

RC25

RC25 DISK FORMATTER
CZRCHAO

AH-T275A-MC
FICHE 1 OF 2

OCT 1983
COPYRIGHT © 1983
MADE IN USA



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
501	502	503	504	505	506	507	508	509	510	511	512	513	514	515	516	517	518	519	520	521	522	523	524	525	526	527	528	529	530	531	532	533	534	535	536	537	538	539	540	541	542	543	544	545	546	547	548	549	550	551	552	553	554	555	556	557	558	559	560	561	562	563	564	565	566	567	568	569	570	571	572	573	574	575	576	577	578	579	580	581	582	583	584	585	586	587	588	589	590	591	592	593	594	595	596	597	598	599	600
601	602	603	604	605	606	607	608	609	610	611	612	613	614	615	616	617	618	619	620	621	622	623	624	625	626	627	628	629	630	631	632	633	634	635	636	637	638	639	640	641	642	643	644	645	646	647	648	649	650	651	652	653	654	655	656	657	658	659	660	661	662	663	664	665	666	667	668	669	670	671	672	673	674	675	676	677	678	679	680	681	682	683	684	685	686	687	688	689	690	691	692	693	694	695	696	697	698	699	700
701	702	703	704	705	706	707	708	709	710	711	712	713	714	715	716	717	718	719	720	721	722	723	724	725	726	727	728	729	730	731	732	733	734	735	736	737	738	739	740	741	742	743	744	745	746	747	748	749	750	751	752	753	754	755	756	757	758	759	760	761	762	763	764	765	766	767	768	769	770	771	772	773	774	775	776	777	778	779	780	781	782	783	784	785	786	787	788	789	790	791	792	793	794	795	796	797	798	799	800
801	802	803	804	805	806	807	808	809	810	811	812	813	814	815	816	817	818	819	820	821	822	823	824	825	826	827	828	829	830	831	832	833	834	835	836	837	838	839	840	841	842	843	844	845	846	847	848	849	850	851	852	853	854	855	856	857	858	859	860	861	862	863	864	865	866	867	868	869	870	871	872	873	874	875	876	877	878	879	880	881	882	883	884	885	886	887	888	889	890	891	892	893	894	895	896	897	898	899	900
901	902	903	904	905	906	907	908	909	910	911	912	913	914	915	916	917	918	919	920	921	922	923	924	925	926	927	928	929	930	931	932	933	934	935	936	937	938	939	940	941	942	943	944	945	946	947	948	949	950	951	952	953	954	955	956	957	958	959	960	961	962	963	964	965	966	967	968	969	970	971	972	973	974	975	976	977	978	979	980	981	982	983	984	985	986	987	988	989	990	991	992	993	994	995	996	997	998	999	1000

RC25

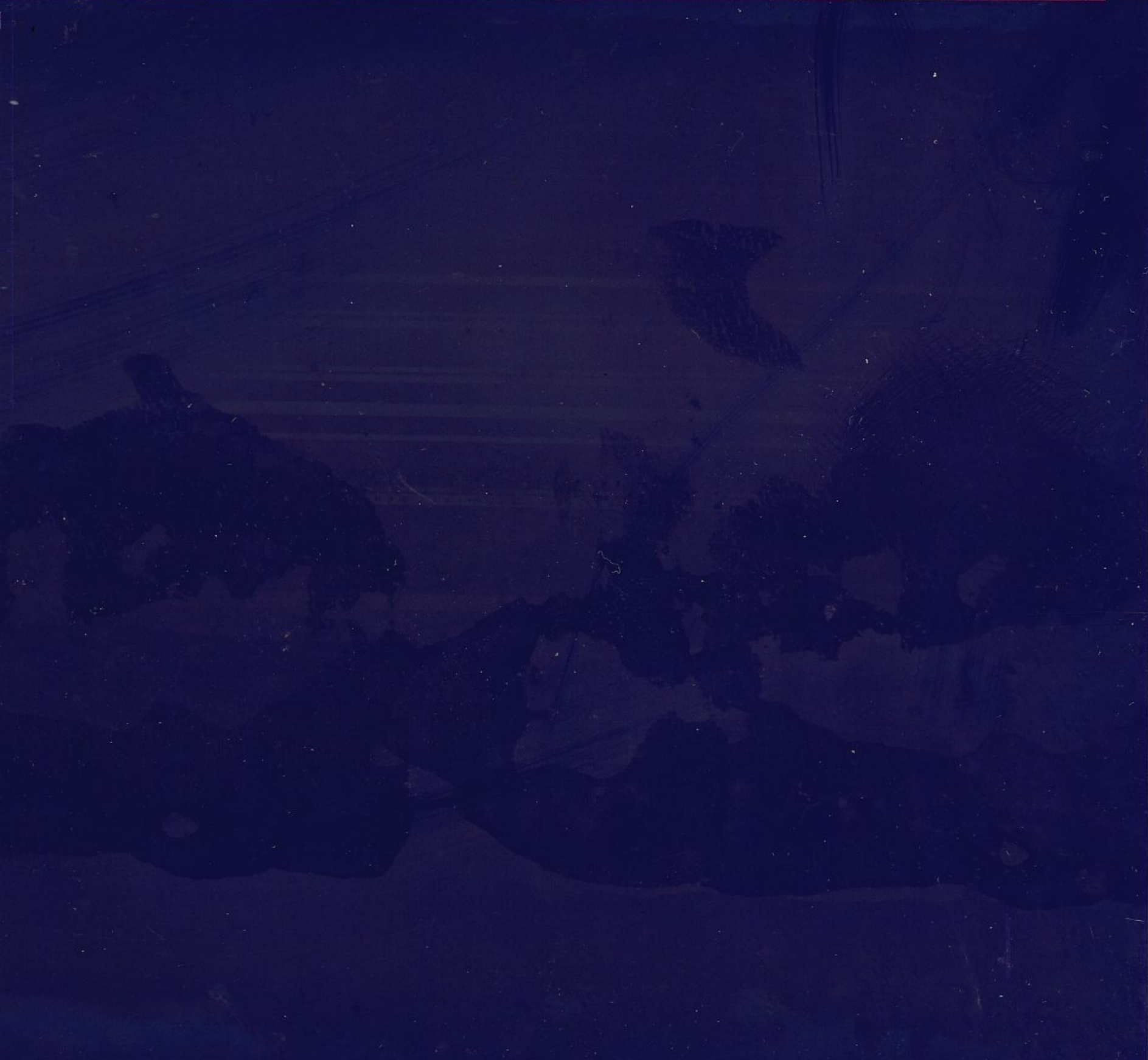
RC25 DISK FORMATTER
CZRCHAO

AH-T275A-MC
FICHE 2 OF 2

OCT 1983
COPYRIGHT © 1983
MADE IN USA



Table with 5 columns and 15 rows of data, including headers like 'STATION', 'DATE', 'TIME', 'FREQ', and 'MODE'. The content is mostly illegible due to low contrast.



CZRCH1

CZRCH RC25 DISK FORMATTER

11-Jul-1983 16:04 50
8-Jul-1983 13:42: 9

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH1.B16;4 (1)

0001 MODULE CZRCH1 (%TITLE 'CZRCH RC25 DISK FORMATTER'
0002 IDENT = 'REV A PATCH 00') =

0003
0004 %SBTTL 'USER DOCUMENTATION'

C 0005 %(
C 0006
C 0007
C 0008
C 0009
C 0010
C 0011
C 0012
C 0013
C 0014
C 0015
C 0016
C 0017
C 0018
C 0019
C 0020
C 0021
C 0022
C 0023
C 0024
C 0025
C 0026
C 0027
C 0028
C 0029
C 0030
C 0031
C 0032
C 0033
C 0034
C 0035
C 0036
C 0037
C 0038
C 0039
C 0040
C 0041
C 0042
C 0043
C 0044
C 0045
C 0046
C 0047
C 0048
C 0049
C 0050

IDENTIFICATION

PRODUCT CODE: AC-T274A-MC

PRODUCT NAME: CZRCHAO RC25 DISK FORMATTER

PRODUCT DATE: 13-JUL-83

MAINTAINER : Disk Engineering

AUTHOR : D.W.Neale

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS DOCUMENT.

NO RESPONSIBILITY IS ASSUMED FOR THE USE OR RELIABILITY OF SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL OR ITS AFFILIATED COMPANIES.

COPYRIGHT (C) 1983 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL	PDP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	

TABLE OF CONTENTS

..	C 0051		
..	C 0052		
..	C 0053		
..	C 0054		
..	C 0055		
..	C 0056	0.0	MAINTENANCE HISTORY
..	C 0057	1.0	GENERAL INFORMATION
..	C 0058	1.1	PROGRAM ABSTRACT
..	C 0059	1.2	PERFORMANCE GOALS
..	C 0060	1.3	SYSTEM REQUIREMENTS
..	C 0061	1.4	RELATED DOCUMENTS AND STANDARDS
..	C 0062	1.5	DIAGNOSTIC HIERARCHY PREREQUISITES
..	C 0063	1.6	ASSUMPTIONS
..	C 0064	1.7	PRODUCT USERS AND USES
..	C 0065	1.8	RESTRICTIONS
..	C 0066	1.9	BAD BLOCK DEFINITION
..	C 0067	1.10	SOURCE MODULE DEFINITION
..	C 0068		
..	C 0069	2.0	OPERATING INSTRUCTIONS
..	C 0070	2.1	COMMANDS
..	C 0071	2.2	SWITCHES
..	C 0072	2.3	FLAGS
..	C 0073	2.4	HARDWARE QUESTIONS
..	C 0074	2.5	SOFTWARE QUESTIONS
..	C 0075	2.6	EXTENDED P-TABLE DIALOGUE
..	C 0076	2.7	QUICK STARTUP PROCEDURE
..	C 0077	2.8	PID (PROCESS INDICATOR) WORDS
..	C 0078	2.9	HOST/DM FORMATTER RUN TIME MESSAGES
..	C 0079		
..	C 0080	3.0	ERROR INFORMATION
..	C 0081	3.1	TYPES OF ERRORS MESSAGES
..	C 0082	3.2	SPECIFIC ERROR MESSAGES
..	C 0083		
..	C 0084	4.0	PERFORMANCE AND PROGRESS REPORTS
..	C 0085		
..	C 0086	5.0	DEVICE INFORMATION TABLES
..	C 0087		
..	C 0088	6.0	DM FORMATTER TEST SUMMARIES
..	C 0089	6.1	REFORMAT MODE PROCEDURE
..	C 0090	6.2	RESTORE MODE PROCEDURE
..	C 0091	6.3	RECONSTRUCT MODE PROCEDURE
..	C 0092		
..	C 0093	7.0	HOST TEST SUMMARIES
..	C 0094	7.1	ATTENDED MODE
..	C 0095	7.2	UN-ATTENDED MODE
..	C 0096		
..	C 0097	8.0	APPENDIX
..	C 0098		APPENDIX A GLOSSARY
..	C 0099		APPENDIX B BAD BLOCK REPLACEMENT
..	C 0100		APPENDIX C FCT
..	C 0101		APPENDIX D RCT
..	C 0102		APPENDIX E DATA PATTERNS
..	C 0103		APPENDIX F PINP-PONG ALGORITHM
..	C 0104		APPENDIX G CREATING CZRCHA DM CODE
..	C 0105		APPENDIX H CREATING CZRCHA HOST CODE

CZRCH1
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
USER DOCUMENTATION

11-Jul-1983 16:04:50
8-Jul-1983 13:42:19

VAX-11 Bliss-16 V3-555
SPIDER\$USERS:[NEALE.AZTEC]CZRCH1.B16;4 (3)

: C 0106
: C 0107
: C 0108
: C 0109
: C 0110
: C 0111

0.0 MAINTENANCE HISTORY

Modified By: D.W.Neale

Date: 13-Jul-83

Version: 1.0

Original release

C 0112 1.0 GENERAL INFORMATION

C 0113
C 0114 1.1 PROGRAM ABSTRACT

C 0115
C 0116 This program will prepare RC25 media for use as addressable storage by
C 0117 providing headers and replacing of bad blocks. This formatter will be
C 0118 composed of two sections; a host-resident section, and a
C 0119 controller-resident DM code section.

C 0120
C 0121 The host section will serially format up to sixteen RC25 subsystems by
C 0122 downline-loading the DM code, and monitor the task. The downline
C 0123 section will consist of overlays of DM code. The first of these will be
C 0124 down-line loaded to the drive itself, and accomplish the actual business
C 0125 of formatting, calling in additional overlays as needed, asking software
C 0126 parameters questions and printing formatter error and informational
C 0127 messages.

C 0128
C 0129 There are three general modes of DM formatter operation and they are:

- C 0130 o REFORMAT - This mode is used to format a medium which has been
- C 0131 previously formatted, and is being reformatted to clear
- C 0132 existing data or to change the mode of the medium to 512 bytes
- C 0133 per sector. It assumes that the FCT is still intact.
- C 0134
- C 0135 o RESTORE - This mode will only be run by DIGITAL Manufacturing
- C 0136 personnel. It provides an external copy of the FCT, produced
- C 0137 when the disk was manufactured and stored offline, to the
- C 0138 RC25 formatter.
- C 0139
- C 0140 o RECONSTRUCT - This mode is used when none of the other modes
- C 0141 is possible. It detects bad blocks by performing repetitive
- C 0142 read checks of each sector. For this reason, a RECONSTRUCT
- C 0143 run takes considerably longer than the other modes.
- C 0144
- C 0145

C 0146 There are two general modes of HOST operation and they are:

- C 0147 o ATTENDED MODE- This is where an operator must be present at the
- C 0148 console terminal to respond to DM formatter software parameter
- C 0149 questions. In this HOST mode the operator can choose any of the
- C 0150 three DM formatting modes.
- C 0151
- C 0152 o UN-ATTENDED MODE- This is where the HOST will automatically answer
- C 0153 DM formatter software parameter questions to perform a REFORMAT
- C 0154 mode to all units selected units via the hardware P_Tables.
- C 0155
- C 0156

C 0157 1.2 Performance Goals

C 0158 Before initiating the format process, simple checks will be made to
C 0159 assure the object drive exists, and that communications are possible
C 0160 between host and downline program sections.

C 0161 To maximize throughput:

- C 0162 o For full Track reads a half Track skew will be employed. That
- C 0163 is, for even physical LBN track numbers each block number in
- C 0164 Track N+1 will occur 16 block times later than in Track N.
- C 0165
- C 0166
- C 0167
- C 0168

o During bad block replacement, primary replacement blocks (RBN) will be used first. A second pass will then use secondary RBN's for any additional bad blocks.

All bad blocks, whether induced by the manufacturing process or wear-caused, will be revectored to Replacement Blocks (RBN). This will assure the full compliment of 25,451 blocks per surface for host applications use.

Product reliability is enhanced through use of four copies of the FCT and RCT.

1.3 SYSTEM REQUIREMENTS

The following are required to run the RC25 Formatter on PDP-11 Systems:

- * A PDP-11 series CPU
- * A minimum of 28K words of main memory
- * An RC25 subsystem
- * An XXDP+ load medium
- * A console terminal
- * Diagnostic Supervisor

1.4 RELATED DOCUMENTS AND STANDARDS

- | | | |
|-----|-------------|--|
| 1. | DUP.V05 | Diagnostic/Utilities Protocol version 0.5 |
| 2. | UQSSP.DOC | Unibus/Q-Bus Storage Systems port V1.5 |
| 4. | MSCP.DOC | Mass Storage Control Protocol |
| 5. | AZTEC.DOC | RC25 Microcode Documentation |
| 6. | DSDFV11.DOC | Standard Disk Format Specification version 1.1 |
| 7. | SUPPRGC.DOC | PDP-11 Diagnostic Supervisor Programming guide |
| 8. | CHQUSD.SEQ | XXDP+/SUPR user manual |
| 9. | BLISS-16 | Language Guide |
| 10. | BLISS-16 | User's Guide |

1.5 DIAGNOSTIC HIERARCHY PREREQUISITES

A fully functional CPU, main memory and RC25 subsystem are required.

1.6 ASSUMPTIONS

Prior to the first formatting of a media, an exhaustive surface analysis is assumed to have taken place to detect bad spots caused by the manufacturing process. The results of this analysis is assumed to have been written as two FCT images per FCT track on the RC25 media.

1.7 PRODUCT USERS AND USES

This program is intended for use on media which has undergone a surface analysis process, resulting in the writing of Factory Control Tables (FCT) on the media.

The program will be used by Engineering and Manufacturing Groups as a product development tool, and by Field Service personnel for media recovery.

.....
C 0169
C 0170
C 0171
C 0172
C 0173
C 0174
C 0175
C 0176
C 0177
C 0178
C 0179
C 0180
C 0181
C 0182
C 0183
C 0184
C 0185
C 0186
C 0187
C 0188
C 0189
C 0190
C 0191
C 0192
C 0193
C 0194
C 0195
C 0196
C 0197
C 0198
C 0199
C 0200
C 0201
C 0202
C 0203
C 0204
C 0205
C 0206
C 0207
C 0208
C 0209
C 0210
C 0211
C 0212
C 0213
C 0214
C 0215
C 0216
C 0217
C 0218
C 0219
C 0220
C 0221
C 0222
C 0223
C 0224
C 0225

CZRCH1
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
USER DOCUMENTATION

11-Jul-1983 16:04:50
8-Jul-1983 13:42:19

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH1.B16;4 (4)

: C 0226
: C 0227
: C 0228
: C 0229
: C 0230

This program is a utility and is not intended for use as a diagnostic.
For future use, the DM formatter code portion of this program will be
used by operating systems as a format utility program.

CZRCH1
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
USER DOCUMENTATION

11-Jul-1983 16:04:50
8-Jul-1983 13:42:19

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH1.B16;4 (5)

```

:      C 0231      1.8 RESTRICTIONS
:      C 0232
:      C 0233      This program will be host loaded.
:      C 0234
:      C 0235      With the exception of the FCT and inner DBN tracks, all information previously
:      C 0236      recorded on the media will be destroyed. This includes all LBN, RBN and RCT
:      C 0237      blocks.
:      C 0238
:      C 0239      The format process will reset the forced error indicator (EDC field) of
:      C 0240      each LBN, DBN, and used RBN and set the forced error indicators of all
:      C 0241      unused RBN's.
:      C 0242
:      C 0243      The FCT preamble will indicate zero for both the size of the 576 byte
:      C 0244      replacement table, and the size of the controller scratch area.
:      C 0245
:      C 0246      All sectors will have 512 byte data fields.
:      C 0247
:      C 0248      This Host program will run on only PDP-11 family CPUs which support the
:      C 0249      Diagnostic Supervisors.
:      C 0250

```



```

:      C 0251      1.9 Bad Block Definition
:      C 0252
:      C 0253      The Formatter's bad block replacement convention is as follows:
:      C 0254
:      C 0255      For Read operations:
:      C 0256      -----
:      C 0257      Any ECC, read or compare data failure will result in four retries.
:      C 0258
:      C 0259      Failure of any retries will be considered a Hard error and
:      C 0260      the block in error will be replaced.
:      C 0261
:      C 0262      The block in question will be considered good and no revectoring
:      C 0263      will be done if all retries are successful.
:      C 0264
:      C 0265      For Write operations:
:      C 0266      -----
:      C 0267      A Write failure will result in four retries.
:      C 0268
:      C 0269      Failure of any retries will be considered a Hard error and
:      C 0270      the block in error will be replaced.
:      C 0271
:      C 0272      The block in question will be considered good and no revectoring
:      C 0273      will be done if all retries are successful.

```

```

: C 0274      1.10 SOURCE MODULE DEFINITION
: C 0275
: C 0276      The following list all source module comprising CZRCHA and a description
: C 0277      of thier contents.
: C 0278
: C 0279      1. CZRCH0.R16      Host source bliss library
: C 0280      2. CZRCH1.B16      Program source document module
: C 0281      3. CZRCH2.B16      Global data module
: C 0282      4. CZRCH3.B16      Init code source module
: C 0283      5. CZRCH4.B16      Test source module
: C 0284      6. CZRCH5.B16      Global routine source module
: C 0285      7. CZRCH6.B16      DM formatter down-line load executable module
: C 0286      8. CZRCH7.B16      DRS> Last address source module
: C 0287      9. CZRCH8.EXE      DMCONV.EXE Dmconv exicutable code
: C 0288      10. CZRCH9.BP2      DMCONV.BP2 Dmconv source module
: C 0289      11. CZRCH10.MAC     AZFMTR.MAC DM formatter source macro module
: C 0290      12. CZRCH11.SAV    AZFMTR.SAV DM formatter executable code
: C 0291      13. CZRCH12.2D    Project functional specification run-off source module
: C 0292      14. CZRCHA.COM     Project indirect command file
: C 0293      15. CZRCHA.DOC    Program document listing file
: C 0294      16. CZRCHA.SEQ    Host sequence listing
: C 0295      17. CZRCHA.BIN    Program XXDP+ .bin file
: C 0296      18. CZRCH13.LST   DM formatter listing file
: C 0297

```


C 0298
C 0299
C 0300
C 0301
C 0302
C 0303
C 0304
C 0305
C 0306
C 0307
C 0308
C 0309
C 0310
C 0311
C 0312
C 0313
C 0314
C 0315
C 0316
C 0317
C 0318
C 0319
C 0320
C 0321
C 0322
C 0323
C 0324
C 0325
C 0326
C 0327
C 0328
C 0329
C 0330
C 0331
C 0332
C 0333
C 0334
C 0335
C 0336
C 0337
C 0338
C 0339
C 0340
C 0341
C 0342
C 0343
C 0344
C 0345
C 0346
C 0347
C 0348
C 0349
C 0350
C 0351
C 0352
C 0353
C 0354

2.0 OPERATING INSTRUCTIONS

This section contains a brief description of the runtime services. for detailed information, refer to the XXDP+ user's manual (CHQUS).

2.1 COMMANDS

There are eleven legal commands for the diagnostic runtime services (supervisor). This section lists the commands and gives a very brief description of them. The XXDP+ user's manual has more details.

COMMAND	EFFECT
START	Start the diagnostic from an initial state
RESTART	Start the diagnostic without initializing
CONTINUE	Continue at test that was interrupted (after ^C) Attempts to continue after a control c will result in the present units formatting to be aborted and the next logical units formatting to commence.
PROCEED	Continue from an error halt. This host program is coded to be a utility and not a diagnostic, therefor no error macros are used. For this reason this DRS> command has no meaning.
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. Report summary coding is remote DM program driven and is not implemented in the host.
DISPLAY	Type a list of all device information
FLAGS	Type the state of all flags (see section 2.3)
ZFLAGS	Clear all flags (see section 2.3)

A command can be recognized by the first three characters. So you may, for example, type "STA" instead of "START".

2.2 SWITCHES

There are several switches which are used to modify supervisor operation. These switches are appended to the legal commands. All of the legal switches are tabulated below with a brief description of each. In the descriptions below, a decimal number is designated by 'DDDD'.

SWITCH	EFFECT
--------	--------

```

: C 0355           /TESTS:LIST      Execute only those tests specified in
: C 0356           the list. List is a string of test
: C 0357           numbers, for example - /TESTS:1:5:7-10.
: C 0358           This list will cause tests 1,5,7,8,9,10 to
: C 0359           be run. All other tests will not be run.
: C 0360
: C 0361           /PASS:DDDDD      Execute DDDDD passes (DDDDD = 1 to 64000)
: C 0362           The host code will perform only one pass.
: C 0363           At completion of pass one the program will
: C 0364           be terminated regardless of this switch.
: C 0365
: C 0366           /FLAGS:FLGS      Set specified flags. flags are described
: C 0367           in section 2.3.
: C 0368
: C 0369           /EOP:DDDDD      Report end of pass message after every
: C 0370           DDDDD passes only. (DDDDD = 1 to 64000)
: C 0371           The host code will perform only one pass.
: C 0372           At completion of pass one the program will
: C 0373           be terminated regardless of this switch.
: C 0374
: C 0375           /UNITS:LIST      TEST/ADD/DROP only those units specified
: C 0376           in the list. List example - /UNITS:0:5:10-12
: C 0377           use units 0,5,10,11,12 (unit numbers = 0-63)
: C 0378
: C 0379
: C 0380
: C 0381
: C 0382
: C 0383
: C 0384
: C 0385
: C 0386
: C 0387
: C 0388
: C 0389
: C 0390
: C 0391
: C 0392
: C 0393
: C 0394
: C 0395
: C 0396
: C 0397
: C 0398
: C 0399
: C 0400
: C 0401
: C 0402
: C 0403
: C 0404
: C 0405
: C 0406
: C 0407
: C 0408
: C 0409
: C 0410
: C 0411

```

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.
3. The end of pass messages will be printed after each 100 passes only.

A Switch can be recognized by the first three characters. You may, for example, type '/TES:1-5' instead of '/TESTS:1-5'.

Below is a table that specifies which switches can be used by each command.

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

2.3 FLAGS

Flags are used to set up certain operational parameters such as looping on error. All flags are cleared at startup and remain cleared until explicitly set using the flags switch. Flags are also cleared after a start command unless set using the flag switch. The ZFLAGS command may also be used to clear all flags, with the exception of the START and ZFLAGS commands. No commands affect the state of the flags; they remain set or cleared as specified by the last flag switch.

FLAG	EFFECT
HOE	Halt on error - control is returned to runtime services command mode
LOE	Loop on error
IER*	Inhibit all error reports
IBR*	Inhibit all error reports except first level (first level contains error type, number, PC, test and unit)
IXR*	Inhibit extended error reports (those called by PRINTX macro's)
PRI	Direct messages to line printer
PNT	Print test number as test executes
BOE	'BELL' on error
UAM	Unattended mode (no manual intervention)
ISR	Inhibit statistical reports (does not apply to diagnostics which do not support statistical reporting)
IDR	Inhibit program dropping of units
ADR	Execute autodrop code
LOT	Loop on test
EVL	Execute evaluation (on diagnostics which have evaluation support)

*error messages are described in section 3.1

See the XXDP+ user's manual for more details on flags. You may specify more than one flag with the flag switch. For example, to cause the program to loop on error, inhibit error reports and type a 'BELL' on error, you may use the following string:

C 0412
C 0413
C 0414
C 0415
C 0416
C 0417
C 0418
C 0419
C 0420
C 0421
C 0422
C 0423
C 0424
C 0425
C 0426
C 0427
C 0428
C 0429
C 0430
C 0431
C 0432
C 0433
C 0434
C 0435
C 0436
C 0437
C 0438
C 0439
C 0440
C 0441
C 0442
C 0443
C 0444
C 0445
C 0446
C 0447
C 0448
C 0449
C 0450
C 0451
C 0452
C 0453
C 0454
C 0455
C 0456
C 0457
C 0458
C 0459
C 0460
C 0461
C 0462
C 0463
C 0464
C 0465
C 0466
C 0467
C 0468

/FLAGS:LOE:IER:BOE

2.4 HARDWARE QUESTIONS

When a Diagnostic is started, the runtime services will prompt the user for hardware information by typing "CHANGE HW (L) ?". You must answer 'Y' after a start command unless the hardware information has been 'preloaded' using the setup utility (see Chapter 6 of the XXDP+ user's Manual). When you answer this question with a 'Y', the runtime services will ask for the number of units (in decimal). You will then be asked the following questions for each unit.

UNITS (D) ?

Answer with the number of units to be tested (no default). This answer will determine how many times the following questions are asked. A unit is a logical disk (single platter) on an RC25. A maximum of sixteen units will be accepted by the init code.

RC25 IP REGISTER ADDRESS (O) 172150 ?

Answer with the address of the IP Register of one RC25 controller as addressed by the processor with memory management turned off (i.e., An even 16 bit address in the range of 160000 to 177774).

RC25 INTERRUPT VECTOR ADDRESS (O) 154 ?

Answer with the interrupt vector address of the RC25 controller. A vector address in the range of 4 to 774 may be specified.

RC25 BUS REQUEST LEVEL (D) 5 ?

Answer with the interrupt priority used by the RC25. Levels 4 to 7 are accepted.

UNIT NUMBER TO BRING ONLINE (D) 0 ?

Answer with the physical platter number you wish to bring online. The removable platter is an even number and the fixed platter is the sequentially following odd number.

2.5 SOFTWARE QUESTIONS

The supervisor will ask the question "CHANGE SW (L) ?". This question is to be answered with 'Y'. The following message will be printed to the console terminal:

FORMAT IN UNATTENDED REFORMAT MODE (L) YES ?

A 'YES' response (the default) will cause the host to run in UN-ATTENDED reformat mode (see section 7.2 for description).

A 'NO' response will cause the host to run in ATTENDED mode (see section 7.1 for description). In this mode an operator must be present at the

C 0526
 C 0527
 C 0528
 C 0529
 C 0530
 C 0531
 C 0532
 C 0533
 C 0534
 C 0535
 C 0536
 C 0537
 C 0538
 C 0539
 C 0540
 C 0541
 C 0542
 C 0543
 C 0544
 C 0545
 C 0546
 C 0547
 C 0548
 C 0549
 C 0550
 C 0551
 C 0552
 C 0553
 C 0554
 C 0555
 C 0556
 C 0557
 C 0558
 C 0559
 C 0560
 C 0561
 C 0562
 C 0563
 C 0564
 C 0565
 C 0566
 C 0567
 C 0568
 C 0569
 C 0570
 C 0571
 C 0572
 C 0573
 C 0574
 C 0575
 C 0576
 C 0577
 C 0578
 C 0579
 C 0580
 C 0581

console terminal to answer the following DM formatter software questions:

The DM will ask the following questions. The question numbers are those used as the DUP question numbers.

0. Enter date:
Enter the current date in MM-DD-YYYY.
1. Enter unit number to format:
Enter the number of the unit to format.
2. Enter sector size to be used (512/576):
This question is for compatibility with other formatters.
The RC25 formatter will skip this question.
3. Enter mode to be used (slow/normal/fast):
This question is for compatibility with other formatters.
The RC25 formatter will skip this question.
4. Use existing bad block information (y/n):
If the answer is YES, a REFORMAT mode format is done. If the answer is NO, the mode of format is determined by question 5. If the answer is YES, question 5 is skipped.
5. DOWN-LINE LOAD bad block information (y/n):
An answer of NO will cause a RECONSTRUCT mode format to be performed. If an answer of YES is given, a RESTORE mode format will be performed. An answer of NO will skip question 6.

NOTE: It is the responsibility of the host program to obtain the name of the file to be used for the DOWN-LINE LOAD.
6. Continue if bad block information is inaccessible (y/n):
An answer of YES will cause a RECONSTRUCT mode format to be done if the formatter cannot read the FCT on the disk or if a RESTORE mode format was attempted and the bad block data file was unavailable. An answer of NO will cause the format to abort if the bad block information is inaccessible. An answer of NO will terminate the question sequence.

FCT blocks will be requested via the special DUP message code. The first data word in this message will be the FCT block (relative to zero) requested by the formatter. All DUP sequence numbers for FCT block requests will be 1. The response is a 3 word block from the host. The first word is 0 if the block was successfully retrieved and non-zero if it was not. The next 2 words contain the UNIBUS address of a buffer containing the 512 byte block from the FCT file.
7. Enter serial number:
Enter the 64-bit decimal serial number of the disk to be formatted. In the RC25 formatter this number is used only if a RECONSTRUCT mode format is used.

2.6 EXTENDED P-TABLE DIALOGUE

When you answer the hardware questions, you are building entries in a table that describes the devices under test. The simplest way to build this table is to answer all questions for each unit to be tested. If you have a multiplexed device such as a mass storage controller with several drives or a communication device with several lines, this becomes tedious since most of the answers are repetitious.

To illustrate a more efficient method, suppose you are testing a fictional device, the XY11. Suppose this device consists of a control module with eight units (sub-devices) attached to it. These units are described by the octal numbers 0 through 7. There is one hardware parameter that can vary among units called the Q-FACTOR. This Q-FACTOR may be 0 or 1. Below is a simple way to build a table for one xy11 with eight units.

```

# UNITS (D) ? 8<CR>

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

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

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

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

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

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

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

UNIT 8
CSR ADDRESS (O) ? 160000<CR>

```

C 0582
C 0583
C 0584
C 0585
C 0586
C 0587
C 0588
C 0589
C 0590
C 0591
C 0592
C 0593
C 0594
C 0595
C 0596
C 0597
C 0598
C 0599
C 0600
C 0601
C 0602
C 0603
C 0604
C 0605
C 0606
C 0607
C 0608
C 0609
C 0610
C 0611
C 0612
C 0613
C 0614
C 0615
C 0616
C 0617
C 0618
C 0619
C 0620
C 0621
C 0622
C 0623
C 0624
C 0625
C 0626
C 0627
C 0628
C 0629
C 0630
C 0631
C 0632
C 0633
C 0634
C 0635
C 0636
C 0637
C 0638

C 0639 SUB-DEVICE # (0) ? 7<CR>
C 0640 Q-FACTOR (0) 1 ? <CR>

C 0641
C 0642 Notice that the default value for the Q-FACTOR changes when a
C 0643 non-default response is given. Be careful when specifying
C 0644 multiple units!

C 0645
C 0646 As you can see from the above example, the hardware parameters
C 0647 do not vary significantly from unit to unit. The procedure shown
C 0648 is not very efficient.

C 0649
C 0650 The runtime services can take multiple unit specifications however.
C 0651 Let's build the same table using the multiple specification feature.

C 0652 # UNITS (0) ? 8<CR>

C 0653 UNIT 1
C 0654 CSR ADDRESS (0) ? 160000<CR>
C 0655 SUB-DEVICE # (0) ? 0,1<CR>
C 0656 Q-FACTOR (0) 0 ? 1,0<CR>

C 0657
C 0658 UNIT 3
C 0659 CSR ADDRESS (0) ? 160000<CR>
C 0660 SUB-DEVICE # (0) ? 2-5<CR>
C 0661 Q-FACTOR (0) 0 ? 0<CR>

C 0662
C 0663 UNIT 7
C 0664 CSR ADDRESS (0) ? 160000<CR>
C 0665 SUB-DEVICE # (0) ? 6,7<CR>
C 0666 Q-FACTOR (0) 0 ? 1<CR>

C 0667
C 0668 As you can see in the above dialogue, the runtime services will
C 0669 build as many entries as it can with the information given in any
C 0670 one pass through the questions. In the first pass, two entries
C 0671 are built since two sub-devices and Q-FACTORS were specified. The
C 0672 services assume that the CSR address is 160000 for both since it
C 0673 was specified only once. In the second pass, four entries were
C 0674 built. This is because four sub-devices were specified. The
C 0675 "-" construct tells the runtime services to increment the data
C 0676 from the first number to the second. In this case, sub-devices
C 0677 2, 3, 4 and 5 were specified. (If the sub-device were specified
C 0678 by addresses, the increment would be by 2 since addresses must
C 0679 be on an even boundary.) The CSR addresses and Q-FACTORS for
C 0680 the four entries are assumed to be 160000 and 0 respectively
C 0681 since they were only specified once. The last two units are
C 0682 specified in the third pass.

C 0683
C 0684 The whole process could have been accomplished in one pass as shown
C 0685 below.

C 0686 # UNITS (0) ? 8<CR>

C 0687 UNIT 1
C 0688 CSR ADDRESS (0) ? 160000<CR>
C 0689 SUB-DEVICE # (0) ? 0-7<CR>
C 0690 Q-FACTOR (0) 0 ? 0,1,0,....,1,1<CR>

C 0691
C 0692
C 0693
C 0694
C 0695

CZRCH1
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
USER DOCUMENTATION

11-Jul-1983 16:04:50
8-Jul-1983 13:42:19

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH1.B16;4 (9)

SEQ 17
Page 17

: C 0696 As you can see from this example, null replies (commas enclosing
: C 0697 a null field) tell the runtime services to repeat the last reply.
: C 0698
: C 0699 2.7 QUICK START-UP PROCEDURE (XXDP+)
: C 0700
: C 0701 To start-up this program:
: C 0702
: C 0703 1. Boot XXDP+
: C 0704
: C 0705 2. Give the date
: C 0706
: C 0707 3. Type 'R CZRCHA', where name is the name of the bin or bic
: C 0708 file for this program
: C 0709
: C 0710 4. Type "START"
: C 0711
: C 0712 5. Answer the "CHANGE HW" question with "Y"
: C 0713
: C 0714 6. Answer all the hardware questions
: C 0715
: C 0716 7. Answer the "CHANGE SW" question with "N"
: C 0717
: C 0718 When you follow this procedure you will be using only the
: C 0719 defaults for flags and software parameters. These defaults
: C 0720 are described in sections 2.3 and 2.5.
: C 0721

```

: C 0722
: C 0723
: C 0724
: C 0725
: C 0726
: C 0727
: C 0728
: C 0729
: C 0730
: C 0731
: C 0732
: C 0733
: C 0734
: C 0735
: C 0736
: C 0737
: C 0738
: C 0739
: C 0740
: C 0741
: C 0742
: C 0743
: C 0744
: C 0745
: C 0746
: C 0747
: C 0748
: C 0749
: C 0750
: C 0751
: C 0752
: C 0753
: C 0754

```

2.8 PID (PROCESS INDICATOR) WORDS

The process indicator words (PID) are maintained by the remote DM program and indicates to the host the DM programs progress running in the controller.

These PID words are then obtained by the host via the DUP GET_DUST_STATUS command. The newly obtained PID are compared to previous PID words. If the PID has not increased since the last GET_DUST_STATUS then the remote program is to be considered dead and the connection to the controller broken.

However if the PID value has increased since the last GET_DUST_STATUS then the remote program is to be considered still running. The new PID words are saved and command time out waits reinitiated.

This host code will print out to the console terminal the PID values after each GET_DUST_STATUS while the formatter is running in the controller. The printing format is as follows:

```
FORMATTER PROGRESSING: PID HI=xxxxxxx(0) PID LO=xxxxxxx(0)
```

To further demonstrate to the host the progress of the DM formatter the following is performed:

Each time the DM formatter calls in a new overlay from host memory the low PID word is cleared and the high PID word is incremented. The operator can then be observed when new overlays are called into DM memory.

2.9 HOST/DM FORMATTER RUN TIME MESSAGES

All host: messages, fatal errors or operator prompts are printed in UPPER case characters while all DM formatter: messages, fatal errors, completion information or operator prompts are printed in lower case characters.

: C 0755
 : C 0756
 : C 0757
 : C 0758
 : C 0759
 : C 0760
 : C 0761
 : C 0762
 : C 0763
 : C 0764
 : C 0765
 : C 0766
 : C 0767
 : C 0768
 : C 0769
 : C 0770
 : C 0771
 : C 0772
 : C 0773
 : C 0774
 : C 0775
 : C 0776
 : C 0777
 : C 0778
 : C 0779
 : C 0780
 : C 0781
 : C 0782
 : C 0783
 : C 0784
 : C 0785
 : C 0786
 : C 0787
 : C 0788
 : C 0789
 : C 0790
 : C 0791
 : C 0792
 : C 0793
 : C 0794
 : C 0795
 : C 0796
 : C 0797

3.0 ERROR INFORMATION

3.1 TYPES OF ERROR MESSAGES

Due to the fact that this is an utility and not a diagnostic error macro calls will not be used by this host code to report errors. Instead all errors will be printed to the operator via the PRINTF macro using the prefix '\$FTLERR- error message text' indicating this to be a fatal error.

3.2 SPECIFIC ERROR MESSAGES

DM Code Error Messages

All errors returned are fatal. Error messages returned by the RC25 formatter are as follows. The error numbers given are the DUP error message numbers.

- 1 'GET STATUS failure''
- 2 'LESI send error''
- 3 'Unsuccessful LESI command''
- 4 'LESI receive error''
- 5 'UNIBUS I/O error''
- 6 'Formatter initialization error''
- 7 'Nonexistent unit number''
- 8 'DBN/XBN format error (drive FORMAT command failed)''
- 9 'FCT does not have enough good copies of each block''
- 10 'SEEK error''
- 11 'RCT does not have enough good copies of each block''
- 12 'LBN format error (drive FORMAT command failed)''
- 13 'FCT write error''
- 14 'RCT read error''
- 15 'RCT write error''
- 16 'RCT full''
- 17 'FCT read error''
- 18 'FCT nonexistent''
- 19 'FCT downline-load error''
- 20 'Drive init timeout''
- 21 'Illegal response to start-up question''
- 22 'WARNING - possible head addressing problems - run diagnostics''
- 23 'INPUT Error''
- 24 'Media degraded''


```
: C 0798      Host Error Messages
: C 0799
: C 0800      |
: C 0801      | Generic self-detected fatal port/controller errors
: C 0802      |
: C 0803      | $FTLERR- UNRECOGNIZABLE ERROR CODE
: C 0804      | $FTLERR- ENVELOPE/PACKET READ (PARITY OR TIMEOUT)
: C 0805      | $FTLERR- ENVELOPE/PACKET WRITE (PARITY OR TIMEOUT)
: C 0806      | $FTLERR- CONTROLLER ROM AND RAM PARITY
: C 0807      | $FTLERR- CONTROLLER RAM PARITY
: C 0808      | $FTLERR- CONTROLLER ROM PARITY
: C 0809      | $FTLERR- RING READ (PARITY OR TIMEOUT)
: C 0810      | $FTLERR- RING WRITE (PARITY OR TIMEOUT)
: C 0811      | $FTLERR- INTERRUPT MASTER
: C 0812      | $FTLERR- HOST ACCESS TIMEOUT
: C 0813      | $FTLERR- CREDIT LIMIT EXCEEDED
: C 0814      | $FTLERR- BUS MASTER ERROR
: C 0815      | $FTLERR- DIAGNOSTIC CONTROLLER FATAL ERROR
: C 0816      | $FTLERR- INSTRUCTION LOOP TIMEOUT
: C 0817      | $FTLERR- INVALID CONNECTION IDENTIFIER
: C 0818      | $FTLERR- INTERRUPT WRITE
: C 0819      | $FTLERR- MAINTENANCE READ/WRITE INVALID REGION IDENTIFIER
: C 0820      | $FTLERR- MAINTENANCE WRITE LOAD TO NON-LOADABLE CONTROLLER
: C 0821      | $FTLERR- CONTROLLER RAM ERROR (NON-PARITY)
: C 0822      | $FTLERR- INIT SEQUENCE ERROR
: C 0823      | $FTLERR- HIGH LEVEL PROTOCOL INCOMPATIBILITY ERROR
: C 0824      | $FTLERR- PURGE/POLL HARDWARE FAILURE
: C 0825      | $FTLERR- MAPPING REGISTER READ ERROR (PARITY OR TIMEOUT)
```

```
.....
C 0826      |
C 0827      | : RC25 Self-detected fatal port/controller errors
C 0828      | :
C 0829      | $FTLERR- VAX READ/WRITE ERROR ON INTERRUPT
C 0830      | $FTLERR- INCONSISTENCY AT U.BFIL
C 0831      | $FTLERR- INCONSISTENCY AT U.BMTY
C 0832      | $FTLERR- INCONSISTENCY AT U.ALOC
C 0833      | $FTLERR- INCONSISTENCY AT SERVO ENTRY (PIP SET)
C 0834      | $FTLERR- INCONSISTENCY AT SERVO ENTRY (ERR SET)
C 0835      | $FTLERR- INCONSISTENCY AT U.SEND
C 0836      | $FTLERR- INCONSISTENCY AT U.RECV
C 0837      | $FTLERR- INCONSISTENCY AT U.ATTN
C 0838      | $FTLERR- INCONSISTENCY AT U.ONLN
C 0839      | $FTLERR- ILLEGAL D REQUEST (U.QDRQ)
C 0840      | $FTLERR- FENCE-POST ERROR AT PROTAB
C 0841      | $FTLERR- BAD PACKET DEQUEUED AT U.DONE
C 0842      | $FTLERR- UNEXPLAINED D-PROC SUSPENSION (U..TDS)
C 0843      | $FTLERR- DUP PACKET D-Q FAILED (XFC 34/35)
C 0844      | $FTLERR- INCONSISTENCY AT U.HTST
C 0845      | $FTLERR- INCONSISTENCY AT U.SEKO
C 0846      | $FTLERR- INCONSISTENCY AT U.CKSV
C 0847      | $FTLERR- D.OPCD FOUND ILLEGAL OPCODE
C 0848      | $FTLERR- D.CSF FOUND ILLEGAL OPCODE
C 0849      | $FTLERR- UNKNOWN BAD DRIVE STATUS AT D.DSTS
C 0850      | $FTLERR- ILLEGAL XFC EXECUTED BY DM
C 0851      | $FTLERR- D PICKED UP A ZERO SCB.DB
C 0852      | $FTLERR- INCONSISTENCY AT D IDLE LOOP
C 0853      | $FTLERR- DM WORD COUNT ERROR ON HOST DMA/SEND/RECV
C 0854      | $FTLERR- UNKNOWN DISPLAY FAULT CODE AT D.DFLT
C 0855      | $FTLERR- DRIVE NOT FAULTING IN P.OFLN STATE
C 0856      | $FTLERR- U POWER UP DIAGNOSTICS FAILED
C 0857      | $FTLERR- D POWER UP DIAGNOSTICS FAILED
C 0858      | $FTLERR- ADAPTER CARD FAILURE
C 0859      | $FTLERR- EC.TMR TIMED OUT
C 0860      | $FTLERR- U.SEND/U.RECV RING READ INCONSISTENCY
C 0861      | $FTLERR- UNKNOWN WAITRV REASON AT D.RVCT
C 0862      | $FTLERR- D.ARCS DID NOT FIND CLOSEST UNDONE ZONE
C 0863      | $FTLERR- U.SEEK FOUND SEEK TO ILLEGAL TRACK
C 0864      | $FTLERR- U.HTST INIT DIAG DMA WRITE FAILED
C 0865      | $FTLERR- U.HTST INIT DIAG DMA COMPARE FAILED
C 0866      | $FTLERR- U.SYDR FOUND SS.DER SET AND SS.SPN NOT SET
C 0867      | $FTLERR- MASTER DRIVES ACLO ASSERTED
.....
```

- : C 0868
- : C 0869
- : C 0870
- : C 0871
- : C 0872
- : C 0873
- : C 0874
- : C 0875
- : C 0876
- : C 0877
- : C 0878
- : C 0879
- : C 0880
- : C 0881
- : C 0882
- : C 0883
- : C 0884
- : C 0885
- : C 0886
- : C 0887
- : C 0888
- : C 0889
- : C 0890
- : C 0891
- : C 0892
- : C 0893
- : C 0894
- : C 0895
- : C 0896

```

: Host runtime fatal error messages
:
$FTLERR- NON-EXISTENT RC25 REGISTER ADRS xxxxxx
$FTLERR- DUP SERVER ACTIVE AFTER INITIALIZATION
$FTLERR- DUP SERVER INACTIVE AFTER EX_SUP_PROG COMMAND
$FTLERR- RESPONSE STATUS ERROR: actual status error message
$FTLERR- HOST/CONTROLLER OUT OF SEQ
$FTLERR- REMOTE PROG NOT RUNNING
$FTLERR- UNKNOWN RETURN STATUS CODE
$FTLERR- COM AREA INIT ERROR
$FTLERR- PORT/HOST SYNC ERROR
$FTLERR- MESSAGE LENGTH ERROR
$FTLERR- UNKNOWN ENDCODE RECEIVED
$FTLERR- ADAPTOR PURGE ERROR
$FTLERR- UNKNOWN INTERRUPT
$FTLERR- INIT SEQ STEP TIMED OUT
$FTLERR- INIT SEQ COMPARE ERROR
$FTLERR- UNEXPECTED ATTENTION END MESSAGE RECEIVED
$FTLERR- UNEXPECTED COMMAND OPCODE IN END MESSAGE RECEIVED
$FTLERR- UNEXPECTED SERIOUS EXCEPTION END MESSAGE RECEIVED
$FTLERR- INVALID COMMAND END MESSAGE RECEIVED
$FTLERR- UNKNOWN MESSAGE TYPE RECEIVED
$FTLERR- OUTSTANDING COMMAND BUFFER FULL
$FTLERR- OUTSTANDING COMMAND BUFFER OUT OF SYNC ERROR
$FTLERR- UNKNOWN MESSAGE NUMBER RECEIVED
$FTLERR- FILE READ ERROR
$FTLERR- PORT/CONTROLLER TIMEOUT ERROR
$FTLERR- ILLEGAL FCT FILE LENGTH

```

- C 0897
- C 0898
- C 0899
- C 0900
- C 0901
- C 0902
- C 0903
- C 0904
- C 0905
- C 0906
- C 0907
- C 0908
- C 0909
- C 0910
- C 0911
- C 0912
- C 0913
- C 0914
- C 0915
- C 0916
- C 0917
- C 0918
- C 0919
- C 0920
- C 0921
- C 0922
- C 0923
- C 0924
- C 0925
- C 0926
- C 0927
- C 0928
- C 0929
- C 0930
- C 0931
- C 0932
- C 0933
- C 0934

```

: Init code fatal error messages
$FTLERR- NO ADDITIONAL UNITS TO FORMAT - ABORTING
$FTLERR- INIT CODE RE-ENTERED DUE TO PWR FAIL
$FTLERR- ABORTING HOST AND REMOTE PROGRAMS
$FTLERR- ILLEGAL NUMBER OF UNITS SELECTED
$FTLERR- LIMIT OF SIXTEEN UNITS PER FORMATING SESSION
$FTLERR- RC25 CONTROLLER INITIALIZATION ERROR
$FTLERR- PROTOCOL VIOLATION ERROR
$FTLERR- COMMUNICATION AREA INIT ERROR

: Dup return status codes
SUCCESSFUL
INVALID COMMAND
NO REGION AVAILABLE
NO REGION SUITABLE
PROGRAM NOT KNOWN
LOAD FAILURE
STANDALONE

: MSCP return status codes
SUCCESS
INVALID COMMAND
COMMAND ABORTED
UNIT-OFFLINE
UNIT-AVAILABLE
MEDIA FORMAT ERROR
WRITE PROTECTED
COMPARE ERROR
DATA ERROR
HOST BUFFER ACCESS ERROR
CONTROLLER ERROR
DRIVE ERROR
MESSAGE FROM AN INTERNAL DIAGNOSTIC

```


4.0 PERFORMANCE AND PROGRESS REPORTS

The formatter issues the following messages upon normal completion. All but the last are sent as DUP informational messages. The last is sent as a DUP termination message.

1. "format completed"
2. "n revectorized LBNS"
Where n is the number of LBNS revectorized in the user data area.
3. "n primary revectorized LBNS"
Where n is the number of LBNS in message #2 which were primary revectorizers.
4. "n secondary/tertiary revectorized LBNS"
Where n is the number of the LBNS in message #2 which were secondary or tertiary revectorizers.
5. "n bad blocks in the RCT area due to data errors"
Where n is the number of blocks in the total RCT area which were bad due to errors in the data portion of their sectors.
6. "n bad blocks in the RCT due to header or timing errors"
Where n is the number of blocks in the total RCT area which were bad due to errors in the header or timing areas of their sectors.
7. "n bad blocks in the DBN area due to data errors"
Where n is the number of blocks in the DBN area which were bad due to errors in the data area of their sectors.
8. "n bad blocks in the DBN area due to header or timing errors"
Where n is the number of blocks in the DBN area which were bad due to errors in the header or timing area of their sectors.
9. "n bad blocks in the XBN area due to data errors"
Where n is the number of blocks in the XBN area which were bad due to errors in the data area of their sectors.
10. "n bad blocks in the XBN area due to header or timing errors"
Where n is the number of blocks in the XBN area which were bad due to errors in the header or timing area of their sectors.
11. "n bad RBNS"
Where n is the number of blocks in the RBN area which were bad due to errors in the data area of their sectors.
12. "n blocks retried on the check pass"
Where n is the number of blocks which had an error on the first read attempt after formatting.

C 0935
C 0936
C 0937
C 0938
C 0939
C 0940
C 0941
C 0942
C 0943
C 0944
C 0945
C 0946
C 0947
C 0948
C 0949
C 0950
C 0951
C 0952
C 0953
C 0954
C 0955
C 0956
C 0957
C 0958
C 0959
C 0960
C 0961
C 0962
C 0963
C 0964
C 0965
C 0966
C 0967
C 0968
C 0969
C 0970
C 0971
C 0972
C 0973
C 0974
C 0975
C 0976
C 0977
C 0978
C 0979
C 0980
C 0981
C 0982
C 0983
C 0984
C 0985
C 0986
C 0987
C 0988
C 0989
C 0990
C 0991

```

: C 0992      13. 'FCT used successfully' or
: C 0993
: C 0994      14. 'FCT was not used'
: C 0995      Depending on the answers to the start-up questions and the
: C 0996      availability of the bad block information (FCT). This
: C 0997      message has the DUP termination message code.
: C 0998
: C 0999      5.0 DEVICE INFORMATION TABLES
: C 1000
: C 1001      |
: C 1002      | Hardware parameter coding questions
: C 1003      |
: C 1004      | HW_Q1_IP      = IP REGISTER ADDRESS
: C 1005      | HW_Q2_VECTOR = INTERRUPT VECTOR ADDRESS
: C 1006      | HW_Q3_BR      = BUS REQUEST LEVEL
: C 1007      | HW_Q4_UNIT   = UNIT NUMBER TO FORMAT
: C 1008

```

```

:      C 1009      6.0 TEST SUMMARIES
:      C 1010
:      C 1011      6.1 Reformat Mode Procedure
:      C 1012
:      C 1013      Reformat mode is accomplished in four sections:
:      C 1014
:      C 1015      * Table Setup
:      C 1016      * Scanning
:      C 1017      * Replacement
:      C 1018      * Check Pass
:      C 1019
:      C 1020
:      C 1021      6.1.1 Table Setup
:      C 1022
:      C 1023      This section will build an RCT skeleton and perform part of the primary
:      C 1024      revectoring.
:      C 1025
:      C 1026      A. The presence of a good FCT will be determined by the ability
:      C 1027      to read each block in an FCT copy, and by verifying the data
:      C 1028      contained in the FCT Information Block.
:      C 1029
:      C 1030      If none of the four FCT copies is good, the Formatter will
:      C 1031      either:
:      C 1032
:      C 1033      1. Abort the program if DM software question #6 was
:      C 1034      answered with a 'NO'.
:      C 1035
:      C 1036      2. Will do a Reconstruct mode if software question #6
:      C 1037      was answered with a 'YES'.
:      C 1038
:      C 1039      (Software question #6 is: Continue if bad block
:      C 1040      information is inaccessible (Y/N):).
:      C 1041
:      C 1042      B. The RCT tracks will be formatted. Bad blocks in the RCT
:      C 1043      area will be marked with a header code of 11. If a bad
:      C 1044      block occurs in the same position in each copy of the RCT,
:      C 1045      the format will be aborted, as the media is unusable.
:      C 1046
:      C 1047      C. Outer DBN tracks will be formatted here.
:      C 1048
:      C 1049      D. A partial RCT will be built from the FCT: The RCT
:      C 1050      Information Block will be assembled. FCT Bad Block
:      C 1051      Descriptors coded for primary revectoring will be
:      C 1052      transformed into RCT entries. This is done by converting
:      C 1053      the PBN address into an LBN address, and entering it into
:      C 1054      the corresponding RBN descriptor slot for the particular
:      C 1055      track, with a code of 02. Other RBN descriptors will be
:      C 1056      designated as unallocated (code 00) if they have a
:      C 1057      corresponding RBN, or as nulls (code 10) if they do not.
:      C 1058
:      C 1059      Two copies of the RCT will be written to each of the two RCT
:      C 1060      tracks.
:      C 1061
:      C 1062      All LBN's marked for secondary revectoring will be stored in
:      C 1063      an 'R2' Table. Replacement RBN's for these LBN's will be
:      C 1064      resolved after primary revectoring is completed.
:      C 1065

```

C 1066
C 1067
C 1068
C 1069
C 1070
C 1071
C 1072
C 1073
C 1074
C 1075
C 1076
C 1077
C 1078
C 1079
C 1080
C 1081
C 1082
C 1083
C 1084
C 1085
C 1086
C 1087
C 1088
C 1089
C 1090
C 1091
C 1092
C 1093
C 1094
C 1095
C 1096
C 1097
C 1098
C 1099
C 1100
C 1101
C 1102
C 1103
C 1104
C 1105
C 1106
C 1107
C 1108
C 1109
C 1110
C 1111
C 1112
C 1113
C 1114
C 1115
C 1116
C 1117
C 1118
C 1119
C 1120
C 1121
C 1122

6.1.2 Scanning

NOTE:

The following section 6.1.2 generally describes the scanning philosophy used by this formatter. Specific scanning philosophies for each formatting mode is as follows:

REFORMAT mode:

In this mode the FCT file is accessible. The RCT file is built from the FCT and the LBN area is scanned for additional bad blocks. This mode will scan only for data pattern W.

RESTORE mode:

In this mode an external FCT file is brought in from the Host and written to the media's FCT track. After the FCT file has been restored a normal Reformat mode is performed scanning for data pattern W only.

RECONSTRUCT mode:

In this mode the FCT file is inaccessible. The FCT I.D. block is not formatted but rather read/write scanned with all data patterns. The I.D> is then rebuilt indicating zero FCT entries. The Reformat mode overlay is then called in to DM space and will read/write scan the LBN area with all data patterns.

This section writes headers throughout the LBN area. It scans for additional bad blocks, and completes the primary revectoring.

Scanning will start at LBN Track 0 (Top Surface OGB) and progress to LBN Track Max (Bottom Surface OGB) following logical ordering of LBN tracks.

E. An LBN track is written with headers and Data Pattern W, using the Format Track on Index XFC. A failure of this XFC will result in 10 retries. If all retries fail, the format will be aborted.

The header codes connote block status. They will be determined by the FCT Bad Block Descriptors as in the following table:

FCT Descriptor Code	Header Code
Not listed	00 - Good LBN
12 or 14	05 - Primary revector
02 or 04	03 - Secondary revector
01	11 - Secondary revector-
	header problem
Any for RBN	11 - Unusable RBN

NOTE

: C 1123
 : C 1124
 : C 1125
 : C 1126
 : C 1127
 : C 1128
 : C 1129
 : C 1130
 : C 1131
 : C 1132
 : C 1133
 : C 1134
 : C 1135
 : C 1136
 : C 1137
 : C 1138
 : C 1139
 : C 1140
 : C 1141
 : C 1142
 : C 1143
 : C 1144
 : C 1145
 : C 1146
 : C 1147
 : C 1148
 : C 1149
 : C 1150
 : C 1151
 : C 1152
 : C 1153
 : C 1154
 : C 1155
 : C 1156
 : C 1157
 : C 1158
 : C 1159
 : C 1160
 : C 1161
 : C 1162
 : C 1163
 : C 1164
 : C 1165
 : C 1166
 : C 1167
 : C 1168
 : C 1169
 : C 1170
 : C 1171
 : C 1172
 : C 1173
 : C 1174
 : C 1175
 : C 1176
 : C 1177
 : C 1178
 : C 1179

If the RBN is not listed in the FCT, a single block per track may be designated for primary revectoring with an FCT descriptor code of 12 or 14.

If there are no bad blocks in the track, the forced error indicator will be set in the RBN at format XFC time by complimenting its expected EDC field.

- F. All blocks in the track will be read without revectoring enabled, and compared to Pattern W. A write-read-compare sequence will then be performed on each block in the track using Pattern M, which is the compliment of Pattern W.

Error-free blocks will be written with LBN-unique Pattern 1. If the track has a primary revector block, its data field will be written with 128 copies of the track's RBN address, and the RBN will be written with LBN-unique Pattern 1 for the replaced block.

Each block, except used RBN and secondary replacement blocks, is read and its data compared to the expected value. This will be done with revectoring enabled for a primary replacement block, and disabled for the others.

This process is repeated using LBN-unique patterns 2, 3 and 4.

- G. Any error, including correctable errors, will result in 4 retries. Any block failing these will be considered a bad block. If no primary revector block already exists in the track, the first to fail will be primary revectorized by making an RCT entry in its RBN descriptor, reheadering it with a code of 05, and resetting the RBN forced error flag.

Other new bad blocks will be reheadered with secondary revector codes of 03 or 11, depending on the presence of header errors.

If any new bad block(s) is found, steps D and E will be partially repeated to include the additional bad block information, and verify the format write. The write-read-compare sequences with patterns 2, 3 and 4 will not be repeated, however.

- H. Addresses of blocks to be secondary revectorized will be accrued in an 'R2 Table' within the program, until 127 entries are made. At this time, 8 copies of the table will be saved on outer DBN tracks starting at DBN block 0 top surface.

6.1.3 Replacement

This section will complete the RCT and resolve all secondary bad block replacement.

- I. When all tracks have undergone the scanning section, The 'R2 Tables' will be read from the media. The secondary

C 1180 revectorors will be resolved in the RCT, finding RBN Bad Block
C 1181 Descriptors through the Ping-pong algorithm.
C 1182

C 1183 Tracks whose RBN is assigned to be used for secondary
C 1184 revectoring will be written to reset the forced error
C 1185 indicator in the RBN.
C 1186
C 1187

C 1188 6.1.4 Check pass
C 1189

C 1190 This section makes write and read passes of the media as a final check
C 1191 to verify revectoring capabilities.
C 1192

C 1193 J. Starting with the LBN track pair nearest the OGB, the data
C 1194 fields of all blocks on the media will be written as
C 1195 follows:
C 1196

Block Type	Data
Good	Pattern 3
Primary revectored	Pattern 3 of replaced block
Secondary revectored	Pattern 3 of replaced block
Used RBN	Pattern 3 of replaced block

C 1197
C 1198
C 1199
C 1200
C 1201 Starting with the top surface LBN track nearest the OGB all
C 1202 Host Application Blocks will be read to test for
C 1203 revectoring capabilities. Failure of any read or write will
C 1204 result in 10 retries. Failure of all retries will result in
C 1205 a fatal error message sent to the host and the format aborted.
C 1206

C 1207
C 1208
C 1209
C 1210
C 1211
C 1212
C 1213
C 1214 6.2 Restore Mode Procedure
C 1215

C 1216 Restore mode will only be run by DIGITAL Manufacturing personnel. It
C 1217 requires an external copy of the FCT produced when the platter was
C 1218 manufactured and is stored on the System Boot Media. This procedure is
C 1219 as follows:
C 1220

C 1221 A. The FCT tracks will be formatted using the Format Track on
C 1222 Index XFC. A failure of this XFC will result in 10 retries.
C 1223 If all retries fail, the format will be aborted.
C 1224

C 1225 The header code of each block will be 12, indicating good
C 1226 FCT blocks. The data will be pattern W.
C 1227

C 1228 B. Each block will be read and its data verified.
C 1229 Write-read-compare sequences will then be performed on each
C 1230 block, using patterns M, 1, 2, 3 and 4.
C 1231

C 1232 Any errors will result in 10 retries. Any block failing a
C 1233 retry will be considered a bad block.
C 1234

C 1235 C. The tracks will be reformatted, if necessary, to change bad
C 1236 block header codes to 11.

C 1237
 C 1238
 C 1239
 C 1240
 C 1241
 C 1242
 C 1243
 C 1244
 C 1245
 C 1246
 C 1247
 C 1248
 C 1249
 C 1250
 C 1251
 C 1252
 C 1253
 C 1254
 C 1255
 C 1256
 C 1257
 C 1258
 C 1259
 C 1260
 C 1261
 C 1262
 C 1263
 C 1264
 C 1265
 C 1266
 C 1267
 C 1268
 C 1269
 C 1270
 C 1271
 C 1272
 C 1273
 C 1274
 C 1275
 C 1276
 C 1277
 C 1278
 C 1279
 C 1280
 C 1281
 C 1282
 C 1283
 C 1284
 C 1285
 C 1286
 C 1287
 C 1288
 C 1289
 C 1290
 C 1291
 C 1292
 C 1293

- D. The operator will be queried for the original FCT file name. The FCT file will then be inputted from the boot device and sent to the DU when requested.
- E. Two copies of the FCT will be written to each FCT track. These will include bad blocks found in step B.
- F. This mode will be aborted if any FCT sector fails in all 4 copies. The RECONSTRUCT mode will then be called in if the option to continue if the FCT file is inaccessible else the formatter will be aborted.
- G. The program will automatically enter the Format Mode, and proceed as in 4.1 above.

6.3 Reconstruct Mode Procedure

This mode will be performed if:

1. The operator has explicitly requested this as the formatting mode by answering 'NO' to the following software questions:
 - . 'Use existing Bad Block information?'
 - . 'Down-line load Bad Block information?'
2. The Operator has responded with a 'YES' to the software question:
 - . 'Continue if Bad Block information is inaccessible?' and the Bad Block file is discovered inaccessible during a Reformat or Restore Formatting Mode.

Reconstruct formatting procedure is as follows:

1. Sector 0 of each FCT copy will be read/write scanned.

The Formatting process will be aborted if all four copies of Sector 0 are bad.
2. The Volume Information Block (Sector 0) will be reconstructed using the following information:
 - . Media Mode = 512 Byte Format
 - . Formatting Instance Number = 0
 - . Volume Serial Number = Operator Supplied S.N.
 - . Time of First Formatting = Operator Supplied Date (VAX/VMS Format)
 - . Time of Most Recent Formatting = Operator Supplied Date (VAX/VMS Format)
 - . Number of Used 512 Table Entries = 0

: C 1294
: C 1295
: C 1296
: C 1297
: C 1298
: C 1299
: C 1300
: C 1301
: C 1302
: C 1303
: C 1304
: C 1305
: C 1306
: C 1307
: C 1308
: C 1309
: C 1310
: C 1311
: C 1312
: C 1313

. Number of Used 576 Table Entries = 0

. Remaining Words Zeroed

3. The Volume Information Block is then written to each FCT copy and read back for data integrity.

Read Failures at all four copies will cause the formatting process to abort.

NOTE

Remaining FCT Sectors 1 thru 15 are not altered.

4. The formatting process will then perform the Reformat Mode Procedure.

8.0 APPENDIX

APPENDIX A

GLOSSARY

C 1340		
C 1341		
C 1342		
C 1343		
C 1344		
C 1345	DBN	Diagnostic Block Number. A number of per surface are reserved for diagnostic use. These tracks are called the DBN area.
C 1346		
C 1347		
C 1348	DM	Diagnostic Machine. An interpreter built into the RC25 firmware which is used to execute programs downline loaded into the RC25's buffer memory.
C 1349		
C 1350		
C 1351		
C 1352	EDC	Error Detecting Code. A means of verifying correct controller operation.
C 1353		
C 1354		
C 1355	FCT	Factory Control Table. A 16 sector table containing volume ID information and a map of bad blocks on the media. The FCT is written as part of the initial media formatting.
C 1356		
C 1357		
C 1358		
C 1359	FRU	Field Replaceable Unit. An item which may be exchanged in the field.
C 1360		
C 1361		
C 1362	IGB	Inner Guard Bands. The guard bands closer to the spindle.
C 1363		
C 1364	IP	Initialization and Polling Register. This register is used for port control.
C 1365		
C 1366		
C 1367	LBN	Logical Block Number. Host visible blocks including the Host Application Area and the RCT tracks.
C 1368		
C 1369		
C 1370	MSCP	Mass Storage Command Protocol. The method of communication used between the host and the RC25 over the bus.
C 1371		
C 1372		
C 1373	OGB	Outer Guard Bands. The guard bands further from the spindle.
C 1374		
C 1375	PBN	Physical Block Number. The absolute address of a block on RC25 media.
C 1376		
C 1377		
C 1378	RBN	Replacement Block Number. RC25 has one RBN per track for bad block replacement.
C 1379		
C 1380		
C 1381	RCT	Replacement and Caching Table. A 16 sector table containing media ID information and a map of the current bad block locations.
C 1382		
C 1383		
C 1384		
C 1385	RZ01	RC25 product name.
C 1386		
C 1387	SA	Status, Address and Purge Register for port control.
C 1388		
C 1389	XFC	Extended Function Call. Instructions which interface Diagnostic Machine code to the RC25 microcode.
C 1390		

APPENDIX B

BAD BLOCK REPLACEMENT

Bad block replacement is an addressing technique for replacing a bad block with a good one. The block which replaces it is called a replacement block, or RBN.

Each RC25 track has 32 blocks. In the host-addressable area, the first thirty-one are for normal data storage, and are called Logical Blocks, or LBNs. The last block on each of these tracks is an RBN.

If a single bad block exists on a track, it can be replaced by the RBN of that track. This is called Primary replacement.

If other bad blocks are found in the same track, they must be replaced by unused RBNs of other tracks. This is called Secondary replacement.

Two types of tables record bad block information for the media. One is the Factory Control Table (FCT), which is written on the media at the factory. It is a map of the bad blocks caused by the manufacturing process. The other is the Replacement and Caching Table (RCT), which includes both the FCT information, and bad block information uncovered by the latest formatting of the media. There are four copies of each table per unit.

APPENDIX C

FCT

: C 1419
 : C 1420
 : C 1421
 : C 1422
 : C 1423
 : C 1424
 : C 1425 The basic unit of the FCT is the Bad Block Descriptor. This descriptor
 : C 1426 gives the Physical Block Address of the bad block, and a four bit code
 : C 1427 indicating why it was retired.
 : C 1428
 : C 1429 The Physical Block Address, or PBN, is the absolute address of a block
 : C 1430 anywhere on the media, without the distinction of host-accessibility.
 : C 1431
 : C 1432 There are a variable number of Bad Block Descriptors in the FCT. These
 : C 1433 descriptors are sorted in descending track order, and in ascending PBN
 : C 1434 order within tracks.

:
 C 1435
 C 1436
 C 1437
 C 1438
 C 1439
 C 1440
 C 1441
 C 1442
 C 1443
 C 1444
 C 1445
 C 1446
 C 1447
 C 1448
 C 1449
 C 1450
 C 1451
 C 1452
 C 1453
 C 1454
 C 1455
 C 1456
 C 1457
 C 1458
 C 1459
 C 1460
 C 1461
 C 1462
 C 1463
 C 1464
 C 1465
 C 1466
 C 1467
 C 1468
 C 1469
 C 1470
 C 1471
 C 1472
 C 1473
 C 1474
 C 1475
 C 1476
 C 1477
 C 1478
 C 1479
 C 1480
 C 1481
 C 1482
 C 1483
 C 1484
 C 1485
 C 1486
 C 1487
 C 1488
 C 1489
 C 1490
 C 1491

APPENDIX D

RCT

The basic unit of the RCT is the double-word RBN Descriptor. There are 1584 of these, one for each RBN in the host-accessible LBN area of the media.

The position of each RBN Descriptor in the RCT is fixed. The first RBN Descriptor is associated with the RBN block in the first host-visible track. The second descriptor with the RBN of the second host-visible track, etc..

In the format process, the FCT is given. Four copies of it are found on the media; two on each FCT track. The RCT is rebuilt from the FCT entries and additional bad block information is gathered in a scan pass.

Primary replacement may be possible for one block per track if identified as bad by either of these sources. The track's RBN must itself be a good, unused block.

Primary replacement is accomplished by:

- * Making the bad block's header code 05.
- * Writing the bad block's data field as 128 copies of its track's RBN address.
- * Writing the address of the bad block to the associated RBN Descriptor in the RCT (with a code of 02).

Secondary replacement is needed for bad blocks which do not have a good, unused RBN block.

Secondary replacement is accomplished similarly to Primary Replacement, but first an unused RBN of another track must be found.

- * Find an unused RBN Descriptor in the RCT, using the Ping-pong algorithm.
- * Make the bad block's header code 03 or 11, depending on error type.
- * Write the bad block's data field with 128 copies of the target RBN address.
- * Write the address of the bad block to the associated RBN Descriptor in the RCT (with a code of 03).

After replacement an attempt by the host to access a replaced block will result in access to the block with which it had been replaced.

Good blocks are headered with codes of 00.

To help detect addressing problems, unused RBNs will have a forced error

CZRCH1 CZRCH RC25 DISK FORMATTER
REV A PATCH 00 USER DOCUMENTATION

11-Jul-1983 16:04:50
8-Jul-1983 13:42:19

VAX-11 Bliss-16 V3-555
SPIDER\$USERS:[NEALE.AZTEC]CZRCH1.B16;4 (22)

: C 1492 indication set by complimenting their EDC fields.

APPENDIX E
DATA PATTERNS

Six patterns will be used during the scan pass.

Pattern W is a pseudo-random pattern, with the LBN address as the first word. Pattern M is its compliment. Its use will assure that all data bits can toggle.

Patterns 1 through 4 are made up of single word elements. These elements are as follows:

- Element A = 071311. This causes peak shift in data encoding.
- Element B = 106466. This causes single transitions spaced by 1, 2, 3 and 4 zeroes.
- Element C = 043146. This causes both the highest and lowest frequency of transitions in the encoding logic.
- Element D = 134631. This also causes both the highest and lowest frequency of transitions in the encoding logic.
- Element L = 17NNNN. Where NNNN is the LBN address to which the pattern is written.

These elements are assembled into patterns 1 through 4 as follows:

- Pattern 1 = LBBAAABBBBAAAAABBBBBBAAAAAABBBBBBBB etc.
- Pattern 2 = BLABBBAAAABBBBBBAAAAAABBBBBBBBAAAAAAA etc.
- Pattern 3 = LDDCCDDDDCCCCDDDDDDCCCCCCCCDDDDDDDD etc.
- Pattern 4 = DLCDDDDCCCCDDDDDDCCCCCCCCDDDDDDDDCCCCCCCC etc.

C 1493
C 1494
C 1495
C 1496
C 1497
C 1498
C 1499
C 1500
C 1501
C 1502
C 1503
C 1504
C 1505
C 1506
C 1507
C 1508
C 1509
C 1510
C 1511
C 1512
C 1513
C 1514
C 1515
C 1516
C 1517
C 1518
C 1519
C 1520
C 1521
C 1522
C 1523
C 1524
C 1525
C 1526
C 1527
C 1528
C 1529
C 1530
C 1531
C 1532
C 1533
C 1534
C 1535
C 1536

```

: C 1537 Embedding the LBN address within the patterns gives them LBN-uniqueness.
: C 1538 This feature is taken advantage of in validating revectoring.
: C 1539
: C 1540 Three steps are taken to insure the data will not interfere with
: C 1541 Secondary Revectoring by matching the "Compare 128" algorithm. These
: C 1542 are:
: C 1543
: C 1544 * The high four bits in every word contain invalid header codes.
: C 1545
: C 1546 * Before secondary revectoring is attempted, blocks not involved
: C 1547 with revectoring will be written with data fields which
: C 1548 contain no 32-bit repetitive patterns.
: C 1549
: C 1550 * When the format is complete, no unrevectorized blocks will
: C 1551 contain data which has 32-bit repetitive patterns.
: C 1552

```

APPENDIX F

Ping-Pong algorithm

The search begins at the primary replacement block descriptor. If the descriptor is not empty then a ping pong search of the sector containing the primary replacement block descriptor ensues. If an empty descriptor is not encountered then a linear scan of the remaining RCT blocks and descriptors within blocks, (with wrap-around at the end of the RCT) ensues until one of two things occur:

1. An unallocated replacement block descriptor is encountered in an overflow location - a secondary.
2. The entire RCT is searched without success - a failure.

The search operates at two levels:

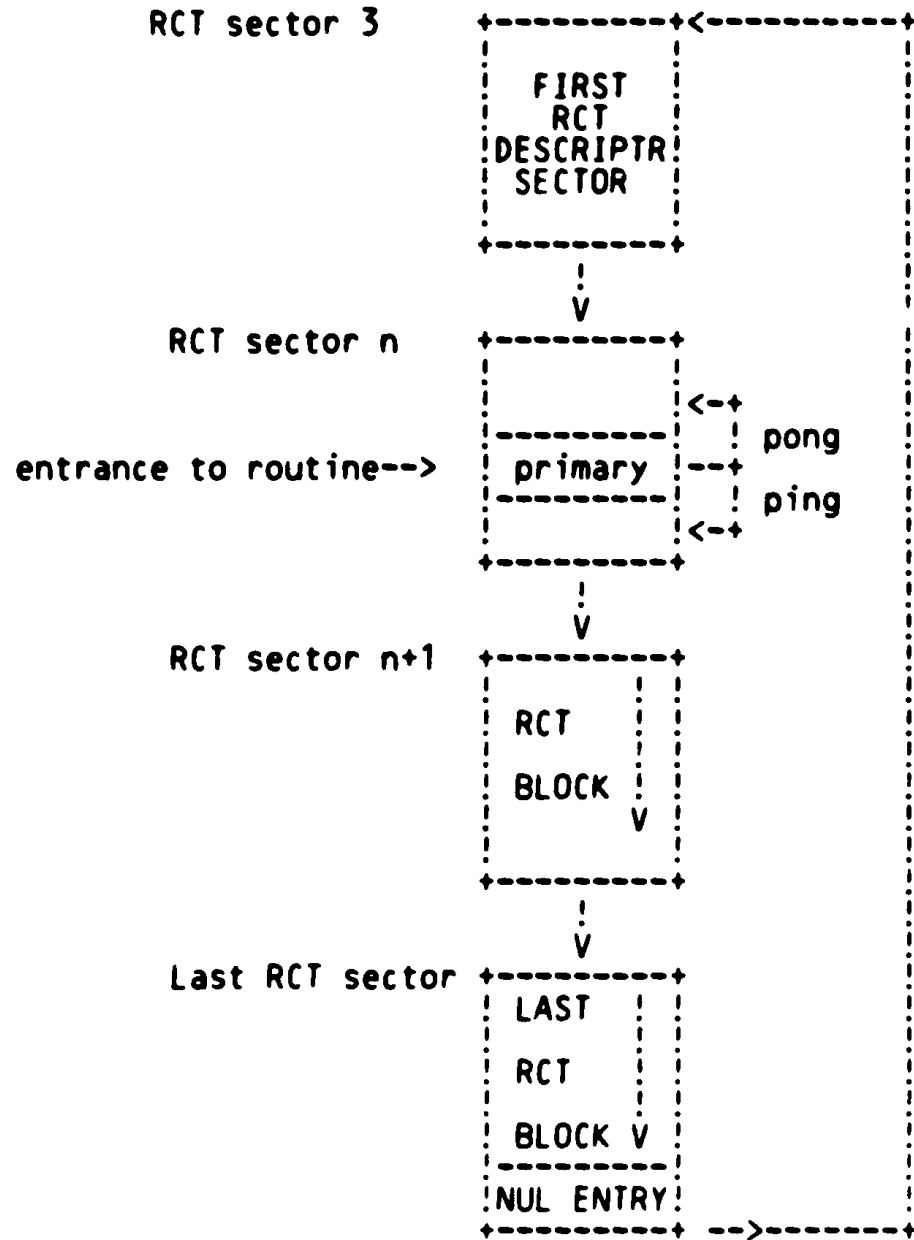
1. Within the primary descriptor RCT sector, outward from the primary descriptor searched (starting with the next highest RBN descriptor). Note that this degenerates to a linear search once the first or last descriptor is encountered.
2. Then a linear search, starting with the next highest RCT sector address, once the initial sector has been completely searched. Each new sector is searched in a linear fashion starting at the lowest RBN descriptor and scanning until the highest RBN descriptor in the sector is encountered.

If at any time during the linear search a null (not an empty) entry is encountered, the search resumes at the first entry in the third RCT sector (the first with descriptors). The search is terminated when it is certain that all the RCT entries have been searched.

C 1553
C 1554
C 1555
C 1556
C 1557
C 1558
C 1559
C 1560
C 1561
C 1562
C 1563
C 1564
C 1565
C 1566
C 1567
C 1568
C 1569
C 1570
C 1571
C 1572
C 1573
C 1574
C 1575
C 1576
C 1577
C 1578
C 1579
C 1580
C 1581
C 1582
C 1583
C 1584
C 1585
C 1586
C 1587

:
C 1588
C 1589
C 1590
C 1591
C 1592
C 1593
C 1594
C 1595
C 1596
C 1597
C 1598
C 1599
C 1600
C 1601
C 1602
C 1603
C 1604
C 1605
C 1606
C 1607
C 1608
C 1609
C 1610
C 1611
C 1612
C 1613
C 1614
C 1615
C 1616
C 1617
C 1618
C 1619
C 1620
C 1621
C 1622
C 1623
C 1624
C 1625
C 1626
C 1627
C 1628
C 1629
C 1630
C 1631
C 1632
:

ALGORITHM FIGURE -



APPENDIX G

CREATING CZRCHA DM CODE

G.1 FILES REQUIRED

A. RTEM (RT-11 emulator) including:

- a. DMACRO.SAV The DM macro assembler
- b. DMACRO.MAC The DM macro library
- c. DTX.SAV The XXDP+ UPD2 emulator including:
- d. DTX DRIVERS including:
 - 1. IZDDAO.DTX
 - 2. IZDKAO.DTX
 - 3. IZDLAO.DTX
 - 4. IZDMAO.DTX
 - 5. IZDTAO.DTX
 - 6. IZDXAO.DTX
 - 7. IZDYAO.DTX
 - 8. IZMMAO.DTX
 - 9. IZMSAO.DTX
 - 10. IZMTAO.DTX

e. STARTM.COM RTEM start up com file. See below assembling DM source files

B. DMCONV.EXE

Converts DMACRO output .SAV files into a Bliss ascii array. This ascii array is then built into a separate Bliss module and compiled. The object code from this module is then linked with other Bliss object modules resulting in a APT compatible, contiguous object file including both host and DM code.

G.1 BUILDING DM SOURCE CODE

DM source code can be built using any of the text editors available under any operating system. Guide lines for its creation can be found in the AZTEC.DOC manual.

G.2 ASSEMBLING DM SOURCE CODE

The DM macro assembler (DMACRO.SAV) is used to assemble DM code modules and is written to be run under RT-11 (or RTEM emulator). The following example demonstrated the RTEM startup com file used to create the released DM formatter .SAV file. This start-up command file includes: allocating more virtual storage, bringing the DM source code into the RTEM system, assembling the DM source code, linking and moving the output .SAV file back to the user directory. The following example is performed on VMS operating system.

```

$ <VMS>
$ set def [neale.rtem]           !Set default directory to the RTEM sub-directory
$ mcr rtem                       !Run the RT-11 emulator
. RTE>/vs                         !Type /vs to prompt RTE>

```

RTEM-11 (VAX/VMS) V01.00

RT-11FB (S) V04.00L

C 1633
C 1634
C 1635
C 1636
C 1637
C 1638
C 1639
C 1640
C 1641
C 1642
C 1643
C 1644
C 1645
C 1646
C 1647
C 1648
C 1649
C 1650
C 1651
C 1652
C 1653
C 1654
C 1655
C 1656
C 1657
C 1658
C 1659
C 1660
C 1661
C 1662
C 1663
C 1664
C 1665
C 1666
C 1667
C 1668
C 1669
C 1670
C 1671
C 1672
C 1673
C 1674
C 1675
C 1676
C 1677
C 1678
C 1679
C 1680
C 1681
C 1682
C 1683
C 1684
C 1685
C 1686
C 1687
C 1688
C 1689

```

C 1690
C 1691
C 1692
C 1693
C 1694
C 1695
C 1696
C 1697
C 1698
C 1699
C 1700
C 1701
C 1702
C 1703
C 1704
C 1705
C 1706
C 1707
C 1708
C 1709
C 1710
C 1711
C 1712
C 1713
C 1714
C 1715
C 1716
C 1717
C 1718
C 1719
C 1720
C 1721
C 1722
C 1723
C 1724
C 1725
C 1726
C 1727
C 1728
C 1729
C 1730
C 1731
C 1732
C 1733
C 1734
C 1735
C 1736
C 1737
C 1738
C 1739
C 1740
C 1741
C 1742
C 1743
C 1744
C 1745
C 1746

```

The following is performed by the startm.com start-up command file. Before running the command file the virtual storage disk VS1: must have already been generated using the JOAT utility.

```

.SET TT:NOCRLF,SCOPE,WIDTH 80      !Set term characteristics
.R JOAT                            !Enter JOAT (Jack of all trades) utility
*MORE/V:VS1                        !Allocate virtual device VS1: to RTEM system
*^C                                 !Exit JOAT

.DEL VS1:*. *                      !Make room in VS1: for this assembly
.R FIP                              !Enter File interchange program
*VS1:=[NEALE.AZTEC]AZFMTR.MAC/F    !Bring in DM source code from user directory
*^C                                 !Exit FIP

.R DMACRO                          !Run the DM macro assembler
*VS1:AZFMTR,AZFMTR=DMACRO,VS1:AZFMTR !Assemble DM source and save in VS1:
*^C                                 !Exit the DM assembler

.R LINK                            !Run the RTEM linker
*VS1:AZFMTR,AZFMTR=VS1:AZFMTR    !Link the .obj file and save in VS1:
*^C                                 !Exit the linker

.R FIP                             !Run FIP
*[NEALE.AZTEC]/F=VS1:AZFMTR.LST   !Output .LST file to user directory
*[NEALE.AZTEC]/F/I=VS1:AZFMTR.SAV !Output .SAV file to user directory
*^C                                 !Exit FIP

.BOO SY                            !Return to VMS
$ <VMS>

```

G.3 BUILDING THE BLISS ASCII ARRAY

The next step is to run the DMCONV program to build the Bliss ascii array from the DMACRO output .SAV file. To do this follow this example:

```

$ run DMCONV.EXE
INPUT FILE NAME (I.E. XXXXXX.YYY) AZFMTR.SAV
OUTPUT FILE NAME (I.E. XXXXXX.YYY) AZFMTR.VEC
BLOCK VECTOR NAME (I.E. XXXXXX.YYY) AZFMTR

```

Note:

In this example it is assumed that DMCONV.EXE and the .SAV file are both in the default directory.

G.4 BUILDING THE DM BLISS MODULE

The output from the DMCONV program (AZFMTR.VEC) is then included into a Bliss module (for the purpose of CZRCHA this module is AZKEL6.B16) and compiled. To compile this module takes a VERY long time so it is suggested that it is submitted to the batch queue after hours.

The entire procedure to compile and link all Bliss source modules (including the DM bliss module) is demonstrated by the following command procedure (azcom.com):

```

: C 1747 $ @azcom !Execute command procedre
: C 1748 $ set verify
: C 1749 $ bliss/pdp11/list/library CZRCH0.R16
: C 1750 $ bliss/pdp11/listing=CZRCHA.DOC CZRCH1.B16
: C 1751 $ bliss/pdp11/list CZRCH2.B16
: C 1752 $ bliss/pdp11/list CZRCH3.B16
: C 1753 $ bliss/pdp11/list CZRCH4.B16
: C 1754 $ bliss/pdp11/list CZRCH5.B16
: C 1755 $ set process/name='killer
: C 1756 $ bliss/pdp11/list CZRCH6.B16
: C 1757 $ set process/name='Fisher man
: C 1758 $ bliss/pdp11/list CZRCH7.B16
: C 1759 $ MCR TKB
: C 1760 CZRCHA/NOHD/NOHM,CZRCHA/CR/-SP=CZRCH2,CZRCH3,CZRCH4,CZRCH5,CZRCH6,CZRCH7,neislb/lb
: C 1761 /
: C 1762 PAR=DUMMY:2000:176000
: C 1763 STACK=0
: C 1764 //
: C 1765 $ !*****
: C 1766 $ !*
: C 1767 $ !* Now run the TKBBIN program to create a .bin file *
: C 1768 $ !*
: C 1769 $ !*****
: C 1770 $ set noverify
: C 1771 $

```

The TKBBIN utility is then run on the output from the TKB linker to produce an XXDP+ executable .BIN file. The following is an example of the procedure:

```

$ RUN TKBBIN
Input filename ? CZRCHA
:load Addr: 2000
:xfer Addr: 1
:Nr char : 74600
End of Job
$

```

G.5 GETTING .BIN FILES ONTO XXDP+ MEDIA

The RT-11 emulator is used to move XXDP+ .BIN files from a VMS user directory onto XXDP+ media. The following is an example of this procedure.

```

$ set def [neale.rtem] !Set default directory to RTEM sub-directory
$ mcr rtem !Run the RT-11 emulator
.RTE>/vs !Type /vs to prompt
:
: The startm.com command file will automatically
: begin to execute when entering RTEM. Control
: c the command procedure at this time and do the
: following:
:
:R JOAT !Run JOAT utility
:DLO:/A !Allocate XXDP+ device to pip .BIN file to
:*^C !Exit JOAT
:
:R FIP !Run FIP utility
*VS1:=[NEALE.AZTEC]CZRCHA.BIN/F/I !Bring .BIN file from user directory into VS1:

```

CZRCH1
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
USER DOCUMENTATION

11-Jul-1983 16:04:50
8-Jul-1983 13:42:19

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH1.B16;4 (27)

```

:      C 1804          *^C          !Exit FIP
:      C 1805
:      C 1806          .R DTX       !Run the XXDP+ UPD2 emulator
:      C 1807          *DEL DLO:CZRCHA.BIN !Delete from XXDP+ media old .BIN file
:      C 1808          *PIP DLO:CZRCHA.BIN=VS1:CZRCHA.BIN !PIP new .BIN file to XXDP+ media
:      C 1809          *LOAD DLO:CZRCHA.BIN !Load the file to verify the chack-sum
:      C 1810          XFR: 000001 CORE:002000,76600
:      C 1811          *^C          !Exit DTX
:      C 1812
:      C 1813          .BOO SY      !Boot back to VMS
:      C 1814          $ <VMS>     !Back at VMS again

```

: C 1815
: C 1816
: C 1817
: C 1818
: C 1819
: C 1820
: C 1821
: C 1822
: C 1823
: C 1824
: C 1825
: C 1826
: C 1827
: C 1828
: C 1829
: C 1830
: C 1831
: C 1832
: C 1833
: C 1834
: C 1835
: 1836)%
: 1837 ELUDOM

APPENDIX H

CREATING CZRCHA HOST CODE

H.1 FILES REQUIRED

- A. Text editor
- B. Bliss16 compiler
- C. TKB linker
- D. TKBBIN Utility This converts Bliss16 .EXE files into XXP+ .BIN files.

H.1 SOURCE FILE CREATING

Using any text editor, follow PDP-11 diagnostic design guidelines demonstrated in SUPPRGC.DOC and CHQUSD.SEQ.

H.2 COMPILING BLISS SOURCE MODULES

Follow steps starting at G.4 to compile Bliss source modules and to get the .BIN file onto XXP+ media.

.TITLE CZRCH1 CZRCH RC25 DISK FORMATTER
.IDENT /REV A /

PSECT SUMMARY

Psect Name	Words	Attributes
------------	-------	------------

COMMAND QUALIFIERS

BLISS /PDP11/LIST=CZRCHA.DOC CZRCH1.B16

: Size: 0 code + 0 data words
 : Run Time: 00:08.6
 : Elapsed Time: 00:15.1
 : Memory Used: 6 pages
 : Compilation Complete

```

0001 %TITLE 'RC25 FORMATTER LIBRARY MODULE'
0002 %sbttl 'MACRO DEFINITIONS'
0003
0004 macro
0005     :
0006     : This macro will transfer the operator inputed date from it storage
0007     : place into the send buffer where it will then be sent to the DM.
0008     :
M 0009     $FILLDATE =
M 0010     begin
M 0011     map
M 0012     snd_buf : vector [74, byte];
M 0013
M 0014     incru i from 0 to 11 do SND_BUF [.i] = .DATETXT [.i];
M 0015
M 0016     end%;
M 0017
M 0018     WRT_RC25 (0, IMAGE) =
M 0019
M 0020     begin
M 0021     local
M 0022     RC$M_REG;
M 0023     RC$M_REG = IMAGE;
M 0024     (.RC25_ADDR + %upval*0) = .RC$M_REG;
M 0025     end%;
M 0026
M 0027     :
M 0028     : Dup Protocol Macros Declarations
M 0029     :
M 0030     REC_BASE =
M 0031     HDR_SIZ, 0, 0, 16, 0%;
M 0032     SND_BASE =
M 0033     HDR_SIZ + REC_ALLOCATE , 0, 0, 16, 0%.
M 0034     :
M 0035     : Macro to clear out the DUP send data text buffer
M 0036     : before requesting input form operator. This by
M 0037     : default puts a null byte at the end of the ascii
M 0038     : input.
M 0039     :
M 0040     $CLRSBUF =
M 0041     incru i from 0 to SNDB_SIZE - 1 do SND_BUF [.i] = ZEROS%;
M 0042     :
M 0043     : General purpose word reference field select
M 0044     :
M 0045     WORD_REF =
M 0046     0, 16, 0%;
M 0047

```

RC25 FORMATTER LIBRARY MODULE
SUPERVISOR DEFINED LITERALS

11-Jul-1983 16:04:41
6-Jul-1983 11:33:38

VAX-11 Bliss-16 V3-555
SPIDERSUSERS:[NEALE.AZTEC]CZRCHO.R16;2 (2)

0048 %sbttl 'SUPERVISOR DEFINED LITERALS'

0049

0050 literal

0051

0052 :+ BIT DIFINITIONS

0053 :-

0054 BIT15 = %o'100000',

0055 BIT14 = %o'40000',

0056 BIT13 = %o'20000',

0057 BIT12 = %o'10000',

0058 BIT11 = %o'4000',

0059 BIT10 = %o'2000',

0060 BIT09 = %o'1000',

0061 BIT08 = %o'400',

0062 BIT07 = %o'200',

0063 BIT06 = %o'100',

0064 BIT05 = %o'40',

0065 BIT04 = %o'20',

0066 BIT03 = %o'10',

0067 BIT02 = %o'4',

0068 BIT01 = %o'2',

0069 BIT00 = %o'1',

0070

0071 BIT9 = BIT09,

0072 BIT8 = BIT08,

0073 BIT7 = BIT07,

0074 BIT6 = BIT06,

0075 BIT5 = BIT05,

0076 BIT4 = BIT04,

0077 BIT3 = BIT03,

0078 BIT2 = BIT02,

0079 BIT1 = BIT01,

0080 BIT0 = BIT00,

0081

0082 :+ EVENT FLAG DEFINITIONS

0083 :- EF32:EF17 RESERVED FOR SUPERVISOR TO PROGRAM COMMUNICATION

0084

0085 EF_START = 32,

0086 EF_RESTART = 31,

0087 EF_CONTINUE = 30,

0088 EF_NEW = 29,

0089 EF_PWR = 28,

! START COMMAND WAS ISSUED
! RESTART COMMAND WAS ISSUED
! CONTINUE COMMAND WAS ISSUED
! A NEW PASS HAS BEEN STARTED
! A POWER-FAIL/POWER-UP OCCURRED

RC25 FORMATTER LIBRARY MODULE
SUPERVISOR DEFINED LITERALS

11-Jul-1983 16:04:41
6-Jul-1983 11:33:38

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCHO.R16;2 (3) SEQ 49
Page 3

```

0090  !+
0091  !: PRIORITY LEVEL DEFINITIONS
0092  !-
0093      PRI07 = %o'340'.
0094      PRI06 = %o'300'.
0095      PRI05 = %o'240'.
0096      PRI04 = %o'200'.
0097      PRI03 = %o'140'.
0098      PRI02 = %o'100'.
0099      PRI01 = %o'40'.
0100      PRI00 = %o'0'.
0101  !+
0102  !: OPERATOR FLAG BITS
0103  !-
0104      EVL = %o'4'.
0105      LOT = %o'10'.
0106      ADR = %o'20'.
0107      IDU = %o'40'.
0108      ISR = %o'100'.
0109      UAM = %o'200'.
0110      BOE = %o'400'.
0111      PNT = %o'1000'.
0112      PRI = %o'2000'.
0113      IXE = %o'4000'.
0114      IBE = %o'10000'.
0115      IER = %o'20000'.
0116      LOE = %o'40000'.
0117      HOE = %o'100000'.

```

```
0118 %sbttl 'FORMATTER DEFINED LITERALS'
0119
0120 : Down line load external FCT copy literals
0121 :
0122 :   FCT_SEC_0 = 0,           !FCT sector zero request
0123 :   FCT_SEC_15 = 15,        !FCT sector fifteen request
0124 :
0125 : Global flag word bit definitions
0126 :
0127 :   NON_EXIST_REG = BIT0,    !Non-existent register flag
0128 :
0129 : Message selection literals
0130 :
0131 :   MSG0 = 0,                !Select message 0
0132 :   MSG1 = 1,                !Select message 1
0133 :   MSG2 = 2,                !Select message 2
0134 :   MSG3 = 3,                !Select message 3
0135 :   MSG4 = 4,                !Select message 4
0136 :   MSG5 = 5,                !Select message 5
0137 :   MSG6 = 6,                !Select message 6
0138 :   MSG7 = 7,                !Select message 7
0139 :   MSG8 = 8,                !Select message 8
0140 :   MSG9 = 9,                !Select message 9
0141 :   MSG10 = 10,              !Select message 10
0142 :   MSG11 = 11,              !Select message 11
0143 :   MSG12 = 12,              !Select message 12
0144 :   MSG13 = 13,              !Select message 13
0145 :   MSG14 = 14,              !Select message 14
0146 :   MSG15 = 15,              !Select message 15
0147 :   MSG16 = 16,              !Select message 16
0148 :   MSG17 = 17,              !Select message 17
0149 :   MSG18 = 18,              !Select message 18
0150 :   MSG19 = 19,              !Select message 19
0151 :   MSG20 = 20,              !Select message 20
0152 :   MSG21 = 21,              !Select message 21
0153 :   MSG22 = 22,              !Select message 22
0154 :   MSG23 = 23,              !Select message 23
0155 :   MSG24 = 24,              !Select message 24
0156 :   MSG25 = 25,              !Select message 25
0157 :   MSG26 = 26,              !Select message 26
0158 :   MSG27 = 27,              !Select message 27
0159 :   MSG28 = 28,              !Select message 28
0160 :   MSG29 = 29,              !Select message 29
0161 :   MSG30 = 30,              !Select message 30
0162 :
0163 : Miscellaneous literals
0164 :
0165 :   DUP = 2,                  !DUP connection ID constant
0166 :   MSCP = 0,                 !MSCP connection ID constant
0167 :   FAILURE = 0,              !Failure return code
0168 :   SUCCESS = 1,              !Success return code
0169 :   BLK0 = 0,                 !Selects block 0 of struct
0170 :   BLK1 = 1,                 !Selects block 1 of struct
0171 :   BLK2 = 2,                 !Selects block 2 of struct
0172 :   BLK3 = 3,                 !Selects block 3 of struct
0173 :   WRD0 = 0,                 !Selects word 0 of block
0174 :   WRD1 = 1,                 !Selects word 1 of block
```

RC25 FORMATTER LIBRARY MODULE
FORMATTER DEFINED LITERALS11-Jul-1983 16:04:41
6-Jul-1983 11:33:38VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCHO.R16;2 (4)

```

0175 WRD2 = 2,           !Selects word 2 of block
0176 WRD3 = 3,           !Selects word 3 of block
0177 WRD4 = 4,           !Selects word 4 of block
0178 WRD5 = 5,           !Selects word 5 of block
0179 WRD6 = 6,           !Selects word 6 of block
0180 ISRD = 0,           !Select the read word (1st) ISD_STRUCTURE
0181 ISWRD = 1,          !Select the write word (2nd) in ISD_STRUCTURE
0182 STEP1 = 0,          !Initialization sequence step one
0183 STEP4 = 3,          !Initialization sequence step two
0184 TRUE = 1,           !True indicator
0185 FALSE = 0,          !False indicator
0186 SET_FLG = 1,         !Set the indicated flag to one
0187 CLR_FLG = 0,         !Clear the indicated flag to zero
0188 ONE = 1,            !Ones data type
0189 ZERO = 0,           !Zero data type
0190 ONES = %o'177777',   !All ones data type
0191 ZEROS = 0,          !All zeros data type
0192 L$LIMIT = 16,        !Allowable number of units to format
0193 DM_SIZE = 10240,     !Size of DM buffer size
0194 FCT_SIZE = 4096,     !Size of fct buffer size
0195
0196 ! Command opcode bits 6 and 7 indicate the type of message
0197 ! (command, end or attention message). The following literals
0198 ! define these field values.
0199
0200 CMD$MSG = %b'00',     !Command opcode message type
0201 ATT$MSG = %b'01',     !Attention opcode msg type
0202 END$MSG = %b'10',     !End opcode message type
0203
0204 ! Error code literals
0205
0206 PAS_CODE = %o'00',    !Pass code
0207 CIE_CODE = %o'01',    !Communication area init error
0208 CTO_CODE = %o'11',    !Controller time out error
0209 PFE_CODE = %o'21',    !Port fatal error
0210 RSE_CODE = %o'31',    !response status error
0211 PVE_CODE = %o'41',    !Host/Controller out of sequence
0212 RPD_CODE = %o'51',    !Remote program died error code
0213 PSE_CODE = %o'61',    !Port/host synchronous error
0214 MLE_CODE = %o'71',    !Message length error code
0215 UEC_CODE = %o'101',  !Unknown end code error
0216 APR_CODE = %o'201',  !Adaptor purge request error
0217 UIN_CODE = %o'301',  !Unknown interrupt error code
0218 ATN_CODE = %o'401',  !Attention msg received
0219 CMD_CODE = %o'501',  !Command msg received
0220 SEX_CODE = %o'601',  !Serious exception error received
0221 IVC_CODE = %o'701',  !Invalid command error received
0222 UMT_CODE = %o'1001', !Unknown message type
0223 OBF_CODE = %o'2001', !Out standing buffer full error
0224 OSE_CODE = %o'3001', !Out$std_buffer out of sync error
0225 UMN_CODE = %o'4001', !Unknown message number
0226 FRE_CODE = %o'5001', !File read error code from load media
0227 ILL_FCT = %o'6001', !Illegal FCT file length
0228
0229 ! Dup Protocol literals
0230
0231 ! Note:

```

```

0232 | The values assigned to the literals
0233 | REC_SIZ, SND_SIZ are represented in
0234 | powers of 2 notation per DUP spec.
0235 |
0236 | By redefining these two literals the
0237 | communications area allocation and init
0238 | sequence is automatically handled and
0239 | no other parameter modification is needed.
0240 |
0241 | Further more the send and receive rings
0242 | can be of different lengths. However the
0243 | maximum value allowed is 7 (2**7 = 128 slots)
0244 | per DUP spec.
0245 |
0246 | REC_SIZ = 2,           !Define number of receive slots
0247 | SND_SIZ = 2,           !Define number of send slots
0248 |
0249 | This one is tricky: (Hdr_siz)
0250 | The communication area is defined to be a
0251 | contiguous Blockvector (two words per block)
0252 | data segment of size, Rec_allocate + Snd_allocate +
0253 | Hdr_siz. The two words per block coming from the
0254 | two words needed to represent the ring descriptors.
0255 |
0256 | Hdr_siz is then really * 2 or 4 words of storage
0257 | to represent the interrupt and purge indicators.
0258 |
0259 | HDR_SIZ = 2,           !Define com area header size 4 words
0260 |
0261 |
0262 | REC_ALLOCATE = 1^REC_SIZ, !Define receive ring allocation
0263 | SND_ALLOCATE = 1^SND_SIZ, !Define send ring allocation
0264 |
0265 | Ring_size equals the total number of ring descriptors
0266 | (number of 2 word blocks) allocated within the
0267 | communications area.
0268 |
0269 | RING_SIZE = REC_ALLOCATE + SND_ALLOCATE + HDR_SIZ,
0270 |
0271 | The RB_SIZ by definition, DUP spec, must be
0272 | a minimum of 60 bytes long (30 words) 64 bytes
0273 | overall. The additional 2 words for the UQ
0274 | port information is accounted for in azkel2
0275 | when the storage is allocated.
0276 |
0277 | RB_SIZE = 30,          !Number of 'words' in response buffer
0278 |
0279 | The biggest command I'll ever send will
0280 | be (I hope) will be 40 bytes, 42 overall,
0281 | and again the UQ port 2 words is allowed
0282 | for in azkel2.
0283 |
0284 | SB_SIZE = 20,          !Number of 'words' in send buffer
0285 |
0286 | SNDB_SIZE = 37,        !Send DUP cmd text buffer size
0287 | RECB_SIZE = 120,       !Receive DUP cmd text buffer size
0288 | PORT_OWNED = 1,        !Descriptor owned by port

```

```

0289     HOST_OWNED = 0,                                !Descriptor owned by host
0290     :
0291     Delay literal values
0292     :
0293     An argument of one results in a 100us
0294     delay.
0295     :
0296     A roman numeral notation is used to
0297     denote values of delay arguments.
0298     The notation is as follows:
0299     :
0300     I (1), V (5), X (10), L (50), C (100),
0301     D (500), M (1000)
0302     :
0303     Any symbol following another of equal or greater
0304     value adds to its value, as II = 2, XI = 11.
0305     :
0306     Any symbol proceeding one of greater value subtracts
0307     from the second and the remainder added to the first
0308     as XIV = 14, LIX = 59.
0309     :
0310     ONE_SEC = 1000,                                  !One second delay argument
0311     ONE_MINUTE = 60000,                              !One minute delay argument
0312     C_US = 1,                                       !100 micro sec delay argument
0313     CC_US = 2,                                       !200 micro sec delay argument
0314     CCC_US = 3,                                      !300 micro sec delay argument
0315     XC_US = 4,                                       !400 micro sec delay argument
0316     D_US = 5,                                       !500 micro sec delay argument
0317     :
0318     RC25 register offsets
0319     :
0320     RCIP = 0,
0321     RCSA = 1,
0322     :
0323     Command packet opcodes
0324     (See note following)
0325     :
0326     OP_ABO = %0'1',                                  !Abort command
0327     OP_ACC = %0'20',                                  !Access command
0328     OP_AVL = %0'10',                                  !Available command
0329     OP_CCD = %0'21',                                  !Compare controller data command
0330     OP_CMP = %0'40',                                  !Compare host data command
0331     OP_DAP = %0'13',                                  !Determine access path
0332     OP_ERS = %0'22',                                  !Erase command
0333     OP_FLU = %0'23',                                  !Flush command
0334     OP_GCS = %0'02',                                  !Get command status command
0335     OP_GUS = %0'03',                                  !Get unit status command
0336     OP_ONL = %0'11',                                  !Online command
0337     OP_RD = %0'41',                                   !Read command
0338     OP_RPL = %0'24',                                  !Replace command
0339     OP_SCC = %0'04',                                  !Set controller characteristics command
0340     OP_SUC = %0'12',                                  !Set unit characteristics command
0341     OP_WR = %0'42',                                   !Write command
0342     OP_MRD = %0'30',                                  !Maintenance read command
0343     OP_MWR = %0'31',                                  !Maintenance write command
0344     :
0345     ! End message and serious exception encodes

```

```

0346      | (see note following)
0347      |
0348      | OP_END = %o'200',           !End packet flag
0349      | OP_SEX = %o'7',           !Serious exception end packet
0350      |
0351      | MSCP Attention message encodes
0352      |
0353      | OP_AVA = %o'100',         !Available attention message
0354      | OP_DUP = %o'101',         !Duplicate unit number attention message
0355      | OP_ACP = %o'102',         !Access path attention message
0356      | OP_RLC = %o'103',         !Reset command limit attention message
0357      |
0358      | The following are the dup op-code commands
0359      |
0360      | OP_GDS = %o'1',           !Get dust status
0361      | OP_ESP = %o'2',           !Execute supplied program
0362      | OP_ELP = %o'3',           !Execute local program
0363      | OP_SED = %o'4',           !Receive data
0364      | OP_RED = %o'5',           !Send data
0365      | OP_ABT = %o'6',           !Abort program
0366      |
0367      | +
0367      | NOTE:
0368      | -----
0369      |
0370      | End message opcodes (also called encodes) are formed by adding the end
0371      | message flag to the command opcode. For example, a READ commands end
0372      | message contains the value OP.RED + OP.END in its opcode field. The Invalid
0373      | command end message contains just the end message flag (i.e., OP.END) in
0374      | its opcode field. The serious exception opcode shown above (i.e. OP.SEX +
0375      | OP.END) in its opcode field.
0376      |
0377      | Commands opcode bits 6 and 7 indicate the type of message (command, end or
0378      | attention message. Command opcodes bits 3 through 5 indicate the command
0379      | category (immediate, sequential or no-sequential) and whether or not the
0380      | command includes a buffer descriptor.
0381      |
0382      | See MSCP document appendix "A-1 NOTE:" for more information on this topic.
0383      |
0384      |
0385      | DUP endcode message types
0386      |
0387      | EOP_GDS = OP_GDS + OP_END,   !Get dust status
0388      | EOP_ESP = OP_ESP + OP_END,   !Execute supplied program
0389      | EOP_ELP = OP_ELP + OP_END,   !Execute local program
0390      | EOP_RED = OP_RED + OP_END,   !Receive data
0391      | EOP_SED = OP_SED + OP_END,   !Send data
0392      | EOP_ABT = OP_ABT + OP_END,   !Abort program
0393      |
0394      | MSCP endcode messag types
0395      |
0396      | EOP_ABO = OP_ABO + OP_END,   !Abort command
0397      | EOP_ACC = OP_ACC + OP_END,   !Access command
0398      | EOP_AVL = OP_AVL + OP_END,   !Available command
0399      | EOP_CCD = OP_CCD + OP_END,   !Compare controller data command
0400      | EOP_CMP = OP_CMP + OP_END,   !Compare host data command
0401      | EOP_ERS = OP_ERS + OP_END,   !Erase command
0402      | EOP_FLU = OP_FLU + OP_END,   !Flush command

```

RC25 FORMATTER LIBRARY MODULE
FORMATTER DEFINED LITERALS

11-Jul-1983 16:04:41
6-Jul-1983 11:33:38

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCHO.R16;2 (4)

```

0403      EOP_GCS = OP_GCS + OP_END,      !Get command status command
0404      EOP_GUS = OP_GUS + OP_END,      !Get unit status command
0405      EOP_GNL = OP_ONL + OP_END,      !Online command
0406      EOP_RD = OP_RD + OP_END,        !Read command
0407      EOP_RPL = OP_RPL + OP_END,      !Replace command
0408      EOP_SCC = OP_SCC + OP_END,      !Set controller characteristics command
0409      EOP_SUC = OP_SUC + OP_END,      !Set unit characteristics command
0410      EOP_WR = OP_WR + OP_END,        !Write command
0411      EOP_MRD = OP_MRD + OP_END,      !Maintenance read command
0412      EOP_MWR = OP_MWR + OP_END,      !Maintenance write command
0413      EOP_SEX = OP_SEX + OP_END,      !Serious exception
0414
0415      !+
0416      ! Dup Command message envelope
0417      ! byte sizes beginning at text+0
0418      ! of the command envelope.
0419      !-
0420      SZ_GDS = 12,                      !Get dust status size
0421      SZ_ESP = 40,                      !Execute supplied program size
0422      SZ_ELP = 48,                      !Execute local program size
0423      SZ_RED = 28,                      !Receive data size
0424      SZ_SED = 28,                      !Send data size
0425      SZ_ABT = 12,                      !Abort program size
0426      !+
0427      ! MSCP Command message envelope
0428      ! byte sizes beginning at text+0
0429      ! of the command envelope.
0430      !-
0431      SZ_SCC = 32,                      !Set Controller characteristics
0432      SZ_ONL = 36,                      !On-line command
0433
0434      !+
0435      ! The following are the expected number of bytes
0436      ! in a commands end packet transmitted by the
0437      ! communications mechanism to the host.
0438      !-
0439      !
0440      ! DUP command end message sizes
0441      !
0442      ESZ_GDS = %DECIMAL '48',          !Get dust status end packet size
0443      ESZ_ESP = %DECIMAL '48',          !Execute supplied prog end packet size
0444      ESZ_ELP = %DECIMAL '48',          !Execute local prog end packet size
0445      ESZ_RED = %DECIMAL '48',          !Receive data end packet size
0446      ESZ_SED = %DECIMAL '48',          !Send data end packet size
0447      ESZ_ABT = %DECIMAL '48',          !Abort program end packet size
0448      !
0449      ! MSCP command end message sizes
0450      !
0451      ESZ_SCC = %DECIMAL '48',          !Set controller characteristics
0452      ESZ_ONL = %DECIMAL '48',          !On line command

```


RC25 FORMATTER LIBRARY MODULE
FIELD DECLARATIONS11-Jul-1983 16:04:41
6-Jul-1983 11:33:38VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCHO.R16;2 (5) SEQ 56
Page 10

0453 %sbttl 'FIELD DECLARATIONS'

0454
0455 field

0456

0457

0458

0459

0460

0461

0462

0463

0464

0465

0466

0467

0468

0469

0470

0471

0472

0473

0474

0475

0476

0477

0478

0479

0480

0481

0482

0483

0484

0485

0486

0487

0488

0489

0490

0491

0492

0493

0494

0495

0496

0497

0498

0499

0500

0501

0502

0503

0504

0505

0506

0507

0508

0509

Definitions:

ISD_FIELD = Initialization Sequence Data Field

ISR_ = Initialization Sequence Read = 0

ISWRT_ = Initialization Sequence Write = 1

S1R_ = Step One Read

S1W_ = Step One Write

etc.

ISD_FIELD =

set

: Miscellaneous status register field
: reference declarations.

RC_ALL = [0, 16, 0],

!RC25 word access

ERR_BIT = [15, 1, 0],

!Error bit

ISR_ALL = [0, 16, 0],

!Initialize sequence read word

ISW_ALL = [0, 16, 0],

!Initialize sequence write word

STP_FIELD = [11, 4, 0],

!All step bit fields

SA_GO = [0, 1, 0],

!Status register GO bit

ERR_CODE = [0, 11, 0],

!SA register fatal error code

: Step one read SA register field reference

S1R_STEP = [11, 1, 0],

!Step one step bit

S1R_NV = [10, 1, 0],

!No host inter vec settable adrs

S1R_QB = [9, 1, 0],

!22-bit addressing support

S1R_DI = [8, 1, 0],

!Enhanced diag implementation

S1R_OD = [7, 1, 0],

!Port allows odd host address

S1R_MP = [6, 1, 0],

!Port supports address mapping

S1R_RSVD = [0, 6, 0],

!Reserved field

: Step one write SA register field reference

S1W_WR = [14, 1, 0],

!Diag wrap around

S1W_CRING = [11, 3, 0],

!Number of C-ring slots 'pwr of 2'

S1W_RRING = [8, 3, 0],

!Number of R-ring slots 'pwr of 2'

S1W_IE = [7, 1, 0],

!Init Sequence interrupt request

S1W_VADR = [0, 7, 0],

!Interrupt vector address

```

0510
0511      | Step two read SA register field reference
0512
0513      S2R_STEP = [12, 1, 0],      |Step two step bit
0514      S2R_PTYP = [8, 3, 0],      |Port type number
0515      S2R_BIT7 = [7, 1, 0],      |Echoed IE bit from step one write
0516      S2R_WR = [6, 1, 0],        |Echoed bit 14 from step one write
0517      S2R_CRING = [3, 3, 0],     |Echoed bits 3-5 from step one write
0518      S2R_RRING = [0, 3, 0],     |Echoed bits 0-2 from step one write
0519
0520      | Step two write SA register field reference
0521
0522      S2W_LRBASE = [0, 16, 0],    |Ring base lower address
0523      S2W_PI = [0, 1, 0],        |Adapter purge interrupt request
0524
0525      | Step three read SA register field reference
0526
0527      S3R_STEP = [13, 1, 0],     |Step three step bit
0528      S3R_RSVD = [8, 3, 0],     |Reserved
0529      S3R_IE = [7, 1, 0],       |Echoed IE bit from step one write
0530      S3R_VADR = [0, 7, 0],     |Echoed VADR from step one write
0531
0532      | Step three write SA register field reference
0533
0534      S3W_PP = [15, 1, 0],       |Purge & Poll test request
0535      S3W_HRBASE = [0, 15, 0],  |Ring base high address
0536
0537      | Step four read SA register field reference
0538
0539      S4R_STEP = [14, 1, 0],     |Step four step bit
0540      S4R_RSVD = [8, 3, 0],     |Reserved
0541      S4R_MOD = [4, 4, 0],      |Encoded controller indentification
0542      S4R_VER = [0, 4, 0],      |Mod 16 value of u-code version
0543
0544      | Step four write SA register field reference
0545
0546      S4W_RSVD = [8, 8, 0],     |Reserved
0547      S4W_BURST = [2, 6, 0],    |Max number longwords per NPR xfer
0548      S4W_LF = [1, 1, 0],      |Last fail request
0549      S4W_GO = [0, 1, 0],       |Go bit
0550      tes,

```

```

0551  :
0552  :
0553  :
0554  :
0555  :
0556  :
0557  :
0558  :
0559  :
0560  :
0561  :
0562  :
0563  :
0564  :
0565  :
0566  :
0567  :
0568  :
0569  :
0570  :
0571  :
0572  :
0573  :
0574  :
0575  :
0576  :
0577  :

```

```

    *
    | Field declaration to define the interrupt
    | indicator and purge words of the communications
    | area.
    |
MDR_FIELD =
    | set
    |
    | Header Word Ringbase -4
RESERVED = [0, 0, 16, 0],
    |
    | Header Word Ringbase -3
RSVD = [1, 0, 8, 0],
ADP_CH = [1, 8, 8, 0],
    |
    | Header Word Ringbase -2
CMD_INT = [2, 0, 16, 0],
    |
    | Header Word Ringbase -1
RSP_INT = [3, 0, 16, 0]
    |
    | tes,

```

```

0578      |
0579      | Field declaration to define the fields within
0580      | the send and receive ring descriptors.
0581      |
0582      |
0583      | DSC_FIELD =
0584      | set
0585      |
0586      | Low order envelope address
0587      |
0588      | LO_EN$AD = [0, 0, 16, 0],
0589      |
0590      | High order 18 bit unibus or Qbus address
0591      |
0592      | HI_EN$AD = [1, 0, 2, 0],
0593      |
0594      | Q_Bus extention address
0595      |
0596      | QB_EXT = [1, 2, 4, 0],
0597      |
0598      | Reserver field
0599      |
0600      | D$RSVD = [1, 6, 8, 0],
0601      |
0602      | Flag bit
0603      |
0604      | FLAG_BIT = [1, 14, 1, 0],
0605      |
0606      | Ownership bit
0607      |
0608      | OWN_BIT = [1, 15, 1, 0]
0609      | tes,

```

```

0610      :+
0611      : Field declaration to define the fields within
0612      : message envelope buffers.
0613      :-
0614
0615      ENV_FIELD =
0616      set
0617      :
0618      : UQ Port envelope header field declaration
0619
0620      MSG_LENGTH = [0, 0, 16, 0],      !Message length
0621      CREDITS = [1, 0, 4, 0],          !Credits
0622      MSG_TYPE = [1, 4, 4, 0],        !Message type
0623      CONN_ID = [1, 8, 8, 0],         !Connection ID
0624
0625      : DUP/MSCP command and response envelope header
0626      : field declaration
0627
0628      CMD_LREF = [2, 0, 16, 0],        !Command Ref number low word
0629      CMD_HREF = [3, 0, 16, 0],        !Command Ref number high word
0630      UNIT_NUM = [4, 0, 16, 0],       !Unit selection field
0631      UN_LOSED = [4, 0, 16, 0],       !Unused low word
0632      UN_HUSED = [5, 0, 16, 0],       !Unused high word
0633      TYPMSG = [6, 6, 2, 0],          !End code message type
0634      OPCODE = [6, 0, 8, 0],          !Opcode
0635      ENDCODE = [6, 0, 8, 0],          !Opcode
0636      ESFLAG = [6, 8, 8, 0],          !End message flag field
0637      RSVD = [6, 8, 8, 0],            !Reserved
0638      STATUS = [7, 0, 16, 0],         !Status
0639      STA_CODE = [7, 0, 5, 0],        !Status code
0640      MODIFIER = [7, 0, 16, 0],       !Modifier
0641
0642      :++
0643      : DUP command and response envelope parameter
0644      : field declarations
0645      :--
0646
0647      :+
0648      : ABORT command and response envelope parameter
0649      : field declaration
0650      :-
0651
0652      : No parameters declared
0653      :
0654      : response FIELD
0655      : No parameters declared
0656
0657      :+
0658      : GET DUST STATUS command and response envelope
0659      : parameter field declaration
0660      :-
0661      :
0662      : COMMAND FIELD
0663      : No parameters declared
0664      :
0665      : response FIELD
0666      PLO_EXT = [8, 0, 16, 0],        !Program Extension low word

```

RC25 FORMATTER LIBRARY MODULE
FIELD DECLARATIONS11-Jul-1983 16:04:41
6-Jul-1983 11:33:38VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCHO.R16;2 (8)

```

0667     PHI_EXT = [9, 0, 8, 0],           !Program Extension high word
0668     F$LAGS = [9, 8, 8, 0],           !Flags
0669     FLG_B0 = [9, 8, 1, 0],           !Flag field bit 0
0670     FLG_B1 = [9, 9, 1, 0],           !Flag field bit 1
0671     FLG_B2 = [9, 10, 1, 0],          !Flag field bit 2
0672     FLG_B3 = [9, 11, 1, 0],          !Flag field bit 3
0673     PLO_IND = [10, 0, 16, 0],        !Progress indicator low word
0674     PHI_IND = [11, 0, 16, 0],        !Progress indicator high word
0675     TIM_OUT = [12, 0, 16, 0],        !Time out
0676
0677     +
0678     EXECUTE SUPPLIED PROGRAM command and response
0679     envelope parameter field declartion
0680     -
0681
0682             COMMAND FIELD
0683
0684     BLO_CNT = [8, 0, 16, 0],           !Byte count low word
0685     BHI_CNT = [9, 0, 16, 0],           !Byte count high word
0686     BPA_LO = [10, 0, 16, 0],          !Buffer physical adrs bits <0-15>
0687     BPA_HI = [11, 0, 2, 0],           !Buffer physical adrs bits <16-17>
0688     QBUS_EXT = [11, 2, 4, 0],         !Q bus extention
0689     RSV = [11, 6, 2, 0],              !Reserved field
0690     UBA_CHAN = [11, 8, 8, 0],         !Unibus adaptor channel number
0691     RSV0 = [12, 0, 16, 0],           !These next four words are not
0692     RSV1 = [13, 0, 16, 0],           !in the UQ port implementation.
0693     RSV2 = [14, 0, 16, 0],
0694     RSV3 = [15, 0, 16, 0],
0695
0696     These next field definitions are the same
0697     as above except they are for the overlay
0698     buffer descriptors. To make life easy for
0699     me I'll use the same names and just prefix
0700     the names with a $ for uniqueness.
0701
0702     $BPA_LO = [16, 0, 16, 0],          !Buffer physical adrs bits <0-15>
0703     $BPA_HI = [17, 0, 2, 0],          !Buffer physical adrs bits <16-17>
0704     $QBUS_EXT = [17, 2, 4, 0],        !Q bus extention
0705     $RSV = [17, 6, 2, 0],             !Reserved field
0706     $UBA_CHAN = [17, 8, 8, 0],        !Unibus adaptor channel number
0707     $RSV0 = [18, 0, 16, 0],           !These next four words are not
0708     $RSV1 = [19, 0, 16, 0],           !in the UQ port implementation.
0709     $RSV2 = [20, 0, 16, 0],
0710     $RSV3 = [21, 0, 16, 0],
0711
0712             response FIELD
0713     No parameters declared
0714
0715     +
0716     EXECUTE LOCAL PROGRAM command and response
0717     parameter field declartion
0718     -
0719
0720             COMMAND FIELD
0721
0722     PN_0 = [8, 0, 16, 0],             !Program name word 0
0723     PN_1 = [9, 0, 16, 0],             !Program name word 1

```

```

0724      PN_2 = [10, 0, 16, 0],           !Program name word 2
0725      |
0726      |         response FIELD
0727      |
0728      | version = [8, 0, 16, 0],       !Version
0729      | TIME_OUT = [9, 0, 8, 0],      !Time out
0730      | FLAGS = [9, 8, 8, 0],         !Flags
0731      |
0732      | +
0733      | SEND DATA/RECEIVE DATA command and response
0734      | parameter field declartion
0735      | -
0736      |         COMMAND FIELD
0737      |         byte count, buffer descriptor are the same
0738      |         as Execute Supplied Program parameters
0739      |
0740      |         response FIELD
0741      |         byte count is the same
0742      |         as Execute Supplied Program parameters
0743      |
0744      |
0745      | ++
0746      | MSCP command and response envelope parameter
0747      | field declarations
0748      | --
0749      |
0750      | +
0751      | SET CONTROLLER CHARACTERISTICS command and response
0752      | parameter field declartion
0753      | -
0754      |         COMMAND FIELD
0755      |
0756      |         MSCP_VER = [ 8, 0, 16, 0], !MSCP version
0757      |         CTL_FLAGS = [ 9, 0, 16, 0], !Controller flags
0758      |         HOST_TOV = [ 10, 0, 16, 0], !Host time out value
0759      |         RSSVD = [ 11, 0, 16, 0],    !Reserved
0760      |         TSD_0 = [ 12, 0, 16, 0],    !Time and Date word 0
0761      |         TSD_1 = [ 13, 0, 16, 0],    !Time and Date word 1
0762      |         TSD_2 = [ 14, 0, 16, 0],    !Time and Date word 2
0763      |         TSD_3 = [ 15, 0, 16, 0],    !Time and Date word 3
0764      |         CDP_LO = [ 16, 0, 16, 0],   !Cntlr dep parameter lo word
0765      |         CDP_HI = [ 17, 0, 16, 0],   !Cntlr dep parameter hi wrd
0766      |
0767      |         RESPONSE FIELD
0768      |
0769      |         !MSCP_VER = [ 8, 0, 16, 0],  !Same as cmd field
0770      |         !CTL_FLAGS = [ 9, 0, 16, 0], !Same as cmd field
0771      |         CTL_TOV = [ 10, 0, 16, 0],   !Cntlr time out value
0772      |         CSVRSN = [ 11, 0, 8, 0],     !Cntlr S/W, F/W, u-code rev num
0773      |         CHVRSN = [ 11, 8, 8, 0],     !Cntlr H/W rev num
0774      |         CID_0 = [ 12, 0, 16, 0],    !Cntlr identifier wrd 0
0775      |         CID_1 = [ 13, 0, 16, 0],    !Cntlr identifier wrd 1
0776      |         CID_2 = [ 14, 0, 16, 0],    !Cntlr identifier wrd 2
0777      |         CID_3 = [ 15, 0, 16, 0],    !Cntlr identifier wrd 3
0778      |
0779      |
0780      | !+

```

```

0781      ! ONLINE COMMAND command and response
0782      ! parameter field declartion
0783      !
0784      !
0785      !           COMMAND FIELD
0786      !
0787      RSV$D= [ 8, 0, 16, 0],           !Reserved
0788      UNT_FLAGS = [ 9, 0, 16, 0],    !Unit flag field
0789      RSV$0 = [ 10, 0, 16, 0],       !Reserved field
0790      RSV$1 = [ 11, 0, 16, 0],       !Reserved field
0791      RSV$2 = [ 12, 0, 16, 0],       !Reserved field
0792      RSV$3 = [ 13, 0, 16, 0],       !Reserved field
0793      RSV$4 = [ 14, 0, 16, 0],       !Reserved field
0794      RSV$5 = [ 15, 0, 16, 0],       !Reserved field
0795      DDP_LO = [ 16, 0, 16, 0],      !Device dependent parameter
0796      DDP_HI = [ 17, 0, 16, 0],      !Device dependent parameter
0797      SHADOW UNIT = [ 18, 0, 16, 0], !Shadow unit
0798      COPY_SPEED = [ 19, 0, 16, 0],  !Copy speed
0799      !
0800      !           RESPONSE FIELD
0801      !
0802      MUNT_CODE = [ 8, 0, 16, 0],     !Multi-unit code
0803      !: UNT_FLAGS = [ 9, 0, 16, 0], !Same as cmd field
0804      !: RSV$0 = [ 10, 0, 16, 0],    !Same as cmd field
0805      !: RSV$1 = [ 11, 0, 16, 0],    !Same as cmd field
0806      !: UID_0 = [ 12, 0, 16, 0],    !Unit ident word 0
0807      !: UID_1 = [ 13, 0, 16, 0],    !Unit ident word 1
0808      !: UID_2 = [ 14, 0, 16, 0],    !Unit ident word 2
0809      !: UID_3 = [ 15, 0, 16, 0],    !Unit ident word 3
0810      !: MTID_LO = [ 16, 0, 16, 0],  !Media type ident word 0
0811      !: MTID_HI = [ 17, 0, 16, 0], !Media type ident word
0812      !: SHADOW UNIT = [ 18, 0, 16, 0], !Same as cmd
0813      !: SHA_STATE = [ 19, 0, 16, 0], !Shadow state
0814      !: USZ_LO = [ 20, 0, 16, 0],   !Unit size lo word
0815      !: USZ_HI = [ 21, 0, 16, 0],   !Unit size hi word
0816      !: VSN_LO = [ 22, 0, 16, 0],   !Volume serial num lo word
0817      !: VSN_HI = [ 23, 0, 16, 0],   !Volume serial num hi word
0818      tes.
0819

```


RC25 FORMATTER LIBRARY MODULE
FIELD DECLARATIONS

11-Jul-1983 16:04:41
6-Jul-1983 11:33:38

VAX-11 Bliss-16 V3-555
SPIDER\$USERS:[NEALE.AZTEC]CZRCHO.R16;2 (9)

```

0820      !+
0821      ! Send Receive command buffer field definition
0822      !-
0823      RECB_FIELD =
0824      set
0825      MSG_NUM = [0, 0, 12, 0],
0826      MSG_TYP = [0, 12, 4, 0],
0827      MSG_TXT = [1, 0, 16, 0]
0828      tes;
0829
0830      !+
0831      ! Outstanding command buffer field declarations
0832      !-
0833      OUT$FIELD =
0834      set
0835      CMD_WRD = [0, 0, 16, 0],           !Command word ref 'word 0 of slot'
0836      REC_FLG = [0, 15, 1, 0],        !Command received indicator flag
0837      CMD_REF = [0, 0, 8, 0],         !Command reference field
0838      ENV_ADR = [1, 0, 16, 0]        !Envelope adrs field
0839      tes;
0840

```

0841 %sbttl 'LINKAGE DELCARATIONS'

0842

0843 Linkage

0844

0845 Call_lnk\$typ

0846

0847

0848

0849

0850

0851

0852

0853

0854

0855

0856

0857

0858

0859

0860

0861

0862 CALL_LNK\$TYP = call (standard),

0863

0864

0865

0866 Int_lnk\$typ

0867

0868

0869

0870

0871

0872

0873

0874

0875

0876

0877

0878

0879 INT_LNK\$TYP = interrupt (standard);

0879

COMMAND QUALIFIERS

BLISS /PDP11/LIST/LIBRARY CZRCHO.R16

: Run Time: 00:06.0
: Elapsed Time: 00:07.6
: Memory Used: 51 pages
: Library Precompilation Complete

```

0001 MODULE CZRCH2 (%TITLE 'CZRCH RC25 DISK FORMATTER'
0002             IDENT = 'REV A PATCH 00',
0003             ADDRESSING MODE (RELATIVE) ,
0004             ENVIRONMENT (NOEIS)
0005             ) =
0006 BEGIN
0007 %sbttl 'PROGRAM HEADER'
0008 |
0009 | Pretty Declarations
0010 |
0011 | <BLF/LOWERCASE_KEY>
0012 |
0013 |
0014 | Library 'CZRCH0';                !Define RC25 Formatter Library
0015 |
0016 | require 'BLSMAC.REQ';          !Define Bliss Macro Require file
1505 |
1506 | +
1507 | The psect named 'code or $code$' is redefined here
1508 | to be called 'aa$code'. This is done to force the TKB
1509 | linker to place the programs header information starting
1510 | at absolute address 2000.
1511 | -
1512 | psect
1513 |     code = aa$code;
1514 |
1515 | literal
1516 |     DS$NBR_OF_TESTS = 1;        !Indicates number of test in Diag
1517 |
1518 | +
1519 | The structure of a diagnostic program may contain any or all of the
1520 | ten optional sections. But five of the optional sections require a
1521 | pointer that is derived by and for the supervisor, and is located in
1522 | the header block. Therefore, in relation to the effective use of
1523 | these five pointers, the optional sections call must be coded to re-
1524 | flect usage (i.e., any,all,or none).
1525 |
1526 | The following coding possibilities exist:
1527 |
1528 |     POINTER (BGNRPT,BGNSW,BGNSFT,BGNAU,BGNDU,ERRTBL,BGNSETUP)
1529 |
1530 |     (or any subset of the args)
1531 |
1532 |     POINTER (ALL)                : All provides pointers for all five
1533 |                                     : sections
1534 |     POINTER (NONE)              : None indicates to supervisor that no
1535 |                                     : pointers are required.
1536 |                                     : this is the default
1537 |
1538 |     No pointers are optional using bliss. Make sure the following
1539 |     sections of codc are in place (in the correct skels),even if
1540 |     the sections are blank.
1541 |
1542 |     ARGUMENT      FUNCTION
1543 |     -----      -
1544 |     RPT           REPORT CODE
1545 |     SW           SOFTWARE TABLE

```

CZRCH2
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
PROGRAM HEADER

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH2.B16;2 (1) SEQ 67
Page 2

```

:      1546 | SFT          SOFTWARE TABLE QUESTIONS
:      1547 | AU          ADD CODE
:      1548 | DU          DROP CODE
:      1549 | TBL        ERROR TABLE
:      1550 | SETUP      ASSEMBLED P-TABLES
:      1551 | -
:      1552 | POINTER (ALL);
:      1553 | +
:      1554 | The program header section contains general information which des-
:      1555 | cribes the major characteristics of the diagnostic program. This in-
:      1556 | cludes, the program name, and revision and patch-order levels. The
:      1557 | header also provides space for an event flag register, and for the
:      1558 | storage of pointers, through which the supervisor may find access to
:      1559 | other key sections of the program(e.g., dispatch table, initialize and
:      1560 | clean-up code, etc.). An argument on the header gives the device type
:      1561 | if it is an XXDP+ bootable device. This enables the supervisor to pro-
:      1562 | vide load medium protection when necessary.
:      1563 | -
:      1564 | HEADER (%ascii'CZRCH ', %ascii'A', %ascii'0', 1200, 0, PRI00);

```

CZRCH2
REV A PATCH 00

CZRCH RC25 DISK FORM/TER
DISPATCH TABLE

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH2.B16;2 (2)

```

:      1565 %sbttl 'DISPATCH TABLE'
:      1566 !+
:      1567 ! The dispatch table section contains address pointers to the various
:      1568 ! tests contained within the diagnostic program. This section requires
:      1569 ! the coding of only the dispatch macro.
:      1570 !-
:      1571 DISPATCH (DSS$NBR_OF_TESTS);
:      1572 ERRTBL;                                !Define Supervisor Error table storage

```

CZRCH2
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
DEFAULT HARDWARE P-TABLE

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH2.B16;L P

```

:      1573 %sbttl 'DEFAULT HARDWARE P-TABLE'
:      1574 :+
:      1575 : The default hardware P-Table contains default values of
:      1576 : the test-device parameters. The structure of this table
:      1577 : is identical to the structure of the hardware P-Tables,
:      1578 : and is used as a "template" for building the P-Tables.
:      1579 :-
:      1580 BGNHW (DFPTBL);
:      1581
:      1582 global
:      1583     HW_IP_ADRS : word initial (%'172150'),      !Define RC25 Controler IP reg
:      1584     HW_VECTOR : word initial (%'154'),      !Define RC25 interrupt vector addrs
:      1585     HW_BR_LEVEL : word initial (5),          !Define RC25 bus request level
:      1586     HW_UNIT_NO : word initial (0);          !Define RC25 unit no. to format
:      1587
:      1588 ENDHW;

```

CZRCH2
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
SOFTWARE P-TABLE

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH2.B16;2 (4)

```

:      1589 %sbttl 'SOFTWARE P-TABLE'
:      1590 :+
:      1591 : The software table contains various data used by the
:      1592 : program as operational parameters. These parameters are
:      1593 : set up at assembly time and may be varied by the operator
:      1594 : at run time.
:      1595 :-
:      1596 BGNSW (SFPTBL);
:      1597
:      1598 global
:      1599     SW_UNATT : word initial (1);           !format in unattended refmt mode
:      1600
:      1601 ENDSW;

```

CZRCH2
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
PROTECTION TABLE

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

VAX-11 Bliss-16 V3-555
SPIDERSUSERS:[NEALE.AZTEC]CZRCH2.B16;2 (5) SEQ 71
Page 6

```
: 1602 %sbttl 'PROTECTION TABLE'  
: 1603 :  
: 1604 : This table is used by the runtime  
: 1605 : services to protect the load media.  
: 1606 :  
: 1607 : 1st arg = Offset into P_Table for csr address  
: 1608 : 2nd arg = Offset into P_Table for massbus address  
: 1609 : 3rd arg = Offset into P_Table for drive number  
: 1610 : -  
: 1611 BGNPROT (-1, -1, -1);  
: 1612 ENDPROT;
```


CZRCH2
REV A PATCH 00CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS11-Jul-1983 16:05:07
7-Jul-1983 08:21:37VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH2.B16;2 (6)SEQ 72
Page 7

```

1613 %sbttl 'MODULE DECLARATIONS'
1614 !+
1615 ! Within BLSMAC.REQ the psect names, plit global and own,
1616 ! are redefined to be aa$code. This is done to force the
1617 ! tkb linker to link the header information starting at
1618 ! at absolute address 2000. Redefine these psect names
1619 ! back to their original names for house keeping purposes.
1620 !
1621 ! Also change the attributes for the psect 'global' so that
1622 ! global data will not be linked starting at absolute address
1623 ! 2000.
1624 !-
1625 psect
1626     plit = $plit$( global),
1627     global = $glob$(nowrite, noexecute, global, concatenate),
1628     own = $own$;
1629
1630 !+
1631 ! Structure declarations used within this
1632 ! module.
1633 !-
1634
1635 structure
1636
1637     !+
1638     ! RC25 register accessing structure. This
1639     ! structure allows RC25 register accessing
1640     ! to be transportable between the PDP-11 and
1641     ! VAX Diagnostic Supervisors.
1642     !
1643     ! This also defines an access algorithm for
1644     ! VAX to allow field reference to MBA address
1645     ! space without generating machine checks.
1646     !-
1647
1648     RC25 [O, P, S, E] =
1649         begin
1650
1651             local
1652                 RC$$_REG;
1653
1654                 RC$$_REG = .(RC25 + %upval*0)<0, %bpval, 0>;
1655                 RC$$_REG
1656             end
1657             <P, S, E>;
1658

```

CZRCH2
REV A PATCH 00CZRCH RC25 DISK FORMATTER
GLOBAL DATA SECTION11-Jul-1983 16:05:07
7-Jul-1983 08:21:37VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH2.B16;2 (7) SEQ 73
Page 8

```

1659 %sbttl 'GLOBAL DATA SECTION'
1660
1661 :+ The global data section contains data that are used
1662 : in more than one test or module.
1663 :-
1664
1665 global
1666 :
1667 : Communication area Declarations
1668 :
1669 COM_AREA : blockvector [REC_ALLOCATE + SND_ALLOCATE + HDR_SIZ, 2, word],
1670 HEAD_AREA : ref block [4, word] field (HDR_FIELD),
1671 RECEIVE_RING : ref blockvector [REC_ALLOCATE, 2, word] field (DSC_FIELD),
1672 SEND_RING : ref blockvector [SND_ALLOCATE, 2, word] field (DSC_FIELD),
1673 REC_ENVELOPE : blockvector [REC_ALLOCATE, RB_SIZE + 2, word] field (ENV_FIELD),
1674 SND_ENVELOPE : blockvector [SND_ALLOCATE, SB_SIZE + 2, word] field (ENV_FIELD),
1675 FCT_BUF : block [256, word],
1676 REC_BUF : block [RECB_SIZE, word] field (RECB_FIELD),
1677 SND_BUF : vector [SNDB_SIZE, word],
1678 OUT$STD_BUF : blockvector [REC_ALLOCATE, 2, word] field (OUT$FIELD),
1679 RET_EN$AD : ref block [RB_SIZE + 2, word] field (ENV_FIELD);
1680
1681 global bind
1682 :
1683 : Diagnostic supervisor printing ascii format strings.
1684 :
1685 FMT1 = uplit (%asciz'%NXT'), !Print one ascii string pointer
1686 FMT2 = uplit (%asciz'%N%N%AINITIALIZING RC25 CONTROLLER ADDRESS:%S%06%A(0)%N'),
1687 FMT3 = uplit (%asciz'%N%ALOGICAL UNIT %D2%A(D)%S%APHYSICAL UNIT %D2%A(D)%S%AFORMAT ABORTED'),
1688 FMT4 = uplit (%asciz'%N%AMICRO CODES MOD 16 VERSION NUMBER =%S%02%A(0)%N'),
1689 FMT5 = uplit (%asciz'%N%ABRINGING ONLINE: LOGICAL UNIT %D2%A(D)%S%APHYSICAL UNIT %D2%A(D)%N'),
1690 FMT6 = uplit (%asciz'%N%A$FTLERR- NON-EXISTENT RC25 CONTROLLER ADDRESS %06%A(0)%N'),
1691 FMT7 = uplit (%asciz'%A$FORMATTING PHYSICAL UNIT %D2%A(D)%N'),
1692 PID_FMT = uplit (%asciz'%N%A$FORMATTER PROGRESSING: PID HI=%06%A(0)%S%$APID LO=%06%A(0)'),
1693 CRLF = uplit (%asciz'%N'),
1694 XCRLF = uplit (%asciz'%N%N%N%N%N%N%N%N%N%N'),
1695 :
1696 : Ring base address declaration
1697 :
1698 RINGBASE = COM_AREA [REC_BASE],
1699 :
1700 :
1701 :
1702 MSGADR = REC_BUF [MSG_TXT];
1703
1704 global
1705 :
1706 : Miscellaneous data declarations
1707 :
1708 DATETXT : vector [12, byte], !Operator data input storage
1709 FLG_WRD : bitvector [16], !Global flag word
1710 NEX_FLAG : word, !Non-existent RC25 register flag
1711 OVSA : word, !Overlay section starting adrs
1712 NXT_CRN : byte, !Stores next cmd ref number
1713 RET_STATUS : word initial (%o'000000'), !Saves various return status codes
1714 LUN : word, !Stores logical unit number being formatted
1715 PID_SAVE : vector [2, word], !Saves proces indicator word

```

CZRCH2
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
GLOBAL DATA SECTION

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

VAX-11 Bliss-16 V3-555
SPIDERSUSERS:[NEALE.AZTEC]CZRCH2.B16;2 (7)

```

1716 NSD_SLOT : word, !Next send Descriptor slot
1717 NRD_SLOT : word, !Next receive Descriptor slot
1718 :
1719 : Hardware P_Table storage declarations
1720 :
1721 RC25_ADDR : ref RC25 field (ISD_FIELD), !Controller register access structure
1722 VEC_ADDR : word, !Interrupt vector address storage
1723 BR_LEVEL : word, !Bus request level storage
1724 UNIT_NO : word, !Unit number to format storage
1725 PTBL_PTR : ref vector [4, word], !Stores P_Table base address
1726 :
1727 : Dup Protocol data structures
1728 :
1729 :+
1730 :Reserved field mask structure declaration
1731 :-
1732 RSVD_STRUCT : vector [4, word] preset ( !Reserved SA reg fields definitions
1733 [0] = %x'002FF', !Step one rsvd field
1734 [1] = %x'0000', !Step two rsvd field
1735 [2] = %x'0700', !Step three rsvd field
1736 [3] = %x'07FF'), !Step four rsvd & ucode field
1737 :
1738 :+
1739 ! Init Sequence Data_Structure declaration
1740 :-
1741 :
1742 ISD_STRUCT : blockvector [4, 2, word] field (ISD_FIELD) preset (
1743 :
1744 : Step one read SA register field declaration
1745 :
1746 [BLK0, WRD0, ERR_BIT] = 0, !Error bit
1747 [BLK0, WRD0, STP_FIELD] = %b'0001', !All step bit fields
1748 [BLK0, WRD0, S1R_NV] = 0, !No host inter vec settable adrs
1749 [BLK0, WRD0, S1R_QB] = 1, !Mask out
1750 [BLK0, WRD0, S1R_DI] = 1, !Enhanced diag implementation
1751 [BLK0, WRD0, S1R_OD] = 1, !Mask out
1752 [BLK0, WRD0, S1R_MP] = 1, !Mask out
1753 [BLK0, WRD0, S1R_RSVD]= %x'3f', !Reserved field
1754 :
1755 : Step one write SA register field declaration
1756 :
1757 [BLK0, WRD1, ERR_BIT] = 1, !Error bit
1758 [BLK0, WRD1, S1W_WR]= 0, !Diag wrap around
1759 [BLK0, WRD1, S1W_CRING]= SND_SIZ, !Number of Send-ring slots 'pwr of 2'
1760 [BLK0, WRD1, S1W_RRING]= REC_SIZ, !Number of Receive-ring slots 'pwr of 2'
1761 [BLK0, WRD1, S1W_IE]= 0, !Init Sequence interrupt request
1762 [BLK0, WRD1, S1W_VADR]= %o'33', !Interrupt vector address
1763 :
1764 : Step two read SA register field declaration
1765 :
1766 [BLK1, WRD0, ERR_BIT] = 0, !Error bit
1767 [BLK1, WRD0, STP_FIELD] = %b'0010', !All step bit fields
1768 [BLK1, WRD0, S2R_PTYP]= 0, !Port type number
1769 [BLK1, WRD0, S2R_BIT7]= 1, !Echoed IE bit from step one write
1770 [BLK1, WRD0, S2R_WR]= 0, !Echoed bit 14 from step one write
1771 [BLK1, WRD0, S2R_CRING]= SND_SIZ, !Echoed bits 3-5 from step one write
1772 [BLK1, WRD0, S2R_RRING]= REC_SIZ, !Echoed bits 0-2 from step one write

```

```

1773
1774      Step two write SA register field declaration
1775
1776      [BLK1, WRD1, S2W_LRBASE]= RINGBASE,           !Ring base lower address
1777
1778      NOTE:
1779      The adapter purge interrupt is loaded within
1780      the bgninit code due to the inability to field
1781      select bits <1, 15, 0> from the ringbase adrs.
1782      [BLK1, WRD1, S2W_PI]= 0,                       !Adapter purge interrupt request
1783
1784      Step three read SA register field declaration
1785
1786      [BLK2, WRD0, ERR_BIT] = 0,                       !Error bit
1787      [BLK2, WRD0, STP_FIELD] = %b'0100',             !All step bit fields
1788      [BLK2, WRD0, S3R_RSVD]= %o'7',                 !Reserved
1789      [BLK2, WRD0, S3R_IE]= 0,                       !Echoed IE bit from step one write
1790      [BLK2, WRD0, S3R_VADR]= %o'33',                !Echoed VADR from step one write
1791
1792      Step three write SA register field declaration
1793
1794      [BLK2, WRD1, S3W_PP]= 0,                       !Purge & Poll test request
1795      [BLK2, WRD1, S3W_HRBASE]= 0,                   !Ring base high address
1796
1797      Step four read SA register field declaration
1798
1799      [BLK3, WRD0, ERR_BIT] = 0,                       !Error bit
1800      [BLK3, WRD0, STP_FIELD] = %b'1000',             !All step bit fields
1801      [BLK3, WRD0, S4R_RSVD]= %o'7',                 !Reserved
1802      [BLK3, WRD0, S4R_MOD] = %x'f',                 !Mask it out
1803      [BLK3, WRD0, S4R_VER] = %x'f',                 !Mask it out
1804
1805      Step four write SA register field declaration
1806
1807      [BLK3, WRD1, S4W_RSVD]= %o'377',                !Reserved
1808      [BLK3, WRD1, S4W_BURST]= 0,                     !Max number longwords per NPR xfer
1809      [BLK3, WRD1, S4W_LF]= 0,                       !Last fail request
1810      [BLK3, WRD1, S4W_GO]= 0);                       !Go bit
1811

```

```

1812 %sbttl 'GLOBAL TEXT SECTION'
1813 !+
1814 ! The global text section contains format statements,
1815 ! messages, and ASCII information that are used in
1816 ! all modules.
1817 !-
1818
1819 global bind
1820 !
1821 ! Self-detected fatal port/controller errors
1822 !
1823 PFE_STRUCT = uplit (
1824     uplit (%asciz'%NXASFTLERR- UNRECOGNIZABLE ERROR CODE'),
1825     uplit (%asciz'%NXASFTLERR- ENVELOPE/PACKET READ (PARITY OR TIMEOUT)'),
1826     uplit (%asciz'%NXASFTLERR- ENVELOPE/PACKET WRITE (PARITY OR TIMEOUT)'),
1827     uplit (%asciz'%NXASFTLERR- CONTROLLER ROM AND RAM PARITY'),
1828     uplit (%asciz'%NXASFTLERR- CONTROLLER RAM PARITY'),
1829     uplit (%asciz'%NXASFTLERR- CONTROLLER ROM PARITY'),
1830     uplit (%asciz'%NXASFTLERR- RING READ (PARITY OR TIMEOUT)'),
1831     uplit (%asciz'%NXASFTLERR- RING WRITE (PARITY OR TIMEOUT)'),
1832     uplit (%asciz'%NXASFTLERR- INTERRUPT MASTER'),
1833     uplit (%asciz'%NXASFTLERR- HOST ACCESS TIMEOUT'),
1834     uplit (%asciz'%NXASFTLERR- CREDIT LIMIT EXCEEDED'),
1835     uplit (%asciz'%NXASFTLERR- BUS MASTER ERROR'),
1836     uplit (%asciz'%NXASFTLERR- DIAGNOSTIC CONTROLLER FATAL ERROR'),
1837     uplit (%asciz'%NXASFTLERR- INSTRUCTION LOOP TIMEOUT'),
1838     uplit (%asciz'%NXASFTLERR- INVALID CONNECTION IDENTIFIER'),
1839     uplit (%asciz'%NXASFTLERR- INTERRUPT WRITE'),
1840     uplit (%asciz'%NXASFTLERR- MAINTENANCE READ/WRITE INVALID REGION IDENTIFIER'),
1841     uplit (%asciz'%NXASFTLERR- MAINTENANCE WRITE LOAD TO NON-LOADABLE CONTROLLER'),
1842     uplit (%asciz'%NXASFTLERR- CONTROLLER RAM ERROR (NON-PARITY)'),
1843     uplit (%asciz'%NXASFTLERR- INIT SEQUENCE ERROR'),
1844     uplit (%asciz'%NXASFTLERR- HIGH LEVEL PROTOCOL INCOMPATIBILITY ERROR'),
1845     uplit (%asciz'%NXASFTLERR- PURGE/POLL HARDWARE FAILURE '),
1846     uplit (%asciz'%NXASFTLERR- MAPPING REGISTER READ ERROR (PARITY OR TIMEOUT)'),
1847     ) : vector [23],
1848 !<BLF/PAGE>

```

```

:      1849      |
:      1850      | Error message structure
:      1851      |
:      1852      | EMSG_STRUCT = uplit (
:      1853      | uplit (%asciz'%ZN$ASFTLERR- RESPONSE STATUS ERROR:%S'),
:      1854      | uplit (%asciz'%ZN$ASFTLERR- HOST/CONTROLLER OUT OF SEQ'),
:      1855      | uplit (%asciz'%ZN$ASFTLERR- REMOTE PROG NOT RUNNING'),
:      1856      | uplit (%asciz'%ZN$ASFTLERR- UNKNOWN RETURN STATUS CODE'),
:      1857      | uplit (%asciz'%ZN$ASFTLERR- COM AREA INIT ERROR'),
:      1858      | uplit (%asciz'%ZN$ASFTLERR- PORT/HOST SYNC ERROR'),
:      1859      | uplit (%asciz'%ZN$ASFTLERR- MESSAGE LENGTH ERROR'),
:      1860      | uplit (%asciz'%ZN$ASFTLERR- UNKNOWN ENDCODE RECEIVED'),
:      1861      | uplit (%asciz'%ZN$ASFTLERR- ADAPTOR PURGE ERROR'),
:      1862      | uplit (%asciz'%ZN$ASFTLERR- UNKNOWN INTERRUPT'),
:      1863      | uplit (%asciz'%ZN$ASFTLERR- INIT SEQ STEP TIMED OUT'),
:      1864      | uplit (%asciz'%ZN$ASFTLERR- INIT SEQ COMPARE ERROR'),
:      1865      | uplit (%asciz'%ZN$ASFTLERR- UNEXPECTED ATTENTION END MESSAGE RECEIVED'),
:      1866      | uplit (%asciz'%ZN$ASFTLERR- UNEXPECTED COMMAND OPCODE IN END MESSAGE RECEIVED'),
:      1867      | uplit (%asciz'%ZN$ASFTLERR- UNEXPECTED SERIOUS EXCEPTION END MESSAGE RECEIVED'),
:      1868      | uplit (%asciz'%ZN$ASFTLERR- INVALID COMMAND END MESSAGE RECEIVED'),
:      1869      | uplit (%asciz'%ZN$ASFTLERR- UNKNOWN MESSAGE TYPE RECEIVED'),
:      1870      | uplit (%asciz'%ZN$ASFTLERR- OUTSTANDING COMMAND BUFFER FULL'),
:      1871      | uplit (%asciz'%ZN$ASFTLERR- OUT STANDING COMMAND BUFFER OUT OF SYNC ERROR'),
:      1872      | uplit (%asciz'%ZN$ASFTLERR- UNKNOWN MESSAGE NUMBER RECEIVED'),
:      1873      | uplit (%asciz'%ZN$ASFTLERR- FILE READ ERROR'),
:      1874      | uplit (%asciz'%ZN$ASFTLERR- PORT/CONTROLLER TIMEOUT ERROR'),
:      1875      | uplit (%asciz'%ZN$ASFTLERR- ILLEGAL FCT FILE LENGTH')) : vector [23],
:      1876      | !<blf/page>

```

```
1877      |
1878      | Self-detected fatal port/controller errors
1879      |
1880      | RC STRUCTURE = uplit (
1881 uplit (%asciz'%N%$FTLERR- VAX READ/WRITE ERROR ON INTERRUPT'),
1882 uplit (%asciz'%N%$FTLERR- INCONSISTENCY AT U.BFIL'),
1883 uplit (%asciz'%N%$FTLERR- INCONSISTENCY AT U.BMTY'),
1884 uplit (%asciz'%N%$FTLERR- INCONSISTENCY AT U.ALOC'),
1885 uplit (%asciz'%N%$FTLERR- INCONSISTENCY AT SERVO ENTRY (PIP SET)'),
1886 uplit (%asciz'%N%$FTLERR- INCONSISTENCY AT SERVO ENTRY (ERR SET)'),
1887 uplit (%asciz'%N%$FTLERR- INCONSISTENCY AT U.SEND'),
1888 uplit (%asciz'%N%$FTLERR- INCONSISTENCY AT U.RECV'),
1889 uplit (%asciz'%N%$FTLERR- INCONSISTENCY AT U.ATTN'),
1890 uplit (%asciz'%N%$FTLERR- INCONSISTENCY AT U.ONLN'),
1891 uplit (%asciz'%N%$FTLERR- ILLEGAL D REQUEST (U.QDRQ)'),
1892 uplit (%asciz'%N%$FTLERR- FENCE-POST ERROR AT PROTAB'),
1893 uplit (%asciz'%N%$FTLERR- BAD PACKET DEQUEUED AT U.DONE'),
1894 uplit (%asciz'%N%$FTLERR- UNEXPLAINED D-PROC SUSPENSION (U..TDS)'),
1895 uplit (%asciz'%N%$FTLERR- DUP PACKET D-Q FAILED (XFC 34/35)'),
1896 uplit (%asciz'%N%$FTLERR- INCONSISTENCY AT U.HTST'),
1897 uplit (%asciz'%N%$FTLERR- INCONSISTENCY AT U.SEKO'),
1898 uplit (%asciz'%N%$FTLERR- INCONSISTENCY AT U.CKSV'),
1899 uplit (%asciz'%N%$FTLERR- D.OPCD FOUND ILLEGAL OPCODE'),
1900 uplit (%asciz'%N%$FTLERR- D.CSF FOUND ILLEGAL OPCODE'),
1901 uplit (%asciz'%N%$FTLERR- UNKNOWN BAD DRIVE STATUS AT D.DSTS'),
1902 uplit (%asciz'%N%$FTLERR- ILLEGAL XFC EXECUTED BY DM'),
1903 uplit (%asciz'%N%$FTLERR- D PICKED UP A ZERO SCB.DB'),
1904 uplit (%asciz'%N%$FTLERR- INCONSISTENCY AT D IDLE LOOP'),
1905 uplit (%asciz'%N%$FTLERR- DM WORD COUNT ERROR ON HOST DMA/SEND/RECV'),
1906 uplit (%asciz'%N%$FTLERR- UNKNOWN DISPLAY FAULT CODE AT D.DFLT'),
1907 uplit (%asciz'%N%$FTLERR- DRIVE NOT FAULTING IN P.OFLN STATE'),
1908 uplit (%asciz'%N%$FTLERR- U POWER UP DIAGNOSTICS FAILED'),
1909 uplit (%asciz'%N%$FTLERR- D POWER UP DIAGNOSTICS FAILED'),
1910 uplit (%asciz'%N%$FTLERR- ADAPTER CARD FAILURE'),
1911 uplit (%asciz'%N%$FTLERR- EC.TMR TIMED OUT'),
1912 uplit (%asciz'%N%$FTLERR- U.SEND/U.RECV RING READ INCONSISTENCY'),
1913 uplit (%asciz'%N%$FTLERR- UNKNOWN WAITRV REASON AT D.RVCT'),
1914 uplit (%asciz'%N%$FTLERR- D.ARCS DID NOT FIND CLOSEST UNDONE ZONE'),
1915 uplit (%asciz'%N%$FTLERR- U.SEEK FOUND SEEK TO ILLEGAL TRACK'),
1916 uplit (%asciz'%N%$FTLERR- U.HTST INIT DIAG DMA WRITE FAILED'),
1917 uplit (%asciz'%N%$FTLERR- U.HTST INIT DIAG DMA COMPARE FAILED'),
1918 uplit (%asciz'%N%$FTLERR- U.SYDR FOUND SS.DER SET AND SS.SPN NOT SET'),
1919 uplit (%asciz'%N%$FTLERR- MASTER DRIVES ACLO ASSERTED')
1920      | ) : vector [39],
1921      | !<blf/page>
```

CZRCH2
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH2.B16;2 (11)

```

:      1922      |
:      1923      | Dup return status codes
:      1924      |
:      1925      | SDUP_STRUCT = uplit (
:      1926      | uplit (%asciz'%A SUCCESSFUL%N'),
:      1927      | uplit (%asciz'%AINVALID COMMAND%N'),
:      1928      | uplit (%asciz'%ANO REGION AVAILABLE%N'),
:      1929      | uplit (%asciz'%ANO REGION SUITABLE%N'),
:      1930      | unlit (%asciz'%APROGRAM NOT KNOWN%N'),
:      1931      | uplit (%asciz'%ALOAD FAILURE%N'),
:      1932      | uplit (%asciz'%ASTANDALONE%N')
:      1933      | ) : vector [7],
:      1934      | !<blf/page>

```


CZRCH2
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH2.B16;2 (12)

```

:
: 1935      |
: 1936      | MSCP return status codes
: 1937      |
: 1938      | SMSCP_STRUCT = uplit (
: 1939      | uplit (%asciz'%SUCCESS%N'),
: 1940      | uplit (%asciz'%INVALID COMMAND%N'),
: 1941      | uplit (%asciz'%COMMAND ABORTED%N'),
: 1942      | uplit (%asciz'%AUNIT-OFFLINE%N'),
: 1943      | uplit (%asciz'%AUNIT-AVAILABLE%N'),
: 1944      | uplit (%asciz'%MEDIA FORMAT ERROR%N'),
: 1945      | uplit (%asciz'%WRITE PROTECTED%N'),
: 1946      | uplit (%asciz'%COMPARE ERROR%N'),
: 1947      | uplit (%asciz'%ADATA ERROR%N'),
: 1948      | uplit (%asciz'%HOST BUFFER ACCESS ERROR%N'),
: 1949      | uplit (%asciz'%CONTROLLER ERROR%N'),
: 1950      | uplit (%asciz'%DRIVE ERROR%N'),
: 1951      | uplit (%asciz'%MESSAGE FROM AN INTERNAL DIAGNOSTIC%N')
: 1952      | ) : vector [13],
: 1953      | !<BLF/PAGE>
:

```

CZRCH2
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH2.B16;2 (13)

```

1954      |
1955      | Init code error and informational messages
1956      |
1957      | DATMSG = uplit (%asciz'ENTER DATE <MM-DD-YYYY> '),
1958      | NO_ADD_UNITS = uplit (%asciz'%NZANO ADDITIONAL UNITS TO FORMAT - ABORTING'),
1959      | PWR_MSG = uplit (%asciz'%NZASFTLERR- INIT CODE RE-ENTERED DUE TO PWR FAIL'),
1960      | ABO_MSG = uplit (%asciz'%NZASFTLERR- ABORTING HOST AND REMOTE PROGRAMS'),
1961      | TO_MANY_UNITS = uplit (%asciz'%NZASFTLERR- ILLEGAL NUMBER OF UNITS SELECTED'),
1962      | GOOD_NUM_UNITS = uplit (%asciz'%NZASFTLERR- LIMIT OF SIXTEEN UNITS PER FORMATING SESSION'),
1963      | BOOT_FAILURE = uplit (%asciz'%NZASFTLERR- RC25 CONTROLLER INITIALIZATION ERROR'),
1964      | PROT_VIOLATION = uplit (%asciz'%NZASFTLERR- PROTOCOL VIOLATION ERROR'),
1965      | PORT_INIT_ERR = uplit (%asciz'%NZASFTLERR- COMMUNICATION AREA INIT ERROR'),
1966      | ACTIVE_DUP_SERVER = uplit (%asciz'%NZASFTLERR- DUP SERVER ACTIVE AFTER INITIALIZING'),
1967      | INACTIVE_DUP_SERVER = uplit (%asciz'%NZASFTLERR- DUP SERVER INACTIVE AFTER EX_SUP_PROG COMMAND'),
1968      |
1969      | Local load media DM module file name.ext
1970      |
1971      | DM_FN$EXT = UPLIT (%ASCIZ'AZFMTR.SAV'),
1972      | FCT_REQ_MSG = uplit (%asciz'ENTER FCT FILE NAME TO RESTORE (FILENAME.EXT): '),
1973      |
1974      | Hardware parameter coding questions
1975      |
1976      | HW_Q1_IP = uplit (%asciz'RC25 IP REGISTER ADDRESS'),
1977      | HW_Q2_VECTOR = uplit (%asciz'RC25 INTERRUPT VECTOR ADDRESS'),
1978      | HW_Q3_BR = uplit (%asciz'RC25 BUS REQUEST LEVEL'),
1979      | HW_Q4_UNIT = uplit (%asciz'UNIT NUMBER TO BRING ONLINE'),
1980      |
1981      | Software parameter coding questions
1982      |
1983      | SW_Q1_UNATT = uplit (%asciz'FORMAT IN UNATTENDED REFORMAT MODE'),
1984      | SW_Q2_NOTICE = uplit (%asciz'%NZ***** NOTICE *****'),
1985      | SW_Q3_OPER = uplit (%asciz'%NZ OPERATOR MUST BE PRESENT IN ATTENDED MODE '),
1986      | SW_Q4_UNATT = uplit (%asciz'%NZ RUNNING IN UN-ATTENDED REFORMAT MODE');
1987      |
1988      | end
1989      |
1990      | eludom

```

.TITLE CZRCH2 CZRCH RC25 DISK FORMATTER
.IDENT /REV A /

Address	Offset	Length	Value	Field	Encoding	Value
000000				.PSECT	AASCODE, RO	
000000	103	132		LSNAME::	.ASCII /CZR/	
000003	103	110	040		.ASCII /CH /	
000006	000				.BYTE 0	
000007	000				.BYTE 0	
000010				LSREV::		
000010	101				.ASCII /A/	
000011	060				.ASCII /O/	
000012	000000G			LSUNIT::	.WORD T\$PTHV	
000014	002260			L\$TIML::	.WORD 2260	
000016	000000G			L\$HPCP::	.WORD L\$HARD	
000020	000000G			L\$SPCP::	.WORD L\$SOFT	
000022	000140'			L\$HPTP::	.WORD L\$HW	
000024	000154'			L\$SPTP::	.WORD L\$SW	

000026	000000G	L\$LADP::	.WORD	L\$LAST
000030	000000	L\$STA::	.WORD	0
000032	000000	L\$SCO::	.WORD	0
000034	000000	L\$DTYP::	.WORD	0
000036	000000	L\$APT::	.WORD	0
000040	000124'	L\$DTP::	.WORD	L\$DISPATCH
000042	000000	L\$PRIO::	.WORD	0
000044	0000C3	L\$ENVI::	.WORD	0
000046	000000	L\$EXP1::	.WORD	0
000050		L\$MREV::		
000050	003		.BYTE	3
000051	003		.BYTE	3
000052	000000	L\$EF::	.WORD	0
000054	000000		.WORD	0
000056	000000	L\$SPC::	.WORD	0
000060	000000G	L\$DEVP::	.WORD	L\$DVTYP
000062	000000G	L\$REPP::	.WORD	L\$RPT
000064	000000	L\$EXP4::	.WORD	0
000066	000000	L\$EXP5::	.WORD	0
000070	000000G	L\$AUT::	.WORD	L\$AU
000072	000000G	L\$DUT::	.WORD	L\$DU
000074	000000	L\$LUN::	.WORD	0
000076	000000G	L\$DESP::	.WORD	L\$DESC
000100	104035	L\$LOAD::	.WORD	-73743
000102	000126'	L\$ETP::	.WORD	L\$ERRTBL
000104	000000G	L\$ICP::	.WORD	L\$INIT
000106	000000G	L\$CCP::	.WORD	L\$CLEAN
000110	000000G	L\$ACP::	.WORD	L\$AUTO
000112	000160'	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	000000C	L\$HWLEN::		
			.WORD	<<L\$NDHW-L\$HWLEN>/2>
000140	172150	HW.IP.ADRS::		
			.WORD	-5630
000142	000154	HW.VECTOR::		
			.WORD	154
000144	000005	HW.BR.LEVEL::		
			.WORD	5
000146	000000	HW.UNIT.NO::		
			.WORD	0
000150		L\$NDHW::	.BLKW	1
000152	000000C	L\$SWLEN::		
			.WORD	<<L\$NDSW-L\$SWLEN>/2>
000154	000001	SW.UNATT::		
			.WORD	1
000156		L\$NDSW::	.BLKW	1
000160	177777	L\$PROT::	.WORD	-1
000162	177777		.WORD	-1

CZRCH2
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH2.B16;2 (13)

000164 177777

.WORD -1

000000					.PSECT SPLITS, RO, D, GBL
000000	045	116	045	P.AAA:	.ASCII /%N%/
000003	124	000	000		.ASCII /T/<00><00>
000006	045	116	045	P.AAB:	.ASCII /%N%/
000011	116	045	101		.ASCII /%A%/
000014	111	116	111		.ASCII /INI/
000017	124	111	101		.ASCII /TIA/
000022	114	111	132		.ASCII /LIZ/
000025	111	116	107		.ASCII /ING/
000030	040	122	103		.ASCII /RC/
000033	062	065	040		.ASCII /25 /
000036	103	117	116		.ASCII /CON/
000041	124	122	117		.ASCII /TRO/
000044	114	114	105		.ASCII /LLE/
000047	122	040	101		.ASCII /R A/
000052	104	104	122		.ASCII /DDR/
000055	105	123	123		.ASCII /ESS/
000060	072	045	123		.ASCII /:XS/
000063	045	117	066		.ASCII /%06/
000066	045	101	050		.ASCII /%A(/
000071	117	051	045		.ASCII /O)%/
000074	116	000			.ASCII /N/<00>
000076	045	116	045	P.AAC:	.ASCII /%N%/
000101	101	114	117		.ASCII /ALO/
000104	107	111	103		.ASCII /GIC/
000107	101	114	040		.ASCII /AL /
000112	125	116	111		.ASCII /UNI/
000115	124	040	045		.ASCII /T %/
000120	104	062	045		.ASCII /D2%/
000123	101	050	104		.ASCII /A(D/
000126	051	045	123		.ASCII /)%S/
000131	045	101	120		.ASCII /%AP/
000134	110	131	123		.ASCII /HYS/
000137	111	103	101		.ASCII /ICA/
000142	114	040	125		.ASCII /L U/
000145	116	111	124		.ASCII /NIT/
000150	040	045	104		.ASCII / %D/
000153	062	045	101		.ASCII /2%A/
000156	050	104	051		.ASCII /(D)/
000161	045	123	045		.ASCII /%S%/
000164	101	106	117		.ASCII /AFO/
000167	122	115	101		.ASCII /RMA/
000172	124	040	101		.ASCII /T A/
000175	102	117	122		.ASCII /BOR/
000200	124	105	104		.ASCII /TED/
000203	0f 0				.ASCII <00>
000204	045	116	045	P.AAD:	.ASCII /%N%/
000207	101	115	111		.ASCII /AMI/
000212	103	122	117		.ASCII /CRO/
000215	040	103	117		.ASCII / CO/
000220	104	105	123		.ASCII /DES/
000223	040	115	117		.ASCII / MO/
000226	104	040	061		.ASCII /D 1/

CZRCH2
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH2.B16;2 (13)

000231	066	040	126	.ASCII	/6 V/
000234	105	122	123	.ASCII	/ERS/
000237	111	117	116	.ASCII	/ION/
000242	040	116	125	.ASCII	/ NU/
000245	115	102	105	.ASCII	/MBE/
000250	122	040	075	.ASCII	/R =/
000253	045	123	045	.ASCII	/XS%/
000256	117	062	045	.ASCII	/O2%/
000261	101	050	117	.ASCII	/A(O/
000264	051	045	116	.ASCII	/)N/
000267	000			.ASCII	<00>
000270	045	116	045	P.AAE: .ASCII	/XN%/
000273	101	102	122	.ASCII	/ABR/
000276	111	116	107	.ASCII	/ING/
000301	111	116	107	.ASCII	/ING/
000304	040	117	116	.ASCII	/ ON/
000307	114	111	116	.ASCII	/LIN/
000312	105	072	040	.ASCII	/E: /
000315	114	117	107	.ASCII	/LOG/
000320	111	103	101	.ASCII	/ICA/
000323	114	040	125	.ASCII	/L U/
000326	116	111	124	.ASCII	/NIT/
000331	040	045	104	.ASCII	/ XD/
000334	062	045	101	.ASCII	/2XA/
000337	050	104	051	.ASCII	/(D)/
000342	045	123	045	.ASCII	/XS%/
000345	101	120	110	.ASCII	/APH/
000350	131	123	111	.ASCII	/YSI/
000353	103	101	114	.ASCII	/CAL/
000356	040	125	116	.ASCII	/ UN/
000361	111	124	040	.ASCII	/IT /
000364	045	104	062	.ASCII	/XD2/
000367	045	101	050	.ASCII	/XA(/
000372	104	051	045	.ASCII	/D)X/
000375	116	000	000	.ASCII	/N/<00><00>
000400	045	116	045	P.AAF: .ASCII	/XN%/
000403	101	044	106	.ASCII	/ASF/
000406	124	114	105	.ASCII	/TLE/
000411	122	122	055	.ASCII	/RR-/
000414	040	116	117	.ASCII	/ NO/
000417	116	055	105	.ASCII	/N-E/
000422	130	111	123	.ASCII	/XIS/
000425	124	105	116	.ASCII	/TEN/
000430	124	040	122	.ASCII	/T R/
000433	103	062	065	.ASCII	/C25/
000436	040	103	117	.ASCII	/ CO/
000441	116	124	122	.ASCII	/NTR/
000444	117	114	114	.ASCII	/OLL/
000447	105	122	040	.ASCII	/ER /
000452	101	104	104	.ASCII	/ADD/
000455	122	105	123	.ASCII	/RES/
000460	123	040	045	.ASCII	/S %/
000463	117	066	045	.ASCII	/O6%/
000466	101	050	117	.ASCII	/A(O/
000471	051	045	116	.ASCII	/)N/
000474	000	000		.ASCII	<00><00>
000476	045	101	106	P.AAG: .ASCII	/XAF/

CZRCH2
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH2.B16;2 (13)

000501	117	122	115	.ASCII	/ORM/
000504	101	124	124	.ASCII	/ATT/
000507	111	116	107	.ASCII	/ING/
000512	040	120	110	.ASCII	/ PH/
000515	131	123	111	.ASCII	/YSI/
000520	103	101	114	.ASCII	/CAL/
000523	040	125	116	.ASCII	/ UN/
000526	111	124	040	.ASCII	/IT /
000531	045	104	062	.ASCII	/XD2/
000534	045	101	050	.ASCII	/XA(/
000537	104	051	045	.ASCII	/D)%/
000542	116	000		.ASCII	/N/<00>
000544	045	116	045	P.AAH:	.ASCII /%N%/
000547	101	106	117	.ASCII	/AFO/
000552	122	115	101	.ASCII	/RMA/
000555	124	124	105	.ASCII	/TTE/
000560	122	040	120	.ASCII	/R P/
000563	122	117	107	.ASCII	/ROG/
000566	122	105	123	.ASCII	/RES/
000571	123	111	116	.ASCII	/SIN/
000574	107	072	040	.ASCII	/G: /
000577	040	120	111	.ASCII	/ PI/
000602	104	040	110	.ASCII	/D H/
000605	111	075	045	.ASCII	/I=%/
000610	117	066	045	.ASCII	/O6%/
000613	101	050	117	.ASCII	/A(O/
000616	051	045	123	.ASCII	/)%S/
000621	045	123	045	.ASCII	/%S%/
000624	101	120	111	.ASCII	/API/
000627	104	040	114	.ASCII	/D L/
000632	117	075	045	.ASCII	/O=%/
000635	117	066	045	.ASCII	/O6%/
000640	101	050	117	.ASCII	/A(O/
000643	051	000	000	.ASCII	/)/<00><00>
000646	045	116	000	P.AAI:	.ASCII /%N/<00>
000651	000			.ASCII	<00>
000652	045	116	045	P.AAJ:	.ASCII /%N%/
000655	116	045	116	.ASCII	/N%N/
000660	045	116	045	.ASCII	/%N%/
000663	116	045	116	.ASCII	/N%N/
000666	045	116	045	.ASCII	/%N%/
000671	116	000	000	.ASCII	/N/<00><00>
000674	045	116	045	P.AAL:	.ASCII /%N%/
000677	101	044	106	.ASCII	/ASF/
000702	124	114	105	.ASCII	/TLE/
000705	122	122	055	.ASCII	/RR-/
000710	040	125	116	.ASCII	/ UN/
000713	122	105	103	.ASCII	/REC/
000716	117	107	116	.ASCII	/OGN/
000721	111	132	101	.ASCII	/IZA/
000724	102	114	105	.ASCII	/BLE/
000727	040	105	122	.ASCII	/ ER/
000732	122	117	122	.ASCII	/ROR/
000735	040	103	117	.ASCII	/ CO/
000740	104	105	000	.ASCII	/DE/<00>
000743	000			.ASCII	<00>
000744	045	116	045	P.AAM:	.ASCII /%N%/

CZRCH2
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH2.B16;2 (13)

000747	101	044	106	.ASCII	/ASF/
000752	124	114	105	.ASCII	/TLE/
000755	122	122	055	.ASCII	/RR-/
000760	040	105	116	.ASCII	/ EN/
000763	126	105	114	.ASCII	/VEL/
000766	117	120	105	.ASCII	/OPE/
000771	057	120	101	.ASCII	<57>/PA/
000774	103	113	105	.ASCII	/CKE/
000777	124	040	122	.ASCII	/T R/
001002	105	101	104	.ASCII	/EAD/
001005	040	050	120	.ASCII	/ (P/
001010	101	122	111	.ASCII	/ARI/
001013	124	131	040	.ASCII	/TY /
001016	117	122	040	.ASCII	/OR /
001021	124	111	115	.ASCII	/TIM/
001024	105	117	125	.ASCII	/EQU/
001027	124	051	000	.ASCII	/T)/<00>
001032	045	116	045	P.AAN: .ASCII	/XN%/
001035	101	044	106	.ASCII	/ASF/
001040	124	114	105	.ASCII	/TLE/
001043	122	122	055	.ASCII	/RR-/
001046	040	105	116	.ASCII	/ EN/
001051	126	105	114	.ASCII	/VEL/
001054	117	120	105	.ASCII	/OPE/
001057	057	120	101	.ASCII	<57>/PA/
001062	103	113	105	.ASCII	/CKE/
001065	124	040	127	.ASCII	/T W/
001070	122	111	124	.ASCII	/RIT/
001073	105	040	050	.ASCII	/E (/
001076	120	101	122	.ASCII	/PAR/
001101	111	124	131	.ASCII	/ITY/
001104	040	117	122	.ASCII	/ OR/
001107	040	124	111	.ASCII	/ TI/
001112	115	105	117	.ASCII	/MEO/
001115	125	124	051	.ASCII	/UT)/
001120	000	000		.ASCII	<00><00>
001122	045	116	045	P.AAO: .ASCII	/XN%/
001125	101	044	106	.ASCII	/ASF/
001130	124	114	105	.ASCII	/TLE/
001133	122	122	055	.ASCII	/RR-/
001136	040	103	117	.ASCII	/ CO/
001141	116	124	122	.ASCII	/NTR/
001144	117	114	114	.ASCII	/OLL/
001147	105	122	040	.ASCII	/ER /
001152	122	117	115	.ASCII	/ROM/
001155	040	101	116	.ASCII	/ AN/
001160	104	040	122	.ASCII	/D R/
001163	101	115	040	.ASCII	/AM /
001166	120	101	122	.ASCII	/PAR/
001171	111	124	131	.ASCII	/ITY/
001174	000	000		.ASCII	<00><00>
001176	045	116	045	P.AAP: .ASCII	/XN%/
001201	101	044	106	.ASCII	/ASF/
001204	124	114	105	.ASCII	/TLE/
001207	122	122	055	.ASCII	/RR-/
001212	040	103	117	.ASCII	/ CO/
001215	116	124	122	.ASCII	/NTR/

CZRCH2
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH2.B16;2 (13)

001220	117	114	114	.ASCII	/OLL/
001223	105	122	040	.ASCII	/ER /
001226	122	101	115	.ASCII	/RAM/
001231	040	120	101	.ASCII	/ PA/
001234	122	111	124	.ASCII	/RIT/
001237	131	000	000	.ASCII	/Y/<00><00>
001242	045	116	045	P.AAQ: .ASCII	/XN%/
001245	101	044	106	.ASCII	/ASF/
001250	124	114	105	.ASCII	/TLE/
001253	122	122	055	.ASCII	/RR-/
001256	040	103	117	.ASCII	/ CO/
001261	116	124	122	.ASCII	/NTR/
001264	117	114	114	.ASCII	/OLL/
001267	105	122	040	.ASCII	/ER /
001272	122	117	115	.ASCII	/ROM/
001275	040	120	101	.ASCII	/ PA/
001300	122	111	124	.ASCII	/RIT/
001303	131	000	000	.ASCII	/Y/<00><00>
001306	045	116	045	P.AAR: .ASCII	/XN%/
001311	101	044	106	.ASCII	/ASF/
001314	124	114	105	.ASCII	/TLE/
001317	122	122	055	.ASCII	/RR-/
001322	040	122	111	.ASCII	/ RI/
001325	116	107	040	.ASCII	/NG /
001330	122	105	101	.ASCII	/REA/
001333	104	040	050	.ASCII	/D (/
001336	120	101	122	.ASCII	/PAR/
001341	111	124	131	.ASCII	/ITY/
001344	040	117	122	.ASCII	/ OR/
001347	040	124	111	.ASCII	/ TI/
001352	115	105	117	.ASCII	/MEO/
001355	125	124	051	.ASCII	/UT)/
001360	000	000		.ASCII	<00><00>
001362	045	116	045	P.AAS: .ASCII	/XN%/
001365	101	044	106	.ASCII	/ASF/
001370	124	114	105	.ASCII	/TLE/
001373	122	122	055	.ASCII	/RR-/
001376	040	122	111	.ASCII	/ RI/
001401	116	107	040	.ASCII	/NG /
001404	127	122	111	.ASCII	/WRI/
001407	124	105	040	.ASCII	/TE /
001412	050	120	101	.ASCII	/(PA/
001415	122	111	124	.ASCII	/RIT/
001420	131	040	117	.ASCII	/Y O/
001423	122	040	124	.ASCII	/R T/
001426	111	115	105	.ASCII	/IME/
001431	117	125	124	.ASCII	/OUT/
001434	051	000		.ASCII	/)/<00>
001436	045	116	045	P.AAT: .ASCII	/XN%/
001441	101	044	106	.ASCII	/ASF/
001444	124	114	105	.ASCII	/TLE/
001447	122	122	055	.ASCII	/RR-/
001452	040	111	116	.ASCII	/ IN/
001455	124	105	122	.ASCII	/TER/
001460	122	125	120	.ASCII	/RUP/
001463	124	040	115	.ASCII	/T M/
001466	101	123	124	.ASCII	/AST/

CZRCH2
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH2.B16;2 (13) SEQ 88
Page 23

001471	105	122	000		.ASCII	/ER/<00>
001474	045	116	045	P.AAU:	.ASCII	/XN%/
001477	101	044	106		.ASCII	/ASF/
001502	124	114	105		.ASCII	/TLE/
001505	122	122	055		.ASCII	/RR-/
001510	040	110	117		.ASCII	/ HO/
001513	123	124	040		.ASCII	/ST /
001516	101	103	103		.ASCII	/ACC/
001521	105	123	123		.ASCII	/ESS/
001524	040	124	111		.ASCII	/ TI/
001527	115	105	117		.ASCII	/MEO/
001532	125	124	000		.ASCII	/UT/<00>
001535	000				.ASCII	<00>
001536	045	116	045	P.AAV:	.ASCII	/XN%/
001541	101	044	106		.ASCII	/ASF/
001544	124	114	105		.ASCII	/TLE/
001547	122	122	055		.ASCII	/RR-/
001552	040	103	122		.ASCII	/ CR/
001555	105	104	111		.ASCII	/EDI/
001560	124	040	114		.ASCII	/T L/
001563	111	115	111		.ASCII	/IMI/
001566	124	040	105		.ASCII	/T E/
001571	130	103	105		.ASCII	/XCE/
001574	105	104	105		.ASCII	/EDE/
001577	104	000	000		.ASCII	/D/<00><00>
001602	045	116	045	P.AAW:	.ASCII	/XN%/
001605	101	044	106		.ASCII	/ASF/
001610	124	114	105		.ASCII	/TLE/
001613	122	122	055		.ASCII	/RR-/
001616	040	102	125		.ASCII	/ BU/
001621	123	040	115		.ASCII	/S M/
001624	101	123	124		.ASCII	/AST/
001627	105	122	040		.ASCII	/ER /
001632	105	122	122		.ASCII	/ERR/
001635	117	122	000		.ASCII	/OR/<00>
001640	045	116	045	P.AAX:	.ASCII	/XN%/
001643	101	044	106		.ASCII	/ASF/
001646	124	114	105		.ASCII	/TLE/
001651	122	122	055		.ASCII	/RR-/
001654	040	104	111		.ASCII	/ DI/
001657	101	107	116		.ASCII	/AGN/
001662	117	123	124		.ASCII	/OST/
001665	111	103	040		.ASCII	/IC /
001670	103	117	116		.ASCII	/CON/
001673	124	122	117		.ASCII	/TRO/
001676	114	114	105		.ASCII	/LLE/
001701	122	040	106		.ASCII	/R F/
001704	101	124	101		.ASCII	/ATA/
001707	114	040	105		.ASCII	/L E/
001712	122	122	117		.ASCII	/RRO/
001715	122	000	000		.ASCII	/R/<00><00>
001720	045	116	045	P.AAY:	.ASCII	/XN%/
001723	101	044	106		.ASCII	/ASF/
001726	124	114	105		.ASCII	/TLE/
001731	122	122	055		.ASCII	/RR-/
001734	040	111	116		.ASCII	/ IN/
001737	123	124	122		.ASCII	/STR/

CZRCH2
REV A PATCH 00
CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

001742	125	103	124	.ASCII	/UCT/
001745	111	117	116	.ASCII	/ION/
001750	040	114	117	.ASCII	/ LO/
001753	117	120	040	.ASCII	/OP /
001756	124	111	115	.ASCII	/TIM/
001761	105	117	125	.ASCII	/EQU/
001764	124	000		.ASCII	/T/<00>
001766	045	116	045	P.AAZ:	.ASCII /%N%/
001771	101	044	106	.ASCII	/ASF/
001774	124	114	105	.ASCII	/TLE/
001777	122	122	055	.ASCII	/RR-/
002002	040	111	116	.ASCII	/ IN/
002005	126	101	114	.ASCII	/VAL/
002010	111	104	040	.ASCII	/ID /
002013	103	117	116	.ASCII	/CON/
002016	116	105	103	.ASCII	/NEC/
002021	124	111	117	.ASCII	/TIO/
002024	116	040	111	.ASCII	/N I/
002027	104	105	116	.ASCII	/DEN/
002032	124	111	106	.ASCII	/TIF/
002035	111	105	122	.ASCII	/IER/
002040	000	000		.ASCII	<00><00>
002042	045	116	045	P.ABA:	.ASCII /%N%/
002045	101	044	106	.ASCII	/ASF/
002050	124	114	105	.ASCII	/TLE/
002053	122	122	055	.ASCII	/RR-/
002056	040	111	116	.ASCII	/ IN/
002061	124	105	122	.ASCII	/TER/
002064	122	125	120	.ASCII	/RUP/
002067	124	040	127	.ASCII	/T W/
002072	122	111	124	.ASCII	/RIT/
002075	105	000	000	.ASCII	/E/<00><00>
002100	045	116	045	P.ABB:	.ASCII /%N%/
002103	101	044	106	.ASCII	/ASF/
002106	124	114	105	.ASCII	/TLE/
002111	122	122	055	.ASCII	/RR-/
002114	040	115	101	.ASCII	/ MA/
002117	111	116	124	.ASCII	/INT/
002122	105	116	101	.ASCII	/ENA/
002125	116	103	105	.ASCII	/NCE/
002130	040	122	105	.ASCII	/ RE/
002133	101	104	057	.ASCII	/AD/<57>
002136	127	122	111	.ASCII	/WRI/
002141	124	105	040	.ASCII	/TE /
002144	111	116	126	.ASCII	/INV/
002147	101	114	111	.ASCII	/ALI/
002152	104	040	122	.ASCII	/D R/
002155	105	107	111	.ASCII	/EGI/
002160	117	116	040	.ASCII	/ON /
002163	111	104	105	.ASCII	/IDE/
002166	116	124	111	.ASCII	/NTI/
002171	106	111	105	.ASCII	/FIE/
002174	122	000		.ASCII	/R/<00>
002176	045	116	045	P.ABC:	.ASCII /%N%/
002201	101	044	106	.ASCII	/ASF/
002204	124	114	105	.ASCII	/TLE/
002207	122	122	055	.ASCII	/RR-/

CZRCH2
REV A PATCH 00 CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

002212	040	115	101	.ASCII	/ MA/
002215	111	116	124	.ASCII	/INT/
002220	105	116	101	.ASCII	/ENA/
002223	116	103	105	.ASCII	/NCE/
002226	040	127	122	.ASCII	/ WR/
002231	111	124	105	.ASCII	/ITE/
002234	040	114	117	.ASCII	/ LO/
002237	101	104	040	.ASCII	/AD /
002242	124	117	040	.ASCII	/TO /
002245	116	117	116	.ASCII	/NON/
002250	055	114	117	.ASCII	/-LO/
002253	101	104	101	.ASCII	/ADA/
002256	102	114	105	.ASCII	/BLE/
002261	040	103	117	.ASCII	/ CO/
002264	116	124	122	.ASCII	/NTR/
002267	117	114	114	.ASCII	/OLL/
002272	105	122	000	.ASCII	/ER/<00>
002275	000			.ASCII	<00>
002276	045	116	045	P.ABD: .ASCII	/XN%/
002301	101	044	106	.ASCII	/ASF/
002304	124	114	105	.ASCII	/TLE/
002307	122	122	055	.ASCII	/RR-/
002312	040	103	117	.ASCII	/ CO/
002315	116	124	122	.ASCII	/NTR/
002320	117	114	114	.ASCII	/OLL/
002323	105	122	040	.ASCII	/ER /
002326	122	101	115	.ASCII	/RAM/
002331	040	105	122	.ASCII	/ ER/
002334	122	117	122	.ASCII	/ROR/
002337	040	050	116	.ASCII	/ (N/
002342	117	116	055	.ASCII	/ON-/
002345	120	101	122	.ASCII	/PAR/
002350	111	124	131	.ASCII	/ITY/
002353	051	000	000	.ASCII	/)/<00><00>
002356	045	116	045	P.ABE: .ASCII	/XN%/
002361	101	044	106	.ASCII	/ASF/
002364	124	114	105	.ASCII	/TLE/
002367	122	122	055	.ASCII	/RR-/
002372	040	111	116	.ASCII	/ IN/
002375	111	124	040	.ASCII	/IT /
002400	123	105	121	.ASCII	/SEQ/
002403	125	105	116	.ASCII	/UEN/
002406	103	105	040	.ASCII	/CE /
002411	105	122	122	.ASCII	/ERR/
002414	117	122	000	.ASCII	/OR/<00>
002417	000			.ASCII	<00>
002420	045	116	045	P.ABF: .ASCII	/XN%/
002423	101	044	106	.ASCII	/ASF/
002426	124	114	105	.ASCII	/TLE/
002431	122	122	055	.ASCII	/RR-/
002434	040	110	111	.ASCII	/ HI/
002437	107	110	040	.ASCII	/GH /
002442	114	105	126	.ASCII	/LEV/
002445	105	114	040	.ASCII	/EL /
002450	120	122	117	.ASCII	/PRO/
002453	124	117	103	.ASCII	/TOC/
002456	117	114	040	.ASCII	/OL /

CZRCH2
REV A PATCH 00
CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

002461	111	116	103	.ASCII	/INC/
002464	117	115	120	.ASCII	/OMP/
002467	101	124	111	.ASCII	/ATI/
002472	102	111	114	.ASCII	/BIL/
002475	111	124	131	.ASCII	/ITY/
002500	040	105	122	.ASCII	/ ER/
002503	122	117	122	.ASCII	/ROR/
002506	000	000		.ASCII	<00><00>
002510	045	116	045	P.ABG: .ASCII	/XN%/
002513	101	044	106	.ASCII	/ASF/
002516	124	114	105	.ASCII	/TLE/
002521	122	122	055	.ASCII	/RR-/
002524	040	120	125	.ASCII	/ PU/
002527	122	107	105	.ASCII	/RGE/
002532	057	120	117	.ASCII	<57>/PO/
002535	114	114	040	.ASCII	/LL /
002540	110	101	122	.ASCII	/HAR/
002543	104	127	101	.ASCII	/DWA/
002546	122	105	040	.ASCII	/RE /
002551	106	101	111	.ASCII	/FAI/
002554	114	125	122	.ASCII	/LUR/
002557	105	040	000	P.ABH: .ASCII	/E /<00>
002562	045	116	045	.ASCII	/XN%/
002565	101	044	106	.ASCII	/ASF/
002570	124	114	105	.ASCII	/TLE/
002573	122	122	055	.ASCII	/RR-/
002576	040	115	101	.ASCII	/ MA/
002601	120	120	111	.ASCII	/PPI/
002604	116	107	040	.ASCII	/NG /
002607	122	105	107	.ASCII	/REG/
002612	111	123	124	.ASCII	/IST/
002615	105	122	040	.ASCII	/ER /
002620	122	105	101	.ASCII	/REA/
002623	104	040	105	.ASCII	/D E/
002626	122	122	117	.ASCII	/RRO/
002631	122	040	050	.ASCII	/R (/
002634	120	101	122	.ASCII	/PAR/
002637	111	124	131	.ASCII	/ITY/
002642	040	117	122	.ASCII	/ OR/
002645	040	124	111	.ASCII	/ TI/
002650	115	105	117	.ASCII	/MEO/
002653	125	124	051	.ASCII	/UT)/
002656	000	000		P.AAK: .ASCII	<00><00>
002660	000674'			.WORD	P.AAL
002662	000744'			.WORD	P.AAM
002664	001032'			.WORD	P.AAN
002666	001122'			.WORD	P.AAO
002670	001176'			.WORD	P.AAP
002672	001242'			.WORD	P.AAQ
002674	001306'			.WORD	P.AAR
002676	001362'			.WORD	P.AAS
002700	001436'			.WORD	P.AAT
002702	001474'			.WORD	P.AAU
002704	001536'			.WORD	P.AAV
002706	001602'			.WORD	P.AAW
002710	001640'			.WORD	P.AAX
002712	001720'			.WORD	P.AAY

CZRCH2
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH2.B16;2 (13)

002714	001766'				.WORD	P.AAZ
002716	002042'				.WORD	P.ABA
002720	002100'				.WORD	P.ABB
002722	002176'				.WORD	P.ABC
002724	002276'				.WORD	P.ABD
002726	002356'				.WORD	P.ABE
002730	002420'				.WORD	P.ABF
002732	002510'				.WORD	P.ABG
002734	002562'				.WORD	P.ABH
002736	045	116	045	P.ABJ:	.ASCII	/XN%/
002741	101	044	106		.ASCII	/ASF/
002744	124	114	105		.ASCII	/TLE/
002747	122	122	055		.ASCII	/RR-/
002752	040	122	105		.ASCII	/RE/
002755	123	120	117		.ASCII	/SPO/
002760	116	123	105		.ASCII	/NSE/
002763	040	123	124		.ASCII	/ST/
002766	101	124	125		.ASCII	/ATU/
002771	123	040	105		.ASCII	/S E/
002774	122	122	117		.ASCII	/RRO/
002777	122	072	045		.ASCII	/R:Z/
003002	123	000			.ASCII	/S/<00>
003004	045	116	045	P.ABK:	.ASCII	/XN%/
003007	101	044	106		.ASCII	/ASF/
003012	124	114	105		.ASCII	/TLE/
003015	122	122	055		.ASCII	/RR-/
003020	040	110	117		.ASCII	/HO/
003023	123	124	057		.ASCII	/ST/<57>
003026	103	117	116		.ASCII	/CON/
003031	124	122	117		.ASCII	/TRO/
003034	114	114	105		.ASCII	/LLE/
003037	122	040	117		.ASCII	/R O/
003042	125	124	040		.ASCII	/UT /
003045	117	106	040		.ASCII	/OF /
003050	123	105	121		.ASCII	/SEQ/
003053	000				.ASCII	<00>
003054	045	116	045	P.ABL:	.ASCII	/XN%/
003057	101	044	106		.ASCII	/ASF/
003062	124	114	105		.ASCII	/TLE/
003065	122	122	055		.ASCII	/RR-/
003070	040	122	105		.ASCII	/RE/
003073	115	117	124		.ASCII	/MOT/
003076	105	040	120		.ASCII	/E P/
003101	122	117	107		.ASCII	/ROG/
003104	040	116	117		.ASCII	/NO/
003107	124	040	122		.ASCII	/T R/
003112	125	116	116		.ASCII	/UNN/
003115	111	116	107		.ASCII	/ING/
003120	000	000			.ASCII	<00><00>
003122	045	116	045	P.ABM:	.ASCII	/XN%/
003125	101	044	106		.ASCII	/ASF/
003130	124	114	105		.ASCII	/TLE/
003133	122	122	055		.ASCII	/RR-/
003136	040	125	116		.ASCII	/UN/
003141	113	116	117		.ASCII	/KNO/
003144	127	116	040		.ASCII	/WN /
003147	122	105	124		.ASCII	/RET/

CZRCH2 CZRCH RC25 DISK FORMATTER
REV A PATCH 00 GLOBAL TEXT SECTION

003152	125	122	116	.ASCII	/URN/
003155	040	123	124	.ASCII	/ST/
003160	101	124	125	.ASCII	/ATU/
003163	123	040	103	.ASCII	/S C/
003166	117	104	105	.ASCII	/ODE/
003171	000			.ASCII	<00>
003172	045	116	045	P.ABN:	.ASCII /%N%/
003175	101	044	106		.ASCII /ASF/
003200	124	114	105		.ASCII /TLE/
003203	122	122	055		.ASCII /RR-/
003206	040	103	117		.ASCII /CO/
003211	115	040	101		.ASCII /M A/
003214	122	105	101		.ASCII /REA/
003217	040	111	116		.ASCII /IN/
003222	111	124	040		.ASCII /IT /
003225	105	122	122		.ASCII /ERR/
003230	117	122	000		.ASCII /OR/<00>
003233	000				.ASCII <00>
003234	045	116	045	P.ABO:	.ASCII /%N%/
003237	101	044	106		.ASCII /ASF/
003242	124	114	105		.ASCII /TLE/
003245	122	122	055		.ASCII /RR-/
003250	040	120	117		.ASCII /PO/
003253	122	124	057		.ASCII /RT/<57>
003256	110	117	123		.ASCII /HOS/
003261	124	040	123		.ASCII /T S/
003264	131	116	103		.ASCII /YNC/
003267	040	105	122		.ASCII /ER/
003272	122	117	122		.ASCII /ROR/
003275	000				.ASCII <00>
003276	045	116	045	P.ABP:	.ASCII /%N%/
003301	101	044	106		.ASCII /ASF/
003304	124	114	105		.ASCII /TLE/
003307	122	122	055		.ASCII /RR-/
003312	040	115	105		.ASCII /ME/
003315	123	123	101		.ASCII /SSA/
003320	107	105	040		.ASCII /GE /
003323	114	105	116		.ASCII /LEN/
003326	107	124	110		.ASCII /GTH/
003331	040	105	122		.ASCII /ER/
003334	122	117	122		.ASCII /ROR/
003337	000				.ASCII <00>
003340	045	116	045	P.ABQ:	.ASCII /%N%/
003343	101	044	106		.ASCII /ASF/
003346	124	114	105		.ASCII /TLE/
003351	122	122	055		.ASCII /RR-/
003354	040	125	116		.ASCII /UN/
003357	113	116	117		.ASCII /KNO/
003362	127	116	040		.ASCII /WN /
003365	105	116	104		.ASCII /END/
003370	103	117	104		.ASCII /COD/
003373	105	040	122		.ASCII /E R/
003376	105	103	105		.ASCII /ECE/
003401	111	126	105		.ASCII /IVE/
003404	104	000			.ASCII /D/<00>
003406	045	116	045	P.ABR:	.ASCII /%N%/
003411	101	044	106		.ASCII /ASF/

CZRCH2
REV A PATCH 00 CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

003414	124	114	105	.ASCII	/TLE/
003417	122	122	055	.ASCII	/RR-/
003422	040	101	104	.ASCII	/AD/
003425	101	120	124	.ASCII	/APT/
003430	117	122	040	.ASCII	/OR /
003433	120	125	122	.ASCII	/PUR/
003436	107	105	040	.ASCII	/GE /
003441	105	122	122	.ASCII	/ERR/
003444	117	122	000	.ASCII	/OR/<00>
003447	000			.ASCII	<00>
003450	045	116	045	P.ABS: .ASCII	/XN%/
003453	101	044	106	.ASCII	/ASF/
003456	124	114	105	.ASCII	/TLE/
003461	122	122	055	.ASCII	/RR-/
003464	040	125	116	.ASCII	/UN/
003467	113	116	117	.ASCII	/KNO/
003472	127	116	040	.ASCII	/WN /
003475	111	116	124	.ASCII	/INT/
003500	105	122	122	.ASCII	/ERR/
003503	125	120	124	.ASCII	/UPT/
003506	000	000		.ASCII	<00><00>
003510	045	116	045	P.ABT: .ASCII	/XN%/
003513	101	044	106	.ASCII	/ASF/
003516	124	114	105	.ASCII	/TLE/
003521	122	122	055	.ASCII	/RR-/
003524	040	111	116	.ASCII	/IN/
003527	111	124	040	.ASCII	/IT /
003532	123	105	121	.ASCII	/SEQ/
003535	040	123	124	.ASCII	/ST/
003540	105	120	040	.ASCII	/EP /
003543	124	111	115	.ASCII	/TIM/
003546	105	104	040	.ASCII	/ED /
003551	117	125	124	.ASCII	/OUT/
003554	000	000		.ASCII	<00><00>
003556	045	116	045	P.ABU: .ASCII	/XN%/
003561	101	044	106	.ASCII	/ASF/
003564	124	114	105	.ASCII	/TLE/
003567	122	122	055	.ASCII	/RR-/
003572	040	111	116	.ASCII	/IN/
003575	111	124	040	.ASCII	/IT /
003600	123	105	121	.ASCII	/SEQ/
003603	040	103	117	.ASCII	/CO/
003606	115	120	101	.ASCII	/MPA/
003611	122	105	040	.ASCII	/RE /
003614	105	122	122	.ASCII	/ERR/
003617	117	122	000	.ASCII	/OR/<00>
003622	045	116	045	P.ABV: .ASCII	/XN%/
003625	101	044	106	.ASCII	/ASF/
003630	124	114	105	.ASCII	/TLE/
003633	122	122	055	.ASCII	/RR-/
003636	040	125	116	.ASCII	/UN/
003641	105	130	120	.ASCII	/EXP/
003644	105	103	124	.ASCII	/ECT/
003647	105	104	040	.ASCII	/ED /
003652	101	124	124	.ASCII	/ATT/
003655	105	116	124	.ASCII	/ENT/
003660	111	117	116	.ASCII	/ION/

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH2.B16;2 (13)

CZRCH2
REV A PATCH CO
CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

003663	040	105	116	.ASCII	/ EN/
003666	104	040	115	.ASCII	/D M/
003671	105	123	123	.ASCII	/ESS/
003674	101	107	105	.ASCII	/AGE/
003677	040	122	105	.ASCII	/ RE/
003702	103	105	111	.ASCII	/CEI/
003705	126	105	104	.ASCII	/VED/
003710	000	000		.ASCII	<00><00>
003712	045	116	045	P.ABW:	.ASCII /%N%/
003715	101	044	106		.ASCII /ASF/
003720	124	114	105		.ASCII /TLE/
003723	122	122	055		.ASCII /RR-/
003726	040	125	116		.ASCII / UN/
003731	105	130	120		.ASCII /EXP/
003734	105	103	124		.ASCII /ECT/
003737	105	104	040		.ASCII /ED /
003742	103	117	115		.ASCII /COM/
003745	115	101	116		.ASCII /MAN/
003750	104	040	117		.ASCII /D O/
003753	120	103	117		.ASCII /PCO/
003756	104	105	040		.ASCII /DE /
003761	111	116	040		.ASCII /IN /
003764	105	116	104		.ASCII /END/
003767	040	115	105		.ASCII / ME/
003772	123	123	101		.ASCII /SSA/
003775	107	105	040		.ASCII /GE /
004000	122	105	103		.ASCII /REC/
004003	105	111	126		.ASCII /EIV/
004006	105	104	000		.ASCII /ED/<00>
004011	000				.ASCII <00>
004012	045	116	045	P.ABX:	.ASCII /%N%/
004015	101	044	106		.ASCII /ASF/
004020	124	114	105		.ASCII /TLE/
004023	122	122	055		.ASCII /RR-/
004026	040	125	116		.ASCII / UN/
004031	105	130	120		.ASCII /EXP/
004034	105	103	124		.ASCII /ECT/
004037	105	104	040		.ASCII /ED /
004042	123	105	122		.ASCII /SER/
004045	111	117	125		.ASCII /IOU/
004050	123	040	105		.ASCII /S E/
004053	130	103	105		.ASCII /XCE/
004056	120	124	111		.ASCII /PTI/
004061	117	116	040		.ASCII /ON /
004064	105	116	104		.ASCII /END/
004067	040	115	105		.ASCII / ME/
004072	123	123	101		.ASCII /SSA/
004075	107	105	040		.ASCII /GE /
004100	122	105	103		.ASCII /REC/
004103	105	111	126		.ASCII /EIV/
004106	105	104	000		.ASCII /ED/<00>
004111	000				.ASCII <00>
004112	045	116	045	P.ABY:	.ASCII /%N%/
004115	101	044	106		.ASCII /ASF/
004120	124	114	105		.ASCII /TLE/
004123	122	122	055		.ASCII /RR-/
004126	040	111	116		.ASCII / IN/

CZRCH2
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

VAX-11 Bliss-16 v3-555
SPIDER\$USERS:[NEALE.AZTEC]CZRCH2.B16;2 (13)

004131	126	101	114	.ASCII	/VAL/	
004134	111	104	040	.ASCII	/ID /	
004137	103	117	115	.ASCII	/COM/	
004142	115	101	116	.ASCII	/MAN/	
004145	104	040	105	.ASCII	/D E/	
004150	116	104	040	.ASCII	/ND /	
004153	115	105	123	.ASCII	/MES/	
004156	123	101	107	.ASCII	/SAG/	
004161	105	040	122	.ASCII	/E R/	
004164	105	103	105	.ASCII	/ECE/	
004167	111	126	105	.ASCII	/IVE/	
004172	104	000		.ASCII	/D/<00>	
004174	045	116	045	P.ABZ:	.ASCII	/XN%/
004177	101	044	106	.ASCII	/ASF/	
004202	124	114	105	.ASCII	/TLE/	
004205	122	122	055	.ASCII	/RR-/	
004210	040	125	116	.ASCII	/ UN/	
004213	113	116	117	.ASCII	/KNO/	
004216	127	116	040	.ASCII	/WN /	
004221	115	105	123	.ASCII	/MES/	
004224	123	101	107	.ASCII	/SAG/	
004227	105	040	124	.ASCII	/E T/	
004232	131	120	105	.ASCII	/YPE/	
004235	040	122	105	.ASCII	/ RE/	
004240	103	105	111	.ASCII	/CEI/	
004243	126	105	104	.ASCII	/VED/	
004246	000	000		.ASCII	<00><00>	
004250	045	116	045	P.ACA:	.ASCII	/XN%/
004253	101	044	106	.ASCII	/ASF/	
004256	124	114	105	.ASCII	/TLE/	
004261	122	122	055	.ASCII	/RR-/	
004264	040	117	125	.ASCII	/ OU/	
004267	124	123	124	.ASCII	/TST/	
004272	101	116	104	.ASCII	/AND/	
004275	111	116	107	.ASCII	/ING/	
004300	040	103	117	.ASCII	/ CO/	
004303	115	115	101	.ASCII	/MMA/	
004306	116	104	040	.ASCII	/ND /	
004311	102	125	106	.ASCII	/BUF/	
004314	106	105	122	.ASCII	/FER/	
004317	040	106	125	.ASCII	/ FU/	
004322	114	114	000	.ASCII	/LL/<00>	
004325	000			.ASCII	<00>	
004326	045	116	045	P.ACB:	.ASCII	/XN%/
004331	101	044	106	.ASCII	/ASF/	
004334	124	114	105	.ASCII	/TLE/	
004337	122	122	055	.ASCII	/RR-/	
004342	040	117	125	.ASCII	/ OU/	
004345	124	040	123	.ASCII	/T S/	
004350	124	101	116	.ASCII	/TAN/	
004353	104	111	116	.ASCII	/DIN/	
004356	107	040	103	.ASCII	/G C/	
004361	117	115	115	.ASCII	/OMM/	
004364	101	116	104	.ASCII	/AND/	
004367	040	102	125	.ASCII	/ BU/	
004372	106	106	105	.ASCII	/FFE/	
004375	122	040	117	.ASCII	/R O/	

CZRCH2
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH2.B16;2 (13)

004400	125	124	040	.ASCII	/UT /
004403	117	106	040	.ASCII	/OF /
004406	123	131	116	.ASCII	/SYN/
004411	103	040	105	.ASCII	/C E/
004414	122	122	117	.ASCII	/RRO/
004417	122	000	000	.ASCII	/R/<00><00>
004422	045	116	045	P.ACC: .ASCII	/XN%/
004425	101	044	106	.ASCII	/ASF/
004430	124	114	105	.ASCII	/TLE/
004433	122	122	055	.ASCII	/RR-/
004436	040	125	116	.ASCII	/ UN/
004441	113	116	117	.ASCII	/KNO/
004444	127	116	040	.ASCII	/WN /
004447	115	105	123	.ASCII	/MES/
004452	123	101	107	.ASCII	/SAG/
004455	105	040	116	.ASCII	/E N/
004460	125	115	102	.ASCII	/UMB/
004463	105	122	040	.ASCII	/ER /
004466	122	105	103	.ASCII	/REC/
004471	105	111	126	.ASCII	/EIV/
004474	105	104	000	.ASCII	/ED/<00>
004477	000			.ASCII	<00>
004500	045	116	045	P.ACD: .ASCII	/XN%/
004503	101	044	106	.ASCII	/ASF/
004506	124	114	105	.ASCII	/TLE/
004511	122	122	055	.ASCII	/RR-/
004514	040	106	111	.ASCII	/ FI/
004517	114	105	040	.ASCII	/LE /
004522	122	105	101	.ASCII	/REA/
004525	104	040	105	.ASCII	/D E/
004530	122	122	117	.ASCII	/RRO/
004533	122	000	000	.ASCII	/R/<00><00>
004536	045	116	045	P.ACE: .ASCII	/XN%/
004541	101	044	106	.ASCII	/ASF/
004544	124	114	105	.ASCII	/TLE/
004547	122	122	055	.ASCII	/RR-/
004552	040	120	117	.ASCII	/ PO/
004555	122	124	057	.ASCII	/RT/<57>
004560	103	117	116	.ASCII	/CON/
004563	124	122	117	.ASCII	/TRO/
004566	114	114	105	.ASCII	/LLE/
004571	122	040	124	.ASCII	/R T/
004574	111	115	105	.ASCII	/IME/
004577	117	125	124	.ASCII	/OUT/
004602	040	105	122	.ASCII	/ ER/
004605	122	117	122	.ASCII	/ROR/
004610	000	000		.ASCII	<00><00>
004612	045	116	045	P.ACF: .ASCII	/XN%/
004615	101	044	106	.ASCII	/ASF/
004620	124	114	105	.ASCII	/TLE/
004623	122	122	055	.ASCII	/RR-/
004626	040	111	114	.ASCII	/ IL/
004631	114	105	107	.ASCII	/LEG/
004634	101	114	040	.ASCII	/AL /
004637	106	103	124	.ASCII	/FCT/
004642	040	106	111	.ASCII	/ FI/
004645	114	105	040	.ASCII	/LE /

CZRCH2
REV A PATCH 00
CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

004650	114	105	116		.ASCII	/LEN/
004653	107	124	110		.ASCII	/GTH/
004656	000	000			.ASCII	<00><00>
004660	002736'			P.ABI:	.WORD	P.ABJ
004662	003004'				.WORD	P.ABK
004664	003054'				.WORD	P.ABL
004666	003122'				.WORD	P.ABM
004670	003172'				.WORD	P.ABN
004672	003234'				.WORD	P.ABO
004674	003276'				.WORD	P.ABP
004676	003340'				.WORD	P.ABQ
004700	003406'				.WORD	P.ABR
004702	003450'				.WORD	P.ABS
004704	003510'				.WORD	P.ABT
004706	003556'				.WORD	P.ABU
004710	003622'				.WORD	P.ABV
004712	003712'				.WORD	P.ABW
004714	004012'				.WORD	P.ABX
004716	004112'				.WORD	P.ABY
004720	004174'				.WORD	P.ABZ
004722	004250'				.WORD	P.ACA
004724	004326'				.WORD	P.ACB
004726	004422'				.WORD	P.ACC
004730	004500'				.WORD	P.ACD
004732	004536'				.WORD	P.ACE
004734	004612'				.WORD	P.ACF
004736	045	116	045	P.ACH:	.ASCII	/XNZ/
004741	101	044	106		.ASCII	/ASF/
004744	124	114	105		.ASCII	/TLE/
004747	122	122	055		.ASCII	/RR-/
004752	040	126	101		.ASCII	/VA/
004755	130	040	122		.ASCII	/X R/
004760	105	101	104		.ASCII	/EAD/
004763	057	127	122		.ASCII	<57>/WR/
004766	111	124	105		.ASCII	/ITE/
004771	040	105	122		.ASCII	/ER/
004774	122	117	122		.ASCII	/ROR/
004777	040	117	116		.ASCII	/ON/
005002	040	111	116		.ASCII	/IN/
005005	124	105	122		.ASCII	/TER/
005010	122	125	120		.ASCII	/RUP/
005013	124	000	000	P.ACI:	.ASCII	/T/<00><00>
005016	045	116	045		.ASCII	/XNZ/
005021	101	044	106		.ASCII	/ASF/
005024	124	114	105		.ASCII	/TLE/
005027	122	122	055		.ASCII	/RR-/
005032	040	111	116		.ASCII	/IN/
005035	103	117	116		.ASCII	/CON/
005040	123	111	123		.ASCII	/SIS/
005043	124	105	116		.ASCII	/TEN/
005046	103	131	040		.ASCII	/CY /
005051	101	124	040		.ASCII	/AT /
005054	125	056	102		.ASCII	/U.B/
005057	106	111	114		.ASCII	/FIL/
005062	000	000			.ASCII	<00><00>
005064	045	116	045	P.ACJ:	.ASCII	/XNZ/
005067	101	044	106		.ASCII	/ASF/

CZRCH2
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH2.B16;2 (13)

005072	124	114	105	.ASCII	/TLE/
005075	122	122	055	.ASCII	/RR-/
005100	040	111	116	.ASCII	/ IN/
005103	103	117	116	.ASCII	/CON/
005106	123	111	123	.ASCII	/SIS/
005111	124	105	116	.ASCII	/TEN/
005114	103	131	040	.ASCII	/CY /
005117	101	124	040	.ASCII	/AT /
005122	125	056	102	.ASCII	/U.B/
005125	115	124	131	.ASCII	/MTY/
005130	000	000		.ASCII	<00><00>
005132	045	116	045	P.ACK: .ASCII	/XNZ/
005135	101	044	106	.ASCII	/ASF/
005140	124	114	105	.ASCII	/TLE/
005143	122	122	055	.ASCII	/RR-/
005146	040	111	116	.ASCII	/ IN/
005151	103	117	116	.ASCII	/CON/
005154	123	111	123	.ASCII	/SIS/
005157	124	105	116	.ASCII	/TEN/
005162	103	131	040	.ASCII	/CY /
005165	101	124	040	.ASCII	/AT /
005170	125	056	101	.ASCII	/U.A/
005173	114	117	103	.ASCII	/LOC/
005176	000	000		.ASCII	<00><00>
005200	045	116	045	P.ACL: .ASCII	/XNZ/
005203	101	044	106	.ASCII	/ASF/
005206	124	114	105	.ASCII	/TLE/
005211	122	122	055	.ASCII	/RR-/
005214	040	111	116	.ASCII	/ IN/
005217	103	117	116	.ASCII	/CON/
005222	123	111	123	.ASCII	/SIS/
005225	124	105	116	.ASCII	/TEN/
005230	103	131	040	.ASCII	/CY /
005233	101	124	040	.ASCII	/AT /
005236	123	105	122	.ASCII	/SER/
005241	126	117	040	.ASCII	/VO /
005244	105	116	124	.ASCII	/ENT/
005247	127	131	040	.ASCII	/RY /
005252	050	120	111	.ASCII	/(PI/
005255	120	040	123	.ASCII	/P S/
005260	105	124	051	.ASCII	/ET)/
005263	000			.ASCII	<00>
005264	045	116	045	P.ACM: .ASCII	/XNZ/
005267	101	044	106	.ASCII	/ASF/
005272	124	114	105	.ASCII	/TLE/
005275	122	122	055	.ASCII	/RR-/
005300	040	111	116	.ASCII	/ IN/
005303	103	117	116	.ASCII	/CON/
005306	123	111	123	.ASCII	/SIS/
005311	124	105	116	.ASCII	/TEN/
005314	103	131	040	.ASCII	/CY /
005317	101	124	040	.ASCII	/AT /
005322	123	105	122	.ASCII	/SER/
005325	126	117	040	.ASCII	/VO /
005330	105	116	124	.ASCII	/ENT/
005333	122	131	040	.ASCII	/RY /
005336	050	105	122	.ASCII	/(ER/

CZRCH2
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH2.B16;2 (13)

005341	122	040	123	.ASCII	/R S/
005344	105	124	051	.ASCII	/ET)/
005347	000			.ASCII	<00>
005350	045	116	045	P.ACN:	.ASCII /XN%/
005353	101	044	106	.ASCII	/ASF/
005356	124	114	105	.ASCII	/TLE/
005361	122	122	055	.ASCII	/RR-/
005364	040	111	116	.ASCII	/ IN/
005367	103	117	116	.ASCII	/CON/
005372	123	111	123	.ASCII	/SIS/
005375	124	105	116	.ASCII	/TEN/
005400	103	131	040	.ASCII	/CY /
005403	101	124	040	.ASCII	/AT /
005406	125	056	123	.ASCII	/U.S/
005411	105	116	104	.ASCII	/END/
005414	000	000		.ASCII	<00><00>
005416	045	116	045	P.ACO:	.ASCII /XN%/
005421	101	044	106	.ASCII	/ASF/
005424	124	114	105	.ASCII	/TLE/
005427	122	122	055	.ASCII	/RR-/
005432	040	111	116	.ASCII	/ IN/
005435	103	117	116	.ASCII	/CON/
005440	123	111	123	.ASCII	/SIS/
005443	124	105	116	.ASCII	/TEN/
005446	103	131	040	.ASCII	/CY /
005451	101	124	040	.ASCII	/AT /
005454	125	056	122	.ASCII	/U.R/
005457	105	103	126	.ASCII	/ECV/
005462	000	000		.ASCII	<00><00>
005464	045	116	045	P.ACP:	.ASCII /XN%/
005467	101	044	106	.ASCII	/ASF/
005472	124	114	105	.ASCII	/TLE/
005475	122	122	055	.ASCII	/RR-/
005500	040	111	116	.ASCII	/ IN/
005503	103	117	116	.ASCII	/CON/
005506	123	111	123	.ASCII	/SIS/
005511	124	105	116	.ASCII	/TEN/
005514	103	131	040	.ASCII	/CY /
005517	101	124	040	.ASCII	/AT /
005522	125	056	101	.ASCII	/U.A/
005525	124	124	116	.ASCII	/TTN/
005530	000	000		.ASCII	<00><00>
005532	045	116	045	P.ACO:	.ASCII /XN%/
005535	101	044	106	.ASCII	/ASF/
005540	124	114	105	.ASCII	/TLE/
005543	122	122	055	.ASCII	/RR-/
005546	040	111	116	.ASCII	/ IN/
005551	103	117	116	.ASCII	/CON/
005554	123	111	123	.ASCII	/SIS/
005557	124	105	116	.ASCII	/TEN/
005562	103	131	040	.ASCII	/CY /
005565	101	124	040	.ASCII	/AT /
005570	125	056	117	.ASCII	/U.O/
005573	116	114	116	.ASCII	/NLN/
005576	000	000		.ASCII	<00><00>
005600	045	116	045	P.ACR:	.ASCII /XN%/
005603	101	044	106	.ASCII	/ASF/

CZRCH2
REV A PATCH 00

CZRCH 9C25 DISK FORMATTER
GLOBAL TEXT SECTION

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH2.B16;2 (13)

005606	124	114	105	.ASCII	/TLE/
005611	122	122	055	.ASCII	/RR-/
005614	040	111	114	.ASCII	/ IL/
005617	114	105	107	.ASCII	/LEG/
005622	101	114	040	.ASCII	/AL /
005625	104	040	122	.ASCII	/D R/
005630	105	121	125	.ASCII	/EQU/
005633	105	123	124	.ASCII	/EST/
005636	040	050	125	.ASCII	/ (U/
005641	056	121	104	.ASCII	/.QD/
005644	122	121	051	.ASCII	/RQ)/
005647	000			.ASCII	<00>
005650	045	116	045	P.ACS: .ASCII	/XNZ/
005653	101	044	106	.ASCII	/ASF/
005656	124	114	105	.ASCII	/TLE/
005661	122	122	055	.ASCII	/RR-/
005664	040	106	105	.ASCII	/ FE/
005667	116	103	105	.ASCII	/NCE/
005672	055	120	117	.ASCII	/-PO/
005675	123	124	040	.ASCII	/ST /
005700	105	122	122	.ASCII	/ERR/
005703	117	122	040	.ASCII	/OR /
005706	101	124	040	.ASCII	/AT /
005711	120	122	117	.ASCII	/PRO/
005714	124	101	102	.ASCII	/TAB/
005717	000			.ASCII	<00>
005720	045	116	045	P.ACT: .ASCII	/XNZ/
005723	101	044	106	.ASCII	/ASF/
005726	124	114	105	.ASCII	/TLE/
005731	122	122	055	.ASCII	/RR-/
005734	040	102	101	.ASCII	/ BA/
005737	104	040	120	.ASCII	/D P/
005742	101	103	113	.ASCII	/ACK/
005745	105	124	040	.ASCII	/ET /
005750	104	105	121	.ASCII	/DEQ/
005753	125	105	125	.ASCII	/UEU/
005756	105	104	040	.ASCII	/ED /
005761	101	124	040	.ASCII	/AT /
005764	125	056	104	.ASCII	/U.D/
005767	117	116	105	.ASCII	/ONE/
005772	000	000		.ASCII	<00><00>
005774	045	116	045	P.ACU: .ASCII	/XNZ/
005777	101	044	106	.ASCII	/ASF/
006002	124	114	105	.ASCII	/TLE/
006005	122	122	055	.ASCII	/RR-/
006010	040	125	116	.ASCII	/ UN/
006013	105	130	120	.ASCII	/EXP/
006016	114	101	111	.ASCII	/LAI/
006021	116	105	104	.ASCII	/NED/
006024	040	104	055	.ASCII	/ D-/
006027	120	122	117	.ASCII	/PRO/
006032	103	040	123	.ASCII	/C S/
006035	125	123	120	.ASCII	/USP/
006040	105	116	123	.ASCII	/ENS/
006043	111	117	116	.ASCII	/ION/
006046	040	050	125	.ASCII	/ (U/
006051	056	056	124	.ASCII	/. .T/

CZRCH2
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH2.B16;2 (13)

006054	104	123	051		.ASCII	/DS)/
006057	000				.ASCII	<00>
006060	045	116	045	P.ACW:	.ASCII	/XNZ/
006063	101	044	106		.ASCII	/ASF/
006066	124	114	105		.ASCII	/TLE/
006071	122	122	055		.ASCII	/RR-/
006074	040	104	125		.ASCII	/ DU/
006077	120	040	120		.ASCII	/P P/
006102	101	103	113		.ASCII	/ACK/
006105	105	124	040		.ASCII	/ET /
006110	104	055	121		.ASCII	/D-Q/
006113	040	106	101		.ASCII	/ FA/
006116	111	114	105		.ASCII	/ILE/
006121	104	040	050		.ASCII	/D (/
006124	130	106	103		.ASCII	/XFC/
006127	040	063	064		.ASCII	/ 34/
006132	057	063	065		.ASCII	<57>/35/
006135	051	000	000		.ASCII	/)/<00><00>
006140	045	116	045	P.ACW:	.ASCII	/XNZ/
006143	101	044	106		.ASCII	/ASF/
006146	124	114	105		.ASCII	/TLE/
006151	122	122	055		.ASCII	/RR-/
006154	040	111	116		.ASCII	/ IN/
006157	103	117	116		.ASCII	/CON/
006162	123	111	123		.ASCII	/SIS/
006165	124	105	116		.ASCII	/TEN/
006170	103	131	040		.ASCII	/CY /
006173	101	124	040		.ASCII	/AT /
006176	125	056	110		.ASCII	/U.H/
006201	124	123	124		.ASCII	/TST/
006204	000	000			.ASCII	<00><00>
006206	045	116	045	P.ACX:	.ASCII	/XNZ/
006211	101	044	106		.ASCII	/ASF/
006214	124	114	105		.ASCII	/TLE/
006217	122	122	055		.ASCII	/RR-/
006222	040	111	116		.ASCII	/ IN/
006225	103	117	116		.ASCII	/CON/
006230	123	111	123		.ASCII	/SIS/
006233	124	105	116		.ASCII	/TEN/
006236	103	131	040		.ASCII	/CY /
006241	101	124	040		.ASCII	/AT /
006244	125	056	123		.ASCII	/U.S/
006247	105	113	117		.ASCII	/EKO/
006252	000	000			.ASCII	<00><00>
006254	045	116	045	P.ACX:	.ASCII	/XNZ/
006257	101	044	106		.ASCII	/ASF/
006262	124	114	105		.ASCII	/TLE/
006265	122	122	055		.ASCII	/RR-/
006270	040	111	116		.ASCII	/ IN/
006273	103	117	116		.ASCII	/CON/
006276	123	111	123		.ASCII	/SIS/
006301	124	105	116		.ASCII	/TEN/
006304	103	131	040		.ASCII	/CY /
006307	101	124	040		.ASCII	/AT /
006312	125	056	103		.ASCII	/U.C/
006315	113	123	126		.ASCII	/KSV/
006320	000	000			.ASCII	<00><00>

CZRCH2
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH2.B16;2 (13)

006322	045	116	045	P.ACZ:	.ASCII	/XN%/
006325	101	044	106		.ASCII	/ASF/
006330	124	114	105		.ASCII	/TLE/
006333	122	122	055		.ASCII	/RR-/
006336	040	104	056		.ASCII	/ D./
006341	117	120	103		.ASCII	/OPC/
006344	104	040	106		.ASCII	/D F/
006347	117	125	116		.ASCII	/OUN/
006352	104	040	111		.ASCII	/D I/
006355	114	114	105		.ASCII	/LLE/
006360	107	101	114		.ASCII	/GAL/
006363	040	117	120		.ASCII	/ OP/
006366	103	117	104		.ASCII	/COD/
006371	105	000	000		.ASCII	/E/<00><00>
006374	045	116	045	P.ADA:	.ASCII	/XN%/
006377	101	044	106		.ASCII	/ASF/
006402	124	114	105		.ASCII	/TLE/
006405	122	122	055		.ASCII	/RR-/
006410	040	104	056		.ASCII	/ D./
006413	103	123	106		.ASCII	/CSF/
006416	040	106	117		.ASCII	/ FO/
006421	125	116	104		.ASCII	/UND/
006424	040	111	114		.ASCII	/ IL/
006427	114	105	107		.ASCII	/LEG/
006432	101	114	040		.ASCII	/AL /
006435	117	120	103		.ASCII	/OPC/
006440	117	104	105		.ASCII	/ODE/
006443	000				.ASCII	<00>
006444	045	116	045	P.ADB:	.ASCII	/XN%/
006447	101	044	106		.ASCII	/ASF/
006452	124	114	105		.ASCII	/TLE/
006455	122	122	055		.ASCII	/RR-/
006460	040	125	116		.ASCII	/ UN/
006463	113	116	117		.ASCII	/KNO/
006466	127	116	040		.ASCII	/WN /
006471	102	101	104		.ASCII	/BAD/
006474	040	104	122		.ASCII	/ DR/
006477	111	126	105		.ASCII	/IVE/
006502	040	123	124		.ASCII	/ ST/
006505	101	124	125		.ASCII	/ATU/
006510	123	040	101		.ASCII	/S A/
006513	124	040	104		.ASCII	/T D/
006516	056	104	123		.ASCII	/.DS/
006521	124	123	000		.ASCII	/TS/<00>
006524	045	116	045	P.ADC:	.ASCII	/XN%/
006527	101	044	106		.ASCII	/ASF/
006532	124	114	105		.ASCII	/TLE/
006535	122	122	055		.ASCII	/RR-/
006540	040	111	114		.ASCII	/ IL/
006543	114	105	107		.ASCII	/LEG/
006546	101	114	040		.ASCII	/AL /
006551	130	106	103		.ASCII	/XFC/
006554	040	105	130		.ASCII	/ EX/
006557	105	103	125		.ASCII	/ECU/
006562	124	105	104		.ASCII	/TED/
006565	040	102	131		.ASCII	/ BY/
006570	040	104	115		.ASCII	/ DM/

CZRCH2
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH2.B16;2 (13)

006573	000				.ASCII	<00>
006574	045	116	045	P.ADD:	.ASCII	/XN%/
006577	101	044	106		.ASCII	/ASF/
006602	124	114	105		.ASCII	/TLE/
006605	122	122	055		.ASCII	/RR-/
006610	040	104	040		.ASCII	/ D /
006613	120	111	103		.ASCII	/PIC/
006616	113	105	104		.ASCII	/KED/
006621	040	125	120		.ASCII	/ UP/
006624	040	101	040		.ASCII	/ A /
006627	132	105	122		.ASCII	/ZER/
006632	117	040	123		.ASCII	/O S/
006635	103	102	056		.ASCII	/CB./
006640	104	102	000		.ASCII	/DB/<00>
006643	000				.ASCII	<00>
006644	045	116	045	P.ADE:	.ASCII	/XN%/
006647	101	044	106		.ASCII	/ASF/
006652	124	114	105		.ASCII	/TLE/
006655	122	122	055		.ASCII	/RR-/
006660	040	111	116		.ASCII	/ IN/
006663	103	117	116		.ASCII	/CON/
006666	123	111	123		.ASCII	/SIS/
006671	124	105	116		.ASCII	/TEN/
006674	103	131	040		.ASCII	/CY /
006677	101	124	040		.ASCII	/AT /
006702	104	040	111		.ASCII	/D I/
006705	104	114	105		.ASCII	/DLE/
006710	040	114	117		.ASCII	/ LO/
006713	117	120	000		.ASCII	/OP/<00>
006716	045	116	045	P.ADF:	.ASCII	/XN%/
006721	101	044	106		.ASCII	/ASF/
006724	124	114	105		.ASCII	/TLE/
006727	122	122	055		.ASCII	/RR-/
006732	040	104	115		.ASCII	/ DM/
006735	040	127	117		.ASCII	/ WO/
006740	122	104	040		.ASCII	/RD /
006743	103	117	125		.ASCII	/COU/
006746	116	124	040		.ASCII	/NT /
006751	105	122	122		.ASCII	/ERR/
006754	117	122	040		.ASCII	/OR /
006757	117	116	040		.ASCII	/ON /
006762	110	117	123		.ASCII	/HOS/
006765	124	040	104		.ASCII	/T D/
006770	115	101	057		.ASCII	/MA/<57>
006773	123	105	116		.ASCII	/SEN/
006776	104	057	122		.ASCII	/D/<57>/R/
007001	105	103	126		.ASCII	/ECV/
007004	000	000			.ASCII	<00><00>
007006	045	116	045	P.ADG:	.ASCII	/XN%/
007011	101	044	106		.ASCII	/ASF/
007014	124	114	105		.ASCII	/TLE/
007017	122	122	055		.ASCII	/RR-/
007022	040	125	116		.ASCII	/ UN/
007025	113	116	117		.ASCII	/KNO/
007030	127	116	040		.ASCII	/WN /
007033	104	111	123		.ASCII	/DIS/
007036	120	114	101		.ASCII	/PLA/

CZRCH2
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

007041	131	040	106	.ASCII	/Y F/	
007044	101	125	114	.ASCII	/AUL/	
007047	124	040	103	.ASCII	/T C/	
007052	117	104	105	.ASCII	/ODE/	
007055	040	101	124	.ASCII	/ AT/	
007060	040	104	056	.ASCII	/ D./	
007063	104	106	114	.ASCII	/DFL/	
007066	124	000		.ASCII	/T/<00>	
007070	045	116	045	P.ADH:	.ASCII	/XN%/
007073	101	044	106	.ASCII	/ASF/	
007076	124	114	105	.ASCII	/TLE/	
007101	122	122	055	.ASCII	/RR-/	
007104	040	104	122	.ASCII	/ DR/	
007107	111	126	105	.ASCII	/IVE/	
007112	040	116	117	.ASCII	/ NO/	
007115	124	040	106	.ASCII	/T F/	
007120	101	125	114	.ASCII	/AUL/	
007123	124	111	116	.ASCII	/TIN/	
007126	107	040	111	.ASCII	/G I/	
007131	116	040	120	.ASCII	/N P/	
007134	056	117	106	.ASCII	/.OF/	
007137	114	116	040	.ASCII	/LN /	
007142	123	124	101	.ASCII	/STA/	
007145	124	105	000	.ASCII	/TE/<00>	
007150	045	116	045	P.ADI:	.ASCII	/XN%/
007153	101	044	106	.ASCII	/ASF/	
007156	124	114	105	.ASCII	/TLE/	
007161	122	122	055	.ASCII	/RR-/	
007164	040	125	040	.ASCII	/ U /	
007167	120	117	127	.ASCII	/POW/	
007172	105	122	040	.ASCII	/ER /	
007175	125	120	040	.ASCII	/UP /	
007200	104	111	101	.ASCII	/DIA/	
007203	107	116	117	.ASCII	/GNO/	
007206	123	124	111	.ASCII	/STI/	
007211	103	123	040	.ASCII	/CS /	
007214	106	101	111	.ASCII	/FAI/	
007217	114	105	104	.ASCII	/LED/	
007222	000	000		.ASCII	<00><00>	
007224	045	116	045	P.ADJ:	.ASCII	/XN%/
007227	101	044	106	.ASCII	/ASF/	
007232	124	114	105	.ASCII	/TLE/	
007235	122	122	055	.ASCII	/RR-/	
007240	040	104	040	.ASCII	/ D /	
007243	120	117	127	.ASCII	/POW/	
007246	105	122	040	.ASCII	/ER /	
007251	125	120	040	.ASCII	/UP /	
007254	104	111	101	.ASCII	/DIA/	
007257	107	116	117	.ASCII	/GNO/	
007262	123	124	111	.ASCII	/STI/	
007265	103	123	040	.ASCII	/CS /	
007270	106	101	111	.ASCII	/FAI/	
007273	114	105	104	.ASCII	/LED/	
007276	000	000		.ASCII	<00><00>	
007300	045	116	045	P.ADK:	.ASCII	/XN%/
007303	101	044	106	.ASCII	/ASF/	
007306	124	114	105	.ASCII	/TLE/	

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

VAX-11 Bliss-16 V3-555
SPIDERSUSERS:[NEALE.AZTEC]CZRCH2.B16;2 (13)

CZRCH2
REV A PATCH 00
CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

007311	122	122	055	.ASCII	/RR-/
007314	040	101	104	.ASCII	/ AD/
007317	101	120	124	.ASCII	/APT/
007322	105	122	040	.ASCII	/ER /
007325	103	101	122	.ASCII	/CAR/
007330	104	040	106	.ASCII	/D F/
007333	101	111	114	.ASCII	/AIL/
007336	125	122	105	.ASCII	/URE/
007341	000			.ASCII	<00>
007342	045	116	045	P.ADL:	.ASCII /%N%/
007345	101	044	106	.ASCII	/ASF/
007350	124	114	105	.ASCII	/TLE/
007353	122	122	055	.ASCII	/RR-/
007356	040	105	103	.ASCII	/ EC/
007361	056	124	115	.ASCII	/.TM/
007364	122	040	124	.ASCII	/R T/
007367	111	115	105	.ASCII	/IME/
007372	104	040	117	.ASCII	/D O/
007375	125	124	000	P.ADM:	.ASCII /UT/<00>
007400	045	116	045	.ASCII	/%N%/
007403	101	044	106	.ASCII	/ASF/
007406	124	114	105	.ASCII	/TLE/
007411	122	122	055	.ASCII	/RR-/
007414	040	125	056	.ASCII	/ U./
007417	123	105	116	.ASCII	/SEN/
007422	104	057	125	.ASCII	/D/<57>/U/
007425	056	122	105	.ASCII	/.RE/
007430	103	126	040	.ASCII	/CV /
007433	122	111	116	.ASCII	/RIN/
007436	107	040	122	.ASCII	/G R/
007441	105	101	104	.ASCII	/EAD/
007444	040	111	116	.ASCII	/ IN/
007447	103	117	116	.ASCII	/CON/
007452	123	111	123	.ASCII	/SIS/
007455	124	105	116	.ASCII	/TEN/
007460	103	131	000	P.ADN:	.ASCII /CY/<00>
007463	000			.ASCII	<00>
007464	045	116	045	.ASCII	/%N%/
007467	101	044	106	.ASCII	/ASF/
007472	124	114	105	.ASCII	/TLE/
007475	122	122	055	.ASCII	/RR-/
007500	040	125	116	.ASCII	/ UN/
007503	113	116	117	.ASCII	/KNO/
007506	127	116	040	.ASCII	/WN /
007511	127	101	111	.ASCII	/WAI/
007514	124	122	126	.ASCII	/TRV/
007517	040	122	105	.ASCII	/ RE/
007522	101	123	117	.ASCII	/ASO/
007525	116	040	101	.ASCII	/N A/
007530	124	040	104	.ASCII	/T D/
007533	056	122	126	.ASCII	/.RV/
007536	103	124	000	P.ADO:	.ASCII /CT/<00>
007541	000			.ASCII	<00>
007542	045	116	045	.ASCII	/%N%/
007545	101	044	106	.ASCII	/ASF/
007550	124	114	105	.ASCII	/TLE/
007553	122	122	055	.ASCII	/RR-/

CZRCH2
REV A PATCH 00 CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

007556	040	104	056	.ASCII	/ D./
007561	101	122	103	.ASCII	/ARC/
007564	123	040	104	.ASCII	/S D/
007567	111	104	040	.ASCII	/ID /
007572	116	117	124	.ASCII	/NOT/
007575	040	106	111	.ASCII	/ FI/
007600	116	104	040	.ASCII	/ND /
007603	103	114	117	.ASCII	/CLO/
007606	123	105	123	.ASCII	/SES/
007611	124	040	125	.ASCII	/T U/
007614	116	104	117	.ASCII	/NDO/
007617	116	105	040	.ASCII	/NE /
007622	132	117	116	.ASCII	/ZON/
007625	105	000	000	.ASCII	/E/<00><00>
007630	045	116	045	P.ADP: .ASCII	/XN%/
007633	101	044	106	.ASCII	/ASF/
007636	124	114	105	.ASCII	/TLE/
007641	122	122	055	.ASCII	/RR-/
007644	040	125	056	.ASCII	/ U./
007647	123	105	105	.ASCII	/SEE/
007652	113	040	106	.ASCII	/K F/
007655	117	125	116	.ASCII	/OUN/
007660	104	040	123	.ASCII	/D S/
007663	105	105	113	.ASCII	/EEK/
007666	040	124	117	.ASCII	/ TO/
007671	040	111	114	.ASCII	/ IL/
007674	114	105	107	.ASCII	/LEG/
007677	101	114	040	.ASCII	/AL /
007702	124	122	101	.ASCII	/TRA/
007705	103	113	000	.ASCII	/CK/<00>
007710	045	116	045	P.ADQ: .ASCII	/XN%/
007713	101	044	106	.ASCII	/ASF/
007716	124	114	105	.ASCII	/TLE/
007721	122	122	055	.ASCII	/RR-/
007724	040	125	056	.ASCII	/ U./
007727	110	124	123	.ASCII	/HTS/
007732	124	040	111	.ASCII	/T I/
007735	116	111	124	.ASCII	/NIT/
007740	040	104	111	.ASCII	/ DI/
007743	101	107	040	.ASCII	/AG /
007746	104	115	101	.ASCII	/DMA/
007751	040	127	122	.ASCII	/ WR/
007754	111	124	105	.ASCII	/ITE/
007757	040	106	101	.ASCII	/ FA/
007762	111	114	105	.ASCII	/ILE/
007765	104	000	000	.ASCII	/D/<00><00>
007770	045	116	045	P.ADR: .ASCII	/XN%/
007773	101	044	106	.ASCII	/ASF/
007776	124	114	105	.ASCII	/TLE/
010001	122	122	055	.ASCII	/RR-/
010004	040	125	056	.ASCII	/ U./
010007	110	124	123	.ASCII	/HTS/
010012	124	040	111	.ASCII	/T I/
010015	116	111	124	.ASCII	/NIT/
010020	040	104	111	.ASCII	/ DI/
010023	101	107	040	.ASCII	/AG /
010026	104	115	101	.ASCII	/DMA/

CZRCH2
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH2.B16;2 (13)

010031	040	103	117
010034	115	120	101
010037	122	105	040
010042	106	101	111
010045	114	105	104
010050	000	000	
010052	045	116	045
010055	101	044	106
010060	124	114	105
010063	122	122	055
010066	040	125	056
010071	123	131	104
010074	122	040	106
010077	117	125	116
010102	104	040	123
010105	123	056	104
010110	105	122	040
010113	123	105	124
010116	040	101	116
010121	104	040	123
010124	123	056	123
010127	120	116	040
010132	116	117	124
010135	040	123	105
010140	124	000	
010142	045	116	045
010145	101	044	106
010150	124	114	105
010153	122	122	055
010156	040	115	101
010161	123	124	105
010164	122	040	104
010167	122	111	126
010172	105	123	040
010175	101	103	114
010200	117	040	101
010203	123	123	105
010206	122	124	105
010211	104	000	000
010214	004736*		
010216	005016*		
010220	005064*		
010222	005132*		
010224	005200*		
010226	005264*		
010230	005350*		
010232	005416*		
010234	005464*		
010236	005532*		
010240	005600*		
010242	005650*		
010244	005720*		
010246	005774*		
010250	006060*		
010252	006140*		
010254	006206*		
010256	006254*		

P.ADS:

.ASCII	/ CO/
.ASCII	/MPA/
.ASCII	/RE /
.ASCII	/FAI/
.ASCII	/LED/
.ASCII	<00><00>
.ASCII	/XN%/
.ASCII	/ASF/
.ASCII	/TLE/
.ASCII	/RR-/
.ASCII	/ U./
.ASCII	/SYD/
.ASCII	/R F/
.ASCII	/OUN/
.ASCII	/D S/
.ASCII	/S.D/
.ASCII	/ER /
.ASCII	/SET/
.ASCII	/ AN/
.ASCII	/D S/
.ASCII	/S.S/
.ASCII	/PN /
.ASCII	/NOT/
.ASCII	/ SE/

P.ADT:

.ASCII	/T/<00>
.ASCII	/XN%/
.ASCII	/ASF/
.ASCII	/TLE/
.ASCII	/RR-/
.ASCII	/ MA/
.ASCII	/STE/
.ASCII	/R D/
.ASCII	/RIV/
.ASCII	/ES /
.ASCII	/ACL/
.ASCII	/O A/
.ASCII	/SSE/
.ASCII	/RTE/
.ASCII	/D/<00><00>

P.ACG:

.WORD	P.ACH
.WORD	P.ACI
.WORD	P.ACJ
.WORD	P.ACK
.WORD	P.ACL
.WORD	P.ACM
.WORD	P.ACN
.WORD	P.ACO
.WORD	P.ACP
.WORD	P.ACQ
.WORD	P.ACR
.WORD	P.ACS
.WORD	P.ACT
.WORD	P.ACU
.WORD	P.ACV
.WORD	P.ACW
.WORD	P.ACX
.WORD	P.ACY

CZRCH2
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH2.B16;2 (13)

010260	006322*					.WORD	P.ACZ
010262	006374*					.WORD	P.ADA
010264	006444*					.WORD	P.ADB
010266	006524*					.WORD	P.ADC
010270	006574*					.WORD	P.ADD
010272	006644*					.WORD	P.ADE
010274	006716*					.WORD	P.ADF
010276	007006*					.WORD	P.ADG
010300	007070*					.WORD	P.ADH
010302	007150*					.WORD	P.ADI
010304	007224*					.WORD	P.ADJ
010306	007300*					.WORD	P.ADK
010310	007342*					.WORD	P.ADL
010312	007400*					.WORD	P.ADM
010314	007464*					.WORD	P.ADN
010316	007542*					.WORD	P.ADO
010320	007630*					.WORD	P.ADP
010322	007710*					.WORD	P.ADQ
010324	007770*					.WORD	P.ADR
010326	010052*					.WORD	P.ADS
010330	010142*					.WORD	P.ADT
010332	045	101	040	P.ADV:	.ASCII	/XA /	
010335	123	125	103		.ASCII	/SUC/	
010340	103	105	123		.ASCII	/CES/	
010343	123	106	125		.ASCII	/SFU/	
010346	114	045	116		.ASCII	/LXN/	
010351	000				.ASCII	<00>	
010352	045	101	111	P.ADW:	.ASCII	/XAI/	
010355	116	126	101		.ASCII	/NVA/	
010360	114	111	104		.ASCII	/LID/	
010363	040	103	117		.ASCII	/ CO/	
010366	115	115	101		.ASCII	/MMA/	
010371	116	104	045		.ASCII	/NDX/	
010374	116	000			.ASCII	/N/<00>	
010376	045	101	116	P.ADX:	.ASCII	/XAN/	
010401	117	040	122		.ASCII	/O R/	
010404	105	107	111		.ASCII	/EGI/	
010407	117	116	040		.ASCII	/ON /	
010412	101	126	101		.ASCII	/AVA/	
010415	111	114	101		.ASCII	/ILA/	
010420	102	114	105		.ASCII	/BLE/	
010423	045	116	000		.ASCII	/XN/<00>	
010426	045	101	116	P.ADY:	.ASCII	/XAN/	
010431	117	040	122		.ASCII	/O R/	
010434	105	107	111		.ASCII	/EGI/	
010437	117	116	040		.ASCII	/ON /	
010442	123	125	111		.ASCII	/SUI/	
010445	124	101	102		.ASCII	/TAB/	
010450	114	105	045		.ASCII	/LEX/	
010453	116	000	000		.ASCII	/N/<00><00>	
010456	045	101	120	P.ADZ:	.ASCII	/XAP/	
010461	122	117	107		.ASCII	/ROG/	
010464	122	101	115		.ASCII	/RAM/	
010467	040	116	117		.ASCII	/ NO/	
010472	124	040	113		.ASCII	/T K/	
010475	116	117	127		.ASCII	/NOW/	
010500	116	045	116		.ASCII	/NXN/	

CZRCH2
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH2.B16;2 (13)

010503	000				.ASCII	<00>
010504	045	101	114	P.AEA:	.ASCII	/XAL/
010507	117	101	104		.ASCII	/OAD/
010512	040	106	101		.ASCII	/FA/
010515	111	114	125		.ASCII	/ILU/
010520	122	105	045		.ASCII	/REX/
010523	116	000	000		.ASCII	/N/<00><00>
010526	045	101	123	P.AEB:	.ASCII	/XAS/
010531	124	101	116		.ASCII	/TAN/
010534	104	101	114		.ASCII	/DAL/
010537	117	116	105		.ASCII	/ONE/
010542	045	116	000		.ASCII	/XN/<00>
010545	000				.ASCII	<00>
010546	010332*			P.ADU:	.WORD	P.ADV
010550	010352*				.WORD	P.ADW
010552	010376*				.WORD	P.ADX
010554	010426*				.WORD	P.ADY
010556	010456*				.WORD	P.ADZ
010560	010504*				.WORD	P.AEA
010562	010526*				.WORD	P.AEB
010564	045	101	123	P.AED:	.ASCII	/XAS/
010567	125	103	103		.ASCII	/UCC/
010572	105	123	123		.ASCII	/ESS/
010575	045	116	000		.ASCII	/XN/<00>
010600	045	101	111	P.AEE:	.ASCII	/XAI/
010603	116	126	101		.ASCII	/NVA/
010606	114	111	104		.ASCII	/LID/
010611	040	103	117		.ASCII	/CO/
010614	115	115	101		.ASCII	/XFA/
010617	116	104	045		.ASCII	/NDX/
010622	116	000			.ASCII	/N/<00>
010624	045	101	103	P.AEF:	.ASCII	/XAC/
010627	117	115	115		.ASCII	/OMM/
010632	101	116	104		.ASCII	/AND/
010635	040	101	102		.ASCII	/AB/
010640	117	122	124		.ASCII	/ORT/
010643	105	104	045		.ASCII	/EDX/
010646	116	000			.ASCII	/N/<00>
010650	045	101	125	P.AEG:	.ASCII	/XAU/
010653	116	111	124		.ASCII	/NIT/
010656	055	117	106		.ASCII	/-OF/
010661	106	114	111		.ASCII	/FLI/
010664	116	105	045		.ASCII	/NEX/
010667	116	000	000		.ASCII	/N/<00><00>
010672	045	101	125	P.AEH:	.ASCII	/XAU/
010675	116	111	124		.ASCII	/NIT/
010700	055	101	126		.ASCII	/-AV/
010703	101	111	114		.ASCII	/AIL/
010706	101	102	114		.ASCII	/ABL/
010711	105	045	116		.ASCII	/EXN/
010714	000	000			.ASCII	<00><00>
010716	045	101	115	P.AEI:	.ASCII	/XAM/
010721	105	104	111		.ASCII	/EDI/
010724	101	040	106		.ASCII	/AF/
010727	117	122	115		.ASCII	/ORM/
010732	101	124	040		.ASCII	/AT /
010735	105	122	122		.ASCII	/ERR/

CZRCH2
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH2.B16;2 (13)

010740	117	122	045		.ASCII	/OR%/
010743	116	000	000		.ASCII	/N/<00><00>
010746	045	101	127	P.AEJ:	.ASCII	/XAW/
010751	122	111	124		.ASCII	/RIT/
010754	105	040	120		.ASCII	/E P/
010757	122	117	124		.ASCII	/ROT/
010762	105	103	124		.ASCII	/ECT/
010765	105	104	045		.ASCII	/ED%/
010770	116	000			.ASCII	/N/<00>
010772	045	101	103	P.AEK:	.ASCII	/XAC/
010775	117	115	120		.ASCII	/OMP/
011000	101	122	105		.ASCII	/ARE/
011003	040	105	122		.ASCII	/ ER/
011006	122	117	122		.ASCII	/ROR/
011011	045	116	000		.ASCII	/XN/<00>
011014	045	101	104	P.AEL:	.ASCII	/XAD/
011017	101	124	101		.ASCII	/ATA/
011022	040	105	122		.ASCII	/ ER/
011025	122	117	122		.ASCII	/ROR/
011030	045	116	000		.ASCII	/XN/<00>
011033	000				.ASCII	<00>
011034	045	101	110	P.AEM:	.ASCII	/XAH/
011037	117	123	124		.ASCII	/OST/
011042	040	102	125		.ASCII	/ BU/
011045	106	106	105		.ASCII	/FFE/
011050	122	040	101		.ASCII	/R A/
011053	103	103	105		.ASCII	/CCE/
011056	123	123	040		.ASCII	/SS /
011061	105	122	122		.ASCII	/ERR/
011064	117	122	045		.ASCII	/OR%/
011067	116	000	000		.ASCII	/N/<00><00>
011072	045	101	103	P.AEN:	.ASCII	/XAC/
011075	117	116	124		.ASCII	/ONT/
011100	122	117	114		.ASCII	/ROL/
011103	114	105	122		.ASCII	/LER/
011106	040	105	122		.ASCII	/ ER/
011111	122	117	122		.ASCII	/ROR/
011114	045	116	000		.ASCII	/XN/<00>
011117	000				.ASCII	<00>
011120	045	101	104	P.AEO:	.ASCII	/XAD/
011123	122	111	126		.ASCII	/RIV/
011126	105	040	105		.ASCII	/E E/
011131	122	122	117		.ASCII	/RRO/
011134	122	045	116		.ASCII	/RXN/
011137	000				.ASCII	<00>
011140	045	101	115	P.AEP:	.ASCII	/XAM/
011143	105	123	123		.ASCII	/ESS/
011146	101	107	105		.ASCII	/AGE/
011151	040	106	122		.ASCII	/ FR/
011154	117	115	040		.ASCII	/OM /
011157	101	116	040		.ASCII	/AN /
011162	111	116	124		.ASCII	/INT/
011165	105	122	116		.ASCII	/ERN/
011170	101	114	040		.ASCII	/AL /
011173	104	111	101		.ASCII	/DIA/
011176	107	116	117		.ASCII	/GNO/
011201	123	124	111		.ASCII	/STI/

CZRCH2
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

VAX-11 Bliss-16 V3-555
SPIDERSUSERS:[NEALE.AZTEC]CZRCH2.B16;2 (13) SEQ 112
Page 47

011204	103	045	116		.ASCII	/C%N/
011207	000				.ASCII	<00>
011210	010564'			P.AEC:	.WORD	P.AED
011212	010600'				.WORD	P.AEE
011214	010624'				.WORD	P.AEF
011216	010650'				.WORD	P.AEG
011220	010672'				.WORD	P.AEH
011222	010716'				.WORD	P.AEI
011224	010746'				.WORD	P.AEJ
011226	010772'				.WORD	P.AEK
011230	011014'				.WORD	P.AEL
011232	011034'				.WORD	P.AEM
011234	011072'				.WORD	P.AEN
011236	011120'				.WORD	P.AEO
011240	011140'				.WORD	P.AEP
011242	105	116	124	P.AEQ:	.ASCII	/ENT/
011245	105	122	040		.ASCII	/ER /
011250	104	101	124		.ASCII	/DAT/
011253	105	040	074		.ASCII	/E </
011256	115	115	055		.ASCII	/MM-/
011261	104	104	055		.ASCII	/DD-/
011264	131	131	131		.ASCII	/YYY/
011267	131	076	040		.ASCII	/Y> /
011272	000	000			.ASCII	<00><00>
011274	045	116	045	P.AER:	.ASCII	/XN% /
011277	101	116	117		.ASCII	/ANO/
011302	040	101	104		.ASCII	/ AD/
011305	104	111	124		.ASCII	/DIT/
011310	111	117	116		.ASCII	/ION/
011313	101	114	040		.ASCII	/AL /
011316	125	116	111		.ASCII	/UNI/
011321	124	123	040		.ASCII	/TS /
011324	124	117	040		.ASCII	/TO /
011327	106	117	122		.ASCII	/FOR/
011332	115	101	124		.ASCII	/MAT/
011335	040	055	040		.ASCII	/ - /
011340	101	102	117		.ASCII	/ABO/
011343	122	124	111		.ASCII	/RTI/
011346	116	107	000		.ASCII	/NG/<00>
011351	000				.ASCII	<00>
011352	045	116	045	P.AES:	.ASCII	/XN% /
011355	101	044	106		.ASCII	/ASF/
011360	124	114	105		.ASCII	/TLE/
011363	122	122	055		.ASCII	/RR-/
011366	040	111	116		.ASCII	/ IN/
011371	111	124	040		.ASCII	/IT /
011374	103	117	104		.ASCII	/COD/
011377	105	040	122		.ASCII	/E R/
011402	105	055	105		.ASCII	/E-E/
011405	116	124	105		.ASCII	/NTE/
011410	122	105	104		.ASCII	/RED/
011413	040	104	125		.ASCII	/ DU/
011416	105	040	124		.ASCII	/E T/
011421	117	040	120		.ASCII	/O P/
011424	127	122	040		.ASCII	/WR /
011427	106	101	111		.ASCII	/FAI/
011432	114	000			.ASCII	/L/<00>

CZRCH2
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

VAX-11 Bliss-16 V3-555
SPIDERSUSERS:[NEALE.AZTEC]CZRCH2.B16;2 (13)

011434	045	116	045	P.AET:	.ASCII	/XN%/
011437	101	044	106		.ASCII	/ASF/
011442	124	114	105		.ASCII	/TLE/
011445	122	122	055		.ASCII	/RR-/
011450	040	101	102		.ASCII	/ AB/
011453	117	122	124		.ASCII	/ORT/
011456	111	116	107		.ASCII	/ING/
011461	040	110	117		.ASCII	/ HO/
011464	123	124	040		.ASCII	/ST /
011467	101	116	104		.ASCII	/AND/
011472	040	122	105		.ASCII	/ RE/
011475	115	117	124		.ASCII	/MOT/
011500	105	040	120		.ASCII	/E P/
011503	122	117	107		.ASCII	/ROG/
011506	122	101	115		.ASCII	/RAM/
011511	123	000	000		.ASCII	/S/<00><00>
011514	045	116	045	P.AEU:	.ASCII	/XN%/
011517	101	044	106		.ASCII	/ASF/
011522	124	114	105		.ASCII	/TLE/
011525	122	122	055		.ASCII	/RR-/
011530	040	111	114		.ASCII	/ IL/
011533	114	105	107		.ASCII	/LEG/
011536	101	114	040		.ASCII	/AL /
011541	116	125	115		.ASCII	/NUM/
011544	102	105	122		.ASCII	/BER/
011547	040	117	106		.ASCII	/ OF/
011552	040	125	116		.ASCII	/ UN/
011555	111	124	123		.ASCII	/ITS/
011560	040	123	105		.ASCII	/ SE/
011563	114	105	103		.ASCII	/LEC/
011566	124	105	104		.ASCII	/TED/
011571	000				.ASCII	<00>
011572	045	116	045	P.AEV:	.ASCII	/XN%/
011575	101	044	106		.ASCII	/ASF/
011600	124	114	105		.ASCII	/TLE/
011603	122	122	055		.ASCII	/RR-/
011606	040	114	111		.ASCII	/ LI/
011611	115	111	124		.ASCII	/MIT/
011614	040	117	106		.ASCII	/ OF/
011617	040	123	111		.ASCII	/ SI/
011622	130	124	105		.ASCII	/XTE/
011625	105	116	040		.ASCII	/EN /
011630	125	116	111		.ASCII	/UNI/
011633	124	123	040		.ASCII	/TS /
011636	120	105	122		.ASCII	/PER/
011641	040	106	117		.ASCII	/ FO/
011644	122	115	101		.ASCII	/RMA/
011647	124	111	116		.ASCII	/TIN/
011652	107	040	123		.ASCII	/G S/
011655	105	123	123		.ASCII	/ESS/
011660	111	117	116		.ASCII	/ION/
011663	000				.ASCII	<00>
011664	045	116	045	P.AEW:	.ASCII	/XN%/
011667	101	044	106		.ASCII	/ASF/
011672	124	114	105		.ASCII	/TLE/
011675	122	122	055		.ASCII	/RR-/
011700	040	122	103		.ASCII	/ RC/

CZRCH2
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

011703	062	065	040	.ASCII	/25 /
011706	103	117	116	.ASCII	/CON/
011711	124	122	117	.ASCII	/TRO/
011714	114	114	105	.ASCII	/LLE/
011717	122	040	111	.ASCII	/R I/
011722	116	111	124	.ASCII	/NIT/
011725	111	101	114	.ASCII	/IAL/
011730	111	132	101	.ASCII	/IZA/
011733	124	111	117	.ASCII	/TIO/
011736	116	040	105	.ASCII	/N E/
011741	122	122	117	.ASCII	/RRO/
011744	122	000		.ASCII	/R/<00>
011746	045	116	045	P.AEX:	.ASCII /%N%/
011751	101	044	106		.ASCII /ASF/
011754	124	114	105		.ASCII /TLE/
011757	122	122	055		.ASCII /RR-/
011762	040	120	122		.ASCII / PR/
011765	117	124	117		.ASCII /OTO/
011770	103	117	114		.ASCII /COL/
011773	040	126	111		.ASCII / VI/
011776	117	114	101		.ASCII /OLA/
012001	124	111	117		.ASCII /TIO/
012004	116	040	105		.ASCII /N E/
012007	122	122	117		.ASCII /RRO/
012012	122	000			.ASCII /R/<00>
012014	045	116	045	P.AEY:	.ASCII /%N%/
012017	101	044	106		.ASCII /ASF/
012022	124	114	105		.ASCII /TLE/
012025	122	122	055		.ASCII /RR-/
012030	040	103	117		.ASCII / CO/
012033	115	115	125		.ASCII /MMU/
012036	116	111	103		.ASCII /NIC/
012041	101	124	111		.ASCII /ATI/
012044	117	116	040		.ASCII /ON /
012047	101	122	105		.ASCII /ARE/
012052	101	040	111		.ASCII /A I/
012055	116	111	124		.ASCII /NIT/
012060	040	105	122		.ASCII / ER/
012063	122	117	122		.ASCII /ROR/
012066	000	000			.ASCII <00><00>
012070	045	116	045	P.AEZ:	.ASCII /%N%/
012073	101	044	106		.ASCII /ASF/
012076	124	114	105		.ASCII /TLE/
012101	122	122	055		.ASCII /RR-/
012104	040	104	125		.ASCII / DU/
012107	120	040	123		.ASCII /P S/
012112	105	122	126		.ASCII /ERV/
012115	105	122	040		.ASCII /ER /
012120	101	103	124		.ASCII /ACT/
012123	111	126	105		.ASCII /IVE/
012126	040	101	106		.ASCII / AF/
012131	124	105	122		.ASCII /TER/
012134	040	111	116		.ASCII / IN/
012137	111	124	111		.ASCII /ITI/
012142	101	114	111		.ASCII /ALI/
012145	132	111	116		.ASCII /ZIN/
012150	107	000			.ASCII /G/<00>

CZRCH2
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

VAX-11 Bliss-16 V3-555
SPIDER\$USERS:[NEALE.AZTEC]CZRCH2.B16;2 (13)

012152	045	116	045	P.AFA:	.ASCII	/XN%/
012155	101	044	106		.ASCII	/ASF/
012160	124	114	105		.ASCII	/TLE/
012163	122	122	055		.ASCII	/RR-/
012166	040	104	125		.ASCII	/ DU/
012171	120	040	123		.ASCII	/P S/
012174	105	122	126		.ASCII	/ERV/
012177	105	122	040		.ASCII	/ER /
012202	111	116	101		.ASCII	/INA/
012205	103	124	111		.ASCII	/CTI/
012210	126	105	040		.ASCII	/VE /
012213	101	106	124		.ASCII	/AFT/
012216	105	122	040		.ASCII	/ER /
012221	105	130	137		.ASCII	/EX /
012224	123	125	120		.ASCII	/SUP/
012227	137	120	122		.ASCII	/ PR/
012232	117	107	040		.ASCII	/DG /
012235	103	117	115		.ASCII	/COM/
012240	115	101	116		.ASCII	/MAN/
012243	104	000	000		.ASCII	/D/<00><00>
012246	105	116	124	P.AFB:	.ASCII	/ENT/
012251	105	122	040		.ASCII	/ER /
012254	106	103	124		.ASCII	/FCT/
012257	040	106	111		.ASCII	/ FI/
012262	114	105	040		.ASCII	/LE /
012265	116	101	115		.ASCII	/NAM/
012270	105	040	124		.ASCII	/E T/
012273	117	040	122		.ASCII	/O R/
012276	105	123	124		.ASCII	/EST/
012301	117	122	105		.ASCII	/ORE/
012304	040	050	106		.ASCII	/ (F/
012307	111	114	105		.ASCII	/ILE/
012312	116	101	115		.ASCII	/NAM/
012315	105	056	105		.ASCII	/E.E/
012320	130	124	051		.ASCII	/XT)/
012323	072	040	000		.ASCII	/: /<00>
012326	122	103	062	P.AFC:	.ASCII	/RC2/
012331	065	040	040		.ASCII	/S /
012334	111	120	040		.ASCII	/IP /
012337	040	040	122		.ASCII	/ R/
012342	105	107	111		.ASCII	/EGI/
012345	123	124	105		.ASCII	/STE/
012350	122	040	040		.ASCII	/R /
012353	040	101	104		.ASCII	/ AD/
012356	104	122	105		.ASCII	/DRE/
012361	123	123	000		.ASCII	/SS/<00>
012364	122	103	062	P.AFD:	.ASCII	/RC2/
012367	065	040	111		.ASCII	/S I/
012372	116	124	105		.ASCII	/NTE/
012375	122	122	125		.ASCII	/RRU/
012400	120	124	040		.ASCII	/PT /
012403	126	105	103		.ASCII	/VEC/
012406	124	117	122		.ASCII	/TOR/
012411	040	101	104		.ASCII	/ AD/
012414	104	122	105		.ASCII	/DRE/
012417	123	123	000		.ASCII	/SS/<00>
012422	122	103	062	P.AFE:	.ASCII	/RC2/

CZRCH2
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH2.B16;2 (13)

012425	065	040	040	.ASCII	/S /
012430	040	040	102	.ASCII	/ B/
012433	125	123	040	.ASCII	/US /
012436	040	040	122	.ASCII	/ R/
012441	105	121	125	.ASCII	/EQU/
012444	105	123	124	.ASCII	/EST/
012447	040	040	040	.ASCII	/ /
012452	114	105	126	.ASCII	/LEV/
012455	105	114	000	.ASCII	/EL/<00>
012460	125	116	111	P.AFF:	.ASCII /UNI/
012463	124	040	040	.ASCII	/T /
012466	116	125	115	.ASCII	/NUM/
012471	102	105	122	.ASCII	/BER/
012474	040	124	117	.ASCII	/ TO/
012477	040	102	122	.ASCII	/ BR/
012502	111	116	107	.ASCII	/ING/
012505	040	040	117	.ASCII	/ O/
012510	116	114	111	.ASCII	/NLI/
012513	116	105	000	.ASCII	/NE/<00>
012516	106	117	122	P.AFG:	.ASCII /FOR/
012521	115	101	124	.ASCII	/MAT/
012524	040	111	116	.ASCII	/ IN/
012527	040	125	116	.ASCII	/ UN/
012532	101	124	124	.ASCII	/ATT/
012535	105	116	104	.ASCII	/END/
012540	105	104	040	.ASCII	/ED /
012543	122	105	106	.ASCII	/REF/
012546	117	122	115	.ASCII	/ORM/
012551	101	124	040	.ASCII	/AT /
012554	115	117	104	.ASCII	/MOD/
012557	105	000	000	.ASCII	/E/<00><00>
012562	045	116	045	P.AFH:	.ASCII /%N%/
012565	101	052	052	.ASCII	/A**/
012570	052	052	052	.ASCII	/***/
012573	052	052	052	.ASCII	/***/
012576	052	052	052	.ASCII	/***/
012601	052	052	052	.ASCII	/***/
012604	052	052	052	.ASCII	/***/
012607	052	052	052	.ASCII	/***/
012612	052	040	116	.ASCII	/* N/
012615	117	124	111	.ASCII	/OTI/
012620	103	105	040	.ASCII	/CE /
012623	052	052	052	.ASCII	/***/
012626	052	052	052	.ASCII	/***/
012631	052	052	052	.ASCII	/***/
012634	052	052	052	.ASCII	/***/
012637	052	052	052	.ASCII	/***/
012642	052	052	052	.ASCII	/***/
012645	052	052	052	.ASCII	/***/
012650	000	000		.ASCII	<00><00>
012652	045	116	045	P.AFI:	.ASCII /%N%/
012655	101	040	040	.ASCII	/A /
012660	040	040	040	.ASCII	/ /
012663	117	120	105	.ASCII	/OPE/
012666	122	101	124	.ASCII	/RAT/
012671	117	122	040	.ASCII	/OR /
012674	115	125	123	.ASCII	/MUS/

CZRCH2
REV A PATCH 00 CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

012677	124	040	102	.ASCII	/T B/
012702	105	040	120	.ASCII	/E P/
012705	122	105	123	.ASCII	/RES/
012710	105	116	124	.ASCII	/ENT/
012713	040	111	116	.ASCII	/ IN/
012716	040	101	124	.ASCII	/ AT/
012721	124	105	116	.ASCII	/TEN/
012724	104	105	104	.ASCII	/DED/
012727	040	115	117	.ASCII	/ MO/
012732	104	105	040	.ASCII	/DE /
012735	000			.ASCII	<00>
012736	045	116	045	P.AFJ: .ASCII	/XNZ/
012741	101	040	040	.ASCII	/A /
012744	040	040	040	.ASCII	/ /
012747	040	040	122	.ASCII	/ R/
012752	125	116	116	.ASCII	/UNN/
012755	111	116	107	.ASCII	/ING/
012760	040	111	116	.ASCII	/ IN/
012763	040	125	116	.ASCII	/ UN/
012766	055	101	124	.ASCII	/-AT/
012771	124	105	116	.ASCII	/TEN/
012774	104	105	104	.ASCII	/DED/
012777	040	122	105	.ASCII	/ RE/
013002	106	117	122	.ASCII	/FOR/
013005	115	101	124	.ASCII	/MAT/
013010	040	115	117	.ASCII	/ MO/
013013	104	105	000	.ASCII	/DE/<00>

000000	.PSECT	\$GLOBS, RO , D , GBL
000000	COM.AREA::	
	.BLKW	24
000050	HEAD.AREA::	
	.BLKW	1
000052	RECEIVE.RING::	
	.BLKW	1
000054	SEND.RING::	
	.BLKW	1
000056	REC.ENVELOPE::	
	.BLKW	200
000456	SND.ENVELOPE::	
	.BLKW	130
000736	FCT.BUF::	
	.BLKW	400
001736	REC.BUF::	
	.BLKW	170
002316	SND.BUF::	
	.BLKW	45
002430	OUT\$STD.BUF::	
	.BLKW	10
002450	RET.ENSAD::	
	.BLKW	1
002452	DATETXT::	
	.BLKW	6
002466	FLG.WRD::	
	.BLKW	1

CZRCH2
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
GLOBAL TEXT SECTION

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

VAX-11 Bliss-16 V3-555
SPIDERSUSERS:[NEALE.AZTEC]CZRCH2.B16;2 (13)

002470		NEX.FLAG::	
		.BLKW	1
002472		OVSA::	.BLKW 1
002474		NXT.CRN::	
		.BLKB	1
		.EVEN	
002476	000000	RET.STATUS::	
		.WORD	0
002500		LUN::	.BLKW 1
002502		PID.SAVE::	
		.BLKW	2
002506		NSD.SLOT::	
		.BLKW	1
002510		NRD.SLOT::	
		.BLKW	1
002512		RC25.ADDR::	
		.BLKW	1
002514		VEC.ADDR::	
		.BLKW	1
002516		BR.LEVEL::	
		.BLKW	1
002520		UNIT.NO::	
		.BLKW	1
002522		PTBL.PTR::	
		.BLKW	1
002524	001377	RSVD.STRUCT::	
		.WORD	1377
002526	000000	.WORD	0
002530	003400	.WORD	3400
002532	003777	.WORD	3777
002534	005777	ISD.STRUCT::	
		.WORD	5777
002536	111033	.WORD	-66745
002540	010222	.WORD	10222
002542	000010	.WORD	RINGBASE
002544	023433	.WORD	23433
002546	000000	.WORD	0
002550	043777	.WORD	43777
002552	177400	.WORD	-400

.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

000126*	L\$ERRTBL==	ERRTYP
000154*	L\$SW==	L\$SWLEN+2
000140*	L\$HW==	L\$HWLEN+2
000011*	L\$DEPO==	L\$REV+1
000140*	DFPTBL==	L\$HWLEN+2
000154*	SFPTBL==	L\$SWLEN+2
000000*	FMT1==	P.AAA
000006*	FMT2==	P.AAB
000076*	FMT3==	P.AAC
000204*	FMT4==	P.AAD
000270*	FMT5==	P.AAE

000400'	FMT6==	P.AAF
000476'	FMT7==	P.AAG
000544'	PID.FMT==	P.AAH
000646'	CRLF==	P.AAI
000652'	XCRLF==	P.AAJ
000010'	RINGBASE==	COM.AREA+10
001740'	MSGADR==	REC.BUF+2
002660'	PFE.STRUCT==	P.AAK
004660'	EMSG.STRUCT==	P.ABI
010214'	RC.STRUCTURE==	P.ACG
010546'	SDUP.STRUCT==	P.ADU
011210'	SMSCP.STRUCT==	P.AEC
011242'	DATMSG==	P.AEQ
011274'	NO.ADD.UNITS==	P.AER
011352'	PWR.MSG==	P.AES
011434'	ABO.MSG==	P.AET
011514'	TO.MANY.UNITS==	P.AEU
011572'	GOOD.NUM.UNITS==	P.AEV
011664'	BOOT.FAILURE==	P.AEW
011746'	PROTO.VIOLATION==	P.AEX
012014'	PORT.INIT.ERR==	P.AEY
012070'	ACTIVE.DUP.SERVER==	P.AEZ
012152'	INACTIVE.DUP.SERVER==	
		P.AFA
012246'	FCT.REQ.MSG==	P.AFB
012326'	HW.Q1.IP==	P.AFC
012364'	HW.Q2.VECTOR==	P.AFD
012422'	HW.Q3.BR==	P.AFE
012460'	HW.Q4.UNIT==	P.AFF
012516'	SW.Q1.UNATT==	P.AFG
012562'	SW.Q2.NOTICE==	P.AFH
012652'	SW.Q3.OPER==	P.AFI
012736'	SW.Q4.UNATT==	P.AFJ

PSECT SUMMARY

Psect Name	Words	Attributes			
AASCODE	59	RO , I	LCL	REL	CON
\$GLOB\$	694	RO , D	GBL	REL	CON
\$PLIT\$	2823	RO , D	GBL	REL	CON

LIBRARY STATISTICS

File	----- Symbols -----		Percent	Blocks Read
	Total	Loaded		
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH0.L16;5	398	176	44	42

CZRCH2 CZRCH RC25 DISK FORMATTER
REV A PATCH 00 GLOBAL TEXT SECTION

11-Jul-1983 16:05:07
7-Jul-1983 08:21:37

VAX-11 Bliss-16 V3-555
SPIDER\$USERS:[NEALE.AZTEC]CZRCH2.B16;2 (13)

: COMMAND QUALIFIERS

: BLISS /PDP11/LIST CZRCH2.B16

: Size: 0 code + 3576 data words
: Run Time: 00:24.0
: Elapsed Time: 00:35.7
: Memory Used: 209 pages
: Compilation Complete

CZRCH3

CZRCH RC25 DISK FORMATTER

11-Jul-1983 16:05:44
7-Jul-1983 08:24:19VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH3.B16;3 (1)

SEQ 121

Page 1

```

:
: 0001 MODULE CZRCH3 (%TITLE 'CZRCH RC25 DISK FORMATTER'
: 0002 IDENT = 'REV A PATCH 00',
: 0003 ADDRESSING MODE (RELATIVE) ,
: 0004 ENVIRONMENT (NOEIS)
: 0005 ) =
: 0006 BEGIN
: 0007 %sbttl 'MODULE DECLARATIONS'
: 0008 |
: 0009 | Pretty Declarations
: 0010 |
: 0011 | <BLF/LOWERCASE_KEY>
: 0012 |
: 0013 |
: 0014 | Library 'CZRCH0'; !Define RC25 Library module
: 0015 |
: 0016 | require 'BLSMAC.REQ'; !Define Bliss Macro Library
: 1505 |
: 1506 | +
: 1507 | Structure declarations used within this module.
: 1508 | -
: 1509 |
: 1510 | structure
: 1511 |
: 1512 | +
: 1513 | RC25 register accessing structure. This
: 1514 | structure allows RC25 register accessing
: 1515 | to be transportable between the PDP-11 and
: 1516 | VAX Diagnostic Supervisors.
: 1517 |
: 1518 | This also defines an access algorithm for
: 1519 | VAX to allow field reference to MBA address
: 1520 | space without generating machine checks.
: 1521 | -
: 1522 |
: 1523 | RC25 [O, P, S, E] =
: 1524 | begin
: 1525 |
: 1526 | local
: 1527 | RC$$_REG;
: 1528 |
: 1529 | RC$$_REG = .(RC25 + %upval*0)<0, %bpval, 0>;
: 1530 | RC$$_REG
: 1531 | end
: 1532 | <P, S, E>;
: 1533 |
: 1534 | !<blf/page>

```

CZRCH3
REV A PATCH 00CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS11-Jul-1983 16:05:44
7-Jul-1983 08:24:19VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH3.B16;3 (2)SEQ 122
Page 2

```
1535 :+
1536 : The psect named "code or $code$" is redefined here
1537 : to be called "aa$code". This is done to force the tkb
1538 : linker to place the programs header information starting
1539 : at absolute address 2000. Then for consistency "a$code"
1540 : is used in place of "code or $code" across all modules.
1541 :-
1542 psect
1543     code = aa$code;
1544
1545 :+
1546 : External Routine declared outside this module.
1547 :-
1548
1549 external routine
1550     DECODE : novalue,
1551     LOAD_FILE;
1552
1553 !<blf/page>
```

CZRCH3
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

11-Jul-1983 16:05:44
7-Jul-1983 08:24:19

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH3.B16;3 (3)

```

1554 !+
1555 ! External Declaration of datums declared outside of this module.
1556 !-
1557
1558 external
1559
1560 ! DM load file from local media definitions
1561
1562 !DM FN$EXT, !DM file name ascii text
1563 AZFMTR : vector [8892, word], !DM host buffer adrs
1564 OVSA : word, !Overlay section starting adrs
1565
1566 ! Hardware question ascii string messages
1567
1568 HW_Q1_IP, !H/W question 1 for IP reg address
1569 HW_Q2_VECTOR, !H/W question 2 for interrupt vector address
1570 HW_Q3_BR, !H/W question 3 for bus req level
1571 HW_Q4_UNIT, !H/W question 4 for unit no. to format
1572
1573 ! Software question ascii string messages
1574
1575 SW_Q1_UNATT, !Unattended reformat software question
1576 SW_Q2_NOTICE, !Notice message
1577 SW_Q3_OPER, !Operator must be present in this mode
1578 SW_Q4_UNATT, !Running in unattended mode
1579
1580 ! Formatting print string
1581
1582 FMT2, !Notifies unit being formatted
1583 FMT3, !Notifies format abort
1584 CRIF, !<CR><LF>
1585 XCRLF, !Prints ten <CR><LF>
1586
1587 ! Init code error and informational messages
1588
1589 NO_ADD_UNITS,
1590 PWR_MSG,
1591 ABO_MSG,
1592 TO_MANY_UNITS,
1593 GOOD_NUM_UNITS,
1594 DATMSG, !Message to ask operator for date
1595
1596 ! Miscellaneous external data declarations
1597
1598 FLG_WRD : bitvector [16], !Global flag word
1599 LUN : byte,
1600 PTBL_PTR : ref vector [4, word],
1601 DATETXT : vector [12, byte], !Storage for operator date input
1602
1603 ! Supervisor defined data declarations
1604
1605 L$UNIT,
1606
1607 ! Hardware P_Table storage declarations
1608
1609 RC25_ADDR : ref RC25 field (ISD_FIELD),
1610 VEC_ADDR : word,

```

CZRCH3
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

11-Jul-1983 16:05:44
7-Jul-1983 08:24:19

VAX-11 Bliss-16 V3-5.5
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH3.B16;3 (3)

```
:      1611      BR_LEVEL : word,  
:      1612      UNIT_NO  : word,  
:      1613      |  
:      1614      | Software question response storage declarations  
:      1615      |  
:      1616      SW_UNATT : word,  
:      1617      |  
:      1618      | Formatter data structures  
:      1619      |  
:      1620      ISD_STRUCT : blockvector [4, 2, word] field (ISD_FIELD);  
:      1621
```

CZRCH3
REV A PATCH 00CZRCH RC25 DISK FORMATTER
TYPE AND DESCRIPTION11-Jul-1983 16:05:44
7-Jul-1983 08:24:19VAX-11 Bliss-16 V3-555
SPIDERSUSERS:[NEALE.AZTEC]CZRCH3.B16;3 (4)

```
: 1622 %sbttl 'TYPE AND DESCRIPTION'
: 1623 |
: 1624 | Two lines of text will be printed to the operator (in addition to the
: 1625 | program name). The first will come from the 'DESCRIPT' macro at start
: 1626 | up time and will identify the diagnostics. The second will come from
: 1627 | the 'DEVTYPE' macro at hardware dialogue time and will identify the
: 1628 | device under test. The arguments of both macros are 72 character
: 1629 | ascii strings enclosed in parentheses:
: 1630 | -
: 1631 | DESCRIPT (%asciz'RC25 DISK FORMATTER');
: 1632 | DEVTYP (%asciz'RC25 DISK DRIVE SUBSYSTEM');
```

CZRCH3
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
HARDWARE PARAMETER CODING

11-Jul-1983 16:05:44
7-Jul-1983 08:24:19

VAX-11 Bliss-16 V3-555
SPIDERSUSERS:[NEALE.AZTEC]CZRCH3.B16;3 (5)

```

: 1633 %sbttl 'HARDWARE PARAMETER CODING'
: 1634 !+
: 1635 ! The hardware parameter coding section contains macros
: 1636 ! that are used by the supervisor to build P-Tables. The
: 1637 ! macros are not executed as machine instructions but are
: 1638 ! interpreted by the supervisor as data structures. The
: 1639 ! macros allow the supervisor to establish communications
: 1640 ! with the operator.
: 1641 !-
: 1642 BGNHRD:
: 1643 GPRMA (HW_Q1_IP, %o'0', 0, %o'16000', %o'177777', YES, 1); !Get RC25 Controller IP register
: 1644 GPRMA (HW_Q2_VECTOR, %o'2', 0, 4, %o'774', YES, 1); !Get RC25 Interrupt Vector address
: 1645 GPRMD (HW_Q3_BR, %o'4', 0, %o'177777', 0, 7, YES, 1); !Get RC25 Bus Request Priority
: 1646 GPRMD (HW_Q4_UNIT, %o'6', 0, %o'177777', 0, %decimal'253', YES, 1); !Get unit number to format
: 1647 ENDHRD:

```

CZRCH3
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
SOFTWARE PARAMETER CODING SECTION

11-Jul-1983 16:05:44
7-Jul-1983 08:24:19

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH3.B16;3 (6)

```

:      1648 %sbttl 'SOFTWARE PARAMETER CODING SECTION'
:      1649 !+
:      1650 ! The software parameter coding section contains macros
:      1651 ! that are used by the supervisor to build P-Tables. The
:      1652 ! macros are not executed as machine instructions but are
:      1653 ! interpreted by the supervisor as data structures. The
:      1654 ! macros allow the supervisor to establish communications
:      1655 ! with the operator.
:      1656 !-
:      1657 BGNSFT;
:      1658 GPRML (SW_Q1_UNATT, 0, 1, YES, 1);
:      1659 ENDSFT;

```


CZRCH3
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
REPORT CODING SECTION

11-Jul-1983 16:05:44
7-Jul-1983 08:24:19

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH3.B16;3 (7) SEQ 128
Page 8

```

:      1660 %sbttl 'REPORT CODING SECTION'
:      1661 |
:      1662 | The statistical report coding section contains the PRINTS macros that
:      1663 | will be used to generate statistical reports. The 'BGNRPT' and 'ENDRPT'
:      1664 | macros are used as begining and ending directives for the coding con-
:      1665 | tained in the section. However, an externally located DORPT call, or
:      1666 | a print command from the operator, may be used to request the execu-
:      1667 | tion of the report coding.
:      1668 | -
:      1669 BGNRPT;
:      1670 return;
:      1671 |
:      1672 | Report summary coding is remote program driven
:      1673 |
:      1674 ENDRPT;

```

```

.TITLE CZRCH3 CZRCH RC25 DISK FORMATTER
.IDENT /REV A /

```

```

000000          .PSECT  AA$CODE,  RO
000000      122      103      062      L$DESC::.ASCII /RC2/
000003      065      040      104      .ASCII /5 D/
000006      111      123      113      .ASCII /ISK/
000011      040      106      117      .ASCII / FO/
000014      122      115      101      .ASCII /RMA/
000017      124      124      105      .ASCII /TTE/
000022      122      000      .ASCII /R/<00>
000024      .BLKB      2
000026      122      103      062      L$DVTYP::
000031          .ASCII /RC2/
000034          .ASCII /5 D/
000037          .ASCII /ISK/
000042          .ASCII / DR/
000045          .ASCII /IVE/
000050          .ASCII / SU/
000053          .ASCII /BSY/
000056          .ASCII /STE/
000060          .ASCII /M/<00>
000062      000000C      .BLKB      2
000064          L$HRDLN::
000064      000031      GP$1::.WORD <<<L$NDHRD-L$HRDLN>/2>-1>
000066      000000G      .WORD      31
000070      016000      .WORD      HW.Q1.IP
000072      177777      .WORD      16000
000074      001031      GP$2::.WORD      -1
000076      000000G      .WORD      1031
000100      000004      .WORD      HW.Q2.VECTOR
000102      000774      .WORD      4
000104      002032      GP$3::.WORD      774
000106      000000G      .WORD      2032
000110      177777      .WORD      HW.Q3.BR
000112      000000      .WORD      -1
000114      000007      .WORD      0
000116      003052      GP$4::.WORD      7
          .WORD      3052

```

CZRCH3 CZRCH RC25 DISK FORMATTER
REV A PATCH 00 REPORT CODING SECTION

11-Jul-1983 16:05:44
7-Jul-1983 08:24:19

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH3.B16;3 (7)

```

000120 000000G .WORD HW.Q4.UNIT
000122 177777 .WORD -1
000124 000000 .WORD 0
000126 000375 .WORD 375
000130 L$NDHRD:: .BLKW 1
000132 000000C L$SFTLN:: .WORD <<<L$NDSFT-L$SFTLN>/2>-1>
000134 000130 GP$S:: .WORD 130
000136 000000G .WORD SW.Q1.UNATT
000140 000001 .WORD 1
000142 L$NDSFT:: .BLKW 1

.GLOBL DECODE, LOAD.FILE, AZFMTR, OVSA
.GLOBL HW.Q1.IP, HW.Q2.VECTOR, HW.Q3.BR
.GLOBL HW.Q4.UNIT, SW.Q1.UNATT, SW.Q2.NOTICE
.GLOBL SW.Q3.OPER, SW.Q4.UNATT, FMT2
.GLOBL FMT3, CRLF, XCRLF, NO.ADD.UNITS
.GLOBL PWR.MSG, ABO.MSG, TO.MANY.UNITS
.GLOBL GOOD.NUM.UNITS, DATMSG, FLG.WRD
.GLOBL LUN, PTBL.PTR, DATETXT, L$UNIT
.GLOBL RC25.ADDR, VEC.ADDR, BR.LEVEL
.GLOBL UNIT.NO, SW.UNATT, ISD.STRUCT

```

```

000064' L$HARD== L$HRDLN+2
000134' L$SOFT== L$SFTLN+2

```

```

000000 000207 LRPT: .SBTTL LRPT REPORT CODING SECTION ; 1659
RTS PC
: Routine Size: 1 word, Routine Base: AA$CODE + 0144
: Maximum stack depth per invocation: 0 words

```

```

000000 004767 177772 L$RPT:: .SBTTL L$RPT REPORT CODING SECTION ; 1670
000004 104425 JSR PC,LRPT
000006 000207 TRAP 25
RTS PC
: Routine Size: 4 words, Routine Base: AA$CODE + 0146
: Maximum stack depth per invocation: 2 words

```

CZRCH3
REV A PATCH 00CZRCH RC25 DISK FORMATTER
INITIALIZE SECTION11-Jul-1983 16:05:44
7-Jul-1983 08:24:19VAX-11 Bliss-16 V3-555
SPIDER\$USERS:[NEALE.AZTEC]CZRCH3.B16;3 (8)SEQ 130
Page 10

```

1675 %sbttl 'INITIALIZE SECTION'
1676 BGNINIT;
1677
1678 !++
1679 ! The initialization code is executed at the beginning of every
1680 ! sub-pass and is primarily used for requesting P Tables. Any
1681 ! other set-up type functions may also be performed in the init
1682 ! code.
1683
1684 ! The initialize code is executed under five conditions. There
1685 ! are supervisor event flags that are used to let the diagnostic
1686 ! know under which condition the execution is taking place. The
1687 ! event flags are read using the 'READEF' macro.
1688
1689 ! The conditions under which the init code is executed and the
1690 ! corresponding event flags are:
1691
1692             START COMMAND           EF.START
1693             RESTART COMMAND         EF.RESTART
1694             CONTINUE COMMAND        EF.CONTINUE
1695             POWERDOWN/POWERUP       EF.PWR
1696             NEW PASS                 EF.NEW
1697
1698             Example of event flag use:
1699
1700             if READEF(EF.START) then
1701                 START_FLAG = 1;
1702             --
1703
1704 !+
1705 ! First read the event flag EF_PWR to see if this init code is being
1706 ! performed due to a system power fail. If it is then report the
1707 ! incident to the operator and abort the DM machine and further execution
1708 ! of this program.
1709 !-
1710
1711 if READEF (EF_PWR)                !Is the PWR event flag set
1712 then
1713     begin                          !Report the incident and abort
1714     PRINTB (PWR_MSG);              !Power fail print message
1715     PRINTB (ABO_MSG);              !Aborting program message
1716     WRT_RC25 (RCIP, ONES);         !Abort DM code execution
1717     DOCLN;                          !Abort further program execution
1718     end;
1719
1720 !
1721 ! See if the DRS commands START or RESTART were used to start
1722 ! this formatting session. If either of them were used then
1723 ! start the formatting session at logical unit zero.
1724 !
1725
1726 if (READEF (EF_START)) or (READEF (EF_RESTART)) !Did start of restart get us here
1727 then
1728     begin
1729     !
1730     ! Check the operator for trying to format
1731     ! more than the defined limit. If they

```

CZRCH3
REV A PATCH 00CZRCH RC25 DISK FORMATTER
INITIALIZE SECTION11-Jul-1983 16:05:44
7-Jul-1983 08:24:19VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH3.B16;3 (8)

SEQ 131

Page 11

```

:      1732      | are then report the error and die.
:      1733      |
:      1734      |
:      1735      | if .LSUNIT gtru L$LIMIT      |Is the formatting limit exceeded
:      1736      | then
:      1737      |     begin
:      1738      |     PRINTB (TO_MANY_UNITS);  |Report the error
:      1739      |     PRINTB (GOOD_NUM_UNITS); |Tell him/her the limit
:      1740      |     DOCLN;                   |Go to cleanup and die
:      1741      |     end;
:      1742      |
:      1743      |
:      1744      | Test to see if this host code is to run in UNATTENDED REFORMAT MODE.
:      1745      | If it is then ask the operator for the date. If not then tell the
:      1746      | operator that an operator must be present during this formatting.
:      1747      |
:      1748      |
:      1749      | if .SW_UNATT
:      1750      | then
:      1751      |     begin
:      1752      |     |
:      1753      |     | Clear out the date text buffer before loading in date.
:      1754      |     |
:      1755      |     |
:      1756      |     | incr i from 0 to 11 do
:      1757      |     |     DATETXT [.i] = ZEROS;
:      1758      |     |
:      1759      |     | GMANID (DATMSG, DATETXT, A, %o'177777', 8, 10, NO);      |Get date from operator
:      1760      |     | PRINTB (SW_Q2_NOTICE);      |Print notice message
:      1761      |     | PRINTB (SW_Q4_UNATT);      |Report running in ATTENDED mode
:      1762      |     | end
:      1763      |     | else
:      1764      |     |     begin
:      1765      |     |     PRINTB (SW_Q2_NOTICE);  |Print notice message
:      1766      |     |     PRINTB (SW_Q3_OPER);    |Tell operator he/she must be present
:      1767      |     |     end;
:      1768      |     |
:      1769      |     | +
:      1770      |     | Everything looks good so far, move on and get the
:      1771      |     | hardware question responses and save them.
:      1772      |     | -
:      1773      |     |
:      1774      |     | LUN = -1;                   |Start formatting at logical unit 0
:      1775      |     |
:      1776      |     | do
:      1777      |     |     begin
:      1778      |     |     LUN = .LUN + 1;        |Up pointer to next logical unit to GPHARD
:      1779      |     |     if .LUN gequ .LSUNIT then DOCLN;      |End host code if no logical units
:      1780      |     |     end
:      1781      |     |
:      1782      |     | until (GPHARD (.LUN, PTBL_PTR)) nequ ZERO; |Repeat GPHARD's until P_Table adrs returned
:      1783      |     |
:      1784      |     |
:      1785      |     |
:      1786      |     | Get the P_Table parameters for this unit and go format it.
:      1787      |     |
:      1788      |     | RC25_ADDR = .PTBL_PTR [wrd0]; |Load up the controllers base address

```

CZRCH3
REV A PATCH 00CZRCH RC25 DISK FORMATTER
INITIALIZE SECTION11-Jul-1983 16:05:44
7-Jul-1983 08:24:19VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH3.B16;3 (8)SEQ 132
Page 12

```

:      1789      VEC_ADDR = .PTBL_PTR [wrđ1];           !Load up the controllers vector address
:      1790      BR [EVEL = .PTBL_PTR [wrđ2];         !Load up the controllers bus request
:      1791      UNIT_NO = .PTBL_PTR [wrđ3];          !Load up the unit number to format
:      1792
:      1793      !+
:      1794      ! Before leaving the init code section we must first do some house keeping
:      1795      ! left behind from the ISD_STRUCT preset declaration. The adapter purge interrupt
:      1796      ! bit must be defined for the type machine this formatter is running under.
:      1797      !-
:      1798
:      1799      if %bliss (bliss16)                       !Define compiler
:      1800      then
:      1801          ISD_STRUCT [BLK1, WRD1, S2W_PI] = ZERO !No purging fo PDP-11
:      1802      else
:      1803          ISD_STRUCT [BLK1, WRD1, S2W_PI] = ONE; !Purging for VAX-11
:      1804
:      1805      !++
:      1806      ! This commented code will load an external copy of the formatter from
:      1807      ! the local load media into azkel6 module space starting at AZFMTR : VECTOR [ .word]
:      1808
:      1809      !+
:      1810      ! Now load in the RC25 formatter DM code from the local boot device
:      1811      ! into the blank buffer allocated in azkel6. Report load error if
:      1812      ! an error code is returned.
:      1813
:      1814      ! The DM code will only be read in during start! or restarts of the
:      1815      ! host code. Block zero of the the DM .sav file will thrown out.
:      1816      !-
:      1817
:      1818      if LOAD_FILE (AZFMTR, DM_FN$EXT, DM_SIZE) then DECODE ();
:      1819
:      1820      !--
:      1821
:      1822      !
:      1823      ! Now calculate the overlay sections starting address and store the result
:      1824      ! in global location 'OVSA' for future reference.
:      1825
:      1826      ! This takes the DM buffer starting adrs and adds to it the number of bytes
:      1827      ! in the (initial load + remote program header size) resulting in the first
:      1828      ! adrs of the overlay section.
:      1829
:      1830      OVSA = AZFMTR + (.AZFMTR [WRD0]);           !Calculate overlay start adrs
:      1831      end
:      1832      else
:      1833          begin
:      1834              !
:      1835              ! The continue flag set will cause the current units formatting to be
:      1836              ! aborted. The next sequential logical unit will then be formatted.
:      1837              !
:      1838
:      1839              if READEF (EF_CONTINUE)
:      1840              then
:      1841                  begin
:      1842                      !
:      1843                      ! End this Host code if this .lun is the the last logical unit to format.
:      1844                      ! Else do another GPHARD macro to get the next logical unit to format.
:      1845

```

CZRCH3
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
INITIALIZE SECTION

11-Jul-1983 16:05:44
7-Jul-1983 08:24:19

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH3.B16;3 (8)

```

:      1846      PRINTB (FMT3, .LUN, .UNIT_NO);           !Report this luns format was aborted
:      1847      WRT_RC25 (RCIP, ONES);                   !Abort this current units formatting
:      1848
:      1849      if (.LUN + 1) gequ .L$UNIT
:      1850      then
:      1851          begin
:      1852              PRINTB (NO_ADD_UNITS);               !Report no more unit to format
:      1853              DOCLN;
:      1854              end;
:      1855
:      1856      end;
:      1857
:      1858      !
:      1859      ! The new pass event flag set means that all logical units have
:      1860      ! been formatted. Therefore end this host code and return to DS>.
:      1861      !
:      1862
:      1863      if READEF (EF_NEW) then DOCLN;                 !New pass means all units formatted
:      1864
:      1865      !
:      1866      ! Repeat doing GPHARDS until a non-zero value is returned or all logical units
:      1867      ! have been GPHARD'ed. End this host code if all logical units have been formatted.
:      1868      !
:      1869
:      1870      do
:      1871          begin
:      1872              LUN = .LUN + 1;                          !Up pointer to next logical unit
:      1873
:      1874              if .LUN gequ .L$UNIT then DOCLN;       !End the host if all logical units GPHARD'ed
:      1875
:      1876              end
:      1877          until (GPHARD (.LUN, PTBL_PTR)) nequ ZERO; !Repeat until a P_table adrs is returned
:      1878
:      1879      !
:      1880      ! Get the P_Table parameters for this unit and go format it.
:      1881      !
:      1882      RC25_ADDR = .PTBL_PTR [wrđ0];                   !Load up the controllers base adres
:      1883      VEC_ADDR = .PTBL_PTR [wrđ1];                   !Load up the controllers vector address
:      1884      BR_LEVEL = .PTBL_PTR [wrđ2];                   !Load up the controllers bus request
:      1885      UNIT_NO = .PTBL_PTR [wrđ3];                   !Load up the unit number to format
:      1886      end;
:      1887
:      1888      !
:      1889      ! Tell the operator which RC25 controller is being initialized this pass.
:      1890      !
:      1891      PRINTB (XCRLF);                                     !Space between each formatting pass
:      1892      PRINTB (FMT2, .PTBL_PTR [WRD0]);
:      1893      ENDINIT;

```

000000	012700	000034	LINIT:	.SBTTL	LINIT INITIALIZE SECTION		
000004	104447			MOV	#34,R0	;	1711
000006	103023			TRAP	47		
000010	012746	000000G		BHIS	1\$		
000014	012746	000001		MOV	#PWR.MSG,-(SP)	;	1714
000020	010600			MOV	#1,-(SP)		
				MOV	SP,R0	;	SP,*

CZRCH3
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
INITIALIZE SECTION

11-Jul-1983 16:05:44
7-Jul-1983 08:24:19

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH3.B16;3 (8)

000022	104414			TRAP	14				
000024	012716	000000G		MOV	#ABO.MSG,(SP)	:			1715
000030	012746	000001		MOV	#1,-(SP)	:			
000034	010600			MOV	SP,R0	:	SP,*		
000036	104414			TRAP	14				
000040	012700	177777		MOV	#-1,R0	:	*RCSM.REG		1716
000044	010077	000000G		MOV	R0,@RC25.ADDR	:	RCSM.REG,*		
000050	104444			TRAP	44				
000052	062706	000006		ADD	#6,SP	:			1713
000056	012700	000040	1\$:	MOV	#40,R0	:			1726
000062	104447			TRAP	47				
000064	103404			BCS	2\$				
000066	012700	000037		MOV	#37,R0				
000072	104447			TRAP	47				
000074	103151			BHIS	9\$				
000076	026727	000000G 000020	2\$:	CMP	LSUNIT,#20	:			1735
000104	101417			BLOS	3\$				
000106	012746	000000G		MOV	#TO.MANY.UNITS,-(SP)	:			1738
000112	012746	000001		MOV	#1,-(SP)	:			
000116	010600			MOV	SP,R0	:	SP,*		
000120	104414			TRAP	14				
000122	012716	000000G		MOV	#GOOD.NUM.UNITS,(SP)	:			1739
000126	012746	000001		MOV	#1,-(SP)	:			
000132	010600			MOV	SP,R0	:	SP,*		
000134	104414			TRAP	14				
000136	104444			TRAP	44				
000140	062706	000006		ADD	#6,SP	:			1737
000144	032767	000001 000000G	3\$:	BIT	#1,SW.UNATT	:			1749
000152	001434			BEQ	5\$				
000154	005000			CLR	R0	:	I		1756
000156	105060	000000G	4\$:	CLRB	DATETXT(R0)	:	*(I)		1757
000162	005200			INC	R0	:	I		1756
000164	020027	000013		CMP	R0,#13	:	I,*		
000170	003772			BLE	4\$				
000172	104443			TRAP	43	:			1759
000174	000406			.WORD	406				
000176	000000G			.WORD	DATETXT				
000200	000142			.WORD	142				
000202	000000G			.WORD	DATMSG				
000204	177777			.WORD	-1				
000206	000010			.WORD	10				
000210	000012			.WORD	12				
000212	012746	000000G		MOV	#SW.Q2.NOTICE,-(SP)	:			1760
000216	012746	000001		MOV	#1,-(SP)	:			
000222	010600			MOV	SP,R0	:	SP,*		
000224	104414			TRAP	14				
000226	012716	000000G		MOV	#SW.Q4.UNATT,(SP)	:			1761
000232	012746	000001		MOV	#1,-(SP)	:			
000236	010600			MOV	SP,R0	:	SP,*		
000240	104414			TRAP	14				
000242	000414			BR	6\$:			1749
000244	012746	000000G	5\$:	MOV	#SW.Q2.NOTICE,-(SP)	:			1765
000250	012746	000001		MOV	#1,-(SP)	:			
000254	010600			MOV	SP,R0	:	SP,*		
000256	104414			TRAP	14				
000260	012716	000000G		MOV	#SW.Q3.OPER,(SP)	:			1766
000264	012746	000001		MOV	#1,-(SP)	:			

CZRCH3
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
INITIALIZE SECTION

11-Jul-1983 16:05:44
7-Jul-1983 08:24:19

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH3.B16;3 (8)

000270	010600			MOV	SP,R0	:	SP,*	
000272	104414			TRAP	14	:		
000274	112767	000377	000000G	6\$:	MOVB	#377,LUN	:	1774
000302	105267	000000G		7\$:	INCB	LUN	:	1778
000306	005000				CLR	R0	:	1780
000310	156700	000000G			BISB	LUN,R0	:	
000314	020067	000000G			CMP	R0,LSUNIT	:	
000320	103401				BLO	8\$:	
000322	104444				TRAP	44	:	
000324	005000			8\$:	CLR	R0	:	1783
000326	156700	000000G			BISB	LUN,R0	:	
000332	104442				TRAP	42	:	
000334	010067	000000G			MOV	R0,PTBL.PTR	:	
000340	001760				BEQ	7\$:	
000342	011067	000000G			MOV	(R0),RC25.ADDR	:	1788
000346	016067	000002	000000G		MOV	2(R0),VEC.ADDR	:	1789
000354	016067	000004	000000G		MOV	4(R0),BR.LEVEL	:	1790
000362	016067	000006	000000G		MOV	6(R0),UNIT.NO	:	1791
000370	142767	000001	000006G		BICB	#1,ISD.STRUCT+6	:	1799
000376	016767	000000G	000000G		MOV	AZFMTR,OVSA	:	1830
000404	062767	000000G	000000G		ADD	#AZFMTR,OVSA	:	
000412	062706	000006			ADD	#6,SP	:	1728
000416	000504				BR	14\$:	1726
000420	012700	000036		9\$:	MOV	#36,R0	:	1839
000424	104447				TRAP	47	:	
000426	103040				BHIS	11\$:	
000430	016746	000000G			MOV	UNIT.NO,-(SP)	:	1846
000434	005046				CLR	-(SP)	:	
000436	116716	000000G			MOVB	LUN,(SP)	:	
000442	012746	000000G			MOV	#FMT3,-(SP)	:	
000446	012746	000003			MOV	#3,-(SP)	:	
000452	010600				MOV	SP,R0	:	SP,*
000454	104414				TRAP	14	:	
000456	012700	177777			MOV	#-1,R0	:	*,RC\$M.REG
000462	010077	000000G			MOV	R0,@RC25.ADDR	:	RC\$M.REG,*
000466	005000				CLR	R0	:	
000470	156700	000000G			BISB	LUN,R0	:	1849
000474	005200				INC	R0	:	
000476	020067	000000G			CMP	R0,LSUNIT	:	
000502	103410				BLO	10\$:	
000504	012716	000000G			MOV	#NO.ADD.UNITS,(SP)	:	1852
000510	012746	000001			MOV	#1,-(SP)	:	
000514	010600				MOV	SP,R0	:	SP,*
000516	104414				TRAP	14	:	
000520	104444				TRAP	44	:	
000522	005726				TST	(SP)+	:	1851
000524	062706	000010		10\$:	ADD	#10,SP	:	1841
000530	012700	000035		11\$:	MOV	#35,R0	:	1863
000534	104447				TRAP	47	:	
000536	103001				BHIS	12\$:	
000540	104444				TRAP	44	:	
000542	105267	000000G		12\$:	INCB	LUN	:	1872
000546	005000				CLR	R0	:	1874
000550	156700	000000G			BISB	LUN,R0	:	
000554	020067	000000G			CMP	R0,LSUNIT	:	
000560	103401				BLO	13\$:	
000562	104444				TRAP	44	:	

CZRCH3
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
INITIALIZE SECTION

11-Jul-1983 16:05:44
7-Jul-1983 08:24:19

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH3.B16;3 (8)

```

000564 005000          13$: CLR      R0                :                1877
000566 156700 000000G   BISB    LUN,R0              :
000572 104442          TRAP    42                :
000574 010067 000000G   MOV     R0,PTBL.PTR    :
000600 001760          BEQ     12$                :
000602 011067 000000G   MOV     (R0),RC25.ADDR :                1882
000606 016067 000002 000000G MOV     2(R0),VEC.ADDR  :                1883
000614 016067 000004 000000G MOV     4(R0),BR.LEVEL :                1884
000622 016067 000006 000000G MOV     6(R0),UNIT.NO  :                1885
000630 012746 000000G   14$: MOV     #XCRLF,-(SP)  :                1891
000634 012746 000001   MOV     #1,-(SP)        :
000640 010600          MOV     SP,R0          : SP,*
000642 104414          TRAP    14                :
000644 017716 000000G   MOV     @PTBL.PTR,(SP) :                1892
000650 012746 000000G   MOV     #FMT2,-(SP)    :
000654 012746 000002   MOV     #2,-(SP)        :
000660 010600          MOV     SP,R0          : SP,*
000662 104414          TRAP    14                :
000664 062706 000010   ADD     #10,SP          :
000670 000207          RTS     PC                :                1674

```

: Routine Size: 221 words, Routine Base: AASCODE + 0156
: Maximum stack depth per invocation: 7 words

```

000000 004767 177102          .SBTTL  LS$INIT INITIALIZE SECTION :                1892
000004 104411          LS$INIT::JSR PC,L$INIT :
000006 000207          TRAP    11                :
                                RTS     PC                :

```

: Routine Size: 4 words, Routine Base: AASCODE + 1050
: Maximum stack depth per invocation: 2 words

CZRCH3
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
AUTODROP SECTION

11-Jul-1983 16:05:44
7-Jul-1983 08:24:19

VAX-11 Bliss-16 V3-555
SPIDERSUSERS:[NEALE.AZTEC]CZRCH3.B16;3 (9)

```

:      1894 %sbttl 'AUTODROP SECTION'
:      1895 |
:      1896 | This code is executed immediately after the initialize code if
:      1897 | the 'ADR' flag was set. The unit(s) under test are checked to
:      1898 | see if they will respond. Those that don't are immediately
:      1899 | dropped from testing.
:      1900 |
:      1901 BGNAUTO;
:      1902 return;
:      1903 ENDAUTO;

```

```

000000 000207          LAUTO: .SBTTL LAUTO AUTODROP SECTION          ;          1893
                                RTS    PC
: Routine Size: 1 word,      Routine Base: AASCODE + 1060
: Maximum stack depth per invocation: 0 words

```

```

000000 004767 177772    LSAUTO: .SBTTL LSAUTO AUTODROP SECTION    ;          1902
000004 104461          :JSR    PC,LAUTO
000006 000207          TRAP   61
                                RTS    PC
: Routine Size: 4 words,      Routine Base: AASCODE + 1062
: Maximum stack depth per invocation: 2 words

```

CZRCH3
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
CLEANUP CODING SECTION

11-Jul-1983 16:05:44
7-Jul-1983 08:24:19

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH3.B16;3 (10)

```

:      1904 %sbttl 'CLEANUP CODING SECTION'
:      1905 |
:      1906 | Cleanup coding is assembled with the diagnostic program, utilizing in-
:      1907 | itiating (BGNCLN) and ending (ENDCLN) directives. The coding can be
:      1908 | used by either the diagnostic program or the supervisor to affect the
:      1909 | return of a test device to a static state.
:      1910 |
:      1911 | The clean-up code is invoked in three different ways:
:      1912 |
:      1913 |     A. At end of every sub-pass
:      1914 |     B. Issuance of DOCLN macro
:      1915 |     C. Operator ^C
:      1916 | -
:      1917 BGNCLN:
:      1918 |
:      1919 | Before initing the RC25 controller check first to make sure that
:      1920 | that this is not a non-existent register.
:      1921 |
:      1922 |
:      1923 if not .FLG_WRD [NON_EXIST_REG] then WRT_RC25 (RCIP, ONES);      !Abort DM code execution
:      1924 |
:      1925 return;
:      1926 ENDCLN;

```

```

000000 032767 000002 000000G      LCLEAN: .SBTTL  LCLEAN CLEANUP CODING SECTION      :      1923
000006 001004                    BIT      #2,FLG.WRD      :
000010 012700 177777              BNE      1$      :
000014 010077 000000G              MOV      #-1,R0      : *RCSM.REG
000020 000207                    MOV      R0,@RC25.ADDR      : RCSM.REG,*
:                                     RTS      PC      :
:                                     :      1903

```

```

; Routine Size: 9 words,      Routine Base: AASCODE + 1072
; Maximum stack depth per invocation: 0 words

```

```

000000 004767 177752      L$CLEAN: .SBTTL  L$CLEAN CLEANUP CODING SECTION      :      1925
000004 104412              JSR      PC,LCLEAN      :
000006 000207              TRAP     12      :
:                                     RTS      PC      :

```

```

; Routine Size: 4 words,      Routine Base: AASCODE + 1114
; Maximum stack depth per invocation: 2 words

```

CZRCH3
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
DROP UNIT SECTION

11-Jul-1983 16:05:44
7-Jul-1983 08:24:19

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH3.B16;3 (11)

```

:      1927 %sbttl 'DROP UNIT SECTION'
:      1928 |
:      1929 | *
:      1930 | The drop code is invoked by a DODU macro or a drop command, and con-
:      1931 | tains any code that needs to be executed in conjunction with the drop-
:      1932 | ping of a unit from the test cycle. No coding is required in this
:      1933 | section.
:      1934 |
:      1935 | The effect of a DODU is the same whether executed in the init code or
:      1936 | in a hardware test. It invokes the drop unit coding and causes subse-
:      1937 | quent GPHARD'S for that logical unit to be returned 'NOT COMPLETE'.
:      1938 | This effect lasts only for the duration of the current command.
:      1939 | -
:      1939 BGNDU;
:      1940 return;
:      1941 ENDDU;

```

```

000000 000207          LDU:      .SBTTL  LDU DROP UNIT SECTION          ;          1926
                                RTS      PC
; Routine Size: 1 word,      Routine Base: AASCODE + 1124
; Maximum stack depth per invocation: 0 words

```

```

000000 004767 177772    LSDU::   .SBTTL  LSDU DROP UNIT SECTION      ;          1940
000004 104453          JSR      PC,LDU
000006 000207          TRAP    53
                                RTS      PC
; Routine Size: 4 words,      Routine Base: AASCODE + 1126
; Maximum stack depth per invocation: 2 words

```

CZRCH3
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
ADD UNIT SECTION

11-Jul-1983 16:05:44
7-Jul-1983 08:24:19

VAX-11 Bliss-16 V3-555
SPIDERSUSERS:[NEALE.AZTEC]CZRCH3.B16;3 (12)

```

:      1942 %sbttl 'ADD UNIT SECTION'
:      1943 |
:      1944 | The add code is invoked by the ADD command, and contains any code that
:      1945 | needs to be executed in conjunction with adding a unit back to the
:      1946 | test cycle. No coding is required in this section.
:      1947 |
:      1948 | Units may be added to the test sequence only through the use of opera-
:      1949 | tor ADD command. Each unit must have a P-TABLE in memory due to an
:      1950 | earlier hardware dialogue (i.e. the unit was previously dropped).
:      1951 | The ADD code must be delimited by BGNAU, ENDAU. There is no particu-
:      1952 | lar coding required in the add code to cause the add to be effective:
:      1953 |
:      1954 | The section is just for programmer housekeeping.
:      1955 | -
:      1956 | BGNAU;
:      1957 | return;
:      1958 | ENDAU;

```

```

000000 000207          LAU:      .SBTTL  LAU ADD UNIT SECTION          ;          1941
                                RTS      PC

```

```

: Routine Size: 1 word,      Routine Base: AASCODE + 1136
: Maximum stack depth per invocation: 0 words

```

```

000000 004767 177772    LSAU::  .SBTTL  LSAU ADD UNIT SECTION          ;          1957
000004 104452          JSR      PC,LAU
000006 000207          TRAP    52
                                RTS      PC

```

```

: Routine Size: 4 words,      Routine Base: AASCODE + 1140
: Maximum stack depth per invocation: 2 words

```

```

:      1959 end
:      1960
:      1961 eludom

```

PSECT SUMMARY

```

:      Psect Name      Words      Attributes
:      AASCODE         308         R0 , I , LCL, REL, CON

```

LIBRARY STATISTICS

```

:      File              ----- Symbols -----      Blocks
:                        Total   Loaded   Percent      Read
:
:      SPIDERSUSERS:[NEALE.AZTEC]CZRCH0.L16;5

```

L 11

CZRCH3 CZRCH RC25 DISK FORMATTER
REV A PATCH 00 ADD UNIT SECTION

11-Jul-1983 16:05:44
7-Jul-1983 08:24:19

VAX-11 Bliss-16 V3-555
SPIDERSUSERS:[NEALE.AZTEC]CZRCH3.B16;3 (12)

SEQ 141
Page 21

: 398 62 15 24

COMMAND QUALIFIERS

: BLISS /PDP11/LIST CZRCH3.B16

: Size: 258 code + 50 data words
: Run Time: 00:17.0
: Elapsed Time: 00:23.5
: Memory Used: 214 pages
: Compilation Complete

CZRCH4

CZRCH RC25 DISK FORMATTER

M 11

11-Jul-1983 16:06:09
7-Jul-1983 08:24:38

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH4.B16;2 (1)

SEQ 142

Page 1

```
0001 MODULE CZRCH4 (%TITLE 'CZRCH RC25 DISK FORMATTER'  
0002 IDENT = 'REV A PATCH 00'  
0003 ADDRESSING MODE (RELATIVE) ,  
0004 ENVIRONMENT (NOEIS)  
0005 ) =  
0006 BEGIN  
0007 %sbttl 'MODULE DECLARATIONS'  
0008 :  
0009 : Pretty Declarations  
0010 :  
0011 : <blf/lowercase_key>  
0012 :  
0013 :  
0014 Library 'CZRCH0'; !Define RC25 formatter library  
0015 :  
0016 require 'BLSMAC.REQ'; !Define Bliss Macro require file  
1505 :  
1506 :+  
1507 : The psect named "code or $code$" is redefined here  
1508 : to be called "ab$code". This is done to organize the  
1509 : formatters test sections into a separate psect.  
1510 :-  
1511 psect  
1512 code = ab$code;  
1513 :  
1514 :+  
1515 : Structure declarations used within this module.  
1516 :-  
1517 :  
1518 structure  
1519 :  
1520 :+  
1521 : RC25 register accessing structure. This  
1522 : structure allows RC25 register accessing  
1523 : to be transportable between the PDP-11 and  
1524 : VAX Diagnostic Supervisors.  
1525 :  
1526 : This also defines an access algorithm for  
1527 : VAX to allow field reference to MBA address  
1528 : space without generating machine checks.  
1529 :-  
1530 :  
1531 RC25 [O, P, S, E] =  
1532 begin  
1533 local  
1534 RC$$_REG;  
1535 :  
1536 RC$$_REG = .(RC25 + %upval*0)<0, %bpval, 0>;  
1537 RC$$_REG  
1538 end  
1539 <P, S, E>;  
1540 :  
1541 :  
1542 !<blf/page>
```

CZRCH4 CZRCH RC25 DISK FORMATTER
REV A PATCH 00 MODULE DECLARATIONS

11-Jul-1983 16:06:09
7-Jul-1983 08:24:38

```

:      1543  !+
:      1544  ! External Routines declared outside this module.
:      1545  !-
:      1546
:      1547  external routine
:      1548      ABORT,
:      1549      GET_DUST_STATUS,
:      1550      EX_SUP_PROG,
:      1551      EX_LOC_PROG,
:      1552      REC_DATA,
:      1553      SEND_DATA,
:      1554      SET_CNTRL_CHAR,
:      1555      ON_LINE,
:      1556      DUPSI_SERVICE : INT_LNKSTYP novalue,
:      1557      INTSI_SERVICE : INT_LNKSTYP novalue,
:      1558      INIT_COM_AREA,
:      1559      BOOT_RC25,
:      1560      DECODE : novalue;
:      1561
:      1562  !<blf/page>

```


CZRCH4
REV A PATCH 00CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS11-Jul-1983 16:06:09
7-Jul-1983 08:24:38VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH4.B16;2 (3)SEQ 144
Page 3

```

1563 !+
1564 ! External Declaration of datums declared outside of this module.
1565 !-
1566
1567 external
1568
1569 ! Micellaneous external data declarations
1570
1571 SW_UNATT, !Unattended reformat mode flag
1572 DATETXT : vector [12, byte], !Stores operator date input
1573 LUN : word, !Logical unit number storage
1574 UNIT_NO : word, !Physical unit number storage
1575 FLG_WRD : bitvector [16], !Global flag word
1576 ISD_STRUCT : blockvector [4, 2, word] field (ISD_FIELD), !Init sequence data
1577 PTBL_PTR : ref vector [4, word] volatile,
1578 NEX_FLAG : word, !RC25 register existence flag
1579 NXT_CRN : byte, !Next seq command ref number
1580 RET_EN$AD : ref block [RB_SIZE + 2, word] field (ENV_FIELD),
1581 FCT_BUF : block [256, word], !External FCT sector load buffer
1582 SND_BUF : vector [SNDB_SIZE, word], !DUP send cmd text buffer
1583 REC_BUF : block [RECB_SIZE, word] field (RECB_FIELD), !DUP receive cmd text buffer
1584 MSGADR, !Pointer to DM sent ascii text
1585 NSD_SLOT : word, !Stores next send ring slot to load cmd into
1586 NRD_SLOT : word, !Stores next receive slot to expect response in
1587 VEC_ADDR : word, !Stores controllers vector address
1588 RET_STATUS : word, !Stores return status of called routines
1589 PID_SAVE : vector [2, word], !Saves process indicator word
1590 RC25_ADDR : ref RC25 field (ISD_FIELD), !RC25 reference structure
1591
1592 ! Init sequence code error
1593
1594 BOOT_FAILURE,
1595 PROT0_VIOLATION,
1596 PORT_INIT_ERR,
1597 ACTIVE_DUP_SERVER,
1598 INACTIVE_DUP_SERVER,
1599
1600 ! External FCT request message during RESTORE formatting mode
1601
1602 FCT_REQ_MSG,
1603
1604 ! Printing format strings and ascii text strings
1605
1606 DATMSG,
1607 CRLF,
1608 FMT1,
1609 FMT5,
1610 FMT6,
1611 FMT7;
1612

```

CZRCH4
REV A PATCH 00CZRCH RC25 DISK FORMATTER
RC25 REGISTER EXISTENCE TEST11-Jul-1983 16:06:09
7-Jul-1983 08:24:38VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH4.B16;2 (4)SEQ 145
Page 4

```

:      1613 %sbttl 'RC25 REGISTER EXISTENCE TEST'
:      1614 BGNTST;
:      1615
:      1616 !++
:      1617 ! Functional Description :
:      1618 ! This section will be the first code to be executed during a formatting
:      1619 ! session and will ensure that the object drive to format actually does
:      1620 ! exist. The format for the unit will be aborted if the drive fails to
:      1621 ! respond to this initial test.
:      1622
:      1623 ! Implicit Inputs :
:      1624 ! none
:      1625
:      1626 ! Implicit Outputs :
:      1627 ! none
:      1628
:      1629 ! Completion Codes :
:      1630 ! none
:      1631
:      1632 ! Side Effects :
:      1633 ! none
:      1634 !--
:      1635
:      1636 local
:      1637     EOF,                                !EOF indicator
:      1638     DUMMY : volatile,                 !Dummy variable for register ext test
:      1639     TEMP : volatile,
:      1640     RETRIES;                          !Store number of retries to perform
:      1641
:      1642 FLG_WRD [NON_EXIST_REG] = ZERO;    !Initially clear nonexistent register flag
:      1643 NEX_FLAG = ZEROS;                   !Clear out nex flag
:      1644 SETVEC (4, INT$I_SERVICE, PRI07);  !Set up for an nex trap
:      1645 TEMP = .PTBL_PTR [WRD0];         !Get this controllers IP address
:      1646 TEMP = .TEMP + 2;                !Make it into the SA register address
:      1647
:      1648 if ..TEMP
:      1649 then
:      1650     begin
:      1651         DUMMY = ZEROS;
:      1652     end;
:      1653
:      1654 CLRVEC (4);                          !correctly.
:      1655
:      1656 if .NEX_FLAG eql ONES
:      1657 then
:      1658     begin
:      1659         PRINTB (FMT6, .PTBL_PTR [WRD0]); !Print error message
:      1660         FLG_WRD [NON_EXIST_REG] = ONE; !Set flag indicating non-existent reg
:      1661         DOCIN;
:      1662     end;
:      1663

```

CZRCH4
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
INITIALIZE RC25 CONTROLLER

11-Jul-1983 16:06:09
7-Jul-1983 08:24:38

VAX-11 Bliss-16 V3-555
SPIDER\$USERS:[NEALE.AZTEC]CZRCH4.B16;2 (5)

```

1664 %sbttl 'INITIALIZE RC25 CONTROLLER'
1665 !+
1666 ! The next thing to be done is to boot the RC25 controller. We will allow
1667 ! a few retries if not successful after the first boot before we considered
1668 ! the Controller dead.
1669
1670 ! But before we do the boot sequence the processors priority and interrupt
1671 ! vector address must first be loaded. During the init sequence the init
1672 ! sequence interrupt service routine will just flag any interrupts and ignore
1673 ! them since interrupts are disabled during init sequence. Later the DUP
1674 ! interrupt service routine will be load and do the DUP communications protocol.
1675
1676 ! The following priorities will be assigned:
1677 ! 1. Processor will run at priority zero.
1678 ! 2. The RC25 runs at priority 5 by default.
1679 ! 3. The DUP interrupt routine will run at priority 7.
1680 !-
1681 SETPRI (PRI00); !Set the processors priority
1682 CLRVEC (.VEC_ADDR); !Clear out the vector before setting
1683 SETVEC (.VEC_ADDR, INT$I_SERVICE, PRI07); !Set the interrupt service priority
1684 !
1685 !+
1686 ! Retry the RC25 booting until the return is true or the retry limit is reached.
1687 !-
1688 RETRIES = -1; !Reset the retry counter
1689
1690 do
1691     begin
1692     RET_STATUS = BOOT_RC25 (); !Boot the RC25 controller
1693     RETRIES = .RETRIES + 1; !Up the retry count
1694     end
1695 until (.RET_STATUS) or (.RETRIES eqlu ONE); !Repeat the Boot until done
1696
1697 !
1698 ! Report booting error if the return status never came back as true.
1699 !
1700
1701 if not .RET_STATUS !Did the Controller boot
1702 then
1703     begin !Report a boot error
1704     PRINTB (BOOT_FAILURE);
1705     DOCLN; !Abort the program
1706     end;
1707
1708 !+
1709 ! Now that the RC25 controller is booted, check to make sure that the controller
1710 ! has done its part of the DUP protocol by clearing out the port communications area.
1711 !
1712 ! While we're there set up the communication area for the up and coming communications
1713 ! between the remote and host program.
1714 !-
1715
1716 if INIT_COM_AREA () !Was the com area cleared out as expected
1717 then
1718     begin !Com area not initied so error and abort
1719     PRINTB (PROTO_VIOLATION);
1720     PRINTB (PORT_INIT_ERR);

```

CZRCH4
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
INITIALIZE RC25 CONTROLLER

11-Jul-1983 16:06:09
7-Jul-1983 08:24:38

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH4.B16;2 (5)

```

:      1721      DOCLN;
:      1722      end;
:      1723
:      1724      |
:      1725      | Before writing the go to the IP register load the DUP interrupt service adrs
:      1726      | into the RC25 vector address for the up and coming communications between
:      1727      | the host and controller.
:      1728      |
:      1729      CLRVEC (.VEC_ADDR);                !Clear out the vector before starting
:      1730      SETVEC (.VEC_ADDR, DUP$I_SERVICE, PRI07);    !Set the interrupt service priority
:      1731      |
:      1732      | Start the controllers functional micro code by writing the SA Register
:      1733      | with step four write data with the go bit set.
:      1734      |
:      1735      WRT_RC25 (RCSA, (.ISD_STRUCT [BLK3, ISWRT, ISW_ALL] or %o'000001'));    !Let the controller go

```

CZRCH4
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
BRING RC25 CONTROLLER ONLINE

11-Jul-1983 16:06:09
7-Jul-1983 08:24:38

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH4.B16;2 (6)

```

:      1736 %sbttl 'BRING RC25 CONTROLLER ONLINE'
:      1737
:      1738 !++
:      1739 ! Functional Description :
:      1740 !   This section will set the RC25 controllers characteristics and place
:      1741 !   the unit on-line.
:      1742
:      1743 ! Implicit Inputs :
:      1744 !   none
:      1745 ! Implicit Outputs :
:      1746 !   none
:      1747 ! Completion Codes :
:      1748 !   none
:      1749 ! Side Effects :
:      1750 !   none
:      1751 !--
:      1752
:      1753 !+
:      1754 ! Set the controller characteristics. Default values will be taken in all cases
:      1755 ! except for the host time out value which will be changed to 'wait for ever'.
:      1756 !-
:      1757
:      1758 if SET_CNTLR_CHAR () then DECODE ();           !Call decode if not successful
:      1759
:      1760 !+
:      1761 ! Do a unit on line cmd which will spin up the device and load the heads. The
:      1762 ! unit to be placed on line will be the last unit number received from the 'gphard'
:      1763 ! macro in the init code.
:      1764 !-
:      1765 PRINTB (FMT5, .LUN, .UNIT_NO);               !Tell operator which units coming online
:      1766 !
:      1767
:      1768 if ON_LINE () then DECODE ();                 !Call decode if not successful
:      1769

```

CZRCH4
REV A PATCH 00CZRCH RC25 DISK FORMATTER
DOWN LINE LOAD FORMATTER AND MONITOR TASK11-Jul-1983 16:06:09
7-Jul-1983 08:24:38VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH4.B16;2 (7)SEQ 149
Page 8

```

:      1770 %sbttl 'DOWN LINE LOAD FORMATTER AND MONITOR TASK'
:      1771
:      1772 |++
:      1773 | Functional Description :
:      1774 |     This section will down line-load the RC25 formatter and monitor
:      1775 |     the formatting task.
:      1776 |
:      1777 | Implicit Inputs :
:      1778 |     none
:      1779 | Implicit Outputs :
:      1780 |     none
:      1781 | Completion Codes :
:      1782 |     none
:      1783 | Side Effects :
:      1784 |     none
:      1785 | --
:      1786 |
:      1787 |
:      1788 | See if the Dup server in the RC25 Controller is in an
:      1789 | idle state. To do this first get the dust status and
:      1790 | then look at the flag field bit 3 for a :
:      1791 |     0 = idle
:      1792 |     1 = active
:      1793 |
:      1794 |
:      1795 | if GET_DUST_STATUS () then DECODE ();           !Call Decode if connection error
:      1796 |
:      1797 |
:      1798 | Look in the flag field bit 3 to see if the server is active.
:      1799 |
:      1800 |
:      1801 | if .RET_EN$AD [FLG_B3]                           !If the server is active exit and reboot
:      1802 | then
:      1803 |     begin
:      1804 |         PRINTB (ACTIVE_DUP_SERVER);
:      1805 |         DOCLN;
:      1806 |     end;
:      1807 |
:      1808 |
:      1809 | The server is not active so down line load the formatter
:      1810 | and start its execution by issuing a 'Execute supplied program'.
:      1811 | Call the decode routine if a connection error is detected.
:      1812 |
:      1813 |
:      1814 | if EX_SUP_PROG () then DECODE ();           !Call decode if connection error
:      1815 |
:      1816 |
:      1817 | Get the dust status to see if the server is in an active
:      1818 | state. An active state is what we want so error if the
:      1819 | server is in an idle state.
:      1820 |
:      1821 | If in the active state then save the progress indicator
:      1822 | in 'Pid_save' for future reference.
:      1823 |
:      1824 |
:      1825 | if GET_DUST_STATUS () then DECODE ();           !Call decode if connection error
:      1826 |

```

```

1827 |
1828 | Look at the flag field bit 3 to see if the server is active.
1829 |
1830 |
1831 | if not (.RET_EN$AD [FLG_B3])           !Reboot if server is idle
1832 | then
1833 |     begin
1834 |     PRINTB (INACTIVE_DUP_SERVER);
1835 |     DOCLN;
1836 |     end
1837 | else
1838 |     begin
1839 |     PID_SAVE [0] = .RET_EN$AD [PLO_IND];       !Save progress indicator lo word
1840 |     PID_SAVE [1] = .RET_EN$AD [PHI_IND];       !Save progress indicator hi word
1841 |     end;
1842 |
1843 | +
1844 | The Dup server is in the active state running the formatter
1845 | program. This DO LOOP will loop on the DUP sub-protocol
1846 | doing the "send and receive" data commands. These commands
1847 | establish the communications between this host program and
1848 | the remote formatter program running in the RC25 controller.
1849 | -
1850 | RETRIES = ZERO;                       !Clear out the retry flag
1851 |
1852 | while TRUE do
1853 |     begin
1854 |
1855 |     +
1856 |     Do a 'Receive_data' command which poll's the remote program
1857 |     for a message. The returned message can either be a:
1858 |
1859 |     1. Question
1860 |         Where the ascii text is a prompt for information.
1861 |
1862 |     2. Default question
1863 |         Where the default question message is identical
1864 |         to the question message except that a null (zero
1865 |         length) send data is taken to be a default answer
1866 |         to the question.
1867 |
1868 |     3. Information
1869 |         Where the ascii text is an informative message.
1870 |
1871 |     4. Termination
1872 |         Where the ascii text is an normal termination message.
1873 |
1874 |     5. Fatal Error
1875 |         Where the ascii text is a fatal error message.
1876 |
1877 |     6. Special
1878 |         This type is used when only a host program could
1879 |         respond.
1880 |     -
1881 |
1882 |     if REC_DATA () then DECODE ();         !Call decode if connection error
1883 |

```

CZRCH4
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
DOWN LINE LOAD FORMATTER AND MONITOR TASK

11-Jul-1983 16:06:09
7-Jul-1983 08:24:38

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH4.B16;2 (7)

SEQ 151
Page 10

```

1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940

```

```

: From the first word in the send/receive data buffer, look
: to see what type message the remote program has sent to
: the Host program. Use this message type number to index
: into the select expression to perform the requested action
: by the remote program.
:
SCLRSBUF:
selectoneu .REC_BUF [MSG_TYP] of
set
[1] :
begin
: Look into the send/receive data buffer at the message
: number field and see what question the remote program
: is asking. Use the fields value to index into the
: select expression to perform the appropriate action.
:
selectoneu .REC_BUF [MSG_NUM] of
set
[0] :
begin
: If in un-attended reformat mode then give to the DM the
: date received from the init code else let the DM prompt
: the operator for the date.
:
if .SW_UNATT
then
begin
if .RETRIES
then
begin
: Clear out the date text buffer before loading in
: date and ask the operator for the date again.
:
incr i from 0 to 11 do
DATETXT [.i] = ZEROS;
GMANID (DATMSG, DATETXT, A, %'177777', 8, 10, NO); !Get date from operator
end;
$FILLDATE;
RETRIES = ONE;
end
else
begin
GMANID (MSGADR, SND_BUF, A, %'177777', 8, 10, NO);

```

!Clear out the send buffer area

!Select the appropriate action

!Question message type

!Select the requested question

!Enter current date <MM-DD-YYYY>

!Is the host running in unattended mode

!Did DM not verify the date the last ime

!Fill the send buffer with the date

!Set the retry flag

CZRCH4
REV A PATCH 00CZRCH RC25 DISK FORMATTER
DOWN LINE LOAD FORMATTER AND MONITOR TASK11-Jul-1983 16:06:09
7-Jul-1983 08:24:38VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH4.B16;2 (7)SEQ 152
Page 11

```

1941         end;
1942
1943         PRINTB (CRLF);
1944
1945         if SEND_DATA () then DECODE ();
1946
1947         end;
1948
1949     [7] :           !Enter a non-zero serial number
1950     begin
1951     GMANID (MSGADR, SND_BUF, A, %o'177777', 1, 64, NO);
1952     PRINTB (CRLF);
1953
1954     if SEND_DATA () then DECODE ();
1955
1956     end;
1957
1958     [otherwise] :   !This message number is unknown
1959     begin
1960     RET_STATUS = UMN_CODE;   !Unknown message number error code
1961     DECODE ();             !Report error and die
1962     end;
1963     tes;
1964
1965     end;
1966
1967     [2] :           !Default Question
1968     begin
1969
1970     selectoneu .REC_BUF [MSG_NUM] of
1971     set
1972
1973     [1] :           !Enter unit number to format <0>
1974     begin
1975     |
1976     | If in un-attended reformat mode then give to the DM the
1977     | unit number obtained from the hardware P Table else let
1978     | the DM prompt the operator for the unit to format.
1979     |
1980
1981     if .SW_UNATT           !Is the host running in unattended mode
1982     then
1983     begin
1984     SND_BUF = .UNIT_NO;
1985     PRINTB (FMT7, .UNIT_NO);   !Report which unit is being formatted
1986     end
1987     else
1988     begin
1989     GMANID (MSGADR, SND_BUF, A, %o'177777', 1, 3, YES);
1990     end;
1991
1992     PRINTB (CRLF);
1993
1994     if SEND_DATA () then DECODE ();
1995
1996     end;
1997

```

CZRCH4
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
DOWN LINE LOAD FORMATTER AND MONITOR TASK

11-Jul-1983 16:06:09
7-Jul-1983 08:24:38

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH4.B16;2 (7)

```

:      1998      [4] :      !Use existing bad block information <N>
:      1999      begin
:      2000      |
:      2001      |   If in un-attended reformat mode then tell the DM to
:      2002      |   do a reformat mode else let the DM prompt the operator
:      2003      |   for the formatting mode to use for this unit.
:      2004      |
:      2005      |
:      2006      |   if .SW_UNATT      !Is the host running in unattended mode
:      2007      |   then
:      2008      |       begin
:      2009      |       SND_BUF = %c'Y';      !Default to do reformat mode
:      2010      |       end
:      2011      |   else
:      2012      |       begin
:      2013      |       GMANID (MSGADR, SND_BUF, A, %o'177777', 1, 3, YES);
:      2014      |       end;
:      2015      |
:      2016      |   PRINTB (CRLF);
:      2017      |
:      2018      |   if SEND_DATA () then DECODE ();
:      2019      |
:      2020      |   end;
:      2021      |
:      2022      |   [5] :      !Use down-line load <N>
:      2023      |   begin
:      2024      |   GMANID (MSGADR, SND_BUF, A, %o'177777', 1, 3, YES);
:      2025      |   PRINTB (CRLF);
:      2026      |
:      2027      |   if SEND_DATA () then DECODE ();
:      2028      |
:      2029      |   end;
:      2030      |
:      2031      |   [6] :      !Continue if bad block information is inaccessible <N>
:      2032      |   begin
:      2033      |
:      2034      |   if .SW_UNATT      !Is the host running in unattended mode
:      2035      |   then
:      2036      |       begin
:      2037      |       SND_BUF = %c'N';      !Default not to continue
:      2038      |       end
:      2039      |   else
:      2040      |       begin
:      2041      |       GMANID (MSGADR, SND_BUF, A, %o'177777', 1, 3, YES);
:      2042      |       end;
:      2043      |
:      2044      |   PRINTB (CRLF);
:      2045      |
:      2046      |   if SEND_DATA () then DECODE ();
:      2047      |
:      2048      |   end;
:      2049      |
:      2050      |   [otherwise] :      !This message number is unknown
:      2051      |   begin
:      2052      |   RET STATUS = UMN_CODE;      !Unknown message number error code
:      2053      |   DECODE ();      !Report error and die
:      2054      |   end;

```

CZRCH4
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
DOWN LINE LOAD FORMATTER AND MONITOR TASK

L 12

11-Jul-1983 16:06:09
7-Jul-1983 08:24:38

VAX-11 Bliss-16 V3-555
SPIDER\$USERS:[NEALE.AZTEC]CZRCH4.B16;2 (7)

SEQ 154
Page 13

```
2055         tes;
2056
2057     end;
2058
2059     [3] :           !Informational message
2060     begin
2061
2062     selectoneu .REC_BUF [MSG_NUM] of
2063     set
2064
2065         [0, 1, 2, 3, 4, 5, 7, 9, 11, 14, 16] : !All possible information msg's
2066         begin
2067             PRINTB (FMT1, MSGADR);
2068         end;
2069
2070     [otherwise] :           !This message number is unknown
2071     begin
2072         RET STATUS = UMN_CODE;           !Unknown message number error code
2073         DECODE ();                       !Report error and die
2074     end;
2075     tes;
2076
2077     end;
2078
2079     [4] :           !Termination message
2080     begin
2081
2082     selectoneu .REC_BUF [MSG_NUM] of
2083     set
2084
2085         [12, 13] :           !Abort and normal termination type
2086         begin
2087             PRINTB (FMT1, MSGADR);
2088             WRT_RC25 (RCIP, ONES);       !Stop the remote program
2089             exitloop;                   !Return to init code
2090         end;
2091
2092     [otherwise] :           !This message number is unknown
2093     begin
2094         RET STATUS = UMN_CODE;           !Unknown message number error code
2095         DECODE ();                       !Report error and die
2096     end;
2097     tes;
2098
2099     end;
2100
2101     [5] :           !Fatal error message
2102     begin
2103
2104     selectoneu .REC_BUF [MSG_NUM] of
2105     set
2106
2107         [1 to 24] :           !Error messages are from 1 to 24
2108         begin
2109             PRINTB (FMT1, MSGADR);
2110             WRT_RC25 (RCIP, ONES);       !Stop the remote program
2111             DOC[N];                     !Kill this formatter and return to init code
```

```
2112         end;
2113
2114         [otherwise] :           !This message number is unknown
2115         begin
2116         RET_STATUS = UMN_CODE;   !Unknown message number error code
2117         DECODE ();              !Report error and die
2118         end;
2119     tes;
2120
2121 end;
2122
2123 [6] :                               !Special message type
2124 begin
2125
2126     selectoneu .REC_BUF [MSG_NUM] of
2127     set
2128
2129     [1] :                               !FCT sector request
2130     begin
2131     |
2132     | If this is the first FCT sector request (sector 0) then ask
2133     | the operator for the external FCT file name and open the file.
2134     | The supervisor will error and return to DRS> monitor if the
2135     | requested file is not found on the local load media.
2136     |
2137     |
2138     if .REC_BUF [MSG_TXT] eql FCT_SEC_0
2139     then
2140     begin
2141     GMANID (FCT_REQ_MSG, SND_BUF, A, %'177777', 1, 10, NO);
2142     OPEN (SND_BUF);
2143     end;
2144
2145     |
2146     | Load into the FCT_BUF the next sequential 256 FCT sector words
2147     |
2148     |
2149     incr FCT_WRD from FCT_BUF to FCT_BUF + %'776' by %'2' do
2150     begin
2151     EOF = GETWORD (.FCT_WRD);
2152     end;
2153
2154     |
2155     | Test to see if the EOF indicator was returned from the GETWORD macro
2156     | before 16 FCT sectors were read indicating an illegal FCT file format.
2157     |
2158     | End of file is reached when .EOF = 0;
2159     |
2160     |
2161     if ((.EOF eql ZERO) and (.REC_BUF [MSG_TXT] lss FCT_SEC_15)) or (.REC_BUF [MSG_TXT] gtr
2162     FCT_SEC_15)
2163     then
2164     begin
2165     SND_BUF [0] = ONES;           !Indicate failure to get good FCT file
2166
2167     if SEND_DATA () then DECODE (); !Send failure to DM
2168
```

CZRCH4
REV A PATCH 00CZRCH RC25 DISK FORMATTER
DOWN LINE LOAD FORMATTER AND MONITOR TASK11-Jul-1983 16:06:09
7-Jul-1983 08:24:38VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH4.B16;2 (7)

```

:      2169          RET STATUS = ILL_FCT;      !Host will also report the failure
:      2170          DECODE ();                !Error out and die
:      2171          end;
:      2172
:      2173          |
:      2174          | Return to the remote program a success indicator and the
:      2175          | unibus lo and hi adrs of where this FCT sector can be found
:      2176          | and send the information back to the remote program.
:      2177          |
:      2178          SND_BUF [0] = ZERO;            !Indicate success
:      2179          SND_BUF [1] = FCT_BUF;      !Lo unibus adrs of FCT sector
:      2180          SND_BUF [2] = ZERO;        !Hi unibus adrs of Fct sector
:      2181
:      2182          if SEND_DATA () then DECODE ();      !Send reply back to DM
:      2183
:      2184          |
:      2185          | Close this External FCT file if this is FCT sector 15.
:      2186          |
:      2187          |
:      2188          if .REC_BUF [MSG_TXT] eql FCT_SEC_1, then CLOSE;
:      2189
:      2190          end;
:      2191
:      2192          [otherwise] :
:      2193          begin
:      2194          RET STATUS = UMN_CODE;
:      2195          DECODE ();
:      2196          end;
:      2197          tes;
:      2198
:      2199          end;
:      2200
:      2201          [otherwise] :                          !This message number is unknown
:      2202          begin
:      2203          RET STATUS = UMT_CODE;                  !Unknown message number error code
:      2204          DECODE ();                          !Report error and die
:      2205          end;
:      2206          tes;
:      2207
:      2208          end;
:      2209
:      2210          ENDTST;

```

```

.TITLE CZRCH4 CZRCH RC25 DISK FORMATTER
.IDENT /REV A /

```

```

.GLOBL ABORT, GET.DUST.STATUS, EX.SUP.PROG
.GLOBL FX.LOC.PROG, REC.DATA, SEND.DATA
.GLOBL SET.CNTRL.CHAR, ON.LINE, DUP$I.SERVICE
.GLOBL INT$I.SERVICE, INIT.COM.AREA, BOOT.RC25
.GLOBL DECODE, SW.UNATT, DATETXT, LUN
.GLOBL UNIT.NO, FLG.WRD, ISD.STRUCT, PTBL.PTR
.GLOBL NEX.FLAG, NXT.CRN, RET.ENSAD, FCT.BUF
.GLOBL SND.BUF, REC.BUF, MSGADR, NSD.SLOT
.GLOBL NRD.SLOT, VEC.ADDR, RET.STATUS
.GLOBL PID.SAVE, RC25.ADDR, BOOT.FAILURE

```

CZRCH4
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
DOWN LINE LOAD FORMATTER AND MONITOR TASK

11-Jul-1983 16:06:09
7-Jul-1983 08:24:38

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH4.B16;2 (7)

.GLOBL PROTO.VIOLATION, PORT.INIT.ERR
.GLOBL ACTIVE.DUP.SERVER, INACTIVE.DUP.SERVER
.GLOBL FCT.REQ.MSG, DATMSG, CRLF, FMT1
.GLOBL FMT5, FMT6, FMT7

.SBTTL \$T1 DOWN LINE LOAD FORMATTER AND MONITOR TASK
.PSECT ABS\$CODE, RO

000000									
000000	004167	000000G		\$T1:	JSR	R1,\$SAVE3	:		1611
000004	024646				CMF	-(SP),-(SP)	:		
000006	142767	000002	000000G		BICB	#2,FLG.WRD	:		1642
000014	005067	000000G			CLR	NEX.FLAG	:		1643
000020	012746	000340			MOV	#340,-(SP)	:		1644
000024	012746	000000G			MOV	#INT\$I.SERVICE,-(SP)	:		
000030	012746	000004			MOV	#4,-(SP)	:		
000034	012746	000003			MOV	#3,-(SP)	:		
000040	104437				TRAP	37	:		
000042	017766	000000G	000010		MOV	@PTBL.PTR,10(SP)	:	*	TEMP
000050	062766	000002	000010		ADD	#2,10(SP)	:	*	TEMP
000056	032776	000001	000010		BIT	#1,@10(SP)	:	*	TEMP
000064	001402				BEQ	1\$:		
000066	005066	000012			CLR	12(SP)	:		DUMMY
000072	012700	000004		1\$:	MOV	#4,R0	:		
000076	104436				TRAP	36	:		
000100	026727	000000G	177777		CMF	NEX.FLAG,#-1	:		1656
000106	001015				BNE	2\$:		
000110	017716	000000G			MOV	@PTBL.PTR,(SP)	:		1659
000114	012746	000000G			MOV	#FMT6,-(SP)	:		
000120	012746	000002			MOV	#2,-(SP)	:		
000124	010600				MOV	SP,R0	:		SP,*
000126	104414				TRAP	14	:		
000130	152767	000002	000000G		BISB	#2,FLG.WRD	:		1660
000136	104444				TRAP	44	:		
000140	022626				CMF	(SP)+,(SP)+	:		1658
000142	005000			2\$:	CLR	R0	:		1681
000144	104441				TRAP	41	:		
000146	016700	000000G			MOV	VEC.ADDR,R0	:		1682
000152	104436				TRAP	36	:		
000154	012716	000340			MOV	#340,(SP)	:		1683
000160	012746	000000G			MOV	#INT\$I.SERVICE,-(SP)	:		
000164	016746	000000G			MOV	VEC.ADDR,-(SP)	:		
000170	012746	000003			MOV	#3,-(SP)	:		
000174	104437				TRAP	37	:		
000176	012703	177777			MOV	#-1,R3	:	*	RETRIES
000202	004767	000000G		3\$:	JSR	PC,BOOT.RC25	:		1688
000206	010067	000000G			MOV	R0,RET.STATUS	:		1692
000212	005203				INC	R3	:		RETRIES
000214	032700	000001			BIT	#1,R0	:	*	RET.STATUS
000220	001003				BNE	4\$:		
000222	020327	000001			CMF	R3,#1	:		RETRIES,*
000226	001365				BNE	3\$:		
000230	032767	000001	000000G	4\$:	BIT	#1,RET.STATUS	:		1701
000236	001010				BNE	5\$:		
000240	012716	000000G			MOV	#BOOT.FAILURE,(SP)	:		1704
000244	012746	000001			MOV	#1,-(SP)	:		
000250	010600				MOV	SP,R0	:		SP,*

CZRCH4
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
DOWN LINE LOAD FORMATTER AND MONITOR TASK

11-Jul-1983 16:06:09
7-Jul-1983 08:24:38

VAX-11 Bliss-16 V3-555
SPIDERSUSERS:[NEALE.AZTEC]CZRCH4.B16;2 (7)

000252	104414			TRAP	14				
000254	104444			TRAP	44				
000256	005726			TST	(SP)+	:			1703
000260	004767	000000G	5\$:	JSR	PC,INIT.COM.AREA	:			1716
000264	006000			ROR	R0				
000266	103016			BCC	6\$				
000270	012716	000000G		MOV	#PROTO.VIOLATION,(SP)	:			1719
000274	012746	000001		MOV	#1,-(SP)				
000300	010600			MOV	SP,R0	:	SP,*		
000302	104414			TRAP	14				
000304	012716	000000G		MOV	#PORT.INIT.ERR,(SP)	:			1720
000310	012746	000001		MOV	#1,-(SP)				
000314	010600			MOV	SP,R0	:	SP,*		
000316	104414			TRAP	14				
000320	104444			TRAP	44				
000322	022626			CMP	(SP)+,(SP)+	:			1718
000324	016700	000000G	6\$:	MOV	VEC.ADDR,R0	:			1729
000330	104436			TRAP	36				
000332	012716	000340		MOV	#340,(SP)	:			1730
000336	012746	000000G		MOV	#DUP\$I.SERVICE,-(SP)				
000342	016746	000000G		MOV	VEC.ADDR,-(SP)				
000346	012746	000003		MOV	#3,-(SP)				
000352	104437			TRAP	37				
000354	016701	000016G		MOV	ISD.STRUCT+16,R1	:	*,RCSM.REG		1735
000360	052701	000001		BIS	#1,R1	:	*,RCSM.REG		
000364	016700	000000G		MOV	RC25.ADDR,R0				
000370	010160	000002		MOV	R1,2(R0)	:	RCSM.REG,*		
000374	004767	000000G		JSR	PC,SET.CNTRLR.CHAR	:			1758
000400	006000			ROR	R0				
000402	103002			BCC	7\$				
000404	004767	000000G		JSR	PC,DECODE				
000410	016716	000000G	7\$:	MOV	UNIT.NO,(SP)	:			1765
000414	016746	000000G		MOV	LUN,-(SP)				
000420	012746	000000G		MOV	#FMT5,-(SP)				
000424	012746	000003		MOV	#3,-(SP)				
000430	010600			MOV	SP,R0	:	SP,*		
000432	104414			TRAP	14				
000434	004767	000000G		JSR	PC,ON.LINE	:			1768
000440	006000			ROR	R0				
000442	103002			BCC	8\$				
000444	004767	000000G		JSR	PC,DECODE				
000450	004767	000000G	8\$:	JSR	PC,GET.DUST.STATUS	:			1795
000454	006000			ROR	R0				
000456	103002			BCC	9\$				
000460	004767	000000G		JSR	PC,DECODE				
000464	016700	000000G	9\$:	MOV	RET.ENSAD,R0	:			1801
000470	032760	004000 000022		BIT	#4000,22(R0)				
000476	001410			BEQ	10\$				
000500	012716	000000G		MOV	#ACTIVE.DUP.SERVER,(SP)	:			1804
000504	012746	000001		MOV	#1,-(SP)				
000510	010600			MOV	SP,R0	:	SP,*		
000512	104414			TRAP	14				
000514	104444			TRAP	44				
000516	005726			TST	(SP)+	:			1803
000520	004767	000000G	10\$:	JSR	PC,EX.SUP.PROG	:			1814
000524	006000			ROR	R0				
000526	103002			BCC	11\$				

CZRCH4
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
DOWN LINE LOAD FORMATTER AND MONITOR TASK

11-Jul-1983 16:06:09
7-Jul-1983 08:24:38

VAX-11 Bliss-16 V3-555
SPIDER\$USERS:[NEALE.AZTEC]CZRCH4.B16;2 (7)

000530	004767	000000G			JSR	PC,DECODE			
000534	004767	000000G		11\$:	JSR	PC,GET.DUST.STATUS	:		1825
000540	006000				ROR	R0			
000542	103002				BCC	12\$			
000544	004767	000000G			JSR	PC,DECODE			
000550	016701	000000G		12\$:	MOV	RET.ENSAD,R1	:		1831
000554	032761	004000	000022		BIT	#4000,22(R1)			
000562	001011				BNE	13\$			
000564	012716	000000G			JV	#INACTIVE.DUP.SERVER,(SP)	:		1834
000570	012746	000001			MOV	#1,-(SP)			
000574	010600				MOV	SP,R0	:	SP,*	
000576	104414				TRAP	14			
000600	104444				TRAP	44			
000602	005726				TST	(SP)+	:		1833
000604	000406				BR	14\$:		1831
000606	016167	000024	000000G	13\$:	MOV	24(R1),PID.SAVE	:		1839
000614	016167	000026	000002G		MOV	26(R1),PID.SAVE+2	:		1840
000622	005003			14\$:	CLR	R3	:	RETRIES	1850
000624	004767	000000G		15\$:	JSR	PC,REC.DATA	:		1882
000630	006000				ROR	R0			
000632	103002				BCC	16\$			
000634	004767	000000G			JSR	PC,DECODE			
000640	005000			16\$:	CLR	R0	:	I	
000642	005060	000000G		17\$:	CLR	SND.BUF(R0)	:	*(I)	
000646	062700	000002			ADD	#2,R0	:	*,I	
000652	020027	000110			CMP	R0,#110	:	I,*	
000656	101771				BLOS	17\$			
000660	016701	000000G			MOV	REC.BUF,R1	:		1894
000664	006201				ASR	R1			
000666	006201				ASR	R1			
000670	006201				ASR	R1			
000672	006201				ASR	R1			
000674	000301				SWAB	R1			
000676	042701	177760			BIC	#177760,R1			
000702	020127	000001			CMP	R1,#1			
000706	001107				BNE	24\$			
000710	016700	000000G			MOV	REC.BUF,R0	:		1906
000714	042700	170000			BIC	#170000,R0			
000720	001060				BNE	23\$			
000722	032767	000001	000000G		BIT	#1,SW.UNATT	:		1917
000730	001435				BEQ	21\$			
000732	032703	000001			BIT	#1,R3	:	*,RETRIES	1921
000736	001417				BEQ	19\$			
000740	005000				CLR	R0	:	I	1929
000742	105060	000000G		18\$:	CLRB	DATETXT(R0)	:	*(I)	1930
000746	005200				INC	R0	:	I	1929
000750	020027	000013			CMP	R0,#13	:	I,*	
000754	003772				BLE	18\$			
000756	104443				TRAP	43	:		1932
000760	000406				.WORD	406			
000762	000000G				.WORD	DATETXT			
000764	000142				.WORD	142			
000766	000000G				.WORD	DATMSG			
000770	177777				.WORD	-1			
000772	000010				.WORD	10			
000774	000012				.WORD	12			
000776	005000			19\$:	CLR	R0	:	I	1933

CZRCH4
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
DOWN LINE LOAD FORMATTER AND MONITOR TASK

11-Jul-1983 16:06:09
7-Jul-1983 08:24:38

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH4.B16;2 (7)

001000	116060	000000G	000000G	20\$:	MOVB	DATETXT(RO),SND.BUF(RO)	:	*(I),*(I)	
001006	005200				INC	RO	:	I	
001010	020027	000013			CMP	RO,#13	:	I,*	
001014	101771				BLOS	20\$:		
001016	012703	000001			MOV	#1,R3	:	*,RETRIES	1936
001022	000410				BR	22\$:		1917
001024	104443			21\$:	TRAP	43	:		1940
001026	000406				.WORD	406	:		
001030	000000G				.WORD	SND.BUF	:		
001032	000142				.WORD	142	:		
001034	000000G				.WORD	MSGADR	:		
001036	177777				.WORD	-1	:		
001040	000010				.WORD	10	:		
001042	000012				.WORD	12	:		
001044	012716	000000G		22\$:	MOV	#CRLF,(SP)	:		1943
001050	012746	000001			MOV	#1,-(SP)	:		
001054	010600				MOV	SP,RO	:	SP,*	
001056	104414				TRAP	14	:		
001060	000547				BR	31\$:		1945
001062	020027	000007		23\$:	CMP	RO,#7	:		1906
001066	001153				BNE	33\$:		
001070	104443				TRAP	43	:		1951
001072	000406				.WORD	406	:		
001074	000000G				.WORD	SND.BUF	:		
001076	000142				.WORD	142	:		
001100	000000G				.WORD	MSGADR	:		
001102	177777				.WORD	-1	:		
001104	000001				.WORD	1	:		
001106	000100				.WORD	100	:		
001110	012716	000000G			MOV	#CRLF,(SP)	:		1952
001114	012746	000001			MOV	#1,-(SP)	:		
001120	010600				MOV	SP,RO	:	SP,*	
001122	104414				TRAP	14	:		
001124	000525				BR	31\$:		1954
001126	020127	000002		24\$:	CMP	R1,#2	:		1894
001132	001175				BNE	39\$:		
001134	016700	000C00G			MOV	REC.BUF,RO	:		1970
001140	042700	170000			BIC	#170000,RO	:		
001144	020027	000001			CMP	RO,#1	:		
001150	001040				BNE	27\$:		
001152	032767	000001	000000G		BIT	#1,SW.UNATT	:		1981
001160	001415				BEQ	25\$:		
001162	016767	000000G	000000G		MOV	UNIT.NO,SND.BUF	:		1984
001170	016716	000000G			MOV	UNIT.NO,(SP)	:		1985
001174	012746	000000G			MOV	#FMT7,-(SP)	:		
001200	012746	000002			MOV	#2,-(SP)	:		
001204	010600				MOV	SP,RO	:	SP,*	
001206	104414				TRAP	14	:		
001210	022626				CMP	(SP)+,(SP)+	:		1983
001212	000410				BR	26\$:		1981
001214	104443			25\$:	TRAP	43	:		1989
001216	000406				.WORD	406	:		
001220	000000G				.WORD	SND.BUF	:		
001222	000152				.WORD	152	:		
001224	000000G				.WORD	MSGADR	:		
001226	177777				.WORD	-1	:		
001230	000001				.WORD	1	:		

CZRCH4
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
DOWN LINE LOAD FORMATTER AND MONITOR TASK

11-Jul-1983 16:06:09
7-Jul-1983 08:24:38

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH4.B16;2 (7)

001232	000003			.WORD	3			
001234	012716	000000G	26\$:	MOV	#CRLF,(SP)	:		1992
001240	012746	000001		MOV	#1,-(SP)	:		
001244	010600			MOV	SP,R0	:	SP,*	
001246	104414			TRAP	14	:		
001250	000453			BR	31\$:		1994
001252	020027	000004	27\$:	CMP	R0,#4	:		1970
001256	001027			BNE	30\$:		
001260	032767	000001	000000G	BIT	#1,SW.UNATT	:		2006
001266	001404			BEQ	28\$:		
001270	012767	000131	000000G	MOV	#131,SND.BUF	:		2009
001276	000410			BR	29\$:		2006
001300	104443		28\$:	TRAP	43	:		2013
001302	000406			.WORD	406	:		
001304	000000G			.WORD	SND.BUF	:		
001306	000152			.WORD	152	:		
001310	000000G			.WORD	MSGADR	:		
001312	177777			.WORD	-1	:		
001314	000001			.WORD	1	:		
001316	000003			.WORD	3	:		
001320	012716	000000G	29\$:	MOV	#CRLF,(SP)	:		2016
001324	012746	000001		MOV	#1,-(SP)	:		
001330	010600			MOV	SP,RC	:	SP,*	
001332	104414			TRAP	14	:		
001334	000421			BR	31\$:		2018
001336	020027	000005	30\$:	CMP	R0,#5	:		1970
001342	001023			BNE	32\$:		
001344	104443			TRAP	43	:		2024
001346	000406			.WORD	406	:		
001350	000000G			.WORD	SND.BUF	:		
001352	000152			.WORD	152	:		
001354	000000G			.WORD	MSGADR	:		
001356	177777			.WORD	-1	:		
001360	000001			.WORD	1	:		
001362	000003			.WORD	3	:		
001364	012716	000000G		MOV	#CRLF,(SP)	:		2025
001370	012746	000001		MOV	#1,-(SP)	:		
001374	010600			MOV	SP,R0	:	SP,*	
001376	104414			TRAP	14	:		
001400	004767	000000G	31\$:	JSR	PC,SEND.DATA	:		2027
001404	006000			ROR	R0	:		
001406	103436			BLO	36\$:		
001410	000437			BR	37\$:		2023
001412	020027	000006	32\$:	CMP	R0,#6	:		1970
001416	001036		33\$:	BNE	38\$:		
001420	032767	000001	000000G	BIT	#1,SW.UNATT	:		2034
001426	001404			BEQ	34\$:		
001430	012767	000116	000000G	MOV	#116,SND.BUF	:		2037
001436	000410			BR	35\$:		2034
001440	104443		34\$:	TRAP	43	:		2041
001442	000406			.WORD	406	:		
001444	000000G			.WORD	SND.BUF	:		
001446	000152			.WORD	152	:		
001450	000000G			.WORD	MSGADR	:		
001452	177777			.WORD	-1	:		
001454	000001			.WORD	1	:		
001456	000003			.WORD	3	:		

CZRCH4
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
DOWN LINE LOAD FORMATTER AND MONITOR TASK

11-Jul-1983 16:06:09
7-Jul-1983 08:24:38

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH4.B16;2 (7)

001460	012716	000000G	35\$:	MOV	#CRLF,(SP)	:	2044
001464	012746	000001		MOV	#1,-(SP)	:	
001470	010600			MOV	SP,R0	: SP,*	
001472	104414			TRAP	14	:	
001474	004767	000000G		JSR	PC,SEND.DATA	:	2046
001500	006000			ROR	R0	:	
001502	103002			BCC	37\$:	
001504	004767	000000G	36\$:	JSR	PC,DECODE	:	
001510	005726		37\$:	TST	(SP)+	:	2032
001512	000533			BR	44\$:	1970
001514	012767	004001 000000G	38\$:	MOV	#4001,RET.STATUS	:	2052
001522	000167	000532		JMP	57\$:	2053
001526	020127	000003	39\$:	CMP	R1,#3	:	1894
001532	001037			BNE	41\$:	
001534	016700	000000G		MOV	REC.BUF,R0	:	2062
001540	042700	170000		BIC	#170000,R0	:	
001544	020027	000005		CMP	R0,#5	:	
001550	101417			BLOS	40\$:	
001552	020027	000007		CMP	R0,#7	:	
001556	001414			BEQ	40\$:	
001560	020027	000011		CMP	R0,#11	:	
001564	001411			BEQ	40\$:	
001566	020027	000013		CMP	R0,#13	:	
001572	001406			BEQ	40\$:	
001574	020027	000016		CMP	R0,#16	:	
001600	001403			BEQ	40\$:	
001602	020027	000020		CMP	R0,#20	:	
001606	001342			BNE	38\$:	
001610	012716	000000G	40\$:	MOV	#MSGADR,(SP)	:	2067
001614	012746	000000G		MOV	#FMT1,-(SP)	:	
001620	012746	000002		MOV	#2,-(SP)	:	
001624	010600			MOV	SP,R0	: SP,*	
001626	104414			TRAP	14	:	
001630	000463			BR	43\$:	2066
001632	020127	000004	41\$:	CMP	R1,#4	:	1894
001636	001030			BNE	42\$:	
001640	016700	000000G		MOV	REC.BUF,R0	:	2082
001644	042700	170000		BIC	#170000,R0	:	
001650	020027	000014		CMP	R0,#14	:	
001654	103717			BLO	38\$:	
001656	020027	000015		CMP	R0,#15	:	
001662	101314			BHI	38\$:	
001664	012716	000000G		MOV	#MSGADR,(SP)	:	2087
001670	012746	000000G		MOV	#FMT1,-(SP)	:	
001674	012746	000002		MOV	#2,-(SP)	:	
001700	010600			MOV	SP,R0	: SP,*	
001702	104414			TRAP	14	:	
001704	022626			CMP	(SP)+,(SP)+	:	
001706	012700	177777		MOV	#-1,R0	: *,RCSM.REG	2088
001712	010077	000000G		MOV	R0,@RC25.ADDR	: RCSM.REG,*	
001716	000564			BR	59\$:	2086
001720	020127	000005	42\$:	CMP	R1,#5	:	1894
001724	001027			BNE	45\$:	
001726	016700	000000G		MOV	REC.BUF,R0	:	2104
001732	042700	170000		BIC	#170000,R0	:	
001736	001666			BEQ	38\$:	
001740	020027	000030		CMP	R0,#30	:	

CZRCH4
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
DOWN LINE LOAD FORMATTER AND MONITOR TASK

11-Jul-1983 16:06:09
7-Jul-1983 08:24:38

VAX-11 Bliss-16 V3-555
SPIDERSUSERS:[NEALE.AZTEC]CZRCH4.B16;2 (7)

001744	101263		BHI	38\$			
001746	012716	000000G	MOV	#MSGADR,(SP)	:		2109
001752	012746	000000G	MOV	#FMT1,-(SP)			
001756	012746	000002	MOV	#2,-(SP)			
001762	010600		MOV	SP,R0	:	SP,*	
001764	104414		TRAP	14			
001766	012700	177777	MOV	#-1,R0	:	*,RCSM.REG	2110
001772	010077	000000G	MOV	R0,@RC25.ADDR	:	RCSM.REG,*	
001776	104444		TRAP	44			
002000	022626		43\$: CMP	(SP)+,(SP)+	:		2108
002002	000530		44\$: BR	58\$:		2104
002004	020127	000006	45\$: CMP	R1,#6	:		1894
002010	001120		BNE	56\$			
002012	016700	000000G	MOV	REC.BUF,R0	:		2126
002016	042700	170000	BIC	#170000,R0			
002022	020027	000001	CMP	R0,#1			
002026	001232		BNE	38\$			
002030	005767	000002G	TST	REC.BUF+2	:		2138
002034	001013		BNE	46\$			
002036	104443		TRAP	43	:		2141
002040	000406		.WORD	406			
002042	000000G		.WORD	SND.BUF			
002044	000142		.WORD	142			
002046	000000G		.WORD	FCT.REQ.MSG			
002050	177777		.WORD	-1			
002052	000001		.WORD	1			
002054	000012		.WORD	12			
002056	012700	000000G	MOV	#SND.BUF,R0	:		2142
002062	104434		TRAP	34			
002064	012701	000000G	46\$: MOV	#FCT.BUF,R1	:	*,FCT.WRD	2149
002070	000412		BR	50\$			
002072	010100		47\$: MOV	R1,R0	:	FCT.WRD,*	2151
002074	104427		TRAP	27			
002076	103004		BHIS	48\$			
002100	010011		MOV	R0,(R1)	:	R0,FCT.WRD	
002102	012702	000001	MOV	#1,R2	:	*,EOF	
002106	000401		BR	49\$			
002110	005002		48\$: CLR	R2	:	EOF	
002112	062701	000002	49\$: ADD	#2,R1	:	*,FCT.WRD	2149
002116	020127	000776G	50\$: CMP	R1,#FCT.BUF+776	:	FCT.WRD,*	
002122	003763		BLE	47\$			
002124	005702		TST	R2	:	EOF	2161
002126	001004		BNE	51\$			
002130	026727	000002G 000017	CMP	REC.BUF+2,#17			
002136	002404		BLT	52\$			
002140	026727	000002G 000017	51\$: CMP	REC.BUF+2,#17			
002146	003416		BLE	54\$			
002150	012767	177777 000000G	52\$: MOV	#-1,SND.BUF	:		2165
002156	004767	000000G	JSR	PC,SEND.DATA	:		2167
002162	006000		ROR	R0			
002164	103002		BCC	53\$			
002166	004767	000000G	JSR	PC,DECODE			
002172	012767	006001 000000G	53\$: MOV	#6001,RET.STATUS	:		2169
002200	004767	000000G	JSR	PC,DECODE	:		2170
002204	005067	000000G	54\$: CLR	SND.BUF	:		2178
002210	012767	000000G 000002G	MOV	#FCT.BUF,SND.BUF+2	:		2179
002216	005067	000004G	CLR	SND.BUF+4	:		2180

CZRCH4
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
DOWN LINE LOAD FORMATTER AND MONITOR TASK

11-Jul-1983 16:06:09
7-Jul-1983 08:24:38

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH4.B16;2 (7)

```

002222 004767 000000G      JSR    PC,SEND.DATA      ;      2182
002226 006000             ROR    RO                  ;
002230 103002             BCC    55$                 ;
002232 004767 000000G      JSR    PC,DECODE          ;
002236 026727 000002G 000017 55$:  CMP    REC.BUF+2,#17      ;      2188
002244 001007             BNE    58$                 ;
002246 104435             TRAP   35                  ;
002250 000405             BR     58$                 ;
002252 012767 001001 000000G 56$:  MOV    #1001,RET.STATUS  ;
002260 004767 000000G      JSR    PC,DECODE          ;
002264 000167 176334      58$:  JMP    15$                 ;
002270 062706 000036      59$:  ADD    #36,SP            ;
002274 000207             RTS    PC                  ;

```

: Routine Size: 607 words, Routine Base: ABS\$CODE + 0000
: Maximum stack depth per invocation: 23 words

```

000000 004767 175476      T1::  .SBTTL  T1 DOWN LINE LOAD FORMATTER AND MONITOR TASK
000000 1$:  JSR    PC,$T1      ;      2208
000004 104466             TRAP   66
000006 006000             ROR    RO
000010 103773             BLO    1$
000012 000207             RTS    PC

```

: Routine Size: 6 words, Routine Base: ABS\$CODE + 2276
: Maximum stack depth per invocation: 2 words

: 2211 end
: 2212
: 2213 eludom

: OTS external references
: .GLOBL \$SAVE3

: PSECT SUMMARY

Psect Name	Words	Attributes
ABS\$CODE	613	RO, I, LCL, REL, CON

: LIBRARY STATISTICS

File	Symbols		Percent	Blocks Read
	Total	Loaded		
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH0.L16;5	398	168	42	38

CZRCH4 CZRCH RC25 DISK FORMATTER
REV A PATCH 00 DOWN LINE LOAD FORMATTER AND MONITOR TASK

J 13

11-Jul-1983 16:06:09
7-Jul-1983 08:24:38

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH4.B16;2 (7)

SEQ 165
Page 24

COMMAND QUALIFIERS

:
: BLISS /PDP11/LIST CZRCH4.B16
: Size: 613 code + 0 data words
: Run Time: 00:23.9
: Elapsed Time: 00:33.5
: Memory Used: 386 pages
: Compilation Complete

```

0001 MODULE CZRCH5 (%TITLE 'CZRCH RC25 DISK FORMATTER'
0002             IDENT = 'REV A PATCH 00',
0003             ADDRESSING MODE (RELATIVE) ,
0004             ENVIRONMENT (NOEIS)
0005             ) =
0006 BEGIN
0007 %sbttl 'MODULE DECLARATIONS'
0008 |
0009 | Pretty Declarations
0010 |
0011 | <blf/lowercase_key>
0012 |
0013 |
0014 | Library 'CZRCH0';           !Define RC25 Formatter Library
0015 |
0016 | require 'BLSMAC.REQ';      !Define Bliss Macro require file
0017 |
0018 |
0019 |
0020 |
0021 |
0022 |
0023 |
0024 |
0025 |
0026 |
0027 |
0028 |
0029 |
0030 |
0031 |
0032 |
0033 |
0034 |
0035 |
0036 |
0037 |
0038 |
0039 |
0040 |
0041 |
0042 |
0043 |
0044 |
0045 |
0046 |
0047 |
0048 |
0049 |
0050 |
0051 |
0052 |
0053 |
0054 |
0055 |
0056 |
0057 |
0058 |
0059 |
0060 |
0061 |
0062 |
0063 |
0064 |
0065 |
0066 |
0067 |
0068 |
0069 |
0070 |
0071 |
0072 |
0073 |
0074 |
0075 |
0076 |
0077 |
0078 |
0079 |
0080 |
0081 |
0082 |
0083 |
0084 |
0085 |
0086 |
0087 |
0088 |
0089 |
0090 |
0091 |
0092 |
0093 |
0094 |
0095 |
0096 |
0097 |
0098 |
0099 |
0100 |
0101 |
0102 |
0103 |
0104 |
0105 |
0106 |
0107 |
0108 |
0109 |
0110 |
0111 |
0112 |
0113 |
0114 |
0115 |
0116 |
0117 |
0118 |
0119 |
0120 |
0121 |
0122 |
0123 |
0124 |
0125 |
0126 |
0127 |
0128 |
0129 |
0130 |
0131 |
0132 |
0133 |
0134 |
0135 |
0136 |
0137 |
0138 |
0139 |
0140 |
0141 |
0142 |
0143 |
0144 |
0145 |
0146 |
0147 |
0148 |
0149 |
0150 |
0151 |
0152 |
0153 |
0154 |
0155 |
0156 |
0157 |
0158 |
0159 |
0160 |
0161 |
0162 |
0163 |
0164 |
0165 |
0166 |
0167 |
0168 |
0169 |
0170 |
0171 |
0172 |
0173 |
0174 |
0175 |
0176 |
0177 |
0178 |
0179 |
0180 |
0181 |
0182 |
0183 |
0184 |
0185 |
0186 |
0187 |
0188 |
0189 |
0190 |
0191 |
0192 |
0193 |
0194 |
0195 |
0196 |
0197 |
0198 |
0199 |
0200 |
0201 |
0202 |
0203 |
0204 |
0205 |
0206 |
0207 |
0208 |
0209 |
0210 |
0211 |
0212 |
0213 |
0214 |
0215 |
0216 |
0217 |
0218 |
0219 |
0220 |
0221 |
0222 |
0223 |
0224 |
0225 |
0226 |
0227 |
0228 |
0229 |
0230 |
0231 |
0232 |
0233 |
0234 |
0235 |
0236 |
0237 |
0238 |
0239 |
0240 |
0241 |
0242 |
0243 |
0244 |
0245 |
0246 |
0247 |
0248 |
0249 |
0250 |
0251 |
0252 |
0253 |
0254 |
0255 |
0256 |
0257 |
0258 |
0259 |
0260 |
0261 |
0262 |
0263 |
0264 |
0265 |
0266 |
0267 |
0268 |
0269 |
0270 |
0271 |
0272 |
0273 |
0274 |
0275 |
0276 |
0277 |
0278 |
0279 |
0280 |
0281 |
0282 |
0283 |
0284 |
0285 |
0286 |
0287 |
0288 |
0289 |
0290 |
0291 |
0292 |
0293 |
0294 |
0295 |
0296 |
0297 |
0298 |
0299 |
0300 |
0301 |
0302 |
0303 |
0304 |
0305 |
0306 |
0307 |
0308 |
0309 |
0310 |
0311 |
0312 |
0313 |
0314 |
0315 |
0316 |
0317 |
0318 |
0319 |
0320 |
0321 |
0322 |
0323 |
0324 |
0325 |
0326 |
0327 |
0328 |
0329 |
0330 |
0331 |
0332 |
0333 |
0334 |
0335 |
0336 |
0337 |
0338 |
0339 |
0340 |
0341 |
0342 |
0343 |
0344 |
0345 |
0346 |
0347 |
0348 |
0349 |
0350 |
0351 |
0352 |
0353 |
0354 |
0355 |
0356 |
0357 |
0358 |
0359 |
0360 |
0361 |
0362 |
0363 |
0364 |
0365 |
0366 |
0367 |
0368 |
0369 |
0370 |
0371 |
0372 |
0373 |
0374 |
0375 |
0376 |
0377 |
0378 |
0379 |
0380 |
0381 |
0382 |
0383 |
0384 |
0385 |
0386 |
0387 |
0388 |
0389 |
0390 |
0391 |
0392 |
0393 |
0394 |
0395 |
0396 |
0397 |
0398 |
0399 |
0400 |
0401 |
0402 |
0403 |
0404 |
0405 |
0406 |
0407 |
0408 |
0409 |
0410 |
0411 |
0412 |
0413 |
0414 |
0415 |
0416 |
0417 |
0418 |
0419 |
0420 |
0421 |
0422 |
0423 |
0424 |
0425 |
0426 |
0427 |
0428 |
0429 |
0430 |
0431 |
0432 |
0433 |
0434 |
0435 |
0436 |
0437 |
0438 |
0439 |
0440 |
0441 |
0442 |
0443 |
0444 |
0445 |
0446 |
0447 |
0448 |
0449 |
0450 |
0451 |
0452 |
0453 |
0454 |
0455 |
0456 |
0457 |
0458 |
0459 |
0460 |
0461 |
0462 |
0463 |
0464 |
0465 |
0466 |
0467 |
0468 |
0469 |
0470 |
0471 |
0472 |
0473 |
0474 |
0475 |
0476 |
0477 |
0478 |
0479 |
0480 |
0481 |
0482 |
0483 |
0484 |
0485 |
0486 |
0487 |
0488 |
0489 |
0490 |
0491 |
0492 |
0493 |
0494 |
0495 |
0496 |
0497 |
0498 |
0499 |
0500 |
0501 |
0502 |
0503 |
0504 |
0505 |
0506 |
0507 |
0508 |
0509 |
0510 |
0511 |
0512 |
0513 |
0514 |
0515 forward routine
0516     LOAD_FILE,           !Load file from local load media
0517     GET_NSD,             !Get next send descriptor slot index
0518     GET_NRD,             !Get next receive descriptor slot index
0519     LOAD_OUT$STD_BUF,    !Load out standing command buffer
0520     GET_CMS$REF,         !Get unique command reference number
0521     DECODE : novalue,    !Decode return status error code
0522     DUP$I SERVICE : INT_LNK$TYP novalue, !Dup/UQ port interrupt service routine
0523     CTO WAIT,           !Command time out wait
0524     ABORT,              !Abort Dup command
0525     GET_DUST STATUS,    !Get Dust Status command
0526     EX_SUP_PROG,        !Execute Supplied Program command
0527     EX_LOC_PROG,        !Execute Local Program command
0528     SEND_DATA,          !Send Data command
0529     REC_DATA,           !Receive Data command
0530     SET_CNTRLR_CHAR,    !Set Controller Characteristics command
0531     ON_LINE,            !On Line command
0532     INT$I SERVICE : INT_LNK$TYP novalue, !Initialization sequence interrupt service
0533     IS_TIMER,           !Initialization sequence time-out wait
0534     BOOT_RC25,          !Initialize sequence for RC25 controller
0535     INIT_COM_AREA;      !Initialize UQ Port communication area
0536 |
0537 | <BLF/PAGE>

```

CZRCH5
REV A PATCH 00CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS11-Jul-1983 16:06:44
7-Jul-1983 08:24:58VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (2)SEQ 167
Page 2

```

:      1538  !+
:      1539  ! The psect named 'code or $code$' is redefined here to be called 'ac$code'.
:      1540  ! This is done to organize formatter routine code into a seperate psect.
:      1541  !-
:      1542  psect
:      1543      code = ac$code;
:      1544
:      1545  !+
:      1546  ! Structure declarations used within this module.
:      1547  !-
:      1548
:      1549  structure
:      1550
:      1551      !+
:      1552      ! RC25 register accessing structure. This structure allows RC25 register
:      1553      ! accessing to be transportable between the PDP-11 and VAX Diagnostic Supervisors.
:      1554      !-
:      1555      ! This also defines an access algorithm for VAX to allow field reference
:      1556      ! to MBA address space without generating machine checks.
:      1557      !-
:      1558
:      1559      RC25 [O, P, S, E] =
:      1560          begin
:      1561              local
:      1562                  RC$$_REG;
:      1563
:      1564                  RC$$_REG = .(RC25 + %upval*0)<0, %bpval, 0>;
:      1565                  RC$$_REG
:      1566                  end
:      1567                  <P, S, E>;
:      1568
:      1569
:      1570  !<blf/page>

```



```

:      1571  !+
:      1572  ! External Declaration of datums declared outside of this module.
:      1573  !-
:      1574
:      1575  external
:      1576  !
:      1577  ! Communications area declarations
:      1578  !
:      1579  COM AREA : blockvector [REC_ALLOCATE + SND_ALLOCATE + HDR_SIZ, 2, word],
:      1580  HEAD AREA : ref block [4, word] field (HDR_FIELD),
:      1581  RECEIVE_RING : ref blockvector [REC_ALLOCATE, 2, word] field (DSC_FIELD),
:      1582  SEND_RING : ref blockvector [SND_ALLOCATE, 2, word] field (DSC_FIELD),
:      1583  REC_ENVELOPE : blockvector [REC_ALLOCATE, RB_SIZE + 2, word] field (ENV_FIELD),
:      1584  SND_ENVELOPE : blockvector [SND_ALLOCATE, SB_SIZE + 2, word] field (ENV_FIELD),
:      1585  RET_EN$AD : ref block [RB_SIZE + 2, word] field (ENV_FIELD),
:      1586  REC_BUF : block [RECB_SIZE, word] field (RECB_FIELD),
:      1587  SND_BUF : vector [SNDB_SIZE, word],
:      1588  OUT$STD_BUF : blockvector [REC_ALLOCATE, 2, word] field (OUT$FIELD),
:      1589  !
:      1590  ! Miscellaneous external data declarations
:      1591  !
:      1592  NEX_FLAG : word,           !Non-existent RC25 register flag
:      1593  NRD_SLOT : word,       !Next receive descriptor slot
:      1594  NSD_SLOT : word,       !Next send descriptor slot
:      1595  RC25_ADDR : ref RC25 field (ISD_FIELD), !Controller reg access struct
:      1596  RET_STATUS : word,      !Global return status location
:      1597  PID_SAVE : vector [2, word], !Saves program indicator field
:      1598  VEC_ADDR : word,       !RC25 interrupt vector address
:      1599  RSVD_STRUCT : vector [4, word], !Stores init seq reserved fields
:      1600  ISD_STRUCT : blockvector [4, 2, word] field (ISD_FIELD), !Init seq data
:      1601  UNIT_NO : word,        !P table unit number to format
:      1602  NXT_CRN : byte,       !Stores next cmd ref number
:      1603  !
:      1604  ! Error Messages Structures
:      1605  !
:      1606  PFE_STRUCT : vector [23], !Port fatal error msg struct
:      1607  EMSG_STRUCT : vector [23], !Error message structure
:      1608  RC_STRUCTURE : vector [39], !RC25 SA register fatal error
:      1609  SDUP_STRUCT : vector [7], !DUP return status code messages
:      1610  SMSCP_STRUCT : vector [13], !MSCP return status code messages
:      1611  !<blf/page>

```

CZRCH5
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555
SPIDERS\$UJERS:[NEALE.AZTEC]CZRCH5.B16;3 (4)

```

:      1612      :
:      1613      : Formatted ascii strings
:      1614      :
:      1615      : PID_FMT,           !Progress indicator printing string
:      1616      : FMTZ,             !Micro code version printing format
:      1617      :
:      1618      : DM code parameter declarations
:      1619      :
:      1620      : The following declarations are declared in module azkel6 'DM code
:      1621      : module' and point to the starting address of specific DM code buffer areas.
:      1622      :
:      1623      : (OVSA is declared in azkel2 and is a memory location however)
:      1624      :
:      1625      : It is a hack but the azfmtr storage allocation must be edited in manually
:      1626      : any time the DM code buffer changes. Get this value from the output of
:      1627      : DMCONV program. (Bliss wants this allocation value to be a compile time constant).
:      1628      :
:      1629      : AZFMTR : vector [8892, word],      !RC25 formatter DM code buffer
:      1630      : HDSA,          !Host DM code header address
:      1631      : DMSA,          !Host DM code initial load address
:      1632      : OVSA;          !Host First overlay address
:      1633      :

```

CZRCH5
REV A PATCH 00CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS11-Jul-1983 16:06:44
7-Jul-1983 08:24:58VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (5)SEQ 170
Page 5

```

1634 global routine LOAD_FILE (BUF_LOC, FN_EXT, BUF_SIZ) = !Load file from local load media
1635
1636 !++
1637 Functional Description :
1638 This multi-purpose routine is used to read in files from the boot
1639 device into host memory space at the specified buffer address.
1640
1641 Formal Parameters :
1642 BUF_LOC This contains the host memory address where
1643 words read in from the boot device are to
1644 be loaded.
1645
1646 FN_EXT This contains a pointer to an asciz string
1647 of the file name to be read in.
1648
1649 BUF_SIZ This contains the number of words allocated
1650 for the buffer where the file is to be read
1651 in. This is used for protect against buffer
1652 over runs.
1653 Implicit Inputs :
1654 none
1655
1656 Implicit Outputs :
1657 none
1658
1659 Completion Codes :
1660 FRE_CODE Is returned if the end-of-file indicator is
1661 not returned before the buffer is full.
1662
1663 PAS_CODE Is returned if end-of-file indicator is returned
1664 before the buffer is full.
1665
1666 Side Effects :
1667 Data previously loaded in the specified load buffer is
1668 destroyed by the incoming file from disk.
1669 --
1670
1671 begin
1672 local
1673 NEOF, !Return status from getword macro
1674 EOB; !End of buffer address storage
1675
1676 OPEN (.FN_EXT); !Open the file for input
1677 EOB = (.BUF_SIZ*2) + .BUF_LOC; !Calculate the last buffer address
1678
1679
1680 !+
1681 Read the first block from the input file and through
1682 it away since it only contains RT-11 information.
1683 !-
1684
1685 incru i from 0 to 255 do
1686 begin
1687 GETWORD (.BUF_LOC);
1688 end;
1689
1690 !+

```

CZRCH5
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (5)

```

: 1691      ! Loop on getting a word from the load file. After
: 1692      ! increment to the next word by adding %'2' to the
: 1693      ! buffer load addrss. Continue getting words until
: 1694      ! the not complete indicator is returned (0) or the
: 1695      ! buffer is over run. Return an error code if the
: 1696      ! buffer is over run.
: 1697      !-
: 1698
: 1699      do
: 1700          begin
: 1701              NEOF = GETWORD (.BUF_LOC);           !Read next word into .mem_loc
: 1702              BUF_LOC = .BUF_LOC + %'2';         !Increment to next word
: 1703          end
: 1704      until ( not .NEOF) or (.BUF_LOC eqlu .EOB); !Repeat until eof or buf overrun
: 1705
: 1706      !
: 1707      ! Return an error code if not eof (neof = 1) else close the file and return a pass code.
: 1708      !
: 1709
: 1710      if .NEOF                                     !Was the eof indicator returned
: 1711      then
: 1712          return RET_STATUS = FRE_CODE             !Return error code to caller
: 1713      else
: 1714          begin
: 1715              CLOSE;                               !Eof was returned so close file
: 1716          end;
: 1717
: 1718      return RET_STATUS = PAS_CODE;               !Return a pass code to the caller
: 1719      end;

```

```

.TITLE CZRCH5 CZRCH RC25 DISK FORMATTER
.IDENT /REV A /

.GLOBL COM.AREA, HEAD.AREA, RECEIVE.RING
.GLOBL SEND.RING, REC.ENVELOPE, SND.ENVELOPE
.GLOBL RET.ENSAD, REC.BUF, SND.BUF, OUT$STD.BUF
.GLOBL NEX.FLAG, NRD.SLOT, NSD.SLOT, RC25.ADDR
.GLOBL RET.STATUS, PID.SAVE, VEC.ADDR
.GLOBL RSVD.STRUCT, ISD.STRUCT, UNIT.NO
.GLOBL NXT.CRN, PFE.STRUCT, EMSG.STRUCT
.GLOBL RC.STRUCTURE, SDUP.STRUCT, SMSCP.STRUCT
.GLOBL PID.FMT, FMT4, AZFMTR, HDSA, DMSA
.GLOBL OVSA

```

```

000000      .SBTTL LOAD.FILE MODULE DECLARATIONS
000000      .PSECT AC$CODE, RO

000000 004167 000000G      LOAD.FILE::
000004 016600 000014      JSR      R1,$SAVE3
000010 104434 000014      MOV      14(SP),R0
000012 016600 000012      TRAP    34
000016 006300 000012      MOV      12(SP),R0
000020 016602 000016      ASL     R0
000024 060200 000016      MOV      16(SP),R2
                                ADD     R2,R0

```

1634
1677
1678

CZRCH5
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (5)

000026	010003			MOV	R0,R3	:	* ,EOB	
000030	005001			CLR	R1	:	I	1685
000032	010200		1\$:	MOV	R2,R0	:		1687
000034	104427			TRAP	27	:		
000036	103001			BHIS	2\$			
000040	010012			MOV	R0,(R2)	:	R0,*	
000042	005201		2\$:	INC	R1	:	I	1685
000044	020127	000377		CMP	R1,#377	:	I,*	
000050	101770			BLOS	1\$			
000052	016600	000016	3\$:	MOV	16(SP),R0	:	BUF.LOC,*	1701
000056	104427			TRAP	27			
000060	103005			BHIS	4\$			
000062	010076	000016		MOV	R0,@16(SP)	:	R0,BUF.LOC	
000066	012702	000001		MOV	#1,R2	:	* ,NEOF	
000072	000401			BR	5\$			
000074	005002		4\$:	CLR	R2	:	NEOF	
000076	062766	000002 000016	5\$:	ADD	#2,16(SP)	:	* ,BUF.LOC	1702
000104	032702	000001		BIT	#1,R2	:	* ,NEOF	1704
000110	001403			BEQ	6\$			
000112	026603	000016		CMP	16(SP),R3	:	BUF.LOC,EOB	
000116	001355			BNE	3\$			
000120	006002		6\$:	ROR	R2	:	NEOF	1710
000122	103005			BCC	7\$			
000124	012700	005001		MOV	#5001,R0	:		1712
000130	010067	000000G		MOV	R0,RET.STATUS			
000134	000207			RTS	PC			
000136	104435		7\$:	TRAP	35	:		1714
000140	005067	000000G		CLR	RET.STATUS	:		1718
000144	005000			CLR	R0	:		1671
000146	000207			RTS	PC	:		1634

: Routine Size: 52 words, Routine Base: AC\$CODE + 0000
: Maximum stack depth per invocation: 6 words

: 1720

CZRCH5
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (6)

```

:      1721 global routine GET_NSD =                !Chooses the next send slot
:      1722
:      1723 |++
:      1724 | Functional Description :
:      1725 |   This routine will determine which send ring descriptor the port/controller
:      1726 |   is polling and returns that dsc slot number to the calling routine.
:      1727 |   This host program will call this routine each time it wishes to deposit
:      1728 |   another command into the command (send) ring.
:      1729 | Formal Parameters :
:      1730 |   none
:      1731 | Implicit Inputs :
:      1732 |   NSD_SLOT :      Global storage for the next send descriptor slot.
:      1733 |                   Stores where the host should place this command for
:      1734 |                   processing by the port/controller.
:      1735 | Implicit Outputs :
:      1736 |   The global storage 'Nsd_slot' is updated to the
:      1737 |   present send slot where the port/controller is polling.
:      1738 | Completion Codes :
:      1739 |   Returns the contents of 'Nsd_slot' to the calling routine.
:      1740 | Side Effects :
:      1741 |   none
:      1742 | --
:      1743
:      1744 | begin
:      1745 |
:      1746 |   Increment the next send descriptor_slot by one
:      1747 |
:      1748 |   NSD_SLOT = .NSD_SLOT + 1;
:      1749 |
:      1750 |   Set the slot pointer back to zero if it wraps around to the top of the ring.
:      1751 |
:      1752 |
:      1753 |   if .NSD_SLOT gtru SND_ALLOCATE - 1 then NSD_SLOT = ZERO;
:      1754 |
:      1755 |
:      1756 |   Return the next send descriptor_slot to the caller
:      1757 |
:      1758 |   return .NSD_SLOT;
:      1759 | end;

```

000000	005267	000000G	GET.NSD::	.SBTTL	GET.NSD MODULE DECLARATIONS		
			INC	NSD.SLOT		:	1748
000004	026727	000000G 000003	CMP	NSD.SLOT,#3		:	1753
000012	101402		BLOS	1\$			
000014	005067	000000G	CLR	NSD.SLOT			
000020	016700	000000G	1\$: MOV	NSD.SLOT,RO		:	1744
000024	000207		RTS	PC		:	1721

```

: Routine Size: 11 words,      Routine Base: AC$CODE + 0150
: Maximum stack depth per invocation: 0 words

```

: 1760

CZRCH5
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (7)

```

:      1761 global routine GET_NRD =                !Chooses the next receive slot
:      1762
:      1763 !++
:      1764 ! Functional Description :
:      1765 ! This routine will determine which receive ring descriptor the port/controller
:      1766 ! is polling and returns that dsc slot number to the calling routine. This
:      1767 ! host program will call this routine each time it wishes to process another
:      1768 ! receive ring descriptor.
:      1769 ! Formal Parameters :
:      1770 ! none
:      1771 ! Implicit Inputs :
:      1772 !   NRD_SLOT :      Global storage for the next receive descriptor slot.
:      1773 !                   Stores where the port should return this commands
:      1774 !                   response indicator.
:      1775 ! Implicit Outputs :
:      1776 ! The global storage 'Nrd_slot' is updated to the present receive slot
:      1777 ! where the port/controller is polling.
:      1778 ! Completion Codes :
:      1779 ! Returns the contents of 'Nrd_slot' to the calling routine.
:      1780 ! Side Effects :
:      1781 ! none
:      1782 !--
:      1783
:      1784     begin
:      1785     !
:      1786     ! Increment the next receive descriptor_slot by one
:      1787     !
:      1788     NRD_SLOT = .NRD_SLOT + 1;
:      1789     !
:      1790     ! Set the slot pointer back to zero if it wraps around to the top of the ring.
:      1791     !
:      1792     !
:      1793     if .NRD_SLOT gtru REC_ALLOCATE - 1 then NRD_SLOT = ZERO;
:      1794     !
:      1795     !
:      1796     ! Return the next receive descriptor_slot to the caller
:      1797     !
:      1798     return .NRD_SLOT;
:      1799     end;

```

000000	005267	000000G	GET.NRD: :	.SBTTL GET.NRD MODULE DECLARATIONS		
				INC NRD_SLOT	:	1788
000004	026727	000000G 000003		CMP NRD_SLOT,#3	:	1793
000012	101402			BLOS 1\$		
000014	005067	000000G		CLR NRD_SLOT		
000020	016700	000000G	1\$:	MOV NRD_SLOT,R0	:	1784
000024	000207			RTS PC	:	1761

```

: Routine Size: 11 words,      Routine Base: AC$CODE + 0176
: Maximum stack depth per invocation: 0 words

```

: 1800

CZRCH5
REV A PATCH 00CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS11-Jul-1983 16:06:44
7-Jul-1983 08:24:58VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (8)SEQ 175
Page 10

```

1801 global routine LOAD_OUT$STD_BUF (REF_NUM) = !Load out$std_buffer with this command
1802
1803 !++
1804 Functional Description :
1805 The outstanding command buffer 'out$std_buf' is used by
1806 this host program to determine if an outstanding command
1807 issued to the port has been processed yet. This is done
1808 by examining the receive flag 'Rec_flg' in a buffer slot
1809 for a '1' which is set by the interrupt service routine
1810 during response ring interrupts.
1811
1812 This buffer can be looked at as a window between the port
1813 driver receiving & processing the response envelopes and the
1814 host class driver issuing commands to the port.
1815
1816 This routine loads into an empty out$std_buf slot the
1817 following values:
1818
1819 1. This commands reference number.
1820 2. Clears 'rec_flg' indicating this command is outstanding.
1821 3. Clears out the second word in slot where the returned
1822 envelope address will go.
1823
1824 IMPORTANT NOTE:
1825 -----
1826 To quarentee a command loaded into the out$std_buffer will
1827 never be lost (i.e. having this routine return a buffer
1828 slot not yet received by the interrupt service routine),
1829 only the cto_wait (controller time out wait) routine is
1830 permitted to return a out$std_buf slot to the unused pool
1831 (i.e. by loading a slots first word with %o'100000').
1832 This routine is therefore quarenteed to return an unused
1833 out$std_buffer slot when this unique value of %o'100000'
1834 is found. To further quarentee this, unique command ref
1835 numbers will never use zero as a reference number.
1836
1837 Formal Parameters :
1838 REF_NUM This is the unique reference number of this command set to the port.
1839
1840 Implicit Inputs :
1841 none
1842
1843 Implicit Outputs :
1844 none
1845
1846 Completion Codes :
1847 The out$standing buffer slot index where this command was put
1848 is routines value and is returned to the caller.
1849
1850 Side Effects :
1851 none
1852
1853 --
1854 begin
1855 !+
1856
1857

```


CZRCH5
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-i1 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (8)

```

:      1858      ! Search through the out standing command buffer and look for the first
:      1859      ! open slot. Return this first slot index to the caller if one is found.
:      1860      ! If no open slots are found then return an error code.
:      1861      !-
:      1862
:      1863      incru i from 0 to REC_ALLOCATE - 1 do          !Find the first open slot
:      1864      !
:      1865      ! An open slot is by definition the value %o'100000' in the slots first word.
:      1866      !
:      1867      !
:      1868      if .OUT$STD_BUF [.i, CMD_WRD] eqlu %o'100000'  !Is this slot open
:      1869      then
:      1870      begin
:      1871      OUT$STD_BUF [.i, REC_FLG] = FALSE;  !Clear the received flag
:      1872      OUT$STD_BUF [.i, CMD_REF] = .REF_NUM;  !Load this cmd's ref num
:      1873      OUT$STD_BUF [.i, ENV_ADR] = ZERO;  !Clear the previous env adr
:      1874      return .i;  !Return buffer index to caller
:      1875      end;
:      1876
:      1877      !
:      1878      ! The buffer is full if the code reaches here. This should never happen so
:      1879      ! report an error for debug purposes.
:      1880
:      1881      return RET_STATUS = OBF_CODE;  !Report an 'out$std buffer full' error
:      1882      end;

```

		.SBTTL	LOAD.OUT\$STD.BUF	MODULE DECLARATIONS	
000000	004167	000000G	LOAD.OUT\$STD.BUF::		
			JSR	R1,\$SAVE2	: 1801
000004	005001		CLR	R1	: 1863
000006	010100		1\$: MOV	R1,R0	: 1,* 1868
000010	006300		ASL	R0	
000012	006300		ASL	R0	
000014	012702	000000G	MOV	#OUT\$STD.BUF,R2	
000020	060002		ADD	R0,R2	
000022	021227	100000	CMP	(R2),#-100000	
000026	001010		BNE	2\$	
000030	042712	100000	BIC	#100000,(R2)	: 1871
000034	116612	000010	MOVB	10(SP),(R2)	: REF.NUM,* 1872
000040	005060	000002G	CLR	OUT\$STD.BUF+2(R0)	: 1873
000044	010100		MOV	R1,R0	: 1,* 1870
000046	000207		RTS	PC	
000050	005201		2\$: INC	R1	: 1 1863
000052	020127	000003	CMP	R1,#3	: 1,*
000056	101753		BLOS	1\$	
000060	012700	002001	MOV	#2001,R0	: 1881
000064	010067	000000G	MOV	R0,RET.STATUS	
000070	000207		RTS	PC	: 1801

```

: Routine Size: 29 words,      Routine Base: AC$CODE + 0224
: Maximum stack depth per invocation: 4 words

```

: 1883

CZRCH5
REV A PATCH 00CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS11-Jul-1983 16:06:44
7-Jul-1983 08:24:58VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (9)SEQ 177
Page 12

```

1884 global routine GET_CMD$REF =                                !Gets next unique cmd ref number
1885
1886 !++
1887 ! Functional Description :
1888 !   A 32 bit unique non-zero number used to identify host commands.
1889 !   Class drivers should supply a unique reference number in each
1890 !   command that they send to a DUP server. A class driver may supply
1891 !   a zero reference number if it does not need to associate a command
1892 !   with its end message.
1893
1894 !   Command reference numbers must be unique across all commands that
1895 !   are outstanding on the same connection i.e., they must be unique
1896 !   across all outstanding commands issued by a single class driver
1897 !   (Host) to a single DUP server. The class driver may re-use a
1898 !   command reference number when the command is no longer
1899 !   outstanding -- i.e., after receiving the command's end message or
1900 !   after re-synchronizing with the DUP server. Command reference
1901 !   numbers need not be unique for commands issued by different class
1902 !   drivers --- i.e. commands issued by different hosts or commands for
1903 !   different DUP servers from the same host. Therefore controllers
1904 !   must internally use the combination of a command reference number
1905 !   and the connection on which the command was received as the unique
1906 !   identifier of an outstanding command.
1907
1908 !   This routine will generate a unique command reference number and
1909 !   will search the outstanding command buffer to see if already used.
1910 !   The first unused unique command reference found will be returned
1911 !   to the calling routine.
1912
1913 ! Formal Parameters :
1914 !   none
1915
1916 ! Implicit Inputs :
1917 !   NXT_CRN           This global location stores the next unique cmd
1918 !                     reference number to be used.
1919
1920 ! Implicit Outputs :
1921 !   NXT_CRN           This global location is loaded with the next
1922 !                     unique command reference number.
1923
1924 ! Completion Codes :
1925 !   The contents of .NXT_CRN is returned to the calling routine.
1926
1927 ! Side Effects :
1928 !   none
1929 !--
1930
1931 begin
1932 local
1933     DONE;
1934
1935     !
1936     ! Increment the global unique command reference number before anything is done.
1937     !
1938     !
1939     NXT_CRN = .NXT_CRN + 1;
1940

```

```

:      1941      |  |
:      1942      |  |
:      1943      |  | Repeat generating and searching the out$standing command buffer until a
:      1944      |  | unique command reference number is found.
:      1945      |  |
:      1946      |  |
:      1947      |  | do
:      1948      |  |   begin
:      1949      |  |   BREAK;                                !Flag any operator control C's
:      1950      |  |   |
:      1951      |  |   | Wrap this next command reference number around
:      1952      |  |   | back to one if it is greater then 255 (decimal).
:      1953      |  |   |
:      1954      |  |   | if .NXT_CRN gtr 255 then NXT_CRN = 1;
:      1955      |  |   |
:      1956      |  |   | DONE = TRUE;                            !Clear the all done indicator flag
:      1957      |  |   |
:      1958      |  |   | |
:      1959      |  |   | | Now search the out$std_buffer for this command
:      1960      |  |   | | reference number. If not there then done stays
:      1961      |  |   | | true and the loop will end else increment to the
:      1962      |  |   | | next unique command ref number and make done false
:      1963      |  |   | | to continue the loop.
:      1964      |  |   | |
:      1965      |  |   | |
:      1966      |  |   | | incru i from 0 to REC_ALLOCATE - 1 do !Search buffer for this cmd ref num
:      1967      |  |   | |
:      1968      |  |   | |   if .OUT$STD_BUF [.i, CMD_REF] eqlu .NXT_CRN !Does it already exist
:      1969      |  |   | |   then
:      1970      |  |   | |   begin                                !It already exists
:      1971      |  |   | |   NXT_CRN = .NXT_CRN + 1;          !Try the next sequential cmd ref num
:      1972      |  |   | |   DONE = FALSE;                    !Make code loop again
:      1973      |  |   | |   exitloop;                          !Exit this incr loop
:      1974      |  |   | |   end;
:      1975      |  |   | |
:      1976      |  |   | | end
:      1977      |  |   | | until .DONE;                                !Repeat loop until done
:      1978      |  |   | |
:      1979      |  |   | |
:      1980      |  |   | | ! Return the unique command reference number to the caller.
:      1981      |  |   | |
:      1982      |  |   | | return .NXT_CRN;
:      1983      |  |   | | end;

```

Address	Offset	Label	SBTTL	GET.CMD\$REF	MODULE DECLARATIONS	Address
000000	010146		GET.CMD\$REF::			1884
000002	105267	000000G	MOV	R1, -(SP)		1939
000006	104422		INCB	NXT.CR		1947
000010	012701	000001	1\$: TRAP	22		1956
000014	005000		MOV	#1, R1	* , DONE	1966
000016	126067	000000G 000000G	2\$: CLR	R0	I	1968
000024	001004		CMPB	OUT\$STD.BUF (R0), NXT.CR	*(I), *	
000026	105267	000000G	BNE	3\$		1971
000032	005001		INCB	NXT.CR		1972
000034	000405		CLR	R1	DONE	1970
			BR	4\$		

CZRCH5 CZRCH RC25 DISK FORMATTER
REV A PATCH 00 MODULE DECLARATIONS

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555
SPIDERSUSERS:[NEALE.AZTEC]CZRCH5.B16;3 (9) SEQ 179
Page 14

000036	062700	000004	3\$:	ADD	#4,R0	:	*.I	1966
000042	020027	000014		CMP	R0,#14	:	I.*	
000046	101763			BLOS	2\$			
000050	006001		4\$:	ROR	R1	:	DONE	1977
000052	103355			BCC	1\$			
000054	005000			CLR	R0	:		1931
000056	156700	000000G		BISB	NXT.CRN,R0	:		
000062	012601			MOV	(SP)+,R1	:		1884
000064	000207			RTS	PC			

: Routine Size: 27 words, Routine Base: AC\$CODE + 0316
: Maximum stack depth per invocation: 3 words

: 1984

7
2

CZRCH5
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (10)

```

:      1985 global routine DECODE : novalue =           !Decodes failing SA reg data
:      1986
:      1987 !++
:      1988 ! Functional Description :
:      1989 ! Due to the implimentation of the DUP and UQ Port protocol there
:      1990 ! are two levels at which an issued command to a port/controller
:      1991 ! can fail and they are:
:      1992
:      1993 !     1. The issued command can time out.
:      1994
:      1995 !     2. An error can be posted in SA register bit 15 by the port to
:      1996 ! report an error.
:      1997
:      1998 !     3. The issued command to the port/controller can be executed
:      1999 ! correctly without any errors but the response packet status
:      2000 ! field could have an error or status other than success posted.
:      2001
:      2002 ! These errors or status's returned are all returned to the host
:      2003 ! routine which queued the DUP command via the global storage
:      2004 ! 'RET_STATUS'. The host to port/controller communications
:      2005 ! connection having the highest priority (ie. if the SA Reg error
:      2006 ! bit is set the DUP interrupt routine returns a PFE_CODE and this
:      2007 ! code is passed up to the calling host routine with out being
:      2008 ! intercepted by any routine on the way up).
:      2009
:      2010 ! This routine will then be called when the return from a queued
:      2011 ! command comes back with an error code or non successfull status
:      2012 ! code. This is by definition when bit 0 in the returned status
:      2013 ! is equal to 1.
:      2014
:      2015 ! An appropriate recovery action will be done for each individual
:      2016 ! error.
:      2017
:      2018 ! Formal Parameters :
:      2019 ! none
:      2020
:      2021 ! Implicit Inputs :
:      2022 ! RET_STATUS:      Stored in this global storage is the returned error
:      2023 !                   code or non-successful status code from a queued
:      2024 !                   command.
:      2025
:      2026 ! Implicit Outputs :
:      2027 ! none
:      2028
:      2029 ! Completion Codes :
:      2030 ! none
:      2031
:      2032 ! Side Effects :
:      2033 ! All formatter errors are fatal, therefor after execution
:      2034 ! of this routine the RC25 controller is initialized
:      2035 ! aborting any DM code running in the controller.
:      2036 ! --
:      2037
:      2038 begin
:      2039
:      2040 local
:      2041     SA_VALUE;           !Save SA register fatal error code

```

```

2042
2043
2044      |* Use the contents of 'RET_STATUS' to select what type error or non-successfu!
2045      | status code is to be processed.
2046      |
2047
2048      selectoneu .RET_STATUS of
2049      set
2050      |
2051      | 'Communication area initialize' error code
2052      |
2053      | This error code indicates that the port did
2054      | not init the com area in the host memory after
2055      | step 2 of the initialization sequence.
2056      |
2057
2058      [CIE_CODE] :                               !Code equals %o'01'
2059      begin
2060      PRINTB (.EMSG_STRUCT [MSG4]);
2061      end;
2062      |
2063      | 'Port/Controller time out' error code
2064      |
2065      | Port/Controller timed out after the specified time out interval.
2066      |
2067
2068      [CTO_CODE] :                               !Code equals %o'11'
2069      begin
2070      PRINTB (.EMSG_STRUCT [MSG21]);
2071      end;
2072      |
2073      | 'Port fatal error' code
2074      |
2075      | The error bit in the SA Register was set when examined in the DUP$I_SERVICE
2076      | routine. This error indicates a Port fatal error code.
2077      |
2078
2079      [PFE_CODE] :                               !Code equals %o'21'
2080      begin
2081      SA_VALUE = .RC25_ADDR [RCSA, ERR_CODE];      !Get error code from SA register
2082      |
2083      | Is the error code within the RC25 error code range, or is it
2084      | an generic, all controllers, error code?
2085      |
2086
2087      if .SA_VALUE gequ 200
2088      then
2089      begin
2090      PRINTB (.RC_STRUCTURE [.SA_VALUE - 200]);      !RC25 error code range
2091      end
2092      else
2093      begin
2094      PRINTB (.PFE_STRUCT [.SA_VALUE]);              !Generic error code
2095      end;
2096
2097      end;
2098

```

```

2099      | 'Return status error' code
2100      |
2101      | This indicates that a non-successful return status code was returned from an issued command.
2102      |
2103      |
2104      [RSE_CODE] :                               !Code equals %'31'
2105      begin
2106      PRINTB (.EMSG_STRUCT [MSG0]);
2107      |
2108      | Look at the UQPORT connection ID field to determine the type of response
2109      |
2110      |
2111      if .RET_EN$AD [CONN_ID] eql DUP
2112      then
2113      begin
2114      PRINTB (.SDUP_STRUCT [.RET_EN$AD [STATUS]]);
2115      end
2116      else
2117      begin
2118      PRINTB (.SMSCP_STRUCT [.RET_EN$AD [STA_CODE]]);
2119      end;
2120      |
2121      end;
2122      |
2123      | 'Port Portocol violation' error code
2124      |
2125      | A protocol violation error was detected during host processing of an issued command.
2126      |
2127      |
2128      [PVE_CODE] :                               !Code equals %'41'
2129      begin
2130      PRINTB (.EMSG_STRUCT [MSG1]);
2131      end;
2132      |
2133      | 'Remote program died' error code
2134      |
2135      | This indicates that the remote program running
2136      | in the DM machine did not responded within the
2137      | designated time out interval and that the progress
2138      | indicator was not increase after subsiquent time out
2139      | delays. It is assumed that the remote program is dead
2140      | and is treated as a fatal error.
2141      |
2142      |
2143      [RPD_CODE] :                               !Code equals %'51'
2144      begin
2145      PRINTB (.EMSG_STRUCT [MSG2]);
2146      end;
2147      |
2148      | 'Port to host synchronious error' code
2149      |
2150      |
2151      [PSE_CODE] :                               !Code equals %'61'
2152      begin
2153      PRINTB (.EMSG_STRUCT [MSG5]);
2154      end;
2155      |

```

```

: 2156      | 'Message length error' code
: 2157      |
: 2158      |
: 2159      | [MLE_CODE] :           !Code equals %'71'
: 2160      |     begin
: 2161      |     PRINTB (.EMSG_STRUCT [MSG6]);
: 2162      |     end;
: 2163      |
: 2164      | 'Unknown end code' error code
: 2165      |
: 2166      |
: 2167      | [UEC_CODE] :           !Code equals %'101'
: 2168      |     begin
: 2169      |     PRINTB (.EMSG_STRUCT [MSG7]);
: 2170      |     end;
: 2171      |
: 2172      | 'Adaptor purge request' error code
: 2173      |
: 2174      |
: 2175      | [APR_CODE] :           !Code equals %'201'
: 2176      |     begin
: 2177      |     PRINTB (.EMSG_STRUCT [MSG8]);
: 2178      |     end;
: 2179      |
: 2180      | 'Unknown interrupt' error code
: 2181      |
: 2182      |
: 2183      | [UIN_CODE] :           !Code equals %'301'
: 2184      |     begin
: 2185      |     PRINTB (.EMSG_STRUCT [MSG9]);
: 2186      |     end;
: 2187      |
: 2188      | 'ATTENTION MSG ENDCODE' error code
: 2189      |
: 2190      |
: 2191      | [ATN_CODE] :           !Code equals %'401'
: 2192      |     begin
: 2193      |     PRINTB (.EMSG_STRUCT [MSG12]);
: 2194      |     end;
: 2195      |
: 2196      | 'COMMAND MSG ENDCODE' error code
: 2197      |
: 2198      |
: 2199      | [CMD_CODE] :           !Code equals %'501'
: 2200      |     begin
: 2201      |     PRINTB (.EMSG_STRUCT [MSG13]);
: 2202      |     end;
: 2203      |
: 2204      | 'SERIOUS EXCEPTION' error code
: 2205      |
: 2206      |
: 2207      | [SEX_CODE] :           !Code equals %'601'
: 2208      |     begin
: 2209      |     PRINTB (.EMSG_STRUCT [MSG14]);
: 2210      |     end;
: 2211      |
: 2212      | 'INVALID COMMAND' error code

```



```

:
:
:
2213      !
2214
2215      [IVC_CODE] :                !Code equals %o'701'
2216          begin
2217          PRINTB (.EMSG_STRUCT [MSG15]);
2218          end;
2219      !
2220      ! 'UNKNOWN MESSAGE TYPE' error code
2221      !
2222
2223      [UMT_CODE] :                !Code equals %o'1001'
2224          begin
2225          PRINTB (.EMSG_STRUCT [MSG16]);
2226          end;
2227      !
2228      ! 'UNKNOWN MESSAGE NUMBER' error code
2229      !
2230
2231      [UMN_CODE] :                !Code equals %o'4001'
2232          begin
2233          PRINTB (.EMSG_STRUCT [MSG19]);
2234          end;
2235      !
2236      ! Out standing buffer slots are all filled up
2237      !
2238
2239      [OBF_CODE] :                !Code equals %o'2001'
2240          begin
2241          PRINTB (.EMSG_STRUCT [MSG17]);
2242          end;
2243      !
2244      ! Out standing command buffer out of sync error
2245      !
2246
2247      [OSE_CODE] :                !Code equals %o'3001'
2248          begin
2249          PRINTB (.EMSG_STRUCT [MSG18]);
2250          end;
2251      !
2252      ! File read error from local load media
2253      !
2254
2255      [FRE_CODE] :                !Code equals %o'5001'
2256          begin
2257          PRINTB (.EMSG_STRUCT [MSG20]);
2258          end;
2259      !
2260      ! Illegal FCT file length
2261      !
2262
2263      [ILL_FCT] :                 !Code equals %o'6001'
2264          begin
2265          PRINTB (.EMSG_STRUCT [MSG22]);
2266          end;
2267      !
2268      ! This is here to trap any unknow return status codes sent to this routine.
2269
:
:

```

CZRCH5
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.816;3 (10)

```

:      2270
:      2271      [otherwise] :      !Code equals non of the above
:      2272      begin
:      2273      PRINTB (.EMSG_STRUCT [MSG3]);
:      2274      end;
:      2275      tes;
:      2276
:      2277      |
:      2278      | All errors are fatal so init the RC25 and jump to the clean-up
:      2279      | code section to abort this units format.
:      2280
:      2281      RC25_ADDR [RCIP, RC ALL] = ONES;      !Init the controller
:      2282      WRT RC25 (RCIP, ONES);      !Init the controller
:      2283      DOCN;      !Jump to the clean-up code section
:      2284      end;

```

```

000000 004167 000000G      .SBTTL      DECODE MODULE DECLARATIONS      1985
000004 024646      DECODE::JSR      R1,$SAVE2      ;
000006 016701 000000G      CMP      -(SP),-(SP)      ;
000012 020127 000001      MOV      RET.STATUS,R1      ;
000016 001007      CMP      R1,#1      ;
000020 016746 000010G      BNE      1$      ;
000024 012746 000001      MOV      EMSG.STRUCT+10,-(SP)      ;
000030 010600      MOV      #1,-(SP)      ;
000032 104414      MOV      SP,R0      ; SP,*
000034 000567      TRAP     14      ;
000036 020127 000011      BR      11$      ;
000042 001007      CMP      R1,#11      ;
000044 016746 000052G      BNE      2$      ;
000050 012746 000001      MOV      EMSG.STRUCT+52,-(SP)      ;
000054 010600      MOV      #1,-(SP)      ;
000056 104414      MOV      SP,R0      ; SP,*
000060 000567      TRAP     14      ;
000062 020127 000021      BR      13$      ;
000066 001034      CMP      R1,#21      ;
000070 016700 000000G      BNE      4$      ;
000074 016066 000002 000002      MOV      RC25.ADDR,R0      ;
000102 016600 000002      MOV      2(R0),2(SP)      ; *RC$$REG
000106 042700 174000      MOV      2(SP),R0      ; RC$$REG,SA.VALUE
000112 020027 000310      BIC      #174000,R0      ; *SA.VALUE
000116 103410      CMP      R0,#310      ; SA.VALUE,*
000120 006300      BLO      3$      ;
000122 016046 177160G      ASL      R0      ;
000126 012746 000001      MOV      RC.STRUCTURE-620(R0),-(SP)      ;
000132 010600      MOV      #1,-(SP)      ;
000134 104414      MOV      SP,R0      ; SP,*
000136 000576      TRAP     14      ;
000140 006300      BR      17$      ;
000142 016046 000000G      ASL      R0      ;
000146 012746 000001      MOV      PFE.STRUCT(R0),-(SP)      ;
000152 010600      MOV      #1,-(SP)      ;
000154 104414      MOV      SP,R0      ; SP,*
000156 000566      TRAP     14      ;
000160 020127 000031      BR      17$      ;
000164 001044      CMP      R1,#31      ;
      BNE      7$      ;

```

CZRCH5
REV A PATCH 00 CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

000166	016746	000000G		MOV	EMSG.STRUCT,-(SP)	:	2106
000172	012746	000001		MOV	#1,-(SP)	:	
000176	010600			MOV	SP,R0	: SP,*	
000200	104414			TRAP	14	:	
000202	016702	000000G		MOV	RET.ENSAD,R2	:	2114
000206	010200			MOV	R2,R0	: RET.ENSAD,*	2111
000210	126027	000003	000002	CMPB	3(R0),#2	:	
000216	001012			BNE	5\$:	
000220	016200	000016		MOV	16(R2),R0	:	2114
000224	006300			ASL	R0	:	
000226	016016	000000G		MOV	SDUP.STRUCT(R0),(SP)	:	
000232	012746	000001		MOV	#1,-(SP)	:	
000236	010600			MOV	SP,R0	: SP,*	
000240	104414			TRAP	14	:	
000242	000413			BR	6\$:	2111
000244	116202	000016	5\$:	MOVB	16(R2),R2	:	2118
000250	042702	177740		BIC	#177740,R2	:	
000254	006302			ASL	R2	:	
000256	016216	000000G		MOV	SMSCP.STRUCT(R2),(SP)	:	
000262	012746	000001		MOV	#1,-(SP)	:	
000266	010600			MOV	SP,R0	: SP,*	
000270	104414			TRAP	14	:	
000272	005726		6\$:	TST	(SP)+	:	2105
000274	000567			BR	22\$:	2048
000276	020127	000041	7\$:	CMP	R1,#41	:	
000302	001007			BNE	8\$:	
000304	016746	000002G		MOV	EMSG.STRUCT+2,-(SP)	:	2130
000310	012746	000001		MOV	#1,-(SP)	:	
000314	010600			MOV	SP,R0	: SP,*	
000316	104414			TRAP	14	:	
000320	000567			BR	24\$:	2048
000322	020127	000051	8\$:	CMP	R1,#51	:	
000326	001007			BNE	9\$:	
000330	016746	000004G		MOV	EMSG.STRUCT+4,-(SP)	:	2145
000334	012746	000001		MOV	#1,-(SP)	:	
000340	010600			MOV	SP,R0	: SP,*	
000342	104414			TRAP	14	:	
000344	000567			BR	26\$:	2048
000346	020127	000061	9\$:	CMP	R1,#61	:	
000352	001007			BNE	10\$:	
000354	016746	000012G		MOV	EMSG.STRUCT+12,-(SP)	:	2153
000360	012746	000001		MOV	#1,-(SP)	:	
000364	010600			MOV	SP,R0	: SP,*	
000366	104414			TRAP	14	:	
000370	000567			BR	28\$:	2048
000372	020127	000071	10\$:	CMP	R1,#71	:	
000376	001007			BNE	12\$:	
000400	016746	000014G		MOV	EMSG.STRUCT+14,-(SP)	:	2161
000404	012746	000001		MOV	#1,-(SP)	:	
000410	010600			MOV	SP,R0	: SP,*	
000412	104414			TRAP	14	:	
000414	000567		11\$:	BR	30\$:	2048
000416	020127	000101	12\$:	CMP	R1,#101	:	
000422	001007			BNE	14\$:	
000424	016746	000016G		MOV	EMSG.STRUCT+16,-(SP)	:	2169
000430	012746	000001		MOV	#1,-(SP)	:	
000434	010600			MOV	SP,R0	: SP,*	

CZRCH5
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (10)

000436	104414		TRAP	14			
000440	000576		BR	33\$:		2048
000442	020127	000201	13\$: 14\$: CMP	R1,#201			
000446	001007		BNE	15\$			
000450	016746	000020G	MOV	EMSG.STRUCT+20,-(SP)	:		2177
000454	012746	000001	MOV	#1,-(SP)			
000460	010600		MOV	SP,R0	:	SP,*	
000462	104414		TRAP	14			
000464	000564		BR	33\$:		2048
000466	020127	000301	15\$: CMP	R1,#301			
000472	001007		BNE	16\$			
000474	016746	000022G	MOV	EMSG.STRUCT+22,-(SP)	:		2185
000500	012746	000001	MOV	#1,-(SP)			
000504	010600		MOV	SP,R0	:	SP,*	
000506	104414		TRAP	14			
000510	000552		BR	33\$:		2048
000512	020127	000401	16\$: CMP	R1,#401			
000516	001007		BNE	18\$			
000520	016746	000030G	MOV	EMSG.STRUCT+30,-(SP)	:		2193
000524	012746	000001	MOV	#1,-(SP)			
000530	010600		MOV	SP,R0	:	SP,*	
000532	104414		TRAP	14			
000534	000540		BR	33\$:		2048
000536	020127	000501	17\$: 18\$: CMP	R1,#501			
000542	001007		BNE	19\$			
000544	016746	000032G	MOV	EMSG.STRUCT+32,-(SP)	:		2201
000550	012746	000001	MOV	#1,-(SP)			
000554	010600		MOV	SP,R0	:	SP,*	
000556	104414		TRAP	14			
000560	000526		BR	33\$:		2048
000562	020127	000601	19\$: CMP	R1,#601			
000566	001007		BNE	20\$			
000570	016746	000034G	MOV	EMSG.STRUCT+34,-(SP)	:		2209
000574	012746	000001	MOV	#1,-(SP)			
000600	010600		MOV	SP,R0	:	SP,*	
000602	104414		TRAP	14			
000604	000514		BR	33\$:		2048
000606	020127	000701	20\$: CMP	R1,#701			
000612	001007		BNE	21\$			
000614	016746	000036G	MOV	EMSG.STRUCT+36,-(SP)	:		2217
000620	012746	000001	MOV	#1,-(SP)			
000624	010600		MOV	SP,R0	:	SP,*	
000626	104414		TRAP	14			
000630	000502		BR	33\$:		2048
000632	020127	001001	21\$: CMP	R1,#1001			
000636	001007		BNE	23\$			
000640	016746	000040G	MOV	EMSG.STRUCT+40,-(SP)	:		2225
000644	012746	000001	MOV	#1,-(SP)			
000650	010600		MOV	SP,R0	:	SP,*	
000652	104414		TRAP	14			
000654	000470		BR	33\$:		2048
000656	020127	004001	22\$: 23\$: CMP	R1,#4001			
000662	001007		BNE	25\$			
000664	016746	000046G	MOV	EMSG.STRUCT+46,-(SP)	:		2233
000670	012746	000001	MOV	#1,-(SP)			
000674	010600		MOV	SP,R0	:	SP,*	
000676	104414		TRAP	14			

CZRCH5
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (10)

000700	000456		24\$:	BR	33\$:	2048
000702	020127	002001	25\$:	CMP	R1,#2001	:	
000706	001007			BNE	27\$:	
000710	016746	000042G		MOV	EMSG.STRUCT+42,-(SP)	:	2241
000714	012746	000001		MOV	#1,-(SP)	:	
000720	010600			MOV	SP,R0	: SP,*	
000722	104414			TRAP	14	:	
000724	000444		26\$:	BR	33\$:	2048
000726	020127	003001	27\$:	CMP	R1,#3001	:	
000732	001007			BNE	29\$:	
000734	016746	000044G		MOV	EMSG.STRUCT+44,-(SP)	:	2249
000740	012746	000001		MOV	#1,-(SP)	:	
000744	010600			MOV	SP,R0	: SP,*	
000746	104414			TRAP	14	:	
000750	000432		28\$:	BR	33\$:	2048
000752	020127	005001	29\$:	CMP	R1,#5001	:	
000756	001007			BNE	31\$:	
000760	016746	000050G		MOV	EMSG.STRUCT+50,-(SP)	:	2257
000764	012746	000001		MOV	#1,-(SP)	:	
000770	010600			MOV	SP,R0	: SP,*	
000772	104414			TRAP	14	:	
000774	000420		30\$:	BR	33\$:	2048
000776	020127	006001	31\$:	CMP	R1,#6001	:	
001002	001007			BNE	32\$:	
001004	016746	000054G		MOV	EMSG.STRUCT+54,-(SP)	:	2265
001010	012746	000001		MOV	#1,-(SP)	:	
001014	010600			MOV	SP,R0	: SP,*	
001016	104414			TRAP	14	:	
001020	000406			BR	33\$:	2048
001022	016746	000006G	32\$:	MOV	EMSG.STRUCT+6,-(SP)	:	2273
001026	012746	000001		MOV	#1,-(SP)	:	
001032	010600			MOV	SP,R0	: SP,*	
001034	104414			TRAP	14	:	
001036	012766	177777 000004	33\$:	MOV	#-1,4(SP)	: *,RC\$\$.REG	2281
001044	012700	177777		MOV	#-1,R0	: *,RC\$\$.REG	2282
001050	010077	000000G		MOV	R0,@RC25.ADDR	: RC\$\$.REG,*	
001054	104444			TRAP	44	:	
001056	062706	000010		ADD	#10,SP	:	1985
001062	000207			RTS	PC	:	

: Routine Size: 282 words, Routine Base: AC\$CODE + 0404
: Maximum stack depth per invocation: 10 words

: 2285

```

:      2286 global routine DUP$I_SERVICE : INT_LNK$TYP novalue =      !Signals receive queue entry
:      2287
:      2288 !++
:      2289 Functional Description :
:      2290 The transmission of a message will result in a host interrupt if and
:      2291 only if interrupts were armed suitably during initialization and one
:      2292 of the following conditions has been met:
:      2293
:      2294 1. The message was a command with F=1 and the port's fetching it
:      2295 caused the command ring to transition from full to not-full.
:      2296 (This interrupt means that the host may place another command
:      2297 in the ring.)
:      2298
:      2299 2. The message was a response with F=1 and the port's depositing
:      2300 it caused the response ring to transition from empty to not-
:      2301 empty. (This interrupt means that there is a response for
:      2302 the host to process.)
:      2303
:      2304 3. The port is interfaced to the host via a bus abapter and a
:      2305 command requires the port/controller to re-access a given
:      2306 location during data transfer. (This interrupt means that
:      2307 the port/controller is requesting the host to purge the
:      2308 indicated chanel of the bus adapter.)
:      2309
:      2310 This interrupt service routine is entered when any of the above
:      2311 conditions occure. When entered it will be determined what type
:      2312 interrupt was executed and take the necessary action.
:      2313
:      2314 Formal Parameters :
:      2315 none
:      2316
:      2317 Implicit Inputs :
:      2318 Nrd_slot: A global flag which points to the next receive descriptor
:      2319 slot where the port/controller should be polling on and
:      2320 where to expect the first response packet to process.
:      2321
:      2322 Implicit Outputs :
:      2323 Ret_status: This global flag is the mechinism by which these DUP
:      2324 and UQ Port protocol routines pass status code back to
:      2325 to the host routine's requesting communications over
:      2326 the established connections. The status returned is
:      2327 decoded by the caller to determine if an error or bad
:      2328 response packet status was discovered.
:      2329
:      2330 Out$std_buf: This buffer is used to save all commands issued to the
:      2331 port and are considered outstanding when in this buffer.
:      2332 This interrupt service routine will indicate this command
:      2333 is nolonger outstanding by setting the rec_flg in the
:      2334 slot matching this response envelope command_ref number.
:      2335
:      2336 Completion Codes :
:      2337 none
:      2338
:      2339 Side Effects :
:      2340 none
:      2341 --
:      2342

```

```
2343 begin
2344
2345 local
2346     TEMP,                !Holds nrd_slot + 1 reference
2347     FOUND_CMD,          !Found command flag
2348     REF_NUM;            !Stores response packets cmd ref number
2349
2350
2351 +
2352 | Before this interrupt service routine does anything
2353 | look at the SA register for any port fatal errors
2354 | posted. If there are errors posted then report the
2355 | error and kick the bucket.
2356 | -
2357 if .RC25_ADDR [RCSA, ERR_BIT]          !Are there any errors posted
2358 then
2359     begin
2360         RET_STATUS = PFE_CODE;          !Indicate the error type
2361         DECODE ();                     !Decode and print the error
2362         return;                          !Just for show. Decode will kill it
2363     end;
2364
2365 +
2366 | See what kind of interrupt got us here.
2367 |
2368 | We could have a:
2369 |
2370 | 1. Response ring transition interrupt.
2371 | 2. Send ring transition interrupt.
2372 | 3. A adaptor purge request interrupt (which is
2373 |    illegal running under the PDP-11 Diagnostic
2374 |    supervisor and is flagged as a fatal controller
2375 |    error.)
2376 | 4. Or an unknown interrupt not known by this program
2377 |    which also results in a fatal controller error.
2378 | -
2379 |
2380 |
2381 |
2382 |
2383 | Check to see if we get here because of a response ring
2384 | transition interrupt. This is more likely to be the
2385 | most frequent interrupt so check it first.
2386 |
2387 |
2388 if .HEAD_AREA [RSP_INT]
2389 then
2390     begin
2391         GET_NRD ();                    !Get the resp slot location to process
2392     |
2393     | Check the host protocol for being out of sequence
2394     | with the controller by making sure that this slot
2395     | is owned by the host (meaning that there is a resp
2396     | envelope at this ring slot to process.
2397     |
2398     |
2399     if .RECEIVE_RING [.NRD_SLOT, OWN_BIT] nequ HOST_OWNED !Is this owned by host
```

```
2400     then
2401     begin
2402     RET_STATUS = PSE_CODE;
2403     DECODE ();
2404     return;
2405     end;
2406
2407     !+
2408     ! Per DUP protocol once interrupted due to a response ring
2409     ! interrupt, the host code should process all response packets
2410     ! found in the response ring. This while loop will continue
2411     ! to process the response packets in the response ring until
2412     ! none remain.
2413     !-
2414
2415     while TRUE do
2416     begin
2417     BREAK;
2418
2419     ! Load the Reference structure 'Ret_en$ad' with the address
2420     ! of this response envelope to process (The minus %o'4' is
2421     ! done to address the first word in the envelope packet
2422     ! and is equal to location 'text-4').
2423
2424     RET_EN$AD = (.RECEIVE_RING [.NRD_SLOT, LO_EN$AD]) - %o'4';
2425
2426     !+
2427     ! Test the end packet for its possible three end types.
2428
2429     ! End message opcodes (also called endcodes) are formed by adding the end
2430     ! message flag to the command opcode. For example, a READ commands end
2431     ! message contains the value OP.RED + OP.END in its opcode field. The Invalid
2432     ! command end message contains just the end message flag (i.e., OP.END) in
2433     ! its opcode field. The serious exception opcode shown above (i.e. OP.SEX +
2434     ! OP.END) in its opcode field.
2435
2436     ! Commands opcode bits 6 and 7 indicate the type of message (command, end or
2437     ! attention message. Command opcodes bits 3 through 5 indicate the command
2438     ! category (immediate, sequential or no-sequential) and whether or not the
2439     ! command includes a buffer descriptor.
2440
2441     ! See MSCP document appendix 'A-1 NOTE:' for more information on this topic.
2442     !-
2443
2444     selectoneu .RET_EN$AD [TYP$MSG] of !Select the endpacket size
2445     set
2446
2447     [END$MSG] :
2448     begin
2449
2450     !+
2451     ! Select off of the endcode to make sure the communications
2452     ! mechanism transferred the correct number of byte for this
2453     ! end packet. If this number of bytes transferred is not
2454     ! correct for the commands end packet then load the error
2455     ! code into return status, call decode to report the
2456     ! error and kick the bucket and die. This endcode is formed
```


CZRCH5
REV A PATCH 00CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS11-Jul-1983 16:06:44
7-Jul-1983 08:24:58VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (11)

```

2457      | by adding the end message flag %o'200' to the commands
2458      | opcode.
2459      |
2460      |
2461      | selectoneu .RET_EN$AD [ENDCODE] of
2462      | set
2463      |
2464      | "RECEIVE DATA" command end packet
2465      |
2466      |
2467      | [EOP_RED] :
2468      | begin
2469      |
2470      |   if .RET_EN$AD [MSG_LENGTH] nequ ESZ_RED      !Is the byte count correct
2471      |   then
2472      |     begin
2473      |       RET STATUS = MLE_CODE; !Return a 'message length error code'
2474      |       DECODE ();           !Report the error and kick the bucket
2475      |       return;              !Just for show. Decode kills it
2476      |     end;
2477      |
2478      |   end;
2479      |
2480      | "SEND DATA" command end packet
2481      |
2482      |
2483      | [EOP_SED] :
2484      | begin
2485      |
2486      |   if .RET_EN$AD [MSG_LENGTH] nequ ESZ_SED      !Is the byte count correct
2487      |   then
2488      |     begin
2489      |       RET STATUS = MLE_CODE; !Return a 'message length error code'
2490      |       DECODE ();           !Report the error and kick the bucket
2491      |       return;              !Just for show. Decode kills it
2492      |     end;
2493      |
2494      |   end;
2495      |
2496      | " GET DUST STATUS" command end packet
2497      |
2498      |
2499      | [EOP_GDS] :
2500      | begin
2501      |
2502      |   if .RET_EN$AD [MSG_LENGTH] nequ ESZ_GDS      !Is the byte count correct
2503      |   then
2504      |     begin
2505      |       RET STATUS = MLE_CODE; !Return a 'message length error code'
2506      |       DECODE ();           !Report the error and kick the bucket
2507      |       return;              !Just for show. Decode kills it
2508      |     end;
2509      |
2510      |   end;
2511      |
2512      | "EXECUTE SUPPLIED PROGRAM" command end packet
2513      |

```

CZRCH5
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

L 15

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (11)

SEQ 193
Page 28

```

2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570

```

```

[EOP ESP] :
  begin
    if .RET_EN$AD [MSG_LENGTH] nequ ESZ_ESP      !Is the byte count correct
    then
      begin
        RET_STATUS = MLE_CODE; !Return a 'message length error code'
        DECODE ();      !Report the error and kick the bucket
        return;         !Just for show. Decode kills it
      end;
    end;
  ! 'EXECUTE LOCAL PROGRAM' command end packet
[EOP ELP] :
  begin
    if .RET_EN$AD [MSG_LENGTH] nequ ESZ_ELP      !Is the byte count correct
    then
      begin
        RET_STATUS = MLE_CODE; !Return a 'message length error code'
        DECODE ();      !Report the error and kick the bucket
        return;         !Just for show. Decode kills it
      end;
    end;
  ! 'ABORT PROGRAM' command end packet
[EOP ABT] :
  begin
    if .RET_EN$AD [MSG_LENGTH] nequ ESZ_ABT      !Is the byte count correct
    then
      begin
        RET_STATUS = MLE_CODE; !Return a 'message length error code'
        DECODE ();      !Report the error and kick the bucket
        return;         !Just for show. Decode kills it
      end;
    end;
  ! 'SET CONTROLLER CHAR' command end packet
[EOP SCC] :
  begin
    if .RET_EN$AD [MSG_LENGTH] nequ ESZ_SCC      !Is the byte count correct
    then
      begin
        RET_STATUS = MLE_CODE; !Return a 'message length error code'
        DECODE ();      !Report the error and kick the bucket

```

CZRCH5
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (11)
SEQ 194
Page 29

```

:      2571          return;          !Just for show. Decode kills it
:      2572          end;
:      2573
:      2574          end;
:      2575          !
:      2576          !'ON LINE' command end packet
:      2577          !
:      2578
:      2579          [EOP_ONL] :
:      2580              begin
:      2581
:      2582              if .RET_EN$AD [MSG_LENGTH] nequ ESZ_ONL      !Is the byte count correct
:      2583              then
:      2584                  begin
:      2585                      RET_STATUS = MLE_CODE; !Return a 'message length error code'
:      2586                      DECODE ();      !Report the error and kick the bucket
:      2587                      return;      !Just for show. Decode kills it
:      2588                      end;
:      2589
:      2590              end;
:      2591
:      2592          ! The 'OP_END' end message flag all by its self tells
:      2593          ! us that the controller is flagging us of an illegal
:      2594          ! command sent over the connection. Error and kick the
:      2595          ! bucket.
:      2596          !
:      2597
:      2598          [OP_END] :
:      2599              begin
:      2600                  RET_STATUS = IVC_CODE;
:      2601                  DECODE ();
:      2602                  return;
:      2603                  end;
:      2604
:      2605          ! The controller is telling us that a serious exception
:      2606          ! has occured. Error and kick the bucket.
:      2607          !
:      2608
:      2609          [EOP_SEX] :
:      2610              begin
:      2611                  RET_STATUS = SEX_CODE;
:      2612                  DECODE ();
:      2613                  return;
:      2614                  end;
:      2615
:      2616          ! Unknown end packet endcode type
:      2617          !
:      2618
:      2619          [otherwise] :
:      2620              begin
:      2621                  RET_STATUS = UEC_CODE;      !Return an 'unknown end code'
:      2622                  DECODE ();      !Report the error and kick the bucket
:      2623                  return;      !Just for show. Decode kills it
:      2624                  end;
:      2625          tes;
:      2626
:      2627          !

```

CZRCH5
REV A PATCH 00CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS11-Jul-1983 16:06:44
7-Jul-1983 08:24:58VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (11)

```
2628 : The port/controller sent the endpacket over the connection
2629 : with out any problems. Now find this commands owner in the
2630 : out$standing buffer and indicate to them that the command
2631 : has been received.
2632 :
2633 REF_NUM = .RET_EN$AD [CMD_LREF]; !Get this rec packets cmd ref number
2634 :
2635 : Search the outstanding command buffer for this commands
2636 : reference number.
2637 :
2638 : If found, load the buffer location with the ret_en$ad
2639 : and set the received flag to signify that this command
2640 : has been received by this interrupt service routine.
2641 :
2642 FOUND_CMD = FALSE; !Clear the found cmd flag
2643 :
2644 incru i from 0 to REC_ALLOCATE - 1 do !Search the buffer
2645 :
2646 if .OUT$STD_BUF [.i, CMD_REF] eqlu .REF_NUM !Is this the cmd ref
2647 then
2648 begin
2649 OUT$STD_BUF [.i, REC_FLG] = TRUE; !Indicate command is received
2650 OUT$STD_BUF [.i, ENV_ADR] = .RET_EN$AD; !Return envelope adrs
2651 FOUND_CMD = TRUE; !Indicate it was found
2652 exitloop; !Exit the loop
2653 end;
2654 :
2655 :
2656 : If the search through the command ref
2657 : buffer failed to find this commands cmd
2658 : reference number then die.
2659 :
2660 :
2661 if not .FOUND_CMD
2662 then
2663 begin
2664 RET_STATUS = PSE_CODE;
2665 DECODE ();
2666 return;
2667 end;
2668 :
2669 end; !End of END$MSG processing
2670 :
2671 : The set controller characteristics command
2672 : disabled the reporting of attentions messages
2673 : so treat this as a fatal error and die.
2674 :
2675 :
2676 [ATT$MSG] : !Attention end message type
2677 begin
2678 RET_STATUS = ATN_CODE;
2679 DECODE ();
2680 return;
2681 end;
2682 :
2683 : It doesn't make any since for this end message
2684 : packet not to have the end message flag added
```

CZRCH5
REV A PATCH 00CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS11-Jul-1983 16:06:44
7-Jul-1983 08:24:58VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (11)SEQ 196
Page 31

```

:      2685      ! to this command opcode as treat it as a fatal
:      2686      ! error and die.
:      2687      !
:      2688
:      2689      [CMD$MSG] :
:      2690      begin
:      2691      RET STATUS = CMD_CODE;
:      2692      DECODE ();
:      2693      return;
:      2694      end;
:      2695
:      2696      ! This end code type is of unknown origin so
:      2697      ! treat is as a fatal error and die.
:      2698      !
:      2699
:      2700      [otherwise] :
:      2701      begin
:      2702      RET STATUS = UEC_CODE;      !Unknown message type code received
:      2703      DECODE ();
:      2704      return;
:      2705      end;
:      2706      tes;
:      2707
:      2708
:      2709      ! Before we leave put this receive envelope message length
:      2710      ! field back to the envelope size, in bytes (Per UQ Spec).
:      2711      ! This size does not include the 2 UQ's words preceding the
:      2712      ! command text area.
:      2713
:      2714      RET_EN$AD [MSG_LENGTH] = RB_SIZE*2;
:      2715
:      2716      ! Return this receive slot descriptor back to
:      2717      ! the port to fullfill my part of the protocol.
:      2718
:      2719      RECEIVE_RING [.NRD_SLOT, OWN_BIT] = PORT_OWNED;
:      2720
:      2721      ! Look at the next response ring descriptor. If its
:      2722      ! host owned then continue this process else exit the
:      2723      ! loop. First see if the ring reference has wrapped
:      2724      ! around to the top of the ring.
:      2725
:      2726
:      2727      if .NRD_SLOT + 1 gtru REC_ALLOCATE - 1      !Has the ring ref wrapped around
:      2728      then
:      2729      TEMP = ZERO      !Wrap it back to zero dsc slot
:      2730      else
:      2731      TEMP = .NRD_SLOT + 1;      !Look at the next seq dsc slot
:      2732
:      2733
:      2734      ! Now see if the next receive descriptor slot is host owned.
:      2735
:      2736
:      2737      if .RECEIVE_RING [.TEMP, OWN_BIT] eqv HOST_OWNED      !Are we done yet
:      2738      then
:      2739      GET_NRD ()      !Get the next resp desc to process
:      2740      else
:      2741      exitloop;      !No more to do so exit

```

CZRCH5
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (11)

```

2742
2743         end;                                !End of WHILE LOOP
2744
2745         |
2746         | All response ring descriptors have been processed with out any
2747         | detected errors so return to the main host code with an pass
2748         | return code. But before we go clear out the interrupt indicators.
2749         |
2750         HEAD_AREA [RSP_INT] = ZEROS;          !Clear the interrupt indicator location
2751         HEAD_AREA [CMD_INT] = ZEROS;         !Clear out the indicator
2752         return RET_STATUS = PAS_CODE;       !Return a 'pass code
2753     end;
2754
2755     !*****END OF RESPONSE RING INTERRUPT PROCESSING*****
2756
2757     | +
2758     | A send ring transition interrupt could happen if at
2759     | some date only one descriptor slot is allocated for
2760     | commands.
2761     |
2762     | Clear the interrupt indicator if this is true and do
2763     | a return with no errors.
2764     | -
2765
2766     if .HEAD_AREA [CMD_INT]                   !Is this a com ring transition interrupt
2767     then
2768         begin
2769             HEAD_AREA [CMD_INT] = ZERO;      !Clear out the indicator
2770             return;                          !Continue on with the host code
2771         end;
2772
2773     !*****END OF COMMAND RING INTERRUPT PROCESSING*****
2774
2775     | +
2776     | Check to see if an adaptor purge is being requested
2777     | by the port/controller in order to complete excution
2778     | of a issued command. Remember that this is illegal
2779     | during PDP-11 formatting and is concidered to be a
2780     | fatal error.
2781     | -
2782
2783     if .HEAD_AREA [ADP_CH] nequ ZERO         !Is the an adaptor purge request?
2784     then
2785         begin
2786             RET_STATUS = APR_CODE;          !Indicate the error code
2787             DECODE ();                     !Report the error and kick the bucket
2788             return;                       !Just for show. Decode kills it
2789         end;
2790
2791     !*****END OF ADAPTOR PURGE INTERRUPT PROCESSING*****
2792
2793     | +
2794     | The host program has been interrupted by an unknown interrupt
2795     | source if the routine program flow reaches here.
2796     |
2797     | Load the error code into return status and call decode to take
2798     | appropriate action.

```

CZRCH5
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (11)

```

:      2799      !-
:      2800
:      2801      RET STATUS = UIN_CODE;
:      2802      DECODE ();
:      2803      return;
:      2804      end;

```

```

000000 010046      .SBTTL DUP$I.SERVICE MODULE DECLARATIONS
                                DUP$I.SERVICE::
                                MOV      R0,-(SP)      ;
000002 010146      MOV      R1,-(SP)
000004 010246      MOV      R2,-(SP)
000006 010346      MOV      R3,-(SP)
000010 010446      MOV      R4,-(SP)
000012 010546      MOV      R5,-(SP)
000014 016700 000000G      MOV      RC25.ADDR,R0      ;
000020 016046 000002      MOV      2(R0),-(SP)      ; *,RCSS.REG
000024 100004      BPL      1$
000026 012767 000021 000000G      MOV      #21,RET.STATUS      ;
000034 000546      BR      9$
000036 016700 000000G      1$: MOV      HEAD.AREA,R0      ;
000042 032760 000001 000006      BIT      #1,6(R0)
000050 001002      BNE      2$
000052 000167 000550      JMP      21$
000056 004767 176424      2$: JSR      PC,GET.NRD
000062 016700 000000G      MOV      NRD.SLOT,R0      ;
000066 006300      ASL      R0
000070 006300      ASL      R0
000072 066700 000000C      ADD      RECEIVE.RING,R0
000076 032760 100000 000002      BIT      #100000,2(R0)
000104 001404      BEQ      4$
000106 012767 000061 000000G      3$: MOV      #61,RET.STATUS      ;
000114 000570      BR      16$
000116 104422      4$: TRAP      22      ;
000120 016700 000000G      MOV      NRD.SLOT,R0
000124 006300      ASL      R0
000126 006300      ASL      R0
000130 066700 000000G      ADD      RECEIVE.RING,R0
000134 011067 000000G      MOV      (R0),RET.FNSAD
000140 162767 000004 000000G      SUB      #4,RET.ENSAD
000146 016701 000000G      MOV      RET.ENSAD,R1      ;
000152 116100 000014      MOV      14(R1),R0
000156 006200      ASR      R0
000160 006200      ASR      R0
000162 006200      ASR      R0
000164 006200      ASR      R0
000166 006200      ASR      R0
000170 006200      ASR      R0
000172 042700 177774      BIC      #177774,R0
000176 020027 000002      CMP      R0,#2
000202 001121      BNE      14$
000204 005002      CLR      R2      ;
000206 156102 000014      BIS      14(R1),R2
000212 020227 000205      CMP      R2,#205
000216 001007      BNE      6$
000220 021127 000060      5$: CMP      (R1),#60      ;

```

CZRCH5
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (11)

000224	001453			BEQ	10\$				
000226	012767	000071	000000G	MOV	#71,RET.STATUS	:			2473
000234	000520			BR	16\$:			2474
000236	020227	000204		6\$:	CMP	R2,#204	:		2461
000242	001766			BEQ	5\$:			2486
000244	020227	000201		CMP	R2,#201	:			2461
000250	001763			BEQ	5\$:			2502
000252	020227	000202		CMP	R2,#202	:			2461
000256	001760			BEQ	5\$:			2518
000260	020227	000203		CMP	R2,#203	:			2461
000264	001755			BEQ	5\$:			2534
000266	020227	000206		CMP	R2,#206	:			2461
000272	001752			BEQ	5\$:			2550
000274	020227	000204		CMP	R2,#204	:			2461
000300	001747			BEQ	5\$:			2566
000302	020227	000211		CMP	R2,#211	:			2461
000306	001744			BEQ	5\$:			2582
000310	020227	000200		CMP	R2,#200	:			2461
000314	001004			BNE	7\$:			
000316	012767	000701	000000G	MOV	#701,RET.STATUS	:			2600
000324	000565			BR	24\$:			2601
000326	020227	000207		7\$:	CMP	R2,#207	:		2461
000332	001004			BNE	8\$:			
000334	012767	000601	000000G	MOV	#601,RET.STATUS	:			2611
000342	000556			BR	24\$:			2612
000344	012767	000101	000000G	8\$:	MOV	#101,RET.STATUS	:		2621
000352	000552			9\$:	BR	24\$:		2622
000354	016700	000000G		10\$:	MOV	RET.ENSAD,R0	:		2633
000360	016005	000004			MOV	4(R0),R5	:	*,REF.NUM	
000364	005004				CLR	R4	:	FOUND.CMD	2642
000366	005000				CLR	R0	:	I	2644
000370	005001			11\$:	CLR	R1	:		2646
000372	156001	000000G			BISB	OUT\$STD.BUF(R0),R1	:	*(I),*	
000376	020105				CMP	R1,R5	:	*,REF.NUM	
000400	001011				BNE	12\$:		
000402	052760	100000	000000G		BIS	#100000,OUT\$STD.BUF(R0)	:	*,*(I)	2649
000410	016760	000000G	000002G		MOV	RET.ENSAD,OUT\$STD.BUF+2(R0)	:	*,*(I)	2650
000416	012704	000001			MOV	#1,R4	:	*,FOUND.CMD	2651
000422	000405				BR	13\$:		2648
000424	062700	000004		12\$:	ADD	#4,R0	:	*,I	2644
000430	020027	000014			CMP	R0,#14	:	I,*	
000434	101755				BLOS	11\$:		
000436	032704	000001		13\$:	BIT	#1,R4	:	*,FOUND.CMD	2661
000442	001016				BNE	17\$:		
000444	000620				BR	3\$:		2664
000446	020027	000001		14\$:	CMP	R0,#1	:		2444
000452	001004				BNE	15\$:		
000454	012767	000401	000000G		MOV	#401,RET.STATUS	:		2678
000462	000506				BR	24\$:		2679
000464	005700			15\$:	TST	R0	:		2444
000466	001326				BNE	8\$:		
000470	012767	000501	000000G		MOV	#501,RET.SiATUS	:		2691
000476	000500			16\$:	BR	24\$:		2692
000500	012777	000074	000000G	17\$:	MOV	#74,RET.ENSAD	:		2714
000506	016700	000000G			MOV	NRD.SLOT,R0	:		2719
000512	006300				ASL	R0	:		
000514	006300				ASL	R0	:		

CZRCH5
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555
SPIDER\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (11)

000516	066700	000000G		ADD	RECEIVE.RING,R0		
000522	052760	100000	000002	BIS	#100000,2(R0)		
000530	016700	000000G		MOV	NRD.SLOT,R0	:	2727
000534	005200			INC	R0		
000536	020027	000003		CMP	R0,#3		
000542	101402			BLOS	18\$		
000544	005003			CLR	R3	: TEMP	2729
000546	000401			BR	19\$:	2727
000550	010003		18\$:	MOV	R0,R3	: *,TEMP	2731
000552	010300		19\$:	MOV	R3,R0	: TEMP,*	2737
000554	006300			ASL	R0		
000556	006300			ASL	R0		
000560	066700	000000G		ADD	RECEIVE.RING,R0		
000564	032760	100000	000002	BIT	#100000,2(R0)		
000572	001004			BNE	20\$		
000574	004767	175706		JSR	PC,GET.NRD	:	2739
000600	000167	177312		JMP	4\$:	2737
000604	016700	000000G	20\$:	MOV	HEAD.AREA,R0	:	2750
000610	005060	000006		CLR	6(R0)		
000614	005060	000004		CLR	4(R0)	:	2751
000620	005067	000000G		CLR	RET.STATUS	:	2752
000624	000427			BR	25\$:	2390
000626	016700	000000G	21\$:	MOV	HEAD.AREA,R0	:	2766
000632	032760	000001	000004	BIT	#1,(R0)		
000640	001403			BEQ	22\$		
000642	005060	000004		CLR	4(R0)	:	2769
000646	000416			BR	25\$:	2768
000650	016700	000000G	22\$:	MOV	HEAD.AREA,R0	:	2783
000654	105760	000003		TSTB	3(R0)		
000660	001404			BEQ	23\$		
000662	012767	000201	000000G	MOV	#201,RET.STATUS	:	2786
000670	000403			BR	24\$:	2787
000672	012767	000301	000000G	MOV	#301,RET.STATUS	:	2801
000700	004767	176010	23\$:	JSR	PC,DECODE	:	2802
000704	005726		24\$:	TST	(SP)+	:	2286
000706	012605		25\$:	MOV	(SP)+,R5		
000710	012604			MOV	(SP)+,R4		
000712	012603			MOV	(SP)+,R3		
000714	012602			MOV	(SP)+,R2		
000716	012601			MOV	(SP)+,R1		
000720	012600			MOV	(SP)+,R0		
000722	000002			RTI			

: Routine Size: 234 words, Routine Base: AC\$CODE + 1470
: Maximum stack depth per invocation: 13 words

: 2805

CZRCH5
REV A PATCH 00CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS11-Jul-1983 16:06:44
7-Jul-1983 08:24:58VAX-11 Bliss-16 V3-555
SPIDER\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (12)

```

:      2806 global routine CTO_WAIT (TO_VALUE, REF_NUM, BUF$LOC) = !Controller time out wait
:      2807
:      2808 |++
:      2809 | Functional Description :
:      2810 |   This routine is called to wait for the port/controller to either
:      2811 |   complete the queued command or time out the command.
:      2812 | Formal Parameters :
:      2813 |   TO_VALUE      Indicate the time-out interval for this command.
:      2814 |   REF_NUM       This argument contains the unique reference number
:      2815 |                 assigned to this command being timed out by this
:      2816 |                 routine.
:      2817 |   BUF$LOC       This argument points to the out$std_buf location
:      2818 |                 where this command is saved. At this location the
:      2819 |                 received flag "rec_flg" bit is examined within the
:      2820 |                 the timeout loop and when it equals true will
:      2821 |                 signal that this command has been received by the
:      2822 |                 interrupt service routine.
:      2823 | Implicit Inputs :
:      2824 |   none
:      2825 | Implicit Outputs :
:      2826 |   The command word in the out$std buffer 'word zero of a command
:      2827 |   slot' is cleared out with the value %0'100000' to indicate this
:      2828 |   is an unused out$std_buffer slot and that it can be reused.
:      2829 | Completion Codes :
:      2830 |   There are two levels of return status returned by this routine.
:      2831 |   1. The DUP interrupt service returns to this routine a status code
:      2832 |      to indicate the success of the connection/communications mechanism
:      2833 |      to complete the queued command. If the port/controller does not
:      2834 |      time out then this return status is returned as this routines
:      2835 |      return status code.
:      2836 |   2. If the port/controller times out then the SA Register error bit
:      2837 |      is examined for the error bit set. If set then an port fatal
:      2838 |      error code is returned to the calling routine else a controller
:      2839 |      time out error code is returned.
:      2840 |   In all cases, if an error code is returned (bit 0 = 1) then the
:      2841 |   routine decode is called to decode the error code and does the
:      2842 |   necessary recovery actions.
:      2843 |   At the next higher level of return from this routine is another level
:      2844 |   of return status returned. This level test the success of the
:      2845 |   connection and also test the status field in the returned response
:      2846 |   envelope for the success of the controller to successfully complete
:      2847 |   the requested command.
:      2848 | Side Effects :
:      2849 |   none
:      2850 | --
:      2851 |
:      2852 | begin
:      2853 |
:      2854 | |++
:      2855 | | Before doing the timeout wait make sure that this buffer location
:      2856 | | that we're suppose to time out actually contains the command ref
:      2857 | | number that was sent to us via the formal argument. Error and
:      2858 | | kick the bucket if not the same.
:      2859 | | -
:      2860 |
:      2861 | if .OUT$STD_BUF [.BUF$LOC, CMD_REF] nequ .REF_NUM !Is this the same ref_num
:      2862 | then

```

```

2863     begin
2864     RET_STATUS = OSE_CODE;           !Indicate the error code
2865     DECODE ();                       !Call decode to report the error
2866     end;
2867
2868     !+
2869     ! Loop on a one micro second delay for the number of times
2870     ! requested by the caller. After each delay see if the flag
2871     ! "rec_flg" has been set yet. Return "Ret_status" and clear the
2872     ! command word to %o'100000' to indicate this command has been
2873     ! received if this flag gets set before the timer expires.
2874     ! Return a error code if the timer expires before the flag gets set.
2875     !-
2876
2877     incru i from 0 to .TO_VALUE do      !Loop for time-out_value
2878     begin
2879     DELAY (C_US);                       !Do the one micro second delay
2880
2881     ! Exit routine with the DUP interrupt service routines "ret_status" if
2882     ! "rec_flg" got set before the timer expires.
2883
2884
2885     if .OUT$STD_BUF [.BUF$LOC, REC_FLG] eqlu TRUE !Is this command received yet
2886     then
2887     begin
2888     OUT$STD_BUF [.BUF$LOC, CMD_WRD] = %o'100000'; !Return the slot to the unused state
2889     return .RET_STATUS; !Return the interrupt service status
2890     end;
2891
2892
2893     !
2894     ! Check the SA register for posted port errors just in case the
2895     ! port does not interrupt the host when a port error occurs.
2896
2897     if .RC25_ADDR [RCSA, ERR_BIT] then return RET_STATUS = PFE_CODE;
2898
2899     BREAK; !Service any control C's
2900     end;
2901
2902
2903     !
2904     ! The port/controller timed out if the code reached here. Return an error
2905     ! code to the caller and exit the routine.
2906
2907     if .RC25_ADDR [RCSA, ERR_BIT] !Is the SA error bit set
2908     then
2909     return RET_STATUS = PFE_CODE !Port timed out with fatal error
2910     else
2911     return RET_STATUS = CTO_CODE; !Port just timed out
2912
2913     end;

```

.GLOBL L\$DLY

.SBTTL CTO.WAIT MODULE DECLARATIONS

CZRCH5
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (12)

000000	004167	000000G		CTO.WAIT::					
000004	162706	000006		JSR	R1,\$SAVE3	:			2806
000010	016600	000020		SUB	#6,SP	:			
000014	006300			MOV	20(SP),R0	:	BUF\$LOC,*		2861
000016	006300			ASL	R0	:			
000020	012703	000000G		ASL	R0	:			
000024	060003			MOV	#OUT\$STD.BUF,R3	:			
000026	005000			ADD	R0,R3	:			
000030	151300			CLR	R0	:			
000032	020066	000022		BISB	(R3),R0	:			
000036	001405			CMP	R0,22(SP)	:	*,REF.NUM		
000040	012767	003001	000000G	BEQ	1\$:			
000046	004767	175716		MOV	#3001,RET.STATUS	:			2864
000052	005002			JSR	PC,DECODE	:			2865
000054	000436			1\$: CLR	R2	:	I		2877
000056	012701	000001		BR	10\$:			
000062	001411			2\$: MOV	#1,R1	:	*,\$\$TMP2		2879
000064	016700	000000G		3\$: BEQ	6\$:			
000070	001404			MOV	L\$DLY,R0	:	*,\$\$TMP1		
000072	005066	000004		BEQ	5\$:			
000076	005300			4\$: CLR	4(SP)	:	\$\$TMP		
000100	001374			DEC	R0	:	\$\$TMP1		
000102	005301			BNE	4\$:			
000104	000766			5\$: DEC	R1	:	\$\$TMP2		
000106	005713			BR	3\$:			
000110	100005			6\$: TST	(R3)	:			2885
000112	012713	100000		BPL	7\$:			
000116	016700	000000G		MOV	#-100000,(R3)	:			2888
000122	000427			MOV	RET.STATUS,R0	:			2887
000124	016700	000000G		BR	12\$:			
000130	016066	000002	000002	7\$: MOV	RC25.ADDR,R0	:			2897
000136	100003			MOV	2(R0),2(SP)	:	*,RC\$\$REG		
000140	012700	000021		BPL	9\$:			
000144	000414			8\$: MOV	#21,R0	:			
000146	104422			BR	11\$:			
000150	005202			9\$: TRAP	22	:			
000152	020266	000024		INC	R2	:	I		2877
000156	101737			10\$: CMP	R2,24(SP)	:	I,TO.VALUE		
000160	016700	000000G		BLOS	2\$:			
000164	016016	000002		MOV	RC25.ADDR,R0	:			2907
000170	100763			MOV	2(R0),(SP)	:	*,RC\$\$REG		
000172	012700	000011		BMI	8\$:			2909
000176	010067	000000G		MOV	#11,R0	:			2911
000202	062706	000006		11\$: MOV	R0,RET.STATUS	:			
000206	000207			12\$: ADD	#6,SP	:			2806
				RTS	PC	:			

; Routine Size: 68 words, Routine Base: AC\$CODE + 2414
; Maximum stack depth per invocation: 9 words

; 2914

```

2915 global routine ABORT = !Aborts remote program
2916
2917 !++
2918 Functional Description :
2919 The abort program command is used to terminate the execution of a
2920 remote program in an orderly fashion. When a successful response
2921 is received to this command the remote program has stopped
2922 executing and the server is in idle state. Note that the sending
2923 of this command does not preclude further send data or receive data
2924 exchanges: On the contrary, the remote program may be designed to
2925 send out termination status and possibly even ask questions during
2926 its forced-exit sequence. The time out for this command is a fixed
2927 10 seconds and if a response is not received by then the
2928 connection to the dust should be terminated. This command is only
2929 legal if the dust is in active state.
2930
2931 Formal Parameters :
2932 none
2933
2934 Implicit Inputs :
2935 NSD_SLOT This global storage gets loaded by the routine
2936 'Get_nsd' and in it is stored the next send ring
2937 descriptor slot where the port/controller should
2938 be polling on and the place to put this commands
2939 command packet.
2940
2941 Implicit Outputs :
2942 none
2943
2944 Completion Codes :
2945 RET_STATUS: Return status passes back to the calling routine
2946 the status of the just issued command.
2947
2948 Side Effects :
2949 Any remote program running in the controllers DM machine will
2950 be aborted.
2951 --
2952
2953 begin
2954
2955 local
2956 REF_NUM, !Stores unique cmd ref number
2957 ABO_BUF$LOC, !Stores outstanding cmd buffer location
2958 TEMP; !A place to put the read IP register data
2959
2960 !
2961 ! Before we load up the command packet up with all this good information get
2962 ! thenext send descriptor slot and a unique command reference number.
2963 !
2964 GET_NSD (); !Get the next send desc slot
2965 REF_NUM = GET_CMD$REF (); !Get a unique command ref num
2966 !
2967 ! UQ Port command envelope Header field definition
2968 !
2969 SND_ENVELOPE [.NSD_SLOT, MSG_LENGTH] = SZ_ABT; !Load the length of envelope
2970 SND_ENVELOPE [.NSD_SLOT, CREDITS] = ONE; !Load credits
2971 SND_ENVELOPE [.NSD_SLOT, MSG_TYPE] = 0; !Load message type

```

CZRCH5
REV A PATCH 00CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS11-Jul-1983 16:06:44
7-Jul-1983 08:24:58VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (13)

```

2972 SND_ENVELOPE [.NSD_SLOT, CONN_ID] = DUP;      !Load connection ID
2973
2974     DUP command envelope field definition
2975
2976 SND_ENVELOPE [.NSD_SLOT, CMD_LREF] = .REF_NUM;      !Define reference number
2977 SND_ENVELOPE [.NSD_SLOT, CMD_HREF] = ZERO;      !Hi order ref number
2978 SND_ENVELOPE [.NSD_SLOT, UN_USED] = ZERO;      !Unused low order
2979 SND_ENVELOPE [.NSD_SLOT, UN_HUSED] = ZERO;      !Unused hi order
2980 SND_ENVELOPE [.NSD_SLOT, OP_CODE] = OP_ABT;      !Load opcode
2981 SND_ENVELOPE [.NSD_SLOT, RSVD] = ZERO;      !Reserved field
2982 SND_ENVELOPE [.NSD_SLOT, MODIFIER] = ZERO;
2983
2984     Call the load outstanding command buffer routine
2985     and load this command into the buffer. The return
2986     from this routine will point us to the buifer location
2987     where this command is stored. Later we can look at
2988     this location to see if the interrupt service routine
2989     has received and process it.
2990
2991 ABO_BUF$LOC = LOAD_OUT$STD_BUF (.REF_NUM);      !Load the command
2992
2993 if .ABO_BUF$LOC eqlu OBF_CODE then DECODE ();      !Error if buffer is full
2994
2995
2996     Set the ownership bit to 1 giving this slot to the port/controller
2997
2998 SEND_RING [.NSD_SLOT, OWN_BIT] = PORT_OWNED;
2999
3000     Read the IP register to stimulate port polling
3001
3002 TEMP = .RC25_ADDR [RCIP, RC_ALL];
3003
3004     Time out the port/controller processing the command.
3005
3006     The first test tests the connections ability to
3007     respond to this command without any errors in the SA
3008     register and for the command not timing out.
3009
3010     The second tests the DUP server for good status. If
3011     bad status is sent back then an error code is returned
3012     to the calling routine where the routine "decode" will
3013     decode and take the appropriate recovery. The time
3014     out routine will loop on delaying and checking the hi
3015     bit of the first word in the out$std_buf for a true.
3016     When true signals us that the interrupt service routine
3017     has received the endpacket and no connection errors
3018     were detected.
3019
3020
3021 if CTO_WAIT (3000, .REF_NUM, .ABO_BUF$LOC) then DECODE ();      !Is return an error
3022
3023
3024     Get the return envelope address from the out$std_buf
3025     at this commands buffer location and check the packet
3026     for good status error and die if bad status was returned
3027
3028 RET_EN$AD = .OUT$STD_BUF [.ABO_BUF$LOC, ENV_ADR];      !Get the ret env adr

```

CZRCH5
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (13)

```

: 3029      |
: 3030      | Now test for good status
: 3031      |
: 3032      |
: 3033      | if .RET_EN$AD [STATUS] nequ ZERO
: 3034      | then
: 3035      |     return RET_STATUS = RSE_CODE
: 3036      | else
: 3037      |     return .RET_STATUS;
: 3038      |
: 3039      | end:

```

```

!Test the status
!Return a 'Response status err' code
!This ret_status is good or bad

```

```

000000 004167 000000G      .SBTTL  ABORT MODULE DECLARATIONS
000004 005746              ABORT:: JSR   R1,$SAVE2                ; 2915
000006 004767 175312      TST   -(SP)                ;
000012 004767 175454      JSR   PC,GET.NSD           ; 2964
000016 010002              JSR   PC,GET.CMD$REF       ; 2965
000020 016746 000000G      MOV   R0,R2                ; *,REF.NUM
000024 012746 000054      MOV   NSD.SLOT,-(SP)       ;
000030 004767 000000G      MOV   #54,-(SP)           ; 2969
000034 012760 000014 000000G JSR   PC,BL$MUL           ;
000042 012701 000002G      MOV   #14,SND.ENVELOPE(R0)
000046 060001              MOV   #SND.ENVELOPE+2,R1 ; 2970
000050 112711 000001      ADD   R0,R1                ;
000054 112761 000002 000001 MOVB  #1,(R1)               ; 2971
000062 010260 000004G      MOVB  #2,1(R1)            ; 2972
000066 005060 000006G      MOV   R2,SND.ENVELOPE+4(R0) ; REF.NUM,* 2976
000072 005060 000010G      CLR   SND.ENVELOPE+6(R0)   ; 2977
000076 005060 000012G      CLR   SND.ENVELOPE+10(R0)  ; 2978
000102 112760 000006 000014G CLR   SND.ENVELOPE+12(R0)   ; 2979
000110 105060 000015G      MOVB  #6,SND.ENVELOPE+14(R0) ; 2980
000114 005060 000016G      CLRB  SND.ENVELOPE+15(R0)  ; 2981
000120 010216              CLR   SND.ENVELOPE+16(R0)  ; 2982
000122 004767 175252      MOV   R2,(SP)              ; REF.NUM,* 2991
000126 010001              JSR   PC,LOAD.OUT$STD.BUF ;
000130 020127 002001      MOV   R0,R1                ; *,ABO.BUF$LOC
000134 001002              CMP   R1,#2001            ; ABO.BUF$LOC,* 2993
000136 004767 175416      BNE   1$                   ;
000142 016700 000000G      JSR   PC,DECODE           ;
000146 006300              MOV   NSD.SLOT,R0         ; 2998
000150 006300              ASL   R0                   ;
000152 066700 000000G      ASL   R0                   ;
000156 052760 100000 000002 ADD   SEND.RING,R0         ;
000164 017766 000000G 000004 BIS   #100000,2(R0)        ;
000172 016600 000004      MOV   @RC25.ADDR,4(SP)    ; *,RC$$REG 3002
000176 012716 005670      MOV   4(SP),R0            ; RC$$REG,TEMP
000202 010246              MOV   #5670,(SP)         ;
000204 010146              MOV   R2,-(SP)           ; REF.NUM,* 3021
000206 004767 177356      MOV   R1,-(SP)           ; ABO.BUF$LOC,*
000212 022626              JSR   PC,CTO.WAIT        ;
000214 006000              CMP   (SP)+,(SP)+       ;
000216 103002              ROR   R0                   ;
000220 004767 175334      BCC   2$                   ;
000224 010100              JSR   PC,DECODE           ;
000226 006300              MOV   R1,R0              ; ABO.BUF$LOC,* 3028
000226 006300              ASL   R0                   ;

```

CZRCH5
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (13)

000230	006300		ASL	R0		
000232	016067	000002G 000000G	MOV	OUT\$STD.BUF+2(R0),RET.ENSAD		
000240	016000	000002G	MOV	OUT\$STD.BUF+2(R0),R0	:	3033
000244	005760	000016	TST	16(R0)		
000250	001405		BEQ	3\$		
000252	012700	000031	MOV	#31,R0	:	3035
000256	010067	000000G	MOV	R0,RET.STATUS		
000262	000402		BR	4\$:	2953
000264	016700	000000G	MOV	RET.STATUS,R0		
000270	062706	000006	ADD	#6,SP	:	2915
000274	000207		RTS	PC		

: Routine Size: 95 words, Routine Base: AC\$CODE + 2624
, Maximum stack depth per invocation: 9 words

: 3040

CZRCH5
REV A PATCH 00CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS11-Jul-1983 16:06:44
7-Jul-1983 08:24:58VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (14)

```

3041 global routine GET_DUST_STATUS =           !Gets DUP server status
3042
3043 !++
3044 Functional Description :
3045 This command allows the host program to interrogate the DUP server
3046 to determine its characteristics, its state and the state of the
3047 program currently running (if any). It is legal in either idle or
3048 active state and does not affect the state of server. It has a
3049 fixed timeout interval of 3 seconds. If the response times out, the
3050 host should break the connection.
3051
3052 Formal Parameters :
3053 none
3054
3055 Implicit Inputs :
3056 NSD_SLOT This global storage gets loaded by the routine
3057 'Get_nsd' and in it is stored the next send ring
3058 descriptor slot where the port/controller should
3059 be polling on and the place to put this commands
3060 command packet.
3061
3062 Implicit Outputs :
3063 none
3064
3065 Completion Codes :
3066 RET_STATUS: Return status passes back to the calling routine
3067 the status of the just issued command.
3068
3069 Side Effects :
3070 --
3071
3072 begin
3073
3074 local
3075     REF_NUM,           !Stores unique cmd ref number
3076     GDS_BUF$LOC,     !Stores outstanding cmd buffer location
3077     TEMP;            !A place to put the IP read data
3078
3079
3080 : Before we load up the command packet up with all this good information get
3081 : the next send descriptor slot and a unique command reference number.
3082
3083 GET_NSD ();           !Get the next send desc slot
3084 REF_NUM = GET_CMD$REF (); !Get a unique command ref num
3085
3086 : UQ Port command envelope Header field definition
3087
3088 SND_ENVELOPE [.NSD_SLOT, MSG_LENGTH] = S7_GDS; !Load the envelope size
3089 SND_ENVELOPE [.NSD_SLOT, CREDITS] = ONE; !Load the credit size
3090 SND_ENVELOPE [.NSD_SLOT, MSG_TYPE] = 0; !Load the message type (Sequential)
3091 SND_ENVELOPE [.NSD_SLOT, CONN_ID] = DUP; !Load the connection ID
3092
3093 : DUP generic command envelope field definition
3094
3095 SND_ENVELOPE [.NSD_SLOT, CMD_LREF] = .REF_NUM; !Load command reference number
3096 SND_ENVELOPE [.NSD_SLOT, CMD_HREF] = ZERO; !Command reference low order
3097 SND_ENVELOPE [.NSD_SLOT, UN_[USED]] = ZERO; !Low order unused

```

CZRCH5
REV A PATCH 00CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS11-Jul-1983 16:06:44
7-Jul-1983 08:24:58VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (14)

SEQ 209

Page 44

```

3098 SND_ENVELOPE [.NSD_SLOT, UN_HUSED] = ZERO; !Hi order unused
3099 SND_ENVELOPE [.NSD_SLOT, OP_CODE] = OP_GDS; !Load opcode
3100 SND_ENVELOPE [.NSD_SLOT, RSVD] = ZERO; !Reserved field
3101 SND_ENVELOPE [.NSD_SLOT, MODIFIER] = ZERO; !Load modifier field
3102
3103 : Call the load out$standing command buffer routine
3104 : and load this command into the buffer. The return
3105 : from this routine will point us to the buffer location
3106 : where this command is stored. Later we can look at
3107 : this location to see if the interrupt service routine
3108 : has received and process it.
3109
3110 GDS_BUF$LOC = LOAD_OUT$STD_BUF (.REF_NUM); !Load the command
3111
3112 if .GDS_BUF$LOC eqlu OBF_CODE then DECODE (); !Error if buffer is full
3113
3114 :
3115 : Set the ownership bit to 1 giving this slot to the port/controller
3116
3117 SEND_RING [.NSD_SLOT, OWN_BIT] = PORT_OWNED;
3118
3119 : Read the IP register to stimulate port polling
3120
3121 TEMP = .RC25_ADDR [RCIP, RC_ALL];
3122
3123 : Time out the port/controller processing the command.
3124
3125 : The first test tests the connections ability to
3126 : respond to this command without any errors in the SA
3127 : register and for the command not timing out.
3128
3129 : The second tests the DUP server for good status. If
3130 : bad status is sent back then an error code is returned
3131 : to the calling routine where the routine "decode" will
3132 : decode and take the appropriate recovery. The time
3133 : out routine will loop on delaying and checking the hi
3134 : bit of the first word in the out$std_buf for a true.
3135 : When true signals us that the interrupt service routine
3136 : has received the endpacket and no connection errors
3137 : were detected.
3138
3139
3140 if CTO_WAIT (3500, .REF_NUM, .GDS_BUF$LOC) then DECODE (); !Is return an error
3141
3142 :
3143 : Get the return envelope address from the out$std_buf
3144 : at this commands buffer location and check the packet
3145 : for good status error and die if bad status was returned
3146
3147 RET_EN$AD = .OUT$STD_BUF [.GDS_BUF$LOC, ENV_ADR]; !Get the ret env adr
3148
3149 : Now test for good status
3150
3151
3152 if .RET_EN$AD [STATUS] nequ ZERO !Test the status
3153 then
3154 return RET_STATUS = RSE_CODE !Return a 'Response status err' code

```


CZRCH5 CZRCH RC25 DISK FORMATTER
REV A PATCH 00 MODULE DECLARATIONS

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (14)

000256	010067	000000G		MOV	RO,RET.STATUS		
000262	000402			BR	4\$:	3072
000264	016700	000000G	3\$:	MOV	RET.STATUS,RO		
000270	062706	000006	4\$:	ADD	#6,SP	:	3041
000274	000207			RTS	PC		

: Routine Size: 95 words, Routine Base: AC\$CODE + 3122
: Maximum stack depth per invocation: 9 words

: 3159

CZRCH5
REV A PATCH 00CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555

SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (15)

```

3160 global routine EX_SUP_PROG =                !Executes supplied program
3161
3162 !++
3163 ! Functional Description :
3164 ! This command causes the server to transfer the program from host
3165 ! memory to an area in the controller and start its execution. The
3166 ! host supplies the address and length (in bytes) of a buffer
3167 ! containing the program header and initial load; the starting
3168 ! of the program, its memory requirements and any relocation information
3169 ! needed to run under the server are in the program header in a format
3170 ! which is none of the host business. This command is only legal when
3171 ! the server is in the idle state and return of a successful end packet
3172 ! puts the server into to active state.
3173
3174 ! The time out interval for this command is 30 seconds.
3175
3176 ! Formal Parameters :
3177 ! none
3178
3179 ! Implicit Inputs :
3180 ! NSD_SLOT          This global storage gets loaded by the routine
3181 !                   'Get_nsd' and in it is stored the next send ring
3182 !                   descriptor slot where the port/controller should
3183 !                   be polling on and the place to put this commands
3184 !                   command packet.
3185
3186
3187 ! Implicit Outputs :
3188 ! AZFMTR:          Azfmtr is the vector produced by DMCONV program and
3189 !                   is decalared in module AZKEL6.
3190 ! DMSA:            These three bound addresses point to specific area
3191 ! OVSA:            in the DM code buffer 'azfmtr' and are used to
3192 ! HDSA:            define the buffer descriptors within this command.
3193
3194 ! Completion Codes :
3195 ! RET_STATUS:      Return status passes back to the calling routine
3196 !                   the status of the just issued command.
3197
3198 ! Side Effects :
3199 ! The DM machine in the controller goes from the idle state to the
3200 ! active state on return of a sucessful return packet.
3201 !--
3202
3203 begin
3204
3205 local
3206     REF_NUM,          !Stores unique cmd ref number
3207     ESP_BUF$LOC,     !Stores out$standing cmd buffer location
3208     TEMP;            !A place to put the read IP register data
3209
3210 !
3211 ! Before we load up the command packet up with all this good information get
3212 ! the next send descriptor slot and a unique command reference number.
3213 !
3214 GET_NSD ();          !Get the next send desc slot
3215 REF_NUM = GET_CMD$REF (); !Get a unique command ref num
3216 !

```

```
3217 : UQ Port command envelope Header field definition
3218 :
3219 SND_ENVELOPE [.NSD_SLOT, MSG_LENGTH] = SZ_ESP;      !Load the message length
3220 SND_ENVELOPE [.NSD_SLOT, CREDITS] = ONE;           !Load the credits field
3221 SND_ENVELOPE [.NSD_SLOT, MSG_TYPE] = 0;           !Define the msg type 'Sequential'
3222 SND_ENVELOPE [.NSD_SLOT, CONN_ID] = DUP;          !Define the conection ID
3223 :
3224 : DUP generic command envelope field definition
3225 :
3226 SND_ENVELOPE [.NSD_SLOT, CMD_LREF] = .REF_NUM;     !Load command ref number
3227 SND_ENVELOPE [.NSD_SLOT, CMD_HREF] = ZERO;       !Zero Hi order word of cmd ref
3228 SND_ENVELOPE [.NSD_SLOT, UN_USED] = ZERO;        !Not used in DUP implimentation
3229 SND_ENVELOPE [.NSD_SLOT, UN_HUSED] = ZERO;       !Not used in DUP implimentation
3230 SND_ENVELOPE [.NSD_SLOT, OP_CODE] = OP_ESP;      !Load the command op-code
3231 SND_ENVELOPE [.NSD_SLOT, RSV0] = ZERO;           !Not used
3232 SND_ENVELOPE [.NSD_SLOT, MODIFIER] = ZERO;
3233 :
3234 : Command specfic command envelope field definition
3235 :
3236 : Byte count of initial transfer (from bytes 0-3 of the program header).
3237 :
3238 SND_ENVELOPE [.NSD_SLOT, BLO_CNT] = .AZFMTR [WRD0]; !Byte count low word
3239 SND_ENVELOPE [.NSD_SLOT, BHI_CNT] = .AZFMTR [WRD1]; !Byte count high word
3240 :
3241 : Buffer descriptor definition for initial load. First
3242 : byte of this buffer is byte 0 of program header.
3243 :
3244 SND_ENVELOPE [.NSD_SLOT, BPA_LO] = HDSA;         !Low unibus adrs <0-15>
3245 SND_ENVELOPE [.NSD_SLOT, BPA_HI] = ZERO;        !Unibus adrs bits <16-17>
3246 SND_ENVELOPE [.NSD_SLOT, QBUS_EXT] = ZERO;      !Q_bus extention adrs
3247 SND_ENVELOPE [.NSD_SLOT, RSV] = ZERO;           !Reserved field
3248 SND_ENVELOPE [.NSD_SLOT, UBA_CHAN] = ZERO;      !Unibus adaptor channel number
3249 SND_ENVELOPE [.NSD_SLOT, RSV0] = ZERO;          !These next four words are not
3250 SND_ENVELOPE [.NSD_SLOT, RSV1] = ZERO;          !used in the DUP implementation
3251 SND_ENVELOPE [.NSD_SLOT, RSV2] = ZERO;
3252 SND_ENVELOPE [.NSD_SLOT, RSV3] = ZERO;
3253 :
3254 : These next field definitions are the same as above except they are for the
3255 : overlay buffer descriptors. To make life easy for me I'll use the same names
3256 : and just prefix them with a $ for uniqueness.
3257 :
3258 : The overlay area immediately follows the initial load image in the program image.
3259 :
3260 SND_ENVELOPE [.NSD_SLOT, $BPA_LO] = .OVSA;       !Low unibus adrs <0-15>
3261 SND_ENVELOPE [.NSD_SLOT, $BPA_HI] = ZERO;       !Unibus adrs bits <16-17>
3262 SND_ENVELOPE [.NSD_SLOT, $QBUS_EXT] = ZERO;     !Q_bus extention adrs
3263 SND_ENVELOPE [.NSD_SLOT, $RSV] = ZERO;          !Reserved field
3264 SND_ENVELOPE [.NSD_SLOT, $UBA_CHAN] = ZERO;     !Unibus adaptor channel number
3265 SND_ENVELOPE [.NSD_SLOT, $RSV0] = ZERO;         !These next four words are not
3266 SND_ENVELOPE [.NSD_SLOT, $RSV1] = ZERO;         !used in the DUP implementation
3267 SND_ENVELOPE [.NSD_SLOT, $RSV2] = ZERO;
3268 SND_ENVELOPE [.NSD_SLOT, $RSV3] = ZERO;
3269 :
3270 : Call the load out$standing command buffer routine
3271 : and load this command into the buffer. The return
3272 : from this routine will point us to the buffer location
3273 : where this command is stored. Later we can look at
```

CZRCH5
REV A PATCH 00CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS11-Jul-1983 16:06:44
7-Jul-1983 08:24:58VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (15)

SEQ 214

Page 49

```

: 3274      ! this location to see if the interrupt service routine
: 3275      ! has received and process it.
: 3276
: 3277      ESP_BUF$LOC = LOAD_OUT$STD_BUF (.REF_NUM); !Load the command
: 3278
: 3279      if .ESP_BUF$LOC eqlu OBF_CODE then DECODE ();      !Error if buffer is full
: 3280
: 3281      !
: 3282      ! Set the ownership bit to 1 giving this slot to the port/controller
: 3283
: 3284      SEND_RING [.NSD_SLOT, OWN_BIT] = PORT_OWNED;
: 3285
: 3286      ! Read the IP register to stimulate port polling
: 3287
: 3288      TEMP = .RC25_ADDR [RCIP, RC_ALL];
: 3289
: 3290      ! Time out the port/controller processing the command.
: 3291
: 3292      ! The first test tests the connections ability to
: 3293      ! respond to this command without any errors in the SA
: 3294      ! register and for the command not timing out.
: 3295
: 3296      ! The second tests the DUP server for good status. If
: 3297      ! bad status is sent back then an error code is returned
: 3298      ! to the calling routine where the routine "decode" will
: 3299      ! decode and take the appropriate recovery. The time
: 3300      ! out routine will loop on delaying and checking the hi
: 3301      ! bit of the first word in the out$std_buf for a true.
: 3302      ! When true signals us that the interrupt service routine
: 3303      ! has received the endpacket and no connection errors
: 3304      ! were detected.
: 3305
: 3306
: 3307      if CTO_WAIT (3500, .REF_NUM, .ESP_BUF$LOC) then DECODE (); !Is return an error
: 3308
: 3309      !
: 3310      ! Get the return envelope address from the out$std_buf
: 3311      ! at this commands buffer location and check the packet
: 3312      ! for good status error and die if bad status was returned
: 3313
: 3314      RET_EN$AD = .OUT$STD_BUF [.ESP_BUF$LOC, ENV_ADR]; !Get the ret env adr
: 3315
: 3316      ! Now test for good status
: 3317
: 3318
: 3319      if .RET_EN$AD [STATUS] nequ ZERO      !Test the status
: 3320      then
: 3321          return RET_STATUS = RSE_CODE      !Return a 'Response status err' code
: 3322      else
: 3323          return .RET_STATUS;              !This ret_status is good or bad
: 3324
: 3325      end;

```

000000 004167 C00000G

```

      .SBTTL EX.SUP.PROG MODULE DECLARATIONS
EX.SUP.PROG::
      JSR    R1,$SAVE2

```

;

3160

CZRCH5
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555
SPIDERSUSERS:[NEALE.AZTEC]CZRCH5.B16;3 (15)

000004	005746			TST	-(SP)		
000006	004767	174516		JSR	PC,GET.NSD	:	3214
000012	004767	174660		JSR	PC,GET.CMD\$REF	:	3215
000016	010002			MOV	R0,R2	: *,REF.NUM	
000020	016746	000000G		MOV	NSD.SLOT,-(SP)	:	3219
000024	012746	000054		MOV	#54,-(SP)	:	
000030	004767	000000G		JSR	PC,BL\$MUL	:	
000034	012760	000050	000000G	MOV	#50,SND.ENVELOPE(R0)	:	
000042	012701	000002G		MOV	#SND.ENVELOPE+2,R1	:	3220
000046	060001			ADD	R0,R1	:	
000050	112711	000001		MOVB	#1,(R1)	:	3221
000054	112761	000002	000001	MOVB	#2,1(R1)	:	3222
000062	010260	000004G		MOV	R2,SND.ENVELOPE+4(R0)	: REF.NUM,*	3226
000066	005060	000006G		CLR	SND.ENVELOPE+6(R0)	:	3227
000072	005060	000010G		CLR	SND.ENVELOPE+10(R0)	:	3228
000076	005060	000012G		CLR	SND.ENVELOPE+12(R0)	:	3229
000102	112760	000002	000014G	MOVB	#2,SND.ENVELOPE+14(R0)	:	3230
000110	105060	000015G		CLRB	SND.ENVELOPE+15(R0)	:	3231
000114	005060	000016G		CLR	SND.ENVELOPE+16(R0)	:	3232
000120	016760	000000G	000020G	MOV	AZFMTR,SND.ENVELOPE+20(R0)	:	3238
000126	016760	000002G	000022G	MOV	AZFMTR+2,SND.ENVELOPE+22(R0)	:	3239
000134	012760	000000G	000024G	MOV	#HDSA,SND.ENVELOPE+24(R0)	:	3244
000142	012701	000026G		MOV	#SND.ENVELOPE+26,R1	:	3245
000146	060001			ADD	R0,R1	:	
000150	105011			CLRB	(R1)	:	3247
000152	105061	000001		CLRB	1(R1)	:	3248
000156	005060	000030G		CLR	SND.ENVELOPE+30(R0)	:	3249
000162	005060	000032G		CLR	SND.ENVELOPE+32(R0)	:	3250
000166	005060	000034G		CLR	SND.ENVELOPE+34(R0)	:	3251
000172	005060	000036G		CLR	SND.ENVELOPE+36(R0)	:	3252
000176	016760	000000G	000040G	MOV	OVSA,SND.ENVELOPE+40(R0)	:	3260
000204	012701	000042G		MOV	#SND.ENVELOPE+42,R1	:	3261
000210	060001			ADD	R0,R1	:	
000212	105011			CLRB	(R1)	:	3263
000214	105061	000001		CLRB	1(R1)	:	3264
000220	005060	000044G		CLR	SND.ENVELOPE+44(R0)	:	3265
000224	005060	000046G		CLR	SND.ENVELOPE+46(R0)	:	3266
000230	005060	000050G		CLR	SND.ENVELOPE+50(R0)	:	3267
000234	005060	000052G		CLR	SND.ENVELOPE+52(R0)	:	3268
000240	010216			MOV	R2,(SP)	: REF.NUM,*	3277
000242	004767	174336		JSR	PC,LOAD.OUT\$STD.BUF	:	
000246	010001			MOV	R0,R1	: *,ESP.BUF\$LOC	
000250	020127	002001		CMP	R1,#2001	: ESP.BUF\$LOC,*	3279
000254	001002			BNE	1\$:	
000256	004767	174502		JSR	PC,DECODE	:	
000262	016700	000000G	1\$:	MOV	NSD.SLOT,R0	:	3284
000266	006300			ASL	R0	:	
000270	006300			ASL	R0	:	
000272	066700	000000G		ADD	SEND.RING,R0	:	
000276	052760	100000	000002	BIS	#100000,2(R0)	:	
000304	017766	000000G	000004	MOV	@RC25.ADDR,4(SP)	: *,RC\$\$REG	3288
000312	016600	000004		MOV	4(SP),R0	: RC\$\$REG,TEMP	
000316	012716	006654		MOV	#6654,(SP)	:	3307
000322	010246			MOV	R2,-(SP)	: REF.NUM,*	
000324	010146			MOV	R1,-(SP)	: ESP.BUF\$LOC,*	
000326	004767	176442		JSR	PC,CTO.WAIT	:	
000332	022626			CMP	(SP)+,(SP)+	:	

CZRCH5 (ZRCH RC25 DISK FORMATTER
REV A PATCH (0) MODULE DECLARATIONS

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555
SPIDERSUSERS:[NEALE.AZTEC]CZRCH5.B16;3 (15)

000334	006000		ROR	R0		
000336	103002		BCC	2\$		
000340	004767	174420	JSR	PC,DECODE		
000344	010100		MOV	R1,R0	: ESP.BUFSLOC,*	3314
000346	006300		ASL	R0		
000350	006300		ASL	R0		
000352	016067	000002G 000000G	MOV	OUT\$STD.BUF+2(R0),RET.ENSAD		
000360	016000	000002G	MOV	OUT\$STD.BUF+2(R0),R0	:	3319
000364	005760	000016	TST	16(R0)		
000370	001405		BEQ	3\$		
000372	012700	000031	MOV	#31,R0	:	3321
000376	010067	000000G	MOV	R0,RET.STATUS	:	3203
000402	000402		BR	4\$:	
000404	016700	000000G	MOV	RET.STATUS,R0	:	3160
000410	062706	000006	ADD	#6,SP	:	
000414	000207		RTS	PC	:	

: Routine Size: 135 words, Routine Base: AC\$CODE + 3420
: Maximum stack depth per invocation: 9 words

: 3326

CZRCH5
REV A PATCH 00CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS11-Jul-1983 16:06:44
7-Jul-1983 08:24:58VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (16)SEQ 217
Page 52

```

3327 global routine EX_LOC_PROG =                !Executes local program
3328
3329 !++
3330 ! Functional Description :
3331 ! Receipt of this command causes the controller to search its local
3332 ! media for the named program, load and execute it. Receipt of a
3333 ! successful response by the host means that the program is executing
3334 ! and the server is in the active state. The time out value for this
3335 ! command is specified in the get dust response
3336
3337 ! Formal Parameters :
3338 ! none
3339
3340 ! Implicit Inputs :
3341 ! NSD_SLOT          This global storage gets loaded by the routine
3342 !                   'Get_nsd' and in it is stored the next send ring
3343 !                   descriptor slot where the port/controller should
3344 !                   be polling on and the place to put this commands
3345 !                   command packet.
3346
3347 ! Implicit Outputs :
3348 ! none
3349
3350 ! Completion Codes :
3351 ! RET_STATUS:      Return status passes back to the calling routine
3352 !                  the status of the just issued command.
3353
3354 ! Side Effects :
3355 ! The DM machine in the controller goes from the idle state to the
3356 ! active state on return of a successful return packet.
3357 !--
3358
3359 begin
3360
3361 local
3362     REF_NUM,                !Stores unique cmd ref number
3363     ELP_BUF$LOC,           !Stores outstanding cmd buffer location
3364     TEMP;                  !A place to store the read IP register data
3365
3366
3367 ! Before we load up the command packet up with all this good information get
3368 ! the next send descriptor slot and a unique command reference number.
3369
3370 GET_NSD ();                !Get the next send desc slot
3371 REF_NUM = GET_CMD$REF (); !Get a unique command ref num
3372
3373 ! UQ Port command envelope Header field definition
3374
3375 SND_ENVELOPE [.NSD_SLOT, MSG_LENGTH] = SZ_ELP;    !Load the message size
3376 SND_ENVELOPE [.NSD_SLOT, CREDITS] = ONE;         !Load the credit size
3377 SND_ENVELOPE [.NSD_SLOT, MSG_TYPE] = 0;         !Define the msg typ 'Sequential'
3378 SND_ENVELOPE [.NSD_SLOT, CONN_ID] = DUP;        !Define the connectio ID
3379
3380 ! DUP generic command envelope field definition
3381
3382 SND_ENVELOPE [.NSD_SLOT, CMD_LREF] = .REF_NUM;   !Load the command ref number
3383 SND_ENVELOPE [.NSD_SLOT, CMD_HREF] = ZERO;      !Zero the Hi order cmd ref num

```

CZRCH5
REV A PATCH 00CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555

SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (16)

SEQ 218

Page 53

```

3384 SND_ENVELOPE [.NSD_SLOT, UN_LUSED] = ZERO; !Not used in DUP implimentation
3385 SND_ENVELOPE [.NSD_SLOT, UN_HUSED] = ZERO; !Not used in DUP implimentation
3386 SND_ENVELOPE [.NSD_SLOT, OP_CODE] = OP_ELP; !Load this commands op-code
3387 SND_ENVELOPE [.NSD_SLOT, RSVD] = ZERO; !Not used field
3388 SND_ENVELOPE [.NSD_SLOT, MODIFIER] = ZERO; !Define modifiers for this cmd
3389
3390 : Command specific command envelope field definition
3391
3392 SND_ENVELOPE [.NSD_SLOT, PN_0] = ZERO; !Program name word 0
3393 SND_ENVELOPE [.NSD_SLOT, PN_1] = ZERO; !Program name word 1
3394 SND_ENVELOPE [.NSD_SLOT, PN_2] = ZERO; !Program name word 2
3395
3396 : Call the load out$standing command buffer routine
3397 : and load this command into the buffer. The return
3398 : from this routine will point us to the buffer location
3399 : where this command is stored. Later we can look at
3400 : this location to see if the interrupt service routine
3401 : has received and process it.
3402
3403 ELP_BUF$LOC = LOAD_OUT$STD_BUF (.REF_NUM); !Load the command
3404
3405 if .ELP_BUF$LOC eqlu OBF_CODE then DECODE (); !Error if buffer is full
3406
3407 :
3408 : Set the ownership bit to 1 giving this slot to the port/controller
3409
3410 SEND_RING [.NSD_SLOT, OWN_BIT] = PORT_OWNED;
3411
3412 : Read the IP register to stimulate port polling
3413
3414 TEMP = .RC25_ADDR [RCIP, RC_ALL];
3415
3416 : Time out the port/controller processing the command.
3417
3418 : The first test tests the connections ability to
3419 : respond to this command without any errors in the SA
3420 : register and for the command not timing out.
3421
3422 : The second tests the DUP server for good status. If
3423 : bad status is sent back then an error code is returned
3424 : to the calling routine where the routine "decode" will
3425 : decode and take the appropriate recovery. The time
3426 : out routine will loop on delaying and checking the hi
3427 : bit of the first word in the out$std_buf for a true.
3428 : When true signals us that the interrupt service routine
3429 : has received the endpacket and no connection errors
3430 : were detected.
3431
3432
3433 if CTO_WAIT (3000, .REF_NUM, .ELP_BUF$LOC) then DECODE (); !Is return an error
3434
3435 :
3436 : Get the return envelope address from the out$std_buf
3437 : at this commands buffer location and check the packet
3438 : for good status error and die if bad status was returned
3439
3440 RET_EN$AD = .OUT$STD_BUF [.ELP_BUF$LOC, ENV_ADR]; !Get the ret env adr

```

CZRCH5
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (16)

```

:      3441      |
:      3442      | : Now test for good status
:      3443      | :
:      3444      |
:      3445      | if .RET_EN$AD [STATUS] nequ ZERO      !Test the status
:      3446      | then
:      3447      |     return RET_STATUS = RSE_CODE      !Return a 'Response status err' code
:      3448      | else
:      3449      |     return .RET_STATUS;               !This ret_status is good or bad
:      3450      |
:      3451      | end;

```

000000	004167	000000G	EX.LOC.PROG::	.SBTTL	EX.LOC.PROG MODULE DECLARATIONS	
000004	005746			JSR	R1,\$SAVE2	3327
000006	004767	174100		TST	-(SP)	
000012	004767	174242		JSR	PC,GET.NSD	3370
000016	010002			JSR	PC,GET.CMD\$REF	3371
000020	016746	000000G		MOV	R0,R2	* ,REF.NUM
000024	012746	000054		MOV	NSD.SLOT,-(SP)	3375
000030	004767	000000G		MOV	#54,-(SP)	
000034	012760	000060	000000G	JSR	PC,BL\$MUL	
000042	012701	000002G		MOV	#60,SND.ENVELOPE(R0)	
000046	060001			MOV	#SND.ENVELOPE+2,R1	3376
000050	112711	000001		ADD	R0,R1	
000054	112761	000002	000001	MOVB	#1,(R1)	3377
000062	010260	000004G		MOVB	#2,1(R1)	3378
000066	005060	000006G		MOV	R2,SND.ENVELOPE+4(R0)	REF.NUM,*
000072	005060	000010G		CLR	SND.ENVELOPE+6(R0)	3382
000076	005060	000012G		CLR	SND.ENVELOPE+10(R0)	3383
000102	112760	000003	000014G	CLR	SND.ENVELOPE+12(R0)	3384
000110	105060	000015G		CLR	SND.ENVELOPE+14(R0)	3385
000114	005060	000016G		MOVB	#3,SND.ENVELOPE+15(R0)	3386
000120	005060	000020G		CLR	SND.ENVELOPE+16(R0)	3387
000124	005060	000022G		CLR	SND.ENVELOPE+20(R0)	3388
000130	005060	000024G		CLR	SND.ENVELOPE+22(R0)	3392
000134	010216			CLR	SND.ENVELOPE+24(R0)	3393
000136	004767	174024		MOV	R2,(SP)	REF.NUM,*
000142	010001			JSR	PC,LOAD.OUT\$STD.BUF	
000144	020127	002001		MOV	R0,R1	* ,ELP.BUF\$LOC
000150	001002			CMP	R1,#2001	ELP.BUF\$LOC,*
000152	004767	174170		BNE	1\$	
000156	016700	000000G	1\$:	JSR	PC,DECODE	
000162	006300			MOV	NSD.SLOT,R0	
000164	006300			ASL	R0	
000166	066700	000000G		ASL	R0	
000172	052760	100000	000002	ADD	SEND.RING,R0	
000200	017766	000000G	000004	BIS	#100000,2(R0)	
000206	016600	000004		MOV	@RC25.ADDR,4(SP)	* ,RC\$\$REG
000212	012716	005670		MOV	4(SP),R0	RC\$\$REG,TEMP
000216	010246			MOV	#5670,(SP)	
000220	010146			MOV	R2,-(SP)	REF.NUM,*
000222	004767	176130		MOV	R1,-(SP)	ELP.BUF\$LOC,*
000226	022626			JSR	PC,CTO.WAIT	
000230	006000			CMP	(SP)+,(SP)+	
				ROR	R0	

CZRCH5
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (16)

000232	103002		BCC	2\$		
000234	004767	174106	JSR	PC,DECODE		
000240	010100		2\$: MOV	R1,R0	:	ELP.BUF\$LOC,* 3440
000242	006300		ASL	R0		
000244	006300		ASL	R0		
000246	016067	000002G 000000G	MOV	OUT\$STD.BUF+2(R0),RET.ENSAD		
000254	016000	000002G	MOV	OUT\$STD.BUF+2(R0),R0	:	3445
000260	005760	000016	TST	16(R0)		
000264	001405		BEQ	3\$		
000266	012700	000031	MOV	#31,R0	:	3447
000272	010067	000000G	MOV	R0,RET.STATUS		
000276	000402		BR	4\$:	3359
000300	016700	000000G	3\$: MOV	RET.STATUS,R0		
000304	062706	000006	4\$: ADD	#6,SP	:	3327
000310	000207		RTS	PC		

: Routine Size: 101 words, Routine Base: AC\$CODE + 4036
: Maximum stack depth per invocation: 9 words

: 3452

```

:      3453 global routine SEND_DATA =                                !Performs host-->port communications
:      3454
:      3455 |**
:      3456 | Functional Description :
:      3457 | These commands are used to communicate between the initiating host
:      3458 | program and the remote program. Both send and receive commands
:      3459 | specify a host buffer descriptor and a byte count. In the case of
:      3460 | send data, the information in the buffer is read by the remote program
:      3461 | and a send data response sent back to the host to acknowledge receipt.
:      3462 | In the case of receive data the remote program writes data into the
:      3463 | buffer up to the amount specified by the byte count and then sends a
:      3464 | receive data response to the host to notify it of the transmission.
:      3465 |
:      3466 | The send data and receive data commands are only legal when the
:      3467 | server is in the active state. If the remote program terminates
:      3468 | abnormally, putting the server back in the idle state, outstanding
:      3469 | send data and receive data commands may be lost. In the event that
:      3470 | the specified timeout interval is exceeded, the host program should
:      3471 | issue a get dust status command to see if the remote program is
:      3472 | still running (is the dup server is active); if it is, the
:      3473 | progress indicator should be remembered and the timeout interval
:      3474 | should be re-installed. If the second timeout expires without a
:      3475 | response and a second get dust status shows the remote program
:      3476 | having made no progress in the interim then the program should be
:      3477 | considered broken and should be aborted.
:      3478 |
:      3479 | Formal Parameters :
:      3480 | none
:      3481 |
:      3482 | Implicit Inputs :
:      3483 | NSD_SLOT This global storage gets loaded by the routine
:      3484 | 'Get_nsd' and in it is stored the next send ring
:      3485 | descriptor slot where the port/controller should
:      3486 | be polling on and the place to put this commands
:      3487 | command packet.
:      3488 |
:      3489 | Implicit Outputs :
:      3490 | none
:      3491 |
:      3492 | Completion Codes :
:      3493 | RET_STATUS: Return status passes back to the calling routine
:      3494 | the status of the just issued command.
:      3495 |
:      3496 | Side Effects :
:      3497 | none
:      3498 | --
:      3499 |
:      3500 | begin
:      3501 |
:      3502 | local
:      3503 |     REF_NUM, !Stores unique cmd ref number
:      3504 |     SND_BUF$LOC, !Stores outstanding cmd buffer location
:      3505 |     TEMP; !A place to put read IP register data
:      3506 |
:      3507 |
:      3508 | : Before we load up the command packet up with all this good information
:      3509 | : get the next send descriptor slot and a unique command reference number.

```

CZRCH5
REV A PATCH 00CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS11-Jul-1983 16:06:44
7-Jul-1983 08:24:58VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (17)SEQ 222
Page 57

```

3510      |
3511      | GET_NSD ();                !Get the next send desc slot
3512      | REF_NUM = GET_CMD$REF (); !Get a unique command ref num
3513      |
3514      | : UQ Port command envelope Header field definition
3515      |
3516      | SND_ENVELOPE [.NSD_SLOT, MSG_LENGTH] = SZ_SED; !Load the message size
3517      | SND_ENVELOPE [.NSD_SLOT, CREDITS] = ONE; !Load the credit size
3518      | SND_ENVELOPE [.NSD_SLOT, MSG_TYPE] = 0; !Define the message typ 'Sequential
3519      | SND_ENVELOPE [.NSD_SLOT, CONN_ID] = DUP; !Define the connection ID
3520      |
3521      | : DUP generic command envelope field definition
3522      |
3523      | SND_ENVELOPE [.NSD_SLOT, CMD_LREF] = .REF_NUM; !Load command reference number
3524      | SND_ENVELOPE [.NSD_SLOT, CMD_HREF] = ZERO; !Zero Hi order cmd ref number
3525      | SND_ENVELOPE [.NSD_SLOT, UN_USED] = ZERO; !Not used in DUP implimentation
3526      | SND_ENVELOPE [.NSD_SLOT, UN_HUSED] = ZERO; !Not used in DUP implimentation
3527      | SND_ENVELOPE [.NSD_SLOT, OP$CODE] = OP_SED; !Load this commands op-code
3528      | SND_ENVELOPE [.NSD_SLOT, RSV$D] = ZERO; !Not used field
3529      | SND_ENVELOPE [.NSD_SLOT, MODIFIER] = ZERO; !Define the commands modifiers
3530      |
3531      | : Command specfic command envelope field definition
3532      |
3533      | : Byte count of transfer
3534      |
3535      | SND_ENVELOPE [.NSD_SLOT, BLO_CNT] = SNDB_SIZE; !Byte count low word
3536      | SND_ENVELOPE [.NSD_SLOT, BHI_CNT] = ZERO; !Byte count high word
3537      |
3538      | : Buffer descriptor definition
3539      |
3540      | SND_ENVELOPE [.NSD_SLOT, BPA_LO] = SND_BUF; !Buffer physical adrs <0-15>
3541      | SND_ENVELOPE [.NSD_SLOT, BPA_HI] = ZERO; !Buffer physical adrs bits <16-17>
3542      | SND_ENVELOPE [.NSD_SLOT, QBUS_EXT] = ZERO; !Q bus extention adrs
3543      | SND_ENVELOPE [.NSD_SLOT, RSV] = ZERO; !Reserved field
3544      | SND_ENVELOPE [.NSD_SLOT, UBA_CHAN] = ZERO; !Unibus adaptor channel number
3545      | SND_ENVELOPE [.NSD_SLOT, RSV0] = ZERO; !These next four words are not
3546      | SND_ENVELOPE [.NSD_SLOT, RSV1] = ZERO; !used in the UQ Port implementation
3547      | SND_ENVELOPE [.NSD_SLOT, RSV2] = ZERO;
3548      | SND_ENVELOPE [.NSD_SLOT, RSV3] = ZERO;
3549      |
3550      | : Call the load out$standing command buffer routine and load this command
3551      | : into the buffer. The return from this routine will point us to the buffer location
3552      | : where this command is stored. Later we can look at this location to
3553      | : see if the interrupt service routine has received and process it.
3554      |
3555      | SND_BUF$LOC = LOAD_OUT$STD_BUF (.REF_NUM); !Load the command
3556      |
3557      | if .SND_BUF$LOC eqlu OBF_CODE then DECODE (); !Error if buffer is full
3558      |
3559      | :
3560      | : Set the ownership bit to 1 giving this slot to the port/controller
3561      |
3562      | SEND_RING [.NSD_SLOT, OWN_BIT] = PORT_OWNED;
3563      |
3564      | : Read the IP register to stimulate port polling
3565      |
3566      | TEMP = .RC25_ADDR [RCIP, RC_ALL];

```

```

3567
3568
3569      + Time out the port/controller for the response from this command.
3570
3571      If the controller times out then:
3572      1. See what kind of error was returned. If the error
3573         is a type other than a CTO_CODE (controller time out)
3574         then call routine Decode which does the appropriate
3575         action based on the error.
3576
3577      2. If the returned error is an CTO_CODE then do a get
3578         dust status and check the progress indicator to look
3579         for an increase, indicating that the remote program is
3580         still running and is not dead.
3581
3582         If the indicator hasn't changed then assume that the
3583         remote program is dead and return an error code of
3584         RPD_CODE (remote program dead code) and exit.
3585
3586         If the indicator has changed then assume that the
3587         remote program is still running, save a copy of its
3588         value and reinstate the controller time out delay and
3589         repeat the loop.
3590
3591         As long as the progress indicator in the remote program
3592         is still increasing this loop will be repeated for ever.
3593
3594      If the controller doesn't time then return with the return
3595      code returned from routine CTO_WAIT () which could be either
3596      a success or error code by definition of this host code.
3597
3598
3599      while TRUE do                                !Repeat for ever
3600      begin
3601      BREAK;                                       !Flag control C's
3602
3603      : Do a controller time out and determine if the controller
3604      : has processed the command or if a fatal error has occured.
3605      :
3606
3607      if CTO_WAIT (4000, .REF_NUM, .SND_BUF$LOC)    !Is return an error
3608      then
3609      begin
3610      :
3611      : If the return status code eql's a CTO_CODE then see if the remote
3612      : program is still running. If it is then save the progress indicator
3613      : and repeat the loop else call routine Decode ().
3614      :
3615
3616      if .RET_STATUS eqlu CTO_CODE                !Is this a controller time out
3617      then
3618      begin
3619
3620      if GET_DUST_STATUS () then DECODE ();        !Get the dust status
3621
3622      if .RET_EN$AD [PLO_IND] gtru .PID_SAVE [0]  !Any progress been made
3623      then

```


CZRCH5
REV A PATCH 00CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS11-Jul-1983 16:06:44
7-Jul-1983 08:24:58VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (17)SEQ 224
Page 59

```

:      3624      begin
:      3625      PID_SAVE [0] = .RET_EN$AD [PLO_IND];      !Still running save Pid
:      3626      PID_SAVE [1] = .RET_EN$AD [PHI_IND];
:      3627      PRINTB (PID_FMT, .PID_SAVE [1], .PID_SAVE [0]);
:      3628      end
:      3629      else
:      3630
:      3631      if .RET_EN$AD [PHI_IND] gtru .PID_SAVE [1]
:      3632      then
:      3633      begin
:      3634      PID_SAVE [0] = .RET_EN$AD [PLO_IND];      !Still running save Pid
:      3635      PID_SAVE [1] = .RET_EN$AD [PHI_IND];
:      3636      PRINTB (PID_FMT, .PID_SAVE [1], .PID_SAVE [0]);
:      3637      end
:      3638      else
:      3639      return RET_STATUS = RPD_CODE;      !No progress so flag error
:      3640
:      3641      end
:      3642      else
:      3643      :
:      3644      : The return status code was not a controller time out code so something
:      3645      : else is wrong. Call the routine Decode () to find out what went wrong.
:      3646      :
:      3647      DECODE ()
:      3648
:      3649      end
:      3650      else
:      3651      begin
:      3652      :
:      3653      : The command has been received by the interrupt service.
:      3654      :
:      3655      : Get this commands return envelope address out of the
:      3656      : out$std buf and check for good return status error and
:      3657      : die if bad status.
:      3658      :
:      3659      RET_EN$AD = .OUT$STD_BUF [.SND_BUF$LOC, ENV_ADR];      !Get the ret env adr
:      3660      :
:      3661      : Test for good status
:      3662      :
:      3663      :
:      3664      if .RET_EN$AD [STATUS] nequ ZERO      !Test the status
:      3665      then
:      3666      return RET_STATUS = RSE_CODE      !Return a 'Response status err' code
:      3667      else
:      3668      return .RET_STATUS;      !This ret_status is good or bad
:      3669
:      3670      end;
:      3671
:      3672      end;
:      3673
:      3674      return .RET_STATUS;      !It won't compile without this here
:      3675      end;

```

000000 004167 C00000G

```

.SBTTL SEND.DATA MODULE DECLARATIONS
SEND.DATA::
JSR R1,$SAVE3 ;

```

3453

CZRCH5
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (17)

000004	005746		TST	-(SP)		
000006	004767	173566	JSR	PC.GET.NSD	:	3511
000012	004767	173730	JSR	PC.GET.CMD\$REF	:	3512
000016	010003		MOV	R0,R3	: *,REF.NUM	
000020	016746	000000G	MOV	NSD.SLOT,-(SP)	:	3516
000024	012746	000054	MOV	#54,-(SP)		
000030	004767	000000G	JSR	PC.BL\$MUL		
000034	012760	000034 000000G	MOV	#34,SND.ENVELOPE(R0)		
000042	012701	000002G	MOV	#SND.ENVELOPE+2,R1	:	3517
000046	060001		ADD	R0,R1		
000050	112711	000001	MOVB	#1,(R1)	:	3518
000054	112761	000002 000001	MOVB	#2,1(R1)	:	3519
000062	010360	000004G	MOV	R3,SND.ENVELOPE+4(R0)	: REF.NUM,*	3523
000066	005060	000006G	CLR	SND.ENVELOPE+6(R0)		3524
000072	005060	000010G	CLR	SND.ENVELOPE+10(R0)		3525
000076	005060	000012G	CLR	SND.ENVELOPE+12(R0)		3526
000102	112760	000004 000014G	MOVB	#4,SND.ENVELOPE+14(R0)		3527
000110	105060	000015G	CLRB	SND.ENVELOPE+15(R0)		3528
000114	005060	000016G	CLR	SND.ENVELOPE+16(R0)		3529
000120	012760	000045 000020G	MOV	#45,SND.ENVELOPE+20(R0)		3535
000126	005060	000022G	CLR	SND.ENVELOPE+22(R0)		3536
000132	012760	000000G 000024G	MOV	#SND.BUF,SND.ENVELOPE+24(R0)		3540
000140	012701	000026G	MOV	#SND.ENVELOPE+26,R1	:	3541
000144	060001		ADD	R0,R1		
000146	105011		CLRB	(R1)	:	3543
000150	105061	000001	CLRB	1(R1)	:	3544
000154	005060	000030G	CLR	SND.ENVELOPE+30(R0)		3545
000160	005060	000032G	CLR	SND.ENVELOPE+32(R0)		3546
000164	005060	000034G	CLR	SND.ENVELOPE+34(R0)		3547
000170	005060	000036G	CLR	SND.ENVELOPE+36(R0)		3548
000174	010316		MOV	R3,(SP)	: REF.NUM,*	3555
000176	004767	173452	JSR	PC.LOAD.OUT\$STD.BUF		
000202	010002		MOV	R0,R2	: *,SND.BUF\$LOC	
000204	020227	002001	CMP	R2,#2001	: SND.BUF\$LOC,*	3557
000210	001002		BNE	1\$		
000212	004767	173616	JSR	PC.DECODE		
000216	016700	000000G	MOV	NSD.SLOT,R0	:	3562
000222	006300		ASL	R0		
000224	006300		ASL	R0		
000226	066700	000000G	ADD	SEND.RING,R0		
000232	052760	100000 000002	BIS	#100000,2(R0)		
000240	017766	000000G 000004	MOV	@RC25.ADDR,4(SP)	: *,RC\$\$REG	3566
000246	016600	000004	MOV	4(SP),R0	: RC\$\$REG,TEMP	
000252	104422		TRAP	22	:	3600
000254	012716	007640	MOV	#7640,(SP)	:	3607
000260	010346		MOV	R3,-(SP)	: REF.NUM,*	
000262	010246		MOV	R2,-(SP)	: SND.BUF\$LOC,*	
000264	004767	175554	JSR	PC.CTO.WAIT		
000270	022626		CMP	(SP)+,(SP)+		
000272	006000		ROR	R0		
000274	103074		BCC	8\$		
000276	026727	000000G 000011	CMP	RET.STATUS,#11	:	3616
000304	001065		BNE	7\$		
000306	004767	176240	JSR	PC.GET.DUST.STATUS	:	3620
000312	006000		ROR	R0		
000314	103002		BCC	3\$		
000316	004767	173512	JSR	PC.DECODE		

CZRCH5
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (17)

000322	016701	000000G	3\$:	MOV	RET.ENSAD,R1	:	3626
000326	010100			MOV	R1,R0	: RET.ENSAD,*	3622
000330	016000	000024		MOV	24(R0),R0		
000334	020067	000000G		CMP	R0,PID.SAVE		
000340	101417			BLOS	4\$		
000342	010067	000000G		MOV	R0,PID.SAVE	:	3625
000346	016167	000026 000002G		MOV	26(R1),PID.SAVE+2	:	3626
000354	010016			MOV	R0,(SP)	: PID.SAVE,*	3627
000356	016746	000002G		MOV	PID.SAVE+2,-(SP)		
000362	012746	000000G		MOV	#PID.FMT,-(SP)		
000366	012746	000003		MOV	#3,-(SP)		
000372	010600			MOV	SP,R0	: SP,*	
000374	104414			TRAP	14		
000376	000422			BR	5\$:	3624
000400	026167	000026 000002G	4\$:	CMP	26(R1),PID.SAVE+2	:	3631
000406	101421			BLOS	6\$		
000410	010067	000000G		MOV	R0,PID.SAVE	:	3634
000414	016167	000026 000002G		MOV	26(R1),PID.SAVE+2	:	3635
000422	010016			MOV	R0,(SP)	: PID.SAVE,*	3636
000424	016746	000002G		MOV	PID.SAVE+2,-(SP)		
000430	012746	000000G		MOV	#PID.FMT,-(SP)		
000434	012746	000003		MOV	#3,-(SP)		
000440	010600			MOV	SP,R0	: SP,*	
000442	104414			TRAP	14		
000444	062706	000006	5\$:	ADD	#6,SP	:	3633
000450	000700			BR	2\$:	3631
000452	012700	000051	6\$:	MOV	#51,R0	:	3639
000456	000420			BR	9\$		
000460	004767	173350	7\$:	JSR	PC,DECODE	:	3647
000464	000672			BR	2\$:	3607
000466	010200		8\$:	MOV	R2,R0	: SND.BUF\$LOC,*	3659
000470	006300			ASL	R0		
000472	006300			ASL	R0		
000474	016067	000002G 000000G		MOV	OUT\$STD.BUF+2(R0),RET.ENSAD		
000502	016000	000002G		MOV	OUT\$STD.BUF+2(R0),R0	:	3664
000506	005760	000016		TST	16(R0)		
000512	001405			BEQ	10\$		
000514	012700	000031		MOV	#31,R0	:	3666
000520	010067	000000G	9\$:	MOV	R0,RET.STATUS	:	3651
000524	000402			BR	11\$:	3651
000526	016700	000000G	10\$:	MOV	RET.STATUS,R0		
000532	062706	000006	11\$:	ADD	#6,SP	:	3453
000536	000207			RTS	PC		

: Routine Size: 176 words, Routine Base: AC\$CODE + 4350
: Maximum stack depth per invocation: 12 words

: 3676

```
3677 global routine REC_DATA = !Performs host-->port communications
3678
3679 !++
3680 Functional Description :
3681 These commands are used to communicate between the initiating host
3682 program and the remote program. Both send and receive commands
3683 specify a host buffer descriptor and a byte count. In the case of
3684 send data, the information in the buffer is read by the remote program
3685 and a send data response sent back to the host to acknowledge receipt.
3686 In the case of receive data the remote program writes data into the
3687 buffer up to the amount specified by the byte count and then sends a
3688 receive data response to the host to notify it of the transmission.
3689
3690 The send data and receive data commands are only legal when the
3691 server is in the active state. If the remote program terminates
3692 abnormally, putting the server back in the idle stae, outstanding
3693 send data and receive data commands may be lost. In the event that
3694 the specified timeout interval is exceeded, the host program should
3695 issue a get dust staus command to see if the remote program is
3696 still running (is. the dup server is active); if it is, the
3697 progress indicator should be remembered and the timeout interval
3698 should be re-installed. If the second timeout expires without a
3699 response and a second get dust status shows the remote program
3700 having made no progress in the interim then the program should be
3701 considered broken and should be aborted.
3702
3703 Formal Parameters :
3704 none
3705
3706 Implicit Inputs :
3707 NSD_SLOT This global storage gets loaded by the routine
3708 'Get_nsd' and in it is stored the next send ring
3709 descriptor slot where the port/controller should
3710 be polling on and the place to put this commands
3711 command packet.
3712
3713 Implicit Outputs :
3714 none
3715
3716 Completion Codes :
3717 RET_STATUS: Return status passes back to the calling routine
3718 the status of the just issued command.
3719
3720 Side Effects :
3721 none
3722 --
3723
3724 begin
3725
3726 local
3727 REF_NUM, !Stores unique cmd ref number
3728 REC_BUF$LOC, !Stores outstanding cmd buffer location
3729 TEMP; !A place to put read IP register data
3730
3731 !
3732 ! Before we load up the command packet up with all this good information
3733 ! get the next send descriptor slot and a unique command reference number.
```

CZRCH5
REV A PATCH 00CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS11-Jul-1983 16:06:44
7-Jul-1983 08:24:58VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (18)SEQ 228
Page 63

```

3734 :
3735 GET_NSD (); !Get the next send desc slot
3736 REF_NUM = GET_CMD$REF (); !Get a unique command ref num
3737 :
3738 : UQ Port command envelope Header field definition
3739 :
3740 SND_ENVELOPE [.NSD_SLOT, MSG_LENGTH] = SZ_RED; !Load message length
3741 SND_ENVELOPE [.NSD_SLOT, CREDITS] = ONE; !Load credit size
3742 SND_ENVELOPE [.NSD_SLOT, MSG_TYPE] = 0; !Define message type 'Sequential'
3743 SND_ENVELOPE [.NSD_SLOT, CONN_ID] = DUP; !Define connection ID
3744 :
3745 : DUP generic command envelope field definition
3746 :
3747 SND_ENVELOPE [.NSD_SLOT, CMD_LREF] = .REF_NUM; !Load command reference number
3748 SND_ENVELOPE [.NSD_SLOT, CMD_HREF] = ZERO; !Zero Hi order cmd ref num
3749 SND_ENVELOPE [.NSD_SLOT, UN_USED] = ZERO; !Not used in DUP implimentation
3750 SND_ENVELOPE [.NSD_SLOT, UN_HUSED] = ZERO; !Not used in DUP implimentation
3751 SND_ENVELOPE [.NSD_SLOT, OP_CODE] = OP_RED; !Load this commands op-code
3752 SND_ENVELOPE [.NSD_SLOT, RSV0] = ZERO; !Not used field
3753 SND_ENVELOPE [.NSD_SLOT, MODIFIER] = ZERO; !Define this commands modifiers
3754 :
3755 : Command specfic command envelope field definition
3756 :
3757 : Byte count of transfer
3758 :
3759 SND_ENVELOPE [.NSD_SLOT, BLO_CNT] = RECB_SIZE; !Byte count low word
3760 SND_ENVELOPE [.NSD_SLOT, BHI_CNT] = ZERO; !Byte count high word
3761 :
3762 : Buffer descriptor definition
3763 :
3764 SND_ENVELOPE [.NSD_SLOT, BPA_LO] = REC_BUF; !Low unibus adrs <0-15>
3765 SND_ENVELOPE [.NSD_SLOT, BPA_HI] = ZERO; !Unibus adrs bits <16-17>
3766 SND_ENVELOPE [.NSD_SLOT, QBUS_EXT] = ZERO; !Q_bus extention adrs
3767 SND_ENVELOPE [.NSD_SLOT, RSV] = ZERO; !Reserved field
3768 SND_ENVELOPE [.NSD_SLOT, UBA_CHAN] = ZERO; !Unibus adaptor channel number
3769 SND_ENVELOPE [.NSD_SLOT, RSV0] = ZERO; !These next four words are not
3770 SND_ENVELOPE [.NSD_SLOT, RSV1] = ZERO; !used in the UQ Port implementation
3771 SND_ENVELOPE [.NSD_SLOT, RSV2] = ZERO;
3772 SND_ENVELOPE [.NSD_SLOT, RSV3] = ZERO;
3773 :
3774 : Call the load out$standing command buffer routine and load this command
3775 : into the buffer. The return from this routine will point us to the
3776 : buffer location where this command is stored. Later we can look at this
3777 : location to see if the interrupt service routine has received and process
3778 : it.
3779 :
3780 REC_BUF$LOC = LOAD_OUT$STD_BUF (.REF_NUM); !Load the command
3781 :
3782 if .REC_BUF$LOC eqlu OBF_CODE then DECODE (); !Error if buffer is full
3783 :
3784 :
3785 : Set the ownership bit to 1 giving this slot to the port/controller.
3786 :
3787 SEND_RING [.NSD_SLOT, OWN_BIT] = PORT_OWNED;
3788 :
3789 : Read the IP register to stimulate port polling
3790 :

```

```
3791 TEMP = .RC25_ADDR [RCIP, RC_ALL];
3792
3793 !+
3794 Time out the port/controller for the response from this command.
3795
3796 If the controller times out then:
3797 1. See what kind of error was returned. If the error
3798 is a type other than a CTO_CODE (controller time out)
3799 then call routine Decode which does the appropriate
3800 action based on the error.
3801
3802 2. If the returned error is an CTO_CODE then do a get
3803 dust status and check the progress indicator to look
3804 for an increase, indicating that the remote program is
3805 still running and is not dead.
3806
3807 If the indicator hasn't changed then assume that the
3808 remote program is dead and return an error code of
3809 RPD_CODE (remote program dead code) and exit.
3810
3811 If the indicator has changed then assume that the
3812 remote program is still running, save a copy of its
3813 value and reinstate the controller time out delay and
3814 repeat the loop.
3815
3816 As long as the progress indicator in the remote program
3817 is still increasing this loop will be repeated for ever.
3818
3819 If the controller doesn't time then return with the return code
3820 returned from routine CTO_WAIT () which could be either a success
3821 or error code by definition of this host code.
3822
3823
3824 while TRUE do !Repeat for ever
3825 begin
3826 BREAK; !Flag control C's
3827
3828 Do a controller time out and determine if the controller
3829 has processed the command or if a fatal error has occurred.
3830
3831
3832 if CTO_WAIT (4000, .REF_NUM, .REC_BUF$LOC) !Is return an error
3833 then
3834 begin
3835
3836 If the return status code eql's a CTO_CODE then see if the remote
3837 program is still running. If it is then save the progress indicator
3838 and repeat the loop else call routine Decode ().
3839
3840
3841 if .RET_STATUS eqlu CTO_CODE !Is this a controller time out
3842 then
3843 begin
3844
3845 if GET_DUST_STATUS () then DECODE (); !Get the dust status
3846
3847 if .RET_EN$AD [PLO_IND] gtru .PID_SAVE [0] !Any progress been made
```

```

3848     then
3849     begin
3850     PID_SAVE [0] = .RET_EN$AD [PLO_IND];      !Still running save Pid
3851     PID_SAVE [1] = .RET_EN$AD [PHI_IND];
3852     PRINTB (PID_FMT, .PID_SAVE [1], .PID_SAVE [0]);
3853     end
3854     else
3855
3856     if .RET_EN$AD [PHI_IND] gtru .PID_SAVE [1]
3857     then
3858     begin
3859     PID_SAVE [0] = .RET_EN$AD [PLO_IND];      !Still running save Pid
3860     PID_SAVE [1] = .RET_EN$AD [PHI_IND];
3861     PRINTB (PID_FMT, .PID_SAVE [1], .PID_SAVE [0]);
3862     end
3863     else
3864     return RET_STATUS = RPD_CODE;      !No progress so flag error
3865
3866     end
3867     else
3868     :
3869     : The return status code was not a controller time out code so something
3870     : else is wrong. Call the routine Decode () to find out what went wrong.
3871     :
3872     : DECODE ()
3873     :
3874     end
3875     else
3876     begin
3877     :
3878     : The command has been received by the interrupt service.
3879     :
3880     : Get this commands return envelope address out of the
3881     : out$std_buf and check for good return status error and
3882     : die if bad status.
3883     :
3884     RET_EN$AD = .OUT$STD_BUF [.REC_BUF$LOC, ENV_ADR];      !Get the ret env adr
3885     :
3886     : Test for good status
3887     :
3888
3889     if .RET_EN$AD [STATUS] nequ ZERO      !Test the status
3890     then
3891     return RET_STATUS = RSE_CODE      !Return a 'Response status err' code
3892     else
3893     return .RET_STATUS;      !This ret_status is good or bad
3894
3895     end;
3896
3897     end;
3898
3899     return .RET_STATUS;      !It won't compile without this here
3900     end;

```

000004	005746		JSR	R1,\$SAVE3	:	3677
000006	004767	173026	TST	-(SP)	:	
000012	004767	173170	JSR	PC,GET.NSD	:	3735
000016	010003		JSR	PC,GET.CMD\$REF	:	3736
000020	016746	000000G	MOV	R0,R3	:*,REF.NUM	
000024	012746	000054	MOV	NSD.SLOT,-(SP)	:	3740
000030	004767	000000G	JSR	PC,BL\$MUL	:	
000034	012760	000034 000000G	MOV	#34,SND.ENVELOPE(R0)	:	
000042	012701	000002G	MOV	#SND.ENVELOPE+2,R1	:	3741
000046	060001		ADD	R0,R1	:	
000050	112711	000001	MOVB	#1,(R1)	:	3742
000054	112761	000002 000001	MOVB	#2,1(R1)	:	3743
000062	010360	000004G	MOV	R3,SND.ENVELOPE+4(R0)	:REF.NUM,*	3747
000066	005060	000006G	CLR	SND.ENVELOPE+6(R0)	:	3748
000072	005060	000010G	CLR	SND.ENVELOPE+10(R0)	:	3749
000076	005060	000012G	CLR	SND.ENVELOPE+12(R0)	:	3750
000102	112760	000005 000014G	MOVB	#5,SND.ENVELOPE+14(R0)	:	3751
000110	105060	000015G	CLRB	SND.ENVELOPE+15(R0)	:	3752
000114	005060	000016G	CLR	SND.ENVELOPE+16(R0)	:	3753
000120	012760	000170 000020G	MOV	#170,SND.ENVELOPE+20(R0)	:	3759
000126	005060	000022G	CLR	SND.ENVELOPE+22(R0)	:	3760
000132	012760	000000G 000024G	MOV	#REC.BUF,SND.ENVELOPE+24(R0)	:	3764
000140	012701	000026G	MOV	#SND.ENVELOPE+26,R1	:	3765
000144	060001		ADD	R0,R1	:	
000146	105011		CLRB	(R1)	:	3767
000150	105061	000001	CLRB	1(R1)	:	3768
000154	005060	000030G	LLR	SND.ENVELOPE+30(R0)	:	3769
000160	005060	000032G	CLR	SND.ENVELOPE+32(R0)	:	3770
000164	005060	000034G	CLR	SND.ENVELOPE+34(R0)	:	3771
000170	005060	000036G	CLR	SND.ENVELOPE+36(R0)	:	3772
000174	010316		MOV	R3,(SP)	:REF.NUM,*	3780
000176	004767	172712	JSR	PC,LOAD.OUT\$STD.BUF	:	
000202	010002		MOV	R0,R2	:*,REC.BUF\$LOC	
000204	020227	002001	CMP	R2,#2001	:REC.BUF\$LOC,*	3782
000210	001002		BNE	1\$:	
000212	004767	173056	JSR	PC,DECODE	:	
000216	016700	000000G	MOV	NSD.SLOT,R0	:	3787
000222	006300		ASL	R0	:	
000224	006300		ASL	R0	:	
000226	066700	000000G	ADD	SEND.RING,R0	:	
000232	052760	100000 000002	BIS	#100000,2(R0)	:	
000240	017766	000000G 000004	MOV	@RC25.ADDR,4(SP)	:*,RC\$\$REG	3791
000246	016600	000004	MOV	4(SP),R0	:RC\$\$REG,TEMP	
000252	104422		TRAP	22	:	3825
000254	012716	007640	MOV	#7640,(SP)	:	3832
000260	010346		MOV	R3,-(SP)	:REF.NUM,*	
000262	010246		MOV	R2,-(SP)	:REC.BUF\$LOC,*	
000264	004767	175014	JSR	PC,CTO.WAIT	:	
000270	022626		CMP	(SP)+,(SP)+	:	
000272	006000		ROR	R0	:	
000274	103074		BCC	8\$:	
000276	026727	000000G 000011	CMP	RET.STATUS,#11	:	3841
000304	001065		BNE	7\$:	
000306	004767	175500	JSR	PC,GET.DUST.STATUS	:	3845
000312	006000		ROR	R0	:	
000314	103002		BCC	3\$:	

000316	004767	172752			JSR	PC,DECODE			
000322	016701	000000G	3\$:		MOV	RET.ENSAD,R1	:		3851
000326	010100				MOV	R1,R0	:	RET.ENSAD,*	3847
000330	016000	000024			MOV	24(R0),R0			
000334	020067	000000G			CMP	R0,PID.SAVE			
000340	101417				BLOS	4\$			
000342	010067	000000G			MOV	R0,PID.SAVE	:		3850
000346	016167	000026	000002G		MOV	26(R1),PID.SAVE+2	:		3851
000354	010016				MOV	R0,(SP)	:	PID.SAVE,*	3852
000356	016746	000002G			MOV	PID.SAVE+2,-(SP)			
000362	012746	000000G			MOV	#PID.FMT,-(SP)			
000366	012746	000003			MOV	#3,-(SP)			
000372	010600				MOV	SP,R0	:	SP,*	
000374	104414				TRAP	14			
000376	000422				BR	5\$:		3849
000400	026167	000026	000002G	4\$:	CMP	26(R1),PID.SAVE+2	:		3856
000406	101421				BLOS	6\$			
000410	010067	000000G			MOV	R0,PID.SAVE	:		3859
000414	016167	000026	000002G		MOV	26(R1),PID.SAVE+2	:		3860
000422	010016				MOV	R0,(SP)	:	PID.SAVE,*	3861
000424	016746	000002G			MOV	PID.SAVE+2,-(SP)			
000430	012746	000000G			MOV	#PID.FMT,-(SP)			
000434	012746	000003			MOV	#3,-(SP)			
000440	010600				MOV	SP,R0	:	SP,*	
000442	104414				TRAP	14			
000444	062706	000006		5\$:	ADD	#6,SP	:		3858
000450	000700				BR	2\$:		3856
000452	012700	000051		6\$:	MOV	#51,R0	:		3864
000456	000420				BR	9\$:		
000460	004767	172610		7\$:	JSR	PC,DECODE	:		3872
000464	000672				BR	2\$:		3832
000466	010200			8\$:	MOV	R2,R0	:	REC.BUF\$LOC,*	3884
000470	006300				ASL	R0			
000472	006300				ASL	R0			
000474	016067	000002G	000000G		MOV	OUT\$STD.BUF+2(R0),RET.ENSAD			
000502	016000	000002G			MOV	OUT\$STD.BUF+2(R0),R0	:		3889
000506	005760	000016			TST	16(R0)			
000512	001405				BEQ	10\$			
000514	012700	000031			MOV	#31,R0	:		3891
000520	010067	000000G		9\$:	MOV	R0,RET.STATUS			
000524	000402				BR	11\$:		3876
000526	016700	000000G		10\$:	MOV	RET.STATUS,R0			
000532	062706	000006		11\$:	ADD	#6,SP	:		3677
000536	000207				RTS	PC			

; Routine Size: 176 words, Routine Base: AC\$CODE + 5110
; Maximum stack depth per invocation: 12 words

; 3901

```
3902 global routine SET_CNTRLR_CHAR = !Sets control characteristics
3903
3904 !++
3905 Functional Description :
3906 The SET CONTROLLER CHARACTERISTICS command is used to set host
3907 settable unit characteristics and obtain those unit
3908 characteristics that are essential for proper class driver
3909 operation. This command never alters the unit's state
3910 ('unit-online', 'unit-available', 'unit-offline'). It is
3911 meaningless to set host settable characteristics for a unit
3912 that is 'unit-available' or 'unit-offline'.
3913
3914 Formal Parameters :
3915 none
3916
3917 Implicit Inputs :
3918 NSD_SLOT This global storage gets loaded by the routine
3919 'Get_nsd' and in it is stored the next send ring
3920 descriptor slot where the port/controller should
3921 be polling on and the place to put this commands
3922 command packet.
3923
3924
3925 Implicit Outputs :
3926 none
3927
3928 Completion Codes :
3929 RET_STATUS: Return status passes back to the calling routine
3930 the status of the just issued command.
3931
3932 Side Effects :
3933 Any previously defined controller characteristics will possibly
3934 be altered after execution of this command.
3935 --
3936
3937 begin
3938
3939 local
3940 REF_NUM, !Stores unique cmd ref number
3941 SCC_BUF$LOC, !Stores outstanding cmd buffer location
3942 TEMP; !A place to put read IP register data
3943
3944
3945 ! Before we load up the command packet up with all this good information
3946 ! get the next send descriptor slot and a unique command reference number.
3947
3948 GET_NSD (); !Get the next send desc slot
3949 REF_NUM = GET_CMD$REF (); !Get a unique command ref num
3950
3951 ! UQ Port command envelope Header field definition
3952
3953 SND_ENVELOPE [.NSD_SLOT, MSG_LENGTH] = SZ_SCC; !Load message length
3954 SND_ENVELOPE [.NSD_SLOT, CREDITS] = ONE; !Load credit size
3955 SND_ENVELOPE [.NSD_SLOT, MSG_TYPE] = 0; !Define message type 'Sequential'
3956 SND_ENVELOPE [.NSD_SLOT, CONN_ID] = MSCP; !Define connection ID
3957
3958 ! MSCP generic command envelope field definition
```

CZRCH5
REV A PATCH 00CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS11-Jul-1983 16:06:44
7-Jul-1983 08:24:58VAX-11 Bliss-16 V3-555
SPIDER\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (19)

SEQ 234

Page 69

```

3959 :
3960 SND_ENVELOPE [.NSD_SLOT, CMD_LREF] = .REF_NUM;      !Load command reference number
3961 SND_ENVELOPE [.NSD_SLOT, CMD_HREF] = ZERO;        !Zero Hi order cmd ref num
3962 SND_ENVELOPE [.NSD_SLOT, UN_USED] = ZERO;         !Not used in DUP implimentation
3963 SND_ENVELOPE [.NSD_SLOT, UN_HUSED] = ZERO;        !Not used in DUP implimentation
3964 SND_ENVELOPE [.NSD_SLOT, OP_CODE] = OP_SCC;       !Load this commands op-code
3965 SND_ENVELOPE [.NSD_SLOT, RSVD] = ZERO;           !Not used field
3966 SND_ENVELOPE [.NSD_SLOT, MODIFIER] = ZERO;        !Define this commands modifiers
3967 :
3968 : Command specific command envelope field definition
3969 :
3970 SND_ENVELOPE [.NSD_SLOT, MSCP_VER] = ZERO;        !MSCP version
3971 SND_ENVELOPE [.NSD_SLOT, CTL_FLAGS] = ZERO;       !Controller flags
3972 SND_ENVELOPE [.NSD_SLOT, HOST_TOV] = ZERO;        !Host time out value
3973 SND_ENVELOPE [.NSD_SLOT, RSSVD] = ZERO;           !Reserved
3974 SND_ENVELOPE [.NSD_SLOT, TSD_0] = ZERO;           !Time and Date word 0
3975 SND_ENVELOPE [.NSD_SLOT, TSD_1] = ZERO;           !Time and Date word 1
3976 SND_ENVELOPE [.NSD_SLOT, TSD_2] = ZERO;           !Time and Date word 2
3977 SND_ENVELOPE [.NSD_SLOT, TSD_3] = ZERO;           !Time and Date word 3
3978 SND_ENVELOPE [.NSD_SLOT, CDP_LO] = ZERO;          !Cntlr dep parameter lo word
3979 SND_ENVELOPE [.NSD_SLOT, CDP_HI] = ZERO;          !Cntlr dep parameter hi wrd
3980 :
3981 : Call the load out$standing command buffer routine and load this command
3982 : into the buffer. The return from this routine will point us to the
3983 : buffer location where this command is stored. Later we can look at this
3984 : location to see if the interrupt service routine has received and process it.
3985 :
3986 SCC_BUF$LOC = LOAD_OUT$STD_BUF (.REF_NUM); !Load the command
3987 :
3988 if .SCC_BUF$LOC eqlu OBF_CODE then DECODE ();      !Error if buffer is full
3989 :
3990 :
3991 : Set the ownership bit to 1 giving this slot to the port/controller
3992 :
3993 SEND_RING [.NSD_SLOT, OWN_BIT] = PORT_OWNED;
3994 :
3995 : Read the IP register to stimulate port polling
3996 :
3997 TEMP = .RC25_ADDR [RCIP, RC_ALL];
3998 :
3999 : Time out the port/controller processing the command.
4000 :
4001 : The first test tests the connections ability to
4002 : respond to this command without any errors in the SA
4003 : register and for the command not timing out.
4004 :
4005 : The second tests the DUP server for good status. If
4006 : bad status is sent back then an error code is returned
4007 : to the calling routine where the routine "decode" will
4008 : decode and take the appropriate recovery. The time
4009 : out routine will loop on delaying and checking the hi
4010 : bit of the first word in the out$std_buf for a true.
4011 : When true signals us that the interrupt service routine
4012 : has received the endpacket and no connection errors
4013 : were detected.
4014 :
4015 :

```

CZRCH5
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (19)

```

:      4016      if CTO_WAIT (3000, .REF_NUM, .SCC_BUF$LOC) then DECODE (); !Is return an error
:      4017
:      4018      |
:      4019      | Get the return envelope address from the out$std_buf
:      4020      | at this commands buffer location and check the packet
:      4021      | for good status error and die if bad status was returned
:      4022      |
:      4023      RET_EN$AD = .OUT$STD_BUF [.SCC_BUF$LOC, ENV_ADR]; !Get the ret env adr
:      4024      |
:      4025      | Now test for good status
:      4026      |
:      4027
:      4028      if .RET_EN$AD [STATUS] nequ ZERO          !Test the status
:      4029      then
:      4030          return RET_STATUS = RSE_CODE        !Return a 'Response status err' code
:      4031      else
:      4032          return .RET_STATUS;                  !This ret_status is good or bad
:      4033
:      4034      end;

```

Address	Offset	Label	Instruction	Comment	Line No.
000000	004167	000000G	.SBTTL SET.CNTRLR.CHAR MODULE DECLARATIONS		
			SET.CNTRLR.CHAR::		
			JSR R1,\$SAVE2	:	3902
			TST -(SP)	:	
000004	005746		JSR PC,GET.NSD	:	3948
000006	004767	172266	JSR PC,GET.CMD\$REF	:	3949
000012	004767	172430	MOV R0,R2	: *.REF.NUM	
000016	010002		MOV NSD.SLOT,-(SP)	:	3953
000020	016746	000000G	MOV #54,-(SP)	:	
000024	012746	000054	JSR PC,BL\$MUL	:	
000030	004767	000000G	MOV #40,SND.ENVELOPE(R0)	:	
000034	012760	000040 000000G	MOV #SND.ENVELOPE+2,R1	:	3954
000042	012701	000002G	ADD R0,R1	:	
000046	060001		MOV#B #1,(R1)	:	3955
000050	112711	000001	CLRB 1(R1)	: REF.NUM,*	3956
000054	105061	000001	MOV R2,SND.ENVELOPE+4(R0)	:	3960
000060	010260	000004G	CLR SND.ENVELOPE+6(R0)	:	3961
000064	005060	000006G	CLR SND.ENVELOPE+10(R0)	:	3962
000070	005060	000010G	CLR SND.ENVELOPE+12(R0)	:	3963
000074	005060	000012G	MOV#B #4,SND.ENVELOPE+14(R0)	:	3964
000100	112760	000004 000014G	CLRB SND.ENVELOPE+15(R0)	:	3965
000106	105060	000015G	CLR SND.ENVELOPE+16(R0)	:	3966
000112	005060	000016G	CLR SND.ENVELOPE+20(R0)	:	3970
000116	005060	000020G	CLR SND.ENVELOPE+22(R0)	:	3971
000122	005060	000022G	CLR SND.ENVELOPE+24(R0)	:	3972
000126	005060	000024G	CLR SND.ENVELOPE+26(R0)	:	3973
000132	005060	000026G	CLR SND.ENVELOPE+30(R0)	:	3974
000136	005060	000030G	CLR SND.ENVELOPE+32(R0)	:	3975
000142	005060	000032G	CLR SND.ENVELOPE+34(R0)	:	3976
000146	005060	000034G	CLR SND.ENVELOPE+36(R0)	:	3977
000152	005060	000036G	CLR SND.ENVELOPE+40(R0)	:	3978
000156	005060	000040G	CLR SND.ENVELOPE+42(R0)	:	3979
000162	005060	000042G	MOV R2,(SP)	: REF.NUM,*	3986
000166	010216		JSR PC,LOAD.OUT\$STD.BUF	:	
000170	004767	172160	MOV R0,R1	: *.SCC.BUF\$LOC	
000174	010001		CMP R1,#2001	: SCC.BUF\$LOC,*	3988
000176	020127	002001			

CZRCH5
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555
SPIDER\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (19)

000202	001002		BNE	1\$		
000204	004767	172324	JSR	PC,DECODE		
000210	016700	000000G	1\$:	MOV	NSD.SLOT,R0	3993
000214	006300			ASL	R0	
000216	006300			ASL	R0	
000220	066700	000000G		ADD	SEND.RING,R0	
000224	052760	100000 000002		BIS	#100000,2(R0)	
000232	017766	000000G 000004		MOV	@RC25.ADDR,4(SP)	: *,RC\$\$REG 3997
000240	016600	000004		MOV	4(SP),R0	: RC\$\$REG,TEMP
000244	012716	005670		MOV	#5670,(SP)	: 4016
000250	010246			MOV	R2,-(SP)	: REF.NUM,*
000252	010146			MOV	R1,-(SP)	: SCC.BUF\$LOC,*
000254	004767	174264		JSR	PC,CTO.WAIT	
000260	022626			CMP	(SP)+,(SP)+	
000262	006000			ROR	R0	
000264	103002			BCC	2\$	
000266	004767	172242		JSR	PC,DECODE	
000272	010100		2\$:	MOV	R1,R0	: SCC.BUF\$LOC,* 4023
000274	006300			ASL	R0	
000276	006300			ASL	R0	
000300	016067	000002G 000000G		MOV	OUT\$STD.BUF+2(R0),RET.EN\$AD	
000306	016000	000002G		MOV	OUT\$STD.BUF+2(R0),R0	: 4028
000312	005760	000016		TST	16(R0)	
000316	001405			BEQ	3\$	
000320	012700	000031		MOV	#31,R0	: 4030
000324	010067	000000G		MOV	R0,RET.STATUS	: 3937
000330	000402			BR	4\$: 3902
000332	016700	000000G	3\$:	MOV	RET.STATUS,R0	
000336	062706	000006	4\$:	ADD	#6,SP	
000342	000207			RTS	PC	

: Routine Size: 114 words, Routine Base: AC\$CODE + 5650
: Maximum stack depth per invocation: 9 words

: 4035

CZRCH5
REV A PATCH 00CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS11-Jul-1983 16:06:44
7-Jul-1983 08:24:58VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (20)

SEQ 237

Page 72

```

:      4036 global routine ON_LINE =                !Makes a unit come online to a host
:      4037
:      4038 :++
:      4039 Functional Description :
:      4040 The online command is used to bring a unit 'unit-online, set
:      4041 host settable unit characteristics and obtain those unit
:      4042 characteristics that are essential for proper class driver
:      4043 operation. The unit is spun-up, if necessary, and is heads
:      4044 are loaded prior to returning the online command's end
:      4045 message. Host settable characteristics are set exactly as if
:      4046 a set unit characteristics command were issued. Host settable
:      4047 characteristics are set after the unit has been successfully spun-up
:      4048 and any other validity checks have succeeded. Note that the unit's
:      4049 host settable characteristics are not altered if the unit is already
:      4050 'unit-online'.
:      4051
:      4052 Formal Parameters :
:      4053 none
:      4054
:      4055 Implicit Inputs :
:      4056 NSD_SLOT This global storage gets loaded by the routine
:      4057 'Get_nsd' and in it is stored the next send ring
:      4058 descriptor slot where the port/controller should
:      4059 be polling on and the place to put this commands
:      4060 command packet.
:      4061
:      4062 Implicit Outputs :
:      4063 none
:      4064
:      4065 Completion Codes :
:      4066 RET_STATUS: Return status passes back to the calling routine
:      4067 the status of the just issued command.
:      4068
:      4069 Side Effects :
:      4070 Any previously defined controller characteristics will possibly
:      4071 be altered after execution of this command.
:      4072 --
:      4073
:      4074 begin
:      4075
:      4076 local
:      4077 REF_NUM, !Stores unique cmd ref number
:      4078 ONL_BUF$LOC, !Stores outstanding cmd buffer location
:      4079 TEMP; !A place to put read IP register data
:      4080
:      4081
:      4082 : Before we load up the command packet up with all this good information
:      4083 : get the next send descriptor slot and a unique command reference number.
:      4084
:      4085 GET_NSD (); !Get the next send desc slot
:      4086 REF_NUM = GET_CMD$REF (); !Get a unique command ref num
:      4087
:      4088 : UQ Port command envelope Header field definition
:      4089
:      4090 SND_ENVELOPE [.NSD_SLOT, MSG_LENGTH] = SZ_ONL; !Load message length
:      4091 SND_ENVELOPE [.NSD_SLOT, CREDITS] = ONE; !Load credit size
:      4092 SND_ENVELOPE [.NSD_SLOT, MSG_TYPE] = 0; !Define message type 'Sequential'

```

CZRCH5
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555
SPIDERSUSERS:[NEALE.AZTEC]CZRCH5.B16;3 (20)

```

4093 SND_ENVELOPE [.NSD_SLOT, CONN_ID] = MSCP; !Define connection ID
4094
4095 : MSCP generic command envelope field definition
4096
4097 SND_ENVELOPE [.NSD_SLOT, CMD_LREF] = .REF_NUM; !Load command reference number
4098 SND_ENVELOPE [.NSD_SLOT, CMD_HREF] = ZERO; !Zero Hi order cmd ref num
4099 SND_ENVELOPE [.NSD_SLOT, UNIT_NUM] = .UNIT_NO; !Select unit to bring online
4100 SND_ENVELOPE [.NSD_SLOT, UN_HUSED] = ZERO; !Not used in DUP implimentation
4101 SND_ENVELOPE [.NSD_SLOT, OP_CODE] = OP_ONL; !Load this commands op-code
4102 SND_ENVELOPE [.NSD_SLOT, RSVD] = ZERO; !Not used field
4103 SND_ENVELOPE [.NSD_SLOT, MODIFIER] = ZERO; !Define this commands modifiers
4104
4105 : Command specific command envelope field definition
4106
4107 SND_ENVELOPE [.NSD_SLOT, RSV$D] = ZERO; !Reserved
4108 SND_ENVELOPE [.NSD_SLOT, UNT_FLAGS] = ZERO; !Unit flag field
4109 SND_ENVELOPE [.NSD_SLOT, RSV$0] = ZERO; !Reserved field
4110 SND_ENVELOPE [.NSD_SLOT, RSV$1] = ZERO; !Reserved field
4111 SND_ENVELOPE [.NSD_SLOT, RSV$2] = ZERO; !Reserved field
4112 SND_ENVELOPE [.NSD_SLOT, RSV$3] = ZERO; !Reserved field
4113 SND_ENVELOPE [.NSD_SLOT, RSV$4] = ZERO; !Reserved field
4114 SND_ENVELOPE [.NSD_SLOT, RSV$5] = ZERO; !Reserved field
4115 SND_ENVELOPE [.NSD_SLOT, DDP_LO] = ZERO; !Device dependent parameter
4116 SND_ENVELOPE [.NSD_SLOT, DDP_HI] = ZERO; !Device dependent parameter
4117 SND_ENVELOPE [.NSD_SLOT, SHADOW_UNIT] = ZERO; !Shadow unit
4118 SND_ENVELOPE [.NSD_SLOT, COPY_SPEED] = ZERO; !Copy speed
4119
4120 : Call the load outstanding command buffer routine
4121 : and load this command into the buffer. The return
4122 : from this routine will point us to the buffer location
4123 : where this command is stored. Later we can look at
4124 : this location to see if the interrupt service routine
4125 : has received and process it.
4126
4127 ONL_BUF$LOC = LOAD_OUT$STD_BUF (.REF_NUM); !Load the command
4128
4129 if .ONL_BUF$LOC eqlu OBF_CODE then DECODE (); !Error if buffer is full
4130
4131 :
4132 : Set the ownership bit to 1 giving this slot to the port/controller
4133
4134 SEND_RING [.NSD_SLOT, OWN_BIT] = PORT_OWNED;
4135
4136 : Read the IP register to stimulate port polling
4137
4138 TEMP = .RC25_ADDR [RCIP, RC_ALL];
4139
4140 : Time out the port/controller processing the command.
4141
4142 : The first test tests the connections ability to
4143 : respond to this command without any errors in the SA
4144 : register and for the command not timing out.
4145
4146 : The second tests the DUP server for good status. If
4147 : bad status is sent back then an error code is returned
4148 : to the calling routine where the routine "decode" will
4149 : decode and take the appropriate recovery. The time

```

```

:      4150      | out routine will loop on delaying and checking the hi
:      4151      | bit of the first word in the out$std_buf for a true.
:      4152      | When true signals us that the interrupt service routine
:      4153      | has received the endpacket and no connection errors
:      4154      | were detected.
:      4155      |
:      4156      |
:      4157      | if CTO_WAIT (ONE_MINUTE, .REF_NUM, .ONL_BUF$LOC) then DECODE ();      !Is return an error
:      4158      |
:      4159      |
:      4160      | Get the return envelope address from the out$std_buf
:      4161      | at this commands buffer location and check the packet
:      4162      | for good status error and die if bad status was returned
:      4163      |
:      4164      | RET_EN$AD = .OUT$STD_BUF [.ONL_BUF$LOC, ENV_ADR];      !Get the ret env adr
:      4165      |
:      4166      | Now test for good status
:      4167      |
:      4168      |
:      4169      | if .RET_EN$AD [STATUS] nequ ZERO      !Test the status
:      4170      | then
:      4171      |     return RET_STATUS = RSE_CODE      !Return a 'Response status err' code
:      4172      | else
:      4173      |     return .RET_STATUS;      !This ret_status is good or bad
:      4174      |
:      4175      | end;

```

000000	004167	000000G	ON.LINE:	.SBTTL ON.LINE MODULE DECLARATIONS	:	
000004	005746		JSR	R1,\$SAVE2	:	4036
000006	004767	171722	TST	-(SP)	:	4085
000012	004767	172064	JSR	PC,GET.NSD	:	4086
000016	010002		JSR	PC,GET.CMD\$REF	:	
000020	016746	000000G	MOV	R0,R2	: * .REF.NUM	
000024	012746	000054	MOV	NSD.SLOT, -(SP)	:	4090
000030	004767	000000G	MOV	#54, -(SP)	:	
000034	012760	000044 000000G	JSR	PC,BL\$MUL	:	
000042	012701	000002G	MOV	#44, SND.ENVELOPE(R0)	:	
000046	060001		MOV	#SND.ENVELOPE+2,R1	:	4091
000050	112711	000001	ADD	R0,R1	:	
000054	105061	000001	MOVB	#1,(R1)	:	4092
000060	010260	000004G	CLRB	1(R1)	:	4093
000064	005060	000006G	MOV	R2, SND.ENVELOPE+4(R0)	: REF.NUM,*	4097
000070	016760	000000G 000010G	CLR	SND.ENVELOPE+6(R0)	:	4098
000076	005060	000012G	MOV	UNIT.NO, SND.ENVELOPE+10(R0)	:	4099
000102	112760	000011 000014G	CLR	SND.ENVELOPE+12(R0)	:	4100
000110	105060	000015G	MOVB	#11, SND.ENVELOPE+14(R0)	:	4101
000114	005060	000016G	CLRB	SND.ENVELOPE+15(R0)	:	4102
000120	005060	000020G	CLR	SND.ENVELOPE+16(R0)	:	4103
000124	005060	000022G	CLR	SND.ENVELOPE+20(R0)	:	4107
000130	005060	000024G	CLR	SND.ENVELOPE+22(R0)	:	4108
000134	005060	000026G	CLR	SND.ENVELOPE+24(R0)	:	4109
000140	005060	000030G	CLR	SND.ENVELOPE+26(R0)	:	4110
000144	005060	000032G	CLR	SND.ENVELOPE+30(R0)	:	4111
000150	005060	000034G	CLR	SND.ENVELOPE+32(R0)	:	4112
			CLR	SND.ENVELOPE+34(R0)	:	4113

CZRCH5
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (20)

000154	005060	000036G		CLR	SND.ENVELOPE+36(R0)	:	4114
000160	005060	000040G		CLR	SND.ENVELOPE+40(R0)	:	4115
000164	005060	000042G		CLR	SND.ENVELOPE+42(R0)	:	4116
000170	005060	000044G		CLR	SND.ENVELOPE+44(R0)	:	4117
000174	005060	000046G		CLR	SND.ENVELOPE+46(R0)	:	4118
000200	010216			MOV	R2,(SP)	:	4127
000202	004767	171602		JSR	PC,LOAD.OUT\$STD.BUF	:	
000206	010001			MOV	R0,R1	:	
000210	020127	002001		CMP	R1,#2001	:	4129
000214	001002			BNE	1\$:	
000216	004767	171746		JSR	PC,DECODE	:	
000222	016700	000000G	1\$:	MOV	NSD.SLOT,R0	:	4134
000226	006300			ASL	R0	:	
000230	006300			ASL	R0	:	
000232	066700	000000G		ADD	SEND.RING,R0	:	
000236	052760	100000	000002	BIS	#100000,2(R0)	:	
000244	017766	000000G	000004	MOV	@RC25.ADDR,4(SP)	:	4138
000252	016600	000004		MOV	4(SP),R0	:	
000256	012716	165140		MOV	#-12640,(SP)	:	4157
000262	010246			MOV	R2,-(SP)	:	
000264	010146			MOV	R1,-(SP)	:	
000266	004767	173706		JSR	PC,CTO.WAIT	:	
000272	022626			CMP	(SP)+,(SP)+	:	
000274	006000			ROR	R0	:	
000276	103002			BCC	2\$:	
000300	004767	171664		JSR	PC,DECODE	:	
000304	010100		2\$:	MOV	R1,R0	:	4164
000306	006300			ASL	R0	:	
000310	006300			ASL	R0	:	
000312	016067	000002G	000000G	MOV	OUT\$STD.BUF+2(R0),RET.ENSAD	:	
000320	016000	000002G		MOV	OUT\$STD.BUF+2(R0),R0	:	4169
000324	005760	000016		TST	16(R0)	:	
000330	001405			BEQ	3\$:	
000332	012700	000031		MOV	#31,R0	:	4171
000336	010067	000000G		MOV	R0,RET.STATUS	:	
000342	000402			BR	4\$:	4074
000344	016700	000000G	3\$:	MOV	RET.STATUS,R0	:	
000350	062706	000006	4\$:	ADD	#6,SP	:	4036
000354	000207			RTS	PC	:	

: Routine Size: 119 words, Routine Base: AC\$CODE + 6214
: Maximum stack depth per invocation: 9 words

: 4176

CZRCH5
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555
SPIDERSUSERS:[NEALE.AZTEC]CZRCH5.B16;3 (21)

```

4177 global routine INT$I_SERVICE : INT_LNK$TYP novalue = !Init sequence interrupt catcher
4178
4179 !++
4180 Functional Description :
4181 During the initialization sequence the IE bit is defined to be
4182 a zero. This means that the host is not requesting interrupts at
4183 the completion of steps 1-3.
4184
4185 Note that no interrupt will be generated at the completion of
4186 step 4 since this step requires only a small number of time.
4187
4188 This interrupt service routine serves to catch any interrupts that the
4189 controller might issue during the initialization sequence. The
4190 interrupt is ignored and control is returned.
4191
4192 This interrupt service routine is also used during the RC25 register
4193 existence test in determining whether P_Table RC25 registers exist.
4194
4195 Formal Parameters :
4196 none
4197
4198 Implicit Inputs :
4199 NEX_FLAG A flag which is loaded with zeros during the RC25 register
4200 existence test and set to all ones by this routine in the
4201 event of attempts to read a non-existent RC25 controller.
4202
4203 Implicit Outputs :
4204 NEX_FLAG Is returned with all ones in the event of RC25 non-existent
4205 register access attempts.
4206
4207 Completion Codes :
4208 none
4209
4210 Side Effects :
4211 none
4212 --
4213
4214 begin
4215 NEX_FLAG = ONES; !Indicate this interrupt occurred
4216 return;
4217 end;

```

```

000000 012767 177777 000000G .SBTTL INT$I.SERVICE MODULE DECLARATIONS
000006 000002 INT$I.SERVICE::
MOV # -1,NEX.FLAG ;
RTI ;

```

4215
4177

: Routine Size: 4 words, Routine Base: AC\$CODE + 6572
: Maximum stack depth per invocation: 0 words

: 4218

CZRCH5
REV A PATCH 00CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS11-Jul-1983 16:06:44
7-Jul-1983 08:24:58VAX-11 Bliss-16 V3-555
SPIDER\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (22)SEQ 242
Page 77

```

:      4219 global routine IS_TIMER (SEQ_NO) =          !Init sequence time out
:      4220
:      4221 !++
:      4222 Functional Description :
:      4223     Steps 1-3 of the init sequence, each are required to complete within
:      4224     10 seconds.  If any of these steps fails to complete within that period,
:      4225     this is to be treated as a host detected fatal error.
:      4226
:      4227     This routine will do one us delays for a total of 10 seconds.  After
:      4228     each delay the step field is examined to see if this init sequence has completed.
:      4229
:      4230 Formal Parameters :
:      4231     SEQ_NO: Indicated which init step is presently being performed within the RC25 init sequence.
:      4232
:      4233 Implicit Inputs :
:      4234     none
:      4235
:      4236 Implicit Outputs :
:      4237     none
:      4238
:      4239 Completion Codes :
:      4240     TRUE:  Indicates to the calling routine that the indicated init sequence step has timed out.
:      4241
:      4242     FALSE: Indicates to the calling routine that the indicated init sequence
:      4243     step has not timed out.
:      4244
:      4245 Side Effects :
:      4246     If the init sequence step times out and an error is posted in the sa register
:      4247     then the routine decode will be call.
:      4248 !--
:      4249
:      4250 begin
:      4251
:      4252 local
:      4253     TO_VALUE : word,          !Step time out value
:      4254     STEP_VAL : word;         !Temp storage of step value
:      4255
:      4256 STEP_VAL = ZERO;           !Make sure the loc is zeroed out
:      4257
:      4258     Select the step value expected from this init sequence step.
:      4259
:      4260
:      4261 selectoneu .SEQ_NO of      !Select the binary step value
:      4262     set
:      4263
:      4264     [0] :
:      4265         begin
:      4266             STEP_VAL = %b'0001';    !Step 1 binary value
:      4267             TO_VALUE = 20000;      !Timeout step one for 60 seconds
:      4268         end;
:      4269
:      4270     [1] :
:      4271         begin
:      4272             STEP_VAL = %b'0010';    !Step 2 binary value
:      4273             TO_VALUE = 5000;       !Timeout step two for 10 seconds
:      4274         end;
:      4275

```

```

4276 [2] :
4277     begin
4278     STEP_VAL = %b'0100';           !Step 3 binary value
4279     TO_VALUE = 5000;             !Timeout step 3 for 10 seconds
4280     end;
4281
4282 [3] :
4283     begin
4284     STEP_VAL = %b'1000';           !Step 4 binary value
4285     TO_VALUE = 5000;             !Timeout step 4 for 10 seconds
4286     end;
4287 tes:
4288
4289 |
4290 | Loop on the 100 micro second delay until either the expected step field
4291 | is read in the SA register or the step times out.
4292 |
4293 |
4294 incru TIM_OUT from 0 to .TO_VALUE do      !Loop on C_US delay's
4295     begin
4296     DELAY (C_US);                       !Do the delay
4297     |
4298     | Check the step bit to see if it is set yet. If it is set then return
4299     | a false indicating the completion else continue delaying.
4300     |
4301     |
4302     if .RC25_ADDR [RCSA, STP_FIELD] eqlu .STEP_VAL then return FALSE;
4303
4304     BREAK;                               !Service any control C's
4305     end;
4306
4307 +
4308 | This step has not completed within the specified time interval. Test
4309 | the sa register for any errors posted and report errors if any. Return
4310 | a true to the caller indicating the error.
4311 |
4312 |
4313 if .RC25_ADDR [RCSA, ERR_BIT]             !Is the error bit set
4314 then
4315     begin
4316     RET_STATUS = PFE_CODE;             !Indicate the port/fatal error code
4317     DECODE ();                         !Report the error
4318     end;
4319
4320 return TRUE;                             !Return a failure to the caller
4321 end;

```

000000	004167	000000G	.SBTTL	IS.TIMER MODULE DECLARATIONS	
			IS.TIMER::		
			JSR	R1,\$SAVE4	4219
000004	162706	000006	SUB	#6,SP	
000010	005002		CLR	R2	: STEP.VAL
000012	016600	000022	MOV	22(SP),R0	: SEQ.NO,*
000016	001005		BNE	1\$	
000020	012702	000001	MOV	#1,R2	: *,STEP.VAL
000024	012703	047040	MOV	#47040,R3	: *,TO.VALUE
					4267

CZRCH5
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

000030	000423			BR	5\$:		4261
000032	020027	000001	1\$:	CMP	R0,#1	:		
000036	001003			BNE	2\$:		
000040	012702	000002		MOV	#2,R2	:	*,STEP.VAL	4272
000044	000413			BR	4\$:		4273
000046	020027	000002	2\$:	CMP	R0,#2	:		4261
000052	001003			BNE	3\$:		
000054	012702	000004		MOV	#4,R2	:	*,STEP.VAL	4278
000060	000405			BR	4\$:		4279
000062	020027	000003	3\$:	CMP	R0,#3	:		4261
000066	001004			BNE	5\$:		
000070	012702	000010		MOV	#10,R2	:	*,STEP.VAL	4284
000074	012703	011610	4\$:	MOV	#11610,R3	:	*,TO.VALUE	4285
000100	005004		5\$:	CLR	R4	:	TIM.OUT	4294
000102	000436			BR	11\$:		
000104	012701	000001	6\$:	MOV	#1,R1	:	*,\$\$TMP2	4296
000110	001411		7\$:	BEQ	10\$:		
000112	016700	000000G		MOV	L\$DLY,R0	:	*,\$\$TMP1	
000116	001404			BEQ	9\$:		
000120	005066	000004	8\$:	CLR	4(SP)	:	\$\$TMP	
000124	005300			DEC	R0	:	\$\$TMP1	
000126	001374			BNE	8\$:		
000130	005301		9\$:	DEC	R1	:	\$\$TMP2	
000132	000766			BR	7\$:		
000134	016700	000000G	10\$:	MOV	RC25.ADDR,R0	:		4302
000140	016066	000002 000002		MOV	2(R0),2(SP)	:	*,RC\$\$REG	
000146	010201			MOV	R2,R1	:	STEP.VAL,*	
000150	016600	000002		MOV	2(SP),R0	:	RC\$\$REG,*	
000154	006200			ASR	R0	:		
000156	006200			ASR	R0	:		
000160	006200			ASR	R0	:		
000162	000300			SWAB	R0	:		
000164	042700	177760		BIC	#177760,R0	:		
000170	020001			CMP	R0,R1	:		
000172	001421			BEQ	13\$:		
000174	104422			TRAP	22	:		
000176	005204			INC	R4	:	TIM.OUT	4294
000200	020403		11\$:	CMP	R4,R3	:	TIM.OUT,TO.VALUE	
000202	101740			BLOS	6\$:		
000204	016700	000000G		MOV	RC25.ADDR,R0	:		4313
000210	016016	000002		MOV	2(R0),(SP)	:	*,RC\$\$REG	
000214	100005			BPL	12\$:		
000216	012767	000021 000000G		MOV	#21,RET.STATUS	:		4316
000224	004767	171352		JSR	PC,DECODE	:		4317
000230	012700	000001	12\$:	MOV	#1,R0	:		4250
000234	000401			BR	14\$:		
000236	005000		13\$:	CLR	R0	:		4219
000240	062706	000006	14\$:	ADD	#6,SP	:		
000244	000207			RTS	PC	:		

; Routine Size: 83 words, Routine Base: AC\$CODE + 6602
; Maximum stack depth per invocation: 10 words

; 4322

!Performs RC25 init sequence

4323 global routine BOOT_RC25 =

4324

4325

4326 !++ Functional Description :

4327 This routine performs the initialization sequence of the RC25
4328 RC25 controller.

4329

4330

4331

4332

4333

4334

4335

4336

4337

4338

4339

4340

4341

4342

4343

4344

4345

4346

4347

4348

4349

4350

4351

4352

4353

4354

4355

4356

4357

4358

4359

4360

4361

4362

4363

4364

4365

4366

4367

4368

4369

4370

4371

4372

4373

4374

4375

4376

4377

4378

4379

!++

Functional Description :

This routine performs the initialization sequence of the RC25
RC25 controller.

The initialization procedure serves to:

1. Identify the parameters of the host-resident communications region to the port.
2. Provide a confidence check of port/controller integrity.
3. Bring the port/controller online to the host (note that the devices attached to the controller are not thereby brought online to the class driver.)

Formal Parameters :

none

Implicit Inputs :

ISD_STRUCT Stores the init sequence read and write data defined
for this program and controller.

Implicit Outputs :

none

Completion Codes :

Success: Is returned to the calling routine if this initialization
sequence was executed successfully.Failure: Is returned to the calling routine if this initialization
sequence was not executed successfully.

Side Effects :

Any DM code that might have been running in the DM machine will be
aborted.Any outstanding commands or response pertaining to a process using
the controller will be lost.

!--

begin

local

TEMP : word; !Temporary storage location

!+

The host begins the initialization sequence either by issuing a bus init or by writing any value into the IP register; the port must guarantee that the host will read zeros in SA on the next bus cycle. Initialization then sequences through steps 1-4 as per UQSSP.DOC Version 1.5.

CZRCH5
REV A PATCH 00CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS11-Jul-1983 16:06:44
7-Jul-1983 08:24:58VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (23)

SEQ 246

Page 81

```

4380      | Write to the IP register and start the init
4381      | sequence going.
4382      |
4383      |
4384      WRT_RC25 (RCIP, ONES);                !Begin init sequence
4385      |
4386      |
4387      |
4388      |
4389      |
4390      |
4391      |
4392      |
4393      |
4394      |
4395      |
4396      |
4397      |
4398      |
4399      |
4400      |
4401      |
4402      |
4403      |
4404      |
4405      |
4406      |
4407      |
4408      |
4409      |
4410      |
4411      |
4412      |
4413      |
4414      |
4415      |
4416      |
4417      |
4418      |
4419      |
4420      |
4421      |
4422      |
4423      |
4424      |
4425      |
4426      |
4427      |
4428      |
4429      |
4430      |
4431      |
4432      |
4433      |
4434      |
4435      |
4436      |

```

This incr loop performs all four steps of the initialization sequence described above. The SA write and read data is preset into the structure `ISD_STRUCT` and stands for "Initialization Sequence Data_STRUCT".

If a step time out error occurs the test invoking this routine will take the necessary retry procedure. A return code of failure is returned.

If any SA register compare error is detected after a step completion the routine Decode will decode the error and load statistical tables up pertanate data.

```

incru SEQ_NO from STEP1 to STEP4 do        !Do the four init seq steps
begin
  |
  | Wait for the controller to load the SA reg up with the step data.
  |
  if IS_TIMER (.SEQ_NO)                    !Did the Controller time out
  then
  begin
    |
    | DO SOME STAT TABLE UP DATA TO SHOW THE TIME OUT
    |
    PRINTB (.EMSG_STRUCT [MSG10]);
    return FAILURE;                        !Notify DRS> init of the failure
  end;

  |
  | The controller did not time out so read the SA register
  | for the expected step data and compare it to the good
  | data stored in ISD_STRUCT.
  |
  | If the read SA data is not what we expect then return a
  | failure code.
  |
  | Note that the reserved fields read in the SA register are
  | or'ed with all ones to mask out the field before compared
  | to the expected data stored in the structure "ISD_STRUCT".
  |
  TEMP = ((.RC25_ADDR [RCSA, RC_ALL]) or (.RSVD_STRUCT [.SEQ_NO]));

  if .TEMP nequ .ISD_STRUCT [.SEQ_NO, ISRD, ISR_ALL] !Compare read to expected
  then

```

CZRCH5
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (23)

```

:      4437      begin
:      4438      |
:      4439      | Load some satistical table up with some data to indicate that the
:      4440      | init sequence had some trouble.
:      4441      |
:      4442      | PRINTB (.EMSG_STRUCT [MSG11]);
:      4443      | return FAILURE;          !Return a failure code
:      4444      | end;
:      4445      |
:      4446      |
:      4447      | If this is step four then print this u-code version number.
:      4448      |
:      4449      |
:      4450      | if .SEQ_NO eqlu STEP4 then PRINTB (FMT4, .RC25_ADDR [RCSA, S4R_VER]);
:      4451      |
:      4452      |
:      4453      | This step read data is what we expected so write the SA register
:      4454      | with this steps write data stored in ISD_STRUCT.
:      4455      |
:      4456      | WRT_RC25 (RCSA, .ISD_STRUCT [.SEQ_NO, ISWRT, ISW_ALL]);
:      4457      | end;
:      4458      |
:      4459      |
:      4460      | The controller initialization sequence was done successfully so return a success code.
:      4461      |
:      4462      | return SUCCESS;
:      4463      | end;

```

000000	004167	000000G	SBTTL	BOOT.RC25 MODULE DECLARATIONS	
			BOOT.RC25::		
			JSR	R1,\$SAVE3	: 4323
000004	024646		CMP	-(SP),-(SP)	:
000006	012700	177777	MOV	#-1,R0	: *,RCSM.REG 4384
000012	010077	000000G	MOV	R0,@RC25.ADDR	: RCSM.REG,*
000016	005002		CLR	R2	: SEQ.NO 4405
000020	010246		1\$: MOV	R2,-(SP)	: SEQ.NO,* 4411
000022	004767	177504	JSR	PC,IS.TIMER	
000026	005726		TST	(SP)+	
000030	006000		ROR	R0	
000032	103007		BCC	2\$	
000034	016746	000024G	MOV	EMSG.STRUCT+24,-(SP)	: 4417
000040	012746	000001	MOV	#1,-(SP)	
000044	010600		MOV	SP,R0	: SP,*
000046	104414		TRAP	14	
000050	000427		BR	3\$: 4411
000052	016700	000000G	2\$: MOV	RC25.ADDR,R0	: 4433
000056	016066	000002 000002	MOV	2(R0),2(SP)	: *,RCSS.REG
000064	010200		MOV	R2,R0	: SEQ.NO,*
000066	006300		ASL	R0	
000070	016603	000002	MOV	2(SP),R3	: RCSS.REG,TEMP
000074	056003	000000G	BIS	RSVD.STRUCT(R0),R3	: *,TEMP
000100	010201		MOV	R2,R1	: SEQ.NO,* 4435
000102	006301		ASL	R1	
000104	006301		ASL	R1	
000106	020361	000000G	CMP	R3,ISD.STRUCT(R1)	: TEMP,*
000112	001410		BEQ	4\$	

CZRCH5
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555
SPIDER\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (23)

000114	016746	000026G	MOV	EMSG.STRUCT+26,-(SP)	:	4442	
000120	012746	000001	MOV	#1,-(SP)	:		
000124	010600		MOV	SP,R0	: SP,*		
000126	104414		TRAP	14	:		
000130	022626	3\$:	CMP	(SP)+,(SP)+	:	4435	
000132	000437		BR	6\$:	4437	
000134	020227	000003	4\$:	CMP	R2,#3	: SEQ.NO,*	4450
000140	001017		BNE	5\$:		
000142	016700	000000G	MOV	RC25.ADDR,R0	:		
000146	016016	000002	MOV	2(R0),(SP)	: *,RC\$S.REG		
000152	011646		MOV	(SP)-,(SP)	: RC\$S.REG,*		
000154	042716	177760	BIC	#177760,(SP)	:		
000160	012746	000000G	MOV	#FMT4,-(SP)	:		
000164	012746	000002	MOV	#2,-(SP)	:		
000170	010600		MOV	SP,R0	: SP,*		
000172	104414		TRAP	14	:		
000174	062706	000006	ADD	#6,SP	:		
000200	016101	000002G	5\$:	MOV	ISD.STRUCT+2(R1),R1	: *,RC\$M.REG	4456
000204	016700	000000G	MOV	RC25.ADDR,R0	:		
000210	010160	000002	MOV	R1,2(R0)	: RC\$M.REG,*		
000214	005202		INC	R2	: SEQ.NO	4405	
000216	020227	000003	CMP	R2,#3	: SEQ.NO,*		
000222	101676		BLOS	1\$:		
000224	012700	000001	MOV	#1,R0	:	4366	
000230	000401		BR	7\$:		
000232	005000	6\$:	CLR	R0	:	4323	
000234	022626	7\$:	CMP	(SP)+,(SP)+	:		
000236	000207		RTS	PC	:		

: Routine Size: 80 words, Routine Base: AC\$CODE + 7050
: Maximum stack depth per invocation: 11 words

: 4464

CZRCH5
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555
SPIDERSUSERS:[NEALE.AZTEC]CZRCH5.B16;3 (24)

SEQ 249

Page 84

```

4465 global routine INIT_COM_AREA =                !Inits DUP Protocol communication area
4466
4467 !++
4468 Functional Description :
4469 After initialization step 3 the port controller clears out
4470 the communication area's ring buffers.
4471
4472 This routine first makes sure that this protocol is accom-
4473 plished by the port before proceeding.
4474
4475 If the port did its part of the protocol then the communic-
4476 ations area is initialized as follows:
4477
4478 1. Defines from the contiguous data storage structure
4479 "COM_AREA" the header area address, receive ring
4480 address and the send ring address (these structures
4481 are initially declared as reference structures and
4482 require an address to be defined as its value per
4483 BLISS language conventions).
4484
4485 2. Clears the interrupt indicators and adaptor purge
4486 (ring base -1, -2, -3, -4) defined as "HEAD_AREA".
4487
4488 3. Loads the receive and send descriptors with the values:
4489 a. Envelope low, high and Q_bus address
4490 b. Reserved field
4491 c. Flag bit
4492 d. Ownership bit
4493
4494 4. Load the receive envelope message length field with the
4495 buffer size in bytes.
4496
4497 5. Initialize the Outstanding command buffer to reflect
4498 that all slots are unused.
4499
4500 Formal Parameters :
4501 none
4502
4503 Implicit Inputs :
4504 HEAD_AREA, RECEIVE_RING, SEND_RING, COM_AREA
4505
4506 Implicit Outputs :
4507 The communication area as a result of this routine will be initialized
4508 for host program to remote program communications per DUP and
4509 UQSSP specifications.
4510
4511 Completion Codes :
4512 TRUE: Error code to indicate the port controller has
4513 not fulfilled its part of the DUP protocol.
4514
4515 FALSE: An error code to indicate the port controller
4516 has fulfilled its part of the DUP protocol.
4517
4518 Side Effects :
4519 none
4520 --
4521

```

CZRCH5
REV A PATCH 00CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS11-Jul-1983 16:06:44
7-Jul-1983 08:24:58VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (24)

SEQ 250

Page 85

```

4522 begin
4523
4524 :+
4525 : Make sure that the controller has done its part of
4526 : the DUP protocol by clearing out the ring buffers.
4527 : If the rings are not cleared out then return with
4528 : an error code of true.
4529 :-
4530
4531 incru i from 2 to RING_SIZE - 1 do           !Test all blocks for zeros
4532
4533     incru j from WRD0 to WRD1 do           !Test all words for zeros
4534
4535         : Test this word for zeros. If not zeros then exit
4536         : this routine with an 'communication area init'
4537         : error code to indicate the Protocol violation.
4538
4539     if .COM_AREA [.i, .j, WORD_REF] nequ ZERO then return CIE_CODE;
4540
4541 :+
4542 : The port did its part of the protocol so now
4543 : define the address locations of the HEAD_AREA,
4544 : RECEIVE_RING and SEND_RING from the contiguous
4545 : storage declared by COM_AREA.
4546 :-
4547
4548 HEAD_AREA = COM_AREA;                       !Define the Header area
4549 RECEIVE_RING = COM_AREA [REC_BASE];         !Define the receive ring area
4550 SEND_RING = COM_AREA [SND_BASE];           !Define the send ring area
4551
4552 :+
4553 : Not quite sure if the port has to clear out
4554 : the header area of the communications area
4555 : so I'll clear it out here just in case.
4556 :-
4557
4558 incru i from WRD0 to WRD3 do
4559     HEAD_AREA [.i, WORD_REF] = ZEROS;
4560
4561 :+
4562 : Load up the Send Ring descriptors with an envelope address,
4563 : define the 'Flag bit' to = 1 (interrupt requested), define
4564 : the 'Ownership bit' to = 0 (owned by host) and load the
4565 : Reserved field with zeros (per DUP spec).
4566 :-
4567
4568 incru i from 0 to SND_ALLOCATE - 1 do
4569     begin
4570         SEND_RING [.i, LO_EN$AD] = SND_ENVELOPE [.i, CMD_LREF]; !Low-order envelope address for all sys
4571         SEND_RING [.i, HI_EN$AD] = ZERO;                       !High-order portion of an 18-bit U/Q bus adrs
4572         SEND_RING [.i, QB_EXT] = ZERO;                         !Q_bus extention
4573         SEND_RING [.i, D$RSVD] = ZERO;                         !Reserved field
4574         SEND_RING [.i, FLAG_BIT] = SET_FLG;                   !Flag bit whose meaning varies depending on dsc state
4575         SEND_RING [.i, OWN_BIT] = HOST_OWNED;                 !Indicates whether dsc is host or port owned
4576     end;
4577
4578

```

```

4579      :
4580      :
4581      : Load up the Receive Ring descriptors with an envelope
4582      : address, define the "Ownership bit" = 1 (owned by port),
4583      : define the "Flag bit" to = 1 (Interrupts requested) and
4584      : the reserved field set to zeros (per DUP spec).
4585      :
4586      :
4587      :
4588      :
4589      :
4590      :
4591      :
4592      :
4593      :
4594      :
4595      :
4596      :
4597      : Reset the communications area pointer to their initial state.
4598      :
4599      :
4600      :
4601      :
4602      :
4603      :
4604      :
4605      :
4606      :
4607      :
4608      :
4609      :
4610      :
4611      :
4612      :
4613      :
4614      :
4615      :
4616      :
4617      :
4618      :
4619      :
4620      :
4621      :
4622      :
4623      :
4624      :
4625      :
4626      :
4627      :
4628      :

```

```

      +
      | Load up the Receive Ring descriptors with an envelope
      | address, define the "Ownership bit" = 1 (owned by port),
      | define the "Flag bit" to = 1 (Interrupts requested) and
      | the reserved field set to zeros (per DUP spec).
      |
      |
      | incru i from 0 to REC_ALLOCATE - 1 do
      |   begin
      |     RECEIVE_RING [.i, LO_EN$AD] = REC_ENVELOPE [.i, CMD_LREF];
      |     RECEIVE_RING [.i, HI_EN$AD] = ZEROS;
      |     RECEIVE_RING [.i, QB_EXT] = ZEROS;
      |     RECEIVE_RING [.i, D$RSVD] = ZEROS;
      |     RECEIVE_RING [.i, FLAG_BIT] = SET_FLG;
      |     RECEIVE_RING [.i, OWN_BIT] = PORT_OWNED;
      |   end;
      |
      |
      | Reset the communications area pointer to their initial state.
      |
      | NRD_SLOT = -1;           !Start ring pointer at zero
      | NSD_SLOT = -1;         !Start ring pointer at zero
      | NXT_CRN = ZERO;        !Start unique cmd ref num at one
      |
      |
      | +
      | | Set the response envelope message length size equal
      | | to the buffer size in bytes starting at text + 0.
      | |
      | |
      | | incru i from 0 to REC_ALLOCATE - 1 do
      | |   REC_ENVELOPE [.i, MSG_LENGTH] = RB_SIZE*2;           !Convert to bytes before loading
      | |
      | |
      | | +
      | | | Init the outstanding command buffer as follows:
      | | | 1. Indicate that all slots are unused by loading
      | | |   the unique value %0'100000'.
      | | | 2. Clear the envelope adrs words to zero.
      | | |
      | | |
      | | | incru i from 0 to REC_ALLOCATE - 1 do
      | | |   begin
      | | |     OUT$STD_BUF [.i, CMD_WRD] = %0'100000'; !Define the slot as unused
      | | |     OUT$STD_BUF [.i, ENV_ADR] = ZERO;       !Clear out the envelope adrs field
      | | |   end;
      | | |
      | | |
      | | | No errors detected by this routine so return with an non-error code of false.
      | | |
      | | | return PAS_CODE;
      | | | end;

```

```

000000 004167 000000G
000004 012701 C00004
0C0010 005002

```

```

      .SBTTL  INIT.COM.AREA MODULE DECLARATIONS
      INIT.COM.AREA::
      JSR    R1,$SAVE3
      MOV    #4,R1
      CLR    R2

```

```

4465
4531
4533

```

CZRCH5
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

000012	010105		2\$:	MOV	R1,R0	:	I,*	4540
000014	060200			ADD	R2,R0	:	J,*	
000016	006300			ASL	R0			
000020	005760	000000G		TST	COM.AREA(R0)			
000024	001403			BEQ	3\$			
000026	012700	000001		MOV	#1,R0			
000032	000207			RTS	PC			
000034	005202		3\$:	INC	R2	:	J	4533
000036	020227	000001		CMP	R2,#1	:	J,*	
000042	101763			BLOS	2\$			
000044	062701	000002		ADD	#2,R1	:	*,I	4531
000050	020127	000022		CMP	R1,#22	:	I,*	
000054	101755			BLOS	1\$			
000056	012767	000000G 000000G		MOV	#COM.AREA,HEAD.AREA	:		4549
000064	012767	000010G 000000G		MOV	#COM.AREA+10,RECEIVE.RING	:		4550
000072	012767	000030G 000000G		MOV	#COM.AREA+30,SEND.RING	:		4551
000100	005000			CLR	R0	:	I	4559
000102	010001		4\$:	MOV	R0,R1	:	I,*	4560
000104	066701	000000G		ADD	HEAD.AREA,R1			
000110	005011			CLR	(R1)			
000112	062700	000002		ADD	#2,R0	:	*,I	4559
000116	020027	000006		CMP	R0,#6	:	I,*	
000122	101767			BLOS	4\$			
000124	005003			CLR	R3	:	I	4569
000126	010301		5\$:	MOV	R3,R1	:	I,*	4571
000130	006301			ASL	R1			
000132	006301			ASL	R1			
000134	010102			MOV	R1,R2			
000136	066702	000000G		ADD	SEND.RING,R2			
000142	010346			MOV	R3,-(SP)	:	I,*	
000144	012746	000054		MOV	#54,-(SP)			
000150	004767	000000G		JSR	PC,BLSMUL			
000154	062700	000004G		ADD	#SND.ENVELOPE+4,R0			
000160	010012			MOV	R0,(R2)			
000162	010100			MOV	R1,R0	:		4572
000164	066700	000000G		ADD	SEND.RING,R0			
000170	062700	000002		ADD	#2,R0			
000174	012710	040000		MOV	#40000,(R0)	:		4576
000200	022626			CMP	(SP)+,(SP)+	:		4570
000202	005203			INC	R3	:	I	4569
000204	020327	000003		CMP	R3,#3	:	I,*	
000210	101746			BLOS	5\$			
000212	005002			CLR	R2	:	I	4586
000214	010201		6\$:	MOV	R2,R1	:	I,*	4588
000216	006301			ASL	R1			
000220	006301			ASL	R1			
000222	010103			MOV	R1,R3			
000224	066703	000000G		ADD	RECEIVE.RING,R3			
000230	010200			MOV	R2,R0	:	I,*	
000232	000300			SWAB	R0			
000234	106000			RORB	R0			
000236	006000			ROR	R0			
000240	006000			ROR	R0			
000242	142700	000077		BICB	#77,R0			
000246	062700	000004G		ADD	#REC.ENVELOPE+4,R0			
000252	010013			MOV	R0,(R3)			
000254	010100			MOV	R1,R0	:		4589

CZRCH5
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
MODULE DECLARATIONS

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (24)

000256	066700	000000G		ADD	RECEIVE.RING,R0			
000262	062700	000002		ADD	#2,R0			
000266	012710	140000		MOV	#140000,(R0)	:		4593
000272	005202			INC	R2	:	I	4586
000274	020227	000003		CMP	R2,#3	:	I,*	
000300	101745			BLOS	6\$			
000302	012767	177777	000000G	MOV	#-1,NRD.SLOT	:		4599
000310	012767	177777	000000G	MOV	#-1,NSD.SLOT	:		4600
000316	105067	000000G		CLRB	NXT.CRN	:		4601
000322	005000			CLR	R0	:	I	4608
000324	012760	000074	000000G	MOV	#74,REC.ENVELOPE(R0)	:	*,*(I)	4609
000332	062700	000100		ADD	#100,R0	:	*,I	4608
000336	020027	000300		CMP	R0,#300	:	I,*	
000342	101770			BLOS	7\$			
000344	005000			CLR	R0	:	I	4618
000346	012760	100000	000000G	MOV	#-100000,OUT\$STD.BUF(R0)	:	*,*(I)	4620
000354	005060	000002G		CLR	OUT\$STD.BUF+2(R0)	:	*(I)	4621
000360	062700	000004		ADD	#4,R0	:	*,I	4618
000364	020027	000014		CMP	R0,#14	:	I,*	
000370	101766			BLOS	8\$			
000372	005000			CLR	R0	:		4522
000374	000207			RTS	PC	:		4465

: Routine Size: 127 words, Routine Base: AC\$CODE + 7310
: Maximum stack depth per invocation: 7 words

: 4629
: 4630 end
: 4631
: 4632 eludom

: OTS external references
: .GLOBL \$SAVE4, \$SAVE3, \$SAVE2, BLSMUL

: PSECT SUMMARY

Psect Name	Words	Attributes
AC\$CODE	2019	R0, I, LCL, REL, CON

: LIBRARY STATISTICS

File	----- Total	Symbols Loaded	----- Percent	Blocks Read
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH0.L16;5	398	278	69	55

CZRCH5 CZRCH RC25 DISK FORMATTER
REV A PATCH 00 MODULE DECLARATIONS

11-Jul-1983 16:06:44
7-Jul-1983 08:24:58

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH5.B16;3 (24)

COMMAND QUALIFIERS

:
: BLISS /PDP11/LIST CZRCH5.B16
: Size: 2019 code + 0 data words
: Run Time: 01:27.2
: Elapsed Time: 01:52.4
: Memory Used: 267 pages
: Compilation Complete

```

0001 MODULE CZRCH7 (%TITLE 'CZRCH RC25 DISK FORMATTER'
0002 IDENT = 'REV A PATCH 00',
0003 ADDRESSING MODE (RELATIVE),
0004 ENVIRONMENT (NOEIS)) =
0005 BEGIN
0006
0007 %SBTTL 'LAST ADDRESS AND SETUP SECTION'
0008
0009 | Pretty Declarations
0010
0011 | <blf/lowercase_key>
0012 |
0013
0014 LIBRARY 'CZRCH0'; !Define RC25 Formatter Library
0015
0016 REQUIRE 'BLSMAC.REQ'; !Define Bliss macro require file
1505
1506 |*
1507 | The LASTAD macro must be the final statement (except .end) in a pro-
1508 | gram. The call generates an even address reflecting the first word of
1509 | memory unused by the program.
1510 | -
1511
1512 LASTAD
1513
1514 |*
1515 | Hardcoded P-TABLES
1516
1517 | These optional hardware P-TABLES are located (when present) between
1518 | the 'LASTAD' macro and the '.END' statement. These hardware P-TABLES
1519 | are above and beyond the default hardware P-TABLE located in the main
1520 | body of the program. These P-TABLES wind up appended to the BIN file
1521 | of the diagnostic, just as though the supervisor or the 'SETUP' utility
1522 | had built them there. Thus the diagnostic can be 'pre-parameterized'
1523 | by the programmer.
1524
1525 | If this hardcoded P_TABLE section is not wanted then define 'number' in
1526 | the BGNSETUP macro to zero and omitt BGNTAB and ENDTAB macros.
1527
1528 | Coding sample is as follows:
1529
1530 | LASTAD
1531 |
1532 | BGNSETUP (Number) !Number of P-TABLES
1533 |
1534 | BGNPTAB
1535 | (DATA)
1536 | (DATA)
1537 | (DATA)
1538 | ENDPTAB
1539 |
1540 | BGNPTAB
1541 | (DATA)
1542 | (DATA)
1543 | (DATA)
1544 | ENDPTAB
1545

```


CZRCH7
REV A PATCH 00

CZRCH RC25 DISK FORMATTER
LAST ADDRESS AND SETUP SECTION

11-Jul-1983 16:08:39
7-Jul-1983 08:26:05

VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH7.B16;2 (1)

```

:      1546 :      ENDSETUP
:      1547 :      .END
:      1548 :      -
:      1549 :
:      1550 BGNSETUP (0);
:      1551 :
:      1552 :      No optional P_Tables are defined
:      1553 :      within this program.
:      1554 :
:      1555 ENDSETUP

```

```

:      .TITLE CZRCH7 CZRCH RC25 DISK FORMATTER
:      .IDENT /REV A /

```

```

000000      .PSECT $XYZ$, RO
000000 000004'  BL$LAS:::WORD  T$FREE
000002 000000C   .WORD  <<T$FREE-<BL$LAS+4>>/2>
000004 000000   T$FREE:::WORD  0

```

```

000004'      L$LAST==      BL$LAS+4
000000      T$Pihv==      0

```

```

000000 000207      .SBTTL SEND.LINK LAST ADDRESS AND SETUP SECTION
SEND.LINK:::
RTS      PC      ;

```

1504

```

; Routine Size: 1 word,      Routine Base: $XYZ$ + 0006
; Maximum stack depth per invocation: 0 words

```

```

:      1556 END
:      1557 ELUDOM

```

PSECT SUMMARY

Psect Name	Words	Attributes
\$XYZ\$	4	RO, I, LCL, REL, CON

LIBRARY STATISTICS

File	Total	Symbols Loaded	Percent	Blocks Read
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH0.L16;5	398	2	0	13

CZRCH7 CZRCH RC25 DISK FORMATTER
REV A PATCH 00 LAST ADDRESS AND SETUP SECTION

M 4
11-Jul-1983 16:08:39
7-Jul-1983 08:26:05

SEQ 257
Page 3
VAX-11 Bliss-16 V3-555
SPIDERS\$USERS:[NEALE.AZTEC]CZRCH7.B16;2 (1)

COMMAND QUALIFIERS

BLISS /PDP11/LIST CZRCH7.B16

: Size: 1 code + 3 data words
: Run Time: 00:06.4
: Elapsed Time: 00:08.5
: Memory Used: 99 pages
: Compilation Complete

CZRCHA.EXE Memory allocation map TKB M40.02 Page 1
11-JUL-83 16:09

Partition name : DUMMY
Identification : REV A
Task UIC : [300,12]
Task attributes: -HD
Total address windows: 1.
Task image size : 15552. words
Task address limits: 002000 076577
R-W disk blk limits: 000002 000076 000075 00061.

*** Root segment: CZRCH2

R/W mem limits: 002000 076577 074600 31104.
Disk blk limits: 000002 000076 000075 00061.

Memory allocation synopsis:

Section	Title	Ident	File
. BLK: (RW,I,LCL,REL,CON)	002000	000000	00000.
AAS\$COD: (RO,I,LCL,REL,CON)	002000	001336	00734.
	002000	000166	00118.
	002166	001150	00616.
AB\$COD: (RO,I,LCL,REL,CON)	003336	002312	01226.
	003336	002312	01226.
AC\$COD: (RO,I,LCL,REL,CON)	005650	007706	04038.
	005650	007706	04038.
AD\$COD: (RO,I,LCL,REL,CON)	015556	000002	00002.
	015556	000002	00002.
BL\$COD: (RO,I,LCL,REL,CON)	015560	000424	00276.
	015560	000316	00206.
	016076	000106	00070.
\$DMCOD: (RO,D,CBL,REL,CON)	016004	042570	17784.
	016204	042570	17784.
\$GLOB\$: (RO,D,GBL,REL,CON)	060774	002554	01388.
	060774	002554	01388.
\$SPLITS: (RO,D,GBL,REL,CON)	063550	013016	05646.
	063550	013016	05646.
\$XYZ\$: (RO,I,LCL,REL,CON)	076566	000010	00008.
	076566	000010	00008.

ABORT	010474-R	BOOT.F	075434-R	DFPTBL	002140-R	ERRTYP	002126-R	FMT4	063754-R	GPS1	002252-R	HW.Q1.	076076-R
ABO.MS	075204-R	BOOT.R	014720-R	DMSA	016244-R	EX.LOC	011706-R	FMT5	064040-R	GPS2	002262-R	HW.Q2.	076134-R
ACTIVE	075640-R	BR.LEV	063512-R	DUMMY	015556-R	EX.SUP	011270-R	FMT6	064150-R	GPS3	002272-R	HW.Q3.	076172-R
AZFMTR	016204-R	COM.AR	060774-R	DUP\$1.	007340-R	FCT.BU	061732-R	FMT7	064246-R	GPS4	002304-R	HW.Q4.	076230-R
BL\$DIV	016004-R	CRLF	064416-R	D\$PCNT	002122-R	FCT.RE	076016-R	GET.CM	006166-R	GPS5	002322-R	HW.UNI	002146-R
BL\$LAS	076566-R	CTO.WA	010264-R	EMSG.S	070430-R	FLG.WR	063462-R	GET.DU	010772-R	HDSA	016204-R	HW.VEC	002142-R
BL\$MOD	016016-R	DATETX	063446-R	ERRBLK	002134-R	FMT1	063550-R	GET.NR	006046-R	HEAD.A	061044-R	INACTI	075722-R
BL\$MUL	015560-R	DATMSG	075012-R	ERRMSG	002132-R	FMT2	063556-R	GET.NS	006020-R	HW.BR.	002144-R	INIT.C	015160-R
BL\$SHF	016030-R	DECODE	006254-R	ERRNBR	002130-R	FMT3	063646-R	GOOD.N	075342-R	HW.IP.	002140-R	INT\$1.	014442-R

CZRCHA.EXE Memory allocation map TKB M40.02
 CZRCH2 11-JUL-83 16:09

Page 2

ISD.ST 063530-R	L\$DISP 002124-R	L\$HPTP 002022-R	L\$PROT 002160-R	NEX.FL 063464-R	RECEIV 061046-R	SW.Q2. 076332-R
IS.TIM 014452-R	L\$DLY 002116-R	L\$HRDL 002250-R	L\$PRT 002112-R	NO.ADD 075044-R	REC.BU 062732-R	SW.Q3. 076422-R
LOAD.F 005650-R	L\$DTP 002040-R	L\$HW 002140-R	L\$REPP 002062-R	NRD.SL 063504-R	REC.DA 012760-R	SW.Q4. 076506-R
LOAD.O 006074-R	L\$DTYP 002034-R	L\$HWLE 002136-R	L\$REV 002010-R	NSD.SL 063502-R	REC.EN 061052-R	SW.UNA 002154-R
LUN 063474-R	L\$DU 003314-R	L\$ICP 002104-R	L\$RPT 002334-R	NXT.CR 063470-R	RET.EN 063444-R	TO.MAN 075264-R
L\$ACP 002110-R	L\$DUT 002072-R	L\$INIT 003236-R	L\$SFTL 002320-R	ON.LIN 014064-R	RET.ST 063472-R	T\$FREE 076572-R
L\$APT 002036-R	L\$DVTY 002214-R	L\$LADP 002026-R	L\$SOFT 002322-R	OUT\$ST 063424-R	RINGBA 061004-R	T\$PTHV 000000
L\$AU 003326-R	L\$EF 002052-R	L\$LAST 076572-R	L\$SPC 002056-R	OVSA 063466-R	RSVD.S 063520-R	T1 005634-R
L\$AUT 002070-R	L\$ENVI 002044-R	L\$LOAD 002100-R	L\$SPCP 002020-R	PFE.ST 066430-R	SDUP.S 074316-R	UNIT.N 063514-R
L\$AUTO 003250-R	L\$ERRT 002126-R	L\$LUN 002074-R	L\$SPTP 002024-R	PID.FM 064314-R	SEND.D 012220-R	VEC.AD 063510-R
L\$CCP 002106-R	L\$ETP 002102-R	L\$MREV 002050-R	L\$STA 002030-R	PID.SA 063476-R	SEND.R 061050-R	XCRLF 064422-R
L\$C EA 003302-R	L\$EXP1 002046-R	L\$NAME 002000-R	L\$SW 002154-R	PORT.I 075564-R	SET.CN 013520-R	\$END.L 076574-R
L\$CO 002032-R	L\$EXP4 002064-R	L\$NDHR 002316-R	L\$SWLE 002152-R	PROTO. 075516-R	SFPTBL 002154-R	\$SAVE2 016076-R
L\$DEPO 002011-R	L\$EXP5 002066-R	L\$NDHW 002150-R	L\$TEST 002114-R	PTBL.P 063516-R	SMSCP. 074760-R	\$SAVE3 016112-R
L\$DESC 002166-R	L\$HARD 002252-R	L\$NDSF 002330-R	L\$TIML 002014-R	PWR.MS 075122-R	SND.BU 063312-R	\$SAVE4 016130-R
L\$DESP 002076-R	L\$HIME 002120-R	L\$NDSW 002156-R	L\$UNIT 002012-R	RC.STR 073764-R	SND.EN 061452-R	\$SAVE5 016150-R
L\$DEVP 002060-R	L\$HPCP 002016-R	L\$PRIO 002042-R	MSGADR 062734-R	RC25.A 063506-R	SW.Q1. 076266-R	

*** Task builder statistics:

Total work file references: 15216.
 Work file reads: 0.
 Work file writes: 0.
 Size of core pool: 5486. words (21. pages)
 Size of work file: 2304. words (9. pages)

Elapsed time:00:00:14

CZRCHA CREATED BY TKB ON 11-JUL-83 AT 16:09 PAGE 1

GLOBAL CROSS REFERENCE

CREF V01

SYMBOL	VALUE	REFERENCES...
ABORT	010474-R	CZRCH4 # CZRCH5
ABO.MS	075204-R	# CZRCH2 CZRCH3
ACTIVE	075640-R	# CZRCH2 CZRCH4
AZFMTR	016204-R	CZRCH3 CZRCH5 # CZRCH6
BL\$DIV	016004-R	# B16MUL
BL\$LAS	076566-R	# CZRCH7
BL\$MOD	016016-R	# B16MUL
BL\$MUL	015560-R	# B16MUL CZRCH5
BL\$SHF	016030-R	# B16MUL
BOOT.F	075434-R	# CZRCH2 CZRCH4
BOOT.R	014720-R	CZRCH4 # CZRCH5
BR.LEV	063512-R	# CZRCH2 CZRCH3
COM.AR	060774-R	# CZRCH2 CZRCH5
CRLF	064416-R	# CZRCH2 CZRCH3 CZRCH4
CTO.WA	010264-R	# CZRCH5
DATETX	063446-R	# CZRCH2 CZRCH3 CZRCH4
DATMSG	075012-R	# CZRCH2 CZRCH3 CZRCH4
DECODE	006254-R	CZRCH3 CZRCH4 # CZRCH5
DFPTBL	002140-R	# CZRCH2
DMSA	016244-R	CZRCH5 # CZRCH6
DUMMY	015556-R	# CZRCH6
DUP\$1.	007340-R	CZRCH4 # CZRCH5
D\$PCNT	002122-R	# CZRCH2
EMSG.S	070430-R	# CZRCH2 CZRCH5
ERRBLK	002134-R	# CZRCH2
ERRMSG	002132-R	# CZRCH2
ERRNBR	002130-R	# CZRCH2
ERRTYP	002126-R	# CZRCH2
EX.LOC	011706-R	CZRCH4 # CZRCH5
EX.SUP	011270-R	CZRCH4 # CZRCH5
FCT.BU	061732-R	# CZRCH2 CZRCH4
FCT.RE	076016-R	# CZRCH2 CZRCH4
FLG.WR	063462-R	# CZRCH2 CZRCH3 CZRCH4
FMT1	063550-R	# CZRCH2 CZRCH4
FMT2	063556-R	# CZRCH2 CZRCH3
FMT3	063646-R	# CZRCH2 CZRCH3
FMT4	063754-R	# CZRCH2 CZRCH5
FMT5	064040-R	# CZRCH2 CZRCH4
FMT6	064150-R	# CZRCH2 CZRCH4
FMT7	064246-R	# CZRCH2 CZRCH4
CET.CM	006166-R	# CZRCH5
GET.DU	010772-R	CZRCH4 # CZRCH5
GET.NR	006046-R	# CZRCH5
GET.NS	006020-R	# CZRCH5
GOOD.N	075342-R	# CZRCH2 CZRCH3
GPS1	002252-R	# CZRCH3
GPS2	002262-R	# CZRCH3
GPS3	002272-R	# CZRCH3
GPS4	002304-R	# CZRCH3
GPS5	002322-R	# CZRCH3
HDSA	016204-R	CZRCH5 # CZRCH6
HEAD.A	061044-R	# CZRCH2 CZRCH5

CZRCHA CREATED BY TKB ON 11-JUL-83 AT 16:09 PAGE 2

GLOBAL CROSS REFERENCE

CREF V01

SYMBOL	VALUE	REFERENCES...
HW.BR.	002144-R	# CZRCH2
HW.IF.	002140-R	# CZRCH2
HW.Q1.	076076-R	# CZRCH2 CZRCH3
HW.Q2.	076134-R	# CZRCH2 CZRCH3
HW.Q3.	076172-R	# CZRCH2 CZRCH3
HW.Q4.	076230-R	# CZRCH2 CZRCH3
HW.UNI	002146-R	# CZRCH2
HW.VEC	002142-R	# CZRCH2
INACTI	075722-R	# CZRCH2 CZRCH4
INIT.C	015160-R	CZRCH4 # CZRCH5
INTSI.	014442-R	CZRCH4 # CZRCH5
ISD.ST	063530-R	# CZRCH2 CZRCH3 CZRCH4 CZRCH5
IS.TIM	014452-R	# CZRCH5
LOAD.F	005650-R	CZRCH3 # CZRCH5
LOAD.O	006074-R	# CZRCH5
LUN	063474-R	# CZRCH2 CZRCH3 CZRCH4
LSACP	002110-R	# CZRCH2
LSAPT	002036-R	# CZRCH2
LSAU	003326-R	CZRCH2 # CZRCH3
LSAUT	002070-R	# CZRCH2
LSAUTO	003250-R	CZRCH2 # CZRCH3
LSCCP	002106-R	# CZRCH2
LSLEA	003302-R	CZRCH2 # CZRCH3
LSCO	002032-R	# CZRCH2
LSDEPO	002011-R	# CZRCH2
LSDESC	002166-R	CZRCH2 # CZRCH3
LSDESP	002076-R	# CZRCH2
LSDEVP	002060-R	# CZRCH2
LSDISP	002124-R	# CZRCH2
LSDLY	002116-R	# CZRCH2 CZRCH5
LSDTP	002040-R	# CZRCH2
LSDTYP	002034-R	# CZRCH2
LSDU	003314-R	CZRCH2 # CZRCH3
LSDUT	002072-R	# CZRCH2
LSDVTY	002214-R	CZRCH2 # CZRCH3
LSEF	002052-R	# CZRCH2
LSENV1	002044-R	# CZRCH2
LSERRT	002126-R	# CZRCH2
LSETP	002102-R	# CZRCH2
LSEXP1	002046-R	# CZRCH2
LSEXP4	002064-R	# CZRCH2
LSEXP5	002066-R	# CZRCH2
LSHARD	002252-R	CZRCH2 # CZRCH3
LSHIME	002120-R	# CZRCH2
LSHPCP	002016-R	# CZRCH2
LSHPTP	002022-R	# CZRCH2
LSHRDL	002250-R	# CZRCH3
LSHW	002140-R	# CZRCH2
LSHWLE	002136-R	# CZRCH2
LSICP	002104-R	# CZRCH2
LSINIT	003236-R	CZRCH2 # CZRCH3
LSLADP	002026-R	# CZRCH2

CZRCHA CREATED BY TKB ON 11-JUL-83 AT 16:09 PAGE 3

GLOBAL CROSS REFERENCE

CREF V01

SYMBOL	VALUE	REFERENCES...
L\$LAST	076572-R	CZRCH2 # CZRCH7
L\$LOAD	002100-R	# CZRCH2
L\$LUN	002074-R	# CZRCH2
L\$MREV	002050-R	# CZRCH2
L\$NAME	002000-R	# CZRCH2
L\$NDHR	002316-R	# CZRCH3
L\$NDHW	002150-R	# CZRCH2
L\$NDSF	002330-R	# CZRCH3
L\$NDSW	002156-R	# CZRCH2
L\$PRIO	002042-R	# CZRCH2
L\$PROT	002160-R	# CZRCH2
L\$PRT	002112-P	# CZRCH2
L\$REPP	002062-R	# CZRCH2
L\$REV	002010-R	# CZRCH2
L\$RPT	002334-R	CZRCH2 # CZRCH3
L\$SFTL	002320-R	# CZRCH3
L\$SOFT	002322-R	CZRCH2 # CZRCH3
L\$SPC	002056-R	# CZRCH2
L\$SPCP	002020-R	# CZRCH2
L\$SPTP	002024-R	# CZRCH2
L\$STA	002030-R	# CZRCH2
L\$SW	002154-R	# CZRCH2
L\$SWLE	002152-R	# CZRCH2
L\$TEST	002114-R	# CZRCH2
L\$TIML	002014-R	# CZRCH2
L\$UNIT	002012-R	# CZRCH2 CZRCH3
MSGADR	062734-R	# CZRCH2 CZRCH4
NEX.FL	063464-R	# CZRCH2 CZRCH4 CZRCH5
NO.ADD	075044-R	# CZRCH2 CZRCH3
NRD.SL	063504-R	# CZRCH2 CZRCH4 CZRCH5
NSD.SL	063502-R	# CZRCH2 CZRCH4 CZRCH5
NXT.CR	063470-R	# CZRCH2 CZRCH4 CZRCH5
ON.LIN	014064-R	CZRCH4 # CZRCH5
OUT\$ST	063424-R	# CZRCH2 CZRCH5
OVSA	063466-R	# CZRCH2 CZRCH3 CZRCH5
PFE.ST	066430-R	# CZRCH2 CZRCH5
PID.FM	064314-R	# CZRCH2 CZRCH5
PID.SA	063476-R	# CZRCH2 CZRCH4 CZRCH5
PORT.I	075564-R	# CZRCH2 CZRCH4
PROTO.	075516-R	# CZRCH2 CZRCH4
PTBL.P	063516-R	# CZRCH2 CZRCH3 CZRCH4
PWR.MS	075122-R	# CZRCH2 CZRCH3
RC.STR	073764-R	# CZRCH2 CZRCH5
RC25.A	063506-R	# CZRCH2 CZRCH3 CZRCH4 CZRCH5
RECEIV	061046-R	# CZRCH2 CZRCH5
REC.BU	062732-R	# CZRCH2 CZRCH4 CZRCH5
REC.DA	012760-R	CZRCH4 # CZRCH5
REC.EN	061052-R	# CZRCH2 CZRCH5
RET.EN	063444-R	# CZRCH2 CZRCH4 CZRCH5
RET.ST	063472-R	# CZRCH2 CZRCH4 CZRCH5
RINGBA	061004-R	# CZRCH?
RSVD.S	063520-R	# CZRCH2 CZRCH5

CZRCHA CREATED BY TKB ON 11-JUL-83 AT 16:09 PAGE 4

GLOBAL CROSS REFERENCE CREF V01

SYMBOL	VALUE	REFERENCES...
SDUP.S	074316-R	# CZRCH2 CZRCH5
SEND.D	012220-R	CZRCH4 # CZRCH5
SEND.R	061050-R	# CZRCH2 CZRCH5
SET.CN	013520-R	CZRCH4 # CZRCH5
SFPTBL	002154-R	# CZRCH2
SMSCP.	074760-R	# CZRCH2 CZRCH5
SND.BU	063_12-R	# CZRCH2 CZRCH4 CZRCH5
SND.EN	061452-R	# CZRCH2 CZRCH5
SW.Q1.	076266-R	# CZRCH2 CZRCH3
SW.Q2.	076332-R	# CZRCH2 CZRCH3
SW.Q3.	076422-R	# CZRCH2 CZRCH3
SW.Q4.	076506-R	# CZRCH2 CZRCH3
SW.UNA	002154-R	# CZRCH2 CZRCH3 CZRCH4
TO.MAN	075264-R	# CZRCH2 CZRCH3
T\$FREE	076572-R	# CZRCH7
T\$PTHV	000000	CZRCH2 # CZRCH7
T1	005634-R	CZRCH2 # CZRCH4
UNIT.N	063514-R	# CZRCH2 CZRCH3 CZRCH4 CZRCH5
VEC.AD	063510-R	# CZRCH2 CZRCH3 CZRCH4 CZRCH5
XCRLF	064422-R	# CZRCH2 CZRCH3
\$END.L	076574-R	# CZRCH7
\$SAVE2	016076-R	B16MUL # B16SAV CZRCH5
\$SAVE3	016112-R	# B16SAV CZRCH4 CZRCH5
\$SAVE4	016130-R	# B16SAV CZRCH5
\$SAVE5	016150-R	B16MUL # B16SAV