

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192	193	194	195	196	197	198	199	200
201	202	203	204	205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	256	257	258	259	260	261	262	263	264	265	266	267	268	269	270	271	272	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	291	292	293	294	295	296	297	298	299	300
301	302	303	304	305	306	307	308	309	310	311	312	313	314	315	316	317	318	319	320	321	322	323	324	325	326	327	328	329	330	331	332	333	334	335	336	337	338	339	340	341	342	343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360	361	362	363	364	365	366	367	368	369	370	371	372	373	374	375	376	377	378	379	380	381	382	383	384	385	386	387	388	389	390	391	392	393	394	395	396	397	398	399	400
401	402	403	404	405	406	407	408	409	410	411	412	413	414	415	416	417	418	419	420	421	422	423	424	425	426	427	428	429	430	431	432	433	434	435	436	437	438	439	440	441	442	443	444	445	446	447	448	449	450	451	452	453	454	455	456	457	458	459	460	461	462	463	464	465	466	467	468	469	470	471	472	473	474	475	476	477	478	479	480	481	482	483	484	485	486	487	488	489	490	491	492	493	494	495	496	497	498	499	500

The image displays a large grid of 120 small, illegible data tables or charts arranged in 10 rows and 12 columns. Each cell contains a small table with multiple columns and rows of text and numbers, which are too small to read. The grid is set against a dark background.

b b w
A ?:
1
e(

SEQ 000

IDENTIFICATION

PRODUCT CODE: AC-T2668-MC
PRODUCT NAME: CZRCDB0 RC25 DISK EXERCISER
PRODUCT DATE: JUNE 17, 1985
MAINTAINER: SMALL STORAGE SYSTEMS DIAGNOSTICS
AUTHOR: JAMES S. DOUCETTE

COPYRIGHT (C) 1983, 1985
DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754

THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE TERMS. TITLE TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DEC.

THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.

DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.

TABLE OF CONTENTS

SEQ 0002

1.0	GENERAL INFORMATION
1.1	PROGRAM ABSTRACT
1.2	SYSTEM REQUIREMENTS
1.3	RELATED DOCUMENTS AND STANDARDS
1.4	DIAGNOSTIC HIERARCHY PREREQUISITES
1.5	ASSUMPTIONS
2.0	OPERATING INSTRUCTIONS
2.1	COMMANDS
2.2	SWITCHES
2.3	FLAGS
2.4	HARDWARE QUESTIONS
2.5	SOFTWARE QUESTIONS
2.6	EXTENDED P-TABLE DIALOGUE
2.7	QUICK STARTUP PROCEDURE
3.0	ERROR INFORMATION
3.1	TYPES OF ERROR MESSAGES
3.2	SPECIFIC ERROR MESSAGES
4.0	PERFORMANCE AND PROGRESS REPORTS
5.0	DEVICE INFORMATION TABLES
6.0	SUBTEST SUMMARIES
7.0	MAINTENANCE HISTORY
	APPENDIX A - DATA PATTERNS
	APPENDIX B - GLOSSARY

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 "DDDD".

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:DDDD	EXECUTE DDDDD PASSES (DDDD = 1 TO 64000)
/FLAGS:FLGS	SET SPECIFIED FLAGS. FLAGS ARE DESCRIBED IN SECTION 2.3.
/EOP:DDDD	REPORT END-OF-PASS MESSAGE AFTER EVERY DDDDD PASSES ONLY. (DDDD = 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 (0) 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 (0) 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 SPECIFY A VECTOR WHICH IS USED FOR ANOTHER PERIPHERAL DEVICE.

BR LEVEL (0) 5 ?

ANSWER WITH THE BUS REQUEST INTERRUPT LEVEL USED BY THE ABOVE RC25. LEVELS 4 THOROUGH 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) (0) 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 (0) ? 2<CR>

UNIT 0

IP ADDRESS (0) 172150 ? <CR>

VECTOR (0) 154 ? <CR>

BR LEVEL (0) 5 ? <CR>

PLATTER ADDRESS ... (0) 0 ? 252<CR>

ALLOW WRITES ... (L) ? Y<CR>

** WARNING - ... CONFIRM (L) ? N<CR> ! USER REVERSES ANSWER

UNIT 1

IP ADDRESS (0) 172150 ? <CR>

VECTOR (0) 154 ? <CR>

BR LEVEL (0) 5 ? <CR>

PLATTER ADDRESS ... (0) 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 SOFTWARE 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 (0) ? 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 (0) ? 160000<CR>

SUB-DEVICE # (0) ? 4<CR>

Q-FACTOR (0) 0 ? <CR>

UNIT 6

CSR ADDRESS (0) ? 160000<CR>

SUB-DEVICE # (0) ? 5<CR>

Q-FACTOR (0) 0 ? <CR>

UNIT 7

CSR ADDRESS (0) ? 160000<CR>

SUB-DEVICE # (0) ? 6<CR>

Q-FACTOR (0) 0 ? 1<CR>

UNIT 8

CSR ADDRESS (0) 160000<CR>

SUB-DEVICE # (0) ? 7<CR>

Q-FACTOR (0) 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 (0) ? 160000<CR>

SUB-DEVICE # (0) ? 0,1<CR>

Q-FACTOR (0) 0 ? 1,0<CR>

UNIT 3

CSR ADDRESS (0) ? 160000<CR>

SUB-DEVICE # (0) ? 2-5<CR>

Q-FACTOR (0) 0 ? 0<CR>

UNIT 7

CSR ADDRESS (0) ? 160000<CR>

SUB-DEVICE # (0) ? 6,7<CR>

Q-FACTOR (0) 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 (0) ? 160000<CR>

SUB-DEVICE # (0) ? 0-7<CR>

Q-FACTOR (0) 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 CTRL 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 OCCASIONAL 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.

5.0 DEVICE INFORMATION TABLES

SEQ 0021

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

```

15
+-----+-----+
}                IP ADDRESS                }
+-----+-----+
}                VECTOR ADDRESS            }
+-----+-----+
}                BR LEVEL                  }
+-----+-----+
}PR)                ) PLATTER ADDRESS    }
+-----+-----+

```

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:

```

15
+-----+-----+
}                IP ADDRESS                }
+-----+-----+
}CS)                ) VECTOR ADDRESS      }
+-----+-----+
}  NUMBER OF UNITS  ) BR LEVEL            }
+-----+-----+
}PR)UP)US) UNIT NUMBER ) PLATTER ADDRESS }
+-----+-----+
}PR)UP)US) UNIT NUMBER ) PLATTER ADDRESS }
+-----+-----+
}PR)UP)US) UNIT NUMBER ) PLATTER ADDRESS }
+-----+-----+
}PR)UP)US) UNIT NUMBER ) PLATTER ADDRESS }
+-----+-----+

```

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 VERIFY 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 EXERCISE 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).

7.0 MAINTENANCE HISTORY

DATE:	AUTHOR:	PRODUCT NAME:
13-JUL-83	JAMES S. DOUCETTE	CZRCDA0.BIN
17-JUN-85	GARY Y. CHIN	CZRCDB0.BIN

NOTE:

CZRCDB0.BIN was modified to run on the PDP-11/84. The four steps of Initialization routines now measure time off of the real-time clock.

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

LBN	LBN	LBN
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
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		
-----	-----	--	--	--	-----	--	--

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 (D6N), 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.

)*

Partition name : DUMMY
Identification : V02.0
Task UIC : [200,250]
Task attributes: -HD
Total address windows: 1.
Task image size : 14720. words
Task address limits: 002000 073307
R-W disk blk limits: 000002 000073 000072 00058.

*** Root segment: CZRCD1

R/W mem limits: 002000 073307 071310 29384.
Disk blk limits: 000002 000073 000072 00058.

Memory allocation synopsis:

Section			Title	Ident	File
-----			-----	-----	-----
. BLK.:(RW,I,LCL,REL,CON)	002000	000000	00000.		
\$CODE\$:(RC,I,LCL,REL,CON)	002000	040426	16662.		
	002000	006550	03432.	CZRCD1	V02.0 CZRCD1.OBJ;4
	010550	011544	04964.	CZRCD2	V02.0 CZRCD2.OBJ;3
	022314	017432	07962.	CZRCD3	V02.0 CZRCD3.OBJ;11
	041746	000034	00028.	B16ABS	V3.0 AZLIB.OLB;1
	042002	000316	00206.	B16MUL	V3.0 AZLIB.OLB;1
	042320	000106	00070.	B16SAV	V3.0 AZLIB.OLB;1
\$FFF\$:(RO,I,LCL,REL,CON)	042426	023714	10188.		
	042426	023714	10188.	CZRCD1	V02.0 CZRCD1.OBJ;4
\$GGG\$:(RO,I,LCL,REL,CON)	066342	002616	01422.		
	066342	002616	01422.	CZRCD3	V02.0 CZRCD3.OBJ;11
\$PLIT\$:(RO,D,LCL,REL,CON)	071160	002066	01078.		
	071160	001716	00974.	CZRCD2	V02.0 CZRCD2.OBJ;3
	073076	000150	00104.	CZRCD3	V02.0 CZRCD3.OBJ;11

\$XYZ\$:(RO,I,LCL,REL,CON)

073246 000040 00032.
073246 000040 00032. CZRCD4 V02.0 CZRCD4.OBJ;3

SEQ 0032

Global symbols:

ADR	000020	BIT10	002000	BIT9	001000	CER.01	004140-R	CST	042756-R	DUM.IE	004450-R	EBD.18	006160-R
BIT0	000001	BIT11	004000	BL\$ABS	041770-R	CER.02	004224-R	CST.AD	043046-R	DUM.UC	004376-R	EBD.19	006224-R
BIT00	000001	BIT12	010000	BL\$DIV	042226-R	CER.03	004270-R	CTLR.C	066246-R	DUM.UE	004526-R	EBD.20	006264-R
BIT01	000002	BIT13	020000	BL\$LAS	073246-R	CLK.CS	066310-R	CUOFF	066244-R	DUM.00	004344-R	EBD.21	006350-R
BIT02	000004	BIT14	040000	BL\$MOD	042240-R	CLK.HE	066306-R	DCT	043050-R	DUR	066250-R	EBD.22	006436-R
BIT03	000010	BIT15	100000	BL\$MUL	042002-R	CLK.IN	014756-R	DCT.AD	043160-R	D\$PCNT	002122-R	EBH.31	006556-R
BIT04	000020	BIT2	000004	BL\$SGN	041746-R	CLK.TY	066304-R	DFPTBL	010450-R	EBD.10	005400-R	EBH.32	006602-R
BIT05	000040	BIT3	000010	BL\$SHF	042252-R	CLK.VE	066312-R	DMC.AD	043506-R	EBD.12	005444-R	EBH.34	006650-R
BIT06	000100	BIT4	000020	BOE	000400	COPY.B	015060-R	DM.COM	043166-R	EBD.13	005526-R	EBH.35	006674-R
BIT07	000200	BIT5	000040	BUFF.D	065466-R	CPLAT	066242-R	DROP.C	016262-R	EBD.14	005604-R	EBH.36	006724-R
BIT08	000400	BIT6	000100	BUFF.O	066066-R	CPT	042736-R	DRV.CT	016366-R	EBD.15	005676-R	EBH.37	006750-R
BIT09	001000	BIT7	000200	BUFF.S	066300-R	CRLF	010442-R	DUM.CE	004420-R	EBD.16	005766-R	EBH.38	007002-R
BIT1	000002	BIT8	000400	CCTLR	066240-R	CRN	066340-R	DUM.HE	004470-R	EBD.17	006062-R	EBH.39	007022-R

EBH.40	007060-R	EOP.FL	066234-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	014254-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	016074-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	015536-R	SWP.UC	010476-R
EF.NEW	000035	EVL	000004	GP\$3	010646-R	L\$DEVP	002060-R	L\$SPTP	002024-R	PUT.EN	015522-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	016010-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	015704-R	SWQ1	002414-R
EF.STA	000040	EX.BD	010070-R	GP\$6	010702-R	L\$DTP	002040-R	L\$SWLE	010462-R	QIO	066270-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	037326-R	SWQ11	003100-R
EGD.11	004714-R	EX.CMD	007474-R	GP\$8	010730-R	L\$DU	014646-R	L\$TIML	002014-R	RCINT1	037344-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	037364-R	SWQ13	003176-R
EGD.13	004766-R	EX.CRN	007304-R	HARD.E	016502-R	L\$DVTY	010550-R	MEM.MG	066235-R	RCINT3	037404-R	SWQ14	003274-R
EGD.14	005014-R	EX.DSC	007400-R	HOE	100000	L\$EF	002052-R	MEM.SI	066274-R	RC25.A	043162-R	SWQ15	003352-R
EGD.15	005052-R	EX.EL	010150-R	HOURS	066314-R	L\$ENVI	002044-R	MINUTE	066316-R	RETPKT	062422-R	SWQ2	002452-R
EGD.16	005104-R	EX.ESP	007546-R	HWPT.B	010454-R	L\$ERRT	002126-R	MSCP.E	045462-R	RPS.X1	043550-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	043552-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	065464-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	065462-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	043510-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	065422-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	066334-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	066326-R	TALLY	044060-R
EGS.01	004556-R	EX.RD	007616-R	HWQ5	002230-R	L\$HPTP	002022-R	MSG.08	004044-R	SECOND	066320-R	TICKS	066322-R
EGS.02	004576-R	EX.SA	007256-R	HWQ6	002314-R	L\$HRDL	010624-R	NEX	066336-R	SEND	017126-R	T\$FREE	073302-R
EMS.01	020706-R	EX.SB	007440-R	IBE	010000	L\$HW	010450-R	NEX.TR	014746-R	SET.CP	015120-R	T\$PTHV	000002
EMS.10	020750-R	EX.SC	007342-R	IDU	000040	L\$HWLE	010446-R	NULL	003464-R	SET.UP	015314-R	T.ADDR	045460-R
EMS.12	021016-R	EX.SCC	007514-R	IER	020000	L\$ICP	002104-R	NUM.BU	066302-R	SFPTBL	010464-R	T.FLAG	066233-R
EMS.13	021060-R	EX.SND	007602-R	IIP.FL	066236-R	L\$INIT	014032-R	OCL.X1	044054-R	STC.00	006542-R	T1	022420-R
EMS.14	021132-R	EX.WRT	007626-R	IN.IOD	016174-R	L\$LADP	002026-R	OCL.X2	044056-R	STC.01	006556-R	UAM	000200
EMS.15	021200-R	FREE.M	066276-R	IODQ	066166-R	L\$LAST	073252-R	OF.RC	066332-R	STC.02	006602-R	UPD.IO	016572-R
EMS.16	021246-R	GET.EN	015412-R	IODQ.I	066226-R	L\$LOAD	002100-R	OUTC.L	043554-R	STC.03	006626-R	WAIT	017372-R
EMS.17	021320-R	GET.IO	015720-R	IODQ.O	066230-R	L\$LUN	002074-R	OUTC.T	043654-R	STC.04	006650-R	XFR.CH	016734-R

EMS.18	021362-R	GET.RE	015576-R	IRC25.	043164-R	L\$MREV	002050-R	OUT.IO	016136-R	STC.05	006674-R	\$END.L	073304-R
EMS.19	021434-R	GP\$DIS	011022-R	ISR	000100	L\$NAME	002000-R	OVF.CH	016660-R	STC.06	006724-R	\$SAVE2	042320-R
EMS.20	021502-R	GP\$1	010626-R	IXE	004000	L\$NDHR	010712-R	PATCH	042426-R	STC.07	006750-R	\$SAVE3	042334-R
EMS.21	021544-R	GP\$10	010750-R	LOE	040000	L\$NDHW	010460-R	PLATT	010416-R	STC.08	007002-R	\$SAVE4	042352-R
EMS.22	021632-R	GP\$11	010760-R	LOT	000010	L\$NDSF	011134-R	PNT	001000	STC.09	007022-R	\$SAVE5	042372-R
EMS.30	021674-R	GP\$12	010766-R	L\$ACP	002110-R	L\$NDSW	010540-R	PRI	002000	STC.10	007060-R		
EMS.42	022044-R	GP\$13	011000-R	L\$APT	002036-R	L\$PRIO	002042-R	PRI00	000000	STC.11	007106-R		
EMS.43	022132-R	GP\$14	011014-R	L\$AU	014736-R	L\$PROT	010542-R	PRI01	000040	STEP	066330-R		
ENTRY.	066232-R	GP\$15	011026-R	L\$AUT	002070-R	L\$PRT	002112-R	PRI02	000100	ST.COD	066324-R		
ENV.US	062262-R	GP\$16	011034-R	L\$AUTO	014044-R	L\$REPP	002062-R	PRI03	000140	SWM1	003370-R		

*** Task builder statistics:

Total work file references: 73188.

Work file reads: 0.

Work file writes: 0.

Size of core pool: 4816. words (18. pages)

Size of work file: 3840. words (15. pages)

Elapsed time:00:00:57

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	041770-R	# B16ABS CZRCD3
BL\$DIV	042226-R	# B16MUL CZRCD3
BL\$LAS	073246-R	# CZRCD4
BL\$MOD	042240-R	# B16MUL CZRCD3
BL\$MUL	042002-R	# B16MUL CZRCD2 CZRCD3

BL\$SGN	041746-R	# B16ABS		
BL\$SHF	042252-R	# B16MUL	CZLCD3	
BOE	000400	# CZLCD1	# CZLCD2	# CZLCD3
BUFF.D	065466-R	# CZLCD1	CZLCD2	CZLCD3
BUFF.O	066066-R	# CZLCD1	CZLCD2	CZLCD3
BUFF.S	066300-R	# CZLCD1	CZLCD2	CZLCD3
CCTLR	066240-R	# CZLCD1	CZLCD2	CZLCD3
CER.01	004140-R	# CZLCD1	CZLCD2	
CER.02	004224-R	# CZLCD1	CZLCD2	
CER.03	004270-R	# CZLCD1	CZLCD2	
CLK.CS	066310-R	# CZLCD1	CZLCD2	CZLCD3
CLK.HE	066306-R	# CZLCD1	CZLCD2	CZLCD3
CLK.IN	014756-R	# CZLCD2		
CLK.TY	066304-R	# CZLCD1	CZLCD2	CZLCD3
CLK.VE	066312-R	# CZLCD1	CZLCD2	CZLCD3
COPY.B	015060-R	# CZLCD2	CZLCD3	
CPLAT	066242-R	# CZLCD1	CZLCD2	CZLCD3
CPT	042736-R	# CZLCD1	CZLCD2	CZLCD3
CRLF	010442-R	# CZLCD1	CZLCD2	
CRN	066340-R	# CZLCD1	CZLCD2	CZLCD3

GLOBAL CROSS REFERENCE CREF V01

SYMBOL	VALUE	REFERENCES...
CST	042756-R	# CZRCD1 CZRCD2 CZRCD3
CST.AD	043046-R	# CZRCD1 CZRCD2 CZRCD3
CTLR.C	066246-R	# CZRCD1 CZRCD2 CZRCD3
CUOFF	066244-R	# CZRCD1 CZRCD2 CZRCD3
DCT	043050-R	# CZRCD1 CZRCD2 CZRCD3
DCT.AD	043160-R	# CZRCD1 CZRCD2 CZRCD3
DFPTBL	010450-R	# CZRCD1
DMC.AD	043506-R	# CZRCD1 CZRCD2 CZRCD3
DM.COM	043166-R	# CZRCD1 CZRCD2 CZRCD3
DROP.C	016262-R	# CZRCD2 CZRCD3
DRV.CT	016366-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.OO	004344-R	# CZRCD1 CZRCD2
DUR	066250-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	

GLOBAL CROSS REFERENCE

CREF V01

SYMBOL	VALUE	REFERENCES...
EGD.13	004766-R	⊕ CZLCD1 CZLCD3
EGD.14	005014-R	⊕ CZLCD1 CZLCD3
EGD.15	005052-R	⊕ CZLCD1 CZLCD3
EGD.16	005104-R	⊕ CZLCD1 CZLCD3
EGD.17	005122-R	⊕ CZLCD1 CZLCD3
EGD.18	005152-R	⊕ CZLCD1 CZLCD3
EGD.19	005170-R	⊕ CZLCD1 CZLCD3
EGD.20	005210-R	⊕ CZLCD1 CZLCD3
EGD.21	005234-R	⊕ CZLCD1 CZLCD3
EGD.22	005260-R	⊕ CZLCD1 CZLCD3
EGH.30	005306-R	⊕ CZLCD1 CZLCD3
EGS.01	004556-R	⊕ CZLCD1 CZLCD2
EGS.02	004576-R	⊕ CZLCD1 CZLCD2
EMS.01	020706-R	⊕ CZLCD2
EMS.10	020750-R	⊕ CZLCD2 CZLCD3
EMS.12	021016-R	⊕ CZLCD2 CZLCD3
EMS.13	021060-R	⊕ CZLCD2 CZLCD3
EMS.14	021132-R	⊕ CZLCD2 CZLCD3
EMS.15	021200-R	⊕ CZLCD2 CZLCD3
EMS.16	021246-R	⊕ CZLCD2 CZLCD3
EMS.17	021320-R	⊕ CZLCD2 CZLCD3
EMS.18	021362-R	⊕ CZLCD2 CZLCD3
EMS.19	021434-R	⊕ CZLCD2 CZLCD3
EMS.20	021502-R	⊕ CZLCD2 CZLCD3
EMS.21	021544-R	⊕ CZLCD2 CZLCD3
EMS.22	021632-R	⊕ CZLCD2 CZLCD3
EMS.30	021674-R	⊕ CZLCD2 CZLCD3
EMS.42	022044-R	⊕ CZLCD2 CZLCD3
EMS.43	022132-R	⊕ CZLCD2 CZLCD3
ENTRY.	066232-R	⊕ CZLCD1 CZLCD2 CZLCD3
ENV.US	062262-R	⊕ CZLCD1 CZLCD2 CZLCD3
EOP.FL	066234-R	⊕ CZLCD1 CZLCD2 CZLCD3
ERRBLK	002134-R	⊕ CZLCD1
ERRMSG	002132-R	⊕ CZLCD1

ERRNBR	002130-R	◆ CZRCD1		
ERRYP	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	

GLOBAL CROSS REFERENCE

CREF V01

SYMBOL	VALUE	REFERENCES...
EX.ONL	007534-R	* CZLCD1 CZLCD2
EX.O3	010356-R	* CZLCD1 CZLCD2 CZLCD3
EX.PA	010212-R	* CZLCD1 CZLCD3
EX.RD	007616-R	* CZLCD1 CZLCD2
EX.SA	007256-R	* CZLCD1 CZLCD2
EX.SB	007440-R	* CZLCD1 CZLCD2 CZLCD3
EX.SC	007342-R	* CZLCD1 CZLCD2
EX.SCC	007514-R	* CZLCD1 CZLCD2
EX.SND	007602-R	* CZLCD1 CZLCD2
EX.WRT	007626-R	* CZLCD1 CZLCD2
FREE.M	066276-R	* CZLCD1 CZLCD2 CZLCD3
GET.EN	015412-R	* CZLCD2 CZLCD3
GET.IO	015720-R	* CZLCD2 CZLCD3
GET.RE	015576-R	* CZLCD2 CZLCD3
GP\$DIS	011022-R	* CZLCD2
GP\$1	010626-R	* CZLCD2
GP\$10	010750-R	* CZLCD2
GP\$11	010760-R	* CZLCD2
GP\$12	010766-R	* CZLCD2
GP\$13	011000-R	* CZLCD2
GP\$14	011014-R	* CZLCD2
GP\$15	011026-R	* CZLCD2
GP\$16	011034-R	* CZLCD2
GP\$17	011044-R	* CZLCD2
GP\$18	011054-R	* CZLCD2
GP\$19	011064-R	* CZLCD2
GP\$2	010636-R	* CZLCD2
GP\$20	011102-R	* CZLCD2
GP\$21	011114-R	* CZLCD2
GP\$3	010646-R	* CZLCD2
GP\$4	010660-R	* CZLCD2
GP\$5	010672-R	* CZLCD2
GP\$6	010702-R	* CZLCD2
GP\$7	010716-R	* CZLCD2

GP#8	010730-R	⊛ CZRCD2		
GP#9	010742-R	⊛ CZRCD2		
HARD.E	016502-R	⊛ CZRCD2	CZRCD3	
HOE	100000	⊛ CZRCD1	⊛ CZRCD2	⊛ CZRCD3
HOURS	056314-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

GLOBAL CROSS REFERENCE

CREF V01

SYMBOL	VALUE	REFERENCES...
IIP.FL	066236-R	* CZRCD1 CZRCD2 CZRCD3
IN.IOD	016174-R	* CZRCD2 CZRCD3
IODQ	066166-R	* CZRCD1 CZRCD2 CZRCD3
IODQ.I	066226-R	* CZRCD1 CZRCD2 CZRCD3
IODQ.0	066230-R	* CZRCD1 CZRCD2 CZRCD3
IRC25.	043164-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	014736-R	CZRCD1 * CZRCD2
L\$AUT	002070-R	* CZRCD1
L\$AUTO	014044-R	CZRCD1 * CZRCD2
L\$CCP	002106-R	* CZRCD1
L\$CLEA	014254-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	014646-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\$EXPS	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	014032-R	CZRCD1	✦ CZRCD2	
L\$LADP	002026-R	✦ CZRCD1		
L\$LAST	073252-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		

GLOBAL CROSS REFERENCE

CREF V01

SYMBOL	VALUE	REFERENCES...
L\$NDHW	010460-R	# CZLCD1
L\$NDSF	011134-R	# CZLCD2
L\$NDSW	010540-R	# CZLCD1
L\$PRIO	002042-R	# CZLCD1
L\$PROT	010542-R	# CZLCD1
L\$PRT	002112-R	# CZLCD1
L\$REPP	002062-R	# CZLCD1
L\$REV	002010-R	# CZLCD1
L\$RPT	012236-R	CZLCD1 # CZLCD2
L\$SFTL	010714-R	# CZLCD2
L\$SOFT	010716-R	CZLCD1 # CZLCD2
L\$SPC	002056-R	# CZLCD1
L\$SPCP	002020-R	# CZLCD1
L\$SPTP	002024-R	# CZLCD1
L\$STA	002030-R	# CZLCD1
L\$SW	010464-R	# CZLCD1
L\$SWLE	010462-R	# CZLCD1
L\$TEST	002114-R	# CZLCD1
L\$TIML	002014-R	# CZLCD1
L\$UNIT	002012-R	# CZLCD1 CZLCD2 CZLCD3
MEM.MG	066235-R	# CZLCD1 CZLCD2 CZLCD3
MEM.SI	066274-R	# CZLCD1 CZLCD2 CZLCD3
MINUTE	066316-R	# CZLCD1 CZLCD2 CZLCD3
MSCP.E	045462-R	# CZLCD1 CZLCD2 CZLCD3
MSG.01	003466-R	# CZLCD1 CZLCD2
MSG.02	003520-R	# CZLCD1 CZLCD3
MSG.03	003616-R	# CZLCD1 CZLCD3
MSG.04	003636-R	# CZLCD1 CZLCD3
MSG.05	003670-R	# CZLCD1 CZLCD3
MSG.06	003730-R	# CZLCD1 CZLCD3
MSG.07	003772-R	# CZLCD1 CZLCD2
MSG.08	004044-R	# CZLCD1 CZLCD3
NEX	066336-R	# CZLCD1 CZLCD2 CZLCD3
NEX.TR	014746-R	# CZLCD2 CZLCD3
NULL	003464-R	# CZLCD1
NUM.BU	066302-R	# CZLCD1 CZLCD2 CZLCD3

OCL.X1	044054-R	# CZRCD1	CZRCD2	CZRCD3
OCL.X2	044056-R	# CZRCD1	CZRCD2	CZRCD3
OF.RC	066332-R	# CZRCD1	CZRCD2	CZRCD3
OUTC.L	043554-R	# CZRCD1	CZRCD2	CZRCD3
OUTC.T	043654-R	# CZRCD1	CZRCD2	CZRCD3
OUT.IO	016136-R	# CZRCD2	CZRCD3	
OVF.CH	016660-R	# CZRCD2	CZRCD3	
PATCH	042426-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

GLOBAL CROSS REFERENCE CREF V01

SYMBOL	VALUE	REFERENCES...
PRI05	000240	# CZLCD1 # CZLCD2 # CZLCD3
PRI06	000300	# CZLCD1 # CZLCD2 # CZLCD3
PRI07	000340	# CZLCD1 # CZLCD2 # CZLCD3
PUTA.B	016074-R	# CZLCD2 CZLCD3
PUTA.E	015536-R	# CZLCD2
PUT.EN	015522-R	# CZLCD2 CZLCD3
PUT.IO	016010-R	# CZLCD2 CZLCD3
PUT.RE	015704-R	# CZLCD2 CZLCD3
QIO	066270-R	# CZLCD1 CZLCD2 CZLCD3
RCINT0	037326-R	# CZLCD3
RCINT1	037344-R	# CZLCD3
RCINT2	037364-R	# CZLCD3
RCINT3	037404-R	# CZLCD3
RC25.A	043162-R	# CZLCD1 CZLCD2 CZLCD3
RETPKT	062422-R	# CZLCD1 CZLCD2 CZLCD3
RPS.X1	043550-R	# CZLCD1 CZLCD2 CZLCD3
RPS.X2	043552-R	# CZLCD1 CZLCD2 CZLCD3
RP.ADD	065464-R	# CZLCD1 CZLCD2 CZLCD3
RP.IND	065462-R	# CZLCD1 CZLCD2 CZLCD3
RP.SAV	043510-R	# CZLCD1 CZLCD2 CZLCD3
RP.USE	065422-R	# CZLCD1 CZLCD2 CZLCD3
SA.REG	066334-R	# CZLCD1 CZLCD2 CZLCD3
SB.COD	066326-R	# CZLCD1 CZLCD2 CZLCD3
SECOND	066320-R	# CZLCD1 CZLCD2 CZLCD3
SEND	017126-R	# CZLCD2 CZLCD3
SET.CP	015120-R	# CZLCD2 CZLCD3
SET.UP	015314-R	# CZLCD2 CZLCD3
SFPTBL	010464-R	# CZLCD1
STC.00	006542-R	# CZLCD1 CZLCD3
STC.01	006556-R	# CZLCD1 CZLCD3
STC.02	006602-R	# CZLCD1 CZLCD3
STC.03	006626-R	# CZLCD1 CZLCD3
STC.04	006650-R	# CZLCD1 CZLCD3
STC.05	006674-R	# CZLCD1 CZLCD3
STC.06	006724-R	# CZLCD1 CZLCD3
STC.07	006750-R	# CZLCD1 CZLCD3
STC.08	007002-R	# CZLCD1 CZLCD3

STC.09	007022-R	# CZRCD1	CZRCD3	
STC.10	007060-R	# CZRCD1	CZRCD3	
STC.11	007106-R	# CZRCD1	CZRCD3	
STEP	066330-R	# CZRCD1	CZRCD2	CZRCD3
ST.COD	056324-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	

GLOBAL CROSS REFERENCE

CREF V01

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	044060-R	# CZRCD1 CZRCD2 CZRCD3
TICKS	066322-R	# CZRCD1 CZRCD2 CZRCD3
T\$FREE	073302-R	# CZRCD4
T\$PTHV	000002	CZRCD1 # CZRCD4
T.ADDR	045460-R	# CZRCD1 CZRCD2 CZRCD3
T.FLAG	066233-R	# CZRCD1 CZRCD2 CZRCD3
T1	022420-R	CZRCD1 # CZRCD3
UAM	000200	# CZRCD1 # CZRCD2 # CZRCD3
UPD.IO	016572-R	# CZRCD2 CZRCD3
WAIT	017372-R	# CZRCD2 CZRCD3
XFR.CH	016734-R	# CZRCD2 CZRCD3
\$END.L	073304-R	# CZRCD4
\$SAVE2	042320-R	B16MUL # B16SAV CZRCD2 CZRCD3
\$SAVE3	042334-R	# B16SAV CZRCD2 CZRCD3
\$SAVE4	042352-R	# B16SAV CZRCD2 CZRCD3
\$SAVE5	042372-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     = 1641,       ! 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      = #0'1',      ! SUPPRESS PRINTING ERROR LOG MESSAGES
SWF_DM       = #0'2',      ! RUN DM EXERCISER INSTEAD OF MULTI-DRIVE SUBTEST
SWF_RDM      = #0'4',      ! RANDOM SEEK MODE
SWF_CRC      = #0'10',     ! READ-COMPARES AT CONTROLLER
SWF_WO       = #0'20',     ! WRITE ONLY
SWF_CWC      = #0'40',     ! WRITE-COMPARES AT CONTROLLER
SWF_HWC      = #0'100',    ! WRITE-COMPARES AT HOST
SWF_UDP      = #0'200',    ! USER-DEFINED DATA PATTERN
SWF_PHWC     = #0'6',      ! 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_CUP       = 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      = ((#0'200') 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       = #0'100000', ! 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      = %0'4',   ! SET CONTROLLER CHARACTERISTICS COMMAND
OP_ONL      = %0'11',  ! ONLINE COMMAND
OP_ACC      = %0'20',  ! ACCESS COMMAND
OP_RD       = %0'41',  ! READ COMMAND
OP_WRT      = %0'42',  ! WRITE COMMAND
OP_MSK      = %0'177', ! OPCODE MASK
OP_END      = %0'200', ! ENDCODE DESIGNATOR

```

```

:***** DUP COMMAND OPCODES
:

```

```

OP_ESP      = %0'2',   ! EXECUTE SUPPLIED PROGRAM
OP_SND      = %0'4',   ! SEND DATA

```

```

:***** MSCP COMMAND MODIFIERS
:

```

```

MD_EXP      = %0'20000', ! EXPRESS REQUEST
MD_CMP      = %0'40000', ! 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      = %0'100', ! ENABLE MISCELLANEOUS ERROR LOG MESSAGES
CF_THS      = %0'20',  ! ENABLE THIS HOST'S ERROR LOG MESSAGES

```

```

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

```

```

UF_WPH      = %0'20000', ! WRITE PROTECT (HARDWARE)

```

```

:***** STATUS / EVENT CODE DEFINITIONS
:

```

```

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

```

```

:***** END MESSAGE FLAGS
:

```

```

EF_BBR      = %0'200', ! BAD BLOCK REPORTED

```

```

:***** RC25 LITERALS
:

```

```

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

```


SA REGISTER BIT DEFINITIONS

```

SA_S1      = 0'004000'  ! STEP 1 STATUS BIT
SA_S2      = 0'010000'  ! : 2
SA_S3      = 0'020000'  ! : 3
SA_S4      = 0'040000'  ! V 4
SA_ERR     = 0'100000'  ! ERROR INDICATOR
SA_INT     = 0'000200'  ! INTERRUPT ENABLE DURING INITIALIZATION
SA_GO      = 0'000001'  ! GO BIT TO START FIRMWARE

```

INITIALIZATION SEQUENCE READ MASKS

```

S1_MASK    = 0'176000'  ! STEP 1 READ BITS
S2_MASK    = 0'174377'  ! : 2
S3_MASK    = 0'174377'  ! : 3
S4_MASK    = 0'174000'  ! V 4

```

MEMORY MANAGEMENT (KT-11) REGISTER LOCATIONS

```

KTPDR0     = 0'172300'  ! PAGE DESCRIPTOR REGISTERS
KTPDR1     = 0'172302'
KTPDR2     = 0'172304'
KTPDR3     = 0'172306'
KTPDR4     = 0'172310'
KTPDR5     = 0'172312'
KTPDR6     = 0'172314'
KTPDR7     = 0'172316'
KTPAR0     = 0'172340'  ! PAGE ADDRESS REGISTERS
KTPAR1     = 0'172342'
KTPAR2     = 0'172344'
KTPAR3     = 0'172346'
KTPAR4     = 0'172350'
KTPAR5     = 0'172352'
KTPAR6     = 0'172354'
KTPAR7     = 0'172356'
MMR0       = 0'177572'  ! MEMORY MANAGEMENT REGISTER 0
MMR3       = 0'172516'  ! 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.
NOT_PRESENT = 0.
UNPROTECTED = 1.
PROTECTED   = 0.
ONLINE      = 1.
OFFLINE     = 0.
ALL_ONES    = 0'177777'.

! PLATTER IS PRESENT IN CONTROLLER
! PLATTER IS NOT PRESENT IN CONTROLLER
! PLATTER HAS UNPROTECTED CUSTOMER LBN'S
! PLATTER HAS PROTECTED CUSTOMER LBN'S

```

CLOCK FLAGS FOR CLK_TYPE

```

NO_CLOCK    = 0.
L_CLOCK     = -1.
P_CLOCK     = 1.

```

```

.....
FIELD
.....
FIELD
.....

```

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 FIELDS

```

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],     !
  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 (0, FIELDNAM, IMAGE) =
BEGIN
LOCAL
RC_REG;
RC_REG <%FIELDEXPAND (FIELDNAM)> = IMAGE;
(.RC25_ADDR + (%UPVAL * 0)) = .RC_REG;
END%;

! S T R U C T U R E S *
! *
! *****

!***** RC25 ACCESS ALGORITHM
!

STRUCTURE

RC25 [0, P, S, E] =

BEGIN
LOCAL

RC_REG;

RC_REG = .(RC25 + #UPVAL * 0) <0, #BPVAL, 0>;

RC_REG

END

<P, S, E>;

```
; 0001 0  MODULE CZRCD1 (
; 0002 0      *TITLE 'CZRCDB0 RC25 DISK EXERCISER'
; 0003 0      IDENT = 'V02.0',
; 0004 0      ADDRESSING_MODE (ABSOLUTE)
; 0005 0      ) =
; 0006 1  BEGIN
; 0007 1
; 0043 1  *SBTTL 'PROGRAM HEADER'
; 0044 1
; 0045 1  LIBRARY 'CZRCDL';           ! RC25 EXERCISER GLOBAL LIBRARY
; 0046 1  REQUIRE 'BLSMAC.REQ';     ! DIAGNOSTIC SUPERVISOR LIBRARY
; 1535 1
; 1536 1  LITERAL
; 1537 1      DS#NBR_OF_TESTS = 1;   ! NUMBER OF TESTS IN THIS DIAGNOSTIC
; 1538 1
; 1539 1  EQUALS;
; 1540 1
; 1541 1  POINTER (ALL);
; 1542 1
; 1543 1  !+
; 1544 1  !   THE PROGRAM HEADER IS THE INTERFACE BETWEEN THE DIAGNOSTIC PROGRAM
; 1545 1  !   AND THE SUPERVISOR. THE ARGUMENTS FOR THE "HEADER" MACRO ARE: PROGRAM
; 1546 1  !   NAME, REV, PATCH, LONGEST TEST TIME, TYPE (WHERE 0 = SEQUENTIAL
; 1547 1  !   DIAGNOSTIC, 1 = EXERCISER), AND, OPTIONALLY, THE PROCESSOR PRIORITY TO
; 1548 1  !   BE SET WHEN STARTING THE DIAGNOSTIC.
; 1549 1  !-
; 1550 1
; 1551 1  HEADER (*ASCII' CZRCD', *ASCII'B', *ASCII'O', 65535, 1, PRI00);
; 1552 1
; 1553 1  !+
; 1554 1  !   THE DISPATCH TABLE CONTAINS THE STARTING ADDRESS OF EACH TEST. IT
; 1555 1  !   IS USED BY THE SUPERVISOR TO DISPATCH TO EACH TEST. THE LITERAL
; 1556 1  !   "DS#NBR_OF_TESTS" REPRESENTS THE NUMBER OF TESTS (1) IN THIS PROGRAM.
; 1557 1  !-
; 1558 1
; 1559 1  DISPATCH (DS#NBR_OF_TESTS);
```



```
: 1560 1 *SBTTL 'GLOBAL DATA SECTION'
: 1561 1
: 1562 1 !+
: 1563 1 ! THE GLOBAL DATA SECTION CONTAINS ALL DYNAMICALLY-MODIFIED DATA.
: 1564 1 !-
: 1565 1
: 1566 1 PSECT GLOBAL = $FFF$ (READ, NOWRITE, EXECUTE, LOCAL, CONCATENATE);
: 1567 1
: 1568 1 GLOBAL
: 1569 1 PATCH : VECTOR [100, WORD], ! PATCH AREA
: 1570 1 CPT : VECTOR [MAX_UNITS, BYTE],
: 1571 1 ! CURRENT PASS TESTING (YES / NO) PER UNIT
: 1572 1 CST : BLOCKVECTOR [MAX_CTLR, CST_LEN, WORD] FIELD (C_FIELDS),
: 1573 1 ! RUN-TIME CONTROLLER STATUS TABLES
: 1574 1 CST_ADDR : REF BLOCK [CST_LEN, WORD] FIELD (C_FIELDS),
: 1575 1 ! CONTROLLER STATUS TABLE ADDRESS OF "CURRENT" CONTROLLER
: 1576 1 DCT : BLOCKVECTOR [MAX_CTLR, DCT_LEN, WORD] FIELD (DC_FIELDS),
: 1577 1 ! DRIVER CONTROLLER TABLES
: 1578 1 DCT_ADDR : REF BLOCK [DCT_LEN, WORD] FIELD (DC_FIELDS),
: 1579 1 ! ADDRESS OF "CURRENT" DRIVER CONTROLLER TABLE
: 1580 1 RC25_ADDR : REF RC25 FIELD (RC_REG),
: 1581 1 ! DEVICE ADDRESS OF "CURRENT" CONTROLLER
: 1582 1 IRC25_ADDR : REF RC25 FIELD (RC_REG),
: 1583 1 ! DEVICE ADDRESS OF INTERRUPTING CONTROLLER
: 1584 1 DM_COMM : BLOCKVECTOR [MAX_CTLR, DMC_LEN, WORD] FIELD (DMC_FIELDS),
: 1585 1 ! DM EXERCISER COMMUNICATION AREA (LINK TO FRONT PANEL TEST)
: 1586 1 DMC_ADDR : REF BLOCK [DMC_LEN, WORD] FIELD (DMC_FIELDS),
: 1587 1 ! ADDRESS OF CURRENT CONTROLLER'S DM EXERCISER COMMUNICATION AREA
: 1588 1 RP_SAVE : VECTOR [MAX_CTLR * RPS_LEN, BYTE, SIGNED],
: 1589 1 ! RETURN PACKET SAVE AREA
: 1590 1 RPS_X1 : WORD, ! STARTING INDEX OF CURRENT CONTROLLER'S RP_SAVE AREA
: 1591 1 RPS_X2 : WORD, ! ENDING INDEX OF CURRENT CONTROLLER'S RP_SAVE AREA
: 1592 1 OUTC_LIST : VECTOR [MAX_CTLR * OUTC_CNT, BYTE, SIGNED],
: 1593 1 ! OUTSTANDING COMMAND LIST (CONTAINS MSCP ENVELOPE INDECES)
: 1594 1 OUTC_TIMR : VECTOR [MAX_CTLR * OUTC_CNT, WORD],
: 1595 1 ! OUTSTANDING COMMAND TIMERS
: 1596 1 OCL_X1 : WORD,
: 1597 1 ! STARTING INDEX OF CURRENT CONTROLLER'S OUTSTANDING COMMAND AREA
: 1598 1 OCL_X2 : WORD,
: 1599 1 ! ENDING INDEX OF CURRENT CONTROLLER'S OUTSTANDING COMMAND AREA
: 1600 1 TALLY : VECTOR [MAX_UNITS * TALLY_LEN, WORD] FIELD (T_FIELDS),
: 1601 1 ! STATISTICS TABLES
: 1602 1 T_ADDR : REF BLOCK [TALLY_LEN, WORD] FIELD (T_FIELDS),
: 1603 1 ! ADDRESS OF STATISTICS TABLE (TALLY) FOR CURRENT UNIT
: 1604 1 MSCP_ENV : BLOCKVECTOR [ENV_CNT, ENV_LEN, WORD] FIELD (E_FIELDS),
: 1605 1 ! MSCP ENVELOPE POOL
: 1606 1 ENV_USE : VECTOR [ENV_CNT, BYTE, SIGNED],
: 1607 1 ! MSCP ENVELOPE POOL ALLOCATION TABLE
: 1608 1 RETPKT : BLOCKVECTOR [RP_CNT, RP_LEN, WORD] FIELD (RP_FIELDS),
: 1609 1 ! RETURN PACKET POOL
: 1610 1 RP_USE : VECTOR [RP_CNT, BYTE, SIGNED],
: 1611 1 ! RETURN PACKET POOL ALLOCATION TABLE
: 1612 1 RP_INDX : WORD, ! CURRENT RETURN PACKET INDEX
: 1613 1 RP_ADDR : REF BLOCK [RP_LEN, WORD] FIELD (RP_FIELDS),
: 1614 1 ! CURRENT RETURN PACKET ADDRESS
: 1615 1 BUFF_DESC : BLOCKVECTOR [MAX_BUF_CNT, DESC_LEN, WORD] FIELD (BD_FIELDS),
: 1616 1 ! TABLE OF I/O BUFFER DESCRIPTORS
```

```

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

```



```
: 1665 1 #SBTTL 'GLOBAL TEXT SECTION'
: 1666 1
: 1667 1 !+
: 1668 1 ! THE GLOBAL TEXT SECTION CONTAINS ALL MESSAGES OUTPUT TO THE OPERATOR
: 1669 1 ! DURING THE OPERATION OF THE EXERCISER. THIS INCLUDES HARDWARE AND
: 1670 1 ! SOFTWARE DIALOG PROMPTS, ERROR MESSAGES, AND DROP UNIT MESSAGES.
: 1671 1 !-
: 1672 1
: 1673 1 GLOBAL BIND
: 1674 1 !
: 1675 1 !***** HARDWARE DIALOG
: 1676 1 !
: 1677 1 HWQ1 = UPLIT (#ASCIZ'IP ADDRESS'),
: 1678 1 HWQ2 = UPLIT (#ASCIZ'VECTOR'),
: 1679 1 HWQ3 = UPLIT (#ASCIZ'BR LEVEL'),
: 1680 1 HWQ4 = UPLIT (#ASCIZ'PLATTER ADDRESS (UNIT PLUG)'),
: 1681 1 HWQ5 = UPLIT (#ASCIZ'ALLOW WRITES TO CUSTOMER DATA AREA ON THIS PLATTER'),
: 1682 1 HWQ6 = UPLIT (#ASCIZ'** WARNING - CUSTOMER DATA AREA MAY BE OVERWRITTEN! ... CONFIRM'),
: 1683 1 !
: 1684 1 !***** SOFTWARE DIALOG
: 1685 1 !
: 1686 1 SWQ1 = UPLIT (#ASCIZ'ERROR LIMIT (0 FOR NO LIMIT)'),
: 1687 1 SWQ2 = UPLIT (#ASCIZ'TRANSFER LIMIT IN MEGABYTES (0 FOR NO LIMIT)'),
: 1688 1 SWQ3 = UPLIT (#ASCIZ'SUPPRESS PRINTING ERROR LOG MESSAGES'),
: 1689 1 SWQ4 = UPLIT (#ASCIZ'RUN DM EXERCISER INSTEAD OF MULTI-DRIVE SUBTEST'),
: 1690 1 SWQ5 = UPLIT (#ASCIZ'RANDOM SEEK MODE'),
: 1691 1 SWQ6 = UPLIT (#ASCIZ'STARTING TRACK'),
: 1692 1 SWQ7 = UPLIT (#ASCIZ'ENDING TRACK'),
: 1693 1 SWQ8 = UPLIT (#ASCIZ'READ-COMPARES PERFORMED AT THE CONTROLLER'),
: 1694 1 SWQ9 = UPLIT (#ASCIZ'WRITE ONLY'),
: 1695 1 SWQ10 = UPLIT (#ASCIZ'WRITE-COMPARES PERFORMED AT THE CONTROLLER'),
: 1696 1 SWQ11 = UPLIT (#ASCIZ'CHECK ALL WRITES AT HOST BY READING'),
: 1697 1 SWQ12 = UPLIT (#ASCIZ'USER-DEFINED DATA PATTERN'),
: 1698 1 SWQ13 = UPLIT (#ASCIZ'SELECT PRE-DEFINED DATA PATTERN (0 FOR SEQUENTIAL SELECTION)'),
: 1699 1 SWQ14 = UPLIT (#ASCIZ'NUMBER OF WORDS IN DATA PATTERN (16 MAXIMUM)'),
: 1700 1 SWQ15 = UPLIT (#ASCIZ'PATTERN VALUE'),
: 1701 1 SWM1 = UPLIT (#ASCIZ'THE REMAINING QUESTIONS ONLY APPLY TO UNPROTECTED PLATTERS. '),
: 1702 1 NULL = UPLIT (#ASCIZ''), ! ADDED TO COVER DRS BUG (NEXT MESSAGE ALSO PRINTED)
: 1703 1 !
: 1704 1 !***** INFORMATION MESSAGES (IF ATTENDED MODE, THEN PRINTF)
: 1705 1 !
: 1706 1 MSG_01 = UPLIT (#ASCIZ'#APOWER DELAY - WAITING#N'),
: 1707 1 MSG_02 = UPLIT (#ASCIZ'#N#AABOUT TO VERIFY VECTOR #03#A(0) FOR DEVICE #06#A(0) ... '),
: 1708 1 MSG_03 = UPLIT (#ASCIZ'#ACOMPLETED.#N'),
: 1709 1 MSG_04 = UPLIT (#ASCIZ'#N#AINIT SUBTEST START#N'),
: 1710 1 MSG_05 = UPLIT (#ASCIZ'#N#AMULTI-DRIVE SUBTEST START#N'),
: 1711 1 MSG_06 = UPLIT (#ASCIZ'#N#ADM EXERCISER SUBTEST START#N'),
: 1712 1 MSG_07 = UPLIT (#ASCIZ'#AUNIT #D2#A. - TRANSFER LIMIT REACHED#N'),
: 1713 1 MSG_08 = UPLIT (#ASCIZ'#D5#A. BLOCKS TRANSFERRED ON UNIT #D2#A. (PLATTER #D3#A.)#N'),
: 1714 1 !
: 1715 1 !***** CONFIGURATION ERROR MESSAGES (IF ATTENDED MODE, THEN PRINTF)
: 1716 1 !
: 1717 1 CER_01 = UPLIT (#ASCIZ'#N#ADUPLICATE PLATTER ADDRESS #D3#A. AT IP #06#A(0)'),
: 1718 1 CER_02 = UPLIT (#ASCIZ'#N#AALREADY 4 UNITS AT IP #06#A(0)'),
: 1719 1 CER_03 = UPLIT (#ASCIZ'#N#AMORE THAN #D1#A DIFFERENT IP ADDRESSES'),
: 1720 1 !
: 1721 1 !***** DROP UNIT MESSAGES
```



```
: 1722 1 !
: 1723 1 !   DUM_00 = UPLIT (ASCIZ' AUNIT D2 A. DROPPED - '),
: 1724 1 !   DUM_UC = UPLIT (ASCIZ' AUSER COMMAND N'),
: 1725 1 !   DUM_CE = UPLIT (ASCIZ' ACONFIGURATION ERROR N'),
: 1726 1 !   DUM_IE = UPLIT (ASCIZ' AINIT ERROR N'),
: 1727 1 !   DUM_HE = UPLIT (ASCIZ' AHARD ERROR LIMIT REACHED N'),
: 1728 1 !   DUM_UE = UPLIT (ASCIZ' AUNRECOVERABLE ERROR N'),
: 1729 1 !
: 1730 1 ! ***** GENERAL ERROR MESSAGES
: 1731 1 !
: 1732 1 !   SYSTEM FATAL (ERRSF)
: 1733 1 !
: 1734 1 !   EGS_01 = UPLIT (ASCIZ' TOO MANY UNITS'),
: 1735 1 !   EGS_02 = UPLIT (ASCIZ' NEITHER P NOR L CLOCK WAS FOUND ON THE SYSTEM'),
: 1736 1 !
: 1737 1 !   DEVICE FATAL (ERRDF)
: 1738 1 !
: 1739 1 !   EGD_10 = UPLIT (ASCIZ' REGISTER EXISTENCE TEST FAILED'),
: 1740 1 !   EGD_11 = UPLIT (ASCIZ' VECTOR TEST FAILED'),
: 1741 1 !   EGD_12 = UPLIT (ASCIZ' BR LEVEL TEST FAILED'),
: 1742 1 !   EGD_13 = UPLIT (ASCIZ' INIT SEQUENCE FAILED'),
: 1743 1 !   EGD_14 = UPLIT (ASCIZ' FATAL PORT / CONTROLLER ERROR'),
: 1744 1 !   EGD_15 = UPLIT (ASCIZ' MESSAGE RESPONSE TIMEOUT'),
: 1745 1 !   EGD_16 = UPLIT (ASCIZ' ONLINE FAILED'),
: 1746 1 !   EGD_17 = UPLIT (ASCIZ' WRITE-PROTECT CONFLICT'),
: 1747 1 !   EGD_18 = UPLIT (ASCIZ' ACCESS FAILED'),
: 1748 1 !   EGD_19 = UPLIT (ASCIZ' FATAL I/O ERROR'),
: 1749 1 !   EGD_20 = UPLIT (ASCIZ' CONTROLLER TIMEOUT'),
: 1750 1 !   EGD_21 = UPLIT (ASCIZ' DUP COMMAND FAILED'),
: 1751 1 !   EGD_22 = UPLIT (ASCIZ' DM EXERCISER TIMEOUT'),
: 1752 1 !
: 1753 1 !   HARD (ERRHRD)
: 1754 1 !
: 1755 1 !   EGH_30 = UPLIT (ASCIZ' I/O REQUEST FAILED'),
: 1756 1 !
: 1757 1 ! ***** BASIC ERROR MESSAGES (PRINTB)
: 1758 1 !
: 1759 1 !   SYSTEM FATAL (ERRSF)
: 1760 1 !
: 1761 1 !   EBS_01 = UPLIT (ASCIZ' AMORE THAN D2 A. UNITS SPECIFIED N'),
: 1762 1 !
: 1763 1 !   DEVICE FATAL (ERRDF)
: 1764 1 !
: 1765 1 !   EBD_10 = UPLIT (ASCIZ' AND RESPONSE AT ADDRESS 06 A(0) N'),
: 1766 1 !   EBD_12 = UPLIT (ASCIZ' AINCORRECT BR LEVEL GIVEN FOR DEVICE 06 A(0) N'),
: 1767 1 !   EBD_13 = UPLIT (ASCIZ' ASTEP D1 A READ ERROR ON DEVICE 06 A(0) N'),
: 1768 1 !   EBD_14 = UPLIT (ASCIZ' AERROR CODE RECEIVED IN SA REGISTER OF DEVICE 06 A(0) N'),
: 1769 1 !   EBD_15 = UPLIT (ASCIZ' AFAILED TO RECEIVE END MESSAGE FROM DEVICE 06 A(0) N'),
: 1770 1 !   EBD_16 = UPLIT (ASCIZ' AERROR IN RESPONSE TO ONLINE COMMAND FOR PLATTER D3 A. N'),
: 1771 1 !   EBD_17 = UPLIT (ASCIZ' APLATTER D3 A. IS SW WRITE-ENABLED BUT HW WRITE-PROTECTED N'),
: 1772 1 !   EBD_18 = UPLIT (ASCIZ' AACCESS FAILED ON PLATTER D3 A. N'),
: 1773 1 !   EBD_19 = UPLIT (ASCIZ' APLATTER D3 A. WENT OFFLINE N'),
: 1774 1 !   EBD_20 = UPLIT (ASCIZ' ADEVICE 06 A(0) NOT PROCESSING COMMAND PACKETS N'),
: 1775 1 !   EBD_21 = UPLIT (ASCIZ' AMESSAGE REJECTED BY DUP SERVER ON DEVICE 06 A(0) N'),
: 1776 1 !   EBD_22 = UPLIT (ASCIZ' AND RESPONSE FROM FRONT PANEL TEST EXECUTING IN DEVICE 06 A(0) N'),
: 1777 1 !
: 1778 1 !   HARD (ERRHRD) - MAINLY BASED ON MSCP STATUS CODES
```



```
: 1779 1 !
: 1780 1   STC_00 = UPLIT (ASCIZ'ASUCCESS'N'),
: 1781 1   STC_01 = UPLIT (ASCIZ'AINVALID COMMAND'N'),
: 1782 1   STC_02 = UPLIT (ASCIZ'ACOMMAND ABORTED'N'),
: 1783 1   STC_03 = UPLIT (ASCIZ'AUNIT-OFFLINE'N'),
: 1784 1   STC_04 = UPLIT (ASCIZ'AUNIT-AVAILABLE'N'),
: 1785 1   STC_05 = UPLIT (ASCIZ'AMEDIA FORMAT ERROR'N'),
: 1786 1   STC_06 = UPLIT (ASCIZ'AWRITE-PROTECTED'N'),
: 1787 1   STC_07 = UPLIT (ASCIZ'ADEVICE COMPARE ERROR'N'),
: 1788 1   STC_08 = UPLIT (ASCIZ'ADATA ERROR'N'),
: 1789 1   STC_09 = UPLIT (ASCIZ'AHOST BUFFER ACCESS ERROR'N'),
: 1790 1   STC_10 = UPLIT (ASCIZ'ACONTROLLER ERROR'N'),
: 1791 1   STC_11 = UPLIT (ASCIZ'ADRIVE ERROR'N'),
: 1792 1 !
: 1793 1   EBH_31 = STC_01,
: 1794 1   EBH_32 = STC_02,
: 1795 1   EBH_34 = STC_04,
: 1796 1   EBH_35 = STC_05,
: 1797 1   EBH_36 = STC_06,
: 1798 1   EBH_37 = STC_07,
: 1799 1   EBH_38 = STC_08,
: 1800 1   EBH_39 = STC_09,
: 1801 1   EBH_40 = STC_10,
: 1802 1   EBH_41 = STC_11,
: 1803 1   EBH_42 = UPLIT (ASCIZ'AHOST-DETECTED WRITE-COMPARE ERROR'N'),
: 1804 1   EBH_43 = UPLIT (ASCIZ'AFAILED TO RECEIVE END MESSAGE FOR I/O COMMAND'N'),
: 1805 1 !
: 1806 1 !***** EXTENDED ERROR (PRINTX) AND ERROR LOG (PRINTF) MESSAGES
: 1807 1 !
: 1808 1   EX_SA = UPLIT (ASCIZ'A SA: %06A(0)N'),
: 1809 1   EX_CRN = UPLIT (ASCIZ'A CMD REF NUM: %06A(0)N'),
: 1810 1   EX_SC = UPLIT (ASCIZ'A STATUS CODE: %02A(0)N'),
: 1811 1   EX_DSC = UPLIT (ASCIZ'A DUP STATUS CODE: %D1AN'),
: 1812 1   EX_SB = UPLIT (ASCIZ'A SUB-CODE: %04A(0)N'),
: 1813 1   EX_CMD = UPLIT (ASCIZ'A COMMAND: '),
: 1814 1   EX_SCC = UPLIT (ASCIZ'ASET CTLR CHAR'),
: 1815 1   EX_ONL = UPLIT (ASCIZ'AONLINE'),
: 1816 1   EX_ESP = UPLIT (ASCIZ'AEXECUTE SUPPLIED PROGRAM'),
: 1817 1   EX_SND = UPLIT (ASCIZ'ASEND DATA'),
: 1818 1   EX_RD = UPLIT (ASCIZ'AREAD'),
: 1819 1   EX_WRT = UPLIT (ASCIZ'AWRITE'),
: 1820 1   EX_CMP = UPLIT (ASCIZ'A-COMPARE'N'),
: 1821 1   EX_BB = UPLIT (ASCIZ'A BAD BLOCK REPORTED: %D5AN'),
: 1822 1   EX_LBN = UPLIT (ASCIZ'A LBN: %D5AN'),
: 1823 1   EX_CBC = UPLIT (ASCIZ'A BYTE COUNT IN COMMAND: %D5AN'),
: 1824 1   EX_BC = UPLIT (ASCIZ'A ACTUAL # OF BYTES TRANSFERRED: %D5AN'),
: 1825 1   EX_BD = UPLIT (ASCIZ'A I/O BUFFER DESCRIPTOR: %07A(0)%07A(0)N'),
: 1826 1   EX_EL = UPLIT (ASCIZ'NAERROR LOG MESSAGE RECEIVED: N'),
: 1827 1   EX_PA = UPLIT (ASCIZ'A PLATTER: %D3AN'),
: 1828 1   EX_FMT = UPLIT (ASCIZ'A FORMAT: '),
: 1829 1   EX_EVC = UPLIT (ASCIZ'A EVENT CODE: '),
: 1830 1   EX_HMA = UPLIT (ASCIZ'A HOST MEM ADDR: %07A(0)%07A(0)N'),
: 1831 1   EX_O3 = UPLIT (ASCIZ'%03A(0)N'),
```

```
: 1832 1      !
: 1833 1      !***** MISCELLANEOUS
: 1834 1      !
: 1835 1      !     ETIME = UPLIT (ASCIZ'D2A:D2A:D2A '),
: 1836 1      !     PLATT = UPLIT (ASCIZ'APLATTER D3A. - '),
: 1837 1      !     CRLF = UPLIT (ASCIZ'N');
: 1838 1
: 1839 1      !SBTTL 'DEFAULT HARDWARE P-TABLE'
: 1840 1
: 1841 1      !.
: 1842 1      !     THE DEFAULT HARDWARE P-TABLE CONTAINS DEFAULT VALUES OF THE TEST-DEVICE
: 1843 1      !     PARAMETERS. THE STRUCTURE OF THIS TABLE IS IDENTICAL TO THE STRUCTURE
: 1844 1      !     OF THE ACTUAL HARDWARE P-TABLES, WHICH RESIDE IN SUPERVISOR SPACE, AND
: 1845 1      !     IS USED AS A "TEMPLATE" FOR BUILDING THE P-TABLES.
: 1846 1      !-
: 1847 1
: 1848 1      !BGNHW (DFPTBL);
: 1849 1
: 1850 1      !GLOBAL
: 1851 1
: 1852 1      !     HWPT_IP_ADDR : WORD INITIAL (0'172150'),      ! IP ADDRESS
: 1853 1      !     HWPT_VECTOR : WORD INITIAL (0'154'),           ! VECTOR ADDRESS
: 1854 1      !     HWPT_BR_LEVEL : WORD INITIAL (5),             ! BR LEVEL
: 1855 1      !     HWPT_PLAT : WORD INITIAL (0);                 ! PLATTER ADDR, PROTECTON BIT
: 1856 1
: 1857 1      !ENDHW;
```


CZLCD1
V02.0

CZLCD0 RC25 DISK EXERCISER
SOFTWARE P-TABLE

14-Jun-1985 09:35:23
14-Jun-1985 09:28:52

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZLCD0]CZLCD1.SRC;6

SEQ 0069
Page 9
(6)

```
: 1858 1  *SBTTL 'SOFTWARE P-TABLE'  
: 1859 1  
: 1860 1  !+  
: 1861 1  ! THE SOFTWARE P-TABLE CONTAINS VARIOUS DATA USED BY THE PROGRAM AS  
: 1862 1  ! OPERATIONAL PARAMETERS. THESE PARAMETERS ARE SET UP AT ASSEMBLY TIME  
: 1863 1  ! AND MAY BE VARIED BY THE OPERATOR AT RUN TIME.  
: 1864 1  !-  
: 1865 1  
: 1866 1  BGNSW (SFPTBL);  
: 1867 1  
: 1868 1  GLOBAL  
: 1869 1  
: 1870 1      SWP_ERROR : WORD INITIAL (32),           ! HARD ERROR LIMIT FOR DROPPING UNIT  
: 1871 1      SWP_XFER  : WORD INITIAL (2),           ! TRANSFER LIMIT PER UNIT PER PASS  
: 1872 1      SWP_STRACK : WORD INITIAL (0),           ! STARTING TRACK  
: 1873 1      SWP_ETRACK : WORD INITIAL (MAX_TRACK),   ! ENDING TRACK  
: 1874 1      SWP_FLAGS  : BYTE INITIAL ('055'),       ! FLAGS (SEE DOCUMENTATION)  
: 1875 1      SWP_DPAT   : BYTE INITIAL (0),           ! DATA PATTERN NUMBER  
: 1876 1      SWP_UCNT   : WORD INITIAL (MAX_UDP_CNT), ! USER DATA PATTERN COUNT  
: 1877 1      SWP_UDPAT  : VECTOR [MAX_UDP_CNT, WORD]; ! USER DATA PATTERN  
: 1878 1  
: 1879 1  ENDSW;
```

CZRCDD1
V02.0

CZRCDB0 RC25 DISK EXERCISER
PROTECTION TABLE

14-Jun-1985 09:35:23
14-Jun-1985 09:28:52

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRCDD1.SRC;6

SEQ 0070
Page 10
(7)

```

: 1880 1 *SBTTL 'PROTECTION TABLE'
: 1881 1
: 1882 1 !+
: 1883 1 ! THIS TABLE IS USED BY THE RUNTIME SERVICES TO PROTECT THE LOAD MEDIA.
: 1884 1 ! 1ST ARG = BYTE OFFSET INTO P-TABLE FOR CSR ADDRESS
: 1885 1 ! 2ND ARG = BYTE OFFSET INTO P-TABLE FOR MASSBUS ADDRESS
: 1886 1 ! 3RD ARG = BYTE OFFSET INTO P-TABLE FOR DRIVE NUMBER
: 1887 1 ! BYTE OFFSET REFERS TO THE NUMBER OF BYTES FROM THE BEGINNING OF A
: 1888 1 ! PTABLE ENTRY TO THE ITEM IN QUESTION. IF THE PARTICULAR ITEM DOES NOT
: 1889 1 ! APPLY, THEN ENTRY IS SET TO -1. WHEN THE RUNTIME SERVICES EXECUTES A
: 1890 1 ! GPWARD, IT USES THESE OFFSETS (IF NOT SET TO -1) TO GET THE ITEMS AND
: 1891 1 ! COMPARE WITH THOSE SAVED IN THE XXDP+ MONITOR. IF THE UNIT BEING
: 1892 1 ! REQUESTED MATCHES THE LOAD DEVICE, THEN THE RUNTIME SERVICES RETURN AN
: 1893 1 ! INCOMPLETE FLAG ON THE GPWARD.
: 1894 1 !-
: 1895 1
: 1896 1 BGNPROT (-1, -1, -1);
: 1897 1
: 1898 1 ENDPROT;
: 1899 1
: 1900 1 END
: 1901 1
: 1902 0 ELUDOM

```

```

.TITLE CZRCDD1 CZRCDB0 RC25 DISK EXERCISER
.IDENT /V02.0/
.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 102 .ASCII /B/
000011 060 .ASCII /O/
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

```


CZRCDD1
V02.0

CZRCDB0 RC25 DISK EXERCISER
PROTECTION TABLE

14-Jun-1985 09:35:23
14-Jun-1985 09:28:52

VAX-11 Bliss-16 V4.0-579
USER\$1:[AZTEC.CZRCDB]CZRCDD1.SRC;6

SEQ 0071
Page 11
(7)

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	/IP /
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 /(O /
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	P.AAH:	.ASCII /)/<00><00>
000452	124	122	101		.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 / (O/
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	/O F/
001242	117	122	040		.ASCII	/OR /
001245	123	105	121		.ASCII	/SEQ/
001250	125	105	116		.ASCII	/UEN/

CZRCDD1
V02.0

CZRCDBO RC25 DISK EXERCISER
PROTECTION TABLE

14-Jun-1985 09:35:23
14-Jun-1985 09:28:52

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRCDD1.SRC;6

SEQ 0075
Page 15
(7)

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>	

CZRC01
V02.0

CZRCDB0 RC25 DISK EXERCISER
PROTECTION TABLE

14-Jun-1985 09:35:23
14-Jun-1985 09:28:52

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC01.SRC;6

SEQ 0076
Page 16
(7)

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	/03/
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	/06%/
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	/2A/
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	/2A/
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	/ %0/
002215	066	045	101		.ASCII	/6A/
002220	050	117	051		.ASCII	/(0)/
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	/06%/	
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>/0/
003201	040	105	122		.ASCII	/ ER/
003204	122	117	122		.ASCII	/ROR/
003207	000				.ASCII	<00>
003210	103	117	116	P.ACA:	.ASCII	/CON/

CZRC01
V02.0

CZRCDB0 RC25 DISK EXERCISER
PROTECTION TABLE

14-Jun-1985 09:35:23
14-Jun-1985 09:28:52

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC01.SRC;6

SEQ 0081
Page 21
(7)

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>/0/
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	/ %0/
003433	066	045	101	.ASCII	/6%A/
003436	050	117	051	.ASCII	/(0)/
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	/06%/	
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	/06%/	
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 /	

CZRC01
V02.0

CZRCDB0 RC25 DISK EXERCISER
PROTECTION TABLE

14-Jun-1985 09:35:23
14-Jun-1985 09:28:52

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC01.SRC;6

SEQ 0083
Page 23
(7)

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	/06%/
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/

CZRCDD1
V02.0

CZRCDB0 RC25 DISK EXERCISER
PROTECTION TABLE

14-Jun-1985 09:35:23
14-Jun-1985 09:28:52

VAX-11 Bliss-16 V4.0-579
USER#1:(AZTEC.CZRCDB)CZRCDD1.SRC;6

SEQ 0084
Page 24
(7)

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	/#06/	
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	/ #0/	
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/	

CZRCDD1
V02.0

CZRCDBO RC25 DISK EXERCISER
PROTECTION TABLE

14-Jun-1985 09:35:23
14-Jun-1985 09:28:52

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRCDD1.SRC;6

SEQ 0085
Page 25
(7)

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	/ACO/
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.ACIV:	.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.ACIV:	.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	115	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>/0/	
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	/ %0/	
005272	066	045	101	.ASCII	/6%A/	
005275	050	117	051	.ASCII	/(0)/	
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	/ %0/	
005331	066	045	101	.ASCII	/6%A/	
005334	050	117	051	.ASCII	/(0)/	
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	/ %0/	
005367	062	045	101	.ASCII	/2%A/	
005372	050	117	051	.ASCII	/(0)/	
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	/ %0/
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/

CZRC01
V02.0

CZRCDB0 RC25 DISK EXERCISER
PROTECTION TABLE

14-Jun-1985 09:35:23
14-Jun-1985 09:28:52

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC01.SRC;6

SEQ 0089
Page 29
(7)

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>/0/
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 /: %0/
006126	067	045	101		.ASCII /7%A/
006131	050	117	051		.ASCII /(0)/

006134	045	117	067	.ASCII	/#07/	
006137	045	101	050	.ASCII	/#A(/	
006142	117	051	045	.ASCII	/0)#/	
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	110	.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	/07#/	
006336	101	050	117	.ASCII	/A(O/	
006341	051	045	117	.ASCII	/)#0/	
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	/#03/
006361	045	101	050	.ASCII	/#A(/	
006364	117	051	045	.ASCII	/0)#/	
006367	116	000	000	.ASCII	/N/<00><00>	
006372	045	104	062	P.AED:	.ASCII	/#D2/
006375	045	101	072	.ASCII	/#A:/	

CZRC01
V02.0

CZRCDB0 RC25 DISK EXERCISER
PROTECTION TABLE

14-Jun-1985 09:35:23
14-Jun-1985 09:28:52

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC01.SRC:6

SEQ 0091
Page 31
(7)

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

CZRC01
V02.0

CZRCDB0 RC25 DISK EXERCISER
PROTECTION TABLE

14-Jun-1985 09:35:23
14-Jun-1985 09:28:52

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC01.SRC;6

SEQ 0092
Page 32
(7)

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

CZRC01
V02.0

CZRCDB0 RC25 DISK EXERCISER
PROTECTION TABLE

14-Jun-1985 09:35:23
14-Jun-1985 09:28:52

VAX-11 Bliss-16 V4.0-579
USER\$1:[AZTEC.CZRCDB]CZRC01.SRC;6

SEQ 0093
Page 33
(7)

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.ACW
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		
:					

CZRC01
V02.0

CZRCDB0 RC25 DISK EXERCISER
PROTECTION TABLE

14-Jun-1985 09:35:23
14-Jun-1985 09:28:52

VAX-11 Bliss-16 V4.0-579
USER\$1:[AZTEC.CZRCDB]CZRC01.SRC;6

SEQ 0097
Page 37
(7)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
USER\$1:[AZTEC.CZRCDB]CZRC01.L16;6	276	138	50	16	00:00.2

COMMAND QUALIFIERS

BLISS/PDP11 CZRC01.SRC/LIST/EN:NOEIS

: Size: 0 code + 6810 data words
: Run Time: 00:58.9
: Elapsed Time: 00:59.9
: Lines/CPU Min: 1938
: Lexemes/CPU-Min: 13067
: Memory Used: 258 pages
: Compilation Complete

```
: 0001 0  MODULE CZRCD2 (  
: 0002 0      *TITLE 'CZRCDB0 RC25 DISK EXERCISER'  
: 0003 0      IDENT = 'V02.0',  
: 0004 0      ADDRESSING_MODE (ABSOLUTE)  
: 0005 0      ) =  
: 0006 1  BEGIN  
: 0007 1  
: 0043 1  *SBTTL 'DECLARATIONS'  
: 0044 1  
: 0045 1  LIBRARY 'CZRCDL';           ! RC25 EXERCISER GLOBAL LIBRARY  
: 0046 1  REQUIRE 'BLSMAC.REQ';     ! DIAGNOSTIC SUPERVISOR LIBRARY  
: 1535 1  
: 1536 1  FORWARD ROUTINE  
: 1537 1      NEX_TRAP : L$ISR NOVALUE,  
: 1538 1      CLK_INT_SERV : L$ISR NOVALUE,  
: 1539 1      EMS_01 : NOVALUE;  
: 1540 1  
: 1541 1  EXTERNAL  
: 1542 1      PATCH : VECTOR [100, WORD],           ! PATCH AREA  
: 1543 1      CPT : VECTOR [MAX_UNITS, BYTE],  
: 1544 1      ! CURRENT PASS TESTING (YES / NO) PER UNIT  
: 1545 1      CST : BLOCKVECTOR [MAX_CTLR, CST_LEN, WORD] FIELD (C_FIELDS),  
: 1546 1      ! RUN-TIME CONTROLLER STATUS TABLES  
: 1547 1      CST_ADDR : REF BLOCK [CST_LEN, WORD] FIELD (C_FIELDS),  
: 1548 1      ! CONTROLLER STATUS TABLE ADDRESS OF "CURRENT" CONTROLLER  
: 1549 1      DCT : BLOCKVECTOR [MAX_CTLR, DCT_LEN, WORD] FIELD (DC_FIELDS),  
: 1550 1      ! DRIVER CONTROLLER TABLES  
: 1551 1      DCT_ADDR : REF BLOCK [DCT_LEN, WORD] FIELD (DC_FIELDS),  
: 1552 1      ! ADDRESS OF "CURRENT" DRIVER CONTROLLER TABLE  
: 1553 1      RC25_ADDR : REF RC25 FIELD (RC_REG),  
: 1554 1      ! DEVICE ADDRESS OF "CURRENT" CONTROLLER
```



```
: 1555 1      IRC25_ADDR : REF RC25 FIELD (RC_REG),
: 1556 1      !      DEVICE ADDRESS OF INTERRUPTING CONTROLLER
: 1557 1      DM_COMM : BLOCKVECTOR [MAX_CTLR, DMC_LEN, WORD] FIELD (DMC_FIELDS),
: 1558 1      !      DM EXERCISER COMMUNICATION AREA (LINK TO FRONT PANEL TEST)
: 1559 1      DMC_ADDR : REF BLOCK [DMC_LEN, WORD] FIELD (DMC_FIELDS),
: 1560 1      !      ADDRESS OF CURRENT CONTROLLER'S DM EXERCISER COMMUNICATION AREA
: 1561 1      RP_SAVE : VECTOR [MAX_CTLR * RPS_LEN, BYTE, SIGNED],
: 1562 1      !      RETURN PACKET SAVE AREA
: 1563 1      RPS_X1 : WORD,                                ! STARTING INDEX OF CURRENT CONTROLLER'S RP_SAVE AREA
: 1564 1      RPS_X2 : WORD,                                ! ENDING INDEX OF CURRENT CONTROLLER'S RP_SAVE AREA
: 1565 1      OUTC_LIST : VECTOR [MAX_CTLR * OUTC_CNT, BYTE, SIGNED],
: 1566 1      !      OUTSTANDING COMMAND LIST (CONTAINS MSCP ENVELOPE INDECES)
: 1567 1      OUTC_TIMR : VECTOR [MAX_CTLR * OUTC_CNT, WORD],
: 1568 1      !      OUTSTANDING COMMAND TIMERS
: 1569 1      OCL_X1 : WORD,
: 1570 1      !      STARTING INDEX OF CURRENT CONTROLLER'S OUTSTANDING COMMAND AREA
: 1571 1      OCL_X2 : WORD,
: 1572 1      !      ENDING INDEX OF CURRENT CONTROLLER'S OUTSTANDING COMMAND AREA
: 1573 1      TALLY : VECTOR [MAX_UNITS * TALLY_LEN, WORD] FIELD (T_FIELDS),
: 1574 1      !      STATISTICS TABLES
: 1575 1      T_ADDR : REF BLOCK [TALLY_LEN, WORD] FIELD (T_FIELDS),
: 1576 1      !      ADDRESS OF STATISTICS TABLE (TALLY) FOR CURRENT UNIT
: 1577 1      MSCP_ENV : BLOCKVECTOR [ENV_CNT, ENV_LEN, WORD] FIELD (E_FIELDS),
: 1578 1      !      MSCP ENVELOPE POOL
: 1579 1      ENV_USE : VECTOR [ENV_CNT, BYTE, SIGNED],
: 1580 1      !      MSCP ENVELOPE POOL ALLOCATION TABLE
: 1581 1      RETPKT : BLOCKVECTOR [RP_CNT, RP_LEN, WORD] FIELD (RP_FIELDS),
: 1582 1      !      RETURN PACKET POOL
: 1583 1      RP_USE : VECTOR [RP_CNT, BYTE, SIGNED],
: 1584 1      !      RETURN PACKET POOL ALLOCATION TABLE
: 1585 1      RP_INDX : WORD,                                ! CURRENT RETURN PACKET INDEX
: 1586 1      RP_ADDR : REF BLOCK [RP_LEN, WORD] FIELD (RP_FIELDS),
: 1587 1      !      CURRENT RETURN PACKET ADDRESS
```

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


CZRCO2
V02.0

CZRCDB0 RC25 DISK EXERCISER
DECLARATIONS

14-Jun-1985 09:36:36
14-Jun-1985 09:32:03

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRCO2.SRC;B

SEQ 0101
Page 4
(2)

:	1645	1	SWQ13,
:	1646	1	SWQ14,
:	1647	1	SWQ15,
:	1648	1	SWM1,
:	1649	1	MSG_01,
:	1650	1	MSG_07,
:	1651	1	CER_01,
:	1652	1	CER_02,
:	1653	1	CER_03,
:	1654	1	DUM_00,
:	1655	1	DUM_UC,
:	1656	1	DUM_CE,
:	1657	1	DUM_IE,
:	1658	1	DUM_HE,
:	1659	1	DUM_UE,
:	1660	1	EGS_01,
:	1661	1	EGS_02,
:	1662	1	EBS_01,
:	1663	1	EBD_10,
:	1664	1	EBD_12,
:	1665	1	EBD_13,
:	1666	1	EBD_14,
:	1667	1	EBD_15,
:	1668	1	EBD_16,
:	1669	1	EBD_17,
:	1670	1	EBD_18,
:	1671	1	EBD_19,
:	1672	1	EBD_20,
:	1673	1	EBD_21,
:	1674	1	EBD_22,
:	1675	1	EBH_31,
:	1676	1	EBH_32,
:	1677	1	EBH_34,
:	1678	1	EBH_35,
:	1679	1	EBH_36,
:	1680	1	EBH_37,
:	1681	1	EBH_38,
:	1682	1	EBH_39,
:	1683	1	EBH_40,
:	1684	1	EBH_41,
:	1685	1	EBH_42,
:	1686	1	EBH_43,
:	1687	1	EX_SA,
:	1688	1	EX_CRN,
:	1689	1	EX_SC,
:	1690	1	EX_DSC,
:	1691	1	EX_SB,
:	1692	1	EX_CMD,
:	1693	1	EX_SCC,
:	1694	1	EX_ONL,
:	1695	1	EX_ESP,
:	1696	1	EX_SND,
:	1697	1	EX_RD,
:	1698	1	EX_WRT,
:	1699	1	EX_CMP,
:	1700	1	EX_BB,
:	1701	1	EX_LBN,

CZLCD2
V02.0

CZLCD0 RC25 DISK EXERCISER
DECLARATIONS

14-Jun-1985 09:36:36
14-Jun-1985 09:32:03

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZLCD0]CZLCD2.SRC;8

SEQ 0102
Page 5
(2)

```
: 1702 1      EX_CBC,  
: 1703 1      EX_BC,  
: 1704 1      EX_BD,  
: 1705 1      EX_O3,  
: 1706 1      ETIME,  
: 1707 1      PLATT,  
: 1708 1      CRLF,  
: 1709 1      SWP_ERROR : WORD,  
: 1710 1      SWP_XFER  : WORD,  
: 1711 1      SWP_STRACK : WORD,  
: 1712 1      SWP_ETRACK : WORD,  
: 1713 1      SWP_FLAGS : BYTE,  
: 1714 1      L$LUN,  
: 1715 1      L$UNIT;  
  
: 1716 1      %SBTTL 'TYPE AND DESCRIPTION'  
: 1717 1  
: 1718 1      EQUALS;  
: 1719 1  
: 1720 1      !+  
: 1721 1      !      THE TEXT WHICH APPEARS IN THESE MACROS IS DISPLAYED TO THE USER WHEN  
: 1722 1      !      THE DIAGNOSTIC IS FIRST LOADED AND RUN.  
: 1723 1      !-  
: 1724 1  
: 1725 1      DEVTYP (%ASCIZ'SINGLE RC25 PLATTER');           ! "UNIT IS SINGLE RC25 PLATTER"  
: 1726 1  
: 1727 1      DESCRIPT (%ASCIZ'RC25 DISK EXERCISER');       ! TEST DESCRIPTION
```



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

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



```

: 1789 1  *SBTTL 'REPORT CODING SECTION'
: 1790 1
: 1791 1  !*
: 1792 1  !
: 1793 1  !   THE REPORT CODING SECTION CONTAINS THE "PRINTS" CALLS THAT GENERATE
: 1794 1  !   STATISTICAL REPORTS. THIS SECTION IS EXECUTED BY THE OPERATOR COMMAND
: 1795 1  !   "PRINT" OR BY THE PROGRAM MACRO "DORPT".
: 1796 1  !-
: 1797 2  BGNRPT;
: 1798 2
: 1799 2  BIND
: 1800 2  SR_HD = UPLIT ('ASCIZ' '***** SUMMARY REPORT *****'N'),
: 1801 2  SR_ET = UPLIT ('ASCIZ' 'ELAPSED TIME:  'D2'A:'D2'A:'D2'N'),
: 1802 2  SR_MM1 = UPLIT ('ASCIZ' 'A          NO.          NO.          NO.          NO. '),
: 1803 2  SR_MM2 = UPLIT ('ASCIZ' 'AUNIT  READS  WRITES  MBYTES  MBYTES  HARD  ERROR'),
: 1804 2  SR_MM3 = UPLIT ('ASCIZ' 'A NO.  (K=1000) (K=1000)  READ  WRITTEN  ERRORS  LOGS'),
: 1805 2  SR_MM4 = UPLIT ('ASCIZ' 'A-----  -----  -----  -----  -----  -----'N'),
: 1806 2  SR_UN = UPLIT ('ASCIZ' 'N'D3'),
: 1807 2  SR_LO = UPLIT ('ASCIZ' 'D9'A '),
: 1808 2  SR_HI = UPLIT ('ASCIZ' 'D9'AK'),
: 1809 2  SR_ML = UPLIT ('ASCIZ' 'D9'D8'D9'D7'),
: 1810 2  SR_EH1 = UPLIT ('ASCIZ' 'N'N'AN SYMBOL ECC ERRORS:'N'),
: 1811 2  SR_EH2 = UPLIT ('ASCIZ' 'N'A          ECC'),
: 1812 2  SR_EH3 = UPLIT ('ASCIZ' 'N'AUNIT  FIELD  N =   N =   N =   N =   N =   N =   N =   N = '),
: 1813 2  SR_EH4 = UPLIT ('ASCIZ' 'N'A NO.  ONLY  1     2     3     4     5     6     7     8'),
: 1814 2  SR_EH5 = UPLIT ('ASCIZ' 'N'A-----  -----  -----  -----  -----  -----  -----'N'),
: 1815 2  SR_EL1 = UPLIT ('ASCIZ' 'N'D3'D8'D7'D7'D7'),
: 1816 2  SR_EL2 = UPLIT ('ASCIZ' 'D7'D7'D7'D7'D7'),
: 1817 2  SR_DH1 = UPLIT ('ASCIZ' 'N'A          NO.          NO.          NO. '),
: 1818 2  SR_DH2 = UPLIT ('ASCIZ' 'N'AUNIT  READS  WRITES  SEEKS  MBYTES  MBYTES  NO. HARD  NO. SOFT'),
: 1819 2  SR_DH3 = UPLIT ('ASCIZ' 'N'A NO.  (K=1000) (K=1000) (K=1000)  READ  WRITTEN  ERRORS  ERRORS'),
: 1820 2  SR_DH4 = UPLIT ('ASCIZ' 'N'A-----  -----  -----  -----  -----  -----  -----'N'),
: 1821 2  SR_DL = UPLIT ('ASCIZ' 'D9'D8'D10'D10');
: 1822 2
: 1823 2  OWN
: 1824 2  ADDR_LO;          ! ADDR OF LOW-ORDER FIELD OF I/O COUNT
: 1825 2
: 1826 2  ROUTINE HI_LO : NOVALUE =
: 1827 2
: 1828 2  !*
: 1829 2  !
: 1830 2  !   THIS ROUTINE IS CALLED FROM THE MAINLINE CODE OF THE REPORT CODING
: 1831 2  !   SECTION WHICH APPEARS BELOW. ITS PURPOSE IS TO PRINT THE HIGH-ORDER
: 1832 2  !   FIELD OF ONE OF THE I/O COUNTS IF THE FIELD IS NON-ZERO. OTHERWISE, THE
: 1833 2  !   LOW-ORDER FIELD IS PRINTED.
: 1834 2  !
: 1835 2  !   IMPLICIT INPUTS:
: 1836 2  !   ADDR_LO - ADDRESS OF THE LOW-ORDER FIELD OF AN I/O COUNT
: 1837 2  !-
: 1838 3  BEGIN
: 1839 3
: 1840 3  LOCAL
: 1841 3  ADDR_HI;          ! ADDR OF HIGH-ORDER FIELD OF I/O COUNT
: 1842 3
: 1843 3  ADDR_HI = .ADDR_LO + 2;  ! GET HIGH-ORDER FIELD ADDRESS
: 1844 3  IF .ADDR_HI EQLU 0      ! IF HIGH-ORDER FIELD IS ZERO
: 1845 3  THEN                  ! THEN

```



```

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
                                .WORD SWM1
000254 000000G        GP$15:: .WORD 4130
000256 004130        .WORD SWQ9
000260 000000G        .WORD 20
000262 000020        GP$16:: .WORD 4130
000264 004130        .WORD SWQ10
000266 000000G        .WORD 40
000270 000040        $SW1: .WORD <<<<$LSW1-$SW1>*400>+4>+20>
000272 000000C        GP$17:: .WORD 4130
000274 004130        .WORD SWQ11
000276 000000G        .WORD 100
000300 000100        $LSW1: .WORD 1004
000302 001004        GP$18:: .WORD 4130
000304 004130        .WORD SWQ12
000306 000000G        .WORD 200
000310 000200        .WORD

```

```

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	P.AAN:	.ASCII	/=/<00><00>
001002	045	116	045		.ASCII	/N%/
001005	101	040	116		.ASCII	/A N/
001010	117	056	040		.ASCII	/0. /
001013	040	040	117		.ASCII	/ 0/
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 / /

CZRC D2
V02.0

CZRCDB0 RC25 DISK EXERCISER
REPORT CODING SECTION

14-Jun-1985 09:36:36
14-Jun-1985 09:32:03

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC D2.SRC;8

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	051	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
001640	055	055	055
001643	055	055	055
001646	055	040	040
001651	055	055	055
001654	055	055	055
001657	055	055	040
001662	040	055	055
001665	055	055	055
001670	055	055	055
001673	045	116	000
001676	045	104	071
001701	045	104	070
001704	045	104	061
001707	060	045	104
001712	061	060	000
001715	000		

P.AAV:

```

.ASCII /- /
.ASCII /---/
.ASCII /---/
.ASCII /- /
.ASCII /---/
.ASCII /---/
.ASCII /-- /
.ASCII / --/
.ASCII /---/
.ASCII /---/
.ASCII /%N/<00>
.ASCII /%D9/
.ASCII /%D8/
.ASCII /%D1/
.ASCII /O%D/
.ASCII /10/<00>
.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

```

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

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

```

```

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

```

```

; 1851 2
; 1852 2   !!!!!!!!!!!!!!!!!!!!! REPORT CODING SECTION - MAINLINE CODE !!!!!!!!!!!!!!!!!!!!!
; 1853 2
; 1854 2   LOCAL
; 1855 2   TA : REF BLOCK [TALLY_LEN] FIELD (T_FIELDS);           ! ADDRESS OF A UNIT'S TALLY BLOCK
; 1856 2
; 1857 2   PRINTS (SR_HD);           ! SUMMARY REPORT HEADER
; 1858 2   PRINTS (SR_ET, .HOURS, .MINUTES, .SECONDS);           ! ELAPSED TIME
; 1859 3   IF BIT_TST (SWP_FLAGS, SWF_DM)           ! IF DM EXERCISER IS BEING RUN
; 1860 2   THEN           ! THEN

```



```
: 1861 3      BEGIN
: 1862 3
: 1863 3      PRINTS (SR_DH1);          ! PRINT HEADER FOR DM EXERCISER STATS
: 1864 3      PRINTS (SR_DH2);
: 1865 3      PRINTS (SR_DH3);
: 1866 3      PRINTS (SR_DH4);
: 1867 3      INCR UNIT FROM 0 TO (.L$UNIT - 1) DO      ! FOR EACH UNIT
: 1868 4          BEGIN
: 1869 4
: 1870 4          TA = TALLY + (.UNIT * TALLY_LEN * 2);  ! CALCULATE UNIT'S TALLY BLOCK ADDR
: 1871 4          PRINTS (SR_UN, .UNIT);                ! PRINT UNIT NUMBER
: 1872 4
: 1873 4          ADDR_LO = TA [READ_LO];              ! SET LOW-ORDER ADDR OF READ COUNT
: 1874 4          HI_LO ();                            ! PRINT EITHER LOW- OR HIGH-ORDER FIELD
: 1875 4          ADDR_LO = .ADDR_LO + 4;              ! ADVANCE TO LOW-ORDER ADDR OF WRITE COUNT
: 1876 4          HI_LO ();                            ! PRINT EITHER LOW- OR HIGH-ORDER FIELD
: 1877 4          ADDR_LO = .ADDR_LO + 4;              ! ADVANCE TO LOW-ORDER ADDR OF SEEK COUNT
: 1878 4          HI_LO ();                            ! PRINT EITHER LOW- OR HIGH-ORDER FIELD
: 1879 4
: 1880 4          PRINTS (SR_DL, .TA [MB_READ], .TA [MB_WRIT], .TA [ER_HRD], .TA [ER_SFT]);
: 1881 4
: 1882 3      END;                                ! UNIT LOOP
: 1883 3
: 1884 3      END
: 1885 2      ELSE                                ! ELSE - MULTI-DRIVE SUBTEST IS BEING RUN
: 1886 3      BEGIN
: 1887 3
: 1888 3      PRINTS (SR_MH1);          ! PRINT HEADER FOR MULTI-DRIVE
: 1889 3      PRINTS (SR_MH2);          ! SUBTEST BASIC STATS
: 1890 3      PRINTS (SR_MH3);
: 1891 3      PRINTS (SR_MH4);
: 1892 3      INCR UNIT FROM 0 TO (.L$UNIT - 1) DO      ! FOR EACH UNIT
: 1893 4          BEGIN
: 1894 4
: 1895 4          TA = TALLY + (.UNIT * TALLY_LEN * 2);  ! CALCULATE UNIT'S TALLY BLOCK ADDR
: 1896 4          PRINTS (SR_UN, .UNIT);                ! PRINT UNIT NUMBER
: 1897 4
: 1898 4          ADDR_LO = TA [READ_LO];              ! SET LOW-ORDER ADDR OF READ COUNT
: 1899 4          HI_LO ();                            ! PRINT EITHER LOW- OR HIGH-ORDER FIELD
: 1900 4          ADDR_LO = .ADDR_LO + 4;              ! ADVANCE TO LOW-ORDER ADDR OF WRITE COUNT
: 1901 4          HI_LO ();                            ! PRINT EITHER LOW- OR HIGH-ORDER FIELD
: 1902 4
: 1903 4          PRINTS (SR_ML, .TA [MB_READ], .TA [MB_WRIT], .TA [ER_HRD], .TA [ER_LOG]);
: 1904 4
: 1905 3      END;                                ! UNIT LOOP
: 1906 3
: 1907 3      PRINTS (SR_EH1);          ! PRINT HEADER FOR ECC ERROR STATS
: 1908 3      PRINTS (SR_EH2);
: 1909 3      PRINTS (SR_EH3);
: 1910 3      PRINTS (SR_EH4);
: 1911 3      PRINTS (SR_EH5);
: 1912 3      INCR UNIT FROM 0 TO (.L$UNIT - 1) DO      ! FOR EACH UNIT
: 1913 4          BEGIN
: 1914 4
: 1915 4          TA = TALLY + (.UNIT * TALLY_LEN * 2);  ! CALCULATE UNIT'S TALLY BLOCK ADDR
: 1916 4          PRINTS (SR_EL1, .UNIT, .TA [ECC_ONLY], .TA [ECC_1], .TA [ECC_2], .TA [ECC_3]);
: 1917 4          PRINTS (SR_EL2, .TA [ECC_4], .TA [ECC_5], .TA [ECC_6], .TA [ECC_7], .TA [ECC_8]);
```

: 1918 4
: 1919 3
: 1920 3
: 1921 2
: 1922 2
: 1923 2
: 1924 2
: 1925 1

END;
END;
PRINTS (CRLF);
ENDRPT;

! UNIT LOOP

! PRINT <CR><LF>

Address	Offset	OpCode	Comment	Address
000000	004137	000000G	LRPT: .SBTTL LRPT REPORT CODING SECTION	
000004	012746	000000'	JSR R1,\$SAVE3	1788
000010	012746	000001	MOV #SR.HD,-(SP)	1857
000014	010600		MOV #1,-(SP)	
000016	104416		MOV SP,R0	: SP,*
000020	013716	000000G	TRAP 16	
000024	013746	000000G	MOV SECONDS,(SP)	: 1858
000030	013746	000000G	MOV MINUTES,-(SP)	
000034	012746	000116'	MOV HOURS,-(SP)	
000040	012746	000004	MOV #SR.ET,-(SP)	
000044	010600		MOV #4,-(SP)	
000046	104416		MOV SP,R0	: SP,*
000050	132737	000002 000000G	TRAP 16	
000056	001516		BITB #2,SWP.FLAGS	: 1859
000060	012716	001264'	BEQ 3\$	
000064	012746	000001	MOV #SR.DH1,(SP)	: 1863
000070	010600		MOV #1,-(SP)	
000072	104416		MOV SP,R0	: SP,*
000074	012716	001330'	TRAP 16	
000100	012746	000001	MOV #SR.DH2,(SP)	: 1864
000104	010600		MOV #1,-(SP)	
000106	104416		MOV SP,R0	: SP,*
000110	012716	001444'	TRAP 16	
000114	012746	000001	MOV #SR.DH3,(SP)	: 1865
000120	010600		MOV #1,-(SP)	
000122	104416		MOV SP,R0	: SP,*
000124	012716	001560'	TRAP 16	
000130	012746	000001	MOV #SR.DH4,(SP)	: 1866
000134	010600		MOV #1,-(SP)	
000136	104416		MOV SP,R0	: SP,*
000140	013703	000000G	TRAP 16	
000144	005002		MOV L\$UNIT,R3	: 1867
000146	000456		CLR R2	: UNIT
000150	010216		BR 2\$	
000152	012746	000060	1\$: MOV R2,(SP)	: UNIT,* 1870
000156	004737	000000G	MOV #60,-(SP)	
000162	062700	000000G	JSR PC,BL\$MUL	
000166	010001		ADD #TALLY,R0	
000170	010216		MOV R0,R1	: *,TA
000172	012746	000550'	MOV R2,(SP)	: UNIT,* 1871
000176	012746	000002	MOV #SR.UN,-(SP)	
000202	010600		MOV #2,-(SP)	
000204	104416		MOV SP,R0	: SP,*
000206	010137	000366'	TRAP 16	
000212	004737	000370'	MOV R1,ADDR.LO	: TA,* 1873
000216	062737	000004 000366'	JSR PC,HI.LO	: 1874
			ADD #4,ADDR.LO	: 1875

CZRC02
V02.0

CZRCDB0 RC25 DISK EXERCISER
REPORT CODING SECTION

14-Jun-1985 09:36:36
14-Jun-1985 09:32:03

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC02.SRC;8

SEQ 0119
Page 23
(6)

000224	004737	000370'		JSR	PC,HI.LO	:	1876
000230	052737	000004	000366'	ADD	#4,ADDR.LO	:	1877
000236	004737	000370'		JSR	PC,HI.LO	:	1878
000242	016116	000034		MOV	34(R1),(SP)	: *(TA),*	1880
000246	016146	000030		MOV	30(R1),-(SP)	: *(TA),*	
000252	016146	000026		MOV	26(R1),-(SP)	: *(TA),*	
000256	016146	000020		MOV	20(R1),-(SP)	: *(TA),*	
000262	012746	001676'		MOV	#SR.DL, -(SP)		
000266	012746	000005		MOV	#5, -(SP)		
000272	010600			MOV	SP,RO	: SP,*	
000274	104416			TRAP	16		
000276	062706	000020		ADD	#20,SP	:	1868
000302	005202			INC	R2	: UNIT	1867
000304	020203		2\$:	CMP	R2,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,RO	: SP,*	
000326	104416			TRAP	16		
000330	012716	000256'		MOV	#SR.MH2,(SP)	:	1889
000334	012746	000001		MOV	#1, -(SP)		
000340	010600			MOV	SP,RO	: SP,*	
000342	104416			TRAP	16		
000344	012716	000354'		MOV	#SR.MH3,(SP)	:	1890
000350	012746	000001		MOV	#1, -(SP)		
000354	010600			MOV	SP,RO	: SP,*	
000356	104416			TRAP	16		
000360	012716	000450'		MOV	#SR.MH4,(SP)	:	1891
000364	012746	000001		MOV	#1, -(SP)		
000370	010600			MOV	SP,RO	: SP,*	
000372	104416			TRAP	16		
000374	013703	000000G		MOV	L\$UNIT,R3	:	1892
000400	005002			CLR	R2	: UNIT	
000402	000451			BR	5\$		
000404	010216		4\$:	MOV	R2,(SP)	: UNIT,*	1895
000406	012746	000060		MOV	#60, -(SP)		
000412	004737	000000G		JSR	PC,BL\$MUL		
000416	062700	000000G		ADD	#TALLY,RO		
000422	010001			MOV	RO,R1	: *,TA	
000424	010216			MOV	R2,(SP)	: UNIT,*	1896
000426	012746	000550'		MOV	#SR.UN, -(SP)		
000432	012746	000002		MOV	#2, -(SP)		
000436	010600			MOV	SP,RO	: SP,*	
000440	104416			TRAP	16		
000442	010137	000366'		MOV	R1,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	016116	000032		MOV	32(R1),(SP)	: *(TA),*	1903
000470	016146	000030		MOV	30(R1),-(SP)	: *(TA),*	
000474	016146	000026		MOV	26(R1),-(SP)	: *(TA),*	
000500	016146	000020		MOV	20(R1),-(SP)	: *(TA),*	
000504	012746	000576'		MOV	#SR.ML, -(SP)		
000510	012746	000005		MOV	#5, -(SP)		
000514	010600			MOV	SP,RO	: SP,*	
000516	104416			TRAP	16		

CZRC02
V02.0

CZRCDB0 RC25 DISK EXERCISER
REPORT CODING SECTION

14-Jun-1985 09:36:36
14-Jun-1985 09:32:03

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC02.SRC;8

SEQ 0120
Page 24
(6)

000520	062706	000020		ADD	#20,SP	:	1893
000524	005202			INC	R2	: UNIT	1892
000526	020203		5\$:	CMP	R2,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	005002			CLR	R2	: UNIT	
000634	000452			BR	7\$		
000636	010216		6\$:	MOV	R2,(SP)	: UNIT,*	1915
000640	012746	000060		MOV	#60,-(SP)		
000644	004737	000000G		JSR	PC,BL\$MUL		
000650	062700	000000G		ADD	#TALLY,R0		
000654	010001			MOV	R0,R1	: *,TA	
000656	016116	000042		MOV	42(R1),(SP)	: *(TA),*	1916
000662	016146	000040		MOV	40(R1),-(SP)	: *(TA),*	
000666	016146	000036		MOV	36(R1),-(SP)	: *(TA),*	
000672	016146	000056		MOV	56(R1),-(SP)	: *(TA),*	
000676	010246			MOV	R2,-(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	016116	000054		MOV	54(R1),(SP)	: *(TA),*	1917
000720	016146	000052		MOV	52(R1),-(SP)	: *(TA),*	
000724	016146	000050		MOV	50(R1),-(SP)	: *(TA),*	
000730	015146	000046		MOV	46(R1),-(SP)	: *(TA),*	
000734	016146	000044		MOV	44(R1),-(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	005202			INC	R2	: UNIT	1912
000762	020203		7\$:	CMP	R2,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)		

CZRC D2
V02.0

CZRCDB0 RC25 DISK EXERCISER
REPORT CODING SECTION

14-Jun-1985 09:36:36
14-Jun-1985 09:32:03

VAX-11 Bliss-16 V4.0-579
USER\$1:[AZTEC.CZRCDB]CZRC D2.SRC;8

SEQ 0121
Page 25
(6)

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

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

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

```
: 1926 1 *SBTTL 'INITIALIZE SECTION'
: 1927 1
: 1928 1 !+
: 1929 1 !
: 1930 1 ! THE INITIALIZE CODE IS EXECUTED UNDER FIVE CONDITIONS. THERE ARE
: 1931 1 ! SUPERVISOR EVENT FLAGS THAT ARE USED TO LET THE DIAGNOSTIC KNOW UNDER
: 1932 1 ! WHICH CONDITION THE EXECUTION IS TAKING PLACE. THE EVENT FLAGS ARE
: 1933 1 ! READ USING THE "READEF" MACRO. THE CONDITIONS UNDER WHICH THE INIT CODE
: 1934 1 ! IS EXECUTED AND THE CORRESPONDING EVENT FLAGS ARE:
: 1935 1 ! START COMMAND EF.START
: 1936 1 ! RESTART COMMAND EF.RESTART
: 1937 1 ! CONTINUE COMMAND EF.CONTINUE
: 1938 1 ! POWERDOWN/POWERUP EF.PWR
: 1939 1 ! NEW PASS EF.NEW
: 1940 1 ! EXAMPLE OF EVENT FLAG USE:
: 1941 1 ! IF READEF(EF.START) THEN START_FLAG = 1;
: 1942 1 ! DURING THE INIT CODE, THE "GPHARD" MACRO IS USED TO OBTAIN P-TABLE
: 1943 1 ! INFORMATION FOR ALL DEVICES. THE NUMBER OF UNITS AVAILABLE IS IN THE
: 1944 1 ! HEADER WORD "L$UNIT".
: 1945 1 !-
: 1946 2 BGNINIT;
: 1947 2
: 1948 2 LOCAL
: 1949 2 FLAG1 : BYTE,
: 1950 2 FLAG2 : BYTE,
: 1951 2 DELAY_MULT : WORD,
: 1952 2 CLK_ADR,
: 1953 2 HWPT_REF : REF BLOCK [HWPT_LEN, WORD] FIELD (HWP_FIELDS);
: 1954 2
: 1955 2 SETPRI (PRI07); ! PRIORITY 7 - NO INTERRUPTS ALLOWED DURING INIT
: 1956 2
: 1957 2 IF READEF (EF_NEW) ! IS THIS A NEW PASS? (ASK NEW_PASS
: 1958 2 THEN ! FIRST IN CASE MULTIPLE FLAGS ARE
: 1959 2 ENTRY_REASON = NEW_PASS; ! SET, E.G., START AND NEW_PASS)
: 1960 2
: 1961 2 IF READEF (EF_CONTINUE) ! IS THIS A CONTINUE?
: 1962 2 THEN
: 1963 2 ENTRY_REASON = CONT;
: 1964 2
: 1965 2 IF READEF (EF_PWR) ! IS THIS A POWER FAIL?
: 1966 2 THEN
: 1967 3 BEGIN
: 1968 3 ENTRY_REASON = PWR_FAIL;
: 1969 4 IF MANUAL ! IF ATTENDED
: 1970 3 THEN ! THEN
: 1971 3 PRINTF (MSG_01); ! "POWER DELAY - WAITING"
: 1972 3
: 1973 3 INCR COUNT FROM 0 TO 60 DO ! WAIT APPROX. 60 SECNDS
: 1974 4 BEGIN
: 1975 4 DELAY_MULT = 400; ! Changed from 333 to 400 for 11/84 cache
: 1976 4 DELAY (.DELAY_MULT);
: 1977 4 BREAK; ! BREAK FOR ACT
: 1978 3 END;
: 1979 3
: 1980 2 END;
: 1981 2
: 1982 2 IF READEF (EF_RESTART) ! IS THIS A RESTART?
```



```
: 1983 2 THEN
: 1984 2 ENTRY_REASON = RESTART;
: 1985 2
: 1986 2 IF READEF (EF_START) ! IS THIS A START?
: 1987 2 THEN
: 1988 3 BEGIN
: 1989 3 ENTRY_REASON = START;
: 1990 3 HOURS = MINUTES = SECONDS = TICKS = 0; ! INIT ELAPSED TIME TO 00:00:00:00
: 1991 2 END;
: 1992 2
: 1993 2 !+
: 1994 2 ! MAKE SURE THAT NOT MORE THAN MAX_UNITS HAVE BEEN SPECIFIED.
: 1995 2 ! IF THERE ARE TOO MANY, NOTIFY USER AND RETURN TO SUPERVISOR.
: 1996 2 ! (DIAGNOSTIC IS ABORTED).
: 1997 2 !-
: 1998 2
: 1999 2 IF .L$UNIT GTRU MAX_UNITS
: 2000 2 THEN
: 2001 3 BEGIN
: 2002 3 ERRSF (1, EGS_01, EMS_01); ! "TOO MANY UNITS"
: 2003 3 DOCLN;
: 2004 2 END;
: 2005 2
: 2006 2 !+
: 2007 2 ! CHECK TO MAKE SURE THERE IS A CLOCK ON THE SYSTEM. IF NO_CLOCK,
: 2008 2 ! THEN ABORT TO SUPERVISOR. OTHERWISE, DETERMINE WHETHER CLOCK IS
: 2009 2 ! AN L OR P CLOCK, AND SET UP PARAMETERS.
: 2010 2 !-
: 2011 2
: 2012 2 CLK_TYPE = NO_CLOCK; ! SET FLAG FOR NO CLOCK
: 2013 2
: 2014 3 IF CLOCK (P, CLK_ADR) ! IS THERE A P_CLOCK?
: 2015 2 THEN
: 2016 3 BEGIN
: 2017 3 CLK_TYPE = P_CLOCK; ! SET TYPE TO P_CLOCK
: 2018 3 CLK_CSR = ..CLK_ADR; ! SAVE THE CSR ADDRESS
: 2019 3 .CLK_CSR = #0'105'; ! ENABLE INTERRUPTS
: 2020 3 END
: 2021 2 ELSE
: 2022 3 BEGIN
: 2023 3
: 2024 4 IF CLOCK (L, CLK_ADR) ! IS THERE AN L_CLOCK?
: 2025 3 THEN
: 2026 4 BEGIN
: 2027 4 CLK_TYPE = L_CLOCK; ! SET TYPE TO L_CLOCK
: 2028 4 CLK_CSR = ..CLK_ADR; ! SAVE THE CSR ADDRESS
: 2029 4 .CLK_CSR = #0'100'; ! ENABLE INTERRUPTS
: 2030 3 END;
: 2031 3
: 2032 2 END;
: 2033 2
: 2034 2 IF .CLK_TYPE EQLU NO_CLOCK ! IF NO CLOCK WAS FOUND
: 2035 2 THEN
: 2036 3 BEGIN
: 2037 3 ERRSF (2, EGS_02, 0); ! "NEITHER P NOR L CLOCK WAS FOUND"
: 2038 3 DOCLN; ! ABORT THE PROGRAM
: 2039 3 END
```

```

: 2040 2 ELSE
: 2041 3 BEGIN
: 2042 3 CLK_VECTOR = .(.CLK_ADR + 4); ! CLOCK VECTOR ADDRESS
: 2043 3 CLK_HERTZ = .(.CLK_ADR + 6); ! CLOCK HERTZ RATE
: 2044 3 SETVEC (.CLK_VECTOR, CLK_INT_SERV, PRI06); ! SET CLOCK VECTOR ADDR
: 2045 2 END;
: 2046 2
: 2047 2 !+
: 2048 2 ! THE FOLLOWING CODE IS EXECUTED FOR ALL ENTRY REASONS EXCEPT NEW_PASS.
: 2049 2 ! ALL RUN-TIME CONTROLLER STATUS TABLES (CST'S) ARE CLEARED TO 0, THEN
: 2050 2 ! LOADED WITH CONFIGURATION DATA FROM THE HARDWARE P-TABLES.
: 2051 2 !-
: 2052 2
: 2053 2 IF .ENTRY_REASON NEQU NEW_PASS
: 2054 2 THEN
: 2055 3 BEGIN
: 2056 3
: 2057 3 INCR COUNT FROM 0 TO ((MAX_CTLR * CST_LEN * 2) - 2) BY 2 DO
: 2058 3 (CST + .COUNT) = 0;
: 2059 3
: 2060 3 INCR UNIT FROM 0 TO (.L$UNIT - 1) DO ! LOOP THROUGH ALL UNITS
: 2061 4 BEGIN
: 2062 4
: 2063 4 CPT [.UNIT] = NO; ! INIT CURRENT PASS TESTING VECTOR
: 2064 4 IF GPHARD (.UNIT, HWPT_REF) NEQA 0 ! IF HWP TABLE FOUND
: 2065 4 THEN
: 2066 5 BEGIN
: 2067 5
: 2068 5 FLAG1 = NOT_FOUND; ! NO EXISTING IP ADDRESS MATCH YET
: 2069 5 INCR CTLR FROM 0 TO (MAX_CTLR - 1) DO ! LOOP THROUGH ALL CST'S
: 2070 6 BEGIN
: 2071 6
: 2072 6 IF .CST [.CTLR, IP_ADDR] EQLA .HWPT_REF [HWP_IP_ADDR]
: 2073 6 THEN ! IF IP ADDR ALREADY EXISTS
: 2074 7 BEGIN
: 2075 7
: 2076 7 FLAG1 = FOUND;
: 2077 7 FLAG2 = NOT_FOUND; ! FLAG INDICATING PLATTER SLOT AVAILABILITY
: 2078 7 INCR OFFSET FROM (0 + OF_UN) TO (3 + OF_UN) DO ! LOOP THROUGH EACH PLATTER SLOT
: 2079 8 BEGIN
: 2080 8
: 2081 8 IF .CST [.CTLR, .OFFSET, P_PRES] EQLU NOT_PRESENT
: 2082 8 THEN ! IF EMPTY SLOT FOUND
: 2083 9 BEGIN
: 2084 9
: 2085 9 FLAG2 = FOUND;
: 2086 9 CST [.CTLR, .OFFSET, ALLBIT] = .HWPT_REF [HWP_PLAT];
: 2087 9 ! COPY PLATTER ADDR AND PROT BIT
: 2088 9 CST [.CTLR, .OFFSET, P_UNIT] = .UNIT;
: 2089 9 CST [.CTLR, .OFFSET, P_PRES] = PRESENT;
: 2090 9 EXITLOOP;
: 2091 9
: 2092 9 END ! END - IF PLATTER SLOT FOUND
: 2093 8 ELSE ! OTHERWISE - SLOT IS OCCUPIED
: 2094 9 BEGIN
: 2095 9
: 2096 9 IF .CST [.CTLR, .OFFSET, P_ADDR] EQLU .HWPT_REF [HWP_PLAT_ADDR]
```



```
: 2154 8
: 2155 7
: 2156 7
: 2157 6
: 2158 6
: 2159 6
: 2160 6
: 2161 7
: 2162 7
: 2163 8
: 2164 7
: 2165 7
: 2166 7
: 2167 7
: 2168 7
: 2169 6
: 2170 6
: 2171 5
: 2172 5
: 2173 4
: 2174 4
: 2175 3
: 2176 3
: 2177 3
: 2178 2
: 2179 3
: 2180 3
: 2181 3
: 2182 4
: 2183 4
: 2184 4
: 2185 4
: 2186 5
: 2187 5
: 2188 5
: 2189 5
: 2190 6
: 2191 6
: 2192 6
: 2193 6
: 2194 7
: 2195 7
: 2196 7
: 2197 7
: 2198 7
: 2199 6
: 2200 6
: 2201 5
: 2202 5
: 2203 4
: 2204 4
: 2205 3
: 2206 3
: 2207 2
: 2208 2
: 2209 2
: 2210 2

      END;
      ! IF EMPTY CST FOUND
      END;
      ! EMPTY CST SEARCH LOOP
      IF .FLAG2 EQLU NOT_FOUND
      THEN
      BEGIN
      ! IF NO EMPTY CST FOUND
      IF MANUAL
      THEN
      BEGIN
      ! IF ATTENDED
      ! THEN
      PRINTF (CER_03, MAX_CTLR); ! "MORE THAN X DIFFERENT IP ADDRESSES."
      DUR [.UNIT] = DU_CONF; ! CONFIGURATION ERROR
      DODU (.UNIT); ! DROP UNIT
      END;
      END;
      ! IF NO IP ADDR MATCH IN CST
      END;
      ! IF GPHARD RETURNED A HWP TABLE
      END;
      ! UNIT LOOP
      END
      ELSE
      BEGIN
      ! OTHERWISE, FOR EACH NEW PASS
      INCR UNIT FROM 0 TO (.L$UNIT - 1) DO
      BEGIN
      ! FOR EACH UNIT
      IF GPHARD (.UNIT, HWPT_REF) NEQA 0
      THEN
      BEGIN
      ! IF UNIT HAS NOT BEEN DROPPED
      ! THEN
      CPT [.UNIT] = YES;
      INCR CTLR FROM 0 TO (MAX_CTLR - 1) DO
      BEGIN
      ! O.K. TO TEST UNIT
      ! FIND UNIT'S CONTROLLER
      IF .CST [.CTLR, IP_ADDR] EQLA .HWPT_REF [HWP_IP_ADDR]
      THEN
      BEGIN
      ! FOUND IT
      CST [.CTLR, U_CNT] = .CST [.CTLR, U_CNT] + 1;
      EXITLOOP;
      ! INCREMENT NO. OF TESTABLE UNITS
      ! DONE
      END;
      END;
      ! CONTROLLER LOOP
      END;
      ! IF UNIT NOT DROPPED
      END;
      ! UNIT LOOP
      END;
      ! END - IF NEW PASS
      IF .ENTRY_REASON LEQU RESTART
      THEN
      ! IF START OR RESTART
      ! THEN
```



```
: 2211 3      BEGIN
: 2212 3
: 2213 3      CRN = 0;                ! COMMAND REFERENCE NUMBER
: 2214 3      CTLR_CNT = 0;          ! NUMBER OF CONFIGURED CONTROLLERS
: 2215 3      INCR CTLR FROM 0 TO (MAX_CTLR - 1) DO
: 2216 4          BEGIN
: 2217 4
: 2218 4          IF .CST [.CTLR, IP_ADDR] NEQA 0      ! IF CONTROLLER IS PRESENT
: 2219 4          THEN                                ! THEN
: 2220 4              CTLR_CNT = .CTLR_CNT + 1;        ! INCREMENT CONTROLLER COUNT
: 2221 4
: 2222 3      END;
: 2223 3
: 2224 3      MEMORY (FREE_MEM_ADDR);                ! GET START OF FREE MEMORY
: 2225 3      MEM_SIZE = ..FREE_MEM_ADDR;           ! GET FREE MEMORY SIZE
: 2226 3
: 2227 3      INCR COUNT FROM 0 TO ((MAX_UNITS * TALLY_LEN) - 1) DO      ! INITIALIZE
: 2228 3          TALLY [.COUNT] = 0;                ! STATISTICS
: 2229 3
: 2230 2      END;                                ! END OF START/RESTART INITIALIZATION
: 2231 2
: 2232 2      !+
: 2233 2      ! MISCELLANEOUS INITIALIZATON
: 2234 2      !-
: 2235 2
: 2236 2      INCR CTLR FROM 0 TO (MAX_CTLR - 1) DO      ! INITIALIZE NO. OF OUTSTANDING QIOS
: 2237 2          QIO [.CTLR] = 0;
: 2238 2      INCR COUNT FROM 0 TO (RP_CNT - 1) DO      ! INITIALIZE RETURN PACKET POOL
: 2239 2          RP_USE [.COUNT] = -1;
: 2240 2      IODQ_IN = IODQ_OUT = 0;                ! INITIALIZE I/O DONE QUEUE POINTERS
: 2241 3      IF ((BIT_TST (SWP_FLAGS, SWF_HWC)) AND    ! IF USER CHANGED FROM HOST WRITE-CHECKS TO
: 2242 3          (BIT_TST (SWP_FLAGS, SWF_CWC)))        ! CONTROLLER WRITE-CHECKS
: 2243 2      THEN                                    ! THEN (BOTH BITS WOULD BE SET)
: 2244 2          SWP_FLAGS <SWF_PHWC,1> = 0;          ! CLEAR HOST WRITE-CHECKS
: 2245 2
: 2246 2      !+
: 2247 2      ! THE FOLLOWING SITUATION (STARTING TRACK NUMBER GREATER THAN ENDING
: 2248 2      ! TRACK NUMBER) CAN OCCUR DUE TO ANOTHER DRS BUG. IF THE USER ENTERS LOW
: 2249 2      ! TRACK LIMITS FOR ONE PASS, THEN ENTERS A HIGH STARTING TRACK NUMBER
: 2250 2      ! AFTER A RESTART OR CONTINUE COMMAND, DRS WILL NOT GIVE AN ERROR IF THE
: 2251 2      ! ENDING TRACK NUMBER IS DEFAULTED TO THE PREVIOUS LOW TRACK NUMBER. AN
: 2252 2      ! ERRONEOUS LBN WOULD BE GENERATED IN QIO_LBN IF THIS SITUATION WERE
: 2253 2      ! ALLOWED.
: 2254 2      !-
: 2255 2
: 2256 2      IF .SWP_STRACK GEQU .SWP_ETRACK
: 2257 2      THEN
: 2258 3          BEGIN
: 2259 3
: 2260 3          LOCAL
: 2261 3              TEMP : WORD;
: 2262 3
: 2263 3          TEMP = .SWP_STRACK;                ! REVERSE STARTING AND ENDING
: 2264 3          SWP_STRACK = .SWP_ETRACK;          ! TRACK NUMBERS
: 2265 3          SWP_ETRACK = .TEMP;
: 2266 3
: 2267 2      END;
```

```

; 2268 2
; 2269 2   SETPRI (PRI00);
; 2270 2
; 2271 1   ENDINIT;
    
```

! SET PROGRAM PRIORITY TO 0

```

                                .GLOBL L$DLY
                                .SBTTL LINIT INITIALIZE SECTION
000000 004137 000000G          LINIT: JSR      R1,$SAVE5           ;           1925
000004 162706 000010          SUB      #10,SP              ;           1955
000010 012700 000340          MOV      #340,R0           ;           1957
000014 104441                    TRAP    41                  ;           1959
000016 012700 000035          MOV      #35,R0           ;           1961
000022 104447                    TRAP    47                  ;           1963
000024 103003                    BHS     1$                  ;           1965
000026 112737 000005 000000G  MOVB    #5,ENTRY.REASON   ;           1968
000034 012700 000036          1$:    MOV      #36,R0           ;           1969
000040 104447                    TRAP    47                  ;           1971
000042 103003                    BHS     2$                  ;           1973
000044 112737 000003 000000G  MOVB    #3,ENTRY.REASON   ;           1975
000052 012700 000034          2$:    MOV      #34,R0           ;           1976
000056 104447                    TRAP    47                  ;           1977
000060 103036                    BHS     9$                  ;           1978
000062 112737 000004 000000G  MOVB    #4,ENTRY.REASON   ;           1979
000070 104450                    TRAP    50                  ;           1980
000072 103007                    BHS     3$                  ;           1981
000074 012746 000000G          MOV      #MSG.01,-(SP)    ;           1982
000100 012746 000001          MOV      #1,-(SP)        ;           1983
000104 010600                    MOV     SP,R0              ; SP,*
000106 104417                    TRAP    17                  ;           1984
000110 022626                    CMP     (SP)+,(SP)+      ;           1985
000112 012702 000075          3$:    MOV      #75,R2           ; *,COUNT           1986
000116 012703 000620          4$:    MOV      #620,R3        ; *,DELAY.MULT       1987
000122 010301                    MOV     R3,R1              ; DELAY.MULT,$$TMP2  1988
000124 001411                    5$:    BEQ     8$                  ;           1989
000126 013700 000000G          MOV     L$DLY,R0         ; *,$$TMP1           1990
000132 001404                    BEQ    7$                  ;           1991
000134 005066 000006          6$:    CLR     6(SP)           ; $$TMP               1992
000140 005300                    DEC     R0                  ; $$TMP1             1993
000142 001374                    BNE    6$                  ;           1994
000144 005301                    7$:    DEC     R1                  ; $$TMP2             1995
000146 000766                    BR     5$                  ;           1996
000150 104422                    8$:    TRAP   22              ;           1997
000152 005302                    DEC     R2                  ; COUNT              1998
000154 001360                    BNE    4$                  ;           1999
000156 012700 000037          9$:    MOV      #37,R0           ;           2000
000162 104447                    TRAP    47                  ;           2001
000164 103003                    BHS    10$                 ;           2002
000166 112737 000002 000000G  MOVB    #2,ENTRY.REASON   ;           2003
000174 012700 000040          10$:   MOV      #40,R0           ;           2004
000200 104447                    TRAP    47                  ;           2005
000202 103013                    BHS    11$                 ;           2006
000204 112737 000001 000000G  MOVB    #1,ENTRY.REASON   ;           2007
000212 005037 000000G          CLR     TICKS              ;           2008
000216 005037 000000G          CLR     SECONDS           ;           2009
    
```


CZRC02
V02.0

CZRCDB0 RC25 DISK EXERCISER
INITIALIZE SECTION

14-Jun-1985 09:36:36
14-Jun-1985 09:32:03

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC02.SRC;8

SEQ 0129
Page 33
(7)

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	:	RO,CLK.ADR
000272	012737	000001	000000G		MOV	#1,CLK.TYPE	:	2017
000300	011137	000000G			MOV	(R1),CLK.CSR	:	CLK.ADR,*
000304	012777	000105	000000G		MOV	#105,@CLK.CSR	:	2018
000312	000415				BR	14\$:	2019
000314	012700	000114		13\$:	MOV	#114,R0	:	2014
000320	104462				TRAP	62		2024
000322	103011				BHIS	14\$		
000324	010001				MOV	R0,R1	:	RO,CLK.ADR
000326	012737	177777	000000G		MOV	#-1,CLK.TYPE	:	2027
000334	011137	000000G			MOV	(R1),CLK.CSR	:	CLK.ADR,*
000340	012777	000100	000000G		MOV	#100,@CLK.CSR	:	2028
000346	005737	000000G		14\$:	TST	CLK.TYPE	:	2029
000352	001006				BNE	15\$:	2034
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),*
000376	016137	000006	000000G		MOV	6(R1),CLK.HERTZ	:	*(CLK.ADR),*
000404	012746	000300			MOV	#300,-(SP)	:	2043
000410	012746	000000V			MOV	#CLK.INT.SERV,-(SP)	:	2044
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	013766	000000G	000002	16\$:	MOV	L\$UNIT,2(SP)	:	2060
000440	005366	000002			DEC	2(SP)		
000444	123727	000000G	000005		CMPB	ENTRY.REASON,#5	:	2053
000452	001002				BNE	17\$		
000454	000137	002744'			JMP	37\$		
000460	005000			17\$:	CLR	R0	:	COUNT
000462	005060	000000G		18\$:	CLR	CST(R0)	:	*(COUNT)
000466	062700	000002			ADD	#2,R0	:	*,COUNT
000472	020027	000066			CMP	R0,#66	:	COUNT,*
000476	003771				BLE	18\$		
000500	005002				CLR	R2	:	UNIT
000502	000137	002730'			JMP	35\$		2060
000506	105062	000000G		19\$:	CLRB	CPT(R2)	:	*(UNIT)
000512	010200				MOV	R2,R0	:	UNIT,*
000514	104442				TRAP	42		2064
000516	010001				MOV	R0,R1	:	*,HWPT.REF

000520	001002			BNE	20\$			
000522	000137	002726'		JMP	34\$			
000526	105016		20\$:	CLRB	(SP)		: FLAG1	2068
000530	005004			CLR	R4		: CTLR	2069
000532	010446		21\$:	MOV	R4,-(SP)		: CTLR,*	2072
000534	012746	000016		MOV	#16,-(SP)			
000540	004737	000000G		JSR	PC,BL\$MUL			
000544	022626			CMP	(SP)+,(SP)+			
000546	026011	000000G		CMP	CST(R0),(R1)		: *,HWPT.REF	
000552	001124			BNE	28\$			
000554	112716	000001		MOVB	#1,(SP)		: *,FLAG1	2076
000560	105005			CLRB	R5		: FLAG2	2077
000562	010446			MOV	R4,-(SP)		: CTLR,*	2081
000564	012746	000007		MOV	#7,-(SP)			
000570	004737	000000G		JSR	PC,BL\$MUL			
000574	010066	000010		MOV	R0,10(SP)			
000600	022626			CMP	(SP)+,(SP)+			
000602	012703	000003		MOV	#3,R3		: *,OFFSET	2078
000606	010300		22\$:	MOV	R3,R0		: OFFSET,*	2081
000610	066600	000004		ADD	4(SP),R0			
000614	006300			ASL	R0			
000616	062700	000000G		ADD	#CST,R0			
000622	032710	040000		BIT	#40000,(R0)			
000626	001016			BNE	23\$			
000630	112705	000001		MOVB	#1,R5		: *,FLAG2	2085
000634	016110	000006		MOV	6(R1),(R0)		: *(HWPT.REF),*	2086
000640	010246			MOV	R2,-(SP)		: UNIT,*	2088
000642	000316			SWAB	(SP)			
000644	042716	160377		BIC	#160377,(SP)			
000650	042710	017400		BIC	#17400,(R0)			
000654	052610			BIS	(SP)+,(R0)			
000656	052710	040000		BIS	#40000,(R0)			
000662	000435			BR	26\$			2089
000664	121061	000006	23\$:	CMPB	(R0),6(R1)		: *,*(HWPT.REF)	2083
000670	001026			BNE	25\$			2096
000672	112705	000001		MOVB	#1,R5		: *,FLAG2	2100
000676	104450			TRAP	50			2101
000700	103014			BHIS	24\$			
000702	011146			MOV	(R1),-(SP)		: HWPT.REF,*	2104
000704	005046			CLR	-(SP)			
000706	116116	000006		MOVB	6(R1),(SP)		: *(HWPT.REF),*	
000712	012746	000000G		MOV	#CER.01,-(SP)			
000716	012746	000003		MOV	#3,-(SP)			
000722	010600			MOV	SP,R0		: SP,*	
000724	104417			TRAP	17			
000726	062706	000010		ADD	#10,SP			
000732	112762	000001	000000G	24\$:	MOVB	#1,DUR(R2)	: *,*(UNIT)	2105
000740	010200			MOV	R2,R0		: UNIT,*	2106
000742	104451			TRAP	51			
000744	000404			BR	26\$			2098
000746	005203		25\$:	INC	R3		: OFFSET	2078
000750	020327	000006		CMP	R3,#6		: OFFSET,*	
000754	003714			BLE	22\$			
000756	105705		26\$:	TSTB	R5		: FLAG2	2115
000760	001025			BNE	29\$			
000762	104450			TRAP	50			2119
000764	103011			BHIS	27\$			

CZRCDD2
V02.0

CZRCDB0 RC25 DISK EXERCISER
INITIALIZE SECTION

14-Jun-1985 09:36:36
14-Jun-1985 09:32:03

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRCDD2.SRC;8

SEQ 0132
Page 36
(7)

001246	005002		37\$:	CLR	R2		; UNIT	2181
001250	000424			BR	42\$			
001252	010200		38\$:	MOV	R2,R0		; UNIT,*	2184
001254	104442			TRAP	42			
001256	010001			MOV	R0,R1		; *,HWPT.REF	
001260	001417			BEQ	41\$			
001262	112762	000001	000000G	MOVB	#1,CPT(R2)		; *,*(UNIT)	2188
001270	005000			CLR	R0		; CTLR	2189
001272	026011	000000G		39\$:	CMP	CST(R0),(R1)	; *(CTLR),HWPT.REF	2192
001276	001003			BNE	40\$			
001300	105260	000005G		INCB	CST+5(R0)		; *(CTLR)	2196
001304	000405			BR	41\$			2194
001306	062700	000016		40\$:	ADD	#16,R0	; *,CTLR	2189
001312	020027	000052		CMP	R0,#52		; CTLR,*	
001316	003765			BLE	39\$			
001320	005202			41\$:	INC	R2	; UNIT	2181
001322	020266	000002		42\$:	CMP	R2,2(SP)	; UNIT,*	
001326	003751			BLE	38\$			
001330	123727	000000G	000002	43\$:	CMPB	ENTRY.REASON,#2		2209
001336	101034			BHI	47\$			
001340	005037	000000G		CLR	CRN			2213
001344	005037	000000G		CLR	CTLR.CNT			2214
001350	005000			CLR	R0		; CTLR	2215
001352	005760	000000G		44\$:	TST	CST(R0)	; *(CTLR)	2218
001356	001402			BEQ	45\$			
001360	005237	000000G		INC	CTLR.CNT			2220
001364	062700	000016		45\$:	ADD	#16,R0	; *,CTLR	2215
001370	020027	000052		CMP	R0,#52		; CTLR,*	
001374	003766			BLE	44\$			
001376	104431			TRAP	31			2224
001400	010037	000000G		MOV	R0,FREE.MEM.ADDR			
001404	011037	000000G		MOV	(R0),MEM.SIZE		; FREE.MEM.ADDR,*	2225
001410	005000			CLR	R0		; COUNT	2227
001412	005060	000000G		46\$:	CLR	TALLY(R0)	; *(COUNT)	2228
001416	062700	000002		ADD	#2,R0		; *,COUNT	2227
001422	020027	001376		CMP	R0,#1376		; COUNT,*	
001426	003771			BLE	46\$			
001430	005000			47\$:	CLR	R0	; CTLR	2236
001432	105060	000000G		48\$:	CLRB	QIO(R0)	; *(CTLR)	2237
001436	005200			INC	R0		; CTLR	2236
001440	020027	000003		CMP	R0,#3		; CTLR,*	
001444	003772			BLE	48\$			
001446	005000			CLR	R0		; COUNT	2238
001450	112760	000377	000000G	49\$:	MOVB	#377,RP.USE(R0)	; *,*(COUNT)	2239
001456	005200			INC	R0		; COUNT	2238
001460	020027	000037		CMP	R0,#37		; COUNT,*	
001464	003771			BLE	49\$			
001466	005037	000000G		CLR	IODQ.OUT			2240
001472	005037	000000G		CLR	IODQ.IN			
001476	132737	000100	000000G	BITB	#100,SWP.FLAGS			2241
001504	001407			BEQ	50\$			
001506	132737	000040	000000G	BITB	#40,SWP.FLAGS			2242
001514	001403			BEQ	50\$			
001516	142737	000100	000000G	BICB	#100,SWP.FLAGS			2244
001524	023737	000000G	000000G	50\$:	CMP	SWP.STRACK,SWP.ETRACK		2256
001532	103407			BLO	51\$			
001534	013700	000000G		MOV	SWP.STRACK,R0		; *,TEMP	2263

CZRC02
V02.0

CZRCDB0 RC25 DISK EXERCISER
INITIALIZE SECTION

14-Jun-1985 09:36:36
14-Jun-1985 09:32:03

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC02.SRC;8

SEQ 0133
Page 37
(7)

001540	013737	000000G	000000G	MOV	SWP.ETRACK,SWP.STRACK	:	2264
001546	010037	000000G		MOV	RO,SWP.ETRACK	: TEMP,*	2265
001552	005000		51\$:	CLR	RO	:	2269
001554	104441			TRAP	41	:	
001556	062706	000010		ADD	#10,SP	:	1925
001562	000207			RTS	PC	:	

: Routine Size: 442 words, Routine Base: \$CODE\$ + 1476
: Maximum stack depth per invocation: 16 words

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

: Routine Size: 4 words, Routine Base: \$CODE\$ + 3262
: Maximum stack depth per invocation: 2 words

CZRC02
V02.0

CZRCDB0 RC25 DISK EXERCISER
AUTODROP SECTION

14-Jun-1985 09:36:36
14-Jun-1985 09:32:03

VAX-11 Bliss-16 V4.0-579
USER\$1:[AZTEC.CZRCDB]CZRC02.SRC;8

SEQ 0134
Page 38
(8)

```
: 2272 1 *SBTTL 'AUTODROP SECTION'  
: 2273 1  
: 2274 1 !*  
: 2275 1 ! THIS CODE IS EXECUTED IMMEDIATELY AFTER THE INITIALIZE CODE IF THE  
: 2276 1 ! "ADR" FLAG WAS SET. THE UNIT(S) UNDER TEST ARE CHECKED TO SEE IF THEY  
: 2277 1 ! WILL RESPOND. THOSE THAT DON'T ARE IMMEDIATELY DROPPED FROM TESTING.  
: 2278 1 !-  
: 2279 1  
: 2280 2 BGNAUTO;  
: 2281 2  
: 2282 2 RETURN;  
: 2283 2  
: 2284 1 ENDAUTO;
```

```
000000 000207 LAUTO: .SBTTL LAUTO AUTODROP SECTION ; 2271  
RTS PC  
: Routine Size: 1 word, Routine Base: $CODE$ + 3272  
: Maximum stack depth per invocation: 0 words
```

```
000000 0C4737 003272' L$AUTO: .SBTTL L$AUTO AUTODROP SECTION ; 2282  
000004 104461 JSR PC,LAUTO  
000006 000207 TRAP 61  
RTS PC  
: Routine Size: 4 words, Routine Base: $CODE$ + 3274  
: Maximum stack depth per invocation: 2 words
```



```
: 2285 1 *SBTTL 'CLEANUP CODING SECTION'
: 2286 1
: 2287 1 !+
: 2288 1 ! THE CLEANUP CODING SECTION CONTAINS THE CODING THAT IS PERFORMED UNDER
: 2289 1 ! ONE OF THREE CONDITIONS: (A) AFTER EACH COMPLETE PASS, (B) WHENEVER
: 2290 1 ! THE "DOCLN" MACRO IS ENCOUNTERED, AND (C) WHEN THE OPERATOR TYPES
: 2291 1 ! "<CTRL-C>". SOME OF THE CODE IN THIS SECTION IS NOT PERFORMED IF THE
: 2292 1 ! REASON FOR ENTRY IS END-OF-PASS ((A) ABOVE).
: 2293 1 !-
: 2294 1
: 2295 2 BGNCLN;
: 2296 2
: 2297 2 IF .CLK_TYPE EQLU NO_CLOCK          ! IF THERE IS NO CLOCK
: 2298 2 THEN                          ! THEN
: 2299 2     RETURN                    ! RETURN
: 2300 2 ELSE                          ! OTHERWISE
: 2301 2     BEGIN
: 2302 2
: 2303 2     .CLK_CSR = 0;              ! TURN OFF CLOCK
: 2304 2     CLRVEC (.CLK_VECTOR);    ! CLEAR CLOCK VECTOR
: 2305 2
: 2306 2     END;
: 2307 2
: 2308 2 IF NOT .EOP_FLAG                ! IF INVOKED BY +C OR DOCLN
: 2309 2 THEN                          ! THEN
: 2310 2     BEGIN
: 2311 2
: 2312 2     SETVEC (4, NEX_TRAP, PRI07); ! IN CASE USER GAVE BAD IP ADDRESS
: 2313 2     INCR CTLR FROM 0 TO (MAX_CTLR - 1) DO ! FOR EACH CONTROLLER
: 2314 2         BEGIN
: 2315 2
: 2316 2         IF (RC25_ADDR = .CST [.CTLR, IP_ADDR]) NEQA 0 ! IF CONTROLLER EXISTS
: 2317 2         THEN                                          ! THEN
: 2318 2             WRT_RC25 (RCIP, RC_ALL, ALL_ONES);    ! WRITE IP TO STOP DEVICE
: 2319 2
: 2320 2         END;
: 2321 2
: 2322 2     CLRVEC (4);          ! RETURN TRAP 4 TO DIAGNOSTIC SUPERVISOR
: 2323 2
: 2324 2     END;              ! IF NOT END OF PASS
: 2325 2
: 2326 2 INCR UNIT FROM 0 TO (MAX_UNITS - 1) DO ! INITIALIZE DROP UNIT REASON VECTOR
: 2327 2     DUR [.UNIT] = 0;
: 2328 2
: 2329 2 IF .MEM_MGMT          ! IF SYSTEM HAS MEMORY MANAGEMENT
: 2330 2 THEN                  ! THEN
: 2331 2     BEGIN
: 2332 2
: 2333 2     KTPAR4 = %0'1000'; ! RESTORE PAR'S
: 2334 2     KTPAR5 = %0'1200';
: 2335 2     MMRO = 0;         ! TURN OFF MEMORY MANAGEMENT
: 2336 2
: 2337 2     END;
: 2338 2
: 2339 2 EOP_FLAG = FALSE;    ! CLEAR END-OF-PASS FLAG
: 2340 2
: 2341 1 ENDCLN;
```

```

.SBTTL LCLEAN CLEANUP CODING SECTION
LCLEAN: MOV R1,-(SP) ; 2284
        TST CLK.TYPE ; 2297
        BEQ 6$ ; 2299
        CLR @CLK.CSR ; 2303
        MOV CLK.VECTOR,R0 ; 2304
        TRAP 36 ;
        BITB #1,EOP.FLAG ; 2308
        BNE 3$ ;
        MOV #340,-(SP) ; 2312
        MOV #NEX.TRAP,-(SP)
        MOV #4,-(SP)
        MOV #3,-(SP)
        TRAP 37
        CLR R0 ; CTLR 2313
        MOV CST(R0),RC25.ADDR ; *(CTLR),* 2316
        BEQ 2$
        MOV #-1,R1 ; *,RC.REG 2318
        MOV R1,@CST(R0) ; RC.REG,*
        ADD #16,R0 ; *,CTLR 2313
        CMP R0,#52 ; CTLR,*
        BLE 1$
        MOV #4,R0 ;
        TRAP 36 ; 2322
        ADD #10,SP ;
        CLR R0 ; UNIT 2310
        CLRB DUR(R0) ; *(UNIT) 2326
        INC R0 ; UNIT 2327
        CMP R0,#17 ; UNIT,* 2326
        BLE 4$
        BITB #1,MEM.MGMT ; 2329
        BEQ 5$
        MOV #1000,@#172350 ;
        MOV #1200,@#172352 ;
        CLR @#177572 ;
        CLRB EOP.FLAG ; 2335
        MOV (SP)+,R1 ; 2339
        RTS PC ; 2284
    
```

; Routine Size: 64 words, Routine Base: \$CODE\$ + 3304
; Maximum stack depth per invocation: 7 words

```

000000 004737 003304' L$CLEAN::
000004 104412 JSR PC,LCLEAN ; 2339
000006 000207 TRAP 12
        RTS PC
    
```

; Routine Size: 4 words, Routine Base: \$CODE\$ + 3504
; Maximum stack depth per invocation: 2 words


```
; 2342 1 *SBTTL 'DROP UNIT SECTION'
; 2343 1
; 2344 1 !+
; 2345 1 ! THE DROP-UNIT SECTION CONTAINS THE CODING THAT CAUSES A PLATTER TO NO
; 2346 1 ! LONGER BE TESTED. IT IS EXECUTED WHENEVER THE MACRO "DODU" IS
; 2347 1 ! ENCOUNTERED IN THE PROGRAM, OR WHEN THE OPERATOR TYPES THE COMMAND
; 2348 1 ! "DROP/UNIT:X"
; 2349 1 !
; 2350 1 ! THE PROGRAM WILL DROP A UNIT FOR ONE OF SEVERAL REASONS:
; 2351 1 ! 1. A CONTROLLER CANNOT BE BROUGHT ONLINE,
; 2352 1 ! 2. A PLATTER CANNOT BE BROUGHT ONLINE TO ITS CONTROLLER,
; 2353 1 ! 3. AN UNRECOVERABLE PORT OR CONTROLLER ERROR HAS BEEN DETECTED,
; 2354 1 ! 4. A PLATTER HAS REACHED THE HARD ERROR LIMIT, OR
; 2355 1 ! 5. AN MSCP END MESSAGE CONTAINS A "UNIT-OFFLINE" STATUS CODE.
; 2356 1 !-
; 2357 1
; 2358 2 BGNDU;
; 2359 2
; 2360 2 OWN
; 2361 2 DUM_TBL : VECTOR [5] INITIAL (DUM_UC, DUM_CE, ! ADDRESS TABLE FOR DROP UNIT MESSAGES
; 2362 2 DUM_IE, DUM_HE, DUM_UE);
; 2363 2
; 2364 2 LOCAL
; 2365 2 UNIT : WORD ! NUMBER OF UNIT BEING DROPPED
; 2366 2 PRINT : WORD; ! O.K TO PRINT DROP UNIT MESSAGE
; 2367 2
; 2368 2 LABEL
; 2369 2 SEARCH;
; 2370 2
; 2371 3 BEGIN
; 2372 3
; 2373 3 REGISTER
; 2374 3 INPUT = 0; ! UNIT NUMBER APPEARS IN RO UPON ENTRY
; 2375 3 UNIT = .INPUT; ! GET UNIT NUMBER
; 2376 3
; 2377 2 END; ! UNDECLARE REGISTER
; 2378 2
; 2379 2 PRINT = FALSE; ! ASSUME NO PRINTING OF MESSAGE
; 2380 2
; 2381 2 SEARCH: ! BEGIN SEARCH BLOCK
; 2382 3 BEGIN
; 2383 3
; 2384 3 INCR CTRL FROM 0 TO (MAX_CTRL - 1) DO ! FOR EACH CST
; 2385 4 BEGIN
; 2386 4
; 2387 4 INCR OFFSET FROM (0 + OF_UN) TO (3 + OF_UN) DO ! FOR EACH UNIT ENTRY IN CST
; 2388 5 BEGIN
; 2389 5
; 2390 6 IF ((.CST [.CTRL, .OFFSET, P_PRES] EQLU PRESENT) AND ! IF UNIT EXISTS AND
; 2391 6 (.CST [.CTRL, .OFFSET, P_UNIT] EQLU .UNIT)) ! UNIT MATCHES CST ENTRY
; 2392 5 THEN ! THEN
; 2393 6 BEGIN
; 2394 6
; 2395 6 IF .CST [.CTRL, .OFFSET, P_STAT] EQLU ONLINE ! IF UNIT IS ONLINE
; 2396 6 THEN ! THEN
; 2397 7 BEGIN
; 2398 7
```

```

; 2399 7          PRINT = TRUE;                ! O.K. TO PRINT MESSAGE
; 2400 7          CST [.CTRL, .OFFSET, P_STAT] = OFFLINE; ! MARK UNIT OFFLINE
; 2401 7          IF .CPT [.UNIT] EQLU YES           ! IF UNIT IS UNDER TEST
; 2402 7          THEN                               ! THEN
; 2403 8              BEGIN
; 2404 8
; 2405 8              CPT [.UNIT] = NO;             ! NO FURTHER TESTING
; 2406 8              CST [.CTRL, U_CNT] = .CST [.CTRL, U_CNT] - 1; ! DECREMENT UNIT COUNT
; 2407 8              DM_COMM [.CTRL, .OFFSET - OF_UN, ALLBIT] = -1; ! IF DM EXER, TELL FPT
; 2408 8
; 2409 7              END;
; 2410 7
; 2411 6          END;                               ! IF UNIT WAS ONLINE
; 2412 6
; 2413 6          LEAVE SEARCH;                    ! EXIT SEARCH BLOCK
; 2414 6
; 2415 5          END;                               ! IF UNIT EXISTS AND MATCHES
; 2416 5
; 2417 4          END;                               ! CST UNIT LOOP
; 2418 4
; 2419 3          END;                               ! CONTROLLER LOOP
; 2420 3
; 2421 2          END;                               ! SEARCH BLOCK
; 2422 2
; 2423 3          IF MANUAL AND (.PRINT OR (.DUR [.UNIT] LEQU DU_INIT)) ! IF O.K. TO PRINT MESSAGE
; 2424 2          THEN                               ! THEN
; 2425 3              BEGIN
; 2426 3
; 2427 3              PRINTF (CRLF);                ! <CR><LF>
; 2428 3              PRINTF (ETIME, .HOURS, .MINUTES, .SECONDS); ! ELAPSED TIME
; 2429 3              PRINTF (DUM_00, .UNIT);       ! "UNIT XX. DROPPED - "
; 2430 3              PRINTF (.DUM_TBL [.DUR [.UNIT]]); ! REASON
; 2431 3
; 2432 2          END;
; 2433 2
; 2434 1          ENDDU;

```

```

003514 000000G          DUM.TBL: .WORD    DUM.UC
003516 000000G          .WORD    DUM.CE
003520 000000G          .WORD    DUM.IE
003522 000000G          .WORD    DUM.HE
003524 000000G          .WORD    DUM.UE

```

```

000000 004137 000000G          .SBTTL  LDU DROP UNIT SECTION
000004 010001          LDU:   JSR      R1,$SAVE5          ;
000006 005004          MOV      R0,R1          ; INPUT,UNIT
000010 005003          CLR      R4          ; PRINT
000012 010346          CLR      R3          ; CTRL
000014 012746 000007          1$:   MOV      R3,-(SP)          ; CTRL,*
000020 004737 000000G          MOV      #7,-(SP)
000024 010005          JSR      PC,BL$MUL
000026 012702 000003          MOV      R0,R5
000032 010500          2$:   MOV      #3,R2          ; *.OFFSET
000034 060200          ADD      R5,R0          ;
                                ; OFFSET,*

```


CZRC02
V02.0

CZRCDB0 RC25 DISK EXERCISER
DROP UNIT SECTION

14-Jun-1985 09:36:36
14-Jun-1985 09:32:03

VAX-11 Bliss-16 V4.0-579
USER\$1:[AZTEC.CZRCDB]CZRC02.SRC;8

SEQ 0139
Page 43
(10)

000036	006300		ASL	R0			
000040	052700	000000G	ADD	#CST,R0			
000044	032710	040000	BIT	#40000,(R0)			
000050	001450		BEQ	4\$			
000052	010146		MOV	R1,-(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	012704	000001	MOV	#1,R4	; *,PRINT		2399
000102	042710	020000	BIC	#20000,(R0)			2400
000106	126104	000000G	CMPB	CPT(R1),R4	; *(UNIT),*		2401
000112	001025		BNE	3\$			
000114	105061	000000G	CLRB	CPT(R1)	; *(UNIT)		2405
000120	010316		MOV	R3,(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	010316		MOV	R3,(SP)	; CTLR,*		2407
000142	012746	000032	MOV	#32,-(SP)			
000146	004737	000000G	JSR	PC,BL\$MUL			
000152	005726		TST	(SP)+			
000154	060200		ADD	R2,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	005202		4\$: INC	R2	; OFFSET		2387
000174	020227	000006	CMP	R2,#6	; OFFSET,*		
000200	003714		BLE	2\$			
000202	022626		CMP	(SP)+,(SP)+			2385
000204	005203		INC	R3	; CTLR		2384
000206	020327	000003	CMP	R3,#3	; CTLR,*		
000212	003677		BLE	1\$			
000214	104450		5\$: TRAP	50			2423
000216	103053		BHIS	7\$			
000220	006004		ROR	R4	; PRINT		
000222	103404		BLO	6\$			
000224	126127	000000G 000002	CMPB	DUR(R1),#2	; *(UNIT),*		
000232	101045		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	010116		MOV	R1,(SP)	; UNIT,*		2429
000302	012746	000000G	MOV	#DUM.00,-(SP)			

CZRCO2
V02.0

CZRCDB0 RC25 DISK EXERCISER
DROP UNIT SECTION

14-Jun-1985 09:36:36
14-Jun-1985 09:32:03

VAX-11 Bliss-16 V4.0-579
USER\$1:[AZTEC.CZRCDB]CZRCO2.SRC;8

SEQ 0140
Page 44
(10)

000306	012746	000002	MOV	#2,-(SP)		
000312	010600		MOV	SP,R0	; SP,*	
000314	104417		TRAP	17		
000316	005000		CLR	R0		
000320	156100	000000G	BISB	DUR(R1),R0	; *(UNIT),*	2430
000324	006300		ASL	R0		
000326	016016	003514'	MOV	DUM.TBL(R0),(SP)		
000332	012746	000001	MOV	#1,-(SP)		
000336	010600		MOV	SP,R0	; SP,*	
000340	104417		TRAP	17		
000342	062706	000022	ADD	#22,SP		2425
000346	000207		RTS	PC		2341

; Routine Size: 116 words, Routine Base: \$CODE\$ + 3526
; Maximum stack depth per invocation: 17 words

000000	004737	003526'		.SBTTL	L\$DU DROP UNIT SECTION	
000004	104453		L\$DU::	JSR	PC,LDU	
000006	000207			TRAP	53	2432
				RTS	PC	

; Routine Size: 4 words, Routine Base: \$CODE\$ + 4076
; Maximum stack depth per invocation: 2 words


```

; 2435 1  %SBTTL 'ADD UNIT SECTION'
; 2436 1
; 2437 1  !*
; 2438 1  !   THE ADD-UNIT SECTION CONTAINS ANY CODE THE PROGRAMMER WISHES TO BE
; 2439 1  !   EXECUTED IN CONJUNCTION WITH THE ADDING OF A UNIT BACK TO THE TEST
; 2440 1  !   CYCLE. THIS CODE WILL BE EXECUTED AFTER THE OPERATOR COMMAND
; 2441 1  !   "ADD/UNIT:X".
; 2442 1  !-
; 2443 1
; 2444 2  BGNAU;
; 2445 2
; 2446 2  LOCAL
; 2447 2      UNIT : WORD,           ! NUMBER OF UNIT BEING ADDED
; 2448 2      STINDX : WORD,
; 2449 2      ENDIDX : WORD;
; 2450 2
; 2451 3  BEGIN
; 2452 3
; 2453 3  REGISTER
; 2454 3      INPUT = 0;           ! UNIT NUMBER APPEARS IN R0 UPON ENTRY
; 2455 3  UNIT = .INPUT;         ! GET UNIT NUMBER
; 2456 3
; 2457 2  END;                   ! UNDECLARE REGISTER
; 2458 2
; 2459 2  DUR [.UNIT] = 0;        ! ZERO OUT DROP UNIT REASON
; 2460 2  STINDX = .UNIT * TALLY_LEN; ! ZERO OUT
; 2461 2  ENDIDX = .STINDX + TALLY_LEN - 1; ! ADDED
; 2462 2  INCR COUNT FROM .STINDX TO .ENDIDX DO ! UNIT'S
; 2463 2      TALLY [.COUNT] = 0; ! STATISTICS
; 2464 2
; 2465 1  ENDAU;

```

000000	004137	000000G	LAU:	.SBTTL	LAU ADD UNIT SECTION		
000004	105060	000000G		JSR	R1,\$SAVE2	;	2434
000010	010046			CLRB	DUR(R0)	;	*(UNIT)
000012	012746	000030		MOV	R0,-(SP)	;	UNIT,*
000016	004737	000000G		MOV	#30,-(SP)		2460
000022	010002			JSR	PC,BL\$MUL		
000024	062702	000027		MOV	R0,R2	;	STINDX,ENDIDX
000030	010001			ADD	#27,R2	;	*,ENDIDX
000032	005301			MOV	R0,R1	;	STINDX,COUNT
000034	000404			DEC	R1	;	COUNT
000036	010100		1\$:	BR	2\$		
000040	006300			MOV	R1,R0	;	COUNT,*
000042	005060	000000G		ASL	R0		2463
000046	005201		2\$:	CLR	TALLY(R0)		
000050	020102			INC	R1	;	COUNT
000052	003771			CMP	R1,R2	;	COUNT,ENDIDX
000054	022626			BLE	1\$		
000056	000207			CMP	(SP)+,(SP)+	;	2434
				RTS	PC		

; Routine Size: 24 words, Routine Base: \$CODE\$ + 4106
; Maximum stack depth per invocation: 6 words

CZRC2
V02.0

CZRCDB0 RC25 DISK EXERCISER
ADD UNIT SECTION

14-Jun-1985 09:36:36
14-Jun-1985 09:32:03

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC2.SRC;8

SEQ 0142
Page 46
(11)

000000	004737	004106'		.SBTTL	L\$AU ADD UNIT SECTION		
000004	104452		L\$AU::	JSR	PC,LAU	:	2463
000006	000207			TRAP	52		
				RTS	PC		

; Routine Size: 4 words, Routine Base: \$CODE\$ + 4166
; Maximum stack depth per invocation: 2 words

CZRC02
V02.0

CZRCDB0 RC25 DISK EXERCISER
NON-EXISTENT MEMORY TRAP HANDLER

14-Jun-1985 09:36:36
14-Jun-1985 09:32:03

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC02.SRC:8

SEQ 0143
Page 47
(12)

```

; 2466 1  #SBTTL 'NON-EXISTENT MEMORY TRAP HANDLER'
; 2467 1
; 2468 1  !+
; 2469 1  ! THIS TRAP HANDLER IS VECTORED FROM LOCATION 4 FOR ALL UNIBUS TIMEOUT
; 2470 1  ! ERRORS, INDICATING THAT AN ATTEMPT WAS MADE TO REFERENCE A NON-EXISTENT
; 2471 1  ! MEMORY LOCATION. ITS MAIN PURPOSE IS TO SET A FLAG FOR THE RC11
; 2472 1  ! REGISTER EXISTENCE TEST, INDICATING THE ABSENCE OF A DEVICE REGISTER.
; 2473 1  !-
; 2474 1
; 2475 2  BGNSRV (NEX_TRAP);
; 2476 2
; 2477 2  NEX = TRUE;                ! NEX TRAP OCCURRED
; 2478 2
; 2479 1  ENDSRV;

```

```

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

```

2477
2475

```

; Routine Size: 4 words,      Routine Base: $CODE$ + 4176
; Maximum stack depth per invocation: 0 words

```

```

: 2480 1  *SBTTL 'CLOCK INTERRUPT SERVICE ROUTINE'
: 2481 1
: 2482 1  !*
: 2483 1  !
: 2484 1  !   THE CLOCK INTERRUPT SERVICE ROUTINE IS ENTERED AT THE CLOCK HERTZ RATE
: 2485 1  !   (50 OR 60 TIMES PER SECOND). ITS PURPOSE IS TO MAINTAIN THE ELAPSED
: 2486 1  !   TIME OF THE EXERCISER, AND, IF ONE SECOND HAS ELAPSED, TO FLAG THE
: 2487 1  !   EVENT BY SETTING T_FLAG TO "TRUE".
: 2488 1  !-
: 2489 2  BGNSRV (CLK_INT_SERV);
: 2490 2
: 2491 2  TICKS = .TICKS + 1;
: 2492 2  IF .TICKS EQLU .CLK_HERTZ
: 2493 2  THEN
: 2494 3      BEGIN
: 2495 3
: 2496 3          TICKS = 0;
: 2497 3          SECONDS = .SECONDS + 1;
: 2498 3          IF .SECONDS EQLU 60
: 2499 3          THEN
: 2500 4              BEGIN
: 2501 4
: 2502 4                  SECONDS = 0;
: 2503 4                  MINUTES = .MINUTES + 1;
: 2504 4                  IF .MINUTES EQLU 60
: 2505 4                  THEN
: 2506 5                      BEGIN
: 2507 5
: 2508 5                          MINUTES = 0;
: 2509 5                          HOURS = .HOURS + 1;
: 2510 5
: 2511 4                      END;
: 2512 4              END;
: 2513 3          END;
: 2514 3      END;
: 2515 2  END;
: 2516 2
: 2517 2  IF .TICKS EQLU 0          ! IF ONE SECOND HAS ELAPSED
: 2518 2  THEN                    ! THEN
: 2519 2      T_FLAG = TRUE;      ! FLAG THE EVENT
: 2520 2
: 2521 1  ENDSRV;

```

000000	005237	000000G	.SBTTL	CLK.INT.SERV	CLOCK INTERRUPT SERVICE ROUTINE	
			CLK.INT.SERV::			
000004	023737	000000G 000000G	INC	TICKS		2491
000012	001024		CMP	TICKS,CLK.HERTZ		2492
000014	005037	000000G	BNE	1\$		
000020	005237	000000G	CLR	TICKS		2496
000024	023727	000000G 000074	INC	SECONDS		2497
000032	001014		CMP	SECONDS,#74		2498
000034	005037	000000G	BNE	1\$		
000040	005237	000000G	CLR	SECONDS		2502
000044	023727	000000G 000074	INC	MINUTES		2503
000052	001004		CMP	MINUTES,#74		2504
			BNE	1\$		

CZRCO2
V02.0

CZRCDBO RC25 DISK EXERCISER
CLOCK INTERRUPT SERVICE ROUTINE

14-Jun-1985 09:36:36
14-Jun-1985 09:32:03

VAX-11 Bliss-16 V4.0-579
USER\$1:[AZTEC.CZRCDB]CZRCO2.SRC;8

SEQ 0145
Page 49
(13)

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\$ + 4206
: Maximum stack depth per invocation: 0 words

CZRC02
V02.0

CZRCDB0 RC25 DISK EXERCISER
GLOBAL ROUTINES

14-Jun-1985 09:36:36
14-Jun-1985 09:32:03

VAX-11 Bliss-16 V4.0-579
USER\$1:[AZTEC.CZRCDB]CZRC02.SRC;8

SEQ 0146
Page 50
(14)

```

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

```

```

000000 005000          .SBTTL COPY.BLK GLOBAL ROUTINES
                                COPY.BLK::
000002 000411          CLR R0 ; I 2531
000004 017676 000006 000004 1$: BR 2$ ; 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: $CODE$ + 4310
; Maximum stack depth per invocation: 0 words

```



```

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

```

```

000000 010146 .SBTTL SET.CPAR GLOBAL ROUTINES
000002 016601 000004 SET.CPAR::
000006 010137 000000G MOV R1, -(SP) ;
000012 010146 MOV R1, CCTLR ; CTLR,*
000014 012746 000016 MOV R1, -(SP) ;
000020 004737 000000G JSR PC, BL$MUL
000024 062700 000000G ADD #CST, R0
000030 010037 000000G MOV R0, CST.ADDR
000034 010116 MOV R1, (SP) ;
000036 012746 000022 MOV #22, -(SP) ;
000042 004737 000000G JSR PC, BL$MUL
000046 062700 000000G ADD #DCT, R0
000052 010037 000000G MOV R0, DCT.ADDR
000056 017737 000000G 000000G MOV @CST.ADDR, RC25.ADDR ;
000064 010116 MOV R1, (SP) ;
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

```

CZRC02
V02.0

CZRCDB0 RC25 DISK EXERCISER
GLOBAL ROUTINES

14-Jun-1985 09:36:36
14-Jun-1985 09:32:03

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB1CZRC02.SRC;8

SEQ 0148
Page 52
(15)

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			
000130	062737	000017 000000G	ADD	#17,OCL.X2		; OCL.X1,*	2568
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			
000156	062737	000007 000000G	ADD	#7,RPS.X2		; RPS.X1,*	2570
000164	062706	000010	ADD	#10,SP			2560
000170	012601		MOV	(SP)+,R1			2539
000172	000207		RTS	PC			

; Routine Size: 62 words, Routine Base: \$CODE\$ + 4350
; Maximum stack depth per invocation: 6 words


```

: 2573 1 GLOBAL ROUTINE SET_UPAR (OFFSET) : NOVALUE =
: 2574 1
: 2575 1 !+
: 2576 1 ! THIS ROUTINE SETS UP THE COMMONLY-USED UNIT-RELATED DATA ITEMS FOR
: 2577 1 ! THE CURRENT CONTROLLER AND GIVEN CST OFFSET.
: 2578 1 !
: 2579 1 ! INPUTS:
: 2580 1 ! OFFSET - WORD OFFSET INTO CURRENT CONTROLLER'S CST WHICH
: 2581 1 ! DESCRIBES A UNIT
: 2582 1 !
: 2583 1 ! IMPLICIT INPUTS:
: 2584 1 ! CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
: 2585 1 !
: 2586 1 ! IMPLICIT OUTPUTS:
: 2587 1 ! CUOFF - CURRENT UNIT'S CST OFFSET
: 2588 1 ! CPLAT - CURRENT PLATTER ADDRESS (MSCP UNIT NUMBER)
: 2589 1 ! L$LUN - CURRENT UNIT NUMBER (DS UNIT NUMBER)
: 2590 1 ! T_ADDR - ADDRESS OF CURRENT UNIT'S STATISTICS BLOCK (TALLY)
: 2591 1 !-
: 2592 1
: 2593 2 BEGIN
: 2594 2
: 2595 2 CUOFF = .OFFSET;
: 2596 2 CPLAT = .CST_ADDR [.OFFSET, P_ADDR];
: 2597 2 L$LUN = .CST_ADDR [.OFFSET, P_UNIT];
: 2598 2 T_ADDR = TALLY + (.L$LUN * TALLY_LEN * 2);
: 2599 2
: 2600 1 END;

```

```

000000 010146 .SBTTL SET.UPAR GLOBAL ROUTINES
SET.UPAR::
000002 016637 000004 000000G MOV R1, -(SP) ; 2573
000010 016600 000004 MOV 4(SP), CUOFF ; OFFSET,* 2595
000014 006300 MOV 4(SP), RO ; CUOFF,* 2596
000016 063700 000000G ASL RO
000022 111037 000000G ADD CST_ADDR, RO
000026 105037 000001G MOVB (RO), CPLAT
000032 011001 CLRB CPLAT+1
000034 000301 MOV (RO), R1 ; 2597
000036 042701 177740 SWAB R1
000042 010137 000000G BIC #177740, R1
000046 010146 MOV R1, L$LUN ; L$LUN,* 2598
000050 010146 000060 MOV R1, -(SP)
000054 000737 000000G MOV #60, -(SP)
000060 062700 000000G JSR PC, BL$MUL
000064 010037 000000G ADD #TALLY, RO
000070 022626 MOV RO, T_ADDR
000072 012601 CMP (SP)+, (SP)+ ; 2593
000074 000207 MOV (SP)+, R1 ; 2573
RTS PC

```

```

; Routine Size: 31 words, Routine Base: $CODE$ + 4544
; Maximum stack depth per invocation: 4 words

```

```

: 2601 1 GLOBAL ROUTINE GET_ENV (CTRL) =
: 2602 1
: 2603 1 !+
: 2604 1 ! THIS ROUTINE SEARCHES THE MSCP ENVELOPE POOL ALLOCATION TABLE (ENV_USE)
: 2605 1 ! FOR A FREE MSCP ENVELOPE TO ALLOCATE TO THE GIVEN CONTROLLER. IF ONE IS
: 2606 1 ! FOUND, THE ENVELOPE IS ZEROED OUT, AND THE ENVELOPE INDEX IS RETURNED
: 2607 1 ! TO THE CALLER. OTHERWISE, A -1 IS RETURNED INDICATING NONE AVAILABLE.
: 2608 1 !
: 2609 1 ! INPUTS:
: 2610 1 ! CTRL - CONTROLLER NUMBER REQUESTING ALLOCATION
: 2611 1 !-
: 2612 1
: 2613 2 BEGIN
: 2614 2
: 2615 2 LOCAL
: 2616 2 INDEX : WORD;
: 2617 2
: 2618 2 INDEX = -1;
: 2619 2
: 2620 2 INCR COUNT FROM 0 TO (ENV_CNT - 1) DO ! FOR EACH ENTRY IN ALLOCATION TABLE
: 2621 3 BEGIN
: 2622 3
: 2623 3 IF .ENV_USE [.COUNT] LSS 0 ! IF ENTRY INDICATES FREE ENVELOPE
: 2624 3 THEN ! THEN
: 2625 4 BEGIN
: 2626 4
: 2627 4 ENV_USE [.COUNT] = .CTRL; ! ALLOCATE ENVELOPE TO CONTROLLER
: 2628 4 INDEX = .COUNT;
: 2629 4 INCR J FROM 2 TO (ENV_LEN - 1) DO ! ZERO OUT ENVELOPE
: 2630 4 MSCP_ENV [.COUNT, .J, ALLBIT] = 0;
: 2631 4 EXITLOOP; ! DONE
: 2632 4
: 2633 3 END;
: 2634 3
: 2635 2 END;
: 2636 2
: 2637 2 RETURN .INDEX;
: 2638 2
: 2639 1 END;

```

		.SBTTL	GET_ENV GLOBAL ROUTINES	
000000	004137	000000G	GET_ENV::	
			JSR R1,\$SAVE4	; 2601
000004	012703	177777	MOV #-1,R3	; *.INDEX 2618
000010	005001		CLR R1	; COUNT 2620
000012	105761	000000G	1\$: TSTB ENV_USE(R1)	; *(COUNT) 2623
000016	002026		BGE 3\$	
000020	116661	000014 000000G	MOVB 14(SP),ENV_USE(R1)	; CTRL,*(COUNT) 2627
000026	010103		MOV R1,R3	; COUNT,INDEX 2628
000030	010146		MOV R1,-(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	010004		2\$: MOV R0,R4	; 2630
000052	060204		ADD R2,R4	; J,*

CZRC D2
V02.0

CZRCDB0 RC25 DISK EXERCISER
GLOBAL ROUTINES

14-Jun-1985 09:36:36
14-Jun-1985 09:32:03

VAX-11 Bliss-16 V4.0-579
USER\$1:[AZTEC.CZRCDB]CZRC D2.SRC;8

SEQ 0151
Page 55
(17)

000054	006304		ASL	R4			
000056	005064	000000G	CLR	MSCP.ENV(R4)			
000062	005202		INC	R2	:	J	2629
000064	020227	000041	CMP	R2,#41	:	J,*	
000070	003767		BLE	2\$			
000072	000404		BR	4\$:		2625
000074	005201		INC	R1	:	COUNT	2620
000076	020127	000137	CMP	R1,#137	:	COUNT,*	
000102	003743		BLE	1\$			
000104	010300		MOV	R3,R0	:	INDEX,*	2613
000106	000207		RTS	PC	:		2601

; Routine Size: 36 words, Routine Base: \$CODE\$ + 4642
; Maximum stack depth per invocation: 8 words

CZRCO2
V02.0

CZRCDBO RC25 DISK EXERCISER
GLOBAL ROUTINES

14-Jun-1985 09:36:36
14-Jun-1985 09:32:03

VAX-11 Bliss-16 V4.0-579
USER\$1:[AZTEC.CZRCDB]CZRCO2.SRC;8

SEQ 0152
Page 56
(18)

```

: 2640 1 GLOBAL ROUTINE PUT_ENV (INDEX) : NOVALUE =
: 2641 1
: 2642 1 !+
: 2643 1 ! THE MSCP ENVELOPE DESIGNATED BY "INDEX" IS RETURNED TO THE POOL BY THIS
: 2644 1 ! ROUTINE.
: 2645 1 !-
: 2646 1
: 2647 2 BEGIN
: 2648 2
: 2649 2 ENV_USE [.INDEX] = -1;
: 2650 2
: 2651 1 END;

```

```

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

```

```

; Routine Size: 6 words, Routine Base: $CODE$ + 4752
; Maximum stack depth per invocation: 0 words

```



```

; 2652 1 GLOBAL ROUTINE PUTA_ENV (CTRL) : NOVALUE =
; 2653 1
; 2654 1 !+
; 2655 1 ! THIS ROUTINE DEALLOCATES ALL MSCP ENVELOPES WHICH HAVE BEEN ALLOCATED
; 2656 1 ! TO A PARTICULAR CONTROLLER.
; 2657 1 !
; 2658 1 ! INPUTS:
; 2659 1 ! CTRL - CONTROLLER NUMBER
; 2660 1 !-
; 2661 1
; 2662 2 BEGIN
; 2663 2
; 2664 2 INCR COUNT FROM 0 TO (ENV_CNT - 1) DO          ! FOR EACH ENTRY IN ALLOCATION TABLE
; 2665 3 BEGIN
; 2666 3
; 2667 3 IF .ENV_USE [.COUNT] EQLU .CTRL              ! IF ENVELOPE IS ALLOCATED TO GIVEN CONTROLLER
; 2668 3 THEN                                          ! THEN
; 2669 3     ENV_USE [.COUNT] = -1;                  ! DEALLOCATE IT
; 2670 3
; 2671 2 END;
; 2672 2
; 2673 1 END;                                          ! ROUTINE PUTA_ENV

```

```

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

```

; Routine Size: 16 words, Routine Base: \$CODE\$ + 4766
; Maximum stack depth per invocation: 2 words

```

; 2674 1 GLOBAL ROUTINE GET_RETPKT (CTRL) =
; 2675 1
; 2676 1 !+
; 2677 1 ! THIS ROUTINE SEARCHES THE RETURN PACKET POOL ALLOCATION TABLE (RP_USE)
; 2678 1 ! FOR A FREE RETURN PACKET TO ALLOCATE TO THE GIVEN CONTROLLER. IF ONE IS
; 2679 1 ! FOUND, THE PACKET IS ZEROED OUT, AND THE PACKET INDEX IS RETURNED TO
; 2680 1 ! THE CALLER. OTHERWISE, A -1 IS RETURNED INDICATING NONE AVAILABLE.
; 2681 1 !
; 2682 1 ! INPUTS:
; 2683 1 ! CTRL - CONTROLLER NUMBER REQUESTING ALLOCATION
; 2684 1 !-
; 2685 1
; 2686 2 BEGIN
; 2687 2
; 2688 2 LOCAL
; 2689 2 INDEX : WORD;
; 2690 2
; 2691 2 INDEX = -1; ! ASSUME NONE AVAILABLE
; 2692 2
; 2693 2 INCR COUNT FROM 0 TO (RP_CNT - 1) DO ! FOR EACH ENTRY IN TABLE
; 2694 3 BEGIN
; 2695 3
; 2696 3 IF .RP_USE [.COUNT] LSS 0 ! IF FREE RETPKT IS FOUND
; 2697 3 THEN
; 2698 4 BEGIN
; 2699 4
; 2700 4 RP_USE [.COUNT] = .CTRL; ! ALLOCATE RETURN PACKET TO CONTROLLER
; 2701 4 INDEX = .COUNT;
; 2702 4 INCR J FROM 0 TO (RP_LEN - 1) DO ! ZERO OUT RETPKT
; 2703 4 RETPKT [.COUNT, .J, ALLBIT] = 0;
; 2704 4 EXITLOOP; ! DONE
; 2705 4
; 2706 3 END;
; 2707 3
; 2708 2 END;
; 2709 2
; 2710 2 RETURN .INDEX; ! RETURN PACKET INDEX (OR -1) TO CALLER
; 2711 2
; 2712 1 END;

```

```

000000 004137 000000G .SBTTL GET.RETPKT GLOBAL ROUTINES
000004 012703 177777 GET.RETPKT::
000010 005001 JSR R1,$SAVE4 ;
000012 105761 000000G 1$: TSTB RP.USE(R1) ; *,INDEX
000016 002025 BGE 3$ ; COUNT
000020 116661 000014 000000G MOVB 14(SP),RP.USE(R1) ; *(COUNT)
000026 010103 MOV R1,R3 ; CTRL,*(COUNT)
000030 010146 MOV R1,-(SP) ; COUNT,INDEX
000032 012746 000030 MOV #30,-(SP) ; COUNT,*
000036 004737 000000G JSR PC,BL$MUL
000042 022626 CMP (SP)+,(SP)+
000044 005002 CLR R2 ; J
000046 010004 2$: MOV R0,R4 ;
000050 060204 ADD R2,R4 ; J,*

```


CZRCO2
V02.0

CZRCDBO RC25 DISK EXERCISER
GLOBAL ROUTINES

14-Jun-1985 09:36:36
14-Jun-1985 09:32:03

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRCO2.SRC:8

SEQ 0155
Page 59
(20)

000052	006304		ASL	R4		
000054	005064	000000G	CLR	RETPKT(R4)		
000060	005202		INC	R2	: J	2702
000062	020227	000027	CMP	R2,#27	: J,*	
000066	003767		BLE	2\$		
000070	000404		BR	4\$		
000072	005201		INC	R1	: COUNT	2698
000074	020127	000037	CMP	R1,#37	: COUNT,*	2693
000100	003744		BLE	1\$		
000102	010300		MOV	R3,R0	: INDEX,*	2686
000104	000207		RTS	PC	:	2674

: Routine Size: 35 words, Routine Base: \$CODE\$ + 5026
: Maximum stack depth per invocation: 8 words

CZRCO2
V02.0

CZRCDB0 RC25 DISK EXERCISER
GLOBAL ROUTINES

14-Jun-1985 09:36:36
14-Jun-1985 09:32:03

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRCO2.SRC;8

SEQ 0156
Page 60
(21)

```
; 2713 1 GLOBAL ROUTINE PUT_RETPKT (INDEX) : NOVALUE =
; 2714 1
; 2715 1 !+
; 2716 1 ! THE RETURN PACKET DESIGNATED BY "INDEX" IS RETURNED TO THE POOL BY THIS
; 2717 1 ! ROUTINE.
; 2718 1 !-
; 2719 1
; 2720 2 BEGIN
; 2721 2
; 2722 2 RP_USE [.INDEX] = -1;
; 2723 2
; 2724 1 END;
```

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

```
; Routine Size: 6 words, Routine Base: $CODE$ + 5134
; Maximum stack depth per invocation: 0 words
```



```

: 2725 1 GLOBAL ROUTINE GET_IO_BUFF (ADDR) : NOVALUE =
: 2726 1
: 2727 1 !+
: 2728 1 ! THIS ROUTINE HANDLES THE ALLOCATION OF AN I/O BUFFER FROM THE BUFFER
: 2729 1 ! POOL.
: 2730 1 !
: 2731 1 ! INPUTS:
: 2732 1 ! ADDR - ADDRESS TO STORE THE 2-WORD BUFFER DESCRIPTOR
: 2733 1 !
: 2734 1 ! IMPLICIT INPUTS:
: 2735 1 ! CCTLR - CURRENT CONTROLLER NUMBER
: 2736 1 !
: 2737 1 ! OUTPUTS:
: 2738 1 ! THE ALLOCATED BUFFER'S DESCRIPTOR IS LOADED INTO THE TWO
: 2739 1 ! WORDS AT "ADDR" AND "ADDR + 2". OTHERWISE, A "-1" IS RETURNED
: 2740 1 ! AT "ADDR" IF NO BUFFERS ARE AVAILABLE.
: 2741 1 !-
: 2743 2 BEGIN
: 2744 2
: 2745 2 .ADDR = -1; ! ASSUME FAILURE
: 2746 2 INCR COUNT FROM 0 TO (.NUM_BUFF - 1) DO ! FOR EACH ENTRY IN BUFFER TABLE
: 2747 3 BEGIN
: 2749 3 IF .BUFF_OWN [.COUNT] LSS 0 ! IF BUFFER IS FREE
: 2750 3 THEN ! THEN
: 2751 4 BEGIN
: 2753 4 BUFF_OWN [.COUNT] = .CCTLR; ! ALLOCATE BUFFER TO CONTROLLER
: 2754 4 .ADDR = .BUFF_DESC [.COUNT, BD_LO]; ! RETURN BUFFER DESCRIPTOR
: 2755 4 (.ADDR + 2) = .BUFF_DESC [.COUNT, BD_HI];
: 2756 4 EXITLOOP; ! DONE
: 2758 3 END;
: 2760 2 END; ! BUFFER TABLE LOOP
: 2762 1 END; ! ROUTINE GET_IO_BUFF

```

```

.SBTTL GET.IO.BUFF GLOBAL ROUTINES
000000 004137 000000G GET.IO.BUFF::
000004 016602 000010 JSR R1,$SAVE2 ; ADDR,* 2725
000010 012712 177777 MOV 10(SP),R2 ; ADDR,* 2745
000014 005001 MOV #-1,(R2) ; COUNT 2746
000016 000420 CLR R1 ; COUNT 2746
000020 105761 000000G 1$: TSTB BUFF.OWN(R1) ; *(COUNT) 2749
000024 002014 BGE 2$
000026 113761 000000G 000000G MOVB CCTLR,BUFF.OWN(R1) ; *,*(COUNT) 2753
000034 010100 MOV R1,R0 ; COUNT,* 2754
000036 006300 ASL R0
000040 006300 ASL R0
000042 016012 000000G MOV BUFF.DESC(R0),(R2)
000046 016062 000002G 000002 MOV BUFF.DESC+2(R0),2(R2) ; 2755
000054 000207 RTS PC ; 2751
000056 005201 2$: INC R1 ; COUNT 2746
000060 020137 000000G 3$: CMP R1,NUM.BUFF ; COUNT,*
000064 002755 BLT 1$
000066 000207 RTS PC ; 2725

```

; Routine Size: 28 words, Routine Base: \$CODE\$ + 5150
; Maximum stack depth per invocation: 4 words

```

: 2763 1 GLOBAL ROUTINE PUT_IO_BUFF (ADDR) : NOVALUE =
: 2764 1
: 2765 1 !*
: 2766 1 ! THIS ROUTINE HANDLES THE DEALLOCATION OF AN I/O BUFFER, RETURNING IT
: 2767 1 ! TO THE BUFFER POOL.
: 2768 1 !
: 2769 1 ! INPUTS:
: 2770 1 ! ADDR - ADDRESS OF THE 2-WORD BUFFER DESCRIPTOR TO BE
: 2771 1 ! DEALLOCATED
: 2772 1 !-
: 2773 1
: 2774 2 BEGIN
: 2775 2
: 2776 2 INCR COUNT FROM 0 TO (.NUM_BUFF - 1) DO ! FOR EACH ENTRY IN BUFFER TABLE
: 2777 3 BEGIN
: 2778 3
: 2779 4 IF ((.BUFF_DESC [.COUNT, BD_LO] EQLA ..ADDR) AND ! IF THIS IS THE BUFFER'S ENTRY
: 2780 4 (.BUFF_DESC [.COUNT, BD_HI] EQLU ..(ADDR + 2)))
: 2781 3 THEN ! THEN
: 2782 4 BEGIN
: 2783 4
: 2784 4 BUFF_OWN [.COUNT] = -1; ! DEALLOCATE BUFFER
: 2785 4 EXITLOOP; ! DONE
: 2786 4
: 2787 3 END;
: 2788 3
: 2789 2 END; ! BUFFER TABLE SEARCH LOOP
: 2790 2
: 2791 1 END; ! ROUTINE PUT_IO_BUFF

```

```

000000 004137 000000G .SBTTL PUT.IO.BUFF GLOBAL ROUTINES
PUT.IO.BUFF::
000004 005002 JSR R1,$SAVE2 ; 2763
000006 000422 CLR R2 ; COUNT 2776
000010 010201 BR 3$
000012 006301 1$: MOV R2,R1 ; COUNT,* 2779
000014 006301 ASL R1
000016 026176 000000G 000010 CMP BUFF.DESC(R1),@10(SP) ; *,ADDR
000024 001012 BNE 2$
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\$ + 5240
; Maximum stack depth per invocation: 4 words


```

: 2792 1 GLOBAL ROUTINE PUTA_BUFF : NOVALUE =
: 2793 1
: 2794 1 !*
: 2795 1 ! THIS ROUTINE DEALLOCATES ALL I/O BUFFERS WHICH HAVE BEEN ALLOCATED TO
: 2796 1 ! THE CURRENT CONTROLLER (CCTLR).
: 2797 1 !-
: 2798 1
: 2799 2 BEGIN
: 2800 2
: 2801 2 INCR COUNT FROM 0 TO (.NUM_BUFF - 1) DO ! FOR EACH ENTRY IN BUFFER TABLE
: 2802 3 BEGIN
: 2803 3
: 2804 3 IF .BUFF_OWN [.COUNT] EQLU .CCTLR ! IF THIS BUFFER IS ALLOCATED TO THE CURRENT CONTROLLER
: 2805 3 THEN ! THEN
: 2806 3 BUFF_OWN [.COUNT] = -1; ! DEALLOCATE IT
: 2807 3
: 2808 2 END; ! BUFFER TABLE ENTRY LOOP
: 2809 2
: 2810 1 END; ! ROUTINE PUTA_BUFF

```

```

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

```

```

; Routine Size: 17 words, Routine Base: $CODE$ + 5324
; Maximum stack depth per invocation: 2 words

```

```

: 2811 1 GLOBAL ROUTINE OUT_IODQ =
: 2812 1
: 2813 1 !+
: 2814 1 ! THIS ROUTINE RETURNS TO THE CALLER THE NEXT RETPKT INDEX TO BE
: 2815 1 ! PROCESSED FROM THE I/O DONE QUEUE (IODQ). THE "OUT" POINTER TO THE
: 2816 1 ! QUEUE IS ALSO UPDATED.
: 2817 1 !
: 2818 1 ! INPUTS:
: 2819 1 ! NONE
: 2820 1 !
: 2821 1 ! OUTPUTS:
: 2822 1 ! THE INDEX OF THE NEXT RETPKT TO BE PROCESSED.
: 2823 1 !-
: 2824 1
: 2825 2 BEGIN
: 2826 2
: 2827 2 LOCAL
: 2828 2 INDEX : WORD;
: 2829 2
: 2830 2 INDEX = .IODQ [.IODQ_OUT]; ! GET NEXT RETPKT INDEX
: 2831 2 IODQ_OUT = .IODQ_OUT + 1; ! ADVANCE "OUT" POINTER
: 2832 2 IF .IODQ_OUT GEQU IODQ_LEN ! IF BEYOND END OF QUEUE
: 2833 2 THEN ! THEN
: 2834 2 IODQ_OUT = 0; ! SET POINTER TO BEGINNING OF QUEUE
: 2835 2 RETURN .INDEX; ! RETURN INDEX TO CALLER
: 2836 2
: 2837 1 END;

```

```

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

```

```

; Routine Size: 15 words, Routine Base: $CODE$ + 5366
; Maximum stack depth per invocation: 0 words

```



```

; 2838 1 GLOBAL ROUTINE IN_IODQ (INDEX) : NOVALUE =
; 2839 1
; 2840 1 !+
; 2841 1 ! THIS ROUTINE INSERTS A RETURN PACKET INDEX INTO THE I/O DONE QUEUE, AND
; 2842 1 ! UPDATES THE IODQ_IN POINTER.
; 2843 1 !-
; 2844 1
; 2845 2 BEGIN
; 2846 2
; 2847 3 IF (((.IODQ_IN + 1) NEQU .IODQ_OUT) AND ! IF I/O DONE QUEUE IS NOT FULL
; 2848 3 (.IODQ_IN - (IODQ_LEN - 1) NEQU .IODQ_OUT))
; 2849 2 THEN ! THEN
; 2850 3 BEGIN
; 2851 3
; 2852 3 IODQ [.IODQ_IN] = .INDEX; ! LOAD INDEX INTO QUEUE
; 2853 3 IODQ_IN = .IODQ_IN + 1; ! ADVANCE "IN" POINTER
; 2854 3 IF .IODQ_IN GEQU IODQ_LEN ! IF BEYOND END OF QUEUE
; 2855 3 THEN ! THEN
; 2856 3 IODQ_IN = 0; ! CYCLE BACK TO BEGINNING OF QUEUE
; 2857 3
; 2858 2 END; ! IF IODQ IS NOT FULL
; 2859 2
; 2860 1 END;

```

```

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

```

```

; Routine Size: 27 words, Routine Base: $CODE$ + 5424
; Maximum stack depth per invocation: 2 words

```

```

: 2861 1 GLOBAL ROUTINE DROP_CTLR (CTLR, REASON) : NOVALUE =
: 2862 1
: 2863 1 !+
: 2864 1 ! THIS ROUTINE DROPS ALL UNITS ASSOCIATED WITH THE CONTROLLER DESIGNATED
: 2865 1 ! BY "CTLR". THE REASON FOR DROPPING THE DEVICE IS LOADED INTO THE DUR
: 2866 1 ! VECTOR FOR EACH ATTACHED UNIT. THIS DATA IS THEN USED BY THE DROP UNIT
: 2867 1 ! SECTION.
: 2868 1 !-
: 2869 1
: 2870 2 BEGIN
: 2871 2
: 2872 2 LOCAL
: 2873 2 UNIT;
: 2874 2
: 2875 2 INCR OFFSET FROM (0 + OF_UN) TO (3 + OF_UN) DO ! FOR EACH UNIT IN CST
: 2876 3 BEGIN
: 2877 3
: 2878 3 IF .CST [.CTLR, .OFFSET, P_PRES] EQLU PRESENT ! IF UNIT IS CONFIGURED
: 2879 3 THEN
: 2880 4 BEGIN
: 2881 4
: 2882 4 UNIT = .CST [.CTLR, .OFFSET, P_UNIT]; ! GET DS UNIT NUMBER
: 2883 4 DUR [.UNIT] = .REASON; ! SET REASON FOR DROPPING UNIT
: 2884 4 DODU (.UNIT); ! DROP UNIT
: 2885 4
: 2886 3 END;
: 2888 2 END;
: 2890 1 END;

```

```

.SBTTL DROP_CTLR GLOBAL ROUTINES
000000 004137 000000G DROP_CTLR::
000004 016646 000014 JSR R1,$SAVE3 ; 2861
000010 012746 000007 MOV 14(SP),-(SP) ; CTLR,* 2878
000014 004737 000000G MOV #7,-(SP)
000020 010003 JSR PC,BL$MUL
000022 012702 000003 MOV R0,R3
000026 010300 1$: MOV #3,R2 ; *,OFFSET 2875
000030 060200 ADD R3,R0 ; 2878
000032 006300 ASL R0 ; OFFSET,*
000034 032760 040000 000000G BIT #40000,CST(R0)
000042 001412 BEQ 2$
000044 016001 000000G MOV CST(R0),R1 ; *,UNIT 2882
000050 000301 SWAB R1 ; UNIT
000052 042701 177740 BIC #177740,R1 ; *,UNIT
000056 115661 000016 000000G MOV# 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)+ ;
000102 000207 RTS PC ; 2870
; 2861

```

; Routine Size: 34 words, Routine Base: \$CODE\$ + 5512
; Maximum stack depth per invocation: 8 words


```

: 2891 1 GLOBAL ROUTINE DRV_CTLERR (CTLR) : NOVALUE =
: 2892 1
: 2893 1 !+
: 2894 1 ! THIS ROUTINE IS CALLED BY DRV_TIMCHK AND FATAL_ERROR WHENEVER AN
: 2895 1 ! UNRECOVERABLE CONTROLLER ERROR HAS BEEN DETECTED. ITS PURPOSE IS TO
: 2896 1 ! CLEAN UP ALL CONTROLLER-RELATED DATA IN THE "DRIVER" PORTION OF THE
: 2897 1 ! PROGRAM. THIS INCLUDES MARKING THE CONTROLLER OFFLINE, CLEARING THE
: 2898 1 ! C-RING COUNT, AND DEALLOCATING MSCP ENVELOPES DESCRIBED IN THE RESPONSE
: 2899 1 ! RING AND OUTSTANDING COMMAND AREA (OUTC_LIST).
: 2900 1 !
: 2901 1 ! INPUTS:
: 2902 1 ! CTLR - DYING CONTROLLER NUMBER
: 2903 1 !-
: 2904 1
: 2905 2 BEGIN
: 2906 2
: 2907 2 LOCAL
: 2908 2 D_ADDR : REF BLOCK [DCT_LEN, WORD] FIELD (DC_FIELDS), ! CONTROLLER'S DCT ADDRESS
: 2909 2 INDEX : WORD,
: 2910 2 STIDX : WORD, ! STARTING OUTC INDEX
: 2911 2 ENIDX : WORD; ! ENDING OUTC INDEX
: 2912 2
: 2913 2 D_ADDR = DCT + (.CTLR * DCT_LEN * 2); ! GET CONTROLLER'S DCT ADDR
: 2914 2 D_ADDR [WORD] = 0; ! MARK DCT OFFLINE AND CLEAR CRING_CNT
: 2915 2 PUTA_ENV (.CTLR); ! RELEASE ALL ENVELOPES ALLOCATED TO CONTROLLER
: 2916 2 STIDX = .CTLR * OUTC_CNT; ! START OF CTLR'S OUTC_LIST AND _TIMR
: 2917 2 ENIDX = .STIDX + OUTC_CNT - 1; ! END OF OUTC
: 2918 2 INCR COUNT FROM .STIDX TO .ENIDX DO ! FOR EACH OUTC ENTRY
: 2919 3 BEGIN
: 2920 3
: 2921 3 OUTC_TMR [.COUNT] = 0; ! TURN OF COMMAND TIMER
: 2922 3 OUTC_LIST [.COUNT] = -1; ! INIT SLOT
: 2923 3
: 2924 2 END; ! OUTC SLOT LOOP
: 2925 2
: 2926 1 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	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	004766'	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

CZRC D2
V02.0

CZRCDB0 RC25 DISK EXERCISER
GLOBAL ROUTINES

14-Jun-1985 09:36:36
14-Jun-1985 09:32:03

VAX-11 Bliss-16 V4.0-579
USER\$1:[AZTEC.CZRCDB]CZRC D2.SRC;8

SEQ 0164
Page 70
(28)

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		INC	R1	:	COUNT	2918
000104	020102	2\$:	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\$ + 5616
: Maximum stack depth per invocation: 6 words


```

: 2927 1 GLOBAL ROUTINE HARD_ERR (ERRCNT) : NOVALUE =
: 2928 1
: 2929 1 !+
: 2930 1 ! THIS ROUTINE IS CALLED BY IO_RETPKT, DM_TALLY, AND OTHERS TO UPDATE THE
: 2931 1 ! HARD ERROR STATISTIC FIELD FOR THE CURRENT UNIT. IF THE HARD ERROR
: 2932 1 ! COUNT HAS EXCEEDED THE OPERATOR-SPECIFIED LIMIT, THEN THE UNIT IS
: 2933 1 ! DROPPED FROM TESTING.
: 2934 1 !
: 2935 1 ! INPUTS:
: 2936 1 ! ERRCNT - NUMBER OF HARD ERRORS TO ADD TO CURRENT TOTAL
: 2937 1 !
: 2938 1 ! IMPLICIT INPUTS:
: 2939 1 ! L$LUN - CURRENT UNIT NUMBER
: 2940 1 ! CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
: 2941 1 ! CUOFF - CST OFFSET FOR CURRENT UNIT
: 2942 1 ! T_ADDR - ADDRESS OF CURRENT UNIT'S STATISTICS BLOCK (TALLY)
: 2943 1 !-
: 2944 1
: 2945 2 BEGIN
: 2946 2
: 2947 2 T_ADDR [ER_HRD] = .T_ADDR [ER_HRD] + .ERRCNT; ! UPDATE UNIT'S HARD ERROR COUNT
: 2948 2 IF .SWP_ERROR NEQU 0 ! IF THE USER SPECIFIED A HARD ERROR LIMIT
: 2949 2 THEN ! THEN
: 2950 3 BEGIN
: 2951 3
: 2952 3 IF .T_ADDR [ER_HRD] GEQU .SWP_ERROR ! IF HARD ERROR LIMIT REACHED
: 2953 3 THEN ! THEN
: 2954 4 BEGIN
: 2955 4
: 2956 4 IF .CST_ADDR [.CUOFF, P_STAT] EQLU ONLINE ! IF UNIT IS STILL ONLINE
: 2957 4 THEN ! THEN
: 2958 5 BEGIN
: 2959 5
: 2960 5 DUR [.L$LUN] = DU_HERR; ! LOAD REASON TO DROP UNIT
: 2961 5 DODU (.L$LUN); ! DROP UNIT
: 2962 5
: 2963 4 END;
: 2964 4
: 2965 3 END; ! IF HARD ERROR LIMIT REACHED
: 2966 3
: 2967 2 END; ! IF THE USER SPECIFIED A HARD ERROR LIMIT
: 2968 2
: 2969 1 END; ! ROUTINE HARD_ERR

```

```

000000 010146 .SBTTL HARD.ERR GLOBAL ROUTINES
000002 013700 000000G MOV R1, -(SP) ; 2927
000006 066660 000004 000030 MOV T.ADDR, R0 ; 2947
000014 013701 000000G ADD 4(SP), 30(R0) ; ERRCNT, *
000020 001421 000000G MOV SWP.ERROR, R1 ; 2948
000022 026001 000030 BEQ 1$ ;
000026 103416 000000G CMP 30(R0), R1 ; 2952
000030 013700 000000G BLO 1$ ;
000034 006300 000000G MOV CUOFF, R0 ; 2956
000035 063700 000000G ASL R0 ;
ADD CST.ADDR, R0

```

CZRC2
V02.0

CZRCDB0 RC25 DISK EXERCISER
GLOBAL ROUTINES

14-Jun-1985 09:36:36
14-Jun-1985 09:32:03

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC2.SRC;8

SEQ 0166
Page 72
(29)

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\$ + 5732
: Maximum stack depth per invocation: 3 words


```
; 2970 1 GLOBAL ROUTINE UPD_IOC (ADDR, COUNT) : NOVALUE =
; 2971 1
; 2972 1 !+
; 2973 1 ! THIS ROUTINE IS CALLED (A) DURING THE MULTI-DRIVE SUBTEST FROM MD_TALLY
; 2974 1 ! FOR ALL I/O TRANSFER RETURN PACKETS WITH "SUCCESS" STATUS CODES, AND
; 2975 1 ! (B) DURING THE DM EXERCISER SUBTEST FROM DM_TALLY WHEN EXAMINING THE
; 2976 1 ! COMMUNICATION AREA (DM_COMM). ITS PURPOSE IS TO UPDATE THE I/O COUNT
; 2977 1 ! (NUMBER OF READS, WRITES, OR SEEKS) FOR THE CURRENT UNIT.
; 2978 1 !
; 2979 1 ! AFTER THE LOW-ORDER FIELD OF THE APPROPRIATE COUNT IS UPDATED, THIS
; 2980 1 ! ROUTINE CHECKS FOR ONE OF TWO OVERFLOW CONDITIONS: (A) IF THE
; 2981 1 ! HIGH-ORDER FIELD OF THE COUNT IS ZERO, THEN THE LOW-ORDER FIELD IS
; 2982 1 ! CHECKED FOR EXCEEDING 65,000. IF TRUE, THEN THE HIGH-ORDER FIELD IS
; 2983 1 ! LOADED WITH 65 (FOR 65,000) AND THE LOW-ORDER FIELD IS REDUCED BY
; 2984 1 ! 65,000. (B) IF THE HIGH-ORDER FIELD OF THE UPDATED COUNT IS NON-ZERO,
; 2985 1 ! THEN THE LOW-ORDER FIELD IS CHECKED FOR EXCEEDING 1000. IF TRUE, THEN
; 2986 1 ! THE HIGH-ORDER FIELD IS INCREMENTED BY 1 FOR EACH 1000 SUBTRACTED FROM
; 2987 1 ! THE LOW-ORDER FIELD.
; 2988 1 !
; 2989 1 ! INPUTS:
; 2990 1 ! ADDR - ADDRESS OF THE LOW-ORDER FIELD OF THE I/O COUNT TO BE
; 2991 1 ! UPDATED (IN THE CURRENT UNIT'S STATISTICS TABLE (TALLY))
; 2992 1 ! COUNT - AMOUNT TO BE ADDED TO THE I/O COUNT
; 2993 1 !-
; 2994 1
; 2995 2 BEGIN
; 2996 2
; 2997 2 .ADDR = ..ADDR + .COUNT; ! UPDATE THE COUNT (LOW-ORDER FIELD)
; 2998 2 IF (.ADDR + 2) EQLU 0 ! IF HIGH-ORDER FIELD IS ZERO
; 2999 2 THEN ! THEN
; 3000 3 BEGIN
; 3001 3
; 3002 3 IF ..ADDR GEQU 65000 ! IF LOW-ORDER FIELD EXCEEDS 65,000
; 3003 3 THEN ! THEN
; 3004 4 BEGIN
; 3005 4
; 3006 4 .ADDR = ..ADDR - 65000; ! BECOMES LOW-ORDER 1000
; 3007 4 (.ADDR + 2) = 65; ! HIGH-ORDER (TIMES 1000)
; 3008 4
; 3009 3 END;
; 3010 3
; 3011 3 END
; 3012 2 ELSE ! ELSE - HIGH-ORDER FIELD IS X1000
; 3013 3 BEGIN
; 3014 3
; 3015 3 WHILE ..ADDR GEQU 1000 DO ! WHILE LOW-ORDER FIELD EXCEEDS 1000
; 3016 4 BEGIN
; 3017 4
; 3018 4 .ADDR = ..ADDR - 1000; ! SUBTRACT 1000 FROM LOW-ORDER FIELD
; 3019 4 (.ADDR + 2) = .(.ADDR + 2) + 1; ! INCREMENT HIGH-ORDER FIELD
; 3020 4
; 3021 3 END;
; 3022 3
; 3023 2 END;
; 3024 2
; 3025 1 END; ! ROUTINE UPD_IOC
```

		.SBTTL	UPD.IOC GLOBAL ROUTINES		
000000	C:0146	UPD.IOC::			
		MOV	R1, -(SP)	:	2970
000002	016600	MOV	6(SP), R0	: ADDR, *	2997
000006	066610	ADD	4(SP), (R0)	: COUNT, *	
000012	012701	MOV	#2, R1	:	2998
000016	060001	ADD	R0, R1		
000020	005711	TST	(R1)		
000022	001010	BNE	1#		
000024	021027	CMP	(R0), #176750	:	3002
000030	103414	BLO	2#		
000032	062710	ADD	#1030, (R0)	:	3006
000036	012711	MOV	#101, (R1)	:	3007
000042	000407	BR	2#	:	2998
000044	021027	CMP	(R0), #1750	:	3015
000050	103404	BLO	2#		
000052	162710	SUB	#1750, (R0)	:	3018
000056	005211	INC	(R1)	:	3019
000060	000771	BR	1#	:	3015
000062	012601	MOV	(SP)+, R1	:	2970
000064	000207	RTS	PC	:	

; Routine Size: 27 words, Routine Base: \$CODE\$ + 6022
; Maximum stack depth per invocation: 2 words


```

; 3026 1 GLOBAL ROUTINE OVF_CHK (ADDR) : NOVALUE =
; 3027 1
; 3028 1 !+
; 3029 1 ! THIS ROUTINE IS CALLED (A) DURING THE MULTI-DRIVE SUBTEST FROM MD_TALLY
; 3030 1 ! FOR ALL I/O TRANSFER RETURN PACKETS WITH "SUCCESS" STATUS CODES, AND
; 3031 1 ! (B) DURING THE DM EXERCISER SUBTEST FROM DM_TALLY WHEN EXAMINING THE
; 3032 1 ! COMMUNICATION AREA (DM_COMM). ITS PURPOSE IS TO CHECK FOR OVERFLOW IN
; 3033 1 ! CERTAIN STATISTICAL FIELDS OF THE CURRENT UNIT. SPECIFICALLY, THE
; 3034 1 ! LOW-ORDER FIELD OF THE NUMBER OF BYTES READ OR WRITTEN IS CHECKED FOR
; 3035 1 ! EXCEEDING 1000. IF TRUE THEN THE HIGH-ORDER COUNT IS INCREMENTED. IF
; 3036 1 ! THAT EXCEEDS 1000, THEN THE MEGABYTE COUNT IS INCREMENTED.
; 3037 1 !
; 3038 1 ! INPUTS:
; 3039 1 ! ADDR - ADDRESS OF THE BR_LO OR BW_LO FIELD FOR THE CURRENT UNIT
; 3040 1 ! (SEE STATISTIC TABLE (TALLY) LAYOUT)
; 3041 1 !-
; 3043 2 BEGIN
; 3044 2
; 3045 2 WHILE ..ADDR GEQU 1000 DO ! IF LO-ORDER OVERFLOW
; 3046 3 BEGIN
; 3047 3
; 3048 3 .ADDR = ..ADDR - 1000; ! SUBTRACT 1000
; 3049 3 (.ADDR + 2) = .(.ADDR + 2) + 1; ! INCR HI-ORDER
; 3050 3
; 3051 2 END;
; 3052 2
; 3053 2 IF .(.ADDR + 2) GEQU 1000 ! IF HI-ORDER OVERFLOW
; 3054 2 THEN ! THEN
; 3055 3 BEGIN
; 3056 3
; 3057 3 (.ADDR + 2) = .(.ADDR + 2) - 1000; ! SUBTRACT 1000
; 3058 3 (.ADDR + 4) = .(.ADDR + 4) + 1; ! INCREMENT MBYTES
; 3059 3
; 3060 2 END;
; 3062 1 END; ! ROUTINE OVF_CHK

```

```

000000 010146 .SBTTL OVF.CHK GLOBAL ROUTINES
OVF.CHK::
000002 016600 000004 MOV R1, -(SP) ;
000006 012701 000002 MOV 4(SP), R0 ; ADDR, *
000012 060001 MOV #2, R1 ;
000014 021027 001750 ADD R0, R1 ;
000020 103404 1#: CMP (R0), #1750 ;
000022 162710 001750 BLO 2# ;
000026 005211 SUB #1750, (R0) ;
000030 000771 INC (R1) ;
000032 021127 001750 BR 1# ;
000036 103404 2#: CMP (R1), #1750 ;
000040 162711 001750 BLO 3# ;
000044 005260 000004 SUB #1750, (R1) ;
000050 012601 3#: INC 4(R0) ;
000052 000207 MOV (SP)+, R1 ;
RTS PC ;

```

```

; Routine Size: 22 words, Routine Base: $CODE$ + 6110
; Maximum stack depth per invocation: 2 words

```

```

: 3063 1 GLOBAL ROUTINE XFR_CHK : NOVALUE =
: 3064 1
: 3065 1 !*
: 3066 1 ! THIS ROUTINE IS CALLED (A) DURING THE MULTI-DRIVE SUBTEST FROM MD_TALLY
: 3067 1 ! FOR ALL I/O TRANSFER RETURN PACKETS WITH "SUCCESS" STATUS CODES, AND
: 3068 1 ! (B) DURING THE DM EXERCISER SUBTEST FROM DM_TALLY WHEN EXAMINING THE
: 3069 1 ! COMMUNICATION AREA (DM_COMM). ITS PURPOSE IS TO CALCULATE THE TOTAL
: 3070 1 ! NUMBER OF BYTES TRANSFERRED THUS FAR ON THE CURRENT UNIT, AND TO CHECK
: 3071 1 ! THIS SUM AGAINST THE OPERATOR-SPECIFIED LIMIT. IF THE LIMIT HAS BEEN
: 3072 1 ! REACHED, THEN THE UNIT IS REMOVED FROM THE CURRENT PASS.
: 3073 1 !
: 3074 1 ! IMPLICIT INPUTS:
: 3075 1 ! T_ADDR - ADDRESS OF THE CURRENT UNIT'S STATISTICS BLOCK (TALLY)
: 3076 1 ! L$LUN - CURRENT UNIT NUMBER
: 3077 1 ! CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
: 3078 1 ! DMC_ADDR - ADDRESS OF CURRENT CONTROLLER'S DM_COMM AREA
: 3079 1 ! CUOFF - CST OFFSET FOR CURRENT UNIT
: 3080 1 !-
: 3081 1
: 3082 2 BEGIN
: 3083 2
: 3084 2 LOCAL
: 3085 2 SUM : WORD; ! TOTAL NUMBER OF BYTES TRANSFERRED TO/FROM A UNIT
: 3086 2
: 3087 2 IF .SWP_XFER NEQU 0 ! IF A TRANSFER LIMIT WAS SPECIFIED
: 3088 2 THEN ! THEN
: 3089 3 BEGIN
: 3090 3
: 3091 3 SUM = .T_ADDR [MB_READ] + .T_ADDR [MB_WRIT]; ! TOTAL BYTES TRANSFERRED
: 3092 3 IF .SUM GEQU .SWP_XFER ! IF TRANSFER LIMIT IS REACHED
: 3093 3 THEN ! THEN
: 3094 4 BEGIN
: 3095 4
: 3096 4 CPT [.L$LUN] = NO; ! MARK UNIT INACTIVE
: 3097 4 CST_ADDR [U_CNT] = .CST_ADDR [U_CNT] - 1; ! DECREMENT ACTIVE UNIT COUNT
: 3098 4 DMC_ADDR [.CUOFF - OF_UN, ALLBIT] = -1; ! IF DM EXER, STOP FPT ON UNIT
: 3099 5 IF MANUAL ! IF ATTENDED
: 3100 4 THEN ! THEN
: 3101 5 BEGIN
: 3102 5
: 3103 5 PRINTF (CRLF);
: 3104 5 PRINTF (ETIME, .HOURS, .MINUTES, .SECONDS); ! ELAPSED TIME
: 3105 5 PRINTF (MSG_07, .L$LUN); ! "UNIT XX. - TRANSFER LIMIT REACHED"
: 3106 5
: 3107 4 END;
: 3108 4
: 3109 3 END;
: 3110 3
: 3111 2 END; ! IF A TRANSFER LIMIT WAS SPECIFIED
: 3112 2
: 3113 1 END; ! ROUTINE XFR_CHK

```

000000 004137 000000G
000004 013702 000000G

.SBTTL XFR.CHK GLOBAL ROUTINES
XFR.CHK::
JSR R1,\$SAVE2
MOV SWP.XFER,R2

3063
3087

CZRC02
V02.0

CZRCDB0 RC25 DISK EXERCISER
GLOBAL ROUTINES

14-Jun-1985 09:36:36
14-Jun-1985 09:32:03

VAX-11 Bliss-16 V4.0-579
USER\$1:[AZTEC.CZRCDB]CZRC02.SRC;8

SEQ 0171
Page 78
(32)

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\$			
000034	013700	000000G	MOV	L\$LUN,R0	:		3096
000040	105060	000000G	CLRB	CPT(R0)			
000044	013700	000000G	MOV	CST.ADDR,R0	:		3097
000050	105360	000005	DECB	5(R0)			
000054	013700	000000G	MOV	CUOFF,R0	:		3098
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\$			
000100	012746	000000G	MOV	#CRLF,-(SP)	:		3103
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		RTS	PC	:		3063

: Routine Size: 61 words, Routine Base: \$CODE\$ + 6164
: Maximum stack depth per invocation: 13 words

```
: 3114 1 GLOBAL ROUTINE SEND (INDEX) =
: 3115 1
: 3116 1 !*
: 3117 1 !
: 3118 1 ! IF THE CURRENT RC25 IS ONLINE AND ITS CRING IS NOT FULL, THEN THIS
: 3119 1 ! ROUTINE "SENDS" A COMMAND TO THE RC25 BY LOADING THE ENVELOPE
: 3120 1 ! DESCRIPTOR OF AN MSCP ENVELOPE INTO THE COMMAND RING AND READING THE
: 3121 1 ! DEVICE'S IP REGISTER. A RECORD OF THE SENT COMMAND IS MADE IN THE
: 3122 1 ! CONTROLLER'S OWN OUTSTANDING COMMAND LIST (OUTC_LIST), AND A COMMAND
: 3123 1 ! TIMER IS STARTED IN THE OUTSTANDING COMMAND TIMER (OUTC_TIMR). IF THE
: 3124 1 ! CURRENT RC25 IS NOT ONLINE, THEN A FAILURE INDICATION IS RETURNED TO
: 3125 1 ! THE CALLER, AND NO ACTION IS TAKEN.
: 3126 1 !
: 3127 1 ! INPUTS:
: 3128 1 ! INDEX - INDEX OF MSCP ENVELOPE CONTAINING THE COMMAND TO
: 3129 1 ! BE SENT
: 3130 1 !
: 3131 1 ! IMPLICIT INPUTS:
: 3132 1 ! CCTRL - CURRENT CONTROLLER NUMBER
: 3133 1 ! DCT_ADDR - ADDRESS OF CURRENT CONTROLLER'S DCT
: 3134 1 ! OCL_X1, OCL_X2 - STARTING/ENDING INDECES OF CURRENT
: 3135 1 ! CONTROLLER'S OUTSTANDING COMMAND AREA
: 3136 1 !-
: 3137 2 BEGIN
: 3138 2
: 3139 2 LOCAL
: 3140 2 SLOT_ADDR,
: 3141 2 TEMP : WORD;
: 3142 2
: 3143 3 IF ((.DCT_ADDR [STAT] EQLU ONLINE) AND ! IF DEVICE IS ONLINE AND
: 3144 3 (.DCT_ADDR [CRING_CNT] LSSU CRING_LEN)) ! ITS CRING IS NOT FULL
: 3145 2 THEN
: 3146 3 BEGIN
: 3147 3
: 3148 3 MSCP_ENV [.INDEX, CRN_LO] = (CRN = .CRN + 1); ! ASSIGN CMD REF NUM
: 3149 3 SLOT_ADDR = .DCT_ADDR [CR_NEXT]; ! ADDR OF NEXT COMMAND SLOT
: 3150 3 .SLOT_ADDR = .MSCP_ENV [.INDEX, ENV_LO]; ! LOAD ENV DESC (LO) INTO COMMAND SLOT
: 3151 3 SLOT_ADDR = .SLOT_ADDR + 2; ! ADVANCE TO NEXT WORD
: 3152 3 .SLOT_ADDR = .MSCP_ENV [.INDEX, ENV_HI]; ! LOAD ENV DESC (HI) INTO COMMAND SLOT
: 3153 3 SLOT_ADDR = .SLOT_ADDR + 2; ! ADVANCE TO NEXT COMMAND SLOT
: 3154 3 IF .SLOT_ADDR GTRA .DCT_ADDR [CR_END] ! IF BEYOND END OF CRING
: 3155 3 THEN ! THEN
: 3156 3 SLOT_ADDR = .DCT_ADDR [CR_BEG]; ! CYCLE BACK TO BEGINNING
: 3157 3 DCT_ADDR [CR_NEXT] = .SLOT_ADDR; ! RESTORE CR_NEXT POINTER IN DCT
: 3158 3 DCT_ADDR [CRING_CNT] = .DCT_ADDR [CRING_CNT] + 1; ! INCR # OF COMMANDS IN CRING
: 3159 3
: 3160 3 INCR COUNT FROM .OCL_X1 TO .OCL_X2 DO ! FOR EACH OUTC ENTRY
: 3161 4 BEGIN
: 3162 4
: 3163 4 IF .OUTC_LIST [.COUNT] LSS 0 ! IF A SPARE SLOT IS FOUND
: 3164 4 THEN
: 3165 5 BEGIN
: 3166 5
: 3167 5 OUTC_LIST [.COUNT] = .INDEX; ! LOAD MSCP_ENV INDEX
: 3168 5 IF .IIP_FLAG ! IF INIT SUBTEST IN PROGRESS
: 3169 5 THEN ! THEN
: 3170 5 OUTC_TIMR [.COUNT] = TO_INIT ! SET INIT SUBTEST COMMAND TIMER
```


CZRC2
V02.0

CZRCDBO RC25 DISK EXERCISER
GLOBAL ROUTINES

14-Jun-1985 09:36:36
14-Jun-1985 09:32:03

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC2.SRC;8

SEQ 0174
Page 81
(33)

000140	006300			ASL	R0			
000142	052700	000000G		ADD	#0UTC.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	BITB	#2,SWP.FLAGS			3174
000172	001403			BEQ	4\$			
000174	012710	000036		MOV	#36,(R0)			3176
000200	000407			BR	6\$			3174
000202	012710	000454		MOV	#454,(R0)			3178
000206	000404			BR	6\$			3165
000210	005201			INC	R1		COUNT	3160
000212	020137	000000G		CMP	R1,0CL.X2		COUNT,*	
000216	003741			BLE	2\$			
000220	017766	000000G	000004	MOV	@RC25.ADDR,4(SP)		*,RC.REG	3188
000226	022626			CMP	(SP)+,(SP)+			3146
000230	012700	000001		MOV	#1,R0			3193
000234	000401			BR	8\$			
000236	005000			CLR	R0			
000240	005726			TST	(SP)+			
000242	000207			RTS	PC			3114

; Routine Size: 82 words, Routine Base: \$CODE\$ + 6356
; Maximum stack depth per invocation: 7 words


```

: 3253 6      OUTC_TIMR [.CX] = .OUTC_TIMR [.CX] - 1; ! DECREMENT TIMER
: 3254 6      IF .OUTC_TIMR [.CX] EQLU 0           ! IF MESSAGE HAS TIMED OUT
: 3255 6      THEN                               ! THEN
: 3256 7      BEGIN
: 3257 7
: 3258 7      RX = GET_RETPKT (.CCTLR);           ! GET A RETURN PACKET
: 3259 7      MX = .OUTC_LIST [.CX];             ! GET COMMAND ENV INDEX
: 3260 7      R_ADDR = RETPKT + (.RX * RP_LEN * 2); ! CALCULATE RETPKT ADDRESS
: 3261 7      M_ADDR = MSCP_ENV + (.MX * ENV_LEN * 2); ! CALCULATE ENV ADDR
: 3262 7      COPY_BLK (.M_ADDR + 4, .R_ADDR, RP_LEN); ! COPY COMMAND TO RETPKT
: 3263 7      R_ADDR [CONID] = CID_DRIVER;        ! SET PACKET SOURCE AS "DRIVER"
: 3264 7      R_ADDR [MESTYP] = MT_TIMEOUT;      ! COMMAND TIMEOUT
: 3265 7      R_ADDR [CTLR] = .CCTLR;           ! CONTROLLER NUMBER
: 3266 7      IN_IODQ (.RX);                     ! "SEND" THE PACKET
: 3267 7      PUT_ENV (.MX);                     ! RETURN COMMAND ENV TO POOL
: 3268 7      OUTC_LIST [.CX] = -1;             ! INIT OUTC_LIST ENTRY
: 3269 7
: 3270 6      END;                               ! IF MESSAGE TIMER EXPIRED
: 3271 6
: 3272 5      END;                               ! IF TIMER STILL ACTIVE
: 3273 5
: 3274 5      SETPRI (PRI00);                    ! RESTORE INIT SUBTEST PRIORITY
: 3275 5
: 3276 4      END;                               ! IF OUTSTANDING COMMAND WAS FOUND
: 3277 4
: 3278 4      T_FLAG = FALSE;                   ! CLEAR ONE SECOND TIMING FLAG
: 3279 4
: 3280 3      END;                               ! IF ONE SECOND HAS ELAPSED
: 3281 3
: 3282 3      END                               ! UNTIL A PACKET IS "SENT" FROM HERE
: 3283 2      UNTIL .IODQ_IN NEQU .IODQ_OUT;    ! OR RC25 INTERRUPT PROCESSING
: 3284 2
: 3285 1      END;                               ! ROUTINE WAIT

```

```

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

```


CZRCDD
V02.0

CZRCDB0 RC25 DISK EXERCISER
GLOBAL ROUTINES

14-Jun-1985 09:36:36
14-Jun-1985 09:32:03

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRCDD.SRC;8

SEQ 0177
Page 84
(34)

000074	062700	000000G	ADD	#OUTC.TIMR,R0		
000100	005710		TST	(R0)		
000102	001471		BEQ	5\$		
000104	005310		DEC	(R0)		
000106	001067		BNE	5\$		3253
000110	013746	000000G	MOV	CCTLR,-(SP)		3254
000114	004737	005026'	JSR	PC,GET.RETPKT		3258
000120	010004		MOV	R0,R4	; *,RX	
000122	116103	000000G	MOVB	OUTC.LIST(R1),R3	; *(CX),MX	3259
000126	010416		MOV	R4,(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	010316		MOV	R3,(SP)	; MX,*	3261
000150	012746	000104	MOV	#104,-(SP)		
000154	004737	000000G	JSR	PC,BL\$MUL		
000160	062700	000000G	ADD	#MSCP.ENV,R0		
000164	010005		MOV	R0,R5	; *,M.ADDR	
000166	010516		MOV	R5,(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	004310'	JSR	PC,COPY.BLK		
000206	012700	000002	MOV	#2,R0		
000212	060200		ADD	R2,R0	; R.ADDR,*	3263
000214	112760	000003 000001	MOVB	#3,1(R0)		
000222	013746	000000G	MOV	CCTLR,-(SP)		
000226	042716	177760	BIC	#177760,(SP)		3265
000232	112710	000100	MOVB	#100,(R0)		
000236	152610		BISB	(SP)+,(R0)		
000240	010416		MOV	R4,(SP)	; RX,*	3266
000242	004737	005424'	JSR	PC,IN.IODQ		
000246	010316		MOV	R3,(SP)	; MX,*	3267
000250	004737	004752'	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\$ + 6622
; Maximum stack depth per invocation: 12 words

```

; 3286 1  *SBTTL 'ERROR MESSAGE SUBROUTINES'
; 3287 1
; 3288 1  ROUTINE EMS_ET : NOVALUE =
; 3289 1
; 3290 1  !+
; 3291 1  !      THIS ROUTINE PRINTS THE ELAPSED TIME THAT PRECEDES ALL BASIC ERROR
; 3292 1  !      MESSAGES.
; 3293 1  !-
; 3294 1
; 3295 1  PRINTB (ETIME, .HOURS, .MINUTES, .SECONDS);

```

```

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

```

```

; Routine Size: 15 words,      Routine Base: $CODE$ + 7132
; Maximum stack depth per invocation: 7 words

```

```

; 3296 1  ROUTINE EMS_SA : NOVALUE =
; 3297 1
; 3298 1  !+
; 3299 1  !      THIS ROUTINE PRINTS (EXTENDED) THE GLOBAL DATUM "SA_REG" WHICH CONTAINS
; 3300 1  !      THE CONTENTS OF THE SA REGISTER.
; 3301 1  !-
; 3302 1
; 3303 1  PRINTX (EX_SA, .SA_REG);          ! " SA: XXXXXX(0)"

```

```

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

```

```

; Routine Size: 11 words,      Routine Base: $CODE$ + 7170
; Maximum stack depth per invocation: 5 words

```

```

; 3304 1  ROUTINE EMS_CRN : NOVALUE =
; 3305 1
; 3306 1  !+
; 3307 1  !      THIS ROUTINE PRINTS (EXTENDED) THE COMMAND REFERENCE NUMBER OF THE
; 3308 1  !      CURRENT RETURN PACKET. ONLY THE LOW-ORDER WORD IS SIGNIFICANT.
; 3309 1  !-
; 3310 1

```



```
; 3311 1 PRINTX (EX_CRN, .RP_ADDR [CRF_LO]); ! " CMD REF NUM: XXXXXX(0)"
```

```

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

```

```
; Routine Size: 13 words, Routine Base: $CODE$ + 7216
; Maximum stack depth per invocation: 5 words
```

```
; 3312 1 ROUTINE EMS_STC : NOVALUE =
; 3313 1
; 3314 1 !+
; 3315 1 ! THIS ROUTINE PRINTS (EXTENDED) THE GLOBAL DATUM "ST_CODE" (STATUS CODE)
; 3316 1 ! IF IT IS NON-ZERO.
; 3317 1 !-
; 3318 1
; 3319 1 IF .ST_CODE NEQU 0
; 3320 1 THEN
; 3321 1 PRINTX (EX_SC, .ST_CODE); ! " STATUS CODE: XX(0)"
```

```

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

```

```
; Routine Size: 13 words, Routine Base: $CODE$ + 7250
; Maximum stack depth per invocation: 5 words
```

```
; 3322 1 ROUTINE EMS_SBC : NOVALUE =
; 3323 1
; 3324 1 !+
; 3325 1 ! THIS ROUTINE PRINTS (EXTENDED) THE GLOBAL DATUM "SB_CODE" (SUB-CODE) IF
; 3326 1 ! EITHER THE STATUS CODE (ST_CODE) OR THE SUB-CODE IS NON-ZERO. (A
; 3327 1 ! NON-ZERO SUB-CODE ALWAYS HAS SIGNIFICANCE, WHEREAS A ZERO SUB-CODE ONLY
; 3328 1 ! HAS MEANING WITH A NON-ZERO STATUS CODE).
; 3329 1 !-
; 3330 1
; 3331 1 IF (.ST_CODE OR .SB_CODE) NEQU 0
; 3332 1 THEN
; 3333 1 PRINTX (EX_SB, .SB_CODE); ! " SUB-CODE: XXXX(0)"
```

```

          .SBTTL  EMS.SBC ERROR MESSAGE SUBROUTINES
000000 013700 000000G      EMS.SBC:MOV  ST.CODE,RO      ;          3331
000004 053700 000000G      BIS      SB.CODE,RO
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,RO      ; SP,*
000030 104415              TRAP     15
000032 062706 000006      ADD      #6,SP
000036 000207              1$:   RTS      PC      ;          3322

```

```

; Routine Size: 16 words,      Routine Base: $CODE$ + 7302
; Maximum stack depth per invocation: 5 words

```

```

; 3334 1  ROUTINE EMS_CMD : NOVALUE =
; 3335 1
; 3336 1  !+
; 3337 1  ! THIS ROUTINE PRINTS (EXTENDED) THE OPCODE AND COMMAND MODIFIER (IF
; 3338 1  ! PRESENT) OF THE CURRENT RETURN PACKET. THESE FIELDS ARE "TRANSLATED"
; 3339 1  ! INTO ENGLISH TEXT RATHER THAN PRINTED AS RAW NUMBERS.
; 3340 1  !
; 3341 1  ! IMPLICIT INPUTS:
; 3342 1  ! RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
; 3343 1  !-
; 3344 1
; 3345 2  BEGIN
; 3346 2
; 3347 2  LOCAL
; 3348 2  COMMAND : WORD;          ! COMMAND OPCODE
; 3349 2
; 3350 2  PRINTX (EX_CMD);          ! " COMMAND: "
; 3351 2  COMMAND = .RP_ADDR [ENDCOD] AND OP_MSK;      ! GET OPCODE ALONE
; 3352 2  SELECTONEU .COMMAND OF
; 3353 2  SET
; 3354 2
; 3355 2  [OP_ESP] : PRINTX (EX_ESP);      ! "EXECUTE SUPPLIED PROGRAM"
; 3356 2  [OP_SCC] : IF .IIP_FLAG          ! IF INIT SUBTEST IN PROGRESS
; 3357 2  THEN          ! THEN (THESE 2 OPCODES ARE THE SAME)
; 3358 3  PRINTX (EX_SCC)          ! "SET CTLR CHAR"
; 3359 2  ELSE          ! OTHERWISE
; 3360 2  PRINTX (EX_SND);          ! "SEND DATA"
; 3361 2  [OP_ONL] : PRINTX (EX_ONL);      ! "ONLINE"
; 3362 2  [OP_RD] : PRINTX (EX_RD);        ! "READ"
; 3363 2  [OP_WRT] : PRINTX (EX_WRT);      ! "WRITE"
; 3364 2  [OTHERWISE] : PRINTX (EX_03, .RP_ADDR [ENDCOD]); ! "XXX(0)"
; 3365 2
; 3366 2  TES;
; 3367 2
; 3368 3  IF (((.COMMAND EQLU OP_RD) OR (.COMMAND EQLU OP_WRT)) AND
; 3369 3  (BIT_TST (RP_ADDR [CMDMOD], MD_CMP)))      ! IF COMPARE MODIFIER IS PRESENT
; 3370 2  THEN          ! THEN
; 3371 3  PRINTX (EX_CMP)          ! "-COMPARE<CR><LF>"
; 3372 2  ELSE          ! OTHERWISE
; 3373 2  PRINTX (CRLF);          ! <CR><LF>

```


; 3374 2
; 3375 1 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,*
000040	001007			BNE 1\$	
000042	012716	000000G		MOV #EX.ESP,(SP)	
000046	012746	000001		MOV #1,-(SP)	
000052	010600			MOV SP,R0	; SP,*
000054	104415			TRAP 15	
000056	000462			BR 6\$	
000060	020227	000004	1\$:	CMP R2,#4	; COMMAND,*
000064	001022			BNE 3\$	
000066	132737	000001 000000G		BITB #1,IIP.FLAG	
000074	0C1407			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	000444			BR 6\$	
000114	012716	000000G	2\$:	MOV #EX.SND,(SP)	3356
000120	012746	000001		MOV #1,-(SP)	3360
000124	010600			MOV SP,R0	; SP,*
000126	104415			TRAP 15	
000130	000435			BR 6\$	
000132	020227	000011	3\$:	CMP R2,#11	; COMMAND,*
000136	001007			BNE 4\$	
000140	012716	000000G		MOV #EX.ONL,(SP)	
000144	012746	000001		MOV #1,-(SP)	
000150	010600			MOV SP,R0	; SP,*
000152	104415			TRAP 15	
000154	000423			BR 6\$	
000156	020227	000041	4\$:	CMP R2,#41	; COMMAND,*
000162	001007			BNE 5\$	
000164	012716	000000G		MOV #EX.RD,(SP)	
000170	012746	000001		MOV #1,-(SP)	
000174	010600			MOV SP,R0	; SP,*
000176	104415			TRAP 15	
000200	000411			BR 6\$	
000202	020227	000042	5\$:	CMP R2,#42	; COMMAND,*
000206	001010			BNE 7\$	
000210	012716	000000G		MOV #EX.WRT,(SP)	
000214	012746	000001		MOV #1,-(SP)	
000220	010600			MOV SP,R0	; SP,*
000222	104415			TRAP 15	
000224	005726		6\$:	TST (SP)+	
000226	000412			BR 8\$	
000230	005016		7\$:	CLR (SP)	3352
					3364

CZRC02
V02.0

CZRCDB0 RC25 DISK EXERCISER
ERROR MESSAGE SUBROUTINES

14-Jun-1985 09:36:36
14-Jun-1985 09:32:03

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC02.SRC;8

SEQ 0183
Page 90
(35)

```

000010 100011          BPL      1$
000012 016046 000040  MOV      40(R0),-(SP)      ;
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$
000034 016046 000050  1$:    MOV      50(R0),-(SP)      ;
000040 012746 000000G  MOV      #EX.LBN,-(SP)
000044 012746 000002  MOV      #2,-(SP)
000050 010600          MOV      SP,R0      ; SP,*
000052 104415          TRAP     15
000054 062706 000006  2$:    ADD      #6,SP      ;
000060 000207          RTS      PC          ;

```

; Routine Size: 25 words, Routine Base: \$CODE\$ + 7706
; Maximum stack depth per invocation: 5 words

```

; 3399 1  ROUTINE EMS_BC : NOVALUE =
; 3400 1
; 3401 1  !+
; 3402 1  ! THIS ROUTINE PRINTS (EXTENDED) BOTH BYTE COUNT FIELDS OF THE CURRENT
; 3403 1  ! RETURN PACKET: THE BYTE COUNT FROM THE COMMAND ENVELOPE AND THE
; 3404 1  ! ACTUAL NUMBER OF BYTES TRANSFERRED (FROM THE RESPONSE ENVELOPE).
; 3405 1  !
; 3406 1  ! IMPLICIT INPUTS:
; 3407 1  ! RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
; 3408 1  !-
; 3409 1
; 3410 2  BEGIN
; 3411 2
; 3412 2  PRINTX (EX_CBC, .RP_ADDR [CBCNT_LO]); ! " BYTE COUNT IN COMMAND: XXXXX."
; 3413 2  PRINTX (EX_BC, .RP_ADDR [BCNT_LO]); ! " ACTUAL # OF BYTES TRANSFERRED: XXXXX."
; 3414 2
; 3415 1  END; ! ROUTINE EMS_BC

```

```

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

```

; Routine Size: 23 words, Routine Base: \$CODE\$ + 7770
; Maximum stack depth per invocation: 7 words


```

: 3416 1  ROUTINE EMS_BD : NOVALUE =
: 3417 1
: 3418 1  !+
: 3419 1  ! THIS ROUTINE PRINTS (EXTENDED) THE TWO-WORD I/O BUFFER DESCRIPTOR
: 3420 1  ! APPEARING IN THE CURRENT RETURN PACKET.
: 3421 1  !
: 3422 1  ! IMPLICIT INPUTS:
: 3423 1  ! RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
: 3424 1  !-
: 3425 1
: 3426 1  PRINTX (EX_BD, .RP_ADDR [BUFF_0], .RP_ADDR [BUFF_1]); ! " I/O BUFFER DESCRIPTOR: XXXXXX(0) XXXXXX(0)"

```

```

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

```

; Routine Size: 15 words, Routine Base: \$CODE\$ + 10046
; Maximum stack depth per invocation: 6 words

```

: 3427 1  ROUTINE EMS_RP : NOVALUE =
: 3428 1
: 3429 1  !+
: 3430 1  ! THIS ROUTINE IS RESPONSIBLE FOR PRINTING (EXTENDED) THE RELEVANT FIELDS
: 3431 1  ! OF THE CURRENT RETURN PACKET.
: 3432 1  !-
: 3433 1
: 3434 2  BEGIN
: 3435 2
: 3436 2  EMS_CRN (); ! COMMAND REFERENCE NUMBER
: 3437 2  EMS_SBC (); ! SUB-CODE
: 3438 2  EMS_CMD (); ! COMMAND (AND MODIFIER)
: 3439 2  EMS_LBN (); ! LBN OR BAD BLOCK NUMBER
: 3440 2  EMS_BC (); ! BYTE COUNTS
: 3441 2  EMS_BD (); ! I/O BUFFER DESCRIPTOR
: 3442 2
: 3443 1  END; ! ROUTINE EMS_RP

```

```

000000 004737 007216'      .SBTTL EMS.RP ERROR MESSAGE SUBROUTINES
000004 004737 007302'      EMS.RP: JSR PC,EMS_CRN ; 3436
000010 004737 007342'      JSR PC,EMS_SBC ; 3437
000014 004737 007706'      JSR PC,EMS_CMD ; 3438
000020 004737 007770'      JSR PC,EMS_LBN ; 3439
000024 004737 010046'      JSR PC,EMS_BC ; 3440
000030 000207      RTS PC ; 3441
                                ; 3427

```

; Routine Size: 13 words, Routine Base: \$CODE\$ + 10104
; Maximum stack depth per invocation: 1 word

; 3444 1 BGNMSG (EMS_01);

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

; Routine Size: 4 words, Routine Base: \$CODE\$ + 10136
; Maximum stack depth per invocation: 2 words

; 3445	2					
; 3446	2	EMS_ET ();		! ELAPSED TIME		
; 3447	2	PRINTB (EBS_01, MAX_UNITS);		! "MORE THAN XX. UNITS SPECIFIED"		
; 3448	2					
; 3449	1	ENDMSG;				

000000	004737	007132'	.SBTTL	M#EMS.01 ERROR MESSAGE SUBROUTINES		
			M#EMS.01:			
000004	012746	000020	JSR	PC,EMS.ET	;	3446
000010	012746	000000G	MOV	#20,-(SP)	;	3447
000014	012746	000002	MOV	#EBS.01,-(SP)		
000020	010600		MOV	#2,-(SP)		
000022	104414		MOV	SP,R0	; SP,*	
000024	062706	000006	TRAP	14		
000030	000207		ADD	#6,SP	;	3444
			RTS	PC		

; Routine Size: 13 words, Routine Base: \$CODE\$ + 10146
; Maximum stack depth per invocation: 5 words

; 3450 1 BGNMSG (EMS_10);

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

; Routine Size: 4 words, Routine Base: \$CODE\$ + 10200
; Maximum stack depth per invocation: 2 words

; 3451	2					
; 3452	2	EMS_ET ();		! ELAPSED TIME		
; 3453	2	PRINTB (EBD_10, .RC25_ADDR + .OF_RC);		! "NO RESPONSE AT ADDRESS XXXXXX(0)"		
; 3454	2					
; 3455	1	ENDMSG;				

000000	004737	007132'	.SBTTL	M#EMS.10 ERROR MESSAGE SUBROUTINES		
			M#EMS.10:			
000004	013746	000000G	JSR	PC,EMS.ET	;	3452
			MOV	RC25.ADDR,-(SP)	;	3453

CZRCO2
V02.0

CZRCDBO RC25 DISK EXERCISER
ERROR MESSAGE SUBROUTINES

14-Jun-1985 09:36:36
14-Jun-1985 09:32:03

VAX-11 Bliss-16 V4.0-579
USER#1:(AZTEC.CZRCDB)CZRCO2.SRC;8

SEQ 0186
Page 94
(36)

```

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                ;
000034 000207      RTS    PC                    ;

```

```

; Routine Size: 15 words,      Routine Base: $CODE$ + 10210
; Maximum stack depth per invocation: 5 words

```

```

; 3456 1 BGNMSG (EMS_12);

```

```

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

```

```

; Routine Size: 4 words,      Routine Base: $CODE$ + 10246
; Maximum stack depth per invocation: 2 words

```

```

; 3457 2
; 3458 2 EMS_ET ();
; 3459 2 PRINTB (EBD_12, .RC25_ADDR);
; 3460 2
; 3461 1 ENDMSG;
! ELAPSED TIME
! "INCORRECT BR LEVEL GIVEN FOR DEVICE XXXXXX(O)"

```

```

000000 004737 007132'      .SBTTL M$EMS.12 ERROR MESSAGE SUBROUTINES
M$EMS.12:
000004 013746 000000G      JSR    PC,EMS.ET            ;
000010 012746 000000G      MOV    RC25.ADDR,-(SP)     ;
000014 012746 000002      MOV    #EBD.12,-(SP)
000020 010600      MOV    #2,-(SP)
000022 104414      MOV    SP,R0                ; SP,*
000024 062706 000006      TRAP   14
000030 000207      ADD    #6,SP                ;
RTS    PC                    ;

```

```

; Routine Size: 13 words,      Routine Base: $CODE$ + 10256
; Maximum stack depth per invocation: 5 words

```

```

; 3462 1 BGNMSG (EMS_13);

```

```

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

```

```

; Routine Size: 4 words,      Routine Base: $CODE$ + 10310
; Maximum stack depth per invocation: 2 words

```



```

; 3463 2
; 3464 2 EMS_ET ();
; 3465 2 PRINTB (EBD_13, .STEP, .RC25_ADDR);
; 3466 2 EMS_SA ();
; 3467 2
; 3468 1 ENDMSG;
! ELAPSED TIME
! "STEP X READ ERROR ON DEVICE XXXXXX(0)"
! PRINTX SA CONTENTS

```

```

000000 004737 007132' .SBTTL M$EMS.13 ERROR MESSAGE SUBROUTINES
M$EMS.13:
000004 013746 000000G JSR PC,EMS.ET ; 3464
000010 013746 000000G MOV RC25.ADDR,-(SP) ; 3465
000014 012746 000000G MOV STEP,-(SP)
000020 012746 000003 MOV #EBD.13,-(SP)
000024 010600 MOV SP,R0 ; SP,*
000026 104414 TRAP 14
000030 004737 007170' JSR PC,EMS.SA ; 3466
000034 062706 000010 ADD #10,SP ; 3462
000040 000207 RTS PC

```

```

; Routine Size: 17 words, Routine Base: $CODE$ + 10320
; Maximum stack depth per invocation: 6 words

```

```

; 3469 1 BGNMSG (EMS_i4);

```

```

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

```

```

; Routine Size: 4 words, Routine Base: $CODE$ + 10362
; Maximum stack depth per invocation: 2 words

```

```

; 3470 2
; 3471 2 EMS_ET ();
; 3472 2 PRINTB (EBD_14, .IRC25_ADDR);
; 3473 2 EMS_SA ();
; 3474 2
; 3475 1 ENDMSG;
! ELAPSED TIME
! "ERROR CODE RECEIVED IN SA REGISTER OF DEVICE XXXXXX(0)"
! PRINTX SA REGISTER CONTENTS

```

```

000000 004737 007132' .SBTTL M$EMS.14 ERROR MESSAGE SUBROUTINES
M$EMS.14:
000004 013746 000000G JSR PC,EMS.ET ; 3471
000010 012746 000000G MOV IRC25.ADDR,-(SP) ; 3472
000014 012746 000002 MOV #EBD.14,-(SP)
000020 010600 MOV SP,R0 ; SP,*
000022 104414 TRAP 14
000024 004737 007170' JSR PC,EMS.SA ; 3473
000030 062706 000006 ADD #6,SP ; 3469
000034 000207 RTS PC

```

; Routine Size: 15 words, Routine Base: \$CODE\$ + 10372
; Maximum stack depth per invocation: 5 words

; 3476 1 BGNMSG (EMS_15);

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

; Routine Size: 4 words, Routine Base: \$CODE\$ + 10430
; Maximum stack depth per invocation: 2 words

; 3477	2				
; 3478	2	EMS_ET ();		! ELAPSED TIME	
; 3479	2	PRINTB (EBD_15, .RC25_ADDR);		! "FAILED TO RECEIVE END MESSAGE FROM DEVICE XXXXXX(O)"	
; 3480	2	EMS_CMD ();		! PRINTX COMMAND	
; 3481	2				
; 3482	1	ENDMSG;			

000000	004737	007132'	.SBTTL	M\$EMS.15 ERROR MESSAGE SUBROUTINES	
000004	013746	000000G	M\$EMS.15:		
000010	012746	000000G	JSR	PC,EMS.ET	3478
000014	012746	000002	MOV	RC25.ADDR,-(SP)	3479
000020	010600		MOV	#EBD.15,-(SP)	
000022	104414		MOV	#2,-(SP)	
000024	004737	007342'	MOV	SP,R0	; SP,*
000030	062706	000006	TRAP	14	
000034	000207		JSR	PC,EMS.CMD	3480
			ADD	#6,SP	3476
			RTS	PC	

; Routine Size: 15 words, Routine Base: \$CODE\$ + 10440
; Maximum stack depth per invocation: 5 words

; 3483 1 BGNMSG (EMS_16);

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

; Routine Size: 4 words, Routine Base: \$CODE\$ + 10476
; Maximum stack depth per invocation: 2 words

; 3484	2				
; 3485	2	EMS_ET ();		! ELAPSED TIME	
; 3486	2	PRINTB (EBD_16, .CPLAT);		! "ERROR IN RESPONSE TO ONLINE COMMAND FOR PLATTER XXX."	
; 3487	2	EMS_STC ();		! PRINTX STATUS CODE IF NOT ZERO	

CZRC02
V02.0

CZRCDB0 RC25 DISK EXERCISER
ERROR MESSAGE SUBROUTINES

14-Jun-1985 09:36:36
14-Jun-1985 09:32:03

VAX-11 Bliss-16 V4.0-579
USER\$1:[AZTEC.CZRCDB]CZRC02.SRC;8

SEQ 0189
Page 97
(36)

; 3488 2 EMS_SBC ();
; 3489 2
; 3490 1 ENDMSG;

! PRINTX SUB-CODE

```
000000 004737 007132'      .SBTTL M$EMS.16 ERROR MESSAGE SUBROUTINES
                                M$EMS.16:
000004 013746 000000G      JSR    PC,EMS.ET          ; 3485
000010 012746 000000G      MOV    CPLAT,-(SP)       ; 3486
000014 012746 000002      MOV    #EBD.16,-(SP)
000020 010600      MOV    #2,-(SP)
000022 104414      MOV    SP,R0           ; SP,*
                                TRAP   14
000024 004737 007250'      JSR    PC,EMS.STC       ; 3487
000030 004737 007302'      JSR    PC,EMS.SBC       ; 3488
000034 062706 000006      ADD    #6,SP           ; 3483
000040 000207      RTS    PC
```

; Routine Size: 17 words, Routine Base: \$CODE\$ + 10506
; Maximum stack depth per invocation: 5 words

; 3491 1 BGNMSG (EMS_17);

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

; Routine Size: 4 words, Routine Base: \$CODE\$ + 10550
; Maximum stack depth per invocation: 2 words

; 3492 2
; 3493 2 EMS_ET ();
; 3494 2 PRINTB (EBD_17, .CPLAT);
; 3495 2
; 3496 1 ENDMSG;

! ELAPSED TIME
! "PLATTER XXX. IS SW WRITE-ENABLED BUT HW WRITE-PROTECTED"

```
000000 004737 007132'      .SBTTL M$EMS.17 ERROR MESSAGE SUBROUTINES
                                M$EMS.17:
000004 013746 000000G      JSR    PC,EMS.ET          ; 3493
000010 012746 000000G      MOV    CPLAT,-(SP)       ; 3494
000014 012746 000002      MOV    #EBD.17,-(SP)
000020 010600      MOV    #2,-(SP)
000022 104414      MOV    SP,R0           ; SP,*
                                TRAP   14
000024 062706 000006      ADD    #6,SP           ; 3491
000030 000207      RTS    PC
```

; Routine Size: 13 words, Routine Base: \$CODE\$ + 10560
; Maximum stack depth per invocation: 5 words

; 3497 1 BGNMSG (EMS_18);

CZRC02
V02.0

CZRCDB0 RC25 DISK EXERCISER
ERROR MESSAGE SUBROUTINES

14-Jun-1985 09:36:36
14-Jun-1985 09:32:03

VAX-11 Bliss-16 V4.0-579
USER\$1:[AZTEC.CZRCDB]CZRC02.SRC;8

SEQ 0191
Page 99
(36)

```

000000 004737 007132'          .SBTTL M$EMS.19 ERROR MESSAGE SUBROUTINES
                                M$EMS.19:
000004 013746 000000G          JSR    PC,EMS.ET          ;          3507
000010 012746 000000G          MOV    CPLAT,-(SP)        ;          3508
000014 012746 000002          MOV    #EBD.19,-(SP)
000020 010600          MOV    #2,-(SP)
000022 104414          MOV    SP,R0             ; SP,*
000024 004737 010104'          TRAP   14
000030 062706 000006          JSR    PC,EMS.RP        ;          3509
000034 000207          ADD    #6,SP            ;          3505
                                RTS    PC

```

```

; Routine Size: 15 words,      Routine Base: $CODE$ + 10674
; Maximum stack depth per invocation: 5 words

```

; 3512 1 BGNMSG (EMS_20);

```

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

```

```

; Routine Size: 4 words,      Routine Base: $CODE$ + 10732
; Maximum stack depth per invocation: 2 words

```

```

; 3513 2
; 3514 2 EMS_ET ();
; 3515 2 PRINTB (EBD_20, .RC25_ADDR);      ! ELAPSED TIME
; 3516 2                                     ! "DEVICE XXXXXX(O) NOT PROCESSING COMMAND PACKETS"
; 3517 1 ENDMSG;

```

```

000000 004737 007132'          .SBTTL M$EMS.20 ERROR MESSAGE SUBROUTINES
                                M$EMS.20:
000004 013746 000000G          JSR    PC,EMS.ET          ;          3514
000010 012746 000000G          MOV    RC25.ADDR,-(SP)   ;          3515
000014 012746 000002          MOV    #EBD.20,-(SP)
000020 010600          MOV    #2,-(SP)
000022 104414          MOV    SP,R0             ; SP,*
000024 062706 000006          TRAP   14
000030 000207          ADD    #6,SP            ;          3512
                                RTS    PC

```

```

; Routine Size: 13 words,      Routine Base: $CODE$ + 10742
; Maximum stack depth per invocation: 5 words

```

; 3518 1 BGNMSG (EMS_21);

```

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

```

; Routine Size: 4 words, Routine Base: \$CODE\$ + 10774
; Maximum stack depth per invocation: 2 words

```

; 3519 2
; 3520 2 EMS_ET ();
; 3521 2 PRINTB (EBD_21, .RC25_ADDR);
; 3522 2 EMS_CMD ();
; 3523 2 PRINTX (EX_DSC, .ST_CODE);
; 3524 2
; 3525 1 ENDMSG;

```

! ELAPSED TIME
! "MESSAGE REJECTED BY DUP SERVER ON DEVICE XXXXXX(0)"
! PRINTX COMMAND
! " DUP STATUS CODE: X."

```

000000 004737 007132' .SBTTL M$EMS.21 ERROR MESSAGE SUBROUTINES
M$EMS.21:
000004 013746 000000G JSR PC,EMS.ET ; 3520
000010 012746 000000G MOV RC25.ADDR,-(SP) ; 3521
000014 012746 000002 MOV #EBD.21,-(SP)
000020 010600 MOV #2,-(SP)
000022 104414 MOV SP,R0 ; SP,*
000024 004737 007342' TRAP 14
000030 013716 000000G JSR PC,EMS.CMD ; 3522
000034 012746 000000G MOV ST.CODE,(SP) ; 3523
000040 012746 000002 MOV #EX.DSC,-(SP)
000044 010600 MOV #2,-(SP)
000046 104415 MOV SP,R0 ; SP,*
000050 062706 000012 TRAP 15
000054 000207 ADD #12,SP ; 3518
RTS PC

```

; Routine Size: 23 words, Routine Base: \$CODE\$ + 11004
; Maximum stack depth per invocation: 7 words

; 3526 1 BGNMSG (EMS_22);

```

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

```

; Routine Size: 4 words, Routine Base: \$CODE\$ + 11062
; Maximum stack depth per invocation: 2 words

```

; 3527 2
; 3528 2 EMS_ET ();
; 3529 2 PRINTB (EBD_22, .RC25_ADDR);
; 3530 2
; 3531 1 ENDMSG;

```

! ELAPSED TIME
! "NO RESPONSE FROM FRONT PANEL TEST EXECUTING IN DEVICE XXXXXX(0)"

```

000000 004737 007132' .SBTTL M$EMS.22 ERROR MESSAGE SUBROUTINES
M$EMS.22:
JSR PC,EMS.ET ; 3528

```



```

000004 013746 000000G      MOV      RC25.ADDR, -(SP)
000010 012746 000000G      MOV      @EBD.22, -(SP)
000014 012746 000002      MOV      @2, -(SP)
000020 010600      MOV      SP, R0
000022 104414      TRAP     14
000024 062706 000006      ADD      @6, SP
000030 000207      RTS      PC

```

; Routine Size: 13 words, Routine Base: \$CODE\$ + 11072
; Maximum stack depth per invocation: 5 words

; 3532 1 BGNMSG (EMS_30);

```

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

```

; Routine Size: 4 words, Routine Base: \$CODE\$ + 11124
; Maximum stack depth per invocation: 2 words

```

; 3533 2
; 3534 2      OWN
; 3535 2      EBH_TB : VECTOR [11] INITIAL (EBH_31, EBH_32, 0, EBH_34, EBH_35, EBH_36,
; 3536 2      EBH_37, EBH_38, EBH_39, EBH_40, EBH_41);
; 3537 2      ! TABLE OF BASIC, HARD ERROR MESSAGE ADDRESSES, INDEXED BY STATUS CODE
; 3538 2
; 3539 2      EMS_ET ();
; 3540 2      PRINTB (PLATT, .CPLAT);
; 3541 3      IF ((.ST_CODE GEQU 1) AND (.ST_CODE LEQU 11))
; 3542 2      THEN
; 3543 3          PRINTB (.EBH_TB [.ST_CODE - 1])
; 3544 2      ELSE
; 3545 2          PRINTB (EX_SC, .ST_CODE);
; 3546 2      EMS_RP ();
; 3547 2
; 3548 1      ENDMSG;

```

```

011134 000000G      EBH_TB: .WORD  EBH.31
011136 000000G      .WORD  EBH.32
011140 000000      .WORD  0
011142 000000G      .WORD  EBH.34
011144 000000G      .WORD  EBH.35
011146 000000G      .WORD  EBH.36
011150 000000G      .WORD  EBH.37
011152 000000G      .WORD  EBH.38
011154 000000G      .WORD  EBH.39
011156 000000G      .WORD  EBH.40
011160 000000G      .WORD  EBH.41

```

```

000000 004737 007132'      .SBTTL  M$EMS.30 ERROR MESSAGE SUBROUTINES
                        M$EMS.30:

```

```

000004 013746 000000G      JSR    PC,EMS.ET      ;
000010 012746 000000G      MOV    CPLAT,-(SP)    ;
000014 012746 000000      MOV    @PLATT,-(SP)   ;
000020 010600              MOV    @2,-(SP)      ;
000022 104414              MOV    SP,R0         ; SP,*
000024 013700 000000G      TRAP   14
000030 001413              MOV    ST.CODE,R0    ;
000032 020027 000013      BEQ    1$            ;
000036 101010              CMP    R0,@13        ;
000040 006300              BHI    1$            ;
000042 016016 011132'      ASL    R0             ;
000046 012746 000001      MOV    EBH.TB-2(R0),(SP) ;
000052 010600              MOV    @1,-(SP)     ;
000054 104414              MOV    SP,R0         ; SP,*
000056 000410              TRAP   14
000060 010016              BR     2$            ;
000062 012746 000000G      1$:  MOV    R0,(SP)    ;
000066 012746 000002      MOV    @EX.SC,-(SP)  ;
000072 010600              MOV    @2,-(SP)     ;
000074 104414              MOV    SP,R0         ; SP,*
000076 005726              TRAP   14
000100 004737 010104'      2$:  TST    (SP)+
000104 062706 000010      JSR    PC,EMS.RP    ;
000110 000207              ADD    @10,SP       ;
                                RTS    PC                ;

```

; Routine Size: 37 words, Routine Base: \$CODE\$ + 11162
; Maximum stack depth per invocation: 7 words

; 3549 1 BGNMSG (EMS_42);

```

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

```

; Routine Size: 4 words, Routine Base: \$CODE\$ + 11274
; Maximum stack depth per invocation: 2 words

```

; 3550 2
; 3551 2  EMS_ET ();
; 3552 2  PRINTB (PLATT, .CPLAT);      ! ELAPSED TIME
; 3553 2  PRINTB (EBH_42);           ! "PLATTER XXX. - "
; 3554 2  EMS_LBN ();                ! "HOST-DETECTED WRITE-COMPARE ERROR"
; 3555 2  EMS_BC ();                ! PRINTX LBN
; 3556 2
; 3557 1  ENDMSG;                    ! PRINTX BYTE COUNTS

```

```

000000 004737 007132'      .SBTTL  M$EMS.42 ERROR MESSAGE SUBROUTINES
000004 013746 000000G      M$EMS.42: JSR    PC,EMS.ET      ;
000010 012746 000000G      MOV    CPLAT,-(SP)    ;
                                MOV    @PLATT,-(SP)

```


CZRC2
V02.0

CZRCDB0 RC25 DISK EXERCISER
ERROR MESSAGE SUBROUTINES

14-Jun-1985 09:36:36
14-Jun-1985 09:32:03

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC2.SRC;8

SEQ 0195
Page 103
(36)

000014	012746	000002	MOV	#2,-(SP)		
000020	010600		MOV	SP,R0	; SP,*	
000022	104414		TRAP	14		
000024	012716	000000G	MOV	#EBH.42,(SP)		
000030	012746	000001	MOV	#1,-(SP)		3553
000034	010600		MOV	SP,R0	; SP,*	
000036	104414		TRAP	14		
000040	004737	007706'	JSR	PC,EMS.LBN		3554
000044	004737	007770'	JSR	PC,EMS.BC		3555
000050	062706	000010	ADD	#10,SP		3549
000054	000207		RTS	PC		

; Routine Size: 23 words, Routine Base: \$CODE\$ + 11304
; Maximum stack depth per invocation: 6 words

; 3558 1 BGNMSG (EMS_43);

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

; Routine Size: 4 words, Routine Base: \$CODE\$ + 11362
; Maximum stack depth per invocation: 2 words

; 3559	2					
; 3560	2	EMS_ET ();		! ELAPSED TIME		
; 3561	2	PRINTB (PLATT, .CPLAT);		! "PLATTER XXX. - "		
; 3562	2	PRINTB (EBH_43);		! "FAILED TO RECEIVE END MESSAGE FOR I/O COMMAND"		
; 3563	2	EMS_CRN ();		! PRINTX COMMAND REFERENCE NUMBER		
; 3564	2	RP_ADDR [CMDMOD] = .RP_ADDR [STATUS];		! STATUS FIELD HOLDS MODIFIER (IF ANY)		
; 3565	2	EMS_CMD ();		! PRINTX COMMAND (AND MODIFIER)		
; 3566	2	PRINTX (EX_LBN, .RP_ADDR [BBLK_LO]);		! PRINTX LBN		
; 3567	2	PRINTX (EX_CBC, .RP_ADDR [BCNT_LO]);		! PRINTX BYTE COUNT		
; 3568	2	EMS_BD ();		! PRINTX I/O BUFFER DESCRIPTOR		
; 3569	2					
; 3570	1	ENDMSG;				

000000	010146		.SBTTL	M#EMS.43 ERROR MESSAGE SUBROUTINES		
			M#EMS.43:			
000002	004737	007132'	MOV	R1,-(SP)		3558
000006	013746	000000G	JSR	PC,EMS.ET		3560
000012	012746	000000G	MOV	CPLAT,-(SP)		3561
000016	012746	000002	MOV	#PLATT,-(SP)		
000022	010600		MOV	#2,-(SP)		
000024	104414		MOV	SP,R0	; SP,*	
000026	012716	000000G	TRAP	14		
000032	012746	000001	MOV	#EBH.43,(SP)		3562
000036	010600		MOV	#1,-(SP)		
000040	104414		MOV	SP,R0	; SP,*	
000042	004737	007216'	TRAP	14		
000046	013701	000000G	JSR	PC,EMS.CRN		3563
			MOV	RP.ADDR,R1		3564

CZRC02
V02.0

CZRCDB0 RC25 DISK EXERCISER
ERROR MESSAGE SUBROUTINES

14-Jun-1985 09:36:36
14-Jun-1985 09:32:03

VAX-11 Bliss-16 V4.0-579
USER\$1:[AZTEC.CZRCDB]CZRC02.SRC;8

SEQ 0196
Page 104
(36)

```

000052 010100          MOV      R1,R0          ; RP.ADDR,*
000054 016061 000016 000012 MOV      16(R0),12(R1)
000062 004737 007342'   JSR      PC,EMS.CMD      ;
000066 013700 000000G   MOV      RP.ADDR,R0      ;
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      ;
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 010046'   JSR      PC,EMS.BD      ;
000142 062706 000020   ADD      #20,SP          ;
000146 012601          MOV      (SP)+,R1
000150 000207          RTS      PC

```

```

; Routine Size: 53 words, Routine Base: $CODE$ + 11372
; Maximum stack depth per invocation: 11 words

```

```

; 3571 1
; 3572 1 END
; 3573 1
; 3574 0 ELUDOM

```

OTS external references

```

.GLOBL $SAVE5, $SAVE4, $SAVE3, $SAVE2
.GLOBL BL$MUL

```

PSECT SUMMARY

```

;
; Psect Name      Words  Attributes
; $CODE$          2482   RO , I , LCL, REL, CON
; $PLIT$          487    RO , D , LCL, REL, CON

```

Library Statistics

```

;
; File              Total  Symbols  Percent  Pages  Processing
;                   -----  Loaded   Percent  Mapped  Time
; USER$1:[AZTEC.CZRCDB]CZRC02.L16;6  276    204      73      16     00:00.2

```

COMMAND QUALIFIERS

```

; BLISS/PDP11 CZRC02.SRC/LIST/EN:NOEIS

```

```

; Size: 2342 code + 627 data words
; Run Time: 04:34.1
; Elapsed Time: 04:35.7
; Lines/CPU Min: 782
; Lexemes/CPU-Min: 8635
; Memory Used: 441 pages
; Compilation Complete

```



```
: 0001 0  MODULE CZRC03 (  
: 0002 0      *TITLE 'CZRCDB0 RC25 DISK EXERCISER'  
: 0003 0      IDENT = 'V02.0',  
: 0004 0      ADDRESSING_MODE (ABSOLUTE)  
: 0005 0      ) =  
: 0006 1  BEGIN  
: 0007 1  
: 0043 1  *SBTTL 'DECLARATIONS'  
: 0044 1  
: 0045 1  LIBRARY 'CZRCDL';           ! RC25 EXERCISER GLOBAL LIBRARY  
: 0046 1  REQUIRE 'BLSMAC.REQ';     ! DIAGNOSTIC SUPERVISOR LIBRARY  
: 1535 1  
: 1536 1  EQUALS;  
: 1537 1  
: 1538 1  
: 1539 1  FORWARD ROUTINE          ! ROUTINES APPEAR IN THIS ORDER  
: 1540 1      INIT_TEST : NOVALUE,   ! INDENTATION IMPLIES CALLED SUBROUTINE  
: 1541 1      DRIVER_INIT : NOVALUE, ! INITIALIZATION SUBTEST  
: 1542 1      CTLR_INIT : NOVALUE,  
: 1543 1      INI_CTLR_DAT : NOVALUE,  
: 1544 1      REG_EXIST,  
: 1545 1      VEC_BR_TEST,  
: 1546 1      INT_GEN,  
: 1547 1      HARD_INIT,  
: 1548 1      INI_RRING : NOVALUE,  
: 1549 1      SET_CTLR_CHAR : NOVALUE,  
: 1550 1      UNIT_INIT : NOVALUE,  
: 1551 1      ACCESS : NOVALUE,
```



```
: 1552 1      DM_EXER : NOVALUE,           ! DM EXERCISER SUBTEST
: 1553 1      DM_INIT : NOVALUE,
: 1554 1      DM_QUIT,
: 1555 1      DM_RETPKT : NOVALUE,
: 1556 1      DM_MSG : NOVALUE,
: 1557 1      DM_ACC,
: 1558 1      DM_TALLY : NOVALUE,
: 1559 1      UPD_DMBC : NOVALUE,
: 1560 1      DM_TIME : NOVALUE,
: 1561 1      MULTI_DRIVE : NOVALUE,       ! MULTI-DRIVE SUBTEST
: 1562 1      MD_INIT : NOVALUE,
: 1563 1      INIT_IO_BUFF : NOVALUE,
: 1564 1      QIO_OK,
: 1565 1      MD_QUIT,
: 1566 1      QIO_GEN : NOVALUE,
: 1567 1      GET_RANDOM : NOVALUE,
: 1568 1      QIO_UNIT : NOVALUE,
: 1569 1      QIO_FUNC : NOVALUE,
: 1570 1      QIO_LBN : NOVALUE,
: 1571 1      ADV_BST : NOVALUE,
: 1572 1      QIO_SIZE : NOVALUE,
: 1573 1      FILL_BUFF : NOVALUE,
: 1574 1      PROC_RETPKT : NOVALUE,
: 1575 1      IO_RETPKT : NOVALUE,
: 1576 1      FSET_UPAR : NOVALUE,
: 1577 1      MD_TALLY : NOVALUE,
: 1578 1      HOST_WRT_CHK : NOVALUE,
: 1579 1      CMP_DATA,
: 1580 1      SWEEP : NOVALUE,
: 1581 1      RPS_REM,
: 1582 1      DR_RETPKT : NOVALUE,
: 1583 1      DRV_TIMCHK : NOVALUE,
: 1584 1      RCINT0 : L$ISR NOVALUE,       ! RC25 INTERRUPT SERVICE ROUTINES
: 1585 1      RCINT1 : L$ISR NOVALUE,
: 1586 1      RCINT2 : L$ISR NOVALUE,
: 1587 1      RCINT3 : L$ISR NOVALUE,
```



```
: 1588 1      RCINT : NOVALUE,
: 1589 1      INT_PROC : NOVALUE,
: 1590 1      ISET_CPAR : NOVALUE,
: 1591 1      FATAL_ERROR : NOVALUE,
: 1592 1      POLL_CRING : NOVALUE,
: 1593 1      POLL_RRING : NOVALUE,
: 1594 1      ENV_TO_RP : NOVALUE,
: 1595 1      DATAGM : NOVALUE;
: 1596 1
: 1597 1      PSECT OWN = $GGG$ (READ, NOWRITE, EXECUTE, LOCAL, CONCATENATE);
: 1598 1
: 1599 1      OWN
: 1600 1      COMM_AREA : BLOCKVECTOR [MAX_CTLR, COMM_LEN, WORD] FIELD (COM_FIELDS),
: 1601 1      ! U/Q PORT COMMUNICATIONS AREA BETWEEN HOST AND RC25 CONTROLLERS
: 1602 1      BST : BLOCKVECTOR [MAX_UNITS, BST_LEN, WORD] FIELD (B_FIELDS),
: 1603 1      ! BLOCK SEQUENCE TABLE FOR SEQUENTIAL LBN (VS. RANDOM SEEK) MODE
: 1604 1      DPST : VECTOR [MAX_UNITS, BYTE], ! DATA PATTERN SEQUENCE TABLE
: 1605 1      ICOM_ADDR : REF BLOCK [COMM_LEN, WORD] FIELD (COM_FIELDS),
: 1606 1      ! ADDRESS OF INTERRUPTING CONTROLLER'S U/Q PORT COMMUNICATION AREA
: 1607 1      IENV_ADDR : REF BLOCK [ENV_LEN, WORD] FIELD (E_FIELDS),
: 1608 1      ! ADDRESS OF AN MSCP ENVELOPE (INTERRUPT PROCESSING)
: 1609 1      ICST_ADDR : REF BLOCK [CST_LEN, WORD] FIELD (C_FIELDS),
: 1610 1      ! ADDRESS OF INTERRUPTING CONTROLLER'S CST
: 1611 1      IDCT_ADDR : REF BLOCK [DCT_LEN, WORD] FIELD (DC_FIELDS),
: 1612 1      ! ADDRESS OF INTERRUPTING CONTROLLER'S DCT
: 1613 1      INT_ADDR : VECTOR [MAX_CTLR] INITIAL (RCINT0, RCINT1, RCINT2, RCINT3),
: 1614 1      ! INTERRUPT SERVICE ROUTINE ADDRESS TABLE
: 1615 1      ICTLR : WORD, ! INTERRUPTING CONTROLLER NUMBER
: 1616 1      MX1 : SIGNED WORD, ! MSCP ENV INDEX FOR FIRST QIO
: 1617 1      MX2 : SIGNED WORD, ! MSCP ENV INDEX FOR SECOND QIO
: 1618 1      MAD1 : REF BLOCK [ENV_LEN, WORD] FIELD (E_FIELDS),
: 1619 1      ! ADDRESS OF MSCP ENVELOPE FOR FIRST QIO
: 1620 1      MAD2 : REF BLOCK [ENV_LEN, WORD] FIELD (E_FIELDS),
: 1621 1      ! ADDRESS OF MSCP ENVELOPE FOR SECOND QIO
: 1622 1      VEC_AD : WORD, ! CURRENT DEVICE'S VECTOR ADDRESS
: 1623 1      BRLEVEL : WORD, ! CURRENT DEVICE'S BR LEVEL
: 1624 1      USIZE : WORD, ! UNIT SIZE (NO. OF LBN'S)
: 1625 1      DM_TIMR : VECTOR [MAX_CTLR], ! TIMERS FOR ONE PASS OF FRONT PANEL TEST (DM EXER)
: 1626 1      SWEEP_FLAG : BYTE, ! CALL / DON'T CALL SWEEP () (MULTI-DRIVE SUBTEST)
: 1627 1      RDM_CNT : WORD INITIAL (RDM_LEN), ! NUMBER OF RANDOM NUMBERS \ KEEP
: 1628 1      RANDOM : VECTOR [RDM_LEN, WORD], ! RANDOM NUMBER TABLE (PATTERN 1) / TOGETHER
: 1629 1      PAT02 : VECTOR [2] INITIAL (1, ! PATTERN 2
: 1630 1      %0'000000'),
: 1631 1      PAT03 : VECTOR [2] INITIAL (1, ! PATTERN 3
: 1632 1      %0'177777'),
: 1633 1      PAT04 : VECTOR [2] INITIAL (1, ! PATTERN 4
: 1634 1      %0'105613'),
: 1635 1      PAT05 : VECTOR [2] INITIAL (1, ! PATTERN 5
: 1636 1      %0'031463'),
: 1637 1      PAT06 : VECTOR [2] INITIAL (1, ! PATTERN 6
: 1638 1      %0'030221'),
: 1639 1      PAT07 : VECTOR [17] INITIAL (16, ! PATTERN 7
: 1640 1      %0'000001', %0'000003', %0'000007', %0'000017',
: 1641 1      %0'000037', %0'000077', %0'000177', %0'000377',
: 1642 1      %0'000777', %0'001777', %0'003777', %0'007777',
: 1643 1      %0'017777', %0'037777', %0'077777', %0'177777'),
: 1644 1      PAT08 : VECTOR [17] INITIAL (16, ! PATTERN 8
```



```
: 1645 1      %0'177776', %0'177774', %0'177770', %0'177760',
: 1646 1      %0'177740', %0'177700', %0'177600', %0'177400',
: 1647 1      %0'177000', %0'176000', %0'174000', %0'170000',
: 1648 1      %0'160000', %0'140000', %0'100000', %0'000000'),
: 1649 1      PAT09 : VECTOR [17] INITIAL (16,          ! PATTERN 9
: 1650 1      %0'000000', %0'000000', %0'000000', %0'177777',
: 1651 1      %0'177777', %0'177777', %0'000000', %0'000000',
: 1652 1      %0'177777', %0'177777',
: 1653 1      %0'000000', %0'177777', %0'000000', %0'177777',
: 1654 1      %0'000000', %0'177777'),
: 1655 1      PAT10 : VECTOR [2] INITIAL (1,          ! PATTERN 10
: 1656 1      %0'133331'),
: 1657 1      PAT11 : VECTOR [17] INITIAL (16,        ! PATTERN 11
: 1658 1      %0'052525', %0'052525', %0'052525', %0'125252',
: 1659 1      %0'125252', %0'125252', %0'052525', %0'052525',
: 1660 1      %0'125252', %0'125252',
: 1661 1      %0'052525', %0'125252', %0'052525', %0'125252',
: 1662 1      %0'052525', %0'125252'),
: 1663 1      PAT12 : VECTOR [21] INITIAL (20,        ! PATTERN 12
: 1664 1      %0'026455', %0'026455', %0'026455', %0'151322',
: 1665 1      %0'151322', %0'151322', %0'026455', %0'026455',
: 1666 1      %0'151322', %0'151322', %0'026455', %0'026455',
: 1667 1      %0'151322', %0'026455', %0'151322', %0'026455',
: 1668 1      %0'151322', %0'026455', %0'151322', %0'026455'),
: 1669 1      PAT13 : VECTOR [2] INITIAL (1,          ! PATTERN 13
: 1670 1      %0'066666'),
: 1671 1      PAT14 : VECTOR [17] INITIAL (16,        ! PATTERN 14
: 1672 1      %0'000001', %0'000002', %0'000004', %0'000010',
: 1673 1      %0'000020', %0'000040', %0'000100', %0'000200',
: 1674 1      %0'000400', %0'001000', %0'002000', %0'004000',
: 1675 1      %0'010000', %0'020000', %0'040000', %0'100000'),
: 1676 1      PAT15 : VECTOR [17] INITIAL (16,        ! PATTERN 15
: 1677 1      %0'177776', %0'177775', %0'177773', %0'177767',
: 1678 1      %0'177757', %0'177737', %0'177677', %0'177577',
: 1679 1      %0'177377', %0'176777', %0'175777', %0'173777',
: 1680 1      %0'167777', %0'157777', %0'137777', %0'077777'),
: 1681 1      PAT16 : VECTOR [17] INITIAL (16,        ! PATTERN 16
: 1682 1      %0'133331', %0'133331', %0'133331', %0'155554',
: 1683 1      %0'155554', %0'155554', %0'133331', %0'133331',
: 1684 1      %0'155554', %0'155554',
: 1685 1      %0'133331', %0'155554', %0'133331', %0'155554',
: 1686 1      %0'133331', %0'155554'),
: 1687 1      PAT17 : VECTOR [22] INITIAL (21,        ! PATTERN 17
: 1688 1      %0'000000', %0'106466', %0'106466', %0'071311',
: 1689 1      %0'071311', %0'071311', %0'106466', %0'106466',
: 1690 1      %0'106466', %0'106466', %0'071311', %0'071311',
: 1691 1      %0'071311', %0'071311', %0'071311', %0'106466',
: 1692 1      %0'106466', %0'106466', %0'106466', %0'106466',
: 1693 1      %0'106466'),
: 1694 1      PAT18 : VECTOR [22] INITIAL (21,        ! PATTERN 18
: 1695 1      %0'106466', %0'000000', %0'071311',
: 1696 1      %0'106466', %0'106466', %0'106466', %0'071311',
: 1697 1      %0'071311', %0'071311', %0'071311', %0'106466',
: 1698 1      %0'106466', %0'106466', %0'106466', %0'106466',
: 1699 1      %0'071311', %0'071311', %0'071311', %0'071311',
: 1700 1      %0'071311', %0'071311'),
: 1701 1      PAT19 : VECTOR [22] INITIAL (21,        ! PATTERN 19
```



```

: 1702 1      %0'000000', %0'134631', %0'134631', %0'043146',
: 1703 1      %0'043146', %0'043146', %0'134631', %0'134631',
: 1704 1      %0'134631', %0'134631', %0'043146', %0'043146',
: 1705 1      %0'043146', %0'043146', %0'043146', %0'134631',
: 1706 1      %0'134631', %0'134631', %0'134631', %0'134631',
: 1707 1      %0'134631'),
: 1708 1      PAT20 : VECTOR [22] INITIAL (21,                ! PATTERN 20
: 1709 1      %0'134631', %0'000000', %0'043146',
: 1710 1      %0'134631', %0'134631', %0'134631', %0'043146',
: 1711 1      %0'043146', %0'043146', %0'043146', %0'134631',
: 1712 1      %0'134631', %0'134631', %0'134631', %0'134631',
: 1713 1      %0'043146', %0'043146', %0'043146', %0'043146',
: 1714 1      %0'043146', %0'043146'),
: 1715 1      PAT21 : VECTOR [2] INITIAL (1,                ! PATTERN 21
: 1716 1      %0'000000'),                                ! (LBN)
: 1717 1      DPA_TBL : VECTOR [DP_CNT] INITIAL            ! DATA PATTERN ADDRESS TABLE
: 1718 1      (RDM_CNT, PAT02, PAT03, PAT04, PAT05,
: 1719 1      PAT06, PAT07, PAT08, PAT09, PAT10, PAT11,
: 1720 1      PAT12, PAT13, PAT14, PAT15, PAT16, PAT17,
: 1721 1      PAT18, PAT19, PAT20, PAT21),

```

```

!+
! THE FOLLOWING CODE IS THE CROM PRIMER WHICH WAS DEVELOPED
! INDEPENDENTLY. ITS BINARY .SAV FILE WAS RUN THROUGH THE PROGRAM
! "DMCONV", PRODUCING THIS COMPILABLE VECTOR.
!-

```

```

: 1728 1      CROMP:VECTOR[206,WORD]
: 1729 1      PRESET (
: 1730 1      [0]      = %0'000140', ! THIS IS THE DM PROGRAM BYTE COUNT.
: 1731 1      [1]      = %0'000000',
: 1732 1      [2]      = %0'000472', ! THIS IS THE DM OVERLAY BYTE COUNT.
: 1733 1      [3]      = %0'000000',
: 1734 1      [4]      = %0'051103', ! NEXT 3 WORDS = PROGRAM NAME (ASCII)
: 1735 1      [5]      = %0'046517', ! PROGRAM NAME IS 'CROMP '
: 1736 1      [6]      = %0'020120',
: 1737 1      [7]      = %0'000001', ! THIS IS THE PROGRAM VERSION
: 1738 1      [8]      = %0'000013', ! UPPER BYTE=TIME OUT VAL. LOWER = FLAGS
: 1739 1      [9]      = %0'000000',
: 1740 1      [10]     = %0'000000',
: 1741 1      [11]     = %0'000000',
: 1742 1      [12]     = %0'000000',
: 1743 1      [13]     = %0'000000',
: 1744 1      [14]     = %0'000000',
: 1745 1      [15]     = %0'000000',
: 1746 1      [16]     = %0'104206', ! DM CODE STARTS HERE
: 1747 1      [17]     = %0'007760',
: 1748 1      [18]     = %0'002754',
: 1749 1      [19]     = %0'000000',
: 1750 1      [20]     = %0'000000',
: 1751 1      [21]     = %0'000000',
: 1752 1      [22]     = %0'000000',
: 1753 1      [23]     = %0'000000',
: 1754 1      [24]     = %0'000000',
: 1755 1      [25]     = %0'000235',
: 1756 1      [26]     = %0'000000',
: 1757 1      [27]     = %0'000000',
: 1758 1      [28]     = %0'104204',

```



```
: 1759 1 [29] = %0'007774'.  
: 1760 1 [30] = %0'104140'.  
: 1761 1 [31] = %0'002752'.  
: 1762 1 [32] = %0'104204'.  
: 1763 1 [33] = %0'002751'.  
: 1764 1 [34] = %0'104203'.  
: 1765 1 [35] = %0'007000'.  
: 1766 1 [36] = %0'104647'.  
: 1767 1 [37] = %0'000001'.  
: 1768 1 [38] = %0'104641'.  
: 1769 1 [39] = %0'000002'.  
: 1770 1 [40] = %0'104142'.  
: 1771 1 [41] = %0'060020'.  
: 1772 1 [42] = %0'102207'.  
: 1773 1 [43] = %0'000037'.  
: 1774 1 [44] = %0'052754'.  
: 1775 1 [45] = %0'104077'.  
: 1776 1 [46] = %0'007006'.  
: 1777 1 [47] = %0'032305'.  
: 1778 1 [48] = %0'000000'.  
: 1779 1 [49] = %0'000000'.  
: 1780 1 [50] = %0'000000'.  
: 1781 1 [51] = %0'000000'.  
: 1782 1 [52] = %0'000000'.  
: 1783 1 [53] = %0'000000'.  
: 1784 1 [54] = %0'104207'.  
: 1785 1 [55] = %0'007000'.  
: 1786 1 [56] = %0'104201'.  
: 1787 1 [57] = %0'000002'.  
: 1788 1 [58] = %0'104202'.  
: 1789 1 [59] = %0'002740'.  
: 1790 1 [60] = %0'107027'.  
: 1791 1 [61] = %0'107017'.  
: 1792 1 [62] = %0'105012'.  
: 1793 1 [63] = %0'060011'.  
: 1794 1 [64] = %0'027107'.  
: 1795 1 [65] = %0'027203'.  
: 1796 1 [66] = %0'027146'.  
: 1797 1 [67] = %0'114000'.  
: 1798 1 [68] = %0'007005'.  
: 1799 1 [69] = %0'114000'.  
: 1800 1 [70] = %0'007776'.  
: 1801 1 [71] = %0'114000'.  
: 1802 1 [72] = %0'007776'.  
: 1803 1 [73] = %0'104077'.  
: 1804 1 [74] = %0'104206'.  
: 1805 1 [75] = %0'007760'.  
: 1806 1 [76] = %0'104301'.  
: 1807 1 [77] = %0'007005'.  
: 1808 1 [78] = %0'104610'.  
: 1809 1 [79] = %0'007234'.  
: 1810 1 [80] = %0'007767'.  
: 1811 1 [81] = %0'115000'.  
: 1812 1 [82] = %0'007767'.  
: 1813 1 [83] = %0'077076'.  
: 1814 1 [84] = %0'027120'.  
: 1815 1 [85] = %0'104200'.
```

! DM OVERLAY CODE STARTS HERE

:	1816	1	[86]	=	%0'007051'
:	1817	1	[87]	=	%0'002750'
:	1818	1	[88]	=	%0'002740'
:	1819	1	[89]	=	%0'027146'
:	1820	1	[90]	=	%0'115000'
:	1821	1	[91]	=	%0'007265'
:	1822	1	[92]	=	%0'057051'
:	1823	1	[93]	=	%0'104200'
:	1824	1	[94]	=	%0'177777'
:	1825	1	[95]	=	%0'007004'
:	1826	1	[96]	=	%0'027130'
:	1827	1	[97]	=	%0'027146'
:	1828	1	[98]	=	%0'115000'
:	1829	1	[99]	=	%0'007265'
:	1830	1	[100]	=	%0'017071'
:	1831	1	[101]	=	%0'114000'
:	1832	1	[102]	=	%0'007004'
:	1833	1	[103]	=	%0'027130'
:	1834	1	[104]	=	%0'007051'
:	1835	1	[105]	=	%0'027162'
:	1836	1	[106]	=	%0'027203'
:	1837	1	[107]	=	%0'114000'
:	1838	1	[108]	=	%0'007264'
:	1839	1	[109]	=	%0'027216'
:	1840	1	[110]	=	%0'115400'
:	1841	1	[111]	=	%0'007005'
:	1842	1	[112]	=	%0'106200'
:	1843	1	[113]	=	%0'000003'
:	1844	1	[114]	=	%0'007005'
:	1845	1	[115]	=	%0'037031'
:	1846	1	[116]	=	%0'114000'
:	1847	1	[117]	=	%0'007005'
:	1848	1	[118]	=	%0'007031'
:	1849	1	[119]	=	%0'104207'
:	1850	1	[120]	=	%0'007002'
:	1851	1	[121]	=	%0'104201'
:	1852	1	[122]	=	%0'000002'
:	1853	1	[123]	=	%0'060023'
:	1854	1	[124]	=	%0'102207'
:	1855	1	[125]	=	%0'000037'
:	1856	1	[126]	=	%0'057107'
:	1857	1	[127]	=	%0'000000'
:	1858	1	[128]	=	%0'104307'
:	1859	1	[129]	=	%0'007005'
:	1860	1	[130]	=	%0'104201'
:	1861	1	[131]	=	%0'000005'
:	1862	1	[132]	=	%0'060031'
:	1863	1	[133]	=	%0'104010'
:	1864	1	[134]	=	%0'007001'
:	1865	1	[135]	=	%0'000000'
:	1866	1	[136]	=	%0'104307'
:	1867	1	[137]	=	%0'007002'
:	1868	1	[138]	=	%0'105207'
:	1869	1	[139]	=	%0'000060'
:	1870	1	[140]	=	%0'114001'
:	1871	1	[141]	=	%0'104202'
:	1872	1	[142]	=	%0'000001'

:	1873	1	[143]	=	'0'104203'
:	1874	1	[144]	=	'0'007004'
:	1875	1	[145]	=	'0'060021'
:	1876	1	[146]	=	'0'102207'
:	1877	1	[147]	=	'0'000037'
:	1878	1	[148]	=	'0'057130'
:	1879	1	[149]	=	'0'000000'
:	1880	1	[150]	=	'0'104307'
:	1881	1	[151]	=	'0'007002'
:	1882	1	[152]	=	'0'114001'
:	1883	1	[153]	=	'0'104202'
:	1884	1	[154]	=	'0'000032'
:	1885	1	[155]	=	'0'104203'
:	1886	1	[156]	=	'0'007234'
:	1887	1	[157]	=	'0'060020'
:	1888	1	[158]	=	'0'102207'
:	1889	1	[159]	=	'0'000037'
:	1890	1	[160]	=	'0'057146'
:	1891	1	[161]	=	'0'000000'
:	1892	1	[162]	=	'0'104200'
:	1893	1	[163]	=	'0'000004'
:	1894	1	[164]	=	'0'007000'
:	1895	1	[165]	=	'0'104207'
:	1896	1	[166]	=	'0'002743'
:	1897	1	[167]	=	'0'104301'
:	1898	1	[168]	=	'0'007001'
:	1899	1	[169]	=	'0'104272'
:	1900	1	[170]	=	'0'105612'
:	1901	1	[171]	=	'0'007240'
:	1902	1	[172]	=	'0'100612'
:	1903	1	[173]	=	'0'007240'
:	1904	1	[174]	=	'0'115401'
:	1905	1	[175]	=	'0'117400'
:	1906	1	[176]	=	'0'007000'
:	1907	1	[177]	=	'0'037171'
:	1908	1	[178]	=	'0'000000'
:	1909	1	[179]	=	'0'114000'
:	1910	1	[180]	=	'0'002743'
:	1911	1	[181]	=	'0'114000'
:	1912	1	[182]	=	'0'002744'
:	1913	1	[183]	=	'0'114000'
:	1914	1	[184]	=	'0'002745'
:	1915	1	[185]	=	'0'114000'
:	1916	1	[186]	=	'0'002746'
:	1917	1	[187]	=	'0'114000'
:	1918	1	[188]	=	'0'002747'
:	1919	1	[189]	=	'0'000000'
:	1920	1	[190]	=	'0'104307'
:	1921	1	[191]	=	'0'007002'
:	1922	1	[192]	=	'0'105207'
:	1923	1	[193]	=	'0'000010'
:	1924	1	[194]	=	'0'114001'
:	1925	1	[195]	=	'0'104202'
:	1926	1	[196]	=	'0'000025'
:	1927	1	[197]	=	'0'104203'
:	1928	1	[198]	=	'0'007240'
:	1929	1	[199]	=	'0'060021'


```
; 1930 1 [200] = %0'102207',
; 1931 1 [201] = %0'000037',
; 1932 1 [202] = %0'057216',
; 1933 1 [203] = %0'000000',
; 1934 1 [204] = %0'143367',
; 1935 1 [205] = %0'000000';
; 1936 1
; 1937 1 EXTERNAL
; 1938 1 PATCH : VECTOR [100, WORD], ! PATCH AREA
; 1939 1 CPT : VECTOR [MAX_UNITS, BYTE],
; 1940 1 ! CURRENT PASS TESTING (YES / NO) PER UNIT
; 1941 1 CST : BLOCKVECTOR [MAX_CTLR, CST_LEN, WORD] FIELD (C_FIELDS),
; 1942 1 ! RUN-TIME CONTROLLER STATUS TABLES
; 1943 1 CST_ADDR : REF BLOCK [CST_LEN, WORD] FIELD (C_FIELDS),
; 1944 1 ! CONTROLLER STATUS TABLE ADDRESS OF "CURRENT" CONTROLLER
; 1945 1 DCT : BLOCKVECTOR [MAX_CTLR, DCT_LEN, WORD] FIELD (DC_FIELDS),
; 1946 1 ! DRIVER CONTROLLER TABLES
; 1947 1 DCT_ADDR : REF BLOCK [DCT_LEN, WORD] FIELD (DC_FIELDS),
; 1948 1 ! ADDRESS OF "CURRENT" DRIVER CONTROLLER TABLE
; 1949 1 RC25_ADDR : REF RC25 FIELD (RC_REG),
; 1950 1 ! DEVICE ADDRESS OF "CURRENT" CONTROLLER
; 1951 1 IRC25_ADDR : REF RC25 FIELD (RC_REG),
; 1952 1 ! DEVICE ADDRESS OF INTERRUPTING CONTROLLER
; 1953 1 DM_COMM : BLOCKVECTOR [MAX_CTLR, DMC_LEN, WORD] FIELD (DMC_FIELDS),
; 1954 1 ! DM EXERCISER COMMUNICATION AREA (LINK TO FRONT PANEL TEST)
; 1955 1 DMC_ADDR : REF BLOCK [DMC_LEN, WORD] FIELD (DMC_FIELDS),
; 1956 1 ! ADDRESS OF CURRENT CONTROLLER'S DM EXERCISER COMMUNICATION AREA
; 1957 1 RP_SAVE : VECTOR [MAX_CTLR * RPS_LEN, BYTE, SIGNED],
; 1958 1 ! RETURN PACKET SAVE AREA
; 1959 1 RPS_X1 : WORD, ! STARTING INDEX OF CURRENT CONTROLLER'S RP_SAVE AREA
; 1960 1 RPS_X2 : WORD, ! ENDING INDEX OF CURRENT CONTROLLER'S RP_SAVE AREA
; 1961 1 OUTC_LIST : VECTOR [MAX_CTLR * OUTC_CNT, BYTE, SIGNED],
; 1962 1 ! OUTSTANDING COMMAND LIST (CONTAINS MSCP ENVELOPE INDECS)
; 1963 1 OUTC_TIMR : VECTOR [MAX_CTLR * OUTC_CNT, WORD],
; 1964 1 ! OUTSTANDING COMMAND TIMERS
; 1965 1 OCL_X1 : WORD,
; 1966 1 ! STARTING INDEX OF CURRENT CONTROLLER'S OUTSTANDING COMMAND AREA
; 1967 1 OCL_X2 : WORD,
; 1968 1 ! ENDING INDEX OF CURRENT CONTROLLER'S OUTSTANDING COMMAND AREA
; 1969 1 TALLY : VECTOR [MAX_UNITS * TALLY_LEN, WORD] FIELD (T_FIELDS),
; 1970 1 ! STATISTICS TABLES
; 1971 1 T_ADDR : REF BLOCK [TALLY_LEN, WORD] FIELD (T_FIELDS),
; 1972 1 ! ADDRESS OF STATISTICS TABLE (TALLY) FOR CURRENT UNIT
; 1973 1 MSCP_ENV : BLOCKVECTOR [ENV_CNT, ENV_LEN, WORD] FIELD (E_FIELDS),
; 1974 1 ! MSCP ENVELOPE POOL
; 1975 1 ENV_USE : VECTOR [ENV_CNT, BYTE, SIGNED],
; 1976 1 ! MSCP ENVELOPE POOL ALLOCATION TABLE
; 1977 1 RETPKT : BLOCKVECTOR [RP_CNT, RP_LEN, WORD] FIELD (RP_FIELDS),
; 1978 1 ! RETURN PACKET POOL
; 1979 1 RP_USE : VECTOR [RP_CNT, BYTE, SIGNED],
; 1980 1 ! RETURN PACKET POOL ALLOCATION TABLE
; 1981 1 RP_INDX : WORD, ! CURRENT RETURN PACKET INDEX
; 1982 1 RP_ADDR : REF BLOCK [RP_LEN, WORD] FIELD (RP_FIELDS),
; 1983 1 ! CURRENT RETURN PACKET ADDRESS
; 1984 1 BUFF_DESC : BLOCKVECTOR [MAX_BUF_CNT, DESC_LEN, WORD] FIELD (BD_FIELDS),
; 1985 1 ! TABLE OF I/O BUFFER DESCRIPTORS
; 1986 1 BUFF_OWN : VECTOR [MAX_BUF_CNT, BYTE, SIGNED],
```



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



```

: 2044 1      STC_01,
: 2045 1      STC_02,
: 2046 1      STC_03,
: 2047 1      STC_04,
: 2048 1      STC_05,
: 2049 1      STC_06,
: 2050 1      STC_07,
: 2051 1      STC_08,
: 2052 1      STC_09,
: 2053 1      STC_10,
: 2054 1      STC_11,
: 2055 1      EX_CRN,
: 2056 1      EX_SB,
: 2057 1      EX_EL,
: 2058 1      EX_PA,
: 2059 1      EX_FMT,
: 2060 1      EX_EVC,
: 2061 1      EX_HMA,
: 2062 1      EX_03,
: 2063 1      SWP_STRACK : WORD,
: 2064 1      SWP_ETRACK : WORD,
: 2065 1      SWP_FLAGS : BYTE,
: 2066 1      SWP_DPAT : BYTE,
: 2067 1      SWP_UCNT : WORD,
: 2068 1      SWP_UDPAT : VECTOR [MAX_UDP_CNT, WORD],
: 2069 1      L$HMEM,
: 2070 1      L$UNIT,
: 2071 1      L$LUN;
: 2072 1
: 2073 1      EXTERNAL ROUTINE
: 2074 1      NEX_TRAP : L$ISR NOVALUE,
: 2075 1      COPY_BLK : NOVALUE,
: 2076 1      SET_CPAR : NOVALUE,
: 2077 1      SET_UPAR : NOVALUE,
: 2078 1      GET_ENV,
: 2079 1      PUT_ENV : NOVALUE,
: 2080 1      GET_RETPKT,
: 2081 1      PUT_RETPKT : NOVALUE,
: 2082 1      GET_IO_BUFF : NOVALUE,
: 2083 1      PUT_IO_BUFF : NOVALUE,
: 2084 1      PUTA_BUFF : NOVALUE,
: 2085 1      OUT_IODQ,
: 2086 1      IN_IODQ : NOVALUE,
: 2087 1      DROP_CTLR : NOVALUE,
: 2088 1      DRV_CTLERR : NOVALUE,
: 2089 1      HARD_ERR : NOVALUE,
: 2090 1      UPD_IOC : NOVALUE,
: 2091 1      OVF_CHK : NOVALUE,
: 2092 1      XFR_CHK : NOVALUE,
: 2093 1      SEND,
: 2094 1      WAIT : NOVALUE,
: 2095 1      EMS_10 : NOVALUE,
: 2096 1      EMS_12 : NOVALUE,
: 2097 1      EMS_13 : NOVALUE,
: 2098 1      EMS_14 : NOVALUE,
: 2099 1      EMS_15 : NOVALUE,
: 2100 1      EMS_16 : NOVALUE,

```

```

! STARTING TRACK
! ENDING TRACK
! FLAGS (SEE DOCUMENTATION)
! DATA PATTERN NUMBER
! USER DATA PATTERN COUNT
! USER DATA PATTERN

```

CZRC03
V02.0

CZRCDB0 RC25 DISK EXERCISER
DECLARATIONS

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC03.SRC;17

SEQ 0208
Page 12
(2)

:	2101	1	EMS_17	: NOVALUE.
:	2102	1	EMS_18	: NOVALUE.
:	2103	1	EMS_19	: NOVALUE.
:	2104	1	EMS_20	: NOVALUE.
:	2105	1	EMS_21	: NOVALUE.
:	2106	1	EMS_22	: NOVALUE.
:	2107	1	EMS_30	: NOVALUE.
:	2108	1	EMS_42	: NOVALUE.
:	2109	1	EMS_43	: NOVALUE.


```

: 2110 1 *SBTTL 'TEST SECTION'
: 2111 1
: 2112 1 !*
: 2113 1 ! THIS SECTION CONTAINS THE TOP-LEVEL TEST CODE FOR THE RC25 DISK
: 2114 1 ! EXERCISER. THE EXERCISER CONSISTS OF ONE TEST WHICH IS SUBDIVIDED INTO
: 2115 1 ! A NUMBER OF SUBTESTS. ALL SUBTESTS ARE DECLARED WITHIN THIS BLOCK.
: 2116 1 !-
: 2117 1
: 2118 3 BGNTST;
: 2119 3
: 2120 3 EOP_FLAG = FALSE;
: 2121 3 IF .ENTRY_REASON NEQU NEW_PASS ! IF START, RESTART, CONT, OR PWR FAIL
: 2122 3 THEN ! THEN
: 2123 4 BEGIN
: 2124 4
: 2125 6 BGNSUB;
: 2126 6 INIT_TEST (); ! INITIALIZATION SUBTEST
: 2127 4 ENDSUB;
: 2128 4
: 2129 3 END;
: 2130 3
: 2131 4 IF BIT_TST (SWP_FLAGS, SWF_DM) ! IF OPERATOR SELECTED DM EXERCISER
: 2132 3 THEN ! THEN
: 2133 4 BEGIN
: 2134 4
: 2135 6 BGNSUB;
: 2136 6 DM_EXER (); ! RUN DM EXERCISER SUBTEST
: 2137 4 ENDSUB;
: 2138 4
: 2139 4 END
: 2140 3 ELSE ! OTHERWISE
: 2141 4 BEGIN
: 2142 4
: 2143 6 BGNSUB;
: 2144 6 MULTI_DRIVE (); ! RUN MULTI-DRIVE SUBTEST
: 2145 4 ENDSUB;
: 2146 4
: 2147 3 END;
: 2148 3
: 2149 3 DORPT; ! PRINT STATISTICS
: 2150 3 EOP_FLAG = TRUE; ! SET END-OF-PASS FLAG
: 2151 3
: 2152 1 ENDTST;

```

```

.TITLE CZLCD3 CZLCD0 RC25 DISK EXERCISER
.IDENT /V02.0/
.ENABL AMA

```

000000
000000

000440
000540
000560

```

.PSECT $GGG$, RO
COMM.AREA:
.BLKW 220
BST: .BLKW 40
DPST: .BLKW 10
ICOM.ADDR:
.BLKW 1

```

CZRC03
V02.0

CZRCDB0 RC25 DISK EXERCISER
TEST SECTION

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC03.SRC;17

SEQ 0210
Page 14
(3)

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	0C0020	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

CZRCDS
V02.0

CZRCDBO RC25 DISK EXERCISER
TEST SECTION

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRCDS.SRC;17

SEQ 0211
Page 15
(3)

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

CZRC03
V02.0

CZRCDB0 RC25 DISK EXERCISER
TEST SECTION

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC03.SRC;17

SEQ 0213
Page 17
(3)

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

CZRC03
V02.0

CZRCDB0 RC25 DISK EXERCISER
TEST SECTION

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER\$1:[AZTEC.CZRCDB]CZRC03.SRC;17

SEQ 0214
Page 18
(3)

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

CZRC03
V02.0

CZRCDB0 RC25 DISK EXERCISER
TEST SECTION

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER\$1:[AZTEC.CZRCDB]CZRC03.SRC;17

SEQ 0215
Page 19
(3)

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

CZRC03
V02.0

CZRCDB0 RC25 DISK EXERCISER
TEST SECTION

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC03.SRC;17

SEQ 0216
Page 20
(3)

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	0C7760	.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

CZRC03
V02.0

CZRCDB0 RC25 DISK EXERCISER
TEST SECTION

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC03.SRC;17

SEQ 0217
Page 21
(3)

002250	027216	.WORD	27215
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

000000			.SBTTL	\$T1 TEST SECTION		
			.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	
000016	004737	000000V		JSR	PC,INIT.TEST	2123
000022	104467			TRAP	67	2126
000024	006000			ROR	RO	
000026	103772			BLO	1\$	
000030	132737	000002 000000G	2\$:	BITB	#2,SMP.FLAGS	2131
000036	001407			BEQ	4\$	
000040	104402		3\$:	TRAP	2	
000042	004737	000000V		JSR	PC,DM.EXER	2133
000046	104467			TRAP	67	2136
000050	006000			ROR	RO	
000052	103007			BCC	5\$	
000054	000771			BR	3\$	
000056	104402		4\$:	TRAP	2	
000060	004737	000000V		JSR	PC,MULTI.DRIVE	2141
000064	104467			TRAP	67	2144
000066	006000			ROR	RO	
000070	103772			BLO	4\$	
000072	104424		5\$:	TRAP	24	
000074	112737	000001 000000G		MOVB	#1,EOP.FLAG	2147
000102	000207			RTS	PC	2150
						2109

; Routine Size: 34 words, Routine Base: \$CODE\$ + 0000
; Maximum stack depth per invocation: 2 words

.SBTTL T1 TEST SECTION

CZRCDS
V02.0

CZRCDB0 RC25 DISK EXERCISER
TEST SECTION

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRCDS.SRC;17

SEQ 0221
Page 25
(3)

000000	004737	000000'	T1::			
000000			1\$:	JSR	PC,\$T1	
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

2150

```
: 2153 1 *SBTTL 'INITIALIZATION SUBTEST ROUTINES'
: 2154 1
: 2155 1 ROUTINE INIT_TEST : NOVALUE =
: 2156 1
: 2157 1 !*
: 2158 1 !
: 2159 1 ! THE INITIALIZATION SUBTEST IS DESIGNED TO VERIFY THE EXISTENCE OF THE
: 2160 1 ! DEVICES AS CONFIGURED BY THE OPERATOR DURING THE HW DIALOG, AND TO
: 2161 1 ! BRING EACH DEVICE ONLINE IN PREPARATION FOR EITHER THE MULTI-DRIVE
: 2162 1 ! SUBTEST OR THE DM EXERCISER SUBTEST.
: 2163 1 !
: 2164 1 ! BASICALLY, THE DEVICES ARE BROUGHT ONLINE VIA "DRIVER_INIT", WHICH IS
: 2165 1 ! INVOKED IMMEDIATELY. ANY DEVICES WHICH FAIL DURING THIS PHASE WILL BE
: 2166 1 ! MARKED OFFLINE IN THEIR DCT AND CST. FOR THOSE DEVICES WHICH SURVIVE
: 2167 1 ! THE INITIALIZATION, THIS ROUTINE WILL ATTEMPT 1 OR 2 ACCESS COMMANDS TO
: 2168 1 ! EACH PLATTER VIA ROUTINE "ACCESS". THE INITIALIZATION SUBTEST IS DEEMED
: 2169 1 ! A SUCCESS IF A BLOCK ON THE INNER TRACK OF EACH PLATTER CAN BE
: 2170 1 ! ACCESSED.
: 2171 1 !-
: 2172 2 BEGIN
: 2173 2
: 2174 3 IF MANUAL ! IF ATTENDED
: 2175 2 THEN ! THEN
: 2176 2 PRINTF (MSG_04); ! "INIT SUBTEST START"
: 2177 2 IIP_FLAG = TRUE; ! SET INIT-IN-PROGRESS FLAG
: 2178 2 DRIVER_INIT (); ! INIT DRIVER DATA AND DEVICES
: 2179 2 INCR CTLR FROM 0 TO (MAX_CTLR - 1) DO ! FOR EACH CONTROLLER
: 2180 3 BEGIN
: 2181 3
: 2182 3 SET_CPAR (.CTLR); ! SET UP COMMONLY-USED CONTROLLER-RELATED DATA ITEMS
: 2183 3 IF .CST_ADDR [STATE] EQLU ONLINE ! IF CONTROLLER IS STILL ALIVE
: 2184 3 THEN ! THEN
: 2185 4 BEGIN
: 2186 4
: 2187 4 INCR OFFSET FROM (0 + OF_UN) TO (3 + OF_UN) DO ! FOR EACH PLATTER
: 2188 5 BEGIN
: 2189 5
: 2190 6 IF ((.CST_ADDR [.OFFSET, P_PRES] EQLU PRESENT) AND
: 2191 6 (.CST_ADDR [.OFFSET, P_STAT] EQLU ONLINE))
: 2192 5 THEN
: 2193 6 BEGIN
: 2194 6
: 2195 6 SET_UPAR (.OFFSET); ! SET UP UNIT-RELATED DATA ITEMS
: 2196 6 ACCESS (); ! TRY ACCESS TO INNER TRACK
: 2197 6
: 2198 5 END; ! IF UNIT IS PRESENT AND ONLINE
: 2199 5
: 2200 4 END; ! UNIT LOOP
: 2201 4
: 2202 3 END; ! IF CONTROLLER IS ONLINE
: 2203 3
: 2204 2 END; ! CONTROLLER LOOP
: 2205 2
: 2206 2 IIP_FLAG = FALSE; ! CLEAR INIT-IN-PROGRESS FLAG
: 2207 2
: 2208 1 END; ! ROUTINE INIT_TEST
```



```

.SBTTL INIT.TEST INITIALIZATION SUBTEST ROUTINES
000000 004137 000000G INIT.TEST:
000004 104450 JSR R1,$SAVE2 ; 2155
000006 103007 TRAP 50 ; 2174
000010 012746 000000G BHIS 1$ ;
000014 012746 000001 MOV #MSG.04,-(SP) ; 2176
000020 010600 MOV #1,-(SP) ;
000022 104417 MOV SP,R0 ; SP,*
000024 022626 TRAP 17 ;
000026 112737 000001 000000G 1$: MOVB (SP)+,(SP)+ ;
000034 004737 000000V JSR #1,IIP.FLAG ; 2177
000040 005002 CLR PC,DRIVER.INIT ; 2178
000042 010246 2$: CLR R2 ; CTLR 2179
000044 004737 000000G JSR R2,-(SP) ; CTLR,* 2182
000050 013700 000000G MOV PC,SET.CPAR ;
000054 005760 000002 TST CST.ADDR,R0 ; 2183
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 ;
000124 005201 4$: INC R1 ; OFFSET 2196
000126 020127 000006 CMP R1,#6 ; OFFSET,* 2187
000132 003755 BLE 3$ ;
000134 005726 5$: TST (SP)+ ;
000136 005202 INC R2 ; CTLR 2180
000140 020227 000003 CMP R2,#3 ; CTLR,* 2179
000144 003736 BLE 2$ ;
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 1 ROUTINE DRIVER_INIT : NOVALUE =
: 2210 1
: 2211 1 !+
: 2212 1 ! THIS ROUTINE IS EQUIVALENT IN FUNCTION TO THE INITIALIZATION ENTRY
: 2213 1 ! POINT OF A STANDARD DEVICE DRIVER. ITS RESPONSIBILITY IS TO INITIALIZE
: 2214 1 ! DRIVER DATA, AND TO BRING EACH RC25 CONTROLLER AND UNIT ONLINE.
: 2215 1 !-
: 2216 1
: 2217 2 BEGIN
: 2218 2
: 2219 2 LOCAL
: 2220 2 ENV_ADDR;
: 2221 2
: 2222 2 ENV_ADDR = MSCP_ENV + 8; ! ADDR (TEXT + 0) OF FIRST MSCP ENVELOPE
: 2223 2 INCR COUNT FROM 0 TO (ENV_CNT - 1) DO ! FOR EACH MSCP ENVELOPE
: 2224 3 BEGIN
: 2225 3
: 2226 3 ENV_USE [.COUNT] = -1; ! MARK ENVELOPE FREE
: 2227 3 MSCP_ENV [.COUNT, ENV_LO] = .ENV_ADDR; ! LOAD ENVELOPE ADDR INTO ENV DESCRIPTOR
: 2228 3 MSCP_ENV [.COUNT, ENV_HI] = 0;
: 2229 3 MSCP_ENV [.COUNT, ENV_F] = 1; ! SET FLAG BIT
: 2230 3 MSCP_ENV [.COUNT, ENV_O] = 1; ! SET OWNERSHIP BIT
: 2231 3 ENV_ADDR = .ENV_ADDR + (ENV_LEN * 2); ! ADVANCE ADDR TO NEXT ENVELOPE
: 2232 3
: 2233 2 END;
: 2234 2
: 2235 2 INCR CTLR FROM 0 TO (MAX_CTLR - 1) DO ! FOR EACH CONTROLLER
: 2236 3 BEGIN
: 2237 3
: 2238 3 IF .CST [.CTLR, IP_ADDR] NEQA 0 ! IF CONTROLLER IS PRESENT
: 2239 3 THEN ! THEN
: 2240 4 BEGIN
: 2241 4
: 2242 4 SET_CPAR (.CTLR); ! SET UP CURRENT CONTROLLER PARAMETERS
: 2243 4 VEC_AD = .CST_ADDR [VEC_ADDR]; ! SET CURRENT CONTROLLER'S VECTOR ADDRESS
: 2244 4 BRLEVEL = .CST_ADDR [BR_LEV] + 5; ! SET CURRENT CONTROLLER'S BR LEVEL
: 2245 4 CTLR_INIT (); ! INIT DEVICE AND CTLR DATA
: 2246 4 IF .DCT_ADDR [STAT] EQLU ONLINE ! IF CONTROLLER IS STILL ALIVE
: 2247 4 THEN ! THEN
: 2248 5 BEGIN
: 2249 5
: 2250 5 INCR OFFSET FROM (0 + OF_UN) TO (3 + OF_UN) DO ! FOR EACH UNIT (PLATTER)
: 2251 6 BEGIN
: 2252 6
: 2253 6 IF .CST_ADDR [.OFFSET, P_PRES] EQLU PRESENT ! IF UNIT EXISTS
: 2254 6 THEN ! THEN
: 2255 7 BEGIN
: 2256 7
: 2257 7 SET_UPAR (.OFFSET); ! SET UP UNIT-RELATED DATA ITEMS
: 2258 7 UNIT_INIT (); ! BRING UNIT ONLINE
: 2259 7
: 2260 6 END; ! IF UNIT EXISTS
: 2261 6
: 2262 5 END; ! UNIT LOOP
: 2263 5
: 2264 4 END; ! IF CONTROLLER IS STILL ALIVE
: 2265 4
```


CZRC03
V02.0

CZRCDB0 RC25 DISK EXERCISER
INITIALIZATION SUBTEST ROUTINES

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC03.SRC;17

SEQ 0226
Page 30
(5)

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

```

; 2328 2 RESULT = HARD_INIT ();
; 2329 2 DCT_ADDR [IG_INT] = NO;
; 2330 2 IF .RESULT EQLU SUCCESS
; 2331 2 THEN
; 2332 3 BEGIN
; 2333 3
; 2334 3 DCT_ADDR [STAT] = ONLINE;
; 2335 3 CST_ADDR [STATE] = ONLINE;
; 2336 3 INI_RRING ();
; 2337 3 WRT_RC25 (RCSA, RC_ALL, SA_GO);
; 2338 3 SET_CTLR_CHAR ();
; 2339 3
; 2340 3 END
; 2341 3
; 2342 2 ELSE
; 2343 2 DROP_CTLR (.CCTLR, DU_INIT);
; 2344 2
; 2345 1 END;
! ATTEMPT HARD DEVICE INIT
! CLEAR "IGNORE INTERRUPT" BIT
! IF HARD INIT WAS SUCCESSFUL
! THEN
! MARK CONTROLLER ONLINE IN "DRIVER"
! MARK CONTROLLER ONLINE IN "PROGRAM"
! INITIALIZE RESPONSE RING
! SET "GO" BIT (START CTLR POLLING)
! SET CONTROLLER CHARACTERISTICS
! HARD INIT FAILED
! DROP ALL CONTROLLER'S UNITS
! ROUTINE CTLR_INIT

```

```

000000 010146 .SBTTL CTLR.INIT INITIALIZATION SUBTEST ROUTINES
000002 0C4737 000000V CTLR.INIT:
000006 012746 000240 MOV R1,-(SP) ; 2271
000012 013700 000000G JSR PC,INI.CTLR.DAT ; 2305
000016 006300 MOV #240,-(SP) ; 2306
000020 016046 000570' MOV CCTLR,R0
000024 013746 000612' ASL R0
000030 012746 000003 MOV INT.ADDR(R0),-(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$LUN
000066 004737 000000V JSR PC,REG.EXIST ; 2310
000072 005700 TST R0 ; 2310
000074 001404 BEQ 1$ ; 2314
000076 004737 000000V JSR PC,VEC.BR.TEST ; 2319
000102 005700 TST R0 ; 2319
000104 001011 BNE 2$
000106 013716 000000G 1$: MOV CCTLR,(SP) ; 2323
000112 012746 000002 MOV #2,-(SP) ;
000116 004737 000000G JSR PC,DROP.CTLR
000122 062706 000012 ADD #12,SP ; 2324
000126 000444 BR 5$ ; 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

```


CZRC03
V02.0

CZRCDB0 RC25 DISK EXERCISER
INITIALIZATION SUBTEST ROUTINES

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER\$1:[AZTEC.CZRCDB]CZRC03.SRC;17

SEQ 0229
Page 33
(6)

000200	013700	000000G		MOV	RC25.ADDR,R0		
000204	010160	000002		MOV	R1,2(R0)	; RC.REG,*	
000210	004737	000000V		JSR	PC,SET.CTLR.CHAR	:	
000214	000407			BR	4\$:	2338
000216	013716	000000G	3\$:	MOV	CCTLR,(SP)	:	2330
000222	012746	000002		MOV	#2,-(SP)	:	2343
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 1 ROUTINE INI_CTLR_DAT : NOVALUE =
; 2347 1
; 2348 1 !+
; 2349 1 ! THIS ROUTINE IS RESPONSIBLE FOR INITIALIZING ALL CONTROLLER-RELATED
; 2350 1 ! DATA IN THE "DRIVER" PORTION OF THE EXERCISER. THIS INCLUDES THE
; 2351 1 ! CONTROLLER'S DCT AND OUTSTANDING COMMAND LIST.
; 2352 1 !
; 2353 1 ! IMPLICIT INPUTS:
; 2354 1 ! CCTLR - CURRENT CONTROLLER NUMBER
; 2355 1 ! DCT_ADDR - ADDRESS OF CURRENT CONTROLLER'S DCT
; 2356 1 ! OCL_X1, OCL_X2 - STARTING AND ENDING INDECES OF CURRENT
; 2357 1 ! CONTROLLER'S OUTSTANDING COMMAND AREA
; 2358 1 !-
; 2359 1
; 2360 2 BEGIN
; 2361 2
; 2362 2 INCR INDEX FROM .OCL_X1 TO .OCL_X2 DO
; 2363 3 BEGIN
; 2364 3
; 2365 3 OUTC_LIST [.INDEX] = -1; ! INIT OUTSTANDING CMD LIST
; 2366 3 OUTC_TIMR [.INDEX] = 0; ! ZERO OUT OUTSTANDING CMD TIMERS
; 2367 3
; 2368 2 END;
; 2369 2
; 2370 2 DCT_ADDR [WORD0] = 0; ! CLEAR FIRST DCT WORD
; 2371 2 DCT_ADDR [RR_BEG] = COMM_AREA + 8 + (.CCTLR * COMM_LEN * 2); ! START OF RESPONSE RING
; 2372 2 DCT_ADDR [RR_END] = .DCT_ADDR [RR_BEG] + ((RRING_LEN - 1) * 4); ! LAST SLOT IN RESPONSE RING
; 2373 2 DCT_ADDR [CR_BEG] = .DCT_ADDR [RR_END] + 4; ! START OF COMMAND RING
; 2374 2 DCT_ADDR [CR_END] = .DCT_ADDR [CR_BEG] + ((CRING_LEN - 1) * 4); ! LAST SLOT IN COMMAND RING
; 2375 2 DCT_ADDR [RR_POLL] = .DCT_ADDR [RR_BEG]; ! FIRST RRING SLOT TO POLL
; 2376 2 DCT_ADDR [CR_POLL] = DCT_ADDR [CR_NEXT] = .DCT_ADDR [CR_BEG]; ! CRING POLL AND NEXT COMMAND POINTERS
; 2377 2
; 2378 1 END;

```

Address	Offset	OpCode	Instruction	Comment	Line No
000000	004137	000000G	.SBTTL INI.CTLR.DAT: INI.CTLR.DAT INITIALIZATION SUBTEST ROUTINES		
000004	013701	000000G	JSR R1,\$SAVE2		2346
000010	005301		MOV OCL.X1,R1	; *,INDEX	2362
000012	000407		DEC R1	; INDEX	
000014	112761	000377 000000G	BR 2\$		
000022	010100		1\$: MOVB #377,OUTC.LIST(R1)	; *,*(INDEX)	2365
000024	006300		MOV R1,R0	; INDEX,*	2366
000026	005060	000000G	ASL R0		
000032	005201		CLR OUTC.TIMR(R0)		
000034	020137	000000G	2\$: INC R1	; INDEX	2362
000040	003765		CMP R1,OCL.X2	; INDEX,*	
000042	013701	000000G	BLE 1\$		
000046	005011		MOV DCT.ADDR,R1		2370
000050	012702	000004	CLR (R1)		
000054	060102		MOV #4,R2		2371
000056	013746	000000G	ADD R1,R2		
000062	012746	000110	MOV CCTLR,-(SP)		
000066	004737	000000G	MOV #110,-(SP)		
000072	062700	000010'	JSR PC,BL\$MUL		
000076	010012		ADD #COMM.AREA+10,R0		
			MOV R0,(R2)		

CZRC03
V02.0

CZRCDB0 RC25 DISK EXERCISER
INITIALIZATION SUBTEST ROUTINES

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC03.SRC;17

SEQ 0231
Page 35
(7)

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

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

```

: 2379 1 ROUTINE REG_EXIST =
: 2380 1
: 2381 1 !*
: 2382 1 ! THIS IS THE REGISTER EXISTENCE (OR "PROBE") TEST DESIGNED TO VERIFY
: 2383 1 ! THE PRESENCE OF AN RC25 DEVICE. THIS OBJECTIVE IS ACCOMPLISHED BY
: 2384 1 ! SETTING UP THE NON-EXISTENT MEMORY (NEX) TRAP VECTOR (LOCATION 4) AND
: 2385 1 ! ATTEMPTING TO READ WHAT IS ASSUMED TO BE THE DEVICE'S SA AND IP
: 2386 1 ! REGISTERS. IF THE NEX TRAP HANDLER IS INVOKED DUE TO AN ABSENT DEVICE,
: 2387 1 ! THEN THE GLOBAL DATUM "NEX" WILL BE SET TO "TRUE". THIS DATUM
: 2388 1 ! DETERMINES THE SUCCESS / FAILURE VALUE OF THIS ROUTINE.
: 2389 1 !
: 2390 1 ! IMPLICIT INPUTS:
: 2391 1 ! RC25_ADDR - ADDRESS OF CURRENT CONTROLLER'S IP REGISTER
: 2392 1 !-
: 2393 1
: 2394 2 BEGIN
: 2395 2
: 2396 2 LOCAL
: 2397 2 TEMP : WORD, ! TEMP FOR READING SA AND IP
: 2398 2 DUMMY : WORD; ! AS THE NAME IMPLIES
: 2399 2
: 2400 2 OF_RC = 2; ! SET UP TO READ SA FIRST
: 2401 2 DO
: 2402 3 BEGIN
: 2403 3
: 2404 3 NEX = FALSE; ! SET TO "TRAP NOT RECEIVED"
: 2405 3 SETVEC (4, NEX_TRAP, PRI07); ! SET LOCATION 4 TRAP VECTOR ADDRESS
: 2406 3 TEMP = (.RC25_ADDR + .OF_RC); ! READ REGISTER (THEN TRAP OR CONTINUE)
: 2407 3 DUMMY = 0; ! DUMMY INSTRUCTION TO COVER TRAP RETURN BUG
: 2408 3 ! (TRAP RETURNS TO NEXT INSTRUCTION)
: 2409 3 CLRVEC (4); ! CLEAR LOCATION 4 TRAP VECTOR ADDRESS
: 2410 3 IF .NEX EQLU TRUE ! IF NEX TRAP OCCURRED
: 2411 3 THEN ! THEN
: 2412 4 BEGIN
: 2413 4
: 2414 4 ERRDF (10, EGD_10, EMS_10); ! "REGISTER EXISTENCE TEST FAILED"
: 2415 4 RETURN FAILURE;
: 2416 4
: 2417 4 END
: 2418 3 ELSE
: 2419 3 OF_RC = .OF_RC - 2; ! SET UP FOR IP REG OR QUIT
: 2420 3
: 2421 3 END
: 2422 3
: 2423 2 UNTIL .OF_RC LSS 0;
: 2424 2
: 2425 2 RETURN SUCCESS;
: 2426 2
: 2427 1 END;

```

```

000000 004137 000000G .SBTTL REG.EXIST INITIALIZATION SUBTEST ROUTINES
REG.EXIST:
000004 012737 000002 000000G JSR R1,$SAVE2
000012 005037 000000G MOV #2,OF_RC
000016 012746 000340 1$: CLR NEX
MOV #340,-(SP)

```

2379
2400
2404
2405

CZRCDB
V02.0

CZRCDB RC25 DISK EXERCISER
INITIALIZATION SUBTEST ROUTINES

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRCDB.SRC:17

SEQ 0233
Page 37
(8)

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	:		
000044	063700	000000G		ADD	OF.RC,R0		2406	
000050	011001			MOV	(R0),R1	:		
000052	005002			CLR	R2	:		
000054	012700	000004		MOV	#4,R0	:	2407	
000060	104436			TRAP	36	:	2409	
000062	023727	000000G	000001	CMP	NEX,#1	:		
000070	001007			BNE	2#	:	2410	
000072	104455			TRAP	55	:		
000074	000012			.WORD	12	:	2414	
000076	000000G			.WORD	EGD.10			
000100	000000G			.WORD	EMS.10			
000102	062706	000010		ADD	#10,SP	:		
000106	000413			BR	3#	:	2415	
000110	162737	000002	000000G	2#:	SUB	#2,OF.RC	:	2412
000116	062706	000010		ADD	#10,SP	:	2419	
000122	005737	000000G		TST	OF.RC	:	2402	
000126	002331			BGE	1#	:	2423	
000130	012700	000001		MOV	#1,R0	:		
000134	000207			RTS	PC	:	2394	
000136	005000			3#:	CLR	R0	:	
000140	000207			RTS	PC	:	2379	

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

```

: 2428 1  ROUTINE VEC_BR_TEST =
: 2429 1
: 2430 1  !*
: 2431 1  !
: 2432 1  !   THIS ROUTINE ATTEMPTS TO VERIFY (A) THAT THE RC25 VECTOR ADDRESS GIVEN
: 2433 1  !   BY THE USER DURING THE HW DIALOG IS VALID, AND (B) THAT THE
: 2434 1  !   USER-SPECIFIED BUS REQUEST LEVEL FOR THE DEVICE IS CORRECT. THE FIRST
: 2435 1  !   OBJECTIVE IS ACCOMPLISHED BY SETTING THE CPU PRIORITY TO 0 AND FORCING
: 2436 1  !   AN RC25 INTERRUPT. IF THE USER SPECIFIED AN INCORRECT VECTOR ADDRESS,
: 2437 1  !   THEN THE RESULT MAY BE UNPREDICTABLE. FOR THIS REASON, THE MESSAGE
: 2438 1  !   "ABOUT TO VERIFY VECTOR XXX(O) FOR DEVICE XXXXXX(O) ..." IS PRINTED
: 2439 1  !   BEFORE THE TEST, AND "COMPLETED" IS PRINTED AT ITS SUCCESSFUL
: 2440 1  !   CONCLUSION. IF THE WORD "COMPLETED" DOES NOT APPEAR, AND AN ERROR IS
: 2441 1  !   NOT REPORTED, THEN PROGRAM CONTROL IS ASSUMED LOST AND A FATAL TRAP HAS
: 2442 1  !   LIKELY OCCURRED. AT THIS POINT, THE EXERCISER MUST BE STARTED AGAIN.
: 2443 1  !
: 2444 1  !   IF THIS TEST SUCCEEDS, THEN THE BR LEVEL TEST IS RUN BY SETTING THE
: 2445 1  !   PROCESSOR PRIORITY TO THE ASSUMED INTERRUPT PRIORITY GIVEN BY THE
: 2446 1  !   USER. A FORCED INTERRUPT SHOULD NOT OCCUR. THEN, BY LOWERING THE
: 2447 1  !   PRIORITY BY ONE, THE DELAYED INTERRUPT SHOULD OCCUR.
: 2448 1  !   RC25_ADDR - ADDRESS OF CURRENT CONTROLLER'S IP REGISTER
: 2449 1  !
: 2450 1  !   IMPLICIT INPUTS:
: 2451 1  !   DCT_ADDR - ADDRESS OF CURRENT CONTROLLER'S DCT
: 2452 1  !   RC25_ADDR - ADDRESS OF CURRENT CONTROLLER'S IP REGISTER
: 2453 1  !   VEC_AD - ASSUMED VECTOR ADDRESS OF THE CURRENT CONTROLLER
: 2454 1  !   BRLEVEL - ASSUMED BUS REQUEST INTERRUPT LEVEL OF THE CURRENT
: 2455 1  !   CONTROLLER
: 2456 1  !-
: 2457 2  BEGIN
: 2458 2
: 2459 3  IF MANUAL          ! IF ATTENDED
: 2460 2  THEN          ! THEN
: 2461 2  PRINTF (MSG_02, .VEC_AD, .RC25_ADDR); ! "ABOUT TO VERIFY VECTOR..."
: 2462 2  IF INT_GEN () EQLU FALSE ! FORCE AN INTERRUPT
: 2463 2  THEN          ! IF INTERRUPT DID NOT OCCUR
: 2464 3  BEGIN
: 2465 3
: 2466 3  ERRDF (11, EGD_11, 0); ! "VECTOR TEST FAILED"
: 2467 3  RETURN FAILURE;
: 2468 3
: 2469 3  END
: 2470 2  ELSE          ! INTERRUPT DID OCCUR
: 2471 3  BEGIN
: 2472 3
: 2473 4  IF MANUAL          ! IF ATTENDED
: 2474 3  THEN          ! THEN
: 2475 3  PRINTF (MSG_03); ! "COMPLETED."
: 2476 3  SETPRI (.BRLEVEL); ! SET PRIORITY TO ASSUMED BR LEVEL
: 2477 3  IF INT_GEN () EQLU FALSE ! FORCE AN INTERRUPT (SHOULD NOT OCCUR)
: 2478 3  THEN          ! IF INTERRUPT DID NOT OCCUR
: 2479 4  BEGIN
: 2480 4
: 2481 4  SETPRI (.BRLEVEL - #0'40'); ! LOWER PRIORITY BY 1
: 2482 4  DELAY (1); ! WAIT
: 2483 4  IF .DCT_ADDR [SA_SAVE] NEQU 0 ! IF INTERRUPT DID OCCUR (SA_SAVE WOULD BE NON-ZERO)
: 2484 4  THEN          ! THEN
```



```

: 2485 5          BEGIN
: 2486 5
: 2487 5          SETPRI (PRI00);          ! RESTORE PROCESSOR PRIORITY TO 0
: 2488 5          RETURN SUCCESS;        ! ONLY SUCCESSFUL EXIT POINT
: 2489 5
: 2490 4          END;
: 2491 4
: 2492 3          END;
: 2493 3
: 2494 2          END;
: 2495 2
: 2496 2          SETPRI (PRI00);          ! COME HERE ONLY FOR BR TEST FAILURE
: 2497 2          ERRDF (12, EGD_12, EMS_12); ! "BR LEVEL TEST FAILED"
: 2498 2          RETURN FAILURE;
: 2499 2
: 2500 1          END;

```

.GLOBL L\$DLY

```

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

```

CZRCDB3
V02.0

CZRCDB0 RC25 DISK EXERCISER
INITIALIZATION SUBTEST ROUTINES

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER\$1:[AZTEC.CZRCDB]CZRCDB3.SRC;17

SEQ 0236
Page 40
(9)

000134	012701	000001		MOV	#1,R1				
000140	001410		4\$:	BEQ	7\$; *,\$\$TMP2		2482
000142	013700	000000G		MOV	L\$DLY,R0				
000146	001403			BEQ	6\$; *,\$\$TMP1		
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				
000166	005760	000002		TST	2(R0)				2483
000172	001405			BEQ	8\$				
000174	005000			CLR	R0				
000176	104441			TRAP	41				2487
000200	012700	000001		MOV	#1,R0				
000204	000407			BR	10\$				2485
000206	005000		8\$:	CLR	R0				
000210	104441			TRAP	41				2496
000212	104455			TRAP	55				
000214	000014			.WORD	14				2497
000216	000000G			.WORD	EGD.12				
000220	000000G			.WORD	EMS.12				
000222	005000		9\$:	CLR	R0				
000224	005726		10\$:	TST	(SP)+				2428
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 1 ROUTINE INT_GEN =
: 2502 1
: 2503 1 !*
: 2504 1 !
: 2505 1 ! THIS ROUTINE BEGINS AN RC25 INITIALIZATION SEQUENCE, BUT ONLY
: 2506 1 ! COMPLETES THROUGH THE STEP 1 WRITE. ITS PURPOSE IS TO CREATE AN RC25
: 2507 1 ! INTERRUPT (AT THE COMPLETION OF STEP 1) IN ORDER TO HELP VERIFY THE
: 2508 1 ! THE USER-SPECIFIED VECTOR ADDRESS AND BUS REQUEST INTERRUPT LEVEL.
: 2509 1 ! A VALUE OF "TRUE" IS RETURNED TO THE CALLER IF AN INTERRUPT OCCURS,
: 2510 1 ! AND "FALSE" OTHERWISE. THE INTERRUPT IS VERIFIED BY A NON-ZERO VALUE
: 2511 1 ! IN THE "SA SAVE" WORD IN THE DEVICE'S DCT.
: 2512 1 !
: 2513 1 ! IMPLICIT INPUTS:
: 2514 1 ! DCT_ADDR - ADDRESS OF CURRENT CONTROLLER'S DCT
: 2515 1 ! RC25_ADDR - ADDRESS OF CURRENT CONTROLLER'S IP REGISTER
: 2516 1 !-
: 2517 2 BEGIN
: 2518 2
: 2519 2 LOCAL
: 2520 2 IE_VEC : WORD, ! IE-BIT-AND-VECTOR-ADDRESS/4 BYTE
: 2521 2 SA : WORD; ! STORAGE FOR STEP 1 READ AND WRITE
: 2522 2
: 2523 2 DCT_ADDR [SA_SAVE] = 0; ! ZERO OUT SA SAVE WORD IN DCT
: 2524 2 IE_VEC = (.VEC_AD + -2) OR SA_INT; ! VEC ADDR / 4 WITH IE = 1
: 2525 2 WRT_RC25 (RCIP, RC_ALL, ALL_ONES); ! WRITE IP TO START INIT SEQUENCE
: 2526 2 DELAY (625); ! WAIT Changed from 500 to 625 for 84 cache
: 2527 2 SA = .RC25_ADDR [RCSA, RC_ALL]; ! STEP 1 READ
: 2528 2 SA = (WR_RING + 8) OR .IE_VEC; ! STEP 1 WRITE VALUE
: 2529 2 WRT_RC25 (RCSA, RC_ALL, .SA); ! STEP 1 WRITE
: 2530 2 INCR COUNT FROM 1 TO 10 DO ! TEN SECOND LIMIT
: 2531 3 BEGIN
: 2532 3
: 2533 3 DELAY (400); ! ABOUT 1 SECOND Changed 333 to 400 for 84
: 2534 3 SA = .RC25_ADDR [RCSA, RC_ALL]; ! READ SA REGISTER
: 2535 4 IF BIT_TST (SA, SA_S2) ! IF STEP 2 HAS BEGUN
: 2536 3 THEN ! THEN
: 2537 3 EXITLOOP; ! BREAK OUT
: 2538 3
: 2539 2 END;
: 2540 2
: 2541 2 IF .DCT_ADDR [SA_SAVE] EQLU 0 ! IF SA SAVE WORD WAS NOT TOUCHED
: 2542 2 THEN ! THEN
: 2543 2 RETURN FALSE ! INTERRUPT DID NOT OCCUR
: 2544 2 ELSE ! OTHERWISE
: 2545 2 RETURN TRUE; ! INTERRUPT DID OCCUR
: 2546 2
: 2547 1 END;

```

000000	004137	000000G	.SBTTL	INT.GEN INITIALIZATION SUBTEST ROUTINES	
000004	162706	000006	INT.GEN:JSR	R1,\$SAVES	2501
000010	013704	000000G	SUB	#6,SP	
000014	005064	000002	MOV	DCT.ADDR,R4	2523
000020	013700	000612'	CLR	2(R4)	
000024	006200		MOV	VEC.AD,R0	2524
000026	006200		ASR	R0	
			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	
000042	010077	000000G	MOV	R0,@RC25.ADDR	; RC.REG,*	2525
000046	012701	001161	MOV	#1161,R1	; *,\$\$TMP2	
000052	001411		1\$: BEQ	4\$		2526
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	011516		MOV	(R5),(SP)	; *,RC.REG	
000112	010302		MOV	R3,R2	; IE.VEC,SA	2528
000114	052702	115400	BIS	#115400,R2	; *,SA	
000120	010200		MOV	R2,R0	; SA,RC.REG	2529
000122	010015		MOV	R0,(R5)	; RC.REG,*	
000124	012703	000012	MOV	#12,R3	; *,COUNT	2530
000130	012701	000620	5\$: MOV	#620,R1	; *,\$\$TMP2	2533
000134	0C1411		6\$: BEQ	9\$		
000136	013700	000000G	MOV	L\$DLY,R0	; *,\$\$TMP1	
000142	001404		BEQ	8\$		
000144	005066	000004	7\$: CLR	4(SP)	; \$\$TMP	
000150	005300		DEC	R0	; \$\$TMP1	
000152	001374		BNE	7\$		
000154	005301		8\$: DEC	R1	; \$\$TMP2	
000156	000766		BR	6\$		
000160	011566	000002	9\$: MOV	(R5),2(SP)	; *,RC.REG	2534
000164	011502		MOV	(R5),R2	; RC.REG,SA	
000166	032702	010000	BIT	#10000,R2	; *,SA	2535
000172	001002		BNE	10\$		
000174	005303		DEC	R3	; COUNT	2530
000176	001354		BNE	5\$		
000200	005764	000002	10\$: TST	2(R4)		2541
000204	001002		BNE	11\$		
000206	005000		CLR	R0		2545
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 1 ROUTINE HARD_INIT =
: 2549 1
: 2550 1 !+
: 2551 1 ! THIS ROUTINE PERFORMS THE FOUR READ / WRITE STEPS REQUIRED TO
: 2552 1 ! INITIALIZE AN RC25 DEVICE. IF NO READ ERRORS ARE DETECTED IN ANY OF
: 2553 1 ! THE FOUR STEPS, THEN A SUCCESS VALUE IS RETURNED TO THE CALLER.
: 2554 1 ! OTHERWISE, ADDITIONAL ATTEMPTS MAY BE MADE TO INITIALIZE THE DEVICE.
: 2555 1 ! IF ALL ATTEMPTS FAIL, A FAILURE INDICATION IS RETURNED.
: 2556 1 !
: 2557 1 ! IMPLICIT INPUTS:
: 2558 1 ! RC25_ADDR - ADDRESS OF CURRENT CONTROLLER'S IP REGISTER
: 2559 1 ! VEC_AD - ASSUMED VECTOR ADDRESS OF THE CURRENT CONTROLLER
: 2560 1 ! DCT_ADDR - ADDRESS OF CURRENT CONTROLLER'S DCT
: 2561 1 !-
: 2562 2 BEGIN
: 2563 2
: 2564 2 LOCAL
: 2565 2 IE_VEC : WORD; ! IE-BIT-AND-VECTOR-ADDRESS/4 BYTE
: 2566 2 ! (USED IN STEP 1 WRITE AND STEP 3 READ)
: 2567 2
: 2568 2 IE_VEC = .VEC_AD + -2; ! GET VECTOR ADDR/4 (IE = 0)
: 2569 2 INCR ATTEMPTS FROM 1 TO INI_ATT DO
: 2570 3 BEGIN
: 2571 3
: 2572 3 WRT_RC25 (RCIP, RC_ALL, ALL_ONES); ! WRITE IP TO START INIT SEQUENCE
: 2573 3 DO
: 2574 4 BEGIN
: 2575 4 DELAY (100);
: 2576 4 END
: 2577 3 UNTIL .T_FLAG EQLU TRUE; ! IF ONE SECOND HAS ELAPSED THEN CONTINUE
: 2578 3 T_FLAG = FALSE;
: 2579 3 DO
: 2580 4 BEGIN
: 2581 4 DELAY (100);
: 2582 4 END
: 2583 3 UNTIL .T_FLAG EQLU TRUE; ! IF ONE SECOND HAS ELAPSED THEN CONTINUE
: 2584 3 T_FLAG = FALSE;
: 2585 3
: 2586 3 !
: 2587 3 ! STEP 1
: 2588 3 !
: 2589 3 STEP = 1;
: 2590 3 SA_REG = .RC25_ADDR [RCSA, RC_ALL]; ! READ SA
: 2591 3 IF (.SA_REG AND S1_MASK) EQLU SA_S1 ! IF STEP 1 READ IS O.K.
: 2592 3 THEN ! THEN
: 2593 4 BEGIN
: 2594 4
: 2595 4 SA_REG = (WR_RING + 8) OR .IE_VEC; ! STEP 1 WRITE VALUE
: 2596 4 WRT_RC25 (RCSA, RC_ALL, .SA_REG); ! STEP 1 WRITE
: 2597 4 INCR COUNT FROM 1 TO 10 DO ! TEN SECOND LIMIT
: 2598 5 BEGIN
: 2599 5
: 2600 5 ! DELAY (400); ! ABOUT 1 SECOND 333 old 400 new
: 2601 5 DO
: 2602 6 BEGIN
: 2603 6 DELAY (100);
: 2604 6 END
```

```
: 2605 5      UNTIL .T_FLAG EQLU TRUE;                ! IF ONE SECOND HAS ELAPSED THEN CONTINUE
: 2606 5      T_FLAG = FALSE;
: 2607 5      SA_REG = .RC25_ADDR [RCSA, RC_ALL]; ! READ SA REGISTER
: 2608 6      IF BIT_TST (SA_REG, SA_S2)           ! IF STEP 2 HAS BEGUN
: 2609 5      THEN                                ! THEN
: 2610 5      EXITLOOP;                          ! BREAK OUT
: 2611 5
: 2612 4      END;
: 2613 4      !
: 2614 4      ! STEP 2
: 2615 4      !
: 2616 4      STEP = .STEP + 1;
: 2617 5      IF (.SA_REG AND S2_MASK) EQLU (SA_S2 OR WR_RING) ! IF STEP 2 READ IS O.K.
: 2618 4      THEN                                ! THEN
: 2619 5      BEGIN
: 2620 5
: 2621 5      WRT_RC25 (RCSA, RC_ALL, .DCT_ADDR [RR_BEG]); ! RINGBASE-LO, PI = 0
: 2622 5      INCR COUNT FROM 1 TO 10 DO          ! TEN SECOND LIMIT
: 2623 6      BEGIN
: 2624 6
: 2625 6      ! DELAY (400);                       ! ABOUT 1 SECOND changed 333 to 400 for 11/84 cache
: 2626 6      !
: 2627 7      DO
: 2628 7      BEGIN
: 2629 7      DELAY (100);
: 2630 6      END
: 2631 6      UNTIL .T_FLAG EQLU TRUE;            ! IF ONE SECOND HAS ELAPSED THEN CONTINUE
: 2632 6      T_FLAG = FALSE;
: 2633 7      SA_REG = .RC25_ADDR [RCSA, RC_ALL]; ! READ SA REGISTER
: 2634 6      IF BIT_TST (SA_REG, SA_S3)         ! IF STEP 3 HAS BEGUN
: 2635 6      THEN                                ! THEN
: 2636 6      EXITLOOP;                          ! BREAK OUT
: 2637 5
: 2638 5      END;
: 2639 5      !
: 2640 5      ! STEP 3
: 2641 5      !
: 2642 6      STEP = .STEP + 1;
: 2643 5      IF (.SA_REG AND S3_MASK) EQLU (SA_S3 OR .IE_VEC) ! IF STEP 3 READ IS O.K.
: 2644 6      THEN                                ! THEN
: 2645 6      BEGIN
: 2646 6
: 2647 6      WRT_RC25 (RCSA, RC_ALL, 0);        ! PP, RINGBASE-HI = 0
: 2648 6      INCR COUNT FROM 1 TO 10 DO          ! TEN SECOND LIMIT
: 2649 7      BEGIN
: 2650 7      ! DELAY (400);                       ! ABOUT 1 SECOND changed 333 to 400 for 11/84 cache
: 2651 7      !
: 2652 8      DO
: 2653 8      BEGIN
: 2654 8      DELAY (100);
: 2655 7      END
: 2656 7      UNTIL .T_FLAG EQLU TRUE;            ! IF ONE SECOND HAS ELAPSED THEN CONTINE
: 2657 7      T_FLAG = FALSE;
: 2658 7
: 2659 8      SA_REG = .RC25_ADDR [RCSA, RC_ALL]; ! READ SA REGISTER
: 2660 7      IF BIT_TST (SA_REG, SA_S4)         ! IF STEP 4 HAS BEGUN
: 2661 7      THEN                                ! THEN
: 2661 7      EXITLOOP;                          ! BREAK OUT
```


CZRC03
V02.0

CZRCDB0 RC25 DISK EXERCISER
INITIALIZATION SUBTEST ROUTINES

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC03.SRC;17

SEQ 0242
Page 46
(11)

000122	010400			MOV	R4,R0			; *,\$\$TMP1	
000124	001404			BEQ	10\$				
000126	005066	000012		9\$: CLR	12(SP)			; \$\$TMP	
000132	005300			DEC	R0			; \$\$TMP1	
000134	001374			BNE	9\$				
000136	005301			10\$: DEC	R1			; \$\$TMP2	
000140	000767			BR	8\$				
000142	123727	000000G	000001	11\$: CMPB	T.FLAG,#1				2583
000150	001361			BNE	7\$				
000152	105037	000000G		CLRB	T.FLAG				2584
000156	012737	000001	000000G	MOV	#1,STEP				2589
000164	011266	000002		MOV	(R2),2(SP)			; *,RC.REG	2590
000170	011237	000000G		MOV	(R2),SA.REG			; RC.REG,*	2591
000174	011200			MOV	(R2),R0			; SA.REG,*	
000176	042700	001777		BIC	#1777,R0				2591
000202	020027	004000		CMP	R0,#4000				
000206	001177			BNE	30\$				
000210	011637	000000G		MOV	(SP),SA.REG			; IE.VEC,*	2595
000214	052737	115400	000000G	BIS	#115400,SA.REG				
000222	013700	000000G		MOV	SA.REG,R0			; *,RC.REG	2596
000226	010012			MOV	R0,(R2)			; RC.REG,*	
000230	012703	000012		MOV	#12,R3			; *,COUNT	2597
000234	012701	000144		12\$: MOV	#144,R1			; *,\$\$TMP2	2603
000240	0C1410			13\$: BEQ	16\$				
000242	010400			MOV	R4,R0			; *,\$\$TMP1	
000244	001404			BEQ	15\$				
000246	005066	000012		14\$: CLR	12(SP)			; \$\$TMP	
000252	005300			DEC	R0			; \$\$TMP1	
000254	001374			BNE	14\$				
000256	005301			15\$: DEC	R1			; \$\$TMP2	
000260	000767			BR	13\$				
000262	123727	000000G	000001	16\$: CMPB	T.FLAG,#1				2605
000270	001361			BNE	12\$				
000272	105037	000000G		CLRB	T.FLAG				2606
000276	011266	000004		MOV	(R2),4(SP)			; *,RC.REG	2607
000302	011237	000000G		MOV	(R2),SA.REG			; RC.REG,*	
000306	032712	010000		BIT	#10000,(R2)			; *,SA.REG	2608
000312	001002			BNE	17\$				
000314	005303			DEC	R3			; COUNT	2597
000316	001346			BNE	12\$				
000320	005237	000000G		17\$: INC	STEP				2616
000324	013700	000000G		MOV	SA.REG,R0				2617
000330	042700	003400		BIC	#3400,R0				
000334	020027	010233		CMP	R0,#10233				
000340	001127			BNE	31\$				
000342	013700	000000G		MOV	DCT.ADDR,R0				2621
000346	016000	000004		MOV	4(R0),R0			; *,RC.REG	
000352	010012			MOV	R0,(R2)			; RC.REG,*	
000354	012703	000012		MOV	#12,R3			; *,COUNT	2622
000360	012701	000144		18\$: MOV	#144,R1			; *,\$\$TMP2	2628
000364	001410			19\$: BEQ	22\$				
000366	010400			MOV	R4,R0			; *,\$\$TMP1	
000370	001404			BEQ	21\$				
000372	005066	000012		20\$: CLR	12(SP)			; \$\$TMP	
000376	005300			DEC	R0			; \$\$TMP1	
000400	001374			BNE	20\$				
000402	005301			21\$: DEC	R1			; \$\$TMP2	

CZRC03
V02.0

CZRCDB0 RC25 DISK EXERCISER
INITIALIZATION SUBTEST ROUTINES

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC03.SRC;17
SEQ 0243
Page 47
(11)

000404	000767			BR	19\$		
000406	123727	000000G 000001	22\$:	CMPB	T.FLAG,#1	:	2630
000414	001361			BNE	18\$		
000416	105037	000000G		CLRB	T.FLAG	:	2631
000422	011266	000000G		MOV	(R2),6(SP)	: *,RC.REG	2632
000426	011237	000000G		MOV	(R2),SA.REG	: RC.REG,*	
000432	032712	020000		BIT	#20000,(R2)	: *,SA.REG	2633
000436	001002			BNE	23\$		
000440	005303			DEC	R3	: COUNT	2622
000442	001346			BNE	18\$		
000444	005237	000000G	23\$:	INC	STEP	:	2641
000450	013701	000000G		MOV	SA.REG,R1	:	2642
000454	042701	003400		BIC	#3400,R1		
000460	011600			MOV	(SP),R0	: IE.VEC,*	
000462	052700	020000		BIS	#20000,R0		
000466	020100			CMP	R1,R0		
000470	001053			BNE	31\$		
000472	005000			CLR	R0	: RC.REG	2646
000474	005012			CLR	(R2)		
000476	012703	000012		MOV	#12,R3	: *,COUNT	2647
000502	012701	000144	24\$:	MOV	#144,R1	: *,\$\$TMP2	2653
000506	001410		25\$:	BEQ	28\$		
000510	010400			MOV	R4,R0	: *,\$\$TMP1	
000512	001404			BEQ	27\$		
000514	005066	000012	26\$:	CLR	12(SP)	: \$\$TMP	
000520	005300			DEC	R0	: \$\$TMP1	
000522	001374			BNE	26\$		
000524	005301		27\$:	DEC	R1	: \$\$TMP2	
000526	000767			BR	25\$		
000530	123727	000000G 000001	28\$:	CMPB	T.FLAG,#1	:	2655
000536	001361			BNE	24\$		
000540	105037	000000G		CLRB	T.FLAG	:	2656
000544	011266	000010		MOV	(R2),10(SP)	: *,RC.REG	2658
000550	011237	000000G		MOV	(R2),SA.REG	: RC.REG,*	
000554	032712	040000		BIT	#40000,(R2)	: *,SA.REG	2659
000560	001002			BNE	29\$		
000562	005303			DEC	R3	: COUNT	2647
000564	001346			BNE	24\$		
000566	005237	000000G	29\$:	INC	STEP	:	2667
000572	013700	000000G		MOV	SA.REG,R0	:	2668
000576	042700	003777		BIC	#3777,R0		
000602	020027	040000		CMP	R0,#40000		
000606	001004		30\$:	BNE	31\$		
000610	005012			CLR	(R2)	:	2672
000612	012700	000001		MOV	#1,R0	:	2670
000616	000411			BR	33\$		
000620	005305		31\$:	DEC	R5	: ATTEMPTS	2569
000622	001402			BEQ	32\$		
000624	000137	002060'		JMP	1\$		
000630	104455		32\$:	TRAP	55	:	2685
000632	000015			.WORD	15		
000634	000000G			.WORD	EGD.13		
000636	000000G			.WORD	EMS.13		
000640	005000			CLR	R0	:	2562
000642	062706	000014	33\$:	ADD	#14,SP	:	2548
000646	000207			RTS	PC		

; Routine Size: 212 words, Routine Base: \$CODE\$ + 2016
; Maximum stack depth per invocation: 14 words

```

; 2689 1  ROUTINE INI_RRING : NOVALUE =
; 2690 1
; 2691 1  !+
; 2692 1  !
; 2693 1  !   THIS ROUTINE IS RESPONSIBLE FOR ALLOCATING ENOUGH MSCP ENVELOPES TO
; 2694 1  !   FILL AN RC25 RESPONSE RING. THE ENVELOPE DESCRIPTOR OF EACH ENVELOPE
; 2695 1  !   (LOCATED IN FRONT OF THE ENVELOPE ITSELF) IS LOADED INTO SUCCESSIVE
; 2696 1  !   RRING SLOTS. NOTE THAT THE ENVELOPE DESCRIPTORS HAVE BEEN INITIALIZED
; 2697 1  !   WITH THE FLAG AND OWNERSHIP BITS SET TO "1", MAKING EACH SLOT
; 2698 1  !   CONTROLLER-OWNED.
; 2699 1  !
; 2700 1  !   IMPLICIT INPUTS:
; 2701 1  !           CCTLN - CURRENT CONTROLLER NUMBER
; 2702 1  !           DCT_ADDR - ADDRESS OF CURRENT CONTROLLER'S DCT
; 2703 1  !-
; 2704 2  BEGIN
; 2705 2
; 2706 2  LOCAL
; 2707 2     INDEX : WORD,
; 2708 2     RRING_ADDR;
; 2709 2
; 2710 2  RRING_ADDR = .DCT_ADDR [RR_BEG];           ! FIRST RESPONSE RING SLOT
; 2711 2  INCR COUNT FROM 1 TO RRING_LEN DO
; 2712 3     BEGIN
; 2713 3
; 2714 3     INDEX = GET_ENV (.CCTLN);           ! GET AN MSCP ENVELOPE
; 2715 3     MSCP_ENV [.INDEX, MSGLEN] = MSG_LEN * 2; ! SET MESSAGE LENGTH FIELD
; 2716 3     .RRING_ADDR = .MSCP_ENV [.INDEX, ENV_LO]; ! LOAD LO-ORDER ENV DESC INTO SLOT
; 2717 3     RRING_ADDR = .RRING_ADDR + 2;       ! ADVANCE TO SECOND WORD
; 2718 3     .RRING_ADDR = .MSCP_ENV [.INDEX, ENV_HI]; ! LOAD HI-ORDER ENV DESC INTO SLOT
; 2719 3     RRING_ADDR = .RRING_ADDR + 2;       ! ADVANCE TO NEXT SLOT
; 2720 3
; 2721 2  END;
; 2723 1  END;

```

```

;SBTTL INI.RRING INITIALIZATION SUBTEST ROUTINES
000000 004137 000000G INI.RRING:
000004 013700 000000G JSR R1,$SAVE3 ;
000010 016001 000004 MOV DCT.ADDR,R0 ; 2689
000014 012702 000010 MOV 4(R0),R1 ; *,RRING.ADDR 2710
000020 013746 000000G 1$: MOV #10,R2 ; *,COUNT 2711
000024 004737 000000G JSR PC,GET_ENV ; 2714
000030 010003 MOV R0,R3 ; *,INDEX
000032 010316 MOV R3,(SP) ; INDEX,* 2715
000034 012746 000104 MOV #104,-(SP)
000040 004737 000000G JSR PC,BL#MUL
000044 012760 000074 000004G MOV #74,MSCP_ENV+4(R0)
000052 016021 000000G MOV MSCP_ENV(R0),(R1)+ ; *,RRING.ADDR 2716
000056 016021 000002G MOV MSCP_ENV+2(R0),(R1)+ ; *,RRING.ADDR 2718
000062 022626 CMP (SP)+,(SP)+ ; 2712
000064 005302 DEC R2 ; COUNT 2711
000066 001354 BNE 1$
000070 000207 RTS PC ; 2689

```


; Routine Size: 29 words, Routine Base: \$CODE\$ + 2666
; Maximum stack depth per invocation: 7 words

```

; 2724 1  ROUTINE SET_CTLR_CHAR : NOVALUE =
; 2725 1
; 2726 1  !+
; 2727 1  !
; 2728 1  !   THIS ROUTINE IS CALLED BY CTLR_INIT AFTER THE RC25 HAS BEEN HARD-
; 2729 1  !   INITIALIZED. ITS PURPOSE IS TO FORMAT AND SEND THE "SET CONTROLLER
; 2730 1  !   CHARACTERISTICS" COMMAND, AND TO VERIFY THAT THE MESSAGE WAS ACCEPTED.
; 2731 1  !
; 2732 1  !   IMPLICIT INPUTS:
; 2733 1  !   CCTLR - CURRENT CONTROLLER NUMBER
; 2734 1  !-
; 2735 2  BEGIN
; 2736 2
; 2737 2  LOCAL
; 2738 2  REASON : WORD,           ! DROP UNIT REASON
; 2739 2  M_INDEX : WORD;         ! MSCP ENVELOPE INDEX
; 2740 2
; 2741 2  M_INDEX = GET_ENV (.CCTLR);           ! GET AN MSCP ENVELOPE
; 2742 2  MSCP_ENV [.M_INDEX, OPCODE] = OP_SCC;   ! OPCODE = SET CTLR CHAR
; 2743 2  MSCP_ENV [.M_INDEX, C_FLAGS] = CF_MSC + CF_TMS; ! CONTROLLER FLAGS
; 2744 2  IF SEND (.M_INDEX) EQLU FAILURE       ! ATTEMPT SEND; IF CTLR IS OFFLINE
; 2745 2  THEN                                     ! THEN
; 2746 2  PUT_ENV (.M_INDEX)                   ! RETURN ENVELOPE TO POOL
; 2747 2  ELSE                                     ! IF SEND WAS SUCCESSFUL
; 2748 3  BEGIN
; 2749 3
; 2750 3  WAIT ();                               ! WAIT FOR RETPKT RESPONSE
; 2751 3  RP_INDX = OUT_IODQ ();                 ! GET INDEX OF RETPKT
; 2752 3  RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2); ! CALCULATE RETPKT ADDRESS
; 2753 3  IF .RP_ADDR [CONID] EQLU CID_DRIVER   ! IF RETPKT IS FROM "DRIVER"
; 2754 3  THEN                                     ! THEN
; 2755 4  BEGIN
; 2756 4
; 2757 4  REASON = DU_FATAL;                     ! ASSUME FATAL SA ERROR
; 2758 4  IF .RP_ADDR [MESTYP] EQLU MT_TIMEOUT ! IF COMMAND TIMED OUT
; 2759 4  THEN                                     ! THEN
; 2760 5  BEGIN
; 2761 5
; 2762 5  REASON = DU_INIT;                       ! SET REASON TO INIT ERROR
; 2763 5  ERRDF (15, EGD_15, EMS_15);           ! "MESSAGE RESPONSE TIMEOUT"
; 2764 5
; 2765 4  END;
; 2766 4
; 2767 4  DROP_CTLR (.CCTLR, .REASON);          ! DROP CONTROLLER'S UNITS
; 2768 4
; 2769 3  END;                                     ! IF RETPKT IS FROM "DRIVER"
; 2770 3
; 2771 3  PUT_RETPKT (.RP_INDX);
; 2772 3
; 2773 2  END;                                     ! IF SEND WAS SUCCESSFUL
; 2775 1  END;                                     ! ROUTINE SET_CTLR_CHAR

```

000000 010146

```

.SBTTL SET.CTLR.CHAR INITIALIZATION SUBTEST ROUTINES
SET.CTLR.CHAR:
MOV R1, -(SP)

```

2724

CZRC03
V02.0

CZRCDB0 RC25 DISK EXERCISER
INITIALIZATION SUBTEST ROUTINES

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC03.SRC:17

SEQ 0246
Page 52
(13)

000002	013746	000000G		MOV	CCTLR, -(SP)	:		
000006	004737	000000G		JSR	PC, GET.ENV	:		2741
000012	010001			MOV	R0, R1	:	*.M.INDEX	
000014	010116			MOV	R1, (SP)	:	M.INDEX, *	2742
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)	:		2743
000042	010116			MOV	R1, (SP)	:	M.INDEX, *	2744
000044	004737	000000G		JSR	PC, SEND			
000050	005700			TST	R0			
000052	001004			BNE	1#			
000054	010116			MOV	R1, (SP)	:	M.INDEX, *	2746
000056	004737	000000G		JSR	PC, PUT.ENV			
000062	000455			BR	4#	:		2744
000064	004737	000000G	1#:	JSR	PC, WAIT	:		2750
000070	004737	000000G		JSR	PC, OUT.IODQ	:		2751
000074	010037	000000G		MOV	R0, RP.INDX	:		
000100	010016			MOV	R0, (SP)	:	RP.INDX, *	2752
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	:		2753
000130	001025			BNE	3#			
000132	012701	000004		MOV	#4, R1	:	*.REASON	2757
000136	116000	000002		MOVB	2(R0), R0	:		2758
000142	042700	177417		BIC	#177417, R0			
000146	020027	000100		CMP	R0, #100			
000152	001006			BNE	2#			
000154	012701	000002		MOV	#2, R1	:	*.REASON	2762
000160	104455			TRAP	55	:		2763
000162	000017			.WORD	17			
000164	000000G			.WORD	EGD.15			
000166	000000G			.WORD	EMS.15			
000170	013716	000000G	2#:	MOV	CCTLR, (SP)	:		2767
000174	010146			MOV	R1, -(SP)	:	REASON, *	
000176	004737	000000G		JSR	PC, DROP.CTLR			
000202	005726			TST	(SP)+	:		2755
000204	013716	000000G	3#:	MOV	RP.INDX, (SP)	:		2771
000210	004737	000000G		JSR	PC, PUT.RETPKT			
000214	005726			TST	(SP)+	:		2748
000216	022626		4#:	CMP	(SP)+, (SP)+	:		2735
000220	012601			MOV	(SP)+, R1	:		2724
000222	000207			RTS	PC			

; Routine Size: 74 words, Routine Base: \$CODE\$ + 2760
; Maximum stack depth per invocation: 6 words


```
: 2776 1 ROUTINE UNIT_INIT : NOVALUE =
: 2777 1
: 2778 1 !*
: 2779 1 ! THIS ROUTINE IS CALLED FROM DRIVER_INIT FOR EACH CONFIGURED UNIT
: 2780 1 ! (PLATTER) WHICH IS ATTACHED TO A CONTROLLER THAT SURVIVED
: 2781 1 ! INITIALIZATION. ITS PURPOSE IS TO FORMAT AND SEND AN "ONLINE"
: 2782 1 ! MESSAGE, AND TO VERIFY THE RESPONSE.
: 2783 1 !
: 2784 1 ! IMPLICIT INPUTS:
: 2785 1 ! CCTLR - CURRENT CONTROLLER NUMBER
: 2786 1 ! CPLAT - CURRENT PLATTER ADDRESS (MSCP UNIT NUMBER)
: 2787 1 ! L$LUN - CURRENT (DRS) UNIT NUMBER
: 2788 1 ! CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
: 2789 1 ! CUOFF - CURRENT UNIT CST OFFSET
: 2790 1 !-
: 2791 1
: 2792 2 BEGIN
: 2793 2
: 2794 2 LOCAL
: 2795 2 REASON : WORD, ! DROP UNIT REASON
: 2796 2 M_INDEX : WORD; ! MSCP ENVELOPE INDEX
: 2797 2
: 2798 2 M_INDEX = GET_ENV (.CCTLR); ! GET AN MSCP ENVELOPE
: 2799 2 MSCP_ENV [.M_INDEX, PL_ADDR] = .CPLAT; ! SET PLATTER ADDRESS (MSCP UNIT NUMBER)
: 2800 2 MSCP_ENV [.M_INDEX, OPCODE] = OP_ONL; ! OPCODE FOR "ONLINE"
: 2801 2 MSCP_ENV [.M_INDEX, DDPAR] = BIT00; ! SHOW ALL ECC ERRORS IN ERROR LOG MESSAGES
: 2802 2 IF SEND (.M_INDEX) EQLU FAILURE ! ATTEMPT TO SEND; IF CTLR IS OFFLINE
: 2803 2 THEN ! THEN
: 2804 2 PUT_ENV (.M_INDEX) ! RETURN ENVELOPE TO POOL
: 2805 2 ELSE ! OTHERWISE (SEND WAS SUCCESSFUL)
: 2806 3 BEGIN
: 2807 3
: 2808 3 WAIT (); ! WAIT FOR RETPKT RESPONSE
: 2809 3 RP_INDX = OUT_IODQ (); ! GET INDEX OF RETPKT
: 2810 3 RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2); ! CALCULATE RETPKT ADDRESS
: 2811 3 IF .RP_ADDR [CONID] EQLU CID_DRIVER ! IF RETPKT IS FROM "DRIVER"
: 2812 3 THEN ! THEN
: 2813 4 BEGIN
: 2814 4
: 2815 4 REASON = DU_FATAL; ! ASSUME FATAL SA ERROR
: 2816 4 IF .RP_ADDR [MESTYP] EQLU MT_TIMEOUT ! IF COMMAND TIMED OUT
: 2817 4 THEN ! THEN
: 2818 5 BEGIN
: 2819 5
: 2820 5 REASON = DU_INIT; ! SET REASON TO INIT ERROR
: 2821 5 ERRDF (15, EGD_15, EMS_15); ! "MESSAGE RESPONSE TIMEOUT"
: 2822 5
: 2823 4 END;
: 2824 4
: 2825 4 DROP_CTLR (CCTLR, .REASON); ! DROP ALL CONTROLLER'S UNITS
: 2826 4
: 2827 4 END
: 2828 3 ELSE ! OTHERWISE, IF RETPKT IS FROM DISK MSCP
: 2829 4 BEGIN
: 2830 4
: 2831 4 ST_CODE = .RP_ADDR [STSCOD]; ! GET STATUS CODE
: 2832 4 SB_CODE = .RP_ADDR [SUBCOD]; ! GET SUB-CODE
```

```

: 2833 4      IF .ST_CODE NEQU ST_SUC      ! IF STATUS CODE IS NOT SUCCESSFUL
: 2834 4      THEN                          ! THEN
: 2835 5          BEGIN
: 2836 5
: 2837 5          ERRDF (16, EGD_16, EMS_16);      ! "ONLINE FAILED"
: 2838 5          DUR [.L$LUN] = DU_INIT;          ! SET UP REASON FOR DROPPING UNIT
: 2839 5          DODU (.L$LUN);                  ! DROP UNIT
: 2840 5
: 2841 5          END
: 2842 4      ELSE                          ! STATUS CODE IS O.K.
: 2843 5          BEGIN
: 2844 5
: 2845 6          IF ((BIT_TST (RP_ADDR [U_FLGS], UF_WPH)) AND      ! IF UNIT IS WRITE-PROTECTED AND
: 2846 6              (.CST_ADDR [.CUOFF, P_PROT] EQLU UNPROTECTED)) ! USER ALLOWED WRITES
: 2847 5          THEN                          ! THEN
: 2848 6              BEGIN
: 2849 6
: 2850 6              ERRDF (17, EGD_17, EMS_17);      ! "WRITE-PROTECT CONFLICT"
: 2851 6              DUR [.L$LUN] = DU_INIT;          ! SET REASON TO DROP UNIT
: 2852 6              DODU (.L$LUN);                  ! DROP UNIT
: 2853 6
: 2854 6              END
: 2855 5          ELSE                          ! WRITE PROTECT SWITCH IS O.K.
: 2856 6              BEGIN
: 2857 6
: 2858 6              USIZE = .RP_ADDR [USIZ_LO];      ! UNIT SIZE (NO. OF LBN'S)
: 2859 6              CST_ADDR [.CUOFF, P_STAT] = ONLINE; ! MARK UNIT ONLINE
: 2860 6              CPT [.L$LUN] = YES;              ! O.K. TO TEST
: 2861 6              CST_ADDR [U_CNT] = .CST_ADDR [U_CNT] + 1; ! INCR NO. OF TESTABLE UNITS
: 2862 6
: 2863 5              END;
: 2864 5
: 2865 4          END;                          ! IF STATUS CODE = SUCCESS
: 2866 4
: 2867 3      END;                          ! IF RETPKT ORIGINATED AT CONTROLLER
: 2868 3
: 2869 3      PUT_RETPKT (.RP_INDX);
: 2870 3
: 2871 2      END;                          ! IF SEND WAS SUCCESSFUL
: 2872 2
: 2873 1      END;                          ! ROUTINE UNIT_INIT

```

```

000000 004137 000000G      .SBTTL UNIT.INIT INITIALIZATION SUBTEST ROUTINES
                                UNIT.INIT:
000004 013746 000000G      JSR      R1,$SAVE2      ;      2776
000010 004737 000000G      MOV      CCTLR,-(SP)      ;      2798
000014 010001      JSR      PC,GET_ENV
000016 010116      MOV      R0,R1      ; *,M.INDEX
000020 012746 000104      MOV      R1,(SP)      ; M.INDEX,*      2799
000024 004737 000000G      MOV      #104,-(SP)
000030 013760 000000G 000014G  JSR      PC,BL$MUL
000036 112760 000011 000020G  MOV      CPLAT,MSCP_ENV+14(R0)
000044 012760 000001 000044G  MOV      #11,MSCP_ENV+20(R0)      ;      2800
000052 010116      MOV      #1,MSCP_ENV+44(R0)      ;      2801
000054 004737 000000G      MOV      R1,(SP)      ; M.INDEX,*      2802
                                JSR      PC,SEND

```


CZRCDB3
V02.0

CZRCDB0 RC25 DISK EXERCISER
INITIALIZATION SUBTEST ROUTINES

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

SEQ 0249
Page 55
VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRCDB3.SRC;17
(14)

000060	005700			TST	R0			
000062	001004			BNE	1\$			
000064	010116			MOV	R1,(SP)		; M.INDEX,*	
000066	004737	000000G		JSR	PC,PUT.ENV			2804
000072	000565			BR	7\$			
000074	004737	000000G		JSR	PC,WAIT			2802
000100	004737	000000G	1\$:	JSR	PC,OUT.IODQ			2808
000104	010037	000000G		MOV	R0,RP.INDX			2809
000110	010016			MOV	R0,(SP)		; RP.INDX,*	
000112	012746	000060		MOV	#60,-(SP)			2810
000116	004737	000000G		JSR	PC,BL\$MUL			
000122	062700	000000G		ADD	#RETPKT,R0			
000126	010037	000000G		MOV	R0,RP.ADDR			
000132	010001			MOV	R0,R1		; RP.ADDR,*	
000134	126127	000003	000003	CMPB	3(R1),#3			2811
000142	001026			BNE	3\$			
000144	012700	000004		MOV	#4,R0		; *,REASON	2815
000150	116102	000002		MOVB	2(R1),R2			2816
000154	042702	177417		BIC	#177417,R2			
000160	020227	000100		CMP	R2,#100			
000164	001006			BNE	2\$			
000166	012700	000002		MOV	#2,R0		; *,REASON	2820
000172	104455			TRAP	55			2821
000174	0C0017			.WORD	17			
000176	000000G			.WORD	EGD.15			
000200	000000G			.WORD	EMS.15			
000202	012716	000000G		MOV	#CCTLR,(SP)			2825
000206	010046		2\$:	MOV	R0,-(SP)		; REASON,*	
000210	004737	000000G		JSR	PC,DROP.CTLR			
000214	005726			TST	(SP)+			2813
000216	000506			BR	6\$			2811
000220	116137	000016	000000G	MOVB	16(R1),ST.CODE			2831
000226	042737	177740	000000G	BIC	#177740,ST.CODE			
000234	016100	000016		MOV	16(R1),R0			2832
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	013702	000000G		MOV	L\$LUN,R2			2838
000266	005737	000000G		TST	ST.CODE			2833
000272	001412			BEQ	4\$			
000274	104455			TRAP	55			2837
000276	000020			.WORD	20			
000300	000000G			.WORD	EGD.16			
000302	000000G			.WORD	EMS.16			
000304	112762	000002	000000G	MOVB	#2,DUR(R2)			2838
000312	010200			MOV	R2,R0			2839
000314	104451			TRAP	51			
000316	000446			BR	6\$			2833
000320	032761	020000	000022	BIT	#20000,22(R1)			2845
000326	001421			BEQ	5\$			
000330	013700	000000G		MOV	CUOFF,R0			2846
000334	006300			ASL	R0			
000336	063700	000000G		ADD	CST.ADDR,R0			

CZRC03
V02.0

CZRCDB0 RC25 DISK EXERCISER
INITIALIZATION SUBTEST ROUTINES

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC03.SRC;17

SEQ 0250
Page 56
(14)

000342	005710			TST	(R0)		
000344	100012			BPL	5\$		
000346	104455			TRAP	55		
000350	000021			.WORD	21	:	2850
000352	000000G			.WORD	EGD.17		
000354	000000G			.WORD	EMS.17		
000356	112762	000002	000000G	MOVB	#2,DUR(R2)	:	2851
000364	010200			MOV	R2,R0	:	2852
000366	104451			TRAP	51		
000370	000421			BR	6\$:	
000372	016137	000050	000616'	MOV	50(R1),USIZE	:	2845
000400	013700	000000G		MOV	CUOFF,R0	:	2858
000404	006300			ASL	R0	:	2859
000406	063700	000000G		ADD	CST.ADDR,R0		
000412	052710	020000		BIS	#20000,(R0)		
000416	112762	000001	000000G	MOVB	#1,CPT(R2)	:	2860
000424	013700	000000G		MOV	CST.ADDR,R0	:	2861
000430	105260	000005		INCB	5(R0)		
000434	013716	000000G	6\$:	MOV	RP.INDX,(SP)	:	2869
000440	004737	000000G		JSR	PC,PUT.RETPKT		
000444	005726			TST	(SP)+	:	2806
000446	022626		7\$:	CMP	(SP)+,(SP)+	:	2792
000450	000207			RTS	PC	:	2776

; Routine Size: 149 words, Routine Base: \$CODE\$ + 3204
; Maximum stack depth per invocation: 8 words


```
: 2874 1 ROUTINE ACCESS : NOVALUE =
: 2875 1
: 2876 1 !*
: 2877 1 ! THIS ROUTINE IS CALLED BY INIT_TEST TO VERIFY THAT THE CURRENT PLATTER
: 2878 1 ! CAN BE ACCESSED. THIS OBJECTIVE IS ACCOMPLISHED BY FORMATTING AND
: 2879 1 ! SENDING ONE OR TWO MSCP ACCESS COMMANDS TO THE PLATTER, AND CHECKING
: 2880 1 ! THE STATUS FIELD OF THE RESPONSE MESSAGE(S).
: 2881 1 !
: 2882 1 ! IMPLICIT INPUTS:
: 2883 1 ! CCTLR - CURRENT CONTROLLER NUMBER
: 2884 1 ! CPLAT - CURRENT PLATTER ADDRESS (MSCP UNIT NUMBER)
: 2885 1 ! L$LUN - CURRENT (DRS) UNIT NUMBER
: 2886 1 !-
: 2887 1
: 2888 2 BEGIN
: 2889 2
: 2890 2 LOCAL
: 2891 2 M_INDEX : WORD,
: 2892 2 RESULT : WORD,
: 2893 2 LBN : WORD,
: 2894 2 PASS : WORD;
: 2895 2
: 2896 2 RESULT = FAILURE;
: 2897 2 ST_CODE = SB_CODE = 0;
: 2898 2 LBN = ((.USIZE + -1) AND #0'77777') - 1;
: 2899 2 PASS = 1;
: 2900 2 DO
: 2901 3 BEGIN
: 2902 3
: 2903 3 M_INDEX = GET_ENV (.CCTLR);
: 2904 3 MSCP_ENV [.M_INDEX, PL_ADDR] = .CPLAT;
: 2905 3 MSCP_ENV [.M_INDEX, OPCODE] = OP_ACC;
: 2906 3 MSCP_ENV [.M_INDEX, BC_LO] = BLK_SIZE;
: 2907 3 MSCP_ENV [.M_INDEX, LBN_L] = .LBN;
: 2908 3 IF SEND (.M_INDEX) EQLU FAILURE
: 2909 3 THEN
: 2910 4 BEGIN
: 2911 4
: 2912 4 PUT_ENV (.M_INDEX);
: 2913 4 PASS = 2;
: 2914 4
: 2915 4 END
: 2916 3 ELSE
: 2917 4 BEGIN
: 2918 4
: 2919 4 WAIT ();
: 2920 4 RP_INDX = OUT_IODQ ();
: 2921 4 RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2);
: 2922 4 IF .RP_ADDR [CONID] EQLU CID_DRIVER
: 2923 4 THEN
: 2924 4 PASS = 2
: 2925 4 ELSE
: 2926 5 BEGIN
: 2927 5
: 2928 5 ST_CODE = .RP_ADDR [STSCOD];
: 2929 5 SB_CODE = .RP_ADDR [SUBCOD];
: 2930 5 IF .ST_CODE EQLU ST_SUC
```

```
! GUILTY UNTIL PROVEN INNOCENT
! STATUS CODE AND SUB-CODE
! START WITH LAST LBN ON TOP SURFACE
! LOOP PASS COUNT
! LOOP STARTS HERE
```

```
! GET AN MSCP ENVELOPE
! SET PLATTER ADDR (MSCP UNIT NUMBER)
! ACCESS OPCODE
! BYTE COUNT (1 BLOCK)
! LOGICAL BLOCK NUMBER
! ATTEMPT TO SEND; IF CTLR NOT ONLINE
! THEN
```

```
! RETURN ENVELOPE TO POOL
! NO MORE TRIES
```

```
! IF SEND WAS SUCCESSFUL
```

```
! WAIT FOR RESPONSE
! GET RETPKT (RESPONSE) INDEX
! CALCULATE RETPKT ADDRESS
! IF RETPKT CAME FROM "DRIVER"
! THEN
! NO MORE TRIES
! OTHERWISE - CHECK OUT RESPONSE
```

```
! GET STATUS CODE FROM PACKET
! GET SUB-CODE FROM PACKET
! IF STATUS CODE INDICATES SUCCESS
```

```

: 2931 5          THEN          ! THEN
: 2932 6          BEGIN
: 2933 6
: 2934 6          RESULT = SUCCESS;
: 2935 6          PASS = 2;          ! NO NEED TO TRY AGAIN
: 2936 6
: 2937 5          END;
: 2938 5
: 2939 4          END;          ! IF RETPKT ORIGINATED AT CONTROLLER
: 2940 4          PUT_RETPKT (.RP_INDX);
: 2941 4
: 2942 4
: 2943 3          END;          ! IF SEND WAS SUCCESSFUL
: 2944 3
: 2945 3          LBN = .LBN + 1;    ! ADVANCE TO FIRST LBN OF BOTTOM SURFACE
: 2946 3          PASS = .PASS + 1; ! SECOND PASS
: 2947 3
: 2948 3          END          ! END OF PASS LOOP
: 2949 3
: 2950 2          UNTIL .PASS GEQU 3;
: 2951 2
: 2952 2          IF .RESULT EQLU FAILURE
: 2953 2          THEN
: 2954 3          BEGIN
: 2955 3
: 2956 3          SETPRI (PRI05);    ! DELAY POSSIBLE ERROR LOG MESSAGE
: 2957 3          ERRDF (18, EGD_18, EMS_18); ! "ACCESS FAILED"
: 2958 3          SETPRI (PRI00);    ! RESTORE PROCESSOR PRIORITY
: 2959 3          DUR [.L$LUN] = DU_INIT; ! SET REASON TO DROP UNIT
: 2960 3          DODU (.L$LUN);    ! DROP UNIT
: 2961 3
: 2962 2          END;          ! IF ACCESS FAILED
: 2963 2
: 2964 1          END;          ! ROUTINE ACCESS

```

000000	004137	000000G	ACCESS: .SBTTL	ACCESS INITIALIZATION SUBTEST ROUTINES	
000004	005003		JSR	R1,\$SAVE5	2874
000006	005037	000000G	CLR	R3	2896
000012	005037	000000G	CLR	SB.CODE	2897
000016	013700	000616'	CLR	ST.CODE	
000022	006200		MOV	USIZE,R0	2898
000024	010004		ASR	R0	
000026	042704	100000	MOV	R0,R4	
000032	005304		BIC	#100000,R4	
000034	012701	000001	DEC	R4	
000040	013746	000000G	MOV	#1,R1	2899
000044	004737	000000G	MOV	CCTLR,-(SP)	2903
000050	010002		JSR	PC,GET.ENV	
000052	010216		MOV	R0,R2	
000054	012746	000104	MOV	R2,(SP)	
000060	004737	000000G	MOV	#104,-(SP)	2904
000064	013760	000000G	JSR	PC,BL\$MUL	
000072	112760	000020	MOV	CPLAT,MSCP.ENV+14(R0)	
000100	012760	001000	MOV	#20,MSCP.ENV+20(R0)	2905
000106	010460	000044G	MOV	#1000,MSCP.ENV+24(R0)	2906
			MOV	R4,MSCP.ENV+44(R0)	2907

000112	010216			MOV	R2,(SP)				
000114	004737	000000G		JSR	PC,SEND		; M.INDEX,*		2908
000120	005700			TST	R0				
000122	001006			BNE	2\$				
000124	010216			MOV	R2,(SP)		; M.INDEX,*		2912
000126	004737	000000G		JSR	PC,PUT.ENV				
000132	012701	000002		MOV	#2,R1		; *,PASS		2913
000136	000460			BR	5\$				2908
000140	004737	000000G	2\$:	JSR	PC,WAIT				2919
000144	004737	000000G		JSR	PC,OUT.IODQ				2920
000150	010037	000000G		MOV	R0,RP.INDX				
000154	010016			MOV	R0,(SP)		; RP.INDX,*		2921
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				2922
000204	001426			BEQ	3\$				2924
000206	116037	000016	000000G	MOVB	16(R0),ST.CODE				2928
000214	042737	177740	000000G	BIC	#177740,ST.CODE				
000222	016005	000016		MOV	16(R0),R5				2929
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				
000254	001004			BNE	4\$				2930
000256	012703	000001		MOV	#1,R3		; *,RESULT		2934
000262	012701	000002	3\$:	MOV	#2,R1		; *,PASS		2935
000266	013716	000000G	4\$:	MOV	RP.INDX,(SP)				2941
000272	004737	000000G		JSR	PC,PUT.RETPKT				
000276	005726			TST	(SP)+				2917
000300	005204		5\$:	INC	R4		; LBN		2945
000302	005201			INC	R1		; PASS		2946
000304	022626			CMP	(SP)+,(SP)+				2901
000306	020127	000003		CMP	R1,#3		; PASS,*		2950
000312	103652			BLO	1\$				
000314	005703			TST	R3		; RESULT		2952
000316	001017			BNE	6\$				
000320	012700	000240		MOV	#240,R0				2956
000324	104441			TRAP	41				
000326	104455			TRAP	55				2957
000330	000022			.WORD	22				
000332	000000G			.WORD	EGD.18				
000334	000000G			.WORD	EMS.18				
000336	005000			CLR	R0				2958
000340	104441			TRAP	41				
000342	013700	000000G		MOV	L\$LUN,R0				2959
000346	112760	000002	000000G	MOVB	#2,DUR(R0)				
000354	104451			TRAP	51				2960
000356	000207		6\$:	RTS	PC				2874

; Routine Size: 120 words, Routine Base: \$CODE\$ + 3656
; Maximum stack depth per invocation: 10 words

```

: 2965 1 *SBTTL 'DM EXERCISER ROUTINES'
: 2966 1
: 2967 1 ROUTINE DM_EXER : NOVALUE =
: 2968 1
: 2969 1 !+
: 2970 1 !
: 2971 1 ! THIS ROUTINE CONTROLS THE OVERALL OPERATION OF THE DM EXERCISER. THE
: 2972 1 ! DM EXERCISER IS A SUBTEST OF THE HOST EXERCISER. ITS BASIC PURPOSE IS
: 2973 1 ! TO SUPERVISE THE OPERATION OF THE FRONT PANEL TEST WHICH EXECUTES IN
: 2974 1 ! THE DM (DIAGNOSTIC MACHINE).
: 2975 1 !
: 2976 1 ! THE HOST INITIATES THE FRONT PANEL TEST BY DOWN-LINE LOADING THE CROM
: 2977 1 ! PRIMER PROGRAM FROM DM_INIT. THIS PROGRAM HAS THE JOB OF LOADING THE
: 2978 1 ! FRONT PANEL TEST INTO CONTROLLER MEMORY AND PASSING EXECUTION CONTROL
: 2979 1 ! TO IT. WHEN AND IF THIS SUCCEEDS, THE HOST SENDS DOWN THE ADDRESS OF
: 2980 1 ! THE DM EXERCISER / FRONT PANEL TEST COMMUNICATION AREA (DM_COMM). THIS
: 2981 1 ! BLOCK OF MEMORY HOLDS THE UNIT NUMBERS OF DISKS WHICH ARE TO BE
: 2982 1 ! DM-EXERCISED, AS WELL AS UNIT STATISTICS (LOADED BY THE FRONT PANEL
: 2983 1 ! TEST AND READ BY THE HOST).
: 2984 1 !
: 2985 1 ! THIS ROUTINE DISPATCHES PROCESSING TO OTHER ROUTINES ON A REGULAR
: 2986 1 ! BASIS UNTIL ALL UNITS HAVE COMPLETED THE REQUIRED NUMBER OF BYTES
: 2987 1 ! TRANSFERRED. THESE OTHER ROUTINES INCLUDE THE PROCESSING OF DUP END
: 2988 1 ! MESSAGES (DM_RETPKT), UPDATING HOST STATISTICS FROM DM_COMM (DM_TALLY),
: 2989 1 ! AND MAINTAINING COMMAND TIMERS (DRV_TIMCHK).
: 2990 1 !-
: 2991 2 BEGIN
: 2992 2
: 2993 3 IF MANUAL ! IF ATTENDED
: 2994 2 THEN ! THEN
: 2995 2 PRINTF (MSG_06); ! "DM EXERCISER SUBTEST START"
: 2996 2 DM_INIT (); ! INIT DM_EXER DATA, SEND CROM PRIMER
: 2997 2
: 2998 2 DO ! "EXECUTIVE" PROCESSING LOOP
: 2999 3 BEGIN
: 3000 3
: 3001 3 BREAK; ! BREAK IN CASE USER TYPED <CTRL-C>
: 3002 3 DM_RETPKT (); ! PROCESS ANY DUP END MESSAGES OR "DRIVER" ERRORS
: 3003 3 IF .T_FLAG EQLU TRUE ! IF ONE SECOND HAS ELAPSED
: 3004 3 THEN ! THEN
: 3005 4 BEGIN
: 3006 4
: 3007 4 INCR CTLR FROM 0 TO (.CTLR_CNT - 1) DO ! FOR EACH CONFIGURED CONTROLLER
: 3008 5 BEGIN
: 3009 5
: 3010 5 SET_CPAR (.CTLR); ! SET UP CONTROLLER-RELATED DATA ITEMS
: 3011 6 IF ((.CST_ADDR [STATE] EQLU ONLINE) AND ! IF CONTROLLER IS ONLINE AND
: 3012 6 (.CST_ADDR [U_CNT] NEQU 0)) ! THERE IS AT LEAST 1 UNIT UNDER TEST
: 3013 5 THEN ! THEN
: 3014 6 BEGIN
: 3015 6
: 3016 6 IF DM_ACC () ! IF ACCESS GAINED TO DM_COMM AREA
: 3017 6 THEN ! THEN
: 3018 6 DM_TALLY (); ! ADD DM_COMM STATS TO HOST TOTALS
: 3019 6 DM_TIME (); ! CHECK FPT SANITY TIMER FOR CONTROLLER
: 3020 6
: 3021 5 END; ! IF CTLR IS ONLINE AND AT LEAST ONE UNIT

```



```

: 3022 5
: 3023 4           END;
: 3024 4
: 3025 4           DRV_TIMCHK ();
: 3026 4
: 3027 3           END;
: 3028 3
: 3029 3           END
: 3030 2           UNTIL DM_QUIT ();
: 3031 2
: 3032 1           END;

```

```

! CONTROLLER LOOP
! CHECK FOR COMMAND TIMEOUTS
! IF ONE SECOND HAS ELAPSED
! END OF "EXECUTIVE" LOOP
! UNTIL ALL UNITS DONE
! ROUTINE DM_EXER

```

```

000000 004137 000000G           .SBTTL  DM.EXER DM EXERCISER ROUTINES
000004 104450           DM.EXER:JSR  R1,$SAVE2           ;           2967
000006 103007           TRAP  50           ;           2993
000010 012746 000000G           BHIS  1$
000014 012746 000001           MOV   #MSG.06,-(SP)           ;           2995
000020 010600           MOV   #1,-(SP)
000022 104417           MOV   SP,R0           ; SP,*
000024 022626           TRAP  17
000026 004737 000000V           CMP   (SP)+,(SP)+
000032 104422           1$: JSR  PC,DM.INIT           ;           2996
000034 004737 000000V           2$: TRAP  22           ;           2999
000040 123727 000000G 000001           JSR  PC,DM.RETPKT           ;           3002
000046 001035           CMPB  T.FLAG,#1           ;           3003
000050 013702 000000G           BNE  7$
000054 005001           MOV   CTLR.CNT,R2           ;           3007
000056 000425           CLR  R1           ; CTLR
000060 010146           BR   6$
000062 004737 000000G           3$: MOV  R1,-(SP)           ; CTLR,*           3010
000066 013700 000000G           JSR  PC,SET.CPAR
000072 005760 000002           MOV  CST.ADDR,R0           ;           3011
000076 100013           TST  2(R0)
000100 105760 000005           BPL  5$
000104 001410           TSTB 5(R0)           ;           3012
000106 004737 000000V           BEQ  5$
000112 006000           JSR  PC,DM.ACC           ;           3016
000114 103002           ROR  R0
000116 004737 000000V           BCC  4$
000122 004737 000000V           4$: JSR  PC,DM.TALLY           ;           3018
000126 005726           5$: JSR  PC,DM.TIME           ;           3019
000130 005201           TST  (SP)+           ;           3008
000132 020102           INC  R1           ; CTLR           3007
000134 002751           6$: CMP  R1,R2           ; CTLR,*
000136 004737 000000V           BLT  3$
000142 004737 000000V           7$: JSR  PC,DRV.TIMCHK           ;           3025
000146 006000           JSR  PC,DM.QUIT           ;           3030
000150 103330           ROR  R0
000152 000207           BCC  2$
           RTS  PC           ;           2967

```

```

; Routine Size: 54 words, Routine Base: $CODE$ + 4236
; Maximum stack depth per invocation: 7 words

```

```

: 3033 1 ROUTINE DM_INIT : NOVALUE =
: 3034 1
: 3035 1 !+
: 3036 1 ! THIS ROUTINE IS CALLED BY DM_EXER WHEN THE DM EXERCISER IS FIRST
: 3037 1 ! STARTED. ITS PURPOSE IS (A) TO INITIALIZE DM EXERCISER DATA, AND (B)
: 3038 1 ! TO START THE FRONT PANEL TEST EXECUTING IN THE DM.
: 3039 1 !
: 3040 1 ! THE MOST SIGNIFICANT DATA INITIALIZATION OCCURS IN THE COMMUNICATION
: 3041 1 ! AREA (DM_COMM) BETWEEN THE HOST AND THE FRONT PANEL TEST. PLATTER
: 3042 1 ! ADDRESSES TO BE EXERCISED UNDER THE FPT ARE LOADED INTO THIS AREA,
: 3043 1 ! AND ALL STATISTICS AND ACCESS WORDS ARE CLEARED TO ZERO.
: 3044 1 !
: 3045 1 ! TO START THE FPT EXECUTING, THIS PROGRAM CAUSES THE CROM PRIMER TO
: 3046 1 ! BE DOWN-LINE LOADED INTO THE DM VIA DUP'S "EXECUTE SUPPLIED PROGRAM"
: 3047 1 ! MESSAGE. THE CROM PRIMER THEN LOADS THE FRONT PANEL TEST INTO RAM AND
: 3048 1 ! STARTS ITS EXECUTION. THE SUCCESS / FAILURE OF THIS OPERATION IS
: 3049 1 ! COMMUNICATED BACK TO THE HOST THROUGH AN END MESSAGE.
: 3050 1 !-
: 3051 1
: 3052 2 BEGIN
: 3053 2
: 3054 2 LOCAL
: 3055 2 MX : WORD; ! MSCP ENVELOPE INDEX
: 3056 2
: 3057 2 INCR CTLR FROM 0 TO (.CTLR_CNT - 1) DO ! FOR EACH CONFIGURED CONTROLLER
: 3058 3 BEGIN
: 3059 3
: 3060 3 SET_CPAR (.CTLR); ! SET UP CONTROLLER-RELATED PARAMETERS
: 3061 3 IF .CST_ADDR [STATE] EQLU ONLINE ! IF CONTROLLER IS ONLINE
: 3062 3 THEN ! THEN
: 3063 4 BEGIN
: 3064 4
: 3065 4 INCR COUNT FROM 8 TO ((DMC_LEN * 2) - 2) BY 2 DO ! CLEAR CTLR'S DM_COMM STATS
: 3066 4 (.DMC_ADDR + .COUNT) = 0;
: 3067 4 INCR OFFSET FROM 0 TO 3 DO ! FOR EACH UNIT UNDER CTLR
: 3068 5 BEGIN
: 3069 5
: 3070 5 IF .CST_ADDR [.OFFSET + OF_UN, P_STAT] EQLU ONLINE ! IF UNIT IS ONLINE
: 3071 5 THEN ! THEN LOAD PLATTER ADDRESS
: 3072 5 DMC_ADDR [.OFFSET, ALLBIT] = .CST_ADDR [.OFFSET + OF_UN, P_ADDR]
: 3073 5 ELSE ! OTHERWISE
: 3074 5 DMC_ADDR [.OFFSET, ALLBIT] = -1; ! DO NOT EXERCISE
: 3075 5
: 3076 4 END; ! END UNIT LOOP
: 3077 4
: 3078 4 IF .ENTRY_REASON NEQU NEW_PASS ! IF START, RESTART, CONT, OR PWR FAIL
: 3079 4 THEN ! THEN
: 3080 5 BEGIN
: 3081 5
: 3082 5 DM_TIMR [.CTLR] = 0; ! INIT FPT SANITY TIMER
: 3083 5 MX = GET_ENV (.CTLR); ! GET AN MSCP ENVELOPE
: 3084 5 MSCP_ENV [.MX, CONNID] = CID_DUP; ! CONNECTION ID (DUP)
: 3085 5 MSCP_ENV [.MX, OPCODE] = OP_ESP; ! OPCODE = EXECUTE SUPPLIED PROGRAM
: 3086 5 MSCP_ENV [.MX, DBC_LO] = .CROMP [0]; ! BYTE COUNT
: 3087 5 MSCP_ENV [.MX, DBUF_0] = CROMP; ! BUFF DESC OF CROM PRIMER
: 3088 5 MSCP_ENV [.MX, OBUF_0] = CROMP + .CROMP [0]; ! OVERLAY BUFF DESC
: 3089 5 IF SEND (.MX) EQLU FAILURE ! ATTEMPT TO SEND. IF FAILURE
```



```

; 3090 5          THEN
; 3091 5          PUT_ENV (.MX);
; 3092 5
; 3093 4          END;
; 3094 4
; 3095 3          END;
; 3096 3
; 3097 2          END;
; 3098 2
; 3099 3          IF ((.ENTRY_REASON NEQU CONT) AND
; 3100 3            (.ENTRY_REASON NEQU PWR_FAIL))
; 3101 2          THEN
; 3102 2            INCR COUNT FROM 0 TO ((MAX_UNITS * TALLY_LEN) - 1) DO
; 3103 2              TALLY [.COUNT] = 0;
; 3104 2
; 3105 2          T_FLAG = FALSE;
; 3106 2
; 3107 1          END;

```

```

! THEN
! RETURN ENVELOPE TO POOL
! IF START, RESTART, CONT, OR PWR FAIL
! END - IF CTLR IS ONLINE
! END CONTROLLER LOOP
! IF START, RESTART, OR NEW PASS
! THEN
! CLEAR ALL STATS
! CLEAR ONE SECOND TIMING FLAG
! ROUTINE DM_INIT

```

			.SBTTL	DM.INIT DM EXERCISER ROUTINES	
000000	004137	000000G	DM.INIT:JSR	R1,\$SAVE5	3033
000004	013705	000000G	MOV	CTLR.CNT,R5	3057
000010	0C5003		CLR	R3	; CTLR
000012	000531		BR	8\$	
000014	010346		1\$:MOV	R3,-(SP)	; CTLR,*
000016	004737	000000G	JSR	PC,SET.CPAR	3060
000022	013700	000000G	MOV	CST.ADDR,R0	
000026	005760	000002	TST	2(R0)	3061
000032	100117		BPL	7\$	
000034	012700	000010	MOV	#10,R0	; *,COUNT
000040	010001		2\$:MOV	R0,R1	; COUNT,*
000042	063701	000000G	ADD	DMC.ADDR,R1	
000046	005011		CLR	(R1)	
000050	062700	000002	ADD	#2,R0	; *,COUNT
000054	020027	000062	CMP	R0,#62	; COUNT,*
000060	003767		BLE	2\$	
000062	005002		CLR	R2	; OFFSET
000064	010201		3\$:MOV	R2,R1	; OFFSET,*
000066	006301		ASL	R1	
000070	063701	000000G	ADD	DMC.ADDR,R1	
000074	010200		MOV	R2,R0	; OFFSET,*
000076	006300		ASL	R0	
000100	063700	000000G	ADD	CST.ADDR,R0	
000104	032760	020000 000006	BIT	#20000,6(R0)	
000112	001405		BEQ	4\$	
000114	116011	000006	MOVB	6(R0),(R1)	
000120	105061	000001	CLRB	1(R1)	
000124	000402		BR	5\$	
000126	012711	177777	4\$:MOV	#-1,(R1)	
000132	005202		5\$:INC	R2	; OFFSET
000134	020227	000003	CMP	R2,#3	; OFFSET,*
000140	003751		BLE	3\$	
000142	123727	000000G 000005	CMPB	ENTRY.REASON,#5	
000150	001450		BEQ	7\$	
000152	010300		MOV	R3,R0	; CTLR,*
000154	006300		ASL	R0	

CZRC03
V02.0

CZRCDB0 RC25 DISK EXERCISER
DM EXERCISER ROUTINES

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC03.SRC;17

SEQ 0258
Page 65
(17)

000156	005060	000620'		CLR	DM.TIMR(R0)			
000162	010316			MOV	R3,(SP)			
000164	004737	000000G		JSR	PC.GET.ENV		: CTRL,*	3083
000170	010004			MOV	R0,R4		: *,MX	
000172	010416			MOV	R4,(SP)		: MX,*	3084
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)			3085
000220	013760	001716'	000024G	MOV	CROMP,MSCP.ENV+24(R0)			3086
000226	012760	001716'	000030G	MOV	#CROMP,MSCP.ENV+30(R0)			3087
000234	016060	000024G	000044G	MOV	MSCP.ENV+24(R0),MSCP.ENV+44(R0)			3088
000242	062760	001716'	000044G	ADD	#CROMP,MSCP.ENV+44(R0)			
000250	010416			MOV	R4,(SP)		: MX,*	3089
000252	004737	000000G		JSR	PC.SEND			
000256	005700			TST	R0			
000260	001003			BNE	6#			
000262	010416			MOV	R4,(SP)		: MX,*	3091
000264	004737	000000G		JSR	PC.PUT.ENV			
000270	005726			6#:	TST	(SP)+		3080
000272	005726			7#:	TST	(SP)+		3058
000274	005203			INC	R3		: CTRL	3057
000276	020305			8#:	CMP	R3,R5	: CTRL,*	
000300	0C2645			BLT	1#			
000302	123727	000000G	000003	CMPB	ENTRY.REASON,#3			3099
000310	001414			BEQ	10#			
000312	123727	000000G	000004	CMPB	ENTRY.REASON,#4			3100
000320	001410			BEQ	10#			
000322	005000			CLR	R0		: COUNT	3102
000324	005060	000000G		9#:	CLR	TALLY(R0)	: *(COUNT)	3103
000330	062700	000002		ADD	#2,R0		: *,COUNT	3102
000334	020027	001376		CMP	R0,#1376		: COUNT,*	
000340	003771			BLE	9#			
000342	105037	000000G		10#:	CLRB	T.FLAG		3105
000346	000207			RTS	PC			3033

; Routine Size: 116 words, Routine Base: \$CODE\$ + 4412
; Maximum stack depth per invocation: 9 words


```

; 3108 1  ROUTINE DM_QUIT =
; 3109 1
; 3110 1  !*
; 3111 1  !
; 3112 1  !   THIS ROUTINE IS CALLED BY THE DM_EXERCISER "EXECUTIVE" (DM_EXER) TO
; 3113 1  !   DETERMINE WHETHER OR NOT THE DM_EXERCISER SUBTEST SHOULD BE TERMINATED.
; 3114 1  !   ITS PURPOSE IS TO EXAMINE THE STATUS OF THE CURRENT PASS TESTING
; 3115 1  !   VECTOR (CPT). IF EACH UNIT'S VALUE IN THIS TABLE IS "NO" (INDICATING
; 3116 1  !   THAT CURRENT PASS TESTING ON THE UNIT IS COMPLETE), THEN THIS ROUTINE
; 3117 1  !   RETURNS A VALUE OF "TRUE" TO ITS CALLER. IF ANY ONE UNIT HAS A VALUE
; 3118 1  !   OF "YES", THEN "FALSE" IS RETURNED.
; 3119 1  !
; 3120 1  !   IMPLICIT INPUTS:
; 3121 1  !   L$UNIT - NUMBER OF UNITS CONFIGURED FOR TEST
; 3122 1  !-
; 3123 2  BEGIN
; 3124 2
; 3125 2  INCR UNIT FROM 0 TO (.L$UNIT - 1) DO           ! FOR EACH CONFIGURED UNIT
; 3126 2  IF .CPT [.UNIT] EQLU YES                       ! IF UNIT STILL UNDER TEST
; 3127 2  THEN                                           ! THEN
; 3128 2  RETURN FALSE;                                ! TEST NOT FINISHED
; 3129 2  RETURN TRUE;                                  ! ALL UNITS DONE
; 3130 2
; 3131 1  END;                                         ! ROUTINE DM_QUIT

```

		.SBTTL	DM.QUIT	DM EXERCISER ROUTINES		
000000	005000		DM.QUIT:CLR	RO	:	UNIT
000002	000405		BR	2\$:	3125
000004	126027	000000G 000001	1\$: CMPB	CPT(RO),#1	:	*(UNIT),*
000012	001407		BEQ	3\$:	3126
000014	005200		INC	RO	:	3128
000016	020037	000000G	2\$: CMP	RO,L\$UNIT	:	UNIT,*
000022	002770		BLT	1\$:	3125
000024	012700	000001	MOV	#1,RO	:	
000030	000207		RTS	PC	:	3123
000032	005000		3\$: CLR	RO	:	
000034	000207		RTS	PC	:	3108

```

; Routine Size: 15 words,      Routine Base: $CODE$ + 4762
; Maximum stack depth per invocation: 0 words

```

```

: 3132 1 ROUTINE DM_RETPKT : NOVALUE =
: 3133 1
: 3134 1 !!
: 3135 1 !! THIS ROUTINE IS CALLED BY DM_EXER TO CHECK FOR AND PROCESS ANY RETURN
: 3136 1 !! PACKETS THAT HAVE BEEN "SENT" BY THE "DRIVER" PORTION OF THE PROGRAM.
: 3137 1 !! THE I/O DONE QUEUE (IODQ) ACTS AS THE LINK BETWEEN THE TWO PROGRAM
: 3138 1 !! PARTS; IT HOLDS INDECES OF RETURN PACKETS WHICH REQUIRE PROCESSING.
: 3139 1 !!
: 3140 1 !! UNDER THE DM EXERCISER SUBTEST, RETURN PACKETS ORIGINATE FROM TWO
: 3141 1 !! SOURCES:
: 3142 1 !! 1. DUP - DESCRIBING AN END MESSAGE RECEIVED FROM THE RC25 IN
: 3143 1 !! RESPONSE TO A HOST COMMAND
: 3144 1 !! 2. THE PROGRAM "DRIVER" - DESCRIBING A CONTROLLER ERROR OR
: 3145 1 !! COMMAND TIMEOUT.
: 3146 1 !!-
: 3147 1
: 3148 2 BEGIN
: 3149 2
: 3150 2 WHILE .IODQ_IN NEQU .IODQ_OUT DO ! DO UNTIL I/O DONE QUEUE IS EMPTY
: 3151 3 BEGIN
: 3152 3
: 3153 3 RP_INDX = OUT_IODQ (); ! GET INDEX OF NEXT RETPKT AND ADVANCE OUT POINTER
: 3154 3 RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2); ! CALCULATE RETPKT ADDRESS
: 3155 3 SET_CPAR (.RP_ADDR [CTLR]); ! SET UP CURRENT CONTROLLER PARAMETERS
: 3156 3 L:LUN = .CST_ADDR [OF_UN, P_UNIT]; ! GET UNIT NUMBER OF FIRST UNIT
: 3157 3 IF .RP_ADDR [CONID] EQLU CID_DRIVER ! IF RETPKT IS FROM PROGRAM "DRIVER"
: 3158 3 THEN ! THEN
: 3159 4 BEGIN
: 3160 4
: 3161 4 IF .RP_ADDR [MESTYP] EQLU MT_TIMEOUT ! IF COMMAND TIMEOUT
: 3162 4 THEN ! THEN
: 3163 4 ERRDF (15, EGD_15, EMS_15); ! "MESSAGE RESPONSE TIMEOUT"
: 3164 4 DROP_CTLR (.CCTLR, DU_FATAL); ! DROP CONTROLLER'S UNITS
: 3165 4 CST_ADDR [STATE] = OFFLINE; ! MARK CONTROLLER OFFLINE
: 3166 4
: 3167 4 END
: 3168 3 ELSE ! OTHERWISE (RETPKT NOT FROM "DRIVER")
: 3169 4 BEGIN
: 3170 4
: 3171 4 IF .RP_ADDR [CONID] EQLU CID_DUP ! IF RETPKT IS FROM DUP
: 3172 4 THEN ! THEN
: 3173 4 DM_MSG (); ! PROCESS THE END MESSAGE
: 3174 4
: 3175 3 END;
: 3176 3
: 3177 3 PUT_RETPKT (.RP_INDX); ! RETURN PACKET TO POOL
: 3178 3
: 3179 2 END; ! END - UNTIL I/O DONE QUEUE IS EMPTY
: 3180 2
: 3181 1 END; ! ROUTINE DM_RETPKT

```

```

000000 010146 .SBTTL DM.RETPKT DM EXERCISER ROUTINES
0000C2 023737 000000G 000000G 1$: MOV R1, -(SP) ; 3132
000010 001506 BEQ IODQ.IN, IODQ.OUT ; 3150

```


CZRC03
V02.0

CZRCDB0 RC25 DISK EXERCISER
DM EXERCISER ROUTINES

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

SEQ 0261
Page 68
VAX-11 Bliss-16 V4.0-579
USER\$1:[AZTEC.CZRCDB]CZRC03.SRC;17
(19)

000012	004737	000000G		JSR	PC,OUT.IODQ				
000016	010037	000000G		MOV	RO,RP.INDX	:			3153
000022	010046			MOV	RO,-(SP)				
000024	012746	000060		MOV	#60,-(SP)	:	RP.INDX,*		3154
000030	004737	000000G		JSR	PC,BL\$MUL				
000034	062700	000000G		ADD	#RETPKT,RO				
000040	010037	000000G		MOV	RO,RP.ADDR				
000044	116016	000002		MOVB	2(RO),(SP)	:			
000050	042716	177760		BIC	#177760,(SP)				3155
000054	004737	000000G		JSR	PC,SET.CPAR				
000060	013700	000000G		MOV	CST.ADDR,RO	:			
000064	016001	000006		MOV	6(RO),R1				3156
000070	000301			SWAB	R1				
000072	042701	177740		BIC	#177740,R1				
000076	010137	000000G		MOV	R1,L\$LUN				
000102	013700	000000G		MOV	RP.ADDR,RO	:			
000106	126027	000003	000003	CMPB	3(RO),#3				3157
000114	001030			BNE	3\$				
000116	116000	000002		MOVB	2(RO),RO	:			
000122	042700	177417		BIC	#177417,RO				3161
000126	020027	000100		CMP	RO,#100				
000132	001004			BNE	2\$				
000134	104455			TRAP	55	:			
000136	0C0017			.WORD	17				3163
000140	000000G			.WORD	EGD.15				
000142	000000G			.WORD	EMS.15				
000144	013716	000000G	2\$:	MOV	CCTLR,(SP)	:			
000150	012746	000004		MOV	#4,-(SP)				3164
000154	004737	000000G		JSR	PC,DROP.CTLR				
000160	013700	000000G		MOV	CST.ADDR,RO	:			
000164	042760	100000	000002	BIC	#100000,2(RO)				3165
000172	005726			TST	(SP)+	:			
000174	000406			BR	4\$:			3159
000176	126027	000003	000002	CMPB	3(RO),#2	:			3157
000204	0C1002			BNE	4\$:			3171
000206	004737	000000V		JSR	PC,DM.MSG	:			
000212	013716	000000G	4\$:	MOV	RP.INDX,(SP)	:			3173
000216	004737	000000G		JSR	PC,PUT.RETPKT	:			3177
000222	022626			CMP	(SP)+,(SP)+	:			
000224	000666			BR	1\$:			3151
000226	012601		5\$:	MOV	(SP)+,R1	:			3150
000230	000207			RTS	PC	:			3132

: Routine Size: 77 words, Routine Base: \$CODE\$ + 5020
: Maximum stack depth per invocation: 5 words


```

: 3239 3      [OP_SND] : DM_TIMR [.CCTLR] = TO_DM;      ! IF "SEND DATA", START FPT SANITY TIMER
: 3240 3
: 3241 3      TES;
: 3242 3
: 3243 2      END;      ! END - IF STATUS CODE = SUCCESS
: 3244 2
: 3245 1      END;      ! ROUTINE DM_MSG

```

```

000000 010146      .SBTTL  DM.MSG DM EXERCISER ROUTINES
000002 013700 000000G DM.MSG: MOV R1,-(SP) ; 3182
000006 016037 000016 000000G MOV RP,ADDR,R0 ; 3211
000014 001413 BEQ 1$ ;
000016 104455 TRAP 55 ; 3215
000020 000025 .WORD 25
000022 000000G .WORD EGD.21
000024 000000G .WORD EMS.21
000026 013746 000000G MOV CCTLR,-(SP) ; 3216
000032 012746 000004 MOV #4,-(SP)
000036 004737 000000G JSR PC,DROP.CTLR
000042 000447 BR 2$ ;
000044 013700 000000G 1$: MOV RP,ADDR,R0 ; 3213
000050 116000 000014 MOVB 14(R0),R0 ; 3222
000054 042700 177600 BIC #177600,R0 ; *,DUP.OP
000060 020027 000002 CMP R0,#2 ; *,DUP.OP
000064 001040 BNE 3$ ; DUP.OP,* 3226
000066 013746 000000G MOV CCTLR,-(SP) ; 3228
000072 004737 000000G JSR PC,GET.ENV
000076 010001 MOV R0,R1 ; *,MX
000100 010116 MOV R1,(SP) ; MX,* 3229
000102 012746 000104 MOV #104,-(SP)
000106 004737 000000G JSR PC,BL$MUL
000112 112760 000002 000007G MOVB #2,MSCP.ENV+7(R0)
000120 112760 000004 000020G MOVB #4,MSCP.ENV+20(R0) ; 3230
000126 012760 000002 000024G MOV #2,MSCP.ENV+24(R0) ; 3231
000134 012760 000000G 000030G MOV #DMC.ADDR,MSCP.ENV+30(R0) ; 3232
000142 010116 MOV R1,(SP) ; MX,* 3233
000144 004737 000000G JSR PC,SEND
000150 005700 TST R0
000152 001003 BNE 2$ ;
000154 010116 MOV R1,(SP) ; MX,* 3235
000156 004737 000000G JSR PC,PUT.ENV
000162 022626 2$: CMP (SP)+,(SP)+ ; 3226
000164 000411 BR 4$ ; 3223
000166 020027 000004 3$: CMP R0,#4 ; DUP.OP,* 3239
000172 001006 BNE 4$
000174 013700 000000G MOV CCTLR,R0
000200 006300 ASL R0
000202 012760 000550 000620' MOV #550,DM.TIMR(R0)
000210 012601 4$: MOV (SP)+,R1 ; 3182
000212 000207 RTS PC

```

```

; Routine Size: 70 words, Routine Base: $CODE$ + 5252
; Maximum stack depth per invocation: 4 words

```

```

: 3246 1 ROUTINE DM_ACC =
: 3247 1
: 3248 1 !+
: 3249 1 ! THIS ROUTINE IS CALLED BY DM_EXER IN ORDER TO ATTEMPT TO GAIN ACCESS
: 3250 1 ! TO THE CURRENT CONTROLLER'S DM_COMM AREA. IF ACCESS IS PERMITTED (I.E.,
: 3251 1 ! IF THE FRONT PANEL TEST (FPT) IS NOT CURRENTLY UPDATING DM_COMM
: 3252 1 ! STATISTICS), THEN THIS ROUTINE RETURNS A VALUE OF "SUCCESS". OTHERWISE,
: 3253 1 ! ACCESS IS DENIED AT THIS TIME, AND "FAILURE" IS RETURNED.
: 3254 1 !
: 3255 1 ! IMPLICIT INPUTS:
: 3256 1 ! DMC_ADDR - ADDRESS OF CURRENT CONTROLLER'S DM_COMM AREA
: 3257 1 !
: 3258 1 ! OUTPUTS:
: 3259 1 ! "SUCCESS" - ACCESS TO DM_COMM IS PERMITTED
: 3260 1 ! "FAILURE" - ACCESS TO DM_COMM IS DENIED
: 3261 1 !-
: 3262 1
: 3263 2 BEGIN
: 3264 2
: 3265 2 LOCAL
: 3266 2 RESULT : WORD; ! RESULT OF ACCESS REQUEST
: 3267 2
: 3268 2 RESULT = SUCCESS; ! ASSUME ACCESS WILL BE GAINED
: 3269 2 DMC_ADDR [HOST_ACC] = ALL_ONES; ! SET HOST ACCESS WORD
: 3270 2 IF .DMC_ADDR [FPT_ACC] NEQU 0 ! IF FPT HAS ACCESS NOW
: 3271 2 THEN ! THEN
: 3272 3 BEGIN
: 3273 3
: 3274 3 DMC_ADDR [HOST_ACC] = 0; ! WITHDRAW ACCESS REQUEST
: 3275 3 RESULT = FAILURE; ! CHANGE RESULT TO FAILURE
: 3276 3
: 3277 2 END;
: 3278 2
: 3279 2 RETURN .RESULT;
: 3280 2
: 3281 1 END; ! ROUTINE DM_ACC

```

000000	004137	000000G	.SBTTL	DM.ACC DM EXERCISER ROUTINES		
000004	012702	000001	DM.ACC: JSR	R1,\$SAVE2	:	3246
000010	013700	000000G	MOV	#1,R2	: *,RESULT	3268
000014	012760	177777 000062	MOV	DMC.ADDR,R0	:	3269
000022	010001		MOV	#-1,62(R0)	:	
000024	005761	000060	MOV	R0,R1	: DMC.ADDR,*	3270
000030	001403		TST	60(R1)		
000032	005060	000062	BEQ	1\$		
000036	005002		CLR	62(R0)	:	3274
000040	010200		CLR	R2	: RESULT	3275
000042	000207		1\$: MOV	R2,R0	: RESULT,*	3263
			RTS	PC	:	3246

```

: Routine Size: 18 words, Routine Base: $CODE$ + 5466
: Maximum stack depth per invocation: 4 words

```



```
; 3282 1 ROUTINE DM_TALLY : NOVALUE =
; 3283 1
; 3284 1 !+
; 3285 1 ! THIS ROUTINE IS CALLED FROM DM_EXER AFTER ACCESS HAS BEEN GAINED TO
; 3286 1 ! THE CURRENT CONTROLLER'S DM_COMM AREA. ITS PURPOSE IS TO EXTRACT THE
; 3287 1 ! STATISTICAL VALUES STORED THERE BY THE FRONT PANEL TEST (EXECUTING IN
; 3288 1 ! THE DM), AND TO ADD THESE VALUES TO THE HOST-KEPT STATS. THE STATS IN
; 3289 1 ! DM_COMM ARE THEN RESET TO 0.
; 3290 1 !
; 3291 1 ! AFTER UPDATING THE HOST STATS, THE NUMBER OF HARD ERRORS AND THE NUMBER
; 3292 1 ! OF MBYTES TRANSFERRED THUS FAR FOR EACH UNIT IS CHECKED AGAINST THE
; 3293 1 ! LIMITS SPECIFIED IN THE SW P-TABLE. IF THE HARD ERROR LIMIT HAS BEEN
; 3294 1 ! REACHED, THEN THE UNIT IS DROPPED FROM ALL TESTING. IF THE TRANSFER
; 3295 1 ! LIMIT HAS BEEN REACHED, THEN THE UNIT IS SIMPLY REMOVED FROM THE
; 3296 1 ! CURRENT PASS.
; 3297 1 !
; 3298 1 ! IMPLICIT INPUTS:
; 3299 1 ! DMC_ADDR - ADDRESS OF THE CURRENT CONTROLLER'S DM_COMM AREA
; 3300 1 !-
; 3301 1
; 3302 2 BEGIN
; 3303 2
; 3304 2 LOCAL
; 3305 2 ST_PTR, ! STATISTICS POINTER IN DM_COMM
; 3306 2 NBXFR : WORD; ! NUMBER OF BLOCKS TRANSFERRED
; 3307 2
; 3308 2 ST_PTR = .DMC_ADDR + 8; ! POINTS TO STATS FOR FIRST UNIT
; 3309 2 INCR OFFSET FROM 0 TO 3 DO ! FOR EACH UNIT
; 3310 3 BEGIN
; 3311 3
; 3312 3 IF .DMC_ADDR [.OFFSET, 0, 16, 1] GEQ 0 ! IF UNIT IS BEING DM-EXERCISED
; 3313 3 THEN ! THEN
; 3314 4 BEGIN
; 3315 4
; 3316 4 SET_UPAR (.OFFSET + OF_UN); ! SET UP UNIT-RELATED PARAMETERS
; 3317 4
; 3318 4 UPD_IOC (T_ADDR [SEEK_LO], ..ST_PTR); ! UPDATE NO. OF SEEKS
; 3319 4 .ST_PTR = 0; ! RESET STAT TO 0
; 3320 4 ST_PTR = .ST_PTR + 2; ! ADVANCE DM_COMM POINTER
; 3321 4
; 3322 4 NBXFR = ..ST_PTR; ! NUMBER OF BLOCKS READ (1 BLOCK PER READ)
; 3323 4 UPD_IOC (T_ADDR [READ_LO], ..ST_PTR); ! UPDATE NO. OF READS
; 3324 4 UPD_DMBC (T_ADDR [BR_LO], ..ST_PTR); ! UPDATE NO. OF BYTES READ, CHECK FOR OVERFLOW
; 3325 4 .ST_PTR = 0; ! RESET STAT TO 0
; 3326 4 ST_PTR = .ST_PTR + 2; ! ADVANCE DM_COMM POINTER
; 3327 4
; 3328 4 NBXFR = .NBXFR + ..ST_PTR; ! ADD NUMBER OF BLOCKS WRITTEN (1 BLOCK PER WRITE)
; 3329 4 UPD_IOC (T_ADDR [WRIT_LO], ..ST_PTR); ! UPDATE NO. OF WRITES
; 3330 4 UPD_DMBC (T_ADDR [BW_LO], ..ST_PTR); ! UPDATE NO. OF BYTES WRITTEN, CHECK FOR OVERFLOW
; 3331 4 .ST_PTR = 0; ! RESET STAT TO 0
; 3332 4 ST_PTR = .ST_PTR + 2; ! ADVANCE DM_COMM POINTER
; 3333 4
; 3334 4 T_ADDR [ER_SFT] = .T_ADDR [ER_SFT] + ..ST_PTR; ! UPDATE NO. OF DM SOFT ERRORS
; 3335 4 .ST_PTR = 0; ! RESET STAT TO 0
; 3336 4 ST_PTR = .ST_PTR + 2; ! ADVANCE DM_COMM POINTER
; 3337 4
; 3338 4 HARD_ERR (..ST_PTR); ! UPDATE HRD ERR CNT + CHK FOR LIMIT
```


CZRCDB3
V02.0

CZRCDB0 RC25 DISK EXERCISER
DM EXERCISER ROUTINES

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRCDB3.SRC;17

SEQ 0267
Page 74
(22)

000126	062716	000004		ADD	#4,(SP)			
000132	011146			MOV	(R1),-(SP)		; ST.PTR,*	
000134	004737	000000G		JSR	PC,UPD.IOC			
000140	013716	000000G		MOV	T.ADDR,(SP)			
000144	062716	000022		ADD	#22,(SP)			3330
000150	011146			MOV	(R1),-(SP)		; ST.PTR,*	
000152	004737	000000V		JSR	PC,UPD.DMBC			
000156	005021			CLR	(R1)+		; ST.PTR	
000160	013700	000000G		MOV	T.ADDR,R0			3331
000164	061160	000034		ADD	(R1),34(R0)			3334
000170	005021			CLR	(R1)+		; ST.PTR,*	
000172	011116			MOV	(R1),(SP)		; ST.PTR	3335
000174	004737	000000G		JSR	PC,HARD.ERR		; ST.PTR,*	3338
000200	005021			CLR	(R1)+		; ST.PTR	
000202	005703			TST	R3		; NBXFR	3339
000204	001425			BEQ	3\$			3342
000206	104450			TRAP	50			
000210	103015			BHIS	2\$			3346
000212	013716	000000G		MOV	CPLAT,(SP)			
000216	013746	000000G		MOV	L\$LUN,-(SP)			3348
000222	010346			MOV	R3,-(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	CCTL,R0			3349
000250	006300			ASL	R0			
000252	012760	000550 000620'		MOV	#550,DM.TIMR(R0)			
000260	004737	000000G	3\$:	JSR	PC,XFR.CHK			3353
000264	062706	000014		ADD	#14,SP			3314
000270	000402			BR	5\$			3312
000272	062701	000012	4\$:	ADD	#12,R1		; *,ST.PTR	3357
000276	005202		5\$:	INC	R2		; OFFSET	3309
000300	020227	000003		CMP	R2,#3		; OFFSET,*	
000304	003644			BLE	1\$			
000306	013700	000000G		MOV	DMC.ADDR,R0			3361
000312	005060	000062		CLR	62(R0)			
000316	000207			RTS	PC			3282

; Routine Size: 104 words, Routine Base: \$CODE\$ + 5532
; Maximum stack depth per invocation: 16 words

```

; 3364 1  ROUTINE UPD_DMBC (ADDR, COUNT) : NOVALUE =
; 3365 1
; 3366 1  !+
; 3367 1  !
; 3368 1  !   THIS ROUTINE IS CALLED FROM DM_TALLY TO UPDATE THE TOTAL NUMBER OF
; 3369 1  !   BYTES READ OR WRITTEN BY THE FRONT PANEL TEST (FPT) FOR THE CURRENT
; 3370 1  !   UNIT. IT IS ASSUMED THAT EACH READ OR WRITE OPERATION PERFORMED BY THE
; 3371 1  !   FPT IS EXACTLY ONE BLOCK (512 BYTES) LONG. (A SIMPLE MULTIPLICATION
; 3372 1  !   (.COUNT * BLK_SIZE) IS NOT FEASIBLE BECAUSE A 16-BIT OVERFLOW WOULD
; 3373 1  !   OCCUR).
; 3374 1  !
; 3375 1  !   INPUTS:
; 3376 1  !   ADDR - TALLY ADDRESS OF THE LOW-ORDER FIELD OF THE NUMBER OF
; 3377 1  !   BYTES READ OR WRITTEN FOR THE CURRENT UNIT
; 3378 1  !   COUNT - THE NUMBER OF READ OR WRITE OPERATIONS PERFORMED BY THE
; 3379 1  !   FPT
; 3380 1  !-
; 3381 2  BEGIN
; 3382 2
; 3383 2  INCR I FROM 1 TO .COUNT DO           ! FOR EACH FPT I/O OPERATION
; 3384 3  BEGIN
; 3385 3
; 3386 3  .ADDR = ..ADDR + BLK_SIZE;           ! UPDATE TOTAL NO. OF BYTES (LOW ORDER)
; 3387 3  IF ..ADDR GEQU 50000                 ! IF LOW ORDER VALUE IS APPROACHING OVERFLOW
; 3388 3  THEN                               ! THEN
; 3389 3  OVF_CHK (.ADDR);                 ! REDUCE LOW ORDER WORD, UPDATE HIGH ORDER WORDS
; 3390 3
; 3391 2  END;                             ! END - I/O OPERATION COUNT LOOP
; 3392 2
; 3393 2  OVF_CHK (.ADDR);                 ! ONE MORE TIME
; 3394 2
; 3395 1  END;                             ! ROUTINE UPD_DMBC

```

000000	004137	000000G	.SBTTL	UPD_DMBC DM EXERCISER ROUTINES		
			UPD_DMBC:			
000004	016601	000012	JSR	R1,\$SAVE2	:	3364
000010	005002		MOV	12(SP),R1	:	3386
000012	000411		CLR	R2	:	3383
000014	062711	001000	BR	2\$:	
000020	021127	141520	1\$:	ADD #1000,(R1)	:	3386
000024	103404			CMP (R1),#141520	:	3387
000026	010146			BLO 2\$:	
000030	004737	000000G		MOV R1,-(SP)	:	3389
000034	005726			JSR PC,OVF.CHK	:	
000036	005202			TST (SP)+	:	
000040	020266	000010	2\$:	INC R2	:	3383
000044	003763			CMP R2,10(SP)	:	
000046	010146			BLE 1\$:	
000050	004737	000000G		MOV R1,-(SP)	:	3393
000054	005726			JSR PC,OVF.CHK	:	
000056	000207			TST (SP)+	:	3381
				RTS PC	:	3364

```

; Routine Size: 24 words,      Routine Base: $CODE$ + 6052
; Maximum stack depth per invocation: 5 words

```



```

; 3396 1 ROUTINE DM_TIME : NOVALUE =
; 3397 1
; 3398 1 !+
; 3399 1 ! THIS ROUTINE IS CALLED ONCE PER SECOND FOR EACH CONFIGURED CONTROLLER
; 3400 1 ! BY THE DM EXERCISER "EXECUTIVE" (DM_EXER). ITS PURPOSE IS TO MAINTAIN
; 3401 1 ! THE SANITY TIMER ON THE FRONT PANEL TEST (FPT) WHICH IS EXECUTING IN
; 3402 1 ! THE DIAGNOSTIC MACHINE (DM).
; 3403 1 !
; 3404 1 ! THE FPT IS EXPECTED TO PROVIDE THE HOST WITH A NEW SET OF OPERATING
; 3405 1 ! STATISTICS IN DM_COMM AT REGULAR INTERVALS. EACH CONTROLLER HAS AN
; 3406 1 ! ENTRY IN DM_TIMR WHICH CONTAINS THE REMAINING TIME (IN SECONDS) TO
; 3407 1 ! RECEIVE A NEW SET OF STATS. IF THE TIMER FOR THE CURRENT CONTROLLER
; 3408 1 ! IS ACTIVE, THEN THIS ROUTINE DECREMENTS THE TIMER, AND DECLARES AN
; 3409 1 ! ERROR IF THE TIMER REACHES ZERO.
; 3410 1 !
; 3411 1 ! INPLICIT INPUTS:
; 3412 1 ! CCTLR - CURRENT CONTROLLER NUMBER
; 3413 1 ! CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
; 3414 1 !-
; 3415 1
; 3416 2 BEGIN
; 3417 2
; 3418 2 IF .DM_TIMR [.CCTLR] NEQU 0 ! IF TIMER IS ACTIVE
; 3419 2 THEN ! THEN
; 3420 3 BEGIN
; 3421 3
; 3422 3 DM_TIMR [.CCTLR] = .DM_TIMR [.CCTLR] - 1; ! DECREMENT
; 3423 3 IF .DM_TIMR [.CCTLR] EQLU 0 ! IF TIME HAS EXPIRED
; 3424 3 THEN ! THEN
; 3425 4 BEGIN
; 3426 4
; 3427 4 L$UNIT = .CST_ADDR [OF_UN, P_UNIT]; ! GET DS UNIT NUMBER OF FIRST PLATTER UNDER CONTROLLER
; 3428 4 ERRDF (22, EGD_22, EMS_22); ! "DM EXERCISER TIMEOUT"
; 3429 4 DROP_CTLR (.CCTLR, DU_FATAL); ! DROP ALL CONTROLLER'S UNITS
; 3430 4 CST_ADDR [STATE] = OFFLINE; ! MARK CONTROLLER OFFLINE
; 3431 4
; 3432 3 END;
; 3433 3
; 3434 2 END;
; 3435 2
; 3436 1 END; ! ROUTINE DM_TIME

```

000000	010146		.SBTTL	DM.TIME DM EXERCISER ROUTINES	
000002	013700	000000G	DM.TIME:MOV	R1, -(SP)	3396
000006	006300		MOV	CCTLR, R0	3418
000010	062700	000620'	ASL	R0	
000014	005710		ADD	#DM.TIMR, R0	
000016	001433		TST	(R0)	
000020	005310		BEQ	1\$	
000022	001031		DEC	(R0)	3422
000024	013700	000000G	BNE	1\$	3423
000030	016001	000006	MOV	CST_ADDR, R0	3427
000034	000301		MOV	6(R0), R1	
000036	042701	177740	SWAB	R1	
000042	010137	000000G	BIC	#177740, R1	
			MOV	R1, L\$UNIT	

CZRC03
V02.0

CZRCDB0 RC25 DISK EXERCISER
DM EXERCISER ROUTINES

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER#1:(AZTEC.CZRCDB)CZRC03.SRC:17

SEQ 0270
Page 78
(24)

000046	104455		TRAP	55			
000050	000026		.WORD	26	:		3428
000052	000000G		.WORD	EGD.22			
000054	000000G		.WORD	EMS.22			
000056	013746	000000G	MOV	CCTLR,-(SP)	:		3429
000062	012746	000004	MOV	#4,-(SP)			
000066	004737	000000G	JSR	PC,DROP.CTLR			
000072	013700	000000G	MOV	CST.ADDR,R0	:		3430
000076	042760	100000 000002	BIC	#100000,2(R0)			
000104	022626		CMP	(SP)+,(SP)+	:		3425
000106	012601	1#:	MOV	(SP)+,R1	:		3396
000110	000207		RTS	PC			

; Routine Size: 37 words, Routine Base: \$CODE\$ + 6132
; Maximum stack depth per invocation: 4 words


```

: 3437 1  *SBTTL 'MULTI-DRIVE SUBTEST ROUTINES'
: 3438 1
: 3439 1  ROUTINE MULTI_DRIVE : NOVALUE =
: 3440 1
: 3441 1  !*
: 3442 1  !
: 3443 1  !   THIS SUBTEST IS THE MOST SIGNIFICANT PART OF THE ENTIRE PROGRAM. THE
: 3444 1  !   MULTI-DRIVE SUBTEST IS A HOST-CONTROLLED EXERCISER DESIGNED TO GIVE THE
: 3445 1  !   USER AN INDICATION OF HOW ONE OR SEVERAL RC25 DRIVES WOULD PERFORM IN
: 3446 1  !   AN OPERATING SYSTEM ENVIRONMENT.
: 3447 1  !
: 3448 1  !   THIS ROUTINE ACTS AS AN "EXECUTIVE" TO THE WHOLE PROCESS. AFTER
: 3449 1  !   INVOKING MD_INIT TO INITIALIZE MULTI-DRIVE SUBTEST DATA, THIS ROUTINE
: 3450 1  !   ENTERS A LOOP WHICH ISSUES QIOS TO ALL ACTIVE CONTROLLERS AND PROCESSES
: 3451 1  !   ANY RESPONSES. IN ADDITION, ALL OUTSTANDING COMMANDS ARE TIMED IN
: 3452 1  !   DRV_TIMCHK WHICH IS INVOKED EVERY SECOND. NORMAL TERMINATION OF THIS
: 3453 1  !   LOOP OCCURS WHEN QIOS ARE NO LONGER BEING ISSUED, AND ALL OUTSTANDING
: 3454 1  !   QIOS HAVE COMPLETED.
: 3455 1  !-
: 3456 2  BEGIN
: 3457 2
: 3458 3  IF MANUAL           ! IF ATTENDED
: 3459 2  THEN             ! THEN
: 3460 2     PRINTF (MSG_05);   ! "MULTI-DRIVE SUBTEST START"
: 3461 2  MD_INIT ();        ! INIT MULTI-DRIVE SUBTEST DATA
: 3462 2
: 3463 2  DO               ! START OF EXECUTIVE LOOP
: 3464 3    BEGIN
: 3465 3
: 3466 3    INCR CTLR FROM 0 TO (.CTLR_CNT - 1) DO    ! FOR EACH CONTROLLER
: 3467 4      BEGIN
: 3468 4
: 3469 4      SET_CPAR (.CTLR);           ! SET UP CURRENT CONTROLLER PARAMETERS
: 3470 4      IF QIO_OK () EQLU TRUE     ! IF O.K. TO ISSUE QIO(S) TO THIS CONTROLLER
: 3471 4      THEN                       ! THEN
: 3472 5        BEGIN
: 3473 5
: 3474 5        QIO_GEN ();              ! GENERATE 1 OR 2 QIOS
: 3475 5        IF .MX1 GEQ 0           ! IF SUCCESS ON FIRST QIO
: 3476 5        THEN                   ! THEN
: 3477 6          BEGIN
: 3478 6
: 3479 6          IF SEND (.MX1) EQLU SUCCESS    ! ATTEMPT TO SEND IT. IF SUCCESS
: 3480 6          THEN                       ! THEN
: 3481 6            QIO [.CTLR] = .QIO [.CTLR] + 1  ! INCR OUTSTANDING QIO COUNT
: 3482 6          ELSE                       ! OTHERWISE
: 3483 6            PUT_ENV (.MX1);          ! RETURN ENVELOPE TO POOL
: 3484 6
: 3485 5          END;
: 3486 5
: 3487 5          IF .MX2 GEQ 0           ! IF SUCCESS ON SECOND QIO
: 3488 5          THEN                   ! THEN
: 3489 6            BEGIN
: 3490 6
: 3491 6            IF SEND (.MX2) EQLU SUCCESS    ! ATTEMPT TO SEND IT. IF SUCCESS
: 3492 6            THEN                       ! THEN
: 3493 6              QIO [.CTLR] = .QIO [.CTLR] + 1  ! INCR OUTSTANDING QIO COUNT

```

```

; 3494 6      ELSE
; 3495 6      PUT_ENV (.MX2);
; 3496 6
; 3497 5      END;
; 3498 5
; 3499 4      END;
; 3500 4      ! O.K. TO ISSUE QIO(S)
; 3501 4      IF .DCT_ADDR [STAT] EQLU ONLINE
; 3502 4      THEN
; 3503 4      INT_PROC ();
; 3504 4      ! IF CONTROLLER IS ONLINE
; 3505 3      END;
; 3506 3      ! THEN
; 3507 3      PROC_RETPKT ();
; 3508 3      IF .T_FLAG EQLU TRUE
; 3509 3      THEN
; 3510 3      DRV_TIMCHK ();
; 3511 3      ! PROCESS ANY RETURN PACKETS
; 3512 3      END
; 3513 3      ! IF ONE SECOND HAS ELAPSED
; 3514 2      UNTIL MD_QUIT ();
; 3515 2      ! THEN
; 3516 1      END;
; 3516 1      ! CHECK OUTSTANDING COMMAND TIMERS
; 3516 1      ! EXECUTIVE PROCESSING LOOP
; 3516 1      ! UNTIL ALL UNITS ARE FINISHED
; 3516 1      ! ROUTINE MULTI_DRIVE

```

```

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

```


CZRC03
V02.0

CZRCDB0 RC25 DISK EXERCISER
MULTI-DRIVE SUBTEST ROUTINES

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC03.SRC;17

SEQ 0273
Page 81
(25)

000134	010016		MOV	R0,(SP)	:		
000136	004737	000000G	JSR	PC,SEND	:		3491
000142	020027	000001	CMP	R0,#1	:		
000146	001003		BNE	6\$:		
000150	105261	000000G	INCB	QIO(R1)	:	*(CTLR)	3493
000154	000404		BR	7\$:		3491
000156	013716	000604'	MOV	MX2,(SP)	:		3495
000162	004737	000000G	JSR	PC,PUT.ENV	:		
000166	005777	000000G	TST	@DCT.ADDR	:		
000172	100002		BPL	8\$:		3501
000174	004737	000000V	JSR	PC,INT.PROC	:		
000200	005726		TST	(SP)+	:		3503
000202	005201		INC	R1	:	CTLR	3467
000204	020102		CMP	R1,R2	:	CTLR,*	3466
000206	002715		BLT	3\$:		
000210	004737	000000V	JSR	PC,PROC.RETPKT	:		3507
000214	123727	000000G 000001	CMPB	T.FLAG,#1	:		3508
000222	001002		BNE	10\$:		
000224	004737	000000V	JSR	PC,DRV.TIMCHK	:		3510
000230	004737	000000V	JSR	PC,MD.QUIT	:		3514
000234	006000		ROR	R0	:		
000236	103275		BCC	2\$:		
000240	000207		RTS	PC	:		3439

: Routine Size: 81 words, Routine Base: \$CODE\$ + 6244
: Maximum stack depth per invocation: 7 words

```

: 3517 1  ROUTINE MD_INIT : NOVALUE =
: 3518 1
: 3519 1  !*
: 3520 1  ! THIS ROUTINE IS CALLED BY ROUTINE MULTI_DRIVE TO INITIALIZE DATA ITEMS
: 3521 1  ! USED BY THE MULTI-DRIVE SUBTEST.
: 3522 1  !-
: 3523 1
: 3524 2  BEGIN
: 3525 2
: 3526 2  INIT_IO_BUFF ();           ! PARTITION FREE MEMORY INTO I/O BUFFERS
: 3527 2  IF .ENTRY_REASON NEQU NEW_PASS ! IF START, RESTART, CONT, PWR FAIL
: 3528 2  THEN                       ! THEN
: 3529 3      BEGIN
: 3530 3
: 3531 3      INCR UNIT FROM 0 TO (MAX_UNITS - 1) DO ! FOR EACH UNIT
: 3532 4          BEGIN
: 3533 4
: 3534 5          IF ((.ENTRY_REASON LEQU RESTART) OR ! IF START OR RESTART OR
: 3535 5              (.BST [.UNIT, TRACK] LSSU .SWP_STRACK) OR ! IF USER CHANGED TRACK LIMITS
: 3536 5              (.BST [.UNIT, TRACK] GTRU .SWP_ETRACK))
: 3537 4          THEN ! THEN
: 3538 5              BEGIN
: 3539 5
: 3540 5              BST [.UNIT, TRACK] = .SWP_STRACK; ! INITIALIZE BLOCK SEQUENCE TABLE
: 3541 5              BST [.UNIT, SECTOR] = 0; ! (USED ONLY FOR SEQUENTIAL LBN MODE)
: 3542 5              DPST [.UNIT] = DP_CNT; ! INITIALIZE DATA PATTERN SEQUENCE TABLE
: 3543 5              ! (USED ONLY IF "PATTERN 0" WAS SELECTED)
: 3544 4              END;
: 3545 4          END;
: 3546 3          END; ! END UNIT LOOP
: 3547 3
: 3548 2      END; ! END - IF START, RESTART, CONT, PWR FAIL
: 3549 2
: 3550 3  IF ((.ENTRY_REASON NEQU CONT) AND ! IF START, RESTART, OR NEW PASS
: 3551 3      (.ENTRY_REASON NEQU PWR_FAIL))
: 3552 2  THEN ! THEN
: 3553 2      INCR COUNT FROM 0 TO ((MAX_UNITS * TALLY_LEN) - 1) DO ! CLEAR ALL STATS
: 3554 2          TALLY [.COUNT] = 0;
: 3555 2  INCR COUNT FROM 0 TO ((MAX_CTLR * RPS_LEN) - 1) DO ! INITIALIZE RETPKT
: 3556 2          RP_SAVE [.COUNT] = -1; ! SAVE AREA
: 3557 2  INCR COUNT FROM 0 TO (.NUM_BUFF - 1) DO ! INITIALIZE I/O BUFFER ALLOCATION
: 3558 2          BUFF_OWN [.COUNT] = -1; ! TABLE
: 3559 2  IF .MEM_MGMT ! IF SYSTEM HAS MEMORY MANAGEMENT
: 3560 2  THEN ! THEN
: 3561 2          MMRO = 1; ! ENABLE RELOCATION
: 3562 2          T_FLAG = FALSE; ! FLAG TO CALL DRV_TIMCHK EVERY SECOND
: 3563 2
: 3564 1  END; ! ROUTINE MD_INIT

```

000000	004137	000000G	.SBTTL MD.INIT MULTI-DRIVE SUBTEST ROUTINES	
000004	004737	000000V	MD.INIT:JSR R1,\$SAVE2	3517
000010	005002		JSR PC,INIT.IO.BUFF	3526
000012	153702	000000G	CLR R2	3527
000016	020227	000005	BISB ENTRY.REASON,R2	
000022	001441		CMP R2,#5	
			BEQ 4\$	


```

: 3565 1 ROUTINE INIT_IO_BUFF : NOVALUE =
: 3566 1
: 3567 1 !*
: 3568 1 ! THIS ROUTINE IS CALLED BY MD_INIT WHEN THE MULTI-DRIVE SUBTEST IS FIRST
: 3569 1 ! STARTED. IT IS RESPONSIBLE FOR PARTITIONING FREE MEMORY INTO A
: 3570 1 ! COLLECTION OF I/O BUFFERS. THE NUMBER OF I/O BUFFERS IS BASED ON THE
: 3571 1 ! NUMBER OF CONTROLLERS CONFIGURED FOR TESTING.
: 3572 1 !
: 3573 1 ! IF THIS HOST PROCESSOR HAS MORE THAN 28K (32K) OF MEMORY, THEN ALL
: 3574 1 ! AVAILABLE MEMORY OVER 32K (BUT UNDER 124K) IS DIVIDED EQUALLY AMONG
: 3575 1 ! THE NUMBER OF I/O BUFFERS (4K PER BUFFER MAXIMUM). MEMORY MANAGEMENT
: 3576 1 ! REGISTERS ARE SET UP TO MAP ALL VIRTUAL ADDRESSES TO THE SAME PHYSICAL
: 3577 1 ! ADDRESSES (FOR NOW). IF THE HOST PROCESSOR HAS ONLY 28K, THEN THE
: 3578 1 ! AMOUNT OF FREE MEMORY UNDER 28K IS SIMILARLY DIVIDED (WITHOUT MEMORY
: 3579 1 ! MANAGEMENT).
: 3580 1 !
: 3581 1 ! ONCE THE STARTING ADDRESS OF FREE MEMORY IS DETERMINED ALONG WITH THE
: 3582 1 ! BUFFER SIZE, THE BUFFER DESCRIPTOR TABLE (BUFF_DESC) IS INITIALIZED.
: 3583 1 ! THIS TABLE REMAINS FIXED THROUGHOUT THE MULTI-DRIVE SUBTEST. IT
: 3584 1 ! PROVIDES THE TWO-WORD BUFFER DESCRIPTORS USED IN MSCP COMMAND
: 3585 1 ! ENVELOPES.
: 3586 1 !
: 3587 1 ! IMPLICIT INPUTS:
: 3588 1 ! CTLR_CNT - THE NUMBER OF CONTROLLERS CONFIGURED
: 3589 1 ! L$HIMEM - TOP OF FREE MEMORY (IN PAR FORMAT)
: 3590 1 ! MEM_SIZE - SIZE (IN WORDS) OF FREE MEMORY
: 3591 1 ! FREE_MEM_ADDR - START OF FREE MEMORY
: 3592 1 !-
: 3593 1
: 3594 2 BEGIN
: 3595 2
: 3596 2 LOCAL
: 3597 2 HIMEM : WORD, ! TOP OF FREE MEMORY (IN PAR FORMAT)
: 3598 2 BLKS_PER_BUFF : WORD, ! NO. OF 32-WORD BLOCKS PER BUFFER
: 3599 2 BD_LOW : WORD, ! BUFFER DESCRIPTOR (LO-ORDER)
: 3600 2 BD_HIGH : WORD, ! BUFFER DESCRIPTOR (HI-ORDER)
: 3601 2 TEMP : WORD,
: 3602 2 KTPDR : WORD; ! PDR INITIALIZATION WORD
: 3603 2
: 3604 2 NUM_BUFF = .CTLR_CNT * QIO_PER_CTLR; ! CALCULATE NO. OF I/O BUFFERS NEEDED
: 3605 2 MEM_MGMT = NO; ! ASSUME NO MEMORY MANAGEMENT
: 3606 2 IF .L$HIMEM GTRU #0'2000' ! IF MORE THAN 32K OF MEMORY
: 3607 2 THEN ! THEN
: 3608 3 BEGIN
: 3609 3
: 3610 3 MEM_MGMT = YES; ! SET MEMORY MANAGEMENT FLAG
: 3611 3 HIMEM = .L$HIMEM; ! GET TOP OF FREE MEMORY
: 3612 3 IF .HIMEM GTRU #0'7577' ! IF GREATER THAN 124K
: 3613 3 THEN ! THEN
: 3614 3 HIMEM = #0'7577'; ! SET LIMIT AT 124K
: 3615 3 BLKS_PER_BUFF = (.HIMEM - #0'1777') / .NUM_BUFF; ! NO. OF 32-WORD BLOCKS PER BUFFER
: 3616 3 IF .BLKS_PER_BUFF GTRU #0'200'
: 3617 3 THEN
: 3618 3 BLKS_PER_BUFF = #0'200'; ! MAX BUFFER SIZE = 4K
: 3619 3 BUFF_SIZE = .BLKS_PER_BUFF * 64; ! BUFFER SIZE IN BYTES
: 3620 3 BD_LOW = 0; ! BUFFERS START AT 32K
: 3621 3 BD_HIGH = 1;

```



```

: 3622 3
: 3623 3      END
: 3624 2      ELSE
: 3625 3      BEGIN
: 3626 3
: 3627 3      BUFF_SIZE = ((.MEM_SIZE * 2) / .NUM_BUFF) AND %0'177776';
: 3628 3      BD_LOW = .FREE_MEM_ADDR;
: 3629 3      BD_HIGH = 0;
: 3630 3
: 3631 2      END;
: 3632 2
: 3633 2      INCR J FROM 0 TO (.NUM_BUFF - 1) DO
: 3634 3      BEGIN
: 3635 3
: 3636 3      BUFF_DESC [.J, BD_LO] = .BD_LOW;
: 3637 3      BUFF_DESC [.J, BD_HI] = .BD_HIGH;
: 3638 3      TEMP = .BD_LOW;
: 3639 3      BD_LOW = .BD_LOW + .BUFF_SIZE;
: 3640 3      IF .TEMP GEQU .BD_LOW
: 3641 3      THEN
: 3642 3          BD_HIGH = .BD_HIGH + 1;
: 3643 3
: 3644 2      END;
: 3645 2
: 3646 2      IF .MEM_MGMT
: 3647 2      THEN
: 3648 3      BEGIN
: 3649 3
: 3650 3      MMRO = 0;
: 3651 3      KTPDR = %0'77406';
: 3652 3      KTPDR0 = .KTPDR;
: 3653 3      KTPDR1 = .KTPDR;
: 3654 3      KTPDR2 = .KTPDR;
: 3655 3      KTPDR3 = .KTPDR;
: 3656 3      KTPDR4 = .KTPDR;
: 3657 3      KTPDR5 = .KTPDR;
: 3658 3      KTPDR6 = .KTPDR;
: 3659 3      KTPDR7 = .KTPDR;
: 3660 3
: 3661 3      KTPAR0 = 0;
: 3662 3      KTPAR1 = %0'200';
: 3663 3      KTPAR2 = %0'400';
: 3664 3      KTPAR3 = %0'600';
: 3665 3      KTPAR4 = %0'1000';
: 3666 3      KTPAR5 = %0'1200';
: 3667 3      KTPAR6 = %0'1400';
: 3668 3      KTPAR7 = %0'7600';
: 3669 3
: 3670 2      END;
: 3671 2
: 3672 1      END;

```

! OTHERWISE - ONLY 28K OF MEMORY

! START OF BUFFER SPACE

! FOR EACH BUFF_DESC ENTRY

! LOAD LOW-ORDER WORD

! LOAD HIGH-ORDER WORD

! ADVANCE TO NEXT BUFFER ADDRESS

! IF 16-BIT CARRY

! THEN

! ADVANCE HIGH-ORDER WORD

! IF SYSTEM HAS MEMORY MANAGEMENT

! THEN

! MAKE SURE MEM MGMT IS OFF

! PDR LOAD VALUE

! LOAD PDR'S

! LOAD PAR'S

! ROUTINE INIT_IO_BUFF

```

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

```

3565
3604

CZRC03
V02.0

CZRCDB0 RC25 DISK EXERCISER
MULTI-DRIVE SUBTEST ROUTINES

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC03.SRC;17

SEQ 0278
Page 86
(27)

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	:		3605
000030	010005		MOV	R0,R5	:	NUM.BUFF,*	3615
000032	023727	000000G 002000	CMP	L\$HIMEM,#2000	:		3606
000040	101441		BLOS	3\$			
000042	112737	000001 000000G	MOVB	#1,MEM.MGMT	:		3610
000050	013700	000000G	MOV	L\$HIMEM,R0	:	*,HIMEM	3611
000054	020027	007577	CMP	R0,#7577	:	HIMEM,*	3612
000060	101402		BLOS	1\$			
000062	012700	007577	MOV	#7577,R0	:	*,HIMEM	3614
000066	010046		MOV	R0,-(SP)	:	HIMEM,*	3615
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,*	3616
000106	101402		BLOS	2\$			
000110	012700	000200	MOV	#200,R0	:	*,BLKS.PER.BUFF	3618
000114	000300		SWAB	R0	:		3619
000116	106000		RORB	R0			
000120	0C6000		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	3620
000136	012703	000001	MOV	#1,R3	:	*,BD.HIGH	3621
000142	000416		BR	4\$:		3606
000144	013746	000000G	MOV	MEM.SIZE,-(SP)	:		3627
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	3628
000176	005003		CLR	R3	:	BD.HIGH	3629
000200	005001		CLR	R1	:	J	3633
000202	000416		BR	7\$			
000204	010100		MOV	R1,R0	:	J,*	3636
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,*	3637
000222	010204		MOV	R2,R4	:	BD.LOW,TEMP	3638
000224	063702	000000G	ADD	BUFF.SIZE,R2	:	*,BD.LOW	3639
000230	020402		CMP	R4,R2	:	TEMP,BD.LOW	3640
000232	103401		BLO	6\$			
000234	005203		INC	R3	:	BD.HIGH	3642
000236	005201		INC	R1	:	J	3633
000240	020105		CMP	R1,R5	:	J,*	
000242	002760		BLT	5\$			
000244	132737	000001 000000G	BITB	#1,MEM.MGMT	:		3646
000252	001453		BEQ	8\$			
000254	005037	177572	CLR	#177572	:		3650
000260	012700	077406	MOV	#77406,R0	:	*,KTPDR	3651

CZRC03
V02.0

CZRCDB0 RC25 DISK EXERCISER
MULTI-DRIVE SUBTEST ROUTINES

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER\$1:[AZTEC.CZRCDB]CZRC03.SRC;17

SEQ 0279
Page 87
(27)

000264	010037	172300		MOV	R0,@#172300	; KTPDR,*	3652
000270	010037	172302		MOV	R0,@#172302	; KTPDR,*	3653
000274	010037	172304		MOV	R0,@#172304	; KTPDR,*	3654
000300	010037	172306		MOV	R0,@#172306	; KTPDR,*	3655
000304	010037	172310		MOV	R0,@#172310	; KTPDR,*	3656
000310	010037	172312		MOV	R0,@#172312	; KTPDR,*	3657
000314	010037	172314		MOV	R0,@#172314	; KTPDR,*	3658
000320	010037	172316		MOV	R0,@#172316	; KTPDR,*	3659
000324	005037	172340		CLR	@#172340	:	3661
000330	012737	000200	172342	MOV	#200,@#172342	:	3662
000336	012737	000400	172344	MOV	#400,@#172344	:	3663
000344	012737	000600	172346	MOV	#600,@#172346	:	3664
000352	012737	001000	172350	MOV	#1000,@#172350	:	3665
000360	012737	001200	172352	MOV	#1200,@#172352	:	3666
000366	012737	001400	172354	MOV	#1400,@#172354	:	3667
000374	012737	007600	172356	MOV	#7600,@#172356	:	3668
000402	022626			8\$: CMP	(SP)+,(SP)+	:	3594
000404	000207			RTS	PC	:	3565

; Routine Size: 131 words, Routine Base: \$CODE\$ + 6756
; Maximum stack depth per invocation: 9 words

```

: 3673 1 ROUTINE QIO_OK =
: 3675 1 !+
: 3676 1 !
: 3677 1 ! THIS ROUTINE IS CALLED BY THE MULTI_DRIVE "EXECUTIVE" IN ORDER TO
: 3678 1 ! DETERMINE WHETHER OR NOT A QIO REQUEST (OR QIO PAIR) SHOULD BE
: 3679 1 ! GENERATED TO THE CURRENT CONTROLLER. A VALUE OF "TRUE" IS RETURNED IF
: 3680 1 ! THE CONTROLLER MEETS 3 REQUIREMENTS:
: 3681 1 !
: 3682 1 ! A. THE CONTROLLER IS ONLINE;
: 3683 1 ! B. THE NUMBER OF OUTSTANDING QIOS IS AT LEAST 2 LESS THAN THE
: 3684 1 ! MAXIMUM ALLOWED FOR ANY ONE CONTROLLER;
: 3685 1 ! C. THERE IS AT LEAST ONE UNIT UNDER THE CONTROLLER WHICH IS
: 3686 1 ! STILL UNDER TEST.
: 3687 1 !
: 3688 1 ! IF ANY OF THESE TESTS FAIL, THEN A VALUE OF "FALSE" IS RETURNED.
: 3689 1 !
: 3690 1 ! IMPLICIT INPUTS:
: 3691 1 ! CCTLR - CURRENT CONTROLLER NUMBER
: 3692 1 ! CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
: 3694 2 !-
: 3695 2 BEGIN
: 3696 2 IF .CST_ADDR [STATE] EQLU ONLINE ! IF CONTROLLER IS ONLINE
: 3697 2 THEN
: 3698 3 BEGIN
: 3699 3
: 3700 3 IF (.QIO [.CCTLR] + 2) LEQU QIO_PER_CTLR ! IF OUTSTANDING QIO COUNT IS O.K.
: 3701 3 THEN
: 3702 4 BEGIN
: 3703 4
: 3704 4 IF .CST_ADDR [U_CNT] NEQU 0 ! IF THERE IS VALID UNIT
: 3705 4 THEN
: 3706 4 RETURN TRUE; ! "TRUE" EXIT POINT
: 3708 3 END;
: 3710 2 END;
: 3711 2
: 3712 2 RETURN FALSE; ! "FALSE" EXIT POINT
: 3714 1 END;

```

000000	013700	000000G	.SBTTL	QIO.OK MULTI-DRIVE SUBTEST ROUTINES	
000004	005760	000002	QIO.OK: MOV	CST.ADDR,R0	3696
000010	100023		TST	2(R0)	
000012	013700	000000G	BPL	1\$	
000016	116000	000000G	MOV	CCTLR,R0	3700
000022	042700	177400	MOVB	QIO(R0),R0	
000026	062700	000002	BIC	#177400,R0	
000032	020027	000020	ADD	#2,R0	
000036	101010		CMP	R0,#20	
000040	013700	000000G	BHI	1\$	
000044	105760	000005	MOV	CST.ADDR,R0	3704
000050	001403		TSTB	5(R0)	
000052	012700	000001	BEQ	1\$	
000056	000207		MOV	#1,R0	3706
000060	005000		RTS	PC	
000062	000207		1\$: CLR	R0	3694
			RTS	PC	3673

; Routine Size: 26 words, Routine Base: \$CODE\$ + 7364
; Maximum stack depth per invocation: 0 words


```

; 3715 1 ROUTINE MD_QUIT =
; 3716 1
; 3717 1 !+
; 3718 1 ! THIS ROUTINE IS CALLED BY THE MULTI_DRIVE EXECUTIVE FOR DETERMINING THE
; 3719 1 ! END OF THE MULTI-DRIVE SUBTEST. ITS PURPOSE IS TO EXAMINE THE NUMBER OF
; 3720 1 ! TESTABLE UNITS REMAINING ON EACH CONTROLLER, AND THE NUMBER OF
; 3721 1 ! OUTSTANDING QIOS. IF BOTH OF THESE PARAMETERS ARE ZERO FOR ALL
; 3722 1 ! CONTROLLERS, THEN THIS ROUTINE RETURNS A VALUE OF "TRUE". OTHERWISE,
; 3723 1 ! THE MULTI-DRIVE SUBTEST EXECUTIVE LOOP MUST CONTINUE, AND A VALUE OF
; 3724 1 ! "FALSE" IS RETURNED.
; 3725 1 !-
; 3726 1
; 3727 2 BEGIN
; 3728 2
; 3729 2 INCR CTLR FROM 0 TO (MAX_CTLR - 1) DO
; 3730 3 BEGIN
; 3731 3
; 3732 3 IF .CST [.CTLR, U_CNT] NEQU 0 ! IF AT LEAST ONE VALID UNIT REMAINS
; 3733 3 THEN ! THEN
; 3734 3 RETURN FALSE ! DON'T TERMINATE MULTI-DRIVE SUBTEST
; 3735 3 ELSE ! OTHERWISE (NO UNITS REMAIN)
; 3736 3 IF .QIO [.CTLR] NEQU 0 ! IF AT LEAST ONE QIO OUTSTANDING
; 3737 3 THEN ! THEN
; 3738 3 RETURN FALSE; ! DON'T TERMINATE SUBTEST
; 3739 3
; 3740 2 END;
; 3741 2
; 3742 2 RETURN TRUE; ! ALL PARAMETERS ARE ZERO - END OF SUBTEST
; 3743 2
; 3744 1 END; ! ROUTINE MD_QUIT

```

```

000000 010146 .SBTTL MD.QUIT MULTI-DRIVE SUBTEST ROUTINES
000002 005001 MD.QUIT:MOV R1,-(SP) ;
000004 010146 1$: CLR R1 ; CTLR
000006 012746 000016 MOV R1,-(SP) ; CTLR,*
000012 004737 000000G MOV #16,-(SP)
000016 022626 JSR PC,BL$MUL
000020 105760 000005G CMP (SP)+,(SP)+
000024 001012 TSTB CST+5(R0)
000026 105761 000000G BNE 2$ ;
000032 001007 TSTB QIO(R1) ; *(CTLR)
000034 005201 BNE 2$ ;
000036 020127 000003 INC R1 ; CTLR
000042 003760 CMP R1,#3 ; CTLR,*
000044 012700 000001 BLE 1$ ;
000050 000401 MOV #1,R0 ;
000052 005000 BR 3$ ;
000054 012601 2$: CLR R0 ;
000056 000207 3$: MOV (SP)+,R1 ;
RTS PC 3715

```

; Routine Size: 24 words, Routine Base: \$CODE\$ + 7450
; Maximum stack depth per invocation: 4 words

```
; 3745 1 ROUTINE QIO_GEN : NOVALUE =
; 3746 1
; 3747 1 !*
; 3748 1 ! THIS ROUTINE IS CALLED BY THE MULTI_DRIVE EXECUTIVE FOR AN ONLINE
; 3749 1 ! CONTROLLER ELIGIBLE TO RECEIVE I/O TRANSFER REQUESTS. IT IS
; 3750 1 ! RESPONSIBLE FOR SECURING ONE OR TWO MSCP ENVELOPES AND LOADING THEM
; 3751 1 ! WITH VARIOUS PARAMETERS COMPRISING THE I/O REQUEST. THE I/O REQUEST
; 3752 1 ! GENERATED HERE IS DESTINED TO A PARTICULAR UNIT SELECTED AT RANDOM FROM
; 3753 1 ! THOSE CONFIGURED UNDER THE CURRENT CONTROLLER.
; 3754 1 !
; 3755 1 ! EACH FIELD OF THE ENVELOPE(S) IS LOADED WITHIN INDIVIDUAL ROUTINES
; 3756 1 ! (QIO_FUNC, QIO_LBN, QIO_SIZE, ETC.). MOST OF THE VALUES SELECTED FOR
; 3757 1 ! EACH FIELD ARE BASED ON A SET OF RANDOM NUMBER GENERATED AT THE START.
; 3758 1 !
; 3759 1 ! UNDER NORMAL CIRCUMSTANCES, ONLY ONE I/O REQUEST IS GENERATED. HOWEVER,
; 3760 1 ! IF THIS I/O REQUEST IS A "WRITE", AND IF THE OPERATOR SELECTED THE
; 3761 1 ! OPTION FOR HOST WRITE-COMPARES, THEN A SECOND "READ" REQUEST WILL BE
; 3762 1 ! GENERATED WITH THE SAME LBN AND BYTE COUNT.
; 3763 1 !
; 3764 1 ! AFTER THE ENVELOPE(S) HAVE BEEN LOADED, THIS ROUTINE REGAINS CONTROL
; 3765 1 ! AND ATTEMPTS TO GET ONE OR TWO I/O BUFFERS FOR THE ACTUAL DATA
; 3766 1 ! TRANSFERS. THE SUCCESS / FAIL STATUS OF THIS ENTIRE OPERATION IS
; 3767 1 ! PASSED BACK TO THE CALLER THROUGH THE GLOBALS "MX1" AND "MX2"; THEY
; 3768 1 ! CONTAIN VALID MSCP ENVELOPE INDECES, OR -1.
; 3769 1 !
; 3770 1 ! IMPLICIT INPUTS:
; 3771 1 ! CCTLR - CURRENT CONTROLLER NUMBER
; 3772 1 !-
; 3773 1
; 3774 2 BEGIN
; 3775 2
; 3776 2 MX2 = -1;
; 3777 2 IF (MX1 = GET_ENV (.CCTLR)) LSS 0 ! ASSUME FAILURE IN SECURING 2ND ENVELOPE
; 3778 2 THEN ! TRY TO GET 1ST ENVELOPE. IF FAILURE
; 3779 2 RETURN; ! THEN
; 3780 2 IF (MX2 = GET_ENV (.CCTLR)) LSS 0 ! NO POINT IN CONTINUING
; 3781 2 THEN ! TRY TO GET 2ND ENVELOPE. IF FAILURE
; 3782 3 BEGIN ! THEN
; 3783 3
; 3784 3 PUT_ENV (.MX1); ! RETURN 1ST ENVELOPE TO POOL
; 3785 3 MX1 = -1; ! INDICATE FAILURE
; 3786 3 RETURN; ! DONE
; 3787 3
; 3788 2 END;
; 3789 2
; 3790 2 MAD1 = MSCP_ENV + (.MX1 * ENV_LEN * 2); ! CALCULATE STARTING ADDRESSES
; 3791 2 MAD2 = MSCP_ENV + (.MX2 * ENV_LEN * 2); ! OF BOTH ENVELOPES
; 3792 2 GET_RANDOM (); ! GENERATE A SET OF RANDOM NUMBERS
; 3793 2 QIO_UNIT (); ! LOAD RANDOM UNIT NUMBER INTO ENVELOPES
; 3794 2 QIO_FUNC (); ! LOAD RANDOM FUNCTION CODE (OPCODE)
; 3795 2 QIO_LBN (); ! LOAD LBN (RANDOM OR SEQUENTIAL)
; 3796 2 QIO_SIZE (); ! LOAD RANDOM BYTE COUNT
; 3797 2 GET_IO_BUFF (MAD1 [BUF_0]); ! TRY TO GET AN I/O BUFFER
; 3798 2 IF .MX2 GEQ 0 ! IF TWO QIOS ARE TO BE ISSUED
; 3799 2 THEN ! THEN
; 3800 3 BEGIN
; 3801 3
```



```

; 3802 3      GET_IO_BUFF (MAD2 [BUF_0]);      ! TRY TO GET 2ND I/O BUFFER
; 3803 3      IF .MAD2 [BUF_0]                ! IF 2ND BUFFER ALLOCATION FAILED
; 3804 3      THEN                            ! THEN
; 3805 4          BEGIN
; 3806 4
; 3807 4          IF NOT .MAD1 [BUF_0]         ! IF 1ST I/O BUFFER WAS ALLOCATED
; 3808 4          THEN                        ! THEN
; 3809 5              BEGIN
; 3810 5
; 3811 5              PUT_IO_BUFF (MAD1 [BUF_0]); ! RETURN 1ST I/O BUFFER TO POOL
; 3812 5              MAD1 [BUF_0] = -1;      ! MARK IT AS FAILED
; 3813 5
; 3814 4          END;
; 3815 4
; 3816 4          PUT_ENV (.MX2);             ! RETURN 2ND ENVELOPE TO POOL
; 3817 4          MX2 = -1;                  ! INDICATE FAILURE
; 3818 4
; 3819 3          END;                       ! IF 2ND I/O BUFFER ALLOCATION FAILED
; 3820 3
; 3821 2      END;                           ! IF TWO QIOS ARE TO BE ISSUED
; 3822 2
; 3823 2      IF .MAD1 [BUF_0]                ! IF 1ST I/O BUFFER ALLOCATION FAILED
; 3824 2      THEN                            ! THEN
; 3825 3          BEGIN
; 3826 3
; 3827 3          PUT_ENV (.MX1);             ! RETURN 1ST ENVELOPE TO POOL
; 3828 3          MX1 = -1;                  ! INDICATE FAILURE
; 3829 3
; 3830 3      END
; 3831 2      ELSE
; 3832 2          IF .MAD1 [OPCODE] EQLU OP_WRT ! OTHERWISE (ALL IS O.K.)
; 3833 2          THEN                        ! IF 1ST OPCODE IS A WRITE
; 3834 2              THEN                    ! THEN
; 3835 2                  FILL_BUFF ();      ! FILL 1ST I/O BUFFER WITH APPROPRIATE DATA PATTERN
; 3836 1      END;                           ! ROUTINE QIO_GEN

```

```

000000 012737 177777 000604'      .SBTTL QIO.GEN MULTI-DRIVE SUBTEST ROUTINES
000006 013746 000000G      QIO.GEN:MOV    #-1,MX2                ; 3776
000012 004737 000000G      MOV    CCTLR,-(SP)                ; 3777
000016 010037 000602'      JSR    PC,GET.ENV
000022 005726      MOV    R0,MX1
000024 005700      TST   (SP)+
000026 002564      TST   R0                          ; MX1
000030 013746 000000G      BLT   6$                          ;
000034 004737 000000G      MOV    CCTLR,-(SP)                ; 3779
000040 010037 000604'      JSR    PC,GET.ENV                ; 3780
000044 005726      MOV    R0,MX2
000046 005700      TST   (SP)+
000050 002011      TST   R0                          ; MX2
000052 013746 000602'      BGE   1$
000056 004737 000000G      MOV    MX1,-(SP)                  ;
000062 012737 177777 000602'      JSR    PC,PUT.ENV                ; 3784
000070 005726      MOV    #-1,MX1                    ;
000072 000207      TST   (SP)+                       ;
000074 013746 000602'      RTS   PC                          ;
1$: MOV    MX1,-(SP)                ; 3785
                                           ; 3786
                                           ; 3782
                                           ; 3790

```

000100	012746	000104		MOV	#104,-(SP)		
000104	004737	000000G		JSR	PC,BL\$MUL		
000110	062700	000000G		ADD	#MSCP.ENV,RO		
000114	010037	000606'		MOV	RO,MAD1		
000120	013716	000604'		MOV	MX2,(SP)	:	
000124	012746	000104		MOV	#104,-(SP)		3791
000130	004737	000000G		JSR	PC,BL\$MUL		
000134	062700	000000G		ADD	#MSCP.ENV,RO		
000140	010037	000610'		MOV	RO,MAD2		
000144	004737	000000V		JSR	PC.GET.RANDOM	:	3792
000150	004737	000000V		JSR	PC,QIO.UNIT	:	3793
000154	004737	000000V		JSR	PC,QIO.FUNC	:	3794
000160	004737	000000V		JSR	PC,QIO.LBN	:	3795
000164	004737	000000V		JSR	PC,QIO.SIZE	:	3796
000170	013716	000606'		MOV	MAD1,(SP)	:	3797
000174	062716	000030		ADD	#30,(SP)		
000200	004737	000000G		JSR	PC.GET.IO.BUFF		
000204	005737	000604'		TST	MX2	:	
000210	002443			BLT	3\$		3798
000212	013716	000610'		MOV	MAD2,(SP)	:	
000216	062716	000030		ADD	#30,(SP)		3802
000222	004737	000000G		JSR	PC.GET.IO.BUFF		
000226	013700	000610'		MOV	MAD2,RO	:	
000232	032760	000001 000030		BIT	#1,30(RO)		3803
000240	001427			BEQ	3\$		
000242	013700	000606'		MOV	MAD1,RO	:	
000246	032760	000001 000030		BIT	#1,30(RO)		3807
000254	001012			BNE	2\$		
000256	012716	000030		MOV	#30,(SP)	:	
000262	060016			ADD	RO,(SP)		3811
000264	004737	000000G		JSR	PC.PUT.IO.BUFF		
000270	013700	000606'		MOV	MAD1,RO	:	
000274	012760	177777 000030		MOV	#-1,30(RO)		3812
000302	013716	000604'	2\$:	MOV	MX2,(SP)	:	
000306	004737	000000G		JSR	PC.PUT.ENV		3816
000312	012737	177777 000604'		MOV	#-1,MX2	:	
000320	013700	000606'	3\$:	MOV	MAD1,RO	:	3817
000324	032760	000001 000030		BIT	#1,30(RO)		3823
000332	001410			BEQ	4\$		
000334	013716	000602'		MOV	MX1,(SP)	:	
000340	004737	000000G		JSR	PC.PUT.ENV		3827
000344	012737	177777 000602'		MOV	#-1,MX1	:	
000352	000410			BR	5\$:	3828
000354	013700	000606'	4\$:	MOV	MAD1,RO	:	3823
000360	126027	000020 000042		CMPB	20(RO),#42	:	3832
000366	001002			BNE	5\$		
000370	004737	000000V		JSR	PC.FILL.BUFF	:	
000374	062706	000006	5\$:	ADD	#6,SP	:	3834
000400	000207		6\$:	RTS	PC	:	3774
						:	3745

; Routine Size: 129 words, Routine Base: \$CODE\$ + 7530
; Maximum stack depth per invocation: 4 words


```

; 3837 1 ROUTINE GET_RANDOM : NOVALUE =
; 3838 1
; 3839 1 !*
; 3840 1 ! THIS ROUTINE IS CALLED BY QIO_GEN TO GENERATE A SET OF RANDOM NUMBERS,
; 3841 1 ! AND TO STORE THEM INTO THE RANDOM NUMBER TABLE (RANDOM). THE RANDOM
; 3842 1 ! NUMBERS ARE USED TO SELECT I/O REQUEST PARAMETERS FOR THE CURRENT QIO
; 3843 1 ! OR QIO PAIR. IN ADDITION, IF DATA PATTERN #1 IS BEING USED, THESE
; 3844 1 ! RANDOM NUMBERS WILL BE USED IN THE WRITE OPERATION.
; 3845 1 !
; 3846 1 ! A TEST IS MADE TO AVOID GENERATING THE RANDOM NUMBER 100000-OCTAL. THIS
; 3847 1 ! NUMBER, IF USED AS THE DIVIDEND IN A "MOD" OPERATION (E.G., QIO_LBN),
; 3848 1 ! REGARDLESS OF THE DIVISOR, RESULTS IN AN OUT-OF-RANGE REMAINDER.
; 3849 1 !-
; 3850 1
; 3851 2 BEGIN
; 3852 2
; 3853 2 OWN
; 3854 2 SEED1 : WORD INITIAL (#0'123456');
; 3855 2
; 3856 2 LOCAL
; 3857 2 SEED2 : WORD;
; 3858 2
; 3859 2 SEED2 = ((.MINUTES + 5) OR .SECONDS) + 5 OR .TICKS;
; 3860 2 DECR COUNT FROM (RDM_LEN - 1) TO 0 DO
; 3861 3 BEGIN
; 3862 3
; 3863 3 DO
; 3864 4 BEGIN
; 3865 4
; 3866 4 SEED1 = (.SEED1 + .SEED2 + 1) * 4;
; 3867 4 SEED2 = (.SEED2 / 4) + .SEED1;
; 3868 4 RANDOM [.COUNT] = .SEED2;
; 3869 4
; 3870 4 END
; 3871 3 UNTIL .RANDOM [.COUNT] NEQU #0'100000';
; 3872 3
; 3873 2 END;
; 3874 2
; 3875 1 END;

```

```

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

```

```

010132 .SBTTL GET_RANDOM MULTI-DRIVE SUBTEST ROUTINES
        .PSECT #CODE#, RO

000000 004137 000000G GET_RANDOM:
000004 013746 000000G JSR R1,#SAVE3
000010 012746 000005 MOV MINUTES,-(SP)
000014 004737 000000G MOV #5,-(SP)
000020 010016 JSR PC,BL#SHF
000022 053716 000000G MOV RO,(SP)
000026 012746 000005 MOV BIS SECONDS,(SP)
                MOV #5,-(SP)

```

3837
3859

CZRC03
V02.0

CZRCDB0 RC25 DISK EXERCISER
MULTI-DRIVE SUBTEST ROUTINES

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC03.SRC;17

SEQ 0286
Page 95
(31)

000032	004737	000000G		JSR	PC,BL\$SHF			
000036	010002			MOV	R0,R2	:	*,SEED2	
000040	053702	000000G		BIS	TICKS,R2	:	*,SEED2	
000044	012701	000036		MOV	#36,R1	:	*,COUNT	
000050	013703	002552'	1\$:	MOV	SEED1,R3	:		3860
000054	010200		2\$:	MOV	R2,R0	:	SEED2,*	3866
000056	060300			ADD	R3,R0			
000060	006300			ASL	R0			
000062	006300			ASL	R0			
000064	010037	002552'		MOV	R0,SEED1			
000070	062737	000004 002552'		ADD	#4,SEED1			
000076	010216			MOV	R2,(SP)	:	SEED2,*	3867
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	010002			MOV	R0,R2	:	*,SEED2	
000120	010261	000634'		MOV	R2,RANDOM(R1)	:	SEED2,*(COUNT)	3868
000124	005726			TST	(SP)+	:		3864
000126	020227	100000		CMP	R2,#-100000	:		3871
000132	001750			BEQ	2\$			
000134	162701	000002'		SUB	#2,R1	:	*,COUNT	3860
000140	100343			BPL	1\$			
000142	062706	000006		ADD	#6,SP	:		3851
000146	000207			RTS	PC	:		3837

: Routine Size: 52 words, Routine Base: \$CODE\$ + 10132
: Maximum stack depth per invocation: 9 words


```

: 3876 1 ROUTINE QIO_UNIT : NOVALUE =
: 3877 1
: 3878 1 !*
: 3879 1 ! THIS ROUTINE IS CALLED BY QIO_GEN TO RANDOMLY SELECT ONE UNIT
: 3880 1 ! CONFIGURED UNDER THE CURRENT CONTROLLER (CCTL) TO BE USED FOR THE
: 3881 1 ! CURRENT QIO OR QIO PAIR. THE UNIT SELECTED IS BASED ON THE NUMBER OF
: 3882 1 ! UNITS ELIGIBLE TO RECEIVE AN I/O REQUEST (FROM 1 TO 4) AND THE FIRST
: 3883 1 ! RANDOM NUMBER IN THE RANDOM NUMBER TABLE (RANDOM).
: 3884 1 !
: 3885 1 ! IMPLICIT INPUTS:
: 3886 1 ! CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
: 3887 1 !
: 3888 1 ! IMPLICIT OUTPUTS:
: 3889 1 ! THE MSCP UNIT NUMBER (PLATTER ADDRESS) IS LOADED INTO THE
: 3890 1 ! APPROPRIATE FIELD OF BOTH MSCP ENVELOPES.
: 3891 1 !-
: 3892 1
: 3893 2 BEGIN
: 3894 2
: 3895 2 LOCAL
: 3896 2 UNIT : WORD, ! UNIT NUMBER
: 3897 2 NTH_UNIT : WORD; ! ORDINAL NUMBER OF CHOSEN UNIT
: 3898 2
: 3899 2 CASE .CST_ADDR [U_CNT] ! DETERMINE ORDINAL UNIT NUMBER
: 3900 2 FROM 1 TO 4 OF
: 3901 2 SET
: 3902 2 [1] : NTH_UNIT = 1; ! 1
: 3903 2 [2] : NTH_UNIT = (.RANDOM [0] AND 1) + 1; ! 1 OR 2
: 3904 2 [3] : NTH_UNIT = ABS (.RANDOM [0] MOD 3) + 1; ! 1, 2, OR 3
: 3905 2 [4] : NTH_UNIT = (.RANDOM [0] AND 3) + 1; ! 1, 2, 3, OR 4
: 3906 2 [OUTRANGE] : NTH_UNIT = 1; ! IN CASE CONTROLLER JUST WENT DOWN
: 3907 2 TES;
: 3908 2
: 3909 2 INCR OFFSET FROM (0 + OF_UN) TO (3 + OF_UN) DO ! LOOP THROUGH EACH CST UNIT
: 3910 3 BEGIN
: 3911 3
: 3912 3 IF .CST_ADDR [.OFFSET, P_PRES] EQLU PRESENT ! IF UNIT IS PRESENT
: 3913 3 THEN ! THEN
: 3914 4 BEGIN
: 3915 4
: 3916 4 UNIT = .CST_ADDR [.OFFSET, P_UNIT]; ! GET (DRS) UNIT NUMBER
: 3917 4 IF .CPT [.UNIT] ! IF THIS UNIT IS ACTIVE
: 3918 4 THEN ! THEN
: 3919 5 BEGIN ! THIS IS AN ELIGIBLE UNIT
: 3920 5
: 3921 5 NTH_UNIT = .NTH_UNIT - 1; ! DECREMENT ORDINAL COUNT
: 3922 5 IF .NTH_UNIT EQLU 0 ! IF DOWN TO 0
: 3923 5 THEN ! THEN
: 3924 6 BEGIN ! THIS IS THE CHOSEN UNIT
: 3925 6
: 3926 6 SET_UPAR (.OFFSET); ! SET UP UNIT-RELATED DATA
: 3927 6 EXITLOOP; ! DONE
: 3928 6
: 3929 5 END;
: 3930 5
: 3931 4 END; ! UNIT IS ACTIVE
: 3932 4
```

```

: 3933 3
: 3934 3
: 3935 2
: 3936 2
: 3937 2
: 3938 2
: 3939 2
: 3940 1
    
```

```

                END;
                END;
MAD1 [PL_ADDR] = .CPLAT;
MAD2 [PL_ADDR] = .CPLAT;
                END;
    
```

```

! UNIT IS PRESENT
! CST UNIT LOOP
! LOAD PLATTER ADDRESS (MSCP UNIT NUMBER)
! BOTH MSCP ENVELOPES
! ROUTINE QIO_UNIT
    
```

```

000000 004137 000000G          .SBTTL QIO.UNIT MULTI-DRIVE SUBTEST ROUTINES
                                QIO.UNIT:
000004 013700 000000G          JSR    R1,$SAVE3                ; 3876
000010 116000 000005          MOV    CST.ADDR,R0                ; 3899
000014 042700 177400          MOVB  5(R0),R0
000020 005300                   BIC   #177400,R0
000022 020027 000003          DEC   R0
000026 101003                   CMP   R0,#3
000030 006300                   BHI  2$
000032 066007 000000'        ASL   R0
000036 012703 000001          2$:  ADD  P.AAA(R0),PC                ; Case dispatch
000042 000427                   MOV  #1,R3                ; *,NTH.UNIT 3902
000044 013703 000634'        BR    7$                ; 3899
000050 042703 177776          3$:  MOV  RANDOM,R3                ; *,NTH.UNIT 3903
000054 000421                   BIC  #177776,R3                ; *,NTH.UNIT
000056 013746 000634'        BR    6$
000062 012746 000003          4$:  MOV  RANDOM,-(SP)                ; 3904
000066 004737 000000G        MOV  #3,-(SP)
000072 010016                   JSR  PC,BL$MOD
000074 004737 000000G        MOV  R0,(SP)
000100 010003                   JSR  PC,BL$ABS
000102 005203                   MOV  R0,R3                ; *,NTH.UNIT
000104 022626                   INC  R3                ; NTH.UNIT
000106 000405                   CMP  (SP)+,(SP)+
000110 013703 000634'        BR    7$                ; 3899
000114 042703 177774          5$:  MOV  RANDOM,R3                ; *,NTH.UNIT 3905
000120 005203                   BIC  #177774,R3                ; *,NTH.UNIT
000122 012702 000003          6$:  INC  R3                ; NTH.UNIT
000126 010200                   MOV  #3,R2                ; *,OFFSET 3909
000130 006300                   MOV  R2,R0                ; OFFSET,* 3912
000132 063700 000000G        ASL  R0
000136 032710 040000          ADD  CST.ADDR,R0
000142 001417                   BIT  #40000,(R0)
000144 011001                   BEQ  9$
000146 000301                   MOV  (R0),R1                ; *,UNIT 3916
000150 042701 177740          SWAB R1                ; UNIT
000154 132761 000001 000000G BIC  #177740,R1                ; *,UNIT
000162 001407                   BITB #1,CPT(R1)                ; *,*(UNIT) 3917
000164 005303                   BEQ  9$
000166 001005                   DEC  R3                ; NTH.UNIT 3921
000170 010246                   BNE  9$                ; 3922
000172 004737 000000G        MOV  R2,-(SP)                ; OFFSET,* 3926
000176 005726                   JSR  PC,SET.UPAR
000200 000404                   TST  (SP)+
000202 005202                   BR   10$
000204 020227 000006          9$:  INC  R2                ; OFFSET 3924
                                CMP  R2,#6                ; OFFSET,* 3909
    
```


CZRC03
V02.0

CZRCDB0 RC25 DISK EXERCISER
MULTI-DRIVE SUBTEST ROUTINES

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER\$1:[AZTEC.CZRCDB]CZRC03.SRC;17

SEQ 0289
Page 98
(32)

000210	003746		BLE	8\$		
000212	013700	000606'	MOV	MAD1,RO	:	3937
000216	013760	000000G 000014	MOV	CPLAT,14(RO)		
000224	013700	000610'	MOV	MAD2,RO	:	3938
000230	013760	000000G 000014	MOV	CPLAT,14(RO)		
000236	000207		RTS	PC	:	3876

; Routine Size: 80 words, Routine Base: \$CODE\$ + 10302
; Maximum stack depth per invocation: 7 words

000000 .PSECT \$PLIT\$, RO, D

000000	000000	P.AAA:			; CASE Table for QIO.UNIT+0032	3899
000002	000006	1\$:	.WORD	0	; [2\$]	
000004	000020		.WORD	6	; [3\$]	
000006	000052		.WORD	20	; [4\$]	
			.WORD	52	; [5\$]	

```

: 3941 1  ROUTINE QIO_FUNC : NOVALUE =
: 3942 1
: 3943 1  !+
: 3944 1  ! THIS ROUTINE IS CALLED BY QIO_GEN TO SELECT THE I/O FUNCTION (OPCODE)
: 3945 1  ! TO BE USED FOR THE CURRENT QIO OR QIO PAIR. THE FUNCTION IS DETERMINED
: 3946 1  ! BY THE FOLLWING ALGORITHM:
: 3947 1  !
: 3948 1  !
: 3949 1  !     IF THE CHOSEN UNIT IS PROTECTED
: 3950 1  !     THEN
: 3951 1  !         FUNCTION = READ
: 3952 1  !     ELSE (UNPROTECTED)
: 3953 1  !         IF OPERATOR SELECTED "WRITE-ONLY" OPTION
: 3954 1  !         THEN
: 3955 1  !             FUNCTION = WRITE
: 3956 1  !         ELSE
: 3957 1  !             FUNCTION (WRITE OR READ) IS BASED ON A RANDOM
: 3958 1  !             NUMBER
: 3959 1  !
: 3960 1  !     IN ADDITION, IF THE OPERATOR SELECTED THE OPTION OF PERFORMING WRITE-
: 3961 1  !     COMPARES AT THE HOST, AND IF A "WRITE" FUNCTION WAS CHOSEN ABOVE FOR
: 3962 1  !     THE FIRST QIO, THEN A "READ" OPCODE IS LOADED INTO THE SECOND MSCP
: 3963 1  !     ENVELOPE. OTHERWISE, THE SECOND MSCP ENVELOPE IS RETURNED TO THE POOL.
: 3964 1  !
: 3965 1  !     IMPLICIT INPUTS:
: 3966 1  !         CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
: 3967 1  !         CUOFF - CURRENT UNIT CST OFFSET
: 3968 1  !
: 3969 1  !     IMPLICIT OUTPUTS:
: 3970 1  !         THE OPCODE FIELD OF ONE OR BOTH MSCP ENVELOPES IS LOADED.
: 3971 1  !-
: 3972 2  BEGIN
: 3973 2
: 3974 2  LOCAL
: 3975 2  FUNC : WORD;                                ! OPCODE (READ OR WRITE)
: 3976 2
: 3977 2  IF .CST_ADDR [.CUOFF, P_PROT] EQLU PROTECTED ! IF UNIT IS PROTECTED
: 3978 2  THEN                                           ! THEN
: 3979 2  FUNC = OP_RD                                ! SET FUNCTION TO READ
: 3980 2  ELSE                                         ! OTHERWISE (UNIT IS UNPROTECTED)
: 3981 3  BEGIN
: 3982 3
: 3983 4  IF BIT_TST (SWP_FLAGS, SWF_WO)                ! IF OPERATOR CHOSE WRITE-ONLY OPTION
: 3984 3  THEN                                           ! THEN
: 3985 3  FUNC = OP_WRT                                ! SET FUNCTION TO WRITE
: 3986 3  ELSE                                         ! OTHERWISE
: 3987 4  BEGIN
: 3988 4
: 3989 5  IF (.RANDOM [1] AND 1)                        ! USE 2ND RANDOM NUMBER TO SELECT
: 3990 4  THEN                                           ! EITHER
: 3991 4  FUNC = OP_RD                                ! READ
: 3992 4  ELSE                                         ! OR
: 3993 4  FUNC = OP_WRT;                               ! WRITE
: 3994 4
: 3995 3  END;
: 3996 3
: 3997 2  END;
```



```

: 3998 2
: 3999 2 IF (MAD1 [OPCODE] = .FUNC) EQLU OP_WRT ! LOAD CHOSEN OPCODE. IF WRITE
: 4000 2 THEN ! THEN
: 4001 3 BEGIN
: 4002 3
: 4003 4 IF BIT_TST (SWP_FLAGS, SWF_CWC) ! IF CONTROLLER DOES WRITE-COMPARES
: 4004 3 THEN ! THEN
: 4005 3 MAD1 [MODIFY] = MD_CMP; ! ADD COMPARE MODIFIER
: 4006 3
: 4007 4 IF BIT_TST (SWP_FLAGS, SWF_HWC) ! IF HOST DOES WRITE-COMPARES
: 4008 3 THEN ! THEN
: 4009 4 BEGIN
: 4010 4
: 4011 4 MAD1 [MODIFY] = MD_EXP; ! SET EXPRESS REQUEST MODIFIER
: 4012 4 MAD2 [OPCODE] = OP_RD; ! SET READ OPCODE INTO 2ND MSCP ENVELOPE
: 4013 4 MAD2 [MODIFY] = MD_EXP; ! SET EXPRESS REQUEST MODIFIER
: 4014 4
: 4015 3 END;
: 4016 3
: 4017 3 END
: 4018 2 ELSE ! OTHERWISE - FUNCTION IS READ
: 4019 3 BEGIN
: 4020 3
: 4021 4 IF BIT_TST (SWP_FLAGS, SWF_CRC) ! IF CONTROLLER DOES READ-COMPARES
: 4022 3 THEN ! THEN
: 4023 3 MAD1 [MODIFY] = MD_CMP; ! ADD COMPARE MODIFIER
: 4024 3
: 4025 2 END;
: 4026 2
: 4027 2 IF .MAD2 [OPCODE] EQLU 0 ! IF NO OPCODE IN 2ND ENVELOPE
: 4028 2 THEN ! THEN
: 4029 3 BEGIN
: 4030 3
: 4031 3 PUT_ENV (.MX2); ! RETURN 2ND ENVELOPE TO POOL
: 4032 3 MX2 = -1; ! MARK IT UNUSED
: 4033 3
: 4034 2 END;
: 4035 2
: 4036 1 END; ! ROUTINE QIO_FUNC

```

010542 .SBTTL QIO.FUNC MULTI-DRIVE SUBTEST ROUTINES
.PSECT \$CODE\$, RO

```

000000 004137 000000G QIO.FUNC:
000004 013700 000000G JSR R1,$SAVE2 ; 3941
000010 006300 MOV CUOFF,R0 ; 3977
000012 063700 000000G ASL R0
000016 032710 100000 ADD CST.ADDR,R0
000022 001410 BEQ 1$ ; 3979
000024 132737 000020 000000G BITB #20,SWP.FLAGS ; 3983
000032 001007 BNE 2$ ; 3985
000034 032737 000001 000636' BIT #1,RANDOM+2 ; 3989
000042 001403 BEQ 2$
000044 012702 000041 1$: MOV #41,R2 ; *,FUNC 3991
000050 000402 BR 3$ ; 3989

```

CZRC03
V02.0

CZRCDB0 RC25 DISK EXERCISER
MULTI-DRIVE SUBTEST ROUTINES

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC03.SRC;17

SEQ 0292
Page 101
(33)

000052	012702	000042	2\$:	MOV	#42,R2	; *,FUNC	3993
000056	005001		3\$:	CLR	R1	;	4003
000060	153701	000000G		BISB	SWP.FLAGS,R1	;	
000064	013700	000606'		MOV	MAD1,R0	;	3999
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	;	4003
000106	001403			BEQ	4\$;	
000110	012760	040000 000022		MOV	#40000,22(R0)	;	4005
000116	032701	000100	4\$:	BIT	#100,R1	;	4007
000122	001422			BEQ	6\$;	
000124	012760	100000 000022		MOV	#-100000,22(R0)	;	4011
000132	013700	000610'		MOV	MAD2,R0	;	4012
000136	112760	000041 000020		MOVB	#41,20(R0)	;	
000144	012760	100000 000022		MOV	#-100000,22(R0)	;	4013
000152	000406			BR	6\$;	3999
000154	032701	000010	5\$:	BIT	#10,R1	;	4021
000160	001403			BEQ	6\$;	
000162	012760	040000 000022		MOV	#40000,22(R0)	;	4023
000170	013700	000610'	6\$:	MOV	MAD2,R0	;	4027
000174	105760	000020		TSTB	20(R0)	;	
000200	001010			BNE	7\$;	
000202	013746	000604'		MOV	MX2,-(SP)	;	4031
000206	004737	000000G		JSR	PC,PUT.ENV	;	
000212	012737	177777 000604'		MOV	#-1,MX2	;	4032
000220	005726			TST	(SP)+	;	4029
000222	000207		7\$:	RTS	PC	;	3941

; Routine Size: 74 words, Routine Base: \$CODE\$ + 10542
; Maximum stack depth per invocation: 5 words


```

; 4037 1  ROUTINE QIO_LBN : NOVALUE =
; 4038 1
; 4039 1  !+
; 4040 1  ! THIS ROUTINE IS CALLED BY QIO_GEN TO SELECT THE LOGICAL BLOCK NUMBER TO
; 4041 1  ! BE USED FOR THE CURRENT QIO OR QIO PAIR.
; 4042 1  !
; 4043 1  ! IF THE OPERATOR CHOSE THE RANDOM SEEK MODE OPTION, THEN THE LBN IS
; 4044 1  ! RANDOMLY CHOSEN WITHIN THE SPECIFIED LIMITS FOR TRACK AND SECTOR.
; 4045 1  ! OTHERWISE, THE NEXT SEQUENTIAL LBN IS DERIVED FROM THE BLOCK SEQUENCE
; 4046 1  ! TABLE (BST).
; 4047 1  !
; 4048 1  ! IMPLICIT INPUTS:
; 4049 1  !     L$LUN - CURRENT (DIAGNOSTIC SUPERVIOR) UNIT NUMBER
; 4050 1  !
; 4051 1  ! IMPLICIT OUTPUTS:
; 4052 1  !     THE LBN IS LOADED INTO ONE OR BOTH MSCP ENVELOPES.
; 4053 1  !-
; 4054 1
; 4055 2  BEGIN
; 4056 2
; 4057 2  LOCAL
; 4058 2  T_RANGE : WORD,           ! PERMISSIBLE RANGE OF TRACK NUMBERS
; 4059 2  S_RANGE : WORD,           ! PERMISSIBLE RANGE OF SECTOR NUMBERS
; 4060 2  NTRACK : WORD,          ! SELECTED TRACK NUMBER
; 4061 2  NSECT : WORD,          ! SELECTED SECTOR NUMBER
; 4062 2  LBN : WORD;           ! LOGICAL BLOCK NUMBER
; 4063 2
; 4064 3  IF BIT_TST (SWP_FLAGS, SWF_RDM)           ! IF RANDOM SEEK MODE
; 4065 2  THEN                                     ! THEN
; 4066 3  BEGIN
; 4067 3
; 4068 3  T_RANGE = (.SWP_ETRACK - .SWP_STRACK) + 1;   ! RANGE OF TRACK NUMBERS
; 4069 3  S_RANGE = MAX_SECT + 1;                   ! RANGE OF SECTOR NUMBERS
; 4070 3  NTRACK = .SWP_STRACK + ABS (.RANDOM [2] MOD .T_RANGE); ! RANDOM TRACK NUMBER WITHIN RANGE
; 4071 3  NSECT = ABS (.RANDOM [3] MOD .S_RANGE);     ! RANDOM SECTOR NUMBER WITHIN RANGE
; 4072 3
; 4073 3  END
; 4074 2  ELSE                                     ! ELSE - SEQUENTIAL LBN MODE
; 4075 3  BEGIN
; 4076 3
; 4077 3  NTRACK = .BST [.L$LUN, TRACK];             ! GET TRACK FROM BST
; 4078 3  NSECT = .BST [.L$LUN, SECTOR];            ! GET SECTOR FROM BST
; 4079 3  ADV_BST ();                                ! ADVANCE TO NEXT LBN
; 4080 3
; 4081 2  END;
; 4082 2
; 4083 2  LBN = (.NTRACK * SEC_PER_TRK) + .NSECT;   ! CALCULATE LBN
; 4084 2  MAD1 [LBN_L] = .LBN;                       ! LOAD LBN INTO 1ST ENVELOPE
; 4085 2  IF .MX2 GEQ 0                               ! IF 2 QIOS
; 4086 2  THEN                                         ! THEN
; 4087 2  MAD2 [LBN_L] = .LBN;                       ! LOAD LBN INTO 2ND ENVELOPE
; 4088 2
; 4089 1  END;                                       ! ROUTINE QIO_LBN

```

000000 004137 000000G

```

.SBTTL QIO.LBN MULTI-DRIVE SUBTEST ROUTINES
QIO.LBN:JSR R1,$SAVE2 ;

```

4037

000004	132737	000004	000000G	BITB	#4,SWP.FLAGS	:		
000012	001436			BEQ	1\$:		4064
000014	013700	000000G		MOV	SWP.ETRACK,R0	:		
000020	163700	000000G		SUB	SWP.STRACK,R0	:		4068
000024	005200			INC	R0			
000026	012702	000037		MOV	#37,R2	:	*,S.RANGE	
000032	013746	000640'		MOV	RANDOM+4,-(SP)	:		4069
000036	010046			MOV	R0,-(SP)	:	T.RANGE,*	4070
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	010001			MOV	R0,R1	:	*,NTRACK	
000060	013716	000642'		MOV	RANDOM+6,(SP)	:		
000064	010246			MOV	R2,-(SP)	:	S.RANGE,*	4071
000066	004737	000000G		JSR	PC,BL\$MOD			
000072	010016			MOV	R0,(SP)			
000074	004737	000000G		JSR	PC,BL\$ABS			
000100	010002			MOV	R0,R2	:	*,NSECT	
000102	062706	000006		ADD	#6,SP	:		4066
000106	000412			BR	2\$:		4064
000110	013700	000000G	1\$:	MOV	L\$LUN,R0	:		4077
000114	006300			ASL	R0			
000116	006300			ASL	R0			
000120	016001	000442'		MOV	BST+2(R0),R1	:	*,NTRACK	
000124	016002	000440'		MOV	BST(R0),R2	:	*,NSECT	
000130	004737	000000V		JSR	PC,ADV.BST	:		4078
000134	010146		2\$:	MOV	R1,-(SP)	:	NTRACK,*	4079
000136	012746	000037		MOV	#37,-(SP)	:		4083
000142	004737	000000G		JSR	PC,BL\$MUL			
000146	060200			ADD	R2,R0	:	NSECT,*	
000150	010001			MOV	R0,R1	:	*,LBN	
000152	013700	000606'		MOV	MAD1,R0	:		
000156	010160	000044		MOV	R1,44(R0)	:	LBN,*	4084
000162	005737	000604'		TST	MX2	:		
000166	002404			BLT	3\$:		4085
000170	013700	000610'		MOV	MAD2,R0	:		
000174	010160	000044		MOV	R1,44(R0)	:	LBN,*	4087
000200	022626		3\$:	CMP	(SP)+,(SP)+	:		
000202	000207			RTS	PC	:		4055
						:		4037

; Routine Size: 66 words, Routine Base: \$CODE\$ + 10766
; Maximum stack depth per invocation: 7 words


```

: 4090 1 ROUTINE ADV_BST : NOVALUE =
: 4091 1
: 4092 1 !+
: 4093 1 ! THIS ROUTINE IS CALLED BY QIO_LBN TO ADVANCE THE CURRNET UNIT'S LBN
: 4094 1 ! IN THE BLOCK SEQUENCE TABLE (BST). THIS IS DONE BY INCREMENTING THE
: 4095 1 ! SECTOR NUMBER (AND TRACK NUMBER, IF NECESSARY), WHILE ENSURING THAT
: 4096 1 ! BOTH VALUES REMAIN WITHIN THE PROPER LIMITS.
: 4097 1 !
: 4098 1 ! IMPLICIT INPUTS:
: 4099 1 ! L$LUN - CURRENT (DRS) UNIT NUMBER
: 4100 1 !-
: 4101 1
: 4102 2 BEGIN
: 4103 2
: 4104 2 BST [.L$LUN, SECTOR] = .BST [.L$LUN, SECTOR] + 1; ! INCREMENT SECTOR NUMBER
: 4105 2 IF .BST [.L$LUN, SECTOR] GTRU MAX_SECT ! IF SECTOR IS BEYOND HIGH LIMIT
: 4106 2 THEN ! THEN
: 4107 3 BEGIN
: 4108 3
: 4109 3 BST [.L$LUN, SECTOR] = 0; ! SET SECTOR TO LOW LIMIT
: 4110 3 BST [.L$LUN, TRACK] = .BST [.L$LUN, TRACK] + 1; ! INCREMENT TRACK NUMBER
: 4111 3 IF .BST [.L$LUN, TRACK] GTRU .SWP_ETRACK ! IF TRACK IS BEYOND HIGH LIMIT
: 4112 3 THEN ! THEN
: 4113 3 BST [.L$LUN, TRACK] = .SWP_STRACK; ! SET TRACK TO LOW LIMIT
: 4114 3
: 4115 2 END;
: 4116 2
: 4117 1 END; ! ROUTINE ADV_BST

```

000000	010146		.SBTTL	ADV.BST MULTI-DRIVE SUBTEST ROUTINES	
000002	013701	000000G	ADV.BST:MOV	R1,-(SP)	4090
000006	006301		MOV	L\$LUN,R1	4104
000010	006301		ASL	R1	
000012	012700	000440'	ASL	R1	
000016	060100		MOV	#BST,R0	
000020	005210		ADD	R1,R0	
000022	021027	000036	INC	(R0)	
000026	101412		CMP	(R0),#36	4105
000030	005010		BLOS	1\$	
000032	012700	000442'	CLR	(R0)	4109
000036	060100		MOV	#BST+2,R0	4110
000040	005210		ADD	R1,R0	
000042	021037	000000G	INC	(R0)	
000046	101402		CMP	(R0),SWP.ETRACK	4111
000050	013710	000000G	BLOS	1\$	
000054	012601		MOV	SWP_STRACK,(R0)	4113
000056	000207		1\$: MOV	(SP)+,R1	4090
			RTS	PC	

```

; Routine Size: 24 words, Routine Base: $CODE$ + 11172
; Maximum stack depth per invocation: 2 words

```

```

; 4118 1 ROUTINE QIO_SIZE : NOVALUE =
; 4119 1
; 4120 1 !+
; 4121 1 ! THIS ROUTINE IS CALLED BY QIO_GEN TO SELECT THE I/O TRANSFER BYTE COUNT
; 4122 1 ! TO BE USED FOR THE CURRENT QIO OR QIO PAIR. THE BYTE COUNT IS
; 4123 1 ! DETERMINED BY A RANDOM NUMBER, AND WILL ALWAYS BE AN EVEN NUMBER
; 4124 1 ! BETWEEN 0 AND THE I/O BUFFER SIZE (BUFF_SIZE), INCLUSIVE.
; 4125 1 !
; 4126 1 ! IMPLICIT OUTPUTS:
; 4127 1 ! THE BYTE COUNT IS LOADED INTO ONE OR BOTH MSCP ENVELOPES.
; 4128 1 !-
; 4129 1
; 4130 2 BEGIN
; 4131 2
; 4132 2 LOCAL
; 4133 2 END_BLK : WORD, ! ENDING BLOCK NUMBER FOR CURRENT I/O
; 4134 2 SIZE : WORD; ! BYTE COUNT
; 4135 2
; 4136 2 SIZE = ABS (.RANDOM [4] MOD (.BUFF_SIZE + 1)) AND #0'17776'; ! GET BYTE COUNT FROM RANDOM NUMBER
; 4137 2 END_BLK = .MAD1 [LBN_L] + ((.SIZE - 1) / BLK_SIZE); ! CALCULATE ENDING BLOCK NUMBER
; 4138 3 IF .END_BLK GTRU (.USIZE - 1) ! IF ENDING BLOCK IS LARGER THAN UNIT SIZE
; 4139 2 THEN ! THEN
; 4140 2 SIZE = (.USIZE - .MAD1 [LBN_L]) * BLK_SIZE; ! SCALE DOWN BYTE COUNT
; 4141 2 MAD1 [BC_LO] = .SIZE; ! LOAD SIZE INTO 1ST MSCP ENVELOPE
; 4142 2 IF .MX2 GEQ 0 ! IF 2 QIOS
; 4143 2 THEN ! THEN
; 4144 2 MAD2 [BC_LO] = .SIZE; ! LOAD SIZE INTO 2ND MSCP ENVELOPE
; 4145 2
; 4146 1 END; ! ROUTINE QIO_SIZE

```

Address	Label	Operation	Comments	Line No.
000000	004137	000000G	.SBTTL QIO.SIZE MULTI-DRIVE SUBTEST ROUTINES	
		QIO.SIZE:		
000004	013746	000644'	JSR R1,\$SAVE3	4118
000010	013746	000000G	MOV RANDOM+10,-(SP)	4136
000014	005216		MOV BUFF.SIZE,-(SP)	
000016	004737	000000G	INC (SP)	
000022	010016		JSR PC,BL\$MOD	
000024	004737	000000G	MOV R0,(SP)	
000030	010003		JSR PC,BL\$ABS	
000032	042703	000001	MOV R0,R3	; *.SIZE
000036	013701	000606'	BIC #1,R3	; *.SIZE
000042	010316		MOV MAD1,R1	; SIZE,*
000044	005316		MOV R3,(SP)	
000046	012746	001000	DEC (SP)	
000052	004737	000000G	MOV #1000,-(SP)	
000056	066100	000044	JSR PC,BL\$DIV	
000062	010002		ADD 44(R1),R0	
000064	013700	000616'	MOV R0,R2	; *.END.BLK
000070	005300		MOV USIZE,R0	;
000072	020200		DEC R0	
000074	101410		CMP R2,R0	; END.BLK,*
000076	013700	000616'	BLOS 1\$	
000102	166100	000044	MOV USIZE,R0	;
000106	000300		SUB 44(R1),R0	
000110	105000		SWAB R0	
			CLRB R0	

CZRC03
V02.0

CZRCDB0 RC25 DISK EXERCISER
MULTI-DRIVE SUBTEST ROUTINES

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC03.SRC;17

SEQ 0297
Page 106
(36)

000112	006300		ASL	R0			
000114	010003		MOV	R0,R3		; *,SIZE	
000116	013700	000606'	MOV	MAD1,R0			
000122	010360	000024	MOV	R3,24(R0)		; SIZE,*	4141
000126	005737	000604'	TST	MX2			
000132	002404		BLT	2#			4142
000134	013700	000610'	MOV	MAD2,R0			
000140	010360	000024	MOV	R3,24(R0)		; SIZE,*	4144
000144	062706	000006	ADD	#6,SP			4130
000150	000207		RTS	PC			4118

; Routine Size: 53 words, Routine Base: \$CODE\$ + 11252
; Maximum stack depth per invocation: 8 words

```

: 4147 1  ROUTINE FILL_BUFF : NOVALUE =
: 4148 1
: 4149 1  !*
: 4150 1  ! THIS ROUTINE IS CALLED BY QIO_GEN TO LOAD THE I/O BUFFER DESCRIBED IN
: 4151 1  ! THE FIRST MSCP ENVELOPE WITH THE APPROPRIATE DATA PATTERN.
: 4152 1  !
: 4153 1  ! THE DATA PATTERN TO BE SELECTED IS BASED ON THE FOLLOWING ALGORITHM:
: 4154 1  !
: 4155 1  !     IF THE OPERATOR DEFINED A DATA PATTERN
: 4156 1  !     THEN
: 4157 1  !         SELECT IT
: 4158 1  !     ELSE
: 4159 1  !         GET DATA PATTERN NUMBER FROM SW P-TABLE
: 4160 1  !         IF DATA PATTERN NUMBER = 0
: 4161 1  !         THEN
: 4162 1  !             GET DATA PATTERN NUMBER FROM THE UNIT'S ENTRY
: 4163 1  !             IN THE DATA PATTERN SEQUENCE TABLE (DPST)
: 4164 1  !
: 4165 1  ! NOTE THAT PATTERN # 1 CONSISTS OF RANDOM NUMBERS, AND PATTERNS # 17 -
: 4166 1  ! 21 USE THE ACTUAL LBN OF THE WRITE REQUEST.
: 4167 1  !
: 4168 1  ! IMPLICIT INPUTS:
: 4169 1  !     L$LUN - CURRENT (DRS) UNIT NUMBER
: 4170 1  !-
: 4171 1
: 4172 2  BEGIN
: 4173 2
: 4174 2  LOCAL
: 4175 2      DP_NUM : WORD,           ! DATA PATTERN NUMBER SELECTED
: 4176 2      DP_ADDR,             ! ADDR OF DATA PATTERN (LENGTH)
: 4177 2      IOB_ADDR,           ! I/O BUFFER ADDRESS (DESTINATION)
: 4178 2      SRC_ADDR,           ! WORKING SOURCE ADDRESS
: 4179 2      DISP : WORD,        ! MEM. MGMT. ADDRESS DISPLACEMENT
: 4180 2      PAF : WORD,         ! PAGE ADDRESS FIELD
: 4181 2      COUNT : WORD;       ! NO. OF WORDS IN DATA PATTERN
: 4182 2
: 4183 3  IF BIT_TST (SWP_FLAGS, SWF_UDP) ! IF USER DEFINED A DATA PATTERN
: 4184 2  THEN
: 4185 2      DP_ADDR = SWP_UCNT        ! THEN
: 4186 2  ELSE
: 4187 3  BEGIN
: 4188 3
: 4189 3      IF .SWP_DPAT NEQU 0
: 4190 3  THEN
: 4191 3          DP_NUM = .SWP_DPAT    ! IF USER SELECTED A PRE-DEFINED DATA PATTERN
: 4192 3  ELSE
: 4193 4  BEGIN
: 4194 4
: 4195 4          DP_NUM = .DPST [.L$LUN]; ! GET PATTERN NUMBER FROM SEQUENCE TABLE
: 4196 4          DPST [.L$LUN] = .DPST [.L$LUN] + 1; ! ADVANCE TO NEXT PATTERN NUMBER
: 4197 4          IF .DPST [.L$LUN] GTRU DP_CNT ! CHECK FOR HIGH LIMIT
: 4198 4  THEN
: 4199 4              DPST [.L$LUN] = 1;
: 4200 4
: 4201 3      END;
: 4202 3
: 4203 3      DP_ADDR = .DPA_TBL [.DP_NUM - 1]; ! ADDRESS OF DATA PATTERN (COUNT)
```



```
: 4204 3      IF .DP_NUM GEQU 17
: 4205 3      THEN
: 4206 4          BEGIN
: 4207 4
: 4208 4          IF .DP_NUM
: 4209 4          THEN
: 4210 4              (.DP_ADDR + 2) = .MAD1 [LBN_L]
: 4211 4          ELSE
: 4212 4              (.DP_ADDR + 4) = .MAD1 [LBN_L];
: 4213 4
: 4214 3          END;
: 4215 3
: 4216 2      END;
: 4217 2
: 4218 2      IOB_ADDR = .MAD1 [BUF_0];
: 4219 2      IF .MEM_MGMT
: 4220 2      THEN
: 4221 3          BEGIN
: 4222 3
: 4223 3          DISP = .IOB_ADDR AND %0'17777';
: 4224 3          PAF = (.IOB_ADDR + -6) AND %0'1600';
: 4225 3          KTPAR4 = (.MAD1 [BUF_1] + 10) OR .PAF;
: 4226 3          IOB_ADDR = %0'100000' OR .DISP;
: 4227 3
: 4228 2      END;
: 4229 2
: 4230 2      COUNT = ..DP_ADDR;
: 4231 2      SRC_ADDR = .DP_ADDR + 2;
: 4232 2      INCR N FROM 1 TO (.MAD1 [BC_LO] / 2) DO
: 4233 3          BEGIN
: 4234 3
: 4235 3          .IOB_ADDR = ..SRC_ADDR;
: 4236 3          IOB_ADDR = .IOB_ADDR + 2;
: 4237 3          SRC_ADDR = .SRC_ADDR + 2;
: 4238 3          IF .MEM_MGMT
: 4239 3          THEN
: 4240 4              BEGIN
: 4241 4
: 4242 4                  IF (.IOB_ADDR AND %0'17777') EQLU 0
: 4243 4                  THEN
: 4244 5                      BEGIN
: 4245 5
: 4246 5                          IOB_ADDR = %0'100000';
: 4247 5                          KTPAR4 = .KTPAR4 + %0'200';
: 4248 5
: 4249 4                      END;
: 4250 4
: 4251 3              END;
: 4252 3
: 4253 3          COUNT = .COUNT - 1;
: 4254 3          IF .COUNT EQLU 0
: 4255 3          THEN
: 4256 4              BEGIN
: 4257 4
: 4258 4                  COUNT = ..DP_ADDR;
: 4259 4                  SRC_ADDR = .DP_ADDR + 2;
: 4260 4
```

! IF PATTERN 17, 19, OR 21
! THEN
! LOAD LBN INTO FIRST WORD OF PATTERN
! ELSE - PATTERN 18 OR 20
! LOAD LBN INTO SECOND WORD OF PATTERN

! I/O BUFFER ADDRESS
! IF SYSTEM HAS MEMORY MANAGEMENT
! THEN

! I/O BUFFER DISPLACEMENT ADDRESS
! PAGE ADDRESS FIELD
! LOAD PAR4
! I/O BUFFER VIRTUAL ADDRESS

! NO. OF WORDS IN DATA PATTERN
! START OF THE ACTUAL DATA PATTERN
! FOR EACH WORD IN THIS WRITE REQUEST

! MOVE 1 WORD
! ADVANCE DESTINATION ADDRESS
! ADVANCE SOURCE ADDRESS
! IF SYSTEM HAS MEMORY MANAGEMENT
! THEN

! IF I/O BUFFER CROSSES A 4K PAGE BOUNDARY
! THEN

! RESET VIRTUAL ADDRESS
! ADVANCE PAR TO NEXT 4K PAGE

! END - IF MEMORY MANAGEMENT

! DECREMENT COUNT
! IF END OF DATA PATTERN
! THEN

! REPEAT DATA PATTERN

CZRC03
V02.0

CZRCDB0 RC25 DISK EXERCISER
MULTI-DRIVE SUBTEST ROUTINES

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER\$1:[AZTEC.CZRCDB]CZRC03.SRC;17

SEQ 0301
Page 110
(37)

000214	016100	000032		MOV	32(R1),R0				
000220	000300			SWAB	R0	:			4225
000222	105000			CLRB	R0				
000224	006300			ASL	R0				
000226	006300			ASL	R0				
000230	010337	172350		MOV	R3,#172350	:	PAF,*		
000234	050037	172350		BIS	R0,#172350				
000240	010504			MOV	R5,R4	:	DISP,IOB.ADDR		4226
000242	052704	100000		BIS	#100000,R4	:	*,IOB.ADDR		
000246	022626			CMP	(SP)+,(SP)+	:			4221
000250	011205		6\$:	MOV	(R2),R5	:	DP.ADDR,COUNT		4230
000252	012716	000002		MOV	#2,(SP)	:			4231
000256	060216			ADD	R2,(SP)	:	DP.ADDR,*		
000260	011603			MOV	(SP),R3	:	*,SRC.ADDR		
000262	016146	000024		MOV	24(R1),-(SP)	:			4232
000266	012746	000002		MOV	#2,-(SP)				
000272	004737	000000G		JSR	PC,BL\$DIV				
000276	005001			CLR	R1	:	N		
000300	000422			BR	9\$				
000302	012324		7\$:	MOV	(R3)+,(R4)+	:	SRC.ADDR,IOB.ADDR		4235
000304	032766	000001	000006	BIT	#1,6(SP)	:			4238
000312	001410			BEQ	8\$				
000314	032704	017777		BIT	#17777,R4	:	*,IOB.ADDR		4242
000320	0C1005			BNE	8\$				
000322	012704	100000		MOV	#-100000,R4	:	*,IOB.ADDR		4246
000326	062737	000200	172350	ADD	#200,#172350	:			4247
000334	005305			DEC	R5	:	COUNT		4253
000336	001003			BNE	9\$:			4254
000340	011205			MOV	(R2),R5	:	DP.ADDR,COUNT		4258
000342	016603	000004		MOV	4(SP),R3	:	*,SRC.ADDR		4259
000346	005201			INC	R1	:	N		4232
000350	020100			CMP	R1,R0	:	N,*		
000352	003753			BLE	7\$				
000354	032766	000001	000006	BIT	#1,6(SP)	:			4265
000362	001403			BEQ	10\$				
000364	012737	001000	172350	MOV	#1000,#172350	:			4267
000372	062706	000010		ADD	#10,SP	:			4147
000376	000207			RTS	PC				

: Routine Size: 128 words, Routine Base: \$CODE\$ + 11424
: Maximum stack depth per invocation: 11 words

```

: 4270 1 ROUTINE PROC_RETPKT : NOVALUE =
: 4272 1 !+
: 4273 1 !
: 4274 1 ! THIS ROUTINE IS CALLED FROM THE MULTI_DRIVE "EXECUTIVE" TO CHECK FOR
: 4275 1 ! AND PROCESS ANY RETURN PACKETS THAT HAVE BEEN "SENT" BY THE "DRIVER"
: 4276 1 ! PORTION OF THE PROGRAM. THE I/O DONE QUEUE (IODQ) ACTS AS THE LINK
: 4277 1 ! BETWEEN THE TWO PROGRAM PARTS; IT HOLDS INDECES OF RETURN PACKETS WHICH
: 4278 1 ! REQUIRE PROCESSING.
: 4279 1 !
: 4280 1 ! UNDER THE MULTI-DRIVE SUBTEST, RETURN PACKETS ORIGINATE FROM TWO
: 4281 1 ! SOURCES:
: 4282 1 ! 1. DISK MSCP - THE MORE COMMON, DESCRIBING A COMPLETED I/O
: 4283 1 ! OPERATION
: 4284 1 ! 2. THE PROGRAM "DRIVER" - DESCRIBING A CONTROLLER ERROR OR
: 4285 1 ! COMMAND TIMEOUT.
: 4287 2 BEGIN
: 4289 2 WHILE .IODQ_IN NEQU .IODQ_OUT DO ! DO UNTIL I/O DONE QUEUE IS EMPTY
: 4290 3 BEGIN
: 4292 3 RP_INDX = OUT_IODQ (); ! GET INDEX OF NEXT RETPKT AND ADVANCE OUT POINTER
: 4293 3 RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2); ! CALCULATE RETPKT ADDRESS
: 4294 3 SET_CPAR (.RP_ADDR [CTLR]); ! SET UP CURRENT CONTROLLER PARAMETERS
: 4295 3 SELECTONEU .RP_ADDR [CONID] OF ! CONNECTION ID INDICATES PACKET SOURCE
: 4296 3 SET
: 4297 3 [CID_DISK] : IO_RETPKT (); ! DISK MSCP (I/O TRANSFER DONE)
: 4298 3 [CID_DRIVER] : DR_RETPKT (); ! MESSAGE FROM "DRIVER"
: 4299 3 TES;
: 4301 2 END; ! UNITL I/O DONE QUEUE IS EMPTY
: 4303 1 END; ! ROUTINE PROC_RETPKT

```

Address	Hex	Dec	Op	Operand	Label
000000	023737	000000G 000000G	PROC.RETPKT:		PROC.RETPKT MULTI-DRIVE SUBTEST ROUTINES
000000			1\$: CMP	IODQ.IN,IODQ.OUT	
000006	001444		BEQ	4\$	4289
000010	004737	000000G	JSR	PC,OUT.IODQ	4292
000014	010037	000000G	MOV	RO,RP.INDX	
000020	010046		MOV	RO,-(SP)	: RP.INDX,*
000022	012746	000060	MOV	#60,-(SP)	4293
000026	004737	000000G	JSR	PC,BL\$MUL	
000032	062700	000000G	ADD	#RETPKT,RO	
000036	010037	000000G	MOV	RO,RP.ADDR	
000042	116016	000002	MOVB	2(RO),(SP)	
000046	042716	177760	BIC	#177760,(SP)	4294
000052	004737	000000G	JSR	PC,SET.CPAR	
000056	013700	000000G	MOV	RP.ADDR,RO	
000062	116000	000003	MOVB	3(RO),RO	4295
000066	042700	177400	BIC	#177400,RO	
000072	001003		BNE	2\$	
000074	004737	000000V	JSR	PC,IO.RETPKT	4297
000100	000405		BR	3\$	
000102	020027	000003	2\$: CMP	RO,#3	4295
000106	001002		BNE	3\$	4298
000110	004737	000000V	JSR	PC,DR.RETPKT	
000114	022626		3\$: CMP	(SP)+,(SP)+	
000116	000730		BR	1\$	4290
000120	000207		4\$: RTS	PC	4289
					4270

; Routine Size: 41 words, Routine Base: \$CODE\$ + 12024
; Maximum stack depth per invocation: 3 words


```
ROUTINE IO_RETPKT : NOVALUE =
: 4304 1
: 4305 1
: 4306 1
: 4307 1
: 4308 1
: 4309 1
: 4310 1
: 4311 1
: 4312 1
: 4313 1
: 4314 1
: 4315 1
: 4316 1
: 4317 2
: 4318 2
: 4319 2
: 4320 2
: 4321 2
: 4322 2
: 4323 2
: 4324 2
: 4325 3
: 4326 3
: 4327 3
: 4328 3
: 4329 4
: 4330 4
: 4331 4
: 4332 4
: 4333 4
: 4334 4
: 4335 4
: 4336 3
: 4337 4
: 4338 4
: 4339 4
: 4340 4
: 4341 4
: 4342 4
: 4343 4
: 4344 4
: 4345 4
: 4346 4
: 4347 4
: 4348 4
: 4349 4
: 4350 4
: 4351 4
: 4352 4
: 4353 4
: 4354 4
: 4355 4
: 4356 4
: 4357 4
: 4358 3
: 4359 3
: 4360 3

ROUTINE IO_RETPKT : NOVALUE =
!+
! THIS ROUTINE IS CALLED BY PROC_RETPKT TO HANDLE ALL I/O TRANSFER
! RETURN PACKETS. PROCESSING OF THESE PACKETS INCLUDES DECLARING ANY
! HARD ERRORS THAT MAY HAVE OCCURRED, UPDATING THE STATISTICS, AND
! PERFORMING HOST WRITE-COMPARES IF REQUIRED.
!
! IMPLICIT INPUTS:
! RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
! CCTLR - CURRENT CONTROLLER NUMBER
!-

BEGIN
SWEEP_FLAG = TRUE;          ! INDICATES THAT SWEEP ( ) WILL BE CALLED
FSET_UPAR ( );             ! FIND UNIT'S ENTRY IN CST AND SET UP UNIT-RELATED DATA
ST_CODE = .RP_ADDR [STSCOD]; ! GET STATUS CODE FROM RETPKT
SB_CODE = .RP_ADDR [SUBCOD]; ! GET SUB-CODE, IF ANY
IF .ST_CODE NEQU ST_SUC    ! IF STATUS CODE INDICATES ERROR
THEN                       ! THEN
  BEGIN
  IF .ST_CODE EQLU ST_OFL  ! IF UNIT OFFLINE
  THEN                     ! THEN
    BEGIN
    ERRDF (19, EGD_19, EMS_19); ! "FATAL I/O ERROR"
    DUR [.L$LUN] = DU_FATAL;    ! LOAD REASON FOR DROPPING UNIT
    DODU (.L$LUN);             ! DROP UNIT
    END
  ELSE                       ! ELSE - ERROR IS NOT UNIT-OFFLINE
  BEGIN
  CASE .ST_CODE
  FROM 1 TO 11 OF
  SET
  [1] : ERRHRD (31, EGH_30, EMS_30); ! INVALID COMMAND
  [2] : ERRHRD (32, EGH_30, EMS_30); ! COMMAND ABORTED
  [3] : ; ! UNIT OFFLINE (DEVICE-FATAL)
  [4] : ERRHRD (34, EGH_30, EMS_30); ! UNIT AVAILABLE
  [5] : ERRHRD (35, EGH_30, EMS_30); ! MEDIA FORMAT ERROR
  [6] : ERRHRD (36, EGH_30, EMS_30); ! WRITE-PROTECTED
  [7] : ERRHRD (37, EGH_30, EMS_30); ! DEVICE COMPARE ERROR
  [8] : ERRHRD (38, EGH_30, EMS_30); ! DATA ERROR
  [9] : ERRHRD (39, EGH_30, EMS_30); ! HOST BUFFER ACCESS ERROR
  [10] : ERRHRD (40, EGH_30, EMS_30); ! CONTROLLER ERROR
  [11] : ERRHRD (41, EGH_30, EMS_30); ! DRIVE ERROR
  [OUTRANGE] : ERRHRD (30, EGH_30, EMS_30); ! UNKNOWN
  TES;
  HARD_ERR (1);             ! INCREMENT HARD ERROR COUNT FOR CURRENT UNIT
  END;
END
```

```

: 4361 2 ELSE
: 4362 3 BEGIN
: 4363 3
: 4364 3 MD_TALLY ();
: 4365 4 IF BIT_TST (SWP_FLAGS, SWF_HWC)
: 4366 3 THEN
: 4367 3 HOST_WRT_CHK ();
: 4368 3
: 4369 2 END;
: 4370 2
: 4371 2 IF .SWEEP_FLAG EQLU TRUE
: 4372 2 THEN
: 4373 2 SWEEP ();
: 4374 2 QIO [.CCTLR] = .QIO [.CCTLR] - 1;
: 4375 2
: 4376 1 END;
! ELSE - I/O WAS SUCCESSFUL
! UPDATE I/O STATISTICS
! IF HOST IS DOING WRITE-COMPARES
! THEN
! SAVE I/O PACKET OR DO WRITE-CHECK
! IF SWEEP_FLAG IS STILL TRUE
! THEN
! DEALLOCATE BUFFER(S) AND RETPKT(S)
! DECREMENT NO. OF OUTSTANDING QIOS
! ROUTINE IO_RETPKT

```

```

000000 010146 .SBTTL IO.RETPKT MULTI-DRIVE SUBTEST ROUTINES
000002 112737 000001 000630' MOV R1,-(SP) ; 4304
000010 004737 000000V MOVB #1,SWEEP.FLAG ; 4319
000014 013700 000000G JSR PC,FSET.UPAR ; 4320
000020 116037 000016 000000G MOV RP.ADDR,R0 ; 4321
000026 042737 177740 000000G MOVB 16(R0),ST.CODE
000034 016001 000016 BIC #177740,ST.CODE
000040 006201 MOV 16(R0),R1 ; 4322
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 ; 4323
000066 001521 BEQ 15$ ; 4327
000070 020027 000003 CMP R0,#3 ; 4331
000074 001013 BNE 1$ ; 4332
000076 104455 TRAP 55 ; 4333
000100 000023 .WORD 23 ; 4339
000102 000000G .WORD EGD.19
000104 000000G .WORD EMS.19
000106 013700 000000G MOV L$LUN,R0 ; 4332
000112 112760 000004 000000G MOVB #4,DUR(R0) ; 4333
000120 104451 TRAP 51 ; 4327
000122 000513 BR 16$ ; 4339
000124 005300 1$: DEC R0 ; 4339
000126 020027 000012 CMP R0,#12 ;
000132 101003 BHI 3$
000134 006300 ASL R0
000136 066007 000010' ADD P.AAB(R0),PC ; Case dispatch
000142 104456 3$: TRAP 56 ; 4353
000144 000036 .WORD 36
000146 000000G .WORD EGH.30
000150 000000G .WORD EMS.30
000152 000461 BR 14$ ; 4339
000154 104456 4$: TRAP 56 ; 4342

```


000156	000037		.WORD	37		
000160	000000G		.WORD	EGH.30		
000162	000000G		.WORD	EMS.30		
000164	000454		BR	14\$		
000166	104456	5\$:	TRAP	56	:	4339
000170	000040		.WORD	40	:	4343
000172	000000G		.WORD	EGH.30		
000174	000000G		.WORD	EMS.30		
000176	000447		BR	14\$		
000200	104456	6\$:	TRAP	56	:	4339
000202	000042		.WORD	42	:	4345
000204	000000G		.WORD	EGH.30		
000206	000000G		.WORD	EMS.30		
000210	000442		BR	14\$		
000212	104456	7\$:	TRAP	56	:	4339
000214	000043		.WORD	43	:	4346
000216	000000G		.WORD	EGH.30		
000220	000000G		.WORD	EMS.30		
000222	000435		BR	14\$		
000224	104456	8\$:	TRAP	56	:	4339
000226	000044		.WORD	44	:	4347
000230	000000G		.WORD	EGH.30		
000232	000000G		.WORD	EMS.30		
000234	000430		BR	14\$		
000236	104456	9\$:	TRAP	56	:	4339
000240	000045		.WORD	45	:	4348
000242	000000G		.WORD	EGH.30		
000244	000000G		.WORD	EMS.30		
000246	000423		BR	14\$		
000250	104456	10\$:	TRAP	56	:	4339
000252	000046		.WORD	46	:	4349
000254	000000G		.WORD	EGH.30		
000256	000000G		.WORD	EMS.30		
000260	000416		BR	14\$		
000262	104456	11\$:	TRAP	56	:	4339
000264	000047		.WORD	47	:	4350
000266	000000G		.WORD	EGH.30		
000270	000000G		.WORD	EMS.30		
000272	000411		BR	14\$		
000274	104456	12\$:	TRAP	56	:	4339
000276	000050		.WORD	50	:	4351
000300	000000G		.WORD	EGH.30		
000302	000000G		.WORD	EMS.30		
000304	000404		BR	14\$		
000306	104456	13\$:	TRAP	56	:	4339
000310	000051		.WORD	51	:	4352
000312	000000G		.WORD	EGH.30		
000314	000000G		.WORD	EMS.30		
000316	012746	000001	14\$:	MOV	#1,-(SP)	
000322	004737	000000G		JSR	PC,HARD.ERR	4356
000326	005726			TST	(SP)+	
000330	000410			BR	16\$	4337
000332	004737	000000V	15\$:	JSR	PC,MD.TALLY	4323
000336	132737	000100 000000G		BITB	#100,SWP.FLAGS	4364
000344	001402			BEQ	16\$	4365
000346	004737	000000V		JSR	PC,HOST.WRT.CHK	
000352	123727	000630' 000001	16\$:	CMPB	SWEEP.FLAG,#1	4367
						4371

CZRC03
V02.0

CZRCDB0 RC25 DISK EXERCISER
MULTI-DRIVE SUBTEST ROUTINES

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER\$1:[AZTEC.CZRCDB]CZRC03.SRC;17

SEQ 0306
Page 116
(39)

000360	001002		BNE	17\$		
000362	004737	000000V	JSR	PC,SWEEP	:	4373
000366	013700	000000G	MOV	CCTL,R0	:	4374
000372	105360	000000G	DECB	QIO(R0)		
000376	012601		MOV	(SP)+,R1	:	4304
000400	000207		RTS	PC		

; Routine Size: 129 words, Routine Base: \$CODE\$ + 12146
 ; Maximum stack depth per invocation: 3 words

000010 .PSECT \$PLIT\$, R0 , D

000010	000012	P.AAB:			; CASE Table for IO.RETPKT+0136	4339
		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\$]	


```

; 4377 1  ROUTINE FSET_UPAR : NOVALUE =
; 4378 1
; 4379 1  !+
; 4380 1  !
; 4381 1  ! THIS ROUTINE IS CALLED BY IO_RETPKT AND OTHERS TO SEARCH THE CURRENT
; 4382 1  ! CONTROLLER STATUS TABLE (CST) FOR THE PLATTER ADDRESS WHICH IS
; 4383 1  ! CONTAINED IN THE CURRENT RETURN PACKET. WHEN FOUND, THE OFFSET INTO THE
; 4384 1  ! CST IS USED AS INPUT TO SET_UPAR, WHICH SETS UP CURRENT UNIT-RELATED
; 4385 1  ! DATA PARAMETERS.
; 4386 1  !
; 4387 1  ! IMPLICIT INPUTS:
; 4388 1  ! RP_ADDR - ADDRESS OF CURRENT RETURN PACKET
; 4389 1  ! CST_ADDR - ADDRESS OF CURRENT CONTROLLER'S CST
; 4390 1  !-
; 4391 2  BEGIN
; 4392 2
; 4393 2  INCR OFFSET FROM (0 + OF_UN) TO (3 + OF_UN) DO ! FOR EACH UNIT IN CST
; 4394 3  BEGIN
; 4395 3
; 4396 3  IF .CST_ADDR [.OFFSET, P_ADDR] EQLU .RP_ADDR [PLAT] ! IF RETPKT UNIT NUMBER MATCHES CST ENTRY
; 4397 3  THEN ! THEN
; 4398 4  BEGIN
; 4399 4
; 4400 4  SET_UPAR (.OFFSET); ! SET UP UNIT-RELATED DATA
; 4401 4  RETURN; ! DONE
; 4402 4
; 4403 3  END;
; 4404 3
; 4405 2  END; ! CST UNIT SEARCH LOOP
; 4406 2
; 4407 1  END; ! ROUTINE FSET_UPAR

```

012550

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

```

000000 004137 000000G  FSET.UPAR:
000004 012702 000003  JSR R1,$SAVE3 ; 4377
000010 010201 000003  MOV #3,R2 ; *,OFFSET 4393
000012 006301 000003  1$: MOV R2,R1 ; OFFSET,* 4396
000014 063701 000000G  ASL R1
000020 013700 000000G  ADD CST.ADDR,R1
000024 005003 000000G  MOV RP.ADDR,RO
000026 151103 000010  CLR R3
000030 020360 000010  BISB (R1),R3
000034 001005 000010  CMP R3,10(RO)
000036 010246 000000G  BNE 2$
000040 004737 000000G  MOV R2,-(SP) ; OFFSET,* 4400
000044 005726 000000G  JSR PC,SET.UPAR
000046 000207 000000G  TST (SP)+ ; 4401
000050 005202 000006  2$: RTS PC ; 4398
000052 020227 000006  INC R2 ; OFFSET 4393
000056 003754 000006  CMP R2,#6 ; OFFSET,*
000060 000207 000006  BLE 1$
; 4377

```

; Routine Size: 25 words, Routine Base: \$CODE\$ + 12550
; Maximum stack depth per invocation: 6 words

```

: 4408 1 ROUTINE MD_TALLY : NOVALUE =
: 4409 1
: 4410 1 !+
: 4411 1 ! THIS ROUTINE IS CALLED FROM IO_RETPKT FOR ALL I/O TRANSFER RETURN
: 4412 1 ! PACKETS WITH "SUCCESS" STATUS CODES. ITS PURPOSE IS TO UPDATE ALL THE
: 4413 1 ! APPROPRIATE STATISTICAL FIELDS FOR THE CURRENT UNIT. A CHECK IS ALSO
: 4414 1 ! MADE ON THE TOTAL NUMBER OF BYTES TRANSFERRED THUS FAR; IF THE
: 4415 1 ! OPERATOR-SPECIFIED LIMIT HAS BEEN REACHED, THEN THE UNIT IS REMOVED
: 4416 1 ! FROM CURRENT PASS TESTING.
: 4417 1 !
: 4418 1 ! IMPLICIT INPUTS:
: 4419 1 ! RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
: 4420 1 ! T_ADDR - ADDRESS OF THE CURRENT UNIT'S STATISTICS BLOCK (TALLY)
: 4421 1 ! L$LUN - CURRENT (DRS) UNIT NUMBER
: 4422 1 !-
: 4423 1
: 4424 2 BEGIN
: 4425 2
: 4426 3 IF .RP_ADDR [ENDCOD] EQLU (OP_RD + OP_END) ! IF ENDCODE IS READ
: 4427 2 THEN ! THEN
: 4428 3 BEGIN
: 4429 3
: 4430 3 UPD_IOC (T_ADDR [READ_LO], 1); ! INCREMENT NO. OF READS
: 4431 3 T_ADDR [BR_LO] = .T_ADDR [BR_LO] + .RP_ADDR [BCNT_LO]; ! UPDATE BYTE COUNT
: 4432 3 OVF_CHK (T_ADDR [BR_LO]); ! CHECK FOR FIELD OVERFLOW
: 4433 3
: 4434 3 END
: 4435 2 ELSE ! ELSE ENDCODE IS WRITE
: 4436 3 BEGIN
: 4437 3
: 4438 3 UPD_IOC (T_ADDR [WRIT_LO], 1); ! INCREMENT NO. OF WRITES
: 4439 3 T_ADDR [BW_LO] = .T_ADDR [BW_LO] + .RP_ADDR [BCNT_LO]; ! UPDATE BYTE COUNT
: 4440 3 OVF_CHK (T_ADDR [BW_LO]); ! CHECK FOR FIELD OVERFLOW
: 4441 3
: 4442 2 END;
: 4443 2
: 4444 2 IF .CPT [.L$LUN] ! IF UNIT IS STILL UNDER TEST
: 4445 2 THEN ! THEN
: 4446 2 XFR_CHK (); ! CHECK MBYTES XFR'D AGAINST LIMIT
: 4447 2
: 4448 1 END; ! ROUTINE MD_TALLY

```

		.SBTTL	MD.TALLY MULTI-DRIVE SUBTEST ROUTINES		
000000	010146		MD.TALLY:		
		MOV	R1, -(SP)	:	
000002	013700	000000G	MOV	RP.ADDR, R0	:
000006	126027	000014 000241	CMPB	14(R0), #241	:
000014	001020		BNE	1\$	
000016	013746	000000G	MOV	T.ADDR, -(SP)	:
000022	012746	000001	MOV	#1, -(SP)	:
000026	004737	000000G	JSR	PC, UPD.IOC	
000032	013700	000000G	MOV	T.ADDR, R0	:
000036	013701	000000G	MOV	RP.ADDR, R1	:
000042	066160	000020 000014	ADD	20(R1), 14(R0)	
000050	012716	000014	MOV	#14, (SP)	:
000054	000421		BR	2\$:

CZRC03
V02.0

CZRCDB0 RC25 DISK EXERCISER
MULTI-DRIVE SUBTEST ROUTINES

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC03.SRC:17

SEQ 0309
Page 120
(41)

000056	013746	000000G	1\$:	MOV	T.ADDR,-(SP)	:	4438
000062	052716	000004		ADD	#4,(SP)		
000066	012746	000001		MOV	#1,-(SP)		
000072	004737	000000G		JSR	PC,UPD.IOC		
000076	013700	000000G		MOV	T.ADDR,R0	:	4439
000102	013701	000000G		MOV	RP.ADDR,R1		
000106	066160	000020	000022	ADD	20(R1),22(R0)		
000114	012716	000022		MOV	#22,(SP)	:	4440
000120	060016		2\$:	ADD	R0,(SP)		
000122	004737	000000G		JSR	PC,OVF.CHK		
000126	013700	000000G		MOV	L\$LUN,R0	:	4444
000132	132760	000001	000000G	BITB	#1,CPT(R0)		
000140	001402			BEQ	3\$		
000142	004737	000000G		JSR	PC,XFR.CHK	:	4446
000146	022626		3\$:	CMP	(SP)+,(SP)+	:	4424
000150	012601			MOV	(SP)+,R1	:	4408
000152	000207			RTS	PC		

; Routine Size: 54 words, Routine Base: \$CODE\$ + 12632
; Maximum stack depth per invocation: 4 words

```
; 4449 1 ROUTINE HOST_WRT_CHK : NOVALUE =
; 4451 1 !*
; 4452 1 !
; 4453 1 ! THIS ROUTINE IS CALLED FROM IO_RETPKT FOR ALL I/O TRANSFER RETURN
; 4454 1 ! PACKETS WITH "SUCCESS" STATUS CODES, BUT ONLY IF THE HOST WRITE-COMPARE
; 4455 1 ! OPTION WAS SELECTED BY THE OPERATOR.
; 4456 1 !
; 4457 1 ! IF THE CURRENT RETPKT BEING PROCESSED IS A WRITE FUNCTION, THEN THE
; 4458 1 ! PACKET INDEX (RP_INDX) IS SAVED IN THE CONTROLLER'S RETURN PACKET SAVE
; 4459 1 ! AREA (RP_SAVE). OTHERWISE, THE PACKET IS A READ, SO ITS ASSOCIATED
; 4460 1 ! WRITE PACKET IS REMOVED FROM THE SAVE AREA, AND A WORD-FOR-WORD
; 4461 1 ! COMPARISON IS PERFORMED ON THE TWO I/O BUFFERS. ANY DIFFERENCES
; 4462 1 ! ENCOUNTERED RESULTS IN THE DECLARATION OF A HARD ERROR.
; 4463 1 !
; 4464 1 ! IMPLICIT INPUTS:
; 4465 1 ! RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
; 4466 1 ! RP_INDX - INDEX OF THE CURRENT RETURN PACKET
; 4467 1 ! RPS_X1, RPS_X2 - STARTING / ENDING INDECES OF THE CURRENT
; 4468 1 ! CONTROLLER'S RETPKT SAVE AREA (RP_SAVE)
; 4469 1 !-
; 4470 2 BEGIN
; 4471 2
; 4472 2 LOCAL
; 4473 2 BUFFW, ! ADDR OF I/O BUFFER DESCRIPTOR
; 4474 2 INDEX : SIGNED WORD;
; 4475 2
; 4476 3 IF .RP_ADDR [ENDCOD] EQLU (OP_WRT + OP_END) ! IF WRITE OPERATION
; 4477 2 THEN ! THEN
; 4478 3 BEGIN
; 4479 3
; 4480 3 INCR INDEX FROM .RPS_X1 TO .RPS_X2 DO ! LOOK FOR SPARE ENTRY IN
; 4481 4 BEGIN ! CONTROLLER'S RP_SAVE
; 4482 4
; 4483 4 IF .RP_SAVE [.INDEX] LSS 0 ! IF SPARE SLOT FOUND
; 4484 4 THEN ! THEN
; 4485 5 BEGIN
; 4486 5
; 4487 5 RP_SAVE [.INDEX] = .RP_INDX; ! SAVE INDEX OF WRITE RETPKT
; 4488 5 EXITLOOP; ! DONE
; 4489 5
; 4490 4 END;
; 4492 3 END;
; 4493 3
; 4494 3 SWEEP_FLAG = FALSE; ! DON'T CALL SWEEP FROM IO_RETPKT
; 4495 3
; 4496 3 END
; 4497 2 ELSE ! ELSE ENDCODE IS READ
; 4498 3 BEGIN
; 4499 3
; 4500 3 IF (INDEX = RPS_REM ()) GEQ 0 ! IF ASSOCIATED WRITE PACKET IS FOUND
; 4501 3 THEN ! THEN
; 4502 4 BEGIN
; 4503 4
; 4504 4 BUFFW = RETPKT [.INDEX,.BUFF_0]; ! ADDR OF ADDR OF WRITE I/O BUFFER
; 4505 4 IF CMP_DATA (.BUFFW) EQLU FAILURE ! COMPARE DATA IN BOTH BUFFERS. IF FAILURE
; 4506 4 THEN ! THEN
; 4507 5 BEGIN
```



```

; 4508 5
; 4509 5
; 4510 5
; 4512 4
; 4513 4
; 4514 4
; 4515 4
; 4516 4
; 4517 3
; 4519 2
; 4521 1

```

ERRHRD (42, EGH_30, EMS_42);
HARD_ERR (1);
END;

PUT_IO_BUFF (.BUFFW);
PUT_RETPKT (.INDEX);
END;

END;

END;

! "I/O REQUEST FAILED"
! INCR HARD ERROR STATISTIC
! IF COMPARE ERROR

! RETURN WRITE I/O BUFFER TO POOL
! PUT BACK WRITE RETPKT

! IF ASSOCIATED WRITE RETPKT WAS FOUND
! IF ENCODED WAS READ
! ROUTINE HOST_WRT_CHK

```

000000 004137 000000G .SBTTL HOST.WRT.CHK MULTI-DRIVE SUBTEST ROUTINES
                                HOST.WRT.CHK:
000004 013700 000000G JSR R1,$SAVE2 ; 4449
000010 126027 000014 000242 MOV RP.ADDR,R0 ; 4476
000016 001022 CMPB 14(R0),#242
000020 013700 000000G BNE 4$
000024 005300 MOV RPS.X1,R0 ; *,INDEX 4480
000026 000407 DEC R0 ; INDEX
000030 105760 000000G BR 2$
000034 002004 1$: TSTB RP.SAVE(R0) ; *(INDEX) 4483
000036 113760 000000G 000000G BGE 2$
000044 000404 MOVB RP.INDX,RP.SAVE(R0) ; *,*(INDEX) 4487
000046 005200 BR 3$ ; 4485
000050 020037 000000G INC R0 ; INDEX 4480
000054 003765 CMP R0,RPS.X2 ; INDEX,*
000056 105037 000630' BLE 1$
000062 000207 3$: CLRB SWEEP.FLAG ; 4494
000064 004737 000000V RTS PC ; 4476
000070 010002 4$: JSR PC,RPS.REM ; 4500
000072 002434 MOV R0,R2 ; *,INDEX
000074 010246 BLT 6$
000076 012746 000060 MOV R2,-(SP) ; INDEX,* 4504
000102 004737 000000G MOV #60,-(SP)
000106 062700 000024G JSR PC,BL#MUL
000112 010001 ADD #RETPKT+24,R0
000114 010116 MOV R0,R1 ; *,BUFFW
000116 004737 000000V JSR PC,CMP.DATA ; BUFFW,* 4505
000122 005700 TST R0
000124 001010 BNE 5$
000126 104456 TRAP 56 ; 4509
000130 000052 .WORD 52
000132 000000G .WORD EGH.30
000134 000000G .WORD EMS.42
000136 012716 000001 MOV #1,(SP) ; 4510
000142 004737 000000G JSR PC,HARD.ERR
000146 010116 5$: MOV R1,(SP) ; BUFFW,* 4514
000150 004737 000000G JSR PC,PUT.IO.BUFF
000154 010216 MOV R2,(SP) ; INDEX,* 4515
000156 004737 000000G JSR PC,PUT.RETPKT
000162 022626 CMP (SP),*(SP). ; 4502
000164 000207 6$: RTS PC ; 4449

```

; Routine Size: 59 words, Routine Base: \$CODE\$ + 13006
; Maximum stack depth per invocation: 7 words

```

: 4522 1 ROUTINE CMP_DATA (BUFFW) =
: 4523 1
: 4524 1 !*
: 4525 1 ! THIS ROUTINE IS CALLED BY MOST_WRT_CHK TO PERFORM THE ACTUAL WORD-BY-
: 4526 1 ! WORD COMPARISON BETWEEN THE READ BUFFER DESCRIBED IN THE CURRENT RETURN
: 4527 1 ! PACKET AND ITS ASSOCIATED WRITE BUFFER. A VALUE OF "SUCCESS" IS
: 4528 1 ! RETURNED IF THE TWO BUFFERS ARE IDENTICAL. OTHERWISE, "FAILURE" IS
: 4529 1 ! RETURNED.
: 4530 1 !
: 4531 1 ! INPUTS:
: 4532 1 ! BUFFW - ADDRESS OF THE WRITE BUFFER DESCRIPTOR
: 4533 1 !
: 4534 1 ! IMPLICIT INPUTS:
: 4535 1 ! RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET (READ OPERATION)
: 4536 1 !-
: 4537 1
: 4538 2 BEGIN
: 4539 2
: 4540 2 LOCAL
: 4541 2 RESULT : WORD, ! SUCCESS / FAILURE
: 4542 2 WBD0 : WORD, ! WRITE BUFFER DESCRIPTOR (WORD 0)
: 4543 2 WBD1 : WORD, ! WRITE BUFFER DESCRIPTOR (WORD 1)
: 4544 2 RBDO : WORD, ! READ BUFFER DESCRIPTOR (WORD 0)
: 4545 2 RBD1 : WORD, ! READ BUFFER DESCRIPTOR (WORD 1)
: 4546 2 DISP : WORD, ! 4K PAGE DISPLACEMENT ADDR
: 4547 2 PAF : WORD, ! PAGE ADDRESS FIELD
: 4548 2 COUNT : WORD; ! BUFFER WORD COUNT
: 4549 2
: 4550 2 RESULT = SUCCESS; ! ASSUME BLUE SKY
: 4551 2 WBD0 = .BUFFW; ! WRITE BUFFER DESCRIPTOR (WORD 0)
: 4552 2 WBD1 = (.BUFFW + 2); ! WRITE BUFFER DESCRIPTOR (WORD 1)
: 4553 2 RBDO = .RP_ADDR [BUFF_0]; ! READ BUFFER DESCRIPTOR (WORD 0)
: 4554 2 RBD1 = .RP_ADDR [BUFF_1]; ! READ BUFFER DESCRIPTOR (WORD 1)
: 4555 2 IF .MEM_MGMT ! IF SYSTEM HAS MEMORY MANAGEMENT
: 4556 2 THEN ! THEN
: 4557 3 BEGIN
: 4558 3
: 4559 3 DISP = .WBD0 AND %0'17777'; ! WRITE BUFFER DISPLACEMENT
: 4560 3 PAF = (.WBD0 + -6) AND %0'1600'; ! PAGE ADDRESS FIELD
: 4561 3 KTPAR4 = (.WBD1 + 10) OR .PAF; ! LOAD PAR4
: 4562 3 WBD0 = %0'100000' OR .DISP; ! WRITE BUFFER VIRTUAL ADDRESS
: 4563 3
: 4564 3 DISP = .RBDO AND %0'17777'; ! READ BUFFER DISPLACEMENT
: 4565 3 PAF = (.RBDO + -6) AND %0'1600'; ! PAGE ADDRESS FIELD
: 4566 3 KTPAR5 = (.RBD1 + 10) OR .PAF; ! LOAD PAR5
: 4567 3 RBDO = %0'120000' OR .DISP; ! READ BUFFER VIRTUAL ADDRESS
: 4568 3
: 4569 2 END;
: 4570 2
: 4571 2 IF (COUNT = .RP_ADDR [BCNT_LO] / 2) NEQU 0 ! IF WORD COUNT IS NON-ZERO
: 4572 2 THEN ! THEN
: 4573 3 BEGIN
: 4574 3
: 4575 3 INCR I FROM 1 TO .COUNT DO ! FOR EACH WORD IN BUFFERS
: 4576 4 BEGIN
: 4577 4
: 4578 4 IF ..WBD0 EQLU ..RBDO ! IF WORDS COMPARE O.K.

```



```

: 4579 4          THEN          ! THEN
: 4580 5          BEGIN
: 4581 5
: 4582 5          WBD0 = .WBD0 + 2;          ! ADVANCE WRITE BUFFER ADDRESS
: 4583 5          RBD0 = .RBD0 + 2;          ! ADVANCE READ BUFFER ADDRESS
: 4584 5          IF .MEM_MGMT          ! IF SYSTEM HAS MEMORY MANAGEMENT
: 4585 5          THEN          ! THEN
: 4586 6          BEGIN
: 4587 6
: 4588 6          IF (.WBD0 AND %0'17777') EQLU 0 ! IF WRITE BUFFER CROSSES 4K PAGE BOUNDARY
: 4589 6          THEN          ! THEN
: 4590 7          BEGIN
: 4591 7
: 4592 7          WBD0 = %0'100000';          ! RESET VIRTUAL ADDRESS
: 4593 7          KTPAR4 = .KTPAR4 + %0'200'; ! ADVANCE PAR TO NEXT 4K PAGE
: 4594 7
: 4595 6          END;
: 4596 6
: 4597 6          IF (.RBD0 AND %0'17777') EQLU 0 ! IF READ BUFFER CROSSES 4K PAGE BOUNDARY
: 4598 6          THEN          ! THEN
: 4599 7          BEGIN
: 4600 7
: 4601 7          RBD0 = %0'120000';          ! RESET VIRTUAL ADDRESS
: 4602 7          KTPAR5 = .KTPAR5 + %0'200'; ! ADVANCE PAR TO NEXT 4K PAGE
: 4603 7
: 4604 6          END;
: 4605 6
: 4606 5          END;          ! END - IF MEMORY MANAGEMENT
: 4607 5
: 4608 5          END
: 4609 4          ELSE          ! OTHERWISE - COMPARE ERROR
: 4610 5          BEGIN
: 4611 5
: 4612 5          RESULT = FAILURE;          ! FAILED
: 4613 5          EXITLOOP;          ! NO NEED TO CONTINUE
: 4614 5
: 4615 4          END;
: 4616 4
: 4617 3          END;          ! WORD COMPARE LOOP
: 4618 3
: 4619 2          END;          ! IF WORD COUNT IS NON-ZERO
: 4620 2
: 4621 2          IF .MEM_MGMT          ! IF SYSTEM HAS MEMORY MANAGEMENT
: 4622 2          THEN          ! THEN
: 4623 3          BEGIN
: 4624 3
: 4625 3          KTPAR4 = %0'1000';          ! RESTORE PAR'S
: 4626 3          KTPAR5 = %0'1200';
: 4627 3
: 4628 2          END;
: 4629 2
: 4630 2          RETURN .RESULT;
: 4631 2
: 4632 1          END;          ! ROUTINE CMP_DATA

```

000000	004137	000000G	CMP.DATA:			
000004	012746	000001	JSR	R1,\$SAVE5	:	4522
000010	017604	000020	MOV	#1,-(SP)	: *,RESULT	4550
000014	016600	000020	MOV	@20(SP),R4	: BUFFW,WBDO	4551
000020	016046	000002	MOV	20(SP),R0	: BUFFW,*	4552
000024	013701	000000G	MOV	2(R0),-(SP)	: *,WBD1	
000030	016105	000024	MOV	RP.ADDR,R1	:	4553
000034	016146	000026	MOV	24(R1),R5	: *,RBDO	
000040	005046		MOV	26(R1),-(SP)	: *,RBD1	4554
000042	113716	000000G	CLR	-(SP)	:	4555
000046	032716	000001	MOV	MEM.MGMT,(SP)		
000052	001462		BIT	#1,(SP)		
000054	010402		BEQ	1\$		
000056	042702	160000	MOV	R4,R2	: WBDO,DISP	4559
000062	010446		BIC	#160000,R2	: *,DISP	
000064	012746	177772	MOV	R4,-(SP)	: WBDO,*	4560
000070	004737	000000G	MOV	#-6,-(SP)		
000074	010003		JSR	PC,BL\$SHF		
000076	042703	176177	MOV	R0,R3	: *,PAF	
000102	016600	000010	BIC	#176177,R3	: *,PAF	
000106	000300		MOV	10(SP),R0	: WBD1,*	4561
000110	105000		SWAB	R0		
000112	006300		CLRB	R0		
000114	006300		ASL	R0		
000116	010337	172350	ASL	R0		
000122	050037	172350	MOV	R3,@#172350	: PAF,*	
000126	010204		BIS	R0,@#172350		
000130	052704	100000	MOV	R2,R4	: DISP,WBDO	4562
000134	010502		BIS	#100000,R4	: *,WBDO	
000136	042702	160000	MOV	R5,R2	: RBDO,DISP	4564
000142	010516		BIC	#160000,R2	: *,DISP	
000144	012746	177772	MOV	R5,(SP)	: RBDO,*	4565
000150	004737	000000G	MOV	#-6,-(SP)		
000154	010003		JSR	PC,BL\$SHF		
000156	042703	176177	MOV	R0,R3	: *,PAF	
000162	016600	000010	BIC	#176177,R3	: *,PAF	
000166	000300		MOV	10(SP),R0	: RBD1,*	4566
000170	105000		SWAB	R0		
000172	006300		CLRB	R0		
000174	006300		ASL	R0		
000176	010337	172352	ASL	R0		
000202	050037	172352	MOV	R3,@#172352	: PAF,*	
000206	010205		BIS	R0,@#172352		
000210	052705	120000	MOV	R2,R5	: DISP,RBDO	4567
000214	062706	000006	BIS	#120000,R5	: *,RBDO	
000220	016146	000020	ADD	#6,SP	:	4557
000224	012746	000002	MOV	20(R1),-(SP)	:	4571
000230	004737	000000G	MOV	#2,-(SP)		
000234	022626		JSR	PC,BL\$DIV		
000236	005700		CMP	(SP)+,(SP)+		
000240	001442		TST	R0	: COUNT	
000242	005001		BEQ	6\$		
000244	000435		CLR	R1	: I	4575
000246	021415		BR	5\$		
000250	001030		2\$: CMP	(R4),(R5)	: WBDO,RBDO	4578
000252	062704	000002	BNE	4\$		
			ADD	#2,R4	: *,WBDO	4582

CZRC03
V02.0

CZRCDB0 RC25 DISK EXERCISER
MULTI-DRIVE SUBTEST ROUTINES

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER\$1:[AZTEC.CZRCDB]CZRC03.SRC;17

SEQ 0315
Page 127
(43)

000256	062705	000002		ADD	#2,R5				
000262	032716	000001		BIT	#1,(SP)		; *,RBDO		4583
000266	001424			BEQ	5\$				4584
000270	032704	017777		BIT	#17777,R4		; *,WBDO		4588
000274	001005			BNE	3\$				
000276	012704	100000		MOV	#-100000,R4		; *,WBDO		4592
000302	062737	000200	172350	ADD	#200,@#172350				4593
000310	032705	017777		BIT	#17777,R5		; *,RBDO		4597
000314	001011			BNE	5\$				
000316	012705	120000		MOV	#-60000,R5		; *,RBDO		4601
000322	062737	000200	172352	ADD	#200,@#172352				4602
000330	000403			BR	5\$				4578
000332	005066	000006		4\$: CLR	6(SP)		; RESULT		4612
000336	000403			BR	6\$				4610
000340	005201			5\$: INC	R1		; I		4575
000342	020100			CMP	R1,R0		; I,COUNT		
000344	003740			BLE	2\$				
000346	032716	000001		6\$: BIT	#1,(SP)				4621
000352	001406			BEQ	7\$				
000354	012737	001000	172350	MOV	#1000,@#172350				4625
000362	012737	001200	172352	MOV	#1200,@#172352				4626
000370	016600	000006		7\$: MOV	6(SP),R0		; RESULT,*		4538
000374	062706	000010		ADD	#10,SP				4522
000400	0C0207			RTS	PC				

; Routine Size: 129 words, Routine Base: \$CODE\$ + 13174
; Maximum stack depth per invocation: 14 words

```

: 4633 1 ROUTINE SWEEP : NOVALUE =
: 4634 1
: 4635 1 !*
: 4636 1 ! THIS ROUTINE IS CALLED FROM IO_RETPKT AND OTHERS TO DEALLOCATE THE
: 4637 1 ! RESOURCES ASSOCIATED WITH THE CURRENT RETURN PACKET. THIS INCLUDES THE
: 4638 1 ! PACKET ITSELF AND THE I/O BUFFER. IN ADDITION, IF THE HOST IS
: 4639 1 ! PERFORMING WRITE-COMPARES, AND IF THE CURRENT RETURN PACKET IS A READ
: 4640 1 ! FUNCTION, THEN THE CURRENT CONTROLLER'S RP_SAVE AREA IS SEARCHED FOR
: 4641 1 ! THE ASSOCIATED WRITE RETPKT SO THAT ITS RESOURCES CAN ALSO BE
: 4642 1 ! DEALLOCATED.
: 4643 1 !
: 4644 1 ! IMPLICIT INPUTS:
: 4645 1 ! RP_ADDR - ADDRESS OF CURRENT RETURN PACKET
: 4646 1 ! RP_INDX - INDEX OF CURRENT RETURN PACKET
: 4647 1 !-
: 4648 1
: 4649 2 BEGIN
: 4650 2
: 4651 2 LOCAL
: 4652 2 INDEX : SIGNED WORD;
: 4653 2
: 4654 2 IF (.RP_ADDR [ENDCOD] AND OP_MSK) EQLU OP_RD ! IF READ OPCODE OR ENDCODE
: 4655 2 THEN ! THEN
: 4656 3 BEGIN
: 4657 3
: 4658 4 IF BIT_TST (SWP_FLAGS, SWF_HWC) ! IF HOST IS DOING WRITE-COMPARES
: 4659 3 THEN ! THEN
: 4660 4 BEGIN
: 4661 4
: 4662 4 IF (INDEX = RPS_REM ()) GEQ 0 ! IF ASSOCIATED WRITE RETPKT IS FOUND
: 4663 4 THEN ! THEN
: 4664 5 BEGIN
: 4665 5
: 4666 5 PUT_IO_BUFF (RETPKT [.INDEX, BUFF_0]); ! RETURN WRITE I/O BUFFER TO POOL
: 4667 5 PUT_RETPKT (.INDEX); ! RETURN WRITE PACKET TO POOL
: 4668 5
: 4669 4 END;
: 4670 4
: 4671 3 END;
: 4672 3
: 4673 2 END;
: 4674 2
: 4675 2 PUT_IO_BUFF (RP_ADDR [BUFF_0]); ! RETURN CURRENT I/O BUFFER TO POOL
: 4676 2 PUT_RETPKT (.RP_INDX); ! RETURN CURRENT RETPKT TO POOL
: 4677 2
: 4678 1 END; ! ROUTINE SWEEP?

```

```

000000 010146          SWEEP: .SBTTL SWEEP MULTI-DRIVE SUBTEST ROUTINES
000002 013700 000000G      MOV      R1, -(SP) ; 4633
000006 116000 000014      MOV      RP_ADDR, R0 ; 4654
000012 042700 177600      MOVB    14(R0), R0
000016 020027 000041      BIC     #177600, R0
000022 001026          CMP     R0, #41
000024 132737 000100 000000G    BNE     1$
000032 001422          BITB    #100, SWP_FLAGS ; 4658
                        BEQ     1$

```


CZRC03
V02.0

CZRCDB0 RC25 DISK EXERCISER
MULTI-DRIVE SUBTEST ROUTINES

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER\$1:[AZTEC.CZRCDB]CZRC03.SRC;17

SEQ 0317
Page 129
(44)

000034	004737	000000V	JSR	PC,RPS.REM	:	4662
000040	010001		MOV	R0,R1	: *,INDEX	
000042	002416		BLT	1\$		
000044	010146		MOV	R1,-(SP)	: INDEX,*	4666
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,*	4667
000072	004737	000000G	JSR	PC,PUT.RETPKT		
000076	022626		CMP	(SP)+,(SP)+	:	4664
000100	013746	000000G	MOV	RP.ADDR,-(SP)	:	4675
000104	062716	000024	ADD	#24,(SP)		
000110	004737	000000G	JSR	PC,PUT.IO.BUFF		
000114	013716	000000G	MOV	RP.INDX,(SP)	:	4676
000120	004737	000000G	JSR	PC,PUT.RETPKT		
000124	005726		TST	(SP)+	:	4649
000126	012601		MOV	(SP)+,R1	:	4633
000130	000207		RTS	PC		

; Routine Size: 45 words, Routine Base: \$CODE\$ + 13576
; Maximum stack depth per invocation: 4 words

```

; 4679 1  ROUTINE RPS_REM =
; 4680 1
; 4681 1  !+
; 4682 1  !
; 4683 1  ! THIS ROUTINE SEARCHES THE CURRENT CONTROLLER'S RP_SAVE AREA FOR A
; 4684 1  ! RETURN PACKET WHOSE COMMAND REFERENCE NUMBER (CRN) IS ONE LESS THAN THE
; 4685 1  ! CRN OF THE CURRENT RETURN PACKET (I.E., SEARCHING FOR THE SAVED WRITE
; 4686 1  ! OPERATION ASSOCIATED WITH THE CURRENT READ OPERATION). IF FOUND, THE
; 4687 1  ! RP_SAVE ENTRY IS CLEARED (TO -1) AND THE RETPKT INDEX OF THE WRITE
; 4688 1  ! OPERATION IS RETURNED TO THE CALLER.
; 4689 1  !
; 4690 1  ! IMPLICIT INPUTS:
; 4691 1  ! RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
; 4692 1  ! RPS_X1, RPS_X2 - STARTING / ENDING INDECES OF THE CURRENT
; 4693 1  ! CONTROLLER'S RP_SAVE AREA
; 4694 1  !
; 4695 1  ! OUTPUTS:
; 4696 1  ! INDEX (VALUE OF THIS ROUTINE) - INDEX OF THE RETPKT CONTAINING
; 4697 1  ! A CRN WHICH IS ONE LESS THAN THE CURRENT
; 4698 1  !-
; 4699 2  BEGIN
; 4700 2
; 4701 2  LOCAL
; 4702 2  INDEX : SIGNED WORD,
; 4703 2  TEMP : SIGNED WORD;
; 4704 2
; 4705 2  INDEX = -1; ! ASSUME NOT FOUND
; 4706 2  INCR COUNT FROM .RPS_X1 TO .RPS_X2 DO ! FOR EACH ENTRY IN RP_SAVE
; 4707 3  BEGIN
; 4708 3
; 4709 3  IF (TEMP = .RP_SAVE [.COUNT]) GEQ 0 ! IF THIS IS A VALID RETPKT INDEX
; 4710 3  THEN ! THEN
; 4711 4  BEGIN
; 4712 4
; 4713 5  IF .RETPKT [.TEMP, CRF_LO] EQLU (.RP_ADDR [CRF_LO] - 1) ! IF CORRECT CRN
; 4714 4  THEN ! THEN
; 4715 5  BEGIN
; 4716 5
; 4717 5  INDEX = .TEMP; ! INDEX TO BE RETURNED
; 4718 5  RP_SAVE [.COUNT] = -1; ! INIT ENTRY
; 4719 5  EXITLOOP; ! DONE
; 4720 5
; 4721 4  END;
; 4722 4
; 4723 3  END; ! IF VALID RETPKT INDEX
; 4724 3
; 4725 2  END; ! RP_SAVE ENTRY LOOP
; 4726 2
; 4727 2  RETURN .INDEX;
; 4728 2
; 4729 1  END; ! ROUTINE RPS_REM

```

000000	004137	000000G	.SBTTL	RPS.REM MULTI-DRIVE SUBTEST ROUTINES	
000004	012704	177777	RPS.REM:JSR	R1,\$SAVE4	4679
000010	013701	000000G	MOV	#-1,R4	4705
			MOV	RPS.X1,R1	4706


```

: 4730 1 ROUTINE DR_RETPKT : NOVALUE =
: 4731 1
: 4732 1 !+
: 4733 1 ! THIS ROUTINE IS CALLED BY PROC_RETPKT FOR ALL PACKETS ORIGINATING AT
: 4734 1 ! THE "DRIVER" PORTION OF THE PROGRAM. THIS INCLUDES PACKETS DESCRIBING
: 4735 1 ! FATAL DEVICE ERRORS AND INDIVIDUAL COMMAND TIMEOUTS.
: 4736 1 !
: 4737 1 ! FOR FATAL DEVICE ERRORS, THIS ROUTINE RELEASES ALL RESOURCES HELD BY
: 4738 1 ! THE CONTROLLER. THE CONTROLLER IS MARKED OFFLINE IN ITS CST, AND ALL
: 4739 1 ! UNITS ATTACHED TO THE CONTROLLER ARE DROPPED. IF THE RETURN PACKET
: 4740 1 ! DESCRIBES A COMMAND TIMEOUT, THEN A HARD ERROR IS DECLARED FOR THE
: 4741 1 ! APPROPRIATE UNIT.
: 4742 1 !
: 4743 1 ! IMPLICIT INPUTS:
: 4744 1 ! RP_INDX - INDEX OF THE CURRENT RETURN PACKET
: 4745 1 ! RP_ADDR - ADDRESS OF THE CURRENT RETURN PACKET
: 4746 1 ! RPS_X1 - STARTING / ENDING INDECES OF THE CURRENT CONTROLLER'S
: 4747 1 ! RETPKT SAVE (RP_SAVE) AREA
: 4748 1 ! CST_ADDR - ADDRESS OF THE CURRENT CONTROLLER'S CST
: 4749 1 ! CCTLR - CURRENT CONTROLLER NUMBER
: 4750 1 !-
: 4751 1
: 4752 2 BEGIN
: 4753 2
: 4754 2 LOCAL
: 4755 2 INDEX : SIGNED WORD;
: 4756 2
: 4757 2 IF .RP_ADDR [MESTYP] EQLU MT_FATAL ! IF FATAL CONTROLLER ERROR
: 4758 2 THEN ! THEN
: 4759 3 BEGIN
: 4760 3
: 4761 3 PUTA_BUFF (); ! RELEASE ALL I/O BUFFERS HELD BY CONTROLLER
: 4762 3 INCR COUNT FROM .RPS_X1 TO .RPS_X2 DO ! FOR EACH ENTRY IN CONTROLLER'S RP_SAVE
: 4763 4 BEGIN
: 4764 4
: 4765 4 IF (INDEX = .RP_SAVE [.COUNT]) GEQ 0 ! IF VALID RETPKT INDEX
: 4766 4 THEN ! THEN
: 4767 5 BEGIN
: 4768 5
: 4769 5 PUT_RETPKT (.INDEX); ! RETURN RETPKT TO POOL
: 4770 5 RP_SAVE [.COUNT] = -1; ! INIT ENTRY
: 4771 5
: 4772 4 END;
: 4773 4
: 4774 3 END; ! RP_SAVE ENTRY LOOP
: 4775 3
: 4776 3 QIO [.CCTLR] = 0; ! CLEAR NO. OF OUTSTANDING QIOS
: 4777 3 CST_ADDR [STATE] = OFFLINE; ! MARK CST OFFLINE
: 4778 3 DROP_CTLR (.CCTLR, DU_FATAL); ! DROP CONTROLLER'S UNITS
: 4779 3 PUT_RETPKT (.RP_INDX); ! PUT BACK RETPKT
: 4780 3
: 4781 3 END
: 4782 2 ELSE ! ELSE - COMMAND TIMEOUT
: 4783 3 BEGIN
: 4784 3
: 4785 3 FSET_UPAR (); ! SET UP UNIT-RELATED PARAMETERS
: 4786 3 ERRHRD (43, EGH_30, EMS_43); ! "I/O REQUEST FAILED"
```



```

: 4787 3      HARD_ERR (1);
: 4788 3      SWEEP ();
: 4789 3      QIO [.CCTLR] = .QIO [.CCTLR] - 1;
: 4790 3
: 4791 2      END;
: 4792 2
: 4793 1      END;

```

```

! INCREMENT HARD ERROR STAT FOR UNIT
! RETURN RESOURCES
! DECREMENT NO. OF OUTSTANDING QIOS

```

! ROUTINE DR_RETPKT

```

000000 004137 000000G      .SBTTL DR.RETPKT MULTI-DRIVE SUBTEST ROUTINES
                                DR.RETPKT:
000004 013700 000000G      JSR    R1,$SAVE3                ; 4730
000010 116000 000002      MOV    RP,ADDR,R0                ; 4757
000014 042700 177417      MOVB  2(R0),R0
000020 020027 000060      BIC   #177417,R0
000024 001051              CMP    R0,#60
000026 004737 000000G      BNE   3$
000032 013703 000000G      JSR   PC,PUTA.BUFF              ; 4761
000036 013701 000000G      MOV   RPS.X2,R3                ; 4762
000042 005301              MOV   RPS.X1,R1                ; *,COUNT
000044 000412              DEC   R1                       ; COUNT
000046 116102 000000G      BR    2$
000052 002407 000000G      1$:  MOVB  RP.SAVE(R1),R2        ; *(COUNT),INDEX 4765
000054 010246              BLT  2$
000056 004737 000000G      MOV   R2,-(SP)                 ; INDEX,* 4769
000062 112761 000377 000000G  JSR   PC,PUT.RETPKT
000070 005726 000377 000000G  MOVB  #377,RP.SAVE(R1)         ; *,*(COUNT) 4770
000072 005201              TST  (SP)+                      ; 4767
000074 020103              2$:  INC  R1                       ; COUNT 4762
000076 003763              CMP  R1,R3                      ; COUNT,*
000100 013701 000000G      BLE  1$
000104 105061 000000G      MOV  CCTLR,R1                  ; 4776
000110 013700 000000G      CLRB QIO(R1)                   ; 4777
000114 042760 100000 000002  MOV  CST.ADDR,R0                ; 4778
000122 010146              BIC  #100000,2(R0)
000124 012746 000004      MOV  R1,-(SP)                  ; 4779
000130 004737 000000G      MOV  #4,-(SP)
000134 013716 000000G      JSR  PC,DROP.CTLR
000140 004737 000000G      MOV  RP,INDX,(SP)              ; 4759
000144 005726              JSR  PC,PUT.RETPKT
000146 000420              TST  (SP)+                      ; 4757
000150 004737 012550'      BR   4$                        ; 4785
000154 104456              3$:  JSR  PC,FSET.UPAR              ; 4786
000156 000053              TRAP 56
000160 000000G      .WORD 53
000162 000000G      .WORD EGH.30
000164 012746 000001      .WORD EMS.43
000170 004737 000000G      MOV  #1,-(SP)                  ; 4787
000174 004737 013576'      JSR  PC,HARD.ERR
000200 013700 000000G      JSR  PC,SWEEP
000204 105360 000000G      MOV  CCTLR,R0                  ; 4788
000210 005726              DEC  QIO(R0)                    ; 4789
000212 000207              4$:  TST  (SP)+                      ; 4752
                                RTS   PC                                ; 4730

```

```

; Routine Size: 70 words,      Routine Base: $CODE$ + 14040
; Maximum stack depth per invocation: 7 words

```

```
: 4794 1 ROUTINE DRV_TIMCHK : NOVALUE =
: 4795 1
: 4796 1 !*
: 4797 1 !
: 4798 1 ! THIS ROUTINE IS CALLED ONCE PER SECOND FROM EITHER THE MULTI-DRIVE
: 4799 1 ! SUBTEST EXECUTIVE (MULTI_DRIVE) OR THE DM EXERCISER SUBTEST EXECUTIVE
: 4800 1 ! (DM_EXER). ITS PURPOSE IS TO DECREMENT THE TIMING COUNTS FOR ALL
: 4801 1 ! OUTSTANDING COMMANDS, AND TO PROCESS ANY THAT REACH ZERO.
: 4802 1 !
: 4803 1 ! A COMMAND TIMER THAT REACHES ZERO CAN MEAN ONE OF TWO THINGS. IF THE
: 4804 1 ! OWNERSHIP BIT OF THE ENVELOPE DESCRIPTOR OF THE TIMED OUT COMMAND
: 4805 1 ! INDICATES THAT THE SLOT IS STILL CONTROLLER-OWNED, THEN THE CONTROLLER
: 4806 1 ! IS ASSUMED DEAD. A DEVICE-FATAL ERROR IS DECLARED AND ALL CONTROLLER-
: 4807 1 ! RELATED DATA IN THE "DRIVER" ARE INITIALIZED. IF THE AFOREMENTIONED
: 4808 1 ! OWNERSHIP BIT INDICATES THAT THE SLOT HAS BEEN RETURNED TO THE HOST, OR
: 4809 1 ! IF THE ENVELOPE DESCRIPTOR SLOT FOR THE TIMED OUT COMMAND CANNOT BE
: 4810 1 ! FOUND (I.E., OVERWRITTEN WITH ANOTHER DESCRIPTOR), THEN ONLY THE
: 4811 1 ! COMMAND IS ASSUMED LOST.
: 4812 1 !
: 4813 1 ! IN EITHER CASE, A RETURN PACKET IS SET UP WHICH DESCRIBES THE
: 4814 1 ! CONDITION, AND "SENT" TO THE "EXERCISER".
: 4815 1 !-
: 4816 2 BEGIN
: 4817 2
: 4818 2 LOCAL
: 4819 2 M_INDEX : SIGNED WORD, ! MSCP ENVELOPE INDEX
: 4820 2 M_ADDR, ! MSCP ENVELOPE ADDRESS
: 4821 2 CR_ADDR, ! COMMAND RING (C-RING) ADDRESS
: 4822 2 TYPE : WORD; ! TYPE OF ERROR
: 4823 2
: 4824 2 INCR CNTLR FROM 0 TO (MAX_CTLR - 1) DO ! FOR EACH CONTROLLER
: 4825 3 BEGIN
: 4826 3
: 4827 3 SET_CPAR (.CNTLR); ! SET UP CONTROLLER-RELATED DATA
: 4828 3 IF .CST_ADDR [STATE] EQLU ONLINE ! IF CONTROLLER IS ONLINE
: 4829 3 THEN ! THEN
: 4830 4 BEGIN
: 4831 4
: 4832 4 INCR COUNT FROM .OCL_X1 TO .OCL_X2 DO ! FOR EACH OUTSTANDING COMMAND ENTRY
: 4833 5 BEGIN
: 4834 5
: 4835 5 IF (M_INDEX = .OUTC_LIST [.COUNT]) GEQ 0 ! IF ENTRY HOLDS VALID MSCP INDEX
: 4836 5 THEN ! THEN
: 4837 6 BEGIN
: 4838 6
: 4839 6 OUTC_TIMR [.COUNT] = .OUTC_TIMR [.COUNT] - 1; ! DECREMENT COMMAND TIMER
: 4840 6 IF .OUTC_TIMR [.COUNT] EQLU 0 ! IF COMMAND TIMED OUT
: 4841 6 THEN ! THEN
: 4842 7 BEGIN
: 4843 7
: 4844 7 OUTC_LIST [.COUNT] = -1; ! KILL OUTC ENTRY
: 4845 7 M_ADDR = MSCP_ENV + (.M_INDEX * ENV_LEN * 2); ! CALCULATE MSCP ENV ADDRESS
: 4846 7 CR_ADDR = .DCT_ADDR [CR_BEG]; ! GET START OF C-RING
: 4847 7 TYPE = MT_TIMEOUT; ! ASSUME COMMAND TIMED OUT
: 4848 7 INCR SLOT FROM 1 TO CRING_LEN DO ! FOR EACH CRING SLOT
: 4849 8 BEGIN
: 4850 8
```



```
: 4851 9          IF ..CR_ADDR EQLA (.M_ADDR + 8)          ! IF THIS SLOT HOLDS TIMED OUT COMMAND
: 4852 8          THEN                                     ! THEN
: 4853 9          BEGIN
: 4854 9
: 4855 9          CR_ADDR = .CR_ADDR + 2;                ! ADVANCE TO HI-ORDER WORD
: 4856 10         IF BIT_TST (.CR_ADDR, ED_OWN)          ! IF SLOT IS STILL CTLR-OWNED
: 4857 9          THEN                                     ! THEN
: 4858 10         BEGIN
: 4859 10
: 4860 10         WRT_RC25 (RCIP, RC_ALL, ALL_ONES);      ! WRITE IP TO STOP DEVICE
: 4861 10         L$LUN = .CST_ADDR [OF_UN, P_UNIT];    ! 1ST UNIT # IN CTLR
: 4862 10         ERRDF (20, EGD_20, EMS_20);          ! "CONTROLLER TIMEOUT"
: 4863 10         DRV_CTLERR (.CNTLR);                 ! CLEAN UP CTLR DATA IN DRIVER
: 4864 10         TYPE = MT_FATAL;                     ! CHANGE TYPE TO FATAL ERROR
: 4865 10
: 4866 9          END;                                    ! IF SLOT STILL CTLR-OWNED
: 4867 9
: 4868 9          EXITLOOP;                              ! EXIT C-RING SLOT SEARCH LOOP
: 4869 9
: 4870 8          END;                                    ! IF COMMAND SLOT IS FOUND
: 4871 8
: 4872 8          CR_ADDR = .CR_ADDR + 4;              ! ADVANCE TO NEXT C-RING SLOT
: 4873 8
: 4874 7          END;                                    ! C-RING SLOT SEARCH LOOP
: 4875 7
: 4876 7          IF (RP_INDX = GET_RETPKT (.CNTLR)) GEQ 0 ! IF RETPKT IS AVAILABLE
: 4877 7          THEN                                     ! THEN
: 4878 8          BEGIN
: 4879 8
: 4880 8          RP_ADDR = RETPKT + (.RP_INDX * RP_LEN * 2);
: 4881 8          IF .TYPE EQLU MT_TIMEOUT              ! IF COMMAND TIMEOUT
: 4882 8          THEN                                     ! THEN
: 4883 9          BEGIN
: 4884 9
: 4885 9          COPY_BLK (.M_ADDR + 4, .RP_ADDR, RP_LEN); ! COPY COMMAND TO RETPKT
: 4886 9          PUT_ENV (.M_INDEX);                  ! RETURN COMMAND ENVELOPE TO POOL
: 4887 9
: 4888 8          END;                                    ! IF TYPE = COMMAND TIMEOUT
: 4889 8
: 4890 8          RP_ADDR [CTLR] = .CNTLR;              ! LOAD CTLR # INTO RETPKT
: 4891 8          RP_ADDR [MESTYP] = .TYPE;            ! LOAD TYPE
: 4892 8          RP_ADDR [CONID] = CID_DRIVER;        ! LOAD PACKET ORIGINATOR
: 4893 8          IN_IODQ (.RP_INDX);                  ! PASS RETPKT TO EXERCISER VIA IODQ
: 4894 8
: 4895 7          END;                                    ! IF RETPKT IS AVAILABLE
: 4896 7
: 4897 6          END;                                    ! IF COMMAND TIMER REACHED ZERO
: 4898 6
: 4899 5          END;                                    ! IF OUTC ENTRY HOLDS A COMMAND ENVELOPE INDEX
: 4900 5
: 4901 4          END;                                    ! OUTSTANDING COMMAND ENTRY LOOP
: 4902 4
: 4903 3          END;                                    ! IF CONTROLLER IS ONLINE
: 4904 3
: 4905 2          END;                                    ! CONTROLLER LOOP
: 4906 2
: 4907 2          T_FLAG = FALSE;                       ! CLEAR ENTRY FLAG
```

; 4908 2
; 4909 1 END;

! ROUTINE DRV_TIMCHK

```

000000 004137 000000G          .SBTTL  DRV.TIMCHK MULTI-DRIVE SUBTEST ROUTINES
                                DRV.TIMCHK:
000004 162706 000006          JSR     R1,#SAVES          ;          4794
000010 005003                SUB     #6,SP
000012 010346                CLR     R3                ; CNTLR    4824
000014 004737 000000G      1$:  MOV     R3,-(SP)        ; CNTLR,*  4827
000020 013700 000000G      JSR     PC,SET.CPAR
000024 005760 000002      MOV     CST.ADDR,R0        ;          4828
000030 100402                TST     2(R0)
000032 000137 014762'      BMI     2$
000036 013701 000000G      JMP     11$
000042 005301                2$:  MOV     OCL.X1,R1        ; *,COUNT 4832
000044 000137 014746'      DEC     R1                ; COUNT
000050 116100 000000G      3$:  JMP     10$
000054 010066 000006      4$:  MOVB    OUTC.LIST(R1),R0  ; *(COUNT),* 4835
                                MOV     R0,6(SP)        ; *,M.INDEX
000060 002771                BLT     3$
000062 010100                MOV     R1,R0            ; COUNT,*    4839
000064 006300                ASL     R0
000066 005360 000000G      DEC     OUTC.TIMR(R0)
000072 001177                BNE     10$
000074 112761 000377 000000G  MOVB    #377,OUTC.LIST(R1) ; *,*(COUNT) 4840
000102 016616 000006      MOV     6(SP),(SP)        ; *,*(COUNT) 4844
000106 012746 000104      MOV     #104,-(SP)       ; M.INDEX,*  4845
000112 004737 000000G      JSR     PC,BL$MUL
000116 062700 000000G      ADD     #MSCP.ENV,R0
000122 010066 000004      MOV     R0,4(SP)        ; *,M.ADDR
000126 013700 000000G      MOV     DCT.ADDR,R0      ;
000132 016005 000010      MOV     10(R0),R5        ; *,CR.ADDR  4846
000136 012766 000004 000006  MOV     #4,6(SP)         ; *,TYPE
000144 016604 000004      MOV     4(SP),R4         ; *,M.ADDR,*  4847
000150 062704 000010      ADD     #10,R4           ; M.ADDR,*  4851
000154 012702 000010      MOV     #10,R2
000160 021504 5$:  CMP     (R5),R4          ; *,SLOT     4848
                                ; CR.ADDR,*  4851
000162 001040                BNE     6$
000164 062705 000002      ADD     #2,R5            ; *,CR.ADDR  4855
000170 011500                MOV     (R5),R0          ; CR.ADDR,*  4856
000172 042700 077777      BIC     #77777,R0
000176 020027 100000      CMP     R0,#-100000
000202 001034                BNE     7$
000204 012700 177777      MOV     #-1,R0           ; *,RC.REG   4860
000210 010077 000000G      MOV     R0,@RC25.ADDR   ; RC.REG,*
000214 013700 000000G      MOV     CST.ADDR,R0
000220 016046 000006      MOV     6(R0),-(SP)     ;          4861
000224 000316                SWAB    (SP)
000226 042716 177740      BIC     #177740,(SP)
000232 012637 000000G      MOV     (SP)+,L$LUN
000236 104455                TRAP    55
000240 000024                .WORD   24
000242 000000G                .WORD   EGD.20
000244 000000G                .WORD   EMS.20
000246 010316                MOV     R3,(SP)        ; CNTLR,*    4863
000250 004737 000000G      JSR     PC,DRV.CTLERR

```


000254	012766	000003	000006		MOV	#3,6(SP)		; *,TYPE	4864
000262	000404				BR	7\$			4853
000264	062705	000004		6\$:	ADD	#4,R5		; *,CR.ADDR	4872
000270	005302				DEC	R2		; SLOT	4848
000272	001332				BNE	5\$			
000274	010316			7\$:	MOV	R3,(SP)		; CNTLR,*	4876
000276	004737	000000G			JSR	PC,GET.RETPKT			
000302	010037	000000G			MOV	RO,RP.INDX			
000306	002470				BLT	9\$			
000310	010016				MOV	RO,(SP)		; RP.INDX,*	4880
000312	012746	000060			MOV	#60,-(SP)			
000316	004737	000000G			JSR	PC,BL\$MUL			
000322	062700	000000G			ADD	#RETPKT,RO			
000326	010037	000000G			MOV	RO,RP.ADDR			
000332	026627	000010	000004		CMP	10(SP),#4		; TYPE,*	4881
000340	001016				BNE	8\$			
000342	016616	000006			MOV	6(SP),(SP)		; M.ADDR,*	4885
000346	062716	000004			ADD	#4,(SP)			
000352	010046				MOV	RO,-(SP)		; RP.ADDR,*	
000354	012746	000030			MOV	#30,-(SP)			
000360	004737	000000G			JSR	PC,COPY.BLK			
000364	016616	000016			MOV	16(SP),(SP)		; M.INDEX,*	4886
000370	004737	000000G			JSR	PC,PUT.ENV			
000374	022626				CMP	(SP)+,(SP)+			
000376	013700	000000G		8\$:	MOV	RP.ADDR,RO			4883
000402	062700	000002			ADD	#2,RO			4890
000406	010302				MOV	R3,R2		; CNTLR,*	
000410	042702	177760			BIC	#177760,R2			
000414	142710	000017			BICB	#17,(R0)			
000420	150210				BISB	R2,(R0)			
000422	016602	000010			MOV	10(SP),R2		; TYPE,*	4891
000426	006302				ASL	R2			
000430	006302				ASL	R2			
000432	006302				ASL	R2			
000434	006302				ASL	R2			
000436	042702	177417			BIC	#177417,R2			
000442	142710	000360			BICB	#360,(R0)			
000446	150210				BISB	R2,(R0)			
000450	112760	000003	000001		MOVB	#3,1(R0)			4892
000456	013716	000000G			MOV	RP.INDX,(SP)			4893
000462	004737	000000G			JSR	PC,IN.IODQ			
000466	005726				TST	(SP)+			4878
000470	005726			9\$:	TST	(SP)+			4842
000472	005201			10\$:	INC	R1		; COUNT	4832
000474	020137	000000G			CMP	R1,OCL.X2		; COUNT,*	
000500	003002				BGT	11\$			
000502	000137	014324'			JMP	4\$			
000506	005726			11\$:	TST	(SP)+			4825
000510	005203				INC	R3		; CNTLR	4824
000512	020327	000003			CMP	R3,#3		; CNTLR,*	
000516	003002				BGT	12\$			
000520	000137	014266'			JMP	1\$			
000524	105037	000000G		12\$:	CLRB	T.FLAG			4907
000530	062706	000006			ADD	#6,SP			4794
000534	000207				RTS	PC			

; Routine Size: 175 words, Routine Base: \$CODE\$ + 14254
; Maximum stack depth per invocation: 15 words

```

: 4910 1  *SBTTL 'RC25 INTERRUPT SERVICE ROUTINES'
: 4911 1
: 4912 1  !*
: 4913 1  !
: 4914 1  !   THERE EXISTS AN RC25 INTERRUPT SERVICE ROUTINE FOR EACH DEVICE
: 4915 1  !   CONTROLLER. EACH SERVICE ROUTINE BEGINS BY SIMPLY SETTING THE
: 4916 1  !   APPROPRIATE CONTROLLER NUMBER INTO "ICTLR". ALL SERVICE ROUTINES THEN
: 4917 1  !   BRANCH TO A COMMON INTERRUPT PROCESSING ROUTINE.
: 4918 1  !-
: 4919 2  BGNSRV (RCINT0);
: 4920 2
: 4921 2  ICTLR = 0;
: 4922 2  RCINT ();
: 4923 2
: 4924 1  ENDSRV;
    
```

```

                                .SBTTL RCINT0 RC25 INTERRUPT SERVICE ROUTINES
000000 010046                    RCINT0::MOV    R0,-(SP)                ; 4919
000002 005037 000600'            CLR     ICTLR                    ; 4921
000006 004737 000000V           JSR    PC,RCINT                ; 4922
000012 012600                    MOV    (SP)+,R0                ; 4919
000014 000002                    RTI
    
```

```

: Routine Size: 7 words,      Routine Base: $CODE$ + 15012
: Maximum stack depth per invocation: 2 words
    
```

```

: 4925 1
: 4926 2  BGNSRV (RCINT1);
: 4927 2
: 4928 2  ICTLR = 1;
: 4929 2  RCINT ();
: 4930 2
: 4931 1  ENDSRV;
    
```

```

                                .SBTTL RCINT1 RC25 INTERRUPT SERVICE ROUTINES
000000 010046                    RCINT1::MOV    R0,-(SP)                ; 4926
000002 012737 000001 000600'    MOV    #1,ICTLR                ; 4928
000010 004737 000000V           JSR    PC,RCINT                ; 4929
000014 012600                    MOV    (SP)+,R0                ; 4926
000016 000002                    RTI
    
```

```

: Routine Size: 8 words,      Routine Base: $CODE$ + 15030
: Maximum stack depth per invocation: 2 words
    
```

```

: 4932 1
: 4933 2  BGNSRV (RCINT2);
: 4934 2
: 4935 2  ICTLR = 2;
: 4936 2  RCINT ();
: 4937 2
: 4938 1  ENDSRV;
    
```



```

.SBTTL RCINT2 RC25 INTERRUPT SERVICE ROUTINES
000000 010046 RCINT2::MOV R0,-(SP) ; 4933
000002 012737 000002 000600' MOV #2,ICTLR ; 4935
000010 004737 000000V JSR PC,RCINT ; 4936
000014 012600 MOV (SP)+,R0 ; 4933
000016 000002 RTI ;

```

```

; Routine Size: 8 words, Routine Base: $CODE$ + 15050
; Maximum stack depth per invocation: 2 words

```

```

; 4939 1
; 4940 2 BGNSRV (RCINT3);
; 4941 2
; 4942 2 ICTLR = 3;
; 4943 2 RCINT ();
; 4944 2
; 4945 1 ENDSRV;

```

```

.SBTTL RCINT3 RC25 INTERRUPT SERVICE ROUTINES
000000 010046 RCINT3::MOV R0,-(SP) ; 4940
000002 012737 000003 000600' MOV #3,ICTLR ; 4942
000010 004737 000000V JSR PC,RCINT ; 4943
000014 012600 MOV (SP)+,R0 ; 4940
000016 000002 RTI ;

```

```

; Routine Size: 8 words, Routine Base: $CODE$ + 15070
; Maximum stack depth per invocation: 2 words

```

```

: 4946 1 ROUTINE RCINT : NOVALUE =
: 4947 1
: 4948 1 !*
: 4949 1 ! THIS IS THE COMMON INTERRUPT SERVICE ROUTINE FOR ALL RC25 CONTROLLERS.
: 4950 1 ! AFTER SETTING UP THE COMMONLY-USED CONTROLLER-RELATED DATA ITEMS, THIS
: 4951 1 ! ROUTINE WILL SAVE THE CURRENT CONTENTS OF THE SA REGISTER IN THE DCT.
: 4952 1 ! THEN, IF THE "IGNORE INTERRUPT" BIT IS SET, NO FURTHER ACTION IS TAKEN.
: 4953 1 !
: 4954 1 ! IF THE INITIALIZATION SUBTEST IS IN PROGRESS OR IF THE DM EXERCISER
: 4955 1 ! SUBTEST BEING RUN, THEN INTERRUPT PROCESSING CONTINUES BY CHECKING THE
: 4956 1 ! ERROR BIT OF THE SA REGISTER, AND POLLING THE COMMAND AND RESPONSE
: 4957 1 ! RINGS. OTHERWISE, THE INTERRUPT IS PROCESSED LATER WHEN INT_PROC IS
: 4958 1 ! CALLED FROM THE MULTI-DRIVE SUBTEST "EXECUTIVE" (MULTI_DRIVE).
: 4959 1 !-
: 4960 1
: 4961 2 BEGIN
: 4962 2
: 4963 2 ISET_CPAR (); ! SET UP CONTROLLER-RELATED PARAMETERS
: 4964 2 IDCT_ADDR [SA_SAVE] = .IRC25_ADDR [RCSA, RC_ALL]; ! SAVE SA REGISTER
: 4965 2 IF .IDCT_ADDR [IG_INT] EQLU YES ! IGNORE INTERRUPT?
: 4966 2 THEN
: 4967 2 RETURN; ! YES - RETURN
: 4968 3 IF ((.IIP_FLAG) OR (BIT_TST (SWP_FLAGS, SWF_DM))) ! IF INIT SUBTEST OR DM EXERCISER
: 4969 2 THEN ! THEN
: 4970 3 BEGIN
: 4971 3
: 4972 4 IF BIT_TST (IDCT_ADDR [SA_SAVE], SA_ERR) ! IF FATAL ERROR
: 4973 3 THEN ! THEN
: 4974 3 FATAL_ERROR (); ! DECLARE ERROR AND CLEAN UP
: 4975 3 ELSE ! OTHERWISE
: 4976 4 BEGIN
: 4977 4
: 4978 4 POLL_CRING (); ! POLL COMMAND RING
: 4979 4 POLL_RRING (); ! POLL RESPONSE RING
: 4980 4
: 4981 3 END;
: 4982 3
: 4983 2 END; ! IF INIT SUBTEST OR DM EXERCISER
: 4984 2
: 4985 1 END; ! ROUTINE RCINT

```

		.SBTTL	RCINT RC25 INTERRUPT SERVICE ROUTINES	
000000	010146		RCINT: MOV R1, -(SP)	4946
000002	005746		TST -(SP)	
000004	004737	000000V	JSR PC, ISET.CPAR	4963
000010	013700	000566'	MOV IDCT.ADDR, R0	4964
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
000034	001026		BNE 3\$	4965
000036	132737	000001 000000G	BITB #1, IIP.FLAG	4967
000044	001004		BNE 1\$	4968
000046	132737	000002 000000G	BITB #2, SWP.FLAGS	
000054	001416		BEQ 3\$	
000056	016000	000002	1\$: MOV 2(R0), R0	4972

CZRCDB
V02.0

CZRCDB0 RC25 DISK EXERCISER
RC25 INTERRUPT SERVICE ROUTINES

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER\$1:[AZTEC.CZRCDB]CZRCDB3.SRC;17

SEQ 0329
Page 143
(49)

000062	042700	077777	BIC	#77777,R0		
000066	020027	100000	CMP	R0,#-100000		
000072	001003		BNE	2\$		
000074	004737	000000V	JSR	PC,FATAL.ERROR	:	4974
000100	000404		BR	3\$:	4972
000102	004737	000000V	JSR	PC,POLL.CRING	:	4978
000106	004737	000000V	JSR	PC,POLL.RRING	:	4979
000112	005726		TST	(SP)+	:	4946
000114	012601		MOV	(SP)+,R1	:	
000116	000207		RTS	PC	:	

: Routine Size: 40 words, Routine Base: \$CODE\$ + 15110
: Maximum stack depth per invocation: 3 words

```

: 4986 1 ROUTINE INT_PROC : NOVALUE =
: 4987 1
: 4988 1 !+
: 4989 1 ! THIS ROUTINE IS CALLED FROM THE "EXECUTIVE" OF THE MULTI-DRIVE SUBTEST
: 4990 1 ! TO PROCESS ANY RC25 INTERRUPTS WHICH MAY HAVE OCCURRED IN THE RECENT
: 4991 1 ! PAST.
: 4992 1 !
: 4993 1 ! IF EITHER THE INITIALIZATION SUBTEST OR DM EXERCISER SUBTEST IS BEING
: 4994 1 ! RUN, THEN INTERRUPTS ARE FULLY PROCESSED AS THEY OCCUR (SEE RCINT).
: 4995 1 ! OTHERWISE, FOR THE MULTI-DRIVE SUBTEST, INTERRUPTS ARE PROCESSED ON A
: 4996 1 ! DELAYED BASIS TO PREVENT PROGRAM PRINTOUTS FROM BEING BROKEN UP BY,
: 4997 1 ! SAY, ERROR LOG MESSAGE PRINTOUTS.
: 4998 1 !
: 4999 1 ! AFTER SOME INITIALIZATION, THIS ROUTINE CONTINUES AS IF AN INTERRUPT
: 5000 1 ! HAS JUST OCCURRED: IF THE ERROR BIT OF THE SA REGISTER IS SET, THEN A
: 5001 1 ! FATAL ERROR IS DECLARED. OTHERWISE, THE COMMAND AND RESPONSE RINGS ARE
: 5002 1 ! POLLED.
: 5003 1 !
: 5004 1 ! IF NO INTERRUPT HAS OCCURRED IN THE RECENT PAST, THEN THE POLLING
: 5005 1 ! ROUTINES TAKE NO ACTION.
: 5006 1 !
: 5007 1 ! NOTE: ALL RC25'S ARE ASSUMED TO BE HARDWIRED AT BR LEVEL 5 OR LESS.
: 5008 1 ! INTERRUPT PROCESSING RUNS AT LEVEL 5, AND CANNOT BE BROKEN BY AN
: 5009 1 ! INTERRUPT FROM A SECOND RC25.
: 5010 1 !
: 5011 1 ! IMPLICIT INPUTS:
: 5012 1 ! CCTLR - CURRENT CONTROLLER NUMBER
: 5013 1 !-
: 5014 1
: 5015 2 BEGIN
: 5016 2
: 5017 2 SETPRI (PRI05); ! LOCK OUT ALL RC25 INTERRUPTS
: 5018 2 ICTLR = .CCTLR; ! FAKE INTERRUPTING CONTROLLER NUMBER
: 5019 2 ISET_CPAR (); ! SET UP CONTROLLER-RELATED DATA
: 5020 2 IDCT_ADDR [SA_SAVE] = .IRC25_ADDR [RCSA, RC_ALL]; ! SAVE SA REGISTER
: 5021 3 IF BIT_TST (IDCT_ADDR [SA_SAVE], SA_ERR) ! IF FATAL ERROR
: 5022 2 THEN
: 5023 2 FATAL_ERROR ()
: 5024 2 ELSE ! OTHERWISE
: 5025 3 BEGIN
: 5026 3
: 5027 3 POLL_CRING (); ! POLL COMMAND RING
: 5028 3 POLL_RRING (); ! POLL RESPONSE RING
: 5029 3
: 5030 2 END;
: 5031 2
: 5032 2 SETPRI (PRI00); ! RESTORE PROCESSOR PRIORITY
: 5033 2
: 5034 1 END; ! ROUTINE INT_PROC

```

```

000000 010146 .SBTTL INT.PROC RC25 INTERRUPT SERVICE ROUTINES
INT.PROC:
000002 005746 MOV R1, -(SP) ; 4986
000004 012700 000240 TST -(SP) ;
000010 104441 MOV #240, R0 ; 5017
TRAP 41

```


CZRC03
V02.0

CZRCDB0 RC25 DISK EXERCISER
RC25 INTERRUPT SERVICE ROUTINES

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER\$1:[AZTEC.CZRCDB]CZRC03.SRC;17

SEQ 0331
Page 145
(50)

000012	013737	000000G	000600'	MOV	CCTLR,ICTLR	:	5018
000020	004737	000000V		JSR	PC,ISET.CPAR	:	5019
000024	013701	000566'		MOV	IDCT.ADDR,R1	:	5020
000030	013700	000000G		MOV	IRC25.ADDR,R0	:	
000034	016016	000002		MOV	2(R0),(SP)	: *,RC.REG	
000040	011661	000002		MOV	(SP),2(R1)	: RC.REG,*	
000044	011601			MOV	(SP),R1	:	5021
000046	042701	077777		BIC	#77777,R1	:	
000052	020127	100000		CMP	R1,#-100000	:	
000056	001003			BNE	1\$:	
000060	004737	000000V		JSR	PC,FATAL.ERROR	:	5023
000064	000404			BR	2\$:	5021
000066	004737	000000V	1\$:	JSR	PC,POLL.CRING	:	5027
000072	004737	000000V		JSR	PC,POLL.RRING	:	5028
000076	005000		2\$:	CLR	R0	:	5032
000100	104441			TRAP	41	:	
000102	005726			TST	(SP)+	:	
000104	012601			MOV	(SP)+,R1	:	4986
000106	000207			RTS	PC	:	

; Routine Size: 36 words, Routine Base: \$CODE\$ + 15230
; Maximum stack depth per invocation: 4 words

```

; 5035 1  ROUTINE ISET_CPAR : NOVALUE =
; 5036 1
; 5037 1  !+
; 5038 1  ! THIS ROUTINE IS CALLED BY RCINT AND INT_PROC TO SET UP THE COMMONLY-
; 5039 1  ! USED DATA ITEMS FOR THE INTERRUPTING CONTROLLER.
; 5040 1  !
; 5041 1  ! IMPLICIT INPUTS:
; 5042 1  !     ICTLR - INTERRUPTING CONTROLLER NUMBER
; 5043 1  !-
; 5044 1
; 5045 2  BEGIN
; 5046 2
; 5047 2  IDCT_ADDR = DCT + (.ICTLR * DCT_LEN * 2);      ! CALCULATE DCT ADDRESS
; 5048 2  ICST_ADDR = CST + (.ICTLR * CST_LEN * 2);      ! CALCULATE CST ADDRESS
; 5049 2  IRC25_ADDR = .ICST_ADDR [IP_ADDR];          ! GET RC25 ADDRESS
; 5050 2  ICOM_ADDR = COMM_AREA + (.ICTLR * COMM_LEN * 2); ! CALCULATE COMM_AREA ADDR
; 5051 2
; 5052 1  END;                                          ! ROUTINE ISET_CPAR
    
```

```

000000 010146      .SBTTL  ISET.CPAR RC25 INTERRUPT SERVICE ROUTINES
                                ISET.CPAR:
000002 013701 000600'  MOV     R1,-(SP)                ; 5035
000006 010146      MOV     ICTLR,R1                ; 5047
000010 012746 000022  MOV     R1,-(SP)
000014 004737 000000G  MOV     #22,-(SP)
000020 062700 000000G  JSR     PC,BL$MUL
000024 010037 000566'  ADD     #DCT,R0
000030 010116      MOV     R0,IDCT.ADDR
000032 012746 000016  MOV     R1,(SP)                ; 5048
000036 004737 000000G  MOV     #16,-(SP)
000042 062700 000000G  JSR     PC,BL$MUL
000046 010037 000564'  ADD     #CST,R0
000052 011037 000000G  MOV     R0,ICST.ADDR
000056 010116      MOV     (R0),IRC25.ADDR        ; ICST.ADDR,* 5049
000060 012746 000110  MOV     R1,(SP)                ; 5050
000064 004737 000000G  MOV     #110,-(SP)
000070 062700 000000'  JSR     PC,BL$MUL
000074 010037 000560'  ADD     #COMM.AREA,R0
000100 062706 000010  MOV     R0,ICOM.ADDR
000104 012601      ADD     #10,SP
000106 000207      MOV     (SP)+,R1                ; 5045
                                RTS     PC                            ; 5035
    
```

```

; Routine Size: 36 words,      Routine Base: $CODE$ + 15340
; Maximum stack depth per invocation: 6 words
    
```



```

; 5053 1  ROUTINE FATAL_ERROR : NOVALUE =
; 5054 1
; 5055 1  !+
; 5056 1  !
; 5057 1  ! THIS ROUTINE IS CALLED BY BOTH RCINT AND INT_PROC UPON DETECTING AN
; 5058 1  ! UNRECOVERABLE ERROR THROUGH THE DEVICE'S SA REGISTER. ITS PURPOSE IS TO
; 5059 1  ! CLEAN UP DEVICE DATA IN THE "DRIVER" PORTION OF THE EXERCISER, AND TO
; 5060 1  ! INFORM THE "PROGRAM" PORTION OF THE EVENT VIA RETURN PACKET.
; 5061 1  !
; 5062 1  ! IMPLICIT INPUTS:
; 5063 1  ! ICTLR - INTERRUPTING CONTROLLER NUMBER
; 5064 1  ! IDCT_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S DCT
; 5065 1  ! ICST_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S CST
; 5066 1  !-
; 5067 2  BEGIN
; 5068 2
; 5069 2  LOCAL
; 5070 2  INDEX : SIGNED WORD,
; 5071 2  U_SAVE : WORD;
; 5072 2
; 5073 2  SA_REG = .IDCT_ADDR [SA_SAVE];
; 5074 2  U_SAVE = .L$LUN;
; 5075 2  L$LUN = .ICST_ADDR [OF_UN, P_UNIT];
; 5076 2  ERRDF (14, EGD_14, EMS_14);
; 5077 2  L$LUN = .U_SAVE;
; 5078 2  DRV_CTLERR (.ICTLR);
; 5079 2  IF (INDEX = GET_RETPKT (.ICTLR)) GEQ 0
; 5080 2  THEN
; 5081 3  BEGIN
; 5082 3
; 5083 3  RETPKT [.INDEX, CONID] = CID_DRIVER;
; 5084 3  RETPKT [.INDEX, MESTYP] = MT_FATAL;
; 5085 3  RETPKT [.INDEX, CTLR] = .ICTLR;
; 5086 3  IN_IODQ (.INDEX);
; 5087 3
; 5088 2  END;
; 5089 2
; 5090 1  END;

```

```

! SAVE PRE-INTERRUPT CURRENT UNIT NUMBER
! SET CURRENT UNIT TO FIRST IN CONTROLLER
! "FATAL PORT / CONTROLLER ERROR"
! RESTORE PRE-INTERRUPT CURRENT UNIT
! CLEAN UP DRIVER DATA FOR CONTROLLER
! TRY TO GET A RETPKT; IF SUCCESS

! SET CONNECTION ID TO "DRIVER"
! FATAL ERROR
! CONTROLLER NUMBER
! LOAD RETPKT INDEX INTO IODQ

! IF RETPKT WAS ALLOCATED

! ROUTINE FATAL_ERR

```

```

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

```

```

;
;
; *,U.SAVE
;
;
;
;
; U.SAVE,*
;
5053
5073
5074
5075
5076
5077
5078

```

CZRC03
V02.0

CZRCDB0 RC25 DISK EXERCISER
RC25 INTERRUPT SERVICE ROUTINES

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER\$1:[AZTEC.CZRCDB]CZRC03.SRC;17

SEQ 0334
Page 148
(52)

000064	004737	000000G	JSR	PC,DRV.CTLERR		
000070	013716	000600'	MOV	ICTLR,(SP)	:	
000074	004737	000000G	JSR	PC,GET.RETPKT		5079
000100	010001		MOV	R0,R1	: *,INDEX	
000102	002425		BLT	1\$		
000104	010116		MOV	R1,(SP)	: INDEX,*	
000106	012746	000060	MOV	#60,-(SP)		5083
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	:	
000134	042702	177760	BIC	#177760,R2		5085
000140	112710	000060	MOVB	#60,(R0)		
000144	150210		BISB	R2,(R0)		
000146	010116		MOV	R1,(SP)	: INDEX,*	
000150	004737	000000G	JSR	PC,IN.IODQ		5086
000154	005726		TST	(SP)+	:	5081
000156	005726	1\$:	TST	(SP)+	:	5067
000160	000207		RTS	PC	:	5053

; Routine Size: 57 words, Routine Base: \$CODE\$ + 15450
; Maximum stack depth per invocation: 6 words


```

: 5091 1 ROUTINE POLL_CRING : NOVALUE =
: 5092 1
: 5093 1 !+
: 5094 1 ! THIS ROUTINE IS CALLED BY BOTH RCINT AND INT_PROC TO SCAN THE DEVICE'S
: 5095 1 ! COMMAND RING AND CHECK FOR ANY COMMAND SLOTS THAT HAVE BEEN "TAKEN" BY
: 5096 1 ! THE CONTROLLER. SUCH SLOTS HAVE BEEN RETURNED TO THE HOST, INDICATED BY
: 5097 1 ! A ZERO OWNERSHIP BIT. FOR EACH SLOT THAT HAS BEEN RETURNED TO THE HOST,
: 5098 1 ! THE CRING COUNT IS DECREMENTED, AND THE CR_POLL ADDRESS IS ADVANCED TO
: 5099 1 ! THE NEXT SLOT IN THE COMMAND RING.
: 5100 1 !
: 5101 1 ! IMPLICIT INPUTS:
: 5102 1 ! ICTLR - INTERRUPTING CONTROLLER NUMBER
: 5103 1 ! IDCT_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S DCT
: 5104 1 ! ICOM_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S COMM_AREA
: 5105 1 !-
: 5107 2 BEGIN
: 5108 2
: 5109 3 WHILE ((.IDCT_ADDR [CRING_CNT] GTRU 0) AND ! WHILE # OF COMMANDS IN CRING > 0 AND
: 5110 2 NOT (BIT_TST ((.IDCT_ADDR [CR_POLL] + 2), ED_OWN))) DO ! CURRENT SLOT IS HOST-OWNED
: 5111 3 BEGIN
: 5112 3
: 5113 3 IDCT_ADDR [CRING_CNT] = .IDCT_ADDR [CRING_CNT] - 1; ! DECREMENT # CMDS IN CRING
: 5114 3 IDCT_ADDR [CR_POLL] = .IDCT_ADDR [CR_POLL] + 4; ! ADVANCE TO NEXT SLOT TO POLL
: 5115 3 IF .IDCT_ADDR [CR_POLL] GTRA .IDCT_ADDR [CR_END] ! IF BEYOND END OF RING
: 5116 3 THEN ! THEN
: 5117 3 IDCT_ADDR [CR_POLL] = .IDCT_ADDR [CR_BEG]; ! SET POINTER TO TOP OF CRING
: 5118 3
: 5119 2 END;
: 5120 2
: 5121 2 ICOM_ADDR [CMD_INT] = 0; ! CLEAR COMMAND INTERRUPT WORD IN RING HEADER
: 5122 2
: 5123 1 END;

```

```

000000 004137 000000G .SBTTL POLL.CRING RC25 INTERRUPT SERVICE ROUTINES
                                POLL.CRING:
000004 013701 000566' JSR R1,$SAVE2 ; 5091
000010 012702 000016 MOV IDCT.ADDR,R1 ; 5109
000014 060102 MOV #16,R2 ; 5114
000016 105711 1$: TSTB (R1) ; 5109
000020 001422 BEQ 2$ ;
000022 016100 000016 MOV 16(R1),R0 ; 5110
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) ; 5113
000046 062712 000004 ADD #4,(R2) ; 5114
000052 021261 000012 CMP (R2),12(R1) ; 5115
000056 101757 BLOS 1$
000060 016112 000010 MOV 10(R1),(R2) ; 5117
000064 000754 BR 1$ ; 5109
000066 013700 000560' 2$: MOV ICOM.ADDR,R0 ; 5121
000072 005060 000004 CLR 4(R0)
000076 000207 RTS PC ; 5091

```

```

: Routine Size: 32 words, Routine Base: $CODE$ + 15632
: Maximum stack depth per invocation: 4 words

```

```

: 5124 1 ROUTINE POLL_RRING : NOVALUE =
: 5125 1
: 5126 1 !+
: 5127 1 !
: 5128 1 ! THIS ROUTINE IS CALLED BY BOTH RCINT AND INT_PROC TO SCAN THE DEVICE'S
: 5129 1 ! RESPONSE RING AND CHECK FOR ANY SLOTS WHICH HAVE BEEN RETURNED TO THE
: 5130 1 ! HOST (OWNERSHIP BIT = 0). FOR EACH SUCH SLOT, THE ASSOCIATED MESSAGE IS
: 5131 1 ! PROCESSED BASED ON ITS CONNECTION ID (DISK OR DUP) AND MESSAGE TYPE
: 5132 1 ! (SEQUENTIAL OR DATAGRAM). AFTER PROCESSING, THE MESSAGE ENVELOPE IS
: 5133 1 ! RE-INITIALIZED AND RETURNED TO THE CONTROLLER (OWNERSHIP BIT SET TO 1).
: 5134 1 !
: 5135 1 ! IMPLICIT INPUTS:
: 5136 1 ! IDCT_ADDR - ADDRESS OF INTERRUPTING CONTROLLER'S DCT
: 5137 1 !-
: 5138 2 BEGIN
: 5139 2
: 5140 2 WHILE NOT (BIT_TST ((.IDCT_ADDR [RR_POLL] + 2), ED_OWN)) DO ! WHILE 0 = 0
: 5141 3 BEGIN
: 5142 3
: 5143 3 IENV_ADDR = ..IDCT_ADDR [RR_POLL] - 8; ! ADDRESS OF RESPONSE ENVELOPE
: 5144 3 IF .IENV_ADDR [CONNID] EQLU CID_DISK ! IF MESSAGE IS FROM DISK MSCP
: 5145 3 THEN ! THEN
: 5146 3 SELECTONEU .IENV_ADDR [MSGTYP] OF ! PROCESS ON MESSAGE TYPE
: 5147 3 SET
: 5148 3
: 5149 3 [MT_SEQ] : ENV_TO_RP (); ! SEQUENTIAL
: 5150 3 [MT_DG] : DATAGM (); ! DATAGRAM
: 5151 3
: 5152 3 TES
: 5153 3 ELSE ! ELSE (MESSAGE NOT FROM DISK MSCP)
: 5154 4 BEGIN
: 5155 4
: 5156 4 IF .IENV_ADDR [CONNID] EQLU CID_DUP ! IF MESSAGE IS FROM DUP
: 5157 4 THEN ! THEN
: 5158 4 ENV_TO_RP (); ! COPY ENV TO RETPKT
: 5159 4
: 5160 3 END;
: 5161 3
: 5162 3 IENV_ADDR [MSGLEN] = MSG_LEN * 2; ! RE-INIT MESSAGE LENGTH
: 5163 3 IDCT_ADDR [RR_POLL] = .IDCT_ADDR [RR_POLL] + 2; ! ADVANCE TO HI ORDER WORD OF RING SLOT
: 5164 3 .IDCT_ADDR [RR_POLL] = .IENV_ADDR [ENV_HI]; ! RETURN SLOT TO CONTROLLER
: 5165 3 IDCT_ADDR [RR_POLL] = .IDCT_ADDR [RR_POLL] + 2; ! ADVANCE TO NEXT RRING SLOT
: 5166 3 IF .IDCT_ADDR [RR_POLL] GTRA .IDCT_ADDR [RR_END] ! IF BEYOND END OF RING
: 5167 3 THEN ! THEN
: 5168 3 IDCT_ADDR [RR_POLL] = .IDCT_ADDR [RR_BEG]; ! CYCLE TO TOP OF RING
: 5169 3
: 5170 2 END; ! WHILE LOOP
: 5171 2
: 5172 2 ICOM_ADDR [RSP_INT] = 0; ! CLEAR RESPONSE INTERRUPT WORD IN RING HEADER
: 5173 2
: 5174 1 END;

```

```

000000 004137 000000G
000004 013702 000566'

```

```

.SBTTL POLL.RRING RC25 INTERRUPT SERVICE ROUTINES
POLL.RRING:
JSR R1,$SAVE2 ;
MOV IDCT.ADDR,R2 ;

```

```

5124
5140

```


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	:	5143
000042	162737	000010	000562'	SUB	#10,IENV.ADDR	:	
000050	013700	000562'		MOV	IENV.ADDR,R0	:	5144
000054	062700	000006		ADD	#6,R0		
000060	105760	000001		TSTB	1(R0)		
000064	001016			BNE	2\$		
000066	111000			MOVB	(R0),R0	:	5146
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\$:	5149
000106	020027	000001		CMP	R0,#1	:	5150
000112	001011			BNE	4\$		
000114	004737	000000V		JSR	PC,DATAGM		
000120	000406			BR	4\$:	5144
000122	126027	000001	000002	2\$:	CMPB	1(R0),#2	5156
000130	001002			BNE	4\$		
000132	004737	000000V		3\$:	JSR	PC,ENV.TO.RP	5158
000136	013700	000562'		4\$:	MOV	IENV.ADDR,R0	5162
000142	012760	000074	000004		MOV	#74,4(R0)	
000150	013701	000566'			MOV	IDCT.ADDR,R1	5163
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)	5164
000174	062712	000002			ADD	#2,(R2)	5165
000200	021261	000006			CMP	(R2),6(R1)	5166
000204	101703				BLOS	1\$	
000206	016112	000004			MOV	4(R1),(R2)	5168
000212	000700				BR	1\$	5140
000214	013700	000560'		5\$:	MOV	ICOM.ADDR,R0	5172
000220	005060	000006			CLR	6(R0)	
000224	000207				RTS	PC	5124

; Routine Size: 75 words, Routine Base: \$CODE\$ + 15732
; Maximum stack depth per invocation: 4 words

```
: 5175 1 ROUTINE ENV_TO_RP : NOVALUE =
: 5176 1
: 5177 1 !*
: 5178 1 ! THIS ROUTINE IS CALLED BY POLL_RRING FOR (A) DISK MSCP RESPONSE
: 5179 1 ! MESSAGES WITH THE "SEQUENTIAL" MESSAGE TYPE, AND (B) DUP RESPONSE
: 5180 1 ! MESSAGES. ITS GENERAL PURPOSE IS TO COPY THE CONTENTS OF THE MESSAGE
: 5181 1 ! ENVELOPE INTO A RETURN PACKET SO THAT THE ENVELOPE CAN BE RETURNED TO
: 5182 1 ! THE CONTROLLER. IF THE RESPONSE MESSAGE HAS AN ASSOCIATED COMMAND IN
: 5183 1 ! THE CONTROLLER'S OUTSTANDING COMMAND LIST (OUTC_LIST), THEN TIMING FOR
: 5184 1 ! THAT COMMAND IS TERMINATED. IN ADDITION, IF THE COMMAND WAS AN I/O
: 5185 1 ! TRANSFER (READ, WRITE, OR ACCESS), THEN SOME FIELDS OF THE COMMAND
: 5186 1 ! ENVELOPE ARE COPIED INTO THE RETURN PACKET.
: 5187 1 !
: 5188 1 ! AFTER THE RETURN PACKET HAS BEEN LOADED, ITS INDEX IS LOADED INTO THE
: 5189 1 ! I/O DONE QUEUE (IODQ) FOR PROCESSING BY THE "PROGRAM" PORTION OF THE
: 5190 1 ! DIAGNOSTIC.
: 5191 1 !
: 5192 1 ! IMPLICIT INPUTS:
: 5193 1 ! ICTLR - INTERRUPTING CONTROLLER NUMBER
: 5194 1 ! IENV_ADDR - ADDRESS OF MSCP ENVELOPE CONTAINING RESPONSE
: 5195 1 !-
: 5196 1
: 5197 2 BEGIN
: 5198 2
: 5199 2 LOCAL
: 5200 2 M_INDEX : SIGNED WORD,
: 5201 2 R_INDEX : SIGNED WORD,
: 5202 2 STIDX : WORD,
: 5203 2 ENDIDX : WORD,
: 5204 2 R_ADDR : REF BLOCK [RP_LEN, WORD] FIELD (RP_FIELDS),
: 5205 2 TEMP : WORD;
: 5206 2
: 5207 2 M_INDEX = -1; ! ASSUME NO ASSOCIATED COMMAND PACKET
: 5208 2 STIDX = .ICTLR * OUTC_CNT; ! START OF CONTROLLER'S OUTC_LIST AND _TIMR
: 5209 2 ENDIDX = .STIDX + OUTC_CNT - 1; ! END OF OUTC
: 5210 2 INCR COUNT FROM .STIDX TO .ENDIDX DO ! SEARCH THROUGH EACH OUTC ENTRY
: 5211 3 BEGIN
: 5212 3
: 5213 3 IF (TEMP = .OUTC_LIST [.COUNT]) GEQ 0 ! IF ENTRY CONTAINS AN OUTSTANDING COMMAND INDEX
: 5214 3 THEN ! THEN
: 5215 4 BEGIN
: 5216 4
: 5217 4 IF .MSCP_ENV [.TEMP, CRN_LO] EQLU .IENV_ADDR [CRN_LO] ! IF THIS IS THE ASSOC CMD
: 5218 4 THEN ! THEN
: 5219 5 BEGIN
: 5220 5
: 5221 5 M_INDEX = .TEMP;
: 5222 5 OUTC_LIST [.COUNT] = -1; ! INIT ENTRY
: 5223 5 OUTC_TIMR [.COUNT] = 0; ! TURN OFF COMMAND TIMER
: 5224 5 EXITLOOP;
: 5225 5
: 5226 4 END;
: 5227 4
: 5228 3 END;
: 5229 3
: 5230 2 END;
: 5231 2
```



```

: 5232 2 IF (R_INDEX = GET_RETPKT (.ICTLR)) GEQ 0 ! IF RETPKT IS AVAILABLE
: 5233 2 THEN
: 5234 3 BEGIN
: 5235 3
: 5236 3 R_ADDR = RETPKT + (.R_INDEX * RP_LEN * 2); ! START OF ALLOCATED RETPKT
: 5237 3 COPY_BLK (.IENV_ADDR + 4, .R_ADDR, RP_LEN); ! COPY RESPONSE ENVELOPE INTO RETURN PACKET
: 5238 3 R_ADDR [CTLR] = .ICTLR; ! LOAD CONTROLLER NUMBER INTO PACKET
: 5239 3 IF .M_INDEX GEQ 0 ! IF ASSOC. CMD ENV WAS FOUND
: 5240 3 THEN ! THEN
: 5241 4 BEGIN
: 5242 4
: 5243 5 IF ((.IENV_ADDR [OPCODE] EQLU (OP_RD + OP_END)) OR ! IF END MESSAGE IS
: 5244 5 (.IENV_ADDR [OPCODE] EQLU (OP_WRT + OP_END)) OR ! READ, WRITE, OR
: 5245 5 (.IENV_ADDR [OPCODE] EQLU (OP_ACC + OP_END))) ! ACCESS
: 5246 4 THEN ! THEN
: 5247 5 BEGIN
: 5248 5
: 5249 5 R_ADDR [CMDMOD] = .MSCP_ENV [.M_INDEX, MODIFY]; ! COPY
: 5250 5 R_ADDR [CBCNT_LO] = .MSCP_ENV [.M_INDEX, BC_LO]; ! RELEVANT
: 5251 5 R_ADDR [CBCNT_HI] = .MSCP_ENV [.M_INDEX, BC_HI]; ! FIELDS
: 5252 5 R_ADDR [LBN_LO] = .MSCP_ENV [.M_INDEX, LBN_L]; ! FROM
: 5253 5 R_ADDR [LBN_HI] = .MSCP_ENV [.M_INDEX, LBN_H]; ! COMMAND
: 5254 5 R_ADDR [BUFF_0] = .MSCP_ENV [.M_INDEX, BUF_0]; ! PACKET
: 5255 5 R_ADDR [BUFF_1] = .MSCP_ENV [.M_INDEX, BUF_1]; ! TO RETPKT
: 5256 5
: 5257 4 END; ! IF ENDCODE WAS READ, WRITE, OR ACCESS
: 5258 4
: 5259 3 END; ! IF ASSOC CMD ENV WAS FOUND
: 5260 3
: 5261 3 IN_IODQ (.R_INDEX); ! PUT RETPKT INDEX INTO IODQ
: 5262 3
: 5263 2 END; ! IF RETPKT WAS ALLOCATED
: 5264 2
: 5265 2 IF .M_INDEX GEQ 0 ! IF ASSOC CMD ENV WAS FOUND
: 5266 2 THEN
: 5267 2 PUT_ENV (.M_INDEX); ! RETURN COMMAND ENVELOPE TO POOL
: 5268 2
: 5269 1 END; ! ROUTINE DISK_RSP

```

			.SBTTL	ENV.TO.RP RC25 INTERRUPT SERVICE ROUTINES	
000000	004137	000000G	ENV.TO.RP:		
			JSR	R1,\$SAVE5	5175
000004	012704	177777	MOV	#-1,R4	5207
000010	013700	000600'	MOV	ICTLR,R0	5208
000014	006300		ASL	R0	
000016	006300		ASL	R0	
000020	006300		ASL	R0	
000022	006300		ASL	R0	
000024	010005		MOV	R0,R5	5209
000026	062705	000017	ADD	#17,R5	5210
000032	010001		MOV	R0,R1	
000034	060401		ADD	R4,R1	
000036	000430		BR	2\$	
000040	116103	000000G	1\$: MOVB	OUTC.LIST(R1),R3	5213
000044	002425		BLT	2\$	
000046	010346		MOV	R3,-(SP)	5217

000050	012746	000104		MOV	#104,-(SP)		
000054	004737	000000G		JSR	PC,BL\$MUL		
000060	022626			CMP	(SP)+,(SP)+		
000062	013702	000562'		MOV	IENV.ADDR,R2		
000066	026062	000010G	000010	CMP	MSCP.ENV+10(R0),10(R2)		
000074	001011			BNE	2\$		
000076	010304			MOV	R3,R4	; TEMP,M.INDEX	5221
000100	112761	000377	000000G	MOVB	#377,OUTC.LIST(R1)	; *,*(COUNT)	5222
000106	010100			MOV	R1,R0	; COUNT,*	5223
000110	006300			ASL	R0		
000112	005060	000000G		CLR	OUTC.TIMR(R0)		
000116	000403			BR	3\$		5219
000120	005201			INC	R1	; COUNT	5210
000122	020105			CMP	R1,R5	; COUNT,ENDIDX	
000124	003745			BLE	1\$		
000126	013746	000600'		MOV	ICTLR,-(SP)		5232
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,*	5236
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)		5237
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		5238
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	5239
000234	002452			BLT	5\$		
000236	013700	000562'		MOV	IENV.ADDR,R0		5243
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		5244
000264	001403			BEQ	4\$		
000266	020027	000220		CMP	R0,#220		5245
000272	001033			BNE	5\$		
000274	010416			MOV	R4,(SP)	; M.INDEX,*	5249
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)	5250
000322	016062	000026G	000046	MOV	MSCP.ENV+26(R0),46(R2)	; *,*(R.ADDR)	5251
000330	016062	000044G	000050	MOV	MSCP.ENV+44(R0),50(R2)	; *,*(R.ADDR)	5252
000336	016062	000046G	000052	MOV	MSCP.ENV+46(R0),52(R2)	; *,*(R.ADDR)	5253
000344	016062	000030G	000024	MOV	MSCP.ENV+30(R0),24(R2)	; *,*(R.ADDR)	5254
000352	016062	000032G	000026	MOV	MSCP.ENV+32(R0),26(R2)	; *,*(R.ADDR)	5255
000360	005726			TST	(SP)+		5247

CZRC03
V02.0

CZRCDB0 RC25 DISK EXERCISER
RC25 INTERRUPT SERVICE ROUTINES

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC03.SRC;17

SEQ 0341
Page 156
(55)

000362	010116		5\$:	MOV	R1,(SP)	; R.INDEX,*	5261
000364	004737	000000G		JSR	PC,IN.IODQ		
000370	062706	000010		ADD	#10,SP		
000374	005704		6\$:	TST	R4	; M.INDEX	5234
000376	002404			BLT	7\$		5265
000400	010446			MOV	R4,-(SP)	; M.INDEX,*	5267
000402	004737	000000G		JSR	PC,PUT.ENV		
000406	005726			TST	(SP)+		
000410	000207		7\$:	RTS	PC		5175

; Routine Size: 133 words, Routine Base: \$CODE\$ + 16160
; Maximum stack depth per invocation: 12 words

```
: 5270 1  ROUTINE DATAGM : NOVALUE =
: 5271 1
: 5272 1  !+
: 5273 1  !
: 5274 1  !   THIS ROUTINE HANDLES ALL DATAGRAM (ERROR LOG) MESSAGES RECEIVED FROM
: 5275 1  !   THE RC25. IF THE DATAGRAM REFERENCES A PARTICULAR UNIT (AND IS NOT
: 5276 1  !   CONTROLLER-RELATED), THEN ITS OCCURRENCE IS TALLIED FOR THAT UNIT. IF
: 5277 1  !   THE USER DID NOT SUPPRESS THE PRINTING OF ERROR LOG MESSAGES DURING THE
: 5278 1  !   SW DIALOG, THEN EACH RELEVANT FIELD IS IDENTIFIED AND PRINTED. IF THE
: 5279 1  !   DATAGRAM INDICATES AN ECC ERROR, THEN IT IS TALLIED FOR THE APPROPRIATE
: 5280 1  !   UNIT.
: 5281 1  !
: 5282 1  !   IMPLICIT INPUTS:
: 5283 1  !       IENV_ADDR - ADDRESS OF MSCP ENVELOPE CONTAINING ERROR LOG
: 5284 1  !       MESSAGE
: 5285 1  !-
: 5286 2  BEGIN
: 5287 2
: 5288 2  LITERAL
: 5289 2      EC1 = 8,           ! SUB-CODE VALUE FOR 1-SYMBOL ECC ERROR
: 5290 2      EC8 = 15,        ! SUB-CODE VALUE FOR 8-SYMBOL ECC ERROR
: 5291 2      ECO = 4;         ! SUB-CODE VALUE FOR ECC-FIELD-ONLY ERROR
: 5292 2
: 5293 2  BIND
: 5294 2      FMT_CE = UPLIT (%ASCIZ'%ACONTROLLER ERROR%N'),
: 5295 2      FMT_HMA = UPLIT (%ASCIZ'%AHOST MEMORY ACCESS ERROR%N'),
: 5296 2      FMT_SDE = UPLIT (%ASCIZ'%ASMALL DISK ERROR%N');
: 5297 2
: 5298 2  OWN
: 5299 2      FMT_TB : VECTOR [5] INITIAL (FMT_CE, FMT_HMA, 0, 0, FMT_SDE),
: 5300 2      STC_TB : VECTOR [12] INITIAL (STC_00, STC_01, STC_02, STC_03, STC_04,
: 5301 2          STC_05, STC_06, STC_07, STC_08, STC_09, STC_10, STC_11);
: 5302 2
: 5303 2  LOCAL
: 5304 2      FMT : WORD,         ! FORMAT CODE
: 5305 2      EVC : WORD,         ! EVENT CODE
: 5306 2      SBC : WORD,         ! SUB-CODE
: 5307 2      PLAD : WORD,        ! PLATTER ADDRESS (MSCP UNIT NO.)
: 5308 2      UNIT : SIGNED WORD, ! DRS UNIT NUMBER
: 5309 2      TAD : REF BLOCK [TALLY_LEN, WORD] FIELD (T_FIELDS); !TALLY ADDRESS
: 5310 2
: 5311 2      FMT = .IENV_ADDR [FORMAT];    ! GET FORMAT CODE
: 5312 2      EVC = .IENV_ADDR [EVENT];     ! GET EVENT CODE
: 5313 2      SBC = .IENV_ADDR [SUBC];     ! GET SUB-CODE
: 5314 2      IF .FMT EQLU FM_SDE         ! IF SMALL DISK ERROR
: 5315 2      THEN                          ! THEN
: 5316 3          BEGIN
: 5317 3
: 5318 3          PLAD = .IENV_ADDR [PL_ADDR];    ! GET PLATTER ADDRESS (MSCP UNIT NO.)
: 5319 3          UNIT = -1;
: 5320 3          INCR OFFSET FROM (0 + OF_UN) TO (3 + OF_UN) DO    ! FOR EACH CST UNIT
: 5321 4              BEGIN
: 5322 4
: 5323 4                  IF .PLAD EQLU .ICST_ADDR [.OFFSET, P_ADDR] ! IF PLAT ADDR MATCHES CST ENTRY
: 5324 4                  THEN                                          ! THEN
: 5325 5                      BEGIN
: 5326 5
```



```
: 5327 5          UNIT = .ICST_ADDR [.OFFSET, P_UNIT];          ! GET DRS UNIT NUMBER
: 5328 5          EXITLOOP;                                     ! DONE SEARCH
: 5329 5
: 5330 4          END;
: 5331 4
: 5332 3          END;                                         ! PLATTER ADDRESS SEARCH LOOP
: 5333 3
: 5334 3          IF .UNIT GEQ 0                               ! IF UNIT NUMBER WAS FOUND
: 5335 3          THEN                                         ! THEN
: 5336 4          BEGIN
: 5337 4
: 5338 4          TAD = TALLY + (.UNIT * TALLY_LEN * 2);      ! CALCULATE UNIT'S TALLY ADDRESS
: 5339 4          TAD [ER_LOG] = .TAD [ER_LOG] + 1;          ! INCR ERROR LOG COUNT
: 5340 4          IF .EVC EQLU ST_DAT                          ! IF EVENT CODE = DATA ERROR
: 5341 4          THEN                                         ! THEN
: 5342 5          BEGIN
: 5343 5
: 5344 5          IF .SBC EQLU ECO                               ! IF ECC-FIELD-ONLY
: 5345 5          THEN                                         ! THEN
: 5346 5          TAD [ECC_ONLY] = .TAD [ECC_ONLY] + 1       ! INCREMENT STAT
: 5347 5          ELSE                                         ! OTHERWISE
: 5348 6          BEGIN
: 5349 6
: 5350 7          IF ((.SBC GEQ EC1) AND (.SBC LEQ EC8))      ! IF SUB-CODE INDICATES ECC SYMBOL ERROR
: 5351 6          THEN                                         ! THEN
: 5352 7          BEGIN
: 5353 7
: 5354 7          TAD = TAD [ECC_1] + ((.SBC - 8) * 2);      ! CALCULATE EXACT ADDR OF ECC STAT
: 5355 7          .TAD = ..TAD + 1;                            ! INCREMENT STATISTIC
: 5356 7
: 5357 6          END;                                         ! IF SUB-CODE IS ECC ERROR
: 5358 6
: 5359 5          END;                                         ! IF NOT ECC-FIELD-ONLY ERROR
: 5360 5
: 5361 4          END;                                         ! IF EVENT CODE = DATA ERROR
: 5362 4
: 5363 3          END;                                         ! IF UNIT NUMBER WAS FOUND
: 5364 3
: 5365 2          END;                                         ! IF FORMAT = SMALL DISK ERROR
: 5366 2
: 5367 3          IF MANUAL AND (NOT (BIT_TST (SWP_FLAGS, SWF_SEL))) ! IF ATTENDED AND ERROR LOG PRINTING WAS NOT SUPPRESSED
: 5368 2          THEN                                         ! THEN
: 5369 3          BEGIN
: 5370 3
: 5371 3          PRINTF (EX_EL);                               ! "ERROR LOG MESSAGE RECEIVED:"
: 5372 3          PRINTF (EX_CRN, .IENV_ADDR [CRN_LO]);        ! "  CMD REF NUM: XXXXXX(O)"
: 5373 3          IF .FMT EQLU FM_SDE                          ! IF FORMAT CODE = SMALL DISK ERROR
: 5374 3          THEN                                         ! THEN
: 5375 3          PRINTF (EX_PA, .PLAD);                       ! "  PLATTER: XXX."
: 5376 3          PRINTF (EX_FMT);                             ! "  FORMAT: "
: 5377 3          SELECTONE .FMT OF
: 5378 3          SET
: 5379 3
: 5380 3          [FM_CNT, FM_BAD, FM_SDE] : PRINTF (&FMT_TB [.FMT]); ! PRINT TEXT
: 5381 3          [OTHERWISE] : PRINTF (EX_03, .FMT);         ! "XXX(O)"
: 5382 3
: 5383 3          TES;
```

```

: 5384 3
: 5385 3      PRINTF (EX_EVC);
: 5386 3      IF .EVC LEQU 11
: 5387 3      THEN
: 5388 4          PRINTF (.STC_TB [.EVC])
: 5389 3      ELSE
: 5390 3          PRINTF (EX_03, .EVC);
: 5391 3      PRINTF (EX_SB, .SBC);
: 5392 3      IF .FMT EQLU FM_BAD
: 5393 3      THEN
: 5394 3          PRINTF (EX_HMA, .IENV_ADDR [MA_LO], .IENV_ADDR [MA_HI]);
: 5395 3
: 5396 2      END;
: 5397 2
: 5398 1      END;

```

! " EVENT CODE: "
! IF EVENT CODE IS KNOWN
! THEN
! PRINT EVENT CODE TEXT
! OTHERWISE
! "XXX(O)"
! " SUB-CODE: XXXX(O)"
! IF HOST MEMORY ACCESS ERROR
! THEN
! PRINT HOST ADDRESS
! IF PRINTING WAS NOT SUPPRESSED
! ROUTINE DATAGM

```

002554
002554 000036'
002556 000064'
002560 000000
002562 000000
002564 0C0122'
002566 000000G
002570 000000G
002572 000000G
002574 000000G
002576 000000G
002600 000000G
002602 000000G
002604 000000G
002606 000000G
002610 000000G
002612 000000G
002614 000000G

```

FMT.TB: .PSECT \$GGG\$, RO
.WORD FMT.CE
.WORD FMT.HMA
.WORD 0
STC.TB: .WORD INT.SDE
.WORD STC.00
.WORD STC.01
.WORD STC.02
.WORD STC.03
.WORD STC.04
.WORD STC.05
.WORD STC.06
.WORD STC.07
.WORD STC.08
.WORD STC.09
.WORD STC.10
.WORD STC.11

```

000036
000036 045 101 103
000041 117 116 124
000044 122 117 114
000047 114 105 122
000052 040 105 122
000055 122 117 122
000060 045 116 000
000063 000
000064 045 101 110
000067 117 123 124
000072 040 115 105
000075 115 117 122
000100 131 040 101
000103 103 103 105
000106 123 123 040
000111 105 122 122
000114 117 122 045

```

P.AAC: .PSECT \$PLIT\$, RO, D
.ASCII /*AC/
.ASCII /ONT/
.ASCII /ROL/
.ASCII /LER/
.ASCII / ER/
.ASCII /ROR/
.ASCII /*N/<00>
P.AAD: .ASCII /*AH/
.ASCII /OST/
.ASCII / ME/
.ASCII /MOR/
.ASCII /Y A/
.ASCII /CCE/
.ASCII /SS /
.ASCII /ERR/
.ASCII /OR*/

CZRC03
V02.0

CZRCDB0 RC25 DISK EXERCISER
RC25 INTERRUPT SERVICE ROUTINES

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER\$1:[AZTEC.CZRCDB]CZRC03.SRC;17

SEQ 0345
Page 160
(56)

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

016572 .SBTTL DATAGM RC25 INTERRUPT SERVICE ROUTINES
.PSECT \$CODE\$, RO

000000	004137	000000G	DATAGM:	JSR	R1,\$SAVE5	:	5270
000004	024646			CMP	-(SP),-(SP)	:	
000006	013700	000562'		MOV	IENV.ADDR,RO	:	5311
000012	005004			CLR	R4	:	
000014	156004	000020		BISB	20(R0),R4	:	
000020	116003	000022		MOVB	22(R0),R3	:	
000024	042703	177740		BIC	#177740,R3	:	5312
000030	016005	000022		MOV	22(R0),R5	:	
000034	006205			ASR	R5	:	5313
000036	006205			ASR	R5	:	
000040	006205			ASR	R5	:	
000042	006205			ASR	R5	:	
000044	006205			ASR	R5	:	
000046	042705	174000		BIC	#174000,R5	:	
000052	005066	000002		CLR	2(SP)	:	
000056	020427	000004		CMP	R4,#4	:	5314
000062	001075			BNE	6\$:	
000064	005266	000002		INC	2(SP)	:	
000070	016016	000014		MOV	14(R0),(SP)	:	
000074	012702	177777		MOV	#-1,R2	:	
000100	012700	000006		MOV	#6,RO	:	
000104	010001		1\$:	MOV	RO,R1	:	5318
000106	063701	000564'		ADD	ICST.ADDR,R1	:	5319
000112	111146			MOVB	(R1),-(SP)	:	5320
000114	105066	000001		CLRB	1(SP)	:	5323
000120	026626	000002		CMP	2(SP),(SP)+	:	
000124	001005			BNE	2\$:	
000126	011102			MOV	(R1),R2	:	
000130	000302			SWAB	R2	:	5327
000132	042702	177740		BIC	#177740,R2	:	
000136	000406			BR	3\$:	
000140	062700	000002	2\$:	ADD	#2,RO	:	5325
000144	020027	000014		CMP	RO,#14	:	5320
000150	003755			BLE	1\$:	
000152	005702			TST	R2	:	
000154	002440		3\$:	BLT	6\$:	5334
000156	010246			MOV	R2,-(SP)	:	
000160	012746	000060		MOV	#60,-(SP)	:	5338

000164	004737	000000G		JSR	PC,BL\$MUL			
000170	052700	000000G		ADD	#TALLY,RO			
000174	005260	000032		INC	32(RO)		; *(TAD)	5339
000200	020327	000010		CMP	R3,#10		; EVC,*	5340
000204	001023			BNE	5\$			
000206	020527	000004		CMP	R5,#4		; SBC,*	5344
000212	001003			BNE	4\$			
000214	005260	000056		INC	56(RO)		; *(TAD)	5346
000220	000415			BR	5\$; SBC,*	5344
000222	020527	000010	4\$:	CMP	R5,#10		; SBC,*	5350
000226	002412			BLT	5\$			
000230	020527	000017		CMP	R5,#17		; SBC,*	
000234	003007			BGT	5\$			
000236	010502			MOV	R5,R2		; SBC,*	5354
000240	006302			ASL	R2			
000242	060002			ADD	RO,R2		; TAD,*	
000244	010200			MOV	R2,RO		; *,TAD	
000246	062700	000016		ADD	#16,RO		; *,TAD	
000252	005210			INC	(RO)		; TAD	5355
000254	022626		5\$:	CMP	(SP)+,(SP)+			5336
000256	104450		6\$:	TRAP	50			5367
000260	103165			BHIS	15\$			
000262	132737	000001	000000G	BITB	#1,SWP.FLAGS			
000270	0C1161			BNE	15\$			
000272	012746	000000G		MOV	#EX.EL,-(SP)			5371
000276	012746	000001		MOV	#1,-(SP)			
000302	010600			MOV	SP,RO		; SP,*	
000304	104417			TRAP	17			
000306	013700	000562'		MOV	IENV.ADDR,RO			5372
000312	016016	000010		MOV	10(RO),(SP)			
000316	012746	000000G		MOV	#EX.CRN,-(SP)			
000322	012746	000002		MOV	#2,-(SP)			
000326	010600			MOV	SP,RO		; SP,*	
000330	104417			TRAP	17			
000332	032766	000001	000012	BIT	#1,12(SP)			5373
000340	001411			BEQ	7\$			
000342	016616	000010		MOV	10(SP),(SP)		; PLAD,*	5375
000346	012746	000000G		MOV	#EX.PA,-(SP)			
000352	012746	000002		MOV	#2,-(SP)			
000356	010600			MOV	SP,RO		; SP,*	
000360	104417			TRAP	17			
000362	022626			CMP	(SP)+,(SP)+			
000364	012716	000000G	7\$:	MOV	#EX.FMT,(SP)			5376
000370	012746	000001		MOV	#1,-(SP)			
000374	010600			MOV	SP,RO		; SP,*	
000376	104417			TRAP	17			
000400	005704			TST	R4		; FMT	5380
000402	002403			BLT	8\$			
000404	020427	000001		CMP	R4,#1		; FMT,*	
000410	003403			BLE	9\$			
000412	020427	000004	8\$:	CMP	R4,#4		; FMT,*	
000416	001012			BNE	10\$			
000420	010400		9\$:	MOV	R4,RO		; FMT,*	
000422	006300			ASL	RO			
000424	016016	002554'		MOV	FMT.TB(RO),(SP)			
000430	012746	000001		MOV	#1,-(SP)			
000434	010600			MOV	SP,RO		; SP,*	

CZRC03
V02.0

CZRCDB0 RC25 DISK EXERCISER
RC25 INTERRUPT SERVICE ROUTINES

14-Jun-1985 09:41:21
14-Jun-1985 09:33:42

VAX-11 Bliss-16 V4.0-579
USER#1:(AZTEC.CZRCDB)CZRC03.SRC;17

SEQ 0347
Page 162
(56)

000436	104417			TRAP	17			
000440	005726			TST	(SP)+			
000442	000410			BR	11\$			
000444	010416		10\$:	MOV	R4,(SP)		; FMT,*	5377
000446	012746	000000G		MOV	#EX.03,-(SP)			5381
000452	012746	000002		MOV	#2,-(SP)			
000456	010600			MOV	SP,R0		; SP,*	
000460	104417			TRAP	17			
000462	022626			CMP	(SP)+,(SP)+			
000464	012716	000000G	11\$:	MOV	#EX.EVC,(SP)			5385
000470	012746	000001		MOV	#1,-(SP)			
000474	010600			MOV	SP,R0		; SP,*	
000476	104417			TRAP	17			
000500	020327	000013		CMP	R3,#13		; EVC,*	5386
000504	101011			BHI	12\$			
000506	010300			MOV	R3,R0		; EVC,*	5388
000510	006300			ASL	R0			
000512	016016	002566'		MOV	STC.TB(R0),(SP)			
000516	012746	000001		MOV	#1,-(SP)			
000522	010600			MOV	SP,R0		; SP,*	
000524	104417			TRAP	17			
000526	000410			BR	13\$			5386
000530	010316		12\$:	MOV	R3,(SP)		; EVC,*	5390
000532	012746	000000G		MOV	#EX.03,-(SP)			
000536	012746	000002		MOV	#2,-(SP)			
000542	010600			MOV	SP,R0		; SP,*	
000544	104417			TRAP	17			
000546	005726			TST	(SP)+			
000550	010516		13\$:	MOV	R5,(SP)		; SBC,*	5391
000552	012746	000000G		MOV	#EX.SB,-(SP)			
000556	012746	000002		MOV	#2,-(SP)			
000562	010600			MOV	SP,R0		; SP,*	
000564	104417			TRAP	17			
000566	020427	000001		CMP	R4,#1		; FMT,*	5392
000572	001016			BNE	14\$			
000574	013700	000562'		MOV	IENV.ADDR,R0			5394
000600	016016	000042		MOV	42(R0),(SP)			
000604	016046	000040		MOV	40(R0),-(SP)			
000610	012746	000000G		MOV	#EX.HMA,-(SP)			
000614	012746	000003		MOV	#3,-(SP)			
000620	010600			MOV	SP,R0		; SP,*	
000622	104417			TRAP	17			
000624	062706	000006		ADD	#6,SP			
000630	062706	000022	14\$:	ADD	#22,SP			5369
000634	022626		15\$:	CMP	(SP)+,(SP)+			5270
000636	000207			RTS	PC			

; Routine Size: 208 words, Routine Base: \$CODE\$ + 16572
; Maximum stack depth per invocation: 22 words

; 5399 1
; 5400 1 END
; 5401 1
; 5402 0 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\$	3981	RO , I , LCL, REL, CON
\$PLIT\$	52	RO , D , LCL, REL, CON

Library Statistics

File	Total	Symbols Loaded	Percent	Pages Mapped	Processing Time
USER#1:[AZTEC.CZRCDB]CZRCDL.L16;6	276	249	90	16	00:00.2

COMMAND QUALIFIERS

BLISS/PDP11 CZRC03.SRC/LIST/EN:NOEIS

: Size: 3981 code + 763 data words
: Run Time: 06:47.6
: Elapsed Time: 07:16.0
: Lines/CPU Min: 795
: Lexemes/CPU-Min: 6489
: Memory Used: 396 pages
: Compilation Complete


```

: 0001 0  MODULE CZRC4 (
: 0002 0      *TITLE 'CZRCDB0 RC25 DISK EXERCISER'
: 0003 0      IDENT = 'V02.0',
: 0004 0      ADDRESSING_MODE (ABSOLUTE)
: 0005 0      ) =
: 0006 1  BEGIN
: 0043 1  *SBTTL 'LASTAD AND SETUP'
: 0044 1
: 0045 1  REQUIRE 'BLSMAC.REQ';           ! DIAGNOSTIC SUPERVISOR LIBRARY
: 1534 1
: 1535 2  LASTAD
: 1536 2
: 1537 2  BGNSETUP (2);
: 1538 2
: P 1539 2  BGNPTAB
: P 1540 2      *0'172150', *0'154', 5, 0      ! IP, VECTOR, BR, PLAT ADDR
: 1541 2  ENDPTAB
: 1542 2
: P 1543 2  BGNPTAB
: P 1544 2      *0'172150', *0'154', 5, 1      ! IP, VECTOR, BR, PLAT ADDR
: 1545 2  ENDPTAB
: 1546 2
: 1547 1  ENDSETUP
    
```

```

000000          .TITLE CZRC4 CZRCDB0 RC25 DISK EXERCISER
000000          .IDENT /V02.0/
000000          .ENABL AMA
000000          .PSECT $XYZ$, RO
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

000000 000207 .SBTTL $END.LINK LASTAD AND SETUP
$END.LINK::
RTS PC ;
    
```

```

: Routine Size: 1 word, Routine Base: $XYZ$ + 0036
: Maximum stack depth per invocation: 0 words
    
```

CZRC04
V02.0

CZRCDB0 RC25 DISK EXERCISER
LASTAD AND SETUP

14-Jun-1985 09:49:05
14-Jun-1985 09:34:28

VAX-11 Bliss-16 V4.0-579
USER#1:[AZTEC.CZRCDB]CZRC04.SRC;4

SEQ 0350
Page 3
(2)

: 1548 1
: 1549 1 END
: 1550 1
: 1551 0 ELUDOM

PSECT SUMMARY

:
: Psect Name Words Attributes
: \$XYZ\$ 16 RO , I , LCL, REL, CON

COMMAND QUALIFIERS

: BLISS/PDP11 CZRC04.SRC/LIST/EN:NOEIS

: Size: 1 code + 15 data words
: Run Time: 00:22.1
: Elapsed Time: 00:22.7
: Lines/CPU Min: 4207
: Lexemes/CPU-Min: 22155
: Memory Used: 103 pages
: Compilation Complete