>
005546  045  101  105  P.ADN:     .ASCII  /%AE/
005551  130  105  103            .ASCII  /XEC/
005554  125  124  105            .ASCII  /UTE/
005557  040  123  125            .ASCII  / SU/
005562  120  120  114            .ASCII  /PPL/
005565  111  105  104            .ASCII  /IED/
005570  040  120  122            .ASCII  / PR/
005573  117  107  122            .ASCII  /OGR/
005576  101  115  000            .ASCII  /AM/<00>
005601  000                      .ASCII  <00>
005602  045  101  123  P.ADO:     .ASCII  /%AS/
005605  105  116  104            .ASCII  /END/
005610  040  104  101            .ASCII  / DA/
005613  124  101  000            .ASCII  /TA/<00>
005616  045  101  122  P.ADP:     .ASCII  /%AR/
005621  105  101  104            .ASCII  /EAD/
005624  000  000                 .ASCII  <00><00>
005626  045  101  127  P.ADQ:     .ASCII  /%AW/
005631  122  111  124            .ASCII  /RIT/
005634  105  000                 .ASCII  /E/<00>
005636  045  101  055  P.ADR:     .ASCII  /%A-/
005641  103  117  115            .ASCII  /COM/
005644  120  101  122            .ASCII  /PAR/
005647  105  045  116            .ASCII  /E%N/
005652  000  000                 .ASCII  <00><00>
005654  045  101  040  P.ADS:     .ASCII  /%A /
005657  040  040  040            .ASCII  /   /
005662  102  101  104            .ASCII  /BAD/
```

```
005665  040  102  114                .ASCII  / BL/
005670  117  103  113                .ASCII  /OCK/
005673  040  122  105                .ASCII  / RE/
005676  120  117  122                .ASCII  /POR/
005701  124  105  104                .ASCII  /TED/
005704  072  040  045                .ASCII  /: %/
005707  104  065  045                .ASCII  /D5%/
005712  101  056  045                .ASCII  /A.%/
005715  116  000  000                .ASCII  /N/<00><00>
005720  045  101  040      P.ADT:    .ASCII  /%A /
005723  040  040  040                .ASCII  /   /
005726  114  102  116                .ASCII  /LBN/
005731  072  040  045                .ASCII  /: %/
005734  104  065  045                .ASCII  /D5%/
005737  101  056  045                .ASCII  /A.%/
005742  116  000                     .ASCII  /N/<00>
005744  045  101  040      P.ADU:    .ASCII  /%A /
005747  040  040  040                .ASCII  /   /
005752  102  131  124                .ASCII  /BYT/
005755  105  040  103                .ASCII  /E C/
005760  117  125  116                .ASCII  /OUN/
005763  124  040  111                .ASCII  /T I/
005766  116  040  103                .ASCII  /N C/
005771  117  115  115                .ASCII  /OMM/
005774  101  116  104                .ASCII  /AND/
005777  072  040  045                .ASCII  /: %/
006002  104  065  045                .ASCII  /D5%/
006005  101  056  045                .ASCII  /A.%/
006010  116  000                     .ASCII  /N/<00>
006012  045  101  040      P.ADV:    .ASCII  /%A /
006015  040  040  040                .ASCII  /   /
006020  101  103  124                .ASCII  /ACT/
006023  125  101  114                .ASCII  /UAL/
006026  040  043  040                .ASCII  / # /
006031  117  106  040                .ASCII  /OF /
006034  102  131  124                .ASCII  /BYT/
006037  105  123  040                .ASCII  /ES /
006042  124  122  101                .ASCII  /TRA/
006045  116  123  106                .ASCII  /NSF/
006050  105  122  122                .ASCII  /ERR/
006053  105  104  072                .ASCII  /ED:/
006056  040  045  104                .ASCII  / %D/
006061  065  045  101                .ASCII  /5%A/
006064  056  045  116                .ASCII  /.%N/
006067  000                         .ASCII  <00>
006070  045  101  040      P.ADW:    .ASCII  /%A /
006073  040  040  040                .ASCII  /   /
006076  111  057  117                .ASCII  /I/<57>/O/
006101  040  102  125                .ASCII  / BU/
006104  106  106  105                .ASCII  /FFE/
006107  122  040  104                .ASCII  /R D/
006112  105  123  103                .ASCII  /ESC/
006115  122  111  120                .ASCII  /RIP/
006120  124  117  122                .ASCII  /TOR/
006123  072  045  117                .ASCII  /:%O/
006126  067  045  101                .ASCII  /7%A/
006131  050  117  051                .ASCII  /(O)/
```

```
006134  045  117  067                .ASCII  /%O7/
006137  045  101  050                .ASCII  /%A(/
006142  117  051  045                .ASCII  /O)%/
006145  116  000  000                .ASCII  /N/<00><00>
006150  045  116  045  P.ADX:        .ASCII  /%N%/
006153  101  105  122                .ASCII  /AER/
006156  122  117  122                .ASCII  /ROR/
006161  040  114  117                .ASCII  / LO/
006164  107  040  115                .ASCII  /G M/
006167  105  123  123                .ASCII  /ESS/
006172  101  107  105                .ASCII  /AGE/
006175  040  122  105                .ASCII  / RE/
006200  103  105  111                .ASCII  /CEI/
006203  126  105  104                .ASCII  /VED/
006206  072  045  116                .ASCII  /:%N/
006211  000                        .ASCII  <00>
006212  045  101  040  P.ADY:        .ASCII  /%A /
006215  040  040  040                .ASCII  /   /
006220  120  114  101                .ASCII  /PLA/
006223  124  124  105                .ASCII  /TTE/
006226  122  072  040                .ASCII  /R: /
006231  045  104  063                .ASCII  /%D3/
006234  045  101  056                .ASCII  /%A./
006237  045  116  000                .ASCII  /%N/<00>
006242  045  101  040  P.ADZ:        .ASCII  /%A /
006245  040  040  040                .ASCII  /   /
006250  106  117  122                .ASCII  /FOR/
006253  115  101  124                .ASCII  /MAT/
006256  072  040  000                .ASCII  /: /<00>
006261  000                        .ASCII  <00>
006262  045  101  040  P.AEA:        .ASCII  /%A /
006265  040  040  040                .ASCII  /   /
006270  105  126  105                .ASCII  /EVE/
006273  116  124  040                .ASCII  /NT /
006276  103  117  104                .ASCII  /COD/
006301  105  072  040                .ASCII  /E: /
006304  000  000                    .ASCII  <00><00>
006306  045  101  040  P.AEB:        .ASCII  /%A /
006311  040  040  040                .ASCII  /   /
006314  110  117  123                .ASCII  /HOS/
006317  124  040  115                .ASCII  /T M/
006322  105  115  040                .ASCII  /EM /
006325  101  104  104                .ASCII  /ADD/
006330  122  072  045                .ASCII  /R:%/
006333  117  067  045                .ASCII  /O7%/
006336  101  050  117                .ASCII  /A(O/
006341  051  045  117                .ASCII  /)%O/
006344  067  045  101                .ASCII  /7%A/
006347  050  117  051                .ASCII  /(O)/
006352  045  116  000                .ASCII  /%N/<00>
006355  000                        .ASCII  <00>
006356  045  117  063  P.AEC:        .ASCII  /%O3/
006361  045  101  050                .ASCII  /%A(/
006364  117  051  045                .ASCII  /O)%/
006367  116  000  000                .ASCII  /N/<00><00>
006372  045  104  062  P.AED:        .ASCII  /%D2/
006375  045  101  072                .ASCII  /%A:/
```

```
006400      045      104      062                    .ASCII  /%D2/
006403      045      101      072                    .ASCII  /%A:/
006406      045      104      062                    .ASCII  /%D2/
006411      045      101      040                    .ASCII  /%A /
006414      040      000                             .ASCII  / /<00>
006416      045      101      120           P.AEE:   .ASCII  /%AP/
006421      114      101      124                    .ASCII  /LAT/
006424      124      105      122                    .ASCII  /TER/
006427      040      045      104                    .ASCII  / %D/
006432      063      045      101                    .ASCII  /3%A/
006435      056      040      055                    .ASCII  /. -/
006440      040      000                             .ASCII  / /<00>
006442      045      116      000           P.AEF:   .ASCII  /%N/<00>
006445      000                                      .ASCII  <00>
006446   000000C                          L$HWLEN::
                                                     .WORD   <<L$NDHW-L$HWLEN>/2>
006450   172150                          HWPT.IP.ADDR::
                                                     .WORD   -5630
006452   000154                          HWPT.VECTOR::
                                                     .WORD   154
006454   000005                          HWPT.BR.LEVEL::
                                                     .WORD   5
006456   000000                          HWPT.PLAT::
                                                     .WORD   0
006460                                   L$NDHW::.BLKW   1
006462   000000C                         L$SWLEN::
                                                     .WORD   <<L$NDSW-L$SWLEN>/2>
006464   000040                          SWP.ERROR::
                                                     .WORD   40
006466   000002                          SWP.XFER::
                                                     .WORD   2
006470   000000                          SWP.STRACK::
                                                     .WORD   0
006472   003151                          SWP.ETRACK::
                                                     .WORD   3151
006474      055                          SWP.FLAGS::
                                                     .BYTE   55
006475      000                          SWP.DPAT::
                                                     .BYTE   0
006476   000020                          SWP.UCNT::
                                                     .WORD   20
006500                                   SWP.UDPAT::
                                                     .BLKW   20
006540                                   L$NDSW::.BLKW   1
006542   177777                          L$PROT::.WORD   -1
006544   177777                                   .WORD   -1
006546   177777                                   .WORD   -1


000000                                               .PSECT  $FFF$,  RO
000000                                   PATCH::  .BLKW   144
000310                                   CPT::    .BLKW   10
000330                                   CST::    .BLKW   34
000420                                   CST.ADDR::
                                                     .BLKW   1
000422                                   DCT::    .BLKW   44
```

```
000532                          DCT.ADDR::
                                        .BLKW   1
000534                          RC25.ADDR::
                                        .BLKW   1
000536                          IRC25.ADDR::
                                        .BLKW   1
000540                          DM.COMM::
                                        .BLKW   150
001060                          DMC.ADDR::
                                        .BLKW   1
001062                          RP.SAVE::
                                        .BLKW   20
001122                          RPS.X1::.BLKW   1
001124                          RPS.X2::.BLKW   1
001126                          OUTC.LIST::
                                        .BLKW   40
001226                          OUTC.TIMR::
                                        .BLKW   100
001426                          OCL.X1::.BLKW   1
001430                          OCL.X2::.BLKW   1
001432                          TALLY:: .BLKW   600
003032                          T.ADDR::.BLKW   1
003034                          MSCP.ENV::
                                        .BLKW   6300
017634                          ENV.USE::
                                        .BLKW   60
017774                          RETPKT::.BLKW   1400
022774                          RP.USE::.BLKW   20
023034                          RP.INDX::
                                        .BLKW   1
023036                          RP.ADDR::
                                        .BLKW   1
023040                          BUFF.DESC::
                                        .BLKW   200
023440                          BUFF.OWN::
                                        .BLKW   40
023540                          IODQ::  .BLKW   20
023600                          IODQ.IN::
                                        .BLKW   1
023602                          IODQ.OUT::
                                        .BLKW   1
023604                          ENTRY.REASON::
                                        .BLKB   1
023605                          T.FLAG::.BLKB   1
023606                          EOP.FLAG::
                                        .BLKB   1
023607                          MEM.MGMT::
                                        .BLKB   1
023610                          IIP.FLAG::
                                        .BLKB   1
                                        .EVEN
023612                          CCTLR:: .BLKW   1
023614                          CPLAT:: .BLKW   1
023616                          CUOFF:: .BLKW   1
023620                          CTLR.CNT::
                                        .BLKW   1
023622                          DUR::   .BLKW   10
```

```
023642                          QIO::    .BLKW   2
023646                          MEM.SIZE::
                                         .BLKW   1
023650                          FREE.MEM.ADDR::
                                         .BLKW   1
023652                          BUFF.SIZE::
                                         .BLKW   1
023654                          NUM.BUFF::
                                         .BLKW   1
023656                          CLK.TYPE::
                                         .BLKW   1
023660                          CLK.HERTZ::
                                         .BLKW   1
023662                          CLK.CSR::
                                         .BLKW   1
023664                          CLK.VECTOR::
                                         .BLKW   1
023666                          HOURS::  .BLKW   1
023670                          MINUTES::
                                         .BLKW   1
023672                          SECONDS::
                                         .BLKW   1
023674                          TICKS::  .BLKW   1
023676                          ST.CODE::
                                         .BLKW   1
023700                          SB.CODE::
                                         .BLKW   1
023702                          STEP::   .BLKW   1
023704                          OF.RC::  .BLKW   1
023706                          SA.REG::.BLKW    1
023710                          NEX::    .BLKW   1
023712                          CRN::    .BLKW   1

                                .GLOBL  L$SOFT, T$PTHV, L$RPT, L$INIT
                                .GLOBL  L$CLEAN, L$LAST, L$HARD, L$DVTYP
                                .GLOBL  L$DESC, L$DU, L$AU, L$AUTO, T1


100000                          BIT15==          -100000
040000                          BIT14==          40000
020000                          BIT13==          20000
010000                          BIT12==          10000
004000                          BIT11==          4000
002000                          BIT10==          2000
001000                          BIT09==          1000
000400                          BIT08==          400
000200                          BIT07==          200
000100                          BIT06==          100
000040                          BIT05==          40
000020                          BIT04==          20
000010                          BIT03==          10
000004                          BIT02==          4
000002                          BIT01==          2
000001                          BIT00==          1
001000                          BIT9==           1000
000400                          BIT8==           400
```

```
000200                  BIT7==          200
000100                  BIT6==          100
000040                  BIT5==          40
000020                  BIT4==          20
000010                  BIT3==          10
000004                  BIT2==          4
000002                  BIT1==          2
000001                  BIT0==          1
000040                  EF.START==      40
000037                  EF.RESTART==    37
000036                  EF.CONTINUE==   36
000035                  EF.NEW==        35
000034                  EF.PWR==        34
000340                  PRI07==         340
000300                  PRI06==         300
000240                  PRI05==         240
000200                  PRI04==         200
000140                  PRI03==         140
000100                  PRI02==         100
000040                  PRI01==         40
000000                  PRI00==         0
000004                  EVL==           4
000010                  LOT==           10
000020                  ADR==           20
000040                  IDU==           40
000100                  ISR==           100
000200                  UAM==           200
000400                  BOE==           400
001000                  PNT==           1000
002000                  PRI==           2000
004000                  IXE==           4000
010000                  IBE==           10000
020000                  IER==           20000
040000                  LOE==           40000
100000                  HOE==           -100000
000126'                 L$ERRTBL==      ERRTYP
006464'                 L$SW==          L$SWLEN+2
006450'                 L$HW==          L$HWLEN+2
000011'                 L$DEPO==        L$REV+1
000136'                 HWQ1==          P.AAA
000152'                 HWQ2==          P.AAB
000162'                 HWQ3==          P.AAC
000174'                 HWQ4==          P.AAD
000230'                 HWQ5==          P.AAE
000314'                 HWQ6==          P.AAF
000414'                 SWQ1==          P.AAG
000452'                 SWQ2==          P.AAH
000530'                 SWQ3==          P.AAI
000576'                 SWQ4==          P.AAJ
000656'                 SWQ5==          P.AAK
000700'                 SWQ6==          P.AAL
000720'                 SWQ7==          P.AAM
000736'                 SWQ8==          P.AAN
001010'                 SWQ9==          P.AAO
001024'                 SWQ10==         P.AAP
001100'                 SWQ11==         P.AAQ
001144'                 SWQ12==         P.AAR
```

```
001176'                 SWQ13==         P.AAS
001274'                 SWQ14==         P.AAT
001352'                 SWQ15==         P.AAU
001370'                 SWM1==          P.AAV
001464'                 NULL==          P.AAW
001466'                 MSG.01==        P.AAX
001520'                 MSG.02==        P.AAY
001616'                 MSG.03==        P.AAZ
001636'                 MSG.04==        P.ABA
001670'                 MSG.05==        P.ABB
001730'                 MSG.06==        P.ABC
001772'                 MSG.07==        P.ABD
002044'                 MSG.08==        P.ABE
002140'                 CER.01==        P.ABF
002224'                 CER.02==        P.ABG
002270'                 CER.03==        P.ABH
002344'                 DUM.00==        P.ABI
002376'                 DUM.UC==        P.ABJ
002420'                 DUM.CE==        P.ABK
002450'                 DUM.IE==        P.ABL
002470'                 DUM.HE==        P.ABM
002526'                 DUM.UE==        P.ABN
002556'                 EGS.01==        P.ABO
002576'                 EGS.02==        P.ABP
002654'                 EGD.10==        P.ABQ
002714'                 EGD.11==        P.ABR
002740'                 EGD.12==        P.ABS
002766'                 EGD.13==        P.ABT
003014'                 EGD.14==        P.ABU
003052'                 EGD.15==        P.ABV
003104'                 EGD.16==        P.ABW
003122'                 EGD.17==        P.ABX
003152'                 EGD.18==        P.ABY
003170'                 EGD.19==        P.ABZ
003210'                 EGD.20==        P.ACA
003234'                 EGD.21==        P.ACB
003260'                 EGD.22==        P.ACC
003306'                 EGH.30==        P.ACD
003332'                 EBS.01==        P.ACE
003400'                 EBD.10==        P.ACF
003444'                 EBD.12==        P.ACG
003526'                 EBD.13==        P.ACH
003604'                 EBD.14==        P.ACI
003676'                 EBD.15==        P.ACJ
003766'                 EBD.16==        P.ACK
004062'                 EBD.17==        P.ACL
004160'                 EBD.18==        P.ACM
004224'                 EBD.19==        P.ACN
004264'                 EBD.20==        P.ACO
004350'                 EBD.21==        P.ACP
004436'                 EBD.22==        P.ACQ
004542'                 STC.00==        P.ACR
004556'                 STC.01==        P.ACS
004602'                 STC.02==        P.ACT
004626'                 STC.03==        P.ACU
004650'                 STC.04==        P.ACV
004674'                 STC.05==        P.ACW
```

```
004724'                         STC.06==        P.ACX
004750'                         STC.07==        P.ACY
005002'                         STC.08==        P.ACZ
005022'                         STC.09==        P.ADA
005060'                         STC.10==        P.ADB
005106'                         STC.11==        P.ADC
004556'                         EBH.31==        P.ACS
004602'                         EBH.32==        P.ACT
004650'                         EBH.34==        P.ACV
004674'                         EBH.35==        P.ACW
004724'                         EBH.36==        P.ACX
004750'                         EBH.37==        P.ACY
005002'                         EBH.38==        P.ACZ
005022'                         EBH.39==        P.ADA
005060'                         EBH.40==        P.ADB
005106'                         EBH.41==        P.ADC
005126'                         EBH.42==        P.ADD
005174'                         EBH.43==        P.ADE
005256'                         EX.SA==         P.ADF
005304'                         EX.CRN==        P.ADG
005342'                         EX.SC==         P.ADH
005400'                         EX.DSC==        P.ADI
005440'                         EX.SB==         P.ADJ
005474'                         EX.CMD==        P.ADK
005514'                         EX.SCC==        P.ADL
005534'                         EX.ONL==        P.ADM
005546'                         EX.ESP==        P.ADN
005602'                         EX.SND==        P.ADO
005616'                         EX.RD==         P.ADP
005626'                         EX.WRT==        P.ADQ
005636'                         EX.CMP==        P.ADR
005654'                         EX.BB==         P.ADS
005720'                         EX.LBN==        P.ADT
005744'                         EX.CBC==        P.ADU
006012'                         EX.BC==         P.ADV
006070'                         EX.BD==         P.ADW
006150'                         EX.EL==         P.ADX
006212'                         EX.PA==         P.ADY
006242'                         EX.FMT==        P.ADZ
006262'                         EX.EVC==        P.AEA
006306'                         EX.HMA==        P.AEB
006356'                         EX.03==         P.AEC
006372'                         ETIME==         P.AED
006416'                         PLATT==         P.AEE
006442'                         CRLF==          P.AEF
006450'                         DFPTBL==        L$HWLEN+2
006464'                         SFPTBL==        L$SWLEN+2


;                               PSECT SUMMARY
;
;       Psect Name              Words     Attributes
;        $CODE$                  1716       RO ,  I  ,  LCL,  REL,  CON
;        $FFF$                   5094       RO ,  I  ,  LCL,  REL,  CON
```

```
;                               LIBRARY STATISTICS
;
;                                          -------- Symbols --------    Blocks
;       File                               Total    Loaded   Percent      Read
;
;  _DUA2:[DOUCETTE.CZRCD]CZRCDL.L16;9        276       138        50        39
```

```
;                                       COMMAND QUALIFIERS

;       BLISS /PDP11 CZRCD1.SRC/LIST/EN:NOEIS

; Size:          0 code + 6810 data words
; Run Time:          00:19.0
; Elapsed Time:      00:24.0
; Memory Used:  237 pages
; Compilation Complete
```

```
;       0001  MODULE CZRCD2 (
;       0002          %TITLE 'CZRCDA0 RC25 DISK EXERCISER'
;       0003          IDENT = 'V01.0',
;       0004          ADDRESSING_MODE (ABSOLUTE)
;       0005          ) =
;       0006  BEGIN
;       0007
;       0043  %SBTTL 'DECLARATIONS'
;       0044
;       0045  LIBRARY 'CZRCDL';                        ! RC25 EXERCISER GLOBAL LIBRARY
;       0046  REQUIRE 'BLSMAC.REQ';                    ! DIAGNOSTIC SUPERVISOR LIBRARY
;       1535
;       1536  FORWARD ROUTINE
;       1537      NEX_TRAP : L$ISR NOVALUE,
;       1538      CLK_INT_SERV : L$ISR NOVALUE,
;       1539      EMS_01 : NOVALUE;
;       1540
;       1541  EXTERNAL
;       1542      PATCH : VECTOR [100, WORD],                ! PATCH AREA
;       1543      CPT : VECTOR [MAX_UNITS, BYTE],
;       1544  !       CURRENT PASS TESTING (YES / NO) PER UNIT
;       1545      CST : BLOCKVECTOR [MAX_CTLR, CST_LEN, WORD] FIELD (C_FIELDS),
;       1546  !       RUN-TIME CONTROLLER STATUS TABLES
;       1547      CST_ADDR : REF BLOCK [CST_LEN, WORD] FIELD (C_FIELDS),
;       1548  !       CONTROLLER STATUS TABLE ADDRESS OF "CURRENT" CONTROLLER
;       1549      DCT : BLOCKVECTOR [MAX_CTLR, DCT_LEN, WORD] FIELD (DC_FIELDS),
;       1550  !       DRIVER CONTROLLER TABLES
;       1551      DCT_ADDR : REF BLOCK [DCT_LEN, WORD] FIELD (DC_FIELDS),
;       1552  !       ADDRESS OF "CURRENT" DRIVER CONTROLLER TABLE
;       1553      RC25_ADDR : REF RC25 FIELD (RC_REG),
;       1554  !       DEVICE ADDRESS OF "CURRENT" CONTROLLER
```

```
;        1555      IRC25_ADDR : REF RC25 FIELD (RC_REG),
;        1556  !       DEVICE ADDRESS OF INTERRUPTING CONTROLLER
;        1557      DM_COMM : BLOCKVECTOR [MAX_CTLR, DMC_LEN, WORD] FIELD (DMC_FIELDS),
;        1558  !       DM EXERCISER COMMUNICATION AREA (LINK TO FRONT PANEL TEST)
;        1559      DMC_ADDR : REF BLOCK [DMC_LEN, WORD] FIELD (DMC_FIELDS),
;        1560  !       ADDRESS OF CURRENT CONTROLLER'S DM EXERCISER COMMUNICATION AREA
;        1561      RP_SAVE : VECTOR [MAX_CTLR * RPS_LEN, BYTE, SIGNED],
;        1562  !       RETURN PACKET SAVE AREA
;        1563      RPS_X1 : WORD,                          ! STARTING INDEX OF CURRENT CONTROLLER'S RP_SAVE AREA
;        1564      RPS_X2 : WORD,                          ! ENDING INDEX OF CURRENT CONTROLLER'S RP_SAVE AREA
;        1565      OUTC_LIST : VECTOR [MAX_CTLR * OUTC_CNT, BYTE, SIGNED],
;        1566  !       OUTSTANDING COMMAND LIST (CONTAINS MSCP ENVELOPE INDECES)
;        1567      OUTC_TIMR : VECTOR [MAX_CTLR * OUTC_CNT, WORD],
;        1568  !       OUTSTANDING COMMAND TIMERS
;        1569      OCL_X1 : WORD,
;        1570  !       STARTING INDEX OF CURRENT CONTROLLER'S OUTSTANDING COMMAND AREA
;        1571      OCL_X2 : WORD,
;        1572  !       ENDING INDEX OF CURRENT CONTROLLER'S OUTSTANDING COMMAND AREA
;        1573      TALLY : VECTOR [MAX_UNITS * TALLY_LEN, WORD] FIELD (T_FIELDS),
;        1574  !       STATISTICS TABLES
;        1575      T_ADDR : REF BLOCK [TALLY_LEN, WORD] FIELD (T_FIELDS),
;        1576  !       ADDRESS OF STATISTICS TABLE (TALLY) FOR CURRENT UNIT
;        1577      MSCP_ENV : BLOCKVECTOR [ENV_CNT, ENV_LEN, WORD] FIELD (E_FIELDS),
;        1578  !       MSCP ENVELOPE POOL
;        1579      ENV_USE : VECTOR [ENV_CNT, BYTE, SIGNED],
;        1580  !       MSCP ENVELOPE POOL ALLOCATION TABLE
;        1581      RETPKT : BLOCKVECTOR [RP_CNT, RP_LEN, WORD] FIELD (RP_FIELDS),
;        1582  !       RETURN PACKET POOL
;        1583      RP_USE : VECTOR [RP_CNT, BYTE, SIGNED],
;        1584  !       RETURN PACKET POOL ALLOCATION TABLE
;        1585      RP_INDX : WORD,                         ! CURRENT RETURN PACKET INDEX
;        1586      RP_ADDR : REF BLOCK [RP_LEN, WORD] FIELD (RP_FIELDS),
;        1587  !       CURRENT RETURN PACKET ADDRESS
```

```
;       1588      BUFF_DESC : BLOCKVECTOR [MAX_BUF_CNT, DESC_LEN, WORD] FIELD (BD_FIELDS),
;       1589  !       TABLE OF I/O BUFFER DESCRIPTORS
;       1590      BUFF_OWN : VECTOR [MAX_BUF_CNT, BYTE, SIGNED],
;       1591  !       I/O BUFFER OWNERSHIP (CONTROLLER NUMBER)
;       1592      IODQ : VECTOR [IODQ_LEN, BYTE],
;       1593  !       I/O DONE QUEUE - CIRCULAR QUEUE OF RETPKT INDECES
;       1594      IODQ_IN : WORD,                             ! I/O DONE QUEUE IN POINTER
;       1595      IODQ_OUT : WORD,                            ! I/O DONE QUEUE OUT POINTER
;       1596      ENTRY_REASON : BYTE,                        ! HOW CURRENT PASS WAS INVOKED
;       1597      T_FLAG : BYTE,                              ! ONE SECOND TIMING FLAG
;       1598      EOP_FLAG : BYTE,                            ! END-OF-PASS FLAG
;       1599      MEM_MGMT : BYTE,                            ! MEMORY MANAGEMENT FLAG
;       1600      IIP_FLAG : BYTE,                            ! INITIALIZATION-IN-PROGRESS FLAG
;       1601      CCTLR : WORD,                               ! NUMBER OF "CURRENT" CONTROLLER
;       1602      CPLAT : WORD,                               ! CURRENT PLATTER ADDRESS (MSCP UNIT NUMBER)
;       1603      CUOFF : WORD,                               ! CST OFFSET FOR CURRENT UNIT
;       1604      CTLR_CNT : WORD,                            ! TOTAL NUMBER OF CONFIGURED CONTROLLERS
;       1605      DUR : VECTOR [MAX_UNITS, BYTE],             ! DROP UNIT REASON
;       1606      QIO : VECTOR [MAX_CTLR, BYTE],              ! NUMBER OF OUTSTANDING QIOS PER CONTROLLER
;       1607      MEM_SIZE : WORD,                            ! AVAILABLE MEMORY (IN WORDS) UP TO 28K
;       1608      FREE_MEM_ADDR,                              ! START OF FREE MEMORY BELOW 28K
;       1609      BUFF_SIZE : WORD,                           ! SIZE (BYTES) OF AN I/O BUFFER
;       1610      NUM_BUFF : WORD,                            ! NUMBER OF I/O BUFFERS
;       1611      CLK_TYPE : WORD,                            ! TYPE OF CLOCK ON SYSTEM
;       1612                                                  !   (0 = NONE, -1 = L-CLOCK, 1 = P_CLOCK)
;       1613      CLK_HERTZ : WORD,                           ! CLOCK HERTZ RATE
;       1614      CLK_CSR,                                    ! CLOCK CSR ADDRESS
;       1615      CLK_VECTOR,                                 ! CLOCK VECTOR ADDRESS
;       1616      HOURS : WORD,                               ! ELAPSED TIME - HOURS,
;       1617      MINUTES : WORD,                             !       MINUTES,
;       1618      SECONDS : WORD,                             !       SECONDS,
;       1619      TICKS : WORD,                               !       TICKS
;       1620      ST_CODE : WORD,                             ! CURRENT STATUS CODE
;       1621      SB_CODE : WORD,                             ! CURRENT SUB-CODE
;       1622      STEP : WORD,                                ! CURRENT STEP IN HARD_INIT
;       1623      OF_RC : SIGNED WORD,                        ! OFFSET (0 OR 2) TO READ IP OR SA
;       1624      SA_REG : WORD,                              ! STORAGE FOR SA REGISTER READS AND WRITES
;       1625      NEX : WORD,                                 ! NON-EXISTENT MEMORY TRAP INDICATOR
;       1626      CRN : WORD,                                 ! COMMAND REF NUMBER OF LAST COMMAND SENT
;       1627      HWQ1,
;       1628      HWQ2,
;       1629      HWQ3,
;       1630      HWQ4,
;       1631      HWQ5,
;       1632      HWQ6,
;       1633      SWQ1,
;       1634      SWQ2,
;       1635      SWQ3,
;       1636      SWQ4,
;       1637      SWQ5,
;       1638      SWQ6,
;       1639      SWQ7,
;       1640      SWQ8,
;       1641      SWQ9,
;       1642      SWQ10,
;       1643      SWQ11,
;       1644      SWQ12,
```

```
;          1645      SWQ13,
;          1646      SWQ14,
;          1647      SWQ15,
;          1648      SWM1,
;          1649      MSG_01,
;          1650      MSG_07,
;          1651      CER_01,
;          1652      CER_02,
;          1653      CER_03,
;          1654      DUM_00,
;          1655      DUM_UC,
;          1656      DUM_CE,
;          1657      DUM_IE,
;          1658      DUM_HE,
;          1659      DUM_UE,
;          1660      EGS_01,
;          1661      EGS_02,
;          1662      EBS_01,
;          1663      EBD_10,
;          1664      EBD_12,
;          1665      EBD_13,
;          1666      EBD_14,
;          1667      EBD_15,
;          1668      EBD_16,
;          1669      EBD_17,
;          1670      EBD_18,
;          1671      EBD_19,
;          1672      EBD_20,
;          1673      EBD_21,
;          1674      EBD_22,
;          1675      EBH_31,
;          1676      EBH_32,
;          1677      EBH_34,
;          1678      EBH_35,
;          1679      EBH_36,
;          1680      EBH_37,
;          1681      EBH_38,
;          1682      EBH_39,
;          1683      EBH_40,
;          1684      EBH_41,
;          1685      EBH_42,
;          1686      EBH_43,
;          1687      EX_SA,
;          1688      EX_CRN,
;          1689      EX_SC,
;          1690      EX_DSC,
;          1691      EX_SB,
;          1692      EX_CMD,
;          1693      EX_SCC,
;          1694      EX_ONL,
;          1695      EX_ESP,
;          1696      EX_SND,
;          1697      EX_RD,
;          1698      EX_WRT,
;          1699      EX_CMP,
;          1700      EX_BB,
;          1701      EX_LBN,
```

```
;        1702      EX_CBC,
;        1703      EX_BC,
;        1704      EX_BD,
;        1705      EX_03,
;        1706      ETIME,
;        1707      PLATT,
;        1708      CRLF,
;        1709      SWP_ERROR : WORD,
;        1710      SWP_XFER : WORD,
;        1711      SWP_STRACK : WORD,
;        1712      SWP_ETRACK : WORD,
;        1713      SWP_FLAGS : BYTE,
;        1714      L$LUN,
;        1715      L$UNIT;

;        1716  %SBTTL 'TYPE AND DESCRIPTION'
;        1717
;        1718  EQUALS;
;        1719
;        1720  !+
;        1721  !       THE TEXT WHICH APPEARS IN THESE MACROS IS DISPLAYED TO THE USER WHEN
;        1722  !       THE DIAGNOSTIC IS FIRST LOADED AND RUN.
;        1723  !-
;        1724
;        1725  DEVTYP (%ASCIZ'SINGLE RC25 PLATTER');          ! 'UNIT IS SINGLE RC25 PLATTER'
;        1726
;        1727  DESCRIPT (%ASCIZ'RC25 DISK EXERCISER');        ! TEST DESCRIPTION
```

```
:          1728  %SBTTL 'HARDWARE PARAMETER CODING SECTION'
:          1729
:          1730  !+
:          1731  !       THE HARDWARE PARAMETER CODING SECTION CONTAINS MACROS THAT ARE USED BY
:          1732  !       THE SUPERVISOR TO BUILD P-TABLES. THE MACROS ARE NOT EXECUTED AS
:          1733  !       MACHINE INSTRUCTIONS BUT ARE INTERPRETED BY THE SUPERVISOR AS DATA
:          1734  !       STRUCTURES. THE MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
:          1735  !       WITH THE OPERATOR.
:          1736  !-
:          1737
:          1738  BGNHRD;
:          1739
:          1740  GPRMA (HWQ1, 0, 0, %O'160000', %O'177777', YES, 1);      ! IP ADDRESS
:          1741  GPRMA (HWQ2, 2, 0, %O'4', %O'774', YES, 1);               ! VECTOR
:          1742  GPRMD (HWQ3, 4, 0, %O'177777', %O'4', %O'7', YES, 1);     ! BR LEVEL
:          1743  GPRMD (HWQ4, 6, D, %O'377', %DECIMAL'0', %DECIMAL'253', YES, 1); ! PLATTER ADDRESS
:          1744  GPRML (HWQ5, 6, %O'100000', NO, 1);                       ! WRITE ON CUST DATA AREA
:          1745  XFERF (HWDONE);                                           ! NO - DONE
:          1746  GPRML (HWQ6, 6, %O'100000', NO, 1);                       ! ** WARNING / CONFIRM
:          1747  $L (HWDONE);
:          1748
:          1749  ENDHRD;
```

```
1750  %SBTTL 'SOFTWARE PARAMETER CODING SECTION'
1751
1752  !+
1753  !       THE SOFTWARE PARAMETER CODING SECTION CONTAINS MACROS THAT ARE USED BY
1754  !       THE SUPERVISOR TO INTERROGATE THE OPERATOR FOR SOFTWARE INFORMATION
1755  !       WHICH WILL BE PLACED IN THE SOFTWARE P-TABLE. THE MACROS ARE NOT
1756  !       EXECUTED AS MACHINE INSTRUCTIONS BUT ARE INTERPRETED BY THE SUPERVISOR
1757  !       AS DATA STRUCTURES. THE MACROS ALLOW THE SUPERVISOR TO ESTABLISH
1758  !       COMMUNICATIONS WITH THE OPERATOR.
1759  !-
1760
1761  BGNSFT;
1762
1763  GPRMD (SWQ1, 0, D, %O'177777', 0, 65535, YES, 1);          ! ERROR LIMIT
1764  GPRMD (SWQ2, 2, D, %O'177777', 0, 65535, YES, 1);          ! TRANSFER LIMIT
1765  GPRML (SWQ3, 8, SWF_SEL, YES, 1);                          ! SUPPRESS ERROR LOG PRINTING
1766  GPRML (SWQ4, 8, SWF_DM, YES, 1);                           ! RUN DM EXERC'SER
1767  XFERT (SW3);                                               ! IF YES - DONE
1768  GPRML (SWQ5, 8, SWF_RDM, YES, 1);                          ! RANDOM SEEK MODE
1769  GPRMD (SWQ6, 4, D, %O'177777', 0, MAX_TRACK, YES, 1);      ! STARTING TRACK
1770  GPRMD (SWQ7, 6, D, %O'177777', GP$ATLO(4), MAX_TRACK, YES, 1); ! ENDING TRACK
1771  GPRML (SWQ8, 8, SWF_CRC, YES, 1);                          ! READ-COMPARES AT CONTROLLER
1772  DISPLAY (SWM1);                                            ! REMAINING QUESTIONS ONLY APPLY ...
1773  GPRML (SWQ9, 8, SWF_WO, YES, 1);                           ! WRITE ONLY
1774  GPRML (SWQ10, 8, SWF_CWC, YES, 1);                         ! WRITE-COMPARES AT CONTROLLER
1775  XFERT (SW1);                                               ! IF YES, SKIP NEXT QUESTION
1776  GPRML (SWQ11, 8, SWF_HWC, YES, 1);                         ! CHECK WRITES AT HOST BY READING
1777  $L (SW1);
1778  GPRML (SWQ12, 8, SWF_UDP, YES, 1);                         ! USER-DEFINED DATA PATTERN
1779  XFERT (SW2);                                               ! IF YES, SKIP NEXT QUESTION
1780  GPRMD (SWQ13, 8, D, %O'177400', 0, DP_CNT, YES, 1);        ! SELECT PRE-DEFINED DATA PATTERN
1781  XFER (SW4);                                                ! DONE
1782  $L (SW2);
1783  GPRMD (SWQ14, 10, D, %O'177777', 1, MAX_UDP_CNT, YES, 1); ! NO. OF WORDS IN USER DATA PATTERN
1784  GPRMD (SWQ15, 12, O, %O'177777', 0, %O'177777', NO, 10); ! PATTERN VALUES
1785  $L (SW3);
1786  $L (SW4);
1787
1788  ENDSFT;
```

```
%SBTTL 'REPORT CODING SECTION'

!+
!       THE REPORT CODING SECTION CONTAINS THE ''PRINTS'' CALLS THAT GENERATE
!       STATISTICAL REPORTS. THIS SECTION IS EXECUTED BY THE OPERATOR COMMAND
!       ''PRINT'' OR BY THE PROGRAM MACRO ''DORPT''.
!-

BGNRPT;

BIND
    SR_HD = UPLIT (%ASCIZ'%N%A************************* S U M M A R Y   R E P O R T *************************%N'),
    SR_ET = UPLIT (%ASCIZ'%AELAPSED TIME:  %D2%A:%D2%A:%D2%N'),
    SR_MH1 = UPLIT (%ASCIZ'%N%A          NO.       NO.                         NO.     NO.'),
    SR_MH2 = UPLIT (%ASCIZ'%N%AUNIT   READS     WRITES    MBYTES  MBYTES    HARD    ERROR'),
    SR_MH3 = UPLIT (%ASCIZ'%N%A NO.  (K=1000)  (K=1000)    READ   WRITTEN  ERRORS  LOGS'),
    SR_MH4 = UPLIT (%ASCIZ'%N%A----  --------  --------  ------  -------  ------  -----%N'),
    SR_UN = UPLIT (%ASCIZ'%N%D3'),
    SR_LO = UPLIT (%ASCIZ'%D9%A '),
    SR_HI = UPLIT (%ASCIZ'%D9%AK'),
    SR_ML = UPLIT (%ASCIZ'%D9%D8%D9%D7'),
    SR_EH1 = UPLIT (%ASCIZ'%N%N%AN SYMBOL ECC ERRORS:%N'),
    SR_EH2 = UPLIT (%ASCIZ'%N%A          ECC'),
    SR_EH3 = UPLIT (%ASCIZ'%N%AUNIT  FIELD   N =    N =    N =    N =    N =    N =    N =    N ='),
    SR_EH4 = UPLIT (%ASCIZ'%N%A NO.   ONLY    1      2      3      4      5      6      7      8'),
    SR_EH5 = UPLIT (%ASCIZ'%N%A----  -----  -----  -----  -----  -----  -----  -----  -----  -----%N'),
    SR_EL1 = UPLIT (%ASCIZ'%N%D3%D8%D7%D7%D7'),
    SR_EL2 = UPLIT (%ASCIZ'%D7%D7%D7%D7%D7'),
    SR_DH1 = UPLIT (%ASCIZ'%N%A          NO.       NO.       NO.'),
    SR_DH2 = UPLIT (%ASCIZ'%N%AUNIT   READS     WRITES    SEEKS    MBYTES  MBYTES   NO. HARD  NO. SOFT'),
    SR_DH3 = UPLIT (%ASCIZ'%N%A NO.  (K=1000)  (K=1000)  (K=1000)   READ   WRITTEN   ERRORS    ERRORS'),
    SR_DH4 = UPLIT (%ASCIZ'%N%A----  --------  --------  --------  ------  -------  --------  --------%N'),
    SR_DL = UPLIT (%ASCIZ'%D9%D8%D10%D10');

OWN
    ADDR_LO;                                    ! ADDR OF LOW-ORDER FIELD OF I/O COUNT

ROUTINE HI_LO : NOVALUE =

!+
!       THIS ROUTINE IS CALLED FROM THE MAINLINE CODE OF THE REPORT CODING
!       SECTION WHICH APPEARS BELOW. ITS PURPOSE IS TO PRINT THE HIGH-ORDER
!       FIELD OF ONE OF THE I/O COUNTS IF THE FIELD IS NON-ZERO. OTHERWISE, THE
!       LOW-ORDER FIELD IS PRINTED.
!
!       IMPLICIT INPUTS:
!               ADDR_LO - ADDRESS OF THE LOW-ORDER FIELD OF AN I/O COUNT
!-

BEGIN

LOCAL
    ADDR_HI;                                    ! ADDR OF HIGH-ORDER FIELD OF I/O COUNT

ADDR_HI = .ADDR_LO + 2;                         ! GET HIGH-ORDER FIELD ADDRESS
IF ..ADDR_HI EQLU 0                             ! IF HIGH-ORDER FIELD IS ZERO
THEN                                            ! THEN
```

```
;        1846      PRINTS (SR_LO, ..ADDR_LO)                    ! PRINT LOW-ORDER FIELD
;        1847  ELSE                                               ! OTHERWISE
;        1848      PRINTS (SR_HI, ..ADDR_HI);                   ! PRINT HIGH-ORDER FIELD (THOUSANDS)
;        1849
;        1850  END;                                               ! ROUTINE HI_LO

                                        .TITLE  CZRCD2 CZRCDA0 RC25 DISK EXERCISER
                                        .IDENT  /V01.0/
                                        .ENABL  AMA


                                        .PSECT  $CODE$,  RO
000000
000000     123     111     116      L$DVTYP::
                                        .ASCII  /SIN/
000003     107     114     105             .ASCII  /GLE/
000006     040     122     103             .ASCII  / RC/
000011     062     065     040             .ASCII  /25 /
000014     120     114     101             .ASCII  /PLA/
000017     124     124     105             .ASCII  /TTE/
000022     122     000                     .ASCII  /R/<00>
000024                                     .BLKB   2
000026     122     103     062      L$DESC::.ASCII  /RC2/
000031     065     040     104             .ASCII  /5 D/
000034     111     123     113             .ASCII  /ISK/
000037     040     105     130             .ASCII  / EX/
000042     105     122     103             .ASCII  /ERC/
000045     111     123     105             .ASCII  /ISE/
000050     122     000                     .ASCII  /R/<00>
000052                                     .BLKB   2
000054  000000C                    L$HRDLN::
                                        .WORD   <<<L$NDHRD-L$HRDLN>/2>-1>
000056  000031                     GP$1::  .WORD   31
000060  000000G                             .WORD   HWQ1
000062  160000                             .WORD   -20000
000064  177777                             .WORD   -1
000066  001031                     GP$2::  .WORD   1031
000070  000000G                             .WORD   HWQ2
000072  000004                             .WORD   4
000074  000774                             .WORD   774
000076  002032                     GP$3::  .WORD   2032
000100  000000G                             .WORD   HWQ3
000102  177777                             .WORD   -1
000104  000004                             .WORD   4
000106  000007                             .WORD   7
000110  003052                     GP$4::  .WORD   3052
000112  000000G                             .WORD   HWQ4
000114  000377                             .WORD   377
000116  000000                             .WORD   0
000120  000375                             .WORD   375
000122  003120                     GP$5::  .WORD   3120
000124  000000G                             .WORD   HWQ5
000126  100000                             .WORD   -100000
000130  000000C                  $HWDONE:.WORD   <<<<$LHWDONE-$HWDONE>*400>+4>+40>
000132  003120                     GP$6::  .WORD   3120
000134  000000G                             .WORD   HWQ6
000136  100000                             .WORD   -100000
```

```
000140  001004                  $LHWDONE:
                                        .WORD   1004
000142                          L$NDHRD::
                                        .BLKW   1
000144  000000C                 L$SFTLN::
                                        .WORD   <<<L$NDSFT-L$SFTLN>/2>-1>
000146  000052                  GP$7::  .WORD   52
000150  000000G                          .WORD   SWQ1
000152  177777                          .WORD   -1
000154  000000                          .WORD   0
000156  177777                          .WORD   -1
000160  001052                  GP$8::  .WORD   1052
000162  000000G                          .WORD   SWQ2
000164  177777                          .WORD   -1
000166  000000                          .WORD   0
000170  177777                          .WORD   -1
000172  004130                  GP$9::  .WORD   4130
000174  000000G                          .WORD   SWQ3
000176  000001                          .WORD   1
000200  004130                  GP$10:: .WORD   4130
000202  000000G                          .WORD   SWQ4
000204  000002                          .WORD   2
000206  000000C                 $SW3:   .WORD   <<<<$LSW3-$SW3>*400>+4>+20>
000210  004130                  GP$11:: .WORD   4130
000212  000000G                          .WORD   SWQ5
000214  000004                          .WORD   4
000216  002052                  GP$12:: .WORD   2052
000220  000000G                          .WORD   SWQ6
000222  177777                          .WORD   -1
000224  000000                          .WORD   0
000226  003151                          .WORD   3151
000230  003452                  GP$13:: .WORD   3452
000232  000000G                          .WORD   SWQ7
000234  177777                          .WORD   -1
000236  000002                          .WORD   2
000240  003151                          .WORD   3151
000242  000001                          .WORD   1
000244  004130                  GP$14:: .WORD   4130
000246  000000G                          .WORD   SWQ8
000250  000010                          .WORD   10
000252  000003                  GP$DISP::
                                        .WORD   3
000254  000000G                         .WORD   SWM1
000256  004130                  GP$15:: .WORD   4130
000260  000000G                          .WORD   SWQ9
000262  000020                          .WORD   20
000264  004130                  GP$16:: .WORD   4130
000266  000000G                          .WORD   SWQ10
000270  000040                          .WORD   40
000272  000000C                 $SW1:   .WORD   <<<<$LSW1-$SW1>*400>+4>+20>
000274  004130                  GP$17:: .WORD   4130
000276  000000G                          .WORD   SWQ11
000300  000100                          .WORD   100
000302  001004                  $LSW1:  .WORD   1004
000304  004130                  GP$18:: .WORD   4130
000306  000000G                          .WORD   SWQ12
000310  000200                          .WORD   200
```

```
000312  000000C                         $SW2:    .WORD   <<<<$LSW2-$SW2>*400>+4>+20>
000314  004052                          GP$19::  .WORD   4052
000316  000000G                                  .WORD   SWQ13
000320  177400                                   .WORD   -400
000322  000000                                   .WORD   0
000324  000025                                   .WORD   25
000326  000000C                         $SW4:    .WORD   <<<$LSW4-$SW4>*400>+4>
000330  001004                          $LSW2:   .WORD   1004
000332  005052                          GP$20::  .WORD   5052
000334  000000G                                  .WORD   SWQ14
000336  177777                                   .WORD   -1
000340  000001                                   .WORD   1
000342  000020                                   .WORD   20
000344  006222                          GP$21::  .WORD   6222
000346  000000G                                  .WORD   SWQ15
000350  177777                                   .WORD   -1
000352  000000                                   .WORD   0
000354  177777                                   .WORD   -1
000356  000005                                   .WORD   5
000360  001004                          $LSW3:   .WORD   1004
000362  001004                          $LSW4:   .WORD   1004
000364                                  L$NDSFT::
                                                 .BLKW   1
000366                                  ADDR.LO::.BLKW   1



000000                                           .PSECT  $PLIT$,  RO ,  D
000000  045     116     045             P.AAA:   .ASCII  /%N%/
000003  101     052     052                      .ASCII  /A**/
000006  052     052     052                      .ASCII  /***/
000011  052     052     052                      .ASCII  /***/
000014  052     052     052                      .ASCII  /***/
000017  052     052     052                      .ASCII  /***/
000022  052     052     052                      .ASCII  /***/
000025  052     052     052                      .ASCII  /***/
000030  052     040     123                      .ASCII  /* S/
000033  040     125     040                      .ASCII  / U /
000036  115     040     115                      .ASCII  /M M/
000041  040     101     040                      .ASCII  / A /
000044  122     040     131                      .ASCII  /R Y/
000047  040     040     040                      .ASCII  /   /
000052  122     040     105                      .ASCII  /R E/
000055  040     120     040                      .ASCII  / P /
000060  117     040     122                      .ASCII  /O R/
000063  040     124     040                      .ASCII  / T /
000066  052     052     052                      .ASCII  /***/
000071  052     052     052                      .ASCII  /***/
000074  052     052     052                      .ASCII  /***/
000077  052     052     052                      .ASCII  /***/
000102  052     052     052                      .ASCII  /***/
000105  052     052     052                      .ASCII  /***/
000110  052     052     052                      .ASCII  /***/
000113  045     116     000                      .ASCII  /%N/<00>
000116  045     101     105             P.AAB:   .ASCII  /%AE/
000121  114     101     120                      .ASCII  /LAP/
000124  123     105     104                      .ASCII  /SED/
```

```
000127  040  124  111                    .ASCII  / TI/
000132  115  105  072                    .ASCII  /ME:/
000135  040  040  045                    .ASCII  /  %/
000140  104  062  045                    .ASCII  /D2%/
000143  101  072  045                    .ASCII  /A:%/
000146  104  062  045                    .ASCII  /D2%/
000151  101  072  045                    .ASCII  /A:%/
000154  104  062  045                    .ASCII  /D2%/
000157  116  000  000                    .ASCII  /N/<00><00>
000162  045  116  045          P.AAC:    .ASCII  /%N%/
000165  101  040  040                    .ASCII  /A  /
000170  040  040  040                    .ASCII  /   /
000173  040  040  040                    .ASCII  /   /
000176  116  117  056                    .ASCII  /NO./
000201  040  040  040                    .ASCII  /   /
000204  040  040  040                    .ASCII  /   /
000207  040  116  117                    .ASCII  / NO/
000212  056  040  040                    .ASCII  /.  /
000215  040  040  040                    .ASCII  /   /
000220  040  040  040                    .ASCII  /   /
000223  040  040  040                    .ASCII  /   /
000226  040  040  040                    .ASCII  /   /
000231  040  040  040                    .ASCII  /   /
000234  040  040  040                    .ASCII  /   /
000237  040  040  040                    .ASCII  /   /
000242  040  116  117                    .ASCII  / NO/
000245  056  040  040                    .ASCII  /.  /
000250  040  040  116                    .ASCII  /  N/
000253  117  056  000                    .ASCII  /O./<00>
000256  045  116  045          P.AAD:    .ASCII  /%N%/
000261  101  125  116                    .ASCII  /AUN/
000264  111  124  040                    .ASCII  /IT /
000267  040  040  122                    .ASCII  /  R/
000272  105  101  104                    .ASCII  /EAD/
000275  123  040  040                    .ASCII  /S  /
000300  040  040  040                    .ASCII  /   /
000303  127  122  111                    .ASCII  /WRI/
000306  124  105  123                    .ASCII  /TES/
000311  040  040  040                    .ASCII  /   /
000314  115  102  131                    .ASCII  /MBY/
000317  124  105  123                    .ASCII  /TES/
000322  040  040  115                    .ASCII  /  M/
000325  102  131  124                    .ASCII  /BYT/
000330  105  123  040                    .ASCII  /ES /
000333  040  040  040                    .ASCII  /   /
000336  110  101  122                    .ASCII  /HAR/
000341  104  040  040                    .ASCII  /D  /
000344  040  105  122                    .ASCII  / ER/
000347  122  117  122                    .ASCII  /ROR/
000352  000  000                        .ASCII  <00><00>
000354  045  116  045          P.AAE:    .ASCII  /%N%/
000357  101  040  116                    .ASCII  /A N/
000362  117  056  040                    .ASCII  /O. /
000365  040  050  113                    .ASCII  / (K/
000370  075  061  060                    .ASCII  /=10/
000373  060  060  051                    .ASCII  /00)/
000376  040  040  050                    .ASCII  /  (/
```

```
000401  113  075  061                    .ASCII  /K=1/
000404  060  060  060                    .ASCII  /000/
000407  051  040  040                    .ASCII  /)  /
000412  040  122  105                    .ASCII  / RE/
000415  101  104  040                    .ASCII  /AD /
000420  040  040  127                    .ASCII  /  W/
000423  122  111  124                    .ASCII  /RIT/
000426  124  105  116                    .ASCII  /TEN/
000431  040  040  105                    .ASCII  /  E/
000434  122  122  117                    .ASCII  /RRO/
000437  122  123  040                    .ASCII  /RS /
000442  040  114  117                    .ASCII  / LO/
000445  107  123  000                    .ASCII  /GS/<00>
000450  045  116  045            P.AAF:  .ASCII  /%N%/
000453  101  055  055                    .ASCII  /A--/
000456  055  055  040                    .ASCII  /-- /
000461  040  055  055                    .ASCII  / --/
000464  055  055  055                    .ASCII  /---/
000467  055  055  055                    .ASCII  /---/
000472  040  040  055                    .ASCII  /  -/
000475  055  055  055                    .ASCII  /---/
000500  055  055  055                    .ASCII  /---/
000503  055  040  040                    .ASCII  /-  /
000506  055  055  055                    .ASCII  /---/
000511  055  055  055                    .ASCII  /---/
000514  040  040  055                    .ASCII  /  -/
000517  055  055  055                    .ASCII  /---/
000522  055  055  055                    .ASCII  /---/
000525  040  040  055                    .ASCII  /  -/
000530  055  055  055                    .ASCII  /---/
000533  055  055  040                    .ASCII  /-- /
000536  040  055  055                    .ASCII  / --/
000541  055  055  055                    .ASCII  /---/
000544  045  116  000                    .ASCII  /%N/<00>
000547  000                            .ASCII  <00>
000550  045  116  045            P.AAG:  .ASCII  /%N%/
000553  104  063  000                    .ASCII  /D3/<00>
000556  045  104  071            P.AAH:  .ASCII  /%D9/
000561  045  101  040                    .ASCII  /%A /
000564  000  000                        .ASCII  <00><00>
000566  045  104  071            P.AAI:  .ASCII  /%D9/
000571  045  101  113                    .ASCII  /%AK/
000574  000  000                        .ASCII  <00><00>
000576  045  104  071            P.AAJ:  .ASCII  /%D9/
000601  045  104  070                    .ASCII  /%D8/
000604  045  104  071                    .ASCII  /%D9/
000607  045  104  067                    .ASCII  /%D7/
000612  000  000                        .ASCII  <00><00>
000614  045  116  045            P.AAK:  .ASCII  /%N%/
000617  116  045  101                    .ASCII  /N%A/
000622  116  040  123                    .ASCII  /N S/
000625  131  115  102                    .ASCII  /YMB/
000630  117  114  040                    .ASCII  /OL /
000633  105  103  103                    .ASCII  /ECC/
000636  040  105  122                    .ASCII  / ER/
000641  122  117  122                    .ASCII  /ROR/
000644  123  072  045                    .ASCII  /S:%/
```

```
000647  116  000  000            .ASCII  /N/<00><00>
000652  045  116  045  P.AAL:     .ASCII  /%N%/
000655  101  040  040            .ASCII  /A  /
000660  040  040  040            .ASCII  /   /
000663  040  040  105            .ASCII  /  E/
000666  103  103  000            .ASCII  /CC/<00>
000671  000                      .ASCII  <00>
000672  045  116  045  P.AAM:     .ASCII  /%N%/
000675  101  125  116            .ASCII  /AUN/
000700  111  124  040            .ASCII  /IT /
000703  040  106  111            .ASCII  / FI/
000706  105  114  104            .ASCII  /ELD/
000711  040  040  040            .ASCII  /   /
000714  116  040  075            .ASCII  /N =/
000717  040  040  040            .ASCII  /   /
000722  040  116  040            .ASCII  / N /
000725  075  040  040            .ASCII  /=  /
000730  040  040  116            .ASCII  /  N/
000733  040  075  040            .ASCII  / = /
000736  040  040  040            .ASCII  /   /
000741  116  040  075            .ASCII  /N =/
000744  040  040  040            .ASCII  /   /
000747  040  116  040            .ASCII  / N /
000752  075  040  040            .ASCII  /=  /
000755  040  040  116            .ASCII  /  N/
000760  040  075  040            .ASCII  / = /
000763  040  040  040            .ASCII  /   /
000766  116  040  075            .ASCII  /N =/
000771  040  040  040            .ASCII  /   /
000774  040  116  040            .ASCII  / N /
000777  075  000  000            .ASCII  /=/<00><00>
001002  045  116  045  P.AAN:     .ASCII  /%N%/
001005  101  040  116            .ASCII  /A N/
001010  117  056  040            .ASCII  /O. /
001013  040  040  117            .ASCII  /  O/
001016  116  114  131            .ASCII  /NLY/
001021  040  040  040            .ASCII  /   /
001024  040  061  040            .ASCII  / 1 /
001027  040  040  040            .ASCII  /   /
001032  040  040  062            .ASCII  /  2/
001035  040  040  040            .ASCII  /   /
001040  040  040  040            .ASCII  /   /
001043  063  040  040            .ASCII  /3  /
001046  040  040  040            .ASCII  /   /
001051  040  064  040            .ASCII  / 4 /
001054  040  040  040            .ASCII  /   /
001057  040  040  065            .ASCII  /  5/
001062  040  040  040            .ASCII  /   /
001065  040  040  040            .ASCII  /   /
001070  066  040  040            .ASCII  /6  /
001073  040  040  040            .ASCII  /   /
001076  040  067  040            .ASCII  / 7 /
001101  040  040  040            .ASCII  /   /
001104  040  040  070            .ASCII  /  8/
001107  000                      .ASCII  <00>
001110  045  116  045  P.AAO:     .ASCII  /%N%/
001113  101  055  055            .ASCII  /A--/
```

```
001116  055  055  040                .ASCII  /-- /
001121  040  055  055                .ASCII  / --/
001124  055  055  055                .ASCII  /---/
001127  040  040  055                .ASCII  /  -/
001132  055  055  055                .ASCII  /---/
001135  055  040  040                .ASCII  /-  /
001140  055  055  055                .ASCII  /---/
001143  055  055  040                .ASCII  /-- /
001146  040  055  055                .ASCII  / --/
001151  055  055  055                .ASCII  /---/
001154  040  040  055                .ASCII  /  -/
001157  055  055  055                .ASCII  /---/
001162  055  040  040                .ASCII  /-  /
001165  055  055  055                .ASCII  /---/
001170  055  055  040                .ASCII  /-- /
001173  040  055  055                .ASCII  / --/
001176  055  055  055                .ASCII  /---/
001201  040  040  055                .ASCII  /  -/
001204  055  055  055                .ASCII  /---/
001207  055  040  040                .ASCII  /-  /
001212  055  055  055                .ASCII  /---/
001215  055  055  045                .ASCII  /--%/
001220  116  000                     .ASCII  /N/<00>
001222  045  116  045          P.AAP: .ASCII  /%N%/
001225  104  063  045                .ASCII  /D3%/
001230  104  070  045                .ASCII  /D8%/
001233  104  067  045                .ASCII  /D7%/
001236  104  067  045                .ASCII  /D7%/
001241  104  067  000                .ASCII  /D7/<00>
001244  045  104  067          P.AAQ: .ASCII  /%D7/
001247  045  104  067                .ASCII  /%D7/
001252  045  104  067                .ASCII  /%D7/
001255  045  104  067                .ASCII  /%D7/
001260  045  104  067                .ASCII  /%D7/
001263  000                         .ASCII  <00>
001264  045  116  045          P.AAR: .ASCII  /%N%/
001267  101  040  040                .ASCII  /A  /
001272  040  040  040                .ASCII  /   /
001275  040  040  040                .ASCII  /   /
001300  116  117  056                .ASCII  /NO./
001303  040  040  040                .ASCII  /   /
001306  040  040  040                .ASCII  /   /
001311  040  116  117                .ASCII  / NO/
001314  056  040  040                .ASCII  /.  /
001317  040  040  040                .ASCII  /   /
001322  040  040  116                .ASCII  /  N/
001325  117  056  000                .ASCII  /O./<00>
001330  045  116  045          P.AAS: .ASCII  /%N%/
001333  101  125  116                .ASCII  /AUN/
001336  111  124  040                .ASCII  /IT /
001341  040  040  122                .ASCII  /  R/
001344  105  101  104                .ASCII  /EAD/
001347  123  040  040                .ASCII  /S  /
001352  040  040  040                .ASCII  /   /
001355  127  122  111                .ASCII  /WRI/
001360  124  105  123                .ASCII  /TES/
001363  040  040  040                .ASCII  /   /
```

```
001366  040  123  105                    .ASCII  / SE/
001371  105  113  123                    .ASCII  /EKS/
001374  040  040  040                    .ASCII  /   /
001377  040  115  102                    .ASCII  / MB/
001402  131  124  105                    .ASCII  /YTE/
001405  123  040  040                    .ASCII  /S  /
001410  115  102  131                    .ASCII  /MBY/
001413  124  105  123                    .ASCII  /TES/
001416  040  040  040                    .ASCII  /   /
001421  116  117  056                    .ASCII  /NO./
001424  040  110  101                    .ASCII  / HA/
001427  122  104  040                    .ASCII  /RD /
001432  040  116  117                    .ASCII  / NO/
001435  056  040  123                    .ASCII  /. S/
001440  117  106  124                    .ASCII  /OFT/
001443  000                            .ASCII  <00>
001444  045  116  045            P.AAT:  .ASCII  /%N%/
001447  101  040  116                    .ASCII  /A N/
001452  117  056  040                    .ASCII  /O. /
001455  040  050  113                    .ASCII  / (K/
001460  075  061  060                    .ASCII  /=10/
001463  060  060  051                    .ASCII  /00)/
001466  040  040  050                    .ASCII  /  (/
001471  113  075  061                    .ASCII  /K=1/
001474  060  060  060                    .ASCII  /000/
001477  051  040  040                    .ASCII  /)  /
001502  050  113  075                    .ASCII  /(K=/
001505  061  060  060                    .ASCII  /100/
001510  060  051  040                    .ASCII  /0) /
001513  040  040  122                    .ASCII  /  R/
001516  105  101  104                    .ASCII  /EAD/
001521  040  040  040                    .ASCII  /   /
001524  127  122  111                    .ASCII  /WRI/
001527  124  124  105                    .ASCII  /TTE/
001532  116  040  040                    .ASCII  /N  /
001535  040  105  122                    .ASCII  / ER/
001540  122  117  122                    .ASCII  /ROR/
001543  123  040  040                    .ASCII  /S  /
001546  040  040  105                    .ASCII  /  E/
001551  122  122  117                    .ASCII  /RRO/
001554  122  123  000                    .ASCII  /RS/<00>
001557  000                            .ASCII  <00>
001560  045  116  045            P.AAU:  .ASCII  /%N%/
001563  101  055  055                    .ASCII  /A--/
001566  055  055  040                    .ASCII  /-- /
001571  040  055  055                    .ASCII  / --/
001574  055  055  055                    .ASCII  /---/
001577  055  055  055                    .ASCII  /---/
001602  040  040  055                    .ASCII  /  -/
001605  055  055  055                    .ASCII  /---/
001610  055  055  055                    .ASCII  /---/
001613  055  040  040                    .ASCII  /-  /
001616  055  055  055                    .ASCII  /---/
001621  055  055  055                    .ASCII  /---/
001624  055  055  040                    .ASCII  /-- /
001627  040  055  055                    .ASCII  / --/
001632  055  055  055                    .ASCII  /---/
```

```
001635  055  040  040                .ASCII  /-  /
001640  055  055  055                .ASCII  /---/
001643  055  055  055                .ASCII  /---/
001646  055  040  040                .ASCII  /-  /
001651  055  055  055                .ASCII  /---/
001654  055  055  055                .ASCII  /---/
001657  055  055  040                .ASCII  /-- /
001662  040  055  055                .ASCII  / --/
001665  055  055  055                .ASCII  /---/
001670  055  055  055                .ASCII  /---/
001673  045  116  000                .ASCII  /%N/<00>
001676  045  104  071      P.AAV:    .ASCII  /%D9/
001701  045  104  070                .ASCII  /%D8/
001704  045  104  061                .ASCII  /%D1/
001707  060  045  104                .ASCII  /0%D/
001712  061  060  000                .ASCII  /10/<00>
001715  000                         .ASCII  <00>

                                    .GLOBL  PATCH, CPT, CST, CST.ADDR, DCT
                                    .GLOBL  DCT.ADDR, RC25.ADDR, IRC25.ADDR
                                    .GLOBL  DM.COMM, DMC.ADDR, RP.SAVE, RPS.X1
                                    .GLOBL  RPS.X2, OUTC.LIST, OUTC.TIMR, OCL.X1
                                    .GLOBL  OCL.X2, TALLY, T.ADDR, MSCP.ENV
                                    .GLOBL  ENV.USE, RETPKT, RP.USE, RP.INDX
                                    .GLOBL  RP.ADDR, BUFF.DESC, BUFF.OWN, IODQ
                                    .GLOBL  IODQ.IN, IODQ.OUT, ENTRY.REASON
                                    .GLOBL  T.FLAG, EOP.FLAG, MEM.MGMT, IIP.FLAG
                                    .GLOBL  CCTLR, CPLAT, CUOFF, CTLR.CNT
                                    .GLOBL  DUR, QIO, MEM.SIZE, FREE.MEM.ADDR
                                    .GLOBL  BUFF.SIZE, NUM.BUFF, CLK.TYPE
                                    .GLOBL  CLK.HERTZ, CLK.CSR, CLK.VECTOR
                                    .GLOBL  HOURS, MINUTES, SECONDS, TICKS
                                    .GLOBL  ST.CODE, SB.CODE, STEP, OF.RC
                                    .GLOBL  SA.REG, NEX, CRN, HWQ1, HWQ2, HWQ3
                                    .GLOBL  HWQ4, HWQ5, HWQ6, SWQ1, SWQ2, SWQ3
                                    .GLOBL  SWQ4, SWQ5, SWQ6, SWQ7, SWQ8, SWQ9
                                    .GLOBL  SWQ10, SWQ11, SWQ12, SWQ13, SWQ14
                                    .GLOBL  SWQ15, SWM1, MSG.01, MSG.07, CER.01
                                    .GLOBL  CER.02, CER.03, DUM.00, DUM.UC
                                    .GLOBL  DUM.CE, DUM.IE, DUM.HE, DUM.UE
                                    .GLOBL  EGS.01, EGS.02, EBS.01, EBD.10
                                    .GLOBL  EBD.12, EBD.13, EBD.14, EBD.15
                                    .GLOBL  EBD.16, EBD.17, EBD.18, EBD.19
                                    .GLOBL  EBD.20, EBD.21, EBD.22, EBH.31
                                    .GLOBL  EBH.32, EBH.34, EBH.35, EBH.36
                                    .GLOBL  EBH.37, EBH.38, EBH.39, EBH.40
                                    .GLOBL  EBH.41, EBH.42, EBH.43, EX.SA
                                    .GLOBL  EX.CRN, EX.SC, EX.DSC, EX.SB, EX.CMD
                                    .GLOBL  EX.SCC, EX.ONL, EX.ESP, EX.SND
                                    .GLOBL  EX.RD, EX.WRT, EX.CMP, EX.BB, EX.LBN
                                    .GLOBL  EX.CBC, EX.BC, EX.BD, EX.03, ETIME
                                    .GLOBL  PLATT, CRLF, SWP.ERROR, SWP.XFER
                                    .GLOBL  SWP.STRACK, SWP.ETRACK, SWP.FLAGS
                                    .GLOBL  L$LUN, L$UNIT
```

```
100000                  BIT15==         -100000
040000                  BIT14==         40000
020000                  BIT13==         20000
010000                  BIT12==         10000
004000                  BIT11==         4000
002000                  BIT10==         2000
001000                  BIT09==         1000
000400                  BIT08==         400
000200                  BIT07==         200
000100                  BIT06==         100
000040                  BIT05==         40
000020                  BIT04==         20
000010                  BIT03==         10
000004                  BIT02==         4
000002                  BIT01==         2
000001                  BIT00==         1
001000                  BIT9==          1000
000400                  BIT8==          400
000200                  BIT7==          200
000100                  BIT6==          100
000040                  BIT5==          40
000020                  BIT4==          20
000010                  BIT3==          10
000004                  BIT2==          4
000002                  BIT1==          2
000001                  BIT0==          1
000040                  EF.START==      40
000037                  EF.RESTART==    37
000036                  EF.CONTINUE==   36
000035                  EF.NEW==        35
000034                  EF.PWR==        34
000340                  PRI07==         340
000300                  PRI06==         300
000240                  PRI05==         240
000200                  PRI04==         200
000140                  PRI03==         140
000100                  PRI02==         100
000040                  PRI01==         40
000000                  PRI00==         0
000004                  EVL==           4
000010                  LOT==           10
000020                  ADR==           20
000040                  IDU==           40
000100                  ISR==           100
000200                  UAM==           200
000400                  BOE==           400
001000                  PNT==           1000
002000                  PRI==           2000
004000                  IXE==           4000
010000                  IBE==           10000
020000                  IER==           20000
040000                  LOE==           40000
100000                  HOE==           -100000
000056'                 L$HARD==        L$HRDLN+2
000146'                 L$SOFT==        L$SFTLN+2
000000'                 SR.HD=          P.AAA
000116'                 SR.ET=          P.AAB
```

```
000162'                 SR.MH1=         P.AAC
000256'                 SR.MH2=         P.AAD
000354'                 SR.MH3=         P.AAE
000450'                 SR.MH4=         P.AAF
000550'                 SR.UN=          P.AAG
000556'                 SR.LO=          P.AAH
000566'                 SR.HI=          P.AAI
000576'                 SR.ML=          P.AAJ
000614'                 SR.EH1=         P.AAK
000652'                 SR.EH2=         P.AAL
000672'                 SR.EH3=         P.AAM
001002'                 SR.EH4=         P.AAN
001110'                 SR.EH5=         P.AAO
001222'                 SR.EL1=         P.AAP
001244'                 SR.EL2=         P.AAQ
001264'                 SR.DH1=         P.AAR
001330'                 SR.DH2=         P.AAS
001444'                 SR.DH3=         P.AAT
001560'                 SR.DH4=         P.AAU
001676'                 SR.DL=          P.AAV


                        .SBTTL  HI.LO REPORT CODING SECTION
000370                  .PSECT  $CODE$,  RO

000000  013700  000366'  HI.LO:  MOV     ADDR.LO,R0                      ; *,ADDR.HI             1843
000004  062700  000002          ADD     #2,R0                           ; *,ADDR.HI
000010  005710                  TST     (R0)                            ; ADDR.HI               1844
000012  001011                  BNE     1$
000014  017746  177756          MOV     @ADDR.LO,-(SP)                  ;                       1846
000020  012746  000556'          MOV     #SR.LO,-(SP)
000024  012746  000002          MOV     #2,-(SP)
000030  010600                  MOV     SP,R0                           ; SP,*
000032  104416                  TRAP    16
000034  000407                  BR      2$                              ;                       1844
000036  011046          1$:     MOV     (R0),-(SP)                      ; ADDR.HI,*             1848
000040  012746  000566'          MOV     #SR.HI,-(SP)
000044  012746  000002          MOV     #2,-(SP)
000050  010600                  MOV     SP,R0                           ; SP,*
000052  104416                  TRAP    16
000054  062706  000006  2$:     ADD     #6,SP                           ;                       1838
000060  000207                  RTS     PC                              ;                       1826

; Routine Size:  25 words,      Routine Base:  $CODE$ + 0370
; Maximum stack depth per invocation:  5 words


;           1851
;           1852  !!!!!!!!!!!!!!! REPORT CODING SECTION - MAINLINE CODE !!!!!!!!!!!!!!!!!!!!!!!!!
;           1853
;           1854  LOCAL
;           1855      TA : REF BLOCK [TALLY_LEN] FIELD (T_FIELDS);        ! ADDRESS OF A UNIT'S TALLY BLOCK
;           1856
;           1857  PRINTS (SR_HD);                                        ! SUMMARY REPORT HEADER
;           1858  PRINTS (SR_ET, .HOURS, .MINUTES, .SECONDS);            ! ELAPSED TIME
;           1859  IF BIT_TST (SWP_FLAGS, SWF_DM)                         ! IF DM EXERCISER IS BEING RUN
;           1860  THEN                                                   ! THEN
```

```
1861        BEGIN
1862
1863        PRINTS (SR_DH1);                                ! PRINT HEADER FOR DM EXERCISER STATS
1864        PRINTS (SR_DH2);
1865        PRINTS (SR_DH3);
1866        PRINTS (SR_DH4);
1867        INCR UNIT FROM 0 TO (.L$UNIT - 1) DO            ! FOR EACH UNIT
1868            BEGIN
1869
1870            TA = TALLY + (.UNIT * TALLY_LEN * 2);       ! CALCULATE UNIT'S TALLY BLOCK ADDR
1871            PRINTS (SR_UN, .UNIT);                      ! PRINT UNIT NUMBER
1872
1873            ADDR_LO = TA [READ_LO];                     ! SET LOW-ORDER ADDR OF READ COUNT
1874            HI_LO ();                                   ! PRINT EITHER LOW- OR HIGH-ORDER FIELD
1875            ADDR_LO = .ADDR_LO + 4;                     ! ADVANCE TO LOW-ORDER ADDR OF WRITE COUNT
1876            HI_LO ();                                   ! PRINT EITHER LOW- OR HIGH-ORDER FIELD
1877            ADDR_LO = .ADDR_LO + 4;                     ! ADVANCE TO LOW-ORDER ADDR OF SEEK COUNT
1878            HI_LO ();                                   ! PRINT EITHER LOW- OR HIGH-ORDER FIELD
1879
1880            PRINTS (SR_DL, .TA [MB_READ], .TA [MB_WRIT], .TA [ER_HRD], .TA [ER_SFT]);
1881
1882            END;                                        ! UNIT LOOP
1883
1884        END
1885    ELSE                                                ! ELSE - MULTI-DRIVE SUBTEST IS BEING RUN
1886        BEGIN
1887
1888        PRINTS (SR_MH1);                                ! PRINT HEADER FOR MULTI-DRIVE
1889        PRINTS (SR_MH2);                                !     SUBTEST BASIC STATS
1890        PRINTS (SR_MH3);
1891        PRINTS (SR_MH4);
1892        INCR UNIT FROM 0 TO (.L$UNIT - 1) DO            ! FOR EACH UNIT
1893            BEGIN
1894
1895            TA = TALLY + (.UNIT * TALLY_LEN * 2);       ! CALCULATE UNIT'S TALLY BLOCK ADDR
1896            PRINTS (SR_UN, .UNIT);                      ! PRINT UNIT NUMBER
1897
1898            ADDR_LO = TA [READ_LO];                     ! SET LOW-ORDER ADDR OF READ COUNT
1899            HI_LO ();                                   ! PRINT EITHER LOW- OR HIGH-ORDER FIELD
1900            ADDR_LO = .ADDR_LO + 4;                     ! ADVANCE TO LOW-ORDER ADDR OF WRITE COUNT
1901            HI_LO ();                                   ! PRINT EITHER LOW- OR HIGH-ORDER FIELD
1902
1903            PRINTS (SR_ML, .TA [MB_READ], .TA [MB_WRIT], .TA [ER_HRD], .TA [ER_LOG]);
1904
1905            END;                                        ! UNIT LOOP
1906
1907        PRINTS (SR_EH1);                                ! PRINT HEADER FOR ECC ERROR STATS
1908        PRINTS (SR_EH2);
1909        PRINTS (SR_EH3);
1910        PRINTS (SR_EH4);
1911        PRINTS (SR_EH5);
1912        INCR UNIT FROM 0 TO (.L$UNIT - 1) DO            ! FOR EACH UNIT
1913            BEGIN
1914
1915            TA = TALLY + (.UNIT * TALLY_LEN * 2);       ! CALCULATE UNIT'S TALLY BLOCK ADDR
1916            PRINTS (SR_EL1, .UNIT, .TA [ECC_ONLY], .TA [ECC_1], .TA [ECC_2], .TA [ECC_3]);
1917            PRINTS (SR_EL2, .TA [ECC_4], .TA [ECC_5], .TA [ECC_6], .TA [ECC_7], .TA [ECC_8]);
```

```
;       1918
;       1919            END;                                    ! UNIT LOOP
;       1920
;       1921        END;
;       1922
;       1923    PRINTS (CRLF);                                  ! PRINT <CR><LF>
;       1924
;       1925    ENDRPT;


                                        .SBTTL  LRPT REPORT CODING SECTION
000000  004137  000000G           LRPT:  JSR     R1,$SAVE3                       ;                       1788
000004  012746  000000'                  MOV     #SR.HD,-(SP)                    ;                       1857
000010  012746  000001                   MOV     #1,-(SP)
000014  010600                           MOV     SP,R0                           ; SP,*
000016  104416                           TRAP    16
000020  013716  000000G                  MOV     SECONDS,(SP)                    ;                       1858
000024  013746  000000G                  MOV     MINUTES,-(SP)
000030  013746  000000G                  MOV     HOURS,-(SP)
000034  012746  000116'                  MOV     #SR.ET,-(SP)
000040  012746  000004                   MOV     #4,-(SP)
000044  010600                           MOV     SP,R0                           ; SP,*
000046  104416                           TRAP    16
000050  132737  000002  000000G         BITB    #2,SWP.FLAGS                    ;                       1859
000056  001516                           BEQ     3$
000060  012716  001264'                  MOV     #SR.DH1,(SP)                    ;                       1863
000064  012746  000001                   MOV     #1,-(SP)
000070  010600                           MOV     SP,R0                           ; SP,*
000072  104416                           TRAP    16
000074  012716  001330'                  MOV     #SR.DH2,(SP)                    ;                       1864
000100  012746  000001                   MOV     #1,-(SP)
000104  010600                           MOV     SP,R0                           ; SP,*
000106  104416                           TRAP    16
000110  012716  001444'                  MOV     #SR.DH3,(SP)                    ;                       1865
000114  012746  000001                   MOV     #1,-(SP)
000120  010600                           MOV     SP,R0                           ; SP,*
000122  104416                           TRAP    16
000124  012716  001560'                  MOV     #SR.DH4,(SP)                    ;                       1866
000130  012746  000001                   MOV     #1,-(SP)
000134  010600                           MOV     SP,R0                           ; SP,*
000136  104416                           TRAP    16
000140  013703  000000G                  MOV     L$UNIT,R3                       ;                       1867
000144  005001                           CLR     R1                              ; UNIT
000146  000456                           BR      2$
000150  010116                   1$:     MOV     R1,(SP)                         ; UNIT,*                1870
000152  012746  000060                   MOV     #60,-(SP)
000156  004737  000000G                  JSR     PC,BL$MUL
000162  062700  000000G                  ADD     #TALLY,R0
000166  010002                           MOV     R0,R2                           ; *,TA
000170  010116                           MOV     R1,(SP)                         ; UNIT,*                1871
000172  012746  000550'                  MOV     #SR.UN,-(SP)
000176  012746  000002                   MOV     #2,-(SP)
000202  010600                           MOV     SP,R0                           ; SP,*
000204  104416                           TRAP    16
000206  010237  000366'                  MOV     R2,ADDR.LO                      ; TA,*                  1873
000212  004737  000370'                  JSR     PC,HI.LO                        ;                       1874
000216  062737  000004  000366'         ADD     #4,ADDR.LO                      ;                       1875
```

```
000224  004737  000370'                    JSR     PC,HI.LO                ;                               1876
000230  062737  000004  000366'            ADD     #4,ADDR.LO              ;                               1877
000236  004737  000370'                    JSR     PC,HI.LO                ;                               1878
000242  016216  000034                      MOV     34(R2),(SP)             ; *(TA),*                       1880
000246  016246  000030                      MOV     30(R2),-(SP)            ; *(TA),*
000252  016246  000026                      MOV     26(R2),-(SP)            ; *(TA),*
000256  016246  000020                      MOV     20(R2),-(SP)            ; *(TA),*
000262  012746  001676'                    MOV     #SR.DL,-(SP)
000266  012746  000005                      MOV     #5,-(SP)
000272  010600                              MOV     SP,R0                   ; SP,*
000274  104416                              TRAP    16
000276  062706  000020                      ADD     #20,SP                  ;                               1868
000302  005201                              INC     R1                      ; UNIT                          1867
000304  020103                  2$:         CMP     R1,R3                   ; UNIT,*
000306  002720                              BLT     1$
000310  000137  001444'                    JMP     8$                      ;                               1859
000314  012716  000162'            3$:     MOV     #SR.MH1,(SP)            ;                               1888
000320  012746  000001                      MOV     #1,-(SP)
000324  010600                              MOV     SP,R0                   ; SP,*
000326  104416                              TRAP    16
000330  012716  000256'                    MOV     #SR.MH2,(SP)            ;                               1889
000334  012746  000001                      MOV     #1,-(SP)
000340  010600                              MOV     SP,R0                   ; SP,*
000342  104416                              TRAP    16
000344  012716  000354'                    MOV     #SR.MH3,(SP)            ;                               1890
000350  012746  000001                      MOV     #1,-(SP)
000354  010600                              MOV     SP,R0                   ; SP,*
000356  104416                              TRAP    16
000360  012716  000450'                    MOV     #SR.MH4,(SP)            ;                               1891
000364  012746  000001                      MOV     #1,-(SP)
000370  010600                              MOV     SP,R0                   ; SP,*
000372  104416                              TRAP    16
000374  013703  000000G                     MOV     L$UNIT,R3               ;                               1892
000400  005001                              CLR     R1                      ; UNIT
000402  000451                              BR      5$
000404  010116                  4$:         MOV     R1,(SP)                 ; UNIT,*                        1895
000406  012746  000060                      MOV     #60,-(SP)
000412  004737  000000G                     JSR     PC,BL$MUL
000416  062700  000000G                     ADD     #TALLY,R0
000422  010002                              MOV     R0,R2                   ; *,TA
000424  010116                              MOV     R1,(SP)                 ; UNIT,*                        1896
000426  012746  000550'                    MOV     #SR.UN,-(SP)
000432  012746  000002                      MOV     #2,-(SP)
000436  010600                              MOV     SP,R0                   ; SP,*
000440  104416                              TRAP    16
000442  010237  000366'                    MOV     R2,ADDR.LO              ; TA,*                          1898
000446  004737  000370'                    JSR     PC,HI.LO                ;                               1899
000452  062737  000004  000366'            ADD     #4,ADDR.LO              ;                               1900
000460  004737  000370'                    JSR     PC,HI.LO                ;                               1901
000464  016216  000032                      MOV     32(R2),(SP)             ; *(TA),*                       1903
000470  016246  000030                      MOV     30(R2),-(SP)            ; *(TA),*
000474  016246  000026                      MOV     26(R2),-(SP)            ; *(TA),*
000500  016246  000020                      MOV     20(R2),-(SP)            ; *(TA),*
000504  012746  000576'                    MOV     #SR.ML,-(SP)
000510  012746  000005                      MOV     #5,-(SP)
000514  010600                              MOV     SP,R0                   ; SP,*
000516  104416                              TRAP    16
```

```
000520  062706  000020                      ADD     #20,SP                  ;                               1893
000524  005201                              INC     R1                      ; UNIT                          1892
000526  020103                      5$:     CMP     R1,R3                   ; UNIT,*
000530  002725                              BLT     4$
000532  012716  000614'                      MOV     #SR.EH1,(SP)            ;                               1907
000536  012746  000001                      MOV     #1,-(SP)
000542  010600                              MOV     SP,R0                   ; SP,*
000544  104416                              TRAP    16
000546  012716  000652'                      MOV     #SR.EH2,(SP)            ;                               1908
000552  012746  000001                      MOV     #1,-(SP)
000556  010600                              MOV     SP,R0                   ; SP,*
000560  104416                              TRAP    16
000562  012716  000672'                      MOV     #SR.EH3,(SP)            ;                               1909
000566  012746  000001                      MOV     #1,-(SP)
000572  010600                              MOV     SP,R0                   ; SP,*
000574  104416                              TRAP    16
000576  012716  001002'                      MOV     #SR.EH4,(SP)            ;                               1910
000602  012746  000001                      MOV     #1,-(SP)
000606  010600                              MOV     SP,R0                   ; SP,*
000610  104416                              TRAP    16
000612  012716  001110'                      MOV     #SR.EH5,(SP)            ;                               1911
000616  012746  000001                      MOV     #1,-(SP)
000622  010600                              MOV     SP,R0                   ; SP,*
000624  104416                              TRAP    16
000626  013703  000000G                     MOV     L$UNIT,R3               ;                               1912
000632  005001                              CLR     R1                      ; UNIT
000634  000452                              BR      7$
000636  010116                      6$:     MOV     R1,(SP)                 ; UNIT,*                        1915
000640  012746  000060                      MOV     #60,-(SP)
000644  004737  000000G                     JSR     PC,BL$MUL
000650  062700  000000G                     ADD     #TALLY,R0
000654  010002                              MOV     R0,R2                   ; *,TA
000656  016216  000042                      MOV     42(R2),(SP)             ; *(TA),*                       1916
000662  016246  000040                      MOV     40(R2),-(SP)            ; *(TA),*
000666  016246  000036                      MOV     36(R2),-(SP)            ; *(TA),*
000672  016246  000056                      MOV     56(R2),-(SP)            ; *(TA),*
000676  010146                              MOV     R1,-(SP)                ; UNIT,*
000700  012746  001222'                      MOV     #SR.EL1,-(SP)
000704  012746  000006                      MOV     #6,-(SP)
000710  010600                              MOV     SP,R0                   ; SP,*
000712  104416                              TRAP    16
000714  016216  000054                      MOV     54(R2),(SP)             ; *(TA),*                       1917
000720  016246  000052                      MOV     52(R2),-(SP)            ; *(TA),*
000724  016246  000050                      MOV     50(R2),-(SP)            ; *(TA),*
000730  016246  000046                      MOV     46(R2),-(SP)            ; *(TA),*
000734  016246  000044                      MOV     44(R2),-(SP)            ; *(TA),*
000740  012746  001244'                      MOV     #SR.EL2,-(SP)
000744  012746  000006                      MOV     #6,-(SP)
000750  010600                              MOV     SP,R0                   ; SP,*
000752  104416                              TRAP    16
000754  062706  000032                      ADD     #32,SP                  ;                               1913
000760  005201                              INC     R1                      ; UNIT                          1912
000762  020103                      7$:     CMP     R1,R3                   ; UNIT,*
000764  002724                              BLT     6$
000766  062706  000012                      ADD     #12,SP                  ;                               1886
000772  012716  000000G             8$:     MOV     #CRLF,(SP)              ;                               1923
000776  012746  000001                      MOV     #1,-(SP)
```

```
001002  010600                          MOV     SP,R0                           ; SP,*
001004  104416                          TRAP    16
001006  062706  000026                  ADD     #26,SP                          ;                                       1788
001012  000207                          RTS     PC

; Routine Size:  262 words,     Routine Base:  $CODE$ + 0452
; Maximum stack depth per invocation:  34 words




                                        .SBTTL  L$RPT REPORT CODING SECTION
000000  004737  000452'          L$RPT:: JSR     PC,LRPT                         ;                                       1923
000004  104425                          TRAP    25
000006  000207                          RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 1466
; Maximum stack depth per invocation:  2 words
```

```
1926  %SBTTL 'INITIALIZE SECTION'
1927
1928  !+
1929  !       THE INITIALIZE CODE IS EXECUTED UNDER FIVE CONDITIONS. THERE ARE
1930  !       SUPERVISOR EVENT FLAGS THAT ARE USED TO LET THE DIAGNOSTIC KNOW UNDER
1931  !       WHICH CONDITION THE EXECUTION IS TAKING PLACE. THE EVENT FLAGS ARE
1932  !       READ USING THE "READEF" MACRO. THE CONDITIONS UNDER WHICH THE INIT CODE
1933  !       IS EXECUTED AND THE CORRRESPONDING EVENT FLAGS ARE:
1934  !               START COMMAND           EF.START
1935  !               RESTART COMMAND         EF.RESTART
1936  !               CONTINUE COMMAND        EF.CONTINUE
1937  !               POWERDOWN/POWERUP       EF.PWR
1938  !               NEW PASS                EF.NEW
1939  !       EXAMPLE OF EVENT FLAG USE:
1940  !               IF READEF(EF.START) THEN START_FLAG = 1;
1941  !       DURING THE INIT CODE, THE "GPHARD" MACRO IS USED TO OBTAIN P-TABLE
1942  !       INFORMATION FOR ALL DEVICES. THE NUMBER OF UNITS AVAILABLE IS IN THE
1943  !       HEADER WORD "L$UNIT".
1944  !-
1945
1946  BGNINIT;
1947
1948  LOCAL
1949      FLAG1 : BYTE,
1950      FLAG2 : BYTE,
1951      DELAY_MULT : WORD,
1952      CLK_ADR,
1953      HWPT_REF : REF BLOCK [HWPT_LEN, WORD] FIELD (HWP_FIELDS);
1954
1955  SETPRI (PRI07);                       ! PRIORITY 7 - NO INTERRUPTS ALLOWED DURING INIT
1956
1957  IF READEF (EF_NEW)                    ! IS THIS A NEW PASS? (ASK NEW_PASS
1958  THEN                                  !       FIRST IN CASE MULTIPLE FLAGS ARE
1959      ENTRY_REASON = NEW_PASS;          !       SET, E.G., START AND NEW_PASS)
1960
1961  IF READEF (EF_CONTINUE)               ! IS THIS A CONTINUE?
1962  THEN
1963      ENTRY_REASON = CONT;
1964
1965  IF READEF (EF_PWR)                    ! IS THIS A POWER FAIL?
1966  THEN
1967      BEGIN
1968      ENTRY_REASON = PWR_FAIL;
1969      IF MANUAL                         ! IF ATTENDED
1970      THEN                              ! THEN
1971          PRINTF (MSG_01);              ! "POWER DELAY - WAITING"
1972
1973      INCR COUNT FROM 0 TO 60 DO        ! WAIT APPROX. 60 SECONDS
1974          BEGIN
1975          DELAY_MULT = 333;
1976          DELAY (.DELAY_MULT);
1977          BREAK;                        ! BREAK FOR ACT
1978          END;
1979
1980      END;
1981
1982  IF READEF (EF_RESTART)                ! IS THIS A RESTART?
```

```
;       1983  THEN
;       1984      ENTRY_REASON = RESTART;
;       1985
;       1986  IF READEF (EF_START)                        ! IS THIS A START?
;       1987  THEN
;       1988      BEGIN
;       1989      ENTRY_REASON = START;
;       1990      HOURS = MINUTES = SECONDS = TICKS = 0;      ! INIT ELAPSED TIME TO 00:00:00:00
;       1991      END;
;       1992
;       1993  !+
;       1994  !       MAKE SURE THAT NOT MORE THAN MAX_UNITS HAVE BEEN SPECIFIED.
;       1995  !       IF THERE ARE TOO MANY, NOTIFY USER AND RETURN TO SUPERVISOR.
;       1996  !       (DIAGNOSTIC IS ABORTED).
;       1997  !-
;       1998
;       1999  IF .L$UNIT GTRU MAX_UNITS
;       2000  THEN
;       2001      BEGIN
;       2002      ERRSF (1, EGS_01, EMS_01);                ! "TOO MANY UNITS"
;       2003      DOCLN;
;       2004      END;
;       2005
;       2006  !+
;       2007  !       CHECK TO MAKE SURE THERE IS A CLOCK ON THE SYSTEM.  IF NO CLOCK,
;       2008  !       THEN ABORT TO SUPERVISOR. OTHERWISE, DETERMINE WHETHER CLOCK IS
;       2009  !       AN L OR P CLOCK, AND SET UP PARAMETERS.
;       2010  !-
;       2011
;       2012  CLK_TYPE = NO_CLOCK;                        ! SET FLAG FOR NO CLOCK
;       2013
;       2014  IF CLOCK (P, CLK_ADR)                       ! IS THERE A P_CLOCK?
;       2015  THEN
;       2016      BEGIN
;       2017      CLK_TYPE = P_CLOCK;                     ! SET TYPE TO P_CLOCK
;       2018      CLK_CSR = ..CLK_ADR;                    ! SAVE THE CSR ADDRESS
;       2019      .CLK_CSR = %O'105';                     ! ENABLE INTERRUPTS
;       2020      END
;       2021  ELSE
;       2022      BEGIN
;       2023
;       2024      IF CLOCK (L, CLK_ADR)                   ! IS THERE AN L_CLOCK?
;       2025      THEN
;       2026          BEGIN
;       2027          CLK_TYPE = L_CLOCK;                 ! SET TYPE TO L_CLOCK
;       2028          CLK_CSR = ..CLK_ADR;                ! SAVE THE CSR ADDRESS
;       2029          .CLK_CSR = %O'100';                 ! ENABLE INTERRUPTS
;       2030          END;
;       2031
;       2032      END;
;       2033
;       2034  IF .CLK_TYPE EQLU NO_CLOCK                  ! IF NO CLOCK WAS FOUND
;       2035  THEN
;       2036      BEGIN
;       2037      ERRSF (2, EGS_02, 0);                   ! "NEITHER P NOR L CLOCK WAS FOUND"
;       2038      DOCLN;                                  ! ABORT THE PROGRAM
;       2039      END
```

```
2040  ELSE
2041      BEGIN
2042      CLK_VECTOR = .(.CLK_ADR + 4);                ! CLOCK VECTOR ADDRESS
2043      CLK_HERTZ = .(.CLK_ADR + 6);                 ! CLOCK HERTZ RATE
2044      SETVEC (.CLK_VECTOR, CLK_INT_SERV, PRI06);   ! SET CLOCK VECTOR ADDR
2045      END;
2046
2047  !+
2048  !       THE FOLLOWING CODE IS EXECUTED FOR ALL ENTRY REASONS EXCEPT NEW_PASS.
2049  !       ALL RUN-TIME CONTROLLER STATUS TABLES (CST'S) ARE CLEARED TO 0, THEN
2050  !       LOADED WITH CONFIGURATION DATA FROM THE HARDWARE P-TABLES.
2051  !-
2052
2053  IF .ENTRY_REASON NEQU NEW_PASS
2054  THEN
2055      BEGIN
2056
2057      INCR COUNT FROM 0 TO ((MAX_CTLR * CST_LEN * 2) - 2) BY 2 DO
2058          (CST + .COUNT) = 0;
2059
2060      INCR UNIT FROM 0 TO (.L$UNIT - 1) DO         ! LOOP THROUGH ALL UNITS
2061          BEGIN
2062
2063          CPT [.UNIT] = NO;                        ! INIT CURRENT PASS TESTING VECTOR
2064          IF GPHARD (.UNIT, HWPT_REF) NEQA 0       ! IF HWP TABLE FOUND
2065          THEN
2066              BEGIN
2067
2068              FLAG1 = NOT_FOUND;                   ! NO EXISTING IP ADDRESS MATCH YET
2069              INCR CTLR FROM 0 TO (MAX_CTLR - 1) DO ! LOOP THROUGH ALL CST'S
2070                  BEGIN
2071
2072                  IF .CST [.CTLR, IP_ADDR] EQLA .HWPT_REF [HWP_IP_ADDR]
2073                  THEN                             ! IF IP ADDR ALREADY EXISTS
2074                      BEGIN
2075
2076                      FLAG1 = FOUND;
2077                      FLAG2 = NOT_FOUND;           ! FLAG INDICATING PLATTER SLOT AVAILABILITY
2078                      INCR OFFSET FROM (0 + OF_UN) TO (3 + OF_UN) DO ! LOOP THROUGH EACH PLATTER SLOT
2079                          BEGIN
2080
2081                          IF .CST [.CTLR, .OFFSET, P_PRES] EQLU NOT_PRESENT
2082                          THEN                     ! IF EMPTY SLOT FOUND
2083                              BEGIN
2084
2085                              FLAG2 = FOUND;
2086                              CST [.CTLR, .OFFSET, ALLBIT] = .HWPT_REF [HWP_PLAT];
2087                                                   ! COPY PLATTER ADDR AND PROT BIT
2088                              CST [.CTLR, .OFFSET, P_UNIT] = .UNIT;
2089                              CST [.CTLR, .OFFSET, P_PRES] = PRESENT;
2090                              EXITLOOP;
2091
2092                              END                  ! END - IF PLATTER SLOT FOUND
2093                          ELSE                     ! OTHERWISE - SLOT IS OCCUPIED
2094                              BEGIN
2095
2096                              IF .CST [.CTLR, .OFFSET, P_ADDR] EQLU .HWPT_REF [HWP_PLAT_ADDR]
```

```
;        2097                            THEN
;        2098                                BEGIN
;        2099
;        2100                                FLAG2 = FOUND;                          ! PLATTER ADDR SLOT ALREADY EXISTS
;        2101                                IF MANUAL                               ! IF ATTENDED
;        2102                                THEN                                    ! THEN
;      P 2103                                    PRINTF (CER_01, .HWPT_REF [HWP_PLAT_ADDR],
;        2104                                        .HWPT_REF [HWP_IP_ADDR]);       ! 'DUPLICATE PLATTER ADDRESS XXX. AT IP XXXX
;        2105                                DUR [.UNIT] = DU_CONF;                  ! LOAD REASON FOR DROPPING UNIT
;        2106                                DODU (.UNIT);                           ! DROP UNIT
;        2107                                EXITLOOP;                               ! DON'T LOOK AT ANY MORE SLOTS
;        2108
;        2109                                END;                            ! IF PLATTER ADDR ALREADY EXISTS
;        2110
;        2111                            END;                            ! IF SLOT IS OCCUPIED
;        2112
;        2113                        END;                            ! PLATTER SLOT SEARCH LOOP
;        2114
;        2115                    IF .FLAG2 EQLU NOT_FOUND    ! IF PLATTER SLOT NOT AVAILABLE
;        2116                    THEN
;        2117                        BEGIN
;        2118
;        2119                        IF MANUAL                                   ! IF ATTENDED
;        2120                        THEN                                        ! THEN
;        2121                            PRINTF (CER_02, .HWPT_REF [HWP_IP_ADDR]); ! "ALREADY 4 UNITS AT IP XXXXXX(O)"
;        2122                        DUR [.UNIT] = DU_CONF;                      ! CONFIGURATION ERROR
;        2123                        DODU (.UNIT);                               ! DROP UNIT
;        2124
;        2125                            END;
;        2126
;        2127                        EXITLOOP;
;        2128
;        2129                            END;                            ! IF EXISTING CST TABLE FOUND
;        2130
;        2131                    END;                            ! CST TABLE SEARCH LOOP
;        2132
;        2133                IF .FLAG1 EQLU NOT_FOUND            ! IF NO IP ADDR MATCH TO EXISTING CST
;        2134                THEN
;        2135                    BEGIN
;        2136
;        2137                    FLAG2 = NOT_FOUND;              ! FLAG INDICATING EMPTY CST AVAILABILITY
;        2138                    INCR CTLR FROM 0 TO (MAX_CTLR - 1) DO ! LOOP THROUGH EACH CST
;        2139                        BEGIN
;        2140
;        2141                        IF .CST [.CTLR, IP_ADDR] EQLA 0 ! IF EMPTY CST FOUND
;        2142                        THEN
;        2143                            BEGIN
;        2144
;        2145                            FLAG2 = FOUND;
;        2146                            CST [.CTLR, IP_ADDR] = .HWPT_REF [HWP_IP_ADDR];
;        2147                            CST [.CTLR, VEC_ADDR] = .HWPT_REF [HWP_VECTOR];
;        2148                            CST [.CTLR, BR_LEV] = .HWPT_REF [HWP_BR_LEVEL];
;        2149
;        2150                            CST [.CTLR, P1_ALL] = .HWPT_REF [HWP_PLAT]; !PLAT ADDR, PROT BIT
;        2151                            CST [.CTLR, P1_UNIT] = .UNIT;
;        2152                            CST [.CTLR, P1_PRES] = PRESENT;
;        2153                            EXITLOOP;
```

```
;        2154
;        2155                        END;                        ! IF EMPTY CST FOUND
;        2156
;        2157                    END;                            ! EMPTY CST SEARCH LOOP
;        2158
;        2159                IF .FLAG2 EQLU NOT_FOUND            ! IF NO EMPTY CST FOUND
;        2160                THEN
;        2161                    BEGIN
;        2162
;        2163                    IF MANUAL                       ! IF ATTENDED
;        2164                    THEN                            ! THEN
;        2165                        PRINTF (CER_03, MAX_CTLR); ! 'MORE THAN X DIFFERENT IP ADDRESSES."
;        2166                    DUR [.UNIT] = DU_CONF;          ! CONFIGURATION ERROR
;        2167                    DODU (.UNIT);                   ! DROP UNIT
;        2168
;        2169                    END;
;        2170
;        2171                END;                                ! IF NO IP ADDR MATCH IN CST
;        2172
;        2173            END;                                    ! IF GPHARD RETURNED A HWP TABLE
;        2174
;        2175        END;                                        ! UNIT LOOP
;        2176
;        2177    END
;        2178ELSE                                                ! OTHERWISE, FOR EACH NEW PASS
;        2179    BEGIN
;        2180
;        2181    INCR UNIT FROM 0 TO (.L$UNIT - 1) DO            ! FOR EACH UNIT
;        2182        BEGIN
;        2183
;        2184        IF GPHARD (.UNIT, HWPT_REF) NEQA 0          ! IF UNIT HAS NOT BEEN DROPPED
;        2185        THEN                                        ! THEN
;        2186            BEGIN
;        2187
;        2188            CPT [.UNIT] = YES;                              ! O.K. TO TEST UNIT
;        2189            INCR CTLR FROM 0 TO (MAX_CTLR - 1) DO           ! FIND UNIT'S CONTROLLER
;        2190                BEGIN
;        2191
;        2192                IF .CST [.CTLR, IP_ADDR] EQLA .HWPT_REF [HWP_IP_ADDR]
;        2193                THEN
;        2194                    BEGIN                               ! FOUND IT
;        2195
;        2196                    CST [.CTLR, U_CNT] = .CST [.CTLR, U_CNT] + 1;       ! INCREMENT NO. OF TESTABLE UNITS
;        2197                    EXITLOOP;                                           ! DONE
;        2198
;        2199                    END;
;        2200
;        2201                END;                                    ! CONTROLLER LOOP
;        2202
;        2203            END;                                ! IF UNIT NOT DROPPED
;        2204
;        2205        END;                                    ! UNIT LOOP
;        2206
;        2207    END;                                        ! END - IF NEW PASS
;        2208
;        2209IF .ENTRY_REASON LEQU RESTART                     ! IF START OR RESTART
;        2210THEN                                            ! THEN
```

```
2211      BEGIN
2212
2213      CRN = 0;                                  ! COMMAND REFERENCE NUMBER
2214      CTLR_CNT = 0;                             ! NUMBER OF CONFIGURED CONTROLLERS
2215      INCR CTLR FROM 0 TO (MAX_CTLR - 1) DO
2216          BEGIN
2217
2218          IF .CST [.CTLR, IP_ADDR] NEQA 0       ! IF CONTROLLER IS PRESENT
2219          THEN                                  ! THEN
2220              CTLR_CNT = .CTLR_CNT + 1;         ! INCREMENT CONTROLLER COUNT
2221
2222          END;
2223
2224      MEMORY (FREE_MEM_ADDR);                   ! GET START OF FREE MEMORY
2225      MEM_SIZE = ..FREE_MEM_ADDR;               ! GET FREE MEMORY SIZE
2226
2227      INCR COUNT FROM 0 TO ((MAX_UNITS * TALLY_LEN) - 1) DO       ! INITIALIZE
2228          TALLY [.COUNT] = 0;                                     ! STATISTICS
2229
2230      END;                                      ! END OF START/RESTART INITIALIZATION
2231
2232  !+
2233  !       MISCELLANEOUS INITIALIZATON
2234  !-
2235
2236  INCR CTLR FROM 0 TO (MAX_CTLR - 1) DO         ! INITIALIZE NO. OF OUTSTANDING QIOS
2237      QIO [.CTLR] = 0;
2238  INCR COUNT FROM 0 TO (RP_CNT - 1) DO          ! INITIALIZE RETURN PACKET POOL
2239      RP_USE [.COUNT] = -1;
2240  IODQ_IN = IODQ_OUT = 0;                       ! INITIALIZE I/O DONE QUEUE POINTERS
2241  IF (BIT_TST (SWP_FLAGS, SWF_HWC)) AND         ! IF USER CHANGED FROM HOST WRITE-CHECKS TO
2242      (BIT_TST (SWP_FLAGS, SWF_CWC))            !     CONTROLLER WRITE-CHECKS
2243  THEN                                          ! THEN (BOTH BITS WOULD BE SET)
2244      SWP_FLAGS <SWF_PHWC,1> = 0;               ! CLEAR HOST WRITE-CHECKS
2245
2246  !+
2247  !       THE FOLLOWING SITUATION (STARTING TRACK NUMBER GREATER THAN ENDING
2248  !       TRACK NUMBER) CAN OCCUR DUE TO ANOTHER DRS BUG. IF THE USER ENTERS LOW
2249  !       TRACK LIMITS FOR ONE PASS, THEN ENTERS A HIGH STARTING TRACK NUMBER
2250  !       AFTER A RESTART OR CONTINUE COMMAND, DRS WILL NOT GIVE AN ERROR IF THE
2251  !       ENDING TRACK NUMBER IS DEFAULTED TO THE PREVIOUS LOW TRACK NUMBER. AN
2252  !       ERRONEOUS LBN WOULD BE GENERATED IN QIO_LBN IF THIS SITUATION WERE
2253  !       ALLOWED.
2254  !-
2255
2256  IF .SWP_STRACK GEQU .SWP_ETRACK
2257  THEN
2258      BEGIN
2259
2260      LOCAL
2261          TEMP : WORD;
2262
2263      TEMP = .SWP_STRACK;                       ! REVERSE STARTING AND ENDING
2264      SWP_STRACK = .SWP_ETRACK;                 !     TRACK NUMBERS
2265      SWP_ETRACK = .TEMP;
2266
2267      END;
```

```
;        2268
;        2269  SETPRI (PRI00);                    ! SET PROGRAM PRIORITY TO 0
;        2270
;        2271  ENDINIT;

                                        .GLOBL  L$DLY

                                        .TTL    LINIT INITIALIZE SECTION
000000  004137  000000G          LINIT: JSR     R1,$SAVE5                       ;                       1925
000004  162706  000010                 SUB     #10,SP
000010  012700  000340                 MOV     #340,R0                         ;                       1955
000014  104441                         TRAP    41
000016  012700  000035                 MOV     #35,R0                          ;                       1957
000022  104447                         TRAP    47
000024  103003                         BHIS    1$
000026  112737  000005  000000G        MOVB    #5,ENTRY.REASON                 ;                       1959
000034  012700  000036          1$:    MOV     #36,R0                          ;                       1961
000040  104447                         TRAP    47
000042  103003                         BHIS    2$
000044  112737  000003  000000G        MOVB    #3,ENTRY.REASON                 ;                       1963
000052  012700  000034          2$:    MOV     #34,R0                          ;                       1965
000056  104447                         TRAP    47
000060  103036                         BHIS    9$
000062  112737  000004  000000G        MOVB    #4,ENTRY.REASON                 ;                       1968
000070  104450                         TRAP    50                              ;                       1969
000072  103007                         BHIS    3$
000074  012746  000000G                MOV     #MSG.01,-(SP)                   ;                       1971
000100  012746  000001                 MOV     #1,-(SP)
000104  010600                         MOV     SP,R0                           ; SP,*
000106  104417                         TRAP    17
000110  022626                         CMP     (SP)+,(SP)+
000112  012702  000075          3$:    MOV     #75,R2                          ; *,COUNT               1973
000116  012703  000515          4$:    MOV     #515,R3                         ; *,DELAY.MULT          1975
000122  010301                         MOV     R3,R1                           ; DELAY.MULT,$$TMP2     1976
000124  001411                 5$:     BEQ     8$
000126  013700  000000G                MOV     L$DLY,R0                        ; *,$$TMP1
000132  001404                         BEQ     7$
000134  005066  000006          6$:    CLR     6(SP)                           ; $$TMP
000140  005300                         DEC     R0                              ; $$TMP1
000142  001374                         BNE     6$
000144  005301                 7$:     DEC     R1                              ; $$TMP2
000146  000766                         BR      5$
000150  104422                 8$:     TRAP    22
000152  005302                         DEC     R2                              ; COUNT                 1973
000154  001360                         BNE     4$
000156  012700  000037          9$:    MOV     #37,R0                          ;                       1982
000162  104447                         TRAP    47
000164  103003                         BHIS    10$
000166  112737  000002  000000G        MOVB    #2,ENTRY.REASON                 ;                       1984
000174  012700  000040          10$:   MOV     #40,R0                          ;                       1986
000200  104447                         TRAP    47
000202  103013                         BHIS    11$
000204  112737  000001  000000G        MOVB    #1,ENTRY.REASON                 ;                       1989
000212  005037  000000G                CLR     TICKS                           ;                       1990
000216  005037  000000G                CLR     SECONDS
```

```
000222  005037  000000G                        CLR     MINUTES
000226  005037  000000G                        CLR     HOURS
000232  023727  000000G 000020          11$:    CMP     L$UNIT,#20                      ;                       1999
000240  101405                                  BLOS    12$
000242  104454                                  TRAP    54                              ;                       2002
000244  000001                                  .WORD   1
000246  000000G                                 .WORD   EGS.01
000250  000000V                                 .WORD   EMS.01
000252  104444                                  TRAP    44
000254  005037  000000G                 12$:    CLR     CLK.TYPE                        ;                       2012
000260  012700  000120                          MOV     #120,R0                         ;                       2014
000264  104462                                  TRAP    62
000266  103012                                  BHIS    13$
000270  010001                                  MOV     R0,R1                           ; R0,CLK.ADR
000272  012737  000001  000000G                 MOV     #1,CLK.TYPE                     ;                       2017
000300  011137  000000G                         MOV     (R1),CLK.CSR                    ; CLK.ADR,*             2018
000304  012777  000105  000000G                 MOV     #105,@CLK.CSR                   ;                       2019
000312  000415                                  BR      14$                             ;                       2014
000314  012700  000114                 13$:    MOV     #114,R0                         ;                       2024
000320  104462                                  TRAP    62
000322  103011                                  BHIS    14$
000324  010001                                  MOV     R0,R1                           ; R0,CLK.ADR
000326  012737  177777  000000G                 MOV     #-1,CLK.TYPE                    ;                       2027
000334  011137  000000G                         MOV     (R1),CLK.CSR                    ; CLK.ADR,*             2028
000340  012777  000100  000000G                 MOV     #100,@CLK.CSR                   ;                       2029
000346  005737  000000G                 14$:    TST     CLK.TYPE                        ;                       2034
000352  001006                                  BNE     15$
000354  104454                                  TRAP    54                              ;                       2037
000356  000002                                  .WORD   2
000360  000000G                                 .WORD   EGS.02
000362  000000                                  .WORD   0
000364  104444                                  TRAP    44
000366  000421                                  BR      16$                             ;                       2034
000370  016137  000004  000000G          15$:    MOV     4(R1),CLK.VECTOR                ; *(CLK.ADR),*          2042
000376  016137  000006  000000G                 MOV     6(R1),CLK.HERTZ                 ; *(CLK.ADR),*          2043
000404  012746  000300                          MOV     #300,-(SP)                      ;                       2044
000410  012746  000000V                         MOV     #CLK.INT.SERV,-(SP)
000414  013746  000000G                         MOV     CLK.VECTOR,-(SP)
000420  012746  000003                          MOV     #3,-(SP)
000424  104437                                  TRAP    37
000426  062706  000010                          ADD     #10,SP                          ;                       2041
000432  013716  000000G                 16$:    MOV     L$UNIT,(SP)                     ;                       2060
000436  005316                                  DEC     (SP)
000440  123727  000000G 000005                  CMPB    ENTRY.REASON,#5                 ;                       2053
000446  001002                                  BNE     17$
000450  000137  003002'                         JMP     37$
000454  005000                          17$:    CLR     R0                              ; COUNT                 2057
000456  005060  000000G                 18$:    CLR     CST(R0)                         ; *(COUNT)              2058
000462  062700  000002                          ADD     #2,R0                           ; *,COUNT               2057
000466  020027  000066                          CMP     R0,#66                          ; COUNT,*
000472  003771                                  BLE     18$
000474  005004                                  CLR     R4                              ; UNIT                  2060
000476  000137  002770'                         JMP     35$
000502  105064  000000G                 19$:    CLRB    CPT(R4)                         ; *(UNIT)               2063
000506  010400                                  MOV     R4,R0                           ; UNIT,*                2064
000510  104442                                  TRAP    42
000512  010066  000004                          MOV     R0,4(SP)                        ; *,HWPT.REF
```

```
000516  001002                          BNE     20$
000520  000137  002766'                 JMP     34$
000524  105005                  20$:    CLRB    R5              ; FLAG1                 2068
000526  005003                          CLR     R3              ; CTLR                  2069
000530  010346                  21$:    MOV     R3,-(SP)        ; CTLR,*                2072
000532  012746  000016                   MOV     #16,-(SP)
000536  004737  000000G                  JSR     PC,BL$MUL
000542  022626                          CMP     (SP)+,(SP)+
000544  026076  000000G 000004           CMP     CST(R0),@4(SP)  ; *,HWPT.REF
000552  001136                          BNE     28$
000554  112705  000001                   MOVB    #1,R5           ; *,FLAG1               2076
000560  105002                          CLRB    R2              ; FLAG2                 2077
000562  010346                          MOV     R3,-(SP)        ; CTLR,*                2081
000564  012746  000007                   MOV     #7,-(SP)
000570  004737  000000G                  JSR     PC,BL$MUL
000574  010066  000006                   MOV     R0,6(SP)
000600  022626                          CMP     (SP)+,(SP)+
000602  012701  000003                   MOV     #3,R1           ; *,OFFSET              2078
000606  010100                  22$:    MOV     R1,R0           ; OFFSET,*              2081
000610  066600  000002                   ADD     2(SP),R0
000614  006300                          ASL     R0
000616  062700  000000G                  ADD     #CST,R0
000622  032710  040000                   BIT     #40000,(R0)
000626  001021                          BNE     23$
000630  112702  000001                   MOVB    #1,R2           ; *,FLAG2               2085
000634  016646  000004                   MOV     4(SP),-(SP)     ; HWPT.REF,*            2086
000640  062716  000006                   ADD     #6,(SP)
000644  013610                          MOV     @(SP)+,(R0)
000646  010446                          MOV     R4,-(SP)        ; UNIT,*                2088
000650  000316                          SWAB    (SP)
000652  042716  160377                   BIC     #160377,(SP)
000656  042710  017400                   BIC     #17400,(R0)
000662  052610                          BIS     (SP)+,(R0)
000664  052710  040000                   BIS     #40000,(R0)     ;                       2089
000670  000443                          BR      26$             ;                       2083
000672  016646  000004          23$:    MOV     4(SP),-(SP)     ; HWPT.REF,*            2096
000676  062716  000006                   ADD     #6,(SP)
000702  121036                          CMPB    (R0),@(SP)+
000704  001031                          BNE     25$
000706  112702  000001                   MOVB    #1,R2           ; *,FLAG2               2100
000712  104450                          TRAP    50              ;                       2101
000714  103017                          BHIS    24$
000716  017646  000004                   MOV     @4(SP),-(SP)    ; HWPT.REF,*            2104
000722  005046                          CLR     -(SP)
000724  016600  000010                   MOV     10(SP),R0       ; HWPT.REF,*
000730  116016  000006                   MOVB    6(R0),(SP)      ; *(HWPT.REF),*
000734  012746  000000G                  MOV     #CER.01,-(SP)
000740  012746  000003                   MOV     #3,-(SP)
000744  010600                          MOV     SP,R0           ; SP,*
000746  104417                          TRAP    17
000750  062706  000010                   ADD     #10,SP
000754  112764  000001  000000G  24$:    MOVB    #1,DUR(R4)      ; *,*(UNIT)             2105
000762  010400                          MOV     R4,R0           ; UNIT,*                2106
000764  104451                          TRAP    51
000766  000404                          BR      26$             ;                       2098
000770  005201                  25$:    INC     R1              ; OFFSET                2078
000772  020127  000006                   CMP     R1,#6           ; OFFSET,*
```

```
000776  003703                          BLE     22$
001000  105702                  26$:    TSTB    R2                      ; FLAG2                 2115
001002  001026                          BNE     29$
001004  104450                          TRAP    50                      ;                       2119
001006  103012                          BHIS    27$
001010  017646  000004                  MOV     @4(SP),-(SP)            ; HWPT.REF,*            2121
001014  012746  000000G                 MOV     #CER.02,-(SP)
001020  012746  000002                  MOV     #2,-(SP)
001024  010600                          MOV     SP,R0                   ; SP,*
001026  104417                          TRAP    17
001030  062706  000006                  ADD     #6,SP
001034  112764  000001  000000G  27$:   MOVB    #1,DUR(R4)              ; *,*(UNIT)             2122
001042  010400                          MOV     R4,R0                   ; UNIT,*                2123
001044  104451                          TRAP    51
001046  000404                          BR      29$                     ;                       2074
001050  005203                  28$:    INC     R3                      ; CTLR                  2069
001052  020327  000003                  CMP     R3,#3                   ; CTLR,*
001056  003624                          BLE     21$
001060  105705                  29$:    TSTB    R5                      ; FLAG1                 2133
001062  001102                          BNE     34$
001064  105002                          CLRB    R2                      ; FLAG2                 2137
001066  005000                          CLR     R0                      ; CTLR                  2138
001070  005760  000000G         30$:    TST     CST(R0)                 ; *(CTLR)               2141
001074  001045                          BNE     31$
001076  112702  000001                  MOVB    #1,R2                   ; *,FLAG2               2145
001102  017660  000004  000000G         MOV     @4(SP),CST(R0)          ; HWPT.REF,*(CTLR)      2146
001110  016603  000004                  MOV     4(SP),R3                ; HWPT.REF,*            2147
001114  016301  000002                  MOV     2(R3),R1                ; *(HWPT.REF),*
001120  042701  177000                  BIC     #177000,R1
001124  042760  000777  000002G         BIC     #777,CST+2(R0)          ; *,*(CTLR)
001132  050160  000002G                 BIS     R1,CST+2(R0)            ; *,*(CTLR)
001136  010301                          MOV     R3,R1                   ; HWPT.REF,*            2148
001140  116160  000004  000004G         MOVB    4(R1),CST+4(R0)         ; *(HWPT.REF),*(CTLR)
001146  012703  000006G                 MOV     #CST+6,R3               ;                       2150
001152  060003                          ADD     R0,R3                   ; CTLR,*
001154  016601  000004                  MOV     4(SP),R1                ; HWPT.REF,*
001160  016113  000006                  MOV     6(R1),(R3)              ; *(HWPT.REF),*
001164  010401                          MOV     R4,R1                   ; UNIT,*                2151
001166  000301                          SWAB    R1
001170  042701  160377                  BIC     #160377,R1
001174  042713  017400                  BIC     #17400,(R3)
001200  050113                          BIS     R1,(R3)
001202  052713  040000                  BIS     #40000,(R3)             ;                       2152
001206  000405                          BR      32$                     ;                       2143
001210  062700  000016          31$:    ADD     #16,R0                  ; *,CTLR                2138
001214  020027  000052                  CMP     R0,#52                  ; CTLR,*
001220  003723                          BLE     30$
001222  105702                  32$:    TSTB    R2                      ; FLAG2                 2159
001224  001021                          BNE     34$
001226  104450                          TRAP    50                      ;                       2163
001230  103012                          BHIS    33$
001232  012746  000004                  MOV     #4,-(SP)                ;                       2165
001236  012746  000000G                 MOV     #CER.03,-(SP)
001242  012746  000002                  MOV     #2,-(SP)
001246  010600                          MOV     SP,R0                   ; SP,*
001250  104417                          TRAP    17
001252  062706  000006                  ADD     #6,SP
```

```
001256  112764  000001  000000G         33$:    MOVB    #1,DUR(R4)              ; *,*(UNIT)                             2166
001264  010400                                  MOV     R4,R0                   ; UNIT,*                                2167
001266  104451                                  TRAP    51
001270  005204                          34$:    INC     R4                      ; UNIT                                  2060
001272  020416                          35$:    CMP     R4,(SP)                 ; UNIT,*
001274  003002                                  BGT     36$
001276  000137  002200'                         JMP     19$
001302  000432                          36$:    BR      43$                     ;                                       2053
001304  005002                          37$:    CLR     R2                      ; UNIT                                  2181
001306  000426                                  BR      42$
001310  010200                          38$:    MOV     R2,R0                   ; UNIT,*                                2184
001312  104442                                  TRAP    42
001314  010066  000004                         MOV     R0,4(SP)                ; *,HWPT.REF
001320  001420                                  BEQ     41$
001322  112762  000001  000000G               MOVB    #1,CPT(R2)              ; *,*(UNIT)                             2188
001330  005000                                  CLR     R0                      ; CTLR                                  2189
001332  026076  000000G 000004         39$:    CMP     CST(R0),@4(SP)          ; *(CTLR),HWPT.REF                      2192
001340  001003                                  BNE     40$
001342  105260  000005G                         INCB    CST+5(R0)               ; *(CTLR)                               2196
001346  000405                                  BR      41$                     ;                                       2194
001350  062700  000016                  40$:    ADD     #16,R0                  ; *,CTLR                                2189
001354  020027  000052                         CMP     R0,#52                  ; CTLR,*
001360  003764                                  BLE     39$
001362  005202                          41$:    INC     R2                      ; UNIT                                  2181
001364  020216                          42$:    CMP     R2,(SP)                 ; UNIT,*
001366  003750                                  BLE     38$
001370  123727  000000G 000002         43$:    CMPB    ENTRY.REASON,#2         ;                                       2209
001376  101034                                  BHI     47$
001400  005037  000000G                         CLR     CRN                     ;                                       2213
001404  005037  000000G                         CLR     CTLR.CNT                ;                                       2214
001410  005000                                  CLR     R0                      ; CTLR                                  2215
001412  005760  000000G                 44$:    TST     CST(R0)                 ; *(CTLR)                               2218
001416  001402                                  BEQ     45$
001420  005237  000000G                         INC     CTLR.CNT                ;                                       2220
001424  062700  000016                  45$:    ADD     #16,R0                  ; *,CTLR                                2215
001430  020027  000052                         CMP     R0,#52                  ; CTLR,*
001434  003766                                  BLE     44$
001436  104431                                  TRAP    31                      ;                                       2224
001440  010037  000000G                         MOV     R0,FREE.MEM.ADDR
001444  011037  000000G                         MOV     (R0),MEM.SIZE           ; FREE.MEM.ADDR,*                       2225
001450  005000                                  CLR     R0                      ; COUNT                                 2227
001452  005060  000000G                 46$:    CLR     TALLY(R0)               ; *(COUNT)                              2228
001456  062700  000002                         ADD     #2,R0                   ; *,COUNT                               2227
001462  020027  001376                         CMP     R0,#1376                ; COUNT,*
001466  003771                                  BLE     46$
001470  005000                          47$:    CLR     R0                      ; CTLR                                  2236
001472  105060  000000G                 48$:    CLRB    QIO(R0)                 ; *(CTLR)                               2237
001476  005200                                  INC     R0                      ; CTLR                                  2236
001500  020027  000003                         CMP     R0,#3                   ; CTLR,*
001504  003772                                  BLE     48$
001506  005000                                  CLR     R0                      ; COUNT                                 2238
001510  112760  000377  000000G         49$:    MOVB    #377,RP.USE(R0)         ; *,*(COUNT)                            2239
001516  005200                                  INC     R0                      ; COUNT                                 2238
001520  020027  000037                         CMP     R0,#37                  ; COUNT,*
001524  003771                                  BLE     49$
001526  005037  000000G                         CLR     IODQ.OUT                ;                                       2240
001532  005037  000000G                         CLR     IODQ.IN
```

```
001536  132737  000100  000000G                 BITB    #100,SWP.FLAGS                  ;                                       2241
001544  001407                                  BEQ     50$
001546  132737  000040  000000G                 BITB    #40,SWP.FLAGS                   ;                                       2242
001554  001403                                  BEQ     50$
001556  142737  000100  000000G                 BICB    #100,SWP.FLAGS                  ;                                       2244
001564  023737  000000G 000000G         50$:    CMP     SWP.STRACK,SWP.ETRACK           ;                                       2256
001572  103407                                  BLO     51$
001574  013700  000000G                         MOV     SWP.STRACK,R0                   ; *,TEMP                                2263
001600  013737  000000G 000000G                 MOV     SWP.ETRACK,SWP.STRACK           ;                                       2264
001606  010037  000000G                         MOV     R0,SWP.ETRACK                   ; TEMP,*                                2265
001612  005000                          51$:    CLR     R0                              ;                                       2269
001614  104441                                  TRAP    41
001616  062706  000010                          ADD     #10,SP                          ;                                       1925
001622  000207                                  RTS     PC

; Routine Size:  458 words,     Routine Base:  $CODE$ + 1476
; Maximum stack depth per invocation:  16 words



                                        .SBTTL  L$INIT INITIALIZE SECTION
000000  004737  001476'         L$INIT::JSR     PC,LINIT                        ;                                       2269
000004  104411                          TRAP    11
000006  000207                          RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 3322
; Maximum stack depth per invocation:  2 words
```

```
;          2272  %SBTTL 'AUTODROP SECTION'
;          2273
;          2274  !+
;          2275  !       THIS CODE IS EXECUTED IMMEDIATELY AFTER THE INITIALIZE CODE IF THE
;          2276  !       "ADR" FLAG WAS SET. THE UNIT(S) UNDER TEST ARE CHECKED TO SEE IF THEY
;          2277  !       WILL RESPOND.  THOSE THAT DON'T ARE IMMEDIATELY DROPPED FROM TESTING.
;          2278  !-
;          2279
;          2280  BGNAUTO;
;          2281
;          2282  RETURN;
;          2283
;          2284  ENDAUTO;


                                        .SBTTL  LAUTO AUTODROP SECTION
000000  000207                  LAUTO:  RTS     PC                              ;                               2271

; Routine Size:  1 word,        Routine Base:  $CODE$ + 3332
; Maximum stack depth per invocation:  0 words



                                        .SBTTL  L$AUTO AUTODROP SECTION
000000  004737  003332'         L$AUTO::JSR     PC,LAUTO                        ;                               2282
000004  104461                          TRAP    61
000006  000207                          RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 3334
; Maximum stack depth per invocation:  2 words
```

```
;       2285  %SBTTL 'CLEANUP CODING SECTION'
;       2286
;       2287  !+
;       2288  !       THE CLEANUP CODING SECTION CONTAINS THE CODING THAT IS PERFORMED UNDER
;       2289  !       ONE OF THREE CONDITIONS: (A) AFTER EACH COMPLETE PASS, (B) WHENEVER
;       2290  !       THE "DOCLN" MACRO IS ENCOUNTERED, AND (C) WHEN THE OPERATOR TYPES
;       2291  !       "<CTRL-C>". SOME OF THE CODE IN THIS SECTION IS NOT PERFORMED IF THE
;       2292  !       REASON FOR ENTRY IS END-OF-PASS ((A) ABOVE).
;       2293  !-
;       2294
;       2295  BGNCLN;
;       2296
;       2297  IF .CLK_TYPE EQLU NO_CLOCK                    ! IF THERE IS NO CLOCK
;       2298  THEN                                          ! THEN
;       2299      RETURN                                    ! RETURN
;       2300  ELSE                                          ! OTHERWISE
;       2301      BEGIN
;       2302
;       2303      .CLK_CSR = 0;                             ! TURN OFF CLOCK
;       2304      CLRVEC (.CLK_VECTOR);                     ! CLEAR CLOCK VECTOR
;       2305
;       2306      END;
;       2307
;       2308  IF NOT .EOP_FLAG                             ! IF INVOKED BY ^C OR DOCLN
;       2309  THEN                                          ! THEN
;       2310      BEGIN
;       2311
;       2312      SETVEC (4, NEX_TRAP, PRI07);              ! IN CASE USER GAVE BAD IP ADDRESS
;       2313      INCR CTLR FROM 0 TO (MAX_CTLR - 1) DO     ! FOR EACH CONTROLLER
;       2314          BEGIN
;       2315
;       2316          IF (RC25_ADDR = .CST [.CTLR, IP_ADDR]) NEQA 0   ! IF CONTROLLER EXISTS
;       2317          THEN                                            ! THEN
;       2318              WRT_RC25 (RCIP, RC_ALL, ALL_ONES);          ! WRITE IP TO STOP DEVICE
;       2319
;       2320          END;
;       2321
;       2322      CLRVEC (4);                               ! RETURN TRAP 4 TO DIAGNOSTIC SUPERVISOR
;       2323
;       2324      END;                                      ! IF NOT END OF PASS
;       2325
;       2326  INCR UNIT FROM 0 TO (MAX_UNITS - 1) DO        ! INITIALIZE DROP UNIT REASON VECTOR
;       2327      DUR [.UNIT] = 0;
;       2328
;       2329  IF .MEM_MGMT                                  ! IF SYSTEM HAS MEMORY MANAGEMENT
;       2330  THEN                                          ! THEN
;       2331      BEGIN
;       2332
;       2333      KTPAR4 = %O'1000';                        ! RESTORE PAR'S
;       2334      KTPAR5 = %O'1200';
;       2335      MMR0 = 0;                                 ! TURN OFF MEMORY MANAGEMENT
;       2336
;       2337      END;
;       2338
;       2339  EOP_FLAG = FALSE;                            ! CLEAR END-OF-PASS FLAG
;       2340
;       2341  ENDCLN;
```

```
                                        .SBTTL  LCLEAN CLEANUP CODING SECTION
000000  010146                  LCLEAN: MOV     R1,-(SP)                ;                               2284
000002  005737  000000G                 TST     CLK.TYPE                ;                               2297
000006  001472                          BEQ     6$                      ;                               2299
000010  005077  000000G                 CLR     @CLK.CSR                ;                               2303
000014  013700  000000G                 MOV     CLK.VECTOR,R0           ;                               2304
000020  104436                          TRAP    36
000022  132737  000001  000000G         BITB    #1,EOP.FLAG             ;                               2308
000030  001034                          BNE     3$
000032  012746  000340                  MOV     #340,-(SP)              ;                               2312
000036  012746  000000V                 MOV     #NEX.TRAP,-(SP)
000042  012746  000004                  MOV     #4,-(SP)
000046  012746  000003                  MOV     #3,-(SP)
000052  104437                          TRAP    37
000054  005001                          CLR     R1                      ; CTLR                          2313
000056  016137  000000G 000000G  1$:     MOV     CST(R1),RC25.ADDR       ; *(CTLR),*                     2316
000064  001404                          BEQ     2$
000066  012700  177777                  MOV     #-1,R0                  ; *,RC.REG                      2318
000072  010071  000000G                 MOV     R0,@CST(R1)             ; RC.REG,*
000076  062701  000016          2$:     ADD     #16,R1                  ; *,CTLR                        2313
000102  020127  000052                  CMP     R1,#52                  ; CTLR,*
000106  003763                          BLE     1$
000110  012700  000004                  MOV     #4,R0                   ;                               2322
000114  104436                          TRAP    36
000116  062706  000010                  ADD     #10,SP                  ;                               2310
000122  005000                  3$:     CLR     R0                      ; UNIT                          2326
000124  105060  000000G         4$:     CLRB    DUR(R0)                 ; *(UNIT)                       2327
000130  005200                          INC     R0                      ; UNIT                          2326
000132  020027  000017                  CMP     R0,#17                  ; UNIT,*
000136  003772                          BLE     4$
000140  132737  000001  000000G         BITB    #1,MEM.MGMT             ;                               2329
000146  001410                          BEQ     5$
000150  012737  001000  172350          MOV     #1000,@#172350          ;                               2333
000156  012737  001200  172352          MOV     #1200,@#172352          ;                               2334
000164  005037  177572                  CLR     @#177572                ;                               2335
000170  105037  000000G         5$:     CLRB    EOP.FLAG                ;                               2339
000174  012601                  6$:     MOV     (SP)+,R1                ;                               2284
000176  000207                          RTS     PC

; Routine Size:  64 words,      Routine Base:  $CODE$ + 3344
; Maximum stack depth per invocation:  7 words



                                        .SBTTL  L$CLEAN CLEANUP CODING SECTION
000000  004737  003344'         L$CLEAN::
                                        JSR     PC,LCLEAN               ;                               2339
000004  104412                          TRAP    12
000006  000207                          RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 3544
; Maximum stack depth per invocation:  2 words
```

```
;       2342  %SBTTL 'DROP UNIT SECTION'
;       2343
;       2344  !+
;       2345  !       THE DROP-UNIT SECTION CONTAINS THE CODING THAT CAUSES A PLATTER TO NO
;       2346  !       LONGER BE TESTED. IT IS EXECUTED WHENEVER THE MACRO "DODU" IS
;       2347  !       ENCOUNTERED IN THE PROGRAM, OR WHEN THE OPERATOR TYPES THE COMMAND
;       2348  !       "DROP/UNIT:X"
;       2349  !
;       2350  !       THE PROGRAM WILL DROP A UNIT FOR ONE OF SEVERAL REASONS:
;       2351  !               1. A CONTROLLER CANNOT BE BROUGHT ONLINE,
;       2352  !               2. A PLATTER CANNOT BE BROUGHT ONLINE TO ITS CONTROLLER,
;       2353  !               3. AN UNRECOVERABLE PORT OR CONTROLLER ERROR HAS BEEN DETECTED,
;       2354  !               4. A PLATTER HAS REACHED THE HARD ERROR LIMIT, OR
;       2355  !               5. AN MSCP END MESSAGE CONTAINS A "UNIT-OFFLINE" STATUS CODE.
;       2356  !-
;       2357
;       2358  BGNDU;
;       2359
;       2360  OWN
;       2361      DUM_TBL : VECTOR [5] INITIAL (DUM_UC, DUM_CE,       ! ADDRESS TABLE FOR DROP UNIT MESSAGES
;       2362                DUM_IE, DUM_HE, DUM_UE);
;       2363
;       2364  LOCAL
;       2365      UNIT : WORD,                                        ! NUMBER OF UNIT BEING DROPPED
;       2366      PRINT : WORD;                                       ! O.K TO PRINT DROP UNIT MESSAGE
;       2367
;       2368  LABEL
;       2369      SEARCH;
;       2370
;       2371  BEGIN
;       2372
;       2373  REGISTER
;       2374      INPUT = 0;                                          ! UNIT NUMBER APPEARS IN R0 UPON ENTRY
;       2375  UNIT = .INPUT;                                         ! GET UNIT NUMBER
;       2376
;       2377  END;                                                    ! UNDECLARE REGISTER
;       2378
;       2379  PRINT = FALSE;                                         ! ASSUME NO PRINTING OF MESSAGE
;       2380
;       2381  SEARCH:                                                ! BEGIN SEARCH BLOCK
;       2382      BEGIN
;       2383
;       2384      INCR CTLR FROM 0 TO (MAX_CTLR - 1) DO               ! FOR EACH CST
;       2385          BEGIN
;       2386
;       2387          INCR OFFSET FROM (0 + OF_UN) TO (3 + OF_UN) DO  ! FOR EACH UNIT ENTRY IN CST
;       2388              BEGIN
;       2389
;       2390              IF ((.CST [.CTLR, .OFFSET, P_PRES] EQLU PRESENT) AND ! IF UNIT EXISTS AND
;       2391                  (.CST [.CTLR, .OFFSET, P_UNIT] EQLU .UNIT))      !    UNIT MATCHES CST ENTRY
;       2392              THEN                                                 ! THEN
;       2393                  BEGIN
;       2394
;       2395                  IF .CST [.CTLR, .OFFSET, P_STAT] EQLU ONLINE     ! IF UNIT IS ONLINE
;       2396                  THEN                                             ! THEN
;       2397                      BEGIN
;       2398
```

```
:          2399                    PRINT = TRUE;                                   ! O.K. TO PRINT MESSAGE
:          2400                    CST [.CTLR, .OFFSET, P_STAT] = OFFLINE;         ! MARK UNIT OFFLINE
:          2401                    IF .CPT [.UNIT] EQLU YES                        ! IF UNIT IS UNDER TEST
:          2402                    THEN                                            ! THEN
:          2403                        BEGIN
:          2404
:          2405                        CPT [.UNIT] = NO;                                   ! NO FURTHER TESTING
:          2406                        CST [.CTLR, U_CNT] = .CST [.CTLR, U_CNT] - 1;       ! DECREMENT UNIT COUNT
:          2407                        DM_COMM [.CTLR, .OFFSET - OF_UN, AL[BIT] = -1;      ! IF DM EXER, TELL FPT
:          2408
:          2409                        END;
:          2410
:          2411                    END;                                            ! IF UNIT WAS ONLINE
:          2412
:          2413                LEAVE SEARCH;                               ! EXIT SEARCH BLOCK
:          2414
:          2415                END;                                        ! IF UNIT EXISTS AND MATCHES
:          2416
:          2417            END;                                            ! CST UNIT LOOP
:          2418
:          2419        END;                                                ! CONTROLLER LOOP
:          2420
:          2421    END;                                                    ! SEARCH BLOCK
:          2422
:          2423    IF MANUAL AND (.PRINT OR (.DUR [.UNIT] LEQU DU_INIT))   ! IF O.K. TO PRINT MESSAGE
:          2424    THEN                                                    ! THEN
:          2425        BEGIN
:          2426
:          2427        PRINTF (CRLF);                                      ! <CR><LF>
:          2428        PRINTF (ETIME, .HOURS, .MINUTES, .SECONDS);         ! ELAPSED TIME
:          2429        PRINTF (DUM_00, .UNIT);                             ! 'UNIT XX. DROPPED - "
:          2430        PRINTF (.DUM_TBL [.DUR [.UNIT]]);                   ! REASON
:          2431
:          2432        END;
:          2433
:          2434    ENDDU;


003554  000000G                 DUM.TBL:.WORD   DUM.UC
003556  000000G                         .WORD   DUM.CE
003560  000000G                         .WORD   DUM.IE
003562  000000G                         .WORD   DUM.HE
003564  000000G                         .WORD   DUM.UE


                                        .SBTTL  LDU DROP UNIT SECTION
000000  004137  000000G         LDU:    JSR     R1,$SAVE5                   ;                               2341
000004  010002                        MOV     R0,R2                       ; INPUT,UNIT                    2375
000006  005003                        CLR     R3                          ; PRINT                         2379
000010  005004                        CLR     R4                          ; CTLR                          2384
000012  010446                1$:     MOV     R4,-(SP)                    ; CTLR,*                        2390
000014  012746  000007                MOV     #7,-(SP)
000020  004737  000000G               JSR     PC,BL$MUL
000024  010005                        MOV     R0,R5
000026  012701  000003                MOV     #3,R1                       ; *,OFFSET                      2387
000032  010500                2$:     MOV     R5,R0                       ;                               2390
000034  060100                        ADD     R1,R0                       ; OFFSET,*
```

```
000036  006300                          ASL     R0
000040  062700  000000G                 ADD     #CST,R0
000044  032710  040000                  BIT     #40000,(R0)
000050  001450                          BEQ     4$
000052  010246                          MOV     R2,-(SP)                ; UNIT,*                2391
000054  011046                          MOV     (R0),-(SP)
000056  000316                          SWAB    (SP)
000060  042716  177740                  BIC     #177740,(SP)
000064  022626                          CMP     (SP)+,(SP)+
000066  001041                          BNE     4$
000070  032710  020000                  BIT     #20000,(R0)             ;                       2395
000074  001434                          BEQ     3$
000076  012703  000001                  MOV     #1,R3                   ; *,PRINT               2399
000102  042710  020000                  BIC     #20000,(R0)             ;                       2400
000106  126203  000000G                 CMPB    CPT(R2),R3              ; *(UNIT),*             2401
000112  001025                          BNE     3$
000114  105062  000000G                 CLRB    CPT(R2)                 ; *(UNIT)               2405
000120  010416                          MOV     R4,(SP)                 ; CTLR,*                2406
000122  012746  000016                  MOV     #16,-(SP)
000126  004737  000000G                 JSR     PC,BL$MUL
000132  005726                          TST     (SP)+
000134  105360  000005G                 DECB    CST+5(R0)
000140  010416                          MOV     R4,(SP)                 ; CTLR,*                2407
000142  012746  000032                  MOV     #32,-(SP)
000146  004737  000000G                 JSR     PC,BL$MUL
000152  005726                          TST     (SP)+
000154  060100                          ADD     R1,R0                   ; OFFSET,*
000156  006300                          ASL     R0
000160  012760  177777  177772G         MOV     #-1,DM.COMM-6(R0)
000166  022626                  3$:     CMP     (SP)+,(SP)+             ;                       2393
000170  000411                          BR      5$
000172  005201                  4$:     INC     R1                      ; OFFSET                2387
000174  020127  000006                  CMP     R1,#6                   ; OFFSET,*
000200  003714                          BLE     2$
000202  022626                          CMP     (SP)+,(SP)+             ;                       2385
000204  005204                          INC     R4                      ; CTLR                  2384
000206  020427  000003                  CMP     R4,#3                   ; CTLR,*
000212  003677                          BLE     1$
000214  104450                  5$:     TRAP    50                      ;                       2423
000216  103054                          BHIS    7$
000220  006003                          ROR     R3                      ; PRINT
000222  103404                          BLO     6$
000224  126227  000000G 000002          CMPB    DUR(R2),#2              ; *(UNIT),*
000232  101046                          BHI     7$
000234  012746  000000G           6$:   MOV     #CRLF,-(SP)             ;                       2427
000240  012746  000001                  MOV     #1,-(SP)
000244  010600                          MOV     SP,R0                   ; SP,*
000246  104417                          TRAP    17
000250  013716  000000G                 MOV     SECONDS,(SP)            ;                       2428
000254  013746  000000G                 MOV     MINUTES,-(SP)
000260  013746  000000G                 MOV     HOURS,-(SP)
000264  012746  000000G                 MOV     #ETIME,-(SP)
000270  012746  000004                  MOV     #4,-(SP)
000274  010600                          MOV     SP,R0                   ; SP,*
000276  104417                          TRAP    17
000300  010216                          MOV     R2,(SP)                 ; UNIT,*                2429
000302  012746  000000G                 MOV     #DUM.00,-(SP)
```

```
000306  012746  000002                          MOV     #2,-(SP)
000312  010600                                  MOV     SP,R0                   ; SP,*
000314  104417                                  TRAP    17
000316  116202  000000G                         MOVB    DUR(R2),R2              ; *(UNIT),*                     2430
000322  042702  177400                          BIC     #177400,R2
000326  006302                                  ASL     R2
000330  016216  003554'                         MOV     DUM.TBL(R2),(SP)
000334  012746  000001                          MOV     #1,-(SP)
000340  010600                                  MOV     SP,R0                   ; SP,*
000342  104417                                  TRAP    17
000344  062706  000022                          ADD     #22,SP                  ;                               2425
000350  000207                          7$:     RTS     PC                      ;                               2341

; Routine Size:  117 words,     Routine Base:  $CODE$ + 3566
; Maximum stack depth per invocation:  17 words



                                                .SBTTL  L$DU DROP UNIT SECTION
000000  004737  003566'                 L$DU::  JSR     PC,LDU                  ;                               2432
000004  104453                                  TRAP    53
000006  000207                                  RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 4140
; Maximum stack depth per invocation:  2 words
```

```
;          2435  %SBTTL 'ADD UNIT SECTION'
;          2436
;          2437  !+
;          2438  !       THE ADD-UNIT SECTION CONTAINS ANY CODE THE PROGRAMMER WISHES TO BE
;          2439  !       EXECUTED IN CONJUNCTION WITH THE ADDING OF A UNIT BACK TO THE TEST
;          2440  !       CYCLE. THIS CODE WILL BE EXECUTED AFTER THE OPERATOR COMMAND
;          2441  !       "ADD/UNIT:X".
;          2442  !-
;          2443
;          2444  BGNAU;
;          2445
;          2446  LOCAL
;          2447      UNIT : WORD,                          ! NUMBER OF UNIT BEING ADDED
;          2448      STINDX : WORD,
;          2449      ENDIDX : WORD;
;          2450
;          2451  BEGIN
;          2452
;          2453  REGISTER
;          2454      INPUT = 0;                            ! UNIT NUMBER APPEARS IN R0 UPON ENTRY
;          2455  UNIT = .INPUT;                            ! GET UNIT NUMBER
;          2456
;          2457  END;                                      ! UNDECLARE REGISTER
;          2458
;          2459  DUR [.UNIT] = 0;                          ! ZERO OUT DROP UNIT REASON
;          2460  STINDX = .UNIT * TALLY_LEN;               ! ZERO OUT
;          2461  ENDIDX = .STINDX + TALLY_LEN - 1;         !    ADDED
;          2462  INCR COUNT FROM .STINDX TO .ENDIDX DO     !    UNIT'S
;          2463      TALLY [.COUNT] = 0;                   !    STATISTICS
;          2464
;          2465  ENDAU;

                                        .SBTTL  LAU ADD UNIT SECTION
000000  004137  000000G          LAU:   JSR     R1,$SAVE2                ;                          2434
000004  105060  000000G                 CLRB    DUR(R0)                  ; *(UNIT)                  2459
000010  010046                         MOV     R0,-(SP)                 ; UNIT,*                   2460
000012  012746  000030                 MOV     #30,-(SP)
000016  004737  000000G                JSR     PC,BL$MUL
000022  010002                         MOV     R0,R2                    ; STINDX,ENDIDX            2461
000024  062702  000027                 ADD     #27,R2                   ; *,ENDIDX
000030  010001                         MOV     R0,R1                    ; STINDX,COUNT             2462
000032  005301                         DEC     R1                       ; COUNT
000034  000404                         BR      2$
000036  010100                  1$:    MOV     R1,R0                    ; COUNT,*                  2463
000040  006300                         ASL     R0
000042  005060  000000G                CLR     TALLY(R0)
000046  005201                  2$:    INC     R1                       ; COUNT                    2462
000050  020102                         CMP     R1,R2                    ; COUNT,ENDIDX
000052  003771                         BLE     1$
000054  022626                         CMP     (SP)+,(SP)+              ;                          2434
000056  000207                         RTS     PC
; Routine Size:  24 words,      Routine Base:  $CODE$ + 4150
; Maximum stack depth per invocation:  6 words
                                        .SBTTL  L$AU ADD UNIT SECTION
000000  004737  004150'          L$AU:: JSR     PC,LAU                   ;                          2463
000004  104452                         TRAP    52
000006  000207                         RTS     PC
; Routine Size:  4 words,       Routine Base:  $CODE$ + 4230
; Maximum stack depth per invocation:  2 words
```

```
;          2466  %SBTTL 'NON-EXISTENT MEMORY TRAP HANDLER'
;          2467
;          2468  !+
;          2469  !       THIS TRAP HANDLER IS VECTORED FROM LOCATION 4 FOR ALL UNIBUS TIMEOUT
;          2470  !       ERRORS, INDICATING THAT AN ATTEMPT WAS MADE TO REFERENCE A NON-EXISTENT
;          2471  !       MEMORY LOCATION. ITS MAIN PURPOSE IS TO SET A FLAG FOR THE RC11
;          2472  !       REGISTER EXISTENCE TEST, INDICATING THE ABSENCE OF A DEVICE REGISTER.
;          2473  !-
;          2474
;          2475  BGNSRV (NEX_TRAP);
;          2476
;          2477  NEX = TRUE;                           ! NEX TRAP OCCURRED
;          2478
;          2479  ENDSRV;

                                        .SBTTL  NEX.TRAP NON-EXISTENT MEMORY TRAP HANDLER
000000  012737  000001  000000G  NEX.TRAP::
                                        MOV     #1,NEX                          ;2477
000006  000002                          RTI                                     ;2475

; Routine Size:  4 words,       Routine Base:  $CODE$ + 4240
; Maximum stack depth per invocation:  0 words
```

```
;          2480  %SBTTL 'CLOCK INTERRUPT SERVICE ROUTINE'
;          2481
;          2482  !+
;          2483  !       THE CLOCK INTERRUPT SERVICE ROUTINE IS ENTERED AT THE CLOCK HERTZ RATE
;          2484  !       (50 OR 60 TIMES PER SECOND). ITS PURPOSE IS TO MAINTAIN THE ELAPSED
;          2485  !       TIME OF THE EXERCISER, AND, IF ONE SECOND HAS ELAPSED, TO FLAG THE
;          2486  !       EVENT BY SETTING T_FLAG TO "TRUE".
;          2487  !-
;          2488
;          2489  BGNSRV (CLK_INT_SERV);
;          2490
;          2491  TICKS = .TICKS + 1;
;          2492  IF .TICKS EQLU .CLK_HERTZ
;          2493  THEN
;          2494      BEGIN
;          2495
;          2496      TICKS = 0;
;          2497      SECONDS = .SECONDS + 1;
;          2498      IF .SECONDS EQLU 60
;          2499      THEN
;          2500          BEGIN
;          2501
;          2502          SECONDS = 0;
;          2503          MINUTES = .MINUTES + 1;
;          2504          IF .MINUTES EQLU 60
;          2505          THEN
;          2506              BEGIN
;          2507
;          2508              MINUTES = 0;
;          2509              HOURS = .HOURS + 1;
;          2510
;          2511              END;
;          2512
;          2513          END;
;          2514
;          2515      END;
;          2516
;          2517  IF .TICKS EQLU 0                    ! IF ONE SECOND HAS ELAPSED
;          2518  THEN                                ! THEN
;          2519      T_FLAG = TRUE;                  ! FLAG THE EVENT
;          2520
;          2521  ENDSRV;
```

```
                                        .SBTTL  CLK.INT.SERV CLOCK INTERRUPT SERVICE ROUTINE
000000  005237  000000G               CLK.INT.SERV::
                                        INC     TICKS                           ;                       2491
000004  023737  000000G 000000G               CMP     TICKS,CLK.HERTZ         ;                       2492
000012  001024                              BNE     1$
000014  005037  000000G                     CLR     TICKS                       ;                       2496
000020  005237  000000G                     INC     SECONDS                     ;                       2497
000024  023727  000000G 000074              CMP     SECONDS,#74                 ;                       2498
000032  001014                              BNE     1$
000034  005037  000000G                     CLR     SECONDS                     ;                       2502
000040  005237  000000G                     INC     MINUTES                     ;                       2503
000044  023727  000000G 000074              CMP     MINUTES,#74                 ;                       2504
000052  001004                              BNE     1$
000054  005037  000000G                     CLR     MINUTES                     ;                       2508
000060  005237  000000G                     INC     HOURS                       ;                       2509
000064  005737  000000G               1$:   TST     TICKS                       ;                       2517
000070  001003                              BNE     2$
000072  112737  000001  000000G             MOVB    #1,T.FLAG                   ;                       2519
000100  000002                        2$:   RTI                                 ;                       2489

; Routine Size:  33 words,      Routine Base:  $CODE$ + 4250
; Maximum stack depth per invocation:  0 words
```

```
;          2522  %SBTTL 'GLOBAL ROUTINES'
;          2523
;          2524  GLOBAL ROUTINE COPY_BLK (SRC, DST, NWRDS) : NOVALUE =
;          2525
;          2526  !+
;          2527  !       THE PURPOSE OF THIS ROUTINE IS TO TRANSFER A BLOCK OF MEMORY OF LENGTH
;          2528  !       "NWRDS" FROM "SRC" TO "DST".
;          2529  !-
;          2530
;          2531  INCR I FROM 1 TO .NWRDS DO                    ! FOR EACH WORD IN BLOCK
;          2532      BEGIN
;          2533
;          2534      .DST = ..SRC;                             ! TRANSFER WORD FROM SOURCE TO DESTINATION
;          2535      DST = .DST + 2;                           ! ADVANCE DESTINATION ADDRESS
;          2536      SRC = .SRC + 2;                           ! ADVANCE SOURCE ADDRESS
;          2537
;          2538      END;


                                        .SBTTL  COPY.BLK GLOBAL ROUTINES
000000  005000                  COPY.BLK::
                                        CLR     R0                      ; I                             2531
000002  000411                         BR      2$
000004  017676  000006  000004  1$:     MOV     @6(SP),@4(SP)           ; SRC,DST                       2534
000012  062766  000002  000004          ADD     #2,4(SP)                ; *,DST                         2535
000020  062766  000002  000006          ADD     #2,6(SP)                ; *,SRC                         2536
000026  005200                  2$:     INC     R0                      ; I                             2531
000030  020066  000002                  CMP     R0,2(SP)                ; I,NWRDS
000034  003763                         BLE     1$
000036  000207                         RTS     PC                      ;                               2524

; Routine Size:  16 words,      Routine Base:  $CODE$ + 4352
; Maximum stack depth per invocation:  0 words
```

```
;         2539  GLOBAL ROUTINE SET_CPAR (CTLR) : NOVALUE =
;         2540
;         2541  !+
;         2542  !       THIS ROUTINE SETS UP THE COMMONLY-USED CONTROLLER-RELATED DATA ITEMS
;         2543  !       FOR THE GIVEN CONTROLLER NUMBER.
;         2544  !
;         2545  !       INPUTS:
;         2546  !               CTLR - CONTROLLER NUMBER
;         2547  !
;         2548  !       IMPLICIT OUTPUTS:
;         2549  !               CCTLR - CURRENT CONTROLLER NUMBER
;         2550  !               CST_ADDR - ADDRESS OF CONTROLLER'S STATUS TABLE
;         2551  !               DCT_ADDR - ADDRESS OF CONTROLLER'S DRIVER TABLE
;         2552  !               RC25_ADDR - ADDRESS OF CONTROLLER'S IP REGISTER
;         2553  !               DMC_ADDR - ADDRESS OF CONTROLLER'S DM_COMM AREA
;         2554  !               OCL_X1,2 - STARTING/ENDING INDECES OF CONTROLLER'S OUTSTANDING
;         2555  !                       COMMAND AREA (OUTC_LIST, OUTC_TIMR)
;         2556  !               RPS_X1,2 - STARTING/ENDING INDECES OF CONTROLLER'S RETURN
;         2557  !                       PACKET SAVE AREA (RP_SAVE)
;         2558  !-
;         2559
;         2560  BEGIN
;         2561
;         2562  CCTLR = .CTLR;                                  ! SET CURRENT CONTROLLER NUMBER
;         2563  CST_ADDR = CST + (.CTLR * CST_LEN * 2);         ! CALCULATE ADDRESS OF CONTROLLER'S CST
;         2564  DCT_ADDR = DCT + (.CTLR * DCT_LEN * 2);         ! CALCULATE ADDRESS OF CONTROLLER'S DCT
;         2565  RC25_ADDR = .CST_ADDR [IP_ADDR];               ! GET CONTROLLER'S DEVICE ADDRESS
;         2566  DMC_ADDR = DM_COMM + (.CTLR * DMC_LEN * 2); ! CALCULATE ADDRESS OF CONTROLLER'S DM_COMM AREA
;         2567  OCL_X1 = .CTLR * OUTC_CNT;                    ! STARTING INDEX OF CTLR'S OUTC AREA
;         2568  OCL_X2 = .OCL_X1 + OUTC_CNT - 1;              ! ENDING INDEX OF CTLR'S OUTC AREA
;         2569  RPS_X1 = .CTLR * RPS_LEN;                     ! STARTING INDEX OF CTLR'S RETPKT SAVE AREA
;         2570  RPS_X2 = .RPS_X1 + RPS_LEN - 1;               ! ENDING INDEX OF CTLR'S RETPKT SAVE AREA
;         2571
;         2572  END;


                                        .SBTTL  SET.CPAR GLOBAL ROUTINES
000000  010146                  SET.CPAR::
                                        MOV     R1,-(SP)                        ;                       2539
000002  016601  000004                  MOV     4(SP),R1                        ; CTLR,*                2562
000006  010137  000000G                 MOV     R1,CCTLR
000012  010146                          MOV     R1,-(SP)                        ;                       2563
000014  012746  000016                  MOV     #16,-(SP)
000020  004737  000000G                 JSR     PC,BL$MUL
000024  062700  000000G                 ADD     #CST,R0
000030  010037  000000G                 MOV     R0,CST.ADDR
000034  010116                          MOV     R1,(SP)                         ;                       2564
000036  012746  000022                  MOV     #22,-(SP)
000042  004737  000000G                 JSR     PC,BL$MUL
000046  062700  000000G                 ADD     #DCT,R0
000052  010037  000000G                 MOV     R0,DCT.ADDR
000056  017737  000000G 000000G         MOV     @CST.ADDR,RC25.ADDR             ;                       2565
000064  010116                          MOV     R1,(SP)                         ;                       2566
000066  012746  000064                  MOV     #64,-(SP)
000072  004737  000000G                 JSR     PC,BL$MUL
000076  062700  000000G                 ADD     #DM.COMM,R0
000102  010037  000000G                 MOV     R0,DMC.ADDR
```

```
000106  010100                          MOV     R1,R0                           ;                                       2567
000110  006300                          ASL     R0
000112  006300                          ASL     R0
000114  006300                          ASL     R0
000116  006300                          ASL     R0
000120  010037  000000G                 MOV     R0,OCL.X1
000124  010037  000000G                 MOV     R0,OCL.X2                       ; OCL.X1,*                              2568
000130  062737  000017  000000G         ADD     #17,OCL.X2
000136  010100                          MOV     R1,R0                           ;                                       2569
000140  006300                          ASL     R0
000142  006300                          ASL     R0
000144  006300                          ASL     R0
000146  010037  000000G                 MOV     R0,RPS.X1
000152  010037  000000G                 MOV     R0,RPS.X2                       ; RPS.X1,*                              2570
000156  062737  000007  000000G         ADD     #7,RPS.X2
000164  062706  000010                  ADD     #10,SP                          ;                                       2560
000170  012601                          MOV     (SP)+,R1                        ;                                       2539
000172  000207                          RTS     PC

; Routine Size:  62 words,      Routine Base:  $CODE$ + 4412
; Maximum stack depth per invocation:  6 words
```

```
;          2573  GLOBAL ROUTINE SET_UPAR (OFFSET) : NOVALUE =
;          2574
;          2575  !+
;          2576  !       THIS ROUTINE SETS UP THE COMMONLY-USED UNIT-RELATED DATA ITEMS FOR
;          2577  !       THE CURRENT CONTROLLER AND GIVEN CST OFFSET.
;          2578  !
;          2579  !       INPUTS:
;          2580  !               OFFSET - WORD OFFSET INTO CURRENT CONTROLLER'S CST WHICH
;          2581  !                        DESCRIBES A UNIT
;          2582  !
;          2583  !       IMPLICIT INPUTS:
;          2584  !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;          2585  !
;          2586  !       IMPLICIT OUTPUTS:
;          2587  !               CUOFF - CURRENT UNIT'S CST OFFSET
;          2588  !               CPLAT - CURRENT PLATTER ADDRESS (MSCP UNIT NUMBER)
;          2589  !               L$LUN - CURRENT UNIT NUMBER (DS UNIT NUMBER)
;          2590  !               T_ADDR - ADDRESS OF CURRENT UNIT'S STATISTICS BLOCK (TALLY)
;          2591  !-
;          2592
;          2593  BEGIN
;          2594
;          2595  CUOFF = .OFFSET;
;          2596  CPLAT = .CST_ADDR [.OFFSET, P_ADDR];
;          2597  L$LUN = .CST_ADDR [.OFFSET, P_UNIT];
;          2598  T_ADDR = TALLY + (.L$LUN * TALLY_LEN * 2);
;          2599
;          2600  END;


                                        .SBTTL  SET.UPAR GLOBAL ROUTINES
000000  010146                  SET.UPAR::
                                        MOV     R1,-(SP)                ;                               2573
000002  016637  000004  000000G         MOV     4(SP),CUOFF             ; OFFSET,*                      2595
000010  016600  000004                  MOV     4(SP),R0                ; CUOFF,*                       2596
000014  006300                          ASL     R0
000016  063700  000000G                 ADD     CST.ADDR,R0
000022  111037  000000G                 MOVB    (R0),CPLAT
000026  105037  000001G                 CLRB    CPLAT+1
000032  011001                          MOV     (R0),R1                 ;                               2597
000034  000301                          SWAB    R1
000036  042701  177740                  BIC     #177740,R1
000042  010137  000000G                 MOV     R1,L$LUN
000046  010146                          MOV     R1,-(SP)                ; L$LUN,*                       2598
000050  012746  000060                  MOV     #60,-(SP)
000054  004737  000000G                 JSR     PC,BL$MUL
000060  062700  000000G                 ADD     #TALLY,R0
000064  010037  000000G                 MOV     R0,T.ADDR
000070  022626                          CMP     (SP)+,(SP)+             ;                               2593
000072  012601                          MOV     (SP)+,R1                ;                               2573
000074  000207                          RTS     PC

; Routine Size:  31 words,      Routine Base:  $CODE$ + 4606
; Maximum stack depth per invocation:  4 words
```

```
;          2601  GLOBAL ROUTINE GET_ENV (CTLR) =
;          2602
;          2603  !+
;          2604  !       THIS ROUTINE SEARCHES THE MSCP ENVELOPE POOL ALLOCATION TABLE (ENV_USE)
;          2605  !       FOR A FREE MSCP ENVELOPE TO ALLOCATE TO THE GIVEN CONTROLLER. IF ONE IS
;          2606  !       FOUND, THE ENVELOPE IS ZEROED OUT, AND THE ENVELOPE INDEX IS RETURNED
;          2607  !       TO THE CALLER. OTHERWISE, A -1 IS RETURNED INDICATING NONE AVAILABLE.
;          2608  !
;          2609  !       INPUTS:
;          2610  !               CTLR - CONTROLLER NUMBER REQUESTING ALLOCATION
;          2611  !-
;          2612
;          2613  BEGIN
;          2614
;          2615  LOCAL
;          2616      INDEX : WORD;
;          2617
;          2618  INDEX = -1;
;          2619
;          2620  INCR COUNT FROM 0 TO (ENV_CNT - 1) DO          ! FOR EACH ENTRY IN ALLOCATION TABLE
;          2621      BEGIN
;          2622
;          2623      IF .ENV_USE [.COUNT] LSS 0                 ! IF ENTRY INDICATES FREE ENVELOPE
;          2624      THEN                                       ! THEN
;          2625          BEGIN
;          2626
;          2627          ENV_USE [.COUNT] = .CTLR;              ! ALLOCATE ENVELOPE TO CONTROLLER
;          2628          INDEX = .COUNT;
;          2629          INCR J FROM 2 TO (ENV_LEN - 1) DO      ! ZERO OUT ENVELOPE
;          2630              MSCP_ENV [.COUNT, .J, ALLBIT] = 0;
;          2631          EXITLOOP;                              ! DONE
;          2632
;          2633          END;
;          2634
;          2635      END;
;          2636
;          2637  RETURN .INDEX;
;          2638
;          2639  END;


                                        .SBTTL  GET.ENV GLOBAL ROUTINES
000000  004137  000000G         GET.ENV::
                                        JSR     R1,$SAVE4                       ;                               2601
000004  012704  177777                  MOV     #-1,R4                          ; *,INDEX                       2618
000010  005003                          CLR     R3                              ; COUNT                         2620
000012  105763  000000G         1$:     TSTB    ENV.USE(R3)                     ; *(COUNT)                      2623
000016  002026                          BGE     3$
000020  116663  000014  000000G         MOVB    14(SP),ENV.USE(R3)              ; CTLR,*(COUNT)                 2627
000026  010304                          MOV     R3,R4                           ; COUNT,INDEX                   2628
000030  010346                          MOV     R3,-(SP)                        ; COUNT,*                       2630
000032  012746  000042                  MOV     #42,-(SP)
000036  004737  000000G                 JSR     PC,BL$MUL
000042  022626                          CMP     (SP)+,(SP)+
000044  012702  000002                  MOV     #2,R2                           ; *,J                           2629
000050  010001                  2$:     MOV     R0,R1                           ;                               2630
000052  060201                          ADD     R2,R1                           ; J,*
```

```
000054  006301                          ASL     R1
000056  005061  000000G                 CLR     MSCP.ENV(R1)
000062  005202                          INC     R2                      ; J                             2629
000064  020227  000041                  CMP     R2,#41                  ; J,*
000070  003767                          BLE     2$
000072  000404                          BR      4$                      ;                               2625
000074  005203                  3$:     INC     R3                      ; COUNT                         2620
000076  020327  000137                  CMP     R3,#137                 ; COUNT,*
000102  003743                          BLE     1$
000104  010400                  4$:     MOV     R4,R0                   ; INDEX,*                       2613
000106  000207                          RTS     PC                      ;                               2601

; Routine Size:  36 words,      Routine Base:  $CODE$ + 4704
; Maximum stack depth per invocation:  8 words
```

```
;          2640  GLOBAL ROUTINE PUT_ENV (INDEX) : NOVALUE =
;          2641
;          2642  !+
;          2643  !       THE MSCP ENVELOPE DESIGNATED BY "INDEX" IS RETURNED TO THE POOL BY THIS
;          2644  !       ROUTINE.
;          2645  !-
;          2646
;          2647  BEGIN
;          2648
;          2649  ENV_USE [.INDEX] = -1;
;          2650
;          2651  END;


                                        .SBTTL  PUT.ENV GLOBAL ROUTINES
000000  016600  000002          PUT.ENV::
                                MOV     2(SP),R0                        ; INDEX,*               2649
000004  112760  000377  000000G        MOVB    #377,ENV.USE(R0)
000012  000207                         RTS     PC                      ;                       2640

; Routine Size:  6 words,       Routine Base:  $CODE$ + 5014
; Maximum stack depth per invocation:  0 words
```

```
:          2652  GLOBAL ROUTINE PUTA_ENV (CTLR) : NOVALUE =
:          2653
:          2654  !+
:          2655  !       THIS ROUTINE DEALLOCATES ALL MSCP ENVELOPES WHICH HAVE BEEN ALLOCATED
:          2656  !       TO A PARTICULAR CONTROLLER.
:          2657  !
:          2658  !       INPUTS:
:          2659  !               CTLR - CONTROLLER NUMBER
:          2660  !-
:          2661
:          2662  BEGIN
:          2663
:          2664  INCR COUNT FROM 0 TO (ENV_CNT - 1) DO          ! FOR EACH ENTRY IN ALLOCATION TABLE
:          2665      BEGIN
:          2666
:          2667      IF .ENV_USE [.COUNT] EQLU .CTLR            ! IF ENVELOPE IS ALLOCATED TO GIVEN CONTROLLER
:          2668      THEN                                       ! THEN
:          2669          ENV_USE [.COUNT] = -1;                 ! DEALLOCATE IT
:          2670
:          2671      END;
:          2672
:          2673  END;                                           ! ROUTINE PUTA_ENV


                                .SBTTL  PUTA.ENV GLOBAL ROUTINES
000000  010146          PUTA.ENV::
                        MOV     R1,-(SP)                ;                               2652
000002  005000                  CLR     R0              ; COUNT                         2664
000004  116001  000000G 1$:     MOVB    ENV.USE(R0),R1  ; *(COUNT),*                    2667
000010  020166  000004          CMP     R1,4(SP)        ; *,CTLR
000014  001003                  BNE     2$
000016  112760  000377  000000G MOVB    #377,ENV.USE(R0); *,*(COUNT)                    2669
000024  005200          2$:     INC     R0              ; COUNT                         2664
000026  020027  000137          CMP     R0,#137         ; COUNT,*
000032  003764                  BLE     1$
000034  012601                  MOV     (SP)+,R1        ;                               2652
000036  000207                  RTS     PC

; Routine Size:  16 words,      Routine Base:  $CODE$ + 5030
; Maximum stack depth per invocation:  2 words
```

```
;        2674  GLOBAL ROUTINE GET_RETPKT (CTLR) =
;        2675
;        2676  !+
;        2677  !       THIS ROUTINE SEARCHES THE RETURN PACKET POOL ALLOCATION TABLE (RP_USE)
;        2678  !       FOR A FREE RETURN PACKET TO ALLOCATE TO THE GIVEN CONTROLLER. IF ONE IS
;        2679  !       FOUND, THE PACKET IS ZEROED OUT, AND THE PACKET INDEX IS RETURNED TO
;        2680  !       THE CALLER. OTHERWISE, A -1 IS RETURNED INDICATING NONE AVAILABLE.
;        2681  !
;        2682  !       INPUTS:
;        2683  !               CTLR - CONTROLLER NUMBER REQUESTING ALLOCATION
;        2684  !-
;        2685
;        2686  BEGIN
;        2687
;        2688  LOCAL
;        2689      INDEX : WORD;
;        2690
;        2691  INDEX = -1;                                    ! ASSUME NONE AVAILABLE
;        2692
;        2693  INCR COUNT FROM 0 TO (RP_CNT - 1) DO           ! FOR EACH ENTRY IN TABLE
;        2694      BEGIN
;        2695
;        2696      IF .RP_USE [.COUNT] LSS 0                  ! IF FREE RETPKT IS FOUND
;        2697      THEN
;        2698          BEGIN
;        2699
;        2700          RP_USE [.COUNT] = .CTLR;               ! ALLOCATE RETURN PACKET TO CONTROLLER
;        2701          INDEX = .COUNT;
;        2702          INCR J FROM 0 TO (RP_LEN - 1) DO       ! ZERO OUT RETPKT
;        2703              RETPKT [.COUNT, .J, ALLBIT] = 0;
;        2704          EXITLOOP;                              ! DONE
;        2705
;        2706          END;
;        2707
;        2708      END;
;        2709
;        2710  RETURN .INDEX;                                 ! RETURN PACKET INDEX (OR -1) TO CALLER
;        2711
;        2712  END;


                                        .SBTTL  GET.RETPKT GLOBAL ROUTINES
000000  004137  000000G         GET.RETPKT::
                                        JSR     R1,$SAVE4                       ;                       2674
000004  012704  177777                  MOV     #-1,R4                          ; *,INDEX               2691
000010  005003                          CLR     R3                              ; COUNT                 2693
000012  105763  000000G         1$:     TSTB    RP.USE(R3)                      ; *(COUNT)              2696
000016  002025                          BGE     3$
000020  116663  000014  000000G         MOVB    14(SP),RP.USE(R3)               ; CTLR,*(COUNT)         2700
000026  010304                          MOV     R3,R4                           ; COUNT,INDEX           2701
000030  010346                          MOV     R3,-(SP)                        ; COUNT,*               2703
000032  012746  000030                  MOV     #30,-(SP)
000036  004737  000000G                 JSR     PC,BL$MUL
000042  022626                          CMP     (SP)+,(SP)+
000044  005002                          CLR     R2                              ; J                     2702
000046  010001                  2$:     MOV     R0,R1                           ;                       2703
000050  060201                          ADD     R2,R1                           ; J,*
```

```
000052  006301                          ASL     R1
000054  005061  000000G                 CLR     RETPKT(R1)
000060  005202                          INC     R2                      ; J                             2702
000062  020227  000027                  CMP     R2,#27                  ; J,*
000066  003767                          BLE     2$
000070  000404                          BR      4$                      ;                               2698
000072  005203                  3$:     INC     R3                      ; COUNT                         2693
000074  020327  000037                  CMP     R3,#37                  ; COUNT,*
000100  003744                          BLE     1$
000102  010400                  4$:     MOV     R4,R0                   ; INDEX,*                       2686
000104  000207                          RTS     PC                      ;                               2674

; Routine Size:  35 words,      Routine Base:  $CODE$ + 5070
; Maximum stack depth per invocation:  8 words
```

```
;          2713  GLOBAL ROUTINE PUT_RETPKT (INDEX) : NOVALUE =
;          2714
;          2715  !+
;          2716  !       THE RETURN PACKET DESIGNATED BY "INDEX" IS RETURNED TO THE POOL BY THIS
;          2717  !       ROUTINE.
;          2718  !-
;          2719
;          2720  BEGIN
;          2721
;          2722  RP_USE [.INDEX] = -1;
;          2723
;          2724  END;


                                                .SBTTL  PUT.RETPKT GLOBAL ROUTINES
000000  016600  000002                    PUT.RETPKT::
                                                MOV     2(SP),R0                        ; INDEX,*                       2722
000004  112760  000377  000000G                 MOVB    #377,RP.USE(R0)
000012  000207                                  RTS     PC                              ;                               2713

; Routine Size:  6 words,       Routine Base:  $CODE$ + 5176
; Maximum stack depth per invocation:  0 words
```

```
;          2725  GLOBAL ROUTINE GET_IO_BUFF (ADDR) : NOVALUE =
;          2727  !+
;          2728  !       THIS ROUTINE HANDLES THE ALLOCATION OF AN I/O BUFFER FROM THE BUFFER
;          2729  !       POOL.
;          2730  !
;          2731  !       INPUTS:
;          2732  !               ADDR - ADDRESS TO STORE THE 2-WORD BUFFER DESCRIPTOR
;          2733  !
;          2734  !       IMPLICIT INPUTS:
;          2735  !               CCTLR - CURRENT CONTROLLER NUMBER
;          2736  !
;          2737  !       OUTPUTS:
;          2738  !               THE ALLOCATED BUFFER'S DESCRIPTOR IS LOADED INTO THE TWO
;          2739  !               WORDS AT ''ADDR'' AND ''ADDR + 2''. OTHERWISE, A ''-1'' IS RETURNED
;          2740  !               AT ''ADDR'' IF NO BUFFERS ARE AVAILABLE.
;          2741  !-
;          2743  BEGIN
;          2744
;          2745  .ADDR = -1;                                   ! ASSUME FAILURE
;          2746  INCR COUNT FROM 0 TO (.NUM_BUFF - 1) DO      ! FOR EACH ENTRY IN BUFFER TABLE
;          2747      BEGIN
;          2748
;          2749      IF .BUFF_OWN [.COUNT] LSS 0               ! IF BUFFER IS FREE
;          2750      THEN                                      ! THEN
;          2751          BEGIN
;          2752
;          2753          BUFF_OWN [.COUNT] = .CCTLR;           ! ALLOCATE BUFFER TO CONTROLLER
;          2754          .ADDR = .BUFF_DESC [.COUNT, BD_LO];   ! RETURN BUFFER DESCRIPTOR
;          2755          (.ADDR + 2) = .BUFF_DESC [.COUNT, BD_HI];
;          2756          EXITLOOP;                             ! DONE
;          2757
;          2758          END;
;          2759
;          2760      END;                                      ! BUFFER TABLE LOOP
;          2761
;          2762  END;                                          ! ROUTINE GET_IO_BUFF
                                        .SBTTL  GET.IO.BUFF GLOBAL ROUTINES
000000  004137  000000G                 GET.IO.BUFF::
                                        JSR     R1,$SAVE2                       ;                       2725
000004  016601  000010                  MOV     10(SP),R1                       ; ADDR,*                2745
000010  012711  177777                  MOV     #-1,(R1)
000014  005002                          CLR     R2                              ; COUNT                 2746
000016  000420                          BR      3$
000020  105762  000000G                 1$:     TSTB    BUFF.OWN(R2)            ; *(COUNT)              2749
000024  002014                          BGE     2$
000026  113762  000000G 000000G         MOVB    CCTLR,BUFF.OWN(R2)              ; *,*(COUNT)            2753
000034  010200                          MOV     R2,R0                           ; COUNT,*               2754
000036  006300                          ASL     R0
000040  006300                          ASL     R0
000042  016011  000000G                 MOV     BUFF.DESC(R0),(R1)
000046  016061  000002G 000002          MOV     BUFF.DESC+2(R0),2(R1)           ;                       2755
000054  000207                          RTS     PC                              ;                       2751
000056  005202                          2$:     INC     R2                      ; COUNT                 2746
000060  020237  000000G                 3$:     CMP     R2,NUM.BUFF             ; COUNT,*
000064  002755                          BLT     1$
000066  000207                          RTS     PC                              ;                       2725
; Routine Size:  28 words,      Routine Base:  $CODE$ + 5212
; Maximum stack depth per invocation:  4 words
```

```
;          2763  GLOBAL ROUTINE PUT_IO_BUFF (ADDR) : NOVALUE =
;          2764
;          2765  !+
;          2766  !       THIS ROUTINE HANDLES THE DEALLOCATION OF AN I/O BUFFER, RETURNING IT
;          2767  !       TO THE BUFFER POOL.
;          2768  !
;          2769  !       INPUTS:
;          2770  !               ADDR - ADDRESS OF THE 2-WORD BUFFER DESCRIPTOR TO BE
;          2771  !                       DEALLOCATED
;          2772  !-
;          2773
;          2774  BEGIN
;          2775
;          2776  INCR COUNT FROM 0 TO (.NUM_BUFF - 1) DO          ! FOR EACH ENTRY IN BUFFER TABLE
;          2777      BEGIN
;          2778
;          2779      IF ((.BUFF_DESC [.COUNT, BD_LO] EQLA ..ADDR) AND     ! IF THIS IS THE BUFFER'S ENTRY
;          2780          (.BUFF_DESC [.COUNT, BD_HI] EQLU .(.ADDR + 2)))
;          2781      THEN                                                 ! THEN
;          2782          BEGIN
;          2783
;          2784          BUFF_OWN [.COUNT] = -1;                          ! DEALLOCATE BUFFER
;          2785          EXITLOOP;                                        ! DONE
;          2786
;          2787          END;
;          2788
;          2789      END;                                                 ! BUFFER TABLE SEARCH LOOP
;          2790
;          2791  END;                                                    ! ROUTINE PUT_IO_BUFF


                                        .SBTTL  PUT.IO.BUFF GLOBAL ROUTINES
000000  004137  000000G         PUT.IO.BUFF::
                                        JSR     R1,$SAVE2                       ;                       2763
000004  005002                         CLR     R2                              ; COUNT                 2776
000006  000422                         BR      3$
000010  010201                 1$:     MOV     R2,R1                           ; COUNT,*               2779
000012  006301                         ASL     R1
000014  006301                         ASL     R1
000016  026176  000000G 000010         CMP     BUFF.DESC(R1),@10(SP)           ; *,ADDR
000024  001012                         BNE     2$
000026  016600  000010                 MOV     10(SP),R0                       ; ADDR,*                2780
000032  026160  000002G 000002         CMP     BUFF.DESC+2(R1),2(R0)
000040  001004                         BNE     2$
000042  112762  000377  000000G       MOVB    #377,BUFF.OWN(R2)               ; *,*(COUNT)            2784
000050  000207                         RTS     PC                              ;                       2782
000052  005202                 2$:     INC     R2                              ; COUNT                 2776
000054  020237  000000G        3$:     CMP     R2,NUM.BUFF                     ; COUNT,*
000060  002753                         BLT     1$
000062  000207                         RTS     PC                              ;                       2763

; Routine Size:  26 words,      Routine Base:  $CODE$ + 5302
; Maximum stack depth per invocation:  4 words
```

```
:          2792  GLOBAL ROUTINE PUTA_BUFF : NOVALUE =
:          2793
:          2794  !+
:          2795  !       THIS ROUTINE DEALLOCATES ALL I/O BUFFERS WHICH HAVE BEEN ALLOCATED TO
:          2796  !       THE CURRENT CONTROLLER (CCTLR).
:          2797  !-
:          2798
:          2799  BEGIN
:          2800
:          2801  INCR COUNT FROM 0 TO (.NUM_BUFF - 1) DO          ! FOR EACH ENTRY IN BUFFER TABLE
:          2802      BEGIN
:          2803
:          2804      IF .BUFF_OWN [.COUNT] EQLU .CCTLR            ! IF THIS BUFFER IS ALLOCATED TO THE CURRENT CONTROLLER
:          2805      THEN                                         ! THEN
:          2806          BUFF_OWN [.COUNT] = -1;                  ! DEALLOCATE IT
:          2807
:          2808      END;                                         ! BUFFER TABLE ENTRY LOOP
:          2809
:          2810  END;                                             ! ROUTINE PUTA_BUFF


                                        .SBTTL  PUTA.BUFF GLOBAL ROUTINES
000000  010146                  PUTA.BUFF::
                                        MOV     R1,-(SP)                        ;                               2792
000002  005000                          CLR     R0                              ; COUNT                         2801
000004  000411                          BR      3$
000006  116001  000000G         1$:     MOVB    BUFF.OWN(R0),R1                 ; *(COUNT),*                    2804
000012  020137  000000G                 CMP     R1,CCTLR
000016  001003                          BNE     2$
000020  112760  000377  000000G        MOVB    #377,BUFF.OWN(R0)               ; *,*(COUNT)                    2806
000026  005200                  2$:     INC     R0                              ; COUNT                         2801
000030  020037  000000G         3$:     CMP     R0,NUM.BUFF                     ; COUNT,*
000034  002764                          BLT     1$
000036  012601                          MOV     (SP)+,R1                        ;                               2792
000040  000207                          RTS     PC

; Routine Size:  17 words,      Routine Base:  $CODE$ + 5366
; Maximum stack depth per invocation:  2 words
```

```
;        2811  GLOBAL ROUTINE OUT_IODQ =
;        2812
;        2813  !+
;        2814  !       THIS ROUTINE RETURNS TO THE CALLER THE NEXT RETPKT INDEX TO BE
;        2815  !       PROCESSED FROM THE I/O DONE QUEUE (IODQ). THE "OUT" POINTER TO THE
;        2816  !       QUEUE IS ALSO UPDATED.
;        2817  !
;        2818  !       INPUTS:
;        2819  !               NONE
;        2820  !
;        2821  !       OUTPUTS:
;        2822  !               THE INDEX OF THE NEXT RETPKT TO BE PROCESSED.
;        2823  !-
;        2824
;        2825  BEGIN
;        2826
;        2827  LOCAL
;        2828      INDEX : WORD;
;        2829
;        2830  INDEX = .IODQ [.IODQ_OUT];                  ! GET NEXT RETPKT INDEX
;        2831  IODQ_OUT = .IODQ_OUT + 1;                   ! ADVANCE "OUT" POINTER
;        2832  IF .IODQ_OUT GEQU IODQ_LEN                  ! IF BEYOND END OF QUEUE
;        2833  THEN                                        ! THEN
;        2834      IODQ_OUT = 0;                           ! SET POINTER TO BEGINNING OF QUEUE
;        2835  RETURN .INDEX;                             ! RETURN INDEX TO CALLER
;        2836
;        2837  END;


                                        .SBTTL  OUT.IODQ GLOBAL ROUTINES
000000  013700  000000G         OUT.IODQ::
                                        MOV     IODQ.OUT,R0             ;                       2830
000004  116000  000000G                 MOVB    IODQ(R0),R0             ; *,INDEX
000010  042700  177400                  BIC     #177400,R0              ; *,INDEX
000014  005237  000000G                 INC     IODQ.OUT                ;                       2831
000020  023727  000000G 000040          CMP     IODQ.OUT,#40            ;                       2832
000026  103402                          BLO     1$
000030  005037  000000G                 CLR     IODQ.OUT                ;                       2834
000034  000207                  1$:     RTS     PC                      ;                       2811

; Routine Size:  15 words,      Routine Base:  $CODE$ + 5430
; Maximum stack depth per invocation:  0 words
```

```
;        2838  GLOBAL ROUTINE IN_IODQ (INDEX) : NOVALUE =
;        2839
;        2840  !+
;        2841  !       THIS ROUTINE INSERTS A RETURN PACKET INDEX INTO THE I/O DONE QUEUE, AND
;        2842  !       UPDATES THE IODQ_IN POINTER.
;        2843  !-
;        2844
;        2845  BEGIN
;        2846
;        2847  IF (((.IODQ_IN + 1) NEQU .IODQ_OUT) AND          ! IF I/O DONE QUEUE IS NOT FULL
;        2848      (.IODQ_IN - (IODQ_LEN - 1) NEQU .IODQ_OUT))
;        2849  THEN                                             ! THEN
;        2850      BEGIN
;        2851
;        2852      IODQ [.IODQ_IN] = .INDEX;                    ! LOAD INDEX INTO QUEUE
;        2853      IODQ_IN = .IODQ_IN + 1;                      ! ADVANCE "IN" POINTER
;        2854      IF .IODQ_IN GEQU IODQ_LEN                    ! IF BEYOND END OF QUEUE
;        2855      THEN                                         ! THEN
;        2856          IODQ_IN = 0;                             ! CYCLE BACK TO BEGINNING OF QUEUE
;        2857
;        2858      END;                                         ! IF IODQ IS NOT FULL
;        2859
;        2860  END;


                                        .SBTTL  IN.IODQ GLOBAL ROUTINES
000000  010146                  IN.IODQ::
                                        MOV     R1,-(SP)                        ;                       2838
000002  013701  000000G                 MOV     IODQ.IN,R1                      ;                       2847
000006  010100                          MOV     R1,R0
000010  005200                          INC     R0
000012  020037  000000G                 CMP     R0,IODQ.OUT
000016  001421                          BEQ     1$
000020  010100                          MOV     R1,R0                           ;                       2848
000022  162700  000037                  SUB     #37,R0
000026  020037  000000G                 CMP     R0,IODQ.OUT
000032  001413                          BEQ     1$
000034  116661  000004  000000G         MOVB    4(SP),IODQ(R1)                  ; INDEX,*               2852
000042  005237  000000G                 INC     IODQ.IN                         ;                       2853
000046  023727  000000G 000040          CMP     IODQ.IN,#40                     ;                       2854
000054  103402                          BLO     1$
000056  005037  000000G                 CLR     IODQ.IN                         ;                       2856
000062  012601                  1$:     MOV     (SP)+,R1                        ;                       2838
000064  000207                          RTS     PC

; Routine Size:  27 words,      Routine Base:  $CODE$ + 5466
; Maximum stack depth per invocation:  2 words
```

```
;          2861  GLOBAL ROUTINE DROP_CTLR (CTLR, REASON) : NOVALUE =
;          2862
;          2863  !+
;          2864  !       THIS ROUTINE DROPS ALL UNITS ASSOCIATED WITH THE CONTROLLER DESIGNATED
;          2865  !       BY "CTLR". THE REASON FOR DROPPING THE DEVICE IS LOADED INTO THE DUR
;          2866  !       VECTOR FOR EACH ATTACHED UNIT. THIS DATA IS THEN USED BY THE DROP UNIT
;          2867  !       SECTION.
;          2868  !-
;          2869
;          2870  BEGIN
;          2871
;          2872  LOCAL
;          2873      UNIT;
;          2874
;          2875  INCR OFFSET FROM (0 + OF_UN) TO (3 + OF_UN) DO  ! FOR EACH UNIT IN CST
;          2876      BEGIN
;          2877
;          2878      IF .CST [.CTLR, .OFFSET, P_PRES] EQLU PRESENT       ! IF UNIT IS CONFIGURED
;          2879      THEN
;          2880          BEGIN
;          2881
;          2882          UNIT = .CST [.CTLR, .OFFSET, P_UNIT];           ! GET DS UNIT NUMBER
;          2883          DUR [.UNIT] = .REASON;                          ! SET REASON FOR DROPPING UNIT
;          2884          DODU (.UNIT);                                   ! DROP UNIT
;          2885
;          2886          END;
;          2887
;          2888      END;
;          2889
;          2890  END;
                                        .SBTTL  DROP.CTLR GLOBAL ROUTINES
000000  004137  000000G                 DROP.CTLR::
                                        JSR     R1,$SAVE3                       ;                       2861
000004  016646  000014                  MOV     14(SP),-(SP)                    ; CTLR,*                2878
000010  012746  000007                  MOV     #7,-(SP)
000014  004737  000000G                 JSR     PC,BL$MUL
000020  010003                          MOV     R0,R3
000022  012702  000003                  MOV     #3,R2                           ; *,OFFSET              2875
000026  010300                  1$:     MOV     R3,R0                           ;                       2878
000030  060200                          ADD     R2,R0                           ; OFFSET,*
000032  006300                          ASL     R0
000034  032760  040000  000000G         BIT     #40000,CST(R0)
000042  001412                          BEQ     2$
000044  016001  000000G                 MOV     CST(R0),R1                      ; *,UNIT                2882
000050  000301                          SWAB    R1                              ; UNIT
000052  042701  177740                  BIC     #177740,R1                      ; *,UNIT
000056  116661  000016  000000G         MOVB    16(SP),DUR(R1)                  ; REASON,*(UNIT)        2883
000064  010100                          MOV     R1,R0                           ; UNIT,*                2884
000066  104451                          TRAP    51
000070  005202                  2$:     INC     R2                              ; OFFSET                2875
000072  020227  000006                  CMP     R2,#6                           ; OFFSET,*
000076  003753                          BLE     1$
000100  022626                          CMP     (SP)+,(SP)+                     ;                       2870
000102  000207                          RTS     PC                              ;                       2861

; Routine Size:  34 words,      Routine Base:  $CODE$ + 5554
; Maximum stack depth per invocation:  8 words
```

```
;         2891  GLOBAL ROUTINE DRV_CTLERR (CTLR) : NOVALUE =
;         2892
;         2893  !+
;         2894  !       THIS ROUTINE IS CALLED BY DRV_TIMCHK AND FATAL_ERROR WHENEVER AN
;         2895  !       UNRECOVERABLE CONTROLLER ERROR HAS BEEN DETECTED. ITS PURPOSE IS TO
;         2896  !       CLEAN UP ALL CONTROLLER-RELATED DATA IN THE "DRIVER" PORTION OF THE
;         2897  !       PROGRAM. THIS INCLUDES MARKING THE CONTROLLER OFFLINE, CLEARING THE
;         2898  !       C-RING COUNT, AND DEALLOCATING MSCP ENVELOPES DESCRIBED IN THE RESPONSE
;         2899  !       RING AND OUTSTANDING COMMAND AREA (OUTC_LIST).
;         2900  !
;         2901  !       INPUTS:
;         2902  !               CTLR - DYING CONTROLLER NUMBER
;         2903  !-
;         2904
;         2905  BEGIN
;         2906
;         2907  LOCAL
;         2908      D_ADDR : REF BLOCK [DCT_LEN, WORD] FIELD (DC_FIELDS),       ! CONTROLLER'S DCT ADDRESS
;         2909      INDEX : WORD,
;         2910      STIDX : WORD,                                               ! STARTING OUTC INDEX
;         2911      ENDIDX : WORD;                                              ! ENDING OUTC INDEX
;         2912
;         2913  D_ADDR = DCT + (.CTLR * DCT_LEN * 2);                         ! GET CONTROLLER'S DCT ADDR
;         2914  D_ADDR [WORD0] = 0;                                            ! MARK DCT OFFLINE AND CLEAR CRING_CNT
;         2915  PUTA_ENV (.CTLR);                                             ! RELEASE ALL ENVELOPES ALLOCATED TO CONTROLLER
;         2916  STIDX = .CTLR * OUTC_CNT;                                     ! START OF CTLR'S OUTC_LIST AND _TIMR
;         2917  ENDIDX = .STIDX + OUTC_CNT - 1;                              ! END OF OUTC
;         2918  INCR COUNT FROM .STIDX TO .ENDIDX DO                          ! FOR EACH OUTC ENTRY
;         2919      BEGIN
;         2920
;         2921      OUTC_TIMR [.COUNT] = 0;                                   ! TURN OF COMMAND TIMER
;         2922      OUTC_LIST [.COUNT] = -1;                                  ! INIT SLOT
;         2923
;         2924      END;                                                      ! OUTC SLOT LOOP
;         2925
;         2926  END;                                                          ! ROUTINE DRV_CTLERR


                                        .SBTTL  DRV.CTLERR GLOBAL ROUTINES
000000  004137  000000G           DRV.CTLERR::
                                        JSR     R1,$SAVE2                       ;                               2891
000004  016601  000010                  MOV     10(SP),R1                       ; CTLR,*                        2913
000010  010146                          MOV     R1,-(SP)
000012  012746  000022                  MOV     #22,-(SP)
000016  004737  000000G                 JSR     PC,BL$MUL
000022  062700  000000G                 ADD     #DCT,R0
000026  005010                          CLR     (R0)                            ; D.ADDR                        2914
000030  010116                          MOV     R1,(SP)                         ;                               2915
000032  004737  005030'                 JSR     PC,PUTA.ENV
000036  010100                          MOV     R1,R0                           ;                               2916
000040  006300                          ASL     R0
000042  006300                          ASL     R0
000044  006300                          ASL     R0
000046  006300                          ASL     R0
000050  010002                          MOV     R0,R2                           ; STIDX,ENDIDX                  2917
000052  062702  000017                  ADD     #17,R2                          ; *,ENDIDX
000056  010001                          MOV     R0,R1                           ; STIDX.COUNT                   2918
```

```
000060  005301                          DEC     R1                              ; COUNT
000062  000407                          BR      2$
000064  010100                  1$:     MOV     R1,R0                           ; COUNT,*                       2921
000066  006300                          ASL     R0
000070  005060  000000G                 CLR     OUTC.TIMR(R0)
000074  112761  000377  000000G         MOVB    #377,OUTC.LIST(R1)              ; *,*(COUNT)                    2922
000102  005201                  2$:     INC     R1                              ; COUNT                         2918
000104  020102                          CMP     R1,R2                           ; COUNT,ENDIDX
000106  003766                          BLE     1$
000110  022626                          CMP     (SP)+,(SP)+                     ;                               2905
000112  000207                          RTS     PC                              ;                               2891

; Routine Size:  38 words,      Routine Base:  $CODE$ + 5660
; Maximum stack depth per invocation:  6 words
```

```
;        2927  GLOBAL ROUTINE HARD_ERR (ERRCNT) : NOVALUE =
;        2928
;        2929  !+
;        2930  !       THIS ROUTINE IS CALLED BY IO_RETPKT, DM_TALLY, AND OTHERS TO UPDATE THE
;        2931  !       HARD ERROR STATISTIC FIELD FOR THE CURRENT UNIT. IF THE HARD ERROR
;        2932  !       COUNT HAS EXCEEDED THE OPERATOR-SPECIFIED LIMIT, THEN THE UNIT IS
;        2933  !       DROPPED FROM TESTING.
;        2934  !
;        2935  !       INPUTS:
;        2936  !               ERRCNT - NUMBER OF HARD ERRORS TO ADD TO CURRENT TOTAL
;        2937  !
;        2938  !       IMPLICIT INPUTS:
;        2939  !               L$LUN - CURRENT UNIT NUMBER
;        2940  !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;        2941  !               CUOFF - CST OFFSET FOR CURRENT UNIT
;        2942  !               T_ADDR - ADDRESS OF CURRENT UNIT'S STATISTICS BLOCK (TALLY)
;        2943  !-
;        2944
;        2945  BEGIN
;        2946
;        2947  T_ADDR [ER_HRD] = .T_ADDR [ER_HRD] + .ERRCNT;    ! UPDATE UNIT'S HARD ERROR COUNT
;        2948  IF .SWP_ERROR NEQU 0                             ! IF THE USER SPECIFIED A HARD ERROR LIMIT
;        2949  THEN                                             ! THEN
;        2950      BEGIN
;        2951
;        2952      IF .T_ADDR [ER_HRD] GEQU .SWP_ERROR          ! IF HARD ERROR LIMIT REACHED
;        2953      THEN                                         ! THEN
;        2954          BEGIN
;        2955
;        2956          IF .CST_ADDR [.CUOFF, P_STAT] EQLU ONLINE    ! IF UNIT IS STILL ONLINE
;        2957          THEN                                         ! THEN
;        2958              BEGIN
;        2959
;        2960              DUR [.L$LUN] = DU_HERR;                  ! LOAD REASON TO DROP UNIT
;        2961              DODU (.L$LUN);                           ! DROP UNIT
;        2962
;        2963              END;
;        2964
;        2965          END;                                         ! IF HARD ERROR LIMIT REACHED
;        2966
;        2967      END;                                             ! IF THE USER SPECIFIED A HARD ERROR LIMIT
;        2968
;        2969  END;                                                 ! ROUTINE HARD_ERR


                                .SBTTL  HARD.ERR GLOBAL ROUTINES
000000  010146                  HARD.ERR::
                                MOV     R1,-(SP)                        ;                       2927
000002  013701  000000G         MOV     T.ADDR,R1                       ;                       2947
000006  066661  000004  000030  ADD     4(SP),30(R1)                    ; ERRCNT,*
000014  013700  000000G         MOV     SWP.ERROR,R0                    ;                       2948
000020  001421                  BEQ     1$
000022  026100  000030          CMP     30(R1),R0                       ;                       2952
000026  103416                  BLO     1$
000030  013700  000000G         MOV     CUOFF,R0                        ;                       2956
000034  006300                  ASL     R0
000036  063700  000000G         ADD     CST.ADDR,R0
```

```
000042  032710  020000                          BIT     #20000,(R0)
000046  001406                                  BEQ     1$
000050  013700  000000G                         MOV     L$LUN,R0                ;                       2960
000054  112760  000003  000000G                 MOVB    #3,DUR(R0)
000062  104451                                  TRAP    51                      ;                       2961
000064  012601                          1$:     MOV     (SP)+,R1                ;                       2927
000066  000207                                  RTS     PC

; Routine Size:  28 words,      Routine Base:  $CODE$ + 5774
; Maximum stack depth per invocation:  3 words
```

```
;       2970  GLOBAL ROUTINE UPD_IOC (ADDR, COUNT) : NOVALUE =
;       2971
;       2972  !+
;       2973  !       THIS ROUTINE IS CALLED (A) DURING THE MULTI-DRIVE SUBTEST FROM MD_TALLY
;       2974  !       FOR ALL I/O TRANSFER RETURN PACKETS WITH ''SUCCESS'' STATUS CODES, AND
;       2975  !       (B) DURING THE DM EXERCISER SUBTEST FROM DM_TALLY WHEN EXAMINING THE
;       2976  !       COMMUNICATION AREA (DM_COMM). ITS PURPOSE IS TO UPDATE THE I/O COUNT
;       2977  !       (NUMBER OF READS, WRITES, OR SEEKS) FOR THE CURRENT UNIT.
;       2978  !
;       2979  !       AFTER THE LOW-ORDER FIELD OF THE APPROPRIATE COUNT IS UPDATED, THIS
;       2980  !       ROUTINE CHECKS FOR ONE OF TWO OVERFLOW CONDITIONS: (A) IF THE
;       2981  !       HIGH-ORDER FIELD OF THE COUNT IS ZERO, THEN THE LOW-ORDER FIELD IS
;       2982  !       CHECKED FOR EXCEEDING 65,000. IF TRUE, THEN THE HIGH-ORDER FIELD IS
;       2983  !       LOADED WITH 65 (FOR 65,000) AND THE LOW-ORDER FIELD IS REDUCED BY
;       2984  !       65,000. (B) IF THE HIGH-ORDER FIELD OF THE UPDATED COUNT IS NON-ZERO,
;       2985  !       THEN THE LOW-ORDER FIELD IS CHECKED FOR EXCEEDING 1000. IF TRUE, THEN
;       2986  !       THE HIGH-ORDER FIELD IS INCREMENTED BY 1 FOR EACH 1000 SUBTRACTED FROM
;       2987  !       THE LOW-ORDER FIELD.
;       2988  !
;       2989  !       INPUTS:
;       2990  !               ADDR - ADDRESS OF THE LOW-ORDER FIELD OF THE I/O COUNT TO BE
;       2991  !                      UPDATED (IN THE CURRENT UNIT'S STATISTICS TABLE (TALLY))
;       2992  !               COUNT - AMOUNT TO BE ADDED TO THE I/O COUNT
;       2993  !-
;       2994
;       2995  BEGIN
;       2996
;       2997  .ADDR = ..ADDR + .COUNT;                  ! UPDATE THE COUNT (LOW-ORDER FIELD)
;       2998  IF .(.ADDR + 2) EQLU 0                    ! IF HIGH-ORDER FIELD IS ZERO
;       2999  THEN                                      ! THEN
;       3000      BEGIN
;       3001
;       3002      IF ..ADDR GEQU 65000                  ! IF LOW-ORDER FIELD EXCEEDS 65,000
;       3003      THEN                                  ! THEN
;       3004          BEGIN
;       3005
;       3006          .ADDR = ..ADDR - 65000;           ! BECOMES LOW-ORDER 1000
;       3007          (.ADDR + 2) = 65;                 ! HIGH-ORDER (TIMES 1000)
;       3008
;       3009          END;
;       3010
;       3011      END
;       3012  ELSE                                      ! ELSE - HIGH-ORDER FIELD IS X1000
;       3013      BEGIN
;       3014
;       3015      WHILE ..ADDR GEQU 1000 DO             ! WHILE LOW-ORDER FIELD EXCEEDS 1000
;       3016          BEGIN
;       3017
;       3018          .ADDR = ..ADDR - 1000;            ! SUBTRACT 1000 FROM LOW-ORDER FIELD
;       3019          (.ADDR + 2) = .(.ADDR + 2) + 1;   ! INCREMENT HIGH-ORDER FIELD
;       3020
;       3021          END;
;       3022
;       3023      END;
;       3024
;       3025  END;                                      ! ROUTINE UPD_IOC
```

```
                                        .SBTTL  UPD.IOC GLOBAL ROUTINES
000000  010146                  UPD.IOC::
                                        MOV     R1,-(SP)                ;                               2970
000002  016600  000006                  MOV     6(SP),R0                ; ADDR,*                        2997
000006  066610  000004                  ADD     4(SP),(R0)              ; COUNT,*
000012  012701  000002                  MOV     #2,R1                   ;                               2998
000016  060001                          ADD     R0,R1
000020  005711                          TST     (R1)
000022  001010                          BNE     1$
000024  021027  176750                  CMP     (R0),#176750            ;                               3002
000030  103414                          BLO     2$
000032  062710  001030                  ADD     #1030,(R0)              ;                               3006
000036  012711  000101                  MOV     #101,(R1)               ;                               3007
000042  000407                          BR      2$                      ;                               2998
000044  021027  001750          1$:     CMP     (R0),#1750              ;                               3015
000050  103404                          BLO     2$
000052  162710  001750                  SUB     #1750,(R0)              ;                               3018
000056  005211                          INC     (R1)                    ;                               3019
000060  000771                          BR      1$                      ;                               3015
000062  012601                  2$:     MOV     (SP)+,R1                ;                               2970
000064  000207                          RTS     PC

; Routine Size:  27 words,      Routine Base:  $CODE$ + 6064
; Maximum stack depth per invocation:  2 words
```

```
;         3026  GLOBAL ROUTINE OVF_CHK (ADDR) : NOVALUE =
;         3027
;         3028  !+
;         3029  !       THIS ROUTINE IS CALLED (A) DURING THE MULTI-DRIVE SUBTEST FROM MD_TALLY
;         3030  !       FOR ALL I/O TRANSFER RETURN PACKETS WITH ''SUCCESS'' STATUS CODES, AND
;         3031  !       (B) DURING THE DM EXERCISER SUBTEST FROM DM_TALLY WHEN EXAMINING THE
;         3032  !       COMMUNICATION AREA (DM_COMM). ITS PURPOSE IS TO CHECK FOR OVERFLOW IN
;         3033  !       CERTAIN STATISTICAL FIELDS OF THE CURRENT UNIT. SPECIFICALLY, THE
;         3034  !       LOW-ORDER FIELD OF THE NUMBER OF BYTES READ OR WRITTEN IS CHECKED FOR
;         3035  !       EXCEEDING 1000. IF TRUE, THEN THE HIGH-ORDER COUNT IS INCREMENTED. IF
;         3036  !       THAT EXCEEDS 1000, THEN THE MEGABYTE COUNT IS INCREMENTED.
;         3037  !
;         3038  !       INPUTS:
;         3039  !               ADDR - ADDRESS OF THE BR_LO OR BW_LO FIELD FOR THE CURRENT UNIT
;         3040  !                       (SEE STATISTIC TABLE (TALLY) LAYOUT)
;         3041  !-
;         3042
;         3043  BEGIN
;         3044
;         3045  WHILE ..ADDR GEQU 1000 DO                        ! IF LO-ORDER OVERFLOW
;         3046      BEGIN
;         3047
;         3048      .ADDR = ..ADDR - 1000;                       ! SUBTRACT 1000
;         3049      (.ADDR + 2) = .(.ADDR + 2) + 1;              ! INCR HI-ORDER
;         3050
;         3051      END;
;         3052
;         3053  IF .(.ADDR + 2) GEQU 1000                        ! IF HI-ORDER OVERFLOW
;         3054  THEN                                             ! THEN
;         3055      BEGIN
;         3056
;         3057      (.ADDR + 2) = .(.ADDR + 2) - 1000;           ! SUBTRACT 1000
;         3058      (.ADDR + 4) = .(.ADDR + 4) + 1;              ! INCREMENT MBYTES
;         3059
;         3060      END;
;         3061
;         3062  END;                                             ! ROUTINE OVF_CHK

                                        .SBTTL  OVF.CHK GLOBAL ROUTINES
000000  010146                  OVF.CHK::
                                        MOV     R1,-(SP)                        ;                       3026
000002  016600  000004                   MOV     4(SP),R0                        ; ADDR,*                3045
000006  012701  000002                   MOV     #2,R1                           ;                       3049
000012  060001                          ADD     R0,R1
000014  021027  001750          1$:     CMP     (R0),#1750                      ;                       3045
000020  103404                          BLO     2$
000022  162710  001750                   SUB     #1750,(R0)                      ;                       3048
000026  005211                          INC     (R1)                            ;                       3049
000030  000771                          BR      1$                              ;                       3045
000032  021127  001750          2$:     CMP     (R1),#1750                      ;                       3053
000036  103404                          BLO     3$
000040  162711  001750                   SUB     #1750,(R1)                      ;                       3057
000044  005260  000004                   INC     4(R0)                           ;                       3058
000050  012601                  3$:     MOV     (SP)+,R1                        ;                       3026
000052  000207                          RTS     PC
; Routine Size:  22 words,      Routine Base:  $CODE$ + 6152
; Maximum stack depth per invocation:  2 words
```

```
;         3063  GLOBAL ROUTINE XFR_CHK : NOVALUE =
;         3064
;         3065  !+
;         3066  !       THIS ROUTINE IS CALLED (A) DURING THE MULTI-DRIVE SUBTEST FROM MD_TALLY
;         3067  !       FOR ALL I/O TRANSFER RETURN PACKETS WITH "SUCCESS" STATUS CODES, AND
;         3068  !       (B) DURING THE DM EXERCISER SUBTEST FROM DM_TALLY WHEN EXAMINING THE
;         3069  !       COMMUNICATION AREA (DM_COMM). ITS PURPOSE IS TO CALCULATE THE TOTAL
;         3070  !       NUMBER OF BYTES TRANSFERRED THUS FAR ON THE CURRENT UNIT, AND TO CHECK
;         3071  !       THIS SUM AGAINST THE OPERATOR-SPECIFIED LIMIT. IF THE LIMIT HAS BEEN
;         3072  !       REACHED, THEN THE UNIT IS REMOVED FROM THE CURRENT PASS.
;         3073  !
;         3074  !       IMPLICIT INPUTS:
;         3075  !               T_ADDR - ADDRESS OF THE CURRENT UNIT'S STATISTICS BLOCK (TALLY)
;         3076  !               L$LUN - CURRENT UNIT NUMBER
;         3077  !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;         3078  !               DMC_ADDR - ADDRESS OF CURRENT CONTROLLER'S DM_COMM AREA
;         3079  !               CUOFF - CST OFFSET FOR CURRENT UNIT
;         3080  !-
;         3081
;         3082  BEGIN
;         3083
;         3084  LOCAL
;         3085      SUM : WORD;                                   ! TOTAL NUMBER OF BYTES TRANSFERRED TO/FROM A UNIT
;         3086
;         3087  IF .SWP_XFER NEQU 0                              ! IF A TRANSFER LIMIT WAS SPECIFIED
;         3088  THEN                                            ! THEN
;         3089      BEGIN
;         3090
;         3091      SUM = .T_ADDR [MB_READ] + .T_ADDR [MB_WRIT];      ! TOTAL BYTES TRANSFERRED
;         3092      IF .SUM GEQU .SWP_XFER                            ! IF TRANSFER LIMIT IS REACHED
;         3093      THEN                                              ! THEN
;         3094          BEGIN
;         3095
;         3096          CPT [.L$LUN] = NO;                            ! MARK UNIT INACTIVE
;         3097          CST_ADDR [U_CNT] = .CST_ADDR [U_CNT] - 1;     ! DECREMENT ACTIVE UNIT COUNT
;         3098          DMC_ADDR [.CUOFF - OF_UN, ALLBIT] = -1;       ! IF DM EXER, STOP FPT ON UNIT
;         3099          IF MANUAL                                     ! IF ATTENDED
;         3100          THEN                                          ! THEN
;         3101              BEGIN
;         3102
;         3103              PRINTF (CRLF);
;         3104              PRINTF (ETIME, .HOURS, .MINUTES, .SECONDS); ! ELAPSED TIME
;         3105              PRINTF (MSG_07, .L$LUN);                    ! "UNIT XX. - TRANSFER LIMIT REACHED"
;         3106
;         3107              END;
;         3108
;         3109          END;
;         3110
;         3111      END;                                          ! IF A TRANSFER LIMIT WAS SPECIFIED
;         3112
;         3113  END;                                              ! ROUTINE XFR_CHK


                                        .SBTTL  XFR.CHK GLOBAL ROUTINES
000000  004137  000000G       XFR.CHK::
                                        JSR     R1,$SAVE2                       ;                       3063
000004  013702  000000G                 MOV     SWP.XFER,R2                     ;                       3087
```

```
000010  001467                          BEQ     1$                              ;
000012  013701  000000G                 MOV     T.ADDR,R1                       ;                                       3091
000016  010100                          MOV     R1,R0                           ; T.ADDR,*
000020  016101  000020                  MOV     20(R1),R1                       ; *,SUM
000024  066001  000026                  ADD     26(R0),R1                       ; *,SUM
000030  020102                          CMP     R1,R2                           ; SUM,*                                 3092
000032  103456                          BLO     1$                                                                      3096
000034  013700  000000G                 MOV     L$LUN,R0                        ;
000040  105060  000000G                 CLRB    CPT(R0)                                                                 3097
000044  013700  000000G                 MOV     CST.ADDR,R0                     ;
000050  105360  000005                  DECB    5(R0)                                                                   3098
000054  013700  000000G                 MOV     CUOFF,R0                        ;
000060  006300                          ASL     R0
000062  063700  000000G                 ADD     DMC.ADDR,R0
000066  012760  177777  177772         MOV     #-1,-6(R0)
000074  104450                          TRAP    50                              ;                                       3099
000076  103034                          BHIS    1$                                                                      3103
000100  012746  000000G                 MOV     #CRLF,-(SP)                     ;
000104  012746  000001                  MOV     #1,-(SP)
000110  010600                          MOV     SP,R0                           ; SP,*
000112  104417                          TRAP    17
000114  013716  000000G                 MOV     SECONDS,(SP)                    ;                                       3104
000120  013746  000000G                 MOV     MINUTES,-(SP)
000124  013746  000000G                 MOV     HOURS,-(SP)
000130  012746  000000G                 MOV     #ETIME,-(SP)
000134  012746  000004                  MOV     #4,-(SP)
000140  010600                          MOV     SP,R0                           ; SP,*
000142  104417                          TRAP    17
000144  013716  000000G                 MOV     L$LUN,(SP)                      ;                                       3105
000150  012746  000000G                 MOV     #MSG.07,-(SP)
000154  012746  000002                  MOV     #2,-(SP)
000160  010600                          MOV     SP,R0                           ; SP,*
000162  104417                          TRAP    17
000164  062706  000020                  ADD     #20,SP                          ;                                       3101
000170  000207                  1$:     RTS     PC                              ;                                       3063

; Routine Size:  61 words,      Routine Base:  $CODE$ + 6226
; Maximum stack depth per invocation:  13 words
```

```
;       3114  GLOBAL ROUTINE SEND (INDEX) =
;       3115
;       3116  !+
;       3117  !       IF THE CURRENT RC25 IS ONLINE AND ITS CRING IS NOT FULL, THEN THIS
;       3118  !       ROUTINE "SENDS" A COMMAND TO THE RC25 BY LOADING THE ENVELOPE
;       3119  !       DESCRIPTOR OF AN MSCP ENVELOPE INTO THE COMMAND RING AND READING THE
;       3120  !       DEVICE'S IP REGISTER. A RECORD OF THE SENT COMMAND IS MADE IN THE
;       3121  !       CONTROLLER'S OWN OUTSTANDING COMMAND LIST (OUTC_LIST), AND A COMMAND
;       3122  !       TIMER IS STARTED IN THE OUTSTANDING COMMAND TIMER (OUTC_TIMR). IF THE
;       3123  !       CURRENT RC25 IN NOT ONLINE, THEN A FAILURE INDICATION IS RETURNED TO
;       3124  !       THE CALLER, AND NO ACTION IS TAKEN.
;       3125  !
;       3126  !       INPUTS:
;       3127  !               INDEX - INDEX OF MSCP ENVELOPE CONTAINING THE COMMAND TO
;       3128  !                       BE SENT
;       3129  !
;       3130  !       IMPLICIT INPUTS:
;       3131  !               CCTLR - CURRENT CONTROLLER NUMBER
;       3132  !               DCT_ADDR - ADDRESS OF CURRENT CONTROLLER'S DCT
;       3133  !               OCL_X1, OCL_X2 - STARTING/ENDING INDECES OF CURRENT
;       3134  !                                       CONTROLLER'S OUTSTANDING COMMAND AREA
;       3135  !-
;       3136
;       3137  BEGIN
;       3138
;       3139  LOCAL
;       3140      SLOT_ADDR,
;       3141      TEMP : WORD;
;       3142
;       3143  IF ((.DCT_ADDR [STAT] EQLU ONLINE) AND                  ! IF DEVICE IS ONLINE AND
;       3144      (.DCT_ADDR [CRING_CNT] LSSU CRING_LEN))              ! ITS CRING IS NOT FULL
;       3145  THEN
;       3146      BEGIN
;       3147
;       3148      MSCP_ENV [.INDEX, CRN_LO] = (CRN = .CRN + 1);        ! ASSIGN CMD REF NUM
;       3149      SLOT_ADDR = .DCT_ADDR [CR_NEXT];                     ! ADDR OF NEXT COMMAND SLOT
;       3150      .SLOT_ADDR = .MSCP_ENV [.INDEX, ENV_LO];             ! LOAD ENV DESC (LO) INTO COMMAND SLOT
;       3151      SLOT_ADDR = .SLOT_ADDR + 2;                          ! ADVANCE TO NEXT WORD
;       3152      .SLOT_ADDR = .MSCP_ENV [.INDEX, ENV_HI];             ! LOAD ENV DESC (HI) INTO COMMAND SLOT
;       3153      SLOT_ADDR = .SLOT_ADDR + 2;                          ! ADVANCE TO NEXT COMMAND SLOT
;       3154      IF .SLOT_ADDR GTRA .DCT_ADDR [CR_END]                ! IF BEYOND END OF CRING
;       3155      THEN                                                 ! THEN
;       3156          SLOT_ADDR = .DCT_ADDR [CR_BEG];                  ! CYCLE BACK TO BEGINNING
;       3157      DCT_ADDR [CR_NEXT] = .SLOT_ADDR;                     ! RESTORE CR_NEXT POINTER IN DCT
;       3158      DCT_ADDR [CRING_CNT] = .DCT_ADDR [CRING_CNT] + 1;    ! INCR # OF COMMANDS IN CRING
;       3159
;       3160      INCR COUNT FROM .OCL_X1 TO .OCL_X2 DO                ! FOR EACH OUTC ENTRY
;       3161          BEGIN
;       3162
;       3163          IF .OUTC_LIST [.COUNT] LSS 0                     ! IF A SPARE SLOT IS FOUND
;       3164          THEN
;       3165              BEGIN
;       3166
;       3167              OUTC_LIST [.COUNT] = .INDEX;                 ! LOAD MSCP_ENV INDEX
;       3168              IF .IIP_FLAG                                 ! IF INIT SUBTEST IN PROGRESS
;       3169              THEN                                         ! THEN
;       3170                  OUTC_TIMR [.COUNT] = TO_INIT              ! SET INIT SUBTEST COMMAND TIMER
```

```
;        3171                ELSE                                        ! OTHERWISE
;        3172                    BEGIN
;        3173
;        3174                    IF BIT_TST (SWP_FLAGS, SWF_DM)          ! IF DM EXERCISER IS BEING RUN
;        3175                    THEN                                    ! THEN
;        3176                        OUTC_TIMR [.COUNT] = TO_DUP         ! SET DUP COMMAND TIMER
;        3177                    ELSE                                    ! OTHERWISE
;        3178                        OUTC_TIMR [.COUNT] = TO_IO;         ! SET I/O TRANSFER COMMAND TIMER
;        3179
;        3180                    END;
;        3181
;        3182                EXITLOOP;
;        3183
;        3184                END;                                    ! IF A SPARE SLOT IS FOUND
;        3185
;        3186            END;                                        ! OUTC_LIST SEARCH LOOP
;        3187
;        3188        TEMP = .RC25_ADDR [RCIP, RC_ALL];               ! READ IP TO FORCE PORT TO POLL
;        3189        RETURN SUCCESS;
;        3190
;        3191        END
;        3192    ELSE                                                ! IF DEVICE IS NOT ONLINE
;        3193        RETURN FAILURE;
;        3194
;        3195    END;                                                ! ROUTINE SEND


                                    .SBTTL  SEND GLOBAL ROUTINES
000000  004137  000000G       SEND::  JSR     R1,$SAVE2               ;                               3114
000004  005746                  TST     -(SP)
000006  013701  000000G               MOV     DCT.ADDR,R1             ;                               3143
000012  005711                  TST     (R1)
000014  100110                  BPL     7$
000016  121127  000010                  CMPB    (R1),#10                ;                               3144
000022  103105                  BHIS    7$
000024  016646  000012                  MOV     12(SP),-(SP)            ; INDEX,*                       3148
000030  012746  000104                  MOV     #104,-(SP)
000034  004737  000000G               JSR     PC,BL$MUL
000040  013702  000000G               MOV     CRN,R2
000044  005202                  INC     R2
000046  010237  000000G               MOV     R2,CRN
000052  010260  000010G               MOV     R2,MSCP.ENV+10(R0)
000056  016102  000020                  MOV     20(R1),R2               ; *,SLOT.ADDR                   3149
000062  016022  000000G               MOV     MSCP.ENV(R0),(R2)+      ; *,SLOT.ADDR                   3150
000066  016022  000002G               MOV     MSCP.ENV+2(R0),(R2)+    ; *,SLOT.ADDR                   3152
000072  020261  000012                  CMP     R2,12(R1)               ; SLOT.ADDR,*                   3154
000076  101402                  BLOS    1$
000100  016102  000010                  MOV     10(R1),R2               ; *,SLOT.ADDR                   3156
000104  010261  000020          1$:     MOV     R2,20(R1)               ; SLOT.ADDR,*                   3157
000110  105211                  INCB    (R1)                    ;                               3158
000112  013701  000000G               MOV     OCL.X1,R1               ; *,COUNT                       3160
000116  005301                  DEC     R1                      ; COUNT
000120  000433                  BR      5$
000122  105761  000000G       2$:     TSTB    OUTC.LIST(R1)           ; *(COUNT)                      3163
000126  002030                  BGE     5$
000130  116661  000016  000000G       MOVB    16(SP),OUTC.LIST(R1)    ; INDEX,*(COUNT)                3167
000136  010100                  MOV     R1,R0                   ; COUNT,*                       3170
```

```
000140  006300                          ASL     R0
000142  062700  000000G                 ADD     #OUTC.TIMR,R0
000146  132737  000001  000000G         BITB    #1,IIP.FLAG             ;                               3168
000154  001403                          BEQ     3$
000156  012710  000170                  MOV     #170,(R0)               ;                               3170
000162  000416                          BR      6$                      ;                               3168
000164  132737  000002  000000G 3$:     BITB    #2,SWP.FLAGS            ;                               3174
000172  001403                          BEQ     4$
000174  012710  000036                  MOV     #36,(R0)                ;                               3176
000200  000407                          BR      6$                      ;                               3174
000202  012710  000454          4$:     MOV     #454,(R0)               ;                               3178
000206  000404                          BR      6$                      ;                               3165
000210  005201                  5$:     INC     R1                      ; COUNT                         3160
000212  020137  000000G                 CMP     R1,OCL.X2               ; COUNT,*
000216  003741                          BLE     2$
000220  017766  000000G 000004  6$:     MOV     @RC25.ADDR,4(SP)        ; *,RC.REG                      3188
000226  022626                          CMP     (SP)+,(SP)+             ;                               3146
000230  012700  000001                  MOV     #1,R0                   ;                               3137
000234  000401                          BR      8$
000236  005000                  7$:     CLR     R0
000240  005726                  8$:     TST     (SP)+                   ;                               3114
000242  000207                          RTS     PC

; Routine Size:  82 words,      Routine Base:  $CODE$ + 6420
; Maximum stack depth per invocation:  7 words
```

```
;       3196  GLOBAL ROUTINE WAIT : NOVALUE =
;       3197
;       3198  !+
;       3199  !       THIS ROUTINE IS CALLED DURING THE INITIALIZATION SUBTEST AFTER EACH
;       3200  !       MSCP MESSAGE IS SENT TO A DEVICE. ITS PURPOSE IS TO LOCATE THE LONE
;       3201  !       OUTSTANDING COMMAND AND TO MAINTAIN ITS TIMER UNITL EITHER (A) THE
;       3202  !       DEVICE RESPONDS TO THE COMMAND THROUGH AN RC25 INTERRUPT, RESULTING IN
;       3203  !       A RETURN PACKET BEING DEPOSITED INTO THE I/O DONE QUEUE (IODQ), OR (B)
;       3204  !       THE COMMAND MESSAGE TIMER EXPIRES.
;       3205  !
;       3206  !       IMPLICIT INPUTS:
;       3207  !               CCTLR - CURRENT CONTROLLER NUMBER
;       3208  !               OCL_X1, OCL_X2 - STARTING/ENDING INDECES OF CURRENT
;       3209  !                                       CONTROLLER'S OUTSTANDING COMMAND AREA
;       3210  !-
;       3211
;       3212  BEGIN
;       3213
;       3214  LOCAL
;       3215      CX : SIGNED WORD,                                         ! OUTSTANDING COMMAND'S OUTC_LIST INDEX
;       3216      MX : WORD,                                                ! INDEX OF MSCP COMMAND ENVELOPE
;       3217      RX : WORD,                                                ! RETURN PACKET INDEX
;       3218      M_ADDR,                                                   ! MSCP ENV ADDRESS
;       3219      R_ADDR : REF BLOCK [RP_LEN, WORD] FIELD (RP_FIELDS); ! RETURN PACKET ADDRESS
;       3220
;       3221  CX = -1;                                                    ! ASSUME COMMAND NOT FOUND
;       3222  INCR COUNT FROM .OCL_X1 TO .OCL_X2 DO                       ! SEARCH OUTC_LIST FOR THE
;       3223      BEGIN                                                   !     OUTSTANDING COMMAND
;       3224
;       3225      IF .OUTC_LIST [.COUNT] GEQ 0                            ! IF THIS IS IT
;       3226      THEN                                                    ! THEN
;       3227          BEGIN
;       3228
;       3229          CX = .COUNT;                                        ! SAVE OUTC INDEX OF COMMAND
;       3230          EXITLOOP;                                           ! DONE HERE
;       3231
;       3232          END;
;       3233
;       3234      END;
;       3235
;       3236  T_FLAG = FALSE;                                            ! CLEAR ONE SECOND TIMING FLAG
;       3237  DO                                                          ! REPEAT UNITL RC25 INTERRUPT
;       3238      BEGIN                                                   !     OR TIMEOUT
;       3239
;       3240      IF .T_FLAG                                              ! IF ONE SECOND HAS ELAPSED
;       3241      THEN                                                    ! THEN
;       3242          BEGIN
;       3243
;       3244          IF .CX GEQ 0                                        ! IF OUTSTANDING COMMAND WAS FOUND
;       3245          THEN                                                ! THEN
;       3246              BEGIN
;       3247
;       3248              SETPRI (PRI05);                                 ! LOCK OUT RC25 INTERRUPTS
;       3249              IF .OUTC_TIMR [.CX] GTRU 0                      ! IF TIMER STILL ACTIVE
;       3250              THEN                                            ! THEN
;       3251                  BEGIN
;       3252
```

```
;        3253                OUTC_TIMR [.CX] = .OUTC_TIMR [.CX] - 1; ! DECREMENT TIMER
;        3254                IF .OUTC_TIMR [.CX] EQL 0               ! IF MESSAGE HAS TIMED OUT
;        3255                THEN                                    ! THEN
;        3256                    BEGIN
;        3257
;        3258                    RX = GET_RETPKT (.CCTLR);                   ! GET A RETURN PACKET
;        3259                    MX = .OUTC_LIST [.CX];                      ! GET COMMAND ENV INDEX
;        3260                    R_ADDR = RETPKT + (.RX * RP_LEN * 2);       ! CALCULATE RETPKT ADDRESS
;        3261                    M_ADDR = MSCP_ENV + (.MX * ENV_LEN * 2);    ! CALCULATE ENV ADDR
;        3262                    COPY_BLK (.M_ADDR + 4, .R_ADDR, RP_LEN);    ! COPY COMMAND TO RETPKT
;        3263                    R_ADDR [CONID] = CID_DRIVER;                ! SET PACKET SOURCE AS "DRIVER"
;        3264                    R_ADDR [MESTYP] = MT_TIMEOUT;               ! COMMAND TIMEOUT
;        3265                    R_ADDR [CTLR] = .CCTLR;                     ! CONTROLLER NUMBER
;        3266                    IN_IODQ (.RX);                              ! "SEND" THE PACKET
;        3267                    PUT_ENV (.MX);                              ! RETURN COMMAND ENV TO POOL
;        3268                    OUTC_LIST [.CX] = -1;                       ! INIT OUTC_LIST ENTRY
;        3269
;        3270                    END;                                ! IF MESSAGE TIMER EXPIRED
;        3271
;        3272                END;                                    ! IF TIMER STILL ACTIVE
;        3273
;        3274            SETPRI (PRI00);                             ! RESTORE INIT SUBTEST PRIORITY
;        3275
;        3276            END;                                        ! IF OUTSTANDING COMMAND WAS FOUND
;        3277
;        3278        T_FLAG = FALSE;                                 ! CLEAR ONE SECOND TIMING FLAG
;        3279
;        3280        END;                                            ! IF ONE SECOND HAS ELAPSED
;        3281
;        3282            END                                         ! UNTIL A PACKET IS "SENT" FROM HERE
;        3283    UNTIL .IODQ_IN NEQU .IODQ_OUT;                      !     OR RC25 INTERRUPT PROCESSING
;        3284
;        3285    END;                                                ! ROUTINE WAIT


                                        .SBTTL  WAIT GLOBAL ROUTINES
000000  004137  000000G           WAIT::  JSR     R1,$SAVE5                       ;                       3196
000004  012701  177777                   MOV     #-1,R1                          ; *,CX                  3221
000010  013700  000000G                  MOV     OCL.X1,R0                       ; *,COUNT               3222
000014  060100                           ADD     R1,R0                           ; *,COUNT
000016  000405                           BR      2$
000020  105760  000000G           1$:     TSTB    OUTC.LIST(R0)                   ; *(COUNT)              3225
000024  002402                           BLT     2$
000026  010001                           MOV     R0,R1                           ; COUNT,CX              3229
000030  000404                           BR      3$                              ;                       3227
000032  005200                   2$:     INC     R0                              ; COUNT                 3222
000034  020037  000000G                  CMP     R0,OCL.X2                       ; COUNT,*
000040  003767                           BLE     1$
000042  105037  000000G           3$:     CLRB    T.FLAG                          ;                       3236
000046  132737  000001  000000G   4$:     BITB    #1,T.FLAG                       ;                       3240
000054  001510                           BEQ     7$
000056  005701                           TST     R1                              ; CX                    3244
000060  002504                           BLT     6$
000062  012700  000240                   MOV     #240,R0                         ;                       3248
000066  104441                           TRAP    41
000070  010100                           MOV     R1,R0                           ; CX,*                  3249
000072  006300                           ASL     R0
```

```
000074  062700  000000G                        ADD     #OUTC.TIMR,R0
000100  005710                                 TST     (R0)
000102  001471                                 BEQ     5$
000104  005310                                 DEC     (R0)                    ;                               3253
000106  001067                                 BNE     5$                      ;                               3254
000110  013746  000000G                        MOV     CCTLR,-(SP)             ;                               3258
000114  004737  005070'                        JSR     PC,GET.RETPKT
000120  010005                                 MOV     R0,R5                   ; *,RX
000122  116104  000000G                        MOVB    OUTC.LIST(R1),R4        ; *(CX),MX                      3259
000126  010516                                 MOV     R5,(SP)                 ; RX,*                          3260
000130  012746  000060                        MOV     #60,-(SP)
000134  004737  000000G                        JSR     PC,BL$MUL
000140  062700  000000G                        ADD     #RETPKT,R0
000144  010002                                 MOV     R0,R2                   ; *,R.ADDR
000146  010416                                 MOV     R4,(SP)                 ; MX,*                          3261
000150  012746  000104                        MOV     #104,-(SP)
000154  004737  000000G                        JSR     PC,BL$MUL
000160  062700  000000G                        ADD     #MSCP.ENV,R0
000164  010003                                 MOV     R0,R3                   ; *,M.ADDR
000166  010316                                 MOV     R3,(SP)                 ; M.ADDR,*                      3262
000170  062716  000004                        ADD     #4,(SP)
000174  010246                                 MOV     R2,-(SP)                ; R.ADDR,*
000176  012746  000030                        MOV     #30,-(SP)
000202  004737  004352'                        JSR     PC,COPY.BLK
000206  012700  000002                        MOV     #2,R0                   ;                               3263
000212  060200                                 ADD     R2,R0                   ; R.ADDR,*
000214  112760  000003  000001                MOVB    #3,1(R0)
000222  013746  000000G                        MOV     CCTLR,-(SP)             ;                               3265
000226  042716  177760                        BIC     #177760,(SP)
000232  112710  000100                        MOVB    #100,(R0)
000236  152610                                 BISB    (SP)+,(R0)
000240  010516                                 MOV     R5,(SP)                 ; RX,*                          3266
000242  004737  005466'                        JSR     PC,IN.IODQ
000246  010416                                 MOV     R4,(SP)                 ; MX,*                          3267
000250  004737  005014'                        JSR     PC,PUT.ENV
000254  112761  000377  000000G               MOVB    #377,OUTC.LIST(R1)      ; *,*(CX)                       3268
000262  062706  000012                        ADD     #12,SP                  ;                               3256
000266  005000                         5$:    CLR     R0                      ;                               3274
000270  104441                                 TRAP    41
000272  105037  000000G                 6$:    CLRB    T.FLAG                  ;                               3278
000276  023737  000000G 000000G        7$:    CMP     IODQ.IN,IODQ.OUT        ;                               3283
000304  001660                                 BEQ     4$
000306  000207                                 RTS     PC                      ;                               3196

; Routine Size:  100 words,    Routine Base:  $CODE$ + 6664
; Maximum stack depth per invocation:  12 words
```

```
;          3286  %SBTTL 'ERROR MESSAGE SUBROUTINES'
;          3287
;          3288  ROUTINE EMS_ET : NOVALUE =
;          3289
;          3290  !+
;          3291  !       THIS ROUTINE PRINTS THE ELAPSED TIME THAT PRECEDES ALL BASIC ERROR
;          3292  !       MESSAGES.
;          3293  !-
;          3294
;          3295  PRINTB (ETIME, .HOURS, .MINUTES, .SECONDS);


                                        .SBTTL  EMS.ET ERROR MESSAGE SUBROUTINES
000000  013746  000000G          EMS.ET: MOV     SECONDS,-(SP)                   ;                       3295
000004  013746  000000G                  MOV     MINUTES,-(SP)
000010  013746  000000G                  MOV     HOURS,-(SP)
000014  012746  000000G                  MOV     #ETIME,-(SP)
000020  012746  000004                   MOV     #4,-(SP)
000024  010600                           MOV     SP,R0                           ; SP,*
000026  104414                           TRAP    14
000030  062706  000012                   ADD     #12,SP
000034  000207                           RTS     PC                              ;                       3288

; Routine Size:  15 words,      Routine Base:  $CODE$ + 7174
; Maximum stack depth per invocation:  7 words


;          3296  ROUTINE EMS_SA : NOVALUE =
;          3297
;          3298  !+
;          3299  !       THIS ROUTINE PRINTS (EXTENDED) THE GLOBAL DATUM "SA_REG" WHICH CONTAINS
;          3300  !       THE CONTENTS OF THE SA REGISTER.
;          3301  !-
;          3302
;          3303  PRINTX (EX_SA, .SA_REG);                        ! ''    SA: XXXXXX(O)''


                                        .SBTTL  EMS.SA ERROR MESSAGE SUBROUTINES
000000  013746  000000G          EMS.SA: MOV     SA.REG,-(SP)                    ;                       3303
000004  012746  000000G                  MOV     #EX.SA,-(SP)
000010  012746  000002                   MOV     #2,-(SP)
000014  010600                           MOV     SP,R0                           ; SP,*
000016  104415                           TRAP    15
000020  062706  000006                   ADD     #6,SP
000024  000207                           RTS     PC                              ;                       3296

; Routine Size:  11 words,      Routine Base:  $CODE$ + 7232
; Maximum stack depth per invocation:  5 words


;          3304  ROUTINE EMS_CRN : NOVALUE =
;          3305
;          3306  !+
;          3307  !       THIS ROUTINE PRINTS (EXTENDED) THE COMMAND REFERENCE NUMBER OF THE
;          3308  !       CURRENT RETURN PACKET. ONLY THE LOW-ORDER WORD IS SIGNIFICANT.
;          3309  !-
;          3310
```

```
;         3311  PRINTX (EX_CRN, .RP_ADDR [CRF_LO]);              ! ''    CMD REF NUM: XXXXXX(O)''


                                        .SBTTL  EMS.CRN ERROR MESSAGE SUBROUTINES
000000  013700  000000G           EMS.CRN:MOV     RP.ADDR,R0                      ;                       3311
000004  016046  000004                   MOV     4(R0),-(SP)
000010  012746  000000G                   MOV     #EX.CRN,-(SP)
000014  012746  000002                   MOV     #2,-(SP)
000020  010600                           MOV     SP,R0                           ; SP,*
000022  104415                           TRAP    15
000024  062706  000006                   ADD     #6,SP
000030  000207                           RTS     PC                              ;                       3304

; Routine Size:  13 words,      Routine Base:  $CODE$ + 7260
; Maximum stack depth per invocation:  5 words


;         3312  ROUTINE EMS_STC : NOVALUE =
;         3313
;         3314  !+
;         3315  !       THIS ROUTINE PRINTS (EXTENDED) THE GLOBAL DATUM ''ST_CODE'' (STATUS CODE)
;         3316  !       IF IT IS NON-ZERO.
;         3317  !-
;         3318
;         3319  IF .ST_CODE NEQU 0
;         3320  THEN
;         3321      PRINTX (EX_SC, .ST_CODE);                   ! ''    STATUS CODE: XX(O)''


                                        .SBTTL  EMS.STC ERROR MESSAGE SUBROUTINES
000000  013700  000000G           EMS.STC:MOV     ST.CODE,R0                      ;                       3319
000004  001411                           BEQ     1$
000006  010046                           MOV     R0,-(SP)                        ;                       3321
000010  012746  000000G                   MOV     #EX.SC,-(SP)
000014  012746  000002                   MOV     #2,-(SP)
000020  010600                           MOV     SP,R0                           ; SP,*
000022  104415                           TRAP    15
000024  062706  000006                   ADD     #6,SP
000030  000207                   1$:     RTS     PC                              ;                       3312

; Routine Size:  13 words,      Routine Base:  $CODE$ + 7312
; Maximum stack depth per invocation:  5 words


;         3322  ROUTINE EMS_SBC : NOVALUE =
;         3323
;         3324  !+
;         3325  !       THIS ROUTINE PRINTS (EXTENDED) THE GLOBAL DATUM ''SB_CODE'' (SUB-CODE) IF
;         3326  !       EITHER THE STATUS CODE (ST_CODE) OR THE SUB-CODE IS NON-ZERO. (A
;         3327  !       NON-ZERO SUB-CODE ALWAYS HAS SIGNIFICANCE, WHEREAS A ZERO SUB-CODE ONLY
;         3328  !       HAS MEANING WITH A NON-ZERO STATUS CODE).
;         3329  !-
;         3330
;         3331  IF (.ST_CODE OR .SB_CODE) NEQU 0
;         3332  THEN
;         3333      PRINTX (EX_SB, .SB_CODE);                   ! ''    SUB-CODE: XXXX(O)''
```

```
                                        .SBTTL  EMS.SBC ERROR MESSAGE SUBROUTINES
000000  013700  000000G          EMS.SBC:MOV     ST.CODE,R0                      ;                       3331
000004  053700  000000G                  BIS     SB.CODE,R0
000010  001412                          BEQ     1$
000012  013746  000000G                  MOV     SB.CODE,-(SP)                   ;                       3333
000016  012746  000000G                  MOV     #EX.SB,-(SP)
000022  012746  000002                   MOV     #2,-(SP)
000026  010600                          MOV     SP,R0                           ; SP,*
000030  104415                          TRAP    15
000032  062706  000006                   ADD     #6,SP
000036  000207                  1$:     RTS     PC                              ;                       3322

; Routine Size:  16 words,      Routine Base:  $CODE$ + 7344
; Maximum stack depth per invocation:  5 words


;          3334  ROUTINE EMS_CMD : NOVALUE =
;          3335
;          3336  !+
;          3337  !       THIS ROUTINE PRINTS (EXTENDED) THE OPCODE AND COMMAND MODIFIER (IF
;          3338  !       PRESENT) OF THE CURRENT RETURN PACKET. THESE FIELDS ARE "TRANSLATED"
;          3339  !       INTO ENGLISH TEXT RATHER THAN PRINTED AS RAW NUMBERS.
;          3340  !
;          3341  !       IMPLICIT INPUTS:
;          3342  !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;          3343  !-
;          3344
;          3345  BEGIN
;          3346
;          3347  LOCAL
;          3348      COMMAND : WORD;                                   ! COMMAND OPCODE
;          3349
;          3350  PRINTX (EX_CMD);                                     ! "    COMMAND: "
;          3351  COMMAND = .RP_ADDR [ENDCOD] AND OP_MSK;              ! GET OPCODE ALONE
;          3352  SELECTONEU .COMMAND OF
;          3353      SET
;          3354
;          3355      [OP_ESP]    : PRINTX (EX_ESP);                    ! "EXECUTE SUPPLIED PROGRAM"
;          3356      [OP_SCC]    : IF .IIP_FLAG                        ! IF INIT SUBTEST IN PROGRESS
;          3357                    THEN                                ! THEN (THESE 2 OPCODES ARE THE SAME)
;          3358                        PRINTX (EX_SCC)                 ! "SET CTLR CHAR"
;          3359                    ELSE                                ! OTHERWISE
;          3360                        PRINTX (EX_SND);                ! "SEND DATA"
;          3361      [OP_ONL]    : PRINTX (EX_ONL);                    ! "ONLINE"
;          3362      [OP_RD]     : PRINTX (EX_RD);                     ! "READ"
;          3363      [OP_WRT]    : PRINTX (EX_WRT);                    ! "WRITE"
;          3364      [OTHERWISE] : PRINTX (EX_03, .RP_ADDR [ENDCOD]); ! "XXX(O)"
;          3365
;          3366      TES;
;          3367
;          3368  IF (((.COMMAND EQLU OP_RD) OR (.COMMAND EQLU OP_WRT)) AND
;          3369       (BIT_TST (RP_ADDR [CMDMOD], MD_CMP)))            ! IF COMPARE MODIFIER IS PRESENT
;          3370  THEN                                                  ! THEN
;          3371      PRINTX (EX_CMP)                                   ! "-COMPARE<CR><LF>"
;          3372  ELSE                                                  ! OTHERWISE
;          3373      PRINTX (CRLF);                                    ! <CR><LF>
```

```
;        3374
;        3375  END;                                    ! ROUTINE EMS_CMD


                                       .SBTTL  EMS.CMD ERROR MESSAGE SUBROUTINES
000000  004137  000000G          EMS.CMD:JSR    R1,$SAVE2                        ;                              3334
000004  012746  000000G                  MOV    #EX.CMD,-(SP)                    ;                              3350
000010  012746  000001                   MOV    #1,-(SP)
000014  010600                           MOV    SP,R0                            ; SP,*
000016  104415                           TRAP   15
000020  013701  000000G                  MOV    RP.ADDR,R1                       ;                              3351
000024  116102  000014                   MOVB   14(R1),R2                        ; *,COMMAND
000030  042702  177600                   BIC    #177600,R2                       ; *,COMMAND
000034  020227  000002                   CMP    R2,#2                            ; COMMAND,*                    3352
000040  001007                           BNE    1$
000042  012716  000000G                  MOV    #EX.ESP,(SP)                     ;                              3355
000046  012746  000001                   MOV    #1,-(SP)
000052  010600                           MOV    SP,R0                            ; SP,*
000054  104415                           TRAP   15
000056  000475                           BR     7$                               ;                              3352
000060  020227  000004          1$:      CMP    R2,#4                            ; COMMAND,*
000064  001022                           BNE    3$
000066  132737  000001  000000G          BITB   #1,IIP.FLAG                      ;                              3356
000074  001407                           BEQ    2$
000076  012716  000000G                  MOV    #EX.SCC,(SP)                     ;                              3358
000102  012746  000001                   MOV    #1,-(SP)
000106  010600                           MOV    SP,R0                            ; SP,*
000110  104415                           TRAP   15
000112  000457                           BR     7$                               ;                              3356
000114  012716  000000G          2$:      MOV    #EX.SND,(SP)                     ;                              3360
000120  012746  000001                   MOV    #1,-(SP)
000124  010600                           MOV    SP,R0                            ; SP,*
000126  104415                           TRAP   15
000130  000450                           BR     7$                               ;                              3352
000132  020227  000011          3$:      CMP    R2,#11                           ; COMMAND,*
000136  001007                           BNE    4$
000140  012716  000000G                  MOV    #EX.ONL,(SP)                     ;                              3361
000144  012746  000001                   MOV    #1,-(SP)
000150  010600                           MOV    SP,R0                            ; SP,*
000152  104415                           TRAP   15
000154  000436                           BR     7$                               ;                              3352
000156  020227  000041          4$:      CMP    R2,#41                           ; COMMAND,*
000162  001007                           BNE    5$
000164  012716  000000G                  MOV    #EX.RD,(SP)                      ;                              3362
000170  012746  000001                   MOV    #1,-(SP)
000174  010600                           MOV    SP,R0                            ; SP,*
000176  104415                           TRAP   15
000200  000424                           BR     7$                               ;                              3352
000202  020227  000042          5$:      CMP    R2,#42                           ; COMMAND,*
000206  001007                           BNE    6$
000210  012716  000000G                  MOV    #EX.WRT,(SP)                     ;                              3363
000214  012746  000001                   MOV    #1,-(SP)
000220  010600                           MOV    SP,R0                            ; SP,*
000222  104415                           TRAP   15
000224  000412                           BR     7$                               ;                              3352
000226  005016                  6$:      CLR    (SP)                             ;                              3364
000230  116116  000014                   MOVB   14(R1),(SP)
```

```
000234  012746  000000G                        MOV     #EX.03,-(SP)
000240  012746  000002                         MOV     #2,-(SP)
000244  010600                                 MOV     SP,R0           ; SP,*
000246  104415                                 TRAP    15
000250  005726                                 TST     (SP)+
000252  020227  000041              7$:        CMP     R2,#41          ; COMMAND,*                     3368
000256  001403                                 BEQ     8$
000260  020227  000042                         CMP     R2,#42          ; COMMAND,*
000264  001015                                 BNE     9$
000266  013700  000000G             8$:        MOV     RP.ADDR,R0      ;                               3369
000272  032760  040000  000012                 BIT     #40000,12(R0)
000300  001407                                 BEQ     9$
000302  012716  000000G                        MOV     #EX.CMP,(SP)    ;                               3371
000306  012746  000001                         MOV     #1,-(SP)
000312  010600                                 MOV     SP,R0           ; SP,*
000314  104415                                 TRAP    15
000316  000406                                 BR      10$             ;                               3368
000320  012716  000000G             9$:        MOV     #CRLF,(SP)      ;                               3373
000324  012746  000001                         MOV     #1,-(SP)
000330  010600                                 MOV     SP,R0           ; SP,*
000332  104415                                 TRAP    15
000334  062706  000010              10$:       ADD     #10,SP          ;                               3345
000340  000207                                 RTS     PC              ;                               3334

; Routine Size:  113 words,    Routine Base:  $CODE$ + 7404
; Maximum stack depth per invocation:  9 words


;          3376  ROUTINE EMS_LBN : NOVALUE =
;          3377
;          3378  !+
;          3379  !       THIS ROUTINE PRINTS (EXTENDED) ONE OF TWO BLOCK NUMBERS APPEARING IN
;          3380  !       THE CURRENT RETURN PACKET. NORMALLY, THE LBN FIELD IS PRINTED; THIS
;          3381  !       FIELD WAS COPIED INTO THE RETURN PACKET FROM THE ASSOCIATED COMMAND
;          3382  !       ENVELOPE. HOWEVER, IF THE "FLAGS" FIELD OF THE CURRENT RETURN PACKET
;          3383  !       INDICATES "BAD BLOCK REPORTED", THEN THE "FIRST BAD BLOCK" FIELD IS
;          3384  !       PRINTED.
;          3385  !
;          3386  !       IMPLICIT INPUTS:
;          3387  !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;          3388  !-
;          3389
;          3390  BEGIN
;          3391
;          3392  IF BIT_TST (RP_ADDR [FLAGS], EF_BBR)          ! IF BAD BLOCK REPORTED
;          3393  THEN                                          ! THEN
;          3394      PRINTX (EX_BB, .RP_ADDR [BBLK_LO])        ! "     BAD BLOCK REPORTED: XXXXX."
;          3395  ELSE                                          ! OTHERWISE
;          3396      PRINTX (EX_LBN, .RP_ADDR [LBN_LO]);       ! "     LBN: XXXXX."
;          3397
;          3398  END;                                          ! ROUTINE EMS_LBN


                                       .SBTTL  EMS.LBN ERROR MESSAGE SUBROUTINES
000000  013700  000000G         EMS.LBN:MOV     RP.ADDR,R0              ;                               3392
000004  005760  000014                   TST     14(R0)
000010  100011                         BPL     1$
```

```
000012  016046  000040                          MOV     40(R0),-(SP)                    ;                               3394
000016  012746  000000G                         MOV     #EX.BB,-(SP)
000022  012746  000002                          MOV     #2,-(SP)
000026  010600                                  MOV     SP,R0                           ; SP,*
000030  104415                                  TRAP    15
000032  000410                                  BR      2$                              ;                               3392
000034  016046  000050                  1$:     MOV     50(R0),-(SP)                    ;                               3396
000040  012746  000000G                         MOV     #EX.LBN,-(SP)
000044  012746  000002                          MOV     #2,-(SP)
000050  010600                                  MOV     SP,R0                           ; SP,*
000052  104415                                  TRAP    15
000054  062706  000006                  2$:     ADD     #6,SP                           ;                               3390
000060  000207                                  RTS     PC                              ;                               3376

; Routine Size:  25 words,      Routine Base:  $CODE$ + 7746
; Maximum stack depth per invocation:  5 words


;           3399  ROUTINE EMS_BC : NOVALUE =
;           3400
;           3401  !+
;           3402  !       THIS ROUTINE PRINTS (EXTENDED) BOTH BYTE COUNT FIELDS OF THE CURRENT
;           3403  !       RETURN PACKET: THE BYTE COUNT FROM THE COMMAND ENVELOPE AND THE
;           3404  !       ACTUAL NUMBER OF BYTES TRANSFERRED (FROM THE RESPONSE ENVELOPE).
;           3405  !
;           3406  !       IMPLICIT INPUTS:
;           3407  !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;           3408  !-
;           3409
;           3410  BEGIN
;           3411
;           3412  PRINTX (EX_CBC, .RP_ADDR [CBCNT_LO]);          ! ''    BYTE COUNT IN COMMAND: XXXXX.''
;           3413  PRINTX (EX_BC, .RP_ADDR [BCNT_LO]);            ! ''    ACTUAL # OF BYTES TRANSFERRED: XXXXX.''
;           3414
;           3415  END;                                            ! ROUTINE EMS_BC


                                        .SBTTL  EMS.BC ERROR MESSAGE SUBROUTINES
000000  013700  000000G         EMS.BC: MOV     RP.ADDR,R0                      ;                               3412
000004  016046  000044                  MOV     44(R0),-(SP)
000010  012746  000000G                 MOV     #EX.CBC,-(SP)
000014  012746  000002                  MOV     #2,-(SP)
000020  010600                          MOV     SP,R0                           ; SP,*
000022  104415                          TRAP    15
000024  013700  000000G                 MOV     RP.ADDR,R0                      ;                               3413
000030  016016  000020                  MOV     20(R0),(SP)
000034  012746  000000G                 MOV     #EX.BC,-(SP)
000040  012746  000002                  MOV     #2,-(SP)
000044  010600                          MOV     SP,R0                           ; SP,*
000046  104415                          TRAP    15
000050  062706  000012                  ADD     #12,SP                          ;                               3410
000054  000207                          RTS     PC                              ;                               3399

; Routine Size:  23 words,      Routine Base:  $CODE$ + 10030
; Maximum stack depth per invocation:  7 words

;           3416  ROUTINE EMS_BD : NOVALUE =
```

```
;          3417
;          3418  !+
;          3419  !       THIS ROUTINE PRINTS (EXTENDED) THE TWO-WORD I/O BUFFER DESCRIPTOR
;          3420  !       APPEARING IN THE CURRENT RETURN PACKET.
;          3421  !
;          3422  !       IMPLICIT INPUTS:
;          3423  !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;          3424  !-
;          3425
;          3426  PRINTX (EX_BD, .RP_ADDR [BUFF_0], .RP_ADDR [BUFF_1]);   ! ''    I/O BUFFER DESCRIPTOR: XXXXXX(0) XXXXXX(0)''


                                        .SBTTL  EMS.BD ERROR MESSAGE SUBROUTINES
000000  013700  000000G          EMS.BD: MOV     RP.ADDR,R0                      ;                                       3426
000004  016046  000026                  MOV     26(R0),-(SP)
000010  016046  000024                  MOV     24(R0),-(SP)
000014  012746  000000G                  MOV     #EX.BD,-(SP)
000020  012746  000003                  MOV     #3,-(SP)
000024  010600                          MOV     SP,R0                           ; SP,*
000026  104415                          TRAP    15
000030  062706  000010                  ADD     #10,SP
000034  000207                          RTS     PC                              ;                                       3416

; Routine Size:  15 words,      Routine Base:  $CODE$ + 10106
; Maximum stack depth per invocation:  6 words


;          3427  ROUTINE EMS_RP : NOVALUE =
;          3428
;          3429  !+
;          3430  !       THIS ROUTINE IS RESPONSIBLE FOR PRINTING (EXTENDED) THE RELEVANT FIELDS
;          3431  !       OF THE CURRENT RETURN PACKET.
;          3432  !-
;          3433
;          3434  BEGIN
;          3435
;          3436  EMS_CRN ();                                   ! COMMAND REFERENCE NUMBER
;          3437  EMS_SBC ();                                   ! SUB-CODE
;          3438  EMS_CMD ();                                   ! COMMAND (AND MODIFIER)
;          3439  EMS_LBN ();                                   ! LBN OR BAD BLOCK NUMBER
;          3440  EMS_BC ();                                    ! BYTE COUNTS
;          3441  EMS_BD ();                                    ! I/O BUFFER DESCRIPTOR
;          3442
;          3443  END;                                         ! ROUTINE EMS_RP


                                        .SBTTL  EMS.RP ERROR MESSAGE SUBROUTINES
000000  004737  007260'          EMS.RP: JSR     PC,EMS.CRN                      ;                                       3436
000004  004737  007344'                  JSR     PC,EMS.SBC                      ;                                       3437
000010  004737  007404'                  JSR     PC,EMS.CMD                      ;                                       3438
000014  004737  007746'                  JSR     PC,EMS.LBN                      ;                                       3439
000020  004737  010030'                  JSR     PC,EMS.BC                       ;                                       3440
000024  004737  010106'                  JSR     PC,EMS.BD                       ;                                       3441
000030  000207                          RTS     PC                              ;                                       3427

; Routine Size:  13 words,      Routine Base:  $CODE$ + 10144
; Maximum stack depth per invocation:  1 word
```

```
;         3444  BGNMSG (EMS_01);


                                        .SBTTL  EMS.01 ERROR MESSAGE SUBROUTINES
000000  004737  000000V           EMS.01::JSR     PC,M$EMS.01                     ;                                       3444
000004  104423                            TRAP    23
000006  000207                            RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 10176
; Maximum stack depth per invocation:  2 words


;         3445
;         3446  EMS_ET ();                                      ! ELAPSED TIME
;         3447  PRINTB (EBS_01, MAX_UNITS);                     ! 'MORE THAN XX. UNITS SPECIFIED''
;         3448
;         3449  ENDMSG;


                                        .SBTTL  M$EMS.01 ERROR MESSAGE SUBROUTINES
000000  004737  007174'           M$EMS.01:
                                        JSR     PC,EMS.ET                       ;                                       3446
000004  012746  000020                    MOV     #20,-(SP)                       ;                                       3447
000010  012746  000000G                   MOV     #EBS.01,-(SP)
000014  012746  000002                    MOV     #2,-(SP)
000020  010600                            MOV     SP,R0                           ; SP,*
000022  104414                            TRAP    14
000024  062706  000006                    ADD     #6,SP                           ;                                       3444
000030  000207                            RTS     PC

; Routine Size:  13 words,     Routine Base:  $CODE$ + 10206
; Maximum stack depth per invocation:  5 words


;         3450  BGNMSG (EMS_10);


                                        .SBTTL  EMS.10 ERROR MESSAGE SUBROUTINES
000000  004737  000000V           EMS.10::JSR     PC,M$EMS.10                     ;                                       3450
000004  104423                            TRAP    23
000006  000207                            RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 10240
; Maximum stack depth per invocation:  2 words


;         3451
;         3452  EMS_ET ();                                      ! ELAPSED TIME
;         3453  PRINTB (EBD_10, .RC25_ADDR + .OF_RC);           ! 'NO RESPONSE AT ADDRESS XXXXXX(O)''
;         3454
;         3455  ENDMSG;


                                        .SBTTL  M$EMS.10 ERROR MESSAGE SUBROUTINES
000000  004737  007174'           M$EMS.10:
                                        JSR     PC,EMS.ET                       ;                                       3452
000004  013746  000000G                   MOV     RC25.ADDR,-(SP)                 ;                                       3453
```

```
000010  063716  000000G                        ADD     OF.RC,(SP)
000014  012746  000000G                        MOV     #EBD.10,-(SP)
000020  012746  000002                         MOV     #2,-(SP)
000024  010600                                 MOV     SP,R0                           ; SP,*
000026  104414                                 TRAP    14
000030  062706  000006                         ADD     #6,SP                           ;                                       3450
000034  000207                                 RTS     PC

; Routine Size:  15 words,      Routine Base:  $CODE$ + 10250
; Maximum stack depth per invocation:  5 words


;         3456  BGNMSG (EMS_12);


                                               .SBTTL  EMS.12 ERROR MESSAGE SUBROUTINES
000000  004737  000000V                EMS.12::JSR     PC,M$EMS.12                     ;                                       3456
000004  104423                                 TRAP    23
000006  000207                                 RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 10306
; Maximum stack depth per invocation:  2 words


;         3457
;         3458  EMS_ET ();                                      ! ELAPSED TIME
;         3459  PRINTB (EBD_12, .RC25_ADDR);                    ! "INCORRECT BR LEVEL GIVEN FOR DEVICE XXXXXX(O)"
;         3460
;         3461  ENDMSG;


                                               .SBTTL  M$EMS.12 ERROR MESSAGE SUBROUTINES
000000  004737  007174'                M$EMS.12:
                                               JSR     PC,EMS.ET                       ;                                       3458
000004  013746  000000G                        MOV     RC25.ADDR,-(SP)                 ;                                       3459
000010  012746  000000G                        MOV     #EBD.12,-(SP)
000014  012746  000002                         MOV     #2,-(SP)
000020  010600                                 MOV     SP,R0                           ; SP,*
000022  104414                                 TRAP    14
000024  062706  000006                         ADD     #6,SP                           ;                                       3456
000030  000207                                 RTS     PC

; Routine Size:  13 words,      Routine Base:  $CODE$ + 10316
; Maximum stack depth per invocation:  5 words


;         3462  BGNMSG (EMS_13);


                                               .SBTTL  EMS.13 ERROR MESSAGE SUBROUTINES
000000  004737  000000V                EMS.13::JSR     PC,M$EMS.13                     ;                                       3462
000004  104423                                 TRAP    23
000006  000207                                 RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 10350
; Maximum stack depth per invocation:  2 words
;         3463
```

```
;         3464  EMS_ET ();                                                    ! ELAPSED TIME
;         3465  PRINTB (EBD_13, .STEP, .RC25_ADDR);                          ! "STEP X READ ERROR ON DEVICE XXXXXX(O)"
;         3466  EMS_SA ();                                                    ! PRINTX SA CONTENTS
;         3467
;         3468  ENDMSG;


                                        .SBTTL  M$EMS.13 ERROR MESSAGE SUBROUTINES
000000  004737  007174'          M$EMS.13:
                                        JSR     PC,EMS.ET                       ;                                                   3464
000004  013746  000000G                  MOV     RC25.ADDR,-(SP)                 ;                                                   3465
000010  013746  000000G                  MOV     STEP,-(SP)
000014  012746  000000G                  MOV     #EBD.13,-(SP)
000020  012746  000003                   MOV     #3,-(SP)
000024  010600                           MOV     SP,R0                           ; SP,*
000026  104414                           TRAP    14
000030  004737  007232'                  JSR     PC,EMS.SA                       ;                                                   3466
000034  062706  000010                   ADD     #10,SP                          ;                                                   3462
000040  000207                           RTS     PC

; Routine Size:  17 words,      Routine Base:  $CODE$ + 10360
; Maximum stack depth per invocation:  6 words

;         3469  BGNMSG (EMS_14);

                                        .SBTTL  EMS.14 ERROR MESSAGE SUBROUTINES
000000  004737  000000V          EMS.14::JSR     PC,M$EMS.14                     ;                                                   3469
000004  104423                           TRAP    23
000006  000207                           RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 10422
; Maximum stack depth per invocation:  2 words


;         3470
;         3471  EMS_ET ();                                                    ! ELAPSED TIME
;         3472  PRINTB (EBD_14, .IRC25_ADDR);                                ! "ERROR CODE RECEIVED IN SA REGISTER OF DEVICE XXXXXX(O)"
;         3473  EMS_SA ();                                                    ! PRINTX SA REGISTER CONTENTS
;         3474
;         3475  ENDMSG;


                                        .SBTTL  M$EMS.14 ERROR MESSAGE SUBROUTINES
000000  004737  007174'          M$EMS.14:
                                        JSR     PC,EMS.ET                       ;                                                   3471
000004  013746  000000G                  MOV     IRC25.ADDR,-(SP)                ;                                                   3472
000010  012746  000000G                  MOV     #EBD.14,-(SP)
000014  012746  000002                   MOV     #2,-(SP)
000020  010600                           MOV     SP,R0                           ; SP,*
000022  104414                           TRAP    14
000024  004737  007232'                  JSR     PC,EMS.SA                       ;                                                   3473
000030  062706  000006                   ADD     #6,SP                           ;                                                   3469
000034  000207                           RTS     PC

; Routine Size:  15 words,      Routine Base:  $CODE$ + 10432
```

```
; Maximum stack depth per invocation:  5 words


;         3476  BGNMSG (EMS_15);


                                        .SBTTL  EMS.15 ERROR MESSAGE SUBROUTINES
000000  004737  000000V          EMS.15::JSR     PC,M$EMS.15                     ;                                       3476
000004  104423                           TRAP    23
000006  000207                           RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 10470
; Maximum stack depth per invocation:  2 words


;         3477
;         3478  EMS_ET ();                                      ! ELAPSED TIME
;         3479  PRINTB (EBD_15, .RC25_ADDR);                    ! "FAILED TO RECEIVE END MESSAGE FROM DEVICE XXXXXX(O)"
;         3480  EMS_CMD ();                                      ! PRINTX COMMAND
;         3481
;         3482  ENDMSG;


                                        .SBTTL  M$EMS.15 ERROR MESSAGE SUBROUTINES
000000  004737  007174'           M$EMS.15:
                                        JSR     PC,EMS.ET                       ;                                       3478
000004  013746  000000G                    MOV     RC25.ADDR,-(SP)                 ;                                       3479
000010  012746  000000G                    MOV     #EBD.15,-(SP)
000014  012746  000002                     MOV     #2,-(SP)
000020  010600                           MOV     SP,R0                           ; SP,*
000022  104414                           TRAP    14
000024  004737  007404'                   JSR     PC,EMS.CMD                      ;                                       3480
000030  062706  000006                   ADD     #6,SP                           ;                                       3476
000034  000207                           RTS     PC

; Routine Size:  15 words,      Routine Base:  $CODE$ + 10500
; Maximum stack depth per invocation:  5 words


;         3483  BGNMSG (EMS_16);


                                        .SBTTL  EMS.16 ERROR MESSAGE SUBROUTINES
000000  004737  000000V          EMS.16::JSR     PC,M$EMS.16                     ;                                       3483
000004  104423                           TRAP    23
000006  000207                           RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 10536
; Maximum stack depth per invocation:  2 words


;         3484
;         3485  EMS_ET ();                                      ! ELAPSED TIME
;         3486  PRINTB (EBD_16, .CPLAT);                        ! "ERROR IN RESPONSE TO ONLINE COMMAND FOR PLATTER XXX."
;         3487  EMS_STC ();                                     ! PRINTX STATUS CODE IF NOT ZERO
;         3488  EMS_SBC ();                                     ! PRINTX SUB-CODE
;         3489
```

```
;         3490  ENDMSG;


                                        .SBTTL  M$EMS.16 ERROR MESSAGE SUBROUTINES
000000  004737  007174'           M$EMS.16:
                                        JSR     PC,EMS.ET                       ;                       3485
000004  013746  000000G                 MOV     CPLAT,-(SP)                     ;                       3486
000010  012746  000000G                 MOV     #EBD.16,-(SP)
000014  012746  000002                  MOV     #2,-(SP)
000020  010600                          MOV     SP,R0                           ; SP,*
000022  104414                          TRAP    14
000024  004737  007312'                 JSR     PC,EMS.STC                      ;                       3487
000030  004737  007344'                 JSR     PC,EMS.SBC                      ;                       3488
000034  062706  000006                  ADD     #6,SP                           ;                       3483
000040  000207                          RTS     PC

; Routine Size:  17 words,      Routine Base:  $CODE$ + 10546
; Maximum stack depth per invocation:  5 words


;         3491  BGNMSG (EMS_17);


                                        .SBTTL  EMS.17 ERROR MESSAGE SUBROUTINES
000000  004737  000000V           EMS.17::JSR     PC,M$EMS.17                     ;                       3491
000004  104423                          TRAP    23
000006  000207                          RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 10610
; Maximum stack depth per invocation:  2 words


;         3492
;         3493  EMS_ET ();                                      ! ELAPSED TIME
;         3494  PRINTB (EBD_17, .CPLAT);                        ! "PLATTER XXX. IS SW WRITE-ENABLED BUT HW WRITE-PROTECTED"
;         3495
;         3496  ENDMSG;


                                        .SBTTL  M$EMS.17 ERROR MESSAGE SUBROUTINES
000000  004737  007174'           M$EMS.17:
                                        JSR     PC,EMS.ET                       ;                       3493
000004  013746  000000G                 MOV     CPLAT,-(SP)                     ;                       3494
000010  012746  000000G                 MOV     #EBD.17,-(SP)
000014  012746  000002                  MOV     #2,-(SP)
000020  010600                          MOV     SP,R0                           ; SP,*
000022  104414                          TRAP    14
000024  062706  000006                  ADD     #6,SP                           ;                       3491
000030  000207                          RTS     PC

; Routine Size:  13 words,      Routine Base:  $CODE$ + 10620
; Maximum stack depth per invocation:  5 words


;         3497  BGNMSG (EMS_18);
```

```
                                        .SBTTL  EMS.18 ERROR MESSAGE SUBROUTINES
000000  004737  000000V            EMS.18::JSR     PC,M$EMS.18                     ;                                       3497
000004  104423                            TRAP    23
000006  000207                            RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 10652
; Maximum stack depth per invocation:  2 words


;        3498
;        3499  EMS_ET ();                                        ! ELAPSED TIME
;        3500  PRINTB (EBD_18, .CPLAT);                          ! "ACCESS FAILED ON PLATTER XXX."
;        3501  EMS_STC ();                                       ! PRINTX STATUS CODE IF NOT ZERO
;        3502  EMS_SBC ();                                       ! PRINTX SUB-CODE
;        3503
;        3504  ENDMSG;


                                        .SBTTL  M$EMS.18 ERROR MESSAGE SUBROUTINES
000000  004737  007174'            M$EMS.18:
                                        JSR     PC,EMS.ET                       ;                                       3499
000004  013746  000000G                     MOV     CPLAT,-(SP)                     ;                                       3500
000010  012746  000000G                     MOV     #EBD.18,-(SP)
000014  012746  000002                      MOV     #2,-(SP)
000020  010600                            MOV     SP,R0                           ; SP,*
000022  104414                            TRAP    14
000024  004737  007312'                    JSR     PC,EMS.STC                      ;                                       3501
000030  004737  007344'                    JSR     PC,EMS.SBC                      ;                                       3502
000034  062706  000006                      ADD     #6,SP                           ;                                       3497
000040  000207                            RTS     PC

; Routine Size:  17 words,     Routine Base:  $CODE$ + 10662
; Maximum stack depth per invocation:  5 words


;        3505  BGNMSG (EMS_19);


                                        .SBTTL  EMS.19 ERROR MESSAGE SUBROUTINES
000000  004737  000000V            EMS.19::JSR     PC,M$EMS.19                     ;                                       3505
000004  104423                            TRAP    23
000006  000207                            RTS     PC

; Routine Size:  4 words,      Routine Base:  $CODE$ + 10724
; Maximum stack depth per invocation:  2 words


;        3506
;        3507  EMS_ET ();                                        ! ELAPSED TIME
;        3508  PRINTB (EBD_19, .CPLAT);                          ! "PLATTER XXX. WENT OFFLINE"
;        3509  EMS_RP ();                                        ! PRINTX RELEVANT RETPKT FIELDS
;        3510
;        3511  ENDMSG;


                                        .SBTTL  M$EMS.19 ERROR MESSAGE SUBROUTINES
000000  004737  007174'            M$EMS.19:
```

```
                                        JSR     PC,EMS.ET                       ;                                       3507
000004  013746  000000G                 MOV     CPLAT,-(SP)                     ;                                       3508
000010  012746  000000G                 MOV     #EBD.19,-(SP)
000014  012746  000002                  MOV     #2,-(SP)
000020  010600                          MOV     SP,R0                           ; SP,*
000022  104414                          TRAP    14
000024  004737  010144'                 JSR     PC,EMS.RP                       ;                                       3509
000030  062706  000006                  ADD     #6,SP                           ;                                       3505
000034  000207                          RTS     PC

; Routine Size:  15 words,      Routine Base:  $CODE$ + 10734
; Maximum stack depth per invocation:  5 words


;         3512  BGNMSG (EMS_20);

                                        .SBTTL  EMS.20 ERROR MESSAGE SUBROUTINES
000000  004737  000000V         EMS.20::JSR     PC,M$EMS.20                     ;                                       3512
000004  104423                          TRAP    23
000006  000207                          RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 10772
; Maximum stack depth per invocation:  2 words


;         3513
;         3514  EMS_ET ();                                      ! ELAPSED TIME
;         3515  PRINTB (EBD_20, .RC25_ADDR);                    ! "DEVICE XXXXXX(0) NOT PROCESSING COMMAND PACKETS"
;         3516
;         3517  ENDMSG;

                                        .SBTTL  M$EMS.20 ERROR MESSAGE SUBROUTINES
000000  004737  007174'         M$EMS.20:
                                        JSR     PC,EMS.ET                       ;                                       3514
000004  013746  000000G                 MOV     RC25.ADDR,-(SP)                 ;                                       3515
000010  012746  000000G                 MOV     #EBD.20,-(SP)
000014  012746  000002                  MOV     #2,-(SP)
000020  010600                          MOV     SP,R0                           ; SP,*
000022  104414                          TRAP    14
000024  062706  000006                  ADD     #6,SP                           ;                                       3512
000030  000207                          RTS     PC

; Routine Size:  13 words,      Routine Base:  $CODE$ + 11002
; Maximum stack depth per invocation:  5 words


;         3518  BGNMSG (EMS_21);

                                        .SBTTL  EMS.21 ERROR MESSAGE SUBROUTINES
000000  004737  000000V         EMS.21::JSR     PC,M$EMS.21                     ;                                       3518
000004  104423                          TRAP    23
000006  000207                          RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 11034
```

```
; Maximum stack depth per invocation:  2 words


;         3519
;         3520  EMS_ET ();                                  ! ELAPSED TIME
;         3521  PRINTB (EBD_21, .RC25_ADDR);                ! 'MESSAGE REJECTED BY DUP SERVER ON DEVICE XXXXXX(O)"
;         3522  EMS_CMD ();                                  ! PRINTX COMMAND
;         3523  PRINTX (EX_DSC, .ST_CODE);                  ! "     DUP STATUS CODE: X."
;         3524
;         3525  ENDMSG;


                                        .SBTTL  M$EMS.21 ERROR MESSAGE SUBROUTINES
000000  004737  007174'          M$EMS.21:
                                        JSR     PC,EMS.ET                       ;                       3520
000004  013746  000000G                 MOV     RC25.ADDR,-(SP)                 ;                       3521
000010  012746  000000G                 MOV     #EBD.21,-(SP)
000014  012746  000002                  MOV     #2,-(SP)
000020  010600                          MOV     SP,R0                           ; SP,*
000022  104414                          TRAP    14
000024  004737  007404'                 JSR     PC,EMS.CMD                      ;                       3522
000030  013716  000000G                 MOV     ST.CODE,(SP)                    ;                       3523
000034  012746  000000G                 MOV     #EX.DSC,-(SP)
000040  012746  000002                  MOV     #2,-(SP)
000044  010600                          MOV     SP,R0                           ; SP,*
000046  104415                          TRAP    15
000050  062706  000012                  ADD     #12,SP                          ;                       3518
000054  000207                          RTS     PC

; Routine Size:  23 words,      Routine Base:  $CODE$ + 11C44
; Maximum stack depth per invocation:  7 words


;         3526  BGNMSG (EMS_22);


                                        .SBTTL  EMS.22 ERROR MESSAGE SUBROUTINES
000000  004737  000000V          EMS.22::JSR    PC,M$EMS.22                      ;                       3526
000004  104423                          TRAP    23
000006  000207                          RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 11122
; Maximum stack depth per invocation:  2 words


;         3527
;         3528  EMS_ET ();                                  ! ELAPSED TIME
;         3529  PRINTB (EBD_22, .RC25_ADDR);                ! "NO RESPONSE FROM FRONT PANEL TEST EXECUTING IN DEVICE XXXXXX(O)"
;         3530
;         3531  ENDMSG;


                                        .SBTTL  M$EMS.22 ERROR MESSAGE SUBROUTINES
000000  004737  007174'          M$EMS.22:
                                        JSR     PC,EMS.ET                       ;                       3528
000004  013746  000000G                 MOV     RC25.ADDR,-(SP)                 ;                       3529
000010  012746  000000G                 MOV     #EBD.22,-(SP)
```

```
000014  012746  000002                        MOV     #2,-(SP)
000020  010600                                MOV     SP,R0                           ; SP,*
000022  104414                                TRAP    14
000024  062706  000006                        ADD     #6,SP                           ;                                       3526
000030  000207                                RTS     PC

; Routine Size:  13 words,      Routine Base:  $CODE$ + 11132
; Maximum stack depth per invocation:  5 words


;         3532  BGNMSG (EMS_30);


                                              .SBTTL  EMS.30 ERROR MESSAGE SUBROUTINES
000000  004737  000000V                EMS.30::JSR     PC,M$EMS.30                     ;                                       3532
000004  104423                                TRAP    23
000006  000207                                RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 11164
; Maximum stack depth per invocation:  2 words


;         3533
;         3534  OWN
;         3535      EBH_TB : VECTOR [11] INITIAL (EBH_31, EBH_32, 0, EBH_34, EBH_35, EBH_36,
;         3536                                    EBH_37, EBH_38, EBH_39, EBH_40, EBH_41);
;         3537          ! TABLE OF BASIC, HARD ERROR MESSAGE ADDRESSES, INDEXED BY STATUS CODE
;         3538
;         3539  EMS_ET ();                                    ! ELAPSED TIME
;         3540  PRINTB (PLATT, .CPLAT);                      ! "PLATTER XXX. - "
;         3541  IF ((.ST_CODE GEQU 1) AND (.ST_CODE LEQU 11)) ! IF STATUS CODE IS WITHIN RANGE
;         3542  THEN                                         ! THEN
;         3543      PRINTB (.EBH_TB [.ST_CODE - 1])          ! PRINTB APPROPRIATE MESSAGE
;         3544  ELSE                                         ! ELSE STATUS CODE OUT OF RANGE
;         3545      PRINTB (EX_SC, .ST_CODE);                ! JUST PRINT STATUS CODE
;         3546  EMS_RP ();                                   ! PRINTX OTHER RETPKT FIELDS
;         3547
;         3548  ENDMSG;


011174  000000G                       EBH.TB: .WORD   EBH.31
011176  000000G                               .WORD   EBH.32
011200  000000                                .WORD   0
011202  000000G                               .WORD   EBH.34
011204  000000G                               .WORD   EBH.35
011206  000000G                               .WORD   EBH.36
011210  000000G                               .WORD   EBH.37
011212  000000G                               .WORD   EBH.38
011214  000000G                               .WORD   EBH.39
011216  000000G                               .WORD   EBH.40
011220  000000G                               .WORD   EBH.41


                                              .SBTTL  M$EMS.30 ERROR MESSAGE SUBROUTINES
000000  004737  007174'                M$EMS.30:
                                              JSR     PC,EMS.ET                       ;                                       3539
000004  013746  000000G                       MOV     CPLAT,-(SP)                     ;                                       3540
```

```
000010  012746  000000G                        MOV     #PLATT,-(SP)
000014  012746  000002                         MOV     #2,-(SP)
000020  010600                                 MOV     SP,R0                           ; SP,*
000022  104414                                 TRAP    14
000024  013700  000000G                        MOV     ST.CODE,R0                      ;                               3541
000030  001413                                 BEQ     1$
000032  020027  000013                         CMP     R0,#13
000036  101010                                 BHI     1$
000040  006300                                 ASL     R0                              ;                               3543
000042  016016  011172'                       MOV     EBH.TB-2(R0),(SP)
000046  012746  000001                         MOV     #1,-(SP)
000052  010600                                 MOV     SP,R0                           ; SP,*
000054  104414                                 TRAP    14
000056  000410                                 BR      2$                              ;                               3541
000060  010016                         1$:     MOV     R0,(SP)                         ;                               3545
000062  012746  000000G                        MOV     #EX.SC,-(SP)
000066  012746  000002                         MOV     #2,-(SP)
000072  010600                                 MOV     SP,R0                           ; SP,*
000074  104414                                 TRAP    14
000076  005726                                 TST     (SP)+
000100  004737  010144'                 2$:     JSR     PC,EMS.RP                       ;                               3546
000104  062706  000010                         ADD     #10,SP                          ;                               3532
000110  000207                                 RTS     PC

; Routine Size:  37 words,      Routine Base:  $CODE$ + 11222
; Maximum stack depth per invocation:  7 words


;         3549  BGNMSG (EMS_42);


                                       .SBTTL  EMS.42 ERROR MESSAGE SUBROUTINES
000000  004737  000000V                EMS.42::JSR     PC,M$EMS.42                     ;                               3549
000004  104423                                 TRAP    23
000006  000207                                 RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 11334
; Maximum stack depth per invocation:  2 words


;         3550
;         3551  EMS_ET ();                                        ! ELAPSED TIME
;         3552  PRINTB (PLATT, .CPLAT);                           ! ''PLATTER XXX. - ''
;         3553  PRINTB (EBH_42);                                  ! ''HOST-DETECTED WRITE-COMPARE ERROR''
;         3554  EMS_LBN ();                                       ! PRINTX LBN
;         3555  EMS_BC ();                                        ! PRINTX BYTE COUNTS
;         3556
;         3557  ENDMSG;


                                       .SBTTL  M$EMS.42 ERROR MESSAGE SUBROUTINES
000000  004737  007174'                M$EMS.42:
                                        JSR     PC,EMS.ET                       ;                               3551
000004  013746  000000G                        MOV     CPLAT,-(SP)                     ;                               3552
000010  012746  000000G                        MOV     #PLATT,-(SP)
000014  012746  000002                         MOV     #2,-(SP)
000020  010600                                 MOV     SP,R0                           ; SP,*
```

```
000022  104414                          TRAP    14
000024  012716  000000G                MOV     #EBH.42,(SP)                    ;                                       3553
000030  012746  000001                  MOV     #1,-(SP)
000034  010600                          MOV     SP,R0                           ; SP,*
000036  104414                          TRAP    14
000040  004737  007746'                 JSR     PC,EMS.LBN                      ;                                       3554
000044  004737  010030'                 JSR     PC,EMS.BC                       ;                                       3555
000050  062706  000010                  ADD     #10,SP                          ;                                       3549
000054  000207                          RTS     PC

; Routine Size:  23 words,      Routine Base:  $CODE$ + 11344
; Maximum stack depth per invocation:  6 words


;         3558  BGNMSG (EMS_43);


                                        .SBTTL  EMS.43 ERROR MESSAGE SUBROUTINES
000000  004737  000000V         EMS.43::JSR     PC,M$EMS.43                     ;                                       3558
000004  104423                          TRAP    23
000006  000207                          RTS     PC

; Routine Size:  4 words,       Routine Base:  $CODE$ + 11422
; Maximum stack depth per invocation:  2 words


;         3559
;         3560  EMS_ET ();                                      ! ELAPSED TIME
;         3561  PRINTB (PLATT, .CPLAT);                         ! "PLATTER XXX. - "
;         3562  PRINTB (EBH_43);                                 ! "FAILED TO RECEIVE END MESSAGE FOR I/O COMMAND"
;         3563  EMS_CRN ();                                      ! PRINTX COMMAND REFERENCE NUMBER
;         3564  RP_ADDR [CMDMOD] = .RP_ADDR [STATUS];           ! STATUS FIELD HOLDS MODIFIER (IF ANY)
;         3565  EMS_CMD ();                                      ! PRINTX COMMAND (AND MODIFIER)
;         3566  PRINTX (EX_LBN, .RP_ADDR [BBLK_LO]);           ! PRINTX LBN
;         3567  PRINTX (EX_CBC, .RP_ADDR [BCNT_LO]);           ! PRINTX BYTE COUNT
;         3568  EMS_BD ();                                      ! PRINTX I/O BUFFER DESCRIPTOR
;         3569
;         3570  ENDMSG;


                                        .SBTTL  M$EMS.43 ERROR MESSAGE SUBROUTINES
000000  010146                  M$EMS.43:
                                        MOV     R1,-(SP)                        ;                                       3558
000002  004737  007174'                 JSR     PC,EMS.ET                       ;                                       3560
000006  013746  000000G                 MOV     CPLAT,-(SP)                     ;                                       3561
000012  012746  000000G                 MOV     #PLATT,-(SP)
000016  012746  000002                  MOV     #2,-(SP)
000022  010600                          MOV     SP,R0                           ; SP,*
000024  104414                          TRAP    14
000026  012716  000000G                 MOV     #EBH.43,(SP)                    ;                                       3562
000032  012746  000001                  MOV     #1,-(SP)
000036  010600                          MOV     SP,R0                           ; SP,*
000040  104414                          TRAP    14
000042  004737  007260'                 JSR     PC,EMS.CRN                      ;                                       3563
000046  013701  000000G                 MOV     RP.ADDR,R1                      ;                                       3564
000052  010100                          MOV     R1,R0                           ; RP.ADDR,*
000054  016061  000016  000012          MOV     16(R0),12(R1)
```

```
000062  004737  007404'                JSR     PC,EMS.CMD                      ;                               3565
000066  013700  000000G                MOV     RP.ADDR,R0                      ;                               3566
000072  016016  000040                 MOV     40(R0),(SP)
000076  012746  000000G                MOV     #EX.LBN,-(SP)
000102  012746  000002                 MOV     #2,-(SP)
000106  010600                         MOV     SP,R0                           ; SP,*
000110  104415                         TRAP    15
000112  013700  000000G                MOV     RP.ADDR,R0                      ;                               3567
000116  016016  000020                 MOV     20(R0),(SP)
000122  012746  000000G                MOV     #EX.CBC,-(SP)
000126  012746  000002                 MOV     #2,-(SP)
000132  010600                         MOV     SP,R0                           ; SP,*
000134  104415                         TRAP    15
000136  004737  010106'                JSR     PC,EMS.BD                       ;                               3568
000142  062706  000020                 ADD     #20,SP                          ;                               3558
000146  012601                         MOV     (SP)+,R1
000150  000207                         RTS     PC

; Routine Size:  53 words,      Routine Base:  $CODE$ + 11432
; Maximum stack depth per invocation:  11 words

;         3571
;         3572  END
;         3573
;         3574  ELUDOM


;                                    OTS external references
                                             .GLOBL  $SAVE5, $SAVE4, $SAVE3, $SAVE2
                                             .GLOBL  BL$MUL

;                                    PSECT SUMMARY
;
;       Psect Name                   Words     Attributes
;         $CODE$                      2498       RO ,  I  ,  LCL,  REL,  CON
;         $PLIT$                       487       RO ,  D  ,  LCL,  REL,  CON


;                                LIBRARY STATISTICS
;                                                 -------- Symbols --------     Blocks
;       File                                      Total    Loaded   Percent       Read
;
;  _DUA2:[DOUCETTE.CZRCD]CZRCDL.L16;9               276       204        73         59



;                                    COMMAND QUALIFIERS
;       BLISS /PDP11 CZRCD2.SRC/LIST/EN:NOEIS

; Size:          2358 code + 627 data words
; Run Time:         01:28.3
; Elapsed Time:     01:58.0
; Memory Used:  396 pages
; Compilation Complete
```

```
;       0001  MODULE CZRCD3 (
;       0002          %TITLE 'CZRCDA0 RC25 DISK EXERCISER'
;       0003          IDENT = 'V01.0',
;       0004          ADDRESSING_MODE (ABSOLUTE)
;       0005          ) =
;       0006  BEGIN
;       0007
;       0043  %SBTTL 'DECLARATIONS'
;       0044
;       0045  LIBRARY 'CZRCDL';                        ! RC25 EXERCISER GLOBAL LIBRARY
;       0046  REQUIRE 'BLSMAC.REQ';                    ! DIAGNOSTIC SUPERVISOR LIBRARY
;       1535
;       1536  EQUALS;
;       1537
;       1538                                          ! ROUTINES APPEAR IN THIS ORDER
;       1539  FORWARD ROUTINE                         !    INDENTATION IMPLIES CALLED SUBROUTINE
;       1540      INIT_TEST : NOVALUE,                ! INITIALIZATION SUBTEST
;       1541          DRIVER_INIT : NOVALUE,
;       1542              CTLR_INIT : NOVALUE,
;       1543                  INI_CTLR_DAT : NOVALUE,
;       1544                  REG_EXIST,
;       1545                  VEC_BR_TEST,
;       1546                  INT_GEN,
;       1547                  HARD_INIT,
;       1548                  INI_RRING : NOVALUE,
;       1549                  SET_CTLR_CHAR : NOVALUE,
;       1550              UNIT_INIT : NOVALUE,
;       1551          ACCESS : NOVALUE,
```

```
1552        DM_EXER : NOVALUE,                  ! DM EXERCISER SUBTEST
1553            DM_INIT : NOVALUE,
1554            DM_QUIT,
1555            DM_RETPKT : NOVALUE,
1556                DM_MSG : NOVALUE,
1557            DM_ACC,
1558            DM_TALLY : NOVALUE,
1559                UPD_DMBC : NOVALUE,
1560            DM_TIME : NOVALUE,
1561        MULTI_DRIVE : NOVALUE,              ! MULTI-DRIVE SUBTEST
1562            MD_INIT : NOVALUE,
1563                INIT_IO_BUFF : NOVALUE,
1564            QIO_OK,
1565            MD_QUIT,
1566            QIO_GEN : NOVALUE,
1567                GET_RANDOM : NOVALUE,
1568                QIO_UNIT : NOVALUE,
1569                QIO_FUNC : NOVALUE,
1570                QIO_LBN : NOVALUE,
1571                    ADV_BST : NOVALUE,
1572                QIO_SIZE : NOVALUE,
1573                FILL_BUFF : NOVALUE,
1574            PROC_RETPKT : NOVALUE,
1575                IO_RETPKT : NOVALUE,
1576                    FSET_UPAR : NOVALUE,
1577                    MD_TALLY : NOVALUE,
1578                    HOST_WRT_CHK : NOVALUE,
1579                        CMP_DATA,
1580                    SWEEP : NOVALUE,
1581                        RPS_REM,
1582                DR_RETPKT : NOVALUE,
1583            DRV_TIMCHK : NOVALUE,
1584        RCINT0 : L$ISR NOVALUE,             ! RC25 INTERRUPT SERVICE ROUTINES
1585        RCINT1 : L$ISR NOVALUE,
1586        RCINT2 : L$ISR NOVALUE,
1587        RCINT3 : L$ISR NOVALUE,
```

```
1588            RCINT : NOVALUE,
1589            INT_PROC : NOVALUE,
1590                ISET_CPAR : NOVALUE,
1591                FATAL_ERROR : NOVALUE,
1592                POLL_CRING : NOVALUE,
1593                POLL_RRING : NOVALUE,
1594                    ENV_TO_RP : NOVALUE,
1595                    DATAGM : NOVALUE;
1596
1597    PSECT OWN = $GGGG$ (READ, NOWRITE, EXECUTE, LOCAL, CONCATENATE);
1598
1599    OWN
1600        COMM_AREA : BLOCKVECTOR [MAX_CTLR, COMM_LEN, WORD] FIELD (COM_FIELDS),
1601    !       U/Q PORT COMMUNICATIONS AREA BETWEEN HOST AND RC25 CONTROLLERS
1602        BST : BLOCKVECTOR [MAX_UNITS, BST_LEN, WORD] FIELD (B_FIELDS),
1603    !       BLOCK SEQUENCE TABLE FOR SEQUENTIAL LBN (VS. RANDOM SEEK) MODE
1604        DPST : VECTOR [MAX_UNITS, BYTE],                    ! DATA PATTERN SEQUENCE TABLE
1605        ICOM_ADDR : REF BLOCK [COMM_LEN, WORD] FIELD (COM_FIELDS),
1606    !       ADDRESS OF INTERRUPTING CONTROLLER'S U/Q PORT COMMUNICATION AREA
1607        IENV_ADDR : REF BLOCK [ENV_LEN, WORD] FIELD (E_FIELDS),
1608    !       ADDRESS OF AN MSCP ENVELOPE (INTERRUPT PROCESSING)
1609        ICST_ADDR : REF BLOCK [CST_LEN, WORD] FIELD (C_FIELDS),
1610    !       ADDRESS OF INTERRUPTING CONTROLLER'S CST
1611        IDCT_ADDR : REF BLOCK [DCT_LEN, WORD] FIELD (DC_FIELDS),
1612    !       ADDRESS OF INTERRUPTING CONTROLLER'S DCT
1613        INT_ADDR : VECTOR [MAX_CTLR] INITIAL (RCINT0, RCINT1, RCINT2, RCINT3),
1614    !       INTERRUPT SERVICE ROUTINE ADDRESS TABLE
1615        ICTLR : WORD,                                       ! INTERRUPTING CONTROLLER NUMBER
1616        MX1 : SIGNED WORD,                                  ! MSCP ENV INDEX FOR FIRST QIO
1617        MX2 : SIGNED WORD,                                  ! MSCP ENV INDEX FOR SECOND QIO
1618        MAD1 : REF BLOCK [ENV_LEN, WORD] FIELD (E_FIELDS),
1619    !       ADDRESS OF MSCP ENVELOPE FOR FIRST QIO
1620        MAD2 : REF BLOCK [ENV_LEN, WORD] FIELD (E_FIELDS),
1621    !       ADDRESS OF MSCP ENVELOPE FOR SECOND QIO
1622        VEC_AD : WORD,                                      ! CURRENT DEVICE'S VECTOR ADDRESS
1623        BRLEVEL : WORD,                                     ! CURRENT DEVICE'S BR LEVEL
1624        USIZE : WORD,                                       ! UNIT SIZE (NO. OF LBN'S)
1625        DM_TIMR : VECTOR [MAX_CTLR],                        ! TIMERS FOR ONE PASS OF FRONT PANEL TEST (DM EXER)
1626        SWEEP_FLAG : BYTE,                                  ! CALL / DON'T CALL SWEEP () (MULTI-DRIVE SUBTEST)
1627        RDM_CNT : WORD INITIAL (RDM_LEN),                   ! NUMBER OF RANDOM NUMBERS          \ KEEP
1628        RANDOM : VECTOR [RDM_LEN, WORD],                    ! RANDOM NUMBER TABLE (PATTERN 1) / TOGETHER
1629        PAT02 : VECTOR [2] INITIAL (1,                                  ! PATTERN 2
1630            %O'000000'),
1631        PAT03 : VECTOR [2] INITIAL (1,                                  ! PATTERN 3
1632            %O'177777'),
1633        PAT04 : VECTOR [2] INITIAL (1,                                  ! PATTERN 4
1634            %O'105613'),
1635        PAT05 : VECTOR [2] INITIAL (1,                                  ! PATTERN 5
1636            %O'031463'),
1637        PAT06 : VECTOR [2] INITIAL (1,                                  ! PATTERN 6
1638            %O'030221'),
1639        PAT07 : VECTOR [17] INITIAL (16,                                ! PATTERN 7
1640            %O'000001', %O'000003', %O'000007', %O'000017',
1641            %O'000037', %O'000077', %O'000177', %O'000377',
1642            %O'000777', %O'001777', %O'003777', %O'007777',
1643            %O'017777', %O'037777', %O'077777', %O'177777'),
1644        PAT08 : VECTOR [17] INITIAL (16,                                ! PATTERN 8
```

```
;       1645                %O'177776', %O'177774', %O'177770', %O'177760',
;       1646                %O'177740', %O'177700', %O'177600', %O'177400',
;       1647                %O'177000', %O'176000', %O'174000', %O'170000',
;       1648                %O'160000', %O'140000', %O'100000', %O'000000'),
;       1649        PAT09 : VECTOR [17] INITIAL (16,                        ! PATTERN 9
;       1650                %O'000000', %O'000000', %O'000000', %O'177777',
;       1651                %O'177777', %O'177777', %O'000000', %O'000000',
;       1652                %O'177777', %O'177777',
;       1653                %O'000000', %O'177777', %O'000000', %O'177777',
;       1654                %O'000000', %O'177777'),
;       1655        PAT10 : VECTOR [2] INITIAL (1,                          ! PATTERN 10
;       1656                %O'133331'),
;       1657        PAT11 : VECTOR [17] INITIAL (16,                        ! PATTERN 11
;       1658                %O'052525', %O'052525', %O'052525', %O'125252',
;       1659                %O'125252', %O'125252', %O'052525', %O'052525',
;       1660                %O'125252', %O'125252',
;       1661                %O'052525', %O'125252', %O'052525', %O'125252',
;       1662                %O'052525', %O'125252'),
;       1663        PAT12 : VECTOR [21] INITIAL (20,                        ! PATTERN 12
;       1664                %O'026455', %O'026455', %O'026455', %O'151322',
;       1665                %O'151322', %O'151322', %O'026455', %O'026455',
;       1666                %O'151322', %O'151322', %O'026455', %O'026455',
;       1667                %O'151322', %O'026455', %O'151322', %O'026455',
;       1668                %O'151322', %O'026455', %O'151322', %O'026455'),
;       1669        PAT13 : VECTOR [2] INITIAL (1,                          ! PATTERN 13
;       1670                %O'066666'),
;       1671        PAT14 : VECTOR [17] INITIAL (16,                        ! PATTERN 14
;       1672                %O'000001', %O'000002', %O'000004', %O'000010',
;       1673                %O'000020', %O'000040', %O'000100', %O'000200',
;       1674                %O'000400', %O'001000', %O'002000', %O'004000',
;       1675                %O'010000', %O'020000', %O'040000', %O'100000'),
;       1676        PAT15 : VECTOR [17] INITIAL (16,                        ! PATTERN 15
;       1677                %O'177776', %O'177775', %O'177773', %O'177767',
;       1678                %O'177757', %O'177737', %O'177677', %O'177577',
;       1679                %O'177377', %O'176777', %O'175777', %O'173777',
;       1680                %O'167777', %O'157777', %O'137777', %O'077777'),
;       1681        PAT16 : VECTOR [17] INITIAL (16,                        ! PATTERN 16
;       1682                %O'133331', %O'133331', %O'133331', %O'155554',
;       1683                %O'155554', %O'155554', %O'133331', %O'133331',
;       1684                %O'155554', %O'155554',
;       1685                %O'133331', %O'155554', %O'133331', %O'155554',
;       1686                %O'133331', %O'155554'),
;       1687        PAT17 : VECTOR [22] INITIAL (21,                        ! PATTERN 17
;       1688                %O'000000', %O'106466', %O'106466', %O'071311',
;       1689                %O'071311', %O'071311', %O'106466', %O'106466',
;       1690                %O'106466', %O'106466', %O'071311', %O'071311',
;       1691                %O'071311', %O'071311', %O'071311', %O'106466',
;       1692                %O'106466', %O'106466', %O'106466', %O'106466',
;       1693                %O'106466'),
;       1694        PAT18 : VECTOR [22] INITIAL (21,                        ! PATTERN 18
;       1695                %O'106466', %O'000000', %O'071311',
;       1696                %O'106466', %O'106466', %O'106466', %O'071311',
;       1697                %O'071311', %O'071311', %O'071311', %O'106466',
;       1698                %O'106466', %O'106466', %O'106466', %O'106466',
;       1699                %O'071311', %O'071311', %O'071311', %O'071311',
;       1700                %O'071311', %O'071311'),
;       1701        PAT19 : VECTOR [22] INITIAL (21,                        ! PATTERN 19
```

```
1702                %O'000000', %O'134631', %O'134631', %O'043146',
1703                %O'043146', %O'043146', %O'134631', %O'134631',
1704                %O'134631', %O'134631', %O'043146', %O'043146',
1705                %O'043146', %O'043146', %O'043146', %O'134631',
1706                %O'134631', %O'134631', %O'134631', %O'134631',
1707                %O'134631'),
1708        PAT20 : VECTOR [22] INITIAL (21,                                ! PATTERN 20
1709                %O'134631', %O'000000', %O'043146',
1710                %O'134631', %O'134631', %O'134631', %O'043146',
1711                %O'043146', %O'043146', %O'043146', %O'134631',
1712                %O'134631', %O'134631', %O'134631', %O'134631',
1713                %O'043146', %O'043146', %O'043146', %O'043146',
1714                %O'043146', %O'043146'),
1715        PAT21 : VECTOR [2] INITIAL (1,                                  ! PATTERN 21
1716                %O'000000'),                                            !   (LBN)
1717        DPA_TBL : VECTOR [DP_CNT] INITIAL                       ! DATA PATTERN ADDRESS TABLE
1718                  (RDM_CNT, PAT02, PAT03, PAT04, PAT05,
1719                  PAT06, PAT07, PAT08, PAT09, PAT10, PAT11,
1720                  PAT12, PAT13, PAT14, PAT15, PAT16, PAT17,
1721                  PAT18, PAT19, PAT20, PAT21),
1722    !+
1723    !       THE FOLLOWING CODE IS THE CROM PRIMER WHICH WAS DEVELOPED
1724    !       INDEPENDENTLY. ITS BINARY .SAV FILE WAS RUN THROUGH THE PROGRAM
1725    !       "DMCONV", PRODUCING THIS COMPILABLE VECTOR.
1726    !-
1727
1728    CROMP:VECTOR[206,WORD]
1729    PRESET (
1730    [0]     =   %O'000140',   ! THIS IS THE DM PROGRAM BYTE COUNT.
1731    [1]     =   %O'000000',
1732    [2]     =   %O'000472',   ! THIS IS THE DM OVERLAY BYTE COUNT.
1733    [3]     =   %O'000000',
1734    [4]     =   %O'051103',   ! NEXT 3 WORDS = PROGRAM NAME (ASCII)
1735    [5]     =   %O'046517',   ! PROGRAM NAME IS 'CROMP '
1736    [6]     =   %O'020120',
1737    [7]     =   %O'000001',   ! THIS IS THE PROGRAM VERSION
1738    [8]     =   %O'000013',   ! UPPER BYTE=TIME OUT VAL. LOWER = FLAGS
1739    [9]     =   %O'000000',
1740    [10]    =   %O'000000',
1741    [11]    =   %O'000000',
1742    [12]    =   %O'000000',
1743    [13]    =   %O'000000',
1744    [14]    =   %O'000000',
1745    [15]    =   %O'000000',
1746    [16]    =   %O'104206',   ! DM CODE STARTS HERE
1747    [17]    =   %O'007760',
1748    [18]    =   %O'002754',
1749    [19]    =   %O'000000',
1750    [20]    =   %O'000000',
1751    [21]    =   %O'000000',
1752    [22]    =   %O'000000',
1753    [23]    =   %O'000000',
1754    [24]    =   %O'000000',
1755    [25]    =   %O'000235',
1756    [26]    =   %O'000000',
1757    [27]    =   %O'000000',
1758    [28]    =   %O'104204',
```

```
;          1759  [29]    =  %O'007774',
;          1760  [30]    =  %O'104140',
;          1761  [31]    =  %O'002752',
;          1762  [32]    =  %O'104204',
;          1763  [33]    =  %O'002751',
;          1764  [34]    =  %O'104203',
;          1765  [35]    =  %O'007000',
;          1766  [36]    =  %O'104647',
;          1767  [37]    =  %O'000001',
;          1768  [38]    =  %O'104641',
;          1769  [39]    =  %O'000002',
;          1770  [40]    =  %O'104142',
;          1771  [41]    =  %O'060020',
;          1772  [42]    =  %O'102207',
;          1773  [43]    =  %O'000037',
;          1774  [44]    =  %O'052754',
;          1775  [45]    =  %O'104077',
;          1776  [46]    =  %O'007006',
;          1777  [47]    =  %O'032305',
;          1778  [48]    =  %O'000000',  ! DM OVERLAY CODE STARTS HERE
;          1779  [49]    =  %O'000000',
;          1780  [50]    =  %O'000000',
;          1781  [51]    =  %O'000000',
;          1782  [52]    =  %O'000000',
;          1783  [53]    =  %O'000000',
;          1784  [54]    =  %O'104207',
;          1785  [55]    =  %O'007000',
;          1786  [56]    =  %O'104201',
;          1787  [57]    =  %O'000002',
;          1788  [58]    =  %O'104202',
;          1789  [59]    =  %O'002740',
;          1790  [60]    =  %O'107027',
;          1791  [61]    =  %O'107017',
;          1792  [62]    =  %O'105012',
;          1793  [63]    =  %O'060011',
;          1794  [64]    =  %O'027107',
;          1795  [65]    =  %O'027203',
;          1796  [66]    =  %O'027146',
;          1797  [67]    =  %O'114000',
;          1798  [68]    =  %O'007005',
;          1799  [69]    =  %O'114000',
;          1800  [70]    =  %O'007776',
;          1801  [71]    =  %O'114000',
;          1802  [72]    =  %O'007776',
;          1803  [73]    =  %O'104077',
;          1804  [74]    =  %O'104206',
;          1805  [75]    =  %O'007760',
;          1806  [76]    =  %O'104301',
;          1807  [77]    =  %O'007005',
;          1808  [78]    =  %O'104610',
;          1809  [79]    =  %O'007234',
;          1810  [80]    =  %O'007767',
;          1811  [81]    =  %O'115000',
;          1812  [82]    =  %O'007767',
;          1813  [83]    =  %O'077076',
;          1814  [84]    =  %O'027120',
;          1815  [85]    =  %O'104200',
```

```
;          1816  [86]    =  %O'007051',
;          1817  [87]    =  %O'002750',
;          1818  [88]    =  %O'002740',
;          1819  [89]    =  %O'027146',
;          1820  [90]    =  %O'115000',
;          1821  [91]    =  %O'007265',
;          1822  [92]    =  %O'057051',
;          1823  [93]    =  %O'104200',
;          1824  [94]    =  %O'177777',
;          1825  [95]    =  %O'007004',
;          1826  [96]    =  %O'027130',
;          1827  [97]    =  %O'027146',
;          1828  [98]    =  %O'115000',
;          1829  [99]    =  %O'007265',
;          1830  [100]   =  %O'017071',
;          1831  [101]   =  %O'114000',
;          1832  [102]   =  %O'007004',
;          1833  [103]   =  %O'027130',
;          1834  [104]   =  %O'007051',
;          1835  [105]   =  %O'027162',
;          1836  [106]   =  %O'027203',
;          1837  [107]   =  %O'114000',
;          1838  [108]   =  %O'007264',
;          1839  [109]   =  %O'027216',
;          1840  [110]   =  %O'115400',
;          1841  [111]   =  %O'007005',
;          1842  [112]   =  %O'106200',
;          1843  [113]   =  %O'000003',
;          1844  [114]   =  %O'007005',
;          1845  [115]   =  %O'037031',
;          1846  [116]   =  %O'114000',
;          1847  [117]   =  %O'007005',
;          1848  [118]   =  %O'007031',
;          1849  [119]   =  %O'104207',
;          1850  [120]   =  %O'007002',
;          1851  [121]   =  %O'104201',
;          1852  [122]   =  %O'000002',
;          1853  [123]   =  %O'060023',
;          1854  [124]   =  %O'102207',
;          1855  [125]   =  %O'000037',
;          1856  [126]   =  %O'057107',
;          1857  [127]   =  %O'000000',
;          1858  [128]   =  %O'104307',
;          1859  [129]   =  %O'007005',
;          1860  [130]   =  %O'104201',
;          1861  [131]   =  %O'000005',
;          1862  [132]   =  %O'060031',
;          1863  [133]   =  %O'104010',
;          1864  [134]   =  %O'007001',
;          1865  [135]   =  %O'000000',
;          1866  [136]   =  %O'104307',
;          1867  [137]   =  %O'007002',
;          1868  [138]   =  %O'105207',
;          1869  [139]   =  %O'000060',
;          1870  [140]   =  %O'114001',
;          1871  [141]   =  %O'104202',
;          1872  [142]   =  %O'000001',
```

```
;          1873  [143]   =  %O'104203',
;          1874  [144]   =  %O'007004',
;          1875  [145]   =  %O'060021',
;          1876  [146]   =  %O'102207',
;          1877  [147]   =  %O'000037',
;          1878  [148]   =  %O'057130',
;          1879  [149]   =  %O'000000',
;          1880  [150]   =  %O'104307',
;          1881  [151]   =  %O'007002',
;          1882  [152]   =  %O'114001',
;          1883  [153]   =  %O'104202',
;          1884  [154]   =  %O'000032',
;          1885  [155]   =  %O'104203',
;          1886  [156]   =  %O'007234',
;          1887  [157]   =  %O'060020',
;          1888  [158]   =  %O'102207',
;          1889  [159]   =  %O'000037',
;          1890  [160]   =  %O'057146',
;          1891  [161]   =  %O'000000',
;          1892  [162]   =  %O'104200',
;          1893  [163]   =  %O'000004',
;          1894  [164]   =  %O'007000',
;          1895  [165]   =  %O'104207',
;          1896  [166]   =  %O'002743',
;          1897  [167]   =  %O'104301',
;          1898  [168]   =  %O'007001',
;          1899  [169]   =  %O'104272',
;          1900  [170]   =  %O'105612',
;          1901  [171]   =  %O'007240',
;          1902  [172]   =  %O'100612',
;          1903  [173]   =  %O'007240',
;          1904  [174]   =  %O'115401',
;          1905  [175]   =  %O'117400',
;          1906  [176]   =  %O'007000',
;          1907  [177]   =  %O'037171',
;          1908  [178]   =  %O'000000',
;          1909  [179]   =  %O'114000',
;          1910  [180]   =  %O'002743',
;          1911  [181]   =  %O'114000',
;          1912  [182]   =  %O'002744',
;          1913  [183]   =  %O'114000',
;          1914  [184]   =  %O'002745',
;          1915  [185]   =  %O'114000',
;          1916  [186]   =  %O'002746',
;          1917  [187]   =  %O'114000',
;          1918  [188]   =  %O'002747',
;          1919  [189]   =  %O'000000',
;          1920  [190]   =  %O'104307',
;          1921  [191]   =  %O'007002',
;          1922  [192]   =  %O'105207',
;          1923  [193]   =  %O'000010',
;          1924  [194]   =  %O'114001',
;          1925  [195]   =  %O'104202',
;          1926  [196]   =  %O'000025',
;          1927  [197]   =  %O'104203',
;          1928  [198]   =  %O'007240',
;          1929  [199]   =  %O'060021',
```

```
1930  [200]   =  %O'102207',
1931  [201]   =  %O'000037',
1932  [202]   =  %O'057216',
1933  [203]   =  %O'000000',
1934  [204]   =  %O'143367',
1935  [205]   =  %O'000000');
1936
1937  EXTERNAL
1938      PATCH : VECTOR [100, WORD],                     ! PATCH AREA
1939      CPT : VECTOR [MAX_UNITS, BYTE],
1940  !       CURRENT PASS TESTING (YES / NO) PER UNIT
1941      CST : BLOCKVECTOR [MAX_CTLR, CST_LEN, WORD] FIELD (C_FIELDS),
1942  !       RUN-TIME CONTROLLER STATUS TABLES
1943      CST_ADDR : REF BLOCK [CST_LEN, WORD] FIELD (C_FIELDS),
1944  !       CONTROLLER STATUS TABLE ADDRESS OF "CURRENT" CONTROLLER
1945      DCT : BLOCKVECTOR [MAX_CTLR, DCT_LEN, WORD] FIELD (DC_FIELDS),
1946  !       DRIVER CONTROLLER TABLES
1947      DCT_ADDR : REF BLOCK [DCT_LEN, WORD] FIELD (DC_FIELDS),
1948  !       ADDRESS OF "CURRENT" DRIVER CONTROLLER TABLE
1949      RC25_ADDR : REF RC25 FIELD (RC_REG),
1950  !       DEVICE ADDRESS OF "CURRENT" CONTROLLER
1951      IRC25_ADDR : REF RC25 FIELD (RC_REG),
1952  !       DEVICE ADDRESS OF INTERRUPTING CONTROLLER
1953      DM_COMM : BLOCKVECTOR [MAX_CTLR, DMC_LEN, WORD] FIELD (DMC_FIELDS),
1954  !       DM EXERCISER COMMUNICATION AREA (LINK TO FRONT PANEL TEST)
1955      DMC_ADDR : REF BLOCK [DMC_LEN, WORD] FIELD (DMC_FIELDS),
1956  !       ADDRESS OF CURRENT CONTROLLER'S DM EXERCISER COMMUNICATION AREA
1957      RP_SAVE : VECTOR [MAX_CTLR * RPS_LEN, BYTE, SIGNED],
1958  !       RETURN PACKET SAVE AREA
1959      RPS_X1 : WORD,                                  ! STARTING INDEX OF CURRENT CONTROLLER'S RP_SAVE AREA
1960      RPS_X2 : WORD,                                  ! ENDING INDEX OF CURRENT CONTROLLER'S RP_SAVE AREA
1961      OUTC_LIST : VECTOR [MAX_CTLR * OUTC_CNT, BYTE, SIGNED],
1962  !       OUTSTANDING COMMAND LIST (CONTAINS MSCP ENVELOPE INDECES)
1963      OUTC_TIMR : VECTOR [MAX_CTLR * OUTC_CNT, WORD],
1964  !       OUTSTANDING COMMAND TIMERS
1965      OCL_X1 : WORD,
1966  !       STARTING INDEX OF CURRENT CONTROLLER'S OUTSTANDING COMMAND AREA
1967      OCL_X2 : WORD,
1968  !       ENDING INDEX OF CURRENT CONTROLLER'S OUTSTANDING COMMAND AREA
1969      TALLY : VECTOR [MAX_UNITS * TALLY_LEN, WORD] FIELD (T_FIELDS),
1970  !       STATISTICS TABLES
1971      T_ADDR : REF BLOCK [TALLY_LEN, WORD] FIELD (T_FIELDS),
1972  !       ADDRESS OF STATISTICS TABLE (TALLY) FOR CURRENT UNIT
1973      MSCP_ENV : BLOCKVECTOR [ENV_CNT, ENV_LEN, WORD] FIELD (E_FIELDS),
1974  !       MSCP ENVELOPE POOL
1975      ENV_USE : VECTOR [ENV_CNT, BYTE, SIGNED],
1976  !       MSCP ENVELOPE POOL ALLOCATION TABLE
1977      RETPKT : BLOCKVECTOR [RP_CNT, RP_LEN, WORD] FIELD (RP_FIELDS),
1978  !       RETURN PACKET POOL
1979      RP_USE : VECTOR [RP_CNT, BYTE, SIGNED],
1980  !       RETURN PACKET POOL ALLOCATION TABLE
1981      RP_INDX : WORD,                                 ! CURRENT RETURN PACKET INDEX
1982      RP_ADDR : REF BLOCK [RP_LEN, WORD] FIELD (RP_FIELDS),
1983  !       CURRENT RETURN PACKET ADDRESS
1984      BUFF_DESC : BLOCKVECTOR [MAX_BUF_CNT, DESC_LEN, WORD] FIELD (BD_FIELDS),
1985  !       TABLE OF I/O BUFFER DESCRIPTORS
1986      BUFF_OWN : VECTOR [MAX_BUF_CNT, BYTE, SIGNED],
```

```
;       1987  !       I/O BUFFER OWNERSHIP (CONTROLLER NUMBER)
;       1988      IODQ : VECTOR [IODQ_LEN, BYTE],
;       1989  !       I/O DONE QUEUE = CIRCULAR QUEUE OF RETPKT INDECES
;       1990      IODQ_IN : WORD,                   ! I/O DONE QUEUE IN POINTER
;       1991      IODQ_OUT : WORD,                  ! I/O DONE QUEUE OUT POINTER
;       1992      ENTRY_REASON : BYTE,              ! HOW CURRENT PASS WAS INVOKED
;       1993      T_FLAG : BYTE,                    ! ONE SECOND TIMING FLAG
;       1994      EOP_FLAG : BYTE,                  ! END-OF-PASS FLAG
;       1995      MEM_MGMT : BYTE,                  ! MEMORY MANAGEMENT FLAG
;       1996      IIP_FLAG : BYTE,                  ! INITIALIZATION-IN-PROGRESS FLAG
;       1997      CCTLR : WORD,                     ! NUMBER OF "CURRENT" CONTROLLER
;       1998      CPLAT : WORD,                     ! CURRENT PLATTER ADDRESS (MSCP UNIT NUMBER)
;       1999      CUOFF : WORD,                     ! CST OFFSET FOR CURRENT UNIT
;       2000      CTLR_CNT : WORD,                  ! TOTAL NUMBER OF CONFIGURED CONTROLLERS
;       2001      DUR : VECTOR [MAX_UNITS, BYTE],   ! DROP UNIT REASON
;       2002      QIO : VECTOR [MAX_CTLR, BYTE],    ! NUMBER OF OUTSTANDING QIOS PER CONTROLLER
;       2003      MEM_SIZE : WORD,                  ! AVAILABLE MEMORY (IN WORDS) UP TO 28K
;       2004      FREE_MEM_ADDR,                    ! START OF FREE MEMORY BELOW 28K
;       2005      BUFF_SIZE : WORD,                 ! SIZE (BYTES) OF AN I/O BUFFER
;       2006      NUM_BUFF : WORD,                  ! NUMBER OF I/O BUFFERS
;       2007      CLK_TYPE : WORD,                  ! TYPE OF CLOCK ON SYSTEM
;       2008                                        !   (0 = NONE, -1 = L-CLOCK, 1 = P_CLOCK)
;       2009      CLK_HERTZ : WORD,                 ! CLOCK HERTZ RATE
;       2010      CLK_CSR,                          ! CLOCK CSR ADDRESS
;       2011      CLK_VECTOR,                       ! CLOCK VECTOR ADDRESS
;       2012      HOURS : WORD,                     ! ELAPSED TIME - HOURS,
;       2013      MINUTES : WORD,                   !       MINUTES,
;       2014      SECONDS : WORD,                   !       SECONDS,
;       2015      TICKS : WORD,                     !       TICKS
;       2016      ST_CODE : WORD,                   ! CURRENT STATUS CODE
;       2017      SB_CODE : WORD,                   ! CURRENT SUB-CODE
;       2018      STEP : WORD,                      ! CURRENT STEP IN HARD_INIT
;       2019      OF_RC : SIGNED WORD,              ! OFFSET (0 OR 2) TO READ IP OR SA
;       2020      SA_REG : WORD,                    ! STORAGE FOR SA REGISTER READS AND WRITES
;       2021      NEX : WORD,                       ! NON-EXISTENT MEMORY TRAP INDICATOR
;       2022      CRN : WORD,                       ! COMMAND REF NUMBER OF LAST COMMAND SENT
;       2023      MSG_02,
;       2024      MSG_03,
;       2025      MSG_04,
;       2026      MSG_05,
;       2027      MSG_06,
;       2028      MSG_08,
;       2029      EGD_10,
;       2030      EGD_11,
;       2031      EGD_12,
;       2032      EGD_13,
;       2033      EGD_14,
;       2034      EGD_15,
;       2035      EGD_16,
;       2036      EGD_17,
;       2037      EGD_18,
;       2038      EGD_19,
;       2039      EGD_20,
;       2040      EGD_21,
;       2041      EGD_22,
;       2042      EGH_30,
;       2043      STC_00,
```

```
;        2044      STC_01,
;        2045      STC_02,
;        2046      STC_03,
;        2047      STC_04,
;        2048      STC_05,
;        2049      STC_06,
;        2050      STC_07,
;        2051      STC_08,
;        2052      STC_09,
;        2053      STC_10,
;        2054      STC_11,
;        2055      EX_CRN,
;        2056      EX_SB,
;        2057      EX_EL,
;        2058      EX_PA,
;        2059      EX_FMT,
;        2060      EX_EVC,
;        2061      EX_HMA,
;        2062      EX_03,
;        2063      SWP_STRACK : WORD,                    ! STARTING TRACK
;        2064      SWP_ETRACK : WORD,                    ! ENDING TRACK
;        2065      SWP_FLAGS : BYTE,                     ! FLAGS (SEE DOCUMENTATION)
;        2066      SWP_DPAT : BYTE,                      ! DATA PATTERN NUMBER
;        2067      SWP_UCNT : WORD,                      ! USER DATA PATTERN COUNT
;        2068      SWP_UDPAT : VECTOR [MAX_UDP_CNT, WORD],   ! USER DATA PATTERN
;        2069      L$HIMEM,
;        2070      L$UNIT,
;        2071      L$LUN;
;        2072
;        2073  EXTERNAL ROUTINE
;        2074      NEX_TRAP : L$ISR NOVALUE,
;        2075      COPY_BLK : NOVALUE,
;        2076      SET_CPAR : NOVALUE,
;        2077      SET_UPAR : NOVALUE,
;        2078      GET_ENV,
;        2079      PUT_ENV : NOVALUE,
;        2080      GET_RETPKT,
;        2081      PUT_RETPKT : NOVALUE,
;        2082      GET_IO_BUFF : NOVALUE,
;        2083      PUT_IO_BUFF : NOVALUE,
;        2084      PUTA_BUFF : NOVALUE,
;        2085      OUT_IODQ,
;        2086      IN_IODQ : NOVALUE,
;        2087      DROP_CTLR : NOVALUE,
;        2088      DRV_CTLERR : NOVALUE,
;        2089      HARD_ERR : NOVALUE,
;        2090      UPD_IOC : NOVALUE,
;        2091      OVF_CHK : NOVALUE,
;        2092      XFR_CHK : NOVALUE,
;        2093      SEND,
;        2094      WAIT : NOVALUE,
;        2095      EMS_10 : NOVALUE,
;        2096      EMS_12 : NOVALUE,
;        2097      EMS_13 : NOVALUE,
;        2098      EMS_14 : NOVALUE,
;        2099      EMS_15 : NOVALUE,
;        2100      EMS_16 : NOVALUE,
```

```
;          2101      EMS_17 : NOVALUE,
;          2102      EMS_18 : NOVALUE,
;          2103      EMS_19 : NOVALUE,
;          2104      EMS_20 : NOVALUE,
;          2105      EMS_21 : NOVALUE,
;          2106      EMS_22 : NOVALUE,
;          2107      EMS_30 : NOVALUE,
;          2108      EMS_42 : NOVALUE,
;          2109      EMS_43 : NOVALUE;
```

```
;        2110  %SBTTL 'TEST SECTION'
;        2111
;        2112  !+
;        2113  !       THIS SECTION CONTAINS THE TOP-LEVEL TEST CODE FOR THE RC25 DISK
;        2114  !       EXERCISER. THE EXERCISER CONSISTS OF ONE TEST WHICH IS SUBDIVIDED INTO
;        2115  !       A NUMBER OF SUBTESTS. ALL SUBTESTS ARE DECLARED WITHIN THIS BLOCK.
;        2116  !-
;        2117
;        2118  BGNTST;
;        2119
;        2120  EOP_FLAG = FALSE;
;        2121  IF .ENTRY_REASON NEQU NEW_PASS                ! IF START, RESTART, CONT, OR PWR FAIL
;        2122  THEN                                          ! THEN
;        2123      BEGIN
;        2124
;        2125      BGNSUB;
;        2126      INIT_TEST ();                             ! INITIALIZATION SUBTEST
;        2127      ENDSUB;
;        2128
;        2129      END;
;        2130
;        2131  IF BIT_TST (SWP_FLAGS, SWF_DM)               ! IF OPERATOR SELECTED DM EXERCISER
;        2132  THEN                                          ! THEN
;        2133      BEGIN
;        2134
;        2135      BGNSUB;
;        2136      DM_EXER ();                               ! RUN DM EXERCISER SUBTEST
;        2137      ENDSUB;
;        2138
;        2139      END
;        2140  ELSE                                          ! OTHERWISE
;        2141      BEGIN
;        2142
;        2143      BGNSUB;
;        2144      MULTI_DRIVE ();                           ! RUN MULTI-DRIVE SUBTEST
;        2145      ENDSUB;
;        2146
;        2147      END;
;        2148
;        2149  DORPT;                                        ! PRINT STATISTICS
;        2150  EOP_FLAG = TRUE;                             ! SET END-OF-PASS FLAG
;        2151
;        2152  ENDTST;


                                .TITLE  CZRCD3 CZRCDA0 RC25 DISK EXERCISER
                                .IDENT  /V01.0/
                                .ENABL  AMA


000000                          .PSECT  $GGG$,  RO
000000                  COMM.AREA:
                                .BLKW   220
000440                  BST:    .BLKW   40
000540                  DPST:   .BLKW   10
000560                  ICOM.ADDR:
                                .BLKW   1
```

```
000562                                  IENV.ADDR:
                                                .BLKW   1
000564                                  ICST.ADDR:
                                                .BLKW   1
000566                                  IDCT.ADDR:
                                                .BLKW   1
000570  000000V                         INT.ADDR:
                                                .WORD   RCINT0
000572  000000V                                   .WORD   RCINT1
000574  000000V                                   .WORD   RCINT2
000576  000000V                                   .WORD   RCINT3
000600                                  ICTLR:  .BLKW   1
000602                                  MX1:    .BLKW   1
000604                                  MX2:    .BLKW   1
000606                                  MAD1:   .BLKW   1
000610                                  MAD2:   .BLKW   1
000612                                  VEC.AD: .BLKW   1
000614                                  BRLEVEL:.BLKW   1
000616                                  USIZE:  .BLKW   1
000620                                  DM.TIMR:.BLKW   4
000630                                  SWEEP.FLAG:
                                                .BLKB   1
                                                .EVEN
000632  000020                          RDM.CNT:.WORD   20
000634                                  RANDOM: .BLKW   20
000674  000001                          PAT02:  .WORD   1
000676  000000                                  .WORD   0
000700  000001                          PAT03:  .WORD   1
000702  177777                                  .WORD   -1
000704  000001                          PAT04:  .WORD   1
000706  105613                                  .WORD   -72165
000710  000001                          PAT05:  .WORD   1
000712  031463                                  .WORD   31463
000714  000001                          PAT06:  .WORD   1
000716  030221                                  .WORD   30221
000720  000020                          PAT07:  .WORD   20
000722  000001                                  .WORD   1
000724  000003                                  .WORD   3
000726  000007                                  .WORD   7
000730  000017                                  .WORD   17
000732  000037                                  .WORD   37
000734  000077                                  .WORD   77
000736  000177                                  .WORD   177
000740  000377                                  .WORD   377
000742  000777                                  .WORD   777
000744  001777                                  .WORD   1777
000746  003777                                  .WORD   3777
000750  007777                                  .WORD   7777
000752  017777                                  .WORD   17777
000754  037777                                  .WORD   37777
000756  077777                                  .WORD   77777
000760  177777                                  .WORD   -1
000762  000020                          PAT08:  .WORD   20
000764  177776                                  .WORD   -2
000766  177774                                  .WORD   -4
000770  177770                                  .WORD   -10
000772  177760                                  .WORD   -20
```

```
000774  177740                          .WORD   -40
000776  177700                          .WORD   -100
001000  177600                          .WORD   -200
001002  177400                          .WORD   -400
001004  177000                          .WORD   -1000
001006  176000                          .WORD   -2000
001010  174000                          .WORD   -4000
001012  170000                          .WORD   -10000
001014  160000                          .WORD   -20000
001016  140000                          .WORD   -40000
001020  100000                          .WORD   -100000
001022  000000                          .WORD   0
001024  000020                  PAT09:  .WORD   20
001026  000000                          .WORD   0
001030  000000                          .WORD   0
001032  000000                          .WORD   0
001034  177777                          .WORD   -1
001036  177777                          .WORD   -1
001040  177777                          .WORD   -1
001042  000000                          .WORD   0
001044  000000                          .WORD   0
001046  177777                          .WORD   -1
001050  177777                          .WORD   -1
001052  000000                          .WORD   0
001054  177777                          .WORD   -1
001056  000000                          .WORD   0
001060  177777                          .WORD   -1
001062  000000                          .WORD   0
001064  177777                          .WORD   -1
001066  000001                  PAT10:  .WORD   1
001070  133331                          .WORD   -44447
001072  000020                  PAT11:  .WORD   20
001074  052525                          .WORD   52525
001076  052525                          .WORD   52525
001100  052525                          .WORD   52525
001102  125252                          .WORD   -52526
001104  125252                          .WORD   -52526
001106  125252                          .WORD   -52526
001110  052525                          .WORD   52525
001112  052525                          .WORD   52525
001114  125252                          .WORD   -52526
001116  125252                          .WORD   -52526
001120  052525                          .WORD   52525
001122  125252                          .WORD   -52526
001124  052525                          .WORD   52525
001126  125252                          .WORD   -52526
001130  052525                          .WORD   52525
001132  125252                          .WORD   -52526
001134  000024                  PAT12:  .WORD   24
001136  026455                          .WORD   26455
001140  026455                          .WORD   26455
001142  026455                          .WORD   26455
001144  151322                          .WORD   -26456
001146  151322                          .WORD   -26456
001150  151322                          .WORD   -26456
001152  026455                          .WORD   26455
001154  026455                          .WORD   26455
```

```
001156  151322                          .WORD   -26456
001160  151322                          .WORD   -26456
001162  026455                          .WORD   26455
001164  026455                          .WORD   26455
001166  151322                          .WORD   -26456
001170  026455                          .WORD   26455
001172  151322                          .WORD   -26456
001174  026455                          .WORD   26455
001176  151322                          .WORD   -26456
001200  026455                          .WORD   26455
001202  151322                          .WORD   -26456
001204  026455                          .WORD   26455
001206  000001                  PAT13:  .WORD   1
001210  066666                          .WORD   66666
001212  000020                  PAT14:  .WORD   20
001214  000001                          .WORD   1
001216  000002                          .WORD   2
001220  000004                          .WORD   4
001222  000010                          .WORD   10
001224  000020                          .WORD   20
001226  000040                          .WORD   40
001230  000100                          .WORD   100
001232  000200                          .WORD   200
001234  000400                          .WORD   400
001236  001000                          .WORD   1000
001240  002000                          .WORD   2000
001242  004000                          .WORD   4000
001244  010000                          .WORD   10000
001246  020000                          .WORD   20000
001250  040000                          .WORD   40000
001252  100000                          .WORD   -100000
001254  000020                  PAT15:  .WORD   20
001256  177776                          .WORD   -2
001260  177775                          .WORD   -3
001262  177773                          .WORD   -5
001264  177767                          .WORD   -11
001266  177757                          .WORD   -21
001270  177737                          .WORD   -41
001272  177677                          .WORD   -101
001274  177577                          .WORD   -201
001276  177377                          .WORD   -401
001300  176777                          .WORD   -1001
001302  175777                          .WORD   -2001
001304  173777                          .WORD   -4001
001306  167777                          .WORD   -10001
001310  157777                          .WORD   -20001
001312  137777                          .WORD   -40001
001314  077777                          .WORD   77777
001316  000020                  PAT16:  .WORD   20
001320  133331                          .WORD   -44447
001322  133331                          .WORD   -44447
001324  133331                          .WORD   -44447
001326  155554                          .WORD   -22224
001330  155554                          .WORD   -22224
001332  155554                          .WORD   -22224
001334  133331                          .WORD   -44447
001336  133331                          .WORD   -44447
```

```
001340  155554                          .WORD   -22224
001342  155554                          .WORD   -22224
001344  133331                          .WORD   -44447
001346  155554                          .WORD   -22224
001350  133331                          .WORD   -44447
001352  155554                          .WORD   -22224
001354  133331                          .WORD   -44447
001356  155554                          .WORD   -22224
001360  000025                  PAT17:  .WORD   25
001362  000000                          .WORD   0
001364  106466                          .WORD   -71312
001366  106466                          .WORD   -71312
001370  071311                          .WORD   71311
001372  071311                          .WORD   71311
001374  071311                          .WORD   71311
001376  106466                          .WORD   -71312
001400  106466                          .WORD   -71312
001402  106466                          .WORD   -71312
001404  106466                          .WORD   -71312
001406  071311                          .WORD   71311
001410  071311                          .WORD   71311
001412  071311                          .WORD   71311
001414  071311                          .WORD   71311
001416  071311                          .WORD   71311
001420  106466                          .WORD   -71312
001422  106466                          .WORD   -71312
001424  106466                          .WORD   -71312
001426  106466                          .WORD   -71312
001430  106466                          .WORD   -71312
001432  106466                          .WORD   -71312
001434  000025                  PAT18:  .WORD   25
001436  106466                          .WORD   -71312
001440  000000                          .WORD   0
001442  071311                          .WORD   71311
001444  106466                          .WORD   -71312
001446  106466                          .WORD   -71312
001450  106466                          .WORD   -71312
001452  071311                          .WORD   71311
001454  071311                          .WORD   71311
001456  071311                          .WORD   71311
001460  071311                          .WORD   71311
001462  106466                          .WORD   -71312
001464  106466                          .WORD   -71312
001466  106466                          .WORD   -71312
001470  106466                          .WORD   -71312
001472  106466                          .WORD   -71312
001474  071311                          .WORD   71311
001476  071311                          .WORD   71311
001500  071311                          .WORD   71311
001502  071311                          .WORD   71311
001504  071311                          .WORD   71311
001506  071311                          .WORD   71311
001510  000025                  PAT19:  .WORD   25
001512  000000                          .WORD   0
001514  134631                          .WORD   -43147
001516  134631                          .WORD   -43147
001520  043146                          .WORD   43146
```

```
001522  043146                          .WORD   43146
001524  043146                          .WORD   43146
001526  134631                          .WORD   -43147
001530  134631                          .WORD   -43147
001532  134631                          .WORD   -43147
001534  134631                          .WORD   -43147
001536  043146                          .WORD   43146
001540  043146                          .WORD   43146
001542  043146                          .WORD   43146
001544  043146                          .WORD   43146
001546  043146                          .WORD   43146
001550  134631                          .WORD   -43147
001552  134631                          .WORD   -43147
001554  134631                          .WORD   -43147
001556  134631                          .WORD   -43147
001560  134631                          .WORD   -43147
001562  134631                          .WORD   -43147
001564  000025                  PAT20:  .WORD   25
001566  134631                          .WORD   -43147
001570  000000                          .WORD   0
001572  043146                          .WORD   43146
001574  134631                          .WORD   -43147
001576  134631                          .WORD   -43147
001600  134631                          .WORD   -43147
001602  043146                          .WORD   43146
001604  043146                          .WORD   43146
001606  043146                          .WORD   43146
001610  043146                          .WORD   43146
001612  134631                          .WORD   -43147
001614  134631                          .WORD   -43147
001616  134631                          .WORD   -43147
001620  134631                          .WORD   -43147
001622  134631                          .WORD   -43147
001624  043146                          .WORD   43146
001626  043146                          .WORD   43146
001630  043146                          .WORD   43146
001632  043146                          .WORD   43146
001634  043146                          .WORD   43146
001636  043146                          .WORD   43146
001640  000001                  PAT21:  .WORD   1
001642  000000                          .WORD   0
001644  000632'                 DPA.TBL:.WORD   RDM.CNT
001646  000674'                         .WORD   PAT02
001650  000700'                         .WORD   PAT03
001652  000704'                         .WORD   PAT04
001654  000710'                         .WORD   PAT05
001656  000714'                         .WORD   PAT06
001660  000720'                         .WORD   PAT07
001662  000762'                         .WORD   PAT08
001664  001024'                         .WORD   PAT09
001666  001066'                         .WORD   PAT10
001670  001072'                         .WORD   PAT11
001672  001134'                         .WORD   PAT12
001674  001206'                         .WORD   PAT13
001676  001212'                         .WORD   PAT14
001700  001254'                         .WORD   PAT15
001702  001316'                         .WORD   PAT16
```

```
001704  001360'                        .WORD   PAT17
001706  001434'                        .WORD   PAT18
001710  001510'                        .WORD   PAT19
001712  001564'                        .WORD   PAT20
001714  001640'                        .WORD   PAT21
001716  000140                CROMP:  .WORD   140
001720  000000                        .WORD   0
001722  000472                        .WORD   472
001724  000000                        .WORD   0
001726  051103                        .WORD   51103
001730  046517                        .WORD   46517
001732  020120                        .WORD   20120
001734  000001                        .WORD   1
001736  000013                        .WORD   13
001740  000000                        .WORD   0
001742  000000                        .WORD   0
001744  000000                        .WORD   0
001746  000000                        .WORD   0
001750  000000                        .WORD   0
001752  000000                        .WORD   0
001754  000000                        .WORD   0
001756  104206                        .WORD   -73572
001760  007760                        .WORD   7760
001762  002754                        .WORD   2754
001764  000000                        .WORD   0
001766  000000                        .WORD   0
001770  000000                        .WORD   0
001772  000000                        .WORD   0
001774  000000                        .WORD   0
001776  000000                        .WORD   0
002000  000235                        .WORD   235
002002  000000                        .WORD   0
002004  000000                        .WORD   0
002006  104204                        .WORD   -73574
002010  007774                        .WORD   7774
002012  104140                        .WORD   -73640
002014  002752                        .WORD   2752
002016  104204                        .WORD   -73574
002020  002751                        .WORD   2751
002022  104203                        .WORD   -73575
002024  007000                        .WORD   7000
002026  104647                        .WORD   -73131
002030  000001                        .WORD   1
002032  104641                        .WORD   -73137
002034  000002                        .WORD   2
002036  104142                        .WORD   -73636
002040  060020                        .WORD   60020
002042  102207                        .WORD   -75571
002044  000037                        .WORD   37
002046  052754                        .WORD   52754
002050  104077                        .WORD   -73701
002052  007006                        .WORD   7006
002054  032305                        .WORD   32305
002056  000000                        .WORD   0
002060  000000                        .WORD   0
002062  000000                        .WORD   0
002064  000000                        .WORD   0
```

```
002066  000000                          .WORD   0
002070  000000                          .WORD   0
002072  104207                          .WORD   -73571
002074  007000                          .WORD   7000
002076  104201                          .WORD   -73577
002100  000002                          .WORD   2
002102  104202                          .WORD   -73576
002104  002740                          .WORD   2740
002106  107027                          .WORD   -70751
002110  107017                          .WORD   -70761
002112  105012                          .WORD   -72766
002114  060011                          .WORD   60011
002116  027107                          .WORD   27107
002120  027203                          .WORD   27203
002122  027146                          .WORD   27146
002124  114000                          .WORD   -64000
002126  007005                          .WORD   7005
002130  114000                          .WORD   -64000
002132  007776                          .WORD   7776
002134  114000                          .WORD   -64000
002136  007776                          .WORD   7776
002140  104077                          .WORD   -73701
002142  104206                          .WORD   -73572
002144  007760                          .WORD   7760
002146  104301                          .WORD   -73477
002150  007005                          .WORD   7005
002152  104610                          .WORD   -73170
002154  007234                          .WORD   7234
002156  007767                          .WORD   7767
002160  115000                          .WORD   -63000
002162  007767                          .WORD   7767
002164  077076                          .WORD   77076
002166  027120                          .WORD   27120
002170  104200                          .WORD   -73600
002172  007051                          .WORD   7051
002174  002750                          .WORD   2750
002176  002740                          .WORD   2740
002200  027146                          .WORD   27146
002202  115000                          .WORD   -63000
002204  007265                          .WORD   7265
002206  057051                          .WORD   57051
002210  104200                          .WORD   -73600
002212  177777                          .WORD   -1
002214  007004                          .WORD   7004
002216  027130                          .WORD   27130
002220  027146                          .WORD   27146
002222  115000                          .WORD   -63000
002224  007265                          .WORD   7265
002226  017071                          .WORD   17071
002230  114000                          .WORD   -64000
002232  007004                          .WORD   7004
002234  027130                          .WORD   27130
002236  007051                          .WORD   7051
002240  027162                          .WORD   27162
002242  027203                          .WORD   27203
002244  114000                          .WORD   -64000
002246  007264                          .WORD   7264
```

```
002250  027216                          .WORD   27216
002252  115400                          .WORD   -62400
002254  007005                          .WORD   7005
002256  106200                          .WORD   -71600
002260  000003                          .WORD   3
002262  007005                          .WORD   7005
002264  037031                          .WORD   37031
002266  114000                          .WORD   -64000
002270  007005                          .WORD   7005
002272  007031                          .WORD   7031
002274  104207                          .WORD   -73571
002276  007002                          .WORD   7002
002300  104201                          .WORD   -73577
002302  000002                          .WORD   2
002304  060023                          .WORD   60023
002306  102207                          .WORD   -75571
002310  000037                          .WORD   37
002312  057107                          .WORD   57107
002314  000000                          .WORD   0
002316  104307                          .WORD   -73471
002320  007005                          .WORD   7005
002322  104201                          .WORD   -73577
002324  000005                          .WORD   5
002326  060031                          .WORD   60031
002330  104010                          .WORD   -73770
002332  007001                          .WORD   7001
002334  000000                          .WORD   0
002336  104307                          .WORD   -73471
002340  007002                          .WORD   7002
002342  105207                          .WORD   -72571
002344  000060                          .WORD   60
002346  114001                          .WORD   -63777
002350  104202                          .WORD   -73576
002352  000001                          .WORD   1
002354  104203                          .WORD   -73575
002356  007004                          .WORD   7004
002360  060021                          .WORD   60021
002362  102207                          .WORD   -75571
002364  000037                          .WORD   37
002366  057130                          .WORD   57130
002370  000000                          .WORD   0
002372  104307                          .WORD   -73471
002374  007002                          .WORD   7002
002376  114001                          .WORD   -63777
002400  104202                          .WORD   -73576
002402  000032                          .WORD   32
002404  104203                          .WORD   -73575
002406  007234                          .WORD   7234
002410  060020                          .WORD   60020
002412  102207                          .WORD   -75571
002414  000037                          .WORD   37
002416  057146                          .WORD   57146
002420  000000                          .WORD   0
002422  104200                          .WORD   -73600
002424  000004                          .WORD   4
002426  007000                          .WORD   7000
002430  104207                          .WORD   -73571
```

```
002432  002743                          .WORD   2743
002434  104301                          .WORD   -73477
002436  007001                          .WORD   7001
002440  104272                          .WORD   -73506
002442  105612                          .WORD   -72166
002444  007240                          .WORD   7240
002446  100612                          .WORD   -77166
002450  007240                          .WORD   7240
002452  115401                          .WORD   -62377
002454  117400                          .WORD   -60400
002456  007000                          .WORD   7000
002460  037171                          .WORD   37171
002462  000000                          .WORD   0
002464  114000                          .WORD   -64000
002466  002743                          .WORD   2743
002470  114000                          .WORD   -64000
002472  002744                          .WORD   2744
002474  114000                          .WORD   -64000
002476  002745                          .WORD   2745
002500  114000                          .WORD   -64000
002502  002746                          .WORD   2746
002504  114000                          .WORD   -64000
002506  002747                          .WORD   2747
002510  000000                          .WORD   0
002512  104307                          .WORD   -73471
002514  007002                          .WORD   7002
002516  105207                          .WORD   -72571
002520  000010                          .WORD   10
002522  114001                          .WORD   -63777
002524  104202                          .WORD   -73576
002526  000025                          .WORD   25
002530  104203                          .WORD   -73575
002532  007240                          .WORD   7240
002534  060021                          .WORD   60021
002536  102207                          .WORD   -75571
002540  000037                          .WORD   37
002542  057216                          .WORD   57216
002544  000000                          .WORD   0
002546  143367                          .WORD   -34411
002550  000000                          .WORD   0

                        .GLOBL  PATCH, CPT, CST, CST.ADDR, DCT
                        .GLOBL  DCT.ADDR, RC25.ADDR, IRC25.ADDR
                        .GLOBL  DM.COMM, DMC.ADDR, RP.SAVE, RPS.X1
                        .GLOBL  RPS.X2, OUTC.LIST, OUTC.TIMR, OCL.X1
                        .GLOBL  OCL.X2, TALLY, T.ADDR, MSCP.ENV
                        .GLOBL  ENV.USE, RETPKT, RP.USE, RP.INDX
                        .GLOBL  RP.ADDR, BUFF.DESC, BUFF.OWN, IODQ
                        .GLOBL  IODQ.IN, IODQ.OUT, ENTRY.REASON
                        .GLOBL  T.FLAG, EOP.FLAG, MEM.MGMT, IIP.FLAG
                        .GLOBL  CCTLR, CPLAT, CUOFF, CTLR.CNT
                        .GLOBL  DUR, QIO, MEM.SIZE, FREE.MEM.ADDR
                        .GLOBL  BUFF.SIZE, NUM.BUFF, CLK.TYPE
                        .GLOBL  CLK.HERTZ, CLK.CSR, CLK.VECTOR
                        .GLOBL  HOURS, MINUTES, SECONDS, TICKS
                        .GLOBL  ST.CODE, SB.CODE, STEP, OF.RC
```

```
                        .GLOBL  SA.REG, NEX, CRN, MSG.02, MSG.03
                        .GLOBL  MSG.04, MSG.05, MSG.06, MSG.08
                        .GLOBL  EGD.10, EGD.11, EGD.12, EGD.13
                        .GLOBL  EGD.14, EGD.15, EGD.16, EGD.17
                        .GLOBL  EGD.18, EGD.19, EGD.20, EGD.21
                        .GLOBL  EGD.22, EGH.30, STC.00, STC.01
                        .GLOBL  STC.02, STC.03, STC.04, STC.05
                        .GLOBL  STC.06, STC.07, STC.08, STC.09
                        .GLOBL  STC.10, STC.11, EX.CRN, EX.SB
                        .GLOBL  EX.EL, EX.PA, EX.FMT, EX.EVC, EX.HMA
                        .GLOBL  EX.03, SWP.STRACK, SWP.ETRACK
                        .GLOBL  SWP.FLAGS, SWP.DPAT, SWP.UCNT
                        .GLOBL  SWP.UDPAT, L$HIMEM, L$UNIT, L$LUN
                        .GLOBL  NEX.TRAP, COPY.BLK, SET.CPAR, SET.UPAR
                        .GLOBL  GET.ENV, PUT.ENV, GET.RETPKT, PUT.RETPKT
                        .GLOBL  GET.IO.BUFF, PUT.IO.BUFF, PUTA.BUFF
                        .GLOBL  OUT.IODQ, IN.IODQ, DROP.CTLR, DRV.CTLERR
                        .GLOBL  HARD.ERR, UPD.IOC, OVF.CHK, XFR.CHK
                        .GLOBL  SEND, WAIT, EMS.10, EMS.12, EMS.13
                        .GLOBL  EMS.14, EMS.15, EMS.16, EMS.17
                        .GLOBL  EMS.18, EMS.19, EMS.20, EMS.21
                        .GLOBL  EMS.22, EMS.30, EMS.42, EMS.43

100000                  BIT15==         -100000
040000                  BIT14==         40000
020000                  BIT13==         20000
010000                  BIT12==         10000
004000                  BIT11==         4000
002000                  BIT10==         2000
001000                  BIT09==         1000
000400                  BIT08==         400
000200                  BIT07==         200
000100                  BIT06==         100
000040                  BIT05==         40
000020                  BIT04==         20
000010                  BIT03==         10
000004                  BIT02==         4
000002                  BIT01==         2
000001                  BIT00==         1
001000                  BIT9==          1000
000400                  BIT8==          400
000200                  BIT7==          200
000100                  BIT6==          100
000040                  BIT5==          40
000020                  BIT4==          20
000010                  BIT3==          10
000004                  BIT2==          4
000002                  BIT1==          2
000001                  BIT0==          1
000040                  EF.START==      40
000037                  EF.RESTART==    37
000036                  EF.CONTINUE==   36
000035                  EF.NEW==        35
000034                  EF.PWR==        34
000340                  PRI07==         340
000300                  PRI06==         300
```

```
                000240          PRI05==         240
                000200          PRI04==         200
                000140          PRI03==         140
                000100          PRI02==         100
                000040          PRI01==         40
                000000          PRI00==         0
                000004          EVL==           4
                000010          LOT==           10
                000020          ADR==           20
                000040          IDU==           40
                000100          ISR==           100
                000200          UAM==           200
                000400          BOE==           400
                001000          PNT==           1000
                002000          PRI==           2000
                004000          IXE==           4000
                010000          IBE==           10000
                020000          IER==           20000
                040000          LOE==           40000
                100000          HOE==           -100000
                                .SBTTL  $T1 TEST SECTION
000000                          .PSECT  $CODE$,  RO

000000  105037  000000G       $T1:    CLRB    EOP.FLAG                ;                       2120
000004  123727  000000G 000005        CMPB    ENTRY.REASON,#5         ;                       2121
000012  001406                        BEQ     2$
000014  104402                1$:     TRAP    2                       ;                       2123
000016  004737  000000V               JSR     PC,INIT.TEST            ;                       2126
000022  104467                        TRAP    67
000024  006000                        ROR     R0
000026  103772                        BLO     1$
000030  132737  000002  000000G 2$:    BIT?    #2,SWP.FLAGS            ;                       2131
000036  001407                        BEQ     4$
000040  104402                3$:     TRAP    2                       ;                       2133
000042  004737  000000V               JSR     PC,DM.EXER              ;                       2136
000046  104467                        TRAP    67
000050  006000                        ROR     R0
000052  103007                        BCC     5$
000054  000771                        BR      3$
000056  104402                4$:     TRAP    2                       ;                       2141
000060  004737  000000V               JSR     PC,MULTI.DRIVE          ;                       2144
000064  104467                        TRAP    67
000066  006000                        ROR     R0
000070  103772                        BLO     4$
000072  104424                5$:     TRAP    24                      ;                       2147
000074  112737  000001  000000G       MOVB    #1,EOP.FLAG             ;                       2150
000102  000207                        RTS     PC                      ;                       2109
; Routine Size:  34 words,      Routine Base:  $CODE$ + 0000
; Maximum stack depth per invocation:  2 words
                                .SBTTL  T1 TEST SECTION
000000  004737  000000'         T1::
000000                          1$:     JSR     PC,$T1                  ;                       2150
000004  104466                        TRAP    66
000006  006000                        ROR     R0
000010  103773                        BLO     1$
000012  000207                        RTS     PC
; Routine Size:  6 words,       Routine Base:  $CODE$ + 0104
; Maximum stack depth per invocation:  2 words
```

```
;       2153  %SBTTL 'INITIALIZATION SUBTEST ROUTINES'
;       2154
;       2155  ROUTINE INIT_TEST : NOVALUE =
;       2156
;       2157  !+
;       2158  !       THE INITIALIZATION SUBTEST IS DESIGNED TO VERIFY THE EXISTENCE OF THE
;       2159  !       DEVICES AS CONFIGURED BY THE OPERATOR DURING THE HW DIALOG, AND TO
;       2160  !       BRING EACH DEVICE ONLINE IN PREPARATION FOR EITHER THE MULTI-DRIVE
;       2161  !       SUBTEST OR THE DM EXERCISER SUBTEST.
;       2162  !
;       2163  !       BASICALLY, THE DEVICES ARE BROUGHT ONLINE VIA "DRIVER_INIT", WHICH IS
;       2164  !       INVOKED IMMEDIATELY. ANY DEVICES WHICH FAIL DURING THIS PHASE WILL BE
;       2165  !       MARKED OFFLINE IN THEIR DCT AND CST. FOR THOSE DEVICES WHICH SURVIVE
;       2166  !       THE INITIALIZATION, THIS ROUTINE WILL ATTEMPT 1 OR 2 ACCESS COMMANDS TO
;       2167  !       EACH PLATTER VIA ROUTINE "ACCESS". THE INITIALIZATION SUBTEST IS DEEMED
;       2168  !       A SUCCESS IF A BLOCK ON THE INNER TRACK OF EACH PLATTER CAN BE
;       2169  !       ACCESSED.
;       2170  !-
;       2171
;       2172  BEGIN
;       2173
;       2174  IF MANUAL                                  ! IF ATTENDED
;       2175  THEN                                       ! THEN
;       2176      PRINTF (MSG_04);                       ! "INIT SUBTEST START"
;       2177  IIP_FLAG = TRUE;                          ! SET INIT-IN-PROGRESS FLAG
;       2178  DRIVER_INIT ();                           ! INIT DRIVER DATA AND DEVICES
;       2179  INCR CTLR FROM 0 TO (MAX_CTLR - 1) DO     ! FOR EACH CONTROLLER
;       2180      BEGIN
;       2181
;       2182      SET_CPAR (.CTLR);                      ! SET UP COMMONLY-USED CONTROLLER-RELATED DATA ITEMS
;       2183      IF .CST_ADDR [STATE] EQLU ONLINE       ! IF CONTROLLER IS STILL ALIVE
;       2184      THEN                                   ! THEN
;       2185          BEGIN
;       2186
;       2187          INCR OFFSET FROM (0 + OF_UN) TO (3 + OF_UN) DO  ! FOR EACH PLATTER
;       2188              BEGIN
;       2189
;       2190              IF ((.CST_ADDR [.OFFSET, P_PRES] EQLU PRESENT) AND
;       2191                  (.CST_ADDR [.OFFSET, P_STAT] EQLU ONLINE))
;       2192              THEN
;       2193                  BEGIN
;       2194
;       2195                  SET_UPAR (.OFFSET);                 ! SET UP UNIT-RELATED DATA ITEMS
;       2196                  ACCESS ();                          ! TRY ACCESS TO INNER TRACK
;       2197
;       2198                  END;                                ! IF UNIT IS PRESENT AND ONLINE
;       2199
;       2200              END;                                ! UNIT LOOP
;       2201
;       2202          END;                                ! IF CONTROLLER IS ONLINE
;       2203
;       2204      END;                                    ! CONTROLLER LOOP
;       2205
;       2206  IIP_FLAG = FALSE;                          ! CLEAR INIT-IN-PROGRESS FLAG
;       2207
;       2208  END;                                       ! ROUTINE INIT_TEST
```

```
                                        .SBTTL  INIT.TEST INITIALIZATION SUBTEST ROUTINES
000000  004137  000000G          INIT.TEST:
                                        JSR     R1,$SAVE2                       ;                       2155
000004  104450                          TRAP    50                              ;                       2174
000006  103007                          BHIS    1$
000010  012746  000000G                 MOV     #MSG.04,-(SP)                   ;                       2176
000014  012746  000001                  MOV     #1,-(SP)
000020  010600                          MOV     SP,R0                           ; SP,*
000022  104417                          TRAP    17
000024  022626                          CMP     (SP)+,(SP)+
000026  112737  000001  000000G  1$:     MOVB    #1,IIP.FLAG                     ;                       2177
000034  004737  000000V                  JSR     PC,DRIVER.INIT                  ;                       2178
000040  005002                          CLR     R2                              ; CTLR                  2179
000042  010246                  2$:     MOV     R2,-(SP)                        ; CTLR,*                2182
000044  004737  000000G                 JSR     PC,SET.CPAR
000050  013700  000000G                 MOV     CST.ADDR,R0                     ;                       2183
000054  005760  000002                  TST     2(R0)
000060  100025                          BPL     5$
000062  012701  000003                  MOV     #3,R1                           ; *,OFFSET              2187
000066  010100                  3$:     MOV     R1,R0                           ; OFFSET,*              2190
000070  006300                          ASL     R0
000072  063700  000000G                 ADD     CST.ADDR,R0
000076  032710  040000                  BIT     #40000,(R0)
000102  001410                          BEQ     4$
000104  032710  020000                  BIT     #20000,(R0)                     ;                       2191
000110  001405                          BEQ     4$
000112  010116                          MOV     R1,(SP)                         ; OFFSET,*              2195
000114  004737  000000G                 JSR     PC,SET.UPAR
000120  004737  000000V                  JSR     PC,ACCESS                       ;                       2196
000124  005201                  4$:     INC     R1                              ; OFFSET                2187
000126  020127  000006                  CMP     R1,#6                           ; OFFSET,*
000132  003755                          BLE     3$
000134  005726                  5$:     TST     (SP)+                           ;                       2180
000136  005202                          INC     R2                              ; CTLR                  2179
000140  020227  000003                  CMP     R2,#3                           ; CTLR,*
000144  003736                          BLE     2$
000146  105037  000000G                 CLRB    IIP.FLAG                        ;                       2206
000152  000207                          RTS     PC                              ;                       2155

; Routine Size:  54 words,      Routine Base:  $CODE$ + 0120
; Maximum stack depth per invocation:  7 words
```

```
2209  ROUTINE DRIVER_INIT : NOVALUE =
2210
2211  !+
2212  !       THIS ROUTINE IS EQUIVALENT IN FUNCTION TO THE INITIALIZATION ENTRY
2213  !       POINT OF A STANDARD DEVICE DRIVER. ITS RESPONSIBILITY IS TO INITIALIZE
2214  !       DRIVER DATA, AND TO BRING EACH RC25 CONTROLLER AND UNIT ONLINE.
2215  !-
2216
2217  BEGIN
2218
2219  LOCAL
2220      ENV_ADDR;
2221
2222  ENV_ADDR = MSCP_ENV + 8;                          ! ADDR (TEXT + 0) OF FIRST MSCP ENVELOPE
2223  INCR COUNT FROM 0 TO (ENV_CNT - 1) DO             ! FOR EACH MSCP ENVELOPE
2224      BEGIN
2225
2226      ENV_USE [.COUNT] = -1;                        ! MARK ENVELOPE FREE
2227      MSCP_ENV [.COUNT, ENV_LO] = .ENV_ADDR;        ! LOAD ENVELOPE ADDR INTO ENV DESCRIPTOR
2228      MSCP_ENV [.COUNT, ENV_HI] = 0;
2229      MSCP_ENV [.COUNT, ENV_F] = 1;                 ! SET FLAG BIT
2230      MSCP_ENV [.COUNT, ENV_O] = 1;                 ! SET OWNERSHIP BIT
2231      ENV_ADDR = .ENV_ADDR + (ENV_LEN * 2);         ! ADVANCE ADDR TO NEXT ENVELOPE
2232
2233      END;
2234
2235  INCR CTLR FROM 0 TO (MAX_CTLR - 1) DO             ! FOR EACH CONTROLLER
2236      BEGIN
2237
2238      IF .CST [.CTLR, IP_ADDR] NEQA 0               ! IF CONTROLLER IS PRESENT
2239      THEN                                          ! THEN
2240          BEGIN
2241
2242          SET_CPAR (.CTLR);                         ! SET UP CURRENT CONTROLLER PARAMETERS
2243          VEC_AD = .CST_ADDR [VEC_ADDR];            ! SET CURRENT CONTROLLER'S VECTOR ADDRESS
2244          BRLEVEL = .CST_ADDR [BR_LEV] ^ 5;         ! SET CURRENT CONTROLLER'S BR LEVEL
2245          CTLR_INIT ();                             ! INIT DEVICE AND CTLR DATA
2246          IF .DCT_ADDR [STAT] EQLU ONLINE           ! IF CONTROLLER IS STILL ALIVE
2247          THEN                                      ! THEN
2248              BEGIN
2249
2250              INCR OFFSET FROM (0 + OF_UN) TO (3 + OF_UN) DO      ! FOR EACH UNIT (PLATTER)
2251                  BEGIN
2252
2253                  IF .CST_ADDR [.OFFSET, P_PRES] EQLU PRESENT     ! IF UNIT EXISTS
2254                  THEN                                            ! THEN
2255                      BEGIN
2256
2257                      SET_UPAR (.OFFSET);                         ! SET UP UNIT-RELATED DATA ITEMS
2258                      UNIT_INIT ();                               ! BRING UNIT ONLINE
2259
2260                      END;                                        ! IF UNIT EXISTS
2261
2262                  END;                                        ! UNIT LOOP
2263
2264              END;                                        ! IF CONTROLLER IS STILL ALIVE
2265
```

```
;        2266          END;                                    ! IF CONTROLLER IS PRESENT
;        2267
;        2268      END;                                        ! CONTROLLER LOOP
;        2269
;        2270  END;                                            ! ROUTINE DRIVER_INIT


                                .SBTTL  DRIVER.INIT INITIALIZATION SUBTEST ROUTINES
000000  004137  000000G        DRIVER.INIT:
                                JSR     R1,$SAVE2                       ;                       2209
000004  012702  000010G                MOV     #MSCP.ENV+10,R2         ; *,ENV.ADDR            2222
000010  005001                        CLR     R1                      ; COUNT                 2223
000012  112761  000377  000000G 1$:   MOVB    #377,ENV.USE(R1)        ; *,*(COUNT)            2226
000020  010146                        MOV     R1,-(SP)                ; COUNT,*               2227
000022  012746  000104                MOV     #104,-(SP)
000026  004737  000000G                JSR     PC,BL$MUL
000032  010260  000000G                MOV     R2,MSCP.ENV(R0)         ; ENV.ADDR,*
000036  062700  000002G                ADD     #MSCP.ENV+2,R0          ;                       2228
000042  005010                        CLR     (R0)
000044  052710  140000                BIS     #140000,(R0)            ;                       2230
000050  062702  000104                ADD     #104,R2                 ; *,ENV.ADDR            2231
000054  022626                        CMP     (SP)+,(SP)+             ;                       2224
000056  005201                        INC     R1                      ; COUNT                 2223
000060  020127  000137                CMP     R1,#137                 ; COUNT,*
000064  003752                        BLE     1$
000066  005002                        CLR     R2                      ; CTLR                  2235
000070  010246                2$:     MOV     R2,-(SP)                ; CTLR,*                2238
000072  012746  000016                MOV     #16,-(SP)
000076  004737  000000G                JSR     PC,BL$MUL
000102  022626                        CMP     (SP)+,(SP)+
000104  005760  000000G                TST     CST(R0)
000110  001454                        BEQ     6$
000112  010246                        MOV     R2,-(SP)                ; CTLR,*                2242
000114  004737  000000G                JSR     PC,SET.CPAR
000120  013700  000000G                MOV     CST.ADDR,R0             ;                       2243
000124  016037  000002  000612'       MOV     2(R0),VEC.AD
000132  042737  177000  000612'       BIC     #177000,VEC.AD
000140  005016                        CLR     (SP)                    ;                       2244
000142  116016  000004                MOVB    4(R0),(SP)
000146  012746  000005                MOV     #5,-(SP)
000152  004737  000000G                JSR     PC,BL$SHF
000156  010037  000614'                MOV     R0,BRLEVEL
000162  004737  000000V                JSR     PC,CTLR.INIT            ;                       2245
000166  005777  000000G                TST     @DCT.ADDR               ;                       2246
000172  100022                        BPL     5$
000174  012701  000003                MOV     #3,R1                   ; *,OFFSET              2250
000200  010100                3$:     MOV     R1,R0                   ; OFFSET,*              2253
000202  006300                        ASL     R0
000204  063700  000000G                ADD     CST.ADDR,R0
000210  032710  040000                BIT     #40000,(R0)
000214  001405                        BEQ     4$
000216  010116                        MOV     R1,(SP)                 ; OFFSET,*              2257
000220  004737  000000G                JSR     PC,SET.UPAR
000224  004737  000000V                JSR     PC,UNIT.INIT            ;                       2258
000230  005201                4$:     INC     R1                      ; OFFSET                2250
000232  020127  000006                CMP     R1,#6                   ; OFFSET,*
000236  003760                        BLE     3$
```

```
000240  022626                          5$:     CMP     (SP)+,(SP)+                     ;                                       2240
000242  005202                          6$:     INC     R2                              ; CTLR                                  2235
000244  020227  000003                          CMP     R2,#3                           ; CTLR,*
000250  003707                                  BLE     2$
000252  000207                                  RTS     PC                              ;                                       2209

; Routine Size:  86 words,      Routine Base:  $CODE$ + 0274
; Maximum stack depth per invocation:  6 words
```

```
;       2271  ROUTINE CTLR_INIT : NOVALUE =
;       2272
;       2273  !+
;       2274  !       THIS "DRIVER" ROUTINE IS CALLED FROM DRIVER_INIT FOR EACH CONTROLLER
;       2275  !       CONFIGURED FOR TESTING. ITS GENERAL PURPOSE IS TO BRING THE RC25 ONLINE
;       2276  !       TO THE HOST. SPECIFICALLY, IT IS WRITTEN TO:
;       2277  !
;       2278  !       1. INITIALIZE DRIVER CONTROLLER DATA, INCLUDING THE DCT,
;       2279  !       2. SET UP THE DEVICE'S INTERRUPT VECTOR ADDRESS, (NOTE THAT RC25
;       2280  !               INTERRUPT PROCESSING RUNS AT LEVEL 5 FOR ALL RC25'S. THIS IS
;       2281  !               BASED ON THE ASSUMPTION THAT ALL RC25'S WILL HAVE A FIXED BR
;       2282  !               LEVEL OF 5 OR LESS. INTERRUPT PROCESSING CANNOT BE BROKEN WITH
;       2283  !               A HIGHER PRIORITY INTERRUPT FROM ANOTHER RC25.)
;       2284  !       3. PERFORM A REGISTER EXISTENCE TEST TO VERIFY THE DEVICE'S PRESENCE,
;       2285  !       4. PERFORM A VECTOR AND BR LEVEL TEST TO VERIFY THE DEVICE'S VECTOR
;       2286  !               ADDRESS AND INTERRUPT REQUEST LEVEL,
;       2287  !       5. DO A HARD INITIALIZATION (FOUR STEPS) ON THE DEVICE.
;       2288  !
;       2289  !       IF ANY OF THESE INITIAL TESTS FAIL, THEN ALL UNITS ASSOCIATED WITH THE
;       2290  !       DEVICE ARE DROPPED.
;       2291  !
;       2292  !       IMPLICIT INPUTS:
;       2293  !               CCTLR - CURRENT CONTROLLER NUMBER
;       2294  !               VEC_AD - ASSUMED VECTOR ADDRESS OF THE CURRENT CONTROLLER
;       2295  !               DCT_ADDR - ADDRESS OF CURRENT CONTROLLER'S DCT
;       2296  !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;       2297  !               RC25_ADDR - ADDRESS OF CURRENT CONTROLLER'S IP REGISTER
;       2298  !-
;       2299
;       2300  BEGIN
;       2301
;       2302  LOCAL
;       2303      RESULT : WORD;
;       2304
;       2305  INI_CTLR_DAT ();                          ! INITIALIZE CONTROLLER DATA
;       2306  SETVEC (.VEC_AD, .INT_ADDR [.CCTLR], PRI05);  ! SET DEVICE'S ASSUMED VECTOR ADDRESS
;       2307  DCT_ADDR [IG_INT] = YES;                  ! SET "IGNORE INTERRUPT" BIT
;       2308  L$LUN = .CST_ADDR [OF_UN, P_UNIT];        ! GET FIRST UNIT NUMBER OF CONTROLLER
;       2309                                           ! (USED BY DRS FOR ERROR REPORTING)
;       2310  IF REG_EXIST () EQLU FAILURE              ! REGISTER EXISTENCE TEST. IF FAILURE
;       2311  THEN                                     ! THEN
;       2312      BEGIN
;       2313
;       2314      DROP_CTLR (.CCTLR, DU_INIT);         ! DROP ALL CONTROLLER'S UNITS
;       2315      RETURN;
;       2316
;       2317      END;
;       2318
;       2319  IF VEC_BR_TEST () EQLU FAILURE           ! VECTOR ADDR AND BR LEVEL TEST. IF FAILURE
;       2320  THEN                                     ! THEN
;       2321      BEGIN
;       2322
;       2323      DROP_CTLR (.CCTLR, DU_INIT);         ! DROP ALL CONTROLLER'S UNITS
;       2324      RETURN;
;       2325
;       2326      END;
;       2327
```

```
;        2328  RESULT = HARD_INIT ();                    ! ATTEMPT HARD DEVICE INIT
;        2329  DCT_ADDR [IG_INT] = NO;                   ! CLEAR "IGNORE INTERRUPT" BIT
;        2330  IF .RESULT EQLU SUCCESS                   ! IF HARD INIT WAS SUCCESSFUL
;        2331  THEN                                      ! THEN
;        2332      BEGIN
;        2333
;        2334      DCT_ADDR [STAT] = ONLINE;            ! MARK CONTROLLER ONLINE IN "DRIVER"
;        2335      CST_ADDR [STATE] = ONLINE;           ! MARK CONTROLLER ONLINE IN "PROGRAM"
;        2336      INI_RRING ();                        ! INITIALIZE RESPONSE RING
;        2337      WRT_RC25 (RCSA, RC_ALL, SA_GO);      ! SET "GO" BIT (START CTLR POLLING)
;        2338      SET_CTLR_CHAR ();                    ! SET CONTROLLER CHARACTERISTICS
;        2339
;        2340      END
;        2341
;        2342  ELSE                                     ! HARD INIT FAILED
;        2343      DROP_CTLR (.CCTLR, DU_INIT);         ! DROP ALL CONTROLLER'S UNITS
;        2344
;        2345  END;                                     ! ROUTINE CTLR_INIT


                                        .SBTTL  CTLR.INIT INITIALIZATION SUBTEST ROUTINES
000000  010146                  CTLR.INIT:
                                        MOV     R1,-(SP)                        ;                       2271
000002  004737  000000V                 JSR     PC,INI.CTLR.DAT                 ;                       2305
000006  012746  000240                  MOV     #240,-(SP)                      ;                       2306
000012  013700  000000G                 MOV     CCTLR,R0
000016  006300                          ASL     R0
000020  016046  000570'                 MOV     INT.ADDR(R0),-(SP)
000024  013746  000612'                 MOV     VEC.AD,-(SP)
000030  012746  000003                  MOV     #3,-(SP)
000034  104437                          TRAP    37
000036  052777  040000  000000G        BIS     #40000,@DCT.ADDR                ;                       2307
000044  013700  000000G                 MOV     CST.ADDR,R0                     ;                       2308
000050  016001  000006                  MOV     6(R0),R1
000054  000301                          SWAB    R1
000056  042701  177740                  BIC     #177740,R1
000062  010137  000000G                 MOV     R1,L$LUN
000066  004737  000000V                 JSR     PC,REG.EXIST                    ;                       2310
000072  005700                          TST     R0
000074  001404                          BEQ     1$                              ;                       2314
000076  004737  000000V                 JSR     PC,VEC.BR.TEST                  ;                       2319
000102  005700                          TST     R0
000104  001011                          BNE     2$
000106  013716  000000G          1$:    MOV     CCTLR,(SP)                      ;                       2323
000112  012746  000002                  MOV     #2,-(SP)
000116  004737  000000G                 JSR     PC,DROP.CTLR
000122  062706  000012                  ADD     #12,SP                          ;                       2319
000126  000444                          BR      5$                              ;                       2321
000130  004737  000000V          2$:    JSR     PC,HARD.INIT                    ;                       2328
000134  042777  040000  000000G        BIC     #40000,@DCT.ADDR                ;                       2329
000142  020027  000001                  CMP     R0,#1                           ; RESULT,*              2330
000146  001023                          BNE     3$
000150  052777  100000  000000G        BIS     #100000,@DCT.ADDR               ;                       2334
000156  013700  000000G                 MOV     CST.ADDR,R0                     ;                       2335
000162  052760  100000  000002        BIS     #100000,2(R0)
000170  004737  000000V                 JSR     PC,INI.RRING                    ;                       2336
000174  012701  000001                  MOV     #1,R1                           ; *,RC.REG              2337
```

```
000200  013700  000000G                       MOV     RC25.ADDR,R0
000204  010160  000002                        MOV     R1,2(R0)                ; RC.REG,*
000210  004737  000000V                       JSR     PC,SET.CTLR.CHAR        ;                       2338
000214  000407                               BR      4$                      ;                       2330
000216  013716  000000G              3$:      MOV     CCTLR,(SP)              ;                       2343
000222  012746  000002                        MOV     #2,-(SP)
000226  004737  000000G                       JSR     PC,DROP.CTLR
000232  005726                               TST     (SP)+
000234  062706  000010               4$:      ADD     #10,SP                  ;                       2300
000240  012601                       5$:      MOV     (SP)+,R1                ;                       2271
000242  000207                               RTS     PC

; Routine Size:  82 words,      Routine Base:  $CODE$ + 0550
; Maximum stack depth per invocation:  7 words
```

```
;          2346  ROUTINE INI_CTLR_DAT : NOVALUE =
;          2347
;          2348  !+
;          2349  !       THIS ROUTINE IS RESPONSIBLE FOR INITIALIZING ALL CONTROLLER-RELATED
;          2350  !       DATA IN THE "DRIVER" PORTION OF THE EXERCISER. THIS INCLUDES THE
;          2351  !       CONTROLLER'S DCT AND OUTSTANDING COMMAND LIST.
;          2352  !
;          2353  !       IMPLICIT INPUTS:
;          2354  !               CCTLR - CURRENT CONTROLLER NUMBER
;          2355  !               DCT_ADDR - ADDRESS OF CURRENT CONTROLLER'S DCT
;          2356  !               OCL_X1, OCL_X2 - STARTING AND ENDING INDECES OF CURRENT
;          2357  !                                       CONTROLLER'S OUTSTANDING COMMAND AREA
;          2358  !-
;          2359
;          2360  BEGIN
;          2361
;          2362  INCR INDEX FROM .OCL_X1 TO .OCL_X2 DO
;          2363      BEGIN
;          2364
;          2365      OUTC_LIST [.INDEX] = -1;                  ! INIT OUTSTANDING CMD LIST
;          2366      OUTC_TIMR [.INDEX] = 0;                   ! ZERO OUT OUTSTANDING CMD TIMERS
;          2367
;          2368      END;
;          2369
;          2370  DCT_ADDR [WORD0] = 0;                                     ! CLEAR FIRST DCT WORD
;          2371  DCT_ADDR [RR_BEG] = COMM_AREA + 8 + (.CCTLR * COMM_LEN * 2);   ! START OF RESPONSE RING
;          2372  DCT_ADDR [RR_END] = .DCT_ADDR [RR_BEG] + ((RRING_LEN - 1) * 4); ! LAST SLOT IN RESPONSE RING
;          2373  DCT_ADDR [CR_BEG] = .DCT_ADDR [RR_END] + 4;                ! START OF COMMAND RING
;          2374  DCT_ADDR [CR_END] = .DCT_ADDR [CR_BEG] + ((CRING_LEN - 1) * 4); ! LAST SLOT IN COMMAND RING
;          2375  DCT_ADDR [RR_POLL] = .DCT_ADDR [RR_BEG];               ! FIRST RRING SLOT TO POLL
;          2376  DCT_ADDR [CR_POLL] = DCT_ADDR [CR_NEXT] = .DCT_ADDR [CR_BEG];  ! CRING POLL AND NEXT COMMAND POINTERS
;          2377
;          2378  END;


                                        .SBTTL  INI.CTLR.DAT INITIALIZATION SUBTEST ROUTINES
000000  004137  000000G         INI.CTLR.DAT:
                                        JSR     R1,$SAVE2                       ;                               2346
000004  013701  000000G                 MOV     OCL.X1,R1                       ; *,INDEX                       2362
000010  005301                          DEC     R1                              ; INDEX
000012  000407                          BR      2$
000014  112761  000377  000000G 1$:     MOVB    #377,OUTC.LIST(R1)              ; *,*(INDEX)                    2365
000022  010100                          MOV     R1,R0                           ; INDEX,*                       2366
000024  006300                          ASL     R0
000026  005060  000000G                 CLR     OUTC.TIMR(R0)
000032  005201                  2$:     INC     R1                              ; INDEX                         2362
000034  020137  000000G                 CMP     R1,OCL.X2                       ; INDEX,*
000040  003765                          BLE     1$
000042  013701  000000G                 MOV     DCT.ADDR,R1                     ;                               2370
000046  005011                          CLR     (R1)
000050  012702  000004                  MOV     #4,R2                           ;                               2371
000054  060102                          ADD     R1,R2
000056  013746  000000G                 MOV     CCTLR,-(SP)
000062  012746  000110                  MOV     #110,-(SP)
000066  004737  000000G                 JSR     PC,BL$MUL
000072  062700  000010'                 ADD     #COMM.AREA+10,R0
000076  010012                          MOV     R0,(R2)
```

```
000100  010061  000006                    MOV     R0,6(R1)                        ;                                       2372
000104  062761  000034  000006            ADD     #34,6(R1)
000112  012700  000010                    MOV     #10,R0                          ;                                       2373
000116  060100                            ADD     R1,R0
000120  016110  000006                    MOV     6(R1),(R0)
000124  062710  000004                    ADD     #4,(R0)
000130  011061  000012                    MOV     (R0),12(R1)                     ;                                       2374
000134  062761  000034  000012            ADD     #34,12(R1)
000142  011261  000014                    MOV     (R2),14(R1)                     ;                                       2375
000146  011061  000020                    MOV     (R0),20(R1)                     ;                                       2376
000152  011061  000016                    MOV     (R0),16(R1)
000156  022626                            CMP     (SP)+,(SP)+                     ;                                       2360
000160  000207                            RTS     PC                              ;                                       2346

; Routine Size:  57 words,      Routine Base:  $CODE$ + 1014
; Maximum stack depth per invocation:  6 words
```

```
;        2379  ROUTINE REG_EXIST =
;        2380
;        2381  !+
;        2382  !       THIS IS THE REGISTER EXISTENCE (OR "PROBE") TEST DESIGNED TO VERIFY
;        2383  !       THE PRESENCE OF AN RC25 DEVICE. THIS OBJECTIVE IS ACCOMPLISHED BY
;        2384  !       SETTING UP THE NON-EXISTENT MEMORY (NEX) TRAP VECTOR (LOCATION 4) AND
;        2385  !       ATTEMPTING TO READ WHAT IS ASSUMED TO BE THE DEVICE'S SA AND IP
;        2386  !       REGISTERS. IF THE NEX TRAP HANDLER IS INVOKED DUE TO AN ABSENT DEVICE,
;        2387  !       THEN THE GLOBAL DATUM "NEX" WILL BE SET TO "TRUE". THIS DATUM
;        2388  !       DETERMINES THE SUCCESS / FAILURE VALUE OF THIS ROUTINE.
;        2389  !
;        2390  !       IMPLICIT INPUTS:
;        2391  !               RC25_ADDR - ADDRESS OF CURRENT CONTROLLER'S IP REGISTER
;        2392  !-
;        2393
;        2394  BEGIN
;        2395
;        2396  LOCAL
;        2397      TEMP : WORD,                          ! TEMP FOR READING SA AND IP
;        2398      DUMMY : WORD;                         ! AS THE NAME IMPLIES
;        2399
;        2400  OF_RC = 2;                               ! SET UP TO READ SA FIRST
;        2401  DO
;        2402      BEGIN
;        2403
;        2404      NEX = FALSE;                          ! SET TO "TRAP NOT RECEIVED"
;        2405      SETVEC (4, NEX_TRAP, PRI07);          ! SET LOCATION 4 TRAP VECTOR ADDRESS
;        2406      TEMP = .(.RC25_ADDR + .OF_RC);        ! READ REGISTER (THEN TRAP OR CONTINUE)
;        2407      DUMMY = 0;                            ! DUMMY INSTRUCTION TO COVER TRAP RETURN BUG
;        2408                                            !       (TRAP RETURNS TO NEXT INSTRUCTION)
;        2409      CLRVEC (4);                           ! CLEAR LOCATION 4 TRAP VECTOR ADDRESS
;        2410      IF .NEX EQLU TRUE                     ! IF NEX TRAP OCCURRED
;        2411      THEN                                  ! THEN
;        2412          BEGIN
;        2413
;        2414          ERRDF (10, EGD_10, EMS_10);       ! "REGISTER EXISTENCE TEST FAILED"
;        2415          RETURN FAILURE;
;        2416
;        2417          END
;        2418      ELSE
;        2419          OF_RC = .OF_RC - 2;               ! SET UP FOR IP REG OR QUIT
;        2420
;        2421      END
;        2422
;        2423  UNTIL .OF_RC LSS 0;
;        2424
;        2425  RETURN SUCCESS;
;        2426
;        2427  END;

                                        .SBTTL  REG.EXIST INITIALIZATION SUBTEST ROUTINES
000000  004137  000000G         REG.EXIST:
                                        JSR     R1,$SAVE2                       ;2379
000004  012737  000002  000000G         MOV     #2,OF.RC                        ;2400
000012  005037  000000G         1$:     CLR     NEX                             ;2404
000016  012746  000340                  MOV     #340,-(SP)                      ;2405
```

```
000022  012746  000000G                         MOV     #NEX.TRAP,-(SP)
000026  012746  000004                          MOV     #4,-(SP)
000032  012746  000003                          MOV     #3,-(SP)
000036  104437                                 TRAP    37
000040  013700  000000G                         MOV     RC25.ADDR,R0                    ;                               2406
000044  063700  C00000G                         ADD     OF.RC,R0
000050  011001                                 MOV     (R0),R1                         ; *,TEMP
000052  005002                                 CLR     R2                              ; DUMMY                         2407
000054  012700  000004                          MOV     #4,R0                           ;                               2409
000060  104436                                 TRAP    36
000062  023727  000000G 000001                 CMP     NEX,#1                          ;                               2410
000070  001007                                 BNE     2$
000072  104455                                 TRAP    55                              ;                               2414
000074  000012                                 .WORD   12
000076  000000G                                .WORD   EGD.10
000100  000000G                                .WORD   EMS.10
000102  062706  000010                          ADD     #10,SP                          ;                               2410
000106  000413                                 BR      3$                              ;                               2412
000110  162737  000002  000000G       2$:       SUB     #2,OF.RC                        ;                               2419
000116  062706  000010                          ADD     #10,SP                          ;                               2402
000122  005737  000000G                         TST     OF.RC                           ;                               2423
000126  002331                                 BGE     1$
000130  012700  000001                          MOV     #1,R0                           ;                               2394
000134  000207                                 RTS     PC
000136  005000                       3$:       CLR     R0                              ;                               2379
000140  000207                                 RTS     PC

; Routine Size:  49 words,      Routine Base:  $CODE$ + 1176
; Maximum stack depth per invocation:  9 words
```

```
;          2428  ROUTINE VEC_BR_TEST =
;          2429
;          2430  !+
;          2431  !       THIS ROUTINE ATTEMPTS TO VERIFY (A) THAT THE RC25 VECTOR ADDRESS GIVEN
;          2432  !       BY THE USER DURING THE HW DIALOG IS VALID, AND (B) THAT THE
;          2433  !       USER-SPECIFIED BUS REQUEST LEVEL FOR THE DEVICE IS CORRECT. THE FIRST
;          2434  !       OBJECTIVE IS ACCOMPLISHED BY SETTING THE CPU PRIORITY TO 0 AND FORCING
;          2435  !       AN RC25 INTERRUPT. IF THE USER SPECIFIED AN INCORRECT VECTOR ADDRESS,
;          2436  !       THEN THE RESULT MAY BE UNPREDICTABLE. FOR THIS REASON, THE MESSAGE
;          2437  !       "ABOUT TO VERIFY VECTOR XXX(O) FOR DEVICE XXXXXX(O) ..." IS PRINTED
;          2438  !       BEFORE THE TEST, AND "COMPLETED" IS PRINTED AT ITS SUCCESSFUL
;          2439  !       CONCLUSION. IF THE WORD "COMPLETED" DOES NOT APPEAR, AND AN ERROR IS
;          2440  !       NOT REPORTED, THEN PROGRAM CONTROL IS ASSUMED LOST AND A FATAL TRAP HAS
;          2441  !       LIKELY OCCURRED. AT THIS POINT, THE EXERCISER MUST BE STARTED AGAIN.
;          2442  !
;          2443  !       IF THIS TEST SUCCEEDS, THEN THE BR LEVEL TEST IS RUN BY SETTING THE
;          2444  !       PROCESSOR PRIORITY TO THE ASSUMED INTERRUPT PRIORITY GIVEN BY THE
;          2445  !       USER. A FORCED INTERRUPT SHOULD NOT OCCUR. THEN, BY LOWERING THE
;          2446  !       PRIORITY BY ONE, THE DELAYED INTERRUPT SHOULD OCCUR.
;          2447  !               RC25_ADDR - ADDRESS OF CURRENT CONTROLLER'S IP REGISTER
;          2448  !
;          2449  !       IMPLICIT INPUTS:
;          2450  !               DCT_ADDR - ADDRESS OF CURRENT CONTROLLER'S DCT
;          2451  !               RC25_ADDR - ADDRESS OF CURRENT CONTROLLER'S IP REGISTER
;          2452  !               VEC_AD - ASSUMED VECTOR ADDRESS OF THE CURRENT CONTROLLER
;          2453  !               BRLEVEL - ASSUMED BUS REQUEST INTERRUPT LEVEL OF THE CURRENT
;          2454  !                       CONTROLLER
;          2455  !-
;          2456
;          2457  BEGIN
;          2458
;          2459  IF MANUAL                                    ! IF ATTENDED
;          2460  THEN                                         ! THEN
;          2461      PRINTF (MSG_02, .VEC_AD, .RC25_ADDR);    ! "ABOUT TO VERIFY VECTOR..."
;          2462  IF INT_GEN () EQLU FALSE                     ! FORCE AN INTERRUPT
;          2463  THEN                                         ! IF INTERRUPT DID NOT OCCUR
;          2464      BEGIN
;          2465
;          2466      ERRDF (11, EGD_11, 0);                   ! "VECTOR TEST FAILED"
;          2467      RETURN FAILURE;
;          2468
;          2469      END
;          2470  ELSE                                         ! INTERRUPT DID OCCUR
;          2471      BEGIN
;          2472
;          2473      IF MANUAL                                ! IF ATTENDED
;          2474      THEN                                     ! THEN
;          2475          PRINTF (MSG_03);                     ! "COMPLETED."
;          2476      SETPRI (.BRLEVEL);                       ! SET PRIORITY TO ASSUMED BR LEVEL
;          2477      IF INT_GEN () EQLU FALSE                 ! FORCE AN INTERRUPT (SHOULD NOT OCCUR)
;          2478      THEN                                     ! IF INTERRUPT DID NOT OCCUR
;          2479          BEGIN
;          2480
;          2481          SETPRI (.BRLEVEL - %O'40');          ! LOWER PRIORITY BY 1
;          2482          DELAY (1);                           ! WAIT
;          2483          IF .DCT_ADDR [SA_SAVE] NEQU 0        ! IF INTERRUPT DID OCCUR (SA_SAVE WOULD BE NON-ZERO)
;          2484          THEN                                 ! THEN
```

```
:          2485                BEGIN
:          2486
:          2487                SETPRI (PRI00);             ! RESTORE PROCESSOR PRIORITY TO 0
:          2488                RETURN SUCCESS;             ! ONLY SUCCESSFUL EXIT POINT
:          2489
:          2490                END;
:          2491
:          2492            END;
:          2493
:          2494        END;
:          2495
:          2496    SETPRI (PRI00);                         ! COME HERE ONLY FOR BR TEST FAILURE
:          2497    ERRDF (12, EGD_12, EMS_12);             ! 'BR LEVEL TEST FAILED'
:          2498    RETURN FAILURE;
:          2499
:          2500    END;


                                        .GLOBL  L$DLY


                                        .SBTTL  VEC.BR.TEST INITIALIZATION SUBTEST ROUTINES
000000  010146                  VEC.BR.TEST:
                                        MOV     R1,-(SP)                ;                       2428
000002  005746                          TST     -(SP)
000004  104450                          TRAP    50                      ;                       2459
000006  103014                          BHIS    1$
000010  013746  000000G                 MOV     RC25.ADDR,-(SP)         ;                       2461
000014  013746  000612'                 MOV     VEC.AD,-(SP)
000020  012746  000000G                 MOV     #MSG.02,-(SP)
000024  012746  000003                  MOV     #3,-(SP)
000030  010600                          MOV     SP,R0                   ; SP,*
000032  104417                          TRAP    17
000034  062706  000010                  ADD     #10,SP
000040  004737  000000V          1$:     JSR     PC,INT.GEN              ;                       2462
000044  005700                          TST     R0
000046  001005                          BNE     2$
000050  104455                          TRAP    55                      ;                       2466
000052  000013                          .WORD   13
000054  000000G                         .WORD   EGD.11
000056  000000                          .WORD   0
000060  000460                          BR      9$                      ;                       2464
000062  104450                  2$:     TRAP    50                      ;                       2473
000064  103007                          BHIS    3$
000066  012746  000000G                 MOV     #MSG.03,-(SP)           ;                       2475
000072  012746  000001                  MOV     #1,-(SP)
000076  010600                          MOV     SP,R0                   ; SP,*
000100  104417                          TRAP    17
000102  022626                          CMP     (SP)+,(SP)+
000104  013700  000614'          3$:     MOV     BRLEVEL,R0              ;                       2476
000110  104441                          TRAP    41
000112  004737  000000V                 JSR     PC,INT.GEN              :                       2477
000116  005700                          TST     R0
000120  001032                          BNE     8$
000122  013700  000614'                 MOV     BRLEVEL,R0              ;                       2481
000126  162700  000040                  SUB     #40,R0
000132  104441                          TRAP    41
```

```
000134  012701  000001                        MOV     #1,R1                   ; *,$$TMP2                      2482
000140  001410                          4$:    BEQ     7$
000142  013700  000000G                       MOV     L$DLY,R0                ; *,$$TMP1
000146  001403                                BEQ     6$
000150  005016                          5$:    CLR     (SP)                    ; $$TMP
000152  005300                                DEC     R0                      ; $$TMP1
000154  001375                                BNE     5$
000156  005301                          6$:    DEC     R1                      ; $$TMP2
000160  000767                                BR      4$
000162  013700  000000G                 7$:    MOV     DCT.ADDR,R0             ;                               2483
000166  005760  000002                        TST     2(R0)
000172  001405                                BEQ     8$
000174  005000                                CLR     R0                      ;                               2487
000176  104441                                TRAP    41
000200  012700  000001                        MOV     #1,R0                   ;                               2485
000204  000407                                BR      10$
000206  005000                          8$:    CLR     R0                      ;                               2496
000210  104441                                TRAP    41
000212  104455                                TRAP    55                      ;                               2497
000214  000014                                .WORD   14
000216  000000G                               .WORD   EGD.12
000220  000000G                               .WORD   EMS.12
000222  005000                          9$:    CLR     R0                      ;                               2428
000224  005726                          10$:   TST     (SP)+
000226  012601                                MOV     (SP)+,R1
000230  000207                                RTS     PC

; Routine Size:  77 words,      Routine Base:  $CODE$ + 1340
; Maximum stack depth per invocation:  8 words
```

```
;         2501  ROUTINE INT_GEN =
;         2502
;         2503  !+
;         2504  !       THIS ROUTINE BEGINS AN RC25 INITIALIZATION SEQUENCE, BUT ONLY
;         2505  !       COMPLETES THROUGH THE STEP 1 WRITE. ITS PURPOSE IS TO CREATE AN RC25
;         2506  !       INTERRUPT (AT THE COMPLETION OF STEP 1) IN ORDER TO HELP VERIFY THE
;         2507  !       THE USER-SPECIFIED VECTOR ADDRESS AND BUS REQUEST INTERRUPT LEVEL.
;         2508  !       A VALUE OF "TRUE" IS RETURNED TO THE CALLER IF AN INTERRUPT OCCURS,
;         2509  !       AND "FALSE" OTHERWISE. THE INTERRUPT IS VERIFIED BY A NON-ZERO VALUE
;         2510  !       IN THE "SA SAVE" WORD IN THE DEVICE'S DCT.
;         2511  !
;         2512  !       IMPLICIT INPUTS:
;         2513  !               DCT_ADDR - ADDRESS OF CURRENT CONTROLLER'S DCT
;         2514  !               RC25_ADDR - ADDRESS OF CURRENT CONTROLLER'S IP REGISTER
;         2515  !-
;         2516
;         2517  BEGIN
;         2518
;         2519  LOCAL
;         2520      IE_VEC : WORD,                      ! IE-BIT-AND-VECTOR-ADDRESS/4 BYTE
;         2521      SA : WORD;                          ! STORAGE FOR STEP 1 READ AND WRITE
;         2522
;         2523  DCT_ADDR [SA_SAVE] = 0;                ! ZERO OUT SA SAVE WORD IN DCT
;         2524  IE_VEC = (.VEC_AD ^ -2) OR SA_INT;     ! VEC ADDR / 4 WITH IE = 1
;         2525  WRT_RC25 (RCIP, RC_ALL, ALL_ONES);     ! WRITE IP TO START INIT SEQUENCE
;         2526  DELAY (500);                           ! WAIT
;         2527  SA = .RC25_ADDR [RCSA, RC_ALL];        ! STEP 1 READ
;         2528  SA = (WR_RING ^ 8) OR .IE_VEC;         ! STEP 1 WRITE VALUE
;         2529  WRT_RC25 (RCSA, RC_ALL, .SA);         ! STEP 1 WRITE
;         2530  INCR COUNT FROM 1 TO 10 DO             ! TEN SECOND LIMIT
;         2531      BEGIN
;         2532
;         2533      DELAY (333);                      ! ABOUT 1 SECOND
;         2534      SA = .RC25_ADDR [RCSA, RC_ALL];   ! READ SA REGISTER
;         2535      IF BIT_TST (SA, SA_S2)            ! IF STEP 2 HAS BEGUN
;         2536      THEN                              ! THEN
;         2537          EXITLOOP;                     ! BREAK OUT
;         2538
;         2539      END;
;         2540
;         2541  IF .DCT_ADDR [SA_SAVE] EQLU 0         ! IF SA SAVE WORD WAS NOT TOUCHED
;         2542  THEN                                  ! THEN
;         2543      RETURN FALSE                      ! INTERRUPT DID NOT OCCUR
;         2544  ELSE                                  ! OTHERWISE
;         2545      RETURN TRUE;                      ! INTERRUPT DID OCCUR
;         2546
;         2547  END;


                                        .SBTTL  INT.GEN INITIALIZATION SUBTEST ROUTINES
000000  004137  000000G         INT.GEN:JSR     R1,$SAVE5                       ;       2501
000004  162706  000006                  SUB     #6,SP
000010  013704  000000G                 MOV     DCT.ADDR,R4                     ;       2523
000014  005064  000002                  CLR     2(R4)
000020  013700  000612'                 MOV     VEC.AD,R0                       ;       2524
000024  006200                          ASR     R0
000026  006200                          ASR     R0
```

```
000030  010003                          MOV     R0,R3                   ; *,IE.VEC
000032  052703  000200                  BIS     #200,R3                 ; *,IE.VEC
000036  012700  177777                  MOV     #-1,R0                  ; *,RC.REG              2525
000042  010077  000000G                 MOV     R0,@RC25.ADDR           ; RC.REG,*
000046  012701  000764                  MOV     #764,R1                 ; *,$$TMP2              2526
000052  001411                  1$:     BEQ     4$
000054  013700  000000G                 MOV     L$DLY,R0                ; *,$$TMP1
000060  001404                          BEQ     3$
000062  005066  000004          2$:     CLR     4(SP)                   ; $$TMP
000066  005300                          DEC     R0                      ; $$TMP1
000070  001374                          BNE     2$
000072  005301                  3$:     DEC     R1                      ; $$TMP2
000074  000766                          BR      1$
000076  013700  000000G         4$:     MOV     RC25.ADDR,R0            ;                       2527
000102  012705  000002                  MOV     #2,R5
000106  060005                          ADD     R0,R5
000110  011566  000002                  MOV     (R5),2(SP)              ; *,RC.REG
000114  010302                          MOV     R3,R2                   ; IE.VEC,SA             2528
000116  052702  115400                  BIS     #115400,R2              ; *,SA
000122  010200                          MOV     R2,R0                   ; SA,RC.REG             2529
000124  010015                          MOV     R0,(R5)                 ; RC.REG,*
000126  012703  000012                  MOV     #12,R3                  ; *,COUNT               2530
000132  012701  000515          5$:     MOV     #515,R1                 ; *,$$TMP2              2533
000136  001411                  6$:     BEQ     9$
000140  013700  000000G                 MOV     L$DLY,R0                ; *,$$TMP1
000144  001404                          BEQ     8$
000146  005066  000004          7$:     CLR     4(SP)                   ; $$TMP
000152  005300                          DEC     R0                      ; $$TMP1
000154  001374                          BNE     7$
000156  005301                  8$:     DEC     R1                      ; $$TMP2
000160  000766                          BR      6$
000162  011516                  9$:     MOV     (R5),(SP)               ; *,RC.REG              2534
000164  011602                          MOV     (SP),R2                 ; *,SA
000166  032702  010000                  BIT     #10000,R2               ; *,SA                  2535
000172  001002                          BNE     10$
000174  005303                          DEC     R3                      ; COUNT                 2530
000176  001355                          BNE     5$
000200  005764  000002          10$:    TST     2(R4)                   ;                       2541
000204  001002                          BNE     11$
000206  005000                          CLR     R0                      ;                       2517
000210  000402                          BR      12$
000212  012700  000001          11$:    MOV     #1,R0
000216  062706  000006          12$:    ADD     #6,SP                   ;                       2501
000222  000207                          RTS     PC

; Routine Size:  74 words,      Routine Base:  $CODE$ + 1572
; Maximum stack depth per invocation:  10 words
```

```
;       2548  ROUTINE HARD_INIT =
;       2549
;       2550  !+
;       2551  !       THIS ROUTINE PERFORMS THE FOUR READ / WRITE STEPS REQUIRED TO
;       2552  !       INITIALIZE AN RC25 DEVICE. IF NO READ ERRORS ARE DETECTED IN ANY OF
;       2553  !       THE FOUR STEPS, THEN A SUCCESS VALUE IS RETURNED TO THE CALLER.
;       2554  !       OTHERWISE, ADDITIONAL ATTEMPTS MAY BE MADE TO INITIALIZE THE DEVICE.
;       2555  !       IF ALL ATTEMPTS FAIL, A FAILURE INDICATION IS RETURNED.
;       2556  !
;       2557  !       IMPLICIT INPUTS:
;       2558  !               RC25_ADDR - ADDRESS OF CURRENT CONTROLLER'S IP REGISTER
;       2559  !               VEC_AD - ASSUMED VECTOR ADDRESS OF THE CURRENT CONTROLLER
;       2560  !               DCT_ADDR - ADDRESS OF CURRENT CONTROLLER'S DCT
;       2561  !-
;       2562
;       2563  BEGIN
;       2564
;       2565  LOCAL
;       2566      IE_VEC : WORD;                            ! IE-BIT-AND-VECTOR-ADDRESS/4 BYTE
;       2567                                                !   (USED IN STEP 1 WRITE AND STEP 3 READ)
;       2568
;       2569  IE_VEC = .VEC_AD ^ -2;                       ! GET VECTOR ADDR/4 (IE = 0)
;       2570  INCR ATTEMPTS FROM 1 TO INI_ATT DO
;       2571      BEGIN
;       2572
;       2573      WRT_RC25 (RCIP, RC_ALL, ALL_ONES);        ! WRITE IP TO START INIT SEQUENCE
;       2574      DELAY (500);                              ! WAIT
;       2575  !
;       2576  !       STEP 1
;       2577  !
;       2578      STEP = 1;
;       2579      SA_REG = .RC25_ADDR [RCSA, RC_ALL];       ! READ SA
;       2580      IF (.SA_REG AND S1_MASK) EQLU SA_S1       ! IF STEP 1 READ IS O.K.
;       2581      THEN                                      ! THEN
;       2582          BEGIN
;       2583
;       2584          SA_REG = (WR_RING ^ 8) OR .IE_VEC;    ! STEP 1 WRITE VALUE
;       2585          WRT_RC25 (RCSA, RC_ALL, .SA_REG);     ! STEP 1 WRITE
;       2586          INCR COUNT FROM 1 TO 10 DO            ! TEN SECOND LIMIT
;       2587              BEGIN
;       2588
;       2589              DELAY (333);                      ! ABOUT 1 SECOND
;       2590              SA_REG = .RC25_ADDR [RCSA, RC_ALL]; ! READ SA REGISTER
;       2591              IF BIT_TST (SA_REG, SA_S2)        ! IF STEP 2 HAS BEGUN
;       2592              THEN                              ! THEN
;       2593                  EXITLOOP;                     ! BREAK OUT
;       2594
;       2595              END;
;       2596  !
;       2597  !       STEP 2
;       2598  !
;       2599          STEP = .STEP + 1;
;       2600          IF (.SA_REG AND S2_MASK) EQLU (SA_S2 OR WR_RING) ! IF STEP 2 READ IS O.K.
;       2601          THEN                                             ! THEN
;       2602              BEGIN
;       2603
;       2604              WRT_RC25 (RCSA, RC_ALL, .DCT_ADDR [RR_BEG]);        ! RINGBASE-LO, PI = 0
```

```
2605                INCR COUNT FROM 1 TO 10 DO                          ! TEN SECOND LIMIT
2606                    BEGIN
2607
2608                    DELAY (333);                                    ! ABOUT 1 SECOND
2609                    SA_REG = .RC25_ADDR [RCSA, RC_ALL];             ! READ SA REGISTER
2610                    IF BIT_TST (SA_REG, SA_S3)                      ! IF STEP 3 HAS BEGUN
2611                    THEN                                            ! THEN
2612                        EXITLOOP;                                   ! BREAK OUT
2613
2614                    END;
2615    !
2616    !           STEP 3
2617    !
2618                STEP = .STEP + 1;
2619                IF (.SA_REG AND S3_MASK) EQLU (SA_S3 OR .IE_VEC)    ! IF STEP 3 READ IS O.K.
2620                THEN                                                ! THEN
2621                    BEGIN
2622
2623                    WRT_RC25 (RCSA, RC_ALL, 0);                     ! PP, RINGBASE-HI = 0
2624                    INCR COUNT FROM 1 TO 10 DO                      ! TEN SECOND LIMIT
2625                        BEGIN
2626
2627                        DELAY (333);                                ! ABOUT 1 SECOND
2628                        SA_REG = .RC25_ADDR [RCSA, RC_ALL];         ! READ SA REGISTER
2629                        IF BIT_TST (SA_REG, SA_S4)                  ! IF STEP 4 HAS BEGUN
2630                        THEN                                        ! THEN
2631                            EXITLOOP;                               ! BREAK OUT
2632
2633                        END;
2634    !
2635    !           STEP 4
2636    !
2637                    STEP = .STEP + 1;
2638                    IF (.SA_REG AND S4_MASK) EQLU SA_S4             ! IF STEP 4 READ IS O.K.
2639                    THEN                                            ! THEN
2640                        BEGIN
2641
2642                        WRT_RC25 (RCSA, RC_ALL, 0);                 ! BURST, LF, GO = 0
2643                        RETURN SUCCESS;                             ! SUCCESS EXIT POINT
2644
2645                        END;
2646
2647                    END;
2648
2649                END;
2650
2651            END;
2652
2653        END;                                                        ! TRY AGAIN OR GIVE UP
2654
2655    ERRDF (13, EGD_13, EMS_13);                                     ! "INIT SEQUENCE FAILED"
2656    RETURN FAILURE;
2657
2658    END;                                                            ! ROUTINE HARD_INIT

                                        .SBTTL  HARD.INIT INITIALIZATION SUBTEST ROUTINES
```

```
000000  004137  000000G                HARD.INIT:
                                                JSR     R1,$SAVE5                       ;                                       2548
000004  162706  000012                          SUB     #12,SP
000010  013704  000612'                        MOV     VEC.AD,R4                       ; *,IE.VEC                              2569
000014  006204                                  ASR     R4                              ; IE.VEC
000016  006204                                  ASR     R4                              ; IE.VEC
000020  013703  000000G                        MOV     L$DLY,R3                        ;                                       2574
000024  013700  000000G                        MOV     RC25.ADDR,R0                    ;                                       2579
000030  012746  000002                        MOV     #2,-(SP)
000034  060016                                ADD     R0,(SP)
000036  012705  000002                        MOV     #2,R5                           ; *,ATTEMPTS                            2570
000042  012700  177777          1$:           MOV     #-1,R0                          ; *,RC.REG                              2573
000046  010077  000000G                       MOV     R0,@RC25.ADDR                   ; RC.REG,*
000052  012701  000764                        MOV     #764,R1                         ; *,$$TMP2                              2574
000056  001410                  2$:           BEQ     5$
000060  010300                                MOV     R3,R0                           ; *,$$TMP1
000062  001404                                BEQ     4$
000064  005066  000012          3$:           CLR     12(SP)                          ; $$TMP
000070  005300                                DEC     R0                              ; $$TMP1
000072  001374                                BNE     3$
000074  005301                  4$:           DEC     R1                              ; $$TMP2
000076  000767                                BR      2$
000100  012737  000001  000000G 5$:           MOV     #1,STEP                         ;                                       2578
000106  017666  000000  000010                MOV     @0(SP),10(SP)                   ; *,RC.REG                              2579
000114  016637  000010  000000G               MOV     10(SP),SA.REG                   ; RC.REG,*
000122  016600  000010                        MOV     10(SP),R0                       ; SA.REG,*                              2580
000126  042700  001777                        BIC     #1777,R0
000132  020027  004000                        CMP     R0,#4000
000136  001177                                BNE     24$
000140  010437  000000G                       MOV     R4,SA.REG                       ; IE.VEC,*                              2584
000144  052737  115400  000000G               BIS     #115400,SA.REG
000152  013700  000000G                       MOV     SA.REG,R0                       ; *,RC.REG                              2585
000156  010076  000000                        MOV     R0,@0(SP)                       ; RC.REG,*
000162  012702  000012                        MOV     #12,R2                          ; *,COUNT                               2586
000166  012701  000515          6$:           MOV     #515,R1                         ; *,$$TMP2                              2589
000172  001410                  7$:           BEQ     10$
000174  010300                                MOV     R3,R0                           ; *,$$TMP1
000176  001404                                BEQ     9$
000200  005066  000012          8$:           CLR     12(SP)                          ; $$TMP
000204  005300                                DEC     R0                              ; $$TMP1
000206  001374                                BNE     8$
000210  005301                  9$:           DEC     R1                              ; $$TMP2
000212  000767                                BR      7$
000214  017666  000000  000006  10$:          MOV     @0(SP),6(SP)                    ; *,RC.REG                              2590
000222  016637  000006  000000G               MOV     6(SP),SA.REG                    ; RC.REG,*
000230  032766  010000  000006                BIT     #10000,6(SP)                    ; *,SA.REG                              2591
000236  001002                                BNE     11$
000240  005302                                DEC     R2                              ; COUNT                                 2586
000242  001351                                BNE     6$
000244  005237  000000G           11$:        INC     STEP                            ;                                       2599
000250  013700  000000G                       MOV     SA.REG,R0                       ;                                       2600
000254  042700  003400                        BIC     #3400,R0
000260  020027  010233                        CMP     R0,#10233
000264  001124                                BNE     24$
000266  013700  000000G                       MOV     DCT.ADDR,R0                     ;                                       2604
000272  016000  000004                        MOV     4(R0),R0                        ; *,RC.REG
000276  010076  000000                        MOV     R0,@0(SP)                       ; RC.REG,*
```

```
000302  012702  000012                    MOV     #12,R2                  ; *,COUNT                       2605
000306  012701  000515          12$:      MOV     #515,R1                 ; *,$$TMP2                      2608
000312  001410                  13$:      BEQ     16$
000314  010300                            MOV     R3,R0                   ; *,$$TMP1
000316  001404                            BEQ     15$
000320  005066  000012          14$:      CLR     12(SP)                  ; $$TMP
000324  005300                            DEC     R0                      ; $$TMP1
000326  001374                            BNE     14$
000330  005301                  15$:      DEC     R1                      ; $$TMP2
000332  000767                            BR      13$
000334  017666  000000  000004  16$:      MOV     @0(SP),4(SP)            ; *,RC.REG                      2609
000342  016637  000004  000000G           MOV     4(SP),SA.REG            ; RC.REG,*
000350  032766  020000  000004            BIT     #20000,4(SP)            ; *,SA.REG                      2610
000356  001002                            BNE     17$
000360  005302                            DEC     R2                      ; COUNT                         2605
000362  001351                            BNE     12$
000364  005237  000000G         17$:      INC     STEP                    ;                               2618
000370  013701  000000G                   MOV     SA.REG,R1               ;                               2619
000374  042701  003400                    BIC     #3400,R1
000400  010400                            MOV     R4,R0                   ; IE.VEC,*
000402  052700  020000                    BIS     #20000,R0
000406  020100                            CMP     R1,R0
000410  001052                            BNE     24$
000412  005000                            CLR     R0                      ; RC.REG                        2623
000414  005076  000000                    CLR     @0(SP)
000420  012702  000012                    MOV     #12,R2                  ; *,COUNT                       2624
000424  012701  000515          18$:      MOV     #515,R1                 ; *,$$TMP2                      2627
000430  001410                  19$:      BEQ     22$
000432  010300                            MOV     R3,R0                   ; *,$$TMP1
000434  001404                            BEQ     21$
000436  005066  000012          20$:      CLR     12(SP)                  ; $$TMP
000442  005300                            DEC     R0                      ; $$TMP1
000444  001374                            BNE     20$
000446  005301                  21$:      DEC     R1                      ; $$TMP2
000450  000767                            BR      19$
000452  017666  000000  000002  22$:      MOV     @0(SP),2(SP)            ; *,RC.REG                      2628
000460  016637  000002  000000G           MOV     2(SP),SA.REG            ; RC.REG,*
000466  032766  040000  000002            BIT     #40000,2(SP)            ; *,SA.REG                      2629
000474  001002                            BNE     23$
000476  005302                            DEC     R2                      ; COUNT                         2624
000500  001351                            BNE     18$
000502  005237  000000G         23$:      INC     STEP                    ;                               2637
000506  013700  000000G                   MOV     SA.REG,R0               ;                               2638
000512  042700  003777                    BIC     #3777,R0
000516  020027  040000                    CMP     R0,#40000
000522  001005                            BNE     24$
000524  005076  000000                    CLR     @0(SP)                  ;                               2642
000530  012700  000001                    MOV     #1,R0                   ;                               2640
000534  000411                            BR      26$
000536  005305                  24$:      DEC     R5                      ; ATTEMPTS                      2570
000540  001402                            BEQ     25$
000542  000137  002060'                    JMP     1$
000546  104455                  25$:      TRAP    55                      ;                               2655
000550  000015                            .WORD   15
000552  000000G                           .WORD   EGD.13
000554  000000G                           .WORD   EMS.13
000556  005000                            CLR     R0                      ;                               2563
```

```
000560  062706  000014                26$:    ADD     #14,SP                          ;                       2548
000564  000207                              RTS     PC

; Routine Size:  187 words,     Routine Base:  $CODE$ + 2016
; Maximum stack depth per invocation:  14 words
```

```
;          2659  ROUTINE INI_RRING : NOVALUE =
;          2660
;          2661  !+
;          2662  !       THIS ROUTINE IS RESPONSIBLE FOR ALLOCATING ENOUGH MSCP ENVELOPES TO
;          2663  !       FILL AN RC25 RESPONSE RING. THE ENVELOPE DESCRIPTOR OF EACH ENVELOPE
;          2664  !       (LOCATED IN FRONT OF THE ENVELOPE ITSELF) IS LOADED INTO SUCCESSIVE
;          2665  !       RRING SLOTS. NOTE THAT THE ENVELOPE DESCRIPTORS HAVE BEEN INITIALIZED
;          2666  !       WITH THE FLAG AND OWNERSHIP BITS SET TO "1", MAKING EACH SLOT
;          2667  !       CONTROLLER-OWNED.
;          2668  !
;          2669  !       IMPLICIT INPUTS:
;          2670  !               CCTLR - CURRENT CONTROLLER NUMBER
;          2671  !               DCT_ADDR - ADDRESS OF CURRENT CONTROLLER'S DCT
;          2672  !-
;          2673
;          2674  BEGIN
;          2675
;          2676  LOCAL
;          2677      INDEX : WORD,
;          2678      RRING_ADDR;
;          2679
;          2680  RRING_ADDR = .DCT_ADDR [RR_BEG];                  ! FIRST RESPONSE RING SLOT
;          2681  INCR COUNT FROM 1 TO RRING_LEN DO
;          2682      BEGIN
;          2683
;          2684      INDEX = GET_ENV (.CCTLR);                     ! GET AN MSCP ENVELOPE
;          2685      MSCP_ENV [.INDEX, MSGLEN] = MSG_LEN * 2;      ! SET MESSAGE LENGTH FIELD
;          2686      .RRING_ADDR = .MSCP_ENV [.INDEX, ENV_LO];     ! LOAD LO-ORDER ENV DESC INTO SLOT
;          2687      RRING_ADDR = .RRING_ADDR + 2;                 ! ADVANCE TO SECOND WORD
;          2688      .RRING_ADDR = .MSCP_ENV [.INDEX, ENV_HI];     ! LOAD HI-ORDER ENV DESC INTO SLOT
;          2689      RRING_ADDR = .RRING_ADDR + 2;                 ! ADVANCE TO NEXT SLOT
;          2690
;          2691      END;
;          2692
;          2693  END;


                                .SBTTL  INI.RRING INITIALIZATION SUBTEST ROUTINES
000000  004137  000000G     INI.RRING:
                            JSR     R1,$SAVE3                       ;                       2659
000004  013700  000000G             MOV     DCT.ADDR,R0             ;                       2680
000010  016002  000004              MOV     4(R0),R2                ; *,RRING.ADDR
000014  012701  000010              MOV     #10,R1                  ; *,COUNT               2681
000020  013746  000000G     1$:     MOV     CCTLR,-(SP)             ;                       2684
000024  004737  000000G             JSR     PC,GET.ENV
000030  010003                      MOV     R0,R3                   ; *,INDEX
000032  010316                      MOV     R3,(SP)                 ; INDEX,*               2685
000034  012746  000104              MOV     #104,-(SP)
000040  004737  000000G             JSR     PC,BL$MUL
000044  012760  000074  000004G     MOV     #74,MSCP.ENV+4(R0)
000052  016022  000000G             MOV     MSCP.ENV(R0),(R2)+      ; *,RRING.ADDR          2686
000056  016022  000002G             MOV     MSCP.ENV+2(R0),(R2)+    ; *,RRING.ADDR          2688
000062  022626                      CMP     (SP)+,(SP)+             ;                       2682
000064  005301                      DEC     R1                      ; COUNT                 2681
000066  001354                      BNE     1$
000070  000207                      RTS     PC                      ;                       2659
```

```
; Routine Size:  29 words,      Routine Base:  $CODE$ + 2604
; Maximum stack depth per invocation:  7 words
```

```
;        2694  ROUTINE SET_CTLR_CHAR : NOVALUE =
;        2695
;        2696  !+
;        2697  !       THIS ROUTINE IS CALLED BY CTLR_INIT AFTER THE RC25 HAS BEEN HARD-
;        2698  !       INITIALIZED. ITS PURPOSE IS TO FORMAT AND SEND THE "SET CONTROLLER
;        2699  !       CHARACTERISTICS" COMMAND, AND TO VERIFY THAT THE MESSAGE WAS ACCEPTED.
;        2700  !
;        2701  !       IMPLICIT INPUTS:
;        2702  !               CCTLR - CURRENT CONTROLLER NUMBER
;        2703  !-
;        2704
;        2705  BEGIN
;        2706
;        2707  LOCAL
;        2708      REASON : WORD,                                  ! DROP UNIT REASON
;        2709      M_INDEX : WORD;                                 ! MSCP ENVELOPE INDEX
;        2710
;        2711  M_INDEX = GET_ENV (.CCTLR);                        ! GET AN MSCP ENVELOPE
;        2712  MSCP_ENV [.M_INDEX, OPCODE] = OP_SCC;              ! OPCODE = SET CTLR CHAR
;        2713  MSCP_ENV [.M_INDEX, C_FLAGS] = CF_MSC + CF_THS;    ! CONTROLLER FLAGS
;        2714  IF SEND (.M_INDEX) EQLU FAILURE                   ! ATTEMPT SEND; IF CTLR IS OFFLINE
;        2715  THEN                                              ! THEN
;        2716      PUT_ENV (.M_INDEX)                            ! RETURN ENVELOPE TO POOL
;        2717  ELSE                                              ! IF SEND WAS SUCCESSFUL
;        2718      BEGIN
;        2719
;        2720      WAIT ();                                      ! WAIT FOR RETPKT RESPONSE
;        2721      RP_INDX = OUT_IODQ ();                        ! GET INDEX OF RETPKT
;        2722      RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2);   ! CALCULATE RETPKT ADDRESS
;        2723      IF .RP_ADDR [CONID] EQLU CID_DRIVER           ! IF RETPKT IS FROM "DRIVER"
;        2724      THEN                                          ! THEN
;        2725          BEGIN
;        2726
;        2727          REASON = DU_FATAL;                        ! ASSUME FATAL SA ERROR
;        2728          IF .RP_ADDR [MESTYP] EQLU MT_TIMEOUT      ! IF COMMAND TIMED OUT
;        2729          THEN                                      ! THEN
;        2730              BEGIN
;        2731
;        2732              REASON = DU_INIT;                     ! SET REASON TO INIT ERROR
;        2733              ERRDF (15, EGD_15, EMS_15);           ! "MESSAGE RESPONSE TIMEOUT"
;        2734
;        2735              END;
;        2736
;        2737          DROP_CTLR (.CCTLR, .REASON);              ! DROP CONTROLLER'S UNITS
;        2738
;        2739          END;                                      ! IF RETPKT IS FROM "DRIVER"
;        2740
;        2741      PUT_RETPKT (.RP_INDX);
;        2742
;        2743      END;                                          ! IF SEND WAS SUCCESSFUL
;        2744
;        2745  END;                                  ! ROUTINE SET_CTLR_CHAR


                                .SBTTL  SET.CTLR.CHAR INITIALIZATION SUBTEST ROUTINES
000000  010146          SET.CTLR.CHAR:
                                MOV     R1,-(SP)                ;                       2694
```

```
000002 013746 000000G                  MOV     CCTLR,-(SP)                     ;                                       2711
000006 004737 000000G                  JSR     PC,GET.ENV
000012 010001                          MOV     R0,R1                           ; *,M.INDEX
000014 010116                          MOV     R1,(SP)                         ; M.INDEX,*                             2712
000016 012746 000104                   MOV     #104,-(SP)
000022 004737 000000G                  JSR     PC,BL$MUL
000026 112760 000004 000020G           MOVB    #4,MSCP.ENV+20(R0)
000034 012760 000120 000026G           MOV     #120,MSCP.ENV+26(R0)            ;                                       2713
000042 010116                          MOV     R1,(SP)                         ; M.INDEX,*                             2714
000044 004737 000000G                  JSR     PC,SEND
000050 005700                          TST     R0
000052 001004                          BNE     1$
000054 010116                          MOV     R1,(SP)                         ; M.INDEX,*                             2716
000056 004737 000000G                  JSR     PC,PUT.ENV
000062 000455                          BR      4$                              ;                                       2714
000064 004737 000000G           1$:    JSR     PC,WAIT                         ;                                       2720
000070 004737 000000G                  JSR     PC,OUT.IODQ                     ;                                       2721
000074 010037 000000G                  MOV     R0,RP.INDX
000100 010016                          MOV     R0,(SP)                         ; RP.INDX,*                             2722
000102 012746 000060                   MOV     #60,-(SP)
000106 004737 000000G                  JSR     PC,BL$MUL
000112 062700 000000G                  ADD     #RETPKT,R0
000116 010037 000000G                  MOV     R0,RP.ADDR
000122 126027 000003 000003            CMPB    3(R0),#3                        ;                                       2723
000130 001025                          BNE     3$
000132 012701 000004                   MOV     #4,R1                           ; *,REASON                              2727
000136 116000 000002                   MOVB    2(R0),R0                        ;                                       2728
000142 042700 177417                   BIC     #177417,R0
000146 020027 000100                   CMP     R0,#100
000152 001006                          BNE     2$
000154 012701 000002                   MOV     #2,R1                           ; *,REASON                              2732
000160 104455                          TRAP    55                              ;                                       2733
000162 000017                          .WORD   17
000164 000000G                         .WORD   EGD.15
000166 000000G                         .WORD   EMS.15
000170 013716 000000G           2$:    MOV     CCTLR,(SP)                      ;                                       2737
000174 010146                          MOV     R1,-(SP)                        ; REASON,*
000176 004737 000000G                  JSR     PC,DROP.CTLR
000202 005726                          TST     (SP)+                           ;                                       2725
000204 013716 000000G           3$:    MOV     RP.INDX,(SP)                    ;                                       2741
000210 004737 000000G                  JSR     PC,PUT.RETPKT
000214 005726                          TST     (SP)+                           ;                                       2718
000216 022626                   4$:    CMP     (SP)+,(SP)+                     ;                                       2705
000220 012601                          MOV     (SP)+,R1                        ;                                       2694
000222 000207                          RTS     PC

; Routine Size:  74 words,      Routine Base:  $CODE$ + 2676
; Maximum stack depth per invocation:  6 words
```

```
2746  ROUTINE UNIT_INIT : NOVALUE =
2747
2748  !+
2749  !       THIS ROUTINE IS CALLED FROM DRIVER_INIT FOR EACH CONFIGURED UNIT
2750  !       (PLATTER) WHICH IS ATTACHED TO A CONTROLLER THAT SURVIVED
2751  !       INITIALIZATION. ITS PURPOSE IS TO FORMAT AND SEND AN ''ONLINE''
2752  !       MESSAGE, AND TO VERIFY THE RESPONSE.
2753  !
2754  !       IMPLICIT INPUTS:
2755  !               CCTLR - CURRENT CONTROLLER NUMBER
2756  !               CPLAT - CURRENT PLATTER ADDRESS (MSCP UNIT NUMBER)
2757  !               L$LUN - CURRENT (DRS) UNIT NUMBER
2758  !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
2759  !               CUOFF - CURRENT UNIT CST OFFSET
2760  !-
2761
2762  BEGIN
2763
2764  LOCAL
2765      REASON : WORD,                                  ! DROP UNIT REASON
2766      M_INDEX : WORD;                                 ! MSCP ENVELOPE INDEX
2767
2768  M_INDEX = GET_ENV (.CCTLR);                         ! GET AN MSCP ENVELOPE
2769  MSCP_ENV [.M_INDEX, PL_ADDR] = .CPLAT;              ! SET PLATTER ADDRESS (MSCP UNIT NUMBER)
2770  MSCP_ENV [.M_INDEX, OPCODE] = OP_ONL;               ! OPCODE FOR ''ONLINE''
2771  MSCP_ENV [.M_INDEX, DDPAR] = BIT00;                 ! SHOW ALL ECC ERRORS IN ERROR LOG MESSAGES
2772  IF SEND (.M_INDEX) EQLU FAILURE                     ! ATTEMPT TO SEND; IF CTLR IS OFFLINE
2773  THEN                                                ! THEN
2774      PUT_ENV (.M_INDEX)                              ! RETURN ENVELOPE TO POOL
2775  ELSE                                                ! OTHERWISE (SEND WAS SUCCESSFUL)
2776      BEGIN
2777
2778      WAIT ();                                        ! WAIT FOR RETPKT RESPONSE
2779      RP_INDX = OUT_IODQ ();                          ! GET INDEX OF RETPKT
2780      RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2);     ! CALCULATE RETPKT ADDRESS
2781      IF .RP_ADDR [CONID] EQLU CID_DRIVER             ! IF RETPKT IS FROM ''DRIVER''
2782      THEN                                            ! THEN
2783          BEGIN
2784
2785          REASON = DU_FATAL;                          ! ASSUME FATAL SA ERROR
2786          IF .RP_ADDR [MESTYP] EQLU MT_TIMEOUT        ! IF COMMAND TIMED OUT
2787          THEN                                        ! THEN
2788              BEGIN
2789
2790              REASON = DU_INIT;                       ! SET REASON TO INIT ERROR
2791              ERRDF (15, EGD_15, EMS_15);             ! ''MESSAGE RESPONSE TIMEOUT''
2792
2793              END;
2794
2795          DROP_CTLR (CCTLR, .REASON);                 ! DROP ALL CONTROLLER'S UNITS
2796
2797          END
2798      ELSE                                            ! OTHERWISE, IF RETPKT IS FROM DISK MSCP
2799          BEGIN
2800
2801          ST_CODE = .RP_ADDR [STSCOD];                ! GET STATUS CODE
2802          SB_CODE = .RP_ADDR [SUBCOD];                ! GET SUB-CODE
```

```
2803            IF .ST_CODE NEQU ST_SUC                 ! IF STATUS CODE IS NOT SUCCESSFUL
2804            THEN                                    ! THEN
2805                BEGIN
2806
2807                ERRDF (16, EGD_16, EMS_16);         ! "ONLINE FAILED"
2808                DUR [.L$LUN] = DU_INIT;             ! SET UP REASON FOR DROPPING UNIT
2809                DODU (.L$LUN);                      ! DROP UNIT
2810
2811                END
2812            ELSE                                    ! STATUS CODE IS O.K.
2813                BEGIN
2814
2815                IF ((BIT_TST (RP_ADDR [U_FLGS], UF_WPH)) AND        ! IF UNIT IS WRITE-PROTECTED AND
2816                    (.CST_ADDR [.CUOFF, P_PROT] EQ[U UNPROTECTED))  !    USER ALLOWED WRITES
2817                THEN                                                ! THEN
2818                    BEGIN
2819
2820                    ERRDF (17, EGD_17, EMS_17);                     ! "WRITE-PROTECT CONFLICT"
2821                    DUR [.L$LUN] = DU_INIT;                         ! SET REASON TO DROP UNIT
2822                    DODU (.L$LUN);                                  ! DROP UNIT
2823
2824                    END
2825                ELSE                                                ! WRITE PROTECT SWITCH IS O.K.
2826                    BEGIN
2827
2828                    USIZE = .RP_ADDR [USIZ_LO];                     ! UNIT SIZE (NO. OF LBN'S)
2829                    CST_ADDR [.CUOFF, P_STAT] = ONLINE;             ! MARK UNIT ONLINE
2830                    CPT [.L$LUN] = YES;                             ! O.K. TO TEST
2831                    CST_ADDR [U_CNT] = .CST_ADDR [U_CNT] + 1;       ! INCR NO. OF TESTABLE UNITS
2832
2833                    END;
2834
2835                END;                                ! IF STATUS CODE = SUCCESS
2836
2837            END;                                    ! IF RETPKT ORIGINATED AT CONTROLLER
2838
2839        PUT_RETPKT (.RP_INDX);
2840
2841        END;                                        ! IF SEND WAS SUCCESSFUL
2842
2843    END;                                            ! ROUTINE UNIT_INIT


                                .SBTTL  UNIT.INIT INITIALIZATION SUBTEST ROUTINES
000000  004137  000000G     UNIT.INIT:
                            JSR     R1,$SAVE2                       ;                       2746
000004  013746  000000G             MOV     CCTLR,-(SP)             ;                       2768
000010  004737  000000G             JSR     PC,GET.ENV
000014  010001                      MOV     R0,R1                   ; *,M.INDEX
000016  010116                      MOV     R1,(SP)                 ; M.INDEX,*             2769
000020  012746  000104              MOV     #104,-(SP)
000024  004737  000000G             JSR     PC,BL$MUL
000030  013760  000000G 000014G     MOV     CPLAT,MSCP.ENV+14(R0)
000036  112760  000011  000020G     MOVB    #11,MSCP.ENV+20(R0)     ;                       2770
000044  012760  000001  000044G     MOV     #1,MSCP.ENV+44(R0)      ;                       2771
000052  010116                      MOV     R1,(SP)                 ; M.INDEX,*             2772
000054  004737  000000G             JSR     PC,SEND
```

```
000060  005700                          TST     R0
000062  001004                          BNE     1$
000064  010116                          MOV     R1,(SP)                 ; M.INDEX,*                     2774
000066  004737  000000G                 JSR     PC,PUT.ENV
000072  000565                          BR      7$                      ;                               2772
000074  004737  000000G           1$:   JSR     PC,WAIT                 ;                               2778
000100  004737  000000G                 JSR     PC,OUT.IODQ             ;                               2779
000104  010037  000000G                 MOV     R0,RP.INDX
000110  010016                          MOV     R0,(SP)                 ; RP.INDX,*                     2780
000112  012746  000060                  MOV     #60,-(SP)
000116  004737  000000G                 JSR     PC,BL$MUL
000122  062700  000000G                 ADD     #RETPKT,R0
000126  010037  000000G                 MOV     R0,RP.ADDR
000132  010002                          MOV     R0,R2                   ; RP.ADDR,*                     2781
000134  126227  000003  000003          CMPB    3(R2),#3
000142  001026                          BNE     3$
000144  012700  000004                  MOV     #4,R0                   ; *,REASON                      2785
000150  116201  000002                  MOVB    2(R2),R1                ;                               2786
000154  042701  177417                  BIC     #177417,R1
000160  020127  000100                  CMP     R1,#100
000164  001006                          BNE     2$
000166  012700  000002                  MOV     #2,R0                   ; *,REASON                      2790
000172  104455                          TRAP    55                      ;                               2791
000174  000017                          .WORD   17
000176  000000G                         .WORD   EGD.15
000200  000000G                         .WORD   EMS.15
000202  012716  000000G           2$:   MOV     #CCTLR,(SP)             ;                               2795
000206  010046                          MOV     R0,-(SP)                ; REASON,*
000210  004737  000000G                 JSR     PC,DROP.CTLR
000214  005726                          TST     (SP)+                   ;                               2783
000216  000506                          BR      6$                      ;                               2781
000220  116237  000016  000000G   3$:   MOVB    16(R2),ST.CODE          ;                               2801
000226  042737  177740  000000G         BIC     #177740,ST.CODE
000234  016200  000016                  MOV     16(R2),R0               ;                               2802
000240  006200                          ASR     R0
000242  006200                          ASR     R0
000244  006200                          ASR     R0
000246  006200                          ASR     R0
000250  006200                          ASR     R0
000252  042700  174000                  BIC     #174000,R0
000256  010037  000000G                 MOV     R0,SB.CODE
000262  013701  000000G                 MOV     L$LUN,R1                ;                               2808
000266  005737  000000G                 TST     ST.CODE                 ;                               2803
000272  001412                          BEQ     4$
000274  104455                          TRAP    55                      ;                               2807
000276  000020                          .WORD   20
000300  000000G                         .WORD   EGD.16
000302  000000G                         .WORD   EMS.16
000304  112761  000002  000000G         MOVB    #2,DUR(R1)              ;                               2808
000312  010100                          MOV     R1,R0                   ;                               2809
000314  104451                          TRAP    51
000316  000446                          BR      6$                      ;                               2803
000320  032762  020000  000022   4$:   BIT     #20000,22(R2)           ;                               2815
000326  001421                          BEQ     5$
000330  013700  000000G                 MOV     CUOFF,R0                ;                               2816
000334  006300                          ASL     R0
000336  063700  000000G                 ADD     CST.ADDR,R0
```

```
000342  005710                          TST     (R0)
000344  100012                          BPL     5$
000346  104455                          TRAP    55                              ;2820
000350  000021                          .WORD   21
000352  000000G                         .WORD   EGD.17
000354  000000G                         .WORD   EMS.17
000356  112761  000002  000000G         MOVB    #2,DUR(R1)                      ;2821
000364  010100                          MOV     R1,R0                           ;2822
000366  104451                          TRAP    51
000370  000421                          BR      6$                              ;2815
000372  016237  000050  000616'  5$:    MOV     50(R2),USIZE                    ;2828
000400  013700  000000G                 MOV     CUOFF,R0                        ;2829
000404  006300                          ASL     R0
000406  063700  000000G                 ADD     CST.ADDR,R0
000412  052710  020000                  BIS     #20000,(R0)
000416  112761  000001  000000G         MOVB    #1,CPT(R1)                      ;2830
000424  013700  000000G                 MOV     CST.ADDR,R0                     ;2831
000430  105260  000005                  INCB    5(R0)
000434  013716  000000G         6$:     MOV     RP.INDX,(SP)                    ;2839
000440  004737  000000G                 JSR     PC,PUT.RETPKT
000444  005726                          TST     (SP)+                           ;2776
000446  022626                  7$:     CMP     (SP)+,(SP)+                     ;2762
000450  000207                          RTS     PC                              ;2746

; Routine Size:  149 words,    Routine Base:  $CODE$ + 3122
; Maximum stack depth per invocation:  8 words
```

```
;        2844  ROUTINE ACCESS : NOVALUE =
;        2845
;        2846  !+
;        2847  !       THIS ROUTINE IS CALLED BY INIT_TEST TO VERIFY THAT THE CURRENT PLATTER
;        2848  !       CAN BE ACCESSED. THIS OBJECTIVE IS ACCOMPLISHED BY FORMATTING AND
;        2849  !       SENDING ONE OR TWO MSCP ACCESS COMMANDS TO THE PLATTER, AND CHECKING
;        2850  !       THE STATUS FIELD OF THE RESPONSE MESSAGE(S).
;        2851  !
;        2852  !       IMPLICIT INPUTS:
;        2853  !               CCTLR - CURRENT CONTROLLER NUMBER
;        2854  !               CPLAT - CURRENT PLATTER ADDRESS (MSCP UNIT NUMBER)
;        2855  !               L$LUN - CURRENT (DRS) UNIT NUMBER
;        2856  !-
;        2857
;        2858  BEGIN
;        2859
;        2860  LOCAL
;        2861      M_INDEX : WORD,
;        2862      RESULT : WORD,
;        2863      LBN : WORD,
;        2864      PASS : WORD;
;        2865
;        2866  RESULT = FAILURE;                          ! GUILTY UNTIL PROVEN INNOCENT
;        2867  ST_CODE = SB_CODE = 0;                     ! STATUS CODE AND SUB-CODE
;        2868  LBN = ((.USIZE ^ -1) AND %O'77777') - 1;   ! START WITH LAST LBN ON TOP SURFACE
;        2869  PASS = 1;                                  ! LOOP PASS COUNT
;        2870  DO                                         ! LOOP STARTS HERE
;        2871      BEGIN
;        2872
;        2873      M_INDEX = GET_ENV (.CCTLR);           ! GET AN MSCP ENVELOPE
;        2874      MSCP_ENV [.M_INDEX, PL_ADDR] = .CPLAT;! SET PLATTER ADDR (MSCP UNIT NUMBER)
;        2875      MSCP_ENV [.M_INDEX, OPCODE] = OP_ACC; ! ACCESS OPCODE
;        2876      MSCP_ENV [.M_INDEX, BC_LO] = BLK_SIZE;! BYTE COUNT (1 BLOCK)
;        2877      MSCP_ENV [.M_INDEX, LBN_L] = .LBN;    ! LOGICAL BLOCK NUMBER
;        2878      IF SEND (.M_INDEX) EQLU FAILURE       ! ATTEMPT TO SEND; IF CTLR NOT ONLINE
;        2879      THEN                                  ! THEN
;        2880          BEGIN
;        2881
;        2882          PUT_ENV (.M_INDEX);               ! RETURN ENVELOPE TO POOL
;        2883          PASS = 2;                         ! NO MORE TRIES
;        2884
;        2885          END
;        2886      ELSE                                  ! IF SEND WAS SUCCESSFUL
;        2887          BEGIN
;        2888
;        2889          WAIT ();                                  ! WAIT FOR RESPONSE
;        2890          RP_INDX = OUT_IODQ ();                    ! GET RETPKT (RESPONSE) INDEX
;        2891          RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2);! CALCULATE RETPKT ADDRESS
;        2892          IF .RP_ADDR [CONID] EQLU CID_DRIVER       ! IF RETPKT CAME FROM 'DRIVER"
;        2893          THEN                                      ! THEN
;        2894              PASS = 2                              ! NO MORE TRIES
;        2895          ELSE                                      ! OTHERWISE - CHECK OUT RESPONSE
;        2896              BEGIN
;        2897
;        2898              ST_CODE = .RP_ADDR [STSCOD];          ! GET STATUS CODE FROM PACKET
;        2899              SB_CODE = .RP_ADDR [SUBCOD];          ! GET SUB-CODE FROM PACKET
;        2900              IF .ST_CODE EQLU ST_SUC               ! IF STATUS CODE INDICATES SUCCESS
```

```
;          2901                THEN                                        ! THEN
;          2902                    BEGIN
;          2903
;          2904                    RESULT = SUCCESS;
;          2905                    PASS = 2;                               ! NO NEED TO TRY AGAIN
;          2906
;          2907                    END;
;          2908
;          2909                END;                                        ! IF RETPKT ORIGINATED AT CONTROLLER
;          2910
;          2911            PUT_RETPKT (.RP_INDX);
;          2912
;          2913            END;                                            ! IF SEND WAS SUCCESSFUL
;          2914
;          2915        LBN = .LBN + 1;                                     ! ADVANCE TO FIRST LBN OF BOTTOM SURFACE
;          2916        PASS = .PASS + 1;                                   ! SECOND PASS
;          2917
;          2918        END                                                 ! END OF PASS LOOP
;          2919
;          2920    UNTIL .PASS GEQU 3;
;          2921
;          2922    IF .RESULT EQLU FAILURE
;          2923    THEN
;          2924        BEGIN
;          2925
;          2926        SETPRI (PRI05);                                     ! DELAY POSSIBLE ERROR LOG MESSAGE
;          2927        ERRDF (18, EGD_18, EMS_18);                         ! "ACCESS FAILED"
;          2928        SETPRI (PRI00);                                     ! RESTORE PROCESSOR PRIORITY
;          2929        DUR [.L$LUN] = DU_INIT;                             ! SET REASON TO DROP UNIT
;          2930        DODU (.L$LUN);                                      ! DROP UNIT
;          2931
;          2932        END;                                                ! IF ACCESS FAILED
;          2933
;          2934    END;                                                    ! ROUTINE ACCESS


                                        .SBTTL  ACCESS INITIALIZATION SUBTEST ROUTINES
000000  004137  000000G         ACCESS: JSR     R1,$SAVE5                       ;                               2844
000004  005002                          CLR     R2                              ; RESULT                        2866
000006  005037  000000G                 CLR     SB.CODE                         ;                               2867
000012  005037  000000G                 CLR     ST.CODE
000016  013700  000616'                 MOV     USIZE,R0                        ;                               2868
000022  006200                          ASR     R0
000024  010003                          MOV     R0,R3                           ; *,LBN
000026  042703  100000                  BIC     #100000,R3                      ; *,LBN
000032  005303                          DEC     R3                              ; LBN
000034  012701  000001                  MOV     #1,R1                           ; *,PASS                        2869
000040  013746  000000G         1$:     MOV     CCTLR,-(SP)                     ;                               2873
000044  004737  000000G                 JSR     PC,GET.ENV
000050  010004                          MOV     R0,R4                           ; *,M.INDEX
000052  010416                          MOV     R4,(SP)                         ; M.INDEX,*                     2874
000054  012746  000104                  MOV     #104,-(SP)
000060  004737  000000G                 JSR     PC,BL$MUL
000064  013760  000000G 000014G         MOV     CPLAT,MSCP.ENV+14(R0)
000072  112760  000020  000020G         MOVB    #20,MSCP.ENV+20(R0)             ;                               2875
000100  012760  001000  000024G         MOV     #1000,MSCP.ENV+24(R0)           ;                               2876
000106  010360  000044G                 MOV     R3,MSCP.ENV+44(R0)              ; LBN,*                         2877
```

```
000112  010416                          MOV     R4,(SP)                 ; M.INDEX,*                             2878
000114  004737  000000G                 JSR     PC,SEND
000120  005700                          TST     R0
000122  001006                          BNE     2$
000124  010416                          MOV     R4,(SP)                 ; M.INDEX,*                             2882
000126  004737  000000G                 JSR     PC,PUT.ENV
000132  012701  000002                  MOV     #2,R1                   ; *,PASS                                2883
000136  000460                          BR      5$                      ;                                       2878
000140  004737  000000G          2$:    JSR     PC,WAIT                 ;                                       2889
000144  004737  000000G                 JSR     PC,OUT.IODQ             ;                                       2890
000150  010037  000000G                 MOV     R0,RP.INDX
000154  010016                          MOV     R0,(SP)                 ; RP.INDX,*                             2891
000156  012746  000060                  MOV     #60,-(SP)
000162  004737  000000G                 JSR     PC,BL$MUL
000166  062700  000000G                 ADD     #RETPKT,R0
000172  010037  000000G                 MOV     R0,RP.ADDR
000176  126027  000003  000003          CMPB    3(R0),#3                ;                                       2892
000204  001426                          BEQ     3$                      ;                                       2894
000206  116037  000016  000000G         MOVB    16(R0),ST.CODE          ;                                       2898
000214  042737  177740  000000G         BIC     #177740,ST.CODE
000222  016005  000016                  MOV     16(R0),R5               ;                                       2899
000226  006205                          ASR     R5
000230  006205                          ASR     R5
000232  006205                          ASR     R5
000234  006205                          ASR     R5
000236  006205                          ASR     R5
000240  042705  174000                  BIC     #174000,R5
000244  010537  000000G                 MOV     R5,SB.CODE
000250  005737  000000G                 TST     ST.CODE                 ;                                       2900
000254  001004                          BNE     4$
000256  012702  000001                  MOV     #1,R2                   ; *,RESULT                              2904
000262  012701  000002          3$:     MOV     #2,R1                   ; *,PASS                                2905
000266  013716  000000G          4$:     MOV     RP.INDX,(SP)            ;                                       2911
000272  004737  000000G                 JSR     PC,PUT.RETPKT
000276  005726                          TST     (SP)+                   ;                                       2887
000300  005203                  5$:     INC     R3                      ; LBN                                   2915
000302  005201                          INC     R1                      ; PASS                                  2916
000304  022626                          CMP     (SP)+,(SP)+             ;                                       2871
000306  020127  000003                  CMP     R1,#3                   ; PASS,*                                2920
000312  103652                          BLO     1$
000314  005702                          TST     R2                      ; RESULT                                2922
000316  001017                          BNE     6$
000320  012700  000240                  MOV     #240,R0                 ;                                       2926
000324  104441                          TRAP    41
000326  104455                          TRAP    55                      ;                                       2927
000330  000022                          .WORD   22
000332  000000G                         .WORD   EGD.18
000334  000000G                         .WORD   EMS.18
000336  005000                          CLR     R0                      ;                                       2928
000340  104441                          TRAP    41
000342  013700  000000G                 MOV     L$LUN,R0                ;                                       2929
000346  112760  000002  000000G         MOVB    #2,DUR(R0)
000354  104451                          TRAP    51                      ;                                       2930
000356  000207                  6$:     RTS     PC                      ;                                       2844

; Routine Size:  120 words,     Routine Base:  $CODE$ + 3574
; Maximum stack depth per invocation:  10 words
```

```
;        2935  %SBTTL 'DM EXERCISER ROUTINES'
;        2936
;        2937  ROUTINE DM_EXER : NOVALUE =
;        2938
;        2939  !+
;        2940  !       THIS ROUTINE CONTROLS THE OVERALL OPERATION OF THE DM EXERCISER. THE
;        2941  !       DM EXERCISER IS A SUBTEST OF THE HOST EXERCISER. ITS BASIC PURPOSE IS
;        2942  !       TO SUPERVISE THE OPERATION OF THE FRONT PANEL TEST WHICH EXECUTES IN
;        2943  !       THE DM (DIAGNOSTIC MACHINE).
;        2944  !
;        2945  !       THE HOST INITIATES THE FRONT PANEL TEST BY DOWN-LINE LOADING THE CROM
;        2946  !       PRIMER PROGRAM FROM DM_INIT. THIS PROGRAM HAS THE JOB OF LOADING THE
;        2947  !       FRONT PANEL TEST INTO CONTROLLER MEMORY AND PASSING EXECUTION CONTROL
;        2948  !       TO IT. WHEN AND IF THIS SUCCEEDS, THE HOST SENDS DOWN THE ADDRESS OF
;        2949  !       THE DM EXERCSER / FRONT PANEL TEST COMMUNICATION AREA (DM_COMM). THIS
;        2950  !       BLOCK OF MEMORY HOLDS THE UNIT NUMBERS OF DISKS WHICH ARE TO BE
;        2951  !       DM-EXERCISED, AS WELL AS UNIT STATISTICS (LOADED BY THE FRONT PANEL
;        2952  !       TEST AND READ BY THE HOST).
;        2953  !
;        2954  !       THIS ROUTINE DISPATCHES PROCESSING TO OTHER ROUTINES ON A REGULAR
;        2955  !       BASIS UNTIL ALL UNITS HAVE COMPLETED THE REQUIRED NUMBER OF BYTES
;        2956  !       TRANSFERRED. THESE OTHER ROUTINES INCLUDE THE PROCESSING OF DUP END
;        2957  !       MESSAGES (DM_RETPKT), UPDATING HOST STATISTICS FROM DM_COMM (DM_TALLY),
;        2958  !       AND MAINTAINING COMMAND TIMERS (DRV_TIMCHK).
;        2959  !-
;        2960
;        2961  BEGIN
;        2962
;        2963  IF MANUAL                                          ! IF ATTENDED
;        2964  THEN                                              ! THEN
;        2965      PRINTF (MSG_06);                              ! "DM EXERCISER SUBTEST START"
;        2966  DM_INIT ();                                        ! INIT DM_EXER DATA, SEND CROM PRIMER
;        2967
;        2968  DO                                                ! "EXECUTIVE" PROCESSING LOOP
;        2969      BEGIN
;        2970
;        2971      BREAK;                                        ! BREAK IN CASE USER TYPED <CTRL-C>
;        2972      DM_RETPKT ();                                 ! PROCESS ANY DUP END MESSAGES OR "DRIVER" ERRORS
;        2973      IF .T_FLAG EQLU TRUE                          ! IF ONE SECOND HAS ELAPSED
;        2974      THEN                                          ! THEN
;        2975          BEGIN
;        2976
;        2977          INCR CTLR FROM 0 TO (.CTLR_CNT - 1) DO    ! FOR EACH CONFIGURED CONTROLLER
;        2978              BEGIN
;        2979
;        2980              SET_CPAR (.CTLR);                             ! SET UP CONTROLLER-RELATED DATA ITEMS
;        2981              IF ((.CST_ADDR [STATE] EQLU ONLINE) AND       ! IF CONTROLLER IS ONLINE AND
;        2982                  (.CST_ADDR [U_CNT] NEQU 0))               !    THERE IS AT LEAST 1 UNIT UNDER TEST
;        2983              THEN                                          ! THEN
;        2984                  BEGIN
;        2985
;        2986                  IF DM_ACC ()                              ! IF ACCESS GAINED TO DM_COMM AREA
;        2987                  THEN                                      ! THEN
;        2988                      DM_TALLY ();                          ! ADD DM_COMM STATS TO HOST TOTALS
;        2989                  DM_TIME ();                               ! CHECK FPT SANITY TIMER FOR CONTROLLER
;        2990
;        2991                  END;                                  ! IF CTLR IS ONLINE AND AT LEAST ONE UNIT
```

```
;          2992
;          2993                END;                                 ! CONTROLLER LOOP
;          2994
;          2995            DRV_TIMCHK ();                           ! CHECK FOR COMMAND TIMEOUTS
;          2996
;          2997            END;                                     ! IF ONE SECOND HAS ELAPSED
;          2998
;          2999        END                                          ! END OF "EXECUTIVE" LOOP
;          3000    UNTIL DM_QUIT ();                                ! UNTIL ALL UNITS DONE
;          3001
;          3002    END;                                             ! ROUTINE DM_EXER


                                        .SBTTL  DM.EXER DM EXERCISER ROUTINES
000000  004137  000000G          DM.EXER:JSR     R1,$SAVE2                       ;                       2937
000004  104450                          TRAP    50                              ;                       2963
000006  103007                          BHIS    1$
000010  012746  000000G                 MOV     #MSG.06,-(SP)                   ;                       2965
000014  012746  000001                  MOV     #1,-(SP)
000020  010600                          MOV     SP,R0                           ; SP,*
000022  104417                          TRAP    17
000024  022626                          CMP     (SP)+,(SP)+
000026  004737  000000V          1$:     JSR     PC,DM.INIT                      ;                       2966
000032  104422                  2$:     TRAP    22                              ;                       2969
000034  004737  000000V                 JSR     PC,DM.RETPKT                    ;                       2972
000040  123727  000000G 000001          CMPB    T.FLAG,#1                       ;                       2973
000046  001035                          BNE     7$
000050  013702  000000G                 MOV     CTLR.CNT,R2                     ;                       2977
000054  005001                          CLR     R1                              ; CTLR
000056  000425                          BR      6$
000060  010146                  3$:     MOV     R1,-(SP)                        ; CTLR,*                2980
000062  004737  000000G                 JSR     PC,SET.CPAR
000066  013700  000000G                 MOV     CST.ADDR,R0                     ;                       2981
000072  005760  000002                  TST     2(R0)
000076  100013                          BPL     5$
000100  105760  000005                  TSTB    5(R0)                           ;                       2982
000104  001410                          BEQ     5$
000106  004737  000000V                 JSR     PC,DM.ACC                       ;                       2986
000112  006000                          ROR     R0
000114  103002                          BCC     4$
000116  004737  000000V                 JSR     PC,DM.TALLY                     ;                       2988
000122  004737  000000V          4$:     JSR     PC,DM.TIME                      ;                       2989
000126  005726                  5$:     TST     (SP)+                           ;                       2978
000130  005201                          INC     R1                              ; CTLR                  2977
000132  020102                  6$:     CMP     R1,R2                           ; CTLR,*
000134  002751                          BLT     3$
000136  004737  000000V                 JSR     PC,DRV.TIMCHK                   ;                       2995
000142  004737  000000V          7$:     JSR     PC,DM.QUIT                      ;                       3000
000146  006000                          ROR     R0
000150  103330                          BCC     2$
000152  000207                          RTS     PC                              ;                       2937

; Routine Size:  54 words,      Routine Base:  $CODE$ + 4154
; Maximum stack depth per invocation:  7 words
```

```
ROUTINE DM_INIT : NOVALUE =

!+
!       THIS ROUTINE IS CALLED BY DM_EXER WHEN THE DM EXERCISER IS FIRST
!       STARTED. ITS PURPOSE IS (A) TO INITIALIZE DM EXERCISER DATA, AND (B)
!       TO START THE FRONT PANEL TEST EXECUTING IN THE DM.
!
!       THE MOST SIGNIFICANT DATA INITIALIZATION OCCURS IN THE COMMUNICATION
!       AREA (DM_COMM) BETWEEN THE HOST AND THE FRONT PANEL TEST. PLATTER
!       ADDRESSES TO BE EXERCISED UNDER THE FPT ARE LOADED INTO THIS AREA,
!       AND ALL STATISTICS AND ACCESS WORDS ARE CLEARED TO ZERO.
!
!       TO START THE FPT EXECUTING, THIS PROGRAM CAUSES THE CROM PRIMER TO
!       BE DOWN-LINE LOADED INTO THE DM VIA DUP'S ''EXECUTE SUPPLIED PROGRAM''
!       MESSAGE. THE CROM PRIMER THEN LOADS THE FRONT PANEL TEST INTO RAM AND
!       STARTS ITS EXECUTION. THE SUCCESS / FAILURE OF THIS OPERATION IS
!       COMMUNICATED BACK TO THE HOST THROUGH AN END MESSAGE.
!-

BEGIN

LOCAL
    MX : WORD;                                          ! MSCP ENVELOPE INDEX

INCR CTLR FROM 0 TO (.CTLR_CNT - 1) DO                  ! FOR EACH CONFIGURED CONTROLLER
    BEGIN

    SET_CPAR (.CTLR);                                           ! SET UP CONTROLLER-RELATED PARAMETERS
    IF .CST_ADDR [STATE] EQLU ONLINE                            ! IF CONTROLLER IS ONLINE
    THEN                                                        ! THEN
        BEGIN

        INCR COUNT FROM 8 TO ((DMC_LEN * 2) - 2) BY 2 DO        ! CLEAR CTLR'S DM_COMM STATS
            (.DMC_ADDR + .COUNT) = 0;
        INCR OFFSET FROM 0 TO 3 DO                              ! FOR EACH UNIT UNDER CTLR
            BEGIN

            IF .CST_ADDR [.OFFSET + OF_UN, P_STAT] EQLU ONLINE  ! IF UNIT IS ONLINE
            THEN                                                ! THEN LOAD PLATTER ADDRESS
                DMC_ADDR [.OFFSET, ALLBIT] = .CST_ADDR [.OFFSET + OF_UN, P_ADDR]
            ELSE                                                ! OTHERWISE
                DMC_ADDR [.OFFSET, ALLBIT] = -1;                ! DO NOT EXERCISE

            END;                                                ! END UNIT LOOP

        IF .ENTRY_REASON NEQU NEW_PASS                          ! IF START, RESTART, CONT, OR PWR FAIL
        THEN                                                    ! THEN
            BEGIN

            DM_TIMR [.CTLR] = 0;                                ! INIT FPT SANITY TIMER
            MX = GET_ENV (.CTLR);                               ! GET AN MSCP ENVELOPE
            MSCP_ENV [.MX, CONNID] = CID_DUP;                   ! CONNECTION ID (DUP)
            MSCP_ENV [.MX, OPCODE] = OP_ESP;                    ! OPCODE = EXECUTE SUPPLIED PROGRAM
            MSCP_ENV [.MX, DBC_LO] = .CROMP [0];                ! BYTE COUNT
            MSCP_ENV [.MX, DBUF_0] = CROMP;                     ! BUFF DESC OF CROM PRIMER
            MSCP_ENV [.MX, OBUF_0] = CROMP + .CROMP [0];        ! OVERLAY BUFF DESC
            IF SEND (.MX) EQLU FAILURE                          ! ATTEMPT TO SEND. IF FAILURE
```

```
:          3060                THEN                                        ! THEN
:          3061                    PUT_ENV (.MX);                          ! RETURN ENVELOPE TO POOL
:          3062
:          3063                END;                                        ! IF START, RESTART, CONT, OR PWR FAIL
:          3064
:          3065            END;                                            ! END - IF CTLR IS ONLINE
:          3066
:          3067        END;                                                ! END CONTROLLER LOOP
:          3068
:          3069    IF ((.ENTRY_REASON NEQU CONT) AND                       ! IF START, RESTART, OR NEW PASS
:          3070        (.ENTRY_REASON NEQU PWR_FAIL))
:          3071    THEN                                                        ! THEN
:          3072        INCR COUNT FROM 0 TO ((MAX_UNITS * TALLY_LEN) - 1) DO   ! CLEAR ALL STATS
:          3073            TALLY [.COUNT] = 0;
:          3074
:          3075    T_FLAG = FALSE;                                         ! CLEAR ONE SECOND TIMING FLAG
:          3076
:          3077    END;                                                    ! ROUTINE DM_INIT


                                    .SBTTL  DM.INIT DM EXERCISER ROUTINES
000000  004137  000000G         DM.INIT:JSR     R1,$SAVE5                       ;                       3003
000004  013704  000000G                 MOV     CTLR.CNT,R4                     ;                       3027
000010  005003                         CLR     R3                              ; CTLR
000012  000531                         BR      8$
000014  010346                 1$:     MOV     R3,-(SP)                        ; CTLR,*                3030
000016  004737  000000G                 JSR     PC,SET.CPAR
000022  013700  000000G                 MOV     CST.ADDR,R0                     ;                       3031
000026  005760  000002                 TST     2(R0)
000032  100117                         BPL     7$
000034  012700  000010                 MOV     #10,R0                          ; *,COUNT               3035
000040  010001                 2$:     MOV     R0,R1                           ; COUNT,*               3036
000042  063701  000000G                 ADD     DMC.ADDR,R1
000046  005011                         CLR     (R1)
000050  062700  000002                 ADD     #2,R0                           ; *,COUNT               3035
000054  020027  000062                 CMP     R0,#62                          ; COUNT,*
000060  003767                         BLE     2$
000062  005001                         CLR     R1                              ; OFFSET                3037
000064  010102                 3$:     MOV     R1,R2                           ; OFFSET,*              3042
000066  006302                         ASL     R2
000070  063702  000000G                 ADD     DMC.ADDR,R2
000074  010100                         MOV     R1,R0                           ; OFFSET,*              3040
000076  006300                         ASL     R0
000100  063700  000000G                 ADD     CST.ADDR,R0
000104  032760  020000  000006         BIT     #20000,6(R0)
000112  001405                         BEQ     4$
000114  116012  000006                 MOVB    6(R0),(R2)                      ;                       3042
000120  105062  000001                 CLRB    1(R2)
000124  000402                         BR      5$                              ;                       3040
000126  012712  177777         4$:     MOV     #-1,(R2)                        ;                       3044
000132  005201                 5$:     INC     R1                              ; OFFSET                3037
000134  020127  000003                 CMP     R1,#3                           ; OFFSET,*
000140  003751                         BLE     3$
000142  123727  000000G 000005         CMPB    ENTRY.REASON,#5                 ;                       3048
000150  001450                         BEQ     7$
000152  010300                         MOV     R3,R0                           ; CTLR,*                3052
000154  006300                         ASL     R0
```

```
000156  005060  000620'                        CLR     DM.TIMR(R0)
000162  010316                                 MOV     R3,(SP)                 ; CTLR,*                        3053
000164  004737  000000G                        JSR     PC,GET.ENV
000170  010005                                 MOV     R0,R5                   ; *,MX
000172  010516                                 MOV     R5,(SP)                 ; MX,*                          3054
000174  012746  000104                        MOV     #104,-(SP)
000200  004737  000000G                        JSR     PC,BL$MUL
000204  112760  000002  000007G               MOVB    #2,MSCP.ENV+7(R0)
000212  112760  000002  000020G               MOVB    #2,MSCP.ENV+20(R0)      ;                               3055
000220  013760  001716' 000024G               MOV     CROMP,MSCP.ENV+24(R0)   ;                               3056
000226  012760  001716' 000030G               MOV     #CROMP,MSCP.ENV+30(R0)  ;                               3057
000234  016060  000024G 000044G               MOV     MSCP.ENV+24(R0),MSCP.ENV+44(R0) ;                       3058
000242  062760  001716' 000044G               ADD     #CROMP,MSCP.ENV+44(R0)
000250  010516                                 MOV     R5,(SP)                 ; MX,*                          3059
000252  004737  000000G                        JSR     PC,SEND
000256  005700                                 TST     R0
000260  001003                                 BNE     6$
000262  010516                                 MOV     R5,(SP)                 ; MX,*                          3061
000264  004737  000000G                        JSR     PC,PUT.ENV
000270  005726                          6$:    TST     (SP)+                   ;                               3050
000272  005726                          7$:    TST     (SP)+                   ;                               3028
000274  005203                                 INC     R3                      ; CTLR                          3027
000276  020304                          8$:    CMP     R3,R4                   ; CTLR,*
000300  002645                                 BLT     1$
000302  123727  000000G 000003                 CMPB    ENTRY.REASON,#3         ;                               3069
000310  001414                                 BEQ     10$
000312  123727  000000G 000004                 CMPB    ENTRY.REASON,#4         ;                               3070
000320  001410                                 BEQ     10$
000322  005000                                 CLR     R0                      ; COUNT                         3072
000324  005060  000000G               9$:     CLR     TALLY(R0)               ; *(COUNT)                      3073
000330  062700  000002                        ADD     #2,R0                   ; *,COUNT                       3072
000334  020027  001376                        CMP     R0,#1376                ; COUNT,*
000340  003771                                 BLE     9$
000342  105037  000000G               10$:    CLRB    T.FLAG                  ;                               3075
000346  000207                                 RTS     PC                      ;                               3003

; Routine Size:  116 words,    Routine Base:  $CODE$ + 4330
; Maximum stack depth per invocation:  9 words
```

```
;        3078  ROUTINE DM_QUIT =
;        3079
;        3080  !+
;        3081  !       THIS ROUTINE IS CALLED BY THE DM_EXERCISER "EXECUTIVE" (DM_EXER) TO
;        3082  !       DETERMINE WHETHER OR NOT THE DM EXERCISER SUBTEST SHOULD BE TERMINATED.
;        3083  !       ITS PURPOSE IS TO EXAMINE THE STATUS OF THE CURRENT PASS TESTING
;        3084  !       VECTOR (CPT). IF EACH UNIT'S VALUE IN THIS TABLE IS "NO" (INDICATING
;        3085  !       THAT CURRENT PASS TESTING ON THE UNIT IS COMPLETE), THEN THIS ROUTINE
;        3086  !       RETURNS A VALUE OF "TRUE" TO ITS CALLER. IF ANY ONE UNIT HAS A VALUE
;        3087  !       OF "YES", THEN "FALSE" IS RETURNED.
;        3088  !
;        3089  !       IMPLICIT INPUTS:
;        3090  !               L$UNIT - NUMBER OF UNITS CONFIGURED FOR TEST
;        3091  !-
;        3092
;        3093  BEGIN
;        3094
;        3095  INCR UNIT FROM 0 TO (.L$UNIT - 1) DO          ! FOR EACH CONFIGURED UNIT
;        3096      IF .CPT [.UNIT] EQLU YES                  ! IF UNIT STILL UNDER TEST
;        3097      THEN                                      ! THEN
;        3098          RETURN FALSE;                         ! TEST NOT FINISHED
;        3099  RETURN TRUE;                                  ! ALL UNITS DONE
;        3100
;        3101  END;                                          ! ROUTINE DM_QUIT


                                        .SBTTL  DM.QUIT DM EXERCISER ROUTINES
000000  005000                  DM.QUIT:CLR     R0                      ; UNIT                          3095
000002  000405                          BR      2$
000004  126027  000000G 000001 1$:     CMPB    CPT(R0),#1              ; *(UNIT),*                     3096
000012  001407                          BEQ     3$                      ;                               3098
000014  005200                          INC     R0                      ; UNIT                          3095
000016  020037  000000G         2$:     CMP     R0,L$UNIT               ; UNIT,*
000022  002770                          BLT     1$
000024  012700  000001                  MOV     #1,R0                   ;                               3093
000030  000207                          RTS     PC
000032  005000                  3$:     CLR     R0                      ;                               3078
000034  000207                          RTS     PC

; Routine Size:  15 words,      Routine Base:  $CODE$ + 4700
; Maximum stack depth per invocation:  0 words
```

```
;         3102  ROUTINE DM_RETPKT : NOVALUE =
;         3103
;         3104  !+
;         3105  !       THIS ROUTINE IS CALLED BY DM_EXER TO CHECK FOR AND PROCESS ANY RETURN
;         3106  !       PACKETS THAT HAVE BEEN "SENT" BY THE "DRIVER" PORTION OF THE PROGRAM.
;         3107  !       THE I/O DONE QUEUE (IODQ) ACTS AS THE LINK BETWEEN THE TWO PROGRAM
;         3108  !       PARTS; IT HOLDS INDECES OF RETURN PACKETS WHICH REQUIRE PROCESSING.
;         3109  !
;         3110  !       UNDER THE DM EXERCISER SUBTEST, RETURN PACKETS ORIGINATE FROM TWO
;         3111  !       SOURCES:
;         3112  !               1. DUP - DESCRIBING AN END MESSAGE RECEIVED FROM THE RC25 IN
;         3113  !                       RESPONSE TO A HOST COMMAND
;         3114  !               2. THE PROGRAM "DRIVER" - DESCRIBING A CONTROLLER ERROR OR
;         3115  !                       COMMAND TIMEOUT.
;         3116  !-
;         3117
;         3118  BEGIN
;         3119
;         3120  WHILE .IODQ_IN NEQU .IODQ_OUT DO                ! DO UNTIL I/O DONE QUEUE IS EMPTY
;         3121      BEGIN
;         3122
;         3123      RP_INDX = OUT_IODQ ();                      ! GET INDEX OF NEXT RETPKT AND ADVANCE OUT POINTER
;         3124      RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2); ! CALCULATE RETPKT ADDRESS
;         3125      SET_CPAR (.RP_ADDR [CTLR]);                 ! SET UP CURRENT CONTROLLER PARAMETERS
;         3126      L$LUN = .CST_ADDR [OF_UN, P_UNIT];          ! GET UNIT NUMBER OF FIRST UNIT
;         3127      IF .RP_ADDR [CONID] EQLU CID_DRIVER         ! IF RETPKT IS FROM PROGRAM "DRIVER"
;         3128      THEN                                        ! THEN
;         3129          BEGIN
;         3130
;         3131          IF .RP_ADDR [MESTYP] EQLU MT_TIMEOUT    ! IF COMMAND TIMEOUT
;         3132          THEN                                    ! THEN
;         3133              ERRDF (15, EGD_15, EMS_15);         ! "MESSAGE RESPONSE TIMEOUT"
;         3134          DROP_CTLR (.CCTLR, DU_FATAL);           ! DROP CONTROLLER'S UNITS
;         3135          CST_ADDR [STATE] = OFFLINE;             ! MARK CONTROLLER OFFLINE
;         3136
;         3137          END
;         3138      ELSE                                        ! OTHERWISE (RETPKT NOT FROM "DRIVER")
;         3139          BEGIN
;         3140
;         3141          IF .RP_ADDR [CONID] EQLU CID_DUP        ! IF RETPKT IS FROM DUP
;         3142          THEN                                    ! THEN
;         3143              DM_MSG ();                          ! PROCESS THE END MESSAGE
;         3144
;         3145          END;
;         3146
;         3147      PUT_RETPKT (.RP_INDX);                      ! RETURN PACKET TO POOL
;         3148
;         3149      END;                                        ! END - UNTIL I/O DONE QUEUE IS EMPTY
;         3150
;         3151  END;                                            ! ROUTINE DM_RETPKT


                                        .SBTTL  DM.RETPKT DM EXERCISER ROUTINES
000000  010146                  DM.RETPKT:
                                        MOV     R1,-(SP)                        ;3102
000002  023737  000000G 000000G 1$:      CMP     IODQ.IN,IODQ.OUT                ;3120
000010  001506                          BEQ     5$
```

```
000012  004737  000000G                        JSR     PC,OUT.IODQ                     ;                       3123
000016  010037  000000G                        MOV     R0,RP.INDX
000022  010046                                 MOV     R0,-(SP)                        ; RP.INDX,*             3124
000024  012746  000060                         MOV     #60,-(SP)
000030  004737  000000G                        JSR     PC,BL$MUL
000034  062700  000000G                        ADD     #RETPKT,R0
000040  010037  000000G                        MOV     R0,RP.ADDR
000044  116016  000002                         MOVB    2(R0),(SP)                      ;                       3125
000050  042716  177760                         BIC     #177760,(SP)
000054  004737  000000G                        JSR     PC,SET.CPAR
000060  013700  000000G                        MOV     CST.ADDR,R0                     ;                       3126
000064  016001  000006                         MOV     6(R0),R1
000070  000301                                 SWAB    R1
000072  042701  177740                         BIC     #177740,R1
000076  010137  000000G                        MOV     R1,L$LUN
000102  013700  000000G                        MOV     RP.ADDR,R0                      ;                       3127
000106  126027  000003  000003                 CMPB    3(R0),#3
000114  001030                                 BNE     3$
000116  116000  000002                         MOVB    2(R0),R0                        ;                       3131
000122  042700  177417                         BIC     #177417,R0
000126  020027  000100                         CMP     R0,#100
000132  001004                                 BNE     2$
000134  104455                                 TRAP    55                              ;                       3133
000136  000017                                 .WORD   17
000140  000000G                                .WORD   EGD.15
000142  000000G                                .WORD   EMS.15
000144  013716  000000G                2$:     MOV     CCTLR,(SP)                      ;                       3134
000150  012746  000004                         MOV     #4,-(SP)
000154  004737  000000G                        JSR     PC,DROP.CTLR
000160  013700  000000G                        MOV     CST.ADDR,R0                     ;                       3135
000164  042760  100000  000002                 BIC     #100000,2(R0)
000172  005726                                 TST     (SP)+                           ;                       3129
000174  000406                                 BR      4$                              ;                       3127
000176  126027  000003  000002         3$:     CMPB    3(R0),#2                        ;                       3141
000204  001002                                 BNE     4$
000206  004737  000000V                        JSR     PC,DM.MSG                       ;                       3143
000212  013716  000000G                4$:     MOV     RP.INDX,(SP)                    ;                       3147
000216  004737  000000G                        JSR     PC,PUT.RETPKT
000222  022626                                 CMP     (SP)+,(SP)+                     ;                       3121
000224  000666                                 BR      1$                              ;                       3120
000226  012601                         5$:     MOV     (SP)+,R1                        ;                       3102
000230  000207                                 RTS     PC

; Routine Size:  77 words,      Routine Base:  $CODE$ + 4736
; Maximum stack depth per invocation:  5 words
```

```
;       3152  ROUTINE DM_MSG : NOVALUE =
;       3153
;       3154  !+
;       3155  !       THIS ROUTINE IS CALLED BY DM_RETPKT FOR ALL DUP END MESSAGES RECEIVED
;       3156  !       BY THE DM EXERCISER.
;       3157  !
;       3158  !       THIS ROUTINE FIRST CHECKS THE STATUS CODE OF THE END MESSAGE; IF IT
;       3159  !       INDICATES ANYTHING OTHER THAN SUCCESSFUL COMPLETION OF THE COMMAND,
;       3160  !       THEN A DEVICE-FATAL ERROR IS DECLARED FOR THE CONTROLLER, AND ALL ITS
;       3161  !       UNITS ARE DROPPED. IF THE STATUS CODE INDICATES SUCCESS, THEN
;       3162  !       PROCESSING IS DEPENDENT ON THE ENDCODE (OPCODE OF THE ASSOCIATED
;       3163  !       COMMAND). FOR THE "EXECUTE SUPPLIED PROGRAM" END MESSAGE, THIS ROUTINE
;       3164  !       FORMATS AND SENDS A "SEND DATA" MESSAGE, PROVIDING THE FRONT PANEL TEST
;       3165  !       WITH THE HOST ADDRESS OF THE CURRENT CONTROLLER'S DM EXERCISER
;       3166  !       COMMUNICATION BLOCK (DM_COMM).
;       3167  !
;       3168  !       IMPLICIT INPUTS:
;       3169  !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;       3170  !               CCTLR - CURRENT CONTROLLER NUMBER
;       3171  !               DMC_ADDR - ADDRESS OF CURRENT CONTROLLER'S DM_COMM AREA
;       3172  !-
;       3173
;       3174  BEGIN
;       3175
;       3176  LOCAL
;       3177      MX : WORD,                                    ! MSCP ENVELOPE INDEX
;       3178      DUP_OP : WORD,                                ! DUP OPCODE
;       3179      UNIT : WORD;                                  ! UNIT NUMBER
;       3180
;       3181  IF (ST_CODE = .RP_ADDR [STATUS]) NEQU ST_SUC     ! IF STATUS CODE NOT SUCCESS
;       3182  THEN                                              ! THEN
;       3183      BEGIN
;       3184
;       3185      ERRDF (21, EGD_21, EMS_21);                   ! "DUP COMMAND FAILED"
;       3186      DROP_CTLR (.CCTLR, DU_FATAL);                 ! DROP CONTROLLER'S UNITS
;       3187
;       3188      END
;       3189  ELSE                                              ! OTHERWISE - SUCCESS
;       3190      BEGIN
;       3191
;       3192      DUP_OP = .RP_ADDR [ENDCOD] AND OP_MSK;        ! GET OPCODE
;       3193      SELECTONEU .DUP_OP OF
;       3194          SET
;       3195
;       3196          [OP_ESP] : BEGIN                                  ! IF "EXECUTE SUPPLIED PROGRAM"
;       3197
;       3198                     MX = GET_ENV (.CCTLR);                 ! GET AN MSCP ENVELOPE
;       3199                     MSCP_ENV [.MX, CONNID] = CID_DUP;      ! CONNECTION ID (DUP)
;       3200                     MSCP_ENV [.MX, OPCODE] = OP_SND;       ! OPCODE = SEND DATA
;       3201                     MSCP_ENV [.MX, DBC_LO] = 2;            ! BYTE COUNT
;       3202                     MSCP_ENV [.MX, DBUF_0] = DMC_ADDR;     ! ADDR OF ADDR OF CTLR'S DM_COMM AREA
;       3203                     IF SEND (.MX) EQLU FAILURE             ! ATTEMPT TO SEND. IF FAILURE
;       3204                     THEN                                   ! THEN
;       3205                         PUT_ENV (.MX);                     ! RETURN ENVELOPE TO POOL
;       3206
;       3207                     END;                                   ! END - IF "EXECUTE SUPPLIED PROGRAM"
;       3208
```

```
;         3209          [OP_SND] : DM_TIMR [.CCTLR] = TO_DM;          ! IF "SEND DATA", START FPT SANITY TIMER
;         3210
;         3211          TES;
;         3212
;         3213      END;                                              ! END - IF STATUS CODE = SUCCESS
;         3214
;         3215  END;                                                  ! ROUTINE DM_MSG


                                        .SBTTL  DM.MSG DM EXERCISER ROUTINES
000000  010146                  DM.MSG: MOV     R1,-(SP)                ;                               3152
000002  013700  000000G                 MOV     RP.ADDR,R0              ;                               3181
000006  016037  000016  000000G         MOV     16(R0),ST.CODE
000014  001413                          BEQ     1$
000016  104455                          TRAP    55                      ;                               3185
000020  000025                          .WORD   25
000022  000000G                         .WORD   EGD.21
000024  000000G                         .WORD   EMS.21
000026  013746  000000G                 MOV     CCTLR,-(SP)             ;                               3186
000032  012746  000004                  MOV     #4,-(SP)
000036  004737  000000G                 JSR     PC,DROP.CTLR
000042  000447                          BR      2$                      ;                               3183
000044  013700  000000G          1$:    MOV     RP.ADDR,R0              ;                               3192
000050  116000  000014                  MOVB    14(R0),R0               ; *,DUP.OP
000054  042700  177600                  BIC     #177600,R0              ; *,DUP.OP
000060  020027  000002                  CMP     R0,#2                   ; DUP.OP,*                      3193
000064  001040                          BNE     3$
000066  013746  000000G                 MOV     CCTLR,-(SP)             ;                               3198
000072  004737  000000G                 JSR     PC,GET.ENV
000076  010001                          MOV     R0,R1                   ; *,MX
000100  010116                          MOV     R1,(SP)                 ; MX,*                          3199
000102  012746  000104                  MOV     #104,-(SP)
000106  004737  000000G                 JSR     PC,BL$MUL
000112  112760  000002  000007G         MOVB    #2,MSCP.ENV+7(R0)
000120  112760  000004  000020G         MOVB    #4,MSCP.ENV+20(R0)      ;                               3200
000126  012760  000002  000024G         MOV     #2,MSCP.ENV+24(R0)      ;                               3201
000134  012760  000000G 000030G         MOV     #DMC.ADDR,MSCP.ENV+30(R0) ;                             3202
000142  010116                          MOV     R1,(SP)                 ; MX,*                          3203
000144  004737  000000G                 JSR     PC,SEND
000150  005700                          TST     R0
000152  001003                          BNE     2$
000154  010116                          MOV     R1,(SP)                 ; MX,*                          3205
000156  004737  000000G                 JSR     PC,PUT.ENV
000162  022626                  2$:     CMP     (SP)+,(SP)+             ;                               3196
000164  000411                          BR      4$                      ;                               3193
000166  020027  000004          3$:     CMP     R0,#4                   ; DUP.OP,*
000172  001006                          BNE     4$
000174  013700  000000G                 MOV     CCTLR,R0                ;                               3209
000200  006300                          ASL     R0
000202  012760  000550  000620'         MOV     #550,DM.TIMR(R0)
000210  012601                  4$:     MOV     (SP)+,R1                ;                               3152
000212  000207                          RTS     PC

; Routine Size:  70 words,      Routine Base:  $CODE$ + 5170
; Maximum stack depth per invocation:  4 words
```

```
;         3216  ROUTINE DM_ACC =
;         3217
;         3218  !+
;         3219  !       THIS ROUTINE IS CALLED BY DM_EXER IN ORDER TO ATTEMPT TO GAIN ACCESS
;         3220  !       TO THE CURRENT CONTROLLER'S DM_COMM AREA. IF ACCESS IS PERMITTED (I.E.,
;         3221  !       IF THE FRONT PANEL TEST (FPT) IS NOT CURRENTLY UPDATING DM_COMM
;         3222  !       STATISTICS), THEN THIS ROUTINE RETURNS A VALUE OF "SUCCESS". OTHERWISE,
;         3223  !       ACCESS IS DENIED AT THIS TIME, AND "FAILURE" IS RETURNED.
;         3224  !
;         3225  !       IMPLICIT INPUTS:
;         3226  !               DMC_ADDR - ADDRESS OF CURRENT CONTROLLER'S DM_COMM AREA
;         3227  !
;         3228  !       OUTPUTS:
;         3229  !               "SUCCESS" - ACCESS TO DM_COMM IS PERMITTED
;         3230  !               "FAILURE" - ACCESS TO DM_COMM IS DENIED
;         3231  !-
;         3232
;         3233  BEGIN
;         3234
;         3235  LOCAL
;         3236      RESULT : WORD;                          ! RESULT OF ACCESS REQUEST
;         3237
;         3238  RESULT = SUCCESS;                          ! ASSUME ACCESS WILL BE GAINED
;         3239  DMC_ADDR [HOST_ACC] = ALL_ONES;            ! SET HOST ACCESS WORD
;         3240  IF .DMC_ADDR [FPT_ACC] NEQU 0              ! IF FPT HAS ACCESS NOW
;         3241  THEN                                       ! THEN
;         3242      BEGIN
;         3243
;         3244      DMC_ADDR [HOST_ACC] = 0;              ! WITHDRAW ACCESS REQUEST
;         3245      RESULT = FAILURE;                     ! CHANGE RESULT TO FAILURE
;         3246
;         3247      END;
;         3248
;         3249  RETURN .RESULT;
;         3250
;         3251  END;                                       ! ROUTINE DM_ACC


                                .SBTTL  DM.ACC DM EXERCISER ROUTINES
000000  004137  000000G         DM.ACC: JSR     R1,$SAVE2                       ;                       3216
000004  012702  000001                 MOV     #1,R2                           ; *,RESULT              3238
000010  013701  000000G                MOV     DMC.ADDR,R1                     ;                       3239
000014  012761  177777  000062         MOV     #-1,62(R1)
000022  010100                         MOV     R1,R0                           ; DMC.ADDR,*            3240
000024  005760  000060                 TST     60(R0)
000030  001403                         BEQ     1$
000032  005061  000062                 CLR     62(R1)                          ;                       3244
000036  005002                         CLR     R2                              ; RESULT                3245
000040  010200                 1$:     MOV     R2,R0                           ; RESULT,*              3233
000042  000207                         RTS     PC                              ;                       3216

; Routine Size:  18 words,      Routine Base:  $CODE$ + 5404
; Maximum stack depth per invocation:  4 words
```

```
;        3252  ROUTINE DM_TALLY : NOVALUE =
;        3253
;        3254  !+
;        3255  !       THIS ROUTINE IS CALLED FROM DM_EXER AFTER ACCESS HAS BEEN GAINED TO
;        3256  !       THE CURRENT CONTROLLER'S DM_COMM AREA. ITS PURPOSE IS TO EXTRACT THE
;        3257  !       STATISTICAL VALUES STORED THERE BY THE FRONT PANEL TEST (EXECUTING IN
;        3258  !       THE DM), AND TO ADD THESE VALUES TO THE HOST-KEPT STATS. THE STATS IN
;        3259  !       DM_COMM ARE THEN RESET TO 0.
;        3260  !
;        3261  !       AFTER UPDATING THE HOST STATS, THE NUMBER OF HARD ERRORS AND THE NUMBER
;        3262  !       OF MBYTES TRANSFERRED THUS FAR FOR EACH UNIT IS CHECKED AGAINST THE
;        3263  !       LIMITS SPECIFIED IN THE SW P-TABLE. IF THE HARD ERROR LIMIT HAS BEEN
;        3264  !       REACHED, THEN THE UNIT IS DROPPED FROM ALL TESTING. IF THE TRANSFER
;        3265  !       LIMIT HAS BEEN REACHED, THEN THE UNIT IS SIMPLY REMOVED FROM THE
;        3266  !       CURRENT PASS.
;        3267  !
;        3268  !       IMPLICIT INPUTS:
;        3269  !               DMC_ADDR - ADDRESS OF THE CURRENT CONTROLLER'S DM_COMM AREA
;        3270  !-
;        3271
;        3272  BEGIN
;        3273
;        3274  LOCAL
;        3275      ST_PTR,                                   ! STATISTICS POINTER IN DM_COMM
;        3276      NBXFR : WORD;                             ! NUMBER OF BLOCKS TRANSFERRED
;        3277
;        3278  ST_PTR = .DMC_ADDR + 8;                      ! POINTS TO STATS FOR FIRST UNIT
;        3279  INCR OFFSET FROM 0 TO 3 DO                   ! FOR EACH UNIT
;        3280      BEGIN
;        3281
;        3282      IF .DMC_ADDR [.OFFSET, 0, 16, 1] GEQ 0    ! IF UNIT IS BEING DM-EXERCISED
;        3283      THEN                                      ! THEN
;        3284          BEGIN
;        3285
;        3286          SET_UPAR (.OFFSET + OF_UN);           ! SET UP UNIT-RELATED PARAMETERS
;        3287
;        3288          UPD_IOC (T_ADDR [SEEK_LO], ..ST_PTR); ! UPDATE NO. OF SEEKS
;        3289          .ST_PTR = 0;                          ! RESET STAT TO 0
;        3290          ST_PTR = .ST_PTR + 2;                 ! ADVANCE DM_COMM POINTER
;        3291
;        3292          NBXFR = ..ST_PTR;                     ! NUMBER OF BLOCKS READ (1 BLOCK PER READ)
;        3293          UPD_IOC (T_ADDR [READ_LO], ..ST_PTR); ! UPDATE NO. OF READS
;        3294          UPD_DMBC (T_ADDR [BR_LO], ..ST_PTR);  ! UPDATE NO. OF BYTES READ, CHECK FOR OVERFLOW
;        3295          .ST_PTR = 0;                          ! RESET STAT TO 0
;        3296          ST_PTR = .ST_PTR + 2;                 ! ADVANCE DM_COMM POINTER
;        3297
;        3298          NBXFR = .NBXFR + ..ST_PTR;            ! ADD NUMBER OF BLOCKS WRITTEN (1 BLOCK PER WRITE)
;        3299          UPD_IOC (T_ADDR [WRIT_LO], ..ST_PTR); ! UPDATE NO. OF WRITES
;        3300          UPD_DMBC (T_ADDR [BW_LO], ..ST_PTR);  ! UPDATE NO. OF BYTES WRITTEN, CHECK FOR OVERFLOW
;        3301          .ST_PTR = 0;                          ! RESET STAT TO 0
;        3302          ST_PTR = .ST_PTR + 2;                 ! ADVANCE DM_COMM POINTER
;        3303
;        3304          T_ADDR [ER_SFT] = .T_ADDR [ER_SFT] + ..ST_PTR;  ! UPDATE NO. OF DM SOFT ERRORS
;        3305          .ST_PTR = 0;                          ! RESET STAT TO 0
;        3306          ST_PTR = .ST_PTR + 2;                 ! ADVANCE DM_COMM POINTER
;        3307
;        3308          HARD_ERR (..ST_PTR);                  ! UPDATE HRD ERR CNT + CHK FOR LIMIT
```

```
:          3309          .ST_PTR = 0;                                    ! RESET STAT TO 0
:          3310          ST_PTR = .ST_PTR + 2;                           ! ADVANCE DM_COMM POINTER
:          3311
:          3312          IF .NBXFR NEQU 0                                ! IF A COMPLETE PASS HAS BEEN MADE
:          3313          THEN                                            ! THEN
:          3314              BEGIN
:          3315
:          3316              IF MANUAL                                       ! IF ATTENDED
:          3317              THEN                                            ! THEN
:          3318                  PRINTF (MSG_08, .NBXFR, .L$LUN, .CPLAT);        ! "XXXXX. BLOCKS TRANSFERRED ON UNIT XX. (PLATTER XX
:          3319              DM_TIMR [.CCTLR] = TO_DM;                       ! RESET FPT SANITY TIMER
:          3320
:          3321              END;
:          3322
:          3323              XFR_CHK ();                                 ! CHECK IF TRANSFER LIMIT REACHED
:          3324
:          3325              END                                         ! END - IF UNIT IS BEING DM-EXERCISED
:          3326          ELSE                                            ! OTHERWISE
:          3327              ST_PTR = .ST_PTR + 10;                      ! ADVANCE DM_COMM POINTER TO NEXT BLOCK
:          3328
:          3329          END;                                            ! UNIT LOOP
:          3330
:          3331      DMC_ADDR [HOST_ACC] = 0;                            ! TELL FPT THAT WE ARE LEAVING
:          3332
:          3333      END;                                                ! ROUTINE DM_TALLY


                                .SBTTL  DM.TALLY DM EXERCISER ROUTINES
000000  004137  000000G     DM.TALLY:
                            JSR     R1,$SAVE3                   ;                       3252
000004  013703  000000G             MOV     DMC.ADDR,R3         ; *,ST.PTR              3278
000010  062703  000010              ADD     #10,R3              ; *,ST.PTR
000014  005001                      CLR     R1                  ; OFFSET                3279
000016  010100              1$:     MOV     R1,R0               ; OFFSET,*              3282
000020  006300                      ASL     R0
000022  063700  000000G             ADD     DMC.ADDR,R0
000026  005710                      TST     (R0)
000030  002520                      BLT     4$
000032  010146                      MOV     R1,-(SP)            ; OFFSET,*              3286
000034  062716  000003              ADD     #3,(SP)
000040  004737  000000G             JSR     PC,SET.UPAR
000044  013716  000000G             MOV     T.ADDR,(SP)         ;                       3288
000050  062716  000010              ADD     #10,(SP)
000054  011346                      MOV     (R3),-(SP)          ; ST.PTR,*
000056  004737  000000G             JSR     PC,UPD.IOC
000062  005023                      CLR     (R3)+               ; ST.PTR                3289
000064  011302                      MOV     (R3),R2             ; ST.PTR,NBXFR          3292
000066  013716  000000G             MOV     T.ADDR,(SP)         ;                       3293
000072  010246                      MOV     R2,-(SP)
000074  004737  000000G             JSR     PC,UPD.IOC
000100  013716  000000G             MOV     T.ADDR,(SP)         ;                       3294
000104  062716  000014              ADD     #14,(SP)
000110  011346                      MOV     (R3),-(SP)          ; ST.PTR,*
000112  004737  000000V             JSR     PC,UPD.DMBC
000116  005023                      CLR     (R3)+               ; ST.PTR                3295
000120  061302                      ADD     (R3),R2             ; ST.PTR,NBXFR          3298
000122  013716  000000G             MOV     T.ADDR,(SP)         ;                       3299
```

```
000126  062716  000004                          ADD     #4,(SP)
000132  011346                                  MOV     (R3),-(SP)          ; ST.PTR,*
000134  004737  000000G                         JSR     PC,UPD.IOC
000140  013716  000000G                         MOV     T.ADDR,(SP)         ;                   3300
000144  062716  000022                          ADD     #22,(SP)
000150  011346                                  MOV     (R3),-(SP)          ; ST.PTR,*
000152  004737  000000V                         JSR     PC,UPD.DMBC
000156  005023                                  CLR     (R3)+               ; ST.PTR            3301
000160  013700  000000G                         MOV     T.ADDR,R0           ;                   3304
000164  061360  000034                          ADD     (R3),34(R0)         ; ST.PTR,*
000170  005023                                  CLR     (R3)+               ; ST.PTR            3305
000172  011316                                  MOV     (R3),(SP)           ; ST.PTR,*          3308
000174  004737  000000G                         JSR     PC,HARD.ERR
000200  005023                                  CLR     (R3)+               ; ST.PTR            3309
000202  005702                                  TST     R2                  ; NBXFR             3312
000204  001425                                  BEQ     3$
000206  104450                                  TRAP    50                  ;                   3316
000210  103015                                  BHIS    2$
000212  013716  000000G                         MOV     CPLAT,(SP)          ;                   3318
000216  013746  000000G                         MOV     L$LUN,-(SP)
000222  010246                                  MOV     R2,-(SP)            ; NBXFR,*
000224  012746  000000G                         MOV     #MSG.08,-(SP)
000230  012746  000004                          MOV     #4,-(SP)
000234  010600                                  MOV     SP,R0               ; SP,*
000236  104417                                  TRAP    17
000240  062706  000010                          ADD     #10,SP
000244  013700  000000G         2$:             MOV     CCTLR,R0            ;                   3319
000250  006300                                  ASL     R0
000252  012760  000550  000620'                 MOV     #550,DM.TIMR(R0)
000260  004737  000000G         3$:             JSR     PC,XFR.CHK          ;                   3323
000264  062706  000014                          ADD     #14,SP              ;                   3284
000270  000402                                  BR      5$                  ;                   3282
000272  062703  000012         4$:             ADD     #12,R3              ; *,ST.PTR          3327
000276  005201                 5$:             INC     R1                  ; OFFSET            3279
000300  020127  000003                          CMP     R1,#3               ; OFFSET,*
000304  003644                                  BLE     1$
000306  013700  000000G                         MOV     DMC.ADDR,R0         ;                   3331
000312  005060  000062                          CLR     62(R0)
000316  000207                                  RTS     PC                  ;                   3252

; Routine Size:  104 words,    Routine Base:  $CODE$ + 5450
; Maximum stack depth per invocation:  16 words
```

```
;         3334  ROUTINE UPD_DMBC (ADDR, COUNT) : NOVALUE =
;         3335
;         3336  !+
;         3337  !       THIS ROUTINE IS CALLED FROM DM_TALLY TO UPDATE THE TOTAL NUMBER OF
;         3338  !       BYTES READ OR WRITTEN BY THE FRONT PANEL TEST (FPT) FOR THE CURRENT
;         3339  !       UNIT. IT IS ASSUMED THAT EACH READ OR WRITE OPERATION PERFORMED BY THE
;         3340  !       FPT IS EXACTLY ONE BLOCK (512 BYTES) LONG. (A SIMPLE MULTIPLICATION
;         3341  !       (.COUNT * BLK_SIZE) IS NOT FEASIBLE BECAUSE A 16-BIT OVERFLOW WOULD
;         3342  !       OCCUR).
;         3343  !
;         3344  !       INPUTS:
;         3345  !               ADDR - TALLY ADDRESS OF THE LOW-ORDER FIELD OF THE NUMBER OF
;         3346  !                      BYTES READ OR WRITTEN FOR THE CURRENT UNIT
;         3347  !               COUNT - THE NUMBER OF READ OR WRITE OPERATIONS PERFORMED BY THE
;         3348  !                       FPT
;         3349  !-
;         3350
;         3351  BEGIN
;         3352
;         3353  INCR I FROM 1 TO .COUNT DO                    ! FOR EACH FPT I/O OPERATION
;         3354      BEGIN
;         3355
;         3356      .ADDR = ..ADDR + BLK_SIZE;                ! UPDATE TOTAL NO. OF BYTES (LOW ORDER)
;         3357      IF ..ADDR GEQU 50000                      ! IF LOW ORDER VALUE IS APPROACHING OVERFLOW
;         3358      THEN                                      ! THEN
;         3359          OVF_CHK (.ADDR);                      ! REDUCE LOW ORDER WORD, UPDATE HIGH ORDER WORDS
;         3360
;         3361      END;                                      ! END - I/O OPERATION COUNT LOOP
;         3362
;         3363  OVF_CHK (.ADDR);                             ! ONE MORE TIME
;         3364
;         3365  END;                                         ! ROUTINE UPD_DMBC


                                .SBTTL  UPD.DMBC DM EXERCISER ROUTINES
000000  004137  000000G  UPD.DMBC:
                        JSR     R1,$SAVE2                   ;                       3334
000004  016601  000012          MOV     12(SP),R1           ; ADDR,*                3356
000010  005002                  CLR     R2                  ; I                     3353
000012  000411                  BR      2$
000014  062711  001000  1$:     ADD     #1000,(R1)          ;                       3356
000020  021127  141520          CMP     (R1),#141520        ;                       3357
000024  103404                  BLO     2$
000026  010146                  MOV     R1,-(SP)            ;                       3359
000030  004737  000000G          JSR     PC,OVF.CHK
000034  005726                  TST     (SP)+
000036  005202          2$:     INC     R2                  ; I                     3353
000040  020266  000010          CMP     R2,10(SP)           ; I,COUNT
000044  003763                  BLE     1$
000046  010146                  MOV     R1,-(SP)            ;                       3363
000050  004737  000000G          JSR     PC,OVF.CHK
000054  005726                  TST     (SP)+               ;                       3351
000056  000207                  RTS     PC                  ;                       3334

; Routine Size:  24 words,      Routine Base:  $CODE$ + 5770
; Maximum stack depth per invocation:  5 words
```

```
;         3366  ROUTINE DM_TIME : NOVALUE =
;         3367
;         3368  !+
;         3369  !       THIS ROUTINE IS CALLED ONCE PER SECOND FOR EACH CONFIGURED CONTROLLER
;         3370  !       BY THE DM EXERCISER "EXECUTIVE" (DM_EXER). ITS PURPOSE IS TO MAINTAIN
;         3371  !       THE SANITY TIMER ON THE FRONT PANEL TEST (FPT) WHICH IS EXECUTING IN
;         3372  !       THE DIAGNOSTIC MACHINE (DM).
;         3373  !
;         3374  !       THE FPT IS EXPECTED TO PROVIDE THE HOST WITH A NEW SET OF OPERATING
;         3375  !       STATISTICS IN DM_COMM AT REGULAR INTERVALS. EACH CONTROLLER HAS AN
;         3376  !       ENTRY IN DM_TIMR WHICH CONTAINS THE REMAINING TIME (IN SECONDS) TO
;         3377  !       RECEIVE A NEW SET OF STATS. IF THE TIMER FOR THE CURRENT CONTROLLER
;         3378  !       IS ACTIVE, THEN THIS ROUTINE DECREMENTS THE TIMER, AND DECLARES AN
;         3379  !       ERROR IF THE TIMER REACHES ZERO.
;         3380  !
;         3381  !       INPLICIT INPUTS:
;         3382  !               CCTLR - CURRENT CONTROLLER NUMBER
;         3383  !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;         3384  !-
;         3385
;         3386  BEGIN
;         3387
;         3388  IF .DM_TIMR [.CCTLR] NEQU 0                       ! IF TIMER IS ACTIVE
;         3389  THEN                                               ! THEN
;         3390      BEGIN
;         3391
;         3392      DM_TIMR [.CCTLR] = .DM_TIMR [.CCTLR] - 1;     ! DECREMENT
;         3393      IF .DM_TIMR [.CCTLR] EQLU 0                   ! IF TIME HAS EXPIRED
;         3394      THEN                                          ! THEN
;         3395          BEGIN
;         3396
;         3397          L$UNIT = .CST_ADDR [OF_UN, P_UNIT];       ! GET DS UNIT NUMBER OF FIRST PLATTER UNDER CONTROLLER
;         3398          ERRDF (22, EGD_22, EMS_22);               ! "DM EXERCISER TIMEOUT"
;         3399          DROP_CTLR (.CCTLR, DU_FATAL);             ! DROP ALL CONTROLLER'S UNITS
;         3400          CST_ADDR [STATE] = OFFLINE;               ! MARK CONTROLLER OFFLINE
;         3401
;         3402          END;
;         3403
;         3404      END;
;         3405
;         3406  END;                                              ! ROUTINE DM_TIME


                                .SBTTL  DM.TIME DM EXERCISER ROUTINES
000000  010146                  DM.TIME:MOV     R1,-(SP)                        ;                       3366
000002  013700  000000G                 MOV     CCTLR,R0                        ;                       3388
000006  006300                          ASL     R0
000010  062700  000620'                  ADD     #DM.TIMR,R0
000014  005710                          TST     (R0)
000016  001433                          BEQ     1$
000020  005310                          DEC     (R0)                            ;                       3392
000022  001031                          BNE     1$                              ;                       3393
000024  013700  000000G                 MOV     CST.ADDR,R0                     ;                       3397
000030  016001  000006                  MOV     6(R0),R1
000034  000301                          SWAB    R1
000036  042701  177740                  BIC     #177740,R1
000042  010137  000000G                 MOV     R1,L$UNIT
```

```
000046  104455                          TRAP    55                              ;                               3398
000050  000026                          .WORD   26
000052  000000G                         .WORD   EGD.22
000054  000000G                         .WORD   EMS.22
000056  013746  000000G                 MOV     CCTLR,-(SP)                     ;                               3399
000062  012746  000004                  MOV     #4,-(SP)
000066  004737  000000G                 JSR     PC,DROP.CTLR
000072  013700  000000G                 MOV     CST.ADDR,R0                     ;                               3400
000076  042760  100000  000002          BIC     #100000,2(R0)
000104  022626                          CMP     (SP)+,(SP)+                     ;                               3395
000106  012601                  1$:     MOV     (SP)+,R1                        ;                               3366
000110  000207                          RTS     PC

; Routine Size:  37 words,      Routine Base:  $CODE$ + 6050
; Maximum stack depth per invocation:  4 words
```

```
;       3407  %SBTTL 'MULTI-DRIVE SUBTEST ROUTINES'
;       3408
;       3409  ROUTINE MULTI_DRIVE : NOVALUE =
;       3410
;       3411  !+
;       3412  !       THIS SUBTEST IS THE MOST SIGNIFICANT PART OF THE ENTIRE PROGRAM. THE
;       3413  !       MULTI-DRIVE SUBTEST IS A HOST-CONTROLLED EXERCISER DESIGNED TO GIVE THE
;       3414  !       USER AN INDICATION OF HOW ONE OR SEVERAL RC25 DRIVES WOULD PERFORM IN
;       3415  !       AN OPERATING SYSTEM ENVIRONMENT.
;       3416  !
;       3417  !       THIS ROUTINE ACTS AS AN "EXECUTIVE" TO THE WHOLE PROCESS. AFTER
;       3418  !       INVOKING MD_INIT TO INITIALIZE MULTI-DRIVE SUBTEST DATA, THIS ROUTINE
;       3419  !       ENTERS A LOOP WHICH ISSUES QIOS TO ALL ACTIVE CONTROLLERS AND PROCESSES
;       3420  !       ANY RESPONSES. IN ADDITION, ALL OUTSTANDING COMMANDS ARE TIMED IN
;       3421  !       DRV_TIMCHK WHICH IS INVOKED EVERY SECOND. NORMAL TERMINATION OF THIS
;       3422  !       LOOP OCCURS WHEN QIOS ARE NO LONGER BEING ISSUED, AND ALL OUTSTANDING
;       3423  !       QIOS HAVE COMPLETED.
;       3424  !-
;       3425
;       3426  BEGIN
;       3427
;       3428  IF MANUAL                                    ! IF ATTENDED
;       3429  THEN                                         ! THEN
;       3430      PRINTF (MSG_05);                         ! "MULTI-DRIVE SUBTEST START"
;       3431  MD_INIT ();                                  ! INIT MULTI-DRIVE SUBTEST DATA
;       3432
;       3433  DO                                           ! START OF EXECUTIVE LOOP
;       3434      BEGIN
;       3435
;       3436      INCR CTLR FROM 0 TO (.CTLR_CNT - 1) DO   ! FOR EACH CONTROLLER
;       3437          BEGIN
;       3438
;       3439          SET_CPAR (.CTLR);                    ! SET UP CURRENT CONTROLLER PARAMETERS
;       3440          IF QIO_OK () EQLU TRUE               ! IF O.K. TO ISSUE QIO(S) TO THIS CONTROLLER
;       3441          THEN                                 ! THEN
;       3442              BEGIN
;       3443
;       3444              QIO_GEN ();                      ! GENERATE 1 OR 2 QIOS
;       3445              IF .MX1 GEQ 0                    ! IF SUCCESS ON FIRST QIO
;       3446              THEN                             ! THEN
;       3447                  BEGIN
;       3448
;       3449                  IF SEND (.MX1) EQLU SUCCESS          ! ATTEMPT TO SEND IT. IF SUCCESS
;       3450                  THEN                                 ! THEN
;       3451                      QIO [.CTLR] = .QIO [.CTLR] + 1   ! INCR OUTSTANDING QIO COUNT
;       3452                  ELSE                                 ! OTHERWISE
;       3453                      PUT_ENV (.MX1);                  ! RETURN ENVELOPE TO POOL
;       3454
;       3455                  END;
;       3456
;       3457              IF .MX2 GEQ 0                    ! IF SUCCESS ON SECOND QIO
;       3458              THEN                             ! THEN
;       3459                  BEGIN
;       3460
;       3461                  IF SEND (.MX2) EQLU SUCCESS          ! ATTEMPT TO SEND IT. IF SUCCESS
;       3462                  THEN                                 ! THEN
;       3463                      QIO [.CTLR] = .QIO [.CTLR] + 1   ! INCR OUTSTANDING QIO COUNT
```

```
;        3464                    ELSE                                    ! OTHERWISE
;        3465                        PUT_ENV (.MX2);                     ! RETURN ENVELOPE TO POOL
;        3466
;        3467                    END;
;        3468
;        3469                END;                                        ! O.K. TO ISSUE QIO(S)
;        3470
;        3471            IF .DCT_ADDR [STAT] EQLU ONLINE                 ! IF CONTROLLER IS ONLINE
;        3472            THEN                                            ! THEN
;        3473                INT_PROC ();                                ! PROCESS ANY INTERRUPTS
;        3474
;        3475            END;                                            ! CONTROLLER LOOP
;        3476
;        3477        PROC_RETPKT ();                                     ! PROCESS ANY RETURN PACKETS
;        3478        IF .T_FLAG EQLU TRUE                                ! IF ONE SECOND HAS ELAPSED
;        3479        THEN                                                ! THEN
;        3480            DRV_TIMCHK ();                                  ! CHECK OUTSTANDING COMMAND TIMERS
;        3481
;        3482        END                                                 ! EXECUTIVE PROCESSING LOOP
;        3483
;        3484    UNTIL MD_QUIT ();                                       ! UNTIL ALL UNITS ARE FINISHED
;        3485
;        3486    END;                                                    ! ROUTINE MULTI_DRIVE


                                        .SBTTL  MULTI.DRIVE MULTI-DRIVE SUBTEST ROUTINES
000000  004137  000000G       MULTI.DRIVE:
                                        JSR     R1,$SAVE2                       ;                       3409
000004  104450                          TRAP    50                              ;                       3428
000006  103007                          BHIS    1$
000010  012746  000000G                 MOV     #MSG.05,-(SP)                   ;                       3430
000014  012746  000001                  MOV     #1,-(SP)
000020  010600                          MOV     SP,R0                           ; SP,*
000022  104417                          TRAP    17
000024  022626                          CMP     (SP)+,(SP)+
000026  004737  000000V       1$:     JSR     PC,MD.INIT                      ;                       3431
000032  013702  000000G       2$:     MOV     CTLR.CNT,R2                     ;                       3436
000036  005001                          CLR     R1                              ; CTLR
000040  000461                          BR      9$
000042  010146                  3$:     MOV     R1,-(SP)                        ; CTLR,*                3439
000044  004737  000000G                 JSR     PC,SET.CPAR
000050  004737  000000V                 JSR     PC,QIO.OK                       ;                       3440
000054  020027  000001                  CMP     R0,#1
000060  001042                          BNE     7$
000062  004737  000000V                 JSR     PC,QIO.GEN                      ;                       3444
000066  013700  000602'                 MOV     MX1,R0                          ;                       3445
000072  002415                          BLT     5$
000074  010016                          MOV     R0,(SP)                         ;                       3449
000076  004737  000000G                 JSR     PC,SEND
000102  020027  000001                  CMP     R0,#1
000106  001003                          BNE     4$
000110  105261  000000G                 INCB    QIO(R1)                         ; *(CTLR)               3451
000114  000404                          BR      5$                              ;                       3449
000116  013716  000602'       4$:     MOV     MX1,(SP)                        ;                       3453
000122  004737  000000G                 JSR     PC,PUT.ENV
000126  013700  000604'       5$:     MOV     MX2,R0                          ;                       3457
000132  002415                          BLT     7$
```

```
000134  010016                          MOV     R0,(SP)                 ;                               3461
000136  004737  000000G                 JSR     PC,SEND
000142  020027  000001                  CMP     R0,#1
000146  001003                          BNE     6$
000150  105261  000000G                 INCB    QIO(R1)                 ; *(CTLR)                       3463
000154  000404                          BR      7$                      ;                               3461
000156  013716  000604'          6$:     MOV     MX2,(SP)                ;                               3465
000162  004737  000000G                 JSR     PC,PUT.ENV
000166  005777  000000G          7$:     TST     @DCT.ADDR               ;                               3471
000172  100002                          BPL     8$
000174  004737  000000V                 JSR     PC,INT.PROC             ;                               3473
000200  005726                  8$:     TST     (SP)+                   ;                               3437
000202  005201                          INC     R1                      ; CTLR                          3436
000204  020102                  9$:     CMP     R1,R2                   ; CTLR,*
000206  002715                          BLT     3$
000210  004737  000000V                 JSR     PC,PROC.RETPKT          ;                               3477
000214  123727  000000G 000001          CMPB    T.FLAG,#1               ;                               3478
000222  001002                          BNE     10$
000224  004737  000000V                 JSR     PC,DRV.TIMCHK           ;                               3480
000230  004737  000000V          10$:    JSR     PC,MD.QUIT              ;                               3484
000234  006000                          ROR     R0
000236  103275                          BCC     2$
000240  000207                          RTS     PC                      ;                               3409

; Routine Size:  81 words,      Routine Base:  $CODE$ + 6162
; Maximum stack depth per invocation:  7 words
```

```
;        3487  ROUTINE MD_INIT : NOVALUE =
;        3488
;        3489  !+
;        3490  !       THIS ROUTINE IS CALLED BY ROUTINE MULTI_DRIVE TO INITIALIZE DATA ITEMS
;        3491  !       USED BY THE MULTI-DRIVE SUBTEST.
;        3492  !-
;        3493
;        3494  BEGIN
;        3495
;        3496  INIT_IO_BUFF ();                                    ! PARTITION FREE MEMORY INTO I/O BUFFERS
;        3497  IF .ENTRY_REASON NEQU NEW_PASS                      ! IF START, RESTART, CONT, PWR FAIL
;        3498  THEN                                                ! THEN
;        3499      BEGIN
;        3500
;        3501      INCR UNIT FROM 0 TO (MAX_UNITS - 1) DO         ! FOR EACH UNIT
;        3502          BEGIN
;        3503
;        3504          IF ((.ENTRY_REASON LEQU RESTART) OR                ! IF START OR RESTART OR
;        3505              (.BST [.UNIT, TRACK] LSSU .SWP_STRACK) OR      ! IF USER CHANGED TRACK LIMITS
;        3506              (.BST [.UNIT, TRACK] GTRU .SWP_ETRACK))
;        3507          THEN                                           ! THEN
;        3508              BEGIN
;        3509
;        3510              BST [.UNIT, TRACK] = .SWP_STRACK;          ! INITIALIZE BLOCK SEQUENCE TABLE
;        3511              BST [.UNIT, SECTOR] = 0;                   !     (USED ONLY FOR SEQUENTIAL LBN MODE)
;        3512              DPST [.UNIT] = DP_CNT;                     ! INITIALIZE DATA PATTERN SEQUENCE TABLE
;        3513                                                         !     (USED ONLY IF "PATTERN 0" WAS SELECTED)
;        3514              END;
;        3515
;        3516          END;                                           ! END UNIT LOOP
;        3517
;        3518      END;                                               ! END - IF START, RESTART, CONT, PWR FAIL
;        3519
;        3520  IF ((.ENTRY_REASON NEQU CONT) AND                     ! IF START, RESTART, OR NEW PASS
;        3521      (.ENTRY_REASON NEQU PWR_FAIL))
;        3522  THEN                                                                ! THEN
;        3523      INCR COUNT FROM 0 TO ((MAX_UNITS * TALLY_LEN) - 1) DO          ! CLEAR ALL STATS
;        3524          TALLY [.COUNT] = 0;
;        3525  INCR COUNT FROM 0 TO ((MAX_CTLR * RPS_LEN) - 1) DO ! INITIALIZE RETPKT
;        3526      RP_SAVE [.COUNT] = -1;                         !     SAVE AREA
;        3527  INCR COUNT FROM 0 TO (.NUM_BUFF - 1) DO             ! INITIALIZE I/O BUFFER ALLOCATION
;        3528      BUFF_OWN [.COUNT] = -1;                        !     TABLE
;        3529  IF .MEM_MGMT                                       ! IF SYSTEM HAS MEMORY MANAGEMENT
;        3530  THEN                                               ! THEN
;        3531      MMR0 = 1;                                      ! ENABLE RELOCATION
;        3532  T_FLAG = FALSE;                                    ! FLAG TO CALL DRV_TIMCHK EVERY SECOND
;        3533
;        3534  END;                                               ! ROUTINE MD_INIT


                                .SBTTL  MD.INIT MULTI-DRIVE SUBTEST ROUTINES
000000  004137  000000G  MD.INIT:JSR     R1,$SAVE2                       ;                       3487
000004  004737  000000V          JSR     PC,INIT.IO.BUFF                 ;                       3496
000010  005002                  CLR     R2                              ;                       3497
000012  153702  000000G          BISB    ENTRY.REASON,R2
000016  020227  000005           CMP     R2,#5
000022  001441                  BEQ     4$
```

```
000024  005001                          CLR     R1                      ; UNIT                          3501
000026  020227  000002          1$:     CMP     R2,#2                   ;                               3504
000032  101416                          BLOS    2$
000034  010100                          MOV     R1,R0                   ; UNIT,*                        3505
000036  006300                          ASL     R0
000040  006300                          ASL     R0
000042  026037  000442' 000000G         CMP     BST+2(R0),SWP.STRACK
000050  103407                          BLO     2$
000052  010100                          MOV     R1,R0                   ; UNIT,*                        3506
000054  006300                          ASL     R0
000056  006300                          ASL     R0
000060  026037  000442' 000000G         CMP     BST+2(R0),SWP.ETRACK
000066  101413                          BLOS    3$
000070  010100                  2$:     MOV     R1,R0                   ; UNIT,*                        3510
000072  006300                          ASL     R0
000074  006300                          ASL     R0
000076  013760  000000G 000442'         MOV     SWP.STRACK,BST+2(R0)
000104  005060  000440'                 CLR     BST(R0)                 ;                               3511
000110  112761  000025  000540'         MOVB    #25,DPST(R1)            ; *,*(UNIT)                     3512
000116  005201                  3$:     INC     R1                      ; UNIT                          3501
000120  020127  000017                  CMP     R1,#17                  ; UNIT,*
000124  003740                          BLE     1$
000126  020227  000003          4$:     CMP     R2,#3                   ;                               3520
000132  001413                          BEQ     6$
000134  020227  000004                  CMP     R2,#4                   ;                               3521
000140  001410                          BEQ     6$
000142  005000                          CLR     R0                      ; COUNT                         3523
000144  005060  000000G         5$:     CLR     TALLY(R0)               ; *(COUNT)                      3524
000150  062700  000002                  ADD     #2,R0                   ; *,COUNT                       3523
000154  020027  001376                  CMP     R0,#1376                ; COUNT,*
000160  003771                          BLE     5$
000162  005000                  6$:     CLR     R0                      ; COUNT                         3525
000164  112760  000377  000000G 7$:     MOVB    #377,RP.SAVE(R0)        ; *,*(COUNT)                    3526
000172  005200                          INC     R0                      ; COUNT                         3525
000174  020027  000037                  CMP     R0,#37                  ; COUNT,*
000200  003771                          BLE     7$
000202  005000                          CLR     R0                      ; COUNT                         3527
000204  000404                          BR      9$
000206  112760  000377  000000G 8$:     MOVB    #377,BUFF.OWN(R0)       ; *,*(COUNT)                    3528
000214  005200                          INC     R0                      ; COUNT                         3527
000216  020037  000000G         9$:     CMP     R0,NUM.BUFF             ; COUNT,*
000222  002771                          BLT     8$
000224  132737  000001  000000G         BITB    #1,MEM.MGMT             ;                               3529
000232  001403                          BEQ     10$
000234  012737  000001  177572          MOV     #1,@#177572             ;                               3531
000242  105037  000000G         10$:    CLRB    T.FLAG                  ;                               3532
000246  000207                          RTS     PC                      ;                               3487

; Routine Size:  84 words,      Routine Base:  $CODE$ + 6424
; Maximum stack depth per invocation:  4 words
```

```
;       3535  ROUTINE INIT_IO_BUFF : NOVALUE =
;       3536
;       3537  !+
;       3538  !       THIS ROUTINE IS CALLED BY MD_INIT WHEN THE MULTI-DRIVE SUBTEST IS FIRST
;       3539  !       STARTED. IT IS RESPONSIBLE FOR PARTITIONING FREE MEMORY INTO A
;       3540  !       COLLECTION OF I/O BUFFERS. THE NUMBER OF I/O BUFFERS IS BASED ON THE
;       3541  !       NUMBER OF CONTROLLERS CONFIGURED FOR TESTING.
;       3542  !
;       3543  !       IF THIS HOST PROCESSOR HAS MORE THAN 28K (32K) OF MEMORY, THEN ALL
;       3544  !       AVAILABLE MEMORY OVER 32K (BUT UNDER 124K) IS DIVIDED EQUALLY AMONG
;       3545  !       THE NUMBER OF I/O BUFFERS (4K PER BUFFER MAXIMUM). MEMORY MANAGEMENT
;       3546  !       REGISTERS ARE SET UP TO MAP ALL VIRTUAL ADDRESSES TO THE SAME PHYSICAL
;       3547  !       ADDRESSES (FOR NOW). IF THE HOST PROCESSOR HAS ONLY 28K, THEN THE
;       3548  !       AMOUNT OF FREE MEMORY UNDER 28K IS SIMILARLY DIVIDED (WITHOUT MEMORY
;       3549  !       MANAGEMENT).
;       3550  !
;       3551  !       ONCE THE STARTING ADDRESS OF FREE MEMORY IS DETERMINED ALONG WITH THE
;       3552  !       BUFFER SIZE, THE BUFFER DESCRIPTOR TABLE (BUFF_DESC) IS INITIALIZED.
;       3553  !       THIS TABLE REMAINS FIXED THROUGHOUT THE MULTI-DRIVE SUBTEST. IT
;       3554  !       PROVIDES THE TWO-WORD BUFFER DESCRIPTORS USED IN MSCP COMMAND
;       3555  !       ENVELOPES.
;       3556  !
;       3557  !       IMPLICIT INPUTS:
;       3558  !               CTLR_CNT - THE NUMBER OF CONTROLLERS CONFIGURED
;       3559  !               L$HIMEM - TOP OF FREE MEMORY (IN PAR FORMAT)
;       3560  !               MEM_SIZE - SIZE (IN WORDS) OF FREE MEMORY
;       3561  !               FREE_MEM_ADDR - START OF FREE MEMORY
;       3562  !-
;       3563
;       3564  BEGIN
;       3565
;       3566  LOCAL
;       3567      HIMEM : WORD,                                 ! TOP OF FREE MEMORY (IN PAR FORMAT)
;       3568      BLKS_PER_BUFF : WORD,                         ! NO. OF 32-WORD BLOCKS PER BUFFER
;       3569      BD_LOW : WORD,                                ! BUFFER DESCRIPTOR (LO-ORDER)
;       3570      BD_HIGH : WORD,                               ! BUFFER DESCRIPTOR (HI-ORDER)
;       3571      TEMP : WORD,
;       3572      KTPDR : WORD;                                 ! PDR INITIALIZATION WORD
;       3573
;       3574  NUM_BUFF = .CTLR_CNT * QIO_PER_CTLR;             ! CALCULATE NO. OF I/O BUFFERS NEEDED
;       3575  MEM_MGMT = NO;                                   ! ASSUME NO MEMORY MANAGEMENT
;       3576  IF .L$HIMEM GTRU %O'2000'                        ! IF MORE THAN 32K OF MEMORY
;       3577  THEN                                             ! THEN
;       3578      BEGIN
;       3579
;       3580      MEM_MGMT = YES;                                   ! SET MEMORY MANAGEMENT FLAG
;       3581      HIMEM = .L$HIMEM;                                 ! GET TOP OF FREE MEMORY
;       3582      IF .HIMEM GTRU %O'7577'                           ! IF GREATER THAN 124K
;       3583      THEN                                              ! THEN
;       3584          HIMEM = %O'7577';                             ! SET LIMIT AT 124K
;       3585      BLKS_PER_BUFF = (.HIMEM - %O'1777') / .NUM_BUFF;  ! NO. OF 32-WORD BLOCKS PER BUFFER
;       3586      IF .BLKS_PER_BUFF GTRU %O'200'
;       3587      THEN
;       3588          BLKS_PER_BUFF = %O'200';                      ! MAX BUFFER SIZE = 4K
;       3589      BUFF_SIZE = .BLKS_PER_BUFF * 64;                  ! BUFFER SIZE IN BYTES
;       3590      BD_LOW = 0;                                       ! BUFFERS START AT 32K
;       3591      BD_HIGH = 1;
```

```
;       3592
;       3593        END
;       3594    ELSE                                        ! OTHERWISE - ONLY 28K OF MEMORY
;       3595        BEGIN
;       3596
;       3597        BUFF_SIZE = ((.MEM_SIZE * 2) / .NUM_BUFF) AND %O'177776';
;       3598        BD_LOW = .FREE_MEM_ADDR;                        ! START OF BUFFER SPACE
;       3599        BD_HIGH = 0;
;       3600
;       3601        END;
;       3602
;       3603    INCR J FROM 0 TO (.NUM_BUFF - 1) DO         ! FOR EACH BUFF_DESC ENTRY
;       3604        BEGIN
;       3605
;       3606        BUFF_DESC [.J, BD_LO] = .BD_LOW;        ! LOAD LOW-ORDER WORD
;       3607        BUFF_DESC [.J, BD_HI] = .BD_HIGH;       ! LOAD HIGH-ORDER WORD
;       3608        TEMP = .BD_LOW;
;       3609        BD_LOW = .BD_LOW + .BUFF_SIZE;          ! ADVANCE TO NEXT BUFFER ADDRESS
;       3610        IF .TEMP GEQU .BD_LOW                   ! IF 16-BIT CARRY
;       3611        THEN                                    ! THEN
;       3612            BD_HIGH = .BD_HIGH + 1;             ! ADVANCE HIGH-ORDER WORD
;       3613
;       3614        END;
;       3615
;       3616    IF .MEM_MGMT                                ! IF SYSTEM HAS MEMORY MANAGEMENT
;       3617    THEN                                        ! THEN
;       3618        BEGIN
;       3619
;       3620        MMR0 = 0;                               ! MAKE SURE MEM MGMT IS OFF
;       3621        KTPDR = %O'77406';                      ! PDR LOAD VALUE
;       3622        KTPDR0 = .KTPDR;                        ! LOAD PDR'S
;       3623        KTPDR1 = .KTPDR;
;       3624        KTPDR2 = .KTPDR;
;       3625        KTPDR3 = .KTPDR;
;       3626        KTPDR4 = .KTPDR;
;       3627        KTPDR5 = .KTPDR;
;       3628        KTPDR6 = .KTPDR;
;       3629        KTPDR7 = .KTPDR;
;       3630
;       3631        KTPAR0 = 0;                             ! LOAD PAR'S
;       3632        KTPAR1 = %O'200';
;       3633        KTPAR2 = %O'400';
;       3634        KTPAR3 = %O'600';
;       3635        KTPAR4 = %O'1000';
;       3636        KTPAR5 = %O'1200';
;       3637        KTPAR6 = %O'1400';
;       3638        KTPAR7 = %O'7600';
;       3639
;       3640        END;
;       3641
;       3642    END;                                        ! ROUTINE INIT_IO_BUFF


                                .SBTTL  INIT.IO.BUFF MULTI-DRIVE SUBTEST ROUTINES
000000  004137  000000G     INIT.IO.BUFF:
                            JSR     R1,$SAVE5                       ;                       3535
000004  013700  000000G     MOV     CTLR.CNT,R0                     ;                       3574
```

```
000010  006300                          ASL     R0
000012  006300                          ASL     R0
000014  006300                          ASL     R0
000016  006300                          ASL     R0
000020  010037  000000G                 MOV     R0,NUM.BUFF
000024  105037  000000G                 CLRB    MEM.MGMT                ;                               3575
000030  010005                          MOV     R0,R5                   ; NUM.BUFF,*                    3585
000032  023727  000000G 002000          CMP     L$HIMEM,#2000           ;                               3576
000040  101441                          BLOS    3$
000042  112737  000001  000000G        MOVB    #1,MEM.MGMT             ;                               3580
000050  013700  000000G                 MOV     L$HIMEM,R0              ; *,HIMEM                       3581
000054  020027  007577                  CMP     R0,#7577                ; HIMEM,*                       3582
000060  101402                          BLOS    1$
000062  012700  007577                  MOV     #7577,R0                ; *,HIMEM                       3584
000066  010046                  1$:     MOV     R0,-(SP)                ; HIMEM,*                       3585
000070  162716  001777                  SUB     #1777,(SP)
000074  010546                          MOV     R5,-(SP)
000076  004737  000000G                 JSR     PC,BL$DIV
000102  020027  000200                  CMP     R0,#200                 ; BLKS.PER.BUFF,*               3586
000106  101402                          BLOS    2$
000110  012700  000200                  MOV     #200,R0                 ; *,BLKS.PER.BUFF               3588
000114  000300                  2$:     SWAB    R0                      ;                               3589
000116  106000                          RORB    R0
000120  006000                          ROR     R0
000122  006000                          ROR     R0
000124  142700  000077                  BICB    #77,R0
000130  010037  000000G                 MOV     R0,BUFF.SIZE
000134  005002                          CLR     R2                      ; BD.LOW                        3590
000136  012703  000001                  MOV     #1,R3                   ; *,BD.HIGH                     3591
000142  000416                          BR      4$                      ;                               3576
000144  013746  000000G          3$:    MOV     MEM.SIZE,-(SP)          ;                               3597
000150  006316                          ASL     (SP)
000152  010546                          MOV     R5,-(SP)
000154  004737  000000G                 JSR     PC,BL$DIV
000160  010037  000000G                 MOV     R0,BUFF.SIZE
000164  042737  000001  000000G        BIC     #1,BUFF.SIZE
000172  013702  000000G                 MOV     FREE.MEM.ADDR,R2        ; *,BD.LOW                      3598
000176  005003                          CLR     R3                      ; BD.HIGH                       3599
000200  005001                  4$:     CLR     R1                      ; J                             3603
000202  000416                          BR      7$
000204  010100                  5$:     MOV     R1,R0                   ; J,*                           3606
000206  006300                          ASL     R0
000210  006300                          ASL     R0
000212  010260  000000G                 MOV     R2,BUFF.DESC(R0)        ; BD.LOW,*
000216  010360  000002G                 MOV     R3,BUFF.DESC+2(R0)      ; BD.HIGH,*                     3607
000222  010204                          MOV     R2,R4                   ; BD.LOW,TEMP                   3608
000224  063702  000000G                 ADD     BUFF.SIZE,R2            ; *,BD.LOW                      3609
000230  020402                          CMP     R4,R2                   ; TEMP,BD.LOW                   3610
000232  103401                          BLO     6$
000234  005203                          INC     R3                      ; BD.HIGH                       3612
000236  005201                  6$:     INC     R1                      ; J                             3603
000240  020105                  7$:     CMP     R1,R5                   ; J,*
000242  002760                          BLT     5$
000244  132737  000001  000000G        BITB    #1,MEM.MGMT             ;                               3616
000252  001453                          BEQ     8$
000254  005037  177572                  CLR     @#177572                ;                               3620
000260  012700  077406                  MOV     #77406,R0               ; *,KTPDR                       3621
```

```
000264  010037  172300                          MOV     R0,a#172300             ; KTPDR,*                       3622
000270  010037  172302                          MOV     R0,a#172302             ; KTPDR,*                       3623
000274  010037  172304                          MOV     R0,a#172304             ; KTPDR,*                       3624
000300  010037  172306                          MOV     R0,a#172306             ; KTPDR,*                       3625
000304  010037  172310                          MOV     R0,a#172310             ; KTPDR,*                       3626
000310  010037  172312                          MOV     R0,a#172312             ; KTPDR,*                       3627
000314  010037  172314                          MOV     R0,a#172314             ; KTPDR,*                       3628
000320  010037  172316                          MOV     R0,a#172316             ; KTPDR,*                       3629
000324  005037  172340                          CLR     a#172340                ;                               3631
000330  012737  000200  172342                  MOV     #200,a#172342           ;                               3632
000336  012737  000400  172344                  MOV     #400,a#172344           ;                               3633
000344  012737  000600  172346                  MOV     #600,a#172346           ;                               3634
000352  012737  001000  172350                  MOV     #1000,a#172350          ;                               3635
000360  012737  001200  172352                  MOV     #1200,a#172352          ;                               3636
000366  012737  001400  172354                  MOV     #1400,a#172354          ;                               3637
000374  012737  007600  172356                  MOV     #7600,a#172356          ;                               3638
000402  022626                          8$:     CMP     (SP)+,(SP)+             ;                               3564
000404  000207                                  RTS     PC                      ;                               3535

; Routine Size:  131 words,     Routine Base:  $CODE$ + 6674
; Maximum stack depth per invocation:  9 words
```

```
;         3643  ROUTINE QIO_OK =
;         3644
;         3645  !+
;         3646  !       THIS ROUTINE IS CALLED BY THE MULTI_DRIVE "EXECUTIVE" IN ORDER TO
;         3647  !       DETERMINE WHETHER OR NOT A QIO REQUEST (OR QIO PAIR) SHOULD BE
;         3648  !       GENERATED TO THE CURRENT CONTROLLER. A VALUE OF "TRUE" IS RETURNED IF
;         3649  !       THE CONTROLLER MEETS 3 REQUIREMENTS:
;         3650  !
;         3651  !               A. THE CONTROLLER IS ONLINE;
;         3652  !               B. THE NUMBER OF OUTSTANDING QIOS IS AT LEAST 2 LESS THAN THE
;         3653  !                       MAXIMUM ALLOWED FOR ANY ONE CONTROLLER;
;         3654  !               C. THERE IS AT LEAST ONE UNIT UNDER THE CONTROLLER WHICH IS
;         3655  !                       STILL UNDER TEST.
;         3656  !
;         3657  !       IF ANY OF THESE TESTS FAIL, THEN A VALUE OF "FALSE" IS RETURNED.
;         3658  !
;         3659  !       IMPLICIT INPUTS:
;         3660  !               CCTLR - CURRENT CONTROLLER NUMBER
;         3661  !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;         3662  !-
;         3663
;         3664  BEGIN
;         3665
;         3666  IF .CST_ADDR [STATE] EQLU ONLINE                  ! IF CONTROLLER IS ONLINE
;         3667  THEN
;         3668      BEGIN
;         3669
;         3670      IF (.QIO [.CCTLR] + 2) LEQU QIO_PER_CTLR      ! IF OUTSTANDING QIO COUNT IS O.K.
;         3671      THEN
;         3672          BEGIN
;         3673
;         3674          IF .CST_ADDR [U_CNT] NEQU 0               ! IF THERE IS VALID UNIT
;         3675          THEN
;         3676              RETURN TRUE;                          ! "TRUE" EXIT POINT
;         3677
;         3678          END;
;         3679
;         3680      END;
;         3681
;         3682  RETURN FALSE;                                     ! "FALSE" EXIT POINT
;         3683
;         3684  END;


                                        .SBTTL  QIO.OK MULTI-DRIVE SUBTEST ROUTINES
000000  013700  000000G         QIO.OK: MOV     CST.ADDR,R0                     ;       3666
000004  005760  000002                  TST     2(R0)
000010  100023                          BPL     1$
000012  013700  000000G                 MOV     CCTLR,R0                        ;       3670
000016  116000  000000G                 MOVB    QIO(R0),R0
000022  042700  177400                  BIC     #177400,R0
000026  062700  000002                  ADD     #2,R0
000032  020027  000020                  CMP     R0,#20
000036  101010                          BHI     1$
000040  013700  000000G                 MOV     CST.ADDR,R0                     ;       3674
000044  105760  000005                  TSTB    5(R0)
000050  001403                          BEQ     1$
```

```
000052  012700  000001                        MOV     #1,R0           ;                               3676
000056  000207                                RTS     PC
000060  005000                          1$:   CLR     R0              ;                               3664
000062  000207                                RTS     PC              ;                               3643

; Routine Size:  26 words,      Routine Base:  $CODE$ + 7302
; Maximum stack depth per invocation:  0 words
```

```
;         3685  ROUTINE MD_QUIT =
;         3686
;         3687  !+
;         3688  !       THIS ROUTINE IS CALLED BY THE MULTI_DRIVE EXECUTIVE FOR DETERMINING THE
;         3689  !       END OF THE MULTI-DRIVE SUBTEST. ITS PURPOSE IS TO EXAMINE THE NUMBER OF
;         3690  !       TESTABLE UNITS REMAINING ON EACH CONTROLLER, AND THE NUMBER OF
;         3691  !       OUTSTANDING QIOS. IF BOTH OF THESE PARAMETERS ARE ZERO FOR ALL
;         3692  !       CONTROLLERS, THEN THIS ROUTINE RETURNS A VALUE OF "TRUE". OTHERWISE,
;         3693  !       THE MULTI-DRIVE SUBTEST EXECUTIVE LOOP MUST CONTINUE, AND A VALUE OF
;         3694  !       "FALSE" IS RETURNED.
;         3695  !-
;         3696
;         3697  BEGIN
;         3698
;         3699  INCR CTLR FROM 0 TO (MAX_CTLR - 1) DO
;         3700      BEGIN
;         3701
;         3702      IF .CST [.CTLR, U_CNT] NEQU 0                 ! IF AT LEAST ONE VALID UNIT REMAINS
;         3703      THEN                                          ! THEN
;         3704          RETURN FALSE                              ! DON'T TERMINATE MULTI-DRIVE SUBTEST
;         3705      ELSE                                          ! OTHERWISE (NO UNITS REMAIN)
;         3706          IF .QIO [.CTLR] NEQU 0                    ! IF AT LEAST ONE QIO OUTSTANDING
;         3707          THEN                                      ! THEN
;         3708              RETURN FALSE;                         ! DON'T TERMINATE SUBTEST
;         3709
;         3710      END;
;         3711
;         3712  RETURN TRUE;                                      ! ALL PARAMETERS ARE ZERO - END OF SUBTEST
;         3713
;         3714  END;                                              ! ROUTINE MD_QUIT


                                        .SBTTL  MD.QUIT MULTI-DRIVE SUBTEST ROUTINES
000000  010146                  MD.QUIT:MOV     R1,-(SP)                        ;                       3685
000002  005001                          CLR     R1                              ; CTLR                  3699
000004  010146                  1$:     MOV     R1,-(SP)                        ; CTLR,*                3702
000006  012746  000016                  MOV     #16,-(SP)
000012  004737  000000G                 JSR     PC,BL$MUL
000016  022626                          CMP     (SP)+,(SP)+
000020  105760  000005G                 TSTB    CST+5(R0)
000024  001012                          BNE     2$                              ;                       3704
000026  105761  000000G                 TSTB    QIO(R1)                         ; *(CTLR)               3706
000032  001007                          BNE     2$                              ;                       3708
000034  005201                          INC     R1                              ; CTLR                  3699
000036  020127  000003                  CMP     R1,#3                           ; CTLR,*
000042  003760                          BLE     1$
000044  012700  000001                  MOV     #1,R0                           ;                       3697
000050  000401                          BR      3$
000052  005000                  2$:     CLR     R0                              ;                       3685
000054  012601                  3$:     MOV     (SP)+,R1
000056  000207                          RTS     PC

; Routine Size:  24 words,      Routine Base:  $CODE$ + 7366
; Maximum stack depth per invocation:  4 words
```

```
;       3715  ROUTINE QIO_GEN : NOVALUE =
;       3716
;       3717  !+
;       3718  !       THIS ROUTINE IS CALLED BY THE MULTI_DRIVE EXECUTIVE FOR AN ONLINE
;       3719  !       CONTROLLER ELIGIBLE TO RECEIVE I/O TRANSFER REQUESTS. IT IS
;       3720  !       RESPONSIBLE FOR SECURING ONE OR TWO MSCP ENVELOPES AND LOADING THEM
;       3721  !       WITH VARIOUS PARAMETERS COMPRISING THE I/O REQUEST. THE I/O REQUEST
;       3722  !       GENERATED HERE IS DESTINED TO A PARTICULAR UNIT SELECTED AT RANDOM FROM
;       3723  !       THOSE CONFIGURED UNDER THE CURRENT CONTROLLER.
;       3724  !
;       3725  !       EACH FIELD OF THE ENVELOPE(S) IS LOADED WITHIN INDIVIDUAL ROUTINES
;       3726  !       (QIO_FUNC, QIO_LBN, QIO_SIZE, ETC.). MOST OF THE VALUES SELECTED FOR
;       3727  !       EACH FIELD ARE BASED ON A SET OF RANDOM NUMBER GENERATED AT THE START.
;       3728  !
;       3729  !       UNDER NORMAL CIRCUMSTANCES, ONLY ONE I/O REQUEST IS GENERATED. HOWEVER,
;       3730  !       IF THIS I/O REQUEST IS A "WRITE", AND IF THE OPERATOR SELECTED THE
;       3731  !       OPTION FOR HOST WRITE-COMPARES, THEN A SECOND "READ" REQUEST WILL BE
;       3732  !       GENERATED WITH THE SAME LBN AND BYTE COUNT.
;       3733  !
;       3734  !       AFTER THE ENVELOPE(S) HAVE BEEN LOADED, THIS ROUTINE REGAINS CONTROL
;       3735  !       AND ATTEMPTS TO GET ONE OR TWO I/O BUFFERS FOR THE ACTUAL DATA
;       3736  !       TRANSFERS. THE SUCCESS / FAIL STATUS OF THIS ENTIRE OPERATION IS
;       3737  !       PASSED BACK TO THE CALLER THROUGH THE GLOBALS "MX1" AND "MX2"; THEY
;       3738  !       CONTAIN VALID MSCP ENVELOPE INDECES, OR -1.
;       3739  !
;       3740  !       IMPLICIT INPUTS:
;       3741  !               CCTLR - CURRENT CONTROLLER NUMBER
;       3742  !-
;       3743
;       3744  BEGIN
;       3745
;       3746  MX2 = -1;                                    ! ASSUME FAILURE IN SECURING 2ND ENVELOPE
;       3747  IF (MX1 = GET_ENV (.CCTLR)) LSS 0            ! TRY TO GET 1ST ENVELOPE. IF FAILURE
;       3748  THEN                                         ! THEN
;       3749      RETURN;                                  ! NO POINT IN CONTINUING
;       3750  IF (MX2 = GET_ENV (.CCTLR)) LSS 0            ! TRY TO GET 2ND ENVELOPE. IF FAILURE
;       3751  THEN                                         ! THEN
;       3752      BEGIN
;       3753
;       3754      PUT_ENV (.MX1);                          ! RETURN 1ST ENVELOPE TO POOL
;       3755      MX1 = -1;                                ! INDICATE FAILURE
;       3756      RETURN;                                  ! DONE
;       3757
;       3758      END;
;       3759
;       3760  MAD1 = MSCP_ENV + (.MX1 * ENV_LEN * 2);     ! CALCULATE STARTING ADDRESSES
;       3761  MAD2 = MSCP_ENV + (.MX2 * ENV_LEN * 2);     !     OF BOTH ENVELOPES
;       3762  GET_RANDOM ();                               ! GENERATE A SET OF RANDOM NUMBERS
;       3763  QIO_UNIT ();                                 ! LOAD RANDOM UNIT NUMBER INTO ENVELOPES
;       3764  QIO_FUNC ();                                 ! LOAD RANDOM FUNCTION CODE (OPCODE)
;       3765  QIO_LBN ();                                  ! LOAD LBN (RANDOM OR SEQUENTIAL)
;       3766  QIO_SIZE ();                                 ! LOAD RANDOM BYTE COUNT
;       3767  GET_IO_BUFF (MAD1 [BUF_0]);                 ! TRY TO GET AN I/O BUFFER
;       3768  IF .MX2 GEQ 0                               ! IF TWO QIOS ARE TO BE ISSUED
;       3769  THEN                                        ! THEN
;       3770      BEGIN
;       3771
```

```
;        3772      GET_IO_BUFF (MAD2 [BUF_0]);                 ! TRY TO GET 2ND I/O BUFFER
;        3773      IF .MAD2 [BUF_0]                            ! IF 2ND BUFFER ALLOCATION FAILED
;        3774      THEN                                        ! THEN
;        3775          BEGIN
;        3776
;        3777          IF NOT .MAD1 [BUF_0]                    ! IF 1ST I/O BUFFER WAS ALLOCATED
;        3778          THEN                                    ! THEN
;        3779              BEGIN
;        3780
;        3781              PUT_IO_BUFF (MAD1 [BUF_0]);         ! RETURN 1ST I/O BUFFER TO POOL
;        3782              MAD1 [BUF_0] = -1;                  ! MARK IT AS FAILED
;        3783
;        3784              END;
;        3785
;        3786          PUT_ENV (.MX2);                         ! RETURN 2ND ENVELOPE TO POOL
;        3787          MX2 = -1;                               ! INDICATE FAILURE
;        3788
;        3789          END;                                    ! IF 2ND I/O BUFFER ALLOCATION FAILED
;        3790
;        3791      END;                                        ! IF TWO QIOS ARE TO BE ISSUED
;        3792
;        3793  IF .MAD1 [BUF_0]                                ! IF 1ST I/O BUFFER ALLOCATION FAILED
;        3794  THEN                                            ! THEN
;        3795      BEGIN
;        3796
;        3797      PUT_ENV (.MX1);                             ! RETURN 1ST ENVELOPE TO POOL
;        3798      MX1 = -1;                                   ! INDICATE FAILURE
;        3799
;        3800      END
;        3801  ELSE                                            ! OTHERWISE (ALL IS O.K.)
;        3802      IF .MAD1 [OPCODE] EQLU OP_WRT               ! IF 1ST OPCODE IS A WRITE
;        3803      THEN                                        ! THEN
;        3804          FILL_BUFF ();                           ! FILL 1ST I/O BUFFER WITH APPROPRIATE DATA PATTERN
;        3805
;        3806  END;                                            ! ROUTINE QIO_GEN


                                    .SBTTL  QIO.GEN MULTI-DRIVE SUBTEST ROUTINES
000000  012737  177777  000604'  QIO.GEN:MOV     #-1,MX2                         ;                       3746
000006  013746  000000G                 MOV     CCTLR,-(SP)                     ;                       3747
000012  004737  000000G                 JSR     PC,GET.ENV
000016  010037  000602'                 MOV     R0,MX1
000022  005726                          TST     (SP)+
000024  005700                          TST     R0                              ; MX1
000026  002564                          BLT     6$                              ;                       3749
000030  013746  000000G                 MOV     CCTLR,-(SP)                     ;                       3750
000034  004737  000000G                 JSR     PC,GET.ENV
000040  010037  000604'                 MOV     R0,MX2
000044  005726                          TST     (SP)+
000046  005700                          TST     R0                              ; MX2
000050  002011                          BGE     1$
000052  013746  000602'                 MOV     MX1,-(SP)                       ;                       3754
000056  004737  000000G                 JSR     PC,PUT.ENV
000062  012737  177777  000602'         MOV     #-1,MX1                         ;                       3755
000070  005726                          TST     (SP)+                           ;                       3750
000072  000207                          RTS     PC                              ;                       3752
000074  013746  000602'          1$:     MOV     MX1,-(SP)                       ;                       3760
```

```
000100  012746  000104                        MOV     #104,-(SP)
000104  004737  000000G                       JSR     PC,BL$MUL
000110  062700  000000G                       ADD     #MSCP.ENV,R0
000114  010037  000606'                       MOV     R0,MAD1
000120  013716  000604'                       MOV     MX2,(SP)                ;                       3761
000124  012746  000104                        MOV     #104,-(SP)
000130  004737  000000G                       JSR     PC,BL$MUL
000134  062700  000000G                       ADD     #MSCP.ENV,R0
000140  010037  000610'                       MOV     R0,MAD2
000144  004737  000000V                       JSR     PC,GET.RANDOM           ;                       3762
000150  004737  000000V                       JSR     PC,QIO.UNIT             ;                       3763
000154  004737  000000V                       JSR     PC,QIO.FUNC             ;                       3764
000160  004737  000000V                       JSR     PC,QIO.LBN              ;                       3765
000164  004737  000000V                       JSR     PC,QIO.SIZE             ;                       3766
000170  013716  000606'                       MOV     MAD1,(SP)               ;                       3767
000174  062716  000030                        ADD     #30,(SP)
000200  004737  000000G                       JSR     PC,GET.IO.BUFF
000204  005737  000604'                       TST     MX2                     ;                       3768
000210  002443                               BLT     3$
000212  013716  000610'                       MOV     MAD2,(SP)               ;                       3772
000216  062716  000030                        ADD     #30,(SP)
000222  004737  000000G                       JSR     PC,GET.IO.BUFF
000226  013700  000610'                       MOV     MAD2,R0                 ;                       3773
000232  032760  000001  000030               BIT     #1,30(R0)
000240  001427                               BEQ     3$
000242  013700  000606'                       MOV     MAD1,R0                 ;                       3777
000246  032760  000001  000030               BIT     #1,30(R0)
000254  001012                               BNE     2$
000256  012716  000030                        MOV     #30,(SP)                ;                       3781
000262  060016                               ADD     R0,(SP)
000264  004737  000000G                       JSR     PC,PUT.IO.BUFF
000270  013700  000606'                       MOV     MAD1,R0                 ;                       3782
000274  012760  177777  000030               MOV     #-1,30(R0)
000302  013716  000604'               2$:     MOV     MX2,(SP)                ;                       3786
000306  004737  000000G                       JSR     PC,PUT.ENV
000312  012737  177777  000604'               MOV     #-1,MX2                 ;                       3787
000320  013700  000606'               3$:     MOV     MAD1,R0                 ;                       3793
000324  032760  000001  000030               BIT     #1,30(R0)
000332  001410                               BEQ     4$
000334  013716  000602'                       MOV     MX1,(SP)                ;                       3797
000340  004737  000000G                       JSR     PC,PUT.ENV
000344  012737  177777  000602'               MOV     #-1,MX1                 ;                       3798
000352  000410                               BR      5$                      ;                       3793
000354  013700  000606'               4$:     MOV     MAD1,R0                 ;                       3802
000360  126027  000020  000042               CMPB    20(R0),#42
000366  001002                               BNE     5$
000370  004737  000000V                       JSR     PC,FILL.BUFF            ;                       3804
000374  062706  000006               5$:     ADD     #6,SP                   ;                       3744
000400  000207                       6$:     RTS     PC                      ;                       3715

; Routine Size:  129 words,     Routine Base:  $CODE$ + 7446
; Maximum stack depth per invocation:  4 words
```

```
;         3807  ROUTINE GET_RANDOM : NOVALUE =
;         3808
;         3809  !+
;         3810  !       THIS ROUTINE IS CALLED BY QIO_GEN TO GENERATE A SET OF RANDOM NUMBERS,
;         3811  !       AND TO STORE THEM INTO THE RANDOM NUMBER TABLE (RANDOM). THE RANDOM
;         3812  !       NUMBERS ARE USED TO SELECT I/O REQUEST PARAMETERS FOR THE CURRENT QIO
;         3813  !       OR QIO PAIR. IN ADDITION, IF DATA PATTERN #1 IS BEING USED, THESE
;         3814  !       RANDOM NUMBERS WILL BE USED IN THE WRITE OPERATION.
;         3815  !
;         3816  !       A TEST IS MADE TO AVOID GENERATING THE RANDOM NUMBER 100000-OCTAL. THIS
;         3817  !       NUMBER, IF USED AS THE DIVIDEND IN A "MOD" OPERATION (E.G., QIO_LBN),
;         3818  !       REGARDLESS OF THE DIVISOR, RESULTS IN AN OUT-OF-RANGE REMAINDER.
;         3819  !-
;         3820
;         3821  BEGIN
;         3822
;         3823  OWN
;         3824      SEED1 : WORD INITIAL (%O'123456');
;         3825
;         3826  LOCAL
;         3827      SEED2 : WORD;
;         3828
;         3829  SEED2 = ((.MINUTES ^ 5) OR .SECONDS) ^ 5 OR .TICKS;
;         3830  DECR COUNT FROM (RDM_LEN - 1) TO 0 DO
;         3831      BEGIN
;         3832
;         3833      DO
;         3834          BEGIN
;         3835
;         3836          SEED1 = (.SEED1 + .SEED2 + 1) * 4;
;         3837          SEED2 = (.SEED2 / 4) + .SEED1;
;         3838          RANDOM [.COUNT] = .SEED2;
;         3839
;         3840          END
;         3841      UNTIL .RANDOM [.COUNT] NEQU %O'100000';
;         3842
;         3843      END;
;         3844
;         3845  END;


002552                                  .PSECT  $GGG$,  RO
002552  123456                  SEED1:  .WORD   -54322


                                        .SBTTL  GET.RANDOM MULTI-DRIVE SUBTEST ROUTINES
010050                                  .PSECT  $CODE$,  RO

000000  004137  000000G         GET.RANDOM:
                                        JSR     R1,$SAVE3                       ;       3807
000004  013746  000000G                 MOV     MINUTES,-(SP)                   ;       3829
000010  012746  000005                  MOV     #5,-(SP)
000014  004737  000000G                 JSR     PC,BL$SHF
000020  010016                          MOV     R0,(SP)
000022  053716  000000G                 BIS     SECONDS,(SP)
000026  012746  000005                  MOV     #5,-(SP)
```

```
000032  004737  000000G                        JSR     PC,BL$SHF
000036  010001                                  MOV     R0,R1           ; *,SEED2
000040  053701  000000G                        BIS     TICKS,R1        ; *,SEED2
000044  012702  000036                         MOV     #36,R2          ; *,COUNT                       3830
000050  013703  002552'                 1$:     MOV     SEED1,R3        ;                               3836
000054  010100                          2$:     MOV     R1,R0           ; SEED2,*
000056  060300                                  ADD     R3,R0
000060  006300                                  ASL     R0              ;                               3834
000062  006300                                  ASL     R0
000064  010037  002552'                         MOV     R0,SEED1        ;                               3836
000070  062737  000004  002552'                 ADD     #4,SEED1
000076  010116                                  MOV     R1,(SP)         ; SEED2,*                       3837
000100  012746  000004                         MOV     #4,-(SP)
000104  004737  000000G                        JSR     PC,BL$DIV
000110  013703  002552'                         MOV     SEED1,R3
000114  060300                                  ADD     R3,R0
000116  010001                                  MOV     R0,R1           ; *,SEED2
000120  010162  000634'                         MOV     R1,RANDOM(R2)   ; SEED2,*(COUNT)                3838
000124  005726                                  TST     (SP)+           ;                               3834
000126  020127  100000                         CMP     R1,#-100000     ;                               3841
000132  001750                                  BEQ     2$
000134  162702  000002                         SUB     #2,R2           ; *,COUNT                       3830
000140  100343                                  BPL     1$
000142  062706  000006                         ADD     #6,SP           ;                               3821
000146  000207                                  RTS     PC              ;                               3807

; Routine Size:  52 words,      Routine Base:  $CODE$ + 10050
; Maximum stack depth per invocation:  9 words
```

```
;       3846  ROUTINE QIO_UNIT : NOVALUE =
;       3847
;       3848  !+
;       3849  !       THIS ROUTINE IS CALLED BY QIO_GEN TO RANDOMLY SELECT ONE UNIT
;       3850  !       CONFIGURED UNDER THE CURRENT CONTROLLER (CCTLR) TO BE USED FOR THE
;       3851  !       CURRENT QIO OR QIO PAIR. THE UNIT SELECTED IS BASED ON THE NUMBER OF
;       3852  !       UNITS ELIGIBLE TO RECEIVE AN I/O REQUEST (FROM 1 TO 4) AND THE FIRST
;       3853  !       RANDOM NUMBER IN THE RANDOM NUMBER TABLE (RANDOM).
;       3854  !
;       3855  !       IMPLICIT INPUTS:
;       3856  !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;       3857  !
;       3858  !       IMPLICIT OUTPUTS:
;       3859  !               THE MSCP UNIT NUMBER (PLATTER ADDRESS) IS LOADED INTO THE
;       3860  !                       APPROPRIATE FIELD OF BOTH MSCP ENVELOPES.
;       3861  !-
;       3862
;       3863  BEGIN
;       3864
;       3865  LOCAL
;       3866      UNIT : WORD,                                  ! UNIT NUMBER
;       3867      NTH_UNIT : WORD;                              ! ORDINAL NUMBER OF CHOSEN UNIT
;       3868
;       3869  CASE .CST_ADDR [U_CNT]                           ! DETERMINE ORDINAL UNIT NUMBER
;       3870      FROM 1 TO 4 OF
;       3871      SET
;       3872      [1] : NTH_UNIT = 1;                           ! 1
;       3873      [2] : NTH_UNIT = (.RANDOM [0] AND 1) + 1;     ! 1 OR 2
;       3874      [3] : NTH_UNIT = ABS (.RANDOM [0] MOD 3) + 1; ! 1, 2, OR 3
;       3875      [4] : NTH_UNIT = (.RANDOM [0] AND 3) + 1;     ! 1, 2, 3, OR 4
;       3876      [OUTRANGE] : NTH_UNIT = 1;                    ! IN CASE CONTROLLER JUST WENT DOWN
;       3877      TES;
;       3878
;       3879  INCR OFFSET FROM (0 + OF_UN) TO (3 + OF_UN) DO   ! LOOP THROUGH EACH CST UNIT
;       3880      BEGIN
;       3881
;       3882      IF .CST_ADDR [.OFFSET, P_PRES] EQLU PRESENT           ! IF UNIT IS PRESENT
;       3883      THEN                                                  ! THEN
;       3884          BEGIN
;       3885
;       3886          UNIT = .CST_ADDR [.OFFSET, P_UNIT];               ! GET (DRS) UNIT NUMBER
;       3887          IF .CPT [.UNIT]                                   ! IF THIS UNIT IS ACTIVE
;       3888          THEN                                              ! THEN
;       3889              BEGIN                                         ! THIS IS AN ELIGIBLE UNIT
;       3890
;       3891              NTH_UNIT = .NTH_UNIT - 1;                     ! DECREMENT ORDINAL COUNT
;       3892              IF .NTH_UNIT EQLU 0                           ! IF DOWN TO 0
;       3893              THEN                                          ! THEN
;       3894                  BEGIN                                     ! THIS IS THE CHOSEN UNIT
;       3895
;       3896                  SET_UPAR (.OFFSET);                       ! SET UP UNIT-RELATED DATA
;       3897                  EXITLOOP;                                 ! DONE
;       3898
;       3899                  END;
;       3900
;       3901              END;                                          ! UNIT IS ACTIVE
;       3902
```

```
;        3903          END;                                    ! UNIT IS PRESENT
;        3904
;        3905      END;                                        ! CST UNIT LOOP
;        3906
;        3907  MAD1 [PL_ADDR] = .CPLAT;                       ! LOAD PLATTER ADDRESS (MSCP UNIT NUMBER)
;        3908  MAD2 [PL_ADDR] = .CPLAT;                       !     BOTH MSCP ENVELOPES
;        3909
;        3910  END;                                           ! ROUTINE QIO_UNIT


                                        .SBTTL  QIO.UNIT MULTI-DRIVE SUBTEST ROUTINES
000000  004137  000000G         QIO.UNIT:
                                        JSR     R1,$SAVE3                       ;                       3846
000004  013700  000000G                 MOV     CST.ADDR,R0                     ;                       3869
000010  116000  000005                  MOVB    5(R0),R0
000014  042700  177400                  BIC     #177400,R0
000020  005300                          DEC     R0
000022  020027  000003                  CMP     R0,#3
000026  101003                          BHI     2$
000030  006300                          ASL     R0
000032  066007  000000'                ADD     P.AAA(R0),PC                    ; Case dispatch
000036  012702  000001          2$:     MOV     #1,R2                           ; *,NTH.UNIT            3872
000042  000427                          BR      7$                              ;                       3869
000044  013702  000634'          3$:     MOV     RANDOM,R2                       ; *,NTH.UNIT            3873
000050  042702  177776                  BIC     #177776,R2                      ; *,NTH.UNIT
000054  000421                          BR      6$
000056  013746  000634'          4$:     MOV     RANDOM,-(SP)                    ;                       3874
000062  012746  000003                  MOV     #3,-(SP)
000066  004737  000000G                 JSR     PC,BL$MOD
000072  010016                          MOV     R0,(SP)
000074  004737  000000G                 JSR     PC,BL$ABS
000100  010002                          MOV     R0,R2                           ; *,NTH.UNIT
000102  005202                          INC     R2                              ; NTH.UNIT
000104  022626                          CMP     (SP)+,(SP)+
000106  000405                          BR      7$                              ;                       3869
000110  013702  000634'          5$:     MOV     RANDOM,R2                       ; *,NTH.UNIT            3875
000114  042702  177774                  BIC     #177774,R2                      ; *,NTH.UNIT
000120  005202                  6$:     INC     R2                              ; NTH.UNIT
000122  012703  000003          7$:     MOV     #3,R3                           ; *,OFFSET              3879
000126  010300                  8$:     MOV     R3,R0                           ; OFFSET,*              3882
000130  006300                          ASL     R0
000132  063700  000000G                 ADD     CST.ADDR,R0
000136  032710  040000                  BIT     #40000,(R0)
000142  001417                          BEQ     9$
000144  011001                          MOV     (R0),R1                         ; *,UNIT                3886
000146  000301                          SWAB    R1                              ; UNIT
000150  042701  177740                  BIC     #177740,R1                      ; *,UNIT
000154  132761  000001  000000G         BITB    #1,CPT(R1)                      ; *,*(UNIT)             3887
000162  001407                          BEQ     9$
000164  005302                          DEC     R2                              ; NTH.UNIT              3891
000166  001005                          BNE     9$                              ;                       3892
000170  010346                          MOV     R3,-(SP)                        ; OFFSET,*              3896
000172  004737  000000G                 JSR     PC,SET.UPAR
000176  005726                          TST     (SP)+
000200  000404                          BR      10$                             ;                       3894
000202  005203                  9$:     INC     R3                              ; OFFSET                3879
000204  020327  000006                  CMP     R3,#6                           ; OFFSET,*
```

```
000210  003746                          BLE     8$                                              3907
000212  013700  000606'                10$:    MOV     MAD1,R0                 ;
000216  013760  000000G 000014                 MOV     CPLAT,14(R0)                            3908
000224  013700  000610'                        MOV     MAD2,R0                 ;
000230  013760  000000G 000014                 MOV     CPLAT,14(R0)                            3846
000236  000207                          RTS     PC                              ;

; Routine Size:  80 words,      Routine Base:  $CODE$ + 10220
; Maximum stack depth per invocation:  7 words


000000                                  .PSECT  $PLIT$,  RO ,  D

                                P.AAA:                                  ; CASE Table for QIO.UNIT+0032  3869
000000  000000                 1$:     .WORD   0                       ; [2$]
000002  000006                         .WORD   6                       ; [3$]
000004  000020                         .WORD   20                      ; [4$]
000006  000052                         .WORD   52                      ; [5$]
```

```
;       3911  ROUTINE QIO_FUNC : NOVALUE =
;       3912
;       3913  !+
;       3914  !       THIS ROUTINE IS CALLED BY QIO_GEN TO SELECT THE I/O FUNCTION (OPCODE)
;       3915  !       TO BE USED FOR THE CURRENT QIO OR QIO PAIR. THE FUNCTION IS DETERMINED
;       3916  !       BY THE FOLLWING ALGORITHM:
;       3917  !
;       3918  !               IF THE CHOSEN UNIT IS PROTECTED
;       3919  !               THEN
;       3920  !                       FUNCTION = READ
;       3921  !               ELSE (UNPROTECTED)
;       3922  !                       IF OPERATOR SELECTED "WRITE-ONLY" OPTION
;       3923  !                       THEN
;       3924  !                               FUNCTION = WRITE
;       3925  !                       ELSE
;       3926  !                               FUNCTION (WRITE OR READ) IS BASED ON A RANDOM
;       3927  !                               NUMBER
;       3928  !
;       3929  !       IN ADDITION, IF THE OPERATOR SELECTED THE OPTION OF PERFORMING WRITE-
;       3930  !       COMPARES AT THE HOST, AND IF A "WRITE" FUNCTION WAS CHOSEN ABOVE FOR
;       3931  !       THE FIRST QIO, THEN A "READ" OPCODE IS LOADED INTO THE SECOND MSCP
;       3932  !       ENVELOPE. OTHERWISE, THE SECOND MSCP ENVELOPE IS RETURNED TO THE POOL.
;       3933  !
;       3934  !       IMPLICIT INPUTS:
;       3935  !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;       3936  !               CUOFF - CURRENT UNIT CST OFFSET
;       3937  !
;       3938  !       IMPLICIT OUTPUTS:
;       3939  !               THE OPCODE FIELD OF ONE OR BOTH MSCP ENVELOPES IS LOADED.
;       3940  !-
;       3941
;       3942  BEGIN
;       3943
;       3944  LOCAL
;       3945      FUNC : WORD;                                  ! OPCODE (READ OR WRITE)
;       3946
;       3947  IF .CST_ADDR [.CUOFF, P_PROT] EQLU PROTECTED     ! IF UNIT IS PROTECTED
;       3948  THEN                                              ! THEN
;       3949      FUNC = OP_RD                                  ! SET FUNCTION TO READ
;       3950  ELSE                                              ! OTHERWISE (UNIT IS UNPROTECTED)
;       3951      BEGIN
;       3952
;       3953      IF BIT_TST (SWP_FLAGS, SWF_WO)                ! IF OPERATOR CHOSE WRITE-ONLY OPTION
;       3954      THEN                                          ! THEN
;       3955          FUNC = OP_WRT                             ! SET FUNCTION TO WRITE
;       3956      ELSE                                          ! OTHERWISE
;       3957          BEGIN
;       3958
;       3959          IF (.RANDOM [1] AND 1)                    ! USE 2ND RANDOM NUMBER TO SELECT
;       3960          THEN                                      !     EITHER
;       3961              FUNC = OP_RD                          !     READ
;       3962          ELSE                                      !     OR
;       3963              FUNC = OP_WRT;                        !     WRITE
;       3964
;       3965          END;
;       3966
;       3967      END;
```

```
;         3968
;         3969  IF (MAD1 [OPCODE] = .FUNC) EQLU OP_WRT          ! LOAD CHOSEN OPCODE. IF WRITE
;         3970  THEN                                            ! THEN
;         3971      BEGIN
;         3972
;         3973      IF BIT_TST (SWP_FLAGS, SWF_CWC)             ! IF CONTROLLER DOES WRITE-COMPARES
;         3974      THEN                                        ! THEN
;         3975          MAD1 [MODIFY] = MD_CMP;                 ! ADD COMPARE MODIFIER
;         3976
;         3977      IF BIT_TST (SWP_FLAGS, SWF_HWC)             ! IF HOST DOES WRITE-COMPARES
;         3978      THEN                                        ! THEN
;         3979          BEGIN
;         3980
;         3981          MAD1 [MODIFY] = MD_EXP;                 ! SET EXPRESS REQUEST MODIFIER
;         3982          MAD2 [OPCODE] = OP_RD;                  ! SET READ OPCODE INTO 2ND MSCP ENVELOPE
;         3983          MAD2 [MODIFY] = MD_EXP;                 ! SET EXPRESS REQUEST MODIFIER
;         3984
;         3985          END;
;         3986
;         3987      END
;         3988  ELSE                                            ! OTHERWISE - FUNCTION IS READ
;         3989      BEGIN
;         3990
;         3991      IF BIT_TST (SWP_FLAGS, SWF_CRC)             ! IF CONTROLLER DOES READ-COMPARES
;         3992      THEN                                        ! THEN
;         3993          MAD1 [MODIFY] = MD_CMP;                 ! ADD COMPARE MODIFIER
;         3994
;         3995      END;
;         3996
;         3997  IF .MAD2 [OPCODE] EQLU 0                       ! IF NO OPCODE IN 2ND ENVELOPE
;         3998  THEN                                            ! THEN
;         3999      BEGIN
;         4000
;         4001      PUT_ENV (.MX2);                           ! RETURN 2ND ENVELOPE TO POOL
;         4002      MX2 = -1;                                 ! MARK IT UNUSED
;         4003
;         4004      END;
;         4005
;         4006  END;                                            ! ROUTINE QIO_FUNC


                                        .SBTTL  QIO.FUNC MULTI-DRIVE SUBTEST ROUTINES
010460                                  .PSECT  $CODE$,  RO

000000  004137  000000G         QIO.FUNC:
                                        JSR     R1,$SAVE2                       ;                       3911
000004  013700  000000G                 MOV     CUOFF,R0                        ;                       3947
000010  006300                          ASL     R0
000012  063700  000000G                 ADD     CST.ADDR,R0
000016  032710  100000                  BIT     #100000,(R0)
000022  001410                          BEQ     1$                              ;                       3949
000024  132737  000020  000000G         BITB    #20,SWP.FLAGS                   ;                       3953
000032  001007                          BNE     2$                              ;                       3955
000034  032737  000001  000636'         BIT     #1,RANDOM+2                     ;                       3959
000042  001403                          BEQ     2$
000044  012702  000041          1$:     MOV     #41,R2                          ; *,FUNC                3961
000050  000402                          BR      3$                              ;                       3959
```

```
000052  012702  000042                2$:     MOV     #42,R2                  ; *,FUNC                                3963
000056  005001                        3$:     CLR     R1                      ;                                       3973
000060  153701  000000G                       BISB    SWP.FLAGS,R1
000064  013700  000606'                       MOV     MAD1,R0                 ;                                       3969
000070  110260  000020                        MOVB    R2,20(R0)               ; FUNC,*
000074  020227  000042                        CMP     R2,#42                  ; FUNC,*
000100  001025                              BNE     5$
000102  032701  000040                        BIT     #40,R1                  ;                                       3973
000106  001403                              BEQ     4$
000110  012760  040000  000022                MOV     #40000,22(R0)           ;                                       3975
000116  032701  000100                4$:     BIT     #100,R1                 ;                                       3977
000122  001422                              BEQ     6$
000124  012760  100000  000022                MOV     #-100000,22(R0)         ;                                       3981
000132  013700  000610'                       MOV     MAD2,R0                 ;                                       3982
000136  112760  000041  000020                MOVB    #41,20(R0)
000144  012760  100000  000022                MOV     #-100000,22(R0)         ;                                       3983
000152  000406                              BR      6$                      ;                                       3969
000154  032701  000010                5$:     BIT     #10,R1                  ;                                       3991
000160  001403                              BEQ     6$
000162  012760  040000  000022                MOV     #40000,22(R0)           ;                                       3993
000170  013700  000610'                6$:     MOV     MAD2,R0                 ;                                       3997
000174  105760  000020                        TSTB    20(R0)
000200  001010                              BNE     7$
000202  013746  000604'                       MOV     MX2,-(SP)               ;                                       4001
000206  004737  000000G                       JSR     PC,PUT.ENV
000212  012737  177777  000604'               MOV     #-1,MX2                 ;                                       4002
000220  005726                              TST     (SP)+                   ;                                       3999
000222  000207                        7$:     RTS     PC                      ;                                       3911

; Routine Size:  74 words,      Routine Base:  $CODE$ + 10460
; Maximum stack depth per invocation:  5 words
```

```
;         4007  ROUTINE QIO_LBN : NOVALUE =
;         4008
;         4009  !+
;         4010  !       THIS ROUTINE IS CALLED BY QIO_GEN TO SELECT THE LOGICAL BLOCK NUMBER TO
;         4011  !       BE USED FOR THE CURRENT QIO OR QIO PAIR.
;         4012  !
;         4013  !       IF THE OPERATOR CHOSE THE RANDOM SEEK MODE OPTION, THEN THE LBN IS
;         4014  !       RANDOMLY CHOSEN WITHIN THE SPECIFIED LIMITS FOR TRACK AND SECTOR.
;         4015  !       OTHERWISE, THE NEXT SEQUENTIAL LBN IS DERIVED FROM THE BLOCK SEQUENCE
;         4016  !       TABLE (BST).
;         4017  !
;         4018  !       IMPLICIT INPUTS:
;         4019  !               L$LUN - CURRENT (DIAGNOSTIC SUPERVIOR) UNIT NUMBER
;         4020  !
;         4021  !       IMPLICIT OUTPUTS:
;         4022  !               THE LBN IS LOADED INTO ONE OR BOTH MSCP ENVELOPES.
;         4023  !-
;         4024
;         4025  BEGIN
;         4026
;         4027  LOCAL
;         4028      T_RANGE : WORD,                           ! PERMISSIBLE RANGE OF TRACK NUMBERS
;         4029      S_RANGE : WORD,                           ! PERMISSIBLE RANGE OF SECTOR NUMBERS
;         4030      NTRACK : WORD,                            ! SELECTED TRACK NUMBER
;         4031      NSECT : WORD,                             ! SELECTED SECTOR NUMBER
;         4032      LBN : WORD;                               ! LOGICAL BLOCK NUMBER
;         4033
;         4034  IF BIT_TST (SWP_FLAGS, SWF_RDM)               ! IF RANDOM SEEK MODE
;         4035  THEN                                          ! THEN
;         4036      BEGIN
;         4037
;         4038      T_RANGE = (.SWP_ETRACK - .SWP_STRACK) + 1;          ! RANGE OF TRACK NUMBERS
;         4039      S_RANGE = MAX_SECT + 1;                             ! RANGE OF SECTOR NUMBERS
;         4040      NTRACK = .SWP_STRACK + ABS (.RANDOM [2] MOD .T_RANGE); ! RANDOM TRACK NUMBER WITHIN RANGE
;         4041      NSECT = ABS (.RANDOM [3] MOD .S_RANGE);             ! RANDOM SECTOR NUMBER WITHIN RANGE
;         4042
;         4043      END
;         4044  ELSE                                          ! ELSE - SEQUENTIAL LBN MODE
;         4045      BEGIN
;         4046
;         4047      NTRACK = .BST [.L$LUN, TRACK];            ! GET TRACK FROM BST
;         4048      NSECT = .BST [.L$LUN, SECTOR];            ! GET SECTOR FROM BST
;         4049      ADV_BST ();                               ! ADVANCE TO NEXT LBN
;         4050
;         4051      END;
;         4052
;         4053  LBN = (.NTRACK * SEC_PER_TRK) + .NSECT;       ! CALCULATE LBN
;         4054  MAD1 [LBN_L] = .LBN;                          ! LOAD LBN INTO 1ST ENVELOPE
;         4055  IF .MX2 GEQ 0                                 ! IF 2 QIOS
;         4056  THEN                                          ! THEN
;         4057      MAD2 [LBN_L] = .LBN;                      ! LOAD LBN INTO 2ND ENVELOPE
;         4058
;         4059  END;                                          ! ROUTINE QIO_LBN

                                        .SBTTL  QIO.LBN MULTI-DRIVE SUBTEST ROUTINES
000000  004137  000000G         QIO.LBN:JSR     R1,$SAVE2                               ;                                       4007
```

```
000004  132737  000004  000000G                 BITB    #4,SWP.FLAGS                    ;                               4034
000012  001436                                  BEQ     1$
000014  013700  000000G                         MOV     SWP.ETRACK,R0                   ;                               4038
000020  163700  000000G                         SUB     SWP.STRACK,R0
000024  005200                                  INC     R0
000026  012701  000037                          MOV     #37,R1                          ; *,S.RANGE                     4039
000032  013746  000640'                         MOV     RANDOM+4,-(SP)                  ;                               4040
000036  010046                                  MOV     R0,-(SP)                        ; T.RANGE,*
000040  004737  000000G                         JSR     PC,BL$MOD
000044  010016                                  MOV     R0,(SP)
000046  004737  000000G                         JSR     PC,BL$ABS
000052  063700  000000G                         ADD     SWP.STRACK,R0
000056  010002                                  MOV     R0,R2                           ; *,NTRACK
000060  013716  000642'                         MOV     RANDOM+6,(SP)                   ;                               4041
000064  010146                                  MOV     R1,-(SP)                        ; S.RANGE,*
000066  004737  000000G                         JSR     PC,BL$MOD
000072  010016                                  MOV     R0,(SP)
000074  004737  000000G                         JSR     PC,BL$ABS
000100  010001                                  MOV     R0,R1                           ; *,NSECT
000102  062706  000006                          ADD     #6,SP                           ;                               4036
000106  000412                                  BR      2$                              ;                               4034
000110  013700  000000G         1$:             MOV     L$LUN,R0                        ;                               4047
000114  006300                                  ASL     R0
000116  006300                                  ASL     R0
000120  016002  000442'                         MOV     BST+2(R0),R2                    ; *,NTRACK
000124  016001  000440'                         MOV     BST(R0),R1                      ; *,NSECT                       4048
000130  004737  000000V                         JSR     PC,ADV.BST                      ;                               4049
000134  010246                          2$:     MOV     R2,-(SP)                        ; NTRACK,*                      4053
000136  012746  000037                          MOV     #37,-(SP)
000142  004737  000000G                         JSR     PC,BL$MUL
000146  060100                                  ADD     R1,R0                           ; NSECT,*
000150  010001                                  MOV     R0,R1                           ; *,LBN
000152  013700  000606'                         MOV     MAD1,R0                         ;                               4054
000156  010160  000044                          MOV     R1,44(R0)                       ; LBN,*
000162  005737  000604'                         TST     MX2                             ;                               4055
000166  002404                                  BLT     3$
000170  013700  000610'                         MOV     MAD2,R0                         ;                               4057
000174  010160  000044                          MOV     R1,44(R0)                       ; LBN,*
000200  022626                          3$:     CMP     (SP)+,(SP)+                     ;                               4025
000202  000207                                  RTS     PC                              ;                               4007

; Routine Size:  66 words,      Routine Base:  $CODE$ + 10704
; Maximum stack depth per invocation:  7 words
```

```
;        4060  ROUTINE ADV_BST : NOVALUE =
;        4061
;        4062  !+
;        4063  !       THIS ROUTINE IS CALLED BY QIO_LBN TO ADVANCE THE CURRNET UNIT'S LBN
;        4064  !       IN THE BLOCK SEQUENCE TABLE (BST). THIS IS DONE BY INCREMENTING THE
;        4065  !       SECTOR NUMBER (AND TRACK NUMBER, IF NECESSARY), WHILE ENSURING THAT
;        4066  !       BOTH VALUES REMAIN WITHIN THE PROPER LIMITS.
;        4067  !
;        4068  !       IMPLICIT INPUTS:
;        4069  !               L$LUN - CURRENT (DRS) UNIT NUMBER
;        4070  !-
;        4071
;        4072  BEGIN
;        4073
;        4074  BST [.L$LUN, SECTOR] = .BST [.L$LUN, SECTOR] + 1;        ! INCREMENT SECTOR NUMBER
;        4075  IF .BST [.L$LUN, SECTOR] GTRU MAX_SECT                    ! IF SECTOR IS BEYOND HIGH LIMIT
;        4076  THEN                                                      ! THEN
;        4077      BEGIN
;        4078
;        4079      BST [.L$LUN, SECTOR] = 0;                            ! SET SECTOR TO LOW LIMIT
;        4080      BST [.L$LUN, TRACK] = .BST [.L$LUN, TRACK] + 1;      ! INCREMENT TRACK NUMBER
;        4081      IF .BST [.L$LUN, TRACK] GTRU .SWP_ETRACK             ! IF TRACK IS BEYOND HIGH LIMIT
;        4082      THEN                                                 ! THEN
;        4083          BST [.L$LUN, TRACK] = .SWP_STRACK;               ! SET TRACK TO LOW LIMIT
;        4084
;        4085      END;
;        4086
;        4087  END;                                                     ! ROUTINE ADV_BST


                                .SBTTL  ADV.BST MULTI-DRIVE SUBTEST ROUTINES
000000  010146                  ADV.BST:MOV     R1,-(SP)                        ;                       4060
000002  013700  000000G                 MOV     L$LUN,R0                        ;                       4074
000006  006300                          ASL     R0
000010  006300                          ASL     R0
000012  012701  000440'                 MOV     #BST,R1
000016  060001                          ADD     R0,R1
000020  005211                          INC     (R1)
000022  021127  000036                  CMP     (R1),#36                        ;                       4075
000026  101411                          BLOS    1$
000030  005011                          CLR     (R1)                            ;                       4079
000032  062700  000442'                 ADD     #BST+2,R0                       ;                       4080
000036  005210                          INC     (R0)
000040  021037  000000G                 CMP     (R0),SWP.ETRACK                 ;                       4081
000044  101402                          BLOS    1$
000046  013710  000000G                 MOV     SWP.STRACK,(R0)                 ;                       4083
000052  012601                  1$:     MOV     (SP)+,R1                        ;                       4060
000054  000207                          RTS     PC

; Routine Size:  23 words,      Routine Base:  $CODE$ + 11110
; Maximum stack depth per invocation:  2 words
```

```
;        4088  ROUTINE QIO_SIZE : NOVALUE =
;        4089
;        4090  !+
;        4091  !       THIS ROUTINE IS CALLED BY QIO_GEN TO SELECT THE I/O TRANSFER BYTE COUNT
;        4092  !       TO BE USED FOR THE CURRENT QIO OR QIO PAIR. THE BYTE COUNT IS
;        4093  !       DETERMINED BY A RANDOM NUMBER, AND WILL ALWAYS BE AN EVEN NUMBER
;        4094  !       BETWEEN 0 AND THE I/O BUFFER SIZE (BUFF_SIZE), INCLUSIVE.
;        4095  !
;        4096  !       IMPLICIT OUTPUTS:
;        4097  !               THE BYTE COUNT IS LOADED INTO ONE OR BOTH MSCP ENVELOPES.
;        4098  !-
;        4099
;        4100  BEGIN
;        4101
;        4102  LOCAL
;        4103      END_BLK : WORD,                                 ! ENDING BLOCK NUMBER FOR CURRENT I/O
;        4104      SIZE : WORD;                                    ! BYTE COUNT
;        4105
;        4106  SIZE = ABS (.RANDOM [4] MOD (.BUFF_SIZE + 1)) AND %O'177776';    ! GET BYTE COUNT FROM RANDOM NUMBER
;        4107  END_BLK = .MAD1 [LBN_L] + ((.SIZE - 1) / BLK_SIZE);             ! CALCULATE ENDING BLOCK NUMBER
;        4108  IF .END_BLK GTRU (.USIZE - 1)                           ! IF ENDING BLOCK IS LARGER THAN UNIT SIZE
;        4109  THEN                                                    ! THEN
;        4110      SIZE = (.USIZE - .MAD1 [LBN_L]) * BLK_SIZE;         ! SCALE DOWN BYTE COUNT
;        4111  MAD1 [BC_LO] = .SIZE;                                   ! LOAD SIZE INTO 1ST MSCP ENVELOPE
;        4112  IF .MX2 GEQ 0                                           ! IF 2 QIOS
;        4113  THEN                                                    ! THEN
;        4114      MAD2 [BC_LO] = .SIZE;                               ! LOAD SIZE INTO 2ND MSCP ENVELOPE
;        4115
;        4116  END;                                                    ! ROUTINE QIO_SIZE


                        .SBTTL  QIO.SIZE MULTI-DRIVE SUBTEST ROUTINES
000000  004137  000000G QIO.SIZE:
                        JSR     R1,$SAVE3                       ;                               4088
000004  013746  000644' MOV     RANDOM+10,-(SP)                 ;                               4106
000010  013746  000000G MOV     BUFF.SIZE,-(SP)
000014  005216          INC     (SP)
000016  004737  000000G JSR     PC,BL$MOD
000022  010016          MOV     R0,(SP)
000024  004737  000000G JSR     PC,BL$ABS
000030  010003          MOV     R0,R3                           ; *,SIZE
000032  042703  000001  BIC     #1,R3                           ; *,SIZE
000036  013701  000606' MOV     MAD1,R1                         ;                               4107
000042  010316          MOV     R3,(SP)                         ; SIZE,*
000044  005316          DEC     (SP)
000046  012746  001000  MOV     #1000,-(SP)
000052  004737  000000G JSR     PC,BL$DIV
000056  066100  000044  ADD     44(R1),R0
000062  010002          MOV     R0,R2                           ; *,END.BLK
000064  013700  000616' MOV     USIZE,R0                        ;                               4108
000070  005300          DEC     R0
000072  020200          CMP     R2,R0                           : END.BLK,*
000074  101410          BLOS    1$
000076  013700  000616' MOV     USIZE,R0                        ;                               4110
000102  166100  000044  SUB     44(R1),R0
000106  000300          SWAB    R0
000110  105000          CLRB    R0
```

```
000112  006300                          ASL     R0
000114  010003                          MOV     R0,R3                   ; *,SIZE                        4111
000116  013700  000606'         1$:     MOV     MAD1,R0                 ;                               4112
000122  010360  000024                  MOV     R3,24(R0)               ; SIZE,*                        4114
000126  005737  000604'                 TST     MX2                     ;                               4100
000132  002404                          BLT     2$                      ;                               4088
000134  013700  000610'                 MOV     MAD2,R0
000140  010360  000024                  MOV     R3,24(R0)               ; SIZE,*
000144  062706  000006         2$:     ADD     #6,SP                   ;
000150  000207                          RTS     PC                      ;

; Routine Size:  53 words,      Routine Base:  $CODE$ + 11166
; Maximum stack depth per invocation:  8 words
```

```
;        4117  ROUTINE FILL_BUFF : NOVALUE =
;        4118
;        4119  !+
;        4120  !       THIS ROUTINE IS CALLED BY QIO_GEN TO LOAD THE I/O BUFFER DESCRIBED IN
;        4121  !       THE FIRST MSCP ENVELOPE WITH THE APPROPRIATE DATA PATTERN.
;        4122  !
;        4123  !       THE DATA PATTERN TO BE SELECTED IS BASED ON THE FOLLOWING ALGORITHM:
;        4124  !
;        4125  !               IF THE OPERATOR DEFINED A DATA PATTERN
;        4126  !               THEN
;        4127  !                       SELECT IT
;        4128  !               ELSE
;        4129  !                       GET DATA PATTERN NUMBER FROM SW P-TABLE
;        4130  !                       IF DATA PATTERN NUMBER = 0
;        4131  !                       THEN
;        4132  !                               GET DATA PATTERN NUMBER FROM THE UNIT'S ENTRY
;        4133  !                               IN THE DATA PATTERN SEQUENCE TABLE (DPST)
;        4134  !
;        4135  !       NOTE THAT PATTERN # 1 CONSISTS OF RANDOM NUMBERS, AND PATTERNS # 17 -
;        4136  !       21 USE THE ACTUAL LBN OF THE WRITE REQUEST.
;        4137  !
;        4138  !       IMPLICIT INPUTS:
;        4139  !               L$LUN - CURRENT (DRS) UNIT NUMBER
;        4140  !-
;        4141
;        4142  BEGIN
;        4143
;        4144  LOCAL
;        4145      DP_NUM : WORD,                          ! DATA PATTERN NUMBER SELECTED
;        4146      DP_ADDR,                                ! ADDR OF DATA PATTERN (LENGTH)
;        4147      IOB_ADDR,                               ! I/O BUFFER ADDRESS (DESTINATION)
;        4148      SRC_ADDR,                               ! WORKING SOURCE ADDRESS
;        4149      DISP : WORD,                            ! MEM. MGMT. ADDRESS DISPLACEMENT
;        4150      PAF : WORD,                             ! PAGE ADDRESS FIELD
;        4151      COUNT : WORD;                           ! NO. OF WORDS IN DATA PATTERN
;        4152
;        4153  IF BIT_TST (SWP_FLAGS, SWF_UDP)             ! IF USER DEFINED A DATA PATTERN
;        4154  THEN                                        ! THEN
;        4155      DP_ADDR = SWP_UCNT                      ! SELECT IT
;        4156  ELSE                                        ! OTHERWISE
;        4157      BEGIN
;        4158
;        4159      IF .SWP_DPAT NEQU 0                     ! IF USER SELECTED A PRE-DEFINED DATA PATTERN
;        4160      THEN                                    ! THEN
;        4161          DP_NUM = .SWP_DPAT                  ! SELECT IT
;        4162      ELSE                                    ! OTHERWISE
;        4163          BEGIN
;        4164
;        4165          DP_NUM = .DPST [.L$LUN];            ! GET PATTERN NUMBER FROM SEQUENCE TABLE
;        4166          DPST [.L$LUN] = .DPST [.L$LUN] + 1; ! ADVANCE TO NEXT PATTERN NUMBER
;        4167          IF .DPST [.L$LUN] GTRU DP_CNT       ! CHECK FOR HIGH LIMIT
;        4168          THEN
;        4169              DPST [.L$LUN] = 1;
;        4170
;        4171          FND;
;        4172
;        4173      DP_ADDR = .DPA_TBL [.DP_NUM - 1];       ! ADDRESS OF DATA PATTERN (COUNT)
```

```
;        4174      IF .DP_NUM GEQU 17
;        4175      THEN
;        4176          BEGIN
;        4177
;        4178          IF .DP_NUM                              ! IF PATTERN 17, 19, OR 21
;        4179          THEN                                    ! THEN
;        4180              (.DP_ADDR + 2) = .MAD1 [LBN_L]      ! LOAD LBN INTO FIRST WORD OF PATTERN
;        4181          ELSE                                    ! ELSE - PATTERN 18 OR 20
;        4182              (.DP_ADDR + 4) = .MAD1 [LBN_L];     ! LOAD LBN INTO SECOND WORD OF PATTERN
;        4183
;        4184          END;
;        4185
;        4186      END;
;        4187
;        4188  IOB_ADDR = .MAD1 [BUF_0];                       ! I/O BUFFER ADDRESS
;        4189  IF .MEM_MGMT                                    ! IF SYSTEM HAS MEMORY MANAGEMENT
;        4190  THEN                                            ! THEN
;        4191      BEGIN
;        4192
;        4193      DISP = .IOB_ADDR AND %O'17777';             ! I/O BUFFER DISPLACEMENT ADDRESS
;        4194      PAF = (.IOB_ADDR ^ -6) AND %O'1600';        ! PAGE ADDRESS FIELD
;        4195      KTPAR4 = (.MAD1 [BUF_1] ^ 10) OR .PAF;      ! LOAD PAR4
;        4196      IOB_ADDR = %O'100000' OR .DISP;             ! I/O BUFFER VIRTUAL ADDRESS
;        4197
;        4198      END;
;        4199
;        4200  COUNT = ..DP_ADDR;                             ! NO. OF WORDS IN DATA PATTERN
;        4201  SRC_ADDR = .DP_ADDR + 2;                       ! START OF THE ACTUAL DATA PATTERN
;        4202  INCR N FROM 1 TO (.MAD1 [BC_LO] / 2) DO        ! FOR EACH WORD IN THIS WRITE REQUEST
;        4203      BEGIN
;        4204
;        4205      .IOB_ADDR = ..SRC_ADDR;                     ! MOVE 1 WORD
;        4206      IOB_ADDR = .IOB_ADDR + 2;                   ! ADVANCE DESTINATION ADDRESS
;        4207      SRC_ADDR = .SRC_ADDR + 2;                   ! ADVANCE SOURCE ADDRESS
;        4208      IF .MEM_MGMT                                ! IF SYSTEM HAS MEMORY MANAGEMENT
;        4209      THEN                                        ! THEN
;        4210          BEGIN
;        4211
;        4212          IF (.IOB_ADDR AND %O'17777') EQLU 0     ! IF I/O BUFFER CROSSES A 4K PAGE BOUNDARY
;        4213          THEN                                    ! THEN
;        4214              BEGIN
;        4215
;        4216              IOB_ADDR = %O'100000';              ! RESET VIRTUAL ADDRESS
;        4217              KTPAR4 = .KTPAR4 + %O'200';         ! ADVANCE PAR TO NEXT 4K PAGE
;        4218
;        4219              END;
;        4220
;        4221          END;                                    ! END - IF MEMORY MANAGEMENT
;        4222
;        4223      COUNT = .COUNT - 1;                         ! DECREMENT COUNT
;        4224      IF .COUNT EQLU 0                            ! IF END OF DATA PATTERN
;        4225      THEN                                        ! THEN
;        4226          BEGIN
;        4227
;        4228          COUNT = ..DP_ADDR;                      ! REPEAT DATA PATTERN
;        4229          SRC_ADDR = .DP_ADDR + 2;
;        4230
```

```
;        4231          END;
;        4232
;        4233      END;                                  ! WORD TRANSFER LOOP
;        4234
;        4235  IF .MEM_MGMT                              ! IF SYSTEM HAS MEMORY MANAGEMENT
;        4236  THEN                                      ! THEN
;        4237      KTPAR4 = %O'1000';                    ! RESTORE PAR4
;        4238
;        4239  END;                                      ! ROUTINE FILL_BUFF


                                        .SBTTL  FILL.BUFF MULTI-DRIVE SUBTEST ROUTINES
000000  004137  000000G         FILL.BUFF:
                                        JSR     R1,$SAVE5                       ;                       4117
000004  024646                          CMP     -(SP),-(SP)
000006  105737  000000G                 TSTB    SWP.FLAGS                       ;                       4153
000012  100003                          BPL     1$
000014  012704  000000G                 MOV     #SWP.UCNT,R4                    ; *,DP.ADDR             4155
000020  000445                          BR      5$                              ;                       4153
000022  005000                  1$:     CLR     R0                              ;                       4159
000024  153700  000000G                 BISB    SWP.DPAT,R0
000030  005700                          TST     R0
000032  001402                          BEQ     2$
000034  010001                          MOV     R0,R1                           ; *,DP.NUM              4161
000036  000414                          BR      3$                              ;                       4159
000040  013700  000000G         2$:     MOV     L$LUN,R0                        ;                       4165
000044  062700  000540'                 ADD     #DPST,R0
000050  005001                          CLR     R1                              ; DP.NUM
000052  151001                          BISB    (R0),R1                         ; *,DP.NUM
000054  105210                          INCB    (R0)                            ;                       4166
000056  121027  000025                  CMPB    (R0),#25                        ;                       4167
000062  101402                          BLOS    3$
000064  112710  000001                  MOVB    #1,(R0)                         ;                       4169
000070  010100                  3$:     MOV     R1,R0                           ; DP.NUM,*              4173
000072  006300                          ASL     R0
000074  016004  001642'                 MOV     DPA.TBL-2(R0),R4                ; *,DP.ADDR
000100  020127  000021                  CMP     R1,#21                          ; DP.NUM,*              4174
000104  103413                          BLO     5$
000106  013700  000606'                 MOV     MAD1,R0                         ;                       4180
000112  006001                          ROR     R1                              ; DP.NUM                4178
000114  103004                          BCC     4$
000116  016064  000044  000002          MOV     44(R0),2(R4)                    ; *,*(DP.ADDR)          4180
000124  000403                          BR      5$                              ;                       4178
000126  016064  000044  000004  4$:     MOV     44(R0),4(R4)                    ; *,*(DP.ADDR)          4182
000134  013701  000606'          5$:     MOV     MAD1,R1                         ;                       4188
000140  016105  000030                  MOV     30(R1),R5                       ; *,IOB.ADDR
000144  005066  000002                  CLR     2(SP)                           ;                       4189
000150  113766  000000G 000002          MOVB    MEM.MGMT,2(SP)
000156  032766  000001  000002          BIT     #1,2(SP)
000164  001431                          BEQ     6$
000166  010503                          MOV     R5,R3                           ; IOB.ADDR,DISP         4193
000170  042703  160000                  BIC     #160000,R3                      ; *,DISP
000174  010546                          MOV     R5,-(SP)                        ; IOB.ADDR,*            4194
000176  012746  177772                  MOV     #-6,-(SP)
000202  004737  000000G                 JSR     PC,BL$SHF
000206  010002                          MOV     R0,R2                           ; *,PAF
000210  042702  176177                  BIC     #176177,R2                      ; *,PAF
```

```
000214  016100  000032                        MOV     32(R1),R0               ;                                       4195
000220  000300                                SWAB    R0
000222  105000                                CLRB    R0
000224  006300                                ASL     R0
000226  006300                                ASL     R0
000230  010246                                MOV     R2,-(SP)                ; PAF,*
000232  050016                                BIS     R0,(SP)
000234  012637  172350                        MOV     (SP)+,@#172350
000240  010305                                MOV     R3,R5                   ; DISP,IOB.ADDR                         4196
000242  052705  100000                        BIS     #100000,R5              ; *,IOB.ADDR
000246  022626                                CMP     (SP)+,(SP)+             ;                                       4191
000250  011403                        6$:     MOV     (R4),R3                 ; DP.ADDR,COUNT                         4200
000252  012702  000002                        MOV     #2,R2                   ;                                       4201
000256  060402                                ADD     R4,R2                   ; DP.ADDR,*
000260  010216                                MOV     R2,(SP)                 ; *,SRC.ADDR
000262  016146  000024                        MOV     24(R1),-(SP)            ;                                       4202
000266  012746  000002                        MOV     #2,-(SP)
000272  004737  000000G                       JSR     PC,BL$DIV
000276  005001                                CLR     R1                      ; N
000300  000426                                BR      9$
000302  017625  000004                7$:     MOV     @4(SP),(R5)+            ; SRC.ADDR,IOB.ADDR                     4205
000306  062766  000002  000004                ADD     #2,4(SP)                ; *,SRC.ADDR                            4207
000314  032766  000001  000006                BIT     #1,6(SP)                ;                                       4208
000322  001410                                BEQ     8$
000324  032705  017777                        BIT     #17777,R5               ; *,IOB.ADDR                            4212
000330  001005                                BNE     8$
000332  012705  100000                        MOV     #-100000,R5             ; *,IOB.ADDR                            4216
000336  062737  000200  172350                ADD     #200,@#172350           ;                                       4217
000344  005303                        8$:     DEC     R3                      ; COUNT                                 4223
000346  001003                                BNE     9$                      ;                                       4224
000350  011403                                MOV     (R4),R3                 ; DP.ADDR,COUNT                         4228
000352  010266  000004                        MOV     R2,4(SP)                ; *,SRC.ADDR                            4229
000356  005201                        9$:     INC     R1                      ; N                                     4202
000360  020100                                CMP     R1,R0                   ; N,*
000362  003747                                BLE     7$
000364  032766  000001  000006                BIT     #1,6(SP)                ;                                       4235
000372  001403                                BEQ     10$
000374  012737  001000  172350                MOV     #1000,@#172350          ;                                       4237
000402  062706  000010                10$:    ADD     #10,SP                  ;                                       4117
000406  000207                                RTS     PC

; Routine Size:  132 words,     Routine Base:  $CODE$ + 11340
; Maximum stack depth per invocation:  11 words
```

```
;         4240  ROUTINE PROC_RETPKT : NOVALUE =
;         4241
;         4242  !+
;         4243  !       THIS ROUTINE IS CALLED FROM THE MULTI_DRIVE "EXECUTIVE" TO CHECK FOR
;         4244  !       AND PROCESS ANY RETURN PACKETS THAT HAVE BEEN "SENT" BY THE "DRIVER"
;         4245  !       PORTION OF THE PROGRAM. THE I/O DONE QUEUE (IODQ) ACTS AS THE LINK
;         4246  !       BETWEEN THE TWO PROGRAM PARTS; IT HOLDS INDECES OF RETURN PACKETS WHICH
;         4247  !       REQUIRE PROCESSING.
;         4248  !
;         4249  !       UNDER THE MULTI-DRIVE SUBTEST, RETURN PACKETS ORIGINATE FROM TWO
;         4250  !       SOURCES:
;         4251  !               1. DISK MSCP - THE MORE COMMON, DESCRIBING A COMPLETED I/O
;         4252  !                  OPERATION
;         4253  !               2. THE PROGRAM "DRIVER" - DESCRIBING A CONTROLLER ERROR OR
;         4254  !                  COMMAND TIMEOUT.
;         4255  !-
;         4256
;         4257  BEGIN
;         4258
;         4259  WHILE .IODQ_IN NEQU .IODQ_OUT DO                ! DO UNTIL I/O DONE QUEUE IS EMPTY
;         4260      BEGIN
;         4261
;         4262      RP_INDX = OUT_IODQ ();                      ! GET INDEX OF NEXT RETPKT AND ADVANCE OUT POINTER
;         4263      RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2); ! CALCULATE RETPKT ADDRESS
;         4264      SET_CPAR (.RP_ADDR [CTLR]);                 ! SET UP CURRENT CONTROLLER PARAMETERS
;         4265      SELECTONEU .RP_ADDR [CONID] OF              ! CONNECTION ID INDICATES PACKET SOURCE
;         4266          SET
;         4267          [CID_DISK]   : IO_RETPKT ();                    ! DISK MSCP (I/O TRANSFER DONE)
;         4268          [CID_DRIVER] : DR_RETPKT ();                    ! MESSAGE FROM "DRIVER"
;         4269          TES;
;         4270
;         4271      END;                                        ! UNITL I/O DONE QUEUE IS EMPTY
;         4272
;         4273  END;                                            ! ROUTINE PROC_RETPKT


                                        .SBTTL  PROC.RETPKT MULTI-DRIVE SUBTEST ROUTINES
000000  023737  000000G 000000G  PROC.RETPKT:
000000                          1$:     CMP     IODQ.IN,IODQ.OUT                ;                       4259
000006  001444                          BEQ     4$
000010  004737  000000G                 JSR     PC,OUT.IODQ                     ;                       4262
000014  010037  000000G                 MOV     R0,RP.INDX
000020  010046                          MOV     R0,-(SP)                        ; RP.INDX,*             4263
000022  012746  000060                  MOV     #60,-(SP)
000026  004737  000000G                 JSR     PC,BL$MUL
000032  062700  000000G                 ADD     #RETPKT,R0
000036  010037  000000G                 MOV     R0,RP.ADDR
000042  116016  000002                  MOVB    2(R0),(SP)                      ;                       4264
000046  042716  177760                  BIC     #177760,(SP)
000052  004737  000000G                 JSR     PC,SET.CPAR
000056  013700  000000G                 MOV     RP.ADDR,R0                      ;                       4265
000062  116000  000003                  MOVB    3(R0),R0
000066  042700  177400                  BIC     #177400,R0
000072  001003                          BNE     2$
000074  004737  000000V                 JSR     PC,IO.RETPKT                    ;                       4267
000100  000405                          BR      3$                              ;                       4265
000102  020027  000003          2$:     CMP     R0,#3
```

```
000106  001002                          BNE     3$
000110  004737  000000V                 JSR     PC,DR.RETPKT                    ;                       4268
000114  022626                  3$:     CMP     (SP)+,(SP)+                     ;                       4260
000116  000730                          BR      1$                              ;                       4259
000120  000207                  4$:     RTS     PC                              ;                       4240

; Routine Size:  41 words,      Routine Base:  $CODE$ + 11750
; Maximum stack depth per invocation:  3 words
```

```
;        4274  ROUTINE IO_RETPKT : NOVALUE =
;        4275
;        4276  !+
;        4277  !       THIS ROUTINE IS CALLED BY PROC_RETPKT TO HANDLE ALL I/O TRANSFER
;        4278  !       RETURN PACKETS. PROCESSING OF THESE PACKETS INCLUDES DECLARING ANY
;        4279  !       HARD ERRORS THAT MAY HAVE OCCURRED, UPDATING THE STATISTICS, AND
;        4280  !       PERFORMING HOST WRITE-COMPARES IF REQUIRED.
;        4281  !
;        4282  !       IMPLICIT INPUTS:
;        4283  !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;        4284  !               CCTLR - CURRENT CONTROLLER NUMBER
;        4285  !-
;        4286
;        4287  BEGIN
;        4288
;        4289  SWEEP_FLAG = TRUE;                                  ! INDICATES THAT SWEEP () WILL BE CALLED
;        4290  FSET_UPAR ();                                       ! FIND UNIT'S ENTRY IN CST AND SET UP UNIT-RELATED DATA
;        4291  ST_CODE = .RP_ADDR [STSCOD];                       ! GET STATUS CODE FROM RETPKT
;        4292  SB_CODE = .RP_ADDR [SUBCOD];                       ! GET SUB-CODE, IF ANY
;        4293  IF .ST_CODE NEQU ST_SUC                            ! IF STATUS CODE INDICATES ERROR
;        4294  THEN                                               ! THEN
;        4295      BEGIN
;        4296
;        4297      IF .ST_CODE EQLU ST_OFL                        ! IF UNIT OFFLINE
;        4298      THEN                                           ! THEN
;        4299          BEGIN
;        4300
;        4301          ERRDF (19, EGD_19, EMS_19);                ! "FATAL I/O ERROR"
;        4302          DUR [.L$LUN] = DU_FATAL;                   ! LOAD REASON FOR DROPPING UNIT
;        4303          DODU (.L$LUN);                             ! DROP UNIT
;        4304
;        4305          END
;        4306      ELSE                                           ! ELSE - ERROR IS NOT UNIT-OFFLINE
;        4307          BEGIN
;        4308
;        4309          CASE .ST_CODE
;        4310              FROM 1 TO 11 OF
;        4311              SET
;        4312              [1]         : ERRHRD (31, EGH_30, EMS_30);    ! INVALID COMMAND
;        4313              [2]         : ERRHRD (32, EGH_30, EMS_30);    ! COMMAND ABORTED
;        4314              [3]         : ;                               ! UNIT OFFLINE (DEVICE-FATAL)
;        4315              [4]         : ERRHRD (34, EGH_30, EMS_30);    ! UNIT AVAILABLE
;        4316              [5]         : ERRHRD (35, EGH_30, EMS_30);    ! MEDIA FORMAT ERROR
;        4317              [6]         : ERRHRD (36, EGH_30, EMS_30);    ! WRITE-PROTECTED
;        4318              [7]         : ERRHRD (37, EGH_30, EMS_30);    ! DEVICE COMPARE ERROR
;        4319              [8]         : ERRHRD (38, EGH_30, EMS_30);    ! DATA ERROR
;        4320              [9]         : ERRHRD (39, EGH_30, EMS_30);    ! HOST BUFFER ACCESS ERROR
;        4321              [10]        : ERRHRD (40, EGH_30, EMS_30);    ! CONTROLLER ERROR
;        4322              [11]        : ERRHRD (41, EGH_30, EMS_30);    ! DRIVE ERROR
;        4323              [OUTRANGE] : ERRHRD (30, EGH_30, EMS_30);     ! UNKNOWN
;        4324              TES;
;        4325
;        4326          HARD_ERR (1);                                 ! INCREMENT HARD ERROR COUNT FOR CURRENT UNIT
;        4327
;        4328          END;
;        4329
;        4330      END
```

```
;          4331  ELSE                                    ! ELSE - I/O WAS SUCCESSFUL
;          4332      BEGIN
;          4333
;          4334      MD_TALLY ();                        ! UPDATE I/O STATISTICS
;          4335      IF BIT_TST (SWP_FLAGS, SWF_HWC)     ! IF HOST IS DOING WRITE-COMPARES
;          4336      THEN                                ! THEN
;          4337          HOST_WRT_CHK ();                ! SAVE I/O PACKET OR DO WRITE-CHECK
;          4338
;          4339      END;
;          4340
;          4341  IF .SWEEP_FLAG EQLU TRUE               ! IF SWEEP_FLAG IS STILL TRUE
;          4342  THEN                                    ! THEN
;          4343      SWEEP ();                           ! DEALLOCATE BUFFER(S) AND RETPKT(S)
;          4344  QIO [.CCTLR] = .QIO [.CCTLR] - 1;       ! DECREMENT NO. OF OUTSTANDING QIOS
;          4345
;          4346  END;                                    ! ROUTINE IO_RETPKT


                                        .SBTTL  IO.RETPKT MULTI-DRIVE SUBTEST ROUTINES
000000  010146                  IO.RETPKT:
                                        MOV     R1,-(SP)                        ;                       4274
000002  112737  000001  000630'         MOVB    #1,SWEEP.FLAG                   ;                       4289
000010  004737  000000V                 JSR     PC,FSET.UPAR                    ;                       4290
000014  013700  000000G                 MOV     RP.ADDR,R0                      ;                       4291
000020  116037  000016  000000G         MOVB    16(R0),ST.CODE
000026  042737  177740  000000G         BIC     #177740,ST.CODE
000034  016001  000016                  MOV     16(R0),R1                       ;                       4292
000040  006201                          ASR     R1
000042  006201                          ASR     R1
000044  006201                          ASR     R1
000046  006201                          ASR     R1
000050  006201                          ASR     R1
000052  042701  174000                  BIC     #174000,R1
000056  010137  000000G                 MOV     R1,SB.CODE
000062  013700  000000G                 MOV     ST.CODE,R0                      ;                       4293
000066  001521                          BEQ     15$
000070  020027  000003                  CMP     R0,#3                           ;                       4297
000074  001013                          BNE     1$
000076  104455                          TRAP    55                              ;                       4301
000100  000023                          .WORD   23
000102  000000G                         .WORD   EGD.19
000104  000000G                         .WORD   EMS.19
000106  013700  000000G                 MOV     L$LUN,R0                        ;                       4302
000112  112760  000004  000000G         MOVB    #4,DUR(R0)
000120  104451                          TRAP    51                              ;                       4303
000122  000513                          BR      16$                             ;                       4297
000124  005300                  1$:     DEC     R0                              ;                       4309
000126  020027  000012                  CMP     R0,#12
000132  101003                          BHI     3$
000134  006300                          ASL     R0
000136  066007  000010'                 ADD     P.AAB(R0),PC                    ; Case dispatch
000142  104456                  3$:     TRAP    56                              ;                       4323
000144  000036                          .WORD   36
000146  000000G                         .WORD   EGH.30
000150  000000G                         .WORD   EMS.30
000152  000461                          BR      14$                             ;                       4309
000154  104456                  4$:     TRAP    56                              ;                       4312
```

```
000156  000037                          .WORD   37
000160  000000G                         .WORD   EGH.30
000162  000000G                         .WORD   EMS.30
000164  000454                          BR      14$                             ;                       4309
000166  104456                  5$:     TRAP    56                              ;                       4313
000170  000040                          .WORD   40
000172  000000G                         .WORD   EGH.30
000174  000000G                         .WORD   EMS.30
000176  000447                          BR      14$                             ;                       4309
000200  104456                  6$:     TRAP    56                              ;                       4315
000202  000042                          .WORD   42
000204  000000G                         .WORD   EGH.30
000206  000000G                         .WORD   EMS.30
000210  000442                          BR      14$                             ;                       4309
000212  104456                  7$:     TRAP    56                              ;                       4316
000214  000043                          .WORD   43
000216  000000G                         .WORD   EGH.30
000220  000000G                         .WORD   EMS.30
000222  000435                          BR      14$                             ;                       4309
000224  104456                  8$:     TRAP    56                              ;                       4317
000226  000044                          .WORD   44
000230  000000G                         .WORD   EGH.30
000232  000000G                         .WORD   EMS.30
000234  000430                          BR      14$                             ;                       4309
000236  104456                  9$:     TRAP    56                              ;                       4318
000240  000045                          .WORD   45
000242  000000G                         .WORD   EGH.30
000244  000000G                         .WORD   EMS.30
000246  000423                          BR      14$                             ;                       4309
000250  104456                  10$:    TRAP    56                              ;                       4319
000252  000046                          .WORD   46
000254  000000G                         .WORD   EGH.30
000256  000000G                         .WORD   EMS.30
000260  000416                          BR      14$                             ;                       4309
000262  104456                  11$:    TRAP    56                              ;                       4320
000264  000047                          .WORD   47
000266  000000G                         .WORD   EGH.30
000270  000000G                         .WORD   EMS.30
000272  000411                          BR      14$                             ;                       4309
000274  104456                  12$:    TRAP    56                              ;                       4321
000276  000050                          .WORD   50
000300  000000G                         .WORD   EGH.30
000302  000000G                         .WORD   EMS.30
000304  000404                          BR      14$                             ;                       4309
000306  104456                  13$:    TRAP    56                              ;                       4322
000310  000051                          .WORD   51
000312  000000G                         .WORD   EGH.30
000314  000000G                         .WORD   EMS.30
000316  012746  000001          14$:    MOV     #1,-(SP)                        ;                       4326
000322  004737  000000G                 JSR     PC,HARD.ERR
000326  005726                          TST     (SP)+                           ;                       4307
000330  000410                          BR      16$                             ;                       4293
000332  004737  000000V         15$:    JSR     PC,MD.TALLY                     ;                       4334
000336  132737  000100  000000G         BITB    #100,SWP.FLAGS                  ;                       4335
000344  001402                          BEQ     16$
000346  004737  000000V                 JSR     PC,HOST.WRT.CHK                 ;                       4337
000352  123727  000630'  000001  16$:    CMPB    SWEEP.FLAG,#1                   ;                       4341
```

```
000360  001002                          BNE     17$
000362  004737  000000V                 JSR     PC,SWEEP                ;                       4343
000366  013700  000000G         17$:    MOV     CCTLR,R0                ;                       4344
000372  105360  000000G                 DECB    QIO(R0)
000376  012601                          MOV     (SP)+,R1                ;                       4274
000400  000207                          RTS     PC

; Routine Size:  129 words,     Routine Base:  $CODE$ + 12072
; Maximum stack depth per invocation:  3 words


000010                                  .PSECT  $PLIT$,  RO ,  D

                                P.AAB:                                  ; CASE Table for IO.RETPKT+0136 4309
000010  000012                  2$:     .WORD   12                      ; [4$]
000012  000024                          .WORD   24                      ; [5$]
000014  000154                          .WORD   154                     ; [14$]
000016  000036                          .WORD   36                      ; [6$]
000020  000050                          .WORD   50                      ; [7$]
000022  000062                          .WORD   62                      ; [8$]
000024  000074                          .WORD   74                      ; [9$]
000026  000106                          .WORD   106                     ; [10$]
000030  000120                          .WORD   120                     ; [11$]
000032  000132                          .WORD   132                     ; [12$]
000034  000144                          .WORD   144                     ; [13$]
```

```
;        4347  ROUTINE FSET_UPAR : NOVALUE =
;        4348
;        4349  !+
;        4350  !       THIS ROUTINE IS CALLED BY IO_RETPKT AND OTHERS TO SEARCH THE CURRENT
;        4351  !       CONTROLLER STATUS TABLE (CST) FOR THE PLATTER ADDRESS WHICH IS
;        4352  !       CONTAINED IN THE CURRENT RETURN PACKET. WHEN FOUND, THE OFFSET INTO THE
;        4353  !       CST IS USED AS INPUT TO SET_UPAR, WHICH SETS UP CURRENT UNIT-RELATED
;        4354  !       DATA PARAMETERS.
;        4355  !
;        4356  !       IMPLICIT INPUTS:
;        4357  !               RP_ADDR - ADDRESS OF CURRENT RETURN PACKET
;        4358  !               CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
;        4359  !-
;        4360
;        4361  BEGIN
;        4362
;        4363  INCR OFFSET FROM (0 + OF_UN) TO (3 + OF_UN) DO  ! FOR EACH UNIT IN CST
;        4364      BEGIN
;        4365
;        4366      IF .CST_ADDR [.OFFSET, P_ADDR] EQLU .RP_ADDR [PLAT] ! IF RETPKT UNIT NUMBER MATCHES CST ENTRY
;        4367      THEN                                                ! THEN
;        4368          BEGIN
;        4369
;        4370          SET_UPAR (.OFFSET);                             ! SET UP UNIT-RELATED DATA
;        4371          RETURN;                                         ! DONE
;        4372
;        4373          END;
;        4374
;        4375      END;                                                ! CST UNIT SEARCH LOOP
;        4376
;        4377  END;                                                    ! ROUTINE FSET_UPAR

                                        .SBTTL  FSET.UPAR MULTI-DRIVE SUBTEST ROUTINES
012474                                  .PSECT  $CODE$,  RO

000000  004137  000000G         FSET.UPAR:
                                        JSR     R1,$SAVE3               ;                       4347
000004  012702  000003                  MOV     #3,R2                   ; *,OFFSET              4363
000010  010201                  1$:     MOV     R2,R1                   ; OFFSET,*              4366
000012  006301                          ASL     R1
000014  063701  000000G                 ADD     CST.ADDR,R1
000020  013700  000000G                 MOV     RP.ADDR,R0
000024  005003                          CLR     R3
000026  151103                          BISB    (R1),R3
000030  020360  000010                  CMP     R3,10(R0)
000034  001005                          BNE     2$
000036  010246                          MOV     R2,-(SP)                ; OFFSET,*              4370
000040  004737  000000G                 JSR     PC,SET.UPAR
000044  005726                          TST     (SP)+                   ;                       4366
000046  000207                          RTS     PC                      ;                       4368
000050  005202                  2$:     INC     R2                      ; OFFSET                4363
000052  020227  000006                  CMP     R2,#6                   ; OFFSET,*
000056  003754                          BLE     1$
000060  000207                          RTS     PC                      ;                       4347

; Routine Size:  25 words,      Routine Base:  $CODE$ + 12474
; Maximum stack depth per invocation:  6 words
```

```
;        4378  ROUTINE MD_TALLY : NOVALUE =
;        4379
;        4380  !+
;        4381  !       THIS ROUTINE IS CALLED FROM IO_RETPKT FOR ALL I/O TRANSFER RETURN
;        4382  !       PACKETS WITH "SUCCESS" STATUS CODES. ITS PURPOSE IS TO UPDATE ALL THE
;        4383  !       APPROPRIATE STATISTICAL FIELDS FOR THE CURRENT UNIT. A CHECK IS ALSO
;        4384  !       MADE ON THE TOTAL NUMBER OF BYTES TRANSFERRED THUS FAR; IF THE
;        4385  !       OPERATOR-SPECIFIED LIMIT HAS BEEN REACHED, THEN THE UNIT IS REMOVED
;        4386  !       FROM CURRENT PASS TESTING.
;        4387  !
;        4388  !       IMPLICIT INPUTS:
;        4389  !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;        4390  !               T_ADDR - ADDRESS OF THE CURRENT UNIT'S STATISTICS BLOCK (TALLY)
;        4391  !               L$LUN - CURRENT (DRS) UNIT NUMBER
;        4392  !-
;        4393
;        4394  BEGIN
;        4395
;        4396  IF .RP_ADDR [ENDCOD] EQLU (OP_RD + OP_END)       ! IF ENDCODE IS READ
;        4397  THEN                                              ! THEN
;        4398      BEGIN
;        4399
;        4400      UPD_IOC (T_ADDR [READ_LO], 1);                        ! INCREMENT NO. OF READS
;        4401      T_ADDR [BR_LO] = .T_ADDR [BR_LO] + .RP_ADDR [BCNT_LO]; ! UPDATE BYTE COUNT
;        4402      OVF_CHK (T_ADDR [BR_LO]);                             ! CHECK FOR FIELD OVERFLOW
;        4403
;        4404      END
;        4405  ELSE                                              ! ELSE ENDCODE IS WRITE
;        4406      BEGIN
;        4407
;        4408      UPD_IOC (T_ADDR [WRIT_LO], 1);                        ! INCREMENT NO. OF WRITES
;        4409      T_ADDR [BW_LO] = .T_ADDR [BW_LO] + .RP_ADDR [BCNT_LO]; ! UPDATE BYTE COUNT
;        4410      OVF_CHK (T_ADDR [BW_LO]);                             ! CHECK FOR FIELD OVERFLOW
;        4411
;        4412      END;
;        4413
;        4414  IF .CPT [.L$LUN]                                  ! IF UNIT IS STILL UNDER TEST
;        4415  THEN                                              ! THEN
;        4416      XFR_CHK ();                                   ! CHECK MBYTES XFR'D AGAINST LIMIT
;        4417
;        4418  END;                                              ! ROUTINE MD_TALLY


                                        .SBTTL  MD.TALLY MULTI-DRIVE SUBTEST ROUTINES
000000  010146                  MD.TALLY:
                                        MOV     R1,-(SP)                        ;                       4378
000002  013700  000000G                 MOV     RP.ADDR,R0                      ;                       4396
000006  126027  000014  000241          CMPB    14(R0),#241
000014  001020                          BNE     1$
000016  013746  000000G                 MOV     T.ADDR,-(SP)                    ;                       4400
000022  012746  000001                  MOV     #1,-(SP)
000026  004737  000000G                 JSR     PC,UPD.IOC
000032  013701  000000G                 MOV     T.ADDR,R1                       ;                       4401
000036  013700  000000G                 MOV     RP.ADDR,R0
000042  066061  000020  000014          ADD     20(R0),14(R1)
000050  012716  000014                  MOV     #14,(SP)                        ;                       4402
000054  000421                          BR      2$
```

```
000056  013746  000000G                   1$:     MOV     T.ADDR,-(SP)                    ;                       4408
000062  062716  000004                               ADD     #4,(SP)
000066  012746  000001                               MOV     #1,-(SP)
000072  004737  000000G                               JSR     PC,UPD.IOC
000076  013701  000000G                               MOV     T.ADDR,R1                       ;                       4409
000102  013700  000000G                               MOV     RP.ADDR,R0
000106  066061  000020  000022                       ADD     20(R0),22(R1)
000114  012716  000022                               MOV     #22,(SP)                        ;                       4410
000120  060116                             2$:     ADD     R1,(SP)
000122  004737  000000G                               JSR     PC,OVF.CHK
000126  013700  000000G                               MOV     L$LUN,R0                        ;                       4414
000132  132760  000001  000000G                       BITB    #1,CPT(R0)
000140  001402                                       BEQ     3$
000142  004737  000000G                               JSR     PC,XFR.CHK                      ;                       4416
000146  022626                             3$:     CMP     (SP)+,(SP)+                     ;                       4394
000150  012601                                       MOV     (SP)+,R1                        ;                       4378
000152  000207                                       RTS     PC

; Routine Size:  54 words,      Routine Base:  $CODE$ + 12556
; Maximum stack depth per invocation:  4 words
```

```
;       4419  ROUTINE HOST_WRT_CHK : NOVALUE =
;       4420
;       4421  !+
;       4422  !       THIS ROUTINE IS CALLED FROM IO_RETPKT FOR ALL I/O TRANSFER RETURN
;       4423  !       PACKETS WITH "SUCCESS" STATUS CODES, BUT ONLY IF THE HOST WRITE-COMPARE
;       4424  !       OPTION WAS SELECTED BY THE OPERATOR.
;       4425  !
;       4426  !       IF THE CURRENT RETPKT BEING PROCESSED IS A WRITE FUNCTION, THEN THE
;       4427  !       PACKET INDEX (RP_INDX) IS SAVED IN THE CONTROLLER'S RETURN PACKET SAVE
;       4428  !       AREA (RP_SAVE). OTHERWISE, THE PACKET IS A READ, SO ITS ASSOCIATED
;       4429  !       WRITE PACKET IS REMOVED FROM THE SAVE AREA, AND A WORD-FOR-WORD
;       4430  !       COMPARISON IS PERFORMED ON THE TWO I/O BUFFERS. ANY DIFFERENCES
;       4431  !       ENCOUNTERED RESULTS IN THE DECLARATION OF A HARD ERROR.
;       4432  !
;       4433  !       IMPLICIT INPUTS:
;       4434  !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;       4435  !               RP_INDX - INDEX OF THE CURRENT RETURN PACKET
;       4436  !               RPS_X1, RPS_X2 - STARTING / ENDING INDECES OF THE CURRENT
;       4437  !                                       CONTROLLER'S RETPKT SAVE AREA (RP_SAVE)
;       4438  !-
;       4439
;       4440  BEGIN
;       4441
;       4442  LOCAL
;       4443      BUFFW,                                          ! ADDR OF I/O BUFFER DESCRIPTOR
;       4444      INDEX : SIGNED WORD;
;       4445
;       4446  IF .RP_ADDR [ENDCOD] EQLU (OP_WRT + OP_END)       ! IF WRITE OPERATION
;       4447  THEN                                               ! THEN
;       4448      BEGIN
;       4449
;       4450      INCR INDEX FROM .RPS_X1 TO .RPS_X2 DO         ! LOOK FOR SPARE ENTRY IN
;       4451          BEGIN                                     !     CONTROLLER'S RP_SAVE
;       4452
;       4453          IF .RP_SAVE [.INDEX] LSS 0                ! IF SPARE SLOT FOUND
;       4454          THEN                                      ! THEN
;       4455              BEGIN
;       4456
;       4457              RP_SAVE [.INDEX] = .RP_INDX;          ! SAVE INDEX OF WRITE RETPKT
;       4458              EXITLOOP;                             ! DONE
;       4459
;       4460              END;
;       4461
;       4462          END;
;       4463
;       4464      SWEEP_FLAG = FALSE;                           ! DON'T CALL SWEEP FROM IO_RETPKT
;       4465
;       4466      END
;       4467  ELSE                                              ! ELSE ENDCODE IS READ
;       4468      BEGIN
;       4469
;       4470      IF (INDEX = RPS_REM ()) GEQ 0                 ! IF ASSOCIATED WRITE PACKET IS FOUND
;       4471      THEN                                          ! THEN
;       4472          BEGIN
;       4473
;       4474          BUFFW = RETPKT [.INDEX, BUFF_O];          ! ADDR OF ADDR OF WRITE I/O BUFFER
;       4475          IF CMP_DATA (.BUFFW) EQLU FAILURE         ! COMPARE DATA IN BOTH BUFFERS. IF FAILURE
```

```
;        4476          THEN                                    ! THEN
;        4477              BEGIN
;        4478
;        4479              ERRHRD (42, EGH_30, EMS_42);        ! "I/O REQUEST FAILED"
;        4480              HARD_ERR (1);                       ! INCR HARD ERROR STATISTIC
;        4481
;        4482              END;                                ! IF COMPARE ERROR
;        4483
;        4484          PUT_IO_BUFF (.BUFFW);                   ! RETURN WRITE I/O BUFFER TO POOL
;        4485          PUT_RETPKT (.INDEX);                    ! PUT BACK WRITE RETPKT
;        4486
;        4487          END;                                    ! IF ASSOCIATED WRITE RETPKT WAS FOUND
;        4488
;        4489      END;                                        ! IF ENDCODE WAS READ
;        4490
;        4491  END;                                            ! ROUTINE HOST_WRT_CHK


                                .SBTTL  HOST.WRT.CHK MULTI-DRIVE SUBTEST ROUTINES
000000  004137  000000G       HOST.WRT.CHK:
                                JSR     R1,$SAVE2                       ;                       4419
000004  013700  000000G                   MOV     RP.ADDR,R0              ;                       4446
000010  126027  000014  000242           CMPB    14(R0),#242
000016  001022                           BNE     4$
000020  013700  000000G                   MOV     RPS.X1,R0               ; *,INDEX               4450
000024  005300                           DEC     R0                      ; INDEX
000026  000407                           BR      2$
000030  105760  000000G       1$:         TSTB    RP.SAVE(R0)             ; *(INDEX)              4453
000034  002004                           BGE     2$
000036  113760  000000G 000000G           MOVB    RP.INDX,RP.SAVE(R0)     ; *,*(INDEX)            4457
000044  000404                           BR      3$                      ;                       4455
000046  005200                   2$:     INC     R0                      ; INDEX                 4450
000050  020037  000000G                   CMP     R0,RPS.X2               ; INDEX,*
000054  003765                           BLE     1$
000056  105037  000630'          3$:     CLRB    SWEEP.FLAG              ;                       4464
000062  000207                           RTS     PC                      ;                       4446
000064  004737  000000V          4$:     JSR     PC,RPS.REM              ;                       4470
000070  010002                           MOV     R0,R2                   ; *,INDEX
000072  002434                           BLT     6$
000074  010246                           MOV     R2,-(SP)                ; INDEX,*               4474
000076  012746  000060                   MOV     #60,-(SP)
000102  004737  000000G                   JSR     PC,BL$MUL
000106  062700  000024G                   ADD     #RETPKT+24,R0
000112  010001                           MOV     R0,R1                   ; *,BUFFW
000114  010116                           MOV     R1,(SP)                 ; BUFFW,*               4475
000116  004737  000000V                   JSR     PC,CMP.DATA
000122  005700                           TST     R0
000124  001010                           BNE     5$
000126  104456                           TRAP    56                      ;                       4479
000130  000052                           .WORD   52
000132  000000G                          .WORD   EGH.30
000134  000000G                          .WORD   EMS.42
000136  012716  000001                   MOV     #1,(SP)                 ;                       4480
000142  004737  000000G                   JSR     PC,HARD.ERR
000146  010116                   5$:     MOV     R1,(SP)                 ; BUFFW,*               4484
000150  004737  000000G                   JSR     PC,PUT.IO.BUFF
000154  010216                           MOV     R2,(SP)                 ; INDEX,*               4485
```

```
000156  004737  000000G                        JSR     PC,PUT.RETPKT
000162  022626                                  CMP     (SP)+,(SP)+             ;                       4472
000164  000207                          6$:     RTS     PC                      ;                       4419

; Routine Size:  59 words,      Routine Base:  $CODE$ + 12732
; Maximum stack depth per invocation:  7 words
```

```
;       4492  ROUTINE CMP_DATA (BUFFW) =
;       4493
;       4494  !+
;       4495  !       THIS ROUTINE IS CALLED BY HOST_WRT_CHK TO PERFORM THE ACTUAL WORD-BY-
;       4496  !       WORD COMPARISON BETWEEN THE READ BUFFER DESCRIBED IN THE CURRENT RETURN
;       4497  !       PACKET AND ITS ASSOCIATED WRITE BUFFER. A VALUE OF "SUCCESS" IS
;       4498  !       RETURNED IF THE TWO BUFFERS ARE IDENTICAL. OTHERWISE, "FAILURE" IS
;       4499  !       RETURNED.
;       4500  !
;       4501  !       INPUTS:
;       4502  !               BUFFW - ADDRESS OF THE WRITE BUFFER DESCRIPTOR
;       4503  !
;       4504  !       IMPLICIT INPUTS:
;       4505  !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET (READ OPERATION)
;       4506  !-
;       4507
;       4508  BEGIN
;       4509
;       4510  LOCAL
;       4511      RESULT : WORD,                        ! SUCCESS / FAILURE
;       4512      WBD0 : WORD,                          ! WRITE BUFFER DESCRIPTOR (WORD 0)
;       4513      WBD1 : WORD,                          ! WRITE BUFFER DESCRIPTOR (WORD 1)
;       4514      RBD0 : WORD,                          ! READ BUFFER DESCRIPTOR (WORD 0)
;       4515      RBD1 : WORD,                          ! READ BUFFER DESCRIPTOR (WORD 1)
;       4516      DISP : WORD,                          ! 4K PAGE DISPLACEMENT ADDR
;       4517      PAF : WORD,                           ! PAGE ADDRESS FIELD
;       4518      COUNT : WORD;                         ! BUFFER WORD COUNT
;       4519
;       4520  RESULT = SUCCESS;                         ! ASSUME BLUE SKY
;       4521  WBD0 = ..BUFFW;                           ! WRITE BUFFER DESCRIPTOR (WORD 0)
;       4522  WBD1 = .(.BUFFW + 2);                     ! WRITE BUFFER DESCRIPTOR (WORD 1)
;       4523  RBD0 = .RP_ADDR [BUFF_0];                ! READ BUFFER DESCRIPTOR (WORD 0)
;       4524  RBD1 = .RP_ADDR [BUFF_1];                ! READ BUFFER DESCRIPTOR (WORD 1)
;       4525  IF .MEM_MGMT                             ! IF SYSTEM HAS MEMORY MANAGEMENT
;       4526  THEN                                     ! THEN
;       4527      BEGIN
;       4528
;       4529      DISP = .WBD0 AND %O'17777';           ! WRITE BUFFER DISPLACEMENT
;       4530      PAF = (.WBD0 ^ -6) AND %O'1600';      ! PAGE ADDRESS FIELD
;       4531      KTPAR4 = (.WBD1 ^ 10) OR .PAF;        ! LOAD PAR4
;       4532      WBD0 = %O'100000' OR .DISP;           ! WRITE BUFFER VIRTUAL ADDRESS
;       4533
;       4534      DISP = .RBD0 AND %O'17777';           ! READ BUFFER DISPLACEMENT
;       4535      PAF = (.RBD0 ^ -6) AND %O'1600';      ! PAGE ADDRESS FIELD
;       4536      KTPAR5 = (.RBD1 ^ 10) OR .PAF;        ! LOAD PAR5
;       4537      RBD0 = %O'120000' OR .DISP;           ! READ BUFFER VIRTUAL ADDRESS
;       4538
;       4539      END;
;       4540
;       4541  IF (COUNT = .RP_ADDR [BCNT_LO] / 2) NEQU 0    ! IF WORD COUNT IS NON-ZERO
;       4542  THEN                                          ! THEN
;       4543      BEGIN
;       4544
;       4545      INCR I FROM 1 TO .COUNT DO                ! FOR EACH WORD IN BUFFERS
;       4546          BEGIN
;       4547
;       4548          IF ..WBD0 EQLU ..RBD0                 ! IF WORDS COMPARE O.K.
```

```
;       4549            THEN                                    ! THEN
;       4550                BEGIN
;       4551
;       4552                WBDO = .WBDO + 2;                   ! ADVANCE WRITE BUFFER ADDRESS
;       4553                RBDO = .RBDO + 2;                   ! ADVANCE READ BUFFER ADDRESS
;       4554                IF .MEM_MGMT                        ! IF SYSTEM HAS MEMORY MANAGEMENT
;       4555                THEN                                ! THEN
;       4556                    BEGIN
;       4557
;       4558                    IF (.WBDO AND %O'17777') EQLU 0 ! IF WRITE BUFFER CROSSES 4K PAGE BOUNDARY
;       4559                    THEN                            ! THEN
;       4560                        BEGIN
;       4561
;       4562                        WBDO = %O'100000';          ! RESET VIRTUAL ADDRESS
;       4563                        KTPAR4 = .KTPAR4 + %O'200'; ! ADVANCE PAR TO NEXT 4K PAGE
;       4564
;       4565                        END;
;       4566
;       4567                    IF (.RBDO AND %O'17777') EQLU 0 ! IF READ BUFFER CROSSES 4K PAGE BOUNDARY
;       4568                    THEN                            ! THEN
;       4569                        BEGIN
;       4570
;       4571                        RBDO = %O'120000';          ! RESET VIRTUAL ADDRESS
;       4572                        KTPAR5 = .KTPAR5 + %O'200'; ! ADVANCE PAR TO NEXT 4K PAGE
;       4573
;       4574                        END;
;       4575
;       4576                    END;                            ! END - IF MEMORY MANAGEMENT
;       4577
;       4578                END
;       4579            ELSE                                    ! OTHERWISE - COMPARE ERROR
;       4580                BEGIN
;       4581
;       4582                RESULT = FAILURE;                   ! FAILED
;       4583                EXITLOOP;                           ! NO NEED TO CONTINUE
;       4584
;       4585                END;
;       4586
;       4587            END;                                    ! WORD COMPARE LOOP
;       4588
;       4589        END;                                        ! IF WORD COUNT IS NON-ZERO
;       4590
;       4591    IF .MEM_MGMT                                    ! IF SYSTEM HAS MEMORY MANAGEMENT
;       4592    THEN                                            ! THEN
;       4593        BEGIN
;       4594
;       4595        KTPAR4 = %O'1000';                          ! RESTORE PAR'S
;       4596        KTPAR5 = %O'1200';
;       4597
;       4598        END;
;       4599
;       4600    RETURN .RESULT;
;       4601
;       4602    END;                                            ! ROUTINE CMP_DATA

                                        .SBTTL  CMP.DATA MULTI-DRIVE SUBTEST ROUTINES
```

```
000000  004137  000000G                CMP.DATA:
                                                JSR     R1,$SAVE5                       ;                                               4492
000004  012746  000001                          MOV     #1,-(SP)                        ; *,RESULT                                      4520
000010  017646  000020                          MOV     @20(SP),-(SP)                   ; BUFFW,WBD0                                    4521
000014  016600  000022                          MOV     22(SP),R0                       ; BUFFW,*                                       4522
000020  016004  000002                          MOV     2(R0),R4                        ; *,WBD1
000024  013701  000000G                         MOV     RP.ADDR,R1                      ;                                               4523
000030  016146  000024                          MOV     24(R1),-(SP)                    ; *,RBD0
000034  016105  000026                          MOV     26(R1),R5                       ; *,RBD1                                        4524
000040  005046                                  CLR     -(SP)                           ;                                               4525
000042  113716  000000G                         MOVB    MEM.MGMT,(SP)
000046  032716  000001                          BIT     #1,(SP)
000052  001470                                  BEQ     1$
000054  016602  000004                          MOV     4(SP),R2                        ; WBD0,DISP                                     4529
000060  042702  160000                          BIC     #160000,R2                      ; *,DISP
000064  016646  000004                          MOV     4(SP),-(SP)                     ; WBD0,*                                        4530
000070  012746  177772                          MOV     #-6,-(SP)
000074  004737  000000G                         JSR     PC,BL$SHF
000100  010003                                  MOV     R0,R3                           ; *,PAF
000102  042703  176177                          BIC     #176177,R3                      ; *,PAF
000106  010400                                  MOV     R4,R0                           ; WBD1,*                                        4531
000110  000300                                  SWAB    R0
000112  105000                                  CLRB    R0
000114  006300                                  ASL     R0
000116  006300                                  ASL     R0
000120  010304                                  MOV     R3,R4                           ; PAF,*
000122  050004                                  BIS     R0,R4
000124  010437  172350                          MOV     R4,@#172350
000130  010266  000010                          MOV     R2,10(SP)                       ; DISP,WBD0                                     4532
000134  052766  100000  000010                  BIS     #100000,10(SP)                  ; *,WBD0
000142  016602  000006                          MOV     6(SP),R2                        ; RBD0,DISP                                     4534
000146  042702  160000                          BIC     #160000,R2                      ; *,DISP
000152  016616  000006                          MOV     6(SP),(SP)                      ; RBD0,*                                        4535
000156  012746  177772                          MOV     #-6,-(SP)
000162  004737  000000G                         JSR     PC,BL$SHF
000166  010003                                  MOV     R0,R3                           ; *,PAF
000170  042703  176177                          BIC     #176177,R3                      ; *,PAF
000174  010500                                  MOV     R5,R0                           ; RBD1,*                                        4536
000176  000300                                  SWAB    R0
000200  105000                                  CLRB    R0
000202  006300                                  ASL     R0
000204  006300                                  ASL     R0
000206  010304                                  MOV     R3,R4                           ; PAF,*
000210  050004                                  BIS     R0,R4
000212  010437  172352                          MOV     R4,@#172352
000216  010266  000010                          MOV     R2,10(SP)                       ; DISP,RBD0                                     4537
000222  052766  120000  000010                  BIS     #120000,10(SP)                  ; *,RBD0
000230  062706  000006                          ADD     #6,SP                           ;                                               4527
000234  016146  000020          1$:             MOV     20(R1),-(SP)                    ;                                               4541
000240  012746  000002                          MOV     #2,-(SP)
000244  004737  000000G                         JSR     PC,BL$DIV
000250  022626                                  CMP     (SP)+,(SP)+
000252  005700                                  TST     R0                              ; COUNT
000254  001452                                  BEQ     6$
000256  005001                                  CLR     R1                              ; I                                             4545
000260  000445                                  BR      5$
000262  027676  000004  000002      2$:         CMP     @4(SP),@2(SP)                   ; WBD0,RBD0                                     4548
```

```
000270  001036                          BNE     4$
000272  062766  000002  000004          ADD     #2,4(SP)        ; *,WBDO                4552
000300  062766  000002  000002          ADD     #2,2(SP)        ; *,RBDO                4553
000306  032716  000001                  BIT     #1,(SP)         ;                       4554
000312  001430                          BEQ     5$
000314  032766  017777  000004          BIT     #17777,4(SP)    ; *,WBDO                4558
000322  001006                          BNE     3$
000324  012766  100000  000004          MOV     #-100000,4(SP)  ; *,WBDO                4562
000332  062737  000200  172350          ADD     #200,@#172350   ;                       4563
000340  032766  017777  000002  3$:     BIT     #17777,2(SP)    ; *,RBDO                4567
000346  001012                          BNE     5$
000350  012766  120000  000002          MOV     #-60000,2(SP)   ; *,RBDO                4571
000356  062737  000200  172352          ADD     #200,@#172352   ;                       4572
000364  000403                          BR      5$              ;                       4548
000366  005066  000006          4$:     CLR     6(SP)           ; RESULT                4582
000372  000403                          BR      6$              ;                       4580
000374  005201                  5$:     INC     R1              ; I                     4545
000376  020100                          CMP     R1,R0           ; I,COUNT
000400  003730                          BLE     2$
000402  032716  000001          6$:     BIT     #1,(SP)         ;                       4591
000406  001406                          BEQ     7$
000410  012737  001000  172350          MOV     #1000,@#172350  ;                       4595
000416  012737  001200  172352          MOV     #1200,@#172352  ;                       4596
000424  016600  000006          7$:     MOV     6(SP),R0        ; RESULT,*              4508
000430  062706  000010                  ADD     #10,SP          ;                       4492
000434  000207                          RTS     PC

; Routine Size:  143 words,     Routine Base:  $CODE$ + 13120
; Maximum stack depth per invocation:  14 words
```

```
;        4603  ROUTINE SWEEP : NOVALUE =
;        4604
;        4605  !+
;        4606  !       THIS ROUTINE IS CALLED FROM IO_RETPKT AND OTHERS TO DEALLOCATE THE
;        4607  !       RESOURCES ASSOCIATED WITH THE CURRENT RETURN PACKET. THIS INCLUDES THE
;        4608  !       PACKET ITSELF AND THE I/O BUFFER. IN ADDITION, IF THE HOST IS
;        4609  !       PERFORMING WRITE-COMPARES, AND IF THE CURRENT RETURN PACKET IS A READ
;        4610  !       FUNCTION, THEN THE CURRENT CONTROLLER'S RP_SAVE AREA IS SEARCHED FOR
;        4611  !       THE ASSOCIATED WRITE RETPKT SO THAT ITS RESOURCES CAN ALSO BE
;        4612  !       DEALLOCATED.
;        4613  !
;        4614  !       IMPLICIT INPUTS:
;        4615  !               RP_ADDR - ADDRESS OF CURRENT RETURN PACKET
;        4616  !               RP_INDX - INDEX OF CURRENT RETURN PACKET
;        4617  !-
;        4618
;        4619  BEGIN
;        4620
;        4621  LOCAL
;        4622      INDEX : SIGNED WORD;
;        4623
;        4624  IF (.RP_ADDR [ENDCOD] AND OP_MSK) EQLU OP_RD     ! IF READ OPCODE OR ENDCODE
;        4625  THEN                                             ! THEN
;        4626      BEGIN
;        4627
;        4628      IF BIT_TST (SWP_FLAGS, SWF_HWC)              ! IF HOST IS DOING WRITE-COMPARES
;        4629      THEN                                         ! THEN
;        4630          BEGIN
;        4631
;        4632          IF (INDEX = RPS_REM ()) GEQ 0            ! IF ASSOCIATED WRITE RETPKT IS FOUND
;        4633          THEN                                     ! THEN
;        4634              BEGIN
;        4635
;        4636              PUT_IO_BUFF (RETPKT [.INDEX, BUFF_O]);      ! RETURN WRITE I/O BUFFER TO POOL
;        4637              PUT_RETPKT (.INDEX);                        ! RETURN WRITE PACKET TO POOL
;        4638
;        4639              END;
;        4640
;        4641          END;
;        4642
;        4643      END;
;        4644
;        4645  PUT_IO_BUFF (RP_ADDR [BUFF_O]);                 ! RETURN CURRENT I/O BUFFER TO POOL
;        4646  PUT_RETPKT (.RP_INDX);                           ! RETURN CURRENT RETPKT TO POOL
;        4647
;        4648  END;                                             ! ROUTINE SWEEP


                                        .SBTTL  SWEEP MULTI-DRIVE SUBTEST ROUTINES
000000  010146                  SWEEP:  MOV     R1,-(SP)                        ;       4603
000002  013700  000000G                 MOV     RP.ADDR,R0                      ;       4624
000006  116000  000014                  MOVB    14(R0),R0
000012  042700  177600                  BIC     #177600,R0
000016  020027  000041                  CMP     R0,#41
000022  001026                          BNE     1$
000024  132737  000100  000000G         BITB    #100,SWP.FLAGS                  ;       4628
000032  001422                          BEQ     1$
```

```
000034  004737  000000V                        JSR     PC,RPS.REM              ;                               4632
000040  010001                                 MOV     R0,R1                   ; *,INDEX
000042  002416                                 BLT     1$
000044  010146                                 MOV     R1,-(SP)                ; INDEX,*                       4636
000046  012746  000060                         MOV     #60,-(SP)
000052  004737  000000G                        JSR     PC,BL$MUL
000056  062700  000024G                        ADD     #RETPKT+24,R0
000062  010016                                 MOV     R0,(SP)
000064  004737  000000G                        JSR     PC,PUT.IO.BUFF
000070  010116                                 MOV     R1,(SP)                 ; INDEX,*                       4637
000072  004737  000000G                        JSR     PC,PUT.RETPKT
000076  022626                                 CMP     (SP)+,(SP)+             ;                               4634
000100  013746  000000G             1$:         MOV     RP.ADDR,-(SP)           ;                               4645
000104  062716  000024                         ADD     #24,(SP)
000110  004737  000000G                        JSR     PC,PUT.IO.BUFF
000114  013716  000000G                        MOV     RP.INDX,(SP)            ;                               4646
000120  004737  000000G                        JSR     PC,PUT.RETPKT
000124  005726                                 TST     (SP)+                   ;                               4619
000126  012601                                 MOV     (SP)+,R1                ;                               4603
000130  000207                                 RTS     PC

; Routine Size:  45 words,      Routine Base:  $CODE$ + 13556
; Maximum stack depth per invocation:  4 words
```

```
;        4649  ROUTINE RPS_REM =
;        4650
;        4651  !+
;        4652  !       THIS ROUTINE SEARCHES THE CURRENT CONTROLLER'S RP_SAVE AREA FOR A
;        4653  !       RETURN PACKET WHOSE COMMAND REFERENCE NUMBER (CRN) IS ONE LESS THAN THE
;        4654  !       CRN OF THE CURRENT RETURN PACKET (I.E., SEARCHING FOR THE SAVED WRITE
;        4655  !       OPERATION ASSOCIATED WITH THE CURRENT READ OPERATION). IF FOUND, THE
;        4656  !       RP_SAVE ENTRY IS CLEARED (TO -1) AND THE RETPKT INDEX OF THE WRITE
;        4657  !       OPERATION IS RETURNED TO THE CALLER.
;        4658  !
;        4659  !       IMPLICIT INPUTS:
;        4660  !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;        4661  !               RPS_X1, RPS_X2 - STARTING / ENDING INDECES OF THE CURRENT
;        4662  !                                CONTROLLER'S RP_SAVE AREA
;        4663  !
;        4664  !       OUTPUTS:
;        4665  !               INDEX (VALUE OF THIS ROUTINE) - INDEX OF THE RETPKT CONTAINING
;        4666  !                                A CRN WHICH IS ONE LESS THAN THE CURRENT
;        4667  !-
;        4668
;        4669  BEGIN
;        4670
;        4671  LOCAL
;        4672      INDEX : SIGNED WORD,
;        4673      TEMP : SIGNED WORD;
;        4674
;        4675  INDEX = -1;                                  ! ASSUME NOT FOUND
;        4676  INCR COUNT FROM .RPS_X1 TO .RPS_X2 DO        ! FOR EACH ENTRY IN RP_SAVE
;        4677      BEGIN
;        4678
;        4679      IF (TEMP = .RP_SAVE [.COUNT]) GEQ 0     ! IF THIS IS A VALID RETPKT INDEX
;        4680      THEN                                    ! THEN
;        4681          BEGIN
;        4682
;        4683          IF .RETPKT [.TEMP, CRF_LO] EQLU (.RP_ADDR [CRF_LO] - 1) ! IF CORRECT CRN
;        4684          THEN                                                    ! THEN
;        4685              BEGIN
;        4686
;        4687              INDEX = .TEMP;                                      ! INDEX TO BE RETURNED
;        4688              RP_SAVE [.COUNT] = -1;                              ! INIT ENTRY
;        4689              EXITLOOP;                                           ! DONE
;        4690
;        4691              END;
;        4692
;        4693          END;                                ! IF VALID RETPKT INDEX
;        4694
;        4695      END;                                    ! RP_SAVE ENTRY LOOP
;        4696
;        4697  RETURN .INDEX;
;        4698
;        4699  END;                                        ! ROUTINE RPS_REM


                                .SBTTL  RPS.REM MULTI-DRIVE SUBTEST ROUTINES
000000  004137  000000G  RPS.REM:JSR     R1,$SAVE4                       ;                       4649
000004  012704  177777             MOV     #-1,R4                          ; *,INDEX               4675
000010  013702  000000G             MOV     RPS.X1,R2                       ; *,COUNT               4676
```

```
000014  060402                          ADD     R4,R2                           ; *,COUNT
000016  000426                          BR      2$
000020  116203  000000G           1$:    MOVB    RP.SAVE(R2),R3                  ; *(COUNT),TEMP                 4679
000024  002423                          BLT     2$
000026  010346                          MOV     R3,-(SP)                        ; TEMP,*                        4683
000030  012746  000060                   MOV     #60,-(SP)
000034  004737  000000G                 JSR     PC,BL$MUL
000040  022626                          CMP     (SP)+,(SP)+
000042  013701  000000G                 MOV     RP.ADDR,R1
000046  016101  000004                   MOV     4(R1),R1
000052  005301                          DEC     R1
000054  026001  000004G                 CMP     RETPKT+4(R0),R1
000060  001005                          BNE     2$
000062  010304                          MOV     R3,R4                           ; TEMP,INDEX                    4687
000064  112762  000377  000000G         MOVB    #377,RP.SAVE(R2)                ; *,*(COUNT)                    4688
000072  000404                          BR      3$                              ;                               4685
000074  005202                  2$:     INC     R2                              ; COUNT                         4676
000076  020237  000000G                 CMP     R2,RPS.X2                       ; COUNT,*
000102  003746                          BLE     1$
000104  010400                  3$:     MOV     R4,R0                           ; INDEX,*                       4669
000106  000207                          RTS     PC                              ;                               4649

; Routine Size:  36 words,      Routine Base:  $CODE$ + 13710
; Maximum stack depth per invocation:  8 words
```

```
;        4700  ROUTINE DR_RETPKT : NOVALUE =
;        4701
;        4702  !+
;        4703  !       THIS ROUTINE IS CALLED BY PROC_RETPKT FOR ALL PACKETS ORIGINATING AT
;        4704  !       THE "DRIVER" PORTION OF THE PROGRAM. THIS INCLUDES PACKETS DESCRIBING
;        4705  !       FATAL DEVICE ERRORS AND INDIVIDUAL COMMAND TIMEOUTS.
;        4706  !
;        4707  !       FOR FATAL DEVICE ERRORS, THIS ROUTINE RELEASES ALL RESOURCES HELD BY
;        4708  !       THE CONTROLLER. THE CONTROLLER IS MARKED OFFLINE IN ITS CST, AND ALL
;        4709  !       UNITS ATTACHED TO THE CONTROLLER ARE DROPPED. IF THE RETURN PACKET
;        4710  !       DESCRIBES A COMMAND TIMEOUT, THEN A HARD ERROR IS DECLARED FOR THE
;        4711  !       APPROPRIATE UNIT.
;        4712  !
;        4713  !       IMPLICIT INPUTS:
;        4714  !               RP_INDX - INDEX OF THE CURRENT RETURN PACKET
;        4715  !               RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
;        4716  !               RPS_X1 - STARTING / ENDING INDECES OF THE CURRENT CONTROLLER'S
;        4717  !                               RETPKT SAVE (RP_SAVE) AREA
;        4718  !               CST_ADDR - ADDRESS OF THE CURRENT CONTROLLER'S CST
;        4719  !               CCTLR - CURRENT CONTROLLER NUMBER
;        4720  !-
;        4721
;        4722  BEGIN
;        4723
;        4724  LOCAL
;        4725      INDEX : SIGNED WORD;
;        4726
;        4727  IF .RP_ADDR [MESTYP] EQLU MT_FATAL                  ! IF FATAL CONTROLLER ERROR
;        4728  THEN                                                ! THEN
;        4729      BEGIN
;        4730
;        4731      PUTA_BUFF ();                                   ! RELEASE ALL I/O BUFFERS HELD BY CONTROLLER
;        4732      INCR COUNT FROM .RPS_X1 TO .RPS_X2 DO           ! FOR EACH ENTRY IN CONTROLLER'S RP_SAVE
;        4733          BEGIN
;        4734
;        4735          IF (INDEX = .RP_SAVE [.COUNT]) GEQ 0        ! IF VALID RETPKT INDEX
;        4736          THEN                                        ! THEN
;        4737              BEGIN
;        4738
;        4739              PUT_RETPKT (.INDEX);                    ! RETURN RETPKT TO POOL
;        4740              RP_SAVE [.COUNT] = -1;                  ! INIT ENTRY
;        4741
;        4742              END;
;        4743
;        4744          END;                                        ! RP_SAVE ENTRY LOOP
;        4745
;        4746      QIO [.CCTLR] = 0;                               ! CLEAR NO. OF OUTSTANDING QIOS
;        4747      CST_ADDR [STATE] = OFFLINE;                     ! MARK CST OFFLINE
;        4748      DROP_CTLR (.CCTLR, DU_FATAL);                   ! DROP CONTROLLER'S UNITS
;        4749      PUT_RETPKT (.RP_INDX);                          ! PUT BACK RETPKT
;        4750
;        4751      END
;        4752  ELSE                                                ! ELSE - COMMAND TIMEOUT
;        4753      BEGIN
;        4754
;        4755      FSET_UPAR ();                                   ! SET UP UNIT-RELATED PARAMETERS
;        4756      ERRHRD (43, EGH_30, EMS_43);                    ! "I/O REQUEST FAILED"
```

```
;         4757      HARD_ERR (1);                          ! INCREMENT HARD ERROR STAT FOR UNIT
;         4758      SWEEP ();                              ! RETURN RESOURCES
;         4759      QIO [.CCTLR] = .QIO [.CCTLR] - 1;      ! DECREMENT NO. OF OUTSTANDING QIOS
;         4760
;         4761      END;
;         4762
;         4763  END;                                       ! ROUTINE DR_RETPKT


                                        .SBTTL  DR.RETPKT MULTI-DRIVE SUBTEST ROUTINES
000000  004137  000000G           DR.RETPKT:
                                        JSR     R1,$SAVE3                       ;                       4700
000004  013700  000000G                 MOV     RP.ADDR,R0                      ;                       4727
000010  116000  000002                  MOVB    2(R0),R0
000014  042700  177417                  BIC     #177417,R0
000020  020027  000060                  CMP     R0,#60
000024  001051                          BNE     3$
000026  004737  000000G                 JSR     PC,PUTA.BUFF                    ;                       4731
000032  013703  000000G                 MOV     RPS.X2,R3                       ;                       4732
000036  013701  000000G                 MOV     RPS.X1,R1                       ; *,COUNT
000042  005301                          DEC     R1                              ; COUNT
000044  000412                          BR      2$
000046  116102  000000G           1$:   MOVB    RP.SAVE(R1),R2                  ; *(COUNT),INDEX        4735
000052  002407                          BLT     2$
000054  010246                          MOV     R2,-(SP)                        ; INDEX,*               4739
000056  004737  000000G                 JSR     PC,PUT.RETPKT
000062  112761  000377  000000G        MOVB    #377,RP.SAVE(R1)                ; *,*(COUNT)            4740
000070  005726                          TST     (SP)+                           ;                       4737
000072  005201                  2$:     INC     R1                              ; COUNT                 4732
000074  020103                          CMP     R1,R3                           ; COUNT,*
000076  003763                          BLE     1$
000100  013701  000000G                 MOV     CCTLR,R1                        ;                       4746
000104  105061  000000G                 CLRB    QIO(R1)
000110  013700  000000G                 MOV     CST.ADDR,R0                     ;                       4747
000114  042760  100000  000002         BIC     #100000,2(R0)
000122  010146                          MOV     R1,-(SP)                        ;                       4748
000124  012746  000004                  MOV     #4,-(SP)
000130  004737  000000G                 JSR     PC,DROP.CTLR
000134  013716  000000G                 MOV     RP.INDX,(SP)                    ;                       4749
000140  004737  000000G                 JSR     PC,PUT.RETPKT
000144  005726                          TST     (SP)+                           ;                       4729
000146  000420                          BR      4$                              ;                       4727
000150  004737  012474'          3$:    JSR     PC,FSET.UPAR                    ;                       4755
000154  104456                          TRAP    56                              ;                       4756
000156  000053                          .WORD   53
000160  000000G                         .WORD   EGH.30
000162  000000G                         .WORD   EMS.43
000164  012746  000001                  MOV     #1,-(SP)                        ;                       4757
000170  004737  000000G                 JSR     PC,HARD.ERR
000174  004737  013556'                 JSR     PC,SWEEP                        ;                       4758
000200  013700  000000G                 MOV     CCTLR,R0                        ;                       4759
000204  105360  000000G                 DECB    QIO(R0)
000210  005726                  4$:     TST     (SP)+                           ;                       4722
000212  000207                          RTS     PC                              ;                       4700

; Routine Size:  70 words,      Routine Base:  $CODE$ + 14020
; Maximum stack depth per invocation:  7 words
```

```
;        4764  ROUTINE DRV_TIMCHK : NOVALUE =
;        4765
;        4766  !+
;        4767  !       THIS ROUTINE IS CALLED ONCE PER SECOND FROM EITHER THE MULTI-DRIVE
;        4768  !       SUBTEST EXECUTIVE (MULTI_DRIVE) OR THE DM EXERCISER SUBTEST EXECUTIVE
;        4769  !       (DM_EXER). ITS PURPOSE IS TO DECREMENT THE TIMING COUNTS FOR ALL
;        4770  !       OUTSTANDING COMMANDS, AND TO PROCESS ANY THAT REACH ZERO.
;        4771  !
;        4772  !       A COMMAND TIMER THAT REACHES ZERO CAN MEAN ONE OF TWO THINGS. IF THE
;        4773  !       OWNERSHIP BIT OF THE ENVELOPE DESCRIPTOR OF THE TIMED OUT COMMAND
;        4774  !       INDICATES THAT THE SLOT IS STILL CONTROLLER-OWNED, THEN THE CONTROLLER
;        4775  !       IS ASSUMED DEAD. A DEVICE-FATAL ERROR IS DECLARED AND ALL CONTROLLER-
;        4776  !       RELATED DATA IN THE "DRIVER" ARE INITIALIZED. IF THE AFOREMENTIONED
;        4777  !       OWNERSHIP BIT INDICATES THAT THE SLOT HAS BEEN RETURNED TO THE HOST, OR
;        4778  !       IF THE ENVELOPE DESCRIPTOR SLOT FOR THE TIMED OUT COMMAND CANNOT BE
;        4779  !       FOUND (I.E., OVERWRITTEN WITH ANOTHER DESCRIPTOR), THEN ONLY THE
;        4780  !       COMMAND IS ASSUMED LOST.
;        4781  !
;        4782  !       IN EITHER CASE, A RETURN PACKET IS SET UP WHICH DESCRIBES THE
;        4783  !       CONDITION, AND "SENT" TO THE "EXERCISER".
;        4784  !-
;        4785
;        4786  BEGIN
;        4787
;        4788  LOCAL
;        4789      M_INDEX : SIGNED WORD,                          ! MSCP ENVELOPE INDEX
;        4790      M_ADDR,                                         ! MSCP ENVELOPE ADDRESS
;        4791      CR_ADDR,                                        ! COMMAND RING (C-RING) ADDRESS
;        4792      TYPE : WORD;                                    ! TYPE OF ERROR
;        4793
;        4794  INCR CNTLR FROM 0 TO (MAX_CTLR - 1) DO              ! FOR EACH CONTROLLER
;        4795      BEGIN
;        4796
;        4797      SET_CPAR (.CNTLR);                              ! SET UP CONTROLLER-RELATED DATA
;        4798      IF .CST_ADDR [STATE] EQLU ONLINE                ! IF CONTROLLER IS ONLINE
;        4799      THEN                                            ! THEN
;        4800          BEGIN
;        4801
;        4802          INCR COUNT FROM .OCL_X1 TO .OCL_X2 DO       ! FOR EACH OUTSTANDING COMMAND ENTRY
;        4803          BEGIN
;        4804
;        4805              IF (M_INDEX = .OUTC_LIST [.COUNT]) GEQ 0    ! IF ENTRY HOLDS VALID MSCP INDEX
;        4806              THEN                                        ! THEN
;        4807                  BEGIN
;        4808
;        4809                  OUTC_TIMR [.COUNT] = .OUTC_TIMR [.COUNT] - 1;   ! DECREMENT COMMAND TIMER
;        4810                  IF .OUTC_TIMR [.COUNT] EQLU 0                   ! IF COMMAND TIMED OUT
;        4811                  THEN                                            ! THEN
;        4812                      BEGIN
;        4813
;        4814                      OUTC_LIST [.COUNT] = -1;                        ! KILL OUTC ENTRY
;        4815                      M_ADDR = MSCP_ENV + (.M_INDEX * ENV_LEN * 2);   ! CALCULATE MSCP ENV ADDRESS
;        4816                      CR_ADDR = .DCT_ADDR [CR_BEG];                   ! GET START OF C-RING
;        4817                      TYPE = MT_TIMEOUT;                              ! ASSUME COMMAND TIMED OUT
;        4818                      INCR SLOT FROM 1 TO CRING_LEN DO                ! FOR EACH CRING SLOT
;        4819                          BEGIN
;        4820
```

```
;        4821                        IF ..CR_ADDR EQLA (.M_ADDR + 8)                 ! IF THIS SLOT HOLDS TIMED OUT COMMAND
;        4822                        THEN                                            ! THEN
;        4823                            BEGIN
;        4824
;        4825                            CR_ADDR = .CR_ADDR + 2;                     ! ADVANCE TO HI-ORDER WORD
;        4826                            IF BIT_TST (.CR_ADDR, ED_OWN)               ! IF SLOT IS STILL CTLR-OWNED
;        4827                            THEN                                        ! THEN
;        4828                                BEGIN
;        4829
;        4830                                WRT_RC25 (RCIP, RC_ALL, ALL_ONES);      ! WRITE IP TO STOP DEVICE
;        4831                                L$LUN = .CST_ADDR [OF_UN, P_UNIT];      ! 1ST UNIT # IN CTLR
;        4832                                ERRDF (20, EGD_20, EMS_20);             ! "CONTROLLER TIMEOUT"
;        4833                                DRV_CTLERR (.CNTLR);                    ! CLEAN UP CTLR DATA IN DRIVER
;        4834                                TYPE = MT_FATAL;                        ! CHANGE TYPE TO FATAL ERROR
;        4835
;        4836                                END;                                    ! IF SLOT STILL CTLR-OWNED
;        4837
;        4838                            EXITLOOP;                                   ! EXIT C-RING SLOT SEARCH LOOP
;        4839
;        4840                            END;                                        ! IF COMMAND SLOT IS FOUND
;        4841
;        4842                        CR_ADDR = .CR_ADDR + 4;                         ! ADVANCE TO NEXT C-RING SLOT
;        4843
;        4844                        END;                                    ! C-RING SLOT SEARCH LOOP
;        4845
;        4846                    IF (RP_INDX = GET_RETPKT (.CNTLR)) GEQ 0    ! IF RETPKT IS AVAILABLE
;        4847                    THEN                                        ! THEN
;        4848                        BEGIN
;        4849
;        4850                        RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2);
;        4851                        IF .TYPE EQLU MT_TIMEOUT                    ! IF COMMAND TIMEOUT
;        4852                        THEN                                        ! THEN
;        4853                            BEGIN
;        4854
;        4855                            COPY_BLK (.M_ADDR + 4, .RP_ADDR, RP_LEN);   ! COPY COMMAND TO RETPKT
;        4856                            PUT_ENV (.M_INDEX);                         ! RETURN COMMAND ENVELOPE TO POOL
;        4857
;        4858                            END;                                    ! IF TYPE = COMMAND TIMEOUT
;        4859
;        4860                        RP_ADDR [CTLR] = .CNTLR;                    ! LOAD CTLR # INTO RETPKT
;        4861                        RP_ADDR [MESTYP] = .TYPE;           ! LOAD TYPE
;        4862                        RP_ADDR [CONID] = CID_DRIVER;       ! LOAD PACKET ORIGINATOR
;        4863                        IN_IODQ (.RP_INDX);                 ! PASS RETPKT TO EXERCISER VIA IODQ
;        4864
;        4865                        END;                                ! IF RETPKT IS AVAILABLE
;        4866
;        4867                    END;                                    ! IF COMMAND TIMER REACHED ZERO
;        4868
;        4869                END;                                ! IF OUTC ENTRY HOLDS A COMMAND ENVELOPE INDEX
;        4870
;        4871            END;                                    ! OUTSTANDING COMMAND ENTRY LOOP
;        4872
;        4873        END;                                        ! IF CONTROLLER IS ONLINE
;        4874
;        4875    END;                                            ! CONTROLLER LOOP
;        4876
;        4877    T_FLAG = FALSE;                                 ! CLEAR ENTRY FLAG
```

```
:       4878
:       4879  END;                                  ! ROUTINE DRV_TIMCHK


                                      .SBTTL  DRV.TIMCHK MULTI-DRIVE SUBTEST ROUTINES
000000  004137  000000G       DRV.TIMCHK:
                              JSR     R1,$SAVE5                       ;                       4764
000004  162706  000006        SUB     #6,SP
000010  005005                CLR     R5                              ; CNTLR                 4794
000012  010546          1$:   MOV     R5,-(SP)                        ; CNTLR,*               4797
000014  004737  000000G       JSR     PC,SET.CPAR
000020  013700  000000G       MOV     CST.ADDR,R0                     ;                       4798
000024  005760  000002        TST     2(R0)
000030  100402                BMI     2$
000032  000137  014744'        JMP     11$
000036  013702  000000G   2$: MOV     OCL.X1,R2                       ; *,COUNT               4802
000042  005302                DEC     R2                              ; COUNT
000044  000137  014730'   3$: JMP     10$
000050  116200  000000G   4$: MOVB    OUTC.LIST(R2),R0                ; *(COUNT),*            4805
000054  010066  000004        MOV     R0,4(SP)                        ; *,M.INDEX
000060  002771                BLT     3$
000062  010200                MOV     R2,R0                           ; COUNT,*               4809
000064  006300                ASL     R0
000066  005360  000000G       DEC     OUTC.TIMR(R0)
000072  001364                BNE     3$                              ;                       4810
000074  112762  000377  000000G MOVB   #377,OUTC.LIST(R2)             ; *,*(COUNT)            4814
000102  016616  000004        MOV     4(SP),(SP)                      ; M.INDEX,*             4815
000106  012746  000104        MOV     #104,-(SP)
000112  004737  000000G       JSR     PC,BL$MUL
000116  062700  000000G       ADD     #MSCP.ENV,R0
000122  010066  000010        MOV     R0,10(SP)                       ; *,M.ADDR
000126  013700  000000G       MOV     DCT.ADDR,R0                     ;                       4816
000132  016066  000010  000004 MOV     10(R0),4(SP)                    ; *,CR.ADDR
000140  012704  000004        MOV     #4,R4                           ; *,TYPE                4817
000144  016603  000010        MOV     10(SP),R3                       ; M.ADDR,*              4821
000150  062703  000010        ADD     #10,R3
000154  012701  000010        MOV     #10,R1                          ; *,SLOT                4818
000160  027603  000004   5$: CMP     @4(SP),R3                       ; CR.ADDR,*             4821
000164  001041                BNE     6$
000166  062766  000002  000004 ADD    #2,4(SP)                        ; *,CR.ADDR             4825
000174  017600  000004        MOV     @4(SP),R0                       ; CR.ADDR,*             4826
000200  042700  077777        BIC     #77777,R0
000204  020027  100000        CMP     R0,#-100000
000210  001034                BNE     7$
000212  012700  177777        MOV     #-1,R0                          ; *,RC.REG              4830
000216  010077  000000G       MOV     R0,@RC25.ADDR                   ; RC.REG,*
000222  013700  000000G       MOV     CST.ADDR,R0                     ;                       4831
000226  016046  000006        MOV     6(R0),-(SP)
000232  000316                SWAB    (SP)
000234  042716  177740        BIC     #177740,(SP)
000240  012637  000000G       MOV     (SP)+,L$LUN
000244  104455                TRAP    55                              :                       4832
000246  000024                .WORD   24
000250  000000G               .WORD   EGD.20
000252  000000G               .WORD   EMS.20
000254  010516                MOV     R5,(SP)                         ; CNTLR,*               4833
000256  004737  000000G       JSR     PC,DRV.CTLERR
```

```
000262  012704  000003                          MOV     #3,R4                           ; *,TYPE                        4834
000266  000405                                  BR      7$                              ;                               4823
000270  062766  000004  000004          6$:     ADD     #4,4(SP)                        ; *,CR.ADDR                     4842
000276  005301                                  DEC     R1                              ; SLOT                          4818
000300  001327                                  BNE     5$
000302  010516                          7$:     MOV     R5,(SP)                         ; CNTLR,*                       4846
000304  004737  000000G                         JSR     PC,GET.RETPKT
000310  010037  000000G                         MOV     R0,RP.INDX
000314  002466                                  BLT     9$
000316  010016                                  MOV     R0,(SP)                         ; RP.INDX,*                     4850
000320  012746  000060                          MOV     #60,-(SP)
000324  004737  000000G                         JSR     PC,BL$MUL
000330  062700  000000G                         ADD     #RETPKT,R0
000334  010037  000000G                         MOV     R0,RP.ADDR
000340  020427  000004                          CMP     R4,#4                           ; TYPE,*                        4851
000344  001016                                  BNE     8$
000346  016616  000012                          MOV     12(SP),(SP)                     ; M.ADDR,*                      4855
000352  062716  000004                          ADD     #4,(SP)
000356  010046                                  MOV     R0,-(SP)                        ; RP.ADDR,*
000360  012746  000030                          MOV     #30,-(SP)
000364  004737  000000G                         JSR     PC,COPY.BLK
000370  016616  000014                          MOV     14(SP),(SP)                     ; M.INDEX,*                     4856
000374  004737  000000G                         JSR     PC,PUT.ENV
000400  022626                                  CMP     (SP)+,(SP)+                     ;                               4853
000402  013700  000000G          8$:     MOV     RP.ADDR,R0                      ;                               4860
000406  062700  000002                          ADD     #2,R0
000412  010501                                  MOV     R5,R1                           ; CNTLR,*
000414  042701  177760                          BIC     #177760,R1
000420  142710  000017                          BICB    #17,(R0)
000424  150110                                  BISB    R1,(R0)
000426  010401                                  MOV     R4,R1                           ; TYPE,*                        4861
000430  006301                                  ASL     R1
000432  006301                                  ASL     R1
000434  006301                                  ASL     R1
000436  006301                                  ASL     R1
000440  042701  177417                          BIC     #177417,R1
000444  142710  000360                          BICB    #360,(R0)
000450  150110                                  BISB    R1,(R0)
000452  112760  000003  000001                  MOVB    #3,1(R0)                        ;                               4862
000460  013716  000000G                         MOV     RP.INDX,(SP)                    ;                               4863
000464  004737  000000G                         JSR     PC,IN.IODQ
000470  005726                                  TST     (SP)+                           ;                               4848
000472  005726                          9$:     TST     (SP)+                           ;                               4812
000474  005202                          10$:    INC     R2                              ; COUNT                         4802
000476  020237  000000G                         CMP     R2,OCL.X2                       ; COUNT,*
000502  003002                                  BGT     11$
000504  000137  014304'                         JMP     4$
000510  005726                          11$:    TST     (SP)+                           ;                               4795
000512  005205                                  INC     R5                              ; CNTLR                         4794
000514  020527  000003                          CMP     R5,#3                           ; CNTLR,*
000520  003002                                  BGT     12$
000522  000137  014246'                         JMP     1$
000526  105037  000000G          12$:    CLRB    T.FLAG                          ;                               4877
000532  062706  000006                          ADD     #6,SP                           ;                               4764
000536  000207                                  RTS     PC
; Routine Size:  176 words,     Routine Base:  $CODE$ + 14234
; Maximum stack depth per invocation:  15 words
```

```
;          4880  %SBTTL 'RC25 INTERRUPT SERVICE ROUTINES'
;          4881
;          4882  !+
;          4883  !       THERE EXISTS AN RC25 INTERRUPT SERVICE ROUTINE FOR EACH DEVICE
;          4884  !       CONTROLLER. EACH SERVICE ROUTINE BEGINS BY SIMPLY SETTING THE
;          4885  !       APPROPRIATE CONTROLLER NUMBER INTO "ICTLR". ALL SERVICE ROUTINES THEN
;          4886  !       BRANCH TO A COMMON INTERRUPT PROCESSING ROUTINE.
;          4887  !-
;          4888
;          4889  BGNSRV (RCINT0);
;          4890
;          4891  ICTLR = 0;
;          4892  RCINT ();
;          4893
;          4894  ENDSRV;


                                        .SBTTL  RCINT0 RC25 INTERRUPT SERVICE ROUTINES
000000  010046                  RCINT0::MOV     R0,-(SP)                        ;                       4889
000002  005037  000600'                 CLR     ICTLR                           ;                       4891
000006  004737  000000V                 JSR     PC,RCINT                        ;                       4892
000012  012600                          MOV     (SP)+,R0                        ;                       4889
000014  000002                          RTI

; Routine Size:  7 words,       Routine Base:  $CODE$ + 14774
; Maximum stack depth per invocation:  2 words


;          4895
;          4896  BGNSRV (RCINT1);
;          4897
;          4898  ICTLR = 1;
;          4899  RCINT ();
;          4900
;          4901  ENDSRV;


                                        .SBTTL  RCINT1 RC25 INTERRUPT SERVICE ROUTINES
000000  010046                  RCINT1::MOV     R0,-(SP)                        ;                       4896
000002  012737  000001  000600'         MOV     #1,ICTLR                        ;                       4898
000010  004737  000000V                 JSR     PC,RCINT                        ;                       4899
000014  012600                          MOV     (SP)+,R0                        ;                       4896
000016  000002                          RTI

; Routine Size:  8 words,       Routine Base:  $CODE$ + 15012
; Maximum stack depth per invocation:  2 words


;          4902
;          4903  BGNSRV (RCINT2);
;          4904
;          4905  ICTLR = 2;
;          4906  RCINT ();
;          4907
;          4908  ENDSRV;
```

```
                                        .SBTTL  RCINT2 RC25 INTERRUPT SERVICE ROUTINES
000000  010046                  RCINT2::MOV     R0,-(SP)                ;                       4903
000002  012737  000002  000600'         MOV     #2,ICTLR                ;                       4905
000010  004737  000000V                 JSR     PC,RCINT                ;                       4906
000014  012600                          MOV     (SP)+,R0                ;                       4903
000016  000002                          RTI

; Routine Size:  8 words,       Routine Base:  $CODE$ + 15032
; Maximum stack depth per invocation:  2 words


;         4909
;         4910  BGNSRV (RCINT3);
;         4911
;         4912  ICTLR = 3;
;         4913  RCINT ();
;         4914
;         4915  ENDSRV;

                                        .SBTTL  RCINT3 RC25 INTERRUPT SERVICE ROUTINES
000000  010046                  RCINT3::MOV     R0,-(SP)                ;                       4910
000002  012737  000003  000600'         MOV     #3,ICTLR                ;                       4912
000010  004737  000000V                 JSR     PC,RCINT                ;                       4913
000014  012600                          MOV     (SP)+,R0                ;                       4910
000016  000002                          RTI

; Routine Size:  8 words,       Routine Base:  $CODE$ + 15052
; Maximum stack depth per invocation:  2 words
```

```
;        4916  ROUTINE RCINT : NOVALUE =
;        4917
;        4918  !+
;        4919  !       THIS IS THE COMMON INTERRUPT SERVICE ROUTINE FOR ALL RC25 CONTROLLERS.
;        4920  !       AFTER SETTING UP THE COMMONLY-USED CONTROLLER-RELATED DATA ITEMS, THIS
;        4921  !       ROUTINE WILL SAVE THE CURRENT CONTENTS OF THE SA REGISTER IN THE DCT.
;        4922  !       THEN, IF THE "IGNORE INTERRUPT" BIT IS SET, NO FURTHER ACTION IS TAKEN.
;        4923  !
;        4924  !       IF THE INITIALIZATION SUBTEST IS IN PROGRESS OR IF THE DM EXERCISER
;        4925  !       SUBTEST BEING RUN, THEN INTERRUPT PROCESSING CONTINUES BY CHECKING THE
;        4926  !       ERROR BIT OF THE SA REGISTER, AND POLLING THE COMMAND AND RESPONSE
;        4927  !       RINGS. OTHERWISE, THE INTERRUPT IS PROCESSED LATER WHEN INT_PROC IS
;        4928  !       CALLED FROM THE MULTI-DRIVE SUBTEST "EXECUTIVE" (MULTI_DRIVE).
;        4929  !-
;        4930
;        4931  BEGIN
;        4932
;        4933  ISET_CPAR ();                                    ! SET UP CONTROLLER-RELATED PARAMETERS
;        4934  IDCT_ADDR [SA_SAVE] = .IRC25_ADDR [RCSA, RC_ALL]; ! SAVE SA REGISTER
;        4935  IF .IDCT_ADDR [IG_INT] EQLU YES                  ! IGNORE INTERRUPT?
;        4936  THEN
;        4937      RETURN;                                      ! YES - RETURN
;        4938  IF ((.IIP_FLAG) OR (BIT_TST (SWP_FLAGS, SWF_DM))) ! IF INIT SUBTEST OR DM EXERCISER
;        4939  THEN                                             ! THEN
;        4940      BEGIN
;        4941
;        4942      IF BIT_TST (IDCT_ADDR [SA_SAVE], SA_ERR)     ! IF FATAL ERROR
;        4943      THEN                                         ! THEN
;        4944          FATAL_ERROR ()                           ! DECLARE ERROR AND CLEAN UP
;        4945      ELSE                                         ! OTHERWISE
;        4946          BEGIN
;        4947
;        4948          POLL_CRING ();                           ! POLL COMMAND RING
;        4949          POLL_RRING ();                           ! POLL RESPONSE RING
;        4950
;        4951          END;
;        4952
;        4953      END;                                         ! IF INIT SUBTEST OR DM EXERCISER
;        4954
;        4955  END;                                             ! ROUTINE RCINT


                                        .SBTTL  RCINT RC25 INTERRUPT SERVICE ROUTINES
000000  010146                  RCINT:  MOV     R1,-(SP)                        ;                       4916
000002  005746                          TST     -(SP)
000004  004737  000000V                 JSR     PC,ISET.CPAR                    ;                       4933
000010  013700  000566'                 MOV     IDCT.ADDR,R0                    ;                       4934
000014  013701  000000G                 MOV     IRC25.ADDR,R1
000020  016116  000002                  MOV     2(R1),(SP)                      ; *,RC.REG
000024  011660  000002                  MOV     (SP),2(R0)                      ; RC.REG,*
000030  032710  040000                  BIT     #40000,(R0)                     ; *,IDCT.ADDR           4935
000034  001026                          BNE     3$                              ;                       4937
000036  132737  000001  000000G         BITB    #1,IIP.FLAG                     ;                       4938
000044  001004                          BNE     1$
000046  132737  000002  000000G         BITB    #2,SWP.FLAGS
000054  001416                          BEQ     3$
000056  016000  000002          1$:     MOV     2(R0),R0                        ;                       4942
```

```
000062  042700  077777                          BIC     #77777,R0
000066  020027  100000                          CMP     R0,#-100000
000072  001003                                  BNE     2$
000074  004737  000000V                        JSR     PC,FATAL.ERROR          ;                       4944
000100  000404                                  BR      3$                      ;                       4942
000102  004737  000000V              2$:       JSR     PC,POLL.CRING           ;                       4948
000106  004737  000000V                        JSR     PC,POLL.RRING           ;                       4949
000112  005726                      3$:       TST     (SP)+                   ;                       4916
000114  012601                                MOV     (SP)+,R1
000116  000207                                RTS     PC

; Routine Size:  40 words,      Routine Base:  $CODE$ + 15072
; Maximum stack depth per invocation:  3 words
```

```
;        4956  ROUTINE INT_PROC : NOVALUE =
;        4957
;        4958  !+
;        4959  !       THIS ROUTINE IS CALLED FROM THE "EXECUTIVE" OF THE MULTI-DRIVE SUBTEST
;        4960  !       TO PROCESS ANY RC25 INTERRUPTS WHICH MAY HAVE OCCURRED IN THE RECENT
;        4961  !       PAST.
;        4962  !
;        4963  !       IF EITHER THE INITIALIZATION SUBTEST OR DM EXERCISER SUBTEST IS BEING
;        4964  !       RUN, THEN INTERRUPTS ARE FULLY PROCESSED AS THEY OCCUR (SEE RCINT).
;        4965  !       OTHERWISE, FOR THE MULTI-DRIVE SUBTEST, INTERRUPTS ARE PROCESSED ON A
;        4966  !       DELAYED BASIS TO PREVENT PROGRAM PRINTOUTS FROM BEING BROKEN UP BY,
;        4967  !       SAY, ERROR LOG MESSAGE PRINTOUTS.
;        4968  !
;        4969  !       AFTER SOME INITIALIZATION, THIS ROUTINE CONTINUES AS IF AN INTERRUPT
;        4970  !       HAS JUST OCCURRED: IF THE ERROR BIT OF THE SA REGISTER IS SET, THEN A
;        4971  !       FATAL ERROR IS DECLARED. OTHERWISE, THE COMMAND AND RESPONSE RINGS ARE
;        4972  !       POLLED.
;        4973  !
;        4974  !       IF NO INTERRUPT HAS OCCURRED IN THE RECENT PAST, THEN THE POLLING
;        4975  !       ROUTINES TAKE NO ACTION.
;        4976  !
;        4977  !       NOTE: ALL RC25'S ARE ASSUMMED TO BE HARDWIRED AT BR LEVEL 5 OR LESS.
;        4978  !       INTERRUPT PROCESSING RUNS AT LEVEL 5, AND CANNOT BE BROKEN BY AN
;        4979  !       INTERRUPT FROM A SECOND RC25.
;        4980  !
;        4981  !       IMPLICIT INPUTS:
;        4982  !               CCTLR - CURRENT CONTROLLER NUMBER
;        4983  !-
;        4984
;        4985  BEGIN
;        4986
;        4987  SETPRI (PRI05);                                 ! LOCK OUT ALL RC25 INTERRUPTS
;        4988  ICTLR = .CCTLR;                                 ! FAKE INTERRUPTING CONTROLLER NUMBER
;        4989  ISET_CPAR ();                                   ! SET UP CONTROLLER-RELATED DATA
;        4990  IDCT_ADDR [SA_SAVE] = .IRC25_ADDR [RCSA, RC_ALL]; ! SAVE SA REGISTER
;        4991  IF BIT_TST (IDCT_ADDR [SA_SAVE], SA_ERR)       ! IF FATAL ERROR
;        4992  THEN
;        4993      FATAL_ERROR ()
;        4994  ELSE                                            ! OTHERWISE
;        4995      BEGIN
;        4996
;        4997      POLL_CRING ();                              ! POLL COMMAND RING
;        4998      POLL_RRING ();                              ! POLL RESPONSE RING
;        4999
;        5000      END;
;        5001
;        5002  SETPRI (PRI00);                                 ! RESTORE PROCESSOR PRIORITY
;        5003
;        5004  END;                                            ! ROUTINE INT_PROC


                                .SBTTL  INT.PROC RC25 INTERRUPT SERVICE ROUTINES
000000  010146                  INT.PROC:
                                MOV     R1,-(SP)                                ;                4956
000002  005746                  TST     -(SP)
000004  012700  000240          MOV     #240,R0                                 ;                4987
000010  104441                  TRAP    41
```

```
000012  013737  000000G 000600'          MOV     CCTLR,ICTLR                     ;                               4988
000020  004737  000000V                  JSR     PC,ISET.CPAR                    ;                               4989
000024  013700  000566'                  MOV     IDCT.ADDR,R0                    ;                               4990
000030  013701  000000G                  MOV     IRC25.ADDR,R1
000034  016116  000002                   MOV     2(R1),(SP)                      ; *,RC.REG
000040  011660  000002                   MOV     (SP),2(R0)                      ; RC.REG,*
000044  011600                           MOV     (SP),R0                         ;                               4991
000046  042700  077777                   BIC     #77777,R0
000052  020027  100000                   CMP     R0,#-100000
000056  001003                           BNE     1$
000060  004737  000000V                  JSR     PC,FATAL.ERROR                  ;                               4993
000064  000404                           BR      2$                              ;                               4991
000066  004737  000000V          1$:     JSR     PC,POLL.CRING                   ;                               4997
000072  004737  000000V                  JSR     PC,POLL.RRING                   ;                               4998
000076  005000                  2$:     CLR     R0                              ;                               5002
000100  104441                           TRAP    41
000102  005726                           TST     (SP)+                           ;                               4956
000104  012601                           MOV     (SP)+,R1
000106  000207                           RTS     PC

; Routine Size:  36 words,      Routine Base:  $CODE$ + 15212
; Maximum stack depth per invocation:  4 words
```

```
;          5005  ROUTINE ISET_CPAR : NOVALUE =
;          5006
;          5007  !+
;          5008  !       THIS ROUTINE IS CALLED BY RCINT AND INT_PROC TO SET UP THE COMMONLY-
;          5009  !       USED DATA ITEMS FOR THE INTERRUPTING CONTROLLER.
;          5010  !
;          5011  !       IMPLICIT INPUTS:
;          5012  !               ICTLR - INTERRUPTING CONTROLLER NUMBER
;          5013  !-
;          5014
;          5015  BEGIN
;          5016
;          5017  IDCT_ADDR = DCT + (.ICTLR * DCT_LEN * 2);        ! CALCULATE DCT ADDRESS
;          5018  ICST_ADDR = CST + (.ICTLR * CST_LEN * 2);        ! CALCULATE CST ADDRESS
;          5019  IRC25_ADDR = .ICST_ADDR [IP_ADDR];               ! GET RC25 ADDRESS
;          5020  ICOM_ADDR = COMM_AREA + (.ICTLR * COMM_LEN * 2); ! CALCULATE COMM_AREA ADDR
;          5021
;          5022  END;                                            ! ROUTINE ISET_CPAR


                                .SBTTL  ISET.CPAR RC25 INTERRUPT SERVICE ROUTINES
000000  010146                ISET.CPAR:
                                MOV     R1,-(SP)                        ;                       5005
000002  013701  000600'         MOV     ICTLR,R1                        ;                       5017
000006  010146                  MOV     R1,-(SP)
000010  012746  000022          MOV     #22,-(SP)
000014  004737  000000G         JSR     PC,BL$MUL
000020  062700  000000G         ADD     #DCT,R0
000024  010037  000566'         MOV     R0,IDCT.ADDR
000030  010116                  MOV     R1,(SP)                 ;               5018
000032  012746  000016          MOV     #16,-(SP)
000036  004737  000000G         JSR     PC,BL$MUL
000042  062700  000000G         ADD     #CST,R0
000046  010037  000564'         MOV     R0,ICST.ADDR
000052  011037  000000G         MOV     (R0),IRC25.ADDR         ; ICST.ADDR,*           5019
000056  010116                  MOV     R1,(SP)                 ;                       5020
000060  012746  000110          MOV     #110,-(SP)
000064  004737  000000G         JSR     PC,BL$MUL
000070  062700  000000'         ADD     #COMM.AREA,R0
000074  010037  000560'         MOV     R0,ICOM.ADDR
000100  062706  000010          ADD     #10,SP                  ;                       5015
000104  012601                  MOV     (SP)+,R1                ;                       5005
000106  000207                  RTS     PC

; Routine Size:  36 words,      Routine Base:  $CODE$ + 15322
; Maximum stack depth per invocation:  6 words
```

```
;         5023  ROUTINE FATAL_ERROR : NOVALUE =
;         5024
;         5025  !+
;         5026  !       THIS ROUTINE IS CALLED BY BOTH RCINT AND INT_PROC UPON DETECTING AN
;         5027  !       UNRECOVERABLE ERROR THROUGH THE DEVICE'S SA REGISTER. ITS PURPOSE IS TO
;         5028  !       CLEAN UP DEVICE DATA IN THE "DRIVER" PORTION OF THE EXERCISER, AND TO
;         5029  !       INFORM THE "PROGRAM" PORTION OF THE EVENT VIA RETURN PACKET.
;         5030  !
;         5031  !       IMPLICIT INPUTS:
;         5032  !               ICTLR - INTERRUPTING CONTROLLER NUMBER
;         5033  !               IDCT_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S DCT
;         5034  !               ICST_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S CST
;         5035  !-
;         5036
;         5037  BEGIN
;         5038
;         5039  LOCAL
;         5040      INDEX : SIGNED WORD,
;         5041      U_SAVE : WORD;
;         5042
;         5043  SA_REG = .IDCT_ADDR [SA_SAVE];
;         5044  U_SAVE = .L$LUN;                          ! SAVE PRE-INTERRUPT CURRENT UNIT NUMBER
;         5045  L$LUN = .ICST_ADDR [OF_UN, P_UNIT];       ! SET CURRENT UNIT TO FIRST IN CONTROLLER
;         5046  ERRDF (14, EGD_14, EMS_14);               ! "FATAL PORT / CONTROLLER ERROR"
;         5047  L$LUN = .U_SAVE;                          ! RESTORE PRE-INTERRUPT CURRENT UNIT
;         5048  DRV_CTLERR (.ICTLR);                      ! CLEAN UP DRIVER DATA FOR CONTROLLER
;         5049  IF (INDEX = GET_RETPKT (.ICTLR)) GEQ 0    ! TRY TO GET A RETPKT; IF SUCCESS
;         5050  THEN
;         5051      BEGIN
;         5052
;         5053      RETPKT [.INDEX, CONID] = CID_DRIVER;  ! SET CONNECTION ID TO "DRIVER"
;         5054      RETPKT [.INDEX, MESTYP] = MT_FATAL;   ! FATAL ERROR
;         5055      RETPKT [.INDEX, CTLR] = .ICTLR;       ! CONTROLLER NUMBER
;         5056      IN_IODQ (.INDEX);                     ! LOAD RETPKT INDEX INTO IODQ
;         5057
;         5058      END;                                  ! IF RETPKT WAS ALLOCATED
;         5059
;         5060  END;                                      ! ROUTINE FATAL_ERR


                                        .SBTTL  FATAL.ERROR RC25 INTERRUPT SERVICE ROUTINES
000000  004137  000000G         FATAL.ERROR:
                                        JSR     R1,$SAVE2                       ;                       5023
000004  013700  000566'                 MOV     IDCT.ADDR,R0                    ;                       5043
000010  016037  000002  000000G         MOV     2(R0),SA.REG
000016  013701  000000G                 MOV     L$LUN,R1                        ; *,U.SAVE              5044
000022  013700  000564'                 MOV     ICST.ADDR,R0                    ;                       5045
000026  016002  000006                  MOV     6(R0),R2
000032  000302                          SWAB    R2
000034  042702  177740                  BIC     #177740,R2
000040  010237  000000G                 MOV     R2,L$LUN
000044  104455                          TRAP    55                              :                       5046
000046  000016                          .WORD   16
000050  000000G                         .WORD   EGD.14
000052  000000G                         .WORD   EMS.14
000054  010137  000000G                 MOV     R1,L$LUN                        ; U.SAVE,*              5047
000060  013746  000600'                 MOV     ICTLR,-(SP)                     ;                       5048
```

```
000064  004737  000000G                        JSR     PC,DRV.CTLERR
000070  013716  000600'                        MOV     ICTLR,(SP)              ;                                       5049
000074  004737  000000G                        JSR     PC,GET.RETPKT
000100  010001                                 MOV     R0,R1                   ; *,INDEX
000102  002425                                 BLT     1$
000104  010116                                 MOV     R1,(SP)                 ; INDEX,*                               5053
000106  012746  000060                        MOV     #60,-(SP)
000112  004737  000000G                        JSR     PC,BL$MUL
000116  062700  000002G                        ADD     #RETPKT+2,R0
000122  112760  000003  000001                 MOVB    #3,1(R0)
000130  013702  000600'                        MOV     ICTLR,R2                ;                                       5055
000134  042702  177760                         BIC     #177760,R2
000140  112710  000060                        MOVB    #60,(R0)
000144  150210                                 BISB    R2,(R0)
000146  010116                                 MOV     R1,(SP)                 ; INDEX,*                               5056
000150  004737  000000G                        JSR     PC,IN.IODQ
000154  005726                                 TST     (SP)+                   ;                                       5051
000156  005726                         1$:     TST     (SP)+                   ;                                       5037
000160  000207                                 RTS     PC                      ;                                       5023

; Routine Size:  57 words,      Routine Base:  $CODE$ + 15432
; Maximum stack depth per invocation:  6 words
```

```
;         5061  ROUTINE POLL_CRING : NOVALUE =
;         5062
;         5063  !+
;         5064  !       THIS ROUTINE IS CALLED BY BOTH RCINT AND INT_PROC TO SCAN THE DEVICE'S
;         5065  !       COMMAND RING AND CHECK FOR ANY COMMAND SLOTS THAT HAVE BEEN "TAKEN" BY
;         5066  !       THE CONTROLLER. SUCH SLOTS HAVE BEEN RETURNED TO THE HOST, INDICATED BY
;         5067  !       A ZERO OWNERSHIP BIT. FOR EACH SLOT THAT HAS BEEN RETURNED TO THE HOST,
;         5068  !       THE CRING COUNT IS DECREMENTED, AND THE CR_POLL ADDRESS IS ADVANCED TO
;         5069  !       THE NEXT SLOT IN THE COMMAND RING.
;         5070  !
;         5071  !       IMPLICIT INPUTS:
;         5072  !               ICTLR - INTERRUPTING CONTROLLER NUMBER
;         5073  !               IDCT_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S DCT
;         5074  !               ICOM_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S COMM_AREA
;         5075  !-
;         5076
;         5077  BEGIN
;         5078
;         5079  WHILE ((.IDCT_ADDR [CRING_CNT] GTRU 0) AND                ! WHILE # OF COMMANDS IN CRING > 0 AND
;         5080     NOT (BIT_TST ((.IDCT_ADDR [CR_POLL] + 2), ED_OWN))) DO      ! CURRENT SLOT IS HOST-OWNED
;         5081       BEGIN
;         5082
;         5083       IDCT_ADDR [CRING_CNT] = .IDCT_ADDR [CRING_CNT] - 1; ! DECREMENT # CMDS IN CRING
;         5084       IDCT_ADDR [CR_POLL] = .IDCT_ADDR [CR_POLL] + 4;     ! ADVANCE TO NEXT SLOT TO POLL
;         5085       IF .IDCT_ADDR [CR_POLL] GTRA .IDCT_ADDR [CR_END]    ! IF BEYOND END OF RING
;         5086       THEN                                                ! THEN
;         5087           IDCT_ADDR [CR_POLL] = .IDCT_ADDR [CR_BEG];      ! SET POINTER TO TOP OF CRING
;         5088
;         5089       END;
;         5090
;         5091  ICOM_ADDR [CMD_INT] = 0;                                 ! CLEAR COMMAND INTERRUPT WORD IN RING HEADER
;         5092
;         5093  END;

                                        .SBTTL  POLL.CRING RC25 INTERRUPT SERVICE ROUTINES
000000  004137  000000G          POLL.CRING:
                                        JSR     R1,$SAVE2                       ;                       5061
000004  013701  000566'                 MOV     IDCT.ADDR,R1                    ;                       5079
000010  012702  000016                  MOV     #16,R2                          ;                       5084
000014  060102                          ADD     R1,R2
000016  105711                  1$:     TSTB    (R1)                            ;                       5079
000020  001422                          BEQ     2$
000022  016100  000016                  MOV     16(R1),R0                       ;                       5080
000026  016000  000002                  MOV     2(R0),R0
000032  042700  077777                  BIC     #77777,R0
000036  020027  100000                  CMP     R0,#-100000
000042  001411                          BEQ     2$
000044  105311                          DECB    (R1)                            ;                       5083
000046  062712  000004                  ADD     #4,(R2)                         ;                       5084
000052  021261  000012                  CMP     (R2),12(R1)                     ;                       5085
000056  101757                          BLOS    1$
000060  016112  000010                  MOV     10(R1),(R2)                     ;                       5087
000064  000754                          BR      1$                              ;                       5079
000066  013700  000560'          2$:     MOV     ICOM.ADDR,R0                    ;                       5091
000072  005060  000004                  CLR     4(R0)
000076  000207                          RTS     PC                              ;                       5061
; Routine Size:  32 words,      Routine Base:  $CODE$ + 15614
; Maximum stack depth per invocation:  4 words
```

```
;          5094  ROUTINE POLL_RRING : NOVALUE =
;          5095
;          5096  !+
;          5097  !       THIS ROUTINE IS CALLED BY BOTH RCINT AND INT_PROC TO SCAN THE DEVICE'S
;          5098  !       RESPONSE RING AND CHECK FOR ANY SLOTS WHICH HAVE BEEN RETURNED TO THE
;          5099  !       HOST (OWNERSHIP BIT = 0). FOR EACH SUCH SLOT, THE ASSOCIATED MESSAGE IS
;          5100  !       PROCESSED BASED ON ITS CONNECTION ID (DISK OR DUP) AND MESSAGE TYPE
;          5101  !       (SEQUENTIAL OR DATAGRAM). AFTER PROCESSING, THE MESSAGE ENVELOPE IS
;          5102  !       RE-INITIALIZED AND RETURNED TO THE CONTROLLER (OWNERSHIP BIT SET TO 1).
;          5103  !
;          5104  !       IMPLICIT INPUTS:
;          5105  !               IDCT_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S DCT
;          5106  !-
;          5107
;          5108  BEGIN
;          5109
;          5110  WHILE NOT (BIT_TST ((.IDCT_ADDR [RR_POLL] + 2), ED_OWN)) DO     ! WHILE 0 = 0
;          5111      BEGIN
;          5112
;          5113      IENV_ADDR = ..IDCT_ADDR [RR_POLL] - 8;                      ! ADDRESS OF RESPONSE ENVELOPE
;          5114      IF .IENV_ADDR [CONNID] EQLU CID_DISK                        ! IF MESSAGE IS FROM DISK MSCP
;          5115      THEN                                                        ! THEN
;          5116          SELECTONEU .IENV_ADDR [MSGTYP] OF                       ! PROCESS ON MESSAGE TYPE
;          5117              SET
;          5118
;          5119              [MT_SEQ]    : ENV_TO_RP ();                             ! SEQUENTIAL
;          5120              [MT_DG]     : DATAGM ();                                ! DATAGRAM
;          5121
;          5122              TES
;          5123      ELSE                                                        ! ELSE (MESSAGE NOT FROM DISK MSCP)
;          5124          BEGIN
;          5125
;          5126          IF .IENV_ADDR [CONNID] EQLU CID_DUP                     ! IF MESSAGE IS FROM DUP
;          5127          THEN                                                    ! THEN
;          5128              ENV_TO_RP ();                                       ! COPY ENV TO RETPKT
;          5129
;          5130          END;
;          5131
;          5132      IENV_ADDR [MSGLEN] = MSG_LEN * 2;                           ! RE-INIT MESSAGE LENGTH
;          5133      IDCT_ADDR [RR_POLL] = .IDCT_ADDR [RR_POLL] + 2;             ! ADVANCE TO HI ORDER WORD OF RING SLOT
;          5134      .IDCT_ADDR [RR_POLL] = .IENV_ADDR [ENV_HI];                 ! RETURN SLOT TO CONTROLLER
;          5135      IDCT_ADDR [RR_POLL] = .IDCT_ADDR [RR_POLL] + 2;             ! ADVANCE TO NEXT RRING SLOT
;          5136      IF .IDCT_ADDR [RR_POLL] GTRA .IDCT_ADDR [RR_END]            ! IF BEYOND END OF RING
;          5137      THEN                                                        ! THEN
;          5138          IDCT_ADDR [RR_POLL] = .IDCT_ADDR [RR_BEG];              ! CYCLE TO TOP OF RING
;          5139
;          5140      END;                                                    ! WHILE LOOP
;          5141
;          5142  ICOM_ADDR [RSP_INT] = 0;                                   ! CLEAR RESPONSE INTERRUPT WORD IN RING HEADER
;          5143
;          5144  END;


                                        .SBTTL  POLL.RRING RC25 INTERRUPT SERVICE ROUTINES
000000  004137  000000G          POLL.RRING:
                                        JSR     R1,$SAVE2                       ;                                       5094
000004  013702  000566'                 MOV     IDCT.ADDR,R2                    ;                                       5110
```

```
000010  062702  000014                          ADD     #14,R2
000014  011200                          1$:     MOV     (R2),R0
000016  016000  000002                          MOV     2(R0),R0
000022  042700  077777                          BIC     #77777,R0
000026  020027  100000                          CMP     R0,#-100000
000032  001470                                  BEQ     5$
000034  017237  000000  000562'                 MOV     @0(R2),IENV.ADDR        ;                       5113
000042  162737  000010  000562'                 SUB     #10,IENV.ADDR
000050  013700  000562'                         MOV     IENV.ADDR,R0            ;                       5114
000054  062700  000006                          ADD     #6,R0
000060  105760  000001                          TSTB    1(R0)
000064  001016                                  BNE     2$
000066  111000                                  MOVB    (R0),R0                 ;                       5116
000070  006200                                  ASR     R0
000072  006200                                  ASR     R0
000074  006200                                  ASR     R0
000076  006200                                  ASR     R0
000100  042700  177760                          BIC     #177760,R0
000104  001412                                  BEQ     3$                      ;                       5119
000106  020027  000001                          CMP     R0,#1                   ;                       5116
000112  001011                                  BNE     4$
000114  004737  000000V                         JSR     PC,DATAGM               ;                       5120
000120  000406                                  BR      4$                      ;                       5114
000122  126027  000001  000002        2$:     CMPB    1(R0),#2                ;                       5126
000130  001002                                  BNE     4$
000132  004737  000000V                 3$:     JSR     PC,ENV.TO.RP            ;                       5128
000136  013700  000562'                 4$:     MOV     IENV.ADDR,R0            ;                       5132
000142  012760  000074  000004                  MOV     #74,4(R0)
000150  013701  000566'                         MOV     IDCT.ADDR,R1            ;                       5133
000154  010102                                  MOV     R1,R2
000156  062702  000014                          ADD     #14,R2
000162  062712  000002                          ADD     #2,(R2)
000166  016072  000002  000000                  MOV     2(R0),@0(R2)            ;                       5134
000174  062712  000002                          ADD     #2,(R2)                 ;                       5135
000200  021261  000006                          CMP     (R2),6(R1)              ;                       5136
000204  101703                                  BLOS    1$
000206  016112  000004                          MOV     4(R1),(R2)              ;                       5138
000212  000700                                  BR      1$                      ;                       5110
000214  013700  000560'                 5$:     MOV     ICOM.ADDR,R0            ;                       5142
000220  005060  000006                          CLR     6(R0)
000224  000207                                  RTS     PC                      ;                       5094

; Routine Size:  75 words,      Routine Base:  $CODE$ + 15714
; Maximum stack depth per invocation:  4 words
```

```
;       5145  ROUTINE ENV_TO_RP : NOVALUE =
;       5146
;       5147  !+
;       5148  !       THIS ROUTINE IS CALLED BY POLL_RRING FOR (A) DISK MSCP RESPONSE
;       5149  !       MESSAGES WITH THE "SEQUENTIAL" MESSAGE TYPE, AND (B) DUP RESPONSE
;       5150  !       MESSAGES. ITS GENERAL PURPOSE IS TO COPY THE CONTENTS OF THE MESSAGE
;       5151  !       ENVELOPE INTO A RETURN PACKET SO THAT THE ENVELOPE CAN BE RETURNED TO
;       5152  !       THE CONTROLLER. IF THE RESPONSE MESSAGE HAS AN ASSOCIATED COMMAND IN
;       5153  !       THE CONTROLLER'S OUTSTANDING COMMAND LIST (OUTC_LIST), THEN TIMING FOR
;       5154  !       THAT COMMAND IS TERMINATED. IN ADDITION, IF THE COMMAND WAS AN I/O
;       5155  !       TRANSFER (READ, WRITE, OR ACCESS), THEN SOME FIELDS OF THE COMMAND
;       5156  !       ENVELOPE ARE COPIED INTO THE RETURN PACKET.
;       5157  !
;       5158  !       AFTER THE RETURN PACKET HAS BEEN LOADED, ITS INDEX IS LOADED INTO THE
;       5159  !       I/O DONE QUEUE (IODQ) FOR PROCESSING BY THE "PROGRAM" PORTION OF THE
;       5160  !       DIAGNOSTIC.
;       5161  !
;       5162  !       IMPLICIT INPUTS:
;       5163  !               ICTLR - INTERRUPTING CONTROLLER NUMBER
;       5164  !               IENV_ADDR - ADDRESS OF MSCP ENVELOPE CONTAINING RESPONSE
;       5165  !-
;       5166
;       5167  BEGIN
;       5168
;       5169  LOCAL
;       5170      M_INDEX : SIGNED WORD,
;       5171      R_INDEX : SIGNED WORD,
;       5172      STIDX : WORD,
;       5173      ENDIDX : WORD,
;       5174      R_ADDR : REF BLOCK [RP_LEN, WORD] FIELD (RP_FIELDS),
;       5175      TEMP : WORD;
;       5176
;       5177  M_INDEX = -1;                                   ! ASSUME NO ASSOCIATED COMMAND PACKET
;       5178  STIDX = .ICTLR * OUTC_CNT;                      ! START OF CONTROLLER'S OUTC_LIST AND _TIMR
;       5179  ENDIDX = .STIDX + OUTC_CNT - 1;                 ! END OF OUTC
;       5180  INCR COUNT FROM .STIDX TO .ENDIDX DO            ! SEARCH THROUGH EACH OUTC ENTRY
;       5181      BEGIN
;       5182
;       5183      IF (TEMP = .OUTC_LIST [.COUNT]) GEQ 0       ! IF ENTRY CONTAINS AN OUTSTANDING COMMAND INDEX
;       5184      THEN                                        ! THEN
;       5185          BEGIN
;       5186
;       5187          IF .MSCP_ENV [.TEMP, CRN_LO] EQLU .IENV_ADDR [CRN_LO]   ! IF THIS IS THE ASSOC CMD
;       5188          THEN                                                    ! THEN
;       5189              BEGIN
;       5190
;       5191              M_INDEX = .TEMP;
;       5192              OUTC_LIST [.COUNT] = -1;                            ! INIT ENTRY
;       5193              OUTC_TIMR [.COUNT] = 0;                             ! TURN OFF COMMAND TIMER
;       5194              EXITLOOP;
;       5195
;       5196              END;
;       5197
;       5198          END;
;       5199
;       5200      END;
;       5201
```

```
:          5202  IF (R_INDEX = GET_RETPKT (.ICTLR)) GEQ 0        ! IF RETPKT IS AVAILABLE
:          5203  THEN
:          5204      BEGIN
:          5205
:          5206      R_ADDR = RETPKT + (.R_INDEX * RP_LEN * 2);  ! START OF ALLOCATED RETPKT
:          5207      COPY_BLK (.IENV_ADDR + 4, .R_ADDR, RP_LEN); ! COPY RESPONSE ENVELOPE INTO RETURN PACKET
:          5208      R_ADDR [CTLR] = .ICTLR;                     ! LOAD CONTROLLER NUMBER INTO PACKET
:          5209      IF .M_INDEX GEQ 0                           ! IF ASSOC. CMD ENV WAS FOUND
:          5210      THEN                                        ! THEN
:          5211          BEGIN
:          5212
:          5213          IF ((.IENV_ADDR [OPCODE] EQLU (OP_RD + OP_END)) OR      ! IF END MESSAGE IS
:          5214              (.IENV_ADDR [OPCODE] EQLU (OP_WRT + OP_END)) OR     !      READ, WRITE, OR
:          5215              (.IENV_ADDR [OPCODE] EQLU (OP_ACC + OP_END)))       !      ACCESS
:          5216          THEN                                                    ! THEN
:          5217              BEGIN
:          5218
:          5219              R_ADDR [CMDMOD] = .MSCP_ENV [.M_INDEX, MODIFY];     ! COPY
:          5220              R_ADDR [CBCNT_LO] = .MSCP_ENV [.M_INDEX, BC_LO];    ! RELEVANT
:          5221              R_ADDR [CBCNT_HI] = .MSCP_ENV [.M_INDEX, BC_HI];    ! FIELDS
:          5222              R_ADDR [LBN_LO] = .MSCP_ENV [.M_INDEX, LBN_L];      ! FROM
:          5223              R_ADDR [LBN_HI] = .MSCP_ENV [.M_INDEX, LBN_H];      ! COMMAND
:          5224              R_ADDR [BUFF_0] = .MSCP_ENV [.M_INDEX, BUF_0];      ! PACKET
:          5225              R_ADDR [BUFF_1] = .MSCP_ENV [.M_INDEX, BUF_1];      ! TO RETPKT
:          5226
:          5227              END;                                        ! IF ENDCODE WAS READ, WRITE, OR ACCESS
:          5228
:          5229          END;                                            ! IF ASSOC CMD ENV WAS FOUND
:          5230
:          5231      IN_IODQ (.R_INDEX);                                 ! PUT RETPKT INDEX INTO IODQ
:          5232
:          5233      END;                                                ! IF RETPKT WAS ALLOCATED
:          5234
:          5235  IF .M_INDEX GEQ 0                                   ! IF ASSOC CMD ENV WAS FOUND
:          5236  THEN
:          5237     PUT_ENV (.M_INDEX);                              ! RETURN COMMAND ENVELOPE TO POOL
:          5238
:          5239  END;                                                ! ROUTINE DISK_RSP

                                        .SBTTL  ENV.TO.RP RC25 INTERRUPT SERVICE ROUTINES
000000  004137  000000G         ENV.TO.RP:
                                        JSR     R1,$SAVE5                       ;                                       5145
000004  012704  177777                  MOV     #-1,R4                          ; *,M.INDEX                             5177
000010  013700  000600'                MOV     ICTLR,R0                        ;                                       5178
000014  006300                          ASL     R0
000016  006300                          ASL     R0
000020  006300                          ASL     R0
000022  006300                          ASL     R0
000024  010005                          MOV     R0,R5                           ; STIDX,ENDIDX                          5179
000026  062705  000017                  ADD     #17,R5                          ; *,ENDIDX
000032  010002                          MOV     R0,R2                           ; STIDX,COUNT                           5180
000034  060402                          ADD     R4,R2                           ; *,COUNT
000036  000430                          BR      2$
000040  116203  000000G         1$:     MOVB    OUTC.LIST(R2),R3                ; *(COUNT),TEMP                         5183
000044  002425                          BLT     2$
000046  010346                          MOV     R3,-(SP)                        ; TEMP,*                                5187
```

```
000050  012746  000104                          MOV     #104,-(SP)
000054  004737  000000G                         JSR     PC,BL$MUL
000060  022626                                  CMP     (SP)+,(SP)+
000062  013701  000562'                         MOV     IENV.ADDR,R1
000066  026061  000010G 000010                  CMP     MSCP.ENV+10(R0),10(R1)
000074  001011                                  BNE     2$
000076  010304                                  MOV     R3,R4                   ; TEMP,M.INDEX                  5191
000100  112762  000377  000000G                MOVB    #377,OUTC.LIST(R2)      ; *,*(COUNT)                    5192
000106  010200                                  MOV     R2,R0                   ; COUNT,*                       5193
000110  006300                                  ASL     R0
000112  005060  000000G                         CLR     OUTC.TIMR(R0)
000116  000403                                  BR      3$                      ;                               5189
000120  005202                          2$:     INC     R2                      ; COUNT                         5180
000122  020205                                  CMP     R2,R5                   ; COUNT,ENDIDX
000124  003745                                  BLE     1$
000126  013746  000600'                  3$:     MOV     ICTLR,-(SP)             ;                               5202
000132  004737  000000G                         JSR     PC,GET.RETPKT
000136  010001                                  MOV     R0,R1                   ; *,R.INDEX
000140  005726                                  TST     (SP)+
000142  005701                                  TST     R1                      ; R.INDEX
000144  002513                                  BLT     6$
000146  010146                                  MOV     R1,-(SP)                ; R.INDEX,*                     5206
000150  012746  000060                         MOV     #60,-(SP)
000154  004737  000000G                         JSR     PC,BL$MUL
000160  062700  000000G                         ADD     #RETPKT,R0
000164  010002                                  MOV     R0,R2                   ; *,R.ADDR
000166  013716  000562'                         MOV     IENV.ADDR,(SP)          ;                               5207
000172  062716  000004                         ADD     #4,(SP)
000176  010246                                  MOV     R2,-(SP)                ; R.ADDR,*
000200  012746  000030                         MOV     #30,-(SP)
000204  004737  000000G                         JSR     PC,COPY.BLK
000210  013700  000600'                         MOV     ICTLR,R0                ;                               5208
000214  042700  177760                         BIC     #177760,R0
000220  142762  000017  000002                 BICB    #17,2(R2)               ; *,*(R.ADDR)
000226  150062  000002                         BISB    R0,2(R2)                ; *,*(R.ADDR)
000232  005704                                  TST     R4                      ; M.INDEX                       5209
000234  002452                                  BLT     5$
000236  013700  000562'                         MOV     IENV.ADDR,R0            ;                               5213
000242  116000  000020                         MOVB    20(R0),R0
000246  042700  177400                         BIC     #177400,R0
000252  020027  000241                         CMP     R0,#241
000256  001406                                  BEQ     4$
000260  020027  000242                         CMP     R0,#242                 ;                               5214
000264  001403                                  BEQ     4$
000266  020027  000220                         CMP     R0,#220                 ;                               5215
000272  001033                                  BNE     5$
000274  010416                          4$:     MOV     R4,(SP)                 ; M.INDEX,*                     5219
000276  012746  000104                         MOV     #104,-(SP)
000302  004737  000000G                         JSR     PC,BL$MUL
000306  016062  000022G 000012                  MOV     MSCP.ENV+22(R0),12(R2)  ; *,*(R.ADDR)
000314  016062  000024G 000044                  MOV     MSCP.ENV+24(R0),44(R2)  ; *,*(R.ADDR)                   5220
000322  016062  000026G 000046                  MOV     MSCP.ENV+26(R0),46(R2)  ; *,*(R.ADDR)                   5221
000330  016062  000044G 000050                  MOV     MSCP.ENV+44(R0),50(R2)  ; *,*(R.ADDR)                   5222
000336  016062  000046G 000052                  MOV     MSCP.ENV+46(R0),52(R2)  ; *,*(R.ADDR)                   5223
000344  016062  000030G 000024                  MOV     MSCP.ENV+30(R0),24(R2)  ; *,*(R.ADDR)                   5224
000352  016062  000032G 000026                  MOV     MSCP.ENV+32(R0),26(R2)  ; *,*(R.ADDR)                   5225
000360  005726                                  TST     (SP)+                   ;                               5217
```

```
000362  010116                          5$:     MOV     R1,(SP)                 ; R.INDEX,*                             5231
000364  004737  000000G                         JSR     PC,IN.IODQ
000370  062706  000010                          ADD     #10,SP                  ;                                       5204
000374  005704                          6$:     TST     R4                      ; M.INDEX                               5235
000376  002404                                  BLT     7$
000400  010446                                  MOV     R4,-(SP)                ; M.INDEX,*                             5237
000402  004737  000000G                         JSR     PC,PUT.ENV
000406  005726                                  TST     (SP)+
000410  000207                          7$:     RTS     PC                      ;                                       5145

; Routine Size:  133 words,     Routine Base:  $CODE$ + 16142
; Maximum stack depth per invocation:  12 words
```

```
;         5240  ROUTINE DATAGM : NOVALUE =
;         5241
;         5242  !+
;         5243  !       THIS ROUTINE HANDLES ALL DATAGRAM (ERROR LOG) MESSAGES RECEIVED FROM
;         5244  !       THE RC25. IF THE DATAGRAM REFERENCES A PARTICULAR UNIT (AND IS NOT
;         5245  !       CONTROLLER-RELATED), THEN ITS OCCURRENCE IS TALLIED FOR THAT UNIT. IF
;         5246  !       THE USER DID NOT SUPPRESS THE PRINTING OF ERROR LOG MESSAGES DURING THE
;         5247  !       SW DIALOG, THEN EACH RELEVANT FIELD IS IDENTIFIED AND PRINTED. IF THE
;         5248  !       DATAGRAM INDICATES AN ECC ERROR, THEN IT IS TALLIED FOR THE APPROPRIATE
;         5249  !       UNIT.
;         5250  !
;         5251  !       IMPLICIT INPUTS:
;         5252  !               IENV_ADDR - ADDRESS OF MSCP ENVELOPE CONTAINING ERROR LOG
;         5253  !                           MESSAGE
;         5254  !-
;         5255
;         5256  BEGIN
;         5257
;         5258  LITERAL
;         5259      EC1 = 8,                                  ! SUB-CODE VALUE FOR 1-SYMBOL ECC ERROR
;         5260      EC8 = 15,                                 ! SUB-CODE VALUE FOR 8-SYMBOL ECC ERROR
;         5261      ECO = 4;                                  ! SUB-CODE VALUE FOR ECC-FIELD-ONLY ERROR
;         5262
;         5263  BIND
;         5264      FMT_CE = UPLIT (%ASCIZ'%ACONTROLLER ERROR%N'),
;         5265      FMT_HMA = UPLIT (%ASCIZ'%AHOST MEMORY ACCESS ERROR%N'),
;         5266      FMT_SDE = UPLIT (%ASCIZ'%ASMALL DISK ERROR%N');
;         5267
;         5268  OWN
;         5269      FMT_TB : VECTOR [5] INITIAL (FMT_CE, FMT_HMA, 0, 0, FMT_SDE),
;         5270      STC_TB : VECTOR [12] INITIAL (STC_00, STC_01, STC_02, STC_03, STC_04,
;         5271                  STC_05, STC_06, STC_07, STC_08, STC_09, STC_10, STC_11);
;         5272
;         5273  LOCAL
;         5274      FMT : WORD,                                   ! FORMAT CODE
;         5275      EVC : WORD,                                   ! EVENT CODE
;         5276      SBC : WORD,                                   ! SUB-CODE
;         5277      PLAD : WORD,                                  ! PLATTER ADDRESS (MSCP UNIT NO.)
;         5278      UNIT : SIGNED WORD,                           ! DRS UNIT NUMBER
;         5279      TAD : REF BLOCK [TALLY_LEN, WORD] FIELD (T_FIELDS); !TALLY ADDRESS
;         5280
;         5281  FMT = .IENV_ADDR [FORMAT];                      ! GET FORMAT CODE
;         5282  EVC = .IENV_ADDR [EVENT];                       ! GET EVENT CODE
;         5283  SBC = .IENV_ADDR [SUBC];                        ! GET SUB-CODE
;         5284  IF .FMT EQLU FM_SDE                            ! IF SMALL DISK ERROR
;         5285  THEN                                            ! THEN
;         5286      BEGIN
;         5287
;         5288      PLAD = .IENV_ADDR [PL_ADDR];                          ! GET PLATTER ADDRESS (MSCP UNIT NO.)
;         5289      UNIT = -1;
;         5290      INCR OFFSET FROM (0 + OF_UN) TO (3 + OF_UN) DO        ! FOR EACH CST UNIT
;         5291          BEGIN
;         5292
;         5293          IF .PLAD EQLU .ICST_ADDR [.OFFSET, P_ADDR]        ! IF PLAT ADDR MATCHES CST ENTRY
;         5294          THEN                                              ! THEN
;         5295              BEGIN
;         5296
```

```
5297                UNIT = .ICST_ADDR [.OFFSET, P_UNIT];        ! GET DRS UNIT NUMBER
5298                EXITLOOP;                                   ! DONE SEARCH
5299
5300                END;
5301
5302            END;                                            ! PLATTER ADDRESS SEARCH LOOP
5303
5304        IF .UNIT GEQ 0                                      ! IF UNIT NUMBER WAS FOUND
5305        THEN                                                ! THEN
5306            BEGIN
5307
5308            TAD = TALLY + (.UNIT * TALLY_LEN * 2);          ! CALCULATE UNIT'S TALLY ADDRESS
5309            TAD [ER_LOG] = .TAD [ER_LOG] + 1;               ! INCR ERROR LOG COUNT
5310            IF .EVC EQLU ST_DAT                             ! IF EVENT CODE = DATA ERROR
5311            THEN                                            ! THEN
5312                BEGIN
5313
5314                IF .SBC EQLU ECO                                    ! IF ECC-FIELD-ONLY
5315                THEN                                                ! THEN
5316                    TAD [ECC_ONLY] = .TAD [ECC_ONLY] + 1            ! INCREMENT STAT
5317                ELSE                                                ! OTHERWISE
5318                    BEGIN
5319
5320                    IF ((.SBC GEQ EC1) AND (.SBC LEQ EC8))          ! IF SUB-CODE INDICATES ECC SYMBOL ERROR
5321                    THEN                                            ! THEN
5322                        BEGIN
5323
5324                        TAD = TAD [ECC_1] + ((.SBC - 8) * 2); ! CALCULATE EXACT ADDR OF ECC STAT
5325                        .TAD = ..TAD + 1;                     ! INCREMENT STATISTIC
5326
5327                        END;                                  ! IF SUB-CODE IS ECC ERROR
5328
5329                    END;                                      ! IF NOT ECC-FIELD-ONLY ERROR
5330
5331                END;                                    ! IF EVENT CODE = DATA ERROR
5332
5333            END;                                        ! IF UNIT NUMBER WAS FOUND
5334
5335        END;                                            ! IF FORMAT = SMALL DISK ERROR
5336
5337    IF MANUAL AND (NOT (BIT_TST (SWP_FLAGS, SWF_SEL)))      ! IF ATTENDED AND ERROR LOG PRINTING WAS NOT SUPPRESSED
5338    THEN                                                    ! THEN
5339        BEGIN
5340
5341        PRINTF (EX_EL);                                     ! "ERROR LOG MESSAGE RECEIVED:"
5342        PRINTF (EX_CRN, .IENV_ADDR [CRN_LO]);               !     CMD REF NUM: XXXXXX(O)"
5343        IF .FMT EQLU FM_SDE                                 ! IF FORMAT CODE = SMALL DISK ERROR
5344        THEN                                                ! THEN
5345            PRINTF (EX_PA, .PLAD);                          !     PLATTER: XXX."
5346        PRINTF (EX_FMT);                                    !     FORMAT: "
5347        SELECTONE .FMT OF
5348            SET
5349
5350            [FM_CNT, FM_BAD, FM_SDE] : PRINTF (.FMT_TB [.FMT]);     ! PRINT TEXT
5351            [OTHERWISE]              : PRINTF (EX_03, .FMT);        ! "XXX(O)"
5352
5353            TES;
```

```
;          5354
;          5355      PRINTF (EX_EVC);                                    ! ''    EVENT CODE: ''
;          5356      IF .EVC LEQU 11                                     ! IF EVENT CODE IS KNOWN
;          5357      THEN                                                ! THEN
;          5358          PRINTF (.STC_TB [.EVC])                         ! PRINT EVENT CODE TEXT
;          5359      ELSE                                                ! OTHERWISE
;          5360          PRINTF (EX_03, .EVC);                           ! ''XXX(O)''
;          5361      PRINTF (EX_SB, .SBC);                               ! ''    SUB-CODE: XXXX(O)''
;          5362      IF .FMT EQLU FM_BAD                                 ! IF HOST MEMORY ACCESS ERROR
;          5363      THEN                                                ! THEN
;          5364          PRINTF (EX_HMA, .IENV_ADDR [MA_LO], .IENV_ADDR [MA_HI]);        ! PRINT HOST ADDRESS
;          5365
;          5366      END;                                                ! IF PRINTING WAS NOT SUPPRESSED
;          5367
;          5368  END;                                                    ! ROUTINE DATAGM


002554                                          .PSECT  $GGG$,  RO
002554  000036'                         FMT.TB: .WORD   FMT.CE
002556  000064'                                 .WORD   FMT.HMA
002560  000000                                  .WORD   0
002562  000000                                  .WORD   0
002564  000122'                                 .WORD   FMT.SDE
002566  000000G                         STC.TB: .WORD   STC.00
002570  000000G                                 .WORD   STC.01
002572  000000G                                 .WORD   STC.02
002574  000000G                                 .WORD   STC.03
002576  000000G                                 .WORD   STC.04
002600  000000G                                 .WORD   STC.05
002602  000000G                                 .WORD   STC.06
002604  000000G                                 .WORD   STC.07
002606  000000G                                 .WORD   STC.08
002610  000000G                                 .WORD   STC.09
002612  000000G                                 .WORD   STC.10
002614  000000G                                 .WORD   STC.11


000036                                          .PSECT  $PLIT$,  RO ,  D
000036      045     101     103         P.AAC:  .ASCII  /%AC/
000041      117     116     124                 .ASCII  /ONT/
000044      122     117     114                 .ASCII  /ROL/
000047      114     105     122                 .ASCII  /LER/
000052      040     105     122                 .ASCII  / ER/
000055      122     117     122                 .ASCII  /ROR/
000060      045     116     000                 .ASCII  /%N/<00>
000063      000                                 .ASCII  <00>
000064      045     101     110         P.AAD:  .ASCII  /%AH/
000067      117     123     124                 .ASCII  /OST/
000072      040     115     105                 .ASCII  / ME/
000075      115     117     122                 .ASCII  /MOR/
000100      131     040     101                 .ASCII  /Y A/
000103      103     103     105                 .ASCII  /CCE/
000106      123     123     040                 .ASCII  /SS /
000111      105     122     122                 .ASCII  /ERR/
000114      117     122     045                 .ASCII  /OR%/
```

```
000117  116  000  000                 .ASCII  /N/<00><00>
000122  045  101  123       P.AAE:    .ASCII  /%AS/
000125  115  101  114                 .ASCII  /MAL/
000130  114  040  104                 .ASCII  /L D/
000133  111  123  113                 .ASCII  /ISK/
000136  040  105  122                 .ASCII  / ER/
000141  122  117  122                 .ASCII  /ROR/
000144  045  116  000                 .ASCII  /%N/<00>
000147  000                           .ASCII  <00>

        000036'                 FMT.CE=         P.AAC
        000064'                 FMT.HMA=        P.AAD
        000122'                 FMT.SDE=        P.AAE

                                .SBTTL  DATAGM RC25 INTERRUPT SERVICE ROUTINES
016554                          .PSECT  $CODE$,  RO

000000  004137  000000G  DATAGM: JSR     R1,$SAVE5                       ;                       5240
000004  024646                   CMP     -(SP),-(SP)
000006  013700  000562'           MOV     IENV.ADDR,R0                    ;                       5281
000012  005004                   CLR     R4                              ; FMT
000014  156004  000020           BISB    20(R0),R4                       ; *,FMT
000020  116003  000022           MOVB    22(R0),R3                       ; *,EVC                 5282
000024  042703  177740           BIC     #177740,R3                      ; *,EVC
000030  016005  000022           MOV     22(R0),R5                       ; *,SBC                 5283
000034  006205                   ASR     R5                              ; SBC
000036  006205                   ASR     R5                              ; SBC
000040  006205                   ASR     R5                              ; SBC
000042  006205                   ASR     R5                              ; SBC
000044  006205                   ASR     R5                              ; SBC
000046  042705  174000           BIC     #174000,R5                      ; *,SBC
000052  005066  000002           CLR     2(SP)                           ;                       5284
000056  020427  000004           CMP     R4,#4                           ; FMT,*
000062  001076                   BNE     6$
000064  005266  000002           INC     2(SP)
000070  016016  000014           MOV     14(R0),(SP)                     ; *,PLAD                5288
000074  012701  177777           MOV     #-1,R1                          ; *,UNIT                5289
000100  012700  000006           MOV     #6,R0                           ; *,OFFSET              5290
000104  010002           1$:     MOV     R0,R2                           ; OFFSET,*              5293
000106  063702  000564'           ADD     ICST.ADDR,R2
000112  111246                   MOVB    (R2),-(SP)
000114  105066  000001           CLRB    1(SP)
000120  026626  000002           CMP     2(SP),(SP)+                     ; PLAD,*
000124  001005                   BNE     2$
000126  011201                   MOV     (R2),R1                         ; *,UNIT                5297
000130  000301                   SWAB    R1                              ; UNIT
000132  042701  177740           BIC     #177740,R1                      ; *,UNIT
000136  000406                   BR      3$                              ;                       5295
000140  062700  000002   2$:     ADD     #2,R0                           ; *,OFFSET              5290
000144  020027  000014           CMP     R0,#14                          ; OFFSET,*
000150  003755                   BLE     1$
000152  005701                   TST     R1                              ; UNIT                  5304
000154  002441           3$:     BLT     6$
000156  010146                   MOV     R1,-(SP)                        ; UNIT,*                5308
000160  012746  000060           MOV     #60,-(SP)
```

```
000164  004737  000000G                        JSR     PC,BL$MUL
000170  062700  000000G                        ADD     #TALLY,R0
000174  010002                                 MOV     R0,R2          ; *,TAD
000176  005262  000032                         INC     32(R2)         ; *(TAD)                5309
000202  020327  000010                         CMP     R3,#10         ; EVC,*                 5310
000206  001023                                 BNE     5$
000210  020527  000004                         CMP     R5,#4          ; SBC,*                 5314
000214  001003                                 BNE     4$
000216  005262  000056                         INC     56(R2)         ; *(TAD)                5316
000222  000415                                 BR      5$             ;                       5314
000224  020527  000010              4$:        CMP     R5,#10         ; SBC,*                 5320
000230  002412                                 BLT     5$
000232  020527  000017                         CMP     R5,#17         ; SBC,*
000236  003007                                 BGT     5$
000240  010500                                 MOV     R5,R0          ; SBC,*                 5324
000242  006300                                 ASL     R0
000244  060200                                 ADD     R2,R0          ; TAD,*                 5322
000246  010002                                 MOV     R0,R2          ; *,TAD                 5324
000250  062702  000016                         ADD     #16,R2         ; *,TAD
000254  005212                                 INC     (R2)           ; TAD                   5325
000256  022626                       5$:       CMP     (SP)+,(SP)+    ;                       5306
000260  104450                       6$:       TRAP    50             ;                       5337
000262  103163                                 BHIS    15$
000264  132737  000001  000000G               BITB    #1,SWP.FLAGS
000272  001157                                 BNE     15$
000274  012746  000000G                        MOV     #EX.EL,-(SP)   ;                       5341
000300  012746  000001                         MOV     #1,-(SP)
000304  010600                                 MOV     SP,R0          ; SP,*
000306  104417                                 TRAP    17
000310  013700  000562'                        MOV     IENV.ADDR,R0   ;                       5342
000314  016016  000010                         MOV     10(R0),(SP)
000320  012746  000000G                        MOV     #EX.CRN,-(SP)
000324  012746  000002                         MOV     #2,-(SP)
000330  010600                                 MOV     SP,R0          ; SP,*
000332  104417                                 TRAP    17
000334  032766  000001  000012                BIT     #1,12(SP)      ;                       5343
000342  001411                                 BEQ     7$
000344  016616  000010                         MOV     10(SP),(SP)    ; PLAD,*                5345
000350  012746  000000G                        MOV     #EX.PA,-(SP)
000354  012746  000002                         MOV     #2,-(SP)
000360  010600                                 MOV     SP,R0          ; SP,*
000362  104417                                 TRAP    17
000364  022626                                 CMP     (SP)+,(SP)+
000366  012716  000000G              7$:        MOV     #EX.FMT,(SP)   ;                       5346
000372  012746  000001                         MOV     #1,-(SP)
000376  010600                                 MOV     SP,R0          ; SP,*
000400  104417                                 TRAP    17
000402  005704                                 TST     R4             ; FMT                   5347
000404  002403                                 BLT     8$
000406  020427  000001                         CMP     R4,#1          ; FMT,*
000412  003403                                 BLE     9$
000414  020427  000004              8$:        CMP     R4,#4          ; FMT,*
000420  001011                                 BNE     10$
000422  010400                       9$:       MOV     R4,R0          ; FMT,*                 5350
000424  006300                                 ASL     R0
000426  016016  002554'                        MOV     FMT.TB(R0),(SP)
000432  012746  000001                         MOV     #1,-(SP)
```

```
000436  010600                          MOV     SP,R0                   ; SP,*
000440  104417                          TRAP    17
000442  000410                          BR      11$                     ;                               5347
000444  010416                  10$:    MOV     R4,(SP)                 ; FMT,*                         5351
000446  012746  000000G                 MOV     #EX.03,-(SP)
000452  012746  000002                  MOV     #2,-(SP)
000456  010600                          MOV     SP,R0                   ; SP,*
000460  104417                          TRAP    17
000462  005726                          TST     (SP)+
000464  012716  000000G         11$:    MOV     #EX.EVC,(SP)            ;                               5355
000470  012746  000001                  MOV     #1,-(SP)
000474  010600                          MOV     SP,R0                   ; SP,*
000476  104417                          TRAP    17
000500  020327  000013                  CMP     R3,#13                  ; EVC,*                         5356
000504  101010                          BHI     12$
000506  006303                          ASL     R3                      ;                               5358
000510  016316  002566'                 MOV     STC.TB(R3),(SP)
000514  012746  000001                  MOV     #1,-(SP)
000520  010600                          MOV     SP,R0                   ; SP,*
000522  104417                          TRAP    17
000524  000410                          BR      13$                     ;                               5356
000526  010316                  12$:    MOV     R3,(SP)                 ; EVC,*                         5360
000530  012746  000000G                 MOV     #EX.03,-(SP)
000534  012746  000002                  MOV     #2,-(SP)
000540  010600                          MOV     SP,R0                   ; SP,*
000542  104417                          TRAP    17
000544  005726                          TST     (SP)+
000546  010516                  13$:    MOV     R5,(SP)                 ; SBC,*                         5361
000550  012746  000000G                 MOV     #EX.SB,-(SP)
000554  012746  000002                  MOV     #2,-(SP)
000560  010600                          MOV     SP,R0                   ; SP,*
000562  104417                          TRAP    17
000564  020427  000001                  CMP     R4,#1                   ; FMT,*                         5362
000570  001016                          BNE     14$
000572  013700  000562'                 MOV     IENV.ADDR,R0            ;                               5364
000576  016016  000042                  MOV     42(R0),(SP)
000602  016046  000040                  MOV     40(R0),-(SP)
000606  012746  000000G                 MOV     #EX.HMA,-(SP)
000612  012746  000003                  MOV     #3,-(SP)
000616  010600                          MOV     SP,R0                   ; SP,*
000620  104417                          TRAP    17
000622  062706  000006                  ADD     #6,SP
000626  062706  000024          14$:    ADD     #24,SP                  ;                               5339
000632  022626                  15$:    CMP     (SP)+,(SP)+             ;                               5240
000634  000207                          RTS     PC

; Routine Size:  207 words,     Routine Base:  $CODE$ + 16554
; Maximum stack depth per invocation:  23 words


;           5369
;           5370  END
;           5371
;           5372  ELUDOM
```

```
;                                       OTS external references
                                                .GLOBL  $SAVE5, $SAVE4, $SAVE3, $SAVE2
                                                .GLOBL  BL$ABS, BL$SHF, BL$DIV, BL$MOD
                                                .GLOBL  BL$MUL


;                                       PSECT SUMMARY
;
;       Psect Name                      Words     Attributes
;        $GGG$                            711       RO ,  I  ,  LCL,  REL,  CON
;        $CODE$                          3973       RO ,  I  ,  LCL,  REL,  CON
;        $PLIT$                            52       RO ,  D  ,  LCL,  REL,  CON


;                               LIBRARY STATISTICS
;                                       -------- Symbols --------     Blocks
;       File                            Total     Loaded   Percent      Read
;
;  _DUA2:[DOUCETTE.CZRCD]CZRCDL.L16;9     276        249        90        65


;                                               COMMAND QUALIFIERS

;       BLISS /PDP11 CZRCD3.SRC/LIST/EN:NOEIS

; Size:          3973 code + 763 data words
; Run Time:        02:09.4
; Elapsed Time:    02:51.6
; Memory Used:  328 pages
; Compilation Complete
```

```
;         0001  MODULE CZRCD4 (
;         0002          %TITLE 'CZRCDA0 RC25 DISK EXERCISER'
;         0003          IDENT = 'V01.0',
;         0004          ADDRESSING_MODE (ABSOLUTE)
;         0005          ) =
;         0006  BEGIN
;         0007
;         0043  %SBTTL 'LASTAD AND SETUP'
;         0044
;         0045  REQUIRE 'BLSMAC.REQ';                    ! DIAGNOSTIC SUPERVISOR LIBRARY
;         1534
;         1535  LASTAD
;         1536
;         1537  BGNSETUP (2);
;         1538
;       P 1539  BGNPTAB
;       P 1540      %O'172150', %O'154', 5, 0            ! IP, VECTOR, BR, PLAT ADDR
;         1541  ENDPTAB
;         1542
;       P 1543  BGNPTAB
;       P 1544      %O'172150', %O'154', 5, 1            ! IP, VECTOR, BR, PLAT ADDR
;         1545  ENDPTAB
;         1546
;         1547  ENDSETUP
                                        .TITLE  CZRCD4 CZRCDA0 RC25 DISK EXERCISER
                                        .IDENT  /V01.0/
                                        .ENABL  AMA
                                        .PSECT  $XYZ$,  RO
000000
000000  000034'                 BL$LAS::.WORD   T$FREE
000002  000000C                         .WORD   <<T$FREE-<BL$LAS+4>>/2>
000004  000024'                 P.AAA:  .WORD   L$LAST+20
000006  000004                         .WORD   4                              ; Plit count word
000010  172150                  P.AAB:  .WORD   -5630
000012  000154                          .WORD   154
000014  000005                          .WORD   5
000016  000000                          .WORD   0
000020  000000                  P.AAC:  .WORD   0
000022  000004                          .WORD   4                              ; Plit count word
000024  172150                  P.AAD:  .WORD   -5630
000026  000154                          .WORD   154
000030  000005                          .WORD   5
000032  000001                          .WORD   1
000034  000000                  T$FREE::.WORD   0

        000004'                 L$LAST==        BL$LAS+4
        000002                  T$PTHV==        2
        000004'                 $LAS3=          P.AAA
        000010'                 $REM3=          P.AAB
        000020'                 $$LAS1=         P.AAC
        000024'                 $REM2=          P.AAD

                                        .SBTTL  $END.LINK LASTAD AND SETUP
000000  000207                  $END.LINK::
                                        RTS     PC                             ;                                1533

; Routine Size:  1 word,        Routine Base:  $XYZ$ + 0036
; Maximum stack depth per invocation:  0 words
;         1548
```

```
;        1549  END
;        1550
;        1551  ELUDOM


;                                       PSECT SUMMARY
;
;       Psect Name                      Words     Attributes
;        $XYZ$                            16       RO ,  I  ,  LCL,  REL,  CON




;                                       COMMAND QUALIFIERS

;       BLISS /PDP11 CZRCD4.SRC/LIST/EN:NOEIS

; Size:          1 code + 15 data words
; Run Time:         00:06.9
; Elapsed Time:     00:11.3
; Memory Used:  94 pages
; Compilation Complete
```