

# RK611

USER DEFINED TEST  
CZR6RC0

AH-9149C-MC

COPYRIGHT © 76-78

FICHE 1 OF 1

MAR 1978

**digital**

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



EOF1CZR6QBSEQ 00010000 780223 PDP10 411 12H0R1CZR6RCSEQ 00010000 780223  
 P02000001 CZR6RCO RK611/06 USR DEFINED MACY11 30(1046) 02-DEC-77 11:59 PAGE 2  
 CZR6RC.P11 02-DEC-77 11:07

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

.REM %

# IDENTIFICATION

PRODUCT CODE: AC-9148C-MC  
 PRODUCT NAME: CZR6RCO RK611/06 USER DEFINED TEST  
 DATE: FEB 1978  
 MAINTAINER: DIAGNOSTIC GROUP  
 AUTHOR: MARV TEGROTENHUIS

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.

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED UNDER A LICENSE AND MAY ONLY BE USED OR COPIED IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE.

DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.

COPYRIGHT (C) 1976, 1978 BY DIGITAL EQUIPMENT CORPORATION

TABLE OF CONTENTS

1.0	ABSTRACT
2.0	REQUIREMENTS
2.1	HARDWARE REQUIREMENTS
2.2	PRELIMINARY PROGRAMS
3.0	OPERATING PROCEDURE AND CONTROL FUNCTIONS
3.1	PROGRAM LOADING
3.2	STARTING LOCATIONS
3.3	CONSOLE SWITCH REGISTERS
3.4	'SOFTWARE' SWITCH REGISTER
3.5	UNIBUS ADDRESSES
3.6	EXECUTION TIME
3.7	TEST PROGRAM SIZE
4.0	USER DEFINED TEST FUNCTIONAL DESCRIPTION
4.1	IMMEDIATE COMMAND SET
4.1.1	DRIVE SELECTION
4.1.2	OUTPUT TEST TO PAPER TAPE
4.1.3	COPY TAPE
4.1.4	INPUT TEST FROM PAPER TAPE
4.1.5	TIMEOUT
4.1.6	ITERATION COUNT
4.1.7	SPECIAL DATA PATTERN
4.1.8	EDIT BUFFER
4.1.9	BUFFER DUMP
4.1.10	COMPILE
4.1.11	RUN
4.1.12	EDIT ADD LINE
4.1.13	EDIT DELETE LINE
4.1.14	PRINT TEST
4.1.15	PRINT LINE
4.1.16	NEW TEST
4.1.17	PRINT REGISTER
4.1.18	HELP
4.1.19	FORMAT SELECT
4.1.20	DRIVE TYPE SELECT
4.2	DEFERRED COMMAND SET
4.2.1	SUBSYSTEM FUNCTION COMMAND
4.2.2	BUFFER INITIALIZE
4.2.3	DATA COMPARE
4.2.4	STATUS COMPARE
4.2.5	REGISTER COMPARE
4.2.6	REGISTER WRITE
4.2.7	STALL
4.2.8	PRINT MESSAGE
4.2.9	UNIBUS INITIALIZE

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

4.3 LINE NUMBERING  
4.4 TIME OUT  
4.5 TEST LOOPING AND LOOP COUNTERS

5.0 ERROR REPORTING FORMATS

5.1 FORMAT 1  
5.1.1 SUBSYSTEM DETECTED ERROR  
5.1.2 UNSOLICITED ATTENTION  
5.1.3 UNEXPECTED DATA TYPE ERROR  
5.1.4 ATTENTION DID NOT RESET WITH DRIVE CLEAR  
5.1.5 ATTENTION DID NOT CLEAR WITH  
SUBSYSTEM CLEAR  
5.1.6 ILLEGAL DRIVE COMMAND  
5.1.7 SUBSYSTEM TIMEOUT  
5.1.8 CLEAR CONTROLLER DID NOT CLEAR ERROR  
5.1.9 NO ATTENTION IN ATTENTION SUMMARY REGISTER  
5.1.10 DATA LATE WHEN UNLOADING HEADER  
5.1.11 CONTROLLER ERROR WHILE DRIVER SERVICING  
5.1.12 DRIVE PARITY WHILE GATHERING STATUS  
5.1.13 MULTIPLE DRIVE SELECT

5.2 FORMAT 2

APPENDIX A - DATA PATTERNS

APPENDIX B - COMMAND SUMMARIES



1.0 ABSTRACT

THE USER DEFINED TEST PROGRAM PROVIDES THE CAPABILITY OF ENTERING, EDITING, SAVING, RECALLING, AND EXECUTING TEST PROGRAMS DESIGNED BY THE USER.

THE USER DEFINED TEST OPERATES INTERACTIVELY TO ALLOW THE USER TO DEVELOP A SPECIFIC TEST MADE UP OF SUBSYSTEM COMMANDS, CHECKING, AND REPORTING IN ANY SEQUENCE.

AN INTERACTIVE COMMAND SET IS DEFINED TO BE USED IN ENTERING, STORING, RETRIEVING, EDITING, AND EXECUTING TESTS. THIS COMMAND SET INCLUDES OTHER COMMANDS THAT PERFORM COMPARE OPERATIONS, CHANGE TEST CONTROL VALUES, INITIALIZE THE BUFFER, AND INITIALIZE THE SUBSYSTEM.

THE INTERACTIVE COMMAND SET IS DIVIDED INTO TWO TYPES OF COMMANDS. THESE ARE:

- \* DEFERRED WHICH ARE THE COMMANDS THAT MAKE UP THE TESTS.
- \* IMMEDIATE WHICH ARE EXECUTED WHEN THEY ARE ENTERED.

WHEN THE DEFERRED COMMANDS ARE ENTERED THEY ARE STORED IN CORE IN A SOURCE AREA. EDITING CAN BE DONE ON THE STORED SOURCE TO ADD OR DELETE SPECIFIC LINES. AFTER THE SOURCE HAS BEEN ENTERED, IT IS "COMPILED" INTO OBJECT CODE, STORED IN CORE IN AN OBJECT AREA, AND EXECUTED. ONCE EXECUTION BEGINS, THE SOURCE CODE IS LOST. THE OBJECT CODE IS PRESERVED UNTIL ANOTHER COMPILE AND CAN BE REEXECUTED. EITHER THE SOURCE CODE BEFORE EXECUTION OR THE OBJECT CODE AFTER COMPILATION CAN BE PUNCHED OUT ON PAPER TAPE ALONG WITH ANY SPECIAL DATA PATTERNS ENTERED. CONVERSELY, EITHER TYPE OF CODE CAN BE READ FROM PAPER TAPE. SOURCE CODE IS PLACED IN THE SOURCE AREA AND OBJECT CODE IS PLACED IN THE OBJECT AREA AND EXECUTED. IF SPECIAL DATA PATTERNS WERE PUNCHED, THESE PATTERNS ARE PLACED IN THE APPROPRIATE BUFFER WHEN THE TAPE IS READ.

THE IMMEDIATE COMMANDS ARE EXECUTED WHEN THEY ARE ENTERED. THESE COMMANDS ARE NOT ENTERED INTO THE SOURCE AREA AND DO NOT BECOME PART OF THE COMPILED TEST. THEY CAN BE EXECUTED AT ANY TIME EXCEPT WHILE COMPILING OR EXECUTING A TEST.

THE GENERAL INTERACTIVE COMMAND (WITH THE EXCEPTION OF THE SPECIAL DATA PATTERN COMMAND, SEE PARAGRAPH 8.1.5) FORMAT IS:

INTERACTIVE COMMAND,PARAMETER1,PARAMETER2,...(RETURN)

THE INTERACTIVE COMMAND IS TESTED FOR LEGALITY. IF IT IS NOT ONE OF THE DEFINED COMMANDS, THE COMMAND IS REJECTED AND AN ERROR MESSAGE IS PRINTED. THE COMMAND IN ERROR IS ECHOED BACK TO SHOW THE ERROR AND THE ENTIRE COMMAND MUST BE REENTERED. THE PARAMETERS ARE ACCEPTED WITHOUT CHECKING UNLESS SPECIFIC CHECKING IS DEFINED FOR THAT COMMAND IN THE FOLLOWING COMMAND

DESCRIPTIONS.

ANY ONE OR ALL OF THE PARAMETERS MAY BE OMITTED. AN OMITTED  
PARAMETER WILL DEFAULT TO THE VALUE LAST SPECIFIED FOR THAT  
PARAMETER. A CARRIAGE RETURN WILL CAUSE THE REMAINING PARAMETERS  
TO DEFAULT. IF A PARAMETER IS TO BE SPECIFIED AFTER ONE THAT IS  
OMITTED, THE SEPARATOR (,) MUST BE PROVIDED.

2.0 REQUIREMENTS

2.1 HARDWARE REQUIREMENTS

THE FOLLOWING HARDWARE IS REQUIRED TO RUN THE USER DEFINED TEST:

PDP-11 SYSTEM (16K MEMORY)  
CONSOLE TERMINAL  
RK06 UNIBUS CONTROLLER  
1 TO 8 RK06/RK07 DRIVES  
RK06-RK07 DISK CARTRIDGE.  
PAPER TAPE READER (OPTIONAL)  
PAPER TAPE PUNCH (OPTIONAL)

2.2 PRELIMINARY PROGRAMS

THE CONTROLLER DIAGNOSTIC AND/OR DRIVE DIAGNOSTIC SHOULD BE RUN  
TO DIAGNOSE FAULTS. HOWEVER, THIS PROGRAM DOES NOT RELY ON AN  
OPERATIONAL SUBSYSTEM. FEATURES ARE PROVIDED TO FACILITATE  
LOOPING ON A TEST OR ERROR FOR TROUBLESHOOTING PURPOSES.

3.0 OPERATING PROCEDURE AND CONTROL FUNCTIONS

3.1 PROGRAM LOADING

THE PROGRAM CAN BE LOADED FROM PAPER TAPE USING THE ABSOLUTE  
LOADER OR FROM ANY MEDIA SUPPORTED BY XXDP BUT IT IS NOT  
CHAINABLE. IT CAN ALSO BE LOADED BY ACT OR APT IN DUMP MODE  
ONLY.

THE PROGRAM DOES NOT DESTROY THE LOADER.

### 3.2 STARTING LOCATIONS

THE STARTING ADDRESS FOR THE USER DEFINED TEST IS 200. THE PROGRAM IDENTIFIES ITSELF, COMPUTES AND TYPES THE MAXIMUM NUMBER OF WORDS FOR DATA TRANSFER COMMANDS (BASED ON MEMORY SIZE) AND TYPES THE MESSAGE "TYPE HP TO PRINT HELP FILE". A STAR (\*) IS THEN PRINTED TO SIGNIFY THE PROGRAM IS READY FOR A COMMAND.

THE RESTART ADDRESS IS ALSO 200. THE SAME MESSAGES ARE PRINTED EXCEPT FOR THE "HELP" MESSAGE. THE HELP FILE IS AVAILABLE ONLY WHEN THE PROGRAM IS INITIALLY LOADED.

### 3.3 CONSOLE SWITCH REGISTER

THE CONSOLE SWITCHES ARE USED TO PROVIDE CONTROL FUNCTIONS. THESE FUNCTIONS AND SWITCH ASSIGNMENTS GENERALLY CONFORM TO THE SYSMAC STANDARD WITH THE EXCEPTIONS NOTED:

SWITCH -----	FUNCTION -----
15	HALT ON ERROR
14	LOOP ON TEST. WHEN THE SWITCH IS RESET AFTER THE TEST HAS BEEN LOOPING THE PROGRAM WILL TYPE NNN=NUMBER OF LOOPS. NNN IS A DECIMAL NUMBER.
13	INHIBIT ERROR TIMEOUT
11	INHIBIT INERATION
10	BELL ON ERROR
9	LOOP ON ERROR. THIS SWITCH CONFORMS TO THE STANDARD IN THAT WHEN THE ERROR IS DETECTED, THE TEST PRESENTLY UNDER EXECUTION IS RESTARTED WITH THE FIRST COMMAND. IF THE TEST EVER COMPLETES WITHOUT AN ERROR THE TEST IS AGAIN STARTED FROM THE FIRST COMMAND AS LONG AS SWITCH 9 REMAINS SET. WHEN SWITCH 9 IS RESET THE LOOP WILL TERMINATE AFTER 1 MORE PASS AND TYPE THE NUMBER OF LOOPS AS WHEN SWITCH 9 WAS RESET.
2	INHIBIT ALL DATA COMPARE ERROR REPORTING.
1	WHEN SET FORCE REPORTING OF ALL DATA COMPARE ERRORS. WHEN RESET REPORT ONLY THE FIRST 10 COMPARE ERRORS.
0	WHEN SET FORCE SHORT ERROR REPORT. WHEN RESET FULL ERROR REPORT IS GIVEN.

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



### 3.4 'SOFTWARE' SWITCH REGISTER

IF THE PROGRAM IS BEING RUN ON A SWITCHLESS PROCESSOR (I.E. AN 11/04 OR 11/34) THE PROGRAM WILL DETERMINE THAT THE HARDWARE SWITCH REGISTER IS NOT PRESENT AND WILL USE A 'SOFTWARE' SWITCH REGISTER. THE 'SOFTWARE' SWITCH REGISTER IS LOCATED AT LOCATION 176(8). THE SETTINGS OF THE 'SOFTWARE' SWITCHES ARE CONTROLLED THROUGH A KEYBOARD ROUTINE WHICH IS CALLED BY TYPING A 'CONTROL G'. THE PROGRAM WILL RECOGNIZE THE 'CONTROL G' AT ANY TIME EXCEPT WHEN THE PROGRAM IS AT A HIGHER PRIORITY PROCESSING AN RK611 INTERRUPT. THE 'SOFTWARE' SWITCH VALUES ARE ENTERED AS AN OCTAL NUMBER IN RESPONSE TO THE PROMPT FROM THE SWITCH ENTRY ROUTINE:

SWR = NNNNNN NEW =

EACH TIME SWITCH SETTING ARE ENTERED, THE ENTIRE SWITCH REGISTER IMAGE MUST BE ENTERED. LEADING ZEROS ARE NOT REQUIRED. 'RUBOUT' AND 'CONTROL U' FUNCTIONS MAY BE USED TO CORRECT TYPING ERRORS DURING SWITCH ENTRY.

ON PROCESSORS WITH HARDWARE SWITCH REGISTERS, THE 'SOFTWARE' SWITCH REGISTER MAY BE USED. IF THE PROGRAM FINDS ALL 16 SWITCHES IN THE 'UP' POSITION, ALL SWITCH REGISTER REFERENCES WILL BE TO THE 'SOFTWARE' REGISTER AND THE PROCEDURES DESCRIBED ABOVE MUST BE FOLLOWED.

### 3.5 UNIBUS ADDRESSES

UNIBUS AND VECTOR ADDRESS OF THE RK06 PAPER TAPE READER, AND PAPER TAPE PUNCH CAN BE CHANGED FROM THE DEFAULT AS PART OF THE PROGRAM STARTUP PROCEDURE. THE METHOD OF CHANGING THE PARAMETERS IS TO ALTER THE MEMORY LOCATIONS SPECIFIED BEFORE THE PROGRAM IS STARTED. THE DEFAULT VALUES OF THESE PARAMETERS ARE:

	UNIBUS ADDRESS	VECTOR
	-----	-----
RK06-RK07	277400	210
TAG NAME	RKBAS	RKVEC
LOCATION	23342	23344
PAPER TAPE READER		
DATA BUFFER	177552	
TAG NAME	PTRDB	NOT
LOCATION	1730	USED
STATUS REG	177550	
TAG NAME	PTRSR	
LOCATION	1726	

PAPER TAPE		
PUNCH		
DATA BUFFER	177556	
TAG NAME	PTPDB	NOT
LOCATION	1734	USED
STATUS REG	177554	
TAG NAME	PTPSR	
LOCATION	1732	

### 3.6 EXECUTION TIME

EXECUTION TIME WILL DEPEND ON THE SPECIFIC TEST DEFINED BY THE USER.

### 3.7 TEST PROGRAM SIZE

THE TEST PROGRAM SIZE IS LIMITED BY THE STORAGE PROVIDED IN THE PROGRAM FOR SOURCE AND OBJECT CODE. IT IS NOT POSSIBLE TO SPECIFY THE EXACT MAXIMUM NUMBER OF SOURCE LINES POSSIBLE SINCE THE SOURCE LINES AND THE RESULTANT OBJECT CODE VARIES FROM COMMAND TO COMMAND. HOWEVER, THE APPROXIMATE MAXIMUM IS 200 SOURCE LINES. WHEN THE TEST IS ENTERED AND WHEN THE TEST IS COMPILED CHECKS ARE PERFORMED TO INSURE THAT NEITHER THE SOURCE OR OBJECT CODE WILL EXCEED ITS RESPECTIVE STORAGE.

### 4.0 USER DEFINED TEST FUNCTIONAL DESCRIPTION

THE INTERACTIVE COMMAND SET IS DESCRIBED IN THE FOLLOWING PARAGRAPHS. THE IMMEDIATE COMMAND SET IS DESCRIBED FIRST, FOLLOWED BY THE DEFERRED COMMAND SET.

THE FOLLOWING IS AN EXAMPLE OF WHAT IS CONSIDERED TYPICAL USAGE OF THE USER DEFINED TEST. THE USER WILL USE THE IMMEDIATE COMMANDS TO SET UP THE ITERATION COUNT AND TIMEOUT, SELECT THE DRIVE, INPUT DATA PATTERNS, ETC. THEN, CHOSING FROM THE DEFERRED COMMANDS, THE TEST IS ENTERED. AT ANY TIME WHILE ENTERING THE TEST THE PRINT AND EDIT COMMANDS MAY BE USED TO DISPLAY AND/OR CHANGE THE ENTERED DEFERRED COMMANDS. AFTER THE TEST HAS BEEN ENTERED AND EDITED, THE COMPILE COMMAND IS EXECUTED. IF THE COMPILE IS SUCCESSFUL (NO ERRORS REPORTED) THE OT S COMMAND IS EXECUTED TO STORE THE SOURCE AND ALL ENTERED DATA PATTERNS ON PAPER TAPE. THE RUN COMMAND IS THEN USED TO HAVE THE TEST EXECUTED. IF A CHANCE TO THE PROGRAM IS DESIRED, THE SOURCE TYPE IS READ IN, THE EDITING PERFORMED, COMPILED AGAIN, PUNCHED AGAIN, ETC. AT ANY TIME THE OBJECT CODE MAY BE SAVED BY DOING AN OT, O COMMAND TO PUNCH A TAPE WITH THAT CODE.

4.1 IMMEDIATE COMMAND SET

4.1.1 DRIVE SELECTION

DN, DRIVE NUMBER

WHERE DRIVE NUMBER IS A SINGLE DIGIT 0 THROUGH 7 TO SPECIFY THE DRIVE TESTED. THIS DRIVE NUMBER WILL BE USED UNTIL EITHER ALTERED BY ANOTHER DN COMMAND OR A DIFFERENT DRIVE NUMBER IS SPECIFIED AS PART OF A SUBSYSTEM COMMAND (SF). WHEN THE PROGRAM IS INITIALLY LOADED THE DRIVE NUMBER IS SET TO 1.

DN, ?

WILL CAUSE THE NUMBER OF THE DRIVE THAT IS PRESENTLY SELECTED TO BE TYPED.

4.1.2 OUTPUT TEST TO PAPER TAPE

OT,0 WHERE 0 IS OBJECT CODE  
OT,S WHERE S IS SOURCE CODE

THIS COMMAND IS PROVIDED TO ALLOW EITHER THE SOURCE OR THE OBJECT CODE TO BE PUNCHED. ONLY THE SOURCE CODE CAN BE PUNCHED BEFORE THE "COMPILE COMMAND HAS BEEN ISSUED, BOTH CAN BE PUNCHED AFTER THE "COMPILE" BUT BEFORE THE "RUN", AND ONLY THE OBJECT CODE CAN BE PUNCHED AFTER THE "RUN". ALL TEST SPECIFIC PARAMETERS (DRIVE

NUMBER, CYLINDER, TRACK, ETC.), THE USER DEFINED DATA PATTERNS, AND THE RANDOM DATA PATTERN ARE ALSO PUNCHED.

4.1.3 COPY TAPE

CT (COPY TAPE)

THE PAPER TAPE LOADED IN THE READER IS REPRODUCED ON THE PUNCH. IT MAY BE EITHER SOURCE OR OBJECT CODE.

4.1.4 INPUT TEST FROM PAPER TAPE.

IT (INPUT TEST)

THE NEXT TEST ON THE PAPER TAPE IS READ. THE TEST WILL BE RECOGNIZED AS SOURCE OR OBJECT CODE AND THE CODE PLACED IN THE APPROPRIATE BUFFER. CONTROL WILL BE RETURNED TO THE CONSOLE



AFTER THE TEST IS LOADED.

IS (INPUT TEST STRING)

THIS COMMAND DIFFERS FROM THE INPUT TEST COMMAND IN THAT IF THE TEST IS OBJECT CODE, THAT TEST IS IMMEDIATELY EXECUTED AND THE NEXT TEST IS READ FROM TAPE. THIS CONTINUES UNTIL A TEST OF SOURCE CODE IS READ OR ALL TESTS HAVE BEEN EXECUTED (TAPE SUPPLY EXHAUSTED). WHEN A TEST OF SOURCE CODE IS READ CONTROL IS RETURNED TO THE CONSOLE AS FOR THE INPUT TEST COMMAND.

#### 4.1.5 TIME OUT

TO,NNNNN

WHERE NNNNN IS A DECIMAL NUMBER THAT WILL VARY THE TIME DURATION OF A SUBSYSTEM TIMEOUT (SEE PARAGRAPH 4.4). THE VALUE HAS NO RELATIONSHIP TO TIME, IT IS SIMPLY THE NUMBER OF TIMES A SOFTWARE LOOP IS EXECUTED. IT IS PRESET TO 2000. EXPERIMENTING WITH VALUES IS SUGGESTED AS THE BEST PROCEDURE TO FIND THE DESIRED VALUE FOR A GIVEN PROCESSOR AND MEMORY CONFIGURATION. THE TIMEOUT VALUE IS OUTPUTED WHEN A TEST IS PUNCHED, EITHER SOURCE OR OBJECT. WHEN THAT TEST IS READ, THE ASSOCIATED TIMEOUT VALUE IS STORED. NNNNN MUST BE 32767(10) OR LESS.

TO,?

WILL CAUSE THE TIME OUT VALUE TO BE PRINTED.

#### 4.1.6 ITERATION COUNT

IC,NNNNN

WHERE NNNNN IS THE DECIMAL NUMBER OF TIMES THE NEXT TEST EXECUTED WILL BE ITERATED (32767(10) OR LESS).

IF THE TEST IS WRITTEN ONTO PAPER TAPE EITHER AS SOURCE OR OBJECT CODE, THE ITERATION COUNT IS ALSO WRITTEN. WHEN THE TEST IS LOADED FROM PAPER TAPE, THE WRITTEN ITERATION COUNT IS USED.

IC,?

WILL TYPE THE CURRENT ITERATION COUNT ON THE CONSOLE.

#### 4.1.7 SPECIAL DATA PATTERNS

DP,BUFFER NAME,DDDD----D(RETURN)  
DDDD----D(RETURN)  
(RETURN)

WHERE PATTERN NAME IS X, Y, OR Z AND WHERE DDDD----D IS THE OCTAL DATA TO BE USED AS A DATA PATTERN.

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

THE TOTAL LENGTH OF D IS 32 WORDS OR LESS. TWO CONSECUTIVE CARRIAGE RETURNS TERMINATE DATA PATTERN ENTRY AND A SINGLE CARRIAGE RETURN RETURNS THE CARRIAGE BUT IS IGNORED AS FAR AS THE DATA PATTERN IS CONCERNED. IF LESS THAN 32 WORDS ARE ENTERED THE REMAINDER OF THE WORDS ARE ZERO FILLED.

EACH LINE MUST BE LIMITED TO 8 WORDS (48 ASCII CHARACTERS) PER LINE OR LESS AT WHICH TIME A CARRIAGE RETURN MUST BE TYPED. ALTHOUGH 6 ASCII CHARACTERS ARE REQUIRED TO FILL A SINGLE WORD, A CARRIAGE RETURN AT SOMETHING OTHER THAN MODULE 6 CAUSES LEFT JUSTIFYING OF THE PARTIAL WORD GIVEN AND USING IT AS A FULL WORD.

THE PATTERN NAME (X, Y, OR Z) MAY BE SPECIFIED IN ANY COMMAND INVOLVING PATTERN SELECTION TO SELECT THE SPECIAL PATTERN. IF THE NAMED SPECIAL PATTERN HAS NOT BEEN DEFINED PRIOR TO ITS USE, A PATTERN OF ALL ZEROS WILL BE SUPPLIED.

THE SPECIAL DATA PATTERNS THAT HAVE BEEN DEFINED WILL BE PUNCHED WITH THE TEST. THESE DATA PATTERNS ARE RETRIEVED WHEN THE TEST IS LOADED.

#### 4.1.8 EDIT BUFFER

EB,BUFFER NAME,WORD POSITION,DDDD---D(RETURN)  
DDDD---D(RETURN)  
(RETURN)

WHERE BUFFER NAME IS X, Y, OR Z; WHERE WORD POSITION IS THE FIRST WORD THAT IS TO BE EDITED; AND WHERE DDDD---D IS THE DATA TO BE ENTERED. WORD POSITION IS AN OCTAL NUMBER AND THE FIRST WORD IN THE BUFFER IS SPECIFIED AS 0.

THE ENTRY PROCEDURE IS THE SAME AS FOR THE DP COMMAND. THE SIGNIFICANT DIFFERENCE IS THAT THE BUFFER IS NOT CLEARED AND ANY WORDS NOT CHANGED BY THE EB COMMAND ARE UNCHANGED. AS BEFORE, PARTIAL WORDS ARE LEFT JUSTIFIED AND ZERO FILLED.

#### 4.1.9 BUFFER DUMP

BD,BUFFER NAME, NUMBER OF WORDS

WHERE BUFFER NAME CAN BE SPECIAL BUFFER X, Y, OR Z, THE READ (R) OR WRITE (W) BUFFER, OR THE HEADER (H) BUFFER (SEE SUBSYSTEM COMMAND READ ALL HEADERS). THE PRINTOUT STARTS WITH WORD 0 AND PRINTS THE NUMBER OF WORDS SPECIFIED (OCTAL).

#### 4.1.10 COMPILE

CO,NC,BII

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

THIS COMMAND CAUSES THE STORED DEFERRED COMMANDS TO BE COMPILED INTO A TEST SEQUENCE. THE OPTIONAL PARAMETER (NC) SPECIFIES IF THE TEST IS TO BE RUN IN A CHECK OR NO-CHECK MODE. IF THE PARAMETER IS OMITTED THE TEST WILL BE COMPILED TO BE RUN WITH NORMAL CHECKING. IF THE PARAMETER IS GIVEN THE TEST WILL BE COMPILED FOR NO-CHECK EXECUTION.

NO-CHECK PERTAINS ONLY TO ALL SUBSYSTEM COMMANDS (SEE DESCRIPTION OF THE SUBSYSTEM FUNCTION COMMAND AND THE COMMANDS LISTED IN APPENDIX B.2) WITH THE EXCEPTION OF THE READ ALL HEADER. THIS COMMAND CANNOT BE EXECUTED IN NO-CHECK MODE.

THE OPTIONAL PARAMETER (I) SPECIFIES THAT ALL DATA TRANSFER OPERATIONS IN THE TEST ARE TO BE EXECUTED WITH "BUS ADDRESS INCREMENT INHIBIT". SPECIFYING BI CAUSES THE PROGRAM TO SUSPEND THE INTERNAL PROGRAM CHECK THAT LOOKS FOR WORD COUNTS THAT ARE GREATER THAN THE BUFFER SIZE I.E. THE PROGRAM WILL ACCEPT A DATA TRANSFER WORD COUNT OF ANY SIZE.

IT SHOULD BE NOTED THAT AFTER THE COMPILE THE SOURCE CODE IS STILL VALID AND CAN BE EDITED OR SAVED.

#### 4.1.11 RUN

RU

THIS COMMAND CAUSES THE OBJECT CODE TO BE EXECUTED. A RUN COMMAND GIVEN BEFORE A TEST IS COMPILED IS REJECTED WITH AN ERROR MESSAGE. EXECUTING A RUN COMMAND CAUSES THE SOURCE CODE TO BE LOST.

#### 4.1.12 EDIT ADD LINE

EA, LN, NEW COMMAND

WHERE LN IS THE NUMBER (DECIMAL) THAT SPECIFIES THE POSITION OF THE LINE INSERTION AND WHERE NEW COMMAND IS THE COMMAND TO BE INSERTED INTO THE SOURCE.

AFTER THE COMMAND IS EXECUTED THE NEW COMMAND WILL HAVE THE LINE NUMBER LN AND THE LINE NUMBERS FROM THE ENTRY POINT (INCLUDING THE LINE THAT WAS AT THE ENTRY POINT) TO THE END OF THE TEST WILL BE INCREMENTED.

#### 4.1.13 EDIT DELETE LINE

ED, LN

WHERE LN IS THE NUMBER IN DECIMAL OF THE LINE TO BE DELETED.



4.1.14 PRINT TEST

PT

THIS COMMAND WILL CAUSE THE STORED SOURCE TO BE PRINTED ON THE  
TELETYPE. LINE NUMBERS (DECIMAL) WILL BE PRINTED AT THE  
BEGINNING OF EACH LINE.

4.1.15 PRINT LINE

PL, LN

WHERE LN IS THE DECIMAL NUMBER OF THE LINE THAT IS TO BE PRINTED.

4.1.16 NEW TEST

NT

THIS COMMAND CAUSES THE PROGRAM TO INITIALIZE ITSELF BY CLEARING  
THE SOURCE COMMANDS, CLEARING THE COMPILED TEST, AND TERMINATING  
ANY TEST PRESENTLY EXECUTING. ALL STORED PARAMETERS (DRIVE  
NUMBER, ITERATION COUNT, STALL, CYLINDER NUMBER, TRACK, SECTOR,  
ETC.) ARE NOT CHANGED.

4.1.17 PRINT REGISTER

PR, REGISTER SELECT

WHERE REGISTER SELECT IS THE UNIBUS ADDRESSABLE REGISTER,  
SPECIFIED AS AN OCTAL NUMBER OR A MNEMONIC NAME. THE  
POSSIBILITIES ARE:

NUMBER	NAME	DESCRIPTION
00	CS1	COMMAND STATUS REG 1
01	WC	WORD COUNT
02	BA	BUS ADDRESS
03	DA	DESIRED ADDRESS - TRACK & SECTOR
04	CS2	COMMAND STATUS REG 2
05	DS	DRIVE STATUS
06	ER	ERROR REGISTER
07	ASOF	ATTENTION SUMMARY & OFFSET
10	DC	DESIRED CYLINDER
11	UNUSED	
12	DB	DATA BUFFER
13	MR1	MAINTENACE REGISTER 1

14	MR2	MAINTENANCE REGISTER 2
15	MR3	MAINTENANCE REGISTER 3
16	POS	ECC/POSITION
17	PAT	ECC/PATTERN

4.1.18 HELP

HP

THIS COMMAND WILL CAUSE A SUMMARY OF THE INTERACTIVE COMMANDS TO BE PRINTED. IT IS VALID ONLY AFTER THE PROGRAM IS FIRST LOADED.

4.1.19 FORMAT SELECT

FT,NN

WHERE NN SPECIFIES THE FORMAT AS AN OCTAL NUMBER (24 FOR 20 SECTOR/TRACK AND 26 FOR 22 SECTOR/TRACK). THE FORMAT DEFAULTS TO 26. THE SELECTED FORMAT APPLIES TO ALL COMMANDS AND WILL REMAIN AS SET UNTIL CHANGED. THIS VALUE IS ALSO OUTPUTTED TO PAPER TAPE AS PART OF THE TEST. CONSEQUENTLY, INPUTTING A TEST CAN ALSO CHANGE THE FORMAT SELECTED.

FT,?

WILL CAUSE THE FORMAT VALUE TO BE PRINTED.

4.1.20 DRIVE TYPE SELECTION

DT,DRIVE TYPE

WHERE DRIVE TYPE IS A SINGLE DIGIT 6 OR 7 TO SPECIFY RK06 OR RK07 RESP.

THIS DRIVE TYPE WILL BE USED UNTIL ALTERED BY ANOTHER DT COMMAND. WHEN THE PROGRAM IS INITIALLY LOADED, THE DRIVE TYPE IS SET TO 6 FOR RK06 OPERATION.

DT,?

WILL CAUSE THE DRIVE TYPE THAT IS PRESENTLY SELECTED TO BE TYPED.

4.2 DEFERRED COMMAND SET

\*\*\*\* IMPORTANT \*\*\*

THE DRIVE TYPE CANNOT BE SPECIFIED OR ALTERED BY

ANY OF THE DEFERRED COMMANDS.  
IT MUST BE SPECIFIED OR ALTERED ONLY BY THE DT  
COMMAND IN THE IMMEDIATE MODE. SEE 4.1.20

#### 4.2.1 SUBSYSTEM FUNCTION COMMAND

SF, SUBSYSTEM COMMAND, DRIVE NUMBER, CCC, T, SS,  
NUMBER OF WORDS, DATA PATTERN

WHERE ALL NUMERIC VALUES ARE OCTAL AND:

- \* SF IS SUBSYSTEM FUNCTION COMMAND.
- \* SUBSYSTEM COMMAND IS ONE OF THE FOLLOWING:

RD	READ DATA
WD	WRITE DATA
WC	WRITE CHECK
WH	WRITE HEADER
RH	READ HEADER
SK	SEEK
CC	CONTROLLER CLEAR
CS	CLEAR SUBSYSTEM
DC	DRIVE CLEAR
RC	RECALIBRATE
DS	DRIVE SELECT
PA	PACK ACKNOWLEDGE
UL	UNLOAD
SS	START SPINDLE
OF	OFFSET
AH	READ ALL HEADER

- \* DRIVE NUMBER IS THE NUMBER OF THE DRIVE TO BE ADDRESSED.  
IF UNSPECIFIED THE LAST DRIVE ADDRESSED WILL BE USED.
- \* CCC IS THE CYLINDER ADDRESS OR THE OFFSET VALUE IF THE  
SUBSYSTEM COMMAND IS OFFSET.
- \* T IS THE TRACK ADDRESS
- \* SS IS THE SECTOR ADDRESS
- \* NUMBER OF WORDS IS THE NUMBER OF WORDS TO BE

TRANSFERRED. THE NUMBER IS CHECKED AGAINST THE MAXIMUM  
ALLOWED CONSISTANT WITH THE BUFFER SIZE. IF THE NUMBER  
IS GREATER THAN THE MAXIMUM AN ERROR IS REPORTED WHEN  
THE TEST IS COMPILED UNLESS THE COMPILE COMMAND IS GIVEN  
WITH THE BII PARAMETER (SEE COMPILE COMMAND  
DESCRIPTION).

- \* DATA PATTERN IS AN ALPHABETIC CHARACTER TO SELECT THE  
DESIRED DATA PATTERN FROM THE VARIOUS PATTERNS AVAILABLE

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



(SEE APPENDIX A). THE CHARACTERS X, Y, OR Z ARE VALID TO USE A USER DEFINED PATTERN. IF THIS PATTERN HAS NOT BEEN DEFINED, DATA OF ALL ZEROS WILL BE USED.

ANY PARAMETER MAY BE OMITTED AND ALLOWED TO DEFAULT TO THE LAST VALUE GIVEN FOR THAT PARAMETER. A CARRIAGE RETURN ANYWHERE IN THE COMMAND ALLOWS THE REMAINING PARAMETERS TO DEFAULT. SEPARATORS ( ) MUST BE SUPPLIED IF A PARAMETER IS OMITTED AND FOLLOWING PARAMETERS ARE GIVEN.

OMITTING THE DRIVE NUMBER PARAMETER IS USEFUL TO HAVE A GENERAL PURPOSE TEST THAT CAN BE RUN ON ANY DRIVE ADDRESS. THE IMMEDIATE COMMAND DN CAN BE USED TO SPECIFY THE DESIRED DRIVE BEFORE THE GENERAL PURPOSE TEST IS EXECUTED. NOTE THAT THIS IS NOT APPLICABLE IF MORE THAN ONE DRIVE IS TO BE USED IN THE GENERAL PURPOSE TEST.

OMITTING THE DATA PATTERN PARAMETER WILL CAUSE THIS COMMAND TO USE THE BUFFER AS IT WAS LAST INITIALIZED. IF THE PARAMETER IS GIVEN THE OUTPUT BUFFER IS INITIALIZED AS PART OF THE TEST. THIS IS ESPECIALLY IMPORTANT WHEN EXECUTION SPEED SHOULD BE FAST FOR SCOPING PURPOSES. THE BUFFER INITIALIZE COMMAND CAN BE ENTERED AND EXECUTED AS A SEPARATE TEST TO AVOID BUFFER LOADING IN A TEST WHERE SPEED IS REQUIRED.

THE NO-CHECK CAPABILITY OF THE COMPILE COMMAND APPLIES TO THE SUBSYSTEM COMMANDS THAT ARE LISTED ABOVE (WITH THE EXCEPTION OF THE READ SPECIFIC HEADER). WHEN THE NO-CHECK MODE OF OPERATION IS INVOKED THE CHECKING FUNCTIONS THAT DETECT THE OCCURRENCE OF OPERATION ERRORS OR FAILURES IN THE CONTROLLER OR DRIVE ARE INHIBITED. THE SUBSYSTEM COMMANDS ARE EXECUTED REGARDLESS OF ERROR CONDITIONS AT THE START OF THE COMMAND OR ERROR OCCURRENCE DURING THE COMMAND. THE ONLY REQUIREMENT FOR THE TEST TO PROCEED TO THE NEXT COMMAND IS THE CONTROLLER MUST INDICATE COMMAND COMPLETION BY SETTING "READY". THE IMPLICATION IS THAT WHEN THE NO-CHECK MODE IS USED THE TEST PROGRAM IS RESPONSIBLE FOR TESTING FOR ERRORS AND CLEARING ERROR CONDITIONS.

THE WRITE HEADER COMMAND IS IMPLEMENTED SUCH THAT THE CYLINDER AND TRACK PARAMETERS SPECIFY THE PHYSICAL LOCATION (CYLINDER & TRACK) THAT IS TO BE FORMATTED. THE SECTOR PARAMETER, IF SPECIFIED AS ZERO, CAUSED THE CORRECT HEADER FOR THAT PHYSICAL LOCATION TO BE GENERATED IN THE OUTPUT BUFFER AND WRITTEN. IF THE SECTOR PARAMETER IS NON-ZERO THE CONTENTS OF THE SPECIAL DATA PATTERN BUFFER X, Y, AND Z ARE USED AND WRITTEN AS THE HEADERS ON THAT CYLINDER AND TRACK. THE WRITE HEADER COMMAND

DOES NOT ALTER THE CONTENTS OF BUFFER X, Y, OR Z. THE CONTENTS OF THESE BUFFERS MUST BE SPECIFIED USING THE SPECIAL DATA PATTERN (DP) COMMAND TO LOAD ALL OF BUFFER X, ALL OF BUFFER Y, AND THE FIRST 2 WORDS OF BUFFER Z (66 WORDS REQUIRED FOR HEADERS).

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

4.2.2 BUFFER INITIALIZE

BI,ALPHABETIC CHARACTER

WHERE THE ALPHABETIC CHARACTER SPECIFIES THE DATA PATTERN TO BE USED (SEE APPENDIX A FOR THE AVAILABLE PATTERNS). THE OUTPUT BUFFER WILL BE INITIALIZED TO THE PATTERN SELECTED. CHARACTERS X, Y OR Z ARE VALID TO SELECT THE USER DEFINED DATA PATTERN. IF THE SPECIAL DATA PATTERN HAS NOT BEEN USER DEFINED BEFORE IT IS SELECTED A PATTERN OF ALL ZEROS WILL BE USED.

4.2.3 DATA COMPARE

DC,NNNNNN

WHERE NNNNNN IS A OCTAL VALUE SPECIFYING THE NUMBER OF WORDS TO BE COMPARED STARTING AT THE BEGINNING OF THE OUTPUT AND INPUT BUFFERS. IF NNNNNN IS OMITTED THE NUMBER OF WORDS IN THE LAST INPUT DATA TRANSFER ARE COMPARED.

A DATA MISCOMPARE WILL CAUSE THE GOOD AND BAD DATA TO BE REPORTED IN THE ERROR REPORT.

4.2.4 STATUS COMPARE

SC,STATUS WORD NUMBER,EXPECTED VALUE,MASK

WHERE ENTERED VALUES ARE OCTAL AND:

- \* STATUS WORD NUMBER IS THE DRIVE STATUS WORD TO BE COMPARED. STATUS WORDS ARE DESIGNATED 0 THROUGH 7 AND ARE ARBITRARILY ASSIGNED AS FOLLOWS:

00	IS	MESSAGE	LINE	A	WORD	0
01	IS	MESSAGE	LINE	B	WORD	0
02	IS	MESSAGE	LINE	A	WORD	1
03	IS	MESSAGE	LINE	B	WORD	1
04	IS	MESSAGE	LINE	A	WORD	2
05	IS	MESSAGE	LINE	B	WORD	2
06	IS	MESSAGE	LINE	A	WORD	3
07	IS	MESSAGE	LINE	B	WORD	3

- \* EXPECTED VALUE IS THE VALUE THE STATUS SPECIFIED SHOULD BE.

- \* MASK SPECIFIES WHICH BITS ARE TO BE COMPARED IN THE STATUS WORD READ AND THE EXPECTED VALUE. A ONE IN A SPECIFIC BIT POSITION ALLOWS THAT BIT POSITION COMPARISON TO OCCUR AND A ZERO INHIBITS COMPARISON. MASK IS INITIALLY SET TO 077777 WHEN THE PROGRAM IS

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

LOADED. (BIT 15 OF DRIVE STATUS WORDS IS ALWAYS ZERO.)  
ONCE MASK HAS BEEN SPECIFIED, THAT VALUE IS STORED AND  
USED FOR SUBSEQUENT SC COMMANDS UNTILL IT IS CHANGED BY  
A SUBSEQUENT SC COMMAND THAT INCLUDES MASK  
SPECIFICATION.

A STATUS MISCOMPARE WILL CAUSE THE GOOD AND BAD STATUS WORDS TO  
BE REPORTED.

#### 4.2.5 REGISTER COMPARE

RC, REGISTER NUMBER, EXPECTED VALUE, MASK

WHERE ENTERED VALUES ARE OCTAL AND:

- \* REGISTER NUMBER IS THE NUMBER OF THE UNIBUS ADDRESSABLE  
REGISTER OF THE RK611 TO BE COMPARED.
- \* EXPECTED VALUE IS THE VALUE THE SPECIFIED REGISTER  
SHOULD CONTAIN.
- \* MASK SPECIFIES WHICH BITS ARE TO BE COMPARED IN THE  
REGISTER READ AND THE EXPECTED VALUE. A ONE IN A  
SPECIFIC BIT POSITION ALLOWS THAT BIT POSITION  
COMPARISON TO OCCUR AND A ZERO INHIBITS THAT COMPARISON.  
MASK IS INITIALLY SET TO 177777 WHEN THE PROGRAM IS  
LOADED. ONCE THE MASK HAS BEEN SPECIFIED, THAT VALUE IS  
STORED AND USED IN SUBSEQUENT RC COMMANDS UNTIL THE MASK  
IS CHANGED BY A SUBSEQUENT RC COMMAND THAT INCLUDES A  
MASK SPECIFICATION.

REGISTER MISCOMPARE WILL CAUSE GOOD AND BAD VALUES TO BE  
REPORTED.

#### 4.2.6 REGISTER WRITE

RW, REGISTER SELECT, VALUE

WHERE ENTERED VALUES ARE OCTAL AND:

- \* REGISTER SELECT IS THE NUMBER OR THE MNEMONIC NAME OF  
THE UNIBUS ADDRESSABLE REGISTER OF THE RK611 TO BE  
WRITTEN. (SEE PARAGRAPH 4.1.14 FOR LIST OF NUMBERS AND  
NAMES.)
- \* VALUE IS THE VALUE TO BE LOADED.

ANY REGISTER AND ANY VALUE MAY BE SPECIFIED. NO CHECK IS MADE  
FOR READ ONLY BITS.

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  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1020  
1021  
1022  
1023  
1024

#### 4.2.7 STALL

ST,NNNNN

WHERE NNNNN IS A DECIMAL NUMBER SPECIFYING A CONSTANT TO DELAY BETWEEN SUBSYSTEM FUNCTION COMMANDS. NNNNN MUST BE 32767(10) OR LESS.

#### 4.2.8 PRINT MESSAGE

PM,MESSAGE

WHERE MESSAGE CAN BE ANY ASCII STRING UP TO 70 CHARACTERS IN LENGTH. THAT MESSAGE IS PRINTED ON THE CONSOLE TERMINAL WHEN THE PM COMMAND IS EXECUTED. THE MESSAGE IS PRINTED ONLY DURING THE FIRST EXECUTION OF THE TEST AFTER A RUN COMMAND IS EXECUTED AND SUPPRESSED DURING SUBSEQUENT TEST ITERATIONS IF THE ITERATION COUNT IS GREATER THAN 1, IF LOOP ON TEST (SW14 SET) OR LOOP ON ERROR (SW9 SET AND ERROR).

#### 4.2.9 UNIBUS INITIALIZE

UI

THIS COMMAND WILL CAUSE A RESET TO BE EXECUTED TO CLEAR ALL UNITS CONNECTED TO THE UNIBUS.

#### 4.3 LINE NUMBERING

AS DEFERRED COMMANDS ARE ENTERED AND STORED, THE PROGRAM ASSIGNS DECIMAL LINE NUMBERS TO THE COMMANDS SEQUENTIALLY. THE LINE NUMBERS ARE USED IN THE EDIT COMMANDS (EA AND ED) AND FOR LINE PRINTING (PL).

WHEN A LINE IS ADDED OR DELETED FROM THE SOURCE, THE STORED LINES ARE RENUMBERED IMMEDIATELY. SUBSEQUENT LINE ORIENTED COMMANDS MUST TAKE THE NEW LINE NUMBERS INTO CONSIDERATION.

#### 4.4 TIMEOUT

TO PREVENT "SILENT DEATH" SITUATIONS (THE PROGRAM STARTS AN OPERATION ON THE RK06 SUBSYSTEM AND THE SUBSYSTEM NEVER SIGNALS COMPLETION) A SOFTWARE TIMER IS EMPLOYED. EACH TIME THE PROGRAM

STARTS AN OPERATION ON THE BK06 SUBSYSTEM. THE TIMER IS USED TO INSURE THAT THE REQUESTED ACTIVITY COMPLETES WITHIN A REASONABLE PERIOD OF TIME. IF THE ACTIVITY DOES NOT COMPLETE, THE PROGRAM WILL DISPLAY A SUBSYSTEM TIMEOUT MESSAGE.

THE REASONABLE PERIOD OF TIME JUST MENTIONED IS NOT CALIBRATED TO REAL TIME. CALIBRATION IS NOT POSSIBLE BECAUSE OF VARIOUS CONFIGURATIONS OF MEMORIES AND PROCESSORS.

TO ENHANCE THE USEFULNESS OF THIS TIMEOUT FEATURE, THE TIMER IS VARIABLE. (SEE TIMEOUT COMMAND DESCRIPTION). THE DEFAULT VALUE OF THE VARIABLE IS LARGE ENOUGH TO INSURE COMMAND COMPLETION.

#### 4.5 TEST LOOPING AND LOOP COUNTERS

USING THE SWITCH OPTIONS PROVIDED, THE PROGRAM WILL LOOP ON THE TEST (SWITCH 14) OR LOOP ON ERROR (SWITCH 9). WHENEVER A TEST IS BEING LOOPEO, EACH LOOP IS COUNTED.

THE LOOP COUNT IS REPORTED IN TWO INSTANCES. THESE ARE WHEN AN ERROR OCCURS AND IS REPORTED (SWITCH 13 RESET) AND WHEN LOOPING IS TERMINATED.

#### 5.0 ERROR REPORTING FORMATS

TWO BASIC REPORT FORMATS ARE DEFINED. FORMAT 1 IS FOR ALL ERRORS (EITHER PROGRAM OR HARDWARE DETECTED) WHERE COMMAND PARAMETERS AND RK611 REGISTER CONTENTS ARE APPLICABLE. FORMAT 2 IS FOR COMPARISON ERROR REPORTING, I.E., STATUS COMPARE, REGISTER COMPARE, AND DATA COMPARE.

##### 5.1 FORMAT 1

FORMAT 1 HAS THE FOLLOWING ENTRIES:

ERROR MESSAGE

XXX = CMND LINE NUM

DRIVE=

CMND=

CURRENT OPERATIONS:

PARAMETERS GIVEN:

CYLNRD SECTOR TRACK OFFSET BAH BAL WDC

APPLICABLE REGISTERS:

CS1	CS2	WC	BA	DA	DC	ASOF
ER	DS	AO	BO			
A1	B1	A2	B2	A3	B3	ECC/POS ECC/PAT

PREVIOUS OPERATION:

DRIVE=

CMND=

PARAMETERS GIVEN:

CYLINDER	SECTOR	TRACK	OFFSET	BAH	BAL	WDC
----------	--------	-------	--------	-----	-----	-----

XXXXXXXXXX = NUMBER OF LOOPS

ALL THE ENTRIES LISTED ABOVE WILL NOT APPEAR IN EVERY REPORT. ENTRIES THAT ARE NOT PERTINENT TO THE OPERATION ARE OMITTED. FOR EXAMPLE, THE PARAMETERS GIVEN ENTRIES ARE NOT APPLICABLE TO A PACK ACKNOWLEDGE OPERATION SO ALL THESE ENTRIES ARE OMITTED IF PA IS THE FAILING COMMAND.

THE NUMBER OF LOOPS ENTRY IS PRINTED ONLY IF THE TEST IS RUNNING WITH LOOP ON ERROR OR LOOP ON TEST SET. WITH THE EXCEPTION OF THE ERROR MESSAGE ENTRY, THE ENTRIES LISTED ABOVE ARE SELF EXPLANATORY. ALL ERROR MESSAGES ARE LISTED BELOW.

5.1.1 SUBSYSTEM DETECTED ERROR

THIS MESSAGE IS PRINTED WHENEVER THE PROGRAM IS ALERTED THAT THE SUBSYSTEM HAS DETECTED AN ERROR. THIS INCLUDES ALL THE ERRORS DETECTED IN THE CONTROLLER OR DRIVE.

5.1.2 UNSOLICITED ATTENTION

THIS MESSAGE INDICATES AN INTERRUPT WAS RECEIVED FROM A DRIVE BUT NO OPERATION HAS BEEN STARTED ON THAT DRIVE. THIS MESSAGE WILL BE SEEN IF WRITE LOCK IS CHANGED OR A DRIVE IS STARTED MANUALLY. THE MESSAGE IS NOT PRINTED WHEN THE CHANGE OCCURS IF THE PROGRAM IS AT COMMAND LEVEL (INTERRUPTS ARE LOCKED OUT) BUT IS PRINTED AS SOON AS CARRIAGE RETURN IS TYPED ON THE CONSOLE.

5.1.3 UNEXPECTED DATA TYPE ERROR

THIS MESSAGE INDICATES AN INTERRUPT OCCURRED THAT WAS CAUSED BY

1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136



DATA ERROR TYPE (DCK, OPI, HUNC, OR WCE) WHEN NO COMMAND OR A  
NON-DATA TRANSFER COMMAND WAS BEING EXECUTED.

5.1.4 ATTENTION DID NOT RESET WITH DRIVE CLEAR

THIS MESSAGE INDICATES A DRIVE CLEAR COMMAND WAS NOT ABLE TO  
RESET THE DRIVE ATTENTION SIGNAL. THIS IS A CATASTROPHIC ERROR  
FOR THE PROGRAM. THE HIGH ATTENTION SIGNAL WILL CAUSE CONTINUOUS  
INTERRUPTS.

5.1.5 ATTENTION DID NOT RESET WITH SUBSYSTEM CLEAR

THIS MESSAGE INDICATES AN ERROR OF THE SAME TYPE AS "ATTENTION  
DID NO RESET WITH DRIVE CLEAR". THE DIFFERENCE IS THAT THE  
SUBSYSTEM CLEAR GENERATES RESET TO ALL DRIVES.

5.1.6 ILLEGAL DRIVER COMMAND

THIS MESSAGE IS AN INDICATION OF AN INTERNAL PROGRAM INTERLAU  
PROBLEM. IT SHOULD NEVER APPEAR. IF IT DOES, PLEASE NOTIFY  
DIAGNOSTIC ENGINEERING.

5.1.7 SUBSYSTEM TIMEOUT

THIS MESSAGE INDICATES THAT THE SUBSYSTEM FAILED TO SEND AN  
INTERRUPT WITHIN A REASONABLE PERIOD OF TIME. "REASONABLE" IS  
SUFFICIENTLY LONG SO THAT THE INTERRUPT SHOULD HAVE OCCURRED.

5.1.8 CLEAR CONTROLLER DID NOT CLEAR ERROR

THIS MESSAGE INDICATES THAT THE CONTROLLER ERROR WAS NOT RESET  
WHEN A CONTROLLER CLEAR WAS DONE. THIS HAS THE SAME IMPLICATION  
AS "DRIVE HARD ERROR" MESSAGE BUT AT THE CONTROLLER LEVEL.

5.1.9 NO ATTENTION IN ATTENTION SUMMARY REGISTER

THIS MESSAGE INDICATES AN INTERRUPT WAS RECEIVED FROM THE  
CONTROLLER BUT NO DRIVE HAS RAISED ATTENTION.

5.1.10 DATA LATE WHEN UNLOADING HEADER

THIS MESSAGE IS PERTINENT TO THE READ ALL HEADER COMMAND AND

1137  
1138  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1190  
1191  
1192

INDICATES PROBLEM REACHING THE DATA BUFFER.

5.1.11 CONTROLLER ERROR WHILE DRIVER SERVICING

THIS MESSAGE INDICATES A CONTROLLER ERROR OCCURRED WHILE THE PROGRAM WAS DOING SERVICE TYPE OPERATIONS. THESE SERVICE OPERATIONS ARE OF THE "DRIVE SELECT" OR "DRIVE CLEAR" NATURE AND ARE PERFORMED WHEN THE PROGRAM IS GATHERING STATUS FOR REPORTING, CLEARING ERRORS, ETC.

5.1.12 DRIVE PARITY WHILE GATHERING S JS

THIS MESSAGE INDICATES THAT THE DRIVE HAS DETECTED A SERCON PARITY ERROR WHILE THE PROGRAM WAS DOING THE SERVICE OPERATION DESCRIBED ABOVE.

5.1.13 MULTIPLE DRIVE SELECT

THIS MESSAGE IS SELF-EXPLANATORY AND WILL APPEAR WITH A SUBSYSTEM DETECTED ERROR MESSAGE.

5.2 FORMAT 2

THREE ERRORS ARE REPORTED USING FORMAT 2. THESE ARE REGISTER COMPARE ERROR, STATUS COMPARE ERROR, AND DATA COMPARE ERROR.

THE REGISTER AND STATUS COMPARE ERROR REPORT FORMAT IS:

XXX=CMND LINE NUM

ERROR MESSAGE (STATUS OR REGISTER COMPARE ERROR)

NN=REGISTER NUMBER OF STATUS WORD NUMBER

(VALUE EXPECTED)=GOOD DATA

(VALUE RECEIVED)=BAD DATA

(SPECIFIED MASK)=NUMBER OF LOOPS

THE DATA COMPARE ERROR REPORT FORMAT IS:

XXX=CMND LINE NUMBER

DATA COMPARE ERR ON WORD NNNN  
(DATA)=GOOD DATA

1193  
1194  
1195  
1196  
1197  
1198  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248

(DATA)=BAD DATA  
XXXX=TOTAL MISCOMPARES  
XXXXXXXXXX=NUMBER OF LOOPS

APPENDIX A

THE FOLLOWING DATA PATTERNS HAVE BEEN DEFINED. ADDITIONAL  
PATTERNS WILL BE INCLUDED WHEN THEY BECOME KNOWN.

PATTERN "A"	PATTERN "B"	PATTERN "C"	PATTERN "D"
-----	-----	-----	-----
177777	000000	125252	052525
000000	177777	125252	052525
.	.	.	.
.	.	.	.
(32 WORDS)	(32 WORDS)	(32 WORDS)	(32 WORDS)
.	.	.	.
.	.	.	.
000000	177777	125252	052525
177777	000000	125252	052525
PATTERN "E"	PATTERN "F"	PATTERN "G"	
-----	-----	-----	
000001	177777	177776	
000003	177776	177775	
000007	177774	177773	
000017	177770	177767	
000037	177760	177757	
000077	177740	177737	
000177	177700	177677	
000377	177600	177577	
000777	177400	177377	
001777	177000	176777	
003777	176000	175777	
007777	174000	173777	
017777	170000	167777	
037777	160000	157777	
077777	140000	137777	
177777	100000	077777	

1305	077777	000000	137777
1306	037777	100000	157777
1307	017777	140000	167777
1308	007777	160000	173777
1309	003777	170000	175777
1310	001777	174000	176777
1311	000777	176000	177377
1312	000377	177000	177577
1313	000177	177400	177677
1314	000077	177600	177737
1315	000037	177700	177757
1316	000017	177740	177767
1317	000007	177760	177773
1318	000003	177770	177775
1319	000001	177774	177776
1320	000000	177776	177777

	PATTERN "H"	PATTERN "I"
	-----	-----
1321	000001	155555
1322	000002	155555
1323	000004	.
1324	000010	.
1325	000020	.
1326	000040	.
1327	000100	(32 WORDS)
1328	000200	.
1329	000400	.
1330	001000	155555
1331	002000	155555
1332	004000	
1333	010000	
1334	020000	
1335	040000	
1336	100000	
1337	100000	
1338	040000	
1339	020000	
1340	010000	
1341	004000	
1342	002000	
1343	001000	
1344	000400	
1345	000200	
1346	000100	
1347	000040	
1348	000020	
1349	000010	
1350	000004	
1351	000002	
1352	000001	
1353		
1354		
1355		
1356		
1357		
1358		
1359		
1360		

## APPENDIX B

## COMMAND SUMMARIES

## B.1 USER DEFINED COMMAND SET

## B.1.1 IMMEDIATE COMMANDS

ALL DECIMAL VALUES MUST BE LESS THAN 32767(10).

COMMANDS -----	MNEMONIC -----	PARAMETERS -----
DRIVE SELECTION	DN DN	,DRIVE NUMBER ,?
OUTPUT TEST	OT	,0 ,S
INPUT TEST	IT	NONE
INPUT STRING	IS	NONE
ITERATION COUNT	IC IC	,NNNN ,?
SPECIAL DATA BUFFER	DP	,PATTERN NAME ,DDDD....D
COMPILE	CO	,NO CHECK ,INCREMENT INHIBIT
EDIT ADD LINE	EA	,LN ,NEW COMMAND
EDIT DELETE LINE	ED	,LN
EDIT BUFFER	EB	,BUFFER NAME ,WORD POSITION ,DDDD--D
BUFFER DUMP	BD	,BUFFER NAME ,NUMBER OF WORDS
PRINT TEST	PT	NONE
PRINT LINE	PL	,LN
NEW TEST	NT	NONE

RUN	RU	NONE
PRINT REGISTER	PR	,REGISTER NUMBER
HELP	HP	NONE
TIME OUT CHANGE	TO	,CONSTANT
	TO	,?
COPY TAPE	CT	,NONE
FORMAT SELECT	FT	,FORMAT
		,?
DRIVE TYPE SELECT	DT	,DRIVE TYPE
		,?

## B.1.2 DEFERRED COMMANDS

ALL DECIMAL VALUES MUST BE LESS THAN 32767(1).

COMMANDS	MNEMONIC	PARAMETERS
-----	-----	-----
SUBSYSTEM FUNCTION	SF	,SUBSYSTEM COMMAND ,DRIVE NUMBER ,CCC ,T ,SS ,NUMBER OF WORDS ,DATA PATTERN
BUFFER INITIALIZE	BI	,PATTERN SELECT
DATA COMPARE	DC	NONE
	DC	,NNNNN
STATUS COMPARE	SC	,STATUS WORD SELECT ,EXPECTED VALUE ,MASK
REGISTER COMPARE NUMBER	RC	,REGISTER NAME OR ,EXPECTED VALUE ,MASK
REGISTER WRITE NUMBER	RW	,REGISTER NAME OR ,VALUE



1473	STALL	ST	,NNNNN
1474			
1475	PRINT MESSAGE	PM	,MESSAGE
1476			
1477	UNIBUS INITIALIZE	UI	NONE
1478			
1479			
1480			

## B.2 SUBSYSTEM COMMANDS

	COMMAND	MNEMONIC
	-----	-----
1481	READ DATA	RD
1482	WRITE DATA	WD
1483	WRITE CHECK	WC
1484	WRITE HEADER & DATA	WH
1485	READ HEADER	RH
1486	SEEK	SK
1487	CLEAR SUBSYSTEM	CS
1488	CONTROLLER CLEAR	CC
1489	DRIVE CLEAR	DC
1490	RECALIBRATE	RC
1491	DRIVE SELECT	DS
1492	PACK ACKNOWLEDGE	PA
1493	UNLOAD	UL
1494	START SPINDLE	SS
1495	OFFSET	OF
1496	READ ALL HEADER	AH
1497		
1498		
1499		
1500		
1501		
1502		
1503		

```
1504  
1505  
1506  
1507  
1508  
1509  
1510 167000  
1511  
1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1520  
1521 000001  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539 001100  
1540  
1541  
1542  
1543  
1544 000011  
1545 000012  
1546 000015  
1547 000200  
1548 177776  
1549  
1550 177774  
1551 177772  
1552 177570  
1553 177570  
1554  
1555  
1556 000000  
1557 000001  
1558 000002  
1559 000003
```

```
*** REV 005 ***  
.NLIST MC,MD,CND  
.LIST ME  
.ENABL ABS,AMA  
.DEFINE SYSMAC MACROS  
$SWR= 167000 :DEFINE SWITCHES  
.TITLE CZR6RCO RK611/06 USR DEFINED  
;*COPYRIGHT (C) 1976,1977  
;*DIGITAL EQUIPMENT CORP.  
;*MAYNARD, MASS. 01754  
;*PROGRAM BY MARV TEGROTENHUIS  
;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC  
;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.  
;*  
$TN=1  
.SBTTL OPERATIONAL SWITCH SETTINGS  
;*  
SWITCH USE  
-----  
15 HALT ON ERROR  
14 LOOP ON TEST  
13 INHIBIT ERROR TYPEOUTS  
11 INHIBIT ITERATIONS  
10 BELL ON ERROR  
9 LOOP ON ERROR  
2 INHIBIT MISCOMPARE PRINTING  
1 REPORT ALL DATA MISCOMPARES  
0 SHORT REPORT FORMAT  
;*  
.SBTTL BASIC DEFINITIONS  
;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***  
STACK= 1100  
.EQUIV EMT,ERROR ;;BASIC DEFINITION OF ERROR CALL  
.EQUIV IOT,SCOPE ;;BASIC DEFINITION OF SCOPE CALL  
;*MISCELLANEOUS DEFINITIONS  
HT= 11 ;;CODE FOR HORIZONTAL TAB  
LF= 12 ;;CODE FOR LINE FEED  
CR= 15 ;;CODE FOR CARRIAGE RETURN  
CRLF= 200 ;;CODE FOR CARRIAGE RETURN-LINE FEED  
PS= 177776 ;;PROCESSOR STATUS WORD  
.EQUIV PS,PSW  
STKLMT= 177774 ;;STACK LIMIT REGISTER  
PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER  
DSWR= 177570 ;;HARDWARE SWITCH REGISTER  
DDISP= 177570 ;;HARDWARE DISPLAY REGISTER  
;*GENERAL PURPOSE REGISTER DEFINITIONS  
R0= %0 ;;GENERAL REGISTER  
R1= %1 ;;GENERAL REGISTER  
R2= %2 ;;GENERAL REGISTER  
R3= %3 ;;GENERAL REGISTER
```

1560	000004	R4=	%4	:: GENERAL REGISTER
1561	000005	R5=	%5	:: GENERAL REGISTER
1562	000006	R6=	%6	:: GENERAL REGISTER
1563	000007	R7=	%7	:: GENERAL REGISTER
1564	000006	SP=	%6	:: STACK POINTER
1565	000007	PC=	%7	:: PROGRAM COUNTER
1566		.*PRIORITY LEVEL DEFINITIONS		
1567		PR0=	0	:: PRIORITY LEVEL 0
1568	000000	PR1=	40	:: PRIORITY LEVEL 1
1569	000040	PR2=	100	:: PRIORITY LEVEL 2
1570	000100	PR3=	140	:: PRIORITY LEVEL 3
1571	000140	PR4=	200	:: PRIORITY LEVEL 4
1572	000200	PR5=	240	:: PRIORITY LEVEL 5
1573	000240	PR6=	300	:: PRIORITY LEVEL 6
1574	000300	PR7=	340	:: PRIORITY LEVEL 7
1575	000340			
1576		.*"SWITCH REGISTER" SWITCH DEFINITIONS		
1577		SW15=	100000	
1578	100000	SW14=	40000	
1579	040000	SW13=	20000	
1580	020000	SW12=	10000	
1581	010000	SW11=	4000	
1582	004000	SW10=	2000	
1583	002000	SW09=	1000	
1584	001000	SW08=	400	
1585	000400	SW07=	200	
1586	000200	SW06=	100	
1587	000100	SW05=	40	
1588	000040	SW04=	20	
1589	000020	SW03=	10	
1590	000010	SW02=	4	
1591	000004	SW01=	2	
1592	000002	SW00=	1	
1593	000001	.EQUIV	SW09, SW9	
1594		.EQUIV	SW08, SW8	
1595		.EQUIV	SW07, SW7	
1596		.EQUIV	SW06, SW6	
1597		.EQUIV	SW05, SW5	
1598		.EQUIV	SW04, SW4	
1599		.EQUIV	SW03, SW3	
1600		.EQUIV	SW02, SW2	
1601		.EQUIV	SW01, SW1	
1602		.EQUIV	SW00, SW0	
1603		.*DATA BIT DEFINITIONS (BIT00 TO BIT15)		
1604		BIT15=	100000	
1605	100000	BIT14=	40000	
1606	040000	BIT13=	20000	
1607	020000	BIT12=	10000	
1608	010000	BIT11=	4000	
1609	004000	BIT10=	2000	
1610	002000	BIT09=	1000	
1611	001000	BIT08=	400	
1612	000400	BIT07=	200	
1613	000200	BIT06=	100	
1614	000100			
1615				

1616 000040  
1617 000020  
1618 000010  
1619 000004  
1620 000002  
1621 000001

BIT05= 40  
BIT04= 20  
BIT03= 10  
BIT02= 4  
BIT01= 2  
BIT00= 1  
.EQUIV BIT09,BIT9  
.EQUIV BIT08,BIT8  
.EQUIV BIT07,BIT7  
.EQUIV BIT06,BIT6  
.EQUIV BIT05,BIT5  
.EQUIV BIT04,BIT4  
.EQUIV BIT03,BIT3  
.EQUIV BIT02,BIT2  
.EQUIV BIT01,BIT1  
.EQUIV BIT00,BIT0

1633  
1634 000004  
1635 000010  
1636 000014  
1637 000014  
1638 000014  
1639 000020  
1640 000024  
1641 000030  
1642 000034  
1643 000060  
1644 000064  
1645 000240

.\*BASIC "CPU" TRAP VECTOR ADDRESSES  
ERRVEC= 4 ; TIME OUT AND OTHER ERRORS  
RESVEC= 10 ; RESERVED AND ILLEGAL INSTRUCTIONS  
TBITVEC= 14 ; "T" BIT  
TRTVEC= 14 ; TRACE TRAP  
BPTVEC= 14 ; BREAKPOINT TRAP (BPT)  
IOTVEC= 20 ; INPUT/OUTPUT TRAP (IOT) \*\*SCOPE\*\*  
PWRVEC= 24 ; POWER FAIL  
EMTVEC= 30 ; EMULATOR TRAP (EMT) \*\*ERROR\*\*  
TRAPVEC= 34 ; "TRAP" TRAP  
TKVEC= 60 ; TTY KEYBOARD VECTOR  
TPVEC= 64 ; TTY PRINTER VECTOR  
PIRQVEC= 240 ; PROGRAM INTERRUPT REQUEST VECTOR  
.SBTTL TRAP CATCHER

1646  
1647  
1648 000000

. = 0  
.\*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"  
.\*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS  
.\*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS

1652 000174 000174  
1653 000174 000000  
1654 000176 000000  
1655 000200 000200  
1656 000200 000137 004116  
1657  
1658

. = 174  
DISPREG: .WORD 0 ; SOFTWARE DISPLAY REGISTER  
SWREG: .WORD 0 ; SOFTWARE SWITCH REGISTER  
. = 200  
JMP 2#UDTSRT

1659  
1660  
1661  
1662  
1663  
1664  
1665 001100  
1666 001100 000000  
1667 001100 000  
1668 001102 000  
1669 001103 000  
1670 001104 000000  
1671 001106 000000  
1672 001110 000000  
1673 001112 000000  
1674 001114 000  
1675 001115 001  
1676 001116 000000  
1677 001120 000000  
1678 001122 000000  
1679 001124 000000  
1680 001126 000000  
1681 001130 000000  
1682 001132 000000  
1683 001134 000  
1684 001135 000  
1685 001136 000000  
1686 001140 177570  
1687 001142 177570  
1688 001144 177560  
1689 001146 177562  
1690 001150 177564  
1691 001152 177566  
1692 001154 000  
1693 001155 002  
1694 001156 012  
1695 001157 000  
1696 001160 000000  
1697 001162 000000  
1698 001164 177607 000377  
1699 001170 077  
1700 001171 015  
1701 001172 000012  
1702  
1703  
1704 001174 000000  
1705 001176 000001  
1706 001200 000000  
1707 001202 000000  
1708 001204 000000  
1709 001206 000400  
1710 001210 000000  
1711 001212 000026  
1712 001214 000000  
1713 001216 000000  
1714 001220 000000

## .SBTTL COMMON TAGS

\*\*\*\*\*  
; THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS  
; \*USED IN THE PROGRAM.

. =1100

SCMTAG: .WORD 0  
\$PASS: .WORD 0  
\$TSTNM: .BYTE 0  
\$ERFLG: .BYTE 0  
\$ICNT: .WORD 0  
\$LPADR: .WORD 0  
\$LPERR: .WORD 0  
\$ERTTL: .WORD 0  
\$ITEMB: .BYTE 0  
\$ERMAX: .BYTE 1  
\$ERPPC: .WORD 0  
\$GDWOR: .WORD 0  
\$BODR: .WORD 0  
\$GDDAT: .WORD 0  
\$BDDAT: .WORD 0  
\$AUTOB: .BYTE 0  
\$INTAG: .BYTE 0  
\$SWR: .WORD DSWR  
\$DISPLAY: .WORD DDISP  
\$TKS: 177560  
\$TKB: 177562  
\$TPS: 177564  
\$TPB: 177566  
\$NULL: .BYTE 0  
\$FILLS: .BYTE 2  
\$FILLC: .BYTE 12  
\$TFILG: .BYTE 0  
\$TIMES: 0  
\$ESCAPE: 0  
\$BELL: .ASCIZ <207><377><377>  
\$QUES: .ASCII /?/  
\$CRLF: .ASCII <15>  
\$LF: .ASCIZ <12>

\*\*\*\*\*  
; SBTTL STORED PARAMETERS

\$PUCODE: .WORD 0 ; PUNCH CODE (OBJECT OR SOURCE)  
\$DRIVE: .WORD 1 ; LAST ADDRESSED DRIVE NUMBER  
\$CYLNUM: .WORD ; LAST GIVEN CYLINDER ADDRESS  
\$TRKNUM: .WORD ; LAST GIVEN TRACK ADDRESS  
\$SECTNUM: .WORD ; LAST GIVEN SECTOR ADDRESS  
\$WDCNT: .WORD 400 ; LAST GIVEN WORD COUNT  
\$FILLER: .WORD 0  
\$FORMAT: .WORD 26 ; LAST FORMAT SELECTED  
\$SUBCMD: .WORD ; LAST ENTERED SUBSYSTEM COMMAND  
\$STATRD: .WORD 0 ; LAST SELECTED STATUS MESSAGE  
\$STVAL: .WORD ; LAST VALUE READ FROM STATRD

START OF COMMON TAGS  
CONTAINS PASS COUNT  
CONTAINS THE TEST NUMBER  
CONTAINS ERROR FLAG  
CONTAINS SUBTEST ITERATION COUNT  
CONTAINS SCOPE LOOP ADDRESS  
CONTAINS SCOPE RETURN FOR ERRORS  
CONTAINS TOTAL ERRORS DETECTED  
CONTAINS ITEM CONTROL BYTE  
CONTAINS MAX. ERRORS PER TEST  
CONTAINS PC OF LAST ERROR INSTRUCTION  
CONTAINS ADDRESS OF 'GOOD' DATA  
CONTAINS ADDRESS OF 'BAD' DATA  
CONTAINS 'GOOD' DATA  
CONTAINS 'BAD' DATA  
RESERVED--NOT TO BE USED  
AUTOMATIC MODE INDICATOR  
INTERRUPT MODE INDICATOR  
ADDRESS OF SWITCH REGISTER  
ADDRESS OF DISPLAY REGISTER  
TTY KBD STATUS  
TTY KBD BUFFER  
TTY PRINTER STATUS REG. ADDRESS  
TTY PRINTER BUFFER REG. ADDRESS  
CONTAINS NULL CHARACTER FOR FILLS  
CONTAINS # OF FILLER CHARACTERS REQUIRED  
INSERT FILL CHARS. AFTER A "LINE FEED"  
"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)  
MAX. NUMBER OF ITERATIONS  
ESCAPE ON ERROR ADDRESS  
CODE FOR BELL  
QUESTION MARK  
CARRIAGE RETURN  
LINE FEED

## STORED PARAMETERS

1715	001222	177777	SMASK: .WORD	177777	; LAST GIVEN STATUS MASK
1716	001224	000000	REGNUM: .WORD	0	; LAST ADDRESSED RK611 REG.
1717	001226	000000	REGVAL: .WORD		; LAST VALUE ENTERED OR READ FROM REGNUM
1718	001230	177777	RMASK: .WORD	177777	; LAST GIVEN REGISTER MASK
1719	001232	000001	ITCNT: .WORD	1	; ITERATION COUNT
1720	001234	001	SFEMP: .BYTE	1	; SOURCE FILE EMPTY (NO VALID SOURCE)
1721	001235	000	VLD OBJ: .BYTE	0	; VALID OBJECT CODE
1722	001236	101	PATSEL: .BYTE	101	; LAST PATTERN SELECTED
1723	001237	000	PATXDF: .BYTE	0	; USER HAS DEFINED PAT X
1724	001240	000	PATYDF: .BYTE	0	; USER HAS DEFINED PAT Y
1725	001241	000	PATZDF: .BYTE	0	; USER HAS DEFINED PAT Z
1726	001242	000	PATROF: .BYTE	0	; USER HAS DEFINED A RANDOM PAT
1727	001243	000	LNCNT: .BYTE		; NUMBER OF LINES IN BUFFER
1728	001244	000	DRVTYP: .BYTE	0	; LAST GIVEN DRIVE TYPE, 0=RK06, 4=RK07
1729					; LOADED IN DIRTE
1730	001245	000	OPFLGS: .BYTE	0	
1731		000002	NCK=	BIT1	; BIT 1 - NO CHECK MODE SWITCH
1732		000004	BAI=	BIT2	; BIT 2 - BUS ADDRESS INCREMENT INHIBIT SWITCH
1733		000010	SECT20=	BIT3	; BIT 3 - 20 SECTOR FORMAT
1734			.EVEN		
1735	001246	000000	SFPTR: .WORD		; SOURCE FILE POINTER
1736	001250	000000	PUFLSZ: .WORD	0	; PUNCH FILE SIZE. NUM OF BYTES IN
1737					; SOURCE OR OBJECT FILE.
1738	001252	000000	STALL: .WORD		; STALL DURATION
1739	001254	000000	COMSZ: .WORD	0	; DATA COMPARE SIZE PARAMETER
1740	001256	000000	LOFFST: .WORD	0	; LAST OFFSET
1741					
1742	001260	003720	TOVAL: .WORD	102000	; TIMEOUT VALUE
1743	001262	000040	PATX: .BLKW	40	; USER DEFINED PATTERN X
1744					
1745	001362	000040	PATY: .BLKW	40	; USER DEFINED PATTERN Y
1746					
1747	001462	000040	PATZ: .BLKW	40	; USER DEFINED PATTERN Z
1748					
1749	001562	000040	PATR: .BLKW	40	; RANDOM PATTERN STORAGE
1750			.SBTTL	CONTROL	PARAMETERS
1751					
1752			.EQUIV	\$ERRPC, LINNUM	
1753	001662	000	PRINH: .BYTE	0	; PRINT INHIBIT SWITCH
1754	001663	000	CHNFLG: .BYTE	0	; CHAINING FLAG
1755	001664	000	CSERR: .BYTE	0	; COMPILE ERROR FLAG
1756	001665	000	OFFLAG: .BYTE	0	; OFFSET FLAG
1757	001666	000	SAMDR: .BYTE	0	; SAME DRIVE SWITCH
1758	001667	000	DONE: .BYTE	0	; DONE FLAG
1759	001670	000	PBSW: .BYTE	0	; PARAMETER BLOCK SELECT SWITCH
1760	001671	000	RPSWIT: .BYTE	0	; REPORT PASS SWITCH
1761	001672	377	HPVLD: .BYTE	377	; HELP VALID SWITCH
1762		001674	.EVEN		
1763	001674	001750	OBJSZ: .WORD	101000	; OBJECT FILESIZE
1764	001676	000000	PATPTR: .WORD		; POINTER TO PATTERN BUFFER
1765					
1766	001700	000400	MAXWDS: .WORD	400	; MAXIMUM WORD COUNT (SET BY PROGRAM)
1767	001702	000000	TEMP1: .WORD	0	; TEMPORARY STORAGE
1768	001704	000000	TEMP2: .WORD	0	; TEMPORARY STORAGE
1769	001706	000000	OBUFPT: .WORD	0	; OUTPUT BUFFER POINTER
1770	001710	000000	IBUFPT: .WORD	0	; INPUT BUFFER POINTER

1771 001712 000000  
1772 001714 000000  
1773 001716 000207  
1774 001720 004437  
1775 001722 000000  
1776 001724 000000  
1777 001726 177550  
1778 001730 177552  
1779 001732 177554  
1780 001734 177556  
1781 001736 000102

OFPTR: .WORD 0  
CNTSTR: .WORD 0  
RTSPC: .WORD 000207  
JSRR4: .WORD 004437  
LPCNT1: .WORD 0  
LPCNT2: .WORD 0  
PTRSR: .WORD 177550  
PTRDB: .WORD 177552  
PTPSR: .WORD 177554  
PTPDB: .WORD 177556  
HDBUFF: .BLKW 102

; OBJECT FILE POINTER  
; STORAGE FOR ITERATION COUNT  
; RETURN CONSTANT  
; JUMP CONSTANT  
; LOW ORDER LOOP COUNTER  
; HI ORDER LOOP COUNTER  
; PAPER TAPE READER STATUS REGISTER  
; PAPER TAPE READER DATA REGISTER  
; PAPER TAPE PUNCH STATUS REGISTER  
; PAPER TAPE PUNCH DATA REGISTER  
; READ ALL HEADERS BUFFER

.SBTTL RK06 CONTROLLER REGISTER DEFINITION

1786 000000  
1787 000002  
1788 000004  
1789 000006  
1790 000010  
1791 000012  
1792 000014  
1793 000016  
1794 000020  
1795 000020  
1796 000024  
1797 000026  
1798 000034  
1799 000036  
1800 000030  
1801 000030  
1802 000032  
1803 000032

RKCS1= 0  
RKWC= 2  
RKBA= 4  
RKDA= 6  
RKCS2= 10  
RKDS= 12  
RKER= 14  
RKASOF= 16  
RKDC= 20  
RKDCYL= 20  
RKDB= 24  
RKMR1= 26  
RKMR2= 34  
RKMR3= 36  
RKPOS= 30  
RKECPS= 30  
RKPAT= 32  
RKECPT= 32

; CONTROL AND STATUS REGISTER 1  
; WORD COUNT REGISTER  
; BUS ADDRESS REGISTER  
; DESIRED TRACK SECTOR REGISTER  
; CONTROL AND STATUS REGISTER 2  
; DRIVE STATUS REGISTER  
; ERROR REGISTER  
; ATTENTION SUMMARY AND OFFSET REGISTER  
; DESIRED CYLINDER REGISTER  
; DESIRED CYLINDER REGISTER  
; DATA BUFFER  
; MAINTENANCE REGISTER 1  
; MAINTENANCE REGISTER 2  
; MAINTENANCE REGISTER 3  
; ECC POSITION INFORMATION  
; ECC POSITION INFORMATION  
; ECC PATTERN INFORMATION  
; ECC PATTERN INFORMATION

.SBTTL DRIVE COMMANDS

1807 000101  
1808 000103  
1809 000105  
1810 000107  
1811 000111  
1812 000113  
1813 000115  
1814 000117  
1815 000121  
1816 000123  
1817 000125  
1818 000127  
1819 000131

SELDRV= 101  
PACK= 103  
CLEAR= 105  
UNLOAD= 107  
SRTSPL= 111  
RECAL= 113  
OFFSET= 115  
SEEK= 117  
RDODATA= 121  
WRDATA= 123  
RDHEAD= 125  
WRHEAD= 127  
WRTCHK= 131

; SELECT DRIVE  
; PACK ACKNOWLEDGE  
; DRIVE CLEAR  
; UNLOAD  
; START SPINDLE  
; RECALIBRATE  
; OFFSET  
; SEEK  
; READ DATA  
; WRITE DATA  
; READ HEADER  
; WRITE HEADER AND DATA  
; WRITE CHECK

; THE FOLLOWING ARE NOT DRIVE COMMANDS BUT ARE USED BY THE DRIVER  
; TO SIMULATE A SPECIFIC DESIRED OPERATION

1824 000140  
1825 000141  
1826 000164

RELEAS= 140  
RDSTAT= 141  
RDALHD= 164

; RELEASE DRIVE  
; GET ALL STATUS FROM DRIVE  
; READ ALL HEADERS



```

1827      000176      CONCLR= 176      ;CONTROLLER CLEAR (BIT 15 OF CS1)
1828      000177      SUBCLR= 177      ;SUBSYSTEM CLEAR (BIT 5 OF CS2)
1829      000300      INTR= 300      ;GENERATE INTERRUPT TO CPU
1830
1831      ;      DRIVER ISSUED SERVICE COMMANDS
1832
1833      000001      DR.SEL= 001      ;DRIVE SELECT
1834      000005      DR.CLR= 005      ;DRIVE CLEAR
1835
1836      .SBTTL  CONTROL AND STATUS REGISTER 1 BITS
1837
1838      000001      GO=      BIT0      ;GO BIT
1839      000100      IE=      BIT6      ;INTERRUPT ENABLE
1840      000200      RDY=      BIT7      ;CONTROLLER READY
1841      000400      BA16=      BIT8      ;BUS ADDRESS BIT 16
1842      001000      BA17=      BIT9      ;BUS ADDRESS BIT 17
1843      002000      CDT=      BIT10      ;CONTROLLER DRIVE TYPE (0=RK06,1=RK07)
1844      004000      CTO=      BIT11      ;CONTROLLER TIMED OUT WAITING FOR
1845      ;      DRIVE RESPONSE
1846      010000      CFMT=      BIT12      ;CONTROLLER DRIVE FORMAT (0=22 SECTOR, 1=20 SECTOR)
1847      020000      SPAR=      BIT13      ;DRIVE BUS PARITY ERROR DETECTED BY CONTROLLER
1848      040000      DI=      BIT14      ;DRIVE INTERRUPT
1849      100000      CERR=      BIT15      ;CONTROLLER ERROR
1850      100000      CCLR=      BIT15      ;CONTROLLER CLEAR
1851
1852      ;      THESE BIT DEFINITIONS ARE USED FOR ADDRESS
1853      ;      THE HIGH BYTE OF RKCS1
1854
1855      000001      B.BA16=      BIT0      ;BUS ADDRESS BIT 16
1856      000002      B.BA17=      BIT1      ;BUS ADDRESS BIT 17
1857      000004      B.CDT=      BIT2      ;CONTROLLER DRIVE TYPE (0=RK06,1=RK07)
1858      000020      B.CFMT=      BIT4      ;CONTROLLER DRIVE FORMAT (0=22 SECTOR, 1=20 SECTOR)
1859
1860      .SBTTL  CONTROL AND STATUS REGISTER 2 BITS
1861
1862      000007      DRVMSK= 7      ;MASK FOR DRIVE SELECTION CODE
1863      000010      DESL=      BIT3      ;DESELECT OR RELEASE DRIVE IN BITS 0-2
1864      000010      RLS=      BIT3      ;DESELECT OR RELEASE DRIVE IN BITS 0-2
1865      000020      BAI=      BIT4      ;BUS ADDRESS INCREMENT INHIBIT
1866      000040      CLR=      BIT5      ;CLEAR CONTROLLER AND ALL DRIVES
1867      000040      SCLR=      BIT5      ;CLEAR CONTROLLER AND ALL DRIVES
1868      000100      IR=      BIT6      ;INPUT READY
1869      000200      OR=      BIT7      ;OUTPUT READY
1870      000400      UFE=      BIT8      ;UNIT FIELD ERROR
1871      001000      MDS=      BIT9      ;MULTIPLE DRIVE SELECT
1872      002000      PGE=      BIT10      ;PROGRAMMING ERROR
1873      004000      NEM=      BIT11      ;NON-EXISTENT MEMORY
1874      010000      NED=      BIT12      ;NON-EXISTENT DRIVE
1875      020000      UPE=      BIT13      ;UNIBUS PARITY ERROR
1876      040000      WCE=      BIT14      ;WRITE CHECK ERROR
1877      100000      DLT=      BIT15      ;DATA LATE ERROR
1878
1879      .SBTTL  ERROR REGISTER BIT DEFINITION
1880
1881      000001      ILC=      BIT0      ;ILLEGAL FUNCTION CODE
1882      ;*ILF=      BIT0      ;ILLEGAL FUNCTION CODE

```

1883	000002	SKI=	BIT1	; SEEK INCOMPLETE
1884	000004	ILF=	BIT2	; ILLEGAL DRIVE FUNCTION
1885	000004	NXF=	BIT2	; ILLEGAL DRIVE FUNCTION
1886	000010	DRPAR=	BIT3	; DRIVE DETECTED DRIVE BUS PARITY ERROR
1887	000020	FMTE=	BIT4	; FORMAT ERROR
1888	000040	DTYPE=	BIT5	; DRIVE TYPE ERROR
1889	000100	ECH=	BIT6	; ECC HARD
1890	000200	BSE=	BIT7	; BAD SECTOR ERROR
1891	000400	HCRC=	BIT8	; HEADER CRC ERROR
1892	000400	HVRC=	BIT8	; HEADER VRC ERROR
1893	001000	COE=	BIT9	; CYLINDER ADDRESS OVERFLOW ERROR
1894	002000	IOAE=	BIT10	; INVALID DISK ADDRESS ERROR
1895	004000	WLE=	BIT11	; WRITE LOCK ERROR
1896	010000	DTE=	BIT12	; DRIVE TIMING ERROR
1897	020000	OPI=	BIT13	; OPERATION (SEARCH) INCOMPLETE
1898	040000	UNS=	BIT14	; DRIVE UNSAFE
1899	100000	DCK=	BIT15	; DATA CHECK
1900				
1901		.SBTTL	STATUS REGISTER BIT DEFINITION	
1902				
1903	000001	DRA=	BIT0	; DRIVE AVAILABLE (CONTROLLER IS SET IF THIS BIT IS RESET)
1904				
1905	000004	OFST=	BIT2	; DRIVE OFFSET
1906	000010	ACLO=	BIT3	; AC LOW
1907	000020	SPDLSS=	BIT4	; SPEED LOSS
1908	000020	DCLO=	BIT4	; DC LOW
1909	000040	OROT=	BIT5	; DRIVE OFF TRACK
1910	000100	VV=	BIT6	; VOLUME VALID
1911	000200	DRY=	BIT7	; DRIVE READY
1912	000200	DRDY=	BIT7	; DRIVE READY
1913	000400	DDT=	BIT8	; DRIVE TYPE (0=RK06, 1=RK07)
1914	004000	WRL=	BIT11	; WRITE LOCK
1915	020000	PIP=	BIT13	; POSITIONING IN PROGRESS
1916	040000	DSC=	BIT14	; DRIVE STATUS CHANGE
1917	100000	SVAL=	BIT15	; STATUS VALID
1918				
1919		.SBTTL	MAINTENANCE REGISTER 1 BIT DEFINITION	
1920				
1921	000017	MESMSK=	17	; MESSAGE MASK
1922				
1923	000020	PAT=	BIT4	; FORCE EVEN PARITY ON DRIVE BUS MESSAGE LINES
1924	000040	DMD=	BIT5	; DIAGNOSTIC MODE
1925	000100	MSP=	BIT5	; MAINTENANCE SECTOR PULSE
1926	000200	MIND=	BIT7	; MAINTENANCE INDEX
1927	000400	MCLK=	BIT8	; MAINTENANCE CLOCK
1928	001000	MERD=	BIT9	; MAINTENANCE ENCODED READ DATA
1929	002000	MEWD=	BIT10	; MAINTENANCE ENCODED WRITE DATA
1930	004000	PCA=	BIT11	; PRECOMPENSATION ADVANCE
1931	010000	PCD=	BIT12	; PRECOMPENSATION DELA
1932	020000	ECCW=	BIT13	; ECC WORD IS BEING RE OR WRITTEN
1933	040000	WRTGAT=	BIT14	; WRITE GATE
1934	100000	RDGATE=	BIT15	; READ GATE
1935				
1936		.SBTTL	DEFINITION OF DRIVE STATUS BYTE 00 MESSAGE A	
1937				
1938	000040	S.DRA=	BIT5	; DRIVE AVAILABLE

1939	000100	S.VV=	BIT6	; VOLUME VALID
1940	000200	S.CRY=	BIT7	; DRIVE READY
1941	000400	S.TYPE=	BIT8	; DRIVE TYPE
1942	001000	S.FORM=	BIT9	; DRIVE FORMAT
1943	002000	S.OFF=	BIT10	; OFFSET
1944	004000	S.WRL=	BIT11	; WRITE LOCK
1945	010000	S.SPIN=	BIT12	; SPINDLE ON
1946	020000	S.PIP=	BIT13	; POSITIONING IN PROGRESS
1947	040000	S.DSC=	BIT14	; DRIVE STATUS CHANGE
1948				
1949		.SBTTL	DEFINITION OF DRIVE STATUS BYTE 00 MESSAGE B	
1950				
1951	000040	S.ICYL=	BIT5	; ILLEGAL CYLINDER ADDRESS
1952	000100	S.ACLO=	BIT6	; AC LOW
1953	000200	S.FLT=	BIT7	; DRIVE FAULT
1954	000400	S.ILF=	BIT8	; ILLEGAL FUNCTION
1955	001000	S.PAR=	BIT9	; DRIVE DETECTED DRIVE BUS PARITY ERROR
1956	002000	S.SKI=	BIT10	; SEEK INCOMPLETE
1957	004000	S.WLE=	BIT11	; WRITE LOCK ERROR
1958	010000	S.SPLS=	BIT12	; SPEED LOSS
1959	010000	S.DCLO=	BIT12	; DC LOW
1960	020000	S.DROT=	BIT13	; DRIVE OFF TRACK
1961	040000	S.UNS=	BIT14	; DRIVE UNSAFE
1962				
1963		.SBTTL	DEFINITION OF DRIVE STATUS BYTE 01 MESSAGE A	
1964				
1965	000020	S.XDOK=	BIT4	; TRANSDUCER OK
1966	000040	S.HDH=	BIT5	; HEADS HOME
1967	000100	S.BRHM=	BIT6	; BRUSHES HOME
1968	000200	S.DOOR=	BIT7	; DOOR INTERLOCKED
1969	000400	S.CART=	BIT8	; CARTRIDGE INTERLOCK
1970	001000	S.SPOK=	BIT9	; SPEED OK
1971	002000	S.FWD=	BIT10	; FORWARD
1972	004000	S.REV=	BIT11	; REVERSE
1973	010000	S.LOAD=	BIT12	; HEADS LOADING
1974	020000	S.RTZ=	BIT13	; RETURN TO ZERO
1975	040000	S.UNLD=	BIT14	; HEADS UNLOADING
1976				
1977		.SBTTL	DEFINITION OF DRIVE STATUS BYTE 01 MESSAGE B	
1978				
1979	000020	S.SECT=	BIT4	; SECTOR ERROR
1980	000040	S.WCLK=	BIT5	; WRITE CLOCK AND NO WRITE GATE
1981	000100	S.WGAT=	BIT6	; WRITE GATE AND NO TRANSISTIONS
1982	000200	S.HDFL=	BIT7	; HEAD FAULT
1983	000400	S.MHD=	BIT8	; MULTIPLE HEAD SELECT
1984	001000	S.XERR=	BIT9	; INDEX ERROR
1985	002000	S.DIB=	BIT10	; DIBIT ERROR
1986	004000	S.PLO=	BIT11	; PLO ERROR
1987	010000	S.NMOV=	BIT12	; SEEK AND NO MOTION
1988	020000	S.LIMD=	BIT13	; LIMIT DETECT ON SEEK
1989	040000	S.BRAKE=	BIT14	; SERVO-BRAKE
1990				
1991		.SBTTL	COMMON MASKS	
1992				
1993	000007	M.DRV=	7	; DRIVE CODE
1994	100000	M.PAR=	BIT15	; PARITY

# M03

CZR6RCD RK611/06 USR DEFINED  
CZR6RC.P11 02-DEC-77 11:07

MACY11 30(1046) 02-DEC-77 11:59 PAGE 39  
COMMON MASKS

SEQ 0038

1995	000003
1996	017760
1997	017760
1998	077770
1999	000760
2000	007000

M.ID=	3
M.COIF=	17760
M.CA0D=	17760
M.SER=	77770
M.SECT=	760
M.HEAD=	7000

;BYTE ID
;CYLINDER DIFFERENCE/OFFSET
;CYLINDER ADDRESS
;DRIVE SERIAL NUMBER
;SECTOR COUNT
;HEAD DECODE

N03

.SBTTL PARAMETER BLOCK ALLOCATION

```

*****
* 1  : COMMAND : DRIVE NO.
* 3  : CYLINDER ADDRESS
* 5  : TRACK : SECTOR
* 7  : BA16-17, FORMAT, DRV TYPE : OFFSET
* 11 : BUS ADDRESS (LOW 16 BITS)
* 13 : WORD COUNT (2'S COMPLEMENT)
* 15 : PROGRAM DRIVE STATUS INFORMATION
* 17 : COMMAND AND STATUS REGISTER 1
* 21 : COMMAND AND STATUS REGISTER 2
* 23 : WORD COUNT REGISTER
* 25 : BUS ADDRESS REGISTER
* 27 : DESIRED TRACK AND SECTOR
* 31 : DESIRED CYLINDER
* 33 : ATTENTION SUMMARY AND DRIVE OFFSET
* 35 : ERROR REGISTER
* 37 : STATUS REGISTER
* 41 : MESSAGE LINE A STATUS BYTE 00
* 43 : MESSAGE LINE B STATUS BYTE 00
* 45 : MESSAGE LINE A STATUS BYTE 01
* 47 : MESSAGE LINE B STATUS BYTE 01
* 51 : MESSAGE LINE A STATUS BYTE 10
* 53 : MESSAGE LINE B STATUS BYTE 10
* 55 : MESSAGE LINE A STATUS BYTE 11
* 57 : MESSAGE LINE B STATUS BYTE 11
* 61 : ECC POSITION INFORMATION
* 63 : ECC PATTERN INFORMATION
*****

```

0  
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

.SBTTL PARAMETERS PASSED TO THE DRIVER

THE FOLLOWING DEFINITIONS ARE USED TO PASS PARAMETERS  
TO THE RK06/RK07 DRIVER

```

P.DRVN= 0 : DRIVE NUMBER
P.CMND= 1 : COMMAND
P.CYLN= 2 : CYLINDER ADDRESS
P.SECT= 4 : SECTOR
P.TRACK= 5 : TRACK
P.OFST= 6 : OFFSET
P.CSIH= 7 : RKCSI BITS 8-15
P.BAHI= 7 : BUS ADDRESS (BITS 16 AND 17)
P.BALO= 10 : BUS ADDRESS (BITS 0-15)
P.WC= 12 : WORD COUNT (2'S COMPLEMENT)
P.PRST= 14 : PROGRAM DRIVE STATUS INFORMATION

```

.SBTTL PROGRAM DEVICE STATUS REGISTER DEFINITION

```

DRVUSE= BIT0 : DRIVE IN USE
DRVPOS= BIT1 : DRIVE POSITIONING
DRVPOD= BIT2 : DRIVE POSITIONED FOR DATA TRANSFER
UEXATT= BIT3 : UNEXPECTED ATTENTION
DRVHRD= BIT4 : DRIVE HAS HARD ERROR
DRVDSO= BIT5 : DRIVE STATUS CHANGE DID NOT CLEAR

```

2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2030  
2031  
2032  
2033  
2034  
2035  
2036  
2037  
2038  
2039  
2040  
2041  
2042  
2043  
2044  
2045  
2046  
2047  
2048  
2049  
2050  
2051  
2052  
2053  
2054  
2055  
2056

000000  
000001  
000002  
000004  
000005  
000006  
000007  
000007  
000010  
000012  
000014  
  
000001  
000002  
000004  
000010  
000020  
000040

```

2057      000100      CMDT0= BIT6      ; NO TERMINATION TO COMMAND FOR AT
2058                                     ; LEAST 1 SECOND
2059      000200      W.WCK= BIT7      ; WRITE FOR WRITE CHECK
2060      000400      NOCHK= BIT8      ; NO CHECK DO NOT SET INTERRUPT ENABLE
2061      001000      PBSVAL= BIT9     ; PARAMETER STATUS WORDS VALID
2062                                     ; (SET WHEN ERROR TERMINATION OR
2063                                     ; READ STATUS COMMAND)
2064      002000      DRPDRV= BIT10     ; DROP DRIVE FROM TEST SEQUENCE
2065      004000      NODSC= BIT11     ; ATTENTION SET BUT DCS AND FAULT RESET
2066      010000      DRVSZD= BIT12     ; DRIVE SEIZED BY OTHER PORT
2067      020000      E.UNLD= BIT13     ; DRIVE UNLOADED DUE TO ERROR
2068      040000      Q.INIT= BIT14     ; PARAMETER BLOCK ENQUEUED IN INITIATION QUEUE
2069      100000      DTBAII= BIT15     ; INHIBIT BUS ADDRESS INCREMENT
2070
2071      .SBTTL  PARAMETERS PASSED FROM DRIVER TO PROGRAM
2072
2073      ;      THE FOLLOWING DEFINITIONS ARE USED FOR REGISTER RETURNS
2074      ;      FROM THE DRIVER TO THE CALLING PROGRAM
2075
2076      000016      P.CS1= 16          ; COMMAND AND STATUS REGISTER 1
2077      000020      P.CS2= 20          ; COMMAND AND STATUS REGISTER 2
2078      000022      P.WCR= 22          ; WORD COUNT REGISTER
2079      000024      P.BAR= 24          ; BUS ADDRESS REGISTER
2080      000026      P.DTS= 26          ; DESIRED TRACK SECTOR REGISTER
2081      000030      P.DCYL= 30         ; DESIRED CYLINDER REGISTER
2082      000032      P.ASOF= 32         ; ATTENTION SUMMARY/OFFSET REGISTER
2083      000034      P.ER= 34          ; ERROR REGISTER
2084      000036      P.OS= 36          ; STATUS REGISTER
2085      000040      P.A00= 40          ; MESSAGE A STATUS BYTE 00
2086      000042      P.B00= 42          ; MESSAGE B STATUS BYTE 00
2087      000044      P.A01= 44          ; MESSAGE A STATUS BYTE 01
2088      000046      P.B01= 46          ; MESSAGE B STATUS BYTE 01
2089      000050      P.A10= 50          ; MESSAGE A STATUS BYTE 10
2090      000052      P.B10= 52          ; MESSAGE B STATUS BYTE 10
2091      000054      P.A11= 54          ; MESSAGE A STATUS BYTE 11
2092      000056      P.B11= 56          ; MESSAGE B STATUS BYTE 11
2093      000060      P.EPOS= 60         ; ECC POSITION INFORMATION
2094      000062      P.EPAT= 62         ; ECC PATTERN INFORMATION
2095      000064      PRTCON=64         ; PRINT CONTROL WORD
2096
2097      .SBTTL  PARAMETER BLOCK 0 FOR DRIVE
2098
2099      002142      000      PARMO: .BYTE 0      ; DRIVE NUMBER
2100      002143      000      .BYTE 0      ; COMMAND
2101      002144      000000     .WORD 0      ; CYLINDER ADDRESS
2102      002146      000      .BYTE 0      ; SECTOR ADDRESS
2103      002147      000      .BYTE 0      ; TRACK ADDRESS
2104      002150      000      .BYTE 0      ; OFFSET VALUE
2105      002151      000      .BYTE 0      ; BUS ADDRESS (BITS 16 AND 17)
2106      002152      000000     .WORD 0      ; BUS ADDRESS (BITS 0 - 15)
2107      002154      000000     .WORD 0      ; WORD COUNT (2'S COMPLEMENT)
2108      002156      000000     .WORD 0      ; PROGRAM DRIVE STATUS INFORMATION
2109      002160      000000     .WORD 0      ; COMMAND AND STATUS REGISTER 1
2110      002162      000000     .WORD 0      ; COMMAND AND STATUS REGISTER 2
2111      002164      000000     .WORD 0      ; WORD COUNT REGISTER
2112      002166      000000     .WORD 0      ; BUS ADDRESS REGISTER

```

2113	002170	000000	.WORD	0	: DESIRED TRACK AND SECTOR REGISTER
2114	002172	000000	.WORD	00	: DESIRED CYLINDER REGISTER
2115	002174	000000	.WORD	00	: ATTENTION SUMMARY/OFFSET REGISTER
2116	002176	000000	.WORD	00	: ERROR REGISTER
2117	002200	000000	.WORD	00	: STATUS REGISTER
2118	002202	000000	.WORD	00	: MESSAGE LINE A STATUS BYTE 00
2119	002204	000000	.WORD	00	: MESSAGE LINE B STATUS BYTE 00
2120	002206	000000	.WORD	00	: MESSAGE LINE A STATUS BYTE 01
2121	002210	000000	.WORD	00	: MESSAGE LINE B STATUS BYTE 01
2122	002212	000000	.WORD	00	: MESSAGE LINE A STATUS BYTE 10
2123	002214	000000	.WORD	00	: MESSAGE LINE B STATUS BYTE 10
2124	002216	000000	.WORD	00	: MESSAGE LINE A STATUS BYTE 11
2125	002220	000000	.WORD	00	: MESSAGE LINE B STATUS BYTE 11
2126	002222	000000	.WORD	00	: ECC POSITION INFORMATION
2127	002224	000000	.WORD	00	: ECC PATTERN INFORMATION
2128	002226	000000	.WORD	0	: PRINT CONTROL WORD

.SBTTL PARAMETER BLOCK 1 FOR DRIVE

2130					
2131					
2132	002230	000	PARM1: .BYTE	0	: DRIVE NUMBER
2133	002231	000	.BYTE	00	: COMMAND
2134	002232	000000	.WORD	00	: CYLINDER ADDRESS
2135	002234	000	.BYTE	00	: SECTOR ADDRESS
2136	002235	000	.BYTE	00	: TRACK ADDRESS
2137	002236	000	.BYTE	00	: OFFSET VALUE
2138	002237	000	.BYTE	00	: BUS ADDRESS (BITS 16 AND 17)
2139	002240	000000	.WORD	00	: BUS ADDRESS (BITS 0 - 15)
2140	002242	000000	.WORD	00	: WORD COUNT (2'S COMPLEMENT)
2141	002244	000000	.WORD	00	: PROGRAM DRIVE STATUS INFORMATION
2142	002246	000000	.WORD	00	: COMMAND AND STATUS REGISTER 1
2143	002250	000000	.WORD	00	: COMMAND AND STATUS REGISTER 2
2144	002252	000000	.WORD	00	: WORD COUNT REGISTER
2145	002254	000000	.WORD	00	: BUS ADDRESS REGISTER
2146	002256	000000	.WORD	00	: DESIRED TRACK AND SECTOR REGISTER
2147	002260	000000	.WORD	00	: DESIRED CYLINDER REGISTER
2148	002262	000000	.WORD	00	: ATTENTION SUMMARY/OFFSET REGISTER
2149	002264	000000	.WORD	00	: ERROR REGISTER
2150	002266	000000	.WORD	00	: STATUS REGISTER
2151	002270	000000	.WORD	00	: MESSAGE LINE A STATUS BYTE 00
2152	002272	000000	.WORD	00	: MESSAGE LINE B STATUS BYTE 00
2153	002274	000000	.WORD	00	: MESSAGE LINE A STATUS BYTE 01
2154	002276	000000	.WORD	00	: MESSAGE LINE B STATUS BYTE 01
2155	002300	000000	.WORD	00	: MESSAGE LINE A STATUS BYTE 10
2156	002302	000000	.WORD	00	: MESSAGE LINE B STATUS BYTE 10
2157	002304	000000	.WORD	00	: MESSAGE LINE A STATUS BYTE 11
2158	002306	000000	.WORD	00	: MESSAGE LINE B STATUS BYTE 11
2159	002310	000000	.WORD	00	: ECC POSITION INFORMATION
2160	002312	000000	.WORD	00	: ECC PATTERN INFORMATION
2161	002314	000000	.WORD	0	: PRINT CONTROL WORD



004

.SBTTL ERROR POINTER TABLE

;;\*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.  
;;\*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN  
;;\*LOCATION SITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.  
;;\*NOTE1: IF SITEMB IS 0 THE ONLY PERTINENT DATA IS (SERRPC).  
;;\*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:

;;\* EM ;:POINTS TO THE ERROR MESSAGE  
;;\* DH ;:POINTS TO THE DATA HEADER  
;;\* DT ;:POINTS TO THE DATA  
;;\* DF ;:POINTS TO THE DATA FORMAT

SERRTB:  
PROGID: .ASCII <12>@\* RK611/RK06-RK07 USER DEFINED TEST \*@<15><12>

.ASCIIZ /CZR6RCO/<15><12>

HELPO: .ASCIIZ /TYPE HP TO PRINT HELP FILE/<15><12>

STAR: .ASCIIZ /\*/  
SCTOBG: .ASCIIZ /TEST PROGRAM TO BIG/<15><12>

NOROOM: .ASCIIZ /SOURCE FILE FULL/<15><12>

EQSGN: .ASCIIZ /\*/  
BADDEC: .ASCIIZ /INVLD DECIMAL/<15><12>

IVDEDT: .ASCIIZ /INVLD EDIT CMND/<15><12>

IVDAOD: .ASCIIZ /INVLD NEW CMND/<15><12>

IVDLN: .ASCIIZ /INVLD LINE NUM/<15><12>

IVDPAR: .ASCIIZ /INVLD PARAMETER/<15><12>

BADOCT: .ASCIIZ /INVLD OCTAL/<15><12>

2162				
2163				
2164				
2165				
2166				
2167				
2168				
2169				
2170				
2171				
2172				
2173				
2174				
2175				
2176	002316			
2177	002316	025012	051040	033113
2178	002324	030461	051057	030113
2179	002332	026466	045522	033460
2180	002340	052440	042523	020122
2181	002346	042504	044506	042516
2182	002354	020104	042524	052123
2183	002362	025040	005015	
2184	002368	055103	033123	041522
2185	002374	006460	000012	
2186	002400	054524	042520	044040
2187	002406	020120	047524	050040
2188	002414	044522	052116	044040
2189	002422	046105	020120	044506
2190	002430	042514	005015	000
2191	002438	052	000	
2192	002437	051124	051505	020124
2193	002444	051120	043517	040522
2194	002452	020115	047524	041040
2195	002460	043511	005015	000
2196	002465	123	052517	041522
2197	002472	020105	044506	042514
2198	002500	043040	046125	006514
2199	002506	000012		
2200	002510	000075		
2201	002512	047111	046126	020104
2202	002520	042504	044503	040515
2203	002526	006514	000012	
2204	002532	047111	046126	020104
2205	002540	042105	052111	041440
2206	002546	047115	006504	000012
2207	002554	047111	046126	020104
2208	002562	042516	020127	046503
2209	002570	042116	005015	000
2210	002575	111	053116	042114
2211	002602	046040	047111	020105
2212	002610	052516	006515	000012
2213	002616	047111	046126	020104
2214	002624	040520	040522	042515
2215	002632	042524	006522	000012
2216	002640	047111	046126	020104
2217	002646	041517	040524	006514

2218	002654	000012				
2219	002656	042510	050114	043040	HPFILE: .ASCIZ	/HELP FILE AVAILABLE ONLY AS FIRST COMMAND AFTER PROGRAM LOAD./<15><12>
2220	002664	046111	020105	053101		
2221	002672	044501	040514	046102		
2222	002700	020105	047117	054514		
2223	002706	040440	020123	044506		
2224	002714	051522	020124	047503		
2225	002722	046515	047101	020104		
2226	002730	043101	042524	020122		
2227	002736	051120	043517	040522		
2228	002744	020115	047514	042101		
2229	002752	006456	000012			
2230	002756	047516	047440	045102	IVDRUN: .ASCIZ	/NO OBJECT/<15><12>
2231	002764	041505	006524	000012		
2232	002772	047503	050115	046111	COMPOK: .ASCIZ	/COMPILE OK/<15><12>
2233	003000	020105	045517	005015		
2234	003006	000				
2235	003007	116	020117	047523	NOSRC: .ASCIZ	/NO SOURCE/<15><12>
2236	003014	051125	042503	005015		
2237	003022	000				
2238	003023	111	052116	051105	INTERR: .ASCIZ	/INTERNAL COMPILER ERR/<15><12>
2239	003030	040516	020114	047503		
2240	003036	050115	046111	051105		
2241	003044	042440	051122	005015		
2242	003052	000				
2243	003053	111	053116	042114	BADCOM: .ASCIZ	/INVLD OR UNDEF CMND/<15><12>
2244	003060	047440	020122	047125		
2245	003066	042504	020106	046503		
2246	003074	042116	005015	000		
2247	003101	127	051117	020104	IVDWCT: .ASCIZ	/WORD CT TOO BIG/<15><12>
2248	003106	052103	052040	047517		
2249	003114	041040	043511	005015		
2250	003122	000				
2251	003123	111	053116	042114	IVDNC: .ASCIZ	/INVLD CMND IN NC/<15><12>
2252	003130	041440	047115	020104		
2253	003136	047111	047040	006503		
2254	003144	000012				
2255	003146	047111	046126	020104	BADDRV: .ASCIZ	/INVLD DRV NUM/<15><12>
2256	003154	051104	020126	052516		
2257	003162	006515	000012			
2258	003166	047111	046126	020104	BADTYP: .ASCIZ	/INVLD DRV TYPE/<15><12>
2259	003174	051104	020126	054524		
2260	003202	042520	005015	000		
2261	003207	120	047125	044103	PUERR: .ASCIZ	/PUNCH ERR/<15><12>
2262	003214	042440	051122	005015		
2263	003222	000				
2264	003223	122	040505	042504	PRERR: .ASCIZ	/READER ERR/<15><12>
2265	003230	020122	051105	006522		
2266	003236	000012				
2267	003240	042012	044522	042526	STNTVD: .ASCIZ	<12>/DRIVE STATUS NOT VALID/<15><12><12>
2268	003246	051440	040524	052524		
2269	003254	020123	047516	020124		
2270	003262	040526	044514	006504		
2271	003270	005012	000			
2272	003273	075	047524	040524	TOTMSC: .ASCIZ	/=TOTAL MISCOMPARES/<15><12><12>
2273	003300	020114	044515	041523		

2274	003306	046517	040520	042522
2275	003314	006523	005012	000
2276	003321	075	040515	020130
2277	003326	047527	042122	041440
2278	003334	052517	052116	043040
2279	003342	051117	042040	052101
2280	003350	020101	051124	047101
2281	003356	043123	051105	005015
2282	003364	000		
2283	003372	111	046114	043505
2284	003379	046101	051040	043505
2285	003400	051440	046105	042047
2286	003406	005015	000	
2287	003411	075	021440	047440
2288	003416	020106	047514	050117
2289	003424	006523	005012	000
2290	003431	052	046040	047517
2291	003436	020120	047503	047125
2292	003444	042524	020122	053117
2293	003452	051105	046106	053517
2294	003460	025040	005015	000
2295	003465	127	051117	020104
2296	003472	020043	020040	020040
2297	003500	020040	020040	020040
2298	003506	047503	052116	047105
2299	003514	051524	005015	000
2300	003521	040	020040	020040
2301	003526	000040		
2302	003530	020040	000	
2303				
2304				
2305				
2306				
2307				
2308				
2309				
2310				
2311				
2312				
2313				
2314				
2315				
2316				
2317				
2318				
2319				
2320	003534	047524		
2321	003536	006062	177777	000000
2322	003544	047104		
2323	003546	006144	177777	000000
2324	003554	052104		
2325	003556	022450	177777	000000
2326	003564	052117		
2327	003566	012426	177777	000000
2328	003574	052111		
2329	003576	013026	177777	000000

IOBFSZ: .ASCIZ /=MAX WORD COUNT FOR DATA TRANSFER/&lt;15&gt;&lt;12&gt;

BADSEL: .ASCIZ /ILLEGAL REG SEL'D/&lt;15&gt;&lt;12&gt;

LPLABL: .ASCIZ /= # OF LOOPS/&lt;15&gt;&lt;12&gt;&lt;12&gt;

LCNTOF: .ASCIZ /\* LOOP COUNTER OVERFLOW \*/&lt;15&gt;&lt;12&gt;

DMPHDR: .ASCIZ /WORD # CONTENTS/&lt;15&gt;&lt;12&gt;

SPACE6: .ASCIZ / / /

SPACE2: .ASCIZ / / /

```

;*****
;SBTTL TABLE OF INTERACTIVE COMMANDS
;THIS TABLE CONTAINS ALL THE INTERACTIVE COMMANDS.
;THERE ARE 4 WORDS PER ENTRY:
;WORD 1      COMMAND MNEUMONIC
;WORD 2      IF IMMEDIATE, ADDRESS OF INTERACTIVE COMMAND PROCESSOR
;             ROUTINE FOR THIS COMMAND. IF DEFERRED, THE
;             ADDRESS OF THE COMPILATION ROUTINE FOR THIS COMMAND.
;WORD 3      IF DEFERRED, NUMBER OF PARAMETERS ASSOCIATED WITH
;             THIS COMMAND. IF IMMEDIATE, -1.
;WORD 4      IF DEFERRED, ADDRESS OF EXECUTE ROUTINE
;             FOR THIS COMMAND. IF IMMEDIATE, ALL 0'S
;THIS TABLE IS USED IN COMMAND ENTRY AND TEST
;COMPILATION
;*****

```

```

;*****
;EVEN
ICTBL: .ASCII /TO/ ;TIMEOUT
        .WORD TORTE,-1,0
        .ASCII /DN/ ;DRIVE SELECT
        .WORD DN RTE,-1,0
        .ASCII /DT/ ;DRIVE TYPE
        .WORD DTRTE,-1,0
        .ASCII /OT/ ;OUTPUT TEST
        .WORD OTRTE,-1,0
        .ASCII /IT/ ;INPUT TEST
        .WORD ITRTE,-1,0
;*****

```

2330	003604	052103			.ASCII	/CT/		;COPY TAPE
2331	003606	022154	177777	000000	.WORD	CTRTE,-1,0		
2332	003614	051511			.ASCII	/IS/		;INPUT STRING
2333	003616	013020	177777	000000	.WORD	ISRTE,-1,0		
2334	003624	041511			.ASCII	/IC/		;ITERATION COUNT
2335	003626	006254	177777	000000	.WORD	ICRTE,-1,0		
2336	003634	052106			.ASCII	/FT/		;FORMAT SELECT
2337	003636	006000	177777	000000	.WORD	FTRTE,-1,0		
2338	003644	041105			.ASCII	/EB/		;EDIT SPECIAL BUFFERS
2339	003646	006346	177777	000000	.WORD	EBRTE,-1,0		
2340	003654	050104			.ASCII	/DP/		;SPECIAL DATA PATTERN
2341	003656	006356	177777	000000	.WORD	DPRTE,-1,0		
2342	003664	047503			.ASCII	/CO/		;COMPILE
2343	003666	010310	177777	000000	.WORD	CORTE,-1,0		
2344	003674	040505			.ASCII	/EA/		;EDIT ADD LINE
2345	003676	004764	177777	000000	.WORD	EARTE,-1,0		
2346	003704	042105			.ASCII	/ED/		;EDIT DELETE LINE
2347	003706	005266	177777	000000	.WORD	EDRTE,-1,0		
2348	003714	052120			.ASCII	/PT/		;PRINT TEST
2349	003716	005700	177777	000000	.WORD	PTRTE,-1,0		
2350	003724	046120			.ASCII	/PL/		;PRINT LINE
2351	003726	005500	177777	000000	.WORD	PLRTE,-1,0		
2352	003734	052116			.ASCII	/NT/		;NEW TEST
2353	003736	007416	177777	000000	.WORD	NTRTE,-1,0		
2354	003744	052522			.ASCII	/RU/		;RUN
2355	003746	010036	177777	000000	.WORD	RURTE,-1,0		
2356	003754	051120			.ASCII	/PR/		;PRINT REGISTER
2357	003756	006742	177777	000000	.WORD	PRRTE,-1,0		
2358	003764	050110			.ASCII	/HP/		;HELP
2359	003766	007366	177777	000000	.WORD	HPRTE,-1,0		
2360	003774	042102			.ASCII	/BD/		;BUFFER DUMP
2361	003776	007530	177777	000000	.WORD	BDRTE,-1,0		
2362	004004	043123			.ASCII	/SF/		;SUBSYSTEM FUNCTION
2363	004006	011576	000010	000000	.WORD	CSSF,10,0		
2364	004014	044502			.ASCII	/BI/		;BUFFER INITIALIZE
2365	004016	011162	000001	20536	.WORD	CSBI,1,ESBI		;SPECIAL TAG FOR BUFFER INIT
2366	004024	041504			.ASCII	/DC/		;DATA COMPARE
2367	004026	011272	000001	020204	.WORD	CSDC,1,ESDATC		
2368	004034	041523			.ASCII	/SC/		;STATUS COMPARE
2369	004036	010770	000003	017610	.WORD	CSSC,3,ESSC		
2370	004044	041522			.ASCII	/RC/		;REGISTER COMPARE
2371	004046	010752	000003	020026	.WORD	CSRC,3,ESREGC		
2372	004054	053522			.ASCII	/RW/		;REGISTER WRITE
2373	004056	010734	000002	020426	.WORD	CSRW,2,ESRW		
2374	004064	052123			.ASCII	/ST/		;STALL
2375	004066	011350	000001	020446	.WORD	CSST,1,ESST		
2376	004074	046520			.ASCII	/PM/		;PRINT MESSAGE
2377	004076	010674	000001	013516	.WORD	CSPM,1,ESPM		
2378	004104	044525			.ASCII	/UI/		;UNIBUS INITIALIZE
2379	004106	011422	000000	021464	.WORD	CSUI,0,ESUI		
2380	004114	025052			.ASCII	/**/		;END OF TABLE
2381					;*****			
2382								
2383	004116				UDTSRT:			
2384					.SBTTL INITIALIZE THE COMMON TAGS			
2385					;CLEAR THE COMMON TAGS (SCMTAG) AREA			

H04

SEQ 0046

```
2386 004116 012706 001100      MOV    #SCMTAG,R6      ;; FIRST LOCATION TO BE CLEARED
2387 004122 012706 001100      CLR    (R6)+          ;; CLEAR MEMORY LOCATION
2388 004124 022706 001140      CMP    #SWR,R6      ;; DONE?
2389 004130 001374 001100      BNE    -6            ;; LOOP BACK IF NO
2390 004132 012706 001100      MOV    #STACK,SP     ;; SETUP THE STACK POINTER
2391                                ;; INITIALIZE A FEW VECTORS
2392 004136 012737 034546 000034  MOV    #STRAP,#STRAPVEC ;; TRAP VECTOR FOR TRAP CALLS
2393 004144 012737 000340 000036  MOV    #340,#STRAPVEC+2; LEVEL 7
2394 004152 012737 034370 000024  MOV    #SPWRON,#SPWRVEC ;; POWER FAILURE VECTOR
2395 004160 012737 000340 000026  MOV    #340,#SPWRVEC+2; LEVEL 7
2396                                ;; SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
2397                                ;; EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
2398 004166 013746 000004 000004  MOV    #ERRVEC, -(SP)   ;; SAVE ERROR VECTOR
2399 004172 012737 004226 000004  MOV    #645,#ERRVEC    ;; SET UP ERROR VECTOR
2400 004200 012737 177570 001140  MOV    #DSWR,SWR        ;; SETUP FOR A HARDWARE SWICH REGISTER
2401 004206 012737 177570 001142  MOV    #DISP,DISPLAY   ;; AND A HARDWARE DISPLAY REGISTER
2402 004214 022777 177777 1 4716  CMP    #-1,DSWR        ;; TRY TO REFERENCE HARDWARE SWR
2403 004222 001012 000000 000000  BNE    66$            ;; BRANCH IF NO TIMEOUT TRAP OCCURRED
2404                                ;; AND THE HARDWARE SWR IS NOT = -1
2405 004224 000403 000000 000000  BR     65$            ;; BRANCH IF NO TIMEOUT
2406 004226 012716 004234 000000 64$: MOV    #65$, (SP)      ;; SET UP FOR TRAP RETURN
2407 004232 000002 000000 000000  RTI
2408 004234 012737 000176 001140 65$: MOV    #SWREG,SWR      ;; POINT TO SOFTWARE SWR
2409 004242 012737 000174 001142  MOV    #DISPREG,DISPLAY
2410 004250 012637 000004 000000 66$: MOV    (SP)+,#ERRVEC ;; RESTORE ERROR VECTOR
2411
2412 004254 013701 023556 000000  MOV    RKVEC,R1        ;; GET VECTOR STORAGE ADDRESS
2413 004260 012721 023764 000000  MOV    #1,INTR (R1)+  ;; LOAD IT WITH INTERRUPT HNDLR ADDR
2414 004264 012711 000340 000000  MOV    #PR?,(R1)      ;; SET PSW TO PRIORITY ?
2415 004270 004737 034272 000000  JSR    PC,$SIZE
2416 004274 013700 034366 000000  MOV    $LASTAD,RO     ;; GET LAST ADDRESS VALUE
2417 004300 012737 041616 001706  MOV    #ENDLOC,OBUFFPT ;; SET OUTPUT BUFFER POINTER
2418 004306 162700 006000 000000  SUB    #6000,RO       ;; ALLOW FOR XXDP LOADER
2419 004312 162700 041616 000000  SUB    #ENDLOC,RO     ;; SUBTRACT FOR PROGRAM CODE
2420 004316 000241 000000 000000  CLC
2421 004320 006000 000000 000000  ROR    RO             ;; DIVIDE BY TWO FOR TWO BUFFERS
2422 004322 042700 000001 000000  BIC    #BIT0,RO       ;; MAKE SURE ITS EVEN
2423 004326 012737 041616 001710  MOV    #ENDLOC,IBUFFPT ;; SET THE INPUT BUFFER POINTER
2424 004334 060037 001710 000000  ADD    RO,IBUFFPT    ;; AT THE MIDDLE OF BUFFER AREA
2425 004340 000241 000000 000000  CLC
2426 004342 006000 000000 000000  ROR    RO             ;; DIVIDE BY TWO FOR MAXIMUM WORDS
2427 004344 162700 000002 000000  SUB    #2,RO          ;; MAKE IT TWO LESS
2428 004350 010037 001700 000000  MOV    RO,MAXWDS      ;; SET MAX WORD VALUE
2429 004354 004737 032370 000000  JSR    PC,$TKINT      ;; INITIALIZE KEYBOARD
2430 004360 104401 002316 000000  TYPE   $PROGID        ;; TYPE PROGRAM NAME
2431 004364 010046 000000 000000  MOV    RO,-(SP)
2432 004366 104402 000000 000000  TYPE   $MAXWDS        ;; TYPE MAX WORDS
2433 004370 104401 003321 000000  TYPE   $IOBFSZ        ;; LABEL IT
2434 004374 005046 000000 000000  CLR    -(SP)          ;; PUT NEW PS ON STACK
2435 004376 012746 004404 000000  MOV    #67$, -(SP)    ;; PUT NEW PC ON STACK
2436 004402 000002 000000 000000  RTI                ;; POP NEW PC AND PS
2437 004404
2438 004404 105737 001672 000000 67$: TSTB   HPVLD        ;; TEST IF HELP FILE VALID
2439 004410 001402 000000 000000  BEQ    COMLEV         ;; NO - DON'T PRINT HELP QUESTION
2440 004412 104401 002400 000000  TYPE   $HELPO        ;; TYPE HELP QUESTION
2441                                ;; *****
```

## .SBTTL COMMAND LEVEL ROUTINE

;\*THIS ROUTINE WAITS FOR AND ACCEPTS A COMMAND FROM THE COSOLE AND DECODES  
;\*IT TO DETERMINE IF THE COMMAND IS IMMEDIATE, DEFERRED, OR AN UNDEFINED.  
;\*IF THE COMMAND IS DEFERRED IT CALLS THE DEFERRED COMMAND PROCESSOR IT  
;\*(DCMDPR). IF THE COMMAND IS IMMEDIATE THE 2ND WORD OF THE  
;\*INTERACTIVE COMMAND TABLE (ICTBL) IS THE ADDRESS OF A SPECIAL  
;\*ROUTINE FOR THAT COMMAND.

;\* RETURN TO COMLEV IS  
;\* NORMAL - TST (R4)+  
;\* RTS R4  
;\* ERROR - MOV (R4),R4  
;\* RTS R4  
;\*\*\*\*\*

COMLEV:	TYPE	,STAR	;TYPE COMMAND LEVEL DESIGNATOR
	RDLIN		;READ COMMAND LINE
	MOV	(SP)+,R1	;MOVE ADDR OF COMMAND R1
	JSR	R4,ICDEC	;CALL INTERACTIVE COMMAND DECODE
	2\$		;ERROR RETURN, GO TO ERROR
	TST	(R2)+	;BUMP TO PARAM WORD
	TST	(R2)	;TEST PARAM WORD
	BPL	1\$	;IF DEFERRED, BRANCH. ELSE
	TST	-(R2)	;DEC TO ADDRESS WORD
	JSR	R4,2(R2)	;BR IMMEDIATE COMMAND
			;ROUTINE WHOSE ADDRESS WAS
			;PLACED IN R2 BY ICDEC.
			;ERROR RETURN
			;RETURN TO COMMAND LEVEL
1\$:	JSR	R4,DCMDPR	;JUMP TO DEFERRED CMD PROCESSOR
	3\$		;ERROR RETURN
4\$:	CLRB	HPVLD	;CLEAR HELP VALID SWITCH
	BR	COMLEV	;RETURN TO COMMAND LEVEL
2\$:	TYPE	,BADCOM	;TYPE BAD COMMAND MESSAGE
	BR	3\$	
3\$:	TYPE	,STTYIN	;TYPE LINE IN ERROR
	TYPE	,SQUES	;FOLLOWED BY QUESTION MARK
	BR	COMLEV	;RETURN TO COMMAND LEVEL

;\*\*\*\*\*  
;SBTTL DEFERRED COMMANDS INPUT PROCESSOR  
;ENTRY: JSR PC,DCMDPR  
;WITH R1 POINTING TO INPUT CMND.  
;RETURN: NORMAL TST (R4)+  
;\* RTS R4  
;\* ERROR MOV (R4),R4  
;\* RTS R4  
;\*THIS ROUTINE WILL PLACE THE DEFERRED COMMAND  
;\*INTO THE SOURCE FILE. THE SOURCE FILE EMPTY  
;\*(SFEMP) FLAG IS CHECKED AND IF SET THE SOURCE  
;\*FILE IS CLEARED AND COUNTERS & POINTER INITIALIZED.  
;\*A LINE NUMBER IS PREFIXED TO THE DEFERRED  
;\*COMMAND AND STORED WITH THE COMMAND. THE

```
2498 ;*LINE TERMINATOR (NULL) IS KEPT WITH THE COMMAND.
2499 ;*1000 WORDS ARE ALLOCATED FOR SOURCE FILE AND
2500 ;*IS NOT ALLOWED TO OVERFLOW. WHEN THIS ROUTINE IS
2501 ;*CALLED R1 MUST BE POINTING TO THE INPUT DEFERRED
2502 ;*COMMAND.
2503 ;*****
2504
2505 004510 010346 DCM DPR: MOV R3, -(SP) ;STORE R3
2506 004512 010546 MOV R5, -(SP) ;STORE R5
2507 004514 105737 001234 TSTB SFEMP ;TEST SOURCE FILE EMPTY
2508 004520 001420 BEQ 2$ ;BR IF NOT EMPTY, APPEND TO SOURCE
2509 004522 105037 001243 CLR R3 ;CLEAR LINE COUNTER
2510 004526 013703 001700 MOV MAXWDS, R3 ;SET BUFFERSIZE
2511 004532 013737 001710 001246 MOV IBUFPT, SFPTR ;SETUP SOURCE FILE POINTER
2512 004540 013705 001710 MOV IBUFPT, R5 ;SET UP TO CLEAR SOURCE FILE.
2513 004544 162705 000002 SUB #2, R5 ;START CLEAR 1 WORD EARLY
2514 004550 005025 1$: CLR (R5)+ ;CLEAR TO ZEROS
2515 004552 005303 DEC R3 ;AND
2516 004554 001375 BNE 1$ ;LOOP UNTIL INPUT BUFFER CLEARED
2517 004556 105037 001234 CLR R3 ;CLEAR SOURCE FILE EMPTY
2518
2519 004562 013705 001246 2$: MOV SFPTR, R5 ;SET UP R5 AS SOURCE FILE PTR
2520 004566 013737 001700 001702 MOV MAXWDS, TEMP1 ;STORE SF SIZE
2521 004574 162737 000016 001702 SUB #16, TEMP1 ;ALLOW FOR THIS LINE
2522 004602 063737 001710 001702 ADD IBUFPT, TEMP1 ;COMPUTE LAST ADDR OF SF
2523 004610 020537 001702 CMP R5, TEMP1 ;TEST FOR SUFFICIENT ROOM FOR
2524 004614 101406 BLOS 3$ ;THIS LINE. BR IF YES
2525 004616 1044C1 002465 TYPE NOROOM ;TYPE NO ROOM MESSAGE
2526 004622 012605 MOV (SP)+, R5 ;RESTORE R5
2527 004624 012603 MOV (SP)+, R3 ;RESTORE R3
2528 004626 011404 MOV (R4), R4 ;SET UP ERROR RETURN
2529 004630 000414 BR 5$ ;GO TO EXIT
2530 004632 105237 001243 3$: INCB LNCNT ;BUMP LINE COUNT
2531 004636 113725 001243 MOV R3, LNCNT ;PUT LINE COUNT IN SOURCE
2532 004642 112115 4$: MOV R3, (R1)+, (R5) ;MOVE INPUT CPT TO SOURCE
2533 004644 105725 TSTB (R5)+ ;TEST IF LAST CHAR MOVED IS NULL
2534 004646 001375 BNE 4$ ;BR IF NOT YET NULL
2535 004650 010537 001246 MOV R5, SFPTR ;STORE OFF NEW SOURCE FILE PTR
2536 004654 012605 MOV (SP)+, R5 ;RESTORE R5
2537 004656 012603 MOV (SP)+, R3 ;RESTORE R3
2538 004660 005724 TST (R4)+ ;SET UP NORMAL RETURN
2539 004662 000204 5$: RTS R4 ;RETURN TO CALLER
2540
2541 ;*****
2542 ;SBTTL SEARCH BYTE STRING FOR COMMA OR NULL
2543 ;*ENTRY
2544 ; JSR PC, SBSCN
2545 ; R1 POINTS TO FIRST CHAR OF STRING
2546 ;
2547 ;*RETURN
2548 ; RTS PC
2549 ; WITH R1 NOW POINTING TO FIRST CHARACTER
2550 ; AFTER A COMMA OR TO THE NULL
2551 ;
2552 ;*****
2553 004664 000 NULL: .BYTE 0
```

2554 004665 054  
2555 004666  
2556 004666 121137 004664  
2557 004672 001403  
2558 004674 122137 004665  
2559 004700 001372  
2560 004702 000207  
2561  
2562  
2563  
2564  
2565  
2566  
2567  
2568  
2569  
2570  
2571  
2572  
2573  
2574  
2575  
2576  
2577  
2578  
2579  
2580  
2581  
2582  
2583  
2584  
2585  
2586 004704 010346  
2587 004706 112137 001702  
2588 004712 111137 001703  
2589 004716 013703 001702  
2590 004722 005301  
2591 004724 012702 003534  
2592 004730 020312  
2593 004732 001410  
2594 004734 121237 002435  
2595 004740 001403  
2596 004742 062702 000010  
2597 004746 000770  
2598 004750 011404  
2599 004752 000402  
2600 004754 005724  
2601 004756 005722  
2602  
2603 004760 012603  
2604 004762 000204  
2605  
2606  
2607  
2608  
2609

```

COMMA: .ASCII /,/
SBSCN:
1$:  CMPB  (R1),NULL      ;TEST FOR NULL
      BEQ   2$             ;BR IF YES
      CMPB  (R1)+,COMMA    ;TEST OF COMMA
      BNE   1$             ;LOOP IF NOT
2$:  RTS    PC             ;RETURN

;*****
;SBTTL INTERACTIVE COMMAND DECODE ROUTINE
;ENTRY:
;      JSR    R4,ICDEC
;      WITH R1 CONTAINING THE ADDRESS OF THE COMMAND LINE
;RETURN:
;      MOV    (R4),R4
;      RTS    R4           ERROR RETURN
;      TST    (R4)+
;      RTS    R4           NORMAL RETURN
;
;WHEN RETURNED R2 WILL POINT TO THE SECOND WORD OF
;THE MATCHING TABLE ENTRY.
;
;THIS ROUTINE SEARCHES THE TABLE OF INTERACTIVE
;COMMANDS, LOOKING FOR A MATCH FOR THE COMMAND
;POINTED TO BY R1. IF NO MATCH THE ROUTINE RETURNS
;BACK TO RETURN 1. IF MATCH OCCURS RETURN IS TO RETURN 2.
;R2 POINTS TO SECOND WORD OF TABLE ENTRY WHICH IS
;ADDRESS OF INTERACTIVE COMMAND PROCESSOR
;SUBROUTINE. THE CALLING ROUTINE MUST
;STORE R2 BEFORE CALLING ICDEC$
;*****
ICDEC:  MOV    R3,-(SP)      ;STORE R3
        MOVB   (R1)+,TEMP1  ;MOVE COMMAND INTO R3
        MOVB   (R1),TEMP1+1 ;TO INSURE WORD
        MOV    TEMP1,R3     ;ALIGNMENT FOR EASY OF COMPARE
        DEC    R1           ;RESTORE R1 TO BEGINNING OF CMD
        MOV    #ICTBL,R2    ;ADDRESS OF TABLE INTO R2
1$:     CMP     R3,(R2)      ;TEST TABLE ENTRY AGAINST
        BEQ    3$           ;ENTERED COMMAND. R IF HIT
        CMPB   (R2),STAR    ;TEST IF END OF TABLE.
        BEQ    2$           ;IF YES DO ERROR RETURN
        ADD    #10,R2       ;BUMP R2 TO NEXT TABLE ENTRY
        BR     1$           ;LOOP - TEST NEXT ENTRY
2$:     MOV    (R4),R4      ;SET UP ERROR RETURN
        BR     4$           ;JUMP TO EXIT
3$:     TST    (R4)+        ;SET UP FOR NO ERROR RETURN
        TST    (R2)+        ;BUMP R2 TO SECOND WORD
                               ;OF TABLE
4$:     MOV    (SP)+,R3     ;RESTORE R3
        RTS    R4          ;RETURN TO CALLER

;*****
;SBTTL EDIT ADD LINE ROUTINE
;ENTRY:
;      JSR    R4,EARTE

```



2610  
2611  
2612  
2613  
2614  
2615  
2616  
2617  
2618  
2619  
2620  
2621  
2622  
2623  
2624  
2625  
2626  
2627  
2628  
2629 004764  
2630 004764 010046  
2631 004766 010146  
2632 004770 010246  
2633 004772 010346  
2634 004774 010546  
2635 004776 013705 001246  
2636 005002 013700 001710  
2637 005006 063700 001700  
2638 005012 162700 000020  
2639 005016 020500  
2640 005020 101403  
2641 005022 104401 002465  
2642 005026 000506  
2643 005030 004737 004666  
2644 005034 121137 004664  
2645 005040 001471  
2646 005042 010146  
2647 005044 004737 031314  
2648 005050 005216  
2649 005052 012603  
2650 005054 113700 001246  
2651 005060 042700 177400  
2652 005064 020300  
2653 005066 101064  
2654 005070 004737 004666  
2655 005074 121137 004664  
2656 005100 001451  
2657 005102 004437 004704  
2658 005106 005232  
2659 005110 005722  
2660 005112 005712  
2661 005114 100446  
2662 005116 005742  
2663 005120 010146  
2664 005122 005200  
2665 005124 110037 001243  
2666 005130 010500

```
* R1 POINTS TO COMMAND LINE R2 POINTS TO 2ND WORD
* IN ICTBL
* RETURN
* RTS R4 IF ERROR
* RTS R4+2 IF NO ERROR
* ROUTINES CALLED
* ICDEC
* DECBIN
* THIS ROUTINE IS USED TO INSERT A NEW LINE INTO THE
* SOURCE FILE. THE LINES BELOW THE ENTERED COMMAND
* ARE SHIFTED DOWN, THE NEW LINE IS ENTERED, AND
* THE COMMAND LINE NUMBERS ARE RESEQUENCED.
* THE LINE ADDED IS CHECKED TO INSURE IT IS A
* DEFERRED COMMAND. IF IT IS NOT, THE COMMAND IS
* NOT ADDED AND AN ERROR IS REPORTED. THE SOURCE
* FILE LENGTH IS CHECKED FOR SUFFICIENT ROOM.

EASTE:
MOV R0, -(SP) ; PUSH R0 ON STACK
MOV R1, -(SP) ; PUSH R1 ON STACK
MOV R2, -(SP) ; PUSH R2 ON STACK
MOV R3, -(SP) ; PUSH R3 ON STACK
MOV R5, -(SP) ; PUSH R5 ON STACK
MOV SFPTR, R5 ; LOAD SOURCE FILE PTR INTO R5
MOV IBUFP, R0 ; GET START OF INPUT BUFFER
ADD MAXWDS, R0 ; ADD THE MAX WORDS
SUB #20, R0 ; ALLOW FOR THIS COMMAND
CMP R5, R0 ; CHECK IF ROOM FOR THIS CMD
BLOS 1$ ; BR IF YES
TYPE NOROOM ; TYPE NO ROOM MESSAGE
BR 25$ ; GO TO EXIT
1$: JSR PC, SBSCN ; BUMP R1 TO NEXT PARAM (LN)
CMPB (R1), NULL ; CHECK IF NULL
BEQ 21$ ; BR IF YES (ERROR)
MOV R1, -(SP) ; ADDRESS OF ASCII LINE NUMBER
JSR PC, DECBIN ; CONVERT IT TO BINARY
20$: MOV (SP)+, R3 ; STORE DECODED LINE NUMBER
MOVB LNCNT, R0 ; GET NUMBER OF LAST LINE
BIC #177400, R0 ; CLEAR ANY PROPAGATED BITS IN R0
CMP R3, R0 ; REQUESTED ADD IN PRESENT SOURCE
BHI 23$ ; BR IF NO
JSR PC, SBSCN ; BUMP R1 TO NEXT PARAM (NEW CMND)
CMPB (R1), NULL ; CHECK IF NULL
BEQ 21$ ; BRANCH IF YES (ERROR)
JSR R4, ICDEC ; DECODE & CHECK NEW COMMAND
22$: ERROR RETURN
TST (R2)+ ; BUMP R2 TO 3RD WORD OF ICTBL
TST (R2) ; TEST IF WORD PLUS
BMI 22$ ; BR IF MINUS, NOT DEFERRED CMND
TST -(R2) ; DEC BACK TO 2ND WORD
MOV R1, -(SP) ; STORE R1 FOR REFERENCE
INC R0 ; ADD 1 TO OLD LINE TOTAL
MOVB R0, LNCNT ; STORE IT OFF
MOV R5, R0 ; STORE R4 FOR REFERENCE
```

M04

```
2666  
2667  
2668  
2669  
2670 005132 005205  
2671 005134 105721  
2672 005136 001375  
2673 005140 005205  
2674 005142 010537 001246  
2675 005146 012601  
2676 005150 114045  
2677  
2678 005152 101376  
2679 005154 6003 000001  
2680  
2681 005160 001373  
2682 005162 062700 000002  
2683 005166 010005  
2684 005170 111120  
2685 005172 105721  
2686 005174 001375  
2687 005176 005203  
2688 005200 105725  
2689 005202 001376  
2690 005204 020537 001246  
2691 005210 001417  
2692 005212 110325  
2693 005214 000770  
2694 005216 104401 002512  
2695 005222 000410  
2696 005224 104401 002532  
2697 005230 000405  
2698 005232 104401 002554  
2699 005236 000402  
2700 005240 104401 002575  
2701 005244 011404  
2702 005246 000401  
2703 005250 005724  
2704 005252  
2705 005252 012605  
2706 005254 012603  
2707 005256 012602  
2708 005260 012601  
2709 005262 012600  
2710 005264 000204  
2711  
2712  
2713  
2714  
2715  
2716  
2717  
2718  
2719  
2720  
2721
```

```
ROUTINE TO DETERMINE HOW FAR TO MOVE SFPTR TO ACCOMMODATE  
NEW LINE & MOVE OLD SOURCE TO MAKE ROOM  
10$: INC R5  
TSTB (R1)+  
BNE 10$  
INC R5  
MOV R5 SFPTR  
MOV (SP)+,R1  
11$: MOVB -(R0),-(R5)  
BNE 11$  
CMPB 1(R0),R3  
BNE 11$  
ADD #2,R0  
MOV R0,R5  
12$: MOVB (R1),(R0)+  
TSTB (R1)+  
BNE 12$  
13$: INC R3  
14$: TSTB (R5)+  
BNE 14$  
CMP R5,SFPTR  
BEQ 30$  
MOVB R3,(R5)+  
BR 13$  
20$: TYPE BADDEC  
BR 25$  
21$: TYPE IVDEDT  
BR 25$  
22$: TYPE IVDADD  
BR 25$  
23$: TYPE IVDLN  
25$: MOV (R4),R4  
BR 35$  
30$: TST (R4)+  
35$:  
MOV (SP)+,R5  
MOV (SP)+,R3  
MOV (SP)+,R2  
MOV (SP)+,R1  
MOV (SP)+,R0  
RTS R4  
; SCAN INPUT COMMAND. LOOK FOR  
; NULL. ADD 1 TO R5 FOR LINE  
; NUMBER. EACH NON-NULL CHAR,  
; AND ONE FOR NULL.  
; STORE NEW SF LINE POINTER  
; RECOVER R1 (COMMAND LINE PTR)  
; MOVE LAST CHAR OF OLD SF TO  
; NEW LAST CHAR LOC.  
; IF NOT NULL, LOOP  
; TEST IF NEXT TO LAST CHAR MOVED IS  
; LINE NUMBER TO BE REPLACED  
; MOVE NOT DONE, LOOP  
; GET R0 OFF NULL PAST LINE NUM  
; STORE R0 FOR REFERENCE  
; MOVE NEW COMMAND INTO SF  
; TEST IF CHAR MOVED IS NULL  
; NOT DONE STORING CMND, LOOP  
; ADD ONE TO LINE NUMBER  
; TEST IF NULL  
; GO UNTIL NULL  
; SF RESEQUENCED, EXIT TEST  
; BR IF YES, EXIT  
; MOVE IN NEW LINE NUMBER  
; BRANCH TO NEXT LINE  
; TYPE BAD NUMBER ENTERED  
; TYPE INVALID EDIT  
; TYPE INVALID NEW COMMAND  
; TYPE INVALID LINE NUMBER  
; ERROR RETURN  
; POP STACK INTO R5  
; POP STACK INTO R3  
; POP STACK INTO R2  
; POP STACK INTO R1  
; POP STACK INTO R0  
;*****  
;SBTTL EDIT DELETE LINE ROUTINE  
;ENTRY: JSR R4,EDRTE  
; WITH R1 POINTING TO INPUT COMMAND LINE (THE EDIT  
; DELETE COMMAND) AND R2 POINTING TO THE SECOND  
; WORD OF THE TABLE (IC TBC)  
; RETURN: RTS R4 ERROR RETURN  
; RTS R4+2 NO ERROR RETURN  
;THIS ROUTINE WILL REMOVE THE COMMAND DESIGNATED BY THE
```

2722			
2723			
2724			
2725			
2726			
2727			
2728			
2729			
2730			
2731			
2732			
2733			
2734			
2735			
2736			
2737			
2738			
2739	005266		
2740	005266	010046	
2741	005270	010146	
2742	005272	010246	
2743	005274	010346	
2744	005276	010546	
2745	005300	004737	004666
2746	005304	105711	
2747	005306	001455	
2748	005310	010146	
2749	005312	004737	031314
2750	005316	005450	
2751	005320	012603	
2752	005322	113702	001243
2753	005326	042702	177400
2754	005332	120203	
2755	005334	103450	
2756	005336	013700	001710
2757	005342	121003	
2758	005344	001406	
2759	005346	020037	001246
2760	005352	101041	
2761	005354	105720	
2762	005356	001376	
2763	005360	000770	
2764			
2765			
2766			
2767			
2768			
2769			
2770	005362	010005	
2771	005364	105720	
2772	005366	001376	
2773	005370	013702	001246
2774	005374	020002	
2775	005376	001404	
2776	005400	105310	
2777	005402	112025	

```

: *LINE NUMBER FROM THE SOURCE FILE. THE REMAINING
: *COMMANDS ARE MOVED UP IN THE SF AND RENUMBERED.
: *THE FOLLOWING SEQUENCE OF OPERATIONS IS PERFORMED:
: *1. RETRIEVE LINE NUMBER FROM INPUT COMMAND
: *2. CONVERT LN FROM ASCII TO BINARY
: *3. CHECK IF LN EXISTS IN SF
: *4. DELETE COMMAND FROM SF BY MOVING REST OF
: *   SF.
: *5. DECREMENT LINE NUMBERS WHILE MOVING COMMANDS.
: *6. ADJUST AND STORE POINTERS AN NEW LINE
: *   COUNT.
: *
: *   ROUTINES CALLED
: *       DECBIN
: *       SBSCN
: *****
EDRTE:
      MOV      R0,-(SP)          ;; PUSH R0 ON STACK
      MOV      R1,-(SP)          ;; PUSH R1 ON STACK
      MOV      R2,-(SP)          ;; PUSH R2 ON STACK
      MOV      R3,-(SP)          ;; PUSH R3 ON STACK
      MOV      R5,-(SP)          ;; PUSH R5 ON STACK
      JSR      PC,SBSCN          ;; BUMP R1 TO NEXT PARAM (LN)
      TSTB     (R1)              ;; CHECK IF NULL
      BEQ      20$               ;; BR IF YES (ERROR)
      MOV      R1,-(SP)          ;; ADDRESS OF LN ON STACK
      JSR      PC,DECBIN         ;; DECODE LN TO BINARY
      21$
      MOV      (SP)+,R3           ;; STORE DECODED LINE NUMBER
      MOVB     LNCNT,R2          ;; GET NUMBER OF LAST LINE
      BIC      #177400,R2        ;; CLEAR UPPER BITS
      CMPB     R2,R3             ;; TEST IF VALID LINE NUMBER
      BLO      22$               ;; NO - GO PRINT ERROR
      MOV      IBUFPT,R0         ;; GET ADDRESS OF START OF SF
1$:    CMPB     (R0),R3           ;; TEST IF THIS IS LINE TO BE
      BEQ      3$               ;; DELETED. BR IF YES
      CMP      R0,SFPTR          ;; CHECK IF STILL IN SF
      BHI      22$               ;; BR IF NO (ERROR)
2$:    TSTB     (R0)+            ;; TEST FOR NULL
      BNE      2$               ;; BR IF NOT NULL, CHECK NEXT
      BR       1$              ;; FOUND NULL, BR TO TEST FOR LN
                                ;; LINE TO BE DELETED HAS
                                ;; BEEN FOUND. NOW FIND THE
                                ;; START OF NEXT LINE. DEC
                                ;; THAT LINE NUMBER AND
                                ;; MOVE IT TO OVERLAY OLD CMD.
3$:    MOV      R0,R5            ;; STORE R0 FOR REFERENCE
4$:    TSTB     (R0)+            ;; LOOK FOR NEXT NULL
      BNE      4$               ;; BR IF NOT, CHECK NEXT CHAR
      MOV      SFPTR,R2         ;; GET SOURCE FILE PTR
5$:    CMP      R0,R2           ;; TEST IF END OF SF
      BEQ      7$               ;; BR IF YES, END OF MOVE
      DECB     (R0)             ;; DEC THAT LN
6$:    MOVB     (R0)+,(R5)+      ;; MOVE BYTE TO OVERLAY LINE

```

```
2778 005404 001773          BEQ      5$          ; TEST IF BYTE MOVED WAS MOVE
2779                                     ; END OF THAT LINE, CHECK IF MORE
2780 005406 000775          BR        6$          ; MOVE NEXT CHAR OF THIS LINE
2781 005410 010537 001246  7$:  MOV     R5,SFPTR ; STORE NEW SF POINTER
2782 005414 105337 001243      DECB     LNCNT    ; DEC TOTAL LINE COUNT
2783 005420 001003          BNE       8$          ; SKIP IF NOT ZERO
2784 005422 152737 000001 001234 BISE     R1,SFEMP ; ELSE SET SOURCE EMPTY FLAG
2785 005430 105025          CLRB     (R5)+      ; CLEAR REST OF SOURCE FILE
2786 005432 020500          CMP      R5,R0      ; CHECK IF FINISHED
2787 005434 001375          BNE       8$          ; LOOP IF NOT DONE
2788                                     ;
2789 005436 005724          TST      (R4)+      ; SET UP NORMAL RETURN
2790 005440 000411          BR        30$        ; GO TO NORMAL EXIT
2791 005442 104401 002532  20$:  TYPE     IVDEDT ; TYPE INVALID EDIT MESSAGE
2792 005446 000405          BR        25$
2793 005450 104401 002512  21$:  TYPE     BADDEC ; TYPE BAD DECIMAL MESSAGE
2794 005454 000402          BR        25$
2795 005456 104401 002575  22$:  TYPE     IVOLN  ; TYPE INVALID LINE NUMBER MESSAGE
2796 005462 011404          25$:  MOV      (R4),R4 ; SET UP ERROR RETURN
2797 005464          30$:
2798          MOV      (SP)+,R5          ; POP STACK INTO R5
2799          MOV      (SP)+,R3          ; POP STACK INTO R3
2800          MOV      (SP)+,R2          ; POP STACK INTO R2
2801          MOV      (SP)+,R1          ; POP STACK INTO R1
2802          MOV      (SP)+,R0          ; POP STACK INTO R0
2803          RTS        R4              ; RETURN TO CALLER
2804
2805
2806
2807 ;*****
2808 ;SBTTL PRINT LINE ROUTINE
2809 ;ENTRY: JSR      R4, PTRTE ON PLRTE
2810 ;WITH R1 POINTING TO COMMAND (PRINT LINE)
2811 ;RETURN: RTS      R4      NORMAL
2812 ;      RTS      R4+2    ERROR RETURN
2813 ;THIS ROUTINE WILL PRINT A SINGLE LINE FROM THE
2814 ;SOURCE FILE. IF A LINE NUMBER IS GIVEN WITH THE PRINT
2815 ;LINE (PL) REQUEST THE ROUTINE PRINTS THE SPECIFIED
2816 ;LN. IF LN IS NOT SUPPLIED THE LAST COMMAND
2817 ;ENTERED IS PRINTED.
2818 ;
2819 ;ROUTINES CALLED
2820 ;      TYPOS
2821 ;      TYPE
2822 ;      DECBIN
2823 ;
2824 ;*****
2825
2826 005500
2827 005500 010046          PLRTE:  MOV     R0,-(SP) ; PUSH R0 ON STACK
2828 005502 010146          MOV     R1,-(SP) ; PUSH R1 ON STACK
2829 005504 010346          MOV     R3,-(SP) ; PUSH R3 ON STACK
2830 005506 010546          MOV     R5,-(SP) ; PUSH R5 ON STACK
2831 005510 105737 001234  TSTB     SFEMP    ; TEST IF ANY SOURCE CODE
2832 005514 001061          BNE      22$        ; BRANCH IF NO SOURCE
2833 005516 004737 004666  JSR      PC,SBSCN   ; BUMP R1 TO NEXT PARAM
```

2834	005522	105711		TSTB	(R1)	; TEST IF NULL
2835	005524	001424		BEQ	4\$	; NO LN PRINT LAST CMND
2836	005526	010146		MOV	R1, -(SP)	; CONVERT ASCII LINE
2837	005530	004737	031314	JSR	PC, DECBIN	; NUMBER TO BINARY
2838	005534	005644		20\$		; ERROR RETURN
2839	005536	012603		MOV	(SP)+, R3	; STORE CONVERTED LN
2840	005540	113705	001243	MOVB	LNCNT, R5	; GET STORED LINE COUNT
2841	005544	042705	177400	BIC	#177400, R5	; CLEAR UPPER BITS
2842	005550	020305		CMP	R3, R5	; TEST IF VALID LINE NUMBER
2843	005552	101037		BHI	21\$	; NO - GO PRINT ERROR
2844	005554	013700	001710	MOV	IBUFPT, R0	; GET ADDRESS OF START OF SF
2845	005556	121003		1\$:	CMPB	(R0), R3
2846	005562	001413		BEQ	3\$	; TEST IF THIS IS LINE TO BE
2847	005564	121005		CMPB	(R0), R5	; PRINTED. BR IF YES FOUND LN
2848	005566	101031		BHI	21\$	; CHECK IF STILL IN SF
2849	005570	105720		2\$:	TSTB	(R0)+
2850	005572	001376		BNE	2\$	; TEST FOR NULL
2851	005574	003771		BR	1\$	; BR IF NOT NULL, TEST NEXT CHAR
2852	005576	013700	001246	4\$:	MOV	SFPT, R0
2853	005582	124040		CMPB	-(R0), -(R0)	; GET SF POINTER
2854	005584	105740		5\$:	TSTB	-(R0)
2855	005586	001376		BNE	5\$	; DEC R0 PAST NULL
2856	005590	105720		TSTB	(R0)+	; TEST FOR NULL
2857						; BR IF NO, LOOP TO FIND NULL
2858	005612	005046		3\$:	CLR	-(SP)
2859	005614	112016		MOVB	(R0)+, (SP)	; BUMP R0 PAST NULL
2860	005616	104405		TYPE		; CLEAR NEXT STACK WORD
2861	005620	104401	002510	TYPE	EQSGN	; PUT LINE NUMBER ON STACK
2862	005624	010037	005632	MOV	R0, 6\$	; TYPE LINE NUMBER
2863	005630	104401		TYPE		; TYPE EQUAL SIGN
2864	005632	000000		6\$:	.WORD	; SET UP PRINT ADDRESS
2865	005634	104401	001171	TYPE	\$CARLF	; TYPE LINE
2866	005640	005724		TST	(R4)+	; TYPE CARRIAGE RETURN
2867	005642	000411		BR	30\$	; SET UP NO ERROR RETURN
2868	005644	104401	002512	20\$:	TYPE	BADDEC
2869	005650	000405		BR	25\$	; TYPE BAD DECIMAL MESSAGE
2870	005652	104401	002575	21\$:	TYPE	IVDLN
2871	005656	000402		BR	25\$	; TYPE INVALID LINE NUMBER
2872	005660	104401	003007	22\$:	TYPE	NOSRC
2873	005664	011404		25\$:	MOV	(R4), R4
2874	005666			30\$:		; TYPE SO SOURCE MESSAGE
2875	005666	012605		MOV	(SP)+, R5	; SET UP ERROR RETURN
2876	005670	012603		MOV	(SP)+, R3	; POP STACK INTO R5
2877	005672	012601		MOV	(SP)+, R1	; POP STACK INTO R3
2878	005674	012600		MOV	(SP)+, R0	; POP STACK INTO R1
2879	005676	000204		RTS	R4	; POP STACK INTO R0
2880						; RETURN
2881						
2882						
2883						
2884						
2885						
2886						
2887						
2888						
2889						

\*\*\*\*\*  
; SBTTL PRINT TEST ROUTINE  
; \*ENTRY: JSR R4, PTRTE  
; \*WITH R1 POINTING TO COMMAND (PRINT TEST)  
; \*RETURN: RTS R4 ERROR RETURN  
; \*RTS R4+2 NORMAL RETURN  
; \*THIS ROUTINE WILL PRINT THE ENTIRE CONTENTS  
; \*OF THE SOURCE FILE.

2890  
2891  
2892  
2893  
2894  
2895 005700  
2896 005700 010046  
2897 005702 010346  
2898 005704 105737 001234  
2899 005710 001025  
2900 005712 013700 001710  
2901 005716 013703 001246  
2902 005722 005046  
2903 005724 112016  
2904 005726 104405  
2905 005730 104401 002510  
2906 005734 010037 005742  
2907 005740 104401  
2908 005742 000000  
2909 005744 104401 001171  
2910 005750 105720  
2911 005752 001376  
2912 005754 020003  
2913 005756 001361  
2914 005760 005724  
2915 005762 000403  
2916 005764 104401 003007  
2917 005770 011404  
2918 005772  
2919 005772 012603  
2920 005774 012600  
2921 005776 0002L.  
2922  
2923  
2924  
2925  
2926  
2927  
2928  
2929  
2930  
2931  
2932  
2933  
2934  
2935  
2936  
2937 006000 004737 004666  
2938 006004 105711  
2939 006006 001417  
2940 006010 121127 000077  
2941 006014 001006  
2942 006016 013746 001212  
2943 006022 104402  
2944 006024 104401 001171  
2945 006030 000406

```

;ROUTINES CALLED
;  TYPE
;  TYPOS
;*****
PTRTE:
      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
      TSTB     SFEMP         ;;TEST IF ANY SOURCE
      BNE      20$           ;;IF NONE, BRANCH EXIT
      MOV      IBUFPT,R0     ;;GET START OF SF
      MOV      SFPTR,R3      ;;GET SOURCE FILE PTR
      CLR      -(SP)         ;;CLEAR NEXT STACK WORD
      MOV      (R0)+,(SP)    ;;PUT LINE NUM ON STACK
      TYPOS     EQSGN        ;;PRINT IT
      TYPE     EQSGN        ;;TYPE EQUAL SIGN
      MOV      R0,1$         ;;SET UP TYPE ADDRESS
      TYPE     WORD         ;;TYPE COMMAND
      1$:      TYPE     $CRLF  ;;TYPE CARRIAGE RETURN
      TSTB     (R0)+         ;;TEST FOR NULL
      BNE      2$           ;;IF NOT LOOP UNTIL NULL
      CMP      R0,R3         ;;TEST IF LAST LINE PRINTED
      BNE      3$           ;;NOT DONE, LOOP
      TST      (R4)+         ;;SET NORMAL RETURN
      BR       25$          ;;GO TO NORMAL EXIT
      20$:     TYPE     NOSRC  ;;TYPE NO SOURCE
      MOV      (R4),R4       ;;SET ERROR RETURN
      25$:     MOV      (SP)+,R3  ;;POP STACK INTO R3
      MOV      (SP)+,R0      ;;POP STACK INTO R0
      RTS      R4            ;;RETURN

;*****
;SBTTL FORMAT SELECT ROUTINE
;ENTRY:      JSR      R4,FTRTE
;            WITH R1 POINTING TO COMMAND
;RETURN:     RTS      R4      ERROR RETURN
;            RTS      R4+2    NO ERROR RETURN
;
;THIS ROUTINE WILL SELECT THE FORMAT (24 OR 26,OCTAL) THAT IS TO
;BE USED FOR THE SUBSYSTEM COMMANDS. ALL COMMANDS WILL BE EXECUTED
;WITH THIS FORMAT UNTIL IT IS CHANGED.
;*****
FTRTE:  JSR      PC,SBSCN     ;;BUMP TO NEXT PARAMETER
      TSTB     (R1)          ;;TEST IF NULL
      BEQ      2$           ;;NO CHANGE - EXIT
      CMPB     (R1),#'?      ;;TEST IF QUESTION
      BNE      1$           ;;NO - SKIP PRINT
      MOV      FORMAT,-(SP)  ;;ELSE GET FORMAT STORED
      TYPOC     $CRLF        ;;PRINT IT
      TYPE     $CRLF        ;;GO TO EXIT
      BR       2$

```

2946 006032 010146  
2947 006034 004737 031160  
2948 006040 006052  
2949 006042 012637 001212  
2950 006046 005724  
2951 006050 000403  
2952 006052 011404  
2953 006054 104401 002640  
2954 006060 000204  
2955  
2956  
2957  
2958  
2959  
2960  
2961  
2962  
2963  
2964  
2965  
2966 006062 004737 004666  
2967 006066 105711  
2968 006070 001417  
2969 006072 121127 000077  
2970 006076 001006  
2971 006100 013746 001260  
2972 006104 104405  
2973 006106 104401 001171  
2974 006112 000406  
2975 006114 010146  
2976 006116 004737 031314  
2977 006122 006134  
2978 006124 012637 001260  
2979 006130 005724  
2980 006132 000403  
2981 006134 011404  
2982 006136 104401 002512  
2983 006142 000204  
2984  
2985  
2986  
2987  
2988  
2989  
2990  
2991  
2992  
2993  
2994  
2995  
2996  
2997  
2998  
2999  
3000  
3001

```
1$:  MOV    R1,-(SP)      ;PUT VALUE FOR CONVERSION ON STACK
      JSR    PC,OCTBIN
      20$
      MOV    (SP)+,FORMAT ;STORE NEW FORMAT
      TST    (R4)+        ;GOOD RETURN
      BR     30$
20$:  MOV    (R4),R4      ;SET ERROR RETURN
      TYPE   BADOCT       ;PRINT ERROR MESSAGE
30$:  RTS     R4          ;RETURN

;*****
;SBTTL TIMEOUT CHANGE ROUTINE
;ENTRY:  JSR     R4,TORTE
;        WITH R1 POINTING TO COMMAND
;
;RETURN  RTS     R4      ERROR RETURN
;        RTS     R4+2    NO ERROR RETURN
;
;THIS ROUTINE WILL CHANGE THE TIME OUT DELAY FOR SUBSYSTEM OPERATIONS.
;*****
TORTE:  JSR    PC,SBSN    ;BUMP TO PARAMETER
      TSTB   (R1)        ;TEST IF 0
      BEQ    1$          ;EXIT - NO TIME OUT CHANGE
      CMPB   (R1),#'?    ;TEST IF QUESTION
      BNE    2$          ;NO - SKIP TYPE
      MOV    TOVAL,-(SP) ;GET TOVAL TO STACK
      TYPDS  SCRLF       ;TYPE IT
      BR     1$
2$:    MOV    R1,-(SP)    ;GET VALUE FOR CONVERSION
      JSR    PC,DECBIN   ;CONVERT VALUE TO BINARY
      20$
      MOV    (SP)+,TOVAL ;STORE VALUE
      TST    (R4)+       ;SET GOOD RETURN
      BR     30$
20$:  MOV    (R4),R4      ;SET ERROR RETURN
      TYPE   BADDEC      ;TYPE MESSAGE
30$:  RTS     R4          ;RETURN

;*****
;SBTTL DRIVE NUMBER CHANGE ROUTINE
;ENTRY:  JSR     R4,DNATE
;        WITH R1 POINTING TO THE COMMAND FIELD IF DN
;        COMMAND OR SUBSYSTEM COMMAND PARAMETER IF
;        SYBSYSTEM FUNCTION COMMAND
;ERROR RETURN:  MOV    (R4),R4
;               RTS     R4
;NO ERROR RETURN:  TST    (R4)+
;               RTS     R4
;
;THIS ROUTINE CHANGES THE "DRIVE" PARAMETER BY STORING
;IN IT THE VALUE SPECIFIED BY THE "DN" OR "SF"
;COMMANDS NOTE R1 MUST POINT TO THE COMMAND
;FIELD (DN) OR THE SUBSYSTEM COMMAND PARAMETER (SF).
;
;ROUTINES CALLED
;OCTBIN
```

3002  
3003  
3004  
3005 006144  
3006 006144 010146  
3007 006146 004737 004666  
3008 006152 105711  
3009 006154 001425  
3010 006156 121127 000077  
3011 006162 001006  
3012 006164 013746 001176  
3013 006170 104402  
3014 006172 104401 001171  
3015 006176 000414  
3016 006200 121127 000054  
3017 006204 001411  
3018 006206 010146  
3019 006210 004737 031160  
3020 006214 006234  
3021 006216 011637 001176  
3022 006222 022627 000007  
3023 006226 101005  
3024 006230 005724  
3025 006232 000406  
3026 006234 104401 002640  
3027 006240 000402  
3028 006242 104401 003146  
3029 006246 011404  
3030 006250  
3031 006250 012601  
3032 006252 000204  
3033  
3034  
3035  
3036  
3037  
3038  
3039  
3040  
3041  
3042  
3043  
3044  
3045  
3046 006254 004737 004666  
3047 006260 105711  
3048 006262 001417  
3049 006264 121127 000077  
3050 006270 001006  
3051 006272 013746 001232  
3052 006276 104405  
3053 006300 104401 001171  
3054 006304 000406  
3055 006306 010146  
3056 006310 004737 031314  
3057 006314 006326

```

;*
;*SBSN
DNRTE:
      MOV      R1,-(SP)          ;: PUSH R1 ON STACK
      JSR      PC,SBSN          ;: BUMP R1 TO NEXT PARAM
      TSTB     (R1)             ;: TEST IF PARAM NULL
      BEQ      1$              ;: NO DRIVE CHANGE EXIT
      CMPB     (R1),#'?         ;: TEST IF QUESTION
      BNE      4$              ;: NO - SKIP TYPE
      MOV      DRIVE,-(SP)      ;: GET DRIVE NUM TO STACK
      TYPOC    TYPE            ;: TYPE IT
      TYPE     $CRLF
      BR       1$
4$:    CMPB     (R1),#'         ;: TEST IF NULL ENTRY
      BEQ      1$              ;: IF YES NO DRIVE CHANGE EXIT
      MOV      R1,-(SP)         ;: ADDRESS OF ASCII CHAR
      JSR      PC,OCTBIN        ;: CONVERT TO BINARY
      2$       ;: ERROR RETURN
      MOV      (SP),DRIVE       ;: STORE NEW DRIVE NUMBER
      CMP      (SP),#?         ;: TEST IF VALID DRIVE
      BHI      3$              ;: YES - TYPE BAD DRIVE NUM
      TST      (R4)+            ;: SET NORMAL RETURN
      BR       30$
2$:    TYPE     $BADOC          ;: TYPE INVALID OCTAL NUMBER
      BR       29$
3$:    TYPE     $BADRV          ;: PRINT MESSAGE
29$:   MOV      (R4),R4         ;: SET ERROR RETURN
30$:   MOV      (SP)+,R1        ;: POP STACK INTO R1
      RTS      R4              ;: RETURN

;*****
;SBTTL ITERATION COUNT CHANGE ROUTINE
;ENTRY: JSR      R4,ICRTE
;       WITH R1 POINTING TO INPUT COMMAND
;RETURN: RTS      R4          ;: ERROR RETURN
;       RTS      R4+2        ;: NO ERROR RETURN
;
;ROUTINES CALLED:
;DECBIN
;SBSN
;*****

ICRTE: JSR      PC,SBSN          ;: BUMP R1 TO NEXT PARAM (IC)
      TSTB     (R1)             ;: TEST FOR NULL
      BEQ      1$              ;: IF NULL, EXIT. NO CHANGE IC
      CMPB     (R1),#'?         ;: TEST IF QUESTION
      BNE      2$              ;: NO - SKIP TYPE
      MOV      ITCNT,-(SP)      ;: GET ITERATION CNT TO STACK
      TYPOD    TYPE            ;: TYPE IT
      TYPE     $CRLF
      BR       1$
2$:    MOV      R1,-(SP)         ;: ADDRESS OF PARAM ON STACK
      JSR      PC,DECBIN        ;: CONVERT IT TO BINARY
      20$      ;: ERROR RETURN

```



3058 006316 012637 001232  
3059 006322 005724  
3060 006324 000403  
3061 006326 011404  
3062 006330 104401 002512  
3063 006334 000204  
3064  
3065  
3066  
3067  
3068  
3069  
3070  
3071  
3072  
3073  
3074  
3075  
3076  
3077  
3078  
3079  
3080  
3081  
3082  
3083  
3084  
3085  
3086  
3087

MOV (SP)+,ITCNT ;STORE ITERATION COUNT  
1\$: TST (R4)+ ;SET UP NORMAL RETURN  
BR 30\$  
20\$: MOV (R4),R4 ;SET UP ERROR RETURN  
TYPE BADDEC ;TYPE BAD DECIMAL MESSAGE  
30\$: RTS R4 ;RETURN

\*\*\*\*\*  
;SBTTL SPECIAL DATA PATTERN ROUTINE  
;ENTRY: JSR R4,DPRTE  
; WITH R1 POINTING TO THE INPUT COMMAND  
;RETURN: RTS R4 ERROR RETURN  
; RTS R4+2 NO ERROR ROUTINE  
;THIS ROUTINE WILL ACCEPT 32 WORDS OR LESS AND STORE  
;THEM WITH THE IDENTIFIER X,Y, OR Z AS CHOSEN BY  
;PARAMETER 1. INPUT DATA MUST BE IN OCTAL AND IN  
;WORD FORMAT (6 OCTAL CHARACTERS WITH THE UPPER BIT A 0).  
;CARRIAGE RETURNS MAY BE USED TO TERMINATE A LINE  
;WITHOUT AFFECTING THE INPUT DATA BUT IT MUST OCCUR  
;ON A WORD BOUNDARY. (IF NOT ON WORD BOUNDARY THE  
;REMAINDER OF THE WORD IS ZERO FILLED.) THE  
;INPUT DATA IS TERMINATED WITH A CARRIAGE RETURN  
;AT THE BEGINNING OF A LINE. IF LESS THAN 32  
;WORDS ARE ENTERED THE REMAINDER WORDS ARE  
;ZERO FILLED.

ROUTINES CALLED:

OCTBIN

SBSCN

3089 006336 000000 000000 000000  
3090 006344 000000  
3091 006346 052737 100000 001702  
3092 006354 000402  
3093 006356 005037 001702  
3094 006362  
3095 006362 010046  
3096 006364 010146  
3097 006366 010246  
3098 006370 010346  
3099 006372 010546  
3100 006374 010446  
3101 006376 004737 004666  
3102 006402 105711  
3103 006404 001541  
3104 006406 121127 000130  
3105 006412 001006  
3106 006414 012703 001262  
3107 006420 152737 000377 001237  
3108 006426 000421  
3109 006430 121127 000131  
3110 006434 001006  
3111 006436 012703 001362  
3112 006442 152737 000377 001240  
3113 006450 000410

CVTBUF: .WORD 0,0,0,0  
EBRTE: BIS #BIT15,TEMP1 ;SET FLAG FOR EDIT BUFFER  
BR DPRTE1 ;SKIP  
DPRTE: CLR TEMP1 ;CLEAR EDIT BUFFER FLAG  
DPRTE1:  
MOV R0,-(SP) ;PUSH R0 ON STACK  
MOV R1,-(SP) ;PUSH R1 ON STACK  
MOV R2,-(SP) ;PUSH R2 ON STACK  
MOV R3,-(SP) ;PUSH R3 ON STACK  
MOV R5,-(SP) ;PUSH R5 ON STACK  
MOV R4,-(SP) ;PUSH R4 ON STACK  
JSR PC,SBSCN ;BUMP R1 TO NEXT PARAM (PAT NAME)  
TSTB (R1) ;TEST PARAM NULL  
BEQ 20\$ ;CANNOT ACCEPT NULL, BRANCH TO ERROR  
CMPB (R1),#'X ;TEST PATTERN NAME FOR  
1\$ BNE 1\$ ;X, Y, OR Z. SET UP R3  
MOV #PATX,R3 ;WITH ADDRESS OF AREA  
BISB #377,PATXDF ;SET PAT X DEFINED SWITCH  
BR 3\$ ;TO STORE DATA PATTERN  
1\$: CMPB (R1),#'Y  
BNE 2\$  
MOV #PATY,R3  
BISB #377,PATYDF ;SET PAT Y DEFINED SWITCH  
BR 3\$

```

3114 006452 121127 000132      2$: CMPB (R1),#2
3115 006456 001114      BNE 20$ ;INVALID PATTERN NAME, ERROR
3116 006460 012703 001462      MOV #PATZ,R3
3117 006464 152737 000377 001241 BISB #377,PATZDF ;SET PAT 2 DEFINED SWITCH
3118 006472 010300      3$: MOV R3,R0 ;STORE R3 (BUFFER AREA)
3119 006474 005737 001702      TST TEMP1 ;TEST EDIT FLAG
3120 006500 100016      BPL 39$ ;IF SET - SKIP
3121 006502 004737 004666      JSR PC,SBSCN ;ELSE BUMP TO NEXT PARAM, WORD NUM
3122 006506 010146      MOV R1,-(SP) ;GET ADDRESS OF PARAM
3123 006510 004737 031160      JSR PC,OCTBIN ;CONVERT IT TO OCTAL
3124 006514 006710      20$ ;ERROR RETURN
3125 006516 011602      MOV (SP),R2 ;GET WORD NUMBER FOR START OF EDIT
3126 006520 006302      ASL R2 ;NOW ITS WORD ADDRESS
3127 006522 060200      ADD R2,R0 ;SET TO INDEX INTO BUFFER
3128 006524 012702 000040      MOV #1032,R2 ;SET MAX OF EDIT
3129 006530 162602      SUB (SP)+,R2 ;NOW SET TO REMAINDER OF BUFFER LENGTH
3130 006532 100466      BMI 20$ ;ERROR, EDIT IS OUT OF BOUNDS
3131 006534 000410      BR 45$
3132 006536 012704 000040      39$: MOV #1032,R4 ;SET R4 FOR COUNT
3133 006542 012723 177777      44$: MOV #1,(R3)+ ;SET PATTERN STORAGE
3134 006546 005304      DEC R4 ;AREA, 32 WORDS
3135 006550 001374      BNE 44$ ;LOOP
3136
3137 006552 012702 000040      MOV #1032,R2 ;SET TOTAL WORD COUNT
3138 ;THE REGISTERS USAGE FOR THE REMAINDER IS AS FOLLOWS:
3139 ; R0 POINTS TO THE PATTERN STORAGE AREA
3140 ; R1 POINTS TO THE ASCII INPUT DATA
3141 ; R3 USED AS A SWITCH, SET WHEN NULL (CARRIAGE
3142 ; RETURN IS DETECTED.
3143 ; R4 COUNTER FOR TICKING OFF 6 ASCII INPUT
3144 ; CHARACTERS (ONE WORD)
3145 ; R5 POINTS TO THE TEMPORARY CONVERSION BUFFER (CVTBUF)
3146 ; R2 COUNTER TO LIMIT INPUT TO 32 WORDS. ALSO
3147 ; USED TO FORCE EXIT IF TWO CONSECUTIVE
3148 ; CARRIAGE RETURNS ARE TYPED.
3149
3150
3151 006556 005003      45$: CLR R3 ;RESET CR SWITCH
3152 006560 004737 004666      JSR PC,SBSCN ;BUMP R1 TO NEXT PARAM (DATA)
3153 006564 012705 006336      4$: MOV #CVTBUF,R5 ;SET UP CVTBUF POINTER
3154 006570 012704 000006      MOV #6,R4 ;SET UP CONVERSION COUNT
3155 006574 112125      5$: MOVB (R1)+,(R5)+ ;MOVE ASCII CHAR TO CVTBUF. IF
3156 006576 001404      BEQ 6$ ;0 (NULL) EXIT LOOP CLEAR CR SWITCH IF
3157 006600 005003      CLR R3 ;SET (NON-NUL CHAR TYPED). DEC CONVERT
3158 006602 005304      DEC R4 ;COUNT AND
3159 006604 001373      BNE 5$ ;LOOP. IF ONE WORD READY,
3160 006606 000415      BR 10$ ;BRANCH TO CONVERSION
3161 006610 005703      6$: TST R3 ;TEST IF CR SWITCH SET. IF NOT
3162 006612 001402      BEQ 7$ ;SET CR SWITCH. IF SET, CLEAR R2
3163 006614 005002      CLR R2 ; (TOTAL WORD COUNT) TO PREPARE FOR
3164 006616 000401      BR 8$ ;EXIT
3165 006620 005103      7$: COM R3 ;SETTING CR SWITCH FOR ABOVE
3166 006622 020427 000006      8$: CMP R4,#6 ;TEST IF PARTIAL WORD TYPED. IF
3167 006626 001414      BEQ 11$ ;YES, 0 FILL REST OF WORD. GO TO CONVERT
3168 006630 005305      DEC R5 ;AND PLACE IN PAT STORE. IF NOT, GO TO
3169 006632 112725 000060      9$: MOVB #0,(R5)+ ;CHECK IF DONE.

```

3170	006636	005304		DEC	R4	
3171	006640	001374		BNE	9\$	
3172	006642	012746	006336	10\$:	MOV	#CVTBUF, -(SP)
3173	006646	004737	031160		JSR	PC, OCTBIN
3174	006652	006716		21\$		
3175	006654	012620		MOV	(SP)+, (R0)+	
3176	006656	005302		DEC	R2	
3177	006660	005702		11\$:	TST	R2
3178	006662	001407			BEQ	12\$
3179	006664	005703			TST	R3
3180	006666	001736			BEQ	4\$
3181	006670	104401	003521		TYPE	, SPACE6
3182	006674	104411			ROLIN	
3183	006676	012601			MOV	(SP)+, R1
3184	006700	000731			BR	4\$
3185	006702	012604		12\$:	MOV	(SP)+, R4
3186	006704	005724			TST	(R4)+
3187	006706	000407			BR	30\$
3188	006710	104401	002616	20\$:	TYPE	, IVDPAR
3189	006714	000402			BR	25\$
3190	006716	104401	002640	21\$:	TYPE	, BADOCT
3191	006722	012604		25\$:	MOV	(SP)+, R4
3192	006724	011404			MOV	(R4), R4
3193	006726			30\$:		
3194	006726	012605			MOV	(SP)+, R5
3195	006730	012603			MOV	(SP)+, R3
3196	006732	012602			MOV	(SP)+, R2
3197	006734	012601			MOV	(SP)+, R1
3198	006736	012600			MOV	(SP)+, R0
3199	006740	000204			RTS	R4

```

; START OF CONVERT. RESET CVTBUF PTR
; CALL CONVERSION
; CONVERSION ERROR ROUTINE
; STORE CONVERTED VALUE IN PAT STORE
; DEC TOTAL WORD COUNTER
; TEST IF WORD CNTR 0.
; EXIT IF YES
; TEST CARRIAGE RETURN SWITCH
; IF NOT SET, GET NEXT 6 CHAR. ELSE
; TYPE 6 SPACES TO ALIGN DATA INPUT
; READ NEXT INPUT LINE
; GET ADDRESS OF INPUT
; LOOP TO PROCESS NEW LINE
; RESTORE R4 FOR RETURN
; NO ERROR RETURN

; TYPE INVALID PARAMETER

; TYPE BAD OCTAL CHARACTERS
; RESTORE R4 FOR RETURN
; ERROR RETURN

; POP STACK INTO R5
; POP STACK INTO R3
; POP STACK INTO R2
; POP STACK INTO R1
; POP STACK INTO R0
; RETURN

```

```

3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219
3220
3221
3222
3223
3224
3225

```

```

;*****
;SBTTL PRINT REGISTER ROUTINE
;ENTRY: JSR R4 PRATE
;      WITH R1 POINTING TO INPUT COMMAND
;RETURN: RTS R4 ERROR RETURN
;      RTS R4+2 NO ERROR RETURN
;
;THIS ROUTINE WILL READ THE RK611 UNIBUS VISIBLE
;REGISTER AND PRINT THE VALUE (OCTAL) ON THE
;TERMINAL. THE REGISTER MAY BE SPECIFIED IN OCTAL
;FROM 00 TO 17 (NUMBER 13 RESERVED FOR FUTURE USE)
;OR NMEMONICALLY AS:
;
;NUMBER NAME DESCRIPTION
;0 CS1 COMMAND STATUS REGISTER 1
;01 WC WORD COUNT
;02 BA BUFFER ADDRESS
;03 DA DESIRED ADDRESS - TRACK AND SECTOR
;04 CS2 COMMAND STATUS REGISTER 2
;05 DS DRIVE STATUS
;06 ER ERROR REGISTER
;07 ASOF ATTENTION SUMMARY AND OFFSET REGISTER
;10 DC DESIRED CYLINDER
;12 DB DATA BUFFER
;13 MR1 MAINTENANCE REGISTER 1
;14 POS ECC POSITION REGISTER

```

3226  
3227  
3228  
3229  
3230  
3231  
3232  
3233  
3234 006742  
3235 006742 010346  
3236 006744 004737 004666  
3237 006750 105711  
3238 006752 001416  
3239 006754 121127 000067  
3240 006760 101404  
3241 006762 004737 007056  
3242 006766 007044  
3243 006770 000405  
3244 006772 010146  
3245 006774 004737 031160  
3246 007000 007036  
3247 007002 012603  
3248 007004 010337 001224  
3249 007010 013703 001224  
3250 007014  
3251 007014 006303  
3252 007016 063703 023554  
3253 007022 011346  
3254 007024 104402  
3255 007026 104401 001171  
3256 007032 005724  
3257 007034 000406  
3258 007036 104401 002640  
3259 007042 000402  
3260 007044 104401 003365  
3261 007050 011404  
3262 007052  
3263 007052 012603  
3264 007054 000274  
3265  
3266  
3267  
3268  
3269  
3270  
3271  
3272  
3273  
3274  
3275  
3276  
3277  
3278 007056 121127 000101  
3279 007062 001003  
3280 007064 012703 000007  
3281 007070 000533

```

; *      15      PAT      ECC PATTERN REGISTER
; *      16      MR2      MAINTENANCE REGISTER 2
; *      17      MR3      MAINTENANCE REGISTER 3
; *ROUTINES CALLED
; *      TYP0C
; *      SBSCN
; *****
PRRTE:
      MOV      R3, -(SP)      ; PUSH R3 ON STACK
      JSR      PC, SBSCN     ; BUMP R1 TO NEXT PARAM (RN)
      TSTB     (R1)          ; TEST IF PARAM NULL
      BEQ      1$            ; NO REG SPECIFIED, USED LAST REG SELECTED
      CMPB     (R1), #67     ; TEST IF NUMERIC PARAMETER
      BLOS     4$            ; YES - SKIP
      JSR      PC, RGDCDE     ; GO DECODE REGISTER
      21$
      BR       3$            ; ERROR RETURN
4$:   MOV      R1, -(SP)     ; GET ADDRESS OF STRING FOR CNVERSION
      JSR      PC, OCTBIN    ; CONVERT IT
      20$
      MOV      (SP)+, R3     ; ERROR RETURN
      MOV      R3, REGNUM    ; GET CONVERTED NUMBER
      3$:   MOV      REGNUM, R3 ; STORE INTEGER
      1$:
      2$:
      ASL      R3            ; SHIFT R3, MULTIPLY INDEX BY 2
      ADD      RKBAS, R3     ; COMPUTE RK611 ADDRESS
      MOV      (R3), -(SP)   ; PUT SELECTED REGISTER CONTENTS ON STACK
      TYP0C     $CRLF        ; TYPE REGISTER CONTENTS
      TYPE     (R4)+         ; TYPE CARRIAGE RETURN & LINE FEED
      TST      30$          ; SET UP NO ERROR RETURN
      BR       30$
20$:  TYPE     BADOCT        ; TYPE BAD OCTAL MESSAGE
      BR       25$          ; GO TO EXIT
21$:  TYPE     BADSEL        ; TYPE BAD REGISTER SELECTION
25$:  MOV      (R4), R4      ; SET UP ERROR RETURN
30$:  MOV      (SP)+, R3     ; POP STACK INTO R3
      RTS      R4           ; RETURN
; *****
; SBTL  CONVERT REGISTER NAME (ASCII) TO REGISTER NUMBER (OCTAL)
; ENTRY:      JSR      PC, RGDCDE
;             WITH R1 POINTING TO THE REG NAME
; RETURN:     RTS      PC    NORMAL RETURN
;             RTS      PC+2  ERROR RETURN
;             WITH R3 CONTAINING THE REG NUMBER
; *THE ASCII NAME OF THE REGISTER IS DECODED INTO AN OCTAL VALUE
; *REQUIRED TO SELECT THE REGISTER. THIS OCTAL VALUE IS PLACED
; *IN R3.
; *****
RGDCDE: CMPB     (R1), #'A    ; TEST IF FIRST CHAR IS A
      BNE      1$            ; NO - SKIP
      MOV      #7, R3        ; SET FOR ASOF
      BR       40$          ; GO TO EXIT
```

## K05

CZR6RC0 RK611/06 USR DEFINED  
CZR6RC.P11 02-DEC-77 11:07

MACY11 30(1046) 02-DEC-77 11:59 PAGE 63  
CONVERT REGISTER NAME (ASCII) TO REGISTER NUMBER (OCTAL)

SEQ 0062

3282	007072	121127	000102	1\$:	CMPB	(R1), #'B	;TEST IF B
3283	007076	001003			BNE	2\$	
3284	007100	012703	000002		MOV	#2, R3	;SET FOR BA
3285	007104	000525			BR	40\$	;GO TO EXIT
3286	007106	121127	000103	2\$:	CMPB	(R1), #'C	;TEST IF C
3287	007112	001012			BNE	5\$	
3288	007114	062701	000002		ADD	#2, R1	;BUMP R1 TO 3RD CHAR
3289	007120	121127	000061		CMPB	(R1), #'1	;TEST IF THIRD CHAR IS 1
3290	007124	001002			BNE	3\$	;NO - BRANCH
3291	007126	005003			CLR	R3	;SET FOR CS1
3292	007130	000513			BR	40\$	
3293	007132	012703	000004	3\$:	MOV	#4, R3	;SET FOR CS2
3294	007136	000510			BR	40\$	
3295	007140	121127	000104	5\$:	CMPB	(R1), #'D	;TEST IF D
3296	007144	001026			BNE	9\$	;NO - SKIP
3297	007146	005201			INC	R1	;BUMP TO 2ND CHAR
3298	007150	121127	000101		CMPB	(R1), #'A	;TEST 2ND CHAR A
3299	007154	001003			BNE	6\$	
3300	007156	012703	000003		MOV	#3, R3	;SET FOR DA
3301	007162	000476			BR	40\$	;EXIT
3302	007164	121127	000123	6\$:	CMPB	(R1), #'S	;TEST IF 2ND CHAR S
3303	007170	001003			BNE	7\$	
3304	007172	012 33	000005		MOV	#5, R3	;SET FOR DS
3305	007176	000470			BR	40\$	
3306	007200	121127	000102	7\$:	CMPB	(R1), #'B	;TEST IF 2ND CHAR B
3307	007204	001003			BNE	8\$	
3308	007206	012703	000012		MOV	#12, R3	;SET FOR DB
3309	007212	000462			BR	40\$	
3310	007214	012703	000010	8\$:	MOV	#10, R3	;SET FOR DC
3311	007220	000457			BR	40\$	
3312	007222	121127	000105	9\$:	CMPB	(R1), #'E	;TEST IF E
3313	007226	001003			BNE	10\$	
3314	007230	012703	000006		MOV	#6, R3	;SET FOR ER
3315	007234	000451			BR	40\$	
3316	007236	121127	000115	10\$:	CMPB	(R1), #'M	;TEST IF M
3317	007242	001021			BNE	13\$	
3318	007244	0627 1	000002		ADD	#2, R1	;BUMP R1 TO 3RD CHAR
3319	007250	121127	000061		CMPB	(R1), #'1	;TEST IF 3RD CHAR 1
3320	007254	001003			BNE	11\$	
3321	007256	012703	000013		MOV	#13, R3	;SET FOR MR1
3322	007262	000436			BR	40\$	
3323	007264	121127	000062	11\$:	CMPB	(R1), #'2	;TEST IF 3RD CHAR 2
3324	007270	001003			BNE	12\$	
3325	007272	012703	000016		MOV	#16, R3	;SET FOR MR2
3326	007276	000430			BR	40\$	
3327	007300	012703	000017	12\$:	MOV	#17, R3	;SET FOR MR3
3328	007304	000425			BR	40\$	
3329	007306	121127	000120	13\$:	CMPB	(R1), #'P	;TEST IF P
3330	007312	001012			BNE	15\$	
3331	007314	005201			INC	R1	;BUMP R1 TO 2ND CHAR
3332	007316	121127	000117		CMPB	(R1), #'O	;TEST 2ND CHAR O
3333	007322	001003			BNE	14\$	
3334	007324	012703	000014		MOV	#14, R3	;SET FOR POS
3335	007330	000413			BR	40\$	
3336	007332	012703	000015	14\$:	MOV	#15, R3	;SET FOR PAT
3337	007336	000410			BR	40\$	

3338 007340 121127 000127  
3339 007344 001003  
3340 007346 012703 000001  
3341 007352 000402  
3342 007354 013616  
3343 007356 000207  
3344 007360 062716 000002  
3345 007364 000207  
3346  
3347  
3348  
3349  
3350  
3351  
3352  
3353  
3354  
3355  
3356  
3357  
3358 007366 105737 001672  
3359 007372 001006  
3360 007374 104401 002656  
3361 007400 104401 041616  
3362 007404 005724  
3363 007406 000204  
3364 007410 104401 034636  
3365 007414 000773  
3366  
3367  
3368  
3369  
3370  
3371  
3372  
3373  
3374  
3375  
3376  
3377  
3378  
3379  
3380  
3381  
3382  
3383 007416  
3384 007416 010346  
3385 007420 010546  
3386 007422 152737 000377 001234  
3387 007430 105037 001235  
3388 007434 012737 034634 001712  
3389 007442 013737 001710 001246  
3390 007450 105037 001116  
3391 007454 105037 001243  
3392 007460 013703 001246  
3393

```
15$:  CMPB    (R1),#W          ;TEST IF W
      BNE     20$              ;BR TO ERROR EXIT
      MOV     #1,R3            ;SET FOR WC
      BR      40$
20$:  MOV     2(SP)+,(SP)       ;SET ERROR RETURN
      RTS     PC
40$:  ADD     #2,(SP)           ;SET FOR GOOD RETURN
      RTS     PC
;*****
;SBTTL  HELP PRINTOUT ROUTINE
;*      ENTRY: JSR     R4,HPRTE
;*      RETURN: RTS     R4
;*
;*THIS ROUTINE PRINTS A SUMMARY OF THE COMMANDS
;*AND PARAMETERS AVAILABLE TO THE USER.
;* ROUTINES CALLED:
;*      TYPE
;*****
HPRTE: TSTB    HPVLD            ;TEST IF HELP FILE VALID
      BNE     1$
      TYPE    ,HPFILE
      TYPE    ,ENDLOC
2$:    TST     (R4)+            ;GOOD RETURN
      RTS     R4                ;RETURN
1$:    TYPE    ,HPDATA          ;TYPE HELP FILE
      BR      2$
;*****
;SBTTL  NEW TEST ROUTINE
;*      ENTRY JSR     R4,NTRTE
;*      RETURN: JMP     COMLEV (RETURN TO COMMAND LEVEL)
;*
;*THIS ROUTINE CLEARS THE SOURCE AND OBJECT FILES
;*AND TERMINATES ANY TEST PRESENTLY EXECUTING. ALL
;*STORED TEST PARAMETERS AND THE OUTPUT BUFFER
;*ARE LEFT UNCHANGED. SINCE THE INPUT BUFFER
;*AND THE SOURCE FILE IS THE SAME MEMORY, THE
;*INPUT BUFFER IS LOST AND MUST BE REINITIALIZED.
;*
;* ROUTINES CALLED
;*      NONE
;*****
NTRTE: MOV     R3,-(SP)          ;PUSH R3 ON STACK
      MOV     R5,-(SP)          ;PUSH R5 ON STACK
      BISB    #377,SFEMP        ;SET SOURCE FILE EMPTY
      CLRB    VLD OBJ          ;CLEAR OBJECT VALID
      MOV     #0,OFPTR          ;RESET OBJECT FILE POINTER
      MOV     IBUFPT,SFPTR      ;RESET SOURCE FILE POINTER
      CLRB    LINNUM            ;CLEAR LINE NUMBER
      CLRB    LNCNT            ;CLEAR LINE COUNT
      MOV     SFPTR,R3
```

3394 007464 013705 001700  
3395 007470 005023  
3396 007472 005305  
3397 007474 001375  
3398 007476 013705 001674  
3399 007502 013703 001712  
3400 007506 005023  
3401 007510 005305  
3402 007512 001375  
3403 007514 012605  
3404 007516 012603  
3405 007520 012706 001100  
3406 007524 000137 004416  
3407  
3408  
3409  
3410  
3411  
3412  
3413  
3414  
3415  
3416 007530  
3417 007530 010046  
3418 007532 010146  
3419 007534 010246  
3420 007536 010346  
3421 007540 005002  
3422 007542 004737 004666  
3423 007546 105711  
3424 007550 001522  
3425 007552 121127 000122  
3426 007556 001003  
3427 007560 013703 001710  
3428 007564 000441  
3429 007566 121127 000127  
3430 007572 001003  
3431 007574 013703 001706  
3432 007600 000433  
3433 007602 012702 000040  
3434 007606 121127 000130  
3435 007612 001003  
3436 007614 012703 001262  
3437 007620 000423  
3438 007622 121127 000131  
3439 007626 001003  
3440 007630 012703 001362  
3441 007634 000415  
3442 007636 121127 000132  
3443 007642 001003  
3444 007644 012703 001462  
3445 007650 000407  
3446 007652 121127 000110  
3447 007656 001057  
3448 007660 012703 001736  
3449 007664 012702 000102

```
1$:  MOV    MAXWDS,R5
      CLR    (R3)+      ;SET UP REGISTERS
      DEC    R5          ;AND CLEAR SOURCE
      BNE    1$          ;FILE
      MOV    OBJSZ,R5
      MOV    OFPTR,R3
2$:  CLR    (R3)+      ;SET UP REGISTERS AND
      DEC    R5          ;CLEAR OBJECT FILE
      BNE    2$
      MOV    (SP)+,R5    ;POP STACK INTO R5
      MOV    (SP)+,R3    ;POP STACK INTO R3
      MOV    #1100,SP    ;CLEAN OFF STACK
      JMP    COMLEV      ;GO TO COMMAND LEVEL
;*****
;SBTTL BUFFER DUMP ROUTINE
;*  ENTRY: JSR    R4,BORTE
;*  RETURN: RTS    R4
;THIS ROUTINE WILL DUMP THE READ, WRITE, HEADER, OR SPECIAL BUFFER. THE
;NUMBER OF WORDS DUMPED IS GIVEN AS A PARAMETER. IF THE NUMBER OF
;WORDS IS NOT GIVEN THE LAST SPECIFIED WORD COUNT IS USED IN THE CASE
;OF THE READ OR WRITE BUFFER OR 32 IN THE CASE OF A SPECIAL DATA BUFFER.
;*****
BORTE: MOV    R0,-(SP)    ;PUSH R0 ON STACK
      MOV    R1,-(SP)    ;PUSH R1 ON STACK
      MOV    R2,-(SP)    ;PUSH R2 ON STACK
      MOV    R3,-(SP)    ;PUSH R3 ON STACK
      CLR    R2          ;CLEAR FOR POSSIBLE WORD COUNT
      JSR    PC,SBSCN    ;GET BUFFER PARAMETER
      TSTB   (R1)        ;TEST IF IT IS NULL
      BEQ    21$         ;YES - SKIP TO ERROR EXIT
      CMPB   (R1),#'R    ;TEST IF READ BUFFER
      BNE    1$         ;NO - SKIP
      MOV    IBUFPT,R3   ;ELSE GET ADDRESS OF READ BUFFER
      BR     5$         ;GO DO IT
1$:  CMPB   (R1),#'W    ;TEST IF WRITE BUFFER
      BNE    2$         ;NO - SKIP
      MOV    OBUFPT,R3   ;ELSE GET ADDRESS OF WRITE BUFFER
      BR     5$         ;GO DO IT
2$:  MOV    #40,R2       ;SET WORD COUNT FOR SPEC BUFF
      CMPB   (R1),#'X    ;TEST IF SPECIAL BUFFER X
      BNE    3$         ;NO - SKIP
      MOV    #PATX,R3    ;ELSE GET ADDRESS OF BUFFER X
      BR     5$         ;GO DO IT
3$:  CMPB   (R1),#'Y    ;TEST IF BUFFER Y
      BNE    4$         ;NO - SKIP
      MOV    #PATY,R3    ;ELSE GET ADDRESS OF BUFFER Y
      BR     5$         ;GO DO IT
4$:  CMPB   (R1),#'Z    ;TEST IF BUFFER Z
      BNE    44$        ;NO - SKIP
      MOV    #PATZ,R3    ;GET ADDRESS OF BUFFER Z
      BR     5$         ;GO DO IT
44$: CMPB   (R1),#'H    ;TEST IF HEADER BUFF DUMP
      BNE    21$        ;NO - SKIP TO ERROR EXIT
      MOV    #HDBUFF,R3 ;SET ADDRESS FOR HEADER BUFF
      MOV    #102,R2    ;SET SPECIAL BUFF LENGTH IF WRD CNT NULL
```

3450	007670	004737	004666	5\$:	JSR	PC,SBSCN	;GET NUMBER OF WORDS PARAM
3451	007674	105711			TSTB	(R1)	;TEST IF NULL
3452	007676	001007			BNE	7\$	;NO - SKIP TO USE GIVEN VALUE
3453	007700	005702			TST	R2	;ELSE USE DEFAULT WORD NUMBER
3454	007702	001402			BEQ	6\$	;IF NO WRD CNT IN R2, GO USE WORD COUNT
3455	007704	010201			MOV	R2,R1	;ELSE SET TO R2 COUNT
3456	007706	000410			BR	8\$	
3457	007710	013701	001206	6\$:	MOV	WDCNT,R1	;GET WORD COUNT
3458	007714	000405			BR	8\$	
3459	007716	010146		7\$:	MOV	R1,-(SP)	;SET UP TO CONVERT PARAMETER
3460	007720	004737	031160		JSR	PC,OCTBIN	;GO CONVERT
3461	007724	010010			20\$		;ERROR RETURN
3462	007726	012601			MOV	(SP)+,R1	;STORE WORD COUNT GIVEN
3463	007730	005002		8\$:	CLR	R2	;CLEAR COUNTERS
3464	007732	005000			CLR	R0	
3465	007734	104401	003465		TYPE	,DMPHDR	;TYPE DUMP HEADERS
3466	007740	012700	000004	9\$:	MOV	R4,R0	;SET NUMBER OF COLUMNS COUNTER
3467	007744	010246			MOV	R2,-(SP)	;SET TO PRINT WORD NUMBER
3468	007746	104402			TYPOC		;TYPE IT
3469	007750	104401	003530	10\$:	TYPE	,SPACE2	;TYPE FORMAT SPACES
3470	007754	012346			MOV	(R3)+,-(SP)	;GET WORD TO TYPE
3471	007756	104402			TYPOC		
3472	007760	005202			INC	R2	;BUMP WORD COUNTER
3473	007762	005301			DEC	R1	;DEC NUMBER OF WORDS TO TYPE COUNT
3474	007764	001405			BEQ	11\$	;IF 0, EXIT
3475	007766	005300			DEC	R0	;DEC NUMBER OF COL COUNT
3476	007770	001367			BNE	10\$	;IF NOT 0, GO TYPE FORMAT SPACES AND NEXT COL
3477	007772	104401	001171		TYPE	,\$CRLF	;ELSE LF-CR AND START NEW LINE
3478	007776	000760			BR	9\$	;LOOP
3479	010000	104401	001171	11\$:	TYPE	,\$CRLF	;RETURN CARRIAGE
3480	010004	005724			TST	(R4)+	;SET UP NO ERROR RETURN
3481	010006	000406			BR	25\$	
3482	010010	104401	002640	20\$:	TYPE	,BADOCT	;REPORT NON-OCTAL PARAMETER
3483	010014	000402			BR	24\$	
3484	010016	104401	002616	21\$:	TYPE	,IVDPAR	;REPORT INVALID PARAM
3485	010022	011404		24\$:	MOV	(R4),R4	;ERROR RETURN
3486	010024			25\$:			
3487	010024	012603			MOV	(SP)+,R3	;POP STACK INTO R3
3488	010026	012602			MOV	(SP)+,R2	;POP STACK INTO R2
3489	010030	012601			MOV	(SP)+,R1	;POP STACK INTO R1
3490	010032	012600			MOV	(SP)+,R0	;POP STACK INTO R0
3491	010034	000204			RTS	R4	
3492							
3493							
3494							
3495							
3496							
3497							
3498							
3499							
3500							
3501							
3502							
3503							
3504							
3505							

;\*\*\*\*\*  
 ;SBTTL RUN ROUTINE  
 ;\* ENTRY: JSR R4,RURTE  
 ;\* RETURN: RESET STACK, JUMP TO COMMAND LEVEL  
 ;\*  
 ;\*THIS ROUTINE CHECKS TO BE SURE OBJECT CODE EXISTS.  
 ;\*IT THEN CHECKS THE NOCK SWITCH TO SEE IF THE OBJECT  
 ;\*WAS COMPILED WITH THE NOCK OPTION. IF IT WAS THE ROUTINE  
 ;\*PROCEEDS TO CLEAN OFF THE STACK AND EXECUTE THE OBJECT CODE.  
 ;\*  
 ;\*IF THE NOCK SWITCH IS OFF THE ROUTINE CHECKS THE CONTROLLER  
 ;\*ERROR BIT AND THE DRIVE INTERRUPT BIT (BIT 15 & 14 OF CS1).



```

3506      *IF EITHER IS SET A SUBSYSTEM CLEAR IS EXECUTED BEFORE THE
3507      *TEST IS STARTED. THIS IS NECESSARY BECAUSE IF EITHER OF THESE
3508      *BITS ARE SET WHEN THE TEST IS STARTED (WITHOUT NO CHECK)
3509      *CONTINUOUS INTERRUPTS WILL BE GENERATED. THIS CASE CAN ONLY
3510      *OCCUR IF THE PREVIOUS TEST WAS A NO CHECK TEST AND AN ERROR
3511      *WAS LEFT UNCLEARED IN THE SUBSYSTEM.
3512      *IT THEN CLEANS OFF THE STACK AND DOES A JSR TO THE
3513      *START OF THE OBJECT FILE.
3514      *
3515      *THIS ROUTINE ALSO ACTS AS A MONITOR TO CONTROL THE
3516      *LOOPING ON THE OBJECT. IT HANDLES THE ITERATION
3517      *COUNTING AND RESTARTING THE OBJECT FILE, MAKING SURE
3518      *THE STACK IS CLEANED UP TO PREVENT ACCIDENTAL OVERFLOW.
3519      *
3520      *WHEN LOOPING IS DONE OR WHEN TEST IS ABORTED
3521      * (ABORTING IS PRESENTLY UNDEFINED) THE ROUTINE RESETS
3522      *THE STACK AND JUMPS TO COM LEV.
3523      *
3524
3525      010036 005037 001722      RUTTE: CLR      LPCNT1      ;CLEAR LOOP COUNTER
3526      010042 005037 001724      CLR      LPCNT2
3527      010046 013737 001260      MOV      TOVAL,W.SEC      ;SET TIMEOUT VALUE
3528      010054 105737 001235      TSTB     VLD0BJ      ;TEST OBJECT VALID
3529      010060 001510      BEQ      RUEXIT      ;NO OBJECT CODE
3530      010062 152737 000377 001234      BISB     #377,SFEMP      ;SET SOURCE FILE EMPTY
3531      010070 013702 023554      MOV      RKBAS,R2      ;GET UNIBUS BASE ADDRESS
3532      010074 105037 001103      CLR      SERFLG      ;CLEAR ERROR FLAG
3533      010100 032762 140000 000000      BIT      #CERR!DI,RKCS1(R2) ;TEST FOR CONT ERR OR DEV INTERRUPT
3534      010106 001403      BEQ      1$      ;BOTH OFF, SKIP CLEAR
3535      010110 052762 000040 000010      BIS      #SCLR,RKCS2(R2) ;SET CLEAR SUBSYSTEM, BIT 5 CS2
3536      010116 013737 001232 001714      1$: MOV      ITCNT,CNTSTR ;STORE OFF ITERATION COUNT
3537      010124 012737 010152 001110      MOV      #LOC2$,SLPERR ;SET UP ERROR LOOP VALUE
3538      010132 012706 001100      LPRET: MOV      #STACK,SP      ;CLEAN STACK
3539      010136 005037 001116      CLR      LINNUM      ;INITIALIZE PSUEDO LINE COUNTER
3540      010142 004737 034634      JSR      PC,OFIL      ;GO TO OBJECT CODE
3541      010146 000401      BR      LOC2$      ;GOOD RETURN
3542      010150 000454      BR      RUEXIT      ;ABORT RETURN
3543      010152 032777 040000 170760      LOC2$: BIT      #SW14,JSWR      ;LOOP ON TEST?
3544      010160 001012      1$      BNE      1$
3545      010162 105737 001103      TSTB     SERFLG      ;TEST IF ERROR FLAG SET
3546      010166 001420      BEQ      3$
3547      010170 032777 001000 170742      BIT      #SW9,JSWR      ;TEST IF LOOP ON ERROR SWITCH
3548      010176 001003      1$      BNE      1$
3549      010200 105037 001103      CLR      SERFLG      ;NO LOOP ON ERROR, CLEAR FLAG
3550      010204 000411      BR      3$
3551      010206 062737 000001 001722      1$: ADD      #1,LPCNT1      ;ADD ONE TO COUNTER
3552      010214 005537 001724      ADC      LPCNT2      ;PROPAGATE CARRY
3553      010220 103002      BCC      2$      ;TEST IF COUNTER OVERFLOWED
3554      010222 104401 003431      TYPE     LCNTOF      ;TYPE OVERFLOW WARNING
3555      010226 000741      2$: BR      LPRET      ;RETURN TO LOOP
3556      010230 032777 004000 170702      3$: BIT      #SW11,JSWR      ;INHIBIT ITERATIONS?
3557      010236 001003      BNE      LOC3$      ;BACK TO MONITOR
3558      010240 005337 001714      DEC      CNTSTR      ;DEC IT COUNT
3559      010244 001332      LPRET      ;BACK TO TEST
3560      010246 004737 016670      LOC3$: JSR      PC,PRLPCT ;GO PRINT LOOP COUNTER
3561      010252 012706 001100      5$: MOV      #STACK,SP      ;CLEAN STACK

```

3562 010256 005037 001116  
 3563 010262 105737 001663  
 3564 010266 001403  
 3565 010270 004437 013026  
 3566 010274 010302  
 3567 010276 000137 004416  
 3568 010302 104401 002756  
 3569 010306 000757

CLR LINNUM ;SET LINE NUMBER TO ZERO  
 TSTB CHNFLG ;TEST IF CHAINING  
 BEQ RURETN ;BRANCH IF NOT, ELSE  
 JSR R4, ITRTE ;JUMP TO INPUT TEST  
 RUEXIT  
 RURETN: JMP COMLEV ;JUMP TO COMMAND LEVEL  
 RUEXIT: TYPE IVDRUN ;TYPE NO OBJECT CODE  
 BR LOC35

```

*****
SBTTL  COMPILE ROUTINE
*      ENTRY  JSR      R4, CORTE
*      RETURN  RTS      R4+2   FOR ALL RETURNS.  ERROR AND ERROR
*                                  MESSAGES ARE ALL HANDLED LOCALLY
*
*THIS ROUTINE ACTS AS A MONITOR FOR THE COMPILE PROCESS.
*THE DEFERRED COMMANDS ARE EXTRACTED FROM THE SOURCE
*FILE. THE INTERACTIVE COMMAND TABLE IS SCANNED TO
*LOCATE THE ENTRY FOR THIS COMMAND. R2 IS SET
*TO POINT TO THE 2ND WORD WHICH IS THE ADDRESS OF
*THE SPECIFIC COMMAND PROCESSOR ROUTINE. CONTROL
*IS THEN GIVEN TO THAT ROUTINE TO GENERATE THE OBJECT
*CODE. RS POINTS TO WHERE THE OBJECT CODE IS TO BE
*PLACED.
*
*THIS ROUTINE CHECKS THE COMPILE COMMAND PARAMETERS. IF
*THE FIRST PARAMETER SPECIFIES NO CHECK AND THE SECOND SPECIFIES
*BUS ADDRESS INCREMENT INHIBIT FOR DATA TRANSFERS. IF THE FIRST IS
*NULL, THE NO CHECK (NOCK) SWITCH IS RESET. IF NOT NULL
*THE NO CHECK SWITCH IS SET. THIS SWITCH IS USED TO DETERMINE
*IF NO CHECKING IS TO BE DONE IN TEST EXECUTION. IF THE SECOND
*IS NULL THE BUS ADDRESS INCREMENT INHIBIT SWITCH IS RESET,
*ELSE IT IS SET.
*
*A CHECK IS MADE TO INSURE THE COMMAND LINE NUMBERS ARE
*RETRIEVED SEQUENTIALLY. IF NOT, AN "INTERNAL ERROR" MESSAGE
*IS PRINTED OUT. THIS LINE COUNT IS TESTED AGAINST THE
*STORED LINE NUMBERS. WHEN EQUAL, AN RTS PC IS INSERTED
*IN THE OBJECT FILE AND, IF NO ERROR HAS BEEN FOUND, THE
*VALID OBJECT CODE FLAG IS SET. A COMPILE OK MESSAGE IS
*THEN PRINTED.
*
*IF ANY OF THE SPECIFIC DEFERRED COMMAND PROCESSOR
*ROUTINES RETURNS AN ERROR, A MESSAGE AND THE
*BAD LINE IS PRINTED. THE COMPILE ERROR FLAG IS SET
*AND THE NEXT LINE IS PROCESSED.
*
*WHEN COMPILATION IS DONE, THE COMPILE ERROR FLAG IS
*CHECKED. IF SET, THE VALID OBJECT CODE FLAG IS NOT
*SET AND CONTROL IS RETURNED TO COMMAND LEVEL.
*
* ROUTINES CALLED
*      SBSCH
*      ICDEC
*      REQUIRED DEFERRED COMMAND PROCESSOR (CSXX)

```

3570  
 3571  
 3572  
 3573  
 3574  
 3575  
 3576  
 3577  
 3578  
 3579  
 3580  
 3581  
 3582  
 3583  
 3584  
 3585  
 3586  
 3587  
 3588  
 3589  
 3590  
 3591  
 3592  
 3593  
 3594  
 3595  
 3596  
 3597  
 3598  
 3599  
 3600  
 3601  
 3602  
 3603  
 3604  
 3605  
 3606  
 3607  
 3608  
 3609  
 3610  
 3611  
 3612  
 3613  
 3614  
 3615  
 3616  
 3617

```

3618 010310 010046
3619 010310 010146
3620 010312 010246
3621 010314 010346
3622 010316 010446
3623 010320 010546
3624 010322 105737 001234
3625 010324 105737 001664
3626 010330 001132 004666
3627 010332 105037 001245
3628 010336 004737 000006
3629 010342 142737 001245
3630 010350 105711
3631 010352 001415
3632 010354 121127 000054
3633 010360 001403
3634 010362 152737 000002
3635 010370 004737 004666
3636 010374 105711
3637 010376 001403
3638 010400 152737 000004
3639 010406 012705 034634
3640 010412 013701 001710
3641 010416 013703 001674
3642 010422 010504
3643 010424 005024
3644 010426 005303
3645 010430 001375
3646 010432 105037 001235
3647 010436 012704 000001
3648 010442 013703 001720
3649 010446 010100
3650 010450 120421
3651 010452 001064
3652 010454 004437 004704
3653 010460 010632
3654 010462 004472 000000
3655 010466 010632
3656 010470 105721
3657 010472 001376
3658 010474 005204
3659 010476 120437 001243
3660 010502 101017
3661 010504 013737 001674
3662 010512 006337 001702
3663 010516 062737 034634
3664 010524 162737 000012
3665 010532 020537 001702
3666 010536 101743
3667 010540 000423
3668 010542 105737 001664
3669 010546 001010
3670
3671
3672
3673

```

```

; *      TYPDS
; *      TYPE
; *****
CORTE:
MOV R0,-(SP)      ; PUSH R0 ON STACK
MOV R1,-(SP)      ; PUSH R1 ON STACK
MOV R2,-(SP)      ; PUSH R2 ON STACK
MOV R3,-(SP)      ; PUSH R3 ON STACK
MOV R4,-(SP)      ; PUSH R4 ON STACK
MOV R5,-(SP)      ; PUSH R5 ON STACK
TSTB SFEMP        ; TEST IF SOURCE FILE EMPTY
BNE 20$           ; NO SOURCE, EXIT
CLRB CSERR        ; CLEAR COMPILE ERROR
JSR PC,SBSCN      ; SEARCH FOR PARAMETER
BICB #NOCK!BAI    ; I OPFLGS; CLEAR SWITCHES
(R1)              ; TEST IF EITHER PC OR AM GIVEN
BEQ 6$           ; NEITHER GIVEN BRANCH
CMPB (R1),#',     ; TEST IF NO CHECK GIVEN
BEQ 5$           ; NO SKIP TO NEXT PARAM TEST
BISB #NOCK,OPFLGS ; ELSE SET NO CHECK SWITCH
JSR PC,SBSCN      ; BUMP TO NEXT PARAMETER
TSTB (R1)         ; TEST IF THAT NULL
BEQ 6$           ; YES - SKIP
BISB #BAI,OPFLGS  ; SET BAI INHIBIT
MOV #OFIL,RS      ; SET OBJECT POINTER
MOV IBUFPT,R1     ; SET SOURCE FILE POINTER
MOV OBJSIZE,R3    ; SET OBJECT SIZE FOR CLEAR
R5,R4
CLR (R4)+         ; CLEAR OBJECT FILE
DEC R3            ; DEC SIZE CNTR
BNE 1$           ; LOOP UNTIL DONE
CLRB VLD OBJ      ; CLEAR VALID OBJECT INDICATOR
MOV #1,R4         ; SET LINE NUMBER
MOV JSRR4,R3      ; PUT JSR R4 CONSTANT IN R3
MOV R1,R0         ; STORE ADDRESS OF THIS LINE
CMPB R4,(R1)+     ; TEST CORRECT LINE NUMBER IN SOURCE
BNE 21$
JSR R4,ICDEC      ; DECODE INTERACTIVE COMMAND
22$
JSR R4,2(R2)      ; JUMP TO COMMAND COMPILE ROUTINE
23$
TSTB (R1)+       ; ERROR RETURN
BNE 3$           ; TEST FOR NULL
INC R4           ; LOOP
CMPB R4,LNCNT    ; BUMP LINE NUMBER
BHI 4$           ; TEST IF LAST LINE PROCESSED
MOV OBJSIZE,TEMP1 ; DONE, EXIT
ASL TEMP1        ; GET OBJECT FILE SIZE(WORDS)
ADD #OFIL,TEMP1  ; MULTIPLY BY TWO(BYTES)
SUB #12,TEMP1    ; GET LAST ADDRESS OF OFILE
CMP R5,TEMP1     ; ALLOW FOR THE NEXT COMMAND
BLOS 2$         ; CHECK IF ROOM
BR 23$          ; YES - GO GET NEXT LINE
TSTB CSERR       ; NO - TYPE MESSAGE
BNE 35$         ; TEST IF COMPILE ERROR

```

```

3674 010550 013725 001716      MOV      RTSPC,(R5)+      ;MOVE RETURN TO OBJ FILE
3675 010554 105137 001235      COMB      VLD0BJ      ;SET OBJECT VALID
3676 010560 010537 001712      MOV      R5,OFPTR      ;SET OBJECT FILE POINTER
3677 010564 104401 002772      TYPE      ,COMPOK      ;TYPE COMPILE OK MESSAGE
3678
3679 010570      35$:      MOV      (SP)+,R5      ;POP STACK INTO R5
3680 010570 012605      MOV      (SP)+,R4      ;POP STACK INTO R4
3681 010572 012604      MOV      (SP)+,R3      ;POP STACK INTO R3
3682 010574 012603      MOV      (SP)+,R2      ;POP STACK INTO R2
3683 010576 012602      MOV      (SP)+,R1      ;POP STACK INTO R1
3684 010600 012601      MOV      (SP)+,R0      ;POP STACK INTO R0
3685 010602 012600      TST      (R4)+      ;SETUP RETURN
3686 010604 005724      RTS      R4      ;RETURN
3687 010606 000204
3688
3689 010610 104401 002437      23$:      TYPE      ,SCT0BG      ;TYPE MESSAGE
3690 010614 000765      BR      35$
3691 010616 104401 003007      20$:      TYPE      ,NOSRC      ;TYPE NO SOURCE MESSAGE
3692 010622 000762      BR      35$
3693 010624 104401 003023      21$:      TYPE      ,INTERR      ;TYPE INTERNAL ERROR
3694 010630 000757      BR      35$
3695 010632 104401 003053      22$:      TYPE      ,BADCOM      ;TYPE BAD COMMAND ERROR
3696 010636 005046      CLR      -(SP)      ;CLEAR NEXT STACK WORD
3697 010640 112016      MOV      (R0)+,(SP)      ;MOVE LINE NUMBER TO STACK
3698 010642 104405      TYPDS      ;TYPE IT
3699 010644 104401 002510      TYPE      ,EQSGN      ;TYPE EQUAL SIGN
3700 010650 010037 010656      MOV      R0,40$      ;ADDRESS OF REST OF BAD LINE
3701 010654 104401      TYPE      ,TYPE BAD LINE
3702 010656 000000      40$:      .WORD
3703 010660 104401 001171      TYPE      ,SCLRF      ;TYPE CARRIAGE RETURN
3704 010664 152737 000377 001664 26$:      BISR      1377,C$ERR      ;SET ERROR FLAG
3705 010672 000676      BR      35$      ;PROCESS NEXT COMMAND
3706
3707
3708
3709
3710
3711
3712
3713
3714
3715
3716
3717
3718
3719
3720
3721
3722
3723
3724
3725
3726
3727
3728
3729

```

```

;*****
;SMTL DEFERRED COMMAND PROCESSOR ROUTINES
;ALL ROUTINES THAT PROCESS DEFERRED COMMANDS ARE
;CALLED AS FOLLOWS:
;* JSR R4,C$XX WHERE XX IS THE COMMAND MNEMONIC
;* THE RETURN IS:
;* RTS R4 FOR ERROR RETURN
;* RTS R4+2 FOR NORMAL RETURN
;*
;* WHEN THE ROUTINE IS CALLED R1 POINTS TO THE COMMAND FIELD,
;* R3 CONTAINS THE JSR R4 CONSTANT,
;* R5 POINTS TO THE OBJECT FILE WHERE
;* THIS ROUTINE MUST INSERT THE OBJECT CODE, AND R2
;* POINTS TO THE 2ND WORD OF THE TABLE.
;*
;* THE OBJECT CODE WILL CONSIST OF JUMPS TO SPECIFIC SUBROUTINES
;* WHERE THE SPECIFIC COMMAND IS EXECUTED. THESE
;* PROCESSOR ROUTINES WILL INSERT THE JSR R4 (004437 OCTAL)
;* FOLLOWED BY THE ADDRESS OF THE COMMAND EXECUTION
;* ROUTINE. THIS ADDRESS IS TAKEN FROM THE 4TH WORD
;* OF THE INTERACTIVE COMMAND TABLE. THE EXCEPTION TO
;* THIS IS THE SUBSYSTEM FUNCTION INTERACTIVE COMMAND

```

3730  
3731  
3732  
3733  
3734  
3735  
3736  
3737  
3738  
3739  
3740  
3741  
3742  
3743  
3744  
3745  
3746  
3747  
3748  
3749  
3750  
3751  
3752  
3753  
3754  
3755  
3756  
3757  
3758  
3759  
3760  
3761  
3762  
3763  
3764  
3765  
3766  
3767  
3768  
3769  
3770  
3771  
3772  
3773  
3774  
3775  
3776  
3777  
3778  
3779  
3780  
3781  
3782  
3783  
3784  
3785

010674	004737	004666
010700	105711	
010702	001411	
010704	010325	
010706	022222	
010710	011225	
010712	112125	
010714	001376	
010716	032705	000001
010722	001401	
010724	105025	
010726	105741	
010730	005724	
010732	000204	

```

; *WHERE THE ADDRESS OF THE SUBSYSTEM COMMAND
; *EXECUTION ROUTINE IS FOUND IN THE SUBSYSTEM COMMAND
; *TABLE (SCTBL).
;
; *THE COMMAND PROCESSOR ROUTINES ALSO PLACE THE
; *PARAMETERS REQUIRED BY THE EXECUTION ROUTINES IN
; *THE OBJECT FILE. THE NUMBER OF PARAMETERS WILL
; *VARY FROM ONE COMMAND TO ANOTHER BUT IS ALWAYS
; *THE SAME FOR A SPECIFIC COMMAND. THUS THE
; *PROCESSOR ROUTINES KNOW HOW MANY PARAMETERS TO
; *PLACE IN THE OBJECT FILE AND THE EXECUTION
; *ROUTINES KNOW HOW MANY TO RETRIEVE.
; *****
; *****
; SBTTL PRINT MESSAGE PROCESSOR
; ENTRY      JSR      R4,C$PM
; RETURN     RTS      R4      ERROR RETURN
;           RTS      R4+2    NO ERROR RETURN
;
; *OBJECT CODE:
; *      JSR      R4,E$PM
; *      (MESSAGE) WHERE MESSAGE IS UP TO 20 WORDS
;
; *THIS ROUTINE GENERATES THE CODE TO PRINT A MESSAGE OF UP
; *TO 20 WORDS (40 CHARACTERS) ON THE TERMINAL
; *****
C$PM:  JSR      PC,SBSCN      ;BUMP R1 PAST COMMAND FIELD
      TSTB     (R1)          ;TEST IF MESSAGE NULL
      BEQ      2$            ;EXIT IF YES (NO MESSAGE)
      MOV      R3,(R5)+      ;INSERT JSR R4 CONSTANT
      CMP      (R2)+,(R2)+   ;BUMP R2 TO 4TH WORD (EXECUTE ADDR)
      MOV      (R2),(R5)+    ;INSERT EXECUTION ROUTINE ADDRESS
1$:    MOVVB   (R1)+,(R5)+   ;INSERT MESSAGE
      BNE      1$            ;LAST CHARACTER MOVED NULL?
      LOOP    IF NO
      BIT      #1,R5         ;TEST IF R5 IS EVEN.
      BEQ      2$            ;IF NOT, CLEAR NEXT LOCATION
      CLRB     (R5)+         ;AND MAKE R5 EVEN.
2$:    STB      -(R1)         ;DEC R1 TO POINT TO NULL IN SOURCE
      TST      (R4)+         ;SET GOOD RETURN
      RTS      R4            ;RETURN
; *****
; SBTTL REGISTER COMPARE, REGISTER WRITE, AND STATUS COMPARE PROCESSORS
; *ENTRIES:
; *      JSR      R4,C$RC FOR REGISTER COMPARE
; *      JSR      R4,C$RW FOR REGISTER WRITE
; *      JSR      R4,C$SC FOR STATUS COMPARE
;
; *RETURN:
; *      RTS      R4      ERROR RETURN
; *      RTS      R4+2    NORMAL RETURN
;
; *OBJECT CODE:
; *      JSR      R4      E$RC OR E$RW OR E$SC

```

G06

```

3786
3787
3788
3789
3790
3791
3792
3793
3794
3795
3796
3797
3798
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3810
3811
3812
3813
3814
3815
3816
3817
3818
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3830
3831
3832
3833
3834
3835
3836
3837
3838
3839
3840
3841

```

```

;
; REGISTER OR STATUS WORD NUMBER
; EXPECTED VALUE OR VALUE TO BE WRITTEN
; MASK (RC & SC ONLY)
;
; THIS ROUTINE HAS THREE ENTRY POINTS, ONE FOR EACH
; OPERATION. THE LINK TO THE PROPER EXECUTION ROUTINE
; IS GENERATED.
;
; EACH PARAMETER IS TAKEN FROM THE COMMAND LINE IF IT IS PROVIDED.
; IF PROVIDED, IT IS ALSO STORED FOR POSSIBLE LATER USE. IF IT IS NOT
; SUPPLIED, THE VALUE GIVEN THE LAST TIME IT WAS SPECIFIED IS USED.
; NOTE THAT THE PARAMETER STORAGE FOR THE REGISTER COMPARE
; AND THE REGISTER WRITE (REGISTER NUMBER AND VALUE) IS THE SAME. THIS MEANS
; THAT WHEN ONE COMMAND CHANGES THE VALUE IT IS CHANGED FOR
; BOTH COMMANDS.
;
; THE REGISTER ASSIGNMENT IS:
;
; 0 CS1 (CONTROL AND STATUS 1)
; 1 WC (WORD COUNT)
; 2 BA (BUFFER ADDRESS)
; 3 DA (DESIRED TRACK AND SECTOR)
; 4 CS2 (CONTROL AND STATUS 2)
; 5 DS (DRIVE STATUS)
; 6 ER (ERROR REGISTER)
; 7 ASOF (ATTENTION SUMMARY AND OFFSET)
; 10 DC (DESIRED CYLINDER)
; 12 DB (DATA BUFFER)
; 13 MR1 (MAINTENANCE REGISTER 1)
; 14 ECC POSITION REGISTER
; 15 ECC PATTERN REGISTER
; 16 MR2 (MAINTENANCE REG 2)
; 17 MR3 (MAINTENANCE REG 3)
;
; THE STATUS WORD ASSIGNMENT IS:
;
; 0 LINE A WORD 0
; 1 LINE B WORD 0
; 2 LINE A WORD 1
; 3 LINE B WORD 1
; 4 LINE A WORD 2
; 5 LINE B WORD 2
; 6 LINE A WORD 3
; 7 LINE B WORD 3
; *****
;
; 010734 012737 001224 001702 CSRW: MOV #REGNUM,TEMP1 ;STORE RW PARAM BASE
; 010742 012737 000002 001704 MOV #2,TEMP2 ;STORE NUMBER OF PARAM
; 010750 000415 BR CSRCWS ;BRANCH TO COMMON RTE
; 010752 012737 001224 001702 CSRC: MOV #REGNUM,TEMP1 ;STORE RC PARAM BASE
; 010760 012737 000003 001704 MOV #3,TEMP2 ;STORE NUMBER OF PARAM
; 010766 000406 BR CSRCWS ;BRANCH TO COMMON RTE
; 010770 012737 001216 001702 CSSC: MOV #STATRD,TEMP1 ;STORE SC PARAM BASE
; 010776 012737 000003 001704 MOV #3,TEMP2 ;STORE NUMBER OF PARAM
; 011004 011004 CSRCWS: MOV RO,-(SP) ;;PUSH RO ON STACK

```

H06

```

3842 011006 010446      MOV      R4, -(SP)      ;: PUSH R4 ON STACK
3843 011010 013704 001704  MOV      TEMP2, R4      ;: SET NUMBER OF PARAM
3844 011014 013700 001702  MOV      TEMP1, RC      ;: SET PARAM BASE
3845
3846 011020 004737 004666      1$: JSR      PC, SBSCN      ;: BUMP R1 TO NEXT PARAM
3847 011024 105711      TSTB      (P1)      ;: TEST FOR NULL (REMAIN PARAM DEFAULT)
3848 011026 001425      BEQ      3$      ;: BRANCH IF YES
3849 011030 121127 000054      CMPEB  (R1), #',      ;: TEST IF THIS PARAM NULL
3850 011034 001417      BEQ      2$      ;: NULL PARAM, SET FOR NEXT
3851 011036 121127 000067      CMPEB  (R1), #67      ;: TEST 1ST CHAR OF PARAM
3852 011042 101407      BLOS      40$      ;: LESS THAN 7 - BRANCH
3853 011044 010346      MOV      R3, -(SP)      ;: STORE R3
3854 011046 004737 007056      JSR      PC, RGOCDE      ;: GO DECODE REGISTER
3855 011052 011146      21$:      ;: ERROR RETURN
3856 011054 010310      MOV      R3, (R0)      ;: STORE REG NUMBER
3857 011056 012603      MOV      (SP)+, R3      ;: RESTORE R3
3858 011060 000405      BR       2$
3859 011062 010146      40$:      ;: SET UP TO CONVERT PARAM
3860 011064 004737 031160      JSR      PC, OCTBIN      ;: TO BINARY
3861 011070 011140      20$:      ;: STORE PARAM
3862 011072 012610      MOV      (SP)+, (R0)      ;: BUMP R0 TO NEXT PARAM STORE
3863 011074 005720      2$:      ;: DEC PARAM COUNT. TAKE
3864 011076 005304      DEC      R4      ;: ONLY 3 PARAMETERS
3865 011100 001347      BNE      1$      ;: RESET TO BASE FOR PARAM INSERTION
3866 011102 013700 001702      3$:      ;: INSERT JSR R4 CONSTANT
3867 011106 010325      MOV      R3, (R5)+      ;: BUMP R2 TO 4TH WORD
3868 011110 022222      CMP      (R2)+, (R2)+      ;: INSERT EXECUTE ADDRESS
3869 011112 011225      MOV      (R2), (R5)+      ;: INSERT REGISTER OR STATUS WD NULL
3870 011114 012025      MOV      (R0)+, (R5)+      ;: INSERT EXPECTED VALUE
3871 011116 012025      MOV      (R0)+, (R5)+      ;: TEST IF RW(ONLY 2 PARAM)
3872 011120 023727 001704 000002  CMP      TEMP2, #2      ;: IF YES, GET OUT. ELSE
3873 011126 001401      BEQ      4$      ;: INSERT MASK
3874 011130 012025      MOV      (R0)+, (R5)+      ;: RESTORE R4 FOR RETURN
3875 011132 012604      4$:      ;: NORMAL RETURN
3876 011134 005724      TST      (R4)+
3877 011136 000407      BR       25$
3878 011140 104401 002640      20$:      ;: TYPE INVALID OCTAL MESSAGE
3879 011144 000402      BR       22$
3880 011146 104401 003365      21$:      ;: BAD REG SELECTION MESSAGE
3881 011152 012604      22$:      ;: RESTORE R4 FOR RETURN
3882 011154 011404      MOV      (SP)+, R4      ;: BAD RETURN
3883 011156
3884 011156 012600      25$:      ;: POP STACK INTO R0
3885 011160 000204      MOV      (SP)+, R0
3886      RTS      R4      ;: RETURN
3887
3888 ;: *****
3889 ;: SBTTL BUFFER INITIALIZE PROCESSOR
3890 ;: *ENTRY: JSR      R4, CSBI OR JSR R4, CSCBI
3891 ;: *RETURN: RTS      R4      ERROR RETURN
3892 ;: *
3893 ;: *
3894 ;: *OBJECT CODE:
3895 ;: * JSR      R4, ESBI
3896 ;: * <EXECUTION LINE COUNT CONTROL> <PATTERN NAME>
3897 ;: *
3898 ;: *THIS ROUTINE GENERATES THE LINK TO THE BUFFER INITIALIZE EXECUTION

```

3898  
3899  
3900  
3901  
3902  
3903  
3904  
3905  
3906  
3907  
3908  
3909  
3910  
3911  
3912  
3913  
3914 011162  
3915 011162 010046  
3916 011164 004737 004666  
3917 011170 111100  
3918 011172 001002  
3919 011174 113700 001236  
3920 011200 052700 100000  
3921 011204 000401  
3922 011206  
3923 011206 010046  
3924 011210 110037 001236  
3925 011214 010325  
3926 011216 022222  
3927 011220 011225  
3928 011222 010025  
3929 011224 120027 000122  
3930 011230 001015  
3931 011232 012700 001562  
3932 011236 004737 034170  
3933 011242 013720 034266  
3934 011246 013720 034270  
3935 011252 022700 001662  
3936 011256 001367  
3937 011260 105137 001242  
3938 011264  
3939 011264 012600  
3940 011266 005724  
3941 011270 000204  
3942  
3943  
3944  
3945  
3946  
3947  
3948  
3949  
3950  
3951  
3952  
3953

;;\*THE PATTERN NAME IS ENTERED AS THE LOW ORDER BYTE OF THE PARAMETER.  
;;\*BIT 15 IF THE PARAMETER BYTE IS USED TO TELL THE BI EXECUTION  
;;\*IF THIS IS AN INTERNALLY GENERATED BI OR A USER COMMAND. IF  
;;\*USER COMMAND, BIT 15 IS A ONE. THIS INDICATOR CONTROLS LINE  
;;\*COUNT INCREMENT DURING EXECUTION.  
;;\*  
;;\*THIS ROUTINE HAS A SPECIAL ENTRY (CSCBI). THIS ENTRY IS USED  
;;\*WHEN A DATA TRANSFER OPERATION REQUIRES BUFFER INITIALIZATION.  
;;\*THE COMPILER ROUTINE INSERTS A JSR TO THE SPECIAL ENTRY INTO THE  
;;\*OBJECT CODE BEFORE THE DATA TRANSFER JSR (SEE DESCRIPTION OF  
;;\*SUBSYSTEM FUNCTION PROCESSOR).  
;;\*  
;;\*THE RANDOM PATTERN IS GENERATED AND STORED IN THIS ROUTINE IF  
;;\*PATTERN R IS SELECTED.  
;;\*\*\*\*\*

C\$BI:       MOV       RO, -(SP)       ;; PUSH RO ON STACK  
              JSR       PC, SBSCN     ;; BUMP R1 TO NEXT PARAM  
              MOVB     (R1), RO     ;; MOVE PAT NAME INTO RO  
              BNE       1\$       ;; IF NOT NULL, BRANCH, ELSE  
              MOVB     PATSEL, RO    ;; MOVE "A" INTO RO  
1\$:         BIS       #BIT15, RO    ;; SET BIT 15 FOR LINE COUNT CONTROL  
              BR       C\$CBII       ;; BRANCH AROUND SPECIAL ENTRY  
C\$CBII:     MOV       RO, -(SP)     ;; PUSH RO ON STACK  
              MOVB     RO, PATSEL    ;; STORE PATTERN SELECT  
              MOV       R3, (R5)+    ;; INSERT JSR R4 CONSTANT  
              CMP       (R2)+, (R2)+  ;; BUMP R2 TO 4TH WORD TO  
              MOV       (R2), (R5)+   ;; INSERT EXECUTE ADDRESS  
              MOV       RO, (R5)+    ;; INSERT INDEX INTO PATBL  
              CMPB     RO, #'R       ;; RANDOM PATTERN?  
              BNE       3\$       ;; NO - BRANCH OUT  
              MOV       #PATR, RO    ;; GET ADDRESS OF PATR STORE  
2\$:         JSR       PC, \$RAND     ;; GENERATE RANDOM VALUES  
              MOV       \$HINUM, (RO)+  ;; LOAD PATR WITH HI VALUE  
              MOV       \$LONUM, (RO)+  ;; LOAD PATR WITH LO VALUE  
              CMP       #PATR+100, RO  ;; PATTERN FULL?  
              BNE       2\$       ;; NO - DO IT AGAIN  
              COMB     PATRDF       ;; SET PAT R DEFINED SWITCH  
3\$:         MOV       (SP)+, RO     ;; POP STACK INTO RO  
              TST       (R4)+       ;; SET NORMAL RETURN  
              RTS       R4       ;; RETURN

;;\*\*\*\*\*  
;;SBTTL DATA COMPARE PROCESSOR  
;;\*ENTRY:       JSR       R4, C\$DC  
;;\*RETURN:      RTS       R4        ERROR RETURN  
;;\*             RTS       R4+2       NORMAL RETURN  
;;\*  
;;\*OBJECT CODE:  
;;\*       JSR       R4, E\$DC  
;;\*       COMPARE LENGTH (OCTAL)  
;;\*  
;;\*THIS ROUTINE GENERATES THE LINK TO THE DATA COMPARE EXECUTE.  
;;\*THE COMPARE LENGTH PARAMETER IS TAKEN FROM THE COMSZE



3954  
3955  
3956  
3957  
3958  
3959  
3960  
3961 011272  
3962 011272 010046  
3963 011274 004737 004666  
3964 011300 105711  
3965 011302 001406  
3966 011304 010146  
3967 011306 004737 031160  
3968 011312 011336  
3969 011314 012637 001254  
3970 011320 010325  
3971 011322 022222  
3972 011324 011225  
3973 011326 013725 001254  
3974 011332 005724  
3975 011334 000403  
3976 011336 104401 002640  
3977 011342 011404  
3978 011344  
3979 011344 012600  
3980 011346 000204  
3981  
3982  
3983  
3984  
3985  
3986  
3987  
3988  
3989  
3990  
3991  
3992  
3993  
3994  
3995  
3996  
3997  
3998  
3999  
4000  
4001 011350 004737 004666  
4002 011354 105711  
4003 011356 001406  
4004 011360 010146  
4005 011362 004737 031314  
4006 011366 011412  
4007 011370 012637 001252  
4008 011374 010325  
4009 011376 022222

;\*PARAMETER STORAGE LOCATION IF IT IS NOT GIVEN WITH THE COMMAND.  
;\*IF IT IS GIVEN, IT IS STORED IN COMSIZE FOR POSSIBLE LATER  
;\*USE. (COMSIZE WILL ALWAYS BE EITHER THE DATA COMPARE  
;\*LENGTH PARAMETER OR THE WORD COUNT OF THE LAST  
;\*INPUT DATA TRANSFER.)  
;\*\*\*\*\*

CSDC:

```
      MOV      R0,-(SP)      ;: PUSH R0 ON STACK
      JSR      PC,SBSCH      ;: BUMP R1 TO NEXT PARAM
      TSTB     (R1)          ;: TEST IF PARAM NULL
      BEQ      2$            ;: BR IF YES
      MOV      R1,-(SP)      ;: SET UP FOR CONVERT
      JSR      PC,OCTBIN      ;: CONVERT PARAM TO BINARY
      20$      ;: ERROR RETURN
      MOV      (SP)+,COMSIZE ;: STORE COMPARE LENGTH
2$:     MOV      R3,(R5)+      ;: INSERT JSR R4 CONSTANT
      CMP      (R2)+,(R2)+    ;: BUMP R2 TO WORD 4 AT ICTBL
      MOV      (R2),(R5)+     ;: INSERT EXECUTE RTE ADDRESS
      MOV      COMSIZE,(R5)+  ;: INSERT COMPARE LENGTH
      TST      (R4)+          ;: SET FOR NORMAL RETURN
      BR       30$
20$:    TYPE     BADOCT        ;: TYPE BAD DECIMAL MESSAGE
      MOV      (R4),R4        ;: SET FOR ERROR RETURN
30$:    MOV      (SP)+,R0      ;: POP STACK INTO R0
      RTS      R4             ;: RETURN
```

;\*\*\*\*\*  
;SBTTL STALL PROCESSOR  
;ENTRY: JSR R4,C\$ST  
;RETURN: RTS R4 ERROR RETURN  
; RTS R4+2 ;NORMAL RETURN  
;\*  
;\*OBJECT CODE:  
;\* JSR R4,ESST  
;\* STALL DURATION (OCTAL)  
;\*  
;\*THIS ROUTINE GENERATES THE LINK TO THE STALL EXECUTE.  
;\*THE PARAMETER, IF GIVEN IN THE INPUT COMMAND, IS  
;\*DECODED FROM ASCII DECIMAL INTO BINARY AND STORED  
;\*IN THE PARAMETER STORAGE LOCATION "STALL". IT IS THEN  
;\*PLACED IN THE OBJECT CODE. IF THE DELAY IS NOT  
;\*SPECIFIED IN THE COMMAND THE OLD VALUE OF  
;\*"STALL" IS USED.  
;\*\*\*\*\*

```
C$ST:  JSR      PC,SBSCH      ;: BUMP R1 TO NEXT PARAM
      TSTB     (R1)          ;: TEST IF PARAM NULL
      BEQ      1$            ;: BR IF NULL
      MOV      R1,-(SP)      ;: SET UP FOR CONVERT
      JSR      PC,DECBIN      ;: CONVERT PARAM TO BINARY
      20$      ;: ERROR RETURN
      MOV      (SP)+,STALL    ;: STORE STALL VALUE
1$:     MOV      R3,(R5)+      ;: INSERT JSR R4 CONSTANT
      CMP      (R2)+,(R2)+    ;: BUMP R2 TO 4TH WORD ICTBL
```

4010 011400 011225  
4011 011402 013725 001252  
4012 011406 005724  
4013 011410 000403  
4014 011412 104401 002512  
4015 011416 011404  
4016 011420 000204  
4017  
4018  
4019  
4020  
4021  
4022  
4023  
4024  
4025  
4026  
4027  
4028  
4029  
4030 011422 010325  
4031 011424 022222  
4032 011426 011225  
4033 011430 005724  
4034 011432 000204  
4035  
4036  
4037  
4038  
4039  
4040  
4041  
4042  
4043  
4044  
4045 011434 042122  
4046 011436 021712 000005  
4047 011442 042127  
4048 011444 02 762 000005  
4049 011450 041527  
4050 011452 021776 000005  
4051 011456 044127  
4052 011460 021672 000005  
4053 011464 044122  
4054 011466 021646 000004  
4055 011472 045523  
4056 011474 021660 000004  
4057 011500 041503  
4058 011502 021540 000001  
4059 011506 051503  
4060 011510 021524 000001  
4061 011514 041504  
4062 011516 021564 000001  
4063 011522 041522  
4064 011524 021474 000001  
4065 011530 051504

```
MOV (R2), (R5)+ ; INSERT EXECUTE ADDRESS
MOV STALL, (R5)+ ; INSERT STALL PARAMETER
TST (R4)+ ; SET UP NORMAL RETURN
BR 30$ ; BR TO RETURN
20$: TYPE BADDEC ; TYPE BAD DECIMAL MESSAGE
MOV (R4), R4 ; SET UP ERROR RETURN
30$: RTS R4 ; RETURN

;*****
;SBTTL UNIBUS INITIALIZE PROCESSOR
;ENTRY: JSR R4, CSUI
;RETURN: RTS R4+2 NO ERROR RETURN
;
;OBJECT CODE:
; JSR R4, ESUI
;
;THIS ROUTINE GENERATES THE UNIBUS INITIALIZE LINK
;INTO THE OBJECT CODE.
;*****
CSUI: MOV R3, (R5)+ ; INSERT JSR R4 CONSTANT
CMP (R2)+, (R2)+ ; BUMP R2 TO 4TH WORD OF TABLE
MOV (R2), (R5)+ ; INSERT EXECUTE ADDRESS
TST (R4)+ ; NORMAL RETURN
RTS R4 ; RETURN

;*****
;SBTTL SUBCOMMAND TABLE
;THIS TABLE CONTAINS ALL THE SUBSYSTEM COMMANDS. THE TABLE
;FORMAT IS:
; WORD1: SUBCOMMAND MNEMONIC
; WORD2: ADDRESS OF EXECUTION SUBROUTINE FOR THIS COMMAND
; WORD3: THE NUMBER OF PARAMETERS THIS COMMAND REQUIRES
;*****
SCTBL: .EVEN
;ASCII /RD/ ; READ DATA
;WORD ESRD, 5
;ASCII /WD/ ; WRITE DATA
;WORD ESWD, 5
;ASCII /WC/ ; WRITE CHECK
;WORD ESWC, 5
;ASCII /WH/ ; WRITE HEADER
;WORD ESWH, 5
;ASCII /RH/ ; READ HEADER
;WORD ESRH, 4
;ASCII /SK/ ; SEEK
;WORD ESSK, 4
;ASCII /CC/ ; CONTROLLER CLEAR
;WORD ESCC, 1
;ASCII /CS/ ; CLEAR SUBSYSTEM
;WORD ESCS, 1
;ASCII /DC/ ; DRIVE CLEAR
;WORD ESDC, 1
;ASCII /RC/ ; RECALIBRATE
;WORD ESRC, 1
;ASCII /DS/ ; DRIVE SELECT
```

4066 011532 021552 000001  
4067 011536 040520  
4068 011540 021510 000001  
4069 011544 046125  
4070 011546 021576 000001  
4071 011552 051523  
4072 011554 021610 000001  
4073 011560 043117  
4074 011562 021622 000002  
4075 011566 044101  
4076 011570 021634 000004  
4077 011574 000000

.WORD ESDS,1  
.ASCII /PA/ ;PACK ACKNOWLEDGE  
.WORD ESPA,1  
.ASCII /UL/ ;UNLOAD  
.WORD ESUL,1  
.ASCII /SS/ ;START SPINDLE  
.WORD ESSS,1  
.ASCII /OF/ ;OFFSET  
.WORD ESOF,2  
.ASCII /AH/ ;ALL HEADER READ  
.WORD ESAH,4  
.WORD 0 ;NULL TO TERMINATE TABLE

```

*****
SCTL SUBSYSTEM FUNCTION PROCESSOR
*ENTRY:      JSR      R4,CSSF
*RETURN:     RTS      R4      ERROR RETURN
              RTS      R4+2    NORMAL RETURN
*OBJECT CODE:
*      JSR      R4,ESXX :WHERE ESXX IS THE COMMAND EXECUTE ROUTINE
*      OPERATION FLAGS :BIT 1 SET = NO CHECKING
*                      :BIT 2 SET = BUS ADDRESS INCREMENT INHIBIT
*                      :BIT 3 SET = 24 SECTOR FORMAT
*      DRIVE NUMBER   :-1 MEANS USE STORED PARAMETER "DRIVE",
*                      ELSE THIS IS DRIVE TO BE USED
*      CYLINDER NUMBER OR OFFSET :DEPENDING ON COMMAND
*      TRACK NUMBER
*      SECTOR NUMBER
*      WORD COUNT
*THE NUMBER OF PARAMETERS IS VARIABLE DEPENDING ON THE
*SUBSYSTEM COMMAND. THE NUMBER IS SPECIFIED IN THE THIRD
*WORD OF THE SUBSYSTEM COMMAND TABLE (SCTL).
*THIS ROUTINE PROVIDES THE LINKS TO THE VARIOUS EXECUTION ROUTINES FOR
*SUBSYSTEM COMMANDS. IT DOES THIS BY PROVIDING THE PROPER DESTINATION
*ADDRESS FOR THE JSR. THE PARAMETERS REQUIRED BY THE
*EXECUTION ROUTINE IS PLACED IN THE OBJECT FILE IMMEDIATELY
*FOLLOWING THE JSR COMMAND.
*WHEN THE ROUTINE IS ENTERED, THE REGISTERS MUST BE SET
*AS FOLLOWS:
*      R0-NA
*      R1-POINTS TO THE SF MNEMONIC IN THE SOURCE FILE LINE
*      R2-NA
*      R3-CONTAINS JSR R4 CONSTANT
*      R4-NA
*      R5-POINTS TO THE 1ST UNUSED LOCATION IN OBJECT FILE
*IF THE SC PARAMETER IS GIVEN IN THE SOURCE LINE THE SC MNEMONIC
*IS USED TO LOCATE THE PROPER ENTRY IN THE TABLE. IF NONE
*IS FOUND, AN ERROR IS REPORTED. IF A HIT OCCURS, THE
*ADDRESS OF THE SECOND WORD OF THE TABLE IF LOADED INTO R2 AND
*STORED IN SUBCMD AS THE INDICATION OF THE LAST COMMAND
*SPECIFIED. IF THE SC PARAMETER IS NULL, THE STORED VALUE
*IN SUBCMD IS PUT IN R2 WHICH EFFECTIVELY LOCATES THE TABLE

```

4078  
4079  
4080  
4081  
4082  
4083  
4084  
4085  
4086  
4087  
4088  
4089  
4090  
4091  
4092  
4093  
4094  
4095  
4096  
4097  
4098  
4099  
4100  
4101  
4102  
4103  
4104  
4105  
4106  
4107  
4108  
4109  
4110  
4111  
4112  
4113  
4114  
4115  
4116  
4117  
4118  
4119  
4120  
4121

4122  
4123  
4124  
4125  
4126  
4127  
4128  
4129  
4130  
4131  
4132  
4133  
4134  
4135  
4136  
4137  
4138  
4139  
4140  
4141  
4142  
4143  
4144  
4145  
4146  
4147  
4148  
4149  
4150  
4151  
4152  
4153  
4154  
4155  
4156  
4157  
4158  
4159  
4160  
4161  
4162  
4163  
4164  
4165  
4166  
4167  
4168  
4169  
4170  
4171  
4172  
4173  
4174  
4175  
4176  
4177

011576		
011576	010046	
011600	010246	
011602	004737	004666
011606	112137	001702
011612	001403	
011614	111137	001703
011620	001015	
011622	005301	
011624	013702	001214
011630	105137	001666
011634	024227	043117
011640	001002	
011642	105137	001665
011646	005722	
011650	000137	012242
011654	124127	000054
011660	001423	

```

*ENTRY.
*
*THE REMAINING PARAMETERS IN THE SOURCE ARE THEN
*PROCESSED. IF THE PARAMETER IS NULL, THE LAST SPECIFIED
*VALUE FOR THAT PARAMETER IS USED. IF THE PARAMETER
*IS GIVEN, IT IS STORED IN PARAMETER STORAGE AND BECOMES
*THE LAST VALUE GIVEN.
*SEVERAL SPECIAL CASES EXIST. THESE ARE:
*   .PATTERN SELECT SPECIFIED
*   .NULL DRIVE PARAMETER
*   .OFFSET COMMAND
*   .INVALID WORD COUNT
*
*SUPPLYING THE PATTERN SELECT PARAMETER REQUIRES OUTPUT BUFFER
*INITIALIZATION. THIS IS ACCOMPLISHED BY USING THE BUFFER
*INITIALIZE COMMAND PROCESSOR ROUTINE WITH A SPECIAL
*ENTRY (JSR R4, CSCBI). THIS GENERATES A LINK TO THE
*BUFFER INITIALIZE EXECUTION ROUTINE THAT IS IDENTICAL TO
*A LINK GENERATED BY A BI INTERACTIVE COMMAND. THE
*LINK TO BI IN THE OBJECT FILE WILL BE INSERTED BEFORE
*THE LINK TO THE SUBSYSTEM COMMAND.
*
*A NULL DRIVE PARAMETER IS SPECIAL BECAUSE IT REQUIRES
*CONSIDERATION WHEN A COMPILED TEST IS READ FROM PAPER TAPE.
*TO SUPPORT THIS, THE PARAMETER IS SET TO -1 IF THE DRIVE IS
*NOT SPECIFIED IN THE COMMAND. THE COMMAND EXECUTION
*ROUTINE KEYS OFF THIS AND GOES TO THE STORED DRIVE
*PARAMETER FOR THE DRIVE NUMBER.
*
*THE OFFSET COMMAND REQUIRES A UNIQUE PARAMETER (OFFSET)
*IN THE POSITION OFF THE CYLINDER NUMBER. TO PRESERVE
*THE STORED CYLINDER NUMBER THIS MUST BE SPECIAL CASE.
*
*THE WORD COUNT PARAMETER IS CHECKED TO INSURE THE
*BUFFER IS LARGE ENOUGH. IF NOT AN ERROR IS PRINTED.
*****

```

```

CSSF:
      MOV    R0,-(SP)      ;: PUSH R0 ON STACK
      MOV    R2,-(SP)      ;: PUSH R2 ON STACK
      JSR    PC,SBSCN      ;: BUMP R1 TO NEXT PARAM
      MOVB   (R1)+,TEMP1   ;: MOVE 1ST CHAR OF SUB CMND.
      BEQ    1$            ;: BRANCH IF NULL
      MOVB   (R1),TEMP1+1  ;: MOVE 2ND CHAR OF SUB CMND
      BNE    3$            ;: BRANCH IF NOT NULL
1$:    DEC    R1            ;: DECREMENT R1 TO INSURE IT DOESN'T
                                ;: GET PAST THE NULL CHARACTER
      MOV    SUBCMD,R2     ;: GET 2ND WORD ADDR FOR LAST CMND
      COMB   SAMDR         ;: SET SAME DRIVE SWITCH
      CMP    -(R2),#''OF   ;: LAST COMMAND OFFSET?
      BNE    2$            ;: IF NO, BRANCH ELSE
      COMB   OFFLAG        ;: SET OFFSET FLAG
2$:    TST    (R2)+        ;: BUMP R2 TO SECOND WORD
      JMP    41$           ;: GO TO FILL OBJECT
3$:    CMPB   -(R1),#',    ;: TEST IF 1ST CHAR COMMA
      BEQ    7$           ;: BRANCH IF YES. ELSE

```

4178	011662	023727	001702	043117	31\$:	CMP	TEMP1, #''OF	; TEST IF OFFSET
4179	011670	001002				BNE	4\$	; BRANCH IF NO. ELSE
4180	011672	105137	001665			COMB	OFFLAG	; SET OFFSET FLAG
4181	011676	012702	011434		4\$:	MOV	#SCTBL, R2	; LOAD ADDRESS OF SUBCMND TABLE
4182	011702	023722	001702		5\$:	CMP	TEMP1, (R2)+	; TEST IF TABLE ENTRY MATCH
4183	011706	001405				BEQ	6\$	; YES, FOUND A HIT. BRANCH
4184	011710	005742				TST	-(R2)	; TEST IF TABLE ENTRY NULL
4185	011712	001502				BEQ	26\$	; IF YES, NO MATCH IN TABLE, ERROR
4186	011714	062702	000006			ADD	#6, R2	; BUMP R2 TO NEXT TABLE ENTRY
4187	011720	000770				BR	5\$	; BRANCH TO TEST NEXT ENTRY
4188	011722	010237	001214		6\$:	MOV	R2, SUBCMD	; STORE ADDRESS 2ND WORD, LAST CMND
4189	011726	000410				BR	9\$	
4190	011730	013702	001214		7\$:	MOV	SUBCMD, R2	; GET 2ND WORD ADDR LAST CMND
4191	011734	024227	043117			CMP	-(R2), #''OF	; TEST IF LAST CMND OFFSET
4192	011740	001002				BNE	8\$	; IF NO, BRANCH. ELSE
4193	011742	105137	001665			COMB	OFFLAG	; SET OFFLAG
4194	011746	005722			8\$:	TST	(R2)+	; BUMP R2 BACK TO SECOND WORD
4195	011750	005000			9\$:	CLR	R0	; CLR R0 FOR PARAM COUNTING
4196	011752	004737	004666		10\$:	JSR	PC, SBSCN	; BUMP R1 TO NEXT PARAM
4197	011756	121127	000054			CMPB	(R1), #',	; TEST IF COMMA
4198	011762	001005				BNE	11\$	; BR IF NO
4199	011764	005700				TST	R0	; TEST IF DRI'E SELECT PARAM
4200	011766	001111				BNE	18\$	; GO TO BUMP & LOOP
4201	011770	105137	001666			COMB	SAMDR	; SET SAME DRIVE SWITCH
4202	011774	000506				BR	18\$	; GO TO BUMP & LOOP
4203	011776	105711			11\$:	TSTB	(R1)	; TEST PARAM NULL
4204	012000	001005				BNE	12\$	; NOT NULL
4205	012002	005700				TST	R0	; TEST IF DRIVE SELECT PARAM
4206	012004	001116				BNE	41\$	; GO TO FILL OBJECT
4207	012006	105137	001666			COMB	SAMDR	; SET SAME DRIVE FLAG
4208	012012	000513				BR	41\$	; GO TO FILL OBJECT
4209	012014	020027	000016		12\$:	CMP	R0, #16	; TEST IF TOO MANY PARAM
4210	012020	002017				BGE	21\$	; BRANCH TO ERROR
4211	012022	020027	000012			CMP	R0, #12	; PATTERN SELECT PARAM?
4212	012026	001401				BEQ	13\$	; IF YES BRANCH, ELSE
4213	012030	000437				BR	14\$	; GO TO BUMP & LOOP
4214	012032	010046			13\$:	MOV	R0, -(SP)	; SAVE R0
4215	012034	010246				MOV	R2, -(SP)	; SAVE R2
4216	012036	012702	004016			MOV	#C\$CBIS, R2	; SET R2 WITH ADDRESS OF "BI" EXEC ROUT
4217	012042	111100				MOV	(R1), R0	; LOAD R0 WITH PAT SELECT CHAR
4218	012044	004437	011206			JSR	R4, C\$CBI	; JUMP TO SPECIAL BUFFER INITIALIZE
4219								; ENTRY. A JSR R4, BI (PATTERN
4220								; NAME) WILL BE INSERTED INTO
4221								; OBJECT FILE.
4222	012050	000000				.WORD	0	; ERROR RETURN POINT
4223	012052	012602				MOV	(SP)+, R2	; RESTORE R2
4224	012054	012600				MOV	(SP)+, R0	; RESTORE R0
4225	012056	000455				BR	18\$	; GO TO BUMP & LOOP
4226	012060	104401	002616		21\$:	TYPE	IVDPAR	; TYPE INVALID PARAMETERS
4227	012064	000415				BR	26\$	
4228	012066				22\$:			
4229	012066	104401	002640		24\$:	TYPE	BADDOCT	; TYPE BAD OCTAL MESSAGE
4230	012072	000412				BR	26\$	
4231	012074	005726			25\$:	TST	(SP)+	; DUMP BAD VALUE
4232	012076	104401	003101			TYPE	IVDWCT	; TYPE INVALID WORD COUNT
4233	012102	000406				BR	26\$	

4234	012104	104401	003123	27\$:	TYPE	IVDNC	;TYPE INVALID COMMAND IN NO CHECK
4235	012110	000403			BR	26\$	
4236	012112	005726		28\$:	TST	(SP)+	;DUMP BAD DRIVE NUMBER
4237	012114	104401	003146		TYPE	,BADDRV	;TYPE MESSAGE
4238	012120			20\$:			
4239	012120	011404		26\$:	MOV	(P4),R4	;SET UP ERROR RETURN
4240	012122			55\$:			
4241	012122	012602			MOV	(SP)+,R2	;POP STACK INTO R2
4242	012124	012600			MOV	(SP)+,R0	;POP STACK INTO R0
4243	012126	000204			RTS	R4	;RETURN
4244	012130	010146		14\$:	MOV	R1,-(SP)	;PUT ADDRESS OF PARAM ON STACK
4245	012132	004737	031160		JSR	PC,OCTBIN	;CONVERT ASCII OCTAL TO BINARY
4246	012136	012066			24\$		;BAD CONVERT, NON OCTAL NUMBER
4247	012140	020027	000010		CMP	R0,#10	;WORD COUNT PARAM?
4248	012144	001010			BNE	17\$	;IF NO, BRANCH. ELSE
4249	012146	021637	001700		CMP	(SP),MAXWDS	;TEST IF WORD COUNT IF TO BIG
4250	012152	101405			BLOS	17\$	;NO - SKIP
4251	012154	032737	000004 001245		BIT	#BAII,OPFLGS	;ELSE TEST IF BAI INHIBIT SET
4252	012162	001001			BNE	17\$	;YES - BIG WORD COUNT OKAY
4253	012164	000743			BR	25\$	;ELSE GO REPORT ERROR
4254	012166	020027	000002	17\$:	CMP	R0,#2	;CYL NUM OR OFFSET PARAM
4255	012172	001412			BEQ	19\$	;IF YES, BRANCH. ELSE STORE
4256	012174	005700			TST	R0	;DRIVE PARAMETER?
4257	012176	001003			BNE	32\$	;NO - SKIP LEGAL DRIVE TEST
4258	012200	021627	000007		CMP	(SP),#7	;TEST DRIVE PARAMETER
4259	012204	101342			BHI	28\$	;TO BIG, ERROR
4260	012206	012660	001176	32\$:	MOV	(SP)+,DRIVE(R0)	;CONVERTED VALUE IN PROPER STORAGE
4261	012212	062700	000002	18\$:	ADD	#2,R0	;BUMP R0 TO NEXT PARAM NUM
4262	012216	000655			BR	10\$	;LOOP FOR NEXT PARAM
4263	012220	105737	001665	19\$:	TSTB	OFFLAG	;TEST IF OFFSET FLAG SET
4264	012224	001003			BNE	40\$	;IF YES, BRANCH TO STORE OFFSET
4265	012226	012637	001200		MOV	(SP)+,CYLNUM	;STORE CYLINDER NUMBER
4266	012232	000767			BR	18\$	;BR TO BUMP & LOOP
4267	012234	012637	001256	40\$:	MOV	(SP)+,LOFFST	;STORE OFFSET VALUE
4268	012240	000764			BR	18\$	;BRANCH TO BUMP & LOOP
4269	012242	010325		41\$:	MOV	R3,(R5)+	;INSERT JSR R4 CONSTANT
4270	012244	012225			MOV	(R2)+,(R5)+	;INSERT EXECUTE ADDRESS
4271	012246	011200			MOV	(R2),R0	;GET NUM OF PARAM FOR THIS CMND
4272	012250	142737	000010 001245		BICB	#SECT20,OPFLGS	;CLEAR 24 SECTOR FLAG
4273	012256	022737	000026 001212		CMP	#26,FORMAT	;CHECK IF THAT IS CORRECT
4274	012264	001403			BEQ	46\$	;YEP - SKIP
4275	012266	152737	000010 001245		BISB	#SECT20,OPFLGS	;NOPE - SET THE FLAG
4276	012274	105737	001666	46\$:	TSTB	SAMDR	;SAME DRIVE SWITCH SET?
4277	012300	001405			BEQ	42\$	
4278	012302	112725	177777		MOVB	#-1,(R5)+	;INSERT DRIVE NUM OF -1
4279	012306	105037	001666		CLRB	SAMDR	;CLEAR SAME DRIVE SWITCH
4280	012312	000402			BR	43\$	
4281	012314	113725	001176	42\$:	MOVB	DRIVE,(R5)+	;INSERT DRIVE NUM PARAM
4282	012320	113725	001245	43\$:	MOVB	OPFLGS,(R5)+	;INSERT OPERATION FLAGS
4283	012324	005300			DEC	R0	;DEC PARAM NUMBER
4284	012326	001435			BEQ	50\$	;IF NOW ZERO, EXIT
4285	012330	105737	001665		TSTB	OFFLAG	;TEST OFFSET FLAG SET
4286	012334	001405			BEQ	44\$	
4287	012336	013725	001256		MOV	LOFFST,(R5)+	;INSERT OFFSET VALUE
4288	012342	105037	001665		CLRB	OFFLAG	;CLEAR OFFSET FLAG
4289	012346	000402			BR	45\$	

4290	012350	013725	001200		44\$: MOV	CYLNUM,(R5)+	; INSERT CYLINDER NUMBER
4291	012354	005300			45\$: DEC	RO	; DEC PARAM NUMBER
4292	012356	001421			BEQ	50\$	; IF ZERO, EXIT
4293	012360	113725	001202		MOVB	TRKNUM,(R5)+	; INSERT TRACK NUMBER PARAM
4294	012364	005300			DEC	RO	; DEC PARAM NUMBER
4295	012366	001415			BEQ	50\$	; IF ZERO, EXIT
4296	012370	113725	001204		MOVB	SECNUM,(R5)+	; INSERT SECTOR NUM PARAM
4297	012374	005300			DEC	RO	; DEC PARAM NUMBER
4298	012376	001411			BEQ	50\$	; IF ZERO, EXIT
4299	012400	013725	001206		MOV	WDCNT,(R5)+	; INSERT WORD COUNT PARAM
4300	012404	023727	001702	042122	CMP	TEMP1,#"RO	; TEST IF READ DATA COMMAND
4301	012412	001003			BNE	50\$	; NO - BRANCH
4302	012414	013737	001206	001254	MOV	WDCNT,COMSIZE	; ELSE STORE WORD COUNT FOR DATA COMPARE
4303	012422	005724			50\$: TST	(R4)+	; SET UP NO ERROR RETURN
4304	012424	000636			BR	55\$	
4305							
4306							
4307							
4308							
4309							
4310							
4311							
4312							
4313							
4314							
4315							
4316							
4317							
4318							
4319							
4320							
4321							
4322							
4323							
4324							
4325							
4326							
4327							
4328	012426						
4329	012426	010046					
4330	012430	010346					
4331	012430	010546					
4332	012434	004737	004666				
4333	012440	105711					
4334	012442	001526					
4335	012444	121127	000117				
4336	012450	001013					
4337	012452	105737	001235				
4338	012456	001523					
4339	012460	012737	043117	001174			
4340	012466	013703	001712				
4341	012472	162703	034634				
4342	012476	000412					
4343	012500	105737	001234				
4344	012504	001113					
4345	012506	012737	043123	001174			

```

*****
;SBTTL OUTPUT TEST ROUTINE
;ENTRY: JSR R4,OTRTE
;RETURN: RTS R4 ERROR RETURN
; RTS R4+2 NORMAL RETURN
;
;TH1 ROUTINE:
; 1. DETERMINES WHICH FILE IS TO BE PUNCHE (SOURCE OR OBJECT)
; 2. CHECKS IF THAT FILE HAS VALID CODE
; 3. COMPUTES THE SIZE OF THAT FILE
; 4. PUNCHES THE STORED PARAMETERS INCLUDING:
;    A. THE TYPE OF CODE BEING PUNCHE (SOURCE OR OBJECT)
;    B. THE SIZE OF THE FILE
;    C. WHICH OF THE USER DEFINED TEST PATTERNS HAVE
;       BEEN DEFINED
;    D. ALL OF THE STORED TEST SPECIFIC PARAMETERS (DRIVE,
;       CYLINDER, TRACK, SECTOR, ITERATION COUNT, ETC.)
; 5. CHECKS WHICH USER DEFINED OR RANDOM TEST PATTERNS HAS BEEN
;    DEFINED AND PUNCHES THAT PATTERN.
; 6. PUNCHES THE SOURCE OR OBJECT FILE
*****

```

```

OTRTE:
MOV RO,-(SP) ;; PUSH RO ON STACK
MOV R3,-(SP) ;; PUSH R3 ON STACK
MOV R5,-(SP) ;; PUSH R5 ON STACK
JSR PC,SBSCN ;; BUMP R1 TO NEXT PARAMETER
TSTB (R1) ;; TEST IF PARAM NULL
BEQ 20$ ;; IF YES, BRANCH TO ERROR
CMPB (R1),#"0 ;; TEST IF PARAMETER IS "0"
BNE 2$
TSTB VLDOBJ ;; TEST IF VALID OBJECT CODE
BEQ 21$ ;; BRANCH TO EXIT FOR ERROR
MOV #"OF,PUCODE ;; STORE OUTPUT DATA TYPE
MOV OFPTR,R3 ;; GET OFILE POINTER(END OF FILE)
SUB #"OFILE,R3 ;; LENGTH OF FILE IS DIFFERENCE
BR 3$
2$: TSTB SFEMP ;; TEST SOURCE FILE EMPTY
BNE 22$ ;; SFEMP SET, ERROR
MOV #"SF,PUCODE ;; SET PUNCH FILE CODE TO SOURCE

```

```

4346 012514 013703 001246      MOV      SFPTR,R3      ;GET ADDR OF END OF SF+1(1ST EMPTY)
4347 012520 163703 001710      SUB      IBUFPT,R3    ;DIFFERENCE IS SOURCE FILE LENGTH
4348 012524 010337 001250      MOV      R3,PUFLSZ    ;STORE BYTE COUNT IN PU FILE SIZE
4349 012530 012700 001174      MOV      #PUCODE,R0    ;GET ADDR OF START OF STORED PARAM
4350 012534 012703 000066      MOV      #66,R3        ;SET PUNCH COUNT
4351 012540 004437 012760      JSR      R4,PUNCH       ;GO PUNCH STORED PARAM
4352 012544 012742 23$          23$          ;ERROR RETURN
4353 012546 012705 001237      MOV      #PATXDF,R5    ;GET ADDR OF PAT DEFINED SWITCHES
4354 012548 105725 23$          TSTB      (R5)+      ;TEST PAT Y DEFINED
4355 012554 001407 4$           BEQ      4$           ;IF NOT BRANCH, ELSE
4356 012556 012703 000100      MOV      #100,R3       ;SET R3 FOR PUNCH COUNT AND
4357 012562 012700 001262      MOV      #PATX,R0       ;GET ADDR OF PATX
4358 012566 004437 012760      JSR      R4,PUNCH       ;PUNCH PATTERN X
4359 012572 012742 23$          23$          ;
4360 012574 105725 4$           TSTB      (R5)+      ;TEST PAT Y DEFINED
4361 012576 001407 5$           BEQ      5$           ;IF NOT SET, BRANCH. ELSE
4362 012600 012700 001362      MOV      #PATY,R0       ;GET ADDR OF PAT Y AND
4363 012604 012703 000100      MOV      #100,R3       ;SET PUNCH COUNT AND
4364 012610 004437 012760      JSR      R4,PUNCH       ;PUNCH IT
4365 012614 012742 23$          23$          ;
4366 012616 105725 5$           TSTB      (R5)+      ;TEST PAT Z DEFINED
4367 012620 001407 6$           BEQ      6$           ;IF NOT, BRANCH. ELSE
4368 012622 012700 001462      MOV      #PATZ,R0       ;GET ADDR OF PAT Z AND
4369 012626 012703 000100      MOV      #100,R3       ;SET PUNCH COUNT AND
4370 012632 004437 012760      JSR      R4,PUNCH       ;PUNCH PAT Z
4371 012636 012742 23$          23$          ;ERROR RETURN
4372 012640 105715 6$           TSTB      (R5)         ;TEST RANDOM PATTERN DEFINED
4373 012642 001407 7$           BEQ      7$           ;IF NOT, BRANCH. ELSE
4374 012644 012703 000100      MOV      #100,R3       ;SET BYTE COUNT
4375 012650 012700 001562      MOV      #PATR,R0       ;SET ADDRESS OF RANDOM PAT
4376 012654 004437 012760      JSR      R4,PUNCH       ;PUNCH RANDOM PATTERN
4377 012660 012742 23$          23$          ;ERROR RETURN
4378 012662 012700 034634      MOV      #OFIL,R0       ;GET ADDRESS OF OBJ FILE
4379 012666 023727 001174      CMP      PUCODE,#OF    ;TEST IF PUNCH CODE SAYS OBJ
4380 012674 001402 043117      BEQ      8$           ;IF EQ, R0 IS CORRECT. ELSE
4381 012676 013700 001710      MOV      IBUFPT,R0     ;LOAD R0 WITH ADDR OF SOURCE
4382 012702 013703 001250      MOV      PUFLSZ,R3     ;SET PUNCH COUNT
4383 012706 004437 012760      JSR      R4,PUNCH       ;PUNCH FILE
4384 012712 012742 23$          23$          ;ERROR RETURN
4385 012714 005724 TST          (R4)+      ;SET NORMAL RETURN
4386 012716 000414 BR          30$          ;GO TO RETURN
4387 012720 104401 002616      TYPE     IVDPAR        ;TYPE INVALID PARAM
4388 012724 000410 20$:        BR          25$          ;
4389 012726 104401 002756      TYPE     IVDRUN        ;TYPE NO OBJECT CODE
4390 012732 000405 21$:        BR          25$          ;
4391 012734 104401 003007      TYPE     NOSRC         ;TYPE NO SOURCE CODE
4392 012740 000402 22$:        BR          25$          ;
4393 012742 104401 003207      TYPE     PUERR         ;TYPE PUNCH ERROR
4394 012746 011404 23$:        MOV      (R4),R4      ;SET ERROR RETURN
4395 012750 30$:              30$:              ;
4396 012750 012605      MOV      (SP)+,R5    ;POP STACK INTO R5
4397 012752 012603      MOV      (SP)+,R3    ;POP STACK INTO R3
4398 012754 012600      MOV      (SP)+,R0    ;POP STACK INTO R0
4399 012756 000204      RTS      R4          ;RETURN
4400
4401 ;*****
;SBTTL PAPER TAPE PUNCH ROUTINE

```



```

4402      ;*      ENTRY: JSR      R4,PUNCH
4403      ;*      RETURN: RTS      R4      ERROR RETURN
4404      ;*      RTS      R4+2      NORMAL RETURN
4405      ;*THIS ROUTINE WILL PUNCH THE NUMBER OF BYTES INDICATED BY THE VALUE
4406      ;*IN R3 FROM THE MEMORY AREA POINTED TO BY R0.
4407      ;*****
4408
4409      012760 042777 000100 166744 PUNCH: BIC      #000100,2PTPSR      ;RESET INTERRUPT ENABLE
4410      012766 032777 100200 166736 1$: BIT      #100200,2PTPSR      ;TEST FOR ERROR OR READY
4411      012774 001774      BEQ      1$      ;LOOP UNTIL ONE OR THE OTHER
4412      012776 100406      BMI      20$      ;BR TO ERROR EXIT IF ERROR
4413      013000 112077 166730      MOVB     (R0)+,2PTPD8      ;LOAD BYTE FOR PUNCH
4414      013004 005303      DEC      R3      ;DEC BYTE COUNT
4415      013006 001367      BNE      1$      ;LOOP IF MORE TO PUNCH
4416      013010 005724      TST      (R4)+      ;SET GOOD RETURN
4417      013012 000204      RTS      R4      ;RETURN
4418      013014 011404      20$: MOV      (R4),R4      ;SET ERROR RETURN
4419      013016 000204      RTS      R4      ;RETURN
4420      ;*****
4421      ;SBTTL INPUT TEST AND INPUT STRING ROUTINE
4422      ;*      ENTRY: JSR      R4,ITRTE OF ISRTE
4423      ;*      RETURN: RTS      R4      ERROR RETURN
4424      ;*      RTS      R4+2      NORMAL RETURN
4425      ;*
4426      ;*THIS ROUTINE HAS TWO ENTRY POINTS. THE ENTRY AT ISRTE SETS THE
4427      ;*CHAIN FLAG.
4428      ;*
4429      ;*FIRST THE STORED PARAMETERS ARE READ IN. THE USER DEFINED
4430      ;*PATTERN FLAGS ARE CHECKED. IF PATXDF IS NOT 0, 100(8)
4431      ;*BYTES ARE READ AND STORED IN PATX; IF PATYDF IS NOT 0,
4432      ;*100(8) BYTES ARE READ AND STORED IN PATY; ETC FOR PATTERN Z.
4433      ;*THE PUCODE IS THEN CHECKED FOR OF OR SF AND THE NUMBER OF BYTES
4434      ;*INDICATED IN PUFLSZ IS READ AND STORED IN THE APPROPRIATE
4435      ;*FILE. THE OUTPUT BUFFER IS THEN INITIALIZED TO THE PATTERN
4436      ;*INDICATED IN PATSEL.
4437      ;*****
4438
4439      013020 152737 000001 001663 ISRTE: BISB      #1,CHNFLG      ;SET CHAIN FLAG
4440      013026 010046      ITRTE:      MOV      R0,-(SP)      ;PUSH R0 ON STACK
4441      013026 010046      MOV      R3,-(SP)      ;PUSH R3 ON STACK
4442      013030 010346      MOV      R5,-(SP)      ;PUSH R5 ON STACK
4443      013032 010546      MOV      #PUCODE,R0      ;GET ADDRESS OF STORED PARAM
4444      013034 012700 001174      L.V      #66,R3      ;SET NUMBER OF BYTES
4445      013040 012703 000066      JSR      R4,READ      ;READ PARAMETERS FROM TAPE
4446      013044 004437 013344      20$:      ;ERROR RETURN
4447      013050 013322      MOV      #PATXDF,R5      ;GET ADDR OF PATD DEFINED FLAGS
4448      013052 012705 001237      TSTB     (R5)+      ;TEST PAT X DEFINED
4449      013056 105725      BEQ      1$      ;BRANCH IF NOT SET, ELSE
4450      013060 001407      MOV      #100,R3      ;SET BYTE COUNT FOR PAT READ
4451      013062 012703 000100      MOV      #PATX,R0      ;SET FOR READ INTO PAT X
4452      013066 012700 001262      JSR      R4,RDCONT      ;READ IN PAT X
4453      013072 004437 013402      20$:      ;ERROR RETURN
4454      013076 013322      1$: TSTB     (R5)+      ;TEST PAT Y DEFINED
4455      013100 105725      BEQ      2$      ;IF NO, BRANCH. ELSE
4456      013102 001407      MOV      #100,R3      ;SET BYTE COUNT
4457      013104 012703 000100

```

```

4458 013110 012700 001362      MOV      #PATY,R0      ;SET FOR READ INTO PATY
4459 013114 004437 013402      JSR      R4,RDCONT    ;READ IN PAT Y
4460 013120 013322              2$: 20$      ;ERROR RETURN
4461 013122 105725              TSTB     (R5)+    ;TEST PAT 2 DEFINED
4462 013124 001407              BEQ      3$      ;IF NOT, BRANCH. ELSE
4463 013126 012703 000100      MOV      #100,R3    ;SET BYTE COUNT
4464 013132 012700 001462      MOV      #PAT2,R0    ;SET FOR READ INTO PAT 2
4465 013136 004437 013402      JSR      R4,RDCONT    ;READ IN PAT 2
4466 013142 013322              20$      ;ERROR RETURN
4467 013144 105715              3$: TSTB     (R5)    ;TEST RANDOM PATTERN DEFINED
4468 013146 001407              BEQ      4$      ;IF NOT, BRANCH. ELSE
4469 013150 012703 000100      MOV      #100,R3    ;SET BYTE COUNT
4470 013154 012700 001562      MOV      #PATR,R0    ;SET FOR READ RANDOM PAT
4471 013160 004437 013402      JSR      R4,RDCONT    ;READ RANDOM PATTERN
4472 013164 013322              20$      ;ERROR RETURN
4473 013166 152737 000377 001234 4$: BISB     #377,SFEMP ;SET NO SOURCE SWITCH
4474 013174 012700 034634      MOV      #OFILE,R0    ;GET ADDR OF OFILE
4475 013200 023727 001174 043117 CMP      PUCODE,#"OF  ;TEST IF CODE IS OBJECT
4476 013206 001006              BNE      8$      ;BR IF NO, ELSE
4477 013210 013737 001250 001712 MOV      PUFLSZ,OFPTR ;GET FILE SIZE
4478 013216 060037 001712      ADD      R0,OFPTR    ;ADD IN START OF FILE ADDRESS
4479 013222 000415              BR       5$
4480 013224 013700 001710      8$: MOV      IBUFPT,R0    ;CORRECT R0 AND
4481 013230 105037 001234      CLRB     SFEMP      ;SOURCE FILE TO BE READ
4482 013234 105037 001235      CLRB     VLDOBJ     ;CLEAR VLDOBJ CAUSE SOURCE IS NEW
4483 013240 010003              MOV      R0,R3      ;LOAD R3 WITH ADDR OF SOURCE BUFF
4484 013242 063703 001250      ADD      PUFLSZ,R3    ;COMPUTE SIZE OF SOURCE
4485 013246 010337 001246      MOV      R3,SFPTR    ;STORE END OF SOURCE CODE ADDR
4486 013252 105037 001663      CLRB     CHNFLG    ;CLEAR CHAIN FLAG
4487 013256 013703 001250      5$: MOV      PUFLSZ,R3    ;GET SIZE OF DATA FILE
4488 013262 004437 013402      JSR      R4,RDCONT    ;READ FILE
4489 013266 013322              20$      ;ERROR RETURN
4490 013270 113737 001236 013302 MOVB     PATSEL,6$    ;GET PATTERN SELECT INDEX
4491 013276 004437 020536      JSR      R4,ESB1     ;GO TO BUFFER INITIALIZE
4492 013302 000000              6$: .WORD     0
4493 013304 105737 001663      TSTB     CHNFLG    ;TEST CHAIN FLAG
4494 013310 001402              BEQ      7$      ;IF CHAIN FLAG SET GO TO RUN
4495 013312 004437 010036      JSR      R4,RURTE    ;SET NORMAL RETURN
4496 013316 005724              7$: TST      (R4)+
4497 013320 000405              BR       30$
4498 013322 104401 003223      20$: TYPE     PRERR      ;TYPE READER ERROR
4499 013326 105037 001663      CLRB     CHNFLG    ;CLEAR CHAIN FLAG IF READ ERR
4500 013332 011404              MOV      (R4),R4    ;SET ERROR RETURN
4501 013334              30$:
4502 013334 012605              MOV      (SP)+,R5    ;;POP STACK INTO R5
4503 013336 012603              MOV      (SP)+,R3    ;;POP STACK INTO R3
4504 013340 012600              MOV      (SP)+,R0    ;;POP STACK INTO R0
4505 013342 000204              RTS      R4          ;RETURN
4506
4507 ;*****
4508 ;SBTTL PAPER TAPE PUNCH ROUTINE
4509 ;*
4510 ;* ENTRY: JSR R4,READ
4511 ;* RETURN: RTS R4
4512 ;*
4513 ;* THIS ROUTINE READS THE NUMBER OF BYTES INDICATED IN R3 AND
;* STORES THEM IN CONSECUTIVE MEMORY LOCATIONS STARTING

```

```

4514 ;*AT THE ADDRESS IN R0.
4515 ;*****
4516
4517 013344 042777 000100 166354 READ: BIC #000100, @PTRSR ;CLEAR INTERRUPT ENABLE
4518 013352 005277 166350 WSTART: INC @PTRSR ;SET READER GO BIT
4519 013356 032777 100200 166342 WLOOP: BIT #100200, @PTRSR ;TEST IF ERROR OR DONE
4520 013364 001774 BEQ WLOOP ;LOOP
4521 013366 100422 BMI XREAD ;GO TO ERROR EXIT IF ERROR
4522 013370 117710 166334 MOVB @PTRDB, (R0) ;EMPTY DATA BUFFER
4523 013374 001766 BEQ WSTART ;IF ZERO GO READ NEXT
4524 013376 105720 TSTB (R0)+ ;ELSE BUMP R0
4525 013400 005303 DEC R3 ;COUNT THAT CHARACTER
4526 013402 005277 166320 RDCONT: INC @PTRSR ;SET READER GO BIT
4527 013406 032777 100200 166312 RDLOOP: BIT #100200, @PTRSR ;TEST FOR ERROR OR DONE
4528 013414 001774 BEQ RDLOOP ;LOOP IF NO BITS SET
4529 013416 100406 BMI XREAD ;IF BIT IS SET GO TO ERROR
4530 013420 117720 166304 MOVB @PTRDB, (R0)+ ;ELSE STORE BYTE READ
4531 013424 005303 DEC R3 ;DEC BYTE COUNT
4532 013426 001365 BNE RDCONT ;IF NOT ZERO, LOOP
4533 013430 005724 TST (R4)+ ;SET NORMAL RETURN
4534 013432 000204 RTS R4 ;RETURN
4535 013434 011404 XREAD: MOV (R4), R4 ;SET ERROR RETURN
4536 013436 000204 RTS R4 ;RETURN
4537 ;*ENTRY: JSR R4, ESPM
4538 ;* <MESSAGE>
4539 ;*RETURN: RTS R4
4540 ;*THIS ROUTINE PRINTS THE MESSAGE FOUND IN THE OBJECT CODE
4541 ;*FOLLOWING THE JSR TO ESPM. AFTER THE MESSAGE IS PRINTED R4 IS
4542 ;*ADJUSTED TO SKIP OVER THE MESSAGE CONTENTS.
4543 ;*ROUTINES CALLED:
4544 ;* TYPE
4545 ;*
4546 ;*****
4547 ;=13516
4548 013516 005237 001116 ESPM: INC LINNUM ;INCREMENT PSEUDO LINE COUNT
4549 013522 023737 001232 001714 CMP ITCNT, CMTSTR ;CHECK IF FIRST PASS?
4550 013530 001011 BNE 2$ ;NO, SKIP PRINT
4551 013532 005737 001722 TST LPCNT1 ;TEST IF LOOP COUNT HAS COUNT
4552 013536 001006 BNE 2$ ;YES - DON'T PRINT MESSAGE
4553 013540 010437 013546 MOV R4, 1$ ;GET ADDRESS OF MESSAGE
4554 013544 104401 TYPE ;TYPE MESSAGE
4555 013546 000000 1$: .WORD
4556 013550 104401 001171 TYPE $CRLF ;TYPE CARRIAGE RETURN
4557 013554 105724 2$: TSTB (R4)+ ;TEST FOR NULL (END OF MESSAGE)
4558 013556 001376 BNE 2$ ;LOOP IF NOT THE END
4559 013560 105714 TSTB (R4) ;TEST FOR SECOND NULL
4560 013562 001001 BNE 3$ ;IF ONLY ONE NULL, EXIT
4561 013564 105724 TSTB (R4)+ ;BUMP PAST 2ND NULL
4562 ;TWO NULLS MAY BE PRESENT TO
4563 ;INSURE WORD ALIGNMENT
4564 013566 000204 3$: RTS R4 ;RETURN
4565 ;*****
4566 ;SBTTL COMMON DRIVER CALL
4567 ;*ENTRY JSR PC, DRVCAL WITH R5 POINTING TO ACTIVE PARAM BLK
4568 ;*RETURN RTS PC
4569 ;*

```

```

4570      ;*THIS ROUTINE CLEARS THE DONE FLAG, CALLS THE DRIVER WITH
4571      ;*THE ACTIVE PARAMETER BLOCK, AND CALLS THE WATCH DOG
4572      ;*TIMER.
4573      ;*
4574      ;*IT THEN LOOPS WAITING FOR "DONE" (INDICATING AN INTERRUPT
4575      ;*WITH OR WITHOUT ERROR) OR WAITING FOR CONTROLLER READY
4576      ;*(IF THE NO CHECK BIT IS SET IN THE ACTIVE PARAM
4577      ;*BLOCK). WHEN EITHER OCCURS THE ROUTINE RETURNS TO
4578      ;*THE CALLER.
4579      ;*****
4580      DRVCAL: CLRB      DONE      ;CLEAR DONE FLAG
4581      MOV      R5,1$      ;GET PARAM BLOCK ADDRESS
4582      JSR      PC,C.INIT  ;CALL DRIVER
4583      JSR      PC,W.WTCH  ;PARAM BLK ADDRESS PARAMETER
4584      JSR      PC,W.WTCH  ;CALL TIMER
4585      TSTB     DONE      ;TEST IF DONE SET
4586      BNE      3$        ;IF YES, INT OCCURRED, OPComp.
4587      BIT      #NOCHK,P.PRST(R5) ;TEST IF NO CHECK OPERATION
4588      BEQ      2$        ;NO CHECK OFF, MUST GET DONE
4589      MOV      RKBAS,R2   ;GET RKBASE ADDRESS
4590      TSTB     RKCS1(R2)  ;TEST FOR READY
4591      BPL      2$        ;NOT READY, LOOP
4592      RTS      PC         ;RETURN
4593      ;*****
4594      ;SBTTL ERROR FREE RETURN FROM DRIVER
4595      ;ENTRY JSR      PC,ERRFRE WITH R5 POINTING TO ACTIVE PARAM BLK
4596      ;RETURN      RTS      PC
4597      ;*THIS ROUTINE SETS THE DONE FLAG AND RETURNS TO
4598      ;*THE INTERRUPT HANDLER.
4599      ;*****
4600      ERRFRE: COMB     DONE      ;SET DONE FLAG
4601      RTS      PC         ;RETURN
4602      ;*****
4603      ;SBTTL ERROR RETURN FROM DRIVER (DRIVE ERROR)
4604      ;ENTRY JSR      PC,DRVERR WITH R5 POINTING TO ACTIVE PARAM BLK
4605      ;RETURN      RTS      PC
4606      ;*
4607      ;*THIS ROUTINE IS CALLED BY THE DRIVER WHEN AN ERROR
4608      ;*OCCURS IN THE SUBSYSTEM OPERATION OR WHEN A TIMEOUT
4609      ;*OCCURS.
4610      ;*
4611      ;*ALL SUBSYSTEM ERROR REPORTS ARE GENERATED IN THIS ROUTINE.
4612      ;*THE ERROR REPORTS WILL CONTAIN PREVIOUS OPERATION
4613      ;*INFORMATION AS WELL AS INFORMATION ABOUT THE FAILING
4614      ;*OPERATION. FAILURE INFORMATION IS TAKEN FROM THE
4615      ;*ACTIVE PARAM BLOCK. PREVIOUS OPERATION INFORMATION
4616      ;*IS TAKEN FROM THE . CTIVE BLOCK.
4617      ;*
4618      ;*THE WORD IN LOCATION 72 OF THE PARAMETER BLOCK IS
4619      ;*A PRINT CONTROL WORD. THIS WORD IS SET UP WHEN THE
4620      ;*PARAMETER BLOCK IS LOADED BEFORE THE CALL TO THE
4621      ;*DRIVER AND TELLS THE ERROR REPORT WHICH GROUPS OF
4622      ;*DATA IN THE PARAMETER BLOCK IS PERTINENT TO THIS
4623      ;*OPERATION. THIS ALLOWS THE REPORT TO BE CUSTOMIZED
4624      ;*AND SPECIFIC FOR THIS ERROR.
4625      ;*

```

4626  
4627  
4628  
4629  
4630  
4631  
4632  
4633  
4634  
4635  
4636  
4637  
4638  
4639  
4640  
4641  
4642  
4643  
4644  
4645  
4646  
4647  
4648  
4649  
4650  
4651  
4652  
4653  
4654  
4655  
4656  
4657  
4658  
4659  
4660  
4661  
4662  
4663  
4664  
4665  
4666  
4667  
4668  
4669  
4670  
4671  
4672  
4673  
4674  
4675  
4676  
4677  
4678  
4679  
4680  
4681013652 041412 051125 042522  
013660 052116 047440 042520  
013666 040522 044524 047117  
013674 006472 000012  
013700 050012 042522 044526  
013706 052517 020123 050117  
013714 051105 052101 047511  
013722 035116 005015 000  
013727 012 050101 046120  
013734 041511 041101 042514  
013742 051040 043505 035123  
013750 005015 000  
013753 012 040520 040522  
013760 042515 042524 051522  
013766 043440 053111 047105  
013774 006472 000012  
014000 051412 041125 054523  
014006 052123 046505 052040  
014014 046511 047505 052125  
014022 005015 000  
014025 012 052523 051502  
014032 051531 042524 020115  
014040 042504 042524 052103  
014046 042105 042440 051122  
014054 051117 005015 000  
014061 075 046503 042116  
014066 046040 047111 020105  
014074 052516 020115 005015  
014102 000  
014103 104 044522 042526  
014110 000075  
014112 046503 042116 000075  
014120 054503 047114 051104: \*EACH BIT IN THE PRINT CONTROL WORD INDICATES THAT A  
: \*GROUP OF WORDS OR BYTES IS OR IS NOT TO BE INCLUDED IN  
: \*THE REPORT. IF SET THAT GROUP IS INCLUDED. THE PARAMETER  
: \*BLOCK ELEMENTS TO PRINT CONTROL WORD BIT ASSIGNMENTS ARE  
: \*AS FOLLOWS:

BIT	PARAMETER BLOCK ELEMENT
0	WORD 2
1	BYTE 4,5,86
2	WORD 10 & 12
3	WORD 16 & 20
4	WORD 22 & 24
5	WORD 26 & 30
6	WORD 32
	WORD 34 THRU 62

: \*THE ERROR REPORT CAN BE ABBREVIATED TO THE FAILING  
: \*LINE NUMBER, THE DRIVE, AND COMMAND BY SETTING  
: \*SWITCH REGISTER 0. THE REST OF THE ERROR REPORT  
: \*IS SUPPRESSED, THE OTHER ERROR REPORTING CONTROLS  
: \*ARE STANDARD: \*\*\*\*\*  
CUROP: .ASCIZ <12>/CURRENT OPERATION:/(15)<12>

PREVOP: .ASCIZ &lt;12&gt;/PREVIOUS OPERATION:/(15)&lt;12&gt;

APRES: .ASCIZ &lt;12&gt;/APPLICABLE REGS:/(15)&lt;12&gt;

PARMGVN: .ASCIZ &lt;12&gt;/PARAMETERS GIVEN:/(15)&lt;12&gt;

SSTO: .ASCIZ &lt;12&gt;/SUBSYSTEM TIMEOUT/(15)&lt;12&gt;

SDETER: .ASCIZ &lt;12&gt;/SUBSYSTEM DETECTED ERROR/(15)&lt;12&gt;

PLINE: .ASCIZ /=CMND LINE NUM /(15)&lt;12&gt;

FDRIVE: .ASCIZ /DRIVE=/  
FCMND: .ASCIZ /CMND=/  
FARM1: .ASCIZ /CYLNR SECTOR TRACK OFFSET /

	4682	014126	020040	042523	052103						
	4683	014134	051117	020040	051124						
	4684	014142	041501	020113	020040						
	4685	014150	043117	051506	052105						
	4686	014156	020040	000							
	4E 7	014161	102	044101	020040	FARM2:	.ASCIZ	/BAH	BAL	WDC/	
	4688	014166	020040	041040	046101						
	4689	014174	020040	020040	053440						
	4690	014202	041504	000							
	4691	014205	103	030523	020040	FARM3:	.ASCIZ	/CS1	CS2	/	
	4692	014212	020040	041440	031123						
	4693	014220	020040	020040	000040						
	4694	014226	041527	020040	020040	FARM4:	.ASCIZ	/WC	BA	/	
	4695	014234	020040	040502	020040						
	4696	014242	020040	020040	000						
	4697	014247	104	020101	020040	FARM5:	.ASCIZ	/DA	DC	/	
	4698	014254	020040	042040	020103						
	4699	014262	020040	020040	000040						
	4700	014270	051501	043117	000	FARM6:	.ASCIZ	/ASOF/			
	4701	014275	105	020122	020040	FARM7:	.ASCIZ	/ER	DS	A0	BO/<15><12>
	4702	014302	020040	042040	020123						
	4703	014310	020040	020040	040440						
	4704	014316	020060	020040	020040						
	4705	014324	041040	006460	000012						
	4706	014332	030501	020040	020040	FARMB:	.ASCII	&A1	B1	A2	B2 A3
	4707	014340	020040	030502	020040						
	4708	014346	020040	020040	031101						
	4709	014354	020040	020040	020040						
	4710	014362	031102	020040	020040						
	4711	014370	020040	031501	020040						
	4712	014376	020040	020040	031502						
	4713	014404	020040	020040	020040						
	4714	014412	041505	027503	047520		.ASCIZ	&ECC/POS ECC/PAT&<15><12>			
	4715	014420	020123	041505	027503						
	4716	014426	040520	006524	000012						
	4717	014434	042012	052101	020101	FDATC:	.ASCIZ	<12>/DATA COMPARE ERR ON WORD /			
	4718	014442	047503	050115	051101						
	4719	014450	020105	051105	020122						
	4720	014456	047117	053440	051117						
	4721	014464	020104	000							
	4722	014467	075	047507	042117	GDDAT:	.ASCIZ	/=GOOD WORD/<15><12>			
	4723	014474	053440	051117	006504						
	4724	014502	000012								
	4725	014504	041075	042101	053440	BDDAT:	.ASCIZ	/=BAD WORD/<15><12>			
	4726	014512	051117	006504	000012						
	4727	014520	051412	040524	052524	SCERR:	.ASCIZ	<12>/STATUS COMPARE ERROR/<15&gt			

K07

CZR6RCO RK611/06 USR DEFINED  
CZR6RC.P11 02-DEC-77 11:07MACY11 30(1046) 02-DEC-77 11:59 PAGE 89  
ERROR RETURN FROM DRIVER (DRIVE ERROR)

SEQ 0088

4738	014614	046517	040520	042522	
4739	014622	042440	051122	051117	
4740	014630	005015	000		
4741	014633	075	042522	020107	REGLAB: .ASCIZ /=REG NUM/<15><12>
4742	014640	052516	006515	000012	
4743	014646	051104	053111	020105	HARDER: .ASCIZ /DRIVE HAS HARD ERROR/<15><12>
4744	014654	040510	020123	040510	
4745	014662	042122	042440	051122	
4746	014670	051117	005015	000	
4747	014675	123	040524	052524	SCNCLR: .ASCIZ /STATUS CHANGE NOT CLEARED/<15><12>
4748	014702	020123	044103	047101	
4749	014710	042507	047040	052117	
4750	014716	041440	042514	051101	
4751	014724	042105	005015	000	
4752	014731	116	020117	051104	DSCNOT: .ASCIZ /NO DRIVE STATUS CHANGE/<15><12>
4753	014736	053111	020105	052123	
4754	014744	052101	051525	041440	
4755	014752	040510	043516	006505	
4756	014760	000012			
4757	014762	042523	055111	042105	STILSZ: .ASCIZ /SEIZED OTHER PORT/<15><12>
4758	014770	047440	044124	051105	
4759	014776	050040	051117	006524	
4760	015004	000012			
4761	015006	047125	054105	042520	BADATT: .ASCIZ /UNEXPECTED ATTENTION/<15><12>
4762	015014	052103	042105	040440	
4763	015022	052124	047105	044524	
4764	015030	047117	005015	000	
4765		015036			
4766	015036				DRVERR: .EVEN
4767	015036	010046			MOV R0,-(SP) ;PUSH R0 ON STACK
4768	015040	010146			MOV R1,-(SP) ;PUSH R1 ON STACK
4769	015042	010246			MOV R2,-(SP) ;PUSH R2 ON STACK
4770	015044	010346			MOV R3,-(SP) ;PUSH R3 ON STACK
4771	015046	010446			MOV R4,-(SP) ;PUSH R4 ON STACK
4772	015050	105037	001671		CLRB RPSWIT ;CLEAR REPORT PASS SWITCH
4773	015054	152737	000377	001667	BISB #377,DONE ;SET DONE FLAG
4774	015062	032777	002000	164050	BIT #SW10,JSWR ;BELL ON ERROR?
4775	015070	001402			BEQ 1\$ ;NO - BRANCH
4776	015072	104401	001164		TYPE \$BELL ;RING BELL
4777	015076	032777	020000	164034	1\$: BIT #SW13,JSWR ;INHIBIT ERROR REPORT?
4778	015104	001407			BEQ 56\$
4779	015106	013702	023554		MOV RKBAS,R2 ;GET BASE
4780	015112	012762	100000	000000	MOV #CCLR,RKCS1(R2) ;RESET IE AND DO CONT CLEAR
4781	015120	000137	015524		JMP 36\$ ;JUMP TO EXIT
4782	015124	105737	023452		56\$: TSTB CEFLG ;TEST CONTROLLER ERROR FLAG
4783	015130	001052			BNE 3\$ ;SKIP TO TYPE OUT
4784	015132	032765	000010	000014	BIT #UEXATT,P.PRST(R5) ;TEST UNEXPECTED ATTENTION
4785	015140	001405			BEQ 55\$
4786	015142	104401	015006		TYPE ,BADATT ;UNEXPECTED ATTENTION REPORT
4787	015146	012765	000044	000064	MOV #44,PRCON(R5) ;SET PRINT CONTROL TO LIMIT REPORT
4788	015154	032765	000020	000014	55\$: BIT #DRVHARD,P.PRST(R5) ;TEST DRIVE HARD ERROR
4789	015162	001402			BEQ 50\$ ;NO - BRANCH
4790	015164	104401	014646		TYPE ,HARDER ;HARD ERROR REPORT
4791	015170	032765	000040	000014	50\$: BIT #DRVDSC,P.PRST(R5) ;TEST STATUS NOT CLEARED ERROR
4792	015176	001402			BEQ 51\$ ;NO - BRANCH
4793	015200	104401	014675		TYPE ,SCNCLR ;REPORT STATUS NOT CLEARED ERROR

4794	015204	032765	004000	000014	51\$:	BIT	#NODSC,P.PRST(R5)	; TEST NO STATUS CHANGE
4795	015212	001402				BEQ	52\$	; NO - BRANCH
4796	015214	104401	014731			TYPE	,DSCNOT	; REPORT NO STATUS CHANGE ERROR
4797	015220	032765	010000	000014	52\$:	BIT	#DRVSZD,P.PRST(R5)	; TEST DRIVE SEIZED
4798	015226	001402				BEQ	54\$	; NO - BRANCH
4799	015230	104401	014762			TYPE	,STILSZ	; REPORT STILL SEIZED ERROR
4800	015234	032765	010100	000014	54\$:	BIT	#CMCTO:DRVSZD,P.PRST(R5)	; TEST TIMEOUT
4801	015242	001403				BEQ	2\$	; BR IF NO
4802	015244	104401	014000			TYPE	,SSTO	; TYPE SUBSYSTEM TIMEOUT
4803	015250	000402				BR	3\$	
4804	015252	104401	014025		2\$:	TYPE	,SDETER	; TYPE SUBSYSTEM DET. ERR.
4805	015256	005002			3\$:	CLR	R2	; CLEAR R2 (BYTE PRINT SWITCH)
4806	015260	013746	001116			MOV	LINUM,-(SP)	; PUT LINE NUMBER ON STACK
4807	015264	104405				TYPDS		; PRINT IT
4808	015266	104401	014061			TYPE	,PLINE	; TYPE LINE NUM MESSAGE
4809	015272	010500			19\$:	MOV	R5,R0	; SET R0 TO BEGINNING OF PARAM BLK
4810	015274	104401	014103			TYPE	,FORIVE	; TYPE "DRIVE"
4811	015300	005102				COM	R2	; SET R2 FOR BYTE OPERATION
4812	015302	104414				FPRINT		; PRINT FIRST LINE (DRIVE NUM)
4813	015304	104401	001171			TYPE	,SCRLF	
4814	015310	104401	014112			TYPE	,FCMND	; TYPE "COMMAND"
4815	015314	104414				FPRINT		; PRINT 2ND LINE (COMMAND)
4816	015316	104401	001171			TYPE	,SCRLF	
4817	015322	032765	000100	000014		BIT	#CMCTO,P.PRST(R5)	; TEST TIMEOUT
4818	015330	001402				BEQ	25\$	
4819	015332	004737	016120			JSR	PC_READRG	; GO READ REGISTERS
4820	015336	105737	001671		25\$:	TSTB	RPSWIT	; TEST 2ND PASS
4821	015342	001006				BNE	17\$	; YES - SKIP (SHORT FORM TEST)
4822	015344	032777	000001	163566		BIT	#SWO,2SWR	; TEST IF SHORT REPORT SWITCH SET
4823	015352	001062				BNE	30\$	; YES - EXIT
4824	015354	104401	013652			TYPE	,CUROP	; TYPE "CURRENT OPERATION"
4825	015360	016503	000064		17\$:	MOV	PRCON(R5),R3	; GET PRINT CONTROL WORD
4826	015364	032703	000003			BIT	#3,R3	; TEST IF GROUP 0 OR 1 TO BE PRINTED
4827	015370	001001				BNE	4\$	; YES SKIP TO PRINTING
4828	015372	000443				BR	6\$	; SKIP TO NEXT REPORT GROUP TEST
4829	015374	104401	013753		4\$:	TYPE	,PARMGVN	; TYPE HEADING "PARAM GIVEN"
4830	015400	032703	000001			BIT	#BIT0,R3	; TEST FOR GROUP 0
4831	015404	001402				BEQ	46\$	
4832	015406	104401	014120			TYPE	,FPARM1	; TYPE GROUP 0 HEADINGS
4833	015412	032703	000002		46\$:	BIT	#BIT1,R3	; TEST FOR GROUP 1
4834	015416	001402				BEQ	45\$	
4835	015420	104401	014161			TYPE	,FPARM2	; TYPE GROUP 1 HEADINGS
4836	015424	104401	001171		45\$:	TYPE	,SCRLF	
4837	015430	032703	000001			BIT	#BIT0,R3	; TEST IF MUST PRINT GROUP 0
4838	015434	001406				BEQ	5\$	
4839	015436	005002				CLR	R2	; CLEAR BYTE REPORT CONTROL
4840	015440	104414				FPRINT		; PRINT LINE (CYLINDER)
4841	015442	005102				COM	R2	; SET BYTE CONTROL
4842	015444	104414				FPRINT		; PRINT LINE (SECTOR)
4843	015446	104414				FPRINT		; PRINT LINE (TRACK)
4844	015450	104414				FPRINT		; PRINT LINE (OFFSET)
4845	015452	010500			5\$:	MOV	R5,R0	; SET TABLE POINTER FOR GROUP 1
4846	015454	062700	000007			ADD	#P.BAHI,R0	; SET ADDRESS
4847	015460	032703	000002			BIT	#BIT1,R3	; TEST IF MUST PRINT GROUP 1
4848	015464	001406				BEQ	6\$	; NO, SKIP TO NEXT GROUP TEST
4849	015466	052702	000001			BIS	#1,R2	; MAKE SURE R2 SAYS BYTE



4850	015472	104414		FPRINT		;PRINT LINE (BUS ADD HI)
4851	015474	005002		CLR	R2	;R2 SAYS WORD
4852	015476	104414		FPRINT		;PRINT LINE (BUS ADD LO)
4853	015500	104414		FPRINT		;PRINT LINE (WORD COUNT)
4854	015502	104401	001171	TYPE	, \$CRLF	
4855	015506	104401	001172	TYPE	, \$LF	
4856	015512	105737	001671	TSTB	APSWIT	;TEST 2ND PASS
4857	015516	001412		BEQ	31\$	;NO - BRANCH AROUND EXIT
4858	015520	004737	016670	JSR	PC, PRLPCT	;PRINT LOOP COUNT
4859	015524					
4860	015524	012604		MOV	(SP)+, R4	;POP STACK INTO R4
4861	015526	012603		MOV	(SP)+, R3	;POP STACK INTO R3
4862	015530	012602		MOV	(SP)+, R2	;POP STACK INTO R2
4863	015532	012601		MOV	(SP)+, R1	;POP STACK INTO R1
4864	015534	012600		MOV	(SP)+, R0	;POP STACK INTO R0
4865	015536	004737	016622	JSR	PC, SWCONT	;GO TO SWR LOOP CONTROL
4866	015542	000207		RTS	PC	;RETURN
4867	015544	104401	013727	TYPE	, APRES	;TYPE "APPLICABLE REGS"
4868	015550	005002		CLR	R2	;MAKE SURE R2 SAYS WORD
4869	015552	032703	000004	BIT	#BIT2, R3	;TEST IF MUST PRINT GROUP 2
4870	015556	001402		BEQ	40\$	
4871	015560	104401	014205	TYPE	, FPARM3	;TYPE GROUP 2 HEADINGS
4872	015564	032703	000010	BIT	#BIT3, R3	;TEST FOR GROUP 3
4873	015570	001402		BEQ	41\$	
4874	015572	104401	014226	TYPE	, FPARM4	;TYPE GROUP 3 HEADINGS
4875	015576	032703	000020	BIT	#BIT4, R3	;TEST FOR GROUP 4
4876	015602	001402		BEQ	42\$	
4877	015604	104401	014247	TYPE	, FPARM5	;TYPE GROUP 4 HEADINGS
4878	015610	032703	000040	BIT	#BIT5, R3	;TEST FOR GROUP 5
4879	015614	001402		BEQ	43\$	
4880	015616	104401	014270	TYPE	, FPARM6	;TYPE GROUP 5 HEADINGS
4881	015622	104401	001171	TYPE	, \$CRLF	
4882	015626	032703	000004	BIT	#BIT2, R3	;TEST IF MUST PRINT GROUP 2
4883	015632	001405		BEQ	7\$	;NO - BRANCH
4884	015634	010500		MOV	R5, R0	;SET TABLE PTR TO GROUP 2 VALUES
4885	015636	062700	000016	ADD	#P.CS1, R0	;SET ADDRESS
4886	015642	104414		FPRINT		;PRINT LINE (CS1)
4887	015644	104414		FPRINT		;PRINT LINE (CS2)
4888	015646	032703	000010	BIT	#BIT3, P3	;TEST IF MUST PRINT GROUP 3
4889	015652	001405		BEQ	8\$	;NO - BRANCH
4890	015654	010500		MOV	R5, R0	;SET TABLE PTR TO GROUP 3 VALUES
4891	015656	062700	000022	ADD	#P.WCR, R0	;SET ADDRESS
4892	015662	104414		FPRINT		;PRINT LINE (WORD COUNT)
4893	015664	104414		FPRINT		;PRINT LINE (BUFFER ADD)
4894	015666	032703	000020	BIT	#BIT4, R3	;TEST IF MUST PRINT GROUP 4
4895	015672	001405		BEQ	9\$	;NO - BRANCH
4896	015674	010500		MOV	R5, R0	;SET TABLE PTR TO GROUP 4 VALUES
4897	015676	062700	000026	ADD	#P.DTS, R0	;SET ADDRESS
4898	015702	104414		FPRINT		;PRINT LINE (DESIRED TRACK/SEC)
4899	015704	104414		FPRINT		;PRINT LINE (DESIRED CYLINDER)
4900	015706	032703	000040	BIT	#BIT5, R3	;TEST IF MUST PRINT GROUP 5
4901	015712	001404		BEQ	10\$	;NO - BR
4902	015714	010500		MOV	R5, R0	;SET PTR TO GROUP 5 VALUE
4903	015716	062700	000032	ADD	#P.ASOF, R0	;SET ADDRESS
4904	015722	104414		FPRINT		;PRINT LINE (ATTN SUM & OFFSET)
4905	015724	104401	001171	TYPE	, \$CRLF	

```
4906 015730 104401 001172      TYPE      SLF
4907 015734 032703 000100      BIT      BIT6,R3      ;TEST IF MUST PRINT GROUP 6
4908 015740 001427              BEQ      15$      ;NO - BRANCH
4909 015742 012704 000004      MOV      #4,R4      ;SET COL COUNT TO 4
4910 015746 104401 014275      TYPE      FPARM7      ;TYPE GROUP 6 HEADINGS
4911 015752 010500              MOV      R5,R0      ;SET TABLE POINTER TO GROUP 6 VALUES
4912 015754 062700 000034      ADD      #P.ER,R0      ;SET ADDRESS
4913 015760 104414              FPRINT      ;PRINT COLUMN
4914 015762 005304              DEC      R4      ;DEC COUNT, LOOP UNTIL ZERO
4915 015764 001375              BNE      11$
4916 015766 104401 001171      TYPE      $CRLF
4917 015772 012704 000010      MOV      #10,R4      ;SET COL COUNT TO 10
4918 015776 104401 014332      TYPE      FPARM8      ;TYPE GROUP 7 HEADINGS
4919 016002 104414              FPRINT      ;PRINT COL
4920 016004 005304              DEC      R4      ;DEC COUNT, LOOP UNTIL ZERO
4921 016006 001375              BNE      12$
4922 016010 104401 001171      TYPE      $CRLF
4923 016014 104401 001172      TYPE      SLF
4924 016020 023727 001116 000001 15$: CMP      LINNUM,#1      ;TEST IF FIRST LINE IS ERROR
4925 016026 101421              BLOS      18$      ;IF YES BYPASS 2ND PASS (PREV OP)
4926 016030 105737 001671      TSTB      RPSWIT      ;TEST IF SECOND PASS IN REPORT
4927 016034 001016              BNE      18$      ;YES - EXIT
4928 016036 105137 001671      COMB      RPSWIT      ;SET SWITCH
4929 016042 012705 002142      MOV      #PARMO,R5      ;SET R5 TO PARAM 0
4930 016046 105737 001670      TSTB      PBSW      ;WAS THAT RIGHT?
4931 016052 001002              BNE      16$      ;YES - SKIP
4932 016054 012705 002230      MOV      #PARM1,R5      ;NO - SET TO PARAM 1
4933 016060 104401 013700 16$: TYPE      PREVOP      ;TYPE "PREVIOUS OP" HDR
4934 016064 005002              CLR      R2
4935 016066 000137 015272      JMP      19$      ;GO PRINT PREVIOUS OPS.
4936 016072 105037 001671 18$: CLRB      RPSWIT      ;RESET 2ND PASS SWITCH
4937 016076 012705 002142      MOV      #PARMO,R5      ;SET R5 TO PARAM 0
4938 016102 105737 001670      TSTB      PBSW      ;TEST PB SWITCH. WAS THAT RIGHT?
4939 016106 001402              BEQ      20$      ;YES - SKIP
4940 016110 012705 002230      MOV      #PARM1,R5      ;NO - SET R5 TO PARAM 1
4941 016114 000137 015520 20$: JMP      30$      ;GO TO EXIT
4942 *****
4943 *****
4944 *****
4945 SBTTL READ RK611 AND DRIVE STATUS REGISTER ROUTINE
4946 *ENTRY:      JSR      PC,READRG
4947 *RETURN:      RTS      PC
4948 *
4949 *THIS ROUTINE READS ALL THE RK611 REGISTERS AND ENTERS THEM INTO THE
4950 *PARAMETER BLOCK. IT THEN TRIES TO READ THE DRIVE STATUS REGISTERS
4951 *AND IF SUCCESSFUL, PUTS THEM INTO THE BLOCK. IF AND ERROR OCCURS
4952 *WHILE READING DRIVE REGISTERS, A MESSAGE IS PRINTED TO ALERT THE
4953 *USER THAT THE DRIVE STATUS IS NOT VALID.
4954 *****
4955 READRG:      MOV      R0,-(SP)      ;PUSH R0 ON STACK
4956              MOV      R2,-(SP)      ;PUSH R2 ON STACK
4957              MOV      R3,-(SP)      ;PUSH R3 ON STACK
4958              MOV      #PARMO,R5      ;GET ADDRESS OF PARMO
4959              TSTB      PBSW      ;IS PARMO PRESENTLY SELECTED?
4960              BEQ      8$      ;YES - SKIP
4961              MOV      #PARM1,R5      ;ELSE GET ADDRESS OF PARM 1
```

4962	016144	010500		8\$:	MOV	R5,R0	;GET PARAM BLOCK ADDRESS	
4963	016146	013702	023554		MOV	RKBAS,R2	;GET RK BASE ADDRESS	
4964	016152	062700	000016		ADD	#P,CS1,R0	;BUMP BASE TO REG RETURN ENTRIES	
4965	016156	016220	000000		MOV	RKCS1(R2),(R0)+	;GET CS1	
4966	016162	016220	000010		MOV	RKCS2(R2),(R0)+	;GET CS2	
4967	016166	016220	000002		MOV	RKWC(R2),(R0)+	;GET WORD COUNT	
4968	016172	016220	000004		MOV	RKBA(R2),(R0)+	;GET BUFFER ADDRESS	
4969	016176	016220	000006		MOV	RKDA(R2),(R0)+	;GET DESIRED ADDRESS	
4970	016202	016220	000020		MOV	RKDC(R2),(R0)+	;GET DESIRED CYLINDER	
4971	016206	016220	000016		MOV	RKASOF(R2),(R0)+	;GET ATTENTION	
4972	016212	016220	000014		MOV	RKER(R2),(R0)+	;GET ERROR REG	
4973	016216	016220	000012		MOV	RKDS(R2),(R0)+	;RET STATUS REG	
4974	016222	005065	000014		CLR	P,PRST(R5)	;CLEAR PROG STATUS	
4975	016226	012737	016416	023564	MOV	#6\$,A,ABNL	;SET TIME OUT RETURN	
4976	016234	016203	000010		MOV	RKCS2(R2),R3	;GET CS2 TO STORE DRIVE NUM	
4977	016240	012762	100000	000000	MOV	#CCLR,RKCS1(R2)	;SET CLEAR CONTROLLER	
4978	016246	042703	177770		BIC	#177770,R3	;CLEAR ALL BUT DRIVE NUMBER	
4979	016252	050362	000010		BIS	R3,RKCS2(R2)	;INSERT DRIVE NUMBER	
4980	016256	005003			CLR	R3		
4981	016260	110362	000026		3\$:	MOV	R3,RKMR1(R2)	;SET DESIRED STATUS BYTE SELECT
4982	016264	012762	000001	000000	MOV	#1,RKCS1(R2)	;DO DRIVE SELECT	
4983	016272	013737	023600	023632	MOV	W,SEC,W,DRV	;SET UP TIMEOUT DURATION	
4984	016300	152737	000377	023616	BISB	#377,W,TIME	;GIVE WATCH DOG SOMETHING TO TIME	
4985	016306	032762	000200	000000	1\$:	BIT	#RDY,RKCS1(R2)	;TEST READY
4986	016314	001007			2\$:	BNE	2\$	
4987	016316	004737	023634		JSR	PC,W,WTCH	;CALL HIM	
4988	016322	032765	000100	000014	BIT	#CHDT0,P,PRST(R5)	;DID HE TIME OUT?	
4989	016330	001033			BNE	5\$	;YES- TYPE WARNING AND EXIT	
4990	016332	000765			1\$:	BR	1\$	;NO - GO TEST READY AGAIN
4991	016334	00562	000000		2\$:	TST	RKCS1(R2)	;TEST CS1 FOR ERROR
4992	016340	100427			BMI	5\$	;YES - SKIP	
4993	016342	032762	000001	000012	BIT	#DRA,RKDS(R2)	;TEST IF DRIVE AVAILABLE	
4994	016350	001423			BEQ	5\$	;NO - PRINT WARNING	
4995	016352	016220	000034		MOV	RKMR2(R2),(R0)+	;STORE BYTE A	
4996	016356	016220	000036		MOV	RKMR3(R2),(R0)+	;STORE BYTE B	
4997	016362	020327	000003		CMP	R3,#3	;TEST IF WORD 3 RETRIEVED	
4998	016366	001402			BEQ	4\$	;YES - EXIT	
4999	016370	005203			INC	R3	;BUMP R3 TO NEXT WORD NUMBER	
5000	016372	000732			BR	3\$	;GO GET IT	
5001	016374	012765	000177	000064	4\$:	MOV	#177,PRSTCON(R5)	;SET PRINT CONTROL
5002	016402	012737	015036	023564	7\$:	MOV	#DRVERR,A,ABNL	;RESTORE DRIVER RETURN
5003	016410	012603			MOV	(SP)+,R3	;POP STACK INTO R3	
5004	016412	012602			MOV	(SP)+,R2	;POP STACK INTO R2	
5005	016414	012600			MOV	(SP)+,R0	;POP STACK INTO R0	
5006	016416	000207			6\$:	RTS	PC	
5007	016420	104401	003240		5\$:	TYPE	STNTVD	;TYPE WARNING ABOUT STATUS VALID
5008	016424	012765	000077	000064	MOV	#77,PRSTCON(R5)	;DO NOT PRINT STATUS WORDS	
5009	016432	005046			CLR	-(SP)	;PUT NEW PS ON STACK	
5010	016434	012746	016442		MOV	#64\$,-(SP)	;PUT NEW PC ON STACK	
5011	016440	000002			RTI		;POP NEW PC AND PS	
5012	016442				64\$:			
5013	016442	000757			BR	7\$		
5014					.SBTTL	ERROR RETURN FROM DRIVER (CONTROLLER ERROR)		
5015					*ENTRY:	JSR	PC,CONERR WITH R5 POINTING TO ACTIVE PARAM BLK	
5016					*RETURN:	RTS	PC	
5017					*			

```

5018 ;*THIS ROUTINE IS CALLED BY THE DRIVER WHEN AN ERROR
5019 ;*OCCURS WITH THE CONTROLLER ERROR BIT SET.
5020 ;*****
5021 CONERR:
5022     MOV     RO,-(SP)      ;PUSH RO ON STACK
5023     MOV     R1,-(SP)      ;PUSH R1 ON STACK
5024     BISB    #377,DONE     ;SET DONE
5025     BIT     #SW10,2SWR    ;BELL ON ERROR?
5026     BEQ     1$           ;NO - BRANCH
5027     TYPE    $BELL        ;RING BELL
5028     BIT     #SW13,2SWR    ;INHIBIT PRINT?
5029     BNE     2$           ;YES - SKIP TO EXIT
5030     MOV     E.CONT,RO     ;GET ERROR WORD
5031     BPL     3$           ;
5032     TYPE    $DTER        ;
5033     CLR     R1           ;
5034     8$:     ASL     RO      ;SHIFT ERROR WORD LEFT TO TEST BIT 15
5035     BCS     9$           ;IF CARRY TRUE GO REPORT ERROR
5036     CMP     R1,#40       ;TEST IF ALL BITS TESTED
5037     BEQ     9$           ;GO REPORT NO ERROR ENTRY
5038     TST     (R1)+        ;BUMP R1 BY 2
5039     BR      8$          ;LOOP
5040     9$:     MOV     ETABL(R1),10$ ;GET ADDRESS OF ERROR MESSAGE
5041     TYPE    $MESSAGE     ;TYPE MESSAGE
5042     10$:    .WORD    ;ADDRESS OF MESSAGE GOES HERE
5043     JSR     PC,READRG     ;GO GET RK611 REGISTERS
5044     CLR     E.CONT        ;CLEAR CONTROLLER ERROR WORD
5045     BISB    #1,CEFLG     ;SET CONTROLLER ERROR FLAG
5046     JSR     PC,DRVERR     ;GO REPORT
5047     CLRB    CEFLG        ;CLEAR ERROR FLAG
5048     MOV     (SP)+,R1      ;POP STACK INTO R1
5049     MOV     (SP)+,RO      ;POP STACK INTO RO
5050     MOV     #SCLR,RKCS2(R2) ;CLEAR CONTROLLER
5051     MOV     #IE,RKCS1(R2) ;SET IE AGAIN
5052     JSR     PC,SWCONT     ;GO TO SWITCH CONTROL
5053     RTS     PC           ;RETURN
5054 ;*****
5055 ;SBTTL SWITCH CONTROL TEST
5056 ;*ENTRY: JSR PC,SWCONT
5057 ;*RETURN: RTS PC
5058 ;
5059 ;     MOV     $LPERR,(SP)  ;IF NO LOOP OR HALT
5060 ;     RTS     PC          ;IF LOOP ON ERROR
5061 ;
5062 ;     HALT      ;IF HALT ON ERROR
5063 ;     PRESS CONTINUE TO EXECUTE NEXT COMMAND OR LOOP ON
5064 ;     ERROR (SWR 9).
5065 ;*****
5066 SWCONT: BIT     #SW15,2SWR ;TEST HALT ON ERROR
5067     BEQ     1$           ;
5068     HALT
5069     1$:     BIT     #SW9,2SWR ;TEST LOOP ON ERROR
5070     BEQ     2$           ;
5071     BISB    #1,$ERFLG     ;SET ERFLG TO LOOP ON TEST FOREVER
5072     MOV     $LPERR,(SP)   ;FUDGE RETURN FOR LOOP
5073     CLR     -(SP)        ;;PUT NEW PS ON STACK

```

5074 016660 012746 016666  
 5075 016664 000002  
 5076 016666  
 5077 016666 000207  
 5078  
 5079  
 5080  
 5081 016670 005737 001722  
 5082 016674 001003  
 5083 016676 005737 001724  
 5084 016702 001412  
 5085 016704 012746 001722  
 5086 016710 004737 033700  
 5087 016714 012637 016722  
 5088 016720 104401  
 5089 016722 000000  
 5090 016724 104401 003411  
 5091 016730 000207  
 5092  
 5093  
 5094  
 5095  
 5096  
 5097  
 5098  
 5099  
 5100  
 5101  
 5102  
 5103 016732 005702  
 5104 016734 001403  
 5105 016736 005046  
 5106 016740 112016  
 5107 016742 000401  
 5108 016744 012046  
 5109 016746 104402  
 5110 016750 104401 003530  
 5111 016754 000002  
 5112  
 5113  
 5114  
 5115  
 5116  
 5117  
 5118  
 5119  
 5120  
 5121  
 5122  
 5123  
 5124  
 5125  
 5126  
 5127  
 5128  
 5129

```

      MOV      #64$, -(SP)      ;; PUT NEW PC ON STACK
      RTI                          ;; POP NEW PC AND PS

64$:
2$:      RTS      PC      ; RETURN
;*****
;SBTTL PRINT LOOP COUNTER IF NOT ZERO ROUTINE
;*****
PALPCT: TST      LPCNT1      ; TEST IF LOOP COUNTER ZERO
      BNE      1$      ; IF ZERO - EXIT
      TST      LPCNT2
      BEQ      3$
1$:      MOV      #LPCNT1, -(SP) ; GET ADDRESS OF LOOP COUNTERS
      JSR      PC, #5082D      ; CALL CONVERSION
      MOV      (SP)+, 2$      ; STORE RESULTS FOR PRINT
      TYPE
2$:      .WORD
      TYPE      LPLABL
3$:      RTS      PC
;*****
;SBTTL ERROR REPORT PRINT ROUTINE
;ENTRY: FPRINT (TRAP CALL): WITH R2 A SWITCH SUCH THAT
;      IF NON-ZERO THE OCTAL TYPE
;      OUT IS A BYTE AND IF ZERO IT
;      IS A WORD.
;      WITH R0 CONTAINING THE
;      ADDRESS OF THE DATA TO
;      BE TYPED.
;RETURN: RTI
;*****
FFPRINT: TST      R2      ; TEST IF BYTE OPERATION
      BEQ      1$      ; BRANCH IF NO, ELSE
      CLR      -(SP)      ; CLEAR WORD ON STACK
      MOVB      (R0)+, (SP) ; MOVE BYTE FOR PRINT ON STACK
      BR      2$
1$:      MOV      (R0)+, -(SP) ; MOVE WORD ON STACK
2$:      TYP0C
      TYPE      ,SPACE2
      RTI                          ; RETURN
;*****
;SBTTL STATUS COMPARE EXECUTION
;ENTRY: JSR      R4, ESSC
;      <STATUS WORD NUMBER>
;      <EXPECTED VALUE>
;      <MASK>
;RETURN: RTS      R4
;
;THIS ROUTINE FIRST GOES TO THE DRIVER PARAMETER BLOCK TO CHECK
;IF THE STATUS HAS ALREADY BEEN RETRIEVED AND STORED THERE. IF
;IT IS, THE STATUS IN THE PARAMETER BLOCK IS USED IN THE
;TEST. IF STATUS HAS NOT BEEN STORED, DRIVE SELECT
;SPECIAL IS ISSUED TO THE DRIVE TO GET ALL THE STATUS
;WORDS.
;
;THE COMPARE ONLY TESTS THE BITS THAT CORRESPOND TO ONES
;IN THE MASK. IF A MISCOMPARE OCCURS THE NUMBER OF THE
;STATUS WORD TESTED, THE EXPECTED VALUE, THE RECEIVED VALUE,

```

```

5130 ;*AND THE MASK ARE REPORTED.
5131 ;*****
5132 ;*****
5133 017610 012705 002142 ESSC: MOV #17610 ;SET R5 TO BLOCK 0
5134 017614 005237 001116 INC LINNUM ;INCREMENT LINE COUNT
5135 017620 105737 001670 TSTB PBSW ;TEST PARAM BLOCK SWITCH
5136 017624 001402 BEQ 1$ ;IF BLOCK 0 SELECTED BRANCH
5137 017626 012705 002230 MOV #PARM1,R5 ;ELSE SET R5 TO BLOCK 1
5138 017632 032765 001000 000014 1$: BIT #PBSVAL,P.PRST(R5) ;TEST STATUS VALID BIT
5139 017640 001005 BNE 2$ ;STATUS VALID, GO TO TEST
5140 017642 112765 000141 000001 MOVB #ROSTAT,P.CMND(R5) ;SET TO DO READ STATUS
5141 017650 004737 013570 JSR PC,DRVCL ;GO TO DRIVER CALL
5142 017654 012400 2$: MOV (R4)+,R0 ;STORE STATUS WORD NUMBER
5143 017656 012401 MOV (R4)+,R1 ;STORE EXPECTED VALUE
5144 017660 012402 MOV (R4)+,R2 ;STORE MASK
5145 017662 006300 ASL R0 ;SHIFT ST WD FOR CORRECT INDEX
5146 017664 062700 000040 ADD #P.ADD,R0 ;ADD BLOCK OFFSET FOR STATUS WORDS
5147 017670 060500 ADD R5,R0 ;COMPUTE STATUS WORD ADDRESS
5148 017672 011003 MOV (R0),R3 ;STORE IT
5149 017674 005102 COM R2 ;COMPLEMENT MASK
5150 017676 040201 BIC R2,R1 ;CLEAR UNTESTED BITS IN
5151 017700 040203 BIC R2,R3 ;EXPECTED & RECEIVED VALUES
5152 017702 020103 CMP R1,R3 ;COMPARE FOR EQUAL
5153 017704 001447 BEQ 3$ ;EQUAL - EXIT
5154 017706 032777 002000 161224 BIT #SW10,ASWR ;BELL ON ERROR?
5155 017714 001402 BEQ 4$ ;NO - BRANCH
5156 017716 104401 001164 TYPE $BELL
5157 017722 032777 020000 161210 4$: BIT #SW13,ASWR ;INHIBIT PRINT?
5158 017730 001033 BNE 5$ ;YES - BRANCH
5159 017732 162704 000006 SUB #6,R4 ;BACK UP R4 TO PARAMETERS
5160 017736 013746 001116 MOV LINNUM,-(SP) ;PUT LINE NUMBER ON STACK
5161 017742 104405 TYPDS ;TYPE IT
5162 017744 104401 014061 TYPE ,PLINE ;LABEL IT
5163 017750 104401 014520 TYPE ,SCERR ;TYPE STATUS COMPARE ERR
5164 017754 012446 MOV (R4)+,-(SP) ;GET STATUS WORD
5165 017756 104402 TYPDC ;TYPE IT
5166 017760 104401 014550 TYPE ,STWDM ;TYPE LABEL
5167 017764 012446 MOV (R4)+,-(SP) ;GET GOOD WORD
5168 017766 104402 TYPDC ;TYPE IT
5169 017770 104401 014467 TYPE ,GDDAT ;TYPE LABEL
5170 017774 011046 MOV (R0)+,-(SP) ;GET BAD WORD
5171 017776 104402 TYPDC ;TYPE IT
5172 020000 104401 014504 TYPE ,BDDAT ;TYPE LABEL
5173 020004 012446 MOV (R4)+,-(SP) ;GET MASK
5174 020006 104402 TYPDC ;TYPE IT
5175 020010 104401 014571 TYPE ,MSKLAB ;TYPE LABEL
5176 020014 004737 016670 JSR PC,PRLPCT ;GO TO SWITCH CONTROL FOR LOOP
5177 020020 004737 016622 JSR PC,SWCONT ;RETURN
5178 020024 000204 RTS R4
5179 ;*****
5180 ;SBTTL REGISTER COMPARE EXECUTION
5181 ;*ENTRY: JSR R4 ESREGC
5182 ;*
5183 ;* <REGISTER NUMBER>
5184 ;* <EXPECTED VALUE>
5185 ;* <MASK>
;*RETURN: RTS R4

```

```

5186
5187
5188
5189
5190
5191
5192
5193
5194
5195
5196
5197
5198
5199
5200 020026 005237 001116
5201 020032 012400
5202 020034 012401
5203 020036 012402
5204 020040 006300
5205 020042 063700 023554
5206 020046 011003
5207 020050 010300
5208 020052 005102
5209 020054 040201
5210 020056 040200
5211 020060 020001
5212 020062 001447
5213 020064 032777 002000 161046
5214 020072 001402
5215 020074 104401 301164
5216 020100 032777 020000 161032 3$:
5217 020106 001033
5218 020110 162704 000006
5219 020114 013746 001116
5220 020120 104405
5221 020122 104401 014061
5222 020126 104401 014601
5223 020132 012446
5224 020134 104402
5225 020136 104401 014633
5226 020142 012446
5227 020144 104402
5228 020146 104401 014467
5229 020152 010346
5230 020154 104402
5231 020156 104401 014504
5232 020162 012446
5233 020164 104402
5234 020166 104401 014571
5235 020172 004737 016670
5236 020176 004737 016622
5237 020202 000204
5238
5239
5240
5241

;*****
;THIS ROUTINE DIRECTLY ACCESSES THE DESIRED RK611 REGISTER AND COMPARES
;IT TO THE EXPECTED VALUE. ONLY THE BITS THAT CORRESPOND TO ONE
;IN THE MASK ARE COMPARED.
;
;THE REGISTER NUMBER APPEARING IN THE PARAMETER IS AN OCTAL
;VALUE. THIS VALUE IS USED AS AN INDEX TO SELECT THE DESIRED
;REGISTER. SINCE THE REGISTERS ARE ON WORD BOUNDARIES, THE VALUE
;MUST BE SHIFTED LEFT ONE BIT BEFORE IT IS USED AS THE INDEX.
;
;IF A COMPARE PRODUCES AN ERROR, THE REGISTER NUMBER, THE EXPECTED VALUE, THE
;VALUE READ, AND THE MASK ARE PRINTED ON THE CONSOLE.
;*****
E$REGC: INC LINNUM ;INCREMENT PSEUDO LINE CNT
MOV (R4)+,R0 ;GET REGISTER NUMBER
MOV (R4)+,R1 ;GET EXPECTED VALUE
MOV (R4)+,R2 ;GET MASK
ASL R0 ;MAKE REGNUM CORRECT FOR INDEX
ADD RKBAS,R0 ;COMPUTE ADDRESS
MOV (R0),R3 ;GET CONTENTS
MOV R3,R0 ;STORE AGAIN FOR REPORT IF ERROR
COM R2 ;COMPLEMENT MASK
BIC R2,R1 ;CLEAR UNTESTED BITS IN EXP. VAL.
BIC R2,R0 ;CLEAR UNTESTED BITS FROM REG
CMP R0,R1 ;COMP TWO VALUES
BEQ 1$ ;IF ZERO, NO ERROR. BRANCH
BIT #SW10,2$SWR ;BELL ON ERROR?
BEQ 3$ ;NO - BRANCH
TYPE $BELL ;RING BELL
BIT #SW13,2$SWR ;INHIBIT PRINT?
BNE 2$ ;YES - BRANCH
SUB #6,R4 ;REPOSITION R4 FOR REPORT DATA
MOV LINNUM,-(SP) ;PUT LINE NUMBER ON STACK
TYPE IT ;TYPE IT
,PLINE ;LABEL IT
TYPE RCERR ;TYPE REG COMP ERROR
MOV (R4)+,-(SP) ;GET REGISTER NUMBER
TYPE IT ;TYPE IT
,REGLAB ;LABEL IT
MOV (R4)+,-(SP) ;GET GOOD WORD
TYPE IT ;TYPE IT
,GDDAT ;LABEL IT
MOV R3,-(SP) ;GET BAD WORD
TYPE IT ;TYPE IT
,BDDAT ;LABEL IT
MOV (R4)+,-(SP) ;GET MASK
TYPE IT ;TYPE IT
,MSKLAB ;LABEL IT
PC,PRLPCT ;GO TO SWITCH CONTROL FOR LOOP
JSR PC,SWCONT
RTS R4
1$:
;*****
;SBTTL DATA COMPARE EXECUTION
;ENTRY: JSR R4,E.DATC
; * (NUMBER OF WORDS)

```

```

5242
5243
5244
5245
5246
5247
5248
5249
5250
5251
5252
5253 020204
5254 020204 010546
5255 020206 005237 001116
5256 020212 013700 001710
5257 020216 013701 001706
5258 020222 012402
5259 020224 005003
5260 020226 005005
5261 020230 022021 1S:
5262 020232 001004
5263 020234 005205 4S:
5264 020236 005302
5265 020240 001452
5266 020242 000772
5267 020244 032777 002000 160666 2S:
5268 020246 001402
5269 020248 104401 001164
5270 020250 032777 000004 160652 5S:
5271 020252 001053
5272 020254 005703
5273 020256 001005
5274 020258 013746 001116
5275 020300 104405
5276 020302 104401 014061
5277 020304 020327 000012 6S:
5278 020306 101404
5279 020308 032777 000002 160616
5280 020310 001417
5281 020312 104401 014434 3S:
5282 020314 010546
5283 020316 104402
5284 020318 104401 001171
5285 020320 024041
5286 020322 012146
5287 020324 104402
5288 020326 104401 014467
5289 020328 012046
5290 020330 104402
5291 020332 104401 014504
5292 020334 005203 13S:
5293 020336 000723
5294 020338 005703 11S:
5295 020340 001414
5296 020342 032777 020000 160540
5297 020400 001006

```

```

; *RETURN: RTS R4
; *THIS ROUTINE WILL COMPARE THE CONTENTS OF THE INPUT BUFFER
; *TO THE CONTENTS OF THE OUTPUT BUFFER. THE NUMBER OF
; *WORDS COMPARED IS SPECIFIED AS THE PARAMETER.
; *
; *IF A MISCOMPARE OCCURS, THE GOOD AND BAD WORD IS
; *REPORTED. THE NUMBER OF THE WORD THAT MISCOMPARED IS
; *ALSO PRINTED. ONLY THE FIRST 10 MISCOMPARES WILL BE
; *PRINTED UNLESS SWITCH REGISTER 1 IS SET. IN THAT
; *CASE ALL MISCOMPARES ARE PRINTED.
; *****
ESDATC:
MOV R5, -(SP) ; PUSH R5 ON STACK
INC LINNUM ; INCREMENT PSEUDO LINE COUNTER
MOV IBUFPT, R0 ; START OF INPUT BUFFER
MOV OBUFPT, R1 ; START OF OUTPUT BUFFER
MOV (R4)+, R2 ; COMPARE COUNT
CLR R3 ; ERROR COUNT
CLR R5 ; CLEAR COUNT FOR REPORT
CMP (R0)+, (R1)+ ; COMPARE DATA
BNE 2S ; ERROR
INC R5 ; COUNT FOR PRINT
DEC R2 ; COUNT FOR COMPARE LENGTH
BEQ 11S ; EXIT - OK
BR 1S ; LOOP
BIT #SW10, 2SWR ; BELL ON ERROR?
BEQ 5S ; NO - BRANCH
TYPE $BELL ; RING BELL
BIT #SW2, 2SWR ; INHIBIT PRINT?
BNE 12S ; YES - SKIP PRINT
TST R3 ; ANY ERRORS YET COUNTED?
BNE 6S ; YES - DON'T PRINT LINE NUMBER
MOV LINNUM, -(SP) ; PUT LINE NUMBER ON STACK
TYPE IT ; TYPE IT
LABEL IT ; LABEL IT
CMP R3, #12 ; TEST FOR TOO MANY ERRORS
BLOS 3S ; NO - SKIP SWR TEST
BIT #SW1, 2SWR ; TEST SWITCH 1 SET
BEQ 13S ; EXIT WITH ERROR
TYPE FDATC ; TYPE DATA COMPARE ERROR
MOV R5, -(SP) ; ERROR WORD POSITION
TYPEOC ; PRINT IT
TYPE $SCRLF ; RETURN CARRIAGE
CMP -(R0), -(R1) ; BACK UP DATA POINTERS
MOV (R1)+, -(SP) ; GOOD WORD
TYPEOC ; TYPE IT
TYPE GDDAT ; TYPE LABEL
MOV (R0)+, -(SP) ; BAD WORD
TYPEOC ; TYPE IT
TYPE BDDAT ; LABEL IT
INC R3 ; COUNT ERROR
BR 4S
TST R3 ; TEST IF ANY ERRORS OCCURED
BEQ 14S ; NO - EXIT
BIT #SW13, 2SWR ; CHECK IF INHIBIT TYPE OUT
BNF 12S ; YES, RETURN

```



5298 020402 010346  
5299 020404 104402  
5300 020406 104401 003273  
5301 020412 004737 016670  
5302 020416 004737 016622  
5303 020422  
5304 020422 012605  
5305 020424 000204  
5306  
5307  
5308  
5309  
5310  
5311  
5312  
5313  
5314  
5315  
5316  
5317  
5318 020426 005237 001116  
5319 020432 012400  
5320 020434 006300  
5321 020436 063700 023554  
5322 020442 012410  
5323 020444 000204  
5324  
5325  
5326  
5327  
5328  
5329  
5330  
5331  
5332  
5333  
5334 020446 005237 001116  
5335 020452 012737 000340 177776  
5336 020460 012437 023632  
5337 020464 112737 000177 023616  
5338  
5339 020472 010637 001704  
5340 020476 012737 020512 023564  
5341 020504 004737 023634  
5342 020510 000775  
5343 020512 013706 001704  
5344 020516 012737 015036 023564  
5345 020524 012737 000000 177776  
5346 020532 000204  
5347  
5348  
5349  
5350  
5351  
5352  
5353

```

MOV R3,-(SP) ;PUT ERROR COUNT ON STACK
TYPOC ;TYPE IT
TYPE TOTMSC ;LABEL IT
JSR PC,PRLPCT
12$: JSR PC,SWCONT ;GO TO SWITCH CONTROL
14$:
MOV (SP)+,R5 ;POP STACK INTO R5
RTS R4 ;RETURN TO OBJECT CODE
;*****
;SMTL REGISTER WRITE EXECUTION
;ENTRY: JSR R4,ESRW
; <REG NUMBER>
; <VALUE>
;RETURN: RTS R4
;
;THIS ROUTINE PLACES THE VALUE GIVEN IN THE REGISTER
;SPECIFIED. NO ATTEMPT IS MADE TO PROTECT THE USER
;AGAINST WRITING READ-ONLY BITS OR AGAINST CAUSING
;UNEXPECTED INTERRUPTS.
;*****
ESRW: INC LINNUM ;INCREMENT PSUEDO LN CNT
MOV (R4)+,RO ;GET REG NUM
ASL RO ;ALIGN RO (REGISTER NUM) FOR INDEX
ADD RKBAS,RO ;COMPUTE ADDRESS
MOV (R4)+,(RO) ;LOAD VALUE
RTS R4 ;RETURN
;*****
;SMTL STALL EXECUTION
;ENTRY: JSR R4,ESST
; <NUM OF MILLISECONDS>
;RETURN: RTS R4
;
;THIS ROUTINE WILL DELAY THE PROGRAM EXECUTION FOR
;THE TIME SPECIFIED. THE DRIVER WATCHDOG TIMER IS
;USED FOR THE DELAY TIMING.
;*****
ESST: INC LINNUM ;INCREMENT LINE COUNT
MOV #PR7,PS ;LOCK OUT ANY INTERRUPTS
MOV (R4)+,W.DRV ;LOAD MS DELAY INTO TIMER
MOVB #177,W.TIME ;FAKE TIMER TO THINK IT IS
;WATCHING A DRIVE
MOV SP,TEMP2 ;STORE STACK POINTER
MOV #2$,A.ABNL ;SET UP TIMER RETURN
1$: JSR PC,W.WTCH ;CALL TIMER
BR 1$ ;LOOP ON TIMER
2$: MOV TEMP2,SP ;RESTORE STACK POINTER
MOV #DRVERA,A.ABNL ;RESTORE ABNORMAL RETURN
MOV #PRO,PS ;ALLOW ALL INTERRUPTS
RTS R4 ;RETURN
;*****
;SMTL BUFFER INITIALIZE EXECUTION
;ENTRY: JSR R4,ESBI
; <EXECUTION LINE COUNTER CONTROL> <PATTERN>
;RETURN: RTS R4
;
;THIS ROUTINE FIRST CHECKS LINE COUNT CONTROL AND IF A ONE

```

```
5354 ;*INCREMENTS LINNUM.
5355 ;*
5356 ;*THE SELECTED PATTERN IS THEN GENERATED AND THE ENTIRE OUTPUT
5357 ;*BUFFER IS INITIALIZED.
5358 ;*****
5359 ;*****
5360 020536 005714 ESBI: TST (R4) ;TEST IF LINE COUNT TO BE INC.
5361 020540 100002 BPL 1$ ;BRANCH IF NO
5362 020542 005237 INC LINNUM ;INC LINE COUNT
5363 020546 013737 001706 001704 1$: MOV OBUFPT,TEMP2 ;COMPUTE AND STORE
5364 020554 063737 001700 001704 ADD MAXWDS,TEMP2 ;THE LAST USABLE BUFFER LOC
5365 020562 063737 001700 001704 ADD MAXWDS,TEMP2 ;DO AGAIN FOR LAST BUFF ADDRESS
5366 020570 013702 001706 MOV OBUFPT,R2 ;SET BUFFER POINTER
5367 020574 012401 MOV (R4)+,R1 ;GET PARAMETER
5368 020576 120127 000122 CMPB R1,#'A ;RANDOM PATTERN?
5369 020602 001004 BNE 2$ ;NO
5370 020604 012701 001562 MOV #PATR,R1 ;GET ADDRESS OF STORED PAT
5371 02 610 000137 021240 JMP 26$ ;GO INITIALIZE WITH PATTERN
5372 020614 120127 000130 2$: CMPB R1,#'X ;PAT X?
5373 020620 001004 BNE 3$ ;NO
5374 020622 012701 001262 MOV #PATX,R1 ;GET ADDRESS OF STORED PAT
5375 020626 000137 021240 JMP 26$ ;GO INITIALIZE WITH PATTERN
5376 020632 120127 000131 3$: CMPB R1,#'Y ;PAT Y?
5377 020636 001003 BNE 4$ ;NO
5378 020640 012701 001362 MOV #PATY,R1 ;GET ADDRESS
5379 020644 000575 BR 26$ ;GO INITIALIZE WITH PATTERN
5380 020646 120127 000132 4$: CMPB R1,#'Z ;PAT Z?
5381 02 652 001003 BNE 5$ ;NO
5382 020654 012701 001462 MOV #PATZ,R1 ;GET ADDRESS
5383 020660 000567 BR 26$ ;GO INITIALIZE WITH PATTERN
5384 020662 120127 000101 5$: CMPB R1,#'A ;TEST IF PAT A
5385 020666 001013 BNE 6$ ;NO
5386 020670 012701 177777 MOV #177777,R1 ;1ST AND LAST WORD OF PATTERN
5387 020674 005003 CLR R3 ;OTHER 30 WORDS
5388 020676 012700 000036 6$: MOV #36,R0 ;SET COUNT FOR OTHER 30 WORDS
5389 020702 010122 MOV R1,(R2)+ ;PUT 1ST WORD IN BUFFER
5390 020704 010322 7$: MOV R3,(R2)+ ;PUT IN OTHER 30 LOOP
5391 020706 005300 DEC R0
5392 020710 001375 BNE 7$
5393 020712 010122 MOV R1,(R2)+ ;PUT LAST WORD OF PAT IN BUFF
5394 020714 000561 BR 28$ ;GO SPREAD THRU BUFFER
5395 020716 120127 000102 8$: CMPB R1,#'B ;PAT B?
5396 020722 001004 BNE 9$ ;NO
5397 020724 005001 CLR R1 ;1ST AND LAST WORD OF PATTERN
5398 020726 012703 177777 MOV #177777,R3 ;OTHER 30 WORDS
5399 020732 000761 BR 6$ ;GO INITIALIZE WITH PATTERN
5400 020734 120127 000103 9$: CMPB R1,#'C ;PAT C?
5401 020740 001004 BNE 10$ ;NO
5402 020742 012701 125252 MOV #125252,R1 ;1ST AND LAST WORD OF PATTERN
5403 020746 010103 MOV R1,R3 ;OTHER 30 THE SAME
5404 020750 000752 BR 6$ ;GO INITIALIZE WITH PATTERN
5405 020752 120127 000104 10$: CMPB R1,#'D ;PAT D?
5406 020756 001004 BNE 11$ ;NO
5407 020760 012701 052525 MOV #052525,R1 ;1ST AND LAST WORD
5408 020764 010103 MOV R1,R3 ;OTHER 30 THE SAME
5409 020766 000743 BR 6$ ;GO INITIALIZE WITH PATTERN
```

5410	020770	120127	000105	11\$:	CMPB	R1, #'E	;PAT E?
5411	020774	001023			BNE	15\$	
5412	020776	012701	000001		MOV	#1, R1	;BASE WORD FOR PATTERN GEN
5413	021002	012700	000020		MOV	#20, R0	;SET A LOOP CNTR
5414	021006	010122		12\$:	MOV	R1, (R2)+	;BASE WORD INTO BUFFER
5415	021010	005300			DEC	R0	;DEC COUNTER
5416	021012	001403			BEQ	13\$	;1ST HALF GENERATED, EXIT
5417	021014	006301			ASL	R1	;TWO OPERATIONS TO
5418	021016	005201			INC	R1	;CONTINUE THE PATTERN
5419	021020	000772			BR	12\$	;LOOP
5420	021022	042701	100000	13\$:	BIC	#100000, R1	;SET BASE FOR SECOND HALF
5421	021026	012700	000020		MOV	#20, R0	;SET A COUNT
5422	021032	010122		14\$:	MOV	R1, (R2)+	;PUT WORD INTO BUFFER
5423	021034	005300			DEC	R0	;DEC COUNTER
5424	021036	001510			BEQ	28\$	;PATTERN GENERATED, GO SPREAD IT
5425	021040	006201			ASR	R1	;SHIFT FOR NEXT WORD OF PAT
5426	021042	000773			BR	14\$	;LOOP
5427	021044	120127	000106	15\$:	CMPB	R1, #'F	;PAT F?
5428	021050	001016			BNE	18\$	
5429	021052	012701	177777		MOV	#177777, R1	;BASE WORD FOR PATTERN
5430	021056	012700	000021		MOV	#21, R0	;SET COUNTER
5431	021062	010122		16\$:	MOV	R1, (R2)+	;PUT IN BUFFER
5432	021064	005300			DEC	R0	;DEC COUNT
5433	021066	001402			BEQ	17\$	;1ST HALF GENERATED, EXIT
5434	021070	006301			ASL	R1	;SHIFT FOR NEXT WD OF PAT
5435	021072	000773			BR	16\$	
5436	021074	052701	100000	17\$:	BIS	#100000, R1	;SET BASE FOR 2ND HALF
5437	021100	012700	000017		MOV	#17, R0	;SET COUNT
5438	021104	000752			BR	14\$	;GO MAKE USE OF PREVIOUS
5439							;PATTERN GENERATION OP.
5440	021106	120127	000107	18\$:	CMPB	R1, #'G	;PAT G?
5441	021112	001021			BNE	22\$	
5442	021114	012701	177777		MOV	#177777, R1	;BASE WORD FOR PATTERN G
5443	021120	006101			ROL	R1	;THIS SETS CARRY AND RESETS BIT 0
5444	021122	012700	000020		MOV	#20, R0	;SET COUNT
5445	021126	010122		19\$:	MOV	R1, (R2)+	;PUT IN BUFFER
5446	021130	005300			DEC	R0	;DEC COUNT
5447	021132	001402			BEQ	20\$	;EXIT 1ST HALF IF ZERO
5448	021134	006101			ROL	R1	;SHIFT PATTERN
5449	021136	000773			BR	19\$	;LOOP
5450	021140	012700	000020	20\$:	MOV	#20, R0	;SET 2ND HALF COUNT
5451	021144	006001		21\$:	ROR	R1	;SHIFT THE PATTERN
5452	021146	010122			MOV	R1, (R2)+	;PUT IN BUFFER
5453	021150	005300			DEC	R0	;DEC COUNT
5454	021152	001442			BEQ	28\$	
5455	021154	000773			BR	21\$	;LOOP
5456	021156	120127	000110	22\$:	CMPB	R1, #'H	;PAT H?
5457	021162	001017			BNE	24\$	
5458	021164	012701	000001		MOV	#1, R1	;BASE WORD FOR PATTERN
5459	021170	012700	000020		MOV	#20, R0	;SET COUNT
5460	021174	010122		31\$:	MOV	R1, (R2)+	;PUT IN BUFFER
5461	021176	005300			DEC	R0	;DEC COUNT
5462	021200	001402			BEQ	23\$	;EXIT 1ST HALF IF ZERO
5463	021202	006301			ASL	R1	;SHIFT PATTERN
5464	021204	000773			BR	31\$	;LOOP
5465	021206	010122		23\$:	MOV	R1, (R2)+	;PUT LAST WD GENERATED IN AGAIN

```
5466 021210 012701 040000      MOV      #40000,R1      ;SET NEW BASE FOR 2ND HALF
5467 021214 012700 000017      MOV      #17,R0       ;SET COUNT
5468 021220 000704              BR      14$          ;GO USE PREV GENERATE OPERATION
5469 021222 120127 000111      24$: CMPB     R1,#'I       ;PAT I
5470 021226 001000              BNE     25$          ;NO BUT FORCE USE OF PAT I
5471 021230 012701 155555      25$: MOV      #155555,R1      ;WORST CASE PAT
5472 021234 010103              MOV      R1,R3       ;ALL WORDS THE SAME
5473 021236 000617              BR      6$           ;GO INITIALIZE WITH PATTERN
5474 021240 012700 000040      26$: MOV      #40,R0       ;SET COUNT
5475 021244 012122              27$: MOV      (R1)+,(R2)+    ;PUT IN BUFFER
5476 021246 020237 001704      CMP      R2,TEMP2      ;CHECK BUFFER FULL
5477 021252 001417              BEQ     30$          ;YES, EXIT
5478 021254 005300              DEC      R0          ;DEC COUNT
5479 021256 001372              BNE     27$          ;NO - LOOP
5480 021260 022701 001562      28$: CMP      #PATR,R1
5481 021264 001007              BNE     29$
5482 021266 010201              MOV      R2,R1
5483 021270 005741              32$: TST      -(R1)
5484 021272 012122              MOV      (R1)+,(R2)+
5485 021274 020237 001704      CMP      R2,TEMP2
5486 021300 001373              BNE     32$
5487 021302 000403              BR      30$
5488 021304 013701 001706      29$: MOV      OBUFPT,R1      ;RESET R1 TO START OF
5489                                ;OUTPUT BUFFER. FROM THIS
5490                                ;POINT ON THE INITIALIZE
5491                                ;PATTERN IS SPREAD THROUGH THE
5492                                ;OBUF.
5493 021310 000753              BR      26$          ;LOOP
5494 021312 000204              30$: RTS      R4          ;RETURN
5495                                ;*****
5496                                ;SBTTL PARAMETER BLOCK SELECTION
5497                                ;*ENTRY: JSR      PC,PBSEL
5498                                ;*RETURN: RTS      PC
5499                                ;*
5500                                ;*THIS ROUTINE SELECTS THE PARAMETER BLOCK TO BE USED IN THE
5501                                ;*NEXT OPERATION AND SETS THE PARAMETER BLOCK SWITCH ACCORDINGLY.
5502                                ;*R5 IS LOADED WITH THE PARAM BLOCK ADDRESS.
5503                                ;*
5504                                ;*IN ADDITION, THIS ROUTINE TAKES CARE OF INSERTING THE DRIVE
5505                                ;*NUMBER AND NO CHECK INDICATOR IN THAT PARAM BLOCK.
5506                                ;*IT ALSO INCREMENTS THE LINNUM (PSUEDO LINE COUNT).
5507                                ;*****
5508 021314 005237 001116      PBSEL: INC      LINNUM      ;INCREMENT LINE NUMBER COUNT
5509 021320 012705 002142      MOV      #PARMO,R5      ;SET R5 WITH PARAMO
5510 021324 105737 001670      TSTB     PBSW        ;IS PARAMO TO BE USED NEXT
5511 021330 001002              BNE     1$           ;YES (PARAM1 WAS USED LAST)
5512 021332 012705 002230      MOV      #PARAM1,R5      ;NO - SET TO PARAM1
5513 021336 105137 001670      1$: COMB     PBSW        ;SET PBSW
5514 021342 111465 000000      MOVB     (R4),P.DRVN(R5) ;MOV IN DRIVE PARAM
5515 021346 105714              TSTB     (R4)         ;WAS IT A DRIVE NUM
5516 021350 100403              BMI     2$           ;NO? GO CHANGE IT
5517 021352 112437 001176      MOVB     (R4)+,DRIVE    ;YES? PUT IT IN PARAM STORAGE TOO.
5518 021356 000404              BR      3$
5519 021360 113765 001176 000000 2$: MOVB     DRIVE,P.DRVN(R5) ;GET STORED DRIVE NUMBER
5520 021366 105724              TSTB     (R4)+      ;BUMP R4
5521 021370 005065 000014      3$: CLR      P.PRST(R5) ;CLEAR PROG STAT WORD
```

```
5522 021374 132714 000004          BITB  #BAII,(R4)          ;TEST BAI INHIBIT SWITCH
5523 021400 001403          BEQ      4$          ;NOT SET, SKIP
5524 021402 052765 100000 000014  BIS      #DTBAII,P.PRST(R5) ;SET FOR INHIBIT INCREMENT
5525 021410 132714 000002          4$: BITB  #NOCK,(R4)          ;TEST IF NO CHECK
5526 021414 001403          BEQ      5$          ;NO - SKIP
5527 021416 052765 000400 000014  BIS      #NOCHK,P.PRST(R5) ;SET NO CHECK
5528 021424 142765 000004 000007  5$: BICB  #B.CDT,P.CSIH(R5) ;CLR OLD NFO
5529 021432 153765 001244 000007  BICB  DRVTP,P.CSIH(R5) ;SET DRIVE TYPE
5530 021440 142765 000020 000007  BICB  #B.CFMT,P.CSIH(R5) ;CLEAR 24 SECTOR MODE
5531 021446 132724 000010          BITB  #SECT20,(R4)+ ;WAS THAT RIGHT?
5532 021452 001403          BEQ      6$          ;YES - SKIP TO EXIT
5533 021454 152765 000020 000007  BISB  #B.CFMT,P.CSIH(R5) ;NO - SET THE 24 SECTOR MODE BIT
5534 021462 000207          6$: RTS      PC          ;RETURN
5535          ;*****
5536          ;SBTTL UNIBUS INITIALIZE EXECUTE
5537          ;ENTRY: JSR      R4,ESUI
5538          ;RETURN: RTS      R4
5539          ;
5540          ;THIS ROUTINE EXECUTES A UNIBUS INITIALIZE
5541          ;
5542          ;*****
5543 021464 000005          ESUI: RESET          ;UNIBUS RESET
5544 021466 005237 001116          INC      LINNUM          ;INCREMENT LINE NUMBER
5545 021472 000204          RTS      R4          ;RETURN
5546          ;*****
5547          ;SBTTL SUBSYSTEM COMMANDS EXECUTION
5548          ;ENTRY: JSR      R4,ESXX WHERE XX IS THE SUBSYSTEM CMND
5549          ;
5550          ;      <NO CHECK> <DRIVE>
5551          ;      <CYL NUM> OR <OFFSET>
5552          ;      <TRACK> <SECTOR>
5553          ;      <WORD COUNT>
5554          ;RETURN: RTS      R4
5555          ;
5556          ;THIS ROUTINE HAS MANY ENTRY POINTS, ONE FOR EACH SUBSYSTEM
5557          ;COMMAND. EACH ENTRY IS SPECIAL IN THAT THE PROPER FUNCTION
5558          ;CODE IS PUT INTO THE PARAMETER BLOCK ALONG WITH THE
5559          ;APPLICABLE PARAMETERS.
5560          ;
5561          ;THE PRINT CONTROL WORD IS PUT INTO THE LAST WORD (WORD 66)
5562          ;OF THE PARAMETER BLOCK. THIS WORD CONTROLS WHICH INFORMATION
5563          ;IN THE PARAMETER BLOCK IS PRINTED IF AN ERROR OCCURS.
5564          ;REFER TO DVERR FOR A DESCRIPTION OF THE WORD AND
5565          ;THE RESULTING DATA REPORTED.
5566          ;*****
5566 021474 112703 000113  ESRC: MOVB  #RECAL,R3          ;RECALIBRATE ENTRY
5567 021500 012700 000104          MOV      #104,R0          ;REPORT FORMAT ENTRY
5568 021504 000137 022132          JMP      ONEP
5569 021510 112703 000103  ESPA: MOVB  #PACK,R3          ;PACK ACK ENTRY
5570 021514 012700 000104          MOV      #104,R0          ;REPORT FORMAT
5571 021520 000137 022132          JMP      ONEP
5572 021524 112703 000177  ESCS: MOVB  #SUBCLR,R3          ;SUBSYSTEM CLEAR ENTRY
5573 021530 012700 000104          MOV      #104,R0          ;REPORT FORMAT
5574 021534 000137 022132          JMP      ONEP
5575 021540 112703 000176  ESCC: MOVB  #CONCLR,R3          ;CONTROLLER CLEAR ENTRY
5576 021544 012700 000104          MOV      #104,R0          ;REPORT FORMAT
5577 021550 000570          BR       ONEP
```

5578	021552	112703	000101		ESDS:	MOVB	#SELDIV,R3	;DRIVE SELECT ENTRY
5579	021556	012700	000104			MOV	#104,R0	;REPORT FORMAT
5580	021562	000563				BR	ONEP	
5581	021564	112703	000105		ESDC:	MOVB	#CLEAR,R3	;DRIVE CLEAR ENTRY
5582	021570	012700	000104			MOV	#104,R0	;REPORT FORMAT
5583	021574	000556				BR	ONEP	
5584	021576	112703	000107		ESUL:	MOVB	#UNLOAD,R3	;UNLOAD ENTRY
5585	021602	012700	000104			MOV	#104,R0	;REPORT FORMAT
5586	021606	000551				BR	ONEP	
5587	021610	112703	000111		ESSS:	MOVB	#SRTSPL,R3	;START SPINDLE ENTRY
5588	021614	012700	000104			MOV	#104,R0	;REPORT FORMAT
5589	021620	000544				BR	ONEP	
5590	021622	112703	000115		ESOF:	MOVB	#OFFSET,R3	;OFFSET ENTRY
5591	021626	012700	000165			MOV	#165,R0	;REPORT FORMAT
5592	021632	000527				BR	TWOP	
5593	021634	112703	000164		ESAH:	MOVB	#RDALHD,R3	;READ ALL HEADERS
5594	021640	012700	000165			MOV	#165,R0	;SET REPORT FORMAT
5595	021644	000501				BR	THREEP	
5596	021646	112703	000125		ESRH:	MOVB	#RDHEAD,R3	;READ HEADER ENTRY
5597	021652	012700	000165			MOV	#165,R0	;REPORT FORMAT
5598	021656	000474				BR	THREEP	
5599	021660	112703	000117		ESSK:	MOVB	#SEEK,R3	;SEEK ENTRY
5600	021664	012700	000165			MOV	#165,R0	;REPORT FORMAT
5601	021670	000467				BR	THREEP	
5602	021672	004737	021314		ESWH:	JSR	PC,PBSEL	
5603	021676	112765	000127	000001		MOVB	#WRHEAD,P.CMND(R5)	
5604	021704	004737	022302			JSR	PC,BLDHOR	
5605	021710	000437				BR	SETADD	
5606	021712	010046			ESRD:	MOV	R0,-(SP)	;STORE R0
5607	021714	010146				MOV	R1,-(SP)	;AND R1
5608	021716	013700	001710			MOV	IBUFPT,R0	;SET R0 AT START OF INPUT BUFF
5609	021722	013701	001700			MOV	MAXWDS,R1	;SET R1 WITH NUMBER OF WORDS
5610	021726	005020			IS:	(R0)+	;CLEAR BUFFER LOCATION	
5611	021730	005301				DEC	R1	;DECREMENT COUNT
5612	021732	001375				BNE	IS	;LOOP UNTIL DONE
5613	021734	012601				MOV	(SP)+,R1	;RESTORE R1
5614	021736	012600				MOV	(SP)+,R0	;AND R0
5615	021740	004737	021314			JSR	PC,PBSEL	;READ DATA, GET PB ASSIGNMENT
5616	021744	112765	000121	000001		MOVB	#RDATA,P.CMND(R5)	;LOAD FUNCTION CODE
5617	021752	013765	001710	000010		MOV	IBUFPT,P.BALO(R5)	;LOAD BUFFER ADDRESS
5618	021760	000416				BR	FOURP	
5619	021762	004737	021314		ESWD:	JSR	PC,PBSEL	;WRITE DATA, GET PB ASSIGNMENT
5620	021766	112765	000123	000001		MOVB	#WDATA,P.CMND(R5)	;LOAD FUNCTION CODE
5621	021774	000405				BR	SETADD	
5622	021776	004737	021314		ESWC:	JSR	PC,PBSEL	;WRITE CHECK, GET PB ASSIGNMENT
5623	022002	112765	000131	000001		MOVB	#WATCH,P.CMND(R5)	;LOAD FUNCTION CODE
5624	022010	013765	001706	000010		SETADD:	MOV	OBUFPT,P.BALO(R5)
5625	022016	012700	000177		FOURP:	MOV	#177,R0	;LOAD BUFFER ADDRESS FOR WRITES
5626	022022	012465	000002			MOV	(R4)+,P.CYLN(R5)	;REPORT FORMAT
5627	022026	112465	000005			MOV	(R4)+,P.CYLN(R5)	;LOAD CYL ADDRESS
5628	022032	112465	000004			MOVB	(R4)+,P.TRCK(R5)	;LOAD TRACK ADDRESS
5629	022036	012465	000012			MOVB	(R4)+,P.SECT(R5)	;LOAD SECTOR ADDRESS
5630	022042	005465	000012			MOV	(R4)+,P.WC(R5)	;LOAD WORD COUNT
5631	022046	000435				NEG	P.WC(R5)	;MAKE WORD COUNT 2'S COMP
5632	022050	004737	021314		THREEP:	JSR	PC,PBSEL	;GET PB ASSIGNMENT
5633	022054	022703	000164			CMP	#RDALHD,R3	;TEST IF READ ALL HEADERS COMMAND

```
5634 022060 001003      BNE      1$      ;NO - SKIP
5635 022062 012765 001736 000010      MOV      #HDBUFF,P.BALO(R5) ;LOAD ADDRESS OF HEADER BUFFER
5636 022070 110365 000001      1$:      MOV      R3,P.CMND(R5) ;LOAD FUNCTION CODE
5637 022074 012465 000002      MOV      (R4)+,P.CYLN(R5) ;LOAD CYL ADDRESS
5638 022100 112465 000005      MOV      (R4)+,P.TRCK(R5) ;LOAD TRACK ADD
5639 022104 112465 000004      MOV      (R4)+,P.SECT(R5) ;LOAD SECTOR ADDRESS
5640 022110 000414      BR      GODRV
5641 022112 004737 021314      TWOP:     JSR      PC,PBSEL      ;GET PB ASSIGNMENT
5642 022116 110365 000001      MOV      R3,P.CMND(R5) ;LOAD FUNCTION CODE
5643 022122 112465 000006      MOV      (R4)+,P.OFST(R5) ;LOAD OFFSET
5644 022126 105724      TST      (R4)+      ;BUMP R4 TO NXT COMD
5645 022130 000404      BR      GODRV
5646 022132 004737 021314      ONEP:     JSR      PC,PBSEL      ;GET PB ASSIGNMENT
5647 022136 110365 000001      MOV      R3,P.CMND(R5) ;LOAD FUNCTION CODE
5648 022142 010065 000064      GODRV:   MOV      R0,PRCON(R5) ;INSERT FORMAT WORD
5649 022146 004737 013570      JSR      PC,DRVCAL      ;CALL DRIVER
5650 022152 000204      RTS      R4      ;RETURN
5651 ;*****
5652 ;SBTTL COPY TAPE ROUTINE
5653 ;*THIS ROUTINE WILL PUNCH A TAPE THAT IS IDENTICAL TO THE
5654 ;*TAPE BEING READ, EITHER SOURCE OR OBJECT.
5655
5656 022154 104412      CTRTE:   SAVREG
5657 000156 042777 000100 157542      BIC      #000100,APTRSR ;RESET IE, PTR
5658 000164 042777 000100 157540      BIC      #000100,APTPSR ;RESET IE, PTP
5659 000172 00277 157530      INC      APTRSR ;SET READER ENABLE
5660 000176 032777 100200 157522      1$:      BIT      #100200,APTRSR ;CHECK PTR DONE OR ERROR
5661 000184 001774      BEQ      1$      ;NO - LOOP
5662 000196 100425      BMI      20$      ;ERROR - EXIT
5663 000210 032777 100200 157514      2$:      BIT      #100200,APTPSR ;CHECK PTP DONE OR ERROR
5664 000216 001774      BEQ      2$      ;NO - LOOP
5665 000230 100423      BMI      30$      ;ERROR - EXIT
5666 000242 00277 157500      5$:      INC      APTRSR ;START READ
5667 000256 032777 100200 157472      3$:      BIT      #100200,APTRSR ;TEST DONE OR ERROR
5668 000264 001774      BEQ      3$      ;NEITHER - LOOP
5669 000276 100411      BMI      20$      ;ERROR - EXIT
5670 000290 117777 157464 157466      MOV      APTRDB,APTPDB ;MOVE DATA TO PUNCH
5671 000304 032777 100200 157456      4$:      BIT      #100200,APTPSR ;CHECK IF DONE OR ERROR
5672 000316 001774      BEQ      4$      ;NEITHER - LOOP
5673 000330 100404      BMI      30$      ;ERROR -EXIT
5674 000344 000760      BR      5$      ;DO NEXT READ
5675 000356 104401 003223      20$:     TYPE      PRERR ;READER ERROR
5676 000368 000402      BR      40$      ;PUNCH ERROR
5677 000380 104401 003207      30$:     TYPE      PUERR
5678 000392 104413      40$:     RESREG
5679 022276 005724      TST      (R4)+      ;GOOD RETURN
5680 022300 000204      RTS      R4      ;RETURN
5681 ;*****
5682 ;SBTTL ROUTINE TO BUILD HEADERS
5683 ;*THIS ROUTINE WILL BUILD HEADERS AS THEY ARE EXPECTED TO BE FOUND
5684 ;*ON THE PACK WHEN THE SECTOR FIELD OF A WRITE HEADER COMMAND IS
5685 ;*SPECIFIED AS ZERO. IF ANY VALUE OTHER THAN 0 IS ENTERED AS THE
5686 ;*SECTOR VALUE THE ROUTINE GOES TO THE SPECIAL DATA PATTERN BUFFERS
5687 ;*AND TRANSFERS THE DATA FOUND IN PAT X, PAT Y, AND THE FIRST TWO
5688 ;*WORDS OF PAT Z INTO THE OUPUT BUFFER TO BE WRITTEN AS THE HEADERS.
5689 ;*****
```

```

5690 022302 104412          BLDHDR: SAVREG
5691 022304 012700 000026    MOV    #26,R0          ;PRESET FOR 26 SECTORS
5692 022310 005037 001702    CLR    TEMP1          ;CLEAR LOCATION TO BE USED AS FLAG
5693 022314 132765 000020 000007 BITB    #B.CFMT,P.CS1H(R5) ;TEST IF THAT IS RIGHT
5694 022322 001405          BEQ    SS              ;YES - SKIP
5695 022324 012700 000024    MOV    #24,R0          ;NO - SET FOR 24 SECTORS
5696 022330 052737 001000 001702 BIS    #BIT9,TEMP1
5697 022336 013705 001706    SS:    MOV    0BUFPT,R5      ;GET BUFFER ADDRESS
5698 022342 012401          MOV    (R4)+,R1          ;GET CYL NUMBER PARAMETER
5699 022344 112402          MOV    (R4)+,R2          ;GET TRACK PARAMETER
5700 022346 112403          MOV    (R4)+,R3          ;GET SECTOR PARAMETER
5701 022350 001025          BNE    2$              ;IF SECTOR NOT 0, GO GETHORS FROM PAT X,Y,Z BUF
5702 022352 006302          ASL    R2              ;ADJUST TRACK FOR CORRECT POSITION
5703 022354 006302          ASL    R2
5704 022356 006302          ASL    R2
5705 022360 006302          ASL    R2
5706 022362 006302          ASL    R2
5707 022364 052702 140000    BIS    #140000,R2      ;SET NO BAD SECTOR BITS
5708 022370 053702 001702    BIS    TEMP1,R2        ;INSERT FORMAT BIT
5709 022374 010103          1$:    MOV    R1,R3        ;COMPUTE VRC
5710 022376 010204          MOV    R2,R4
5711 022400 040104          BIC    #1,R4
5712 022402 040203          BIC    #3,R3
5713 022404 050403          BIS    #4,R3
5714 022406 010125          MOV    R1,(R5)+          ;INSERT WORD 1
5715 022410 010225          MOV    R2,(R5)+          ;INSERT WORD 2
5716 022412 010325          MOV    R3,(R5)+          ;INSERT WORD 3
5717 022414 005202          INC    R2              ;BUMP SECTOR COUNT
5718 022416 005300          DEC    R0              ;DECREMENT COUNTER
5719 022420 001365          BNE    1$              ;LOOP IF NOT YET ZERO
5720 022422 000410          BR     4$              ;EXIT
5721 022424 012701 001262    2$:    MOV    #PATX,R1    ;GET ADDRESS OF PAT X BUFFER
5722 022430 010003          MOV    R0,R3            ;MULTIPLY SECTOR COUNT BY 3
5723 022432 006300          ASL    R0              ;TO GET THE NUMBER OF WORDS REQUIRED
5724 022434 060300          ADD    R3,R0
5725 022436 012125          3$:    MOV    (R1)+,(R5)+  ;MOVE HEADER WORD
5726 022440 005300          DEC    R0              ;DEC COUNT
5727 022442 001375          BNE    3$              ;LOOP UNTIL DONE
5728 022444 104413          4$:    RESREG
5729 022446 000207          RTS    PC
5730                                     ;*****
5731                                     ;SBTTL DRIVE TYPE CHANGE ROUTINE
5732                                     ;ENTRY      JSR    R4,DTRTE
5733                                     ;
5734                                     ;      WITH R1 POINTING TO COMMAND FIELD FOR DT COMMAND
5735                                     ;
5736                                     ;ERROR RETURN:      MOV    (R4),R4
5737                                     ;                      RTS    R4
5738                                     ;NO ERROR RETURN:    TST    (R4)+
5739                                     ;                      RTS    R4
5740                                     ;
5741                                     ;THIS ROUTINE CHANGES THE "DRVTP" PARAMETER BY STORING
5742                                     ;IN IT THE VALUE SPECIFIED BY THE 'DT'COMMAND
5743                                     ;*****
5744
5745 022450          DTRTE:

```



```

5746 022450 010146      MOV      R1,-(SP)      ;:PUSH R1 ON STACK
5747 022452 004737 004666 JSR      PC,SBSCN      ;:BUMP R1 TO NEXT PARAM
5748 022456 105741      TSTB     (R1)      ;:TEST IF NULL
5749 022460 001453      BEQ      1$      ;:BR IF YES & EXIT
5750 022462 121127 000077 CMPB     (R1),#'?      ;:SEE IF "?"
5751 022466 001014      BNE      4$      ;:BR IF NO
5752 022470 105737 001244 TSTB     DRVTP      ;:ELSE TEST EXISTING DRIVE TYPE
5753 022474 001403      BEQ      5$      ;:BR IF RK06
5754 022476 012746 000007 MOV      #7,-(SP)      ;:ELSE SETUP FOR RK07
5755 022502 000402      BR       6$
5756 022504 012746 000006 5$:      MOV      #6,-(SP)
5757 022510 104402      6$:      TYPOC
5758 022512 104401 001171      TYPE     SCRLF
5759 022516 000434      BR       1$
5760
5761 022520 121127 000054 4$:      CMPB     (R1),#',      ;:SEE IF NULL
5762 022524 001431      BEQ      1$      ;:BR IF YES, NO CHANGE-EXIT
5763 022526 010146      MOV      R1,-(SP)      ;:ELSE GET ADDR OF ASCII CHAR
5764 022530 004737 031160 JSR      PC,OCTBIN      ;:CONVERT TO BINARY
5765 022534 022614      2$:      ERROR RETURN
5766 022536 112637 001244 MOVB     (SP)+,DRVTP      ;:PUT CHAR IN REG
5767 022542 123727 001244 000006 CMPB     DRVTP,#6      ;:SEE IF RK06
5768 022550 001411      BEQ      7$      ;:BR IF YES
5769 022552 123727 001244 000007 CMPB     DRVTP,#7      ;:SEE IF RK07
5770 022560 001410      BEQ      8$      ;:BR IF YES
5771 022562 104401 003166      TYPE     BADTYP      ;:PRINT ERROR MSG
5772 022566 011404      29$:      MOV      (R4),R4      ;:SET ERR RETURN
5773 022570      30$:
5774 022570 012601      MOV      (SP)+,R1      ;:POP STACK INTO R1
5775 022572 000204      RTS      R4
5776
5777 022574 105037 001244 7$:      CLRB     DRVTP      ;:CLEAR FOR RK06
5778 022600 000403      BR       1$      ;:EXIT
5779 022602 112737 000004 001244 8$:      MOVB     #4,DRVTP      ;:SET TO LOAD RK07 TO P.CS1H(R5)
5780 022610 005724      1$:      TST      (R4)+      ;:SET NO ERR RETURN
5781 022612 000766      BR       30$      ;:EXIT
5782
5783 022614 104401 002640 2$:      TYPE     BADOCT      ;:INVALID OCT NUMBER
5784 022620 000762      BR       29$      ;:EXIT
5785 ;:*****
5786 ;:SBTTL ERROR MESSAGES FOR CONTROLLER ERROR RETURN
5787 CERRO: .ASCIZ /CLEAR CONTROLLER DID NOT CLEAR ERROR/<15><12>
5788
5789
5790
5791
5792
5793
5794
5795
5796
5797
5798
5799
5800
5801
022622 046103 040505 020122
022630 047503 052116 047522
022636 046114 051105 042040
022644 042111 047040 052117
022652 041440 042514 051101
022660 042440 051122 051117
022666 005015 000
022671 116 020117 052101
022676 042524 052116 047511
022704 020116 051511 040440
022712 052124 047105 044524
022720 047117 051440 046525
022726 040515 054522 051040
022734 043505 051511 042524
022742 006522 000012

```

5802	022746	047125	047523	044514	CERR2: .ASCIZ /UNSOLICITED ATTENTION/<15><12>
5803	022754	044503	042524	020104	
5804	022762	052101	042524	052116	
5805	022770	047511	006516	000012	
5806	022776	047125	054105	042520	CERR3: .ASCIZ /UNEXPECTED DATA TYPE ERROR/<15><12>
5807	023004	052103	042105	042040	
5808	023012	052101	020101	054524	
5809	023020	042520	042440	051122	
5810	023026	051117	005015	000	
5811	023033	101	052124	047105	CERR4: .ASCIZ /ATTENTION DID NOT RESET WITH DRIVE CLEAR/<15><12>
5812	023040	044524	047117	042040	
5813	023046	042111	047040	052117	
5814	023054	051040	051505	052105	
5815	023062	053440	052111	020110	
5816	023070	051104	053111	020105	
5817	023076	046103	040505	006522	
5818	023104	000012			
5819	023106	052101	042524	052116	CERR5: .ASCIZ /ATTENTION DID NOT CLEAR WITH SUS-SYSTEM CLEAR/<15><12>
5820	023114	047511	020116	044504	
5821	023122	020104	047516	020124	
5822	023130	046103	040505	020122	
5823	023136	044527	044124	051440	
5824	023144	051525	051455	051531	
5825	023152	042524	020115	046103	
5826	023160	040505	006522	000012	
5827	023166	046111	042514	040507	CERR6: .ASCIZ /ILLEGAL DRIVER COMMAND/<15><12>
5828	023174	020114	051104	053111	
5829	023202	051105	041440	046517	
5830	023210	040515	042116	005015	
5831	023216	000			
5832	023217	104	052101	020101	CERR8: .ASCIZ /DATA LATE WHEN UNLOADING HEADER/<15><12>
5833	023224	040514	042524	053440	
5834	023232	042510	020116	047125	
5835	023240	047514	042101	047111	
5836	023246	020107	042510	042101	
5837	023254	051105	005015	000	
5838	023261	103	047117	051124	CERR9: .ASCIZ /CONTROLLER ERROR DURING DRIVE SERVICING/<15><12>
5839	023266	046117	042514	020122	
5840	023274	051105	047522	020122	
5841	023302	052504	044522	043516	
5842	023310	042040	044522	042526	
5843	023316	051440	051105	044526	
5844	023324	044503	043516	005015	
5845	023332	000			
5846	023333	104	044522	042526	CERR10: .ASCIZ /DRIVE PARITY WHILE GATHERING STATUS/<15><12>
5847	023340	050040	051101	052111	
5848	023346	020131	044127	046111	
5849	023354	020105	040507	044124	
5850	023362	051105	047111	020107	
5851	023370	052123	052101	051525	
5852	023376	005015	000		
5853	023401	115	046125	044524	CERR15: .ASCIZ /MULTIPLE DRIVE SELECT/<15><12>
5854	023406	046120	020105	051104	
5855	023414	053111	020105	042523	
5856	023422	042514	052103	005015	
5857	023430	000			

```

5858 023431 116 020117 051105 CERR7: .ASCIIZ /NO ERROR ENTRY/<15><12>
5859 023436 047522 020122 047105
5860 023444 051124 006531 000012
5861 .EQUIV CERR7,CERR11
5862 .EQUIV CERR7,CERR12
5863 .EQUIV CERR7,CERR13
5864 .EQUIV CERR7,CERR14
5865 .EQUIV CERR7,CERR16
5866 023452 000 CEFLG: .BYTE 0 ;CONTROLLER ERROR FLAG
5867 023454 023454
5868 023454 023401
5869 023456 023431
5870 023460 023431
5871 023462 023431
5872 023464 023431
5873 023466 023333
5874 023470 023261
5875 023472 023217
5876 023474 023431
5877 023476 023166
5878 023500 023106
5879 023502 023033
5880 023504 022776
5881 023506 022746
5882 023510 022671
5883 023512 022622
5884 023514 023431
5885
5886 .SBTTL TEMPORARY CONTROLLER REGISTER STORAGE
5887
5888 023516 000000 T.CS1: .WORD 0 ;TEMPORARY STORAGE FOR COMMAND AND STATUS
5889 ; REGISTER 1
5890 023520 000000 T.CS2: .WORD 0 ;TEMPORARY STORAGE FOR COMMAND AND STATUS
5891 ; REGISTER 2
5892 023522 000000 T.WCR: .WORD 0 ;TEMPORARY STORAGE FOR WORD COUNT REGISTER
5893 023524 000000 T.BA: .WORD 0 ;TEMPORARY STORAGE FOR BUS ADDRESS REGISTER
5894 023526 000000 T.DA: .WORD 0 ;TEMPORARY STORAGE FOR DISK TRACK AND SECTOR
5895 023530 000000 T.DC: .WORD 0 ;TEMPORARY STORAGE FOR DRIVE CYLINDER
5896 023532 000000 T.ASOF: .WORD 0 ;TEMPORARY STORAGE FOR ATTENTION SUMMARY
5897 ; AND OFFSET
5898 023534 000000 T.ER: .WORD 0 ;TEMPORARY STORAGE FOR ERROR REGISTER
5899 023536 000000 T.DS: .WORD 0 ;TEMPORARY STORAGE FOR DRIVE STATUS REGISTER
5900 023540 000000 T.MR1: .WORD 0 ;TEMPORARY STORAGE FOR MAINTENANCE REGISTER 1
5901 023542 000000 T.MR2: .WORD 0 ;TEMPORARY STORAGE FOR MAINTENANCE REGISTER 2
5902 023544 000000 T.MR3: .WORD 0 ;TEMPORARY STORAGE FOR MAINTENANCE REGISTER 3
5903 023546 000000 T.POS: .WORD 0 ;TEMPORARY STORAGE FOR ECC POSITION
5904 023550 000000 T.PAT: .WORD 0 ;TEMPORARY STORAGE FOR ECC PATTERN
5905 023552 000000 T.DB: .WORD 0 ;TEMPORARY STORAGE FOR DATA BUFFER REGISTER
5906
5907 .SBTTL DRIVER PARAMETERS
5908
5909 023554 177440 RKBAS: .WORD 177440 ;ADDRESS OF RK611 UNIBUS ADDRESS BLOCK
5910 023556 000210 RKVEC: .WORD 210 ;ADDRESS OF R611 VECTOR
5911 023560 000240 RKPRI: .WORD PR5 ;RK611 INTERRUPT PRIORITY
5912 023562 013644 A.NORM: .ERRFRE ;ADDRESS OF NORMAL RETURN FROM DRIVER
5913 023564 015036 A.ABNL: .DRVERR ;ADDRESS OF ABNORMAL RETURN FROM DRIVER

```

5914	023556	016444			A.CONT: CONERR				ADDRESS OF CONTROLLER ERROR RETURN
5915	023570	000000			E.CONT: .WORD	0			CONTROLLER ERROR STATUS
5916									THIS LOCATION IS CLEARED WHEN EVERY COMMAND
5917									IS INITIATED. IF A CONTROLLER ERROR
5918									OCCURS THE FOLLOWING BIT ASSIGNMENT IS
5919									USED:
5920									
5921		000001			E.CCLR=	BIT0			CLEAR CONTROLLER DID NOT CLEAR ERROR
5922		000002			E.NOAT=	BIT1			NO ATTENTION IN ATTENTION SUMMARY REG
5923		000004			E.UATT=	BIT2			UNSOLICITED ATTENTION (SEQUENTIAL ONLY)
5924		000010			E.UDAT=	BIT3			UNEXPECTED DATA TYPE ERROR
5925		000020			E.CLAT=	BIT4			ATTENTION DID NOT RESET WITH CLEAR
5926		000040			E.SCLR=	BIT5			SUBSYSTEM CLEAR DID NOT CLEAR DRIVE
5927									ATTENTION
5928		000100			E.ILLD=	BIT6			ILLEGAL DRIVER COMMAND
5929		000400			E.DLT=	BIT8			DATA LATE WHEN UNLOADING HEADER
5930		001000			E.CERR=	BIT9			CONTROLLER ERROR DURING DRIVER SERVICING
5931		002000			E.DPAR=	BIT10			DRIVE DETECTED PARITY ERROR
5932		040000			E.CMTO=	BIT14			CONTROLLER COMMAND TIME OUT (QUEUED ONLY)
5933		100000			E.MDS=	BIT15			MULTIPLE DRIVE SELECT
5934									
5935	023572	000000			O.WAIT: .WORD	0			PARAMETER BLOCK OF THE DRIVE
5936									WAITING FOR COMMAND COMPLETION
5937	023574	000400			W.MTIM: .WORD	400			LOOP COUNTER FOR MILLISECOND SCAN OF DRIVE
5938	023576	000400			W.MILI: .WORD	400			16 MILLISECOND TIME FOR PROGRAM
5939									
5940									
5941									
5942									
5943									
5944									
5945									
5946									
5947									
5948									
5949									
5950									
5951	023600	000300			W.SEC: .WORD	300			SECOND COUNT FOR ALL COMMANDS
5952									EXCEPT START SPINDLE
5953	023602	003000			W.8SEC: .WORD	3000			8 SECOND FOR DRIVE CYCLE DOWN
5954	023604	030000			W.MIN: .WORD	30000			MINUTE TIME FOR START SPINDLE
5955	023606	000000			HDR.AD: .WORD	0			ADDRESS USED FOR READ ALL HEADERS
5956	023610	000000			HDR.CT: .WORD	0			NUMBER OF HEADERS LEFT TO READ FOR READ
5957									ALL HEADERS
5958	023612	000			I.ISRL: .BYTE	0			INTERRUPT OR RELEASED COMMAND ISSUED
5959	023613	002	004	010	H.HEAD: .BYTE	2,4,10			HEAD DECODES
5960	023616	000			W.TIME: .BYTE	0			DRIVES BEING WATCH-DOG TIMED
5961									
5962					.SBTTL	INTERRUPT MASKS			
5963									
5964	023617	000			INTMSK: .BYTE	0			INTERRUPT MASKS FOR DRIVE IN PARAMETER BLOCK
5965									
5966						INTERRUPT MASK TABLE			
5967									
5968	023620	001			I.DRV: .BYTE	1			INTERRUPT MASK FOR DRIVE 0
5969	023621	002			.BYTE	2			INTERRUPT MASK FOR DRIVE 1

CZR6RCO RK611/06 USR DEFINED  
CZR6RC.P11 02-DEC-77 11:07

MACY11 30(1046) 02-DEC-77 11:59 PAGE 111  
INTERRUPT MASKS

SEQ 0110

5970 023622 004  
5971 023623 010  
5972 023624 020  
5973 023625 040  
5974 023626 100  
5975 023627 200

.E TE 4  
.BYTE 10  
.BYTE 20  
.BYTE 40  
.BYTE 100  
.BYTE 200

; INTERRUPT MASK FOR DRIVE 2  
; INTERRUPT MASK FOR DRIVE 3  
; INTERRUPT MASK FOR DRIVE 4  
; INTERRUPT MASK FOR DRIVE 5  
; INTERRUPT MASK FOR DRIVE 6  
; INTERRUPT MASK FOR DRIVE 7

.SBTTL PARAMETER BLOCK TABLE

5976  
5977  
5978  
5979 023630 002142  
5980  
5981  
5982  
5983  
5984  
5985 023632 000000

PBLKT: PARMO

; ADDRESS OF PARAMETER BLOCK GIVEN WITH  
; DRIVE CALL. MUST BE LOADED INTO PBLKT

.SBTTL TIME FOR WATCH-DOG TIMER

W.DRV: .WORD 0

; TIME FOR INSTRUCTION IN PARAMETER BLOCK

H09

.SBTTL RK611/RK06-RK07 UNIBUS DRIVER FOR SEQUENTIAL OPERATIONS (REV. 0.10)

;\*COPYRIGHT (C) 1975,1976,1977  
;\*DIGITAL EQUIPMENT CORP.  
;\*MAYNARD, MA. 01754  
;\*AUTHOR: ROY SPITZER

.SBTTL \*WATCH-DOG TIMER

```
*****
*
*   THE WATCH-DOG TIMER DOES A PSEUDO-TIMING OF RK06-RK07 UNIBUS
*   SUBSYSTEM COMMAND. SINCE ONE CAN NOT GUARANTEE THAT A
*   REAL-TIME CLOCK (KW11-P OR KW11-L) IS ON THE SYSTEM
*   THE RK06-RK07 DRIVER WILL USE THE LOCATION W.MTIM FOR
*   MILLI-SECOND TIMING. WHEN W.MTIM REACHES ZERO THE
*   WATCH-DOG TIMER WILL SCAN THE DRIVES IN USE AS
*   DETERMINED BY THE LOCATION W.TIME. THE TIMER COUNTS
*   (ONE FOR EACH DRIVE) ARE KEPT IN THE TABLE W.DRV.
*   IF ANY COUNT IN THE TABLE W.DRV REACHES ZERO A COMMAND
*   TIME-OUT WILL BE DESIGNATED IN THE PROGRAM DEVICE STATUS
*   REGISTER OF THAT DRIVE'S PARAMETER BLOCK.
*
*   THE DRIVER WILL USE THE LOCATION W.MIN AS THE NUMBER
*   OF MILLISECDS FOR AN UNLOAD OR START SPINDLE COMMAND.
*   THE DRIVER WILL USE THE LOCATION W.SEC AS THE TIME
*   LIMIT FOR ALL OTHER COMMANDS.
*
*   FOR QUEUED OPERATIONS THE WATCH-DOG TIMER WILL
*   WATCH UP TO 8 OPERATIONS SIMULTANEOUSLY. FOR SEQUENTIAL
*   OPERATIONS ONLY ONE OPERATION WILL BE WATCHED.
*
*CALL JSR PC,W.WTCH
*   RETURN IF NO DRIVE ORDER EXCEEDED ITS TIME LIMIT
*
*   OTHERWISE AN ABNORMAL RETURN TO THE ROUTINE ADDRESS
*   BY LOCATION A.ABNL WILL OCCUR AND THE CMDTO FLAG
*   IN THE PROGRAM DEVICE STATUS REGISTER OF THE
*   APPROPRIATE PARAMETER BLOCK WILL BE SET.
*****
```

```
W.WTCH: MOV R5,-(SP) ;SAVE R5 ON THE STACK
        MOV R4,-(SP) ;SAVE R4 ON THE STACK
        MOV R3,-(SP) ;SAVE R3 ON THE STACK
        MOV R2,-(SP) ;SAVE R2 ON STACK
        MOV PS,-(SP) ;SAVE PROGRAM STATUS WORD ON STACK
        DEC W.MTIM ;DECREMENT MILLISECOND TIMER
        BNE 20$ ;IF NOT ZERO RETURN
        MOV W.MILI,W.MTIM ;REINITIALIZE MILLISECOND TIMER
        TSTB W.TIME ;CHECK IF DRIVE IS BEING TIMED
        SEQ 20$ ;NO, RETURN
        MOV RKPRI,PS ;LOCK OUT RK06-RK07 INTERRUPTS
        MOV RKBAS,R2 ;LOAD BASE OF RK06-RK07 REGISTERS
        DEC W.DRV ;DECREMENT COMMAND TIMER
        BNE 20$ ;RETURN IF NO TIME OUT
```

```
5986
5987
5988
5989
5990
5991
5992
5993
5994
5995
5996
5997
5998
5999
6000
6001
6002
6003
6004
6005
6006
6007
6008
6009
6010
6011
6012
6013
6014
6015
6016
6017
6018
6019
6020
6021
6022
6023
6024
6025
6026
6027
6028 023634 010546
6029 023636 010446
6030 023640 010346
6031 023642 010246
6032 023644 013746 177776
6033 023650 005337 023574
6034 023654 001034
6035 023656 013737 023576 023574
6036 023664 105737 023616
6037 023670 001426
6038 023672 013737 023560 177776
6039 023700 013702 023554
6040 023704 005337 023632
6041 023710 001016
```

CZR6RCD RK611/06 USR DEFINED  
CZR6RC.P11 02-DEC-77 11:07

MACY11 30(1046) 02-DEC-77 11:59 PAGE 113  
\*WATCH-DOG TIMER

SEQ 0112

6042	023712	105037	023616		CLRB	W.TIME	:RESET TIMING INDICATOR
6043	023716	013705	023630		MOV	PBLKT,R5	:LOAD ADDRESS OF PARAMETER BLOCK
6044							:TABLE FOR INDEXING
6045	023722	052765	000100	000014	BIS	#CMDTO,P.PRST(R5)	:SET COMMAND TIME OUT
6046	023730	020537	023572		CMP	R5,O.WAIT	:CHECK IF DRIVER IS WAITING FOR
6047							:COMMAND COMPLETION
6048	023734	001002			BNE	SS	:NO, DO NOT ALTER WAITING FOR
6049							:COMMAND COMPLETION
6050	023736	005037	023572		CLR	O.WAIT	:CLEAR WAIT FOR COMMAND COMPLETION
6051	023742	004737	027216	SS:	JSR	PC,R.ABNL	:BRANCH TO ERROR ROUTINE
6052	023746	012637	177776	20\$:	MOV	(SP)+,PS	:RESTORE PSW
6053	023752	012602			MOV	(SP)+,R2	:RESTORE R2
6054	023754	012603			MOV	(SP)+,R3	:RESTORE R3
6055	023756	012604			MOV	(SP)+,R4	:RESTORE R4
6056	023760	012605			MOV	(SP)+,R5	:RESTORE R5
6057	023762	000207			RTS	PC	:RETURN

J09

.SBTTL \*RK06 INTERRUPT SERVICE ROUTINE

\*\*\*\*\*

THIS ROUTINE WILL SERVICE ALL RK06 INTERRUPTS.

UPON RECEIVING AN INTERRUPT, THIS ROUTINE WILL  
PERFORM ONE OF THE FOLLOWING SERVICES:

- 1.) SERVICE PORT WAS SEIZED BY OTHER PORT
- 2.) SERVICE DRIVER IS WAIT FOR COMMAND COMPLETION
- 3.) SERVICE POSITIONING COMPLETION
- 4.) REQUEUE COMMAND IF DRIVE WAS RELEASED  
FOR THE QUEUED RK06 DRIVER.
- 5.) IF NO SERVICE IS REQUIRED, THE COMMAND WILL BE ISSUED  
FOR THE QUEUED RK06 DRIVER.

THREE LINKS ARE PROVIDED TO THE DRIVING PROGRAM.  
THEY ARE:

- 1.) A.NORM ADDRESS OF NORMAL RETURN (SUCESSFUL COMPLETION OF COMMAND)
- 2.) A.ABNL ADDRESS OF ABNORMAL RETURN (UNSUCESSFUL COMPLETION OF COMMAND)
- 3.) A.CONT ADDRESS OF CONTROL ERROR RETURN

FOR NORMAL AND ABNORMAL RETURNS, THE ADDRESS OF THE APPROPRIATE  
PARAMETER BLOCK WILL BE IN R5.

FOR THE CONTROLLER ERROR RETURN, THE LOCATION E.CONT CONTAINS  
THE REASON FOR THE CONTROLLER ERROR.

ROUTINES USED:

C.OPT (QUEUED ONLY)  
Q.PUSH (QUEUED ONLY)  
Q.RMOV (QUEUED ONLY)  
R.CONT (SEQUENTIAL ONLY)  
R.NORM (SEQUENTIAL ONLY)  
R.ABNL (SEQUENTIAL ONLY)  
I.CSTS  
I.STAT  
I.ISSU  
I.CCLR

\*\*\*\*\*

6058				
6059				
6060				
6061				
6062				
6063				
6064				
6065				
6066				
6067				
6068				
6069				
6070				
6071				
6072				
6073				
6074				
6075				
6076				
6077				
6078				
6079				
6080				
6081				
6082				
6083				
6084				
6085				
6086				
6087				
6088				
6089				
6090				
6091				
6092				
6093				
6094				
6095				
6096				
6097				
6098				
6099				
6100				
6101				
6102	023764	010546		
6103	023766	010446		
6104	023770	010346		
6105	023772	010246		
6106	023774	010146		
6107	023776	010046		
6108	024000	013702	023554	
6109	024004	016237	000010	023520
6110	024012	032737	001000	023520
6111	024020	001407		
6112	024022	052737	100000	023570
6113	024030	004737	027242	

I.INTR:	MOV	R5, -(SP)	:STORE R5 ON THE STACK
	MOV	R4, -(SP)	:STORE R4 ON THE STACK
	MOV	R3, -(SP)	:STORE R3 ON THE STACK
	MOV	R2, -(SP)	:STORE R2 ON THE STACK
	MOV	R1, -(SP)	:STORE R1 ON THE STACK
	MOV	R0, -(SP)	:STORE R0 ON THE STACK
	MOV	RKBAS, R2	:LOAD R2 TO ADDRESS RK06 REGISTER
	MOV	RKCS2(R2), T.CS2	:STORE CS2
	BIT	#MDS, T.CS2	:CHECK IF MULTIPLE DRIVE SELECT
	BEQ	1\$	:NO, CONTINUE PROCESSING
	BIS	#E.MDS, E.CONT	:SET MULTIPLE DRIVE SELECT
	JSR	PC, R.CONT	:REPORT ERROR



```
6114 024034 000137 026214      JMP      I.ATRN      ;RETURN
6115
6116 024040 105737 023612      1$:  TSTB     I.ISRL      ;CHECK IF INTERRUPT OR RELEASE
6117 024044 001410              BEQ      6$          ;NO, CHECK IF DRIVE AVAILABLE
6118 024046 100403              BMI      5$          ;CHECK IF RELEASE COMMAND
6119 024050 105037 023612      CLRB     I.ISRL      ;YES, CLEAR FLAG
6120 024054 000473              BR       I.ICO      ;CONTINUE PROCESSING INTERRUPT
6121
6122 024056 105037 023612      5$:  CLRB     I.ISRL      ;CLEAR FLAG
6123 024062 000137 025176      JMP      I.ATTN      ;GO PROCESS DRIVE ATTENTIONS
6124
6125 024066 032737 010400 023520 6$:  BIT      #NED!UFE,T.CS2 ;CHECK FOR NON-EXISTENT DRIVE OR
6126                                ;UNIT FIELD ERROR
6127 024074 001413              BEQ      7$          ;NO, WAIT FOR DUAL ACCESS INTERRUPT
6128 024076 013704 023520      MOV      T.CS2,R4      ;LOAD R4 FOR DRIVE NUMBER
6129 024102 042704 177770      BIC      #C<DRVMSK>,R4 ;KEEP DRIVE BITS
6130 024106 013705 023630      MOV      PBLKT,R5      ;STORE PARAMETER BLOCK ADDRESS
6131 024112 016237 000000 023516  MOV      RKCS1(R2),T.CS1 ;LOAD TEMPORARY CS1 FOR STATUS REPORT
6132 024120 000137 024406      JMP      I.ERRC      ;REPORT ERROR
6133
6134 024124 016237 000012 023536 7$:  MOV      RKDS(R2),T.DS ;STORE STATUS REGISTER FOR COMPARISON
6135 024132 032737 000001 023536  BIT      #DRA,T.DS ;CHECK IF DRIVE SEIZED BY OTHER
6136                                ;PORT
6137 024140 001041              BNE      I.I00      ;NO, CONTINUE PROCESSING INTERRUPT
6138
6139                                ;CHECK IF ANY DATA TRANSFER ERROR EXISTS
6140 024142 032737 164000 023520  BIT      #DLT!WCE!UPE!NEM,T.CS2
6141
6142 024150 001007              BNE      10$         ;INDICATE ERROR
6143 024152 016237 000014 023534  MOV      RKER(R2),T.ER ;STORE ERROR REGISTER
6144
6145                                ;CHECK FOR DATA TRANSFER ERROR TYPE ERROR
6146 024160 032737 125700 023534  BIT      #DCK!OPI!WLE!COE!HVRC!BSE!ECH,T.ER
6147
6148 024166 001407              BEQ      11$         ;NO, WAIT FOR RELEASE OF RK06 DRIVE
6149
6150 024170 052737 000010 023570 10$:  BIS      #E.UDAT,E.CONT ;SET UNEXPECTED DATA TYPE ERROR
6151 024176 004737 027242      JSR      PC,R.CONT ;REPORT ERROR
6152 024202 000137 026214      JMP      I.ATRN      ;RESTORE REGISTERS
6153
6154 024206 105037 023616      11$:  CLRB     W.TIME      ;RESET TIMING ON THIS DRIVE
6155 024212 005037 023632      CLR      W.DRV      ;CLEAR TIMING COUNT FOR THIS DRIVE
6156 024216 013705 023630      MOV      PBLKT,R5      ;LOAD R5 WITH PARAMETER BLOCK
6157                                ;ADDRESS
6158 024222 052765 010000 000014  BIS      #DRVSZD,P.PRST(R5) ;SET DRIVE SEIZED IN THE
6159                                ;PROGRAM DRIVE STATUS REGISTER
6160 024230 005037 023572      CLR      O.WAIT      ;CLEAR WAIT FOR COMMAND COMPLETION
6161 024234 004737 027216      JSR      PC,R.ABNL ;INDICATE ABNORMAL TERMINATION
6162 024240 000137 026214      JMP      I.ATRN      ;GO RESTORE REGISTERS
6163
6164 024244 013705 023572      I.I00: MOV      O.WAIT,R5 ;LOAD PARAMETER BLOCK ADDRESS INTO R5
6165 024250 001002              BNE      2$          ;IS COMMAND WAITING PROCESSING
6166                                ;YES, DO PROCESSING
6167 024252 000137 025176      JMP      I.ATTN      ;NO, PROCESS ATTENTION
6168
6169 024256 013704 023520      2$:  MOV      T.CS2,R4      ;STORE RKCS2 FOR DRIVE NUMBER
```

6170	024262	042704	177770			BIC	#C<DRVMSK>,R4	;MASK OUT UNNECESSARY BITS
6171								
6172								
6173	024266	126504	000000			CMPB	P.DRVN(R5),R4	;CHECK IF DRIVE NUMBER IS EXPECTED
6174	024272	001401				BEQ	3\$	;YES, CONTINUE
6175	024274	000000				HALT		;NO, DRIVER ERROR
6176	024276	122765	000164	000001	3\$:	CMPB	#RDALHD,P.CMND(R5)	;CHECK IF READ ALL HEADERS
6177	024304	001002				BNE	10\$	;NO, EXECUTE NORMAL DATA TRANSFER
6178	024306	000137	024644			JMP	I.HDAL	;GO EXECUTE SPECIAL HEADER SEQUENCE
6179								
6180	024312	005037	023572		10\$:	CLR	O.WAIT	;CLEAR WAIT FOR COMMAND COMPLETION
6181	024316	005037	023632			CLR	W.DRV	;CLEAR WATCH-DOG TIME
6182	024322	105037	023616			CLRB	W.TIME	;RESET TIMING ON THIS DRIVE
6183	024326	016237	000000	023516		MOV	RKCS1(R2),T.CS1	;STORE COMMAND AND STATUS REGISTER 1
6184	024334	032737	100000	023516		BIT	#CERR,T.CS1	;CHECK IF CONTROLLER ERROR
6185	024342	001021				BNE	I.ERRC	;YES, PROCESS ERROR
6186	024344	016237	000016	023532		MOV	RKASOF(R2),T.ASOF	;STORE ATTENTION SUMMARY
6187	024352	133737	023617	023533		BITB	INTMSK,T.ASOF+1	;CHECK IF DRIVE ATTENTION SET
6188	024360	001004				BNE	15\$	;YES, REPORT ERROR
6189	024362	004737	027230			JSR	PC,R.NORM	;INDICATE NORMAL RETURN
6190	024366	000137	026214			JMP	I.ATRN	;RESTORE REGISTERS
6191								
6192	024372	052765	000010	000014	15\$:	BIS	#UEXATT,P.PRST(R5)	;SET UNEXPECTED ATTENTION
6193								
6194	024400	004737	026664		I.ERRA:	JSR	PC,I.CSTS	;STORE CONTROLLER STATUS
6195	024404	000405				BR	I.ERR	;STORE PATTERN AND POSITION INFORMATION
6196								
6197	024406	013765	023516	000016	I.ERRC:	MOV	T.CS1,P.CS1(R5)	;GET ERROR RKCS1
6198	024414	004737	026706			JSR	PC,I.CST1	;GET REST OF CONTROLLER STATUS
6199	024420	016265	000032	000062	I.ERR:	MOV	RKECPT(R2),P.EPAT(R5)	;STORE ECC PATTERN
6200	024426	016265	000030	000060		MOV	RKECPS(R2),P.EPOS(R5)	;STORE ECC POSITION
6201	024434	004037	026232			JSR	RD,I.CCLR	;CLEAR CONTROLLER
6202	024440	026214				I.ATRN		;ERROR RETURN
6203	024442	032765	010400	000020		BIT	#NED!UFE,P.CS2(R5)	;CHECK IF IT WAS NON-EXISTENT DRIVE OR
6204								UNIT FIELD ERROR
6205	024450	001046				BNE	5\$	;YES, REPORT ERROR
6206	024452	004037	026770			JSR	RD,I.STAT	;GATHER DRIVE STATUS
6207	024456	026214				I.ATRN		;ERROR RETURN
6208	024460	112737	000005	023516		MOV	#DR,CLR,T.CS1	;LOAD COMMAND
6209	024466	004037	026314			JSR	RD,I.ISSU	;ISSUE DRIVE CLEAR
6210	024472	026214				I.ATRN		;ERROR RETURN
6211	024474	133737	023617	023533		BITB	INTMSK,T.ASOF+1	;CHECK IF ATTENTION RESET
6212	024502	001407				BEQ	2\$	;NO, INDICATE DRIVE ERROR
6213	024504	052737	000020	023570		BIS	#E.CLAT,E.CONT	;SET ATTENTION DID NOT RESET
6214								WITH CLEAR
6215	024512	004737	027242			JSR	PC,R.CONT	;REPORT CONTROLLER ERROR
6216	024516	000137	026214			JMP	I.ATRN	;GO RESTORE REGISTERS
6217								
6218	024522	032737	040000	023542	2\$:	BIT	#S.DSC,T.MR2	;CHECK IF DRIVE STATUS CHANGE CLEARED
6219	024530	001403				BEQ	3\$	;YES, CHECK FAULT
6220	024532	052765	000040	000014		BIS	#DRVDSC,P.PRST(R5)	;SET DSC DID NOT CLEAR
6221	024540	032737	001000	023544	3\$:	BIT	#S.PAR,T.MR3	;CHECK IF DRIVE PARITY ERROR
6222	024546	001407				BEQ	5\$	;NO, INDICATE ABNORMAL TERMINATION
6223	024550	052737	002000	023570		BIS	#E.DPAR,E.CONT	;SET DRIVE PARITY ERROR
6224	024556	004737	027242			JSR	PC,R.CONT	;INDICATE CONTROLLER ERROR
6225	024562	000137	026214			JMP	I.ATRN	;RETURN

```
6226
6227 024566 032765 000020 000014 5$: BIT #DRVHRD,P.PRST(R5) ;CHECK IF HARD DRIVE ERROR
6228 024574 001017 BNE 10$ ;YES, GO REPORT ERROR
6229 024576 032737 020000 023542 BIT #S.PIP,T.MR2 ;CHECK IF DRIVE IS CYCLING DOWN
6230 024604 001413 BEQ 10$ ;NO, REPORT ERROR
6231 024606 052765 020000 000014 BIS #E.UNLD,P.PRST(R5) ;SET DRIVE UNLOADING
6232 024614 113737 023617 023616 MOVB INTMSK,W.TIME ;SET UP 8 SECONDS FOR DRIVE TO CYCLE UP
6233 024622 013737 023602 023632 MOV W.8SEC,W.DRV
6234 024630 000137 026214 JMP I.ATRN ;GO RESTORE REGISTERS
6235
6236 024634 004737 027216 10$: JSR PC,R.ABNL ;GO REPORT ERROR
6237 024640 000137 026214 JMP I.ATRN ;GO RESTORE REGISTERS
6238
6239 .SBTTL *READ ALL HEADERS INTERRUPT SEQUENCE
6240
6241 024644 016237 000000 023516 I.HDAL: MOV RKCS1(R2),T.CS1 ;STORE CS1 TO CHECK CONTROLLER
6242 ERROR
6243 024652 032737 100000 023516 BIT #CERR,T.CS1 ;CHECK IF CONTROLLER ERROR
6244 024660 001422 BEQ 5$ ;NO, CHECK FOR ATTENTION
6245
6246 024662 005037 023572 CLR O.WAIT ;CLEAR WAITING FOR COMMAND COMPLETE
6247 024666 105037 023616 CLRB W.TIME ;RESET TIMING ON DRIVE
6248 024672 005037 023632 CLR W.DRV ;CLEAR TIME OUT COUNT
6249 024676 013765 023516 000016 MOV T.CS1,P.CS1(R5) ;STORE ERROR RKCS1
6250 024704 004737 026706 JSR PC,I.CST1 ;STORE CONTROLLER REGISTERS
6251 024710 004037 026232 JSR RO,I.CCLR ;CLEAR CONTROLLER
6252 024714 026214 I.ATRN ;ERROR RETURN
6253 024716 004737 027216 JSR PC,R.ABNL ;INDICATE ERROR RETURN
6254 024722 000137 026214 JMP I.ATRN ;RESTORE REGISTERS
6255
6256 024726 016237 000016 023532 5$: MOV RKASOF(R2),T.ASOF ;STORE ATTENTION SUMMARY
6257 024734 133737 023617 023533 BITB INTMSK,T.ASOF+1 ;CHECK IF DRIVE ATTENTION 1, SET
6258 024742 001410 BEQ 7$ ;NO, CHECK IF READ ALL HEADERS
6259 024744 005037 023572 CLR O.WAIT ;CLEAR WAITING FOR COMMAND COMPLETION
6260 024750 105037 023616 CLRB W.TIME ;RESET TIMING ON DRIVE
6261 024754 005037 023632 CLR W.DRV ;CLEAR TIME OUT COUNT
6262 024760 000137 024400 JMP I.ERRA ;GO REPORT ERROR
6263
6264 024764 013701 023606 7$: MOV HDR.AD,R1 ;GET MAIN MEMORY ADDRESS
6265 024770 016221 000024 MOV FKDB(R2),(R1)+ ;GET FIRST WORD OF HEADER
6266 024774 016221 000024 MOV RKDB(R2),(R1)+ ;GET SECOND WORD OF HEADER
6267 025000 016221 000024 MOV RKDB(R2),(R1)+ ;GET THIRD WORD OF HEADER
6268 025004 010137 023606 MOV R1,HDR.AD ;STORE ADDRESS FOR NEXT HEADER
6269 025010 016237 000010 023520 MOV RKCS2(R2),T.CS2 ;STORE CS2 TO CHECK FOR DATA LATE
6270 025016 032737 100000 023520 BIT #DLT,T.CS2 ;CHECK FOR DATA LATE
6271 025024 001055 BNE 35$ ;YES, REPORT ERROR
6272 025026 005337 023610 DEC HDR.CT ;DECREMENT NUMBER OF HEADER YET TO READ
6273 025032 001026 BNE 25$ ;IF NON-ZERO, GO ISSUE NEXT READ HEADER
6274 025034 005037 023572 CLR O.WAIT ;CLEAR DRIVER WAITING FOR COMMAND COMPLETION
6275 025040 005037 023632 CLR W.DRV ;CLEAR TIME OUT COUNT FOR THIS DRIVE
6276 025044 105037 023616 CLRB W.TIME ;CLEAR WATCH DOG TIME ON THIS DRIVE
6277 025050 012762 000003 000026 MOV #3,RKMR1(R2) ;LOAD MAINTENANCE REGISTER FOR SECTOR COUNT
6278 025056 112737 000001 023516 MOVB #DR.SEL,T.CS1 ;LOAD SELECT COMMAND
6279 025064 004037 026314 JSR RO,I.ISSU ;GET SECTOR COUNT
6280 025070 026214 I.ATRN ;ERROR RETURN
6281 025072 013765 023544 000056 MOV T.MR3,P.B11(R5) ;LOAD SECTOR COUNT
```

```

6282 025100 004737 027230      JSR    PC.R.NORM      ;INDICATE NORMAL TERMINATION
6283 025104 000137 026214      JMP    I.RTRN        ;RESTORE REGISTERS
6284
6295 025110 016562 000002 000020 25$: MOV    P.CYLN(R5),RKDCYL(R2) ;LOAD CYLINDER ADDRESS REGISTER
6286 025116 016562 000004 000006      MOV    P.SECT(R5),RKDA(R2) ;LOAD SECTOR AND TRACK
6287 025124 116565 000007 000017      MOV    P.CS1H(R5),P.CS1+1(R5) ;STORE BITS 8-15 OF CS1
6288 025132 042765 165777 000016      BIC    #1C<CDT!CFMT>,P.CS1(R5) ;CLEAR ALL BITS EXCEPT FORMAT AND
6289                                     ;DRIVE TYPE
6290 025140 112765 000125 000016      MOV    #RDHEAD,P.CS1(R5) ;STORE COMMAND ISSUED
6291 025146 016562 000016 000000      MOV    P.CS1(R5),RKCS1(R2) ;ISSUE READ HEADER
6292 025154 000137 026214      JMP    I.RTRN        ;RESTORE REGISTERS
6293
6294 025160 052737 000400 023570 35$: BIS    #E.DLT,E.CONT    ;SET DATA LATE WHILE UNLOADING HEADER
6295 025166 004737 027242      JSR    PC.R.CONT    ;REPORT ERROR
6296 025172 000137 026214      JMP    I.RTRN        ;RESTORE REGISTERS
6297
6298 .SBTTL  *DRIVE ATTENTION SCANNER
6299
6300 025176 016237 000000 023516 1.ATTN: MOV    RKCS1(R2),T.CS1 ;STORE COMMAND AND STATUS
6301                                     ;REGISTER 1 FOR COMPARISON
6302 025204 032737 100000 023516      BIT     #CERR,T.CS1    ;CHECK IF CONTROLLER ERROR OCCURRED
6303 025212 001441                                     BEQ     5$          ;NO, CHECK IF ATTENTION
6304                                     ;CHECK IF ANY DATA TRANSFER TYPE ERROR EXISTS
6305                                     BIT     #DLT!WCE!UPE!NEM,T.CS2
6306 025214 032737 164000 023520      BNE     1$          ;INDICATE ERROR
6307 025222 001007                                     MOV    RKER(R2),T.ER    ;STORE ERROR REGISTER
6308 025224 016237 000014 023534      ;
6309                                     ;CHECK FOR DATA TRANSFER ERROR TYPE
6310                                     BIT     #DCK!OPI!WLE!COE!HVRC!BSE!ECH,T.ER
6311 025232 032737 125700 023534      BEQ     2$          ;NO DATA TRANSFER ERROR
6312
6313 025240 001407                                     ;
6314 025242 052737 000010 023570 1$: BIS    #E.UDAT,E.CONT    ;SET UNEXPECTED DATA TYPE ERROR
6315 025250 004737 027242      JSR    PC.R.CONT    ;REPORT ERROR
6316 025254 000137 026214      JMP    I.RTRN        ;RESTORE REGISTERS
6317
6318 025260 013704 023520 2$: MOV    T.CS2,R4    ;SAVE CS2 FOR REGISTER NUMBER
6319 025264 042704 177770      BIC    #1C<DRVMSK>,R4 ;STRIP OFF JUNK
6320 025270 105037 023616      CLR    W.TIME    ;CLEAR WATCH DOG TIMER
6321 025274 005037 023632      CLR    W.DRV    ;RESET TIMER VALUE
6322 025300 013705 023630      MOV    PBLKT,R5    ;STORE PARAMETER BLOCK ADDRESS IN R5
6323
6324                                     ;
6325                                     ;CLEAR DRIVE POSITIONING AND DRIVE POSITIONED FOR DATA TRANSFER
6326                                     ;IN PROGRAM DEVICE STATUS REGISTER
6327                                     BIC    #DRVPOS!DRVPDT,P.PRST(R5)
6328 025304 042765 000006 000014      JMP    I.ERRC        ;GO REPORT ERROR
6329
6330 025312 000137 024406      ;
6331 025316 032737 040000 023516 5$: BIT     #DI,T.CS1    ;CHECK IF ANY DRIVE ATTENTION
6332 025324 001002                                     BNE     6$          ;YES, PROCESS INTERRUPT
6333 025326 000137 026214      JMP    I.RTRN        ;RESTORE REGISTERS
6334
6335 025332 016237 000016 023532 6$: MOV    RKASOF(R2),T.ASOF ;STORE ATTENTION SUMMARY
6336 025340 105737 023533      TSTB   T.ASOF+1    ;CHECK IF ANY ATTENTIONS SET
6337

```

6338	025344	001007			BNE	7\$		; YES GO PROCESS INTERRUPT
6339	025346	052737	000002	023570	BIS	#E.NOAT,E.CONT		; SET NO ATTENTION IN ATTENTION SUMMARY
6340	025354	004737	027242		JSR	PC,R.CONT		; GO REPORT ERROR
6341	025360	000137	026214		JMP	I.ATRN		; GO RESTORE REGISTERS
6342								
6343	025364	133737	023617	023533	7\$: BITB	INTMSK,T.ASOF+1		; CHECK IF DESIRED INTERRUPT
6344	025372	001007			BNE	8\$		; YES, GO PROCESS IT
6345	025374	052737	000004	023570	BIS	#E.UATT,E.CONT		; SET UNSOLICATED ATTENTION
6346	025402	004737	027242		JSR	PC,R.CONT		; GO REPORT ERROR
6347	025406	000137	026214		JMP	I.ATRN		; GO RESTORE REGISTERS
6348								
6349	025412	013705	023630		8\$: MOV	PBLKT,R5		; STORE PARAMETER BLOCK TABLE
6350	025416	116504	000000		MOVB	P.DRVN(R5),R4		; STORE DRIVE NUMBER
6351	025422	032765	020000	000014	BIT	#E.UNLD,P.PRST(R5)		; CHECK IF DRIVE UNLOADING
6352	025430	001402			BEQ	11\$		; NO, CONTINUE
6353	025432	000137	026114		JMP	I.UNLD		; SERVICE DRIVE IN POSITION AFTER ERROR
6354								
6355	025436	042765	000002	000014	11\$: BIC	#DRVPOS,P.PRST(R5)		; RESET DRIVE POSITIONING
6356	025444	005062	000026		CLR	RKMR1(R2)		; CLEAR MAINTENANCE REGISTER 1
6357	025450	112737	000001	023516	MOVB	#DR.SEL,T.CS1		; LOAD COMMAND
6358	025456	004037	026314		JSR	RO,I.ISSU		; SELECT DRIVE WITH ATTENTION HIGH
6359	025462	026214			I.ATRN			; ERROR RETURN
6360	025464	013765	023544	000042	MOV	T.MR3,P.800(R5)		; STORE STATUS BYTE 00 MESS B
6361	025472	032765	000200	000042	BIT	#S.FLT,P.800(R5)		; CHECK IF DRIVE FAULT
6362	025500	001401			BEQ	12\$		; NO, CHECK FOR DRIVE STATUS CHANGE
6363	025502	000461			BR	I.AERR		; PROCESS ERROR
6364								
6365	025504	013765	023542	000040	12\$: MOV	T.MR2,P.A00(R5)		; STORE MAINTENANCE REGISTER 2
6366	025512	032765	040000	000040	BIT	#S.DSC,P.A00(R5)		; CHECK FOR DRIVE STATUS CHANGE
6367	025520	001004			BNE	13\$		; YES, PROCESS DRIVE STATUS CHANGE
6368	025522	052765	004000	000014	BIS	#NODSC,P.PRST(R5)		; SET NO DRIVE STATUS CHANGE
6369	025530	000446			BR	I.AERR		; PROCESS ERROR
6370								
6371	025532	112737	000005	023516	13\$: MOVB	#DR.CLR,T.CS1		; LOAD COMMAND
6372	025540	004037	026314		JSR	RO,I.ISSU		; CLEAR DRIVE STATUS CHANGE
6373	025544	026214			I.ATRN			; ERROR RETURN
6374	025546	013765	023532	000032	MOV	T.ASOF,P.ASOF(R5)		; STORE ATTENTION SUMMARY
6375	025554	133765	023617	000033	BITB	INTMSK,P.ASOF+1(R5)		; CHECK IF ATTENTION RESET
6376	025562	001407			BEQ	15\$		; YES, CONTINUE INTERRUPT PROCESSING
6377	025564	052737	000020	023570	BIS	#E.CLAT,E.CONT		; SET ATTENTION DID NOT RESET
6378								; WITH DRIVE CLEAR
6379	025572	004737	027242		JSR	PC,R.CONT		; FLAG ERROR
6380	025576	000137	026214		JMP	I.ATRN		; RESTORE REGISTERS
6381								
6382	025602	013765	023542	000040	15\$: MOV	T.MR2,P.A00(R5)		; STORE MAINTENANCE REGISTER 2
6383	025610	032765	040000	000040	BIT	#S.DSC,P.A00(R5)		; CHECK IF DRIVE STATUS CHANGE
6384								; RESET
6385	025616	001404			BEQ	16\$		; YES, CONTINUE INTERRUPT PROCESSING
6386	025620	052765	000040	000014	BIS	#DRVDSC,P.PRST(R5)		; SET DRIVE STATUS CHANGE DID NOT CLEAR
6387	025626	000407			BR	I.AERR		; GO PROCESS ERROR
6388								
6389	025630	105037	023616		16\$: CLRB	W.TIME		; RESET TIMING ON THIS DRIVE
6390	025634	005037	023632		CLR	W.DRV		; CLEAR DRIVE TIMING COUNT
6391	025640	004737	027230		JSR	PC,R.NORM		; REPORT SUCCESSFUL COMMAND COMPLETION
6392	025644	000563			BR	I.ATRN		; RESTORE REGISTERS
6393								

```

6394          .SBTTL      *ATTENTION ERROR HANDLER
6395
6396 025646 042765 000004 000014 I.AERR: BIC      #DRV PDT,P.PRST(R5) ;RESET POSITIONING IN PROGRESS BECAUSE
6397                                     OF DATA TRANSFER
6398 025654 105037 023616          CLR B          W.TIME          ;CLEAR TIMING FOR THIS DRIVE
6399 025660 005037 023632          CLR          W.DRV          ;RESET WATCH-DOG TIME
6400 025664 042765 177741 000016 BIC      #177741,P.CS1(R5) ;KEEP COMMAND ISSUED
6401 025672 042737 000036 023516 BIC      #36,T.CS1      ;KEEP CURRENT CONTROLLER STATUS
6402 025700 053765 023516 000016 BIS      T.CS1,P.CS1(R5) ;MAKE GOOD MESSAGE
6403 025706 013765 023520 000020 MOV      T.CS2,P.CS2(R5) ;STORE CONTROLLER REGISTERS
6404 025714 013765 023522 000022 MOV      T.WCR,P.WCR(R5)
6405 025722 013765 023524 000024 MOV      T.BA,P.BAR(R5)
6406 025730 013765 023526 000026 MOV      T.DA,P.DTS(R5)
6407 025736 013765 023530 000030 MOV      T.DC,P.DCYL(R5)
6408 025744 013765 023532 000032 MOV      T.ASOF,P.ASOF(R5)
6409 025752 013765 023534 000034 MOV      T.ER,P.ER(R5)
6410 025760 013765 023536 000036 MOV      T.DS,P.DS(R5)
6411 025766 004037 026770          JSR      RO,I.STAT      ;GATHER DRIVE STATUS
6412 025772 026214          I.RTRN      ;ERROR RETURN
6413 025774 112737 000005 023516 MOV B    #DR.CLR,T.CS1 ;LOAD COMMAND
6414 026002 004037 026314          JSR      RO,I.ISSU      ;CLEAR DRIVE ERRORS
6415 026006 026214          I.RTRN      ;ERROR RETURN
6416 026010 133737 023617 023533 BIT B    INTMSK,T.ASOF+1 ;CHECK IF ATTENTION RESET
6417 026016 001407          BEQ      2$      ;YES, FLAG DRIVE ERROR
6418 026020 052737 000020 023570 BIS      #E.CLAT,E.CONT ;SET ATTENTION DID NOT RESET
6419 026026 004737 027242          JSR      PC,R.CONT      ;REPORT ERROR
6420 026032 000137 026214          JMP      I.RTRN      ;RESTORE REGISTERS
6421
6422 026036 032765 000020 000014 2$: BIT      #DRVH RD,P.PRST(R5) ;CHECK IF AWARD DRIVE ERROR
6423 026044 001017          BNE      10$      ;YES, REPORT ERROR
6424 026046 032737 020000 023542 BIT      #S.PIP,T.MR2      ;CHECK IF DRIVE IS UNLOADING
6425 026054 001413          BEQ      10$      ;NO, REPORT ERROR
6426 026056 052765 020000 000014 BIS      #E.UNLD,P.PRST(R5) ;SET DRIVE UNLOADING DUE TO ERROR
6427 026064 113737 023617 023616 MOV B    INTMSK,W.TIME ;SET TIMING ON THIS DRIVE
6428 026072 013737 023602 023632 MOV      W.BSEC,W.DRV ;LOAD 8 SECONDS FOR CYCLE UP TIME
6429 026100 000137 026214          JMP      I.RTRN      ;RESTORE REGISTERS
6430
6431 026104 004737 027216          10$: JSR      PC,R.ABNL      ;REPORT ERROR
6432 026110 000137 026214          JMP      I.RTRN      ;RESTORE REGISTERS
6433
6434          .SBTTL      *ERROR CAUSING DRIVE TO UNLOAD
6435
6436 026114 052765 020000 000014 I.UNLD: BIS      #E.UNLD,P.PRST(R5) ;CLEAR DRIVE UNLOADING BECAUSE OF ERROR
6437 026122 112737 000005 023516 MOV B    #DR.CLR,T.CS1 ;LOAD IN DRIVE CLEAR
6438 026130 004037 026314          JSR      RO,I.ISSU      ;GO ISSUE DRIVE CLEAR
6439 026134 026214          I.RTRN      ;ERROR RETURN
6440 026136 136437 023617 023533 BIT B    INTMSK(R4),T.ASOF+1 ;CHECK IF ATTENTION CLEARED
6441 026144 001406          BEQ      15$      ;YES, CONTINUE
6442 026146 012737 000020 023570 MOV      #E.CLAT,E.CONT ;SET ATTENTION DID NOT RESET
6443 026154 004737 027242          JSR      PC,R.CONT      ;REPORT ERROR
6444 026160 000415          BR      I.RTRN      ;RESTORE REGISTERS
6445
6446 026162 032737 040000 023542 15$: BIT      #S.DSC,T.MR2      ;CHECK IF DRIVE STATUS CHANGE RESET
6447 026170 001403          BEQ      20$      ;YES, CONTINUE
6448 026172 052765 000040 000014 BIS      #DRV DSC,P.PRST(R5) ;SET DRIVE STATUS CHANGE DID NOT CLEAR
6449 026200 105037 023616          20$: CLR B    W.TIME          ;RESET TIMING ON THIS DRIVE

```

CZR6RC0 RK611/06 USR DEFINED  
CZR6RC.P11 02-DEC-77 11:07

MACY11 30(1046) 02-DEC-77 11:59 PAGE 121  
\*ERROR CAUSING DRIVE TO UNLOAD

SEQ 0120

6450	026204	005037	023632	CLR	W.DRV	;CLEAR TIME COUNT
6451	026210	004737	027216	JSR	PC,R.ABNL	;REPORT ERROR
6452						
6453	026214	012600		I.RTRN: MOV	(SP)+,R0	;RESTORE R0
6454	026216	012601		MOV	(SP)+,R1	;RESTORE R1
6455	026220	012602		MOV	(SP)+,R2	;RESTORE R2
6456	026222	012603		MOV	(SP)+,R3	;RESTORE R3
6457	026224	012604		MOV	(SP)+,R4	;RESTORE R4
6458	026226	012605		MOV	(SP)+,R5	;RESTORE R5
6459	026230	000002		RTI		;RETURN
6460						

.SBTTL \*CONTROLLER CLEAR ROUTINE

;\*\*\*\*\*

; THIS ROUTINE WILL BE USED BY THE DRIVER TO CLEAR THE CONTROLLER  
; AND CHECK IF THE CONTROLLER ERRORS ARE RESET. IF THE ERROR IS NOT  
; CLEARED, THE ROUTINE AS SPECIFIED IN A.CONT WILL BE CALLED WITH  
; E.CCLR SET IN E.CONT.; REGISTER USE  
; -----; R2 ADDRESS OF RK06 REGISTERS  
; R5 ADDRESS OF PARAMETER BLOCK; \*CALL JSR R0, I.CCLR  
; <ADDRESS OF ERROR RETURN>  
; RETURN

;\*\*\*\*\*

6461								
6462								
6463								
6464								
6465								
6466								
6467								
6468								
6469								
6470								
6471								
6472								
6473								
6474								
6475								
6476								
6477								
6478								
6479								
6480								
6481								
6482	026232	012762	100000	000000	I.CCLR: MOV	#CCLR, RKCS1(R2)	; CLEAR CONTROLLER	
6483	026240	016237	000000	023516	MOV	RKCS1(R2), T.CS1	; STORE COMMAND AND STATUS REGISTER 1	
6484	026246	032737	100000	023516	BIT	#CERR, T.CS1	; CHECK IF CONTROLLER CLEAR DID	
6485							; CLEAR ERROR	
6486	026254	001407			BEQ	SS	; YES, RETURN TO DRIVER PROCESSING	
6487	026256	052737	000001	023570	BIS	#E.CCLR, E.CONT	; SET CLEAR CONTROLLER DID NOT CLEAR ERROR	
6488	026264	004737	027242		JSR	PC, R.CONT	; REPORT CONTROLLER ERROR	
6489	026270	011000			MOV	(R0), R0	; SET UP ERROR RETURN	
6490	026272	000200			RTS	R0	; RETURN	
6491								
6492	026274	012762	000100	000000	SS: MOV	#IE, RKCS1(R2)	; SET INTERRUPT ENABLE	
6493	026302	112737	177777	023612	MOVB	#-1, I.ISRL	; SET INTERRUPT ENABLE ISSUED	
6494	026310	005720			TST	(R0)+	; ADJUST FOR NORMAL RETURN	
6495	026312	000200			RTS	R0	; RETURN	



.SBTTL \*COMMAND ISSUED BY DRIVER SERVICE ROUTINE

```

*****
* THIS ROUTINE WILL ISSUE THE COMMAND AS SPECIFIED IN T.CS1
* AND CHECK IF A CONTROLLER ERROR OCCURRED. IF A CONTROLLER
* ERROR OCCURRED, E.CERR WILL BE SET IN E.CONT AND
* CONTROL WILL BE TURN OVER TO THE ROUTINE SPECIFIED BY THE
* ADDRESS IN A.CONT.

```

```

REGISTER      USE
-----      ---

```

```

R2            ADDRESS OF RK06 REGISTERS
R5            ADDRESS OF PARAMETER BLOCK

```

```

*CALL JSR      RO I.ISSU
*      <ADDRESS OF ERROR RETURN>
*      RETURN

```

ROUTINES USED:

```

-----
I.CCLR
I.STOR

```

```

*****
I.ISSU: MOV      T.CS1, -(SP)      ;STORE COMMAND ISSUED
        CLR      T.CS2            ;CLEAR TEMPORARY CS2
        MOV      P.DRVN(R5), T.CS2 ;LOAD IN DRIVE NUMBER
        MOV      T.CS2, RKCS2(R2) ;LOAD DRIVE NUMBER FOR COMMAND
        MOV      P.CS1H(R5), T.CS1+1 ;STORE BITS 8-15 OF CS1
        BICB     #T.CB.CDT!B.CFMT, T.CS1+1 ;CLEAR ALL BITS EXCEPT
        ;FORMAT AND DRIVE TYPE
        MOV      T.CS1, RKCS1(R2) ;ISSUE COMMAND
15:      TSTB     RKCS1(R2)        ;WAIT FOR READY
        BPL      15$
        JSR      PC I.STOR        ;GO STORE REGISTERS
        BIT      #CERR, T.CS1     ;CHECK IF CONTROLLER ERROR OCCURED
        BEQ      55$              ;NO, RETURN
        BIT      #MDS, T.CS2      ;CHECK IF MULTIPLE DRIVE SELECT
        BEQ      25$              ;NO, CHECK FOR OTHER CONTROLLER ERRORS
        BIS      #E.MDS, E.CONT    ;SET MULTIPLE DRIVE SELECT FLAG
        JSR      PC R.CONT        ;REPORT CONTROLLER ERROR
        BR       105$             ;RETURN
        ;CHECK IF ANY CONTROLLER ERROR IS SET
25:      BIT      #CTO!SPAR, T.CS1
        BNE      75$
        BIT      #UFE!PGE!NEM!NED!UPE!WCE!DLT, T.CS2
        BNE      75$
        BIT      #ILC!DTYE!FMTE!ECH!BSE!HVRC!COE!DTE!OPI!DCK, T.ER
        BNE      75$
        CMPB     #DR.CLR, (SP)    ;CHECK IF CLEAR DRIVE

```

```

6496
6497
6498
6499
6500
6501
6502
6503
6504
6505
6506
6507
6508
6509
6510
6511
6512
6513
6514
6515
6516
6517
6518
6519
6520
6521
6522
6523
6524 026314 013746 023516
6525 026320 005037 023520
6526 026324 116537 000000 023520
6527 026332 013762 023520 000010
6528 026340 116537 000007 023517
6529 026346 142737 177753 023517
6530
6531 026354 013762 023516 000000
6532 026362 105762 000000
6533 026366 100375
6534 026370 004737 026536
6535 026374 032737 100000 023516
6536 026402 001437
6537 026404 032737 001000 023520
6538 026412 001406
6539 026414 052737 100000 023570
6540 026422 004737 027242
6541 026426 000440
6542
6543
6544 026430 032737 024000 023516 25:
6545 026436 001027
6546 026440 032737 176400 023520
6547 026446 001023
6548 026450 032737 131761 023534
6549 026456 001017
6550
6551 026460 122716 000005

```

## G10

CZR6RCO RK611/06 USR DEFINED  
CZR6RC.P11 02-DEC-77 11:07MACY11 30(1046) 02-DEC-77 11:59 PAGE 124  
\*COMMAND ISSUED BY DRIVER SERVICE ROUTINE

SEQ 0123

6552	026464	001003				BNE	3\$	:NO, DO NOT SET DRIVE HARD ERROR
6553	026466	052765	000020	000014		BIS	#DRVHRD,P.PRST(R5)	:SET HARD DRIVE ERROR
6554	026474	004037	026232		3\$:	JSR	RO,I.CCLR	:GO ISSUE A CONTROLLER CLEAR
6555	026500	026530				10\$		:ERROR RETURN
6556	026502	012762	000100	000000	5\$:	MOV	#IE,RKCS1(R2)	:SET INTERRUPT ENABLE
6557	026510	005726				TST	(SP)+	:ADJUST STACK
6558	026512	005720				TST	(RO)+	:ADJUST RO FOR NORMAL RETURN
6559	026514	000200				RTS	RO	:RETURN
6560								
6561	026516	052737	001000	023570	7\$:	BIS	#E.CERR,E.CONT	:SET CONTROLLER ERROR DURING
6562								: DRIVER SERVICING
6563	026524	004737	027242			JSR	PC,R.CONT	:REPORT ERROR
6564	026530	005726			10\$:	TST	(SP)+	:ADJUST STACK
6565	026532	011000				MOV	(RO),RO	:ADJUST RO FOR ERROR RETURN
6566	026534	000200				RTS	RO	:RETURN

CZR6RCO RK611/06 USR DEFINED  
CZR6RC.P11 02-DEC-77 11:07

MACY11 30(1046) 02-DEC-77 11:59 PAGE 125  
\*STORE RK611 UNIBUS REGISTERS

H10

SEQ 0124

.SBTTL \*STORE RK611 UNIBUS REGISTERS

```
*****  
* THIS SUBROUTINE IS CALLED BY THE RK06 DRIVER TO STORE ALL  
* RK611 REGISTER IN TEMPORARY LOCATIONS.  
*  
*CALL JSR PC,I.STOR  
* RETURN  
*  
* REGISTER USE  
* -----  
*  
* R2 ADDRESS OF RK611 REGISTERS  
*  
*****
```

6567				
6568				
6569				
6570				
6571				
6572				
6573				
6574				
6575				
6576				
6577				
6578				
6579				
6580				
6581				
6582				
6583				
6584	026536	016237	000000	023516
6585	026544	016237	000010	023520
6586	026552	016237	000002	023522
6587	026560	016237	000004	023524
6588	026566	016237	000006	023526
6589	026574	016237	000012	023536
6590	026602	016237	000014	023534
6591	026610	016237	000016	023532
6592	026616	016237	000020	023530
6593	026624	016237	000026	023540
6594	026632	016237	000034	023542
6595	026640	016237	000036	023544
6596	026646	016237	000030	023546
6597	026654	016237	000032	023550
6598	026662	000207		

```
I.STOR: MOV RKCS1(R2),T.CS1 ;STORE ALL CONTROLLER REGISTERS  
MOV RKCS2(R2),T.CS2 ; EXCEPT DATA BUFFER  
MOV RKWC(R2),T.WCR  
MOV RKBA(R2),T.BA  
MOV RKDA(R2),T.DA  
MOV RKDS(R2),T.DS  
MOV RKER(R2),T.ER  
MOV RKASOF(R2),T.ASOF  
MOV RKDCYL(R2),T.DC  
MOV RKMR1(R2),T.MR1  
MOV RKMR2(R2),T.MR2  
MOV RKMR3(R2),T.MR3  
MOV RKECPS(R2),T.POS  
MOV RKECPT(R2),T.PAT  
RTS PC ;RETURN
```

.SBTTL \*STORE CONTROLLER STATUS

```

*****
*
* THIS SUBROUTINE IS CALLED BY THE RK06 DRIVER AT PRIORITY 7.
* THE FOLLOWING REGISTERS WILL BE STORED:
*
*     COMMAND AND STATUS REGISTER 2
*     WORD COUNT REGISTER
*     BUS ADDRESS REGISTER
*     DESIRED TRACK AND SECTOR
*     STATUS REGISTER
*     ERROR REGISTER
*     ATTENTION SUMMARY/OFFSET REGISTER
*     CYLINDER ADDRESS REGISTER
*
*CALL  JSR    PC,I.CSTS
*      RETURN
*
* THIS ROUTINE ASSUMES THE FOLLOWING REGISTERS CONTAIN:
*
*     REGISTER      CONTENTS
*     -----
*
*     R2            RK06 BASE ADDRESS
*     R5            ADDRESS OF PARAMETER BLOCK
*****

```

6599							
6600							
6601							
6602							
6603							
6604							
6605							
6606							
6607							
6608							
6609							
6610							
6611							
6612							
6613							
6614							
6615							
6616							
6617							
6618							
6619							
6620							
6621							
6622							
6623							
6624							
6625							
6626							
6627							
6628							
6629	026664	042765	177741	000016	I.CSTS: BIC	#177741,P.CS1(R5)	; CLEAR ALL BITS EXCEPT FUNCTION
6630							; OF LAST COMMAND ISSUED
6631	026672	042737	000036	023516	BIC	#36,T.CS1	; CLEAR FUNCTION OF CS1 STATUS
6632	026700	053765	023516	000016	BIS	T.CS1,P.CS1(R5)	; GENERATE CS1 STATUS INFORMATION
6633	026706	016265	000010	000020	I.CST1: MOV	RKCS2(R2),P.CS2(R5)	; STORE COMMAND AND STATUS REGISTER 2
6634	026714	016265	000002	000022	MOV	RKWC(R2),P.WCR(R5)	; STORE WORD COUNT REGISTER
6635	026722	016265	000004	000024	MOV	RKBA(R2),P.BAR(R5)	; STORE BUS ADDRESS REGISTER
6636	026730	016265	000006	000026	MOV	RKDA(R2),P.DTS(R5)	; STORE DESIRED TRACK AND SECTOR
6637	026736	016265	000012	000036	MOV	RKDS(R2),P.DS(R5)	; STORE DRIVE STATUS REGISTER
6638	026744	016265	000014	000034	MOV	RKER(R2),P.ER(R5)	; STORE ERROR REGISTER
6639	026752	016265	000016	000032	MOV	RKASOF(R2),P.ASOF(R5)	; STORE ATTENTION SUMMARY AND
6640							; OFFSET
6641	026760	016265	000020	000030	MOV	RKDCYL(R2),P.DCYL(R5)	; STORE CYLINDER ADDRESS
6642	026766	000207			RTS	PC	; RETURN
6643							

.SBTTL \*GATHER DRIVE STATUS

```
*****
*
* THIS SUBROUTINE WILL BE USED TO GATHER DRIVE STATUS
* BYTE 01, 10, AND 11. IT IS ASSUMED THAT THE DRIVE
* HAS PREVIOUSLY BEEN SEIZED. IT RUNS AT PRIORITY 7.
*
*CALL JSR RO,I. STAT
*      <ADDRESS OF ERROR RETURN>
*      RETURN
*
* THIS ROUTINE ASSUMES THE FOLLOWING REGISTERS CONTAIN:
*
*      REGISTER      CONTENTS
*      -----
*
*      R2            RK06 BASE ADDRESS
*      R5            ADDRESS OF PARAMETER BLOCK
*
* ROUTINES USED:
*      I.ISSU
*****
```

6670	026770	012762	000001	000026	I. STAT: MOV	#1, RKMR1(R2)	: LOAD MAINTENANCE REGISTER 1
6671							: FOR STATUS BYTE 01
6672	026776	112737	000001	023516	MOVB	#DR. SEL, T.CS1	: LOAD COMMAND
6673	027004	004037	026314		JSR	RO, I.ISSU	: GET STATUS BYTES 01
6674	027010	027200			3\$		: ERROR RETURN
6675	027012	013765	023542	000044	MOV	T.MR2, P.A01(R5)	: STORE STATUS BYTE 01 MESS A
6676	027020	013765	023544	000046	MOV	T.MR3, P.B01(R5)	: STORE STATUS BYTE 01 MESS B
6677	027026	012762	000002	000026	MOV	#2, RKMR1(R2)	: LOAD MAINTENANCE REGISTER 1
6678							: FOR STATUS BYTE 10
6679	027034	112737	000001	023516	MOVB	#DR. SEL, T.CS1	: LOAD COMMAND
6680	027042	004037	026314		JSR	RO, I.ISSU	: GET STATUS BYTES 10
6681	027046	027200			3\$		: ERROR RETURN
6682	027050	013765	023542	000050	MOV	T.MR2, P.A10(R5)	: STORE STATUS BYTE 10 MESS A
6683	027056	013765	023544	000052	MOV	T.MR3, P.B10(R5)	: STORE STATUS BYTE 10 MESS B
6684	027064	012762	000003	000026	MOV	#3, RKMR1(R2)	: LOAD MAINTENANCE REGISTER 1
6685							: FOR STATUS BYTE 11
6686	027072	112737	000001	023516	MOVB	#DR. SEL, T.CS1	: LOAD COMMAND
6687	027100	004037	026314		JSR	RO, I.ISSU	: GET STATUS BYTES 11
6688	027104	027200			3\$		: ERROR RETURN
6689	027106	013765	023542	000054	MOV	T.MR2, P.A11(R5)	: STORE STATUS BYTE 11 MESS A
6690	027114	013765	023544	000056	MOV	T.MR3, P.B11(R5)	: STORE STATUS BYTE 11 MESS B
6691	027122	005062	000026		CLR	RKMR1(R2)	: LOAD MAINTENANCE REGISTER 1
6692							: FOR STATUS BYTE 00
6693	027126	112737	000001	023516	MOVB	#DR. SEL, T.CS1	: LOAD COMMAND
6694	027134	004037	026314		JSR	RO, I.ISSU	: GET STATUS BYTES 00
6695	027140	027200			3\$		: ERROR RETURN
6696	027142	013765	023542	000040	MOV	T.MR2, P.A00(R5)	: STORE STATUS BYTE 00 MESS A
6697	027150	013765	023544	000042	MOV	T.MR3, P.B00(R5)	: STORE STATUS BYTE 00 MESS B
6698	027156	032737	001000	023544	BIT	#S. PAR, T.MR3	: CHECK IF BAD PARITY DETECTED BY DRIVE
6699	027164	001407			BEQ	SS	: NO, RETURN NORMALLY

K10

CZR6RCO RK611/06 USR DEFINED  
CZR6RC.P11 02-DEC-77 11:07MACY11 30(1046) 02-DEC-77 11:59 PAGE 128  
\*GATHER DRIVE STATUS

SEQ 0127

6700	027166	052737	002000	023570	BIS	#E.DPAR,E.CONT	;INDICATE BAD PARITY DETECTED BY DRIVE
6701	027174	004737	027242		JSR	PC,R.CONT	;REPORT ERROR
6702	027200	011000		3\$:	MOV	(R0),R0	;LOAD R0 FOR ERROR RETURN
6703	027202	000200			RTS	R0	;RETURN
6704							
6705	027204	052765	001000	000014	5\$:	BIS	#PBSVAL,P.PRST(R5)
6706	027212	005720			TST	(R0),+	;SET PARAMETER BLOCK STATUS VALID
6707	027214	000200			RTS	R0	;ADJUST R0 FOR NORMAL RETURN
6708							;RETURN

CZR6RCO RK611/06 USR DEFINED  
CZR6RC.P11 02-DEC-77 11:07

MACY11 30(1046) 02-DEC-77 11:59 PAGE 129  
\*COMMON DRIVER RETURNS

L10

SEQ 0128

6709				.SBTTL	*COMMON DRIVER RETURNS	
6710						
6711	027216	105037	023617	R.ABNL:	CLRB INTMSK	;INHIBIT FUTURE DRIVE INTERRUPT REPORTING
6712	027222	004777	174336		JSR PC,QA.ABNL	;INDICATE ABNORMAL RETURN
6713	027226	000207			RTS PC	;RETURN
6714						
6715	027230	105037	023617	R.NORM:	CLRB INTMSK	;INHIBIT FUTURE DRIVE INTERRUPT REPORTING
6716	027234	004777	174322		JSR PC,QA.NORM	;INDICATE NORMAL RETURN
6717	027240	000207			RTS PC	;RETURN
6718						
6719	027242	105037	023617	R.CONT:	CLRB INTMSK	;INHIBIT FUTURE DRIVE INTERRUPT REPORTING
6720	027246	105037	023616		CLRB W.TIME	;RESET WATCH DOG TIMING ON THIS DRIVE
6721	027252	005037	023632		CLR W.DRV	;CLEAR TIMING COUNT FOR THIS DRIVE
6722	027256	004777	174304		JSR PC,QA.CONT	;INDICATE CONTROLLER ERROR RETURN
6723	027262	000207			RTS PC	;RETURN

M10

.SBTTL \*COMMAND INITATOR

\*\*\*\*\*  
\* THIS SUBROUTINE WILL INITIATE ALL COMMANDS AS SPECIFIED  
\* BY THE COMMAND FIELD OF THE PARAMETER BLOCK. THE FOLLOWING  
\* SPECIAL COMMAND ARE ALSO EXECUTED:

RELEASE  
CONROLLER CLEAR  
SUBSYSTEM CLEAR  
READ ALL DRIVE STATUS  
READ SPECIFIED HEADER

THE ABOVE COMMANDS ARE TRANSLATED INTO A SEQUENCE OF COMMANDS

\*CALL JSR PC.C.INIT  
<ADDRESS OF PARAMETER BLOCK>  
RETURN

FOR THE SEQUENTIAL OPERATIONS, THE DRIVER WILL LOAD THE  
LOCATIONS, PBLKT AND INTMSK.

ROUTINES USED:

W.WTCH  
I.CSTS  
I.STAT  
I.CCLR

\*\*\*\*\*

C.INIT: MOV R5, -(SP) ;STORE R5 ON STACK  
MOV R4, -(SP) ;STORE R4 ON STACK  
MOV R3, -(SP) ;STORE R3 ON STACK  
MOV R2, -(SP) ;STORE R2 ON STACK  
MOV R1, -(SP) ;STORE R1 ON STACK  
MOV R0, -(SP) ;STORE R0 ON STACK  
MOV PS, -(SP) ;STORE PSW ON STACK  
MOV RKPRI, PS ;LOCK OUT RK06 INTERRUPTS  
MOV @16(SP), R5 ;STORE PARAMETER BLOCK ADDRESS  
ADD #2, 16(SP) ;ADJUST RETURN  
MOV P.DRVN(R5), R4 ;STORE DRIVE NUMBER  
BIC #1C<DRVMSK>, R4 ;MASK OUT JUNK  
MOV R5, PBLKT ;LOAD PARAMETER BLOCK TABLE  
MOVB I.DRV(R4), INTMSK ;LOAD INTERRUPT MASK  
MOVB I.DRV(R4), W.TIME ;SET WATCH-DOG TIMER FLAG  
MOV W.SEC, W.DRV ;LOAD WATCH-DOG TIME

MOV RKBAS, R2 ;LOAD R2 WITH RK06 ADDRESS BASE

RESET ALL BITS IN PROGRAM DEVICE STATUS REGISTER EXCEPT  
DRIVE IN USE  
WRITE FOR WRITE CHECK  
NO CHECK  
DROP DRIVE FROM TEST SEQUENCE  
INHIBIT BUS ADDRESS INCREMENT

6724  
6725  
6726  
6727  
6728  
6729  
6730  
6731  
6732  
6733  
6734  
6735  
6736  
6737  
6738  
6739  
6740  
6741  
6742  
6743  
6744  
6745  
6746  
6747  
6748  
6749  
6750  
6751  
6752  
6753  
6754  
6755 027264 010546  
6756 027266 010546  
6757 027270 010546  
6758 027272 010246  
6759 027274 010146  
6760 027276 010046  
6761 027300 013746 177776  
6762 027304 013737 023560 177776  
6763 027312 017605 000016  
6764 027316 062766 000002 000016  
6765 027324 016504 000000  
6766 027330 042704 177770  
6767 027334 010537 023630  
6768 027340 116437 023620 023617  
6769 027346 116437 023620 023616  
6770 027354 013737 023600 023632  
6771  
6772 027362 013702 023554  
6773  
6774  
6775  
6776  
6777  
6778  
6779



6780	027366	042765	075176	000014		BIC	#1C<DRVUSE!W.WCK!NOCHK!DRPDV!DTBAII>,P.PRST(R5)	
6781								
6782	027374	010500				MOV	R5,R0	;STORE PARAMETER BLOCK ADDRESS
6783	027376	062700	000016			ADD	#P.CS1,R0	;CALCULATE FIRST LOCATION TO BE CLEARED
6784	027402	010501				MOV	R5,R1	;STORE PARAMETER BLOCK ADDRESS
6785	027404	062701	000062			ADD	#P.EPAT,R1	;CALCULATE LAST LOCATION TO BE CLEARED
6786								
6787	027410	005020			1\$:	CLR	(R0)+	;CLEAR RETURN PARAMETER
6788	027412	020001				CMP	R0,R1	;CHECK IF FINISHED
6789	027414	101775				BLOS	1\$	;NO, CLEAR NEXT RETURN PARAMETER
6790	027416	105037	023612			CLRB	I.ISRL	;CLEAR RELEASE OR INTERRUPT ISSUED
6791	027422	010465	000020			MOV	R4,P.CS2(R5)	;STORE DRIVE NUMBER
6792	027426	005062	000026			CLR	RKMR1(R2)	;CLEAR RK06 MAINTENANCE REGISTER 1
6793	027432	132765	000040	000001		BITB	#BITS,P.CMND(R5)	;CHECK IF SPECIAL COMMAND
6794	027440	001402				BEQ	3\$	;NO, PROCESS
6795	027442	000137	030156			JMP	C.SPEC	;JUMP TO SPECIAL COMMAND PROCESSOR
6796								
6797	027446	122765	000107	000001	3\$:	CMPB	#UNLOAD,P.CMND(R5)	;CHECK IF POSITIONING COMMAND
6798								START SPINDLE
6799								RECALIBRATE
6800								OFFSET
6801								SEEK
6802								UNLOAD
6803								
6804	027454	101174				BHI	25\$	;NO, DRIVE COMMAND
6805								SELECT DRIVE
6806								PACK ACKNOWLEDGE
6807								CLEAR
6808								
6809	027456	122765	000117	000001		CMPB	#SEEK,P.CMND(R5)	;CHECK IF DATA TRANSFER
6810	027464	103540				BLO	20\$	;YES, DATA TRANSFER COMMAND
6811								READ DATA
6812								WRITE DATA
6813								READ HEADER
6814								WRITE HEADER
6815								WRITE CHECK
6816	027466	016562	000020	000010		MOV	P.CS2(R5),RKCS2(R2)	;LOAD DRIVE NUMBER
6817	027474	052765	000002	000014		BIS	#DRVPOS,P.PRST(R5)	;SET DRIVE POSITIONING
6818	027502	005037	023572			CLR	0.WAIT	;CLEAR WAIT FOR COMMAND
6819	027506	122765	000117	000001		CMPB	#SEEK,P.CMND(R5)	;CHECK IF SEEK
6820	027514	001007				BNE	5\$	;NO, CHECK FOR OFFSET
6821	027516	016562	000002	000020		MOV	P.CYLN(R5),RKDCY(R2)	;LOAD CYLINDER ADDRESS
6822	027524	016562	000004	000006		MOV	P.SECT(R5),RKDA(R2)	;LOAD SECTOR AND TRACK
6823	027532	000431				BR	8\$	;GO ISSUE COMMAND
6824								
6825	027534	122765	000115	000001	5\$:	CMPB	#OFFSET,P.CMND(R5)	;CHECK IF OFFSET
6826	027542	001007				BNE	6\$	;NO, CHECK FOR UNLOAD
6827	027544	116565	000006	000032		MOVB	P.OFST(R5),P.ASOF(R5)	;STORE OFFSET
6828	027552	016562	000032	000016		MOV	P.ASOF(R5),RKASOF(R2)	;LOAD OFFSET REGISTER
6829	027560	000416				BR	8\$	;GO ISSUE COMMAND
6830								
6831	027562	122765	000111	000001	6\$:	CMPB	#SRTSPL,P.CMND(R5)	;CHECK IF START SPINDLE
6832	027570	001003				BNE	7\$	;NO, CHECK IF RECAL
6833	027572	013737	023604	023632		MOV	W.MIN,W.DRV	;LOAD WATCH DOG TIME FOR 1 MINUTE
6834	027600	122765	000113	000001	7\$:	CMPB	#RECAL,P.CMND(R5)	;CHECK IF RECAL
6835	027606	001003				BNE	8\$	;NO, CONTINUE

6836	027610	013737	023602	023632		MOV	W.BSEC,W.DRV	:LOAD RECAL TIME FOR 8 SECONDS
6837	027616	116565	000007	000017	8\$:	MOVB	P.CS1H(R5),P.CS1+1(R5)	:STORE BITS 8-15 OF CS1
6838	027624	042765	165777	000016		BIC	#1C<CFMT!COT>,P.CS1(R5)	:CLEAR ALL BITS EXCEPT FORMAT
6839								:AND DRIVE TYPE
6840	027632	116565	000001	000016		MOVB	P.CMND(R5),P.CS1(R5)	:MOVE COMMAND INTO CS1
6841	027640	042765	000200	000014		BIC	#W.WCK,P.PRST(R5)	:RESET WRITE FOR WRITE CHECK
6842	027646	032765	000400	000014		BIT	#NOCHK,P.PRST(R5)	:CHECK IN NO CHECK MODE
6843	027654	001533				BEQ	30\$	:NO, SKIP CLEAR OF INTERRUPT ENABLE
6844	027656	042765	000100	000016		BIC	#IE,P.CS1(R5)	:CLEAR INTERRUPT ENABLE
6845	027664	016562	000016	000000		MOV	P.CS1(R5),RKCS1(R2)	:ISSUE COMMAND
6846	027672	004737	023634		10\$:	JSR	PC,W.WTCH	:CALL WATCH DOG TIMER
6847	027676	016237	000000	023516		MOV	RKCS1(R2),T.CS1	:STORE COMMAND AND STATUS REGISTER 1
6848	027704	032737	000200	023516		BIT	#RDY,T.CS1	:WAIT FOR READY
6849	027712	001767				BEQ	10\$	
6850	027714	032737	100000	023516		BIT	#CERR,T.CS1	:CHECK FOR ERROR
6851	027722	001011				BNE	15\$	:YES, GIVE NORMAL RETURN
6852	027724	004737	023634		11\$:	JSR	PC,W.WTCH	:CALL WATCH DOG TIMER
6853	027730	016237	000016	023532		MOV	RKASOF(R2),T.ASOF	:STORE ATTENTION SUMMARY
6854	027736	133737	023617	023533		BITB	INTMSK,T.ASOF+1	:CHECK IF INTERRUPT HAS OCCURRED
6855	027744	001767				BEQ	11\$	:WAIT FOR DRIVE INTERRUPT
6856	027746	105037	023616		15\$:	CLRB	W.TIME	:RESET TIMING ON THIS DRIVE
6857	027752	005037	023632			CLR	W.DRV	:CLEAR DRIVE TIMING COUNT
6858	027756	004737	027230			JSR	PC,R.NORM	:INDICATE COMMAND IS FINISHED
6859	027762	000137	031136			JMP	C.ATRN	:RESTORE REGISTERS
6860								
6861	027766	016562	000010	000004	20\$:	MOV	P.BALO(R5),RKBA(R2)	:LOAD BUS ADDRESS REGISTER
6862	027774	016562	000012	000002		MOV	P.WC(R5),RKWC(R2)	:LOAD WORD COUNT REGISTER
6863	030002	016562	000002	000020		MOV	P.CYLN(R5),RKDCYL(R2)	:LOAD CYLINDER ADDRESS REGISTER
6864	030010	016562	000004	000006		MOV	P.SECT(R5),RKDA(R2)	:LOAD SECTOR AND TRACK NUMBER
6865	030016	122765	000131	000001		CMPB	#WATCHK,P.CMND(R5)	:CHECK IF WRITE CHECK COMMAND
6866	030024	001010				BNE	25\$	:NO, GO ISSUE THE COMMAND
6867	030026	032765	000200	000014		BIT	#W.WCK,P.PRST(R5)	:CHECK IF WRITE COMMAND SHOULD BE ISSUED
6868	030034	001404				BEQ	25\$	:NO, GO ISSUE THE COMMAND
6869	030036	012765	000123	000016		MOV	#WRDATA,P.CS1(R5)	:ISSUE WRITE COMMAND
6870	030044	000406				BR	26\$	:GO ISSUE COMMAND
6871								
6872	030046	116565	000001	000016	25\$:	MOVB	P.CMND(R5),P.CS1(R5)	:MOVE COMMAND INTO CS1
6873	030054	042765	000200	000014		BIC	#W.WCK,P.PRST(R5)	:RESET WRITE FOR WRITE CHECK
6874	030062	116565	000007	000017	26\$:	MOVB	P.CS1H(R5),P.CS1+1(R5)	:STORE BITS 8-15 OF CS1
6875	030070	142765	177750	000017		BICB	#1C<B.CFMT!B.COT!B.BA16!B.BA17>,P.CS1+1(R5)	:CLEAR ALL BITS EXCEPT
6876								:FORMAT, DRIVE TYPE, AND BUS ADDRESS
6877								:BITS 16-17
6878	030076	010537	023572			MOV	R5,0.WAIT	:LOAD WAITING FOR COMMAND
6879	030102	032765	100000	000014		BIT	#DIBAI,P.PRST(R5)	:CHECK IF INHIBIT BUS ADDRESS INCREMENT
6880	030110	001403				BEQ	27\$	:NO, LOAD CS2
6881	030112	052765	000020	000020		BIS	#BAI,P.CS2(R5)	:SET INHIBIT BUS ADDRESS INCREMENT
6882	030120	016562	000020	000010	27\$:	MOV	P.CS2(R5),RKCS2(R2)	:LOAD CS2
6883	030126	032765	000400	000014		BIT	#NOCHK,P.PRST(R5)	:CHECK IN NO CHECK MODE
6884	030134	001403				BEQ	30\$	:NO, SKIP CLEAR OF INTERRUPT ENABLE
6885	030136	042765	000100	000016		BIC	#IE,P.CS1(R5)	:CLEAR INTERRUPT ENABLE
6886	030144	016562	000016	000000	30\$:	MOV	P.CS1(R5),RKCS1(R2)	:ISSUE COMMAND
6887	030152	000137	031136			JMP	C.RTRN	:RESTORE REGISTERS
6888								
6889								
6890								
6891	030156	122765	000141	000001		C.SPEC: CMPB	#RDSTAT,P.CMND(R5)	:CHECK IF READ DRIVE STATUS

.SBTTL \*SPECIAL COMMAND PROCESSING

6892	030164	001132			BNE	10\$	:NO, PROCESS OTHER COMMANDS
6893	030166	016562	000020	000010	MOV	P.CS2(R5),RKCS2(R2)	:LOAD CS2 FOR COMMAND
6894	030174	116565	000007	000017	MOVB	P.CS1H(R5),P.CS1+1(R5)	:STORE BITS 8-15 OF CS1
6895	030202	042765	165777	000016	BIC	#1C<CFMT!COT>,P.CS1(R5)	:CLEAR ALL BITS EXCEPT FORMAT
6896							: AND DRIVE TYPE
6897	030210	112765	000001	000016	MOVB	#DR,SEL,P.CS1(R5)	:STORE COMMAND
6898	030216	016562	000016	000000	MOV	P.CS1(R5),RKCS1(R2)	:ISSUE COMMAND
6899	030224	004737	023634		JSR	PC.W.WTCH	:CALL WATCH-DOG TIMER
6900	030230	016265	000000	000016	MOV	RKCS1(R2),P.CS1(R5)	:STORE COMMAND AND STATUS REG. 1
6901	030236	032765	000200	000016	BIT	#RDY,P.CS1(R5)	:WAIT FOR READY
6902	030244	001767			BEQ	2\$	
6903	030246	004737	026706		JSR	PC.I.CST1	:STORE CONTROLLER REGISTERS
6904	030252	016265	000034	000040	MOV	RKMR2(R2),P.A00(R5)	:STORE STATUS BYTE 00 MESSAGE A
6905	030260	016265	000036	000042	MOV	RKMR3(R2),P.B00(R5)	:STORE STATUS BYTE 00 MESSAGE B
6906	030266	032765	100000	000016	BIT	#CERR,P.CS1(R5)	:CHECK IF CONTROLLER ERROR
6907	030274	001436			BEQ	6\$	:NO, GATHER DRIVE STATUS
6908	030276	105037	023616		CLRB	W.TIME	:RESET WATCH DOG TIMING ON THIS DRIVE
6909	030302	005037	023632		CLR	W.DRV	:CLEAR WATCH DOG COUNT
6910	030306	032765	000400	000014	BIT	#NOCHK,P.PRST(R5)	:CHECK IF NO CHECK MODE
6911	030314	001043			BNE	8\$	:YES, INDICATE NORMAL RETURN
6912	030316	032765	001000	000020	BIT	#MDS,P.CS2(R5)	:CHECK IF MULTIPLE DRIVE SELECT
6913	030324	001043			BNE	9\$	:YES, INDICATE CONTROLLER ERROR
6914	030326	004037	026232		JSR	RO,I.CCLR	:CLEAR ERROR
6915	030332	031136			C.RTRN		:ERROR RETURN
6916	030334	032765	010400	000020	BIT	#NED!UFE,P.CS2(R5)	:CHECK IF NON-EXISTENT DRIVE OR UNIT FIELD ERROR
6917	030342	001007			BNE	5\$	:REPORT ERROR
6918	030344	032765	000001	000036	BIT	#DRA,P.DS(R5)	:CHECK IF DRIVE AVAILABLE
6919	030352	001003			BNE	5\$	:YES, REPORT ERROR
6920	030354	052765	010000	000014	BIS	#DRVSZD,P.PRST(R5)	:INDICATE DRIVE IS SEIZED BY OTHER PORT
6921	030362	004737	027216		JSR	PC.R.ABNL	:INDICATE ABNORMAL RETURN
6922	030366	000137	031136		JMP	C.RTRN	:RESTORE REGISTERS
6923							
6924	030372	004037	026770		JSR	RO,I.STAT	:GATHER DRIVE STATUS
6925	030376	031136			C.RTRN		:ERROR RETURN
6926	030400	105037	023616		CLRB	W.TIME	:STOP WATCH-DOG TIMING ON DRIVE
6927	030404	005037	023632		CLR	W.DRV	:RESET WATCH-DOG TIME
6928	030410	032765	000400	000014	BIT	#NOCHK,P.PRST(R5)	:CHECK IF NO CHECK MODE
6929	030416	001402			BEQ	8\$	:NO, REPORT ERROR
6930	030420	005062	000000		CLR	RKCS1(R2)	:CLEAR INTERRUPT ENABLE
6931	030424	004737	027230		JSR	PC.R.NORM	:REPORT COMMAND COMPLETE
6932	030430	000137	031136		JMP	C.RTRN	:RESTORE REGISTERS
6933							
6934	030434	052737	100000	023570	BIS	#E.MDS,E.CONT	:SET MULTIPLE DRIVE SELECT
6935	030442	004737	027242		JSR	PC.R.CONT	:INDICATE CONTROLLER ERROR
6936	030446	000137	031136		JMP	C.RTRN	
6937							
6938	030452	122765	000140	000001	10\$: CMPB	#RELEAS,P.CMND(R5)	:CHECK IF RELEASE COMMAND
6939	030460	001040			BNE	13\$	:NO, CHECK IF READ ALL HEADERS
6940	030462	010537	023572		MOV	RS,O.WAIT	:STORE PARAMETER BLOCK ADDRESS IN
6941							: WAIT FOR COMMAND
6942	030466	052765	000010	000020	BIS	#RLS,P.CS2(R5)	:SET RELEASE BIT
6943	030474	016562	000020	000010	MOV	P.CS2(R5),RKCS2(R2)	:LOAD CS2 FOR DESELECT
6944	030502	112737	000001	023612	MOVB	#1,I.ISRL	:SET FLAG FOR RELEASE COMMAND
6945	030510	116565	000007	000017	MOVB	P.CS1H(R5),P.CS1+1(R5)	:STORE BITS 8-15 OF CS1
6946	030516	042765	165777	000016	BIC	#1C<CFMT!COT>,P.CS1(R5)	:CLEAR ALL BITS EXCEPT FORMAT
6947							: AND DRIVE TYPE

6948	030524	112765	000101	000016		MOV	#SELDIV,P.CS1(R5) ;STORE COMMAND
6949	030532	032765	000400	000014		BIT	#NOCHK,P.PRST(R5) ;CHECK IF NO CHECK MODE
6950	030540	001403				BEQ	11\$ ;NO DO NOT RESET INTERRUPT ENABLE
6951	030542	042765	000100	000016		BIC	#IE,P.CS1(R5) ;RESET INTERRUPT ENABLE
6952	030550	016562	000016	000000	11\$:	MOV	P.CS1(R5),RKCS1(R2) ;ISSUE COMMAND
6953	030556	000137	031136			JMP	C.RTRN ;RESTORE REGISTERS
6954							
6955	030562	122765	000164	000001	13\$:	CMPB	#RDALHD,P.CMND(R5) ;CHECK IF READ ALL HEADERS
6956	030570	001053				BNE	30\$ ;NO CHECK IF CONTROLLER CLEAR
6957	030572	010537	023572			MOV	RS,0.WAIT ;SET WAITING FOR COMMAND COMPLETION
6958	030576	016537	000010	023606		MOV	P.BAL0(R5),HDR.A0 ;LOAD HEADER ADDRESS
6959	030604	132765	000020	000007		BITB	#B.CFMT,P.CS1H(R5) ;CHECK IF 22 SECTOR FORMANT
6960	030612	001404				BEQ	14\$ ;YES LOAD 22 IN HEADER COUNT
6961	030614	012737	000024	023610		MOV	#20.,HDR.CT ;LOAD 27 IN SECTOR COUNT
6962	030622	000403				BR	22\$ ;GO ISSUE READ HEADER COMMAND
6963							
6964	030624	012737	000026	023610	14\$:	MOV	#22.,HDR.CT ;LOAD 22 IN SECTOR COUNT
6965	030632	016562	000002	000020	22\$:	MOV	P.CYL(N(R5),RKDCYL(R2) ;LOAD CYLINDER ADDRESS
6966	030640	016562	000004	000006		MOV	P.SECT(R5),RKDA(R2) ;LOAD TRACK NUMBER
6967	030646	016562	000020	000010		MOV	P.CS2(R5),RKCS2(R2) ;LOAD DRIVE NUMBER
6968	030654	116565	000007	000017		MOVB	P.CS1H(R5),P.CS1+1(R5) ;STORE BITS 8-15 OF CS1
6969	030662	042765	165777	000016		BIC	#1C<CFMT!CDT>,P.CS1(R5) ;CLEAR ALL BITS EXCEPT DRIVE TYPE
6970							AND FORMAT
6971	030670	112765	000125	000016		MOVB	#RDHEAD,P.CS1(R5) ;STORE READ HEADER COMMAND
6972	030676	032765	000400	000014		BIT	#NOCHK,P.PRST(R5) ;CHECK IF NO CHECK MODE
6973	030704	001027				BNE	34\$ ;YES INDICATE ILLEGAL DRIVER COMMAND
6974	030706	016562	000016	000000		MOV	P.CS1(R5),RKCS1(R2) ;ISSUE READ HEADER
6975	030714	000137	031136			JMP	C.RTRN ;RESTORE REGISTERS
6976							
6977	030720	122765	000176	000001	30\$:	CMPB	#CONCLR,P.CMND(R5) ;CHECK IF CONTROLLER CLEAR
6978	030726	001012				BNE	32\$ ;NO CHECK IF SUBSYSTEM CLEAR
6979	030730	004037	026232			JSR	RD,I.CCLR ;CLEAR CONTROLLER
6980	030734	031136				C.RTRN	ERROR RETURN
6981	030736	032765	000400	000014		BIT	#NOCHK,P.PRST(R5) ;CHECK IF NO CHECK MODE
6982	030744	001472				BEQ	40\$ ;NO INDICATE NORMAL RETURN
6983	030746	005062	000000			CLR	RKCS1(R2) ;RESET INTERRUPT ENABLE
6984	030752	000467				BR	40\$ ;INDICATE NORMAL RETURN
6985							
6986	030754	122765	000177	000001	32\$:	CMPB	#SUBCLR,P.CMND(R5) ;CHECK IF SUBSYSTEM CLEAR
6987	030762	001406				BEQ	36\$ ;YES CLEAR SUBSYSTEM
6988	030764	052737	000100	023570	34\$:	BIS	#E.ILLD,E.CONT ;SET ILLEGAL DRIVER COMMAND
6989	030772	004737	027242			JSR	PC,R.CONT ;REPORT ERROR
6990	030776	000457				BR	C.RTRN ;RESTORE REGISTERS
6991							
6992	031000	012762	000040	000010	36\$:	MOV	#SCLR,RKCS2(R2) ;ISSUE SUBSYSTEM CLEAR
6993	031006	016265	000000	000016		MOV	RKCS1(R2),P.CS1(R5) ;STORE COMMAND AND STATUS REGISTER 1
6994	031014	032765	100000	000016		BIT	#CERR,P.CS1(R5) ;CLEAR IF CONTROLLER ERROR RESET
6995	031022	001406				BEQ	37\$ ;NO FINISH COMMAND
6996	031024	052737	000001	023570		BIS	#BIT0,E.CONT ;SET CLEAR SUBSYSTEM DID NOT CLEAR
6997							CONTROLLER ERROR
6998	031032	004737	027242			JSR	PC,R.CONT ;REPORT ERROR
6999	031036	000437				BR	C.RTRN ;RESTORE REGISTERS
7000							
7001	031040	013746	023576		37\$:	MOV	W.MILI,-(SP) ;LOAD 16 MILI-SECOND COUNT FOR ATTENTION
7002							TO DISAPPEAR
7003	031044	016265	000000	000016	38\$:	MOV	RKCS1(R2),P.CS1(R5) ;STORE CS1

```

7004 0310 032765 040000 000016 BIT #DI,P.CS1(R5) ;CHECK IF ATTENTIONS CLEARED
7005 031060 001411 BEQ 39$ ;YES, FINISH COMMAND
7006 031062 005316 DEC (SP) ;DECREMENT 16 MILISCOND COUNT
7007 031064 001367 BNE 38$ ;CHECK DRIVE INTERRUPT AGAIN
7008 031066 005726 TST (SP)+ ;ADJUST STACK
7009 031070 052737 000040 023570 BIS #E.SCLR,E.CONT ;SET SUBSYSTEM CLEAR DID NOT CLEAR
7010 ;DRIVE ATTENTIONS
7011 031076 004737 027242 JSR PC,R.CONT ;REPORT ERROR
7012 031102 000415 BR C.RTRN ;RESTORE REGISTER
7013
7014 031104 005726 39$: TST (SP)+ ;ADJUST STACK
7015 031106 032765 000400 000014 BIT #NOCHK,P.PRST(R5) ;CHECK IF NO CHECK MODE
7016 031114 001010 BNE C.RTRN ;YES, RESTORE REGISTERS
7017 031116 112737 177777 023612 MOVB #-1,I.ISRL ;SET INTERRUPT ENABLE SET
7018 031124 012762 000100 000000 MOV #IE,RKCS1(R2) ;SET INTERRUPT ENABLE
7019 031132 004737 027230 40$: JSR PC,R.NORM ;INDICATE NORMAL TERMINATION
7020
7021 031136 012637 177776 C.RTRN: MOV (SP)+,PS ;RESTORE PSW
7022 031142 012600 MOV (SP)+,R0 ;RESTORE R0
7023 031144 012601 MOV (SP)+,R1 ;RESTORE R1
7024 031146 012602 MOV (SP)+,R2 ;RESTORE R2
7025 031150 012603 MOV (SP)+,R3 ;RESTORE R3
7026 031152 012604 MOV (SP)+,R4 ;RESTORE R4
7027 031154 012605 MOV (SP)+,R5 ;RESTORE R5
7028 031156 000207 RTS PC ;RETURN
7029
7030 .SBTTL OCTAL TO BINARY CONVERSION ROUTINE
7031
7032 ;*****
7033 ;
7034 ; THIS ROUTINE WILL CHECK A STRING OF ASCII CHARACTERS TERMINATED
7035 ; WITH A NULL <000> OR COMMA. IF THE CHARACTERS ARE LEGAL
7036 ; IT WILL GENERATE TWO BINARY WORDS PLACING THE LOW 16 BITS
7037 ; ON THE STACK AND THE HIGH 16 BITS IN LOCATION $HIOCT.
7038 ;
7039 ;CALL
7040 ; MOV <ADDRESS OF ASCII STRING>,-(SP)
7041 ; JSR PC,OCTBIN
7042 ; <ADDRESS OF ERROR RETURN>
7043 ; RETURN
7044 ;
7045 ;*****
7046 031160 010046 OCTBIN: MOV R0,-(SP) ;SAVE R0
7047 031162 010146 MOV R1,-(SP) ;SAVE R1
7048 031164 010246 MOV R2,-(SP) ;SAVE R2
7049 031166 016600 000010 MOV 10(SP),R0 ;GET ADDRESS OF ASCII STRING
7050 031172 005001 CLR R1 ;CLEAR DATA WORDS
7051 031174 005002 CLR R2
7052 031176 112046 2$: MOVB (R0)+,-(SP) ;PICK THIS CHARACTER
7053 031200 001423 BEQ 3$ ;IF ZERO GET OUT
7054 031202 121627 000054 CMPB (SP),#', ;CHECK IF COMMA
7055 031206 001420 BEQ 3$ ;IF COMMA GET OUT
7056 031210 122716 000060 CMPB #'0,(SP) ;MAKE SURE THIS CHARACTER IS
7057 031214 003030 BGT 4$ ; AN OCTAL DIGIT
7058 031216 122716 000067 CMPB #'7,(SP)
7059 031222 002425 BLT 4$

```

```

7060 031224 006301      ASL      R1      ; *2
7061 031226 006102      ROL      R2
7062 031230 006301      ASL      R1      ; *4
7063 031232 006102      ROL      R2
7064 031234 006301      ASL      R1      ; *8
7065 031236 006102      ROL      R2
7066 031240 042716 177770 BIC      #1C7,(SP) ;STRIP THE ASCII JUNK
7067 031244 062601      ADD      (SP)+,R1 ;ADD THIS DIGIT
7068 031246 000753      BR       2$      ;LOOP
7069 031250 005726      3$:      TST      (SP)+ ;CLEAN PARTIAL FROM STACK
7070 031252 010166 000010 MOV      R1,10(SP) ;SAVE RESULT
7071 031256 010237 031312 MOV      R2,$HI OCT
7072 031262 012602      MOV      (SP)+,R2 ;RESTORE R2
7073 031264 012601      MOV      (SP)+,R1 ;RESTORE R1
7074 031266 012600      MOV      (SP)+,R0 ;RESTORE R0
7075 031270 062716 000002 ADD      #2,(SP) ;ADJUST RETURN
7076 031274 000207      RTS      PC      ;RETURN
7077
7078 031276 005726      4$:      TST      (SP)+ ;CLEAN UP PARTIAL FROM STACK
7079 031300 012602      MOV      (SP)+,R2 ;RESTORE R2
7080 031302 012601      MOV      (SP)+,R1 ;RESTORE R1
7081 031304 012600      MOV      (SP)+,R0 ;RESTORE R0
7082 031306 013616      MOV      2(SP)+,(SP) ;PUT ADDRESS OF ERROR ROUTINE ON STACK
7083 031310 000207      RTS      PC      ;GO PROCESS ERROR
7084 031312 000000      $HI OCT: .WORD 0 ;HIGH ORDER BITS GO HERE
7085
7086      .SBTTL DECIMAL TO BINARY CONVERSION ROUTINE
7087
7088      ;*****
7089      ; THIS ROUTINE WILL CHECK A STRING OF ASCII CHARACTERS TERMINATED
7090      ; WITH A NULL <000> OR COMMA. IF THE CHARACTERS ARE LEGAL,
7091      ; IT WILL GENERATE A BINARY WORD PLACING IT ON THE STACK.
7092      ;
7093      ;CALL
7094      ; MOV      <ADDRESS OF ASCII STRING>,-(SP)
7095      ; JSR      PC,DECBIN
7096      ; <ADDRESS OF ERROR RETURN>
7097      ; RETURN
7098      ;
7099      ;*****
7100
7101 031314 010046      DECBIN: MOV      R0,-(SP) ;SAVE R0
7102 031316 010146      MOV      R1,-(SP) ;SAVE R1
7103 031320 010246      MOV      R2,-(SP) ;SAVE R2
7104 031322 016600 000010 MOV      10(SP),R0 ;GET ADDRESS OF ASCII STRING
7105 031326 005046      CLR      -(SP) ;CLEAR DATA WORD
7106 031330 005002      CLR      R2 ;SIGN SET POSITIVE
7107 031332 122710 000055 CMPB     #'-(R0) ;SEE IF A MINUS SIGN
7108 031336 001001      BNE      2$ ;BRANCH IF NO MINUS SIGN
7109 031340 112002      MOVB     (R0)+,R2 ;SAVE FOR LATER USE
7110 031342 112001      2$:      MOVB     (R0)+,R1 ;PICKUP THIS CHARACTER
7111 031344 001427      BEQ      3$ ;GET OUT IF ZERO
7112 031346 120127 000054 CMPB     R1,#' ;CHECK IF COMMA
7113 031352 001424      BEQ      3$ ;GET OUT IF COMMA
7114 031354 122701 000060 CMPB     #'0,R1 ;MAKE SURE THIS CHARACTER IS
7115 031360 003034      BGT      5$ ; A DIGIT BETWEEN 0 & 9

```

```
7116 031362 122701 000071      CMPB    #'9,R1
7117 031366 002431      BLT     5$
7118 031370 032716 170000      BIT     #170000,(SP)      ;DON'T LET NUMBER GET TO BIG
7119 031374 001026      BNE     5$      ;BRANCH IF NUMBER WOULD OVERFLOW
7120 031376 006316      ASL     (SP)      ;#2
7121 031400 011646      MOV     (SP),-(SP)      ;SAVE FOR LATER
7122 031402 006316      ASL     (SP)      ;#4
7123 031404 006316      ASL     (SP)      ;#8
7124 031406 062616      ADD     (SP)+,(SP)      ;#10
7125 031410 102420      BVS     5$      ;OVERFLOW ISN'T ALLOWED
7126 031412 162701 000060      SUB     #'0,R1      ;STRIP AWAY THE ASCII JUNK
7127 031416 060116      ADD     R1,(SP)      ;ADD IN THIS DIGIT
7128 031420 102414      BVS     5$      ;OVERFLOW ISN'T ALLOWED
7129 031422 000747      BR      2$      ;LOOP
7130 031424 005702      3$: TST     R2      ;CHECK IF NUMBER IS NEGATIVE
7131 031426 001401      BEQ     4$      ;BRANCH IF NO
7132 031430 005416      NEG     (SP)      ;YES--NEGATE THE NUMBER
7133 031432 012666 000010      4$: MOV     (SP)+,10(SP)      ;SAVE RESULT
7134 031436 012602      MOV     (SP)+,R2      ;RESTORE R2
7135 031440 012601      MOV     (SP)+,R1      ;RESTORE R1
7136 031442 012600      MOV     (SP)+,R0      ;RESTORE R0
7137 031444 062716 000002      ADD     #2,(SP)      ;ADJUST RETURN
7138 031450 000207      RTS     PC      ;RETURN
7139
7140 031452 005726      5$: TST     (SP)+      ;CLEAN PARTIAL NUMBER FROM STACK
7141 031454 012602      MOV     (SP)+,R2      ;RESTORE R2
7142 031456 012601      MOV     (SP)+,R1      ;RESTORE R1
7143 031460 012600      MOV     (SP)+,R0      ;RESTORE R0
7144 031462 013616      MOV     2(SP)+,(SP)      ;PUT ADDRESS OF ERROR ON STACK
7145 031464 000207      RTS     PC      ;GO PROCESS ERROR
7146
7147      .SBTTL  TYPE ROUTINE
7148
7149      ;*****
7150      ;ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
7151      ;THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
7152      ;NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
7153      ;NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
7154      ;NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
7155      ;
7156      ;CALL:
7157      ;1) USING A TRAP INSTRUCTION
7158      ;* TYPE ,MESADR      ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
7159      ;*OR
7160      ;* TYPE
7161      ;* MESADR
7162      ;*
7163 031466 105737 001157      $TYPE: TSTB    $TPFLG      ;; IS THERE A TERMINAL?
7164 031472 100002      BPL     1$      ;; BR IF YES
7165 031474 000000      HALT     ;      ;HALT HERE IF NO TERMINAL
7166 031476 000407      BR      3$      ;LEAVE
7167 031500 010046      1$: MOV     R0,-(SP)      ;SAVE R0
7168 031502 017600 000002      MOV     22(SP),R0      ;GET ADDRESS OF ASCIZ STRING
7169 031506 112046      2$: MOVB    (R0)+,-(SP)      ;PUSH CHARACTER TO BE TYPED ONTO STACK
7170 031510 001005      BNE     4$      ;BR IF IT ISN'T THE TERMINATOR
7171 031512 005726      TST     (SP)+      ;IF TERMINATOR POP IT OFF THE STACK
```

## TYPE ROUTINE

```
7172 031514 012600 000002 60S: MOV (SP)+,RO ;:RESTORE RO
7173 031516 062716 000002 3S: ADD #2,(SP) ;:ADJUST RETURN PC
7174 031522 000002 RTI ;:RETURN
7175 031524 122716 000011 4S: CMPB #HT,(SP) ;:BRANCH IF <HT>
7176 031530 001430 BEQ #S ;:
7177 031532 122716 000200 CMPB #CRLF,(SP) ;:BRANCH IF NOT <CRLF>
7178 031536 001006 BNE #S ;:
7179 031540 005726 TST (SP)+ ;:POP <CR><LF> EQUIV
7180 031542 104401 TYPE ;:TYPE A CR AND LF
7181 031544 001171 SCRLF ;:
7182 031546 105037 031702 CLRB $CHARCNT ;:CLEAR CHARACTER COUNT
7183 031552 000755 BR #S ;:GET NEXT CHARACTER
7184 031554 004737 031636 5S: JSR PC,$TYPEC ;:GO TYPE THIS CHARACTER
7185 031560 123726 001156 6S: CMPB $FILLC,(SP)+ ;:IS IT TIME FOR FILLER CHARS.?
7186 031564 001350 BNE #S ;:IF NO GO GET NEXT CHAR.
7187 031566 013746 001154 MOV $NULL,-(SP) ;:GET # OF FILLER CHARS. NEEDED
7188 ;:AND THE NULL CHAR.
7189 031572 105366 000001 7S: DECB 1(SP) ;:DOES A NULL NEED TO BE TYPED?
7190 031576 002770 BLT #S ;:BR IF NO--GO POP THE NULL OFF OF STACK
7191 031600 004737 031636 JSR PC,$TYPEC ;:GO TYPE A NULL
7192 031604 105337 031702 DECB $CHARCNT ;:DO NOT COUNT AS A COUNT
7193 031610 000770 BR #S ;:LOOP
7194
7195 ;:HORIZONTAL TAB PROCESSOR
7196
7197 031612 112716 000040 8S: MOVB #' (SP) ;:REPLACE TAB WITH SPACE
7198 031616 004737 031636 9S: JSR PC,$TYPEC ;:TYPE A SPACE
7199 031622 132737 000007 031702 BITB #7,$CHARCNT ;:BRANCH IF NOT AT
7200 031630 001372 BNE #S ;:TAB STOP
7201 031632 005726 TST (SP)+ ;:POP SPACE OFF STACK
7202 031634 000724 BR #S ;:GET NEXT CHARACTER
7203 031636 105777 147306 $TYPEC: TSTB #STPS ;:WAIT UNTIL PRINTER IS READY
7204 031642 100375 BPL $TYPEC ;:
7205 031644 116677 000002 147300 MOVB 2(SP),#STPB ;:LOAD CHAR TO BE TYPED INTO DATA REG.
7206 031652 122766 000015 000002 CMPB #CR,2(SP) ;:IS CHARACTER A CARRIAGE RETURN?
7207 031660 001003 BNE #S ;:BRANCH IF NO
7208 031662 105037 031702 CLRB $CHARCNT ;:YES--CLEAR CHARACTER COUNT
7209 031666 000406 BR $TYPEX ;:EXIT
7210 031670 122766 000012 000002 1S: CMPB #LF,2(SP) ;:IS CHARACTER A LINE FEED?
7211 031676 001402 BEQ $TYPEX ;:BRANCH IF YES
7212 031700 105227 INCB (PC)+ ;:COUNT THE CHARACTER
7213 031702 000000 $CHARCNT: WORD 0 ;:CHARACTER COUNT STORAGE
7214 031704 000207 $TYPEX: RTS PC
7215
7216 .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
7217
7218 ;:*****
7219 ;:THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
7220 ;:SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
7221 ;:NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
7222 ;:BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
7223 ;:REPLACED WITH SPACES.
7224 ;:CALL:
7225 ;: MOV NUM,-(SP) ;:PUT THE BINARY NUMBER ON THE STACK
7226 ;: TYPOS ;:GO TO THE ROUTINE
7227
```



```
7228 031706          STYPDS:
7229 031706 010046      MOV    R0,-(SP)      ; PUSH R0 ON STACK
7230 031710 010146      MOV    R1,-(SP)      ; PUSH R1 ON STACK
7231 031712 010246      MOV    R2,-(SP)      ; PUSH R2 ON STACK
7232 031714 010346      MOV    R3,-(SP)      ; PUSH R3 ON STACK
7233 031716 010546      MOV    R5,-(SP)      ; PUSH R5 ON STACK
7234 031720 012746 020200 MOV    #20200,-(SP) ; SET BLANK SWITCH AND SIGN
7235 031724 016605 000020 MOV    20(SP),R5 ; GET THE INPUT NUMBER
7236 031730 100004      BPL     1$          ; BR IF INPUT IS POS.
7237 031732 005405      NEG     R5          ; MAKE THE BINARY NUMBER POS.
7238 031734 112766 000055 000001 1$: MOV    #'-,1(SP) ; MAKE THE ASCII NUMBER NEG.
7239 031742 005000      CLR     R0          ; ZERO THE CONSTANTS INDEX
7240 031744 012703 032122      MOV    #SDBLK,R3 ; SETUP THE OUTPUT POINTER
7241 031750 112723 000040      MOV    #'',(R3)+ ; SET THE FIRST CHARACTER TO A BLANK
7242 031754 005002      CLR     R2          ; CLEAR THE BCD NUMBER
7243 031756 016001 032112      MOV    $DTBL(R0),R1 ; GET THE CONSTANT
7244 031762 160105      3$: SUB     R1,R5 ; FORM THIS BCD DIGIT
7245 031764 002402      BLT     4$          ; BR IF DONE
7246 031766 005202      INC     R2          ; INCREASE THE BCD DIGIT BY 1
7247 031770 000774      BR      3$
7248 031772 060105      4$: ADD     R1,R5 ; ADD BACK THE CONSTANT
7249 031774 005702      TST     R2          ; CHECK IF BCD DIGIT=0
7250 031776 001002      BNE     5$          ; FALL THROUGH IF 0
7251 032000 105716      TSTB    (SP)        ; STILL DOING LEADING 0'S?
7252 032002 100407      BMI     7$          ; BR IF YES
7253 032004 106316      5$: ASLB     (SP)        ; MSD?
7254 032006 103003      BCC     6$          ; BR IF NO
7255 032010 116663 000001 177777 6$: MOV    1(SP),-1(R3) ; YES--SET THE SIGN
7256 032016 052702 000060      BIS    #'0,R2 ; MAKE THE BCD DIGIT ASCII
7257 032022 052702 000040 7$: BIS    #' ,R2 ; MAKE IT A SPACE IF NOT ALREADY A DIGIT
7258 032026 110223      MOV    R2,(R3)+ ; PUT THIS CHARACTER IN THE OUTPUT BUFFER
7259 032030 005720      TST     (R0)+ ; JUST INCREMENTING
7260 032032 020027 000010      CMP    R0,#10 ; CHECK THE TABLE INDEX
7261 032036 002746      BLT     2$          ; GO DO THE NEXT DIGIT
7262 032040 003002      BGT     8$          ; GO TO EXIT
7263 032042 010502      MOV     R5,R2
7264 032044 000764      BR      6$          ; GET THE LSD
7265 032046 105726      8$: TSTB    (SP)+ ; GO CHANGE TO ASCII
7266 032050 100003      BPL     9$          ; WAS THE LSD THE FIRST NON-ZERO?
7267 032052 116663 177777 177776 9$: MOV    -1(SP),-2(R3) ; BR IF NO
7268 032060 105013      CLRB    (R3)        ; YES--SET THE SIGN FOR TYPING
7269 032062 012605      CLR     (SP)+,R5 ; SET THE TERMINATOR
7270 032064 012603      MOV     (SP)+,R3 ; POP STACK INTO R5
7271 032066 012602      MOV     (SP)+,R2 ; POP STACK INTO R3
7272 032070 012601      MOV     (SP)+,R1 ; POP STACK INTO R2
7273 032072 012600      MOV     (SP)+,R0 ; POP STACK INTO R1
7274 032074 104401 032122      TYPE   $SDBLK ; POP STACK INTO R0
7275 032100 016666 000002 000004      MOV    2(SP),4(SP) ; NOW TYPE THE NUMBER
7276 032106 012616      MOV     (SP)+,(SP) ; ADJUST THE STACK
7277 032110 000002      RTI                ; RETURN TO USER
7278 032112 023420      $DTBL: 10000.
7279 032114 001750      1000.
7280 032116 000144      100.
7281 032120 000012      10.
7282 032122 000004      $SDBLK: .BLKW 4
7283                                .SBTTL BINARY TO OCTAL (ASCII) AND TYPE
```

```
7284
7285
7286
7287
7288
7289
7290
7291
7292
7293
7294
7295
7296
7297
7298
7299
7300
7301
7302
7303
7304
7305
7306
7307
7308 032132 017646 000000
7309 032136 116637 000001 032355
7310 032144 112637 032357
7311 032150 062716 000002
7312 032154 000406
7313 032156 112737 000001 032355
7314 032164 112737 000006 032357
7315 032172 112737 000005 032354
7316 032200 010346
7317 032202 010446
7318 032204 010546
7319 032206 113704 032357
7320 032212 005404
7321 032214 062704 000006
7322 032220 110437 032356
7323 032224 113704 032355
7324 032230 016605 000012
7325 032234 005003
7326 032236 006105
7327 032240 000404
7328 032242 006105
7329 032244 006105
7330 032246 006105
7331 032250 010503
7332 032252 006103
7333 032254 105337 032356
7334 032260 100016
7335 032262 042703 177770
7336 032266 001002
7337 032270 005704
7338 032272 001403
7339 032274 005204

*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
*OCTAL (ASCII) NUMBER AND TYPE IT.
*STYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
*CALL:
*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*      TYPPOS      ;;CALL FOR TYPEOUT
*      .BYTE      N      ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
*      .BYTE      M      ;;M=1 OR 0
*                               ;;1=TYPE LEADING ZEROS
*                               ;;0=SUPPRESS LEADING ZEROS
*STYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
*STYPOS OR STYPOC
*CALL:
*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*      TYPON      ;;CALL FOR TYPEOUT
*STYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
*CALL:
*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*      TYPOC      ;;CALL FOR TYPEOUT
*STYPOS: MOV      2(SP),-(SP)      ;;PICKUP THE MODE
*      MOV      1(SP),SOFILL      ;;LOAD ZERO FILL SWITCH
*      MOV      (SP)+,SOMODE+1      ;;NUMBER OF DIGITS TO TYPE
*      ADD      #2,(SP)      ;;ADJUST RETURN ADDRESS
*      BR      STYPON
*STYPOC: MOV      #1,SOFILL      ;;SET THE ZERO FILL SWITCH
*      MOV      #6,SOMODE+1      ;;SET FOR SIX(6) DIGITS
*STYPON: MOV      #5,SOCNT      ;;SET THE ITERATION COUNT
*      MOV      R3,-(SP)      ;;SAVE R3
*      MOV      R4,-(SP)      ;;SAVE R4
*      MOV      R5,-(SP)      ;;SAVE R5
*      MOV      SOMODE+1,R4      ;;GET THE NUMBER OF DIGITS TO TYPE
*      NEG      R4
*      ADD      #6,R4      ;;SUBTRACT IT FOR MAX. ALLOWED
*      MOV      R4,SOMODE      ;;SAVE IT FOR USE
*      MOV      SOFILL,R4      ;;GET THE ZERO FILL SWITCH
*      MOV      12(SP),R5      ;;PICKUP THE INPUT NUMBER
*      CLR      R3      ;;CLEAR THE OUTPUT WORD
*      ROL      R5      ;;ROTATE MSB INTO "C"
*      BR      3$      ;;GO DO MSB
*      ROL      R5      ;;FORM THIS DIGIT
*      ROL      R5
*      ROL      R5
*      MOV      R5,R3
*      ROL      R3      ;;GET LSB OF THIS DIGIT
*      DEC      SOMODE      ;;TYPE THIS DIGIT?
*      BPL      7$      ;;BR IF NO
*      BIC      #177770,R3      ;;GET RID OF JUNK
*      BNE      4$      ;;TEST FOR 0
*      TST      R4      ;;SUPPRESS THIS 0?
*      BEQ      5$      ;;BR IF YES
*      INC      R4      ;;DON'T SUPPRESS ANYMORE 0'S
*      1$: ROL      R5
*      2$: ROL      R5
*      3$: ROL      R3
*      4$: BNE      4$
*      5$: BEQ      5$
*      6$: INC      R4
```

```
7340 032276 052703 000060
7341 032302 052703 000040
7342 032306 110337 032352
7343 032312 104401 032352
7344 032316 105337 032354
7345 032322 003347
7346 032324 002402
7347 032326 005204
7348 032330 000744
7349 032332 012605
7350 032334 012604
7351 032336 012603
7352 032340 016666 000002 000004
7353 032346 012616
7354 032350 000002
7355 032352 000
7356 032353 000
7357 032354 000
7358 032355 000
7359 032356 000000
7360
7361
7362
7363
7364 032360 000000
7365 032362 000000
7366 032364 000000
7367 032366 000001
7368 032367
7369 032370
7370
7371
7372
7373
7374
7375
7376
7377
7378
7379 032370 005037 032360
7380 032374 012737 032366 032362
7381 032402 013737 032362 032364
7382 032410 012737 032440 000060
7383 032416 012737 000200 000062
7384 032424 005777 146516
7385 032430 012777 000100 146506
7386 032436 000207
7387
7388
7389
7390
7391
7392
7393
7394
7395 032440 117746 146502
```

```
55: BIS #D,R3 ;;MAKE THIS DIGIT ASCII
    BIS #' ,R3 ;;MAKE ASCII IF NOT ALREADY
    MOVB R3,8$ ;;SAVE FOR TYPING
    TYPE 8$ ;;GO TYPE THIS DIGIT
7$: DECB $OCNT ;;COUNT BY 1
    BGT 2$ ;;BR IF MORE TO DO
    BLT 6$ ;;BR IF DONE
    INC R4 ;;INSURE LAST DIGIT ISN'T A BLANK
    BR 2$ ;;GO DO THE LAST DIGIT
6$: MOV (SP)+,R5 ;;RESTORE R5
    MOV (SP)+,R4 ;;RESTORE R4
    MOV (SP)+,R3 ;;RESTORE R3
    MOV 2(SP),4(SP) ;;SET THE STACK FOR RETURNING
    MOV (SP)+,(SP)
    RTI ;;RETURN
8$: .BYTE 0 ;;STORAGE FOR ASCII DIGIT
    .BYTE 0 ;;TERMINATOR FOR TYPE ROUTINE
$OCNT: .BYTE 0 ;;OCTAL DIGIT COUNTER
$OFILL: .BYTE 0 ;;ZERO FILL SWITCH
$OMODE: .WORD 0 ;;NUMBER OF DIGITS TO TYPE
.SBTTL TTY INPUT ROUTINE

;*****
.ENABL LSB
$TKCNT: .WORD 0 ;;NUMBER OF ITEMS IN QUEUE
$TKQIN: .WORD 0 ;;INPUT POINTER
$TKQOUT: .WORD 0 ;;OUTPUT POINTER
$TKQSRT: .BLKB 1 ;;TTY KEYBOARD QUEUE
$TKQEND=.
.EVEN

;*TK INITIALIZE ROUTINE
;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
;*CALL:
;* JSR PC,$TKINT
;* RETURN
$TKINT: CLR $TKCNT ;;CLEAR COUNT OF ITEMS IN QUEUE
    MOV $TKQSRT,$TKQIN ;;MOVE THE STARTING ADDRESS OF THE
    MOV $TKQIN,$TKQOUT ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
    MOV $TKSRV,$TKVEC ;;INITIALIZE THE KEYBOARD VECTOR
    MOV #200,$TKVEC+2 ;;"BR" LEVEL 4
    TST $TKB ;;CLEAR DONE FLAG
    MOV #100,$TKS ;;ENABLE TTY KEYBOARD INTERRUPT
    RTS PC ;;RETURN TO CALLER

;*TK SERVICE ROUTINE
;*THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
;*BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
;*IT IN THE QUEUE.
;*IF THE CHARACTER IS A "CONTROL-C" (1C) $TKINT IS CALLED AND
;*UPON RETURN EXIT IS MADE TO THE "CONTROL-C" RESTART ADDRESS (UDTSRT)
$TKSRV: MOVB $TKB,-(SP) ;;PICKUP THE CHARACTER
```

```
7396 032444 042716 177600      BIC      #C177,(SP)      ;STRIP THE JUNK
7397 032450 021627 000003      CMP      (SP),#3      ;IS IT A CONTROL C?
7398 032454 001007              BNE      1$      ;BRANCH IF NO
7399 032456 104401 033636      TYPE     $CNTLC      ;TYPE A CONTROL-C (↑C)
7400 032462 004737 032370      JSR      PC,STKINT    ;INIT THE KEYBOARD
7401 032466 005726              TST      (SP)+      ;CLEAN UP STACK
7402 032470 000137 004116      JMP      UDTSRT      ;CONTROL C RESTART
7403 032474 021627 000007      1$:      CMP      (SP),#7      ;IS IT A CONTROL G?
7404 032500 001004              BNE      2$      ;BRANCH IF NO
7405 032502 022737 000176 001140  CMP      $SWREG,SWR      ;IS SOFT-SWR SELECTED?
7406 032510 001500              BEQ      6$      ;GO TO SWR CHANGE
7407
7408 032512              2$:
7409 032512 022737 000001 032360  CMP      #1,STKCNT      ;IS THE QUEUE FULL?
7410 032520 001004              BNE      3$      ;BRANCH IF NO
7411 032522 104401 001164      TYPE     $BELL      ;RING THE TTY BELL
7412 032526 005726              TST      (SP)+      ;CLEAN CHARACTER OFF OF STACK
7413 032530 000451              BR       5$      ;EXIT
7414 032532 021627 000023      3$:      CMP      (SP),#23      ;IS IT A CONTROL-S?
7415 032536 001021              BNE      32$      ;BRANCH IF NO
7416 032540 005077 146400      CLR      $STKS      ;DISABLE TTY KEYBOARD INTERRUPTS
7417 032544 005726              TST      (SP)+      ;CLEAN CHAR OFF STACK
7418 032546 105777 146372      31$:    TSTB     $STKS      ;WAIT FOR A CHAR
7419 032552 100375              BPL      31$      ;LOOP UNTIL ITS THERE
7420 032554 117746 146366      MOVB     $STKB,-(SP)    ;GET THE CHARACTER
7421 032560 042716 177600      BIC      #C177,(SP)    ;MAKE IT 7-BIT ASCII
7422 032564 022627 000021      CMP      (SP)+,#21      ;IS IT A CONTROL-Q?
7423 032570 001366              BNE      31$      ;BRANCH IF NO
7424 032572 012777 000100 146344  MOV      #100,$STKS      ;REENABLE TTY KEYBOARD INTERRUPTS
7425 032600 000002              RTI      ;RETURN
7426 032602 005237 032360      32$:    INC      STKCNT      ;COUNT THIS CHARACTER
7427 032606 021627 000140      CMP      (SP),#140      ;IS IT UPPER CASE?
7428 032612 002405              BLT      4$      ;BRANCH IF YES
7429 032614 021627 000175      CMP      (SP),#175      ;IS IT A SPECIAL CHAR?
7430 032620 003002              BGT      4$      ;BRANCH IF YES
7431 032622 042716 000040      BIC      #40,(SP)      ;MAKE IT UPPER CASE
7432 032626 112677 177530      4$:      MOVB     (SP)+,$STKQIN  ;AND PUT IT IN QUEUE
7433 032632 005237 032362      INC      STKQIN      ;UPDATE THE POINTER
7434 032636 023727 032362 032367  CMP      STKQIN,$STKQEND  ;GO OFF THE END?
7435 032644 001003              BNE      5$      ;BRANCH IF NO
7436 032646 012737 032366 032362  MOV      $STKQSRST,$STKQIN ;RESET THE POINTER
7437 032654 000002      5$:      RTI      ;RETURN
7438
7439 ;*****
7440 ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
7441 ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
7442 ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP
7443 ;*CALL WHEN OPERATING IN TTY INTERRUPT MODE.
7444 032656 022737 000176 001140 $CKSWR: CMP      $SWREG,SWR      ;IS THE SOFT-SWR SELECTED
7445 032664 001124              BNE      15$      ;EXIT IF NOT
7446 032666 105777 146252      TSTB     $STKS      ;IS A CHAR WAITING?
7447 032672 100121              BPL      15$      ;IF NOT, EXIT
7448 032674 117746 146246      MOVB     $STKB,-(SP)    ;YES
7449 032700 042716 177600      BIC      #C177,(SP)    ;MAKE IT 7-BIT ASCII
7450 032704 021627 000007      CMP      (SP),#7      ;IS IT A CONTROL-G?
7451 032710 001300              BNE      2$      ;IF NOT, PUT IT IN THE TTY QUEUE
```

```
7452                                     ;;AND EXIT
7453
7454                                     ;*****
7455                                     ;*CONTROL IS PASSED TO THIS POINT FROM EITHER THE TTY INTERRUPT SERVICE
7456                                     ;*ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A
7457                                     ;*CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.
7458 032712 123727 001134 000001 6$: CMPB $AUTOB,#1 ;;ARE WE RUNNING IN AUTO-MODE?
7459 032720 001674 BEQ 2$ ;;BRANCH IF YES
7460 032722 005726 TST (SP)+ ;;CLEAR CONTROL-G OFF STACK
7461 032724 004737 032370 JSR PC,$TKINT ;;FLUSH THE TTY INPUT QUEUE
7462 032730 005077 146210 CLR 2$TKS ;;DISABLE TTY KEYBOARD INTERRUPTS
7463 032734 112737 000001 001135 MOVB #1,$INTAG ;;SET INTERRUPT MODE INDICATOR
7464
7465 032742 104401 033650 SGTSWR: TYPE , $CNTLG ;;ECHO THE CONTROL-G (↑G)
7466 032746 104401 033655 TYPE , $MSWR ;;TYPE CURRENT CONTENTS
7467 032752 013746 000176 MOV SWREG,-(SP) ;;SAVE SWREG FOR TYPEOUT
7468 032756 104402 TPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
7469 032760 104401 033666 TYPE , $MNEW ;;PROMPT FOR NEW SWR
7470 032764 005046 19$: CLR -(SP) ;;CLEAR COUNTER
7471 032766 005046 CLR -(SP) ;;THE NEW SWR
7472 032770 105777 146150 7$: TSTB 2$TKS ;;CHAR THERE?
7473 032774 100375 BPL 7$ ;;IF NOT TRY AGAIN
7474
7475 032776 117746 146144 MOVB 2$TKB,-(SP) ;;PICK UP CHAR
7476 033002 042716 177600 BIC #1C177,(SP) ;;MAKE IT 7-BIT ASCII
7477
7478 033006 021627 000003 CMP (SP),#3 ;;IS IT A CONTROL-C?
7479 033012 001015 BNE 9$ ;;BRANCH IF NOT
7480 033014 104401 033636 TYPE , $CNTLC ;;YES, ECHO CONTROL-C (↑C)
7481 033020 062706 000006 ADD #6,SP ;;CLEAR UP STACK
7482 033024 123727 001135 000001 CMPB $INTAG,#1 ;;RE-ENABLE TTY KEYBOARD INTERRUPTS?
7483 033032 001003 BNE 8$ ;;BRANCH IF NO
7484 033034 012777 000100 146102 MOV #100,2$TKS ;;ALLOW TTY KEYBOARD INTERRUPTS
7485 033042 000137 004116 8$: JMP UDTSTRT ;;CONTROL-C RESTART
7486
7487
7488 033046 021627 000025 9$: CMP (SP),#25 ;;IS IT A CONTROL-U?
7489 033052 001005 BNE 10$ ;;BRANCH IF NOT
7490 033054 104401 033643 TYPE , $CNTLU ;;YES, ECHO CONTROL-U (↑U)
7491 033060 062706 000006 20$: ADD #6,SP ;;IGNORE PREVIOUS INPUT
7492 033064 000737 BR 19$ ;;LET'S TRY IT AGAIN
7493
7494
7495 033066 021627 000015 10$: CMP (SP),#15 ;;IS IT A <CR>?
7496 033072 001022 BNE 16$ ;;BRANCH IF NO
7497 033074 001766 000004 TST 4(SP) ;;YES, IS IT THE FIRST CHAR?
7498 033100 001403 BEQ 11$ ;;BRANCH IF YES
7499 033102 016677 000002 146030 MOV 2(SP),2$SWR ;;SAVE NEW SWR
7500 033110 062706 000006 11$: ADD #6,SP ;;CLEAR UP STACK
7501 033114 104401 001171 14$: TYPE , $CRLF ;;ECHO <CR> AND <LF>
7502 033120 123727 001135 000001 CMPB $INTAG,#1 ;;RE-ENABLE TTY KBD INTERRUPTS?
7503 033126 001003 BNE 15$ ;;BRANCH IF NOT
7504 033130 012777 000100 146006 MOV #100,2$TKS ;;RE-ENABLE TTY KBD INTERRUPTS
7505 033136 000002 15$: RTI ;;RETURN
7506 033140 004737 031636 16$: JSR PC,$TYPEC ;;ECHO CHAR
7507 033144 021627 000060 CMP (SP),#60 ;;CHAR < 0?
```

```

7508 033150 002420          BLT      18$          ;; BRANCH IF YES
7509 033152 021627 000067    CMP      (SP),#67    ;; CHAR > 7?
7510 033156 003015          BGT      18$          ;; BRANCH IF YES
7511 033160 042726 000060    BIC      #60,(SP)+    ;; STRIP-OFF ASCII
7512 033164 005766 000002    TST      2(SP)       ;; IS THIS THE FIRST CHAR
7513 033170 001403          BEQ      17$          ;; BRANCH IF YES
7514 033172 006316          ASL      (SP)         ;; NO, SHIFT PRESENT
7515 033174 006316          ASL      (SP)         ;; CHAR OVER TO MAKE
7516 033176 006316          ASL      (SP)         ;; ROOM FOR NEW ONE.
7517 033200 005266 000002    17$: INC      2(SP)     ;; KEEP COUNT OF CHAR
7518 033204 056616 177776    BIS      -2(SP),(SP)  ;; SET IN NEW CHAR
7519 033210 000667          BR       7$          ;; GET THE NEXT ONE
7520 033212 104401 001170    18$: TYPE    $QUES    ;; TYPE ?(CR)(LF)
7521 033216 000720          BR       20$          ;; SIMULATE CONTROL-U
7522          .DSABL    LSB
7523
7524
7525          ;; *****
7526          ;; THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
7527          ;; *CALL:
7528          ;;      R0CHR          ;; GET A CHARACTER FROM THE QUEUE
7529          ;;      RETURN HERE    ;; CHARACTER IS ON THE STACK
7530          ;;
7531          ;;
7532
7533 033220 011646 000004 000002 $R0CHR: MOV      (SP),-(SP)    ;; PUSH DOWN THE PC AND
7534 033222 016666 000004          MOV      4(SP),2(SP)    ;; THE PS
7535 033230 005 66 000004          CLR      4(SP)          ;; GET READY FOR A CHARACTER
7536 033234 005J46          CLR      -(SP)          ;; PUT NEW PS ON STACK
7537 033236 012746 033244          MOV      #E4$,-(SP)     ;; PUT NEW PC ON STACK
7538 033242 000002          RTI          ;; POP NEW PC AND PS
7539 033244
7540 033244 005737 032360    64$: TST      $TKCNT          ;; WAIT ON A CHARACTER
7541 033250 001775          1$: BEQ      1$
7542 033252 005337 032360          DEC      $TKCNT          ;; DECREMENT THE COUNTER
7543 033256 117766 177102 000004  MOVB     $TKQOUT,4(SP)  ;; GET ONE CHARACTER
7544 033254 005237 032364          INC      $TKQOUT        ;; UPDATE THE POINTER
7545 033270 023727 032364 032367  CMP      $TKQOUT,$$TKQEND ;; DID IT GO OFF OF THE END?
7546 033276 001003          BNE      2$          ;; BRANCH IF NO
7547 033300 012737 032366 032364  MOV      $$TKQSRT,$TKQOUT ;; RESET THE POINTER
7548 033306 000002          RTI          ;; RETURN
7549
7550          ;; *****
7551          ;; THIS ROUTINE WILL INPUT A STRING FROM THE TTY
7552          ;; *CALL:
7553          ;;      RDLIN          ;; INPUT A STRING FROM THE TTY
7554          ;;      RETURN HERE    ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
7555          ;;
7556          ;;
7557          ;;
7558          ;;
7559          ;;
7560          ;;
7561          ;;
7562          ;;
7563          ;;

```

```

7564 033336 001022      BNE      5$      ;:BR IF NO
7565 033340 005716      TST      (SP)      ;:IS THIS THE FIRST RUBOUT?
7566 033342 001007      BNE      6$      ;:BR IF NO
7567 033344 112737 000134 033542      MOV     #' \, 9$      ;:TYPE A BACK SLASH
7568 033352 104401 033542      TYPE     9$      ;:
7569 033356 012716 177777      MOV     4-1, (SP)      ;:SET THE RUBOUT KEY
7570 033362 005303      6$: DEC     R3      ;:BACKUP BY ONE
7571 033364 020327 033544      CMP     R3, #STTYIN      ;:STACK EMPTY?
7572 033370 103434      BLO     4$      ;:BR IF YES
7573 033372 111337 033542      MOV     (R3), 9$      ;:SETUP TO TYPEOUT THE DELETED CHAR.
7574 033376 104401 033542      TYPE     9$      ;:GO T'PE
7575 033402 000746      BR      2$      ;:GO READ ANOTHER CHAR.
7576 033404 005716      5$: TST      (SP)      ;:RUBOUT KEY SET?
7577 033406 001406      BEQ     7$      ;:BR IF NO
7578 033410 112737 000134 033542      MOV     #' \, 9$      ;:TYPE A BACK SLASH
7579 033416 104401 033542      TYPE     9$      ;:
7580 033422 005016      CLR     (SP)      ;:CLEAR THE RUBOUT KEY
7581 033424 122713 000025      7$: CMPB   #25, (R3)      ;:IS CHARACTER A CTRL U?
7582 033430 001003      BNE      8$      ;:BR IF NO
7583 033432 104401 033643      TYPE     $CNTLU      ;:TYPE A CONTROL "U"
7584 033436 000726      BR      1$      ;:GO START OVER
7585 033440 122713 000022      8$: CMPB   #22, (R3)      ;:IS CHARACTER A "↑R"?
7586 033444 001011      BNE      3$      ;:BRANCH IF NO
7587 033446 105013      CLRB    (R3)      ;:CLEAR THE CHARACTER
7588 033450 104401 001171      TYPE     $CRLF      ;:TYPE A "CR" & "LF"
7589 033454 104401 033544      TYPE     $TTYIN      ;:TYPE THE INPUT STRING
7590 033460 000717      BR      2$      ;:GO PICKUP ANOTHER CHACTER
7591 033462 104401 001170      4$: TYPE     $QUES      ;:TYPE A '?'
7592 033466 000712      BR      1$      ;:CLEAR THE BUFFER AND LOOP
7593 033470 111337 033542      3$: MOV     (R3), 9$      ;:ECHO THE CHARACTER
7594 033474 104401 033542      TYPE     9$      ;:
7595 033500 122723 000015      CMPB   #15, (R3)+      ;:CHECK FOR RETURN
7596 033504 001305      BNE      2$      ;:LOOP IF NOT RETURN
7597 033506 105063 177777      CLRB    -1(R3)      ;:CLEAR RETURN (THE 15)
7598 033512 104401 001172      TYPE     $LF      ;:TYPE A LINE FEED
7599 033516 005726      TST      (SP)+      ;:CLEAN RUBOUT KEY FROM THE STACK
7600 033520 012603      MOV     (SP)+, R3      ;:RESTORE R3
7601 033522 011646      MOV     (SP), -(SP)      ;:ADJUST THE STACK AND PUT ADDRESS OF THE
7602 033524 016666 000004 000002      MOV     4(SP), 2(SP)      ;:FIRST ASCII CHARACTER ON IT
7603 033532 012766 033544 000004      MOV     #STTYIN, 4(SP)      ;:
7604 033540 000002      RTI      ;:RETURN
7605 033542 000      9$: .BYTE    0      ;:STORAGE FOR ASCII CHAR. TO TYPE
7606 033543 000      .BYTE    0      ;:TERMINATOR
7607 033544 000072      STTYIN: .BLKB   72      ;:RESERVE 72 BYTES FOR TTY INPUT
7608 033636 041536 005015 000      $CNTLC: .ASCIZ  /↑C/<15><12>      ;:CONTROL "C"
7609 033643 136 006525 000012      $CNTLU: .ASCIZ  /↑U/<15><12>      ;:CONTROL "U"
7610 033650 043536 005015 000      $CNTLG: .ASCIZ  /↑G/<15><12>      ;:CONTROL "G"
7611 033655 015 051412 051127      $MSWR: .ASCIZ  <15><12>/SWR = /
7612 033662 036440 000040      $MNEW: .ASCIZ  / NEW = /
7613 033666 020040 042516 020127      .EVEN
7614 033674 020075 000      .SBTTL  DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE
7615 033700
7616
7617
7618
7619

```

\*\*\*\*\*  
; THIS ROUTINE WILL CONVERT A 32-BIT BINARY NUMBER TO AN UNSIGNED

```

7620 ;*DECIMAL (ASCII) NUMBER. THE SIGN OF THE BINARY NUMBER MUST BE
7621 ;*POSITIVE.
7622 ;*CALL
7623 ;*      MOV      #PNTR, -(SP)      ;; POINTER TO LOW WORD OF BINARY NUMBER
7624 ;*      JSR      PC, @#SDB2D      ;; THE FIRST ADDRESS OF ASCII
7625 ;*      RETURN                      ;; IS ON THE STACK
7626
7627
7628
7629 033700 104412      SDB2D: SAVREG      ;; SAVE REGISTERS
7630 033702 016602 000002 MOV      2(SP), R2      ;; PICKUP THE DATA POINTER
7631 033706 012700 034060 MOV      #SDECVL, R0      ;; GET ADDRESS OF "SDECVL" STRING
7632 033712 010066 000002 MOV      R0, 2(SP)      ;; PUT ADDRESS OF ASCII STRING ON STACK
7633 033716 012201      MOV      (R2)+, R1      ;; PICKUP THE BINARY NUMBER
7634 033720 012202      MOV      (R2)+, R2
7635 033722 012737 000012 MOV      #10, R4      ;; SET UP TO DO 10 CONVERSIONS
7636 033730 012704 034010 MOV      #STNPWR, R4      ;; ADDRESS OF TEN POWER
7637 033734 012705 034012 MOV      #STNPWR+2, R5
7638 033740 005003      1$: CLR      R3      ;; CLEAR PARTIAL
7639 033742 161401      2$: JBC      (R4), R1      ;; SUBTRACT TEN POWER
7640 033744 005602      JBC      R2
7641 033746 161502      SUB      (R5), R2
7642 033750 002402      BLT      3$      ;; BR IF TEN POWER TOO LARGE
7643 033752 005203      INC      R3      ;; ADD 1 TO PARTIAL
7644 033754 000772      BR      2$      ;; LOOP
7645 033756 062401      3$: ADD      (R4)+, R1      ;; RESTORE SUBTRACTED VALUE
7646 033760 005502      ADC      R2
7647 033762 062402      ADD      (R4)+, R2
7648 033764 022525      CMP      (R5)+, (R5)+      ;; MOVE TO NEXT TEN POWER
7649 033766 052703 000060 BIS      #0, R3      ;; CHANGE PARTIAL TO ASCII
7650 033772 110320      MOV      R3, (R0)+      ;; SAVE IT
7651 033774 005327      DEC      (PC)+      ;; DONE?
7652 033776 000000      4$: .WORD      0
7653 034000 001357      BNE      1$      ;; BR IF NO
7654 034002 105020      CLRB      (R0)+      ;; TERMINATOR
7655 034004 104413      RESREG      ;; RESTORE REGISTERS
7656 034006 000207      RTS      PC      ;; RETURN
7657 034010 145000      STNPWR: 145000      ;; 1.0E09
7658 034012 035632      35632
7659 034014 160400      160400      ;; 1.0E08
7660 034016 002765      2765
7661 034020 113200      113200      ;; 1.0E07
7662 034022 000230      230
7663 034024 041100      041100      ;; 1.0E06
7664 034026 000017      17
7665 034030 103240      103240      ;; 1.0E05
7666 034032 000001      1
7667 034034 023420      23420      ;; 1.0E04
7668 034036 000000      0
7669 034040 001750      1750      ;; 1.0E03
7670 034042 000000      0
7671 034044 000144      144      ;; 1.0E02
7672 034046 000000      0
7673 034050 000012      12      ;; 1.0E01
7674 034052 000000      0
7675 034054 000001      1      ;; 1.0E00

```



CZR6RCO RK611/06 USR DEFINED  
CZR6RC.P11 02-DEC-77 11:07

MACY11 30(1046) 02-DEC-77 11:59 PAGE 147  
DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE

SEQ 0146

7676 034056 000000  
7677 034060 000014  
7678  
7679  
7680  
7681  
7682  
7683  
7684  
7685  
7686  
7687  
7688  
7689  
7690  
7691  
7692  
7693  
7694  
7695 034074  
7696 034074 010046  
7697 034076 010146  
7698 034100 010246  
7699 034102 010346  
7700 034104 010446  
7701 034106 010546  
7702 034110 016646 000022  
7703 034114 016646 000022  
7704 034120 016646 000022  
7705 034124 016646 000022  
7706 034130 000002  
7707  
7708  
7709  
7710  
7711 034132  
7712 034132 012666 000022  
7713 034136 012666 000022  
7714 034142 012666 000022  
7715 034146 012666 000022  
7716 034152 012605  
7717 034154 012604  
7718 034156 012603  
7719 034160 012602  
7720 034162 012601  
7721 034164 012600  
7722 034166 000002  
7723  
7724  
7725  
7726  
7727  
7728  
7729  
7730  
7731

0  
\$DECVL: .BLKB 12. ;RESERVE STORAGE FOR ASCII STRING  
.SBTTL SAVE AND RESTORE RO-R5 ROUTINES

\*\*\*\*\*  
;SAVE RO-R5  
;CALL:  
; SAVREG  
;UPON RETURN FROM \$SAVREG THE STACK WILL LOOK LIKE:  
;TOP---(+16)  
; +2---(+18)  
; +4---R5  
; +6---R4  
; +8---R3  
; +10---R2  
; +12---R1  
; +14---R0

\$SAVREG:  
MOV RO, -(SP) ;PUSH RO ON STACK  
MOV R1, -(SP) ;PUSH R1 ON STACK  
MOV R2, -(SP) ;PUSH R2 ON STACK  
MOV R3, -(SP) ;PUSH R3 ON STACK  
MOV R4, -(SP) ;PUSH R4 ON STACK  
MOV R5, -(SP) ;PUSH R5 ON STACK  
MOV 22(SP), -(SP) ;SAVE PS OF MAIN FLOW  
MOV 22(SP), -(SP) ;SAVE PC OF MAIN FLOW  
MOV 22(SP), -(SP) ;SAVE PS OF CALL  
MOV 22(SP), -(SP) ;SAVE PC OF CALL  
RTI

;RESTORE RO-R5

;CALL:

; RESREG

\$RESREG:  
MOV (SP)+, 22(SP) ;RESTORE PC OF CALL  
MOV (SP)+, 22(SP) ;RESTORE PS OF CALL  
MOV (SP)+, 22(SP) ;RESTORE PC OF MAIN FLOW  
MOV (SP)+, 22(SP) ;RESTORE PS OF MAIN FLOW  
MOV (SP)+, R5 ;POP STACK INTO R5  
MOV (SP)+, R4 ;POP STACK INTO R4  
MOV (SP)+, R3 ;POP STACK INTO R3  
MOV (SP)+, R2 ;POP STACK INTO R2  
MOV (SP)+, R1 ;POP STACK INTO R1  
MOV (SP)+, R0 ;POP STACK INTO R0  
RTI

.SBTTL RANDOM NUMBER GENERATOR ROUTINE

\*\*\*\*\*  
;THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATOR  
;WITH A RANGE OF 0 TO 2(+33)-1.  
;CALL:  
; JSR PC, \$RAND ;CALL THE ROUTINE  
; RETURN ;RETURN HERE THE RANDOM  
; ;NUMBER WILL BE IN

```

7732
7733
7734 034170
7735 034170 010046
7736 034172 010146
7737 034174 010246
7738 034176 013700 034270
7739 034202 013701 034266
7740 034206 012702 177771
7741 034212 006300
7742 034214 006101
7743 034216 005202
7744 034220 001374
7745 034222 063700 034270
7746 034226 005501
7747 034230 063701 034266
7748 034234 062700 001057
7749 034240 005501
7750 034242 062701 047401
7751 034246 010037 034270
7752 034252 010137 034266
7753 034256 012602
7754 034260 012601
7755 034262 012600
7756 034264 000207
7757 034266 176543
7758 034270 123456
7759
7760
7761
7762
7763
7764
7765
7766
7767 034272 010046
7768 034274 010146
7769 034276 013746 000004
7770 034302 013746 000006
7771 034306 010600
7772
7773 034310 104400
7774 034312 012637 000006
7775 034316 012737 034336 000004
7776 034324 012701 020000
7777 034330 005711
7778 034332 005721
7779 034334 000775
7780 034336 162701 000002
7781 034342 010006
7782 034344 012637 000006
7783 034350 012637 000004
7784 034354 010137 034366
7785 034360 012601
7786 034362 012600
7787 034364 000207

```

```

;*                                     ;;SHINUM,$LONUM
$RAND: MOV RO,-(SP)                ;;PUSH RO ON STACK
        MOV R1,-(SP)                ;;PUSH R1 ON STACK
        MOV R2,-(SP)                ;;PUSH R2 ON STACK
        MOV $LONUM,R0                ;;SET RO WITH LOW
        MOV $SHINUM,R1                ;;SET R1 WITH HIGH
        MOV #-7,R2                    ;;SET SHIFT COUNT
1$: ASL RO                            ;;SHIFT RO LEFT AND
    ROL R1                            ;;ROTATE CARRY INTO R1 AND
    INC R2                            ;;CHECK FOR DONE
    BNE 1$                            ;;CONTINUE SHIFT LOOP
    ADD $LONUM,R0                    ;;ADD NUMBER TO MAKE X 129
    ADC R1                            ;;PROPAGATE CARRY
    ADD $SHINUM,R1                    ;;ADD NUMBER TO MAKE X 129
    ADD #1057,R0                      ;;ADD LOW CONSTANT
    ADC R1                            ;;PROPAGATE CARRY
    ADD #47401,R1                     ;;ADD HIGH CONSTANT
    MOV RO,$LONUM                    ;;SAVE RO
    MOV R1,$SHINUM                   ;;SAVE R1
    MOV (SP)+,R2                      ;;POP STACK INTO R2
    MOV (SP)+,R1                      ;;POP STACK INTO R1
    MOV (SP)+,R0                      ;;POP STACK INTO R0
    RTS PC                            ;;RETURN
$SHINUM: .WORD 176543
$LONUM: .WORD 123456
.SBTTL ROUTINE TO SIZE MEMORY

*****
*CALL:
* JSR PC,$SIZE
* RETURN
*$LSTAD WILL CONTAIN THE LAST AVAILABLE MEMORY LOCATION

$SIZE: MOV RO,-(SP)                ;;SAVE RO ON THE STACK
        MOV R1,-(SP)                ;;SAVE R1 ON THE STACK
        MOV @#ERRVEC,-(SP)          ;;SAVE PRESENT ERROR VECTOR PS & PC
        MOV @#ERRVEC+2,-(SP)
        MOV SP,R0                    ;;SAVE THE STACK POINTER
;;SET THE ERRVEC PS TO THE PRESENT PS
TRAP
        MOV (SP)+,@#ERRVEC+2        ;;PUSH OLD PSW AND PC ON STACK
        MOV #2$,@#ERRVEC            ;;SAVE THE PSW IN @#ERRVEC+2
        MOV #20000,R1                ;;SET FOR TIMEOUT
        MOV @#ERRVEC                ;;FIRST ADDRESS
1$: TST (R1)                          ;;TEST THIS ADDRESS
    TST (R1)+                        ;;STEP TO NEXT ADDRESS
    BR 1$                            ;;TRY ANOTHER
2$: SUB #2,R1                        ;;DROP BACK
    MOV RO,SP                        ;;RESTORE THE STACK
    MOV (SP)+,@#ERRVEC+2            ;;RESTORE ERROR VECTOR
    MOV (SP)+,@#ERRVEC
    MOV R1,$LSTAD                    ;;LAST ADDRESS
    MOV (SP)+,R1                    ;;RESTORE R1
    MOV (SP)+,R0                    ;;RESTORE RO
    RTS PC

```

```

7788 034366 000000 $LSTAD: .WORD 0 ;CONTAINS THE LAST ADDRESS
7789 .SBTTL POWER DOWN AND UP ROUTINES
7790
7791 ;*****
7792 ;POWER DOWN ROUTINE
7793 034370 012737 034530 000024 $PWRDN: MOV $SILLUP,2*PWRVEC ;SET FOR FAST UP
7794 034376 012737 000340 000026 MOV #340,2*PWRVEC+2 ;PRIO:7
7795 034404 010046 MOV R0,-(SP) ;PUSH R0 ON STACK
7796 034406 010146 MOV R1,-(SP) ;PUSH R1 ON STACK
7797 034410 010246 MOV R2,-(SP) ;PUSH R2 ON STACK
7798 034412 010346 MOV R3,-(SP) ;PUSH R3 ON STACK
7799 034414 010446 MOV R4,-(SP) ;PUSH R4 ON STACK
7800 034416 010546 MOV R5,-(SP) ;PUSH R5 ON STACK
7801 034420 017746 144514 MOV 2SWR,-(SP) ;PUSH 2SWR ON STACK
7802 034424 010637 034534 MOV SP,$SAVR6 ;SAVE SP
7803 034430 012737 034442 000024 MOV $PWRUP,2*PWRVEC ;SET UP VECTOR
7804 034436 000000 HALT
7805 034440 000776 BR .-2 ;HANG UP
7806
7807 ;*****
7808 ;POWER UP ROUTINE
7809 034442 012737 034530 000024 $PWRUP: MOV $SILLUP,2*PWRVEC ;SET FOR FAST DOWN
7810 034450 013706 034534 MOV $SAVR6,SP ;GET SP
7811 034454 005037 034534 CLR $SAVR6 ;WAIT LOOP FOR THE TTY
7812 034460 005237 034534 1$: INC $SAVR6 ;WAIT FOR THE INC
7813 034464 001375 BNE 1$ ;OF WORD
7814 034466 012677 144446 MOV (SP)+,2SWR ;POP STACK INTO 2SWR
7815 034472 012605 MOV (SP)+,R5 ;POP STACK INTO R5
7816 034474 012604 MOV (SP)+,R4 ;POP STACK INTO R4
7817 034476 012603 MOV (SP)+,R3 ;POP STACK INTO R3
7818 034500 012602 MOV (SP)+,R2 ;POP STACK INTO R2
7819 034502 012601 MOV (SP)+,R1 ;POP STACK INTO R1
7820 034504 012600 MOV (SP)+,R0 ;POP STACK INTO R0
7821 034506 012737 034370 000024 MOV $PWRDN,2*PWRVEC ;SET UP THE POWER DOWN VECTOR
7822 034514 012737 000340 000026 MOV #340,2*PWRVEC+2 ;PRIO:7
7823 034522 104401 TYPE ;REPORT THE POWER FAILURE
7824 034524 034536 $PWRMG: .WORD $POWER ;POWER FAIL MESSAGE POINTER
7825 034526 000002 RTI
7826 034530 000000 $SILLUP: HALT ;THE POWER UP SEQUENCE WAS STARTED
7827 034532 000776 BR .-2 ;BEFORE THE POWER DOWN WAS COMPLETE
7828 034534 000000 $SAVR6: 0 ;PUT THE SP HERE
7829 034536 005015 047520 042527 $POWER: .ASCIZ <15><12>"POWER"
7830 034544 000122
7831 .EVEN
7832 .SBTTL TRAP DECODER
7833
7834 ;*****
7835 ;THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
7836 ;AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
7837 ;OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
7838 ;GO TO THAT ROUTINE.
7839
7840 034546 010046 000002 $TRAP: MOV R0,-(SP) ;SAVE R0
7841 034550 016600 MOV 2(SP),R0 ;GET TRAP ADDRESS
7842 034554 005740 TST -(R0) ;BACKUP BY 2
7843 034556 111000 MOVB (R0),R0 ;GET RIGHT BYTE OF TRAP

```

7844 034560 006300  
7845 034562 016000 034602  
7846 034566 000200

ASL RO ;: POSITION FOR INDEXING  
MOV \$TRPAD(RO),RO ;: INDEX TO TABLE  
RTS RO ;: GO TO ROUTINE

7847  
7848  
7849

;: THIS IS USE TO HANDLE THE "GETPRI" MACRO

7850  
7851 034570 011646  
7852 034572 016666 000004 000002  
7853 034600 000002

\$TRAP2: MOV (SP),-(SP) ;: MOVE THE PC DOWN  
MOV 4(SP),2(SP) ;: MOVE THE PSW DOWN  
RTI ;: RESTORE THE PSW

7854  
7855  
7856

.SBTTL TRAP TABLE

7857  
7858  
7859

;: \*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED  
;: \*BY THE "TRAP" INSTRUCTION.

7860  
7861

ROUTINE

7862 034602 034570  
7863 034604 031466  
7864 034606 032156  
7865 034610 032132  
7866 034612 032172  
7867 034614 031706

\$TRPAD: .WORD \$TRAP2  
\$TYPE ;: CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE  
\$TYPOC ;: CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)  
\$TYP0S ;: CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)  
\$TYPON ;: CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)  
\$TYPDS ;: CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)

7868  
7869 034616 032746  
7870

\$GTSWR ;: CALL=GTSWR TRAP+6(104406) GET SOFT-SWR SETTING

7871 034620 032656  
7872 034622 033220  
7873 034624 033310

\$CKSWR ;: CALL=CKSWR TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR  
\$RDCHR ;: CALL=RDCHR TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE  
\$ROLIN ;: CALL=ROLIN TRAP+11(104411) TTY TYPEIN STRING ROUTINE  
\$SAVREG ;: CALL=SAVREG TRAP+12(104412) SAVE R0-R5 ROUTINE  
\$RESREG ;: CALL=RESREG TRAP+13(104413) RESTORE R0-R5 ROUTINE  
\$FPRINT ;: CALL=FPRINT TRAP+14(104414) FAILURE PRINT ROUTINE

7874 034626 034074  
7875 034630 034132  
7876 034632 016732  
7877 034634 000000

OFIL: .WORD 0  
HPDATA: .ASCII /IMMEDIATE COMMAND SUMMARY:/<15><12><12>

7878 034636 046511 042515 044504  
7879 034644 052101 020105 047503  
7880 034652 046515 047101 020104

7881 034660 052523 046515 051101  
7882 034666 035131 005015 012  
7883 034673 103 046517 040515

.ASCII /COMMAND NMENONIC PARAMETERS/<15><12><12>

7884 034700 042116 020040 020040  
7885 034706 020040 020040 020040  
7886 034714 020040 020040 02 40

7887 034722 047040 042515 047516  
7888 034730 044516 020103 020040  
7889 034736 020040 050040 051101

7890 034744 046501 052105 051105  
7891 034752 006523 005012

.ASCII /DRIVE SELECT DN ,DRIVE NUMBER/<15><12>

7892 034756 051104 053111 020105  
7893 034764 042523 042514 052103  
7894 034772 020040 020040 020040

7895 035000 020040 020040 020040  
7896 035006 020040 042040 020116  
7897 035014 020040 020040 020040

7898 035022 020040 042054 044522  
7899 035030 042526 047040 046525

CZR6RCO RK611/06 USR DEFINED  
CZR6RC.P11 02-DEC-77 11:07

MACY11 30(1046) 02-DEC-77 11:59 PAGE 151  
TRAP TABLE

SEQ 0150

7900	035036	042502	006522	012		
7901	035043	040	03040	020040	.ASCII /	,? (PRINT DRIVE SELECTED)/<15><12>
7902	035050	020040	03040	020040		
7903	035056	020040	03040	020040		
7904	035064	020040	03040	020040		
7905	035072	020040	03040	020040		
7906	035080	020040	03040	020040		
7907	035108	020040	03040	020077		
7908	035114	020040	044522	052116		
7909	035122	042040	044522	042526		
7910	035130	051440	046105	041505		
7911	035136	042524	024504	005015		
7912	035144	012				
7913	035145	0106	051117	040515	.ASCII /FORMAT SELECT	FT ,FORMAT(24 OR 26)/<15><12>
7914	035152	020124	042523	042514		
7915	035160	052103	020040	020040		
7916	035166	03040	03040	020040		
7917	035174	03040	03040	052106		
7918	035202	03040	03040	020040		
7919	035210	020040	026040	047506		
7920	035216	046522	052101	031050		
7921	035224	020064	051117	031040		
7922	035232	024456	005015			
7923	035236	020040	020040	020040	.ASCII /	,? (PRINT FORMAT SELECTED)/<15><1>
7924	035244	020040	020040	020040		
7925	035252	020040	020040	020040		
7926	035260	03040	03040	020040		
7927	035268	03040	03040	020040		
7928	035274	03040	03040	020040		
7929	035302	020040	037454	024040		
7930	035310	051120	047111	020124		
7931	035316	047506	046522	052101		
7932	035324	051440	046105	041505		
7933	035332	042524	024504	005015		
7934	035340	0517	050124	052125	.ASCII /OUTPUT TEST	OT ,O (OBJECT)/<15><12>
7935	035346	03040	051505	020124		
7936	035354	03040	020040	020040		
7937	035362	03040	020040	020040		
7938	035370	03040	047440	020124		
7939	035376	03040	020040	020040		
7940	035404	020040	047454	024040		
7941	035412	041117	042512	052103		
7942	035420	006451	012			
7943	035423	03040	020040	020040	.ASCII /	,S (SOURCE)/<15><12><12>
7944	035430	020040	020040	020040		
7945	035436	020040	020040	020040		
7946	035444	020040	020040	020040		
7947	035452	020040	020040	020040		
7948	035460	020040	020040	020040		
7949	035466	020040	026040	020123		
7950	035474	051450	052517	041522		
7951	035502	024505	005015	012		
7952	035507	111	050116	052125	.ASCII /INPUT TEST	IT/<15><12><12>
7953	035514	052040	051505	020124		
7954	035522	020040	020040	020040		
7955	035530	020040	020040	020040		

7956	035536	020040	020040	052111		
7957	035544	005015	012			
7958	035547	111	050116	052125	.ASCII /INPUT STRING	IS/<15><12><12>
7959	035554	051440	051124	047111		
7960	035562	020107	020040	020040		
7961	035570	020040	020040	020040		
7962	035576	020040	020040	051511		
7963	035604	005015	012			
7964	035607	103	050117	020131	.ASCII /COPY TAPE	CT/<15><12><12>
7965	035614	040524	04257	020040		
7966	035622	020040	02004	020040		
7967	035630	020040	020040	020040		
7968	035636	020040	020040	052103		
7969	035644	005015	012			
7970	035647	111	042524	040522	.ASCII /ITERATION COUNT	IC ,NNNNN (DECIMAL COUNT)/<15><12>
7971	035654	044524	047117	041440		
7972	035662	052517	052116	020040		
7973	035670	020040	020040	020040		
7974	035676	020040	020040	041511		
7975	035704	020040	020040	020040		
7976	035712	020040	026040	047116		
7977	035720	047116	020116	042050		
7978	035726	041505	046511	046101		
7979	035734	041440	052517	052116		
7980	035742	006451	012			
7981	035745	040	020040	020040	.ASCII /	,? (PRINT COUNT)/<15><12><12>
7982	035752	020040	020040	020040		
7983	035760	020040	020040	020040		
7984	035766	020040	020040	020040		
7985	035774	020040	020040	020040		
7986	036002	020040	020040	020040		
7987	036010	020040	026040	020077		
7988	036016	050050	044522	752116		
7989	036024	041440	052517	052116		
7990	036032	006451	005012			
7991	036036	02502	043106	051105	.ASCII /BUFFER DUMP	BD ,BUFFER NAME(H,R,W,X,Y, OR Z)/<15
7992	036044	042040	046525	020120		
7993	036052	020040	020040	020040		
7994	036060	020040	020040	020040		
7995	036066	020040	041040	020104		
7996	036074	020040	020040	020040		
7997	036102	020040	041054	043125		
7998	036110	042506	020122	040516		
7999	036116	042515	044050	051054		
8000	036124	053454	054054	054454		
8001	036132	020054	051117	055040		
8002	036140	006451	012			
8003	036143	040	020040	020040	.ASCII /	,NUMBER OF WORDS/<15><12>
8004	036150	020040	020040	020040		
8005	036156	020040	020040	020040		
8006	036164	020040	020040	020040		
8007	036172	020040	020040	020040		
8008	036200	020040	020040	020040		
8009	036206	020040	026040	052516		
8010	036214	041115	051105	047440		
8011	036222	020106	047527	042122		

8012	036230	006523	012				
8013	036233	123	042520	044503	.ASCII /SPECIAL DATA PATTERN	DP	,PATTERN NAME (X,Y,Z)/<15><12>
8014	036240	046101	042040	052101			
8015	036246	020101	040520	052124			
8016	036254	051105	020116	020040			
8017	036262	020040	020040	050104			
8018	036270	020040	020040	020040			
8019	036276	020040	026040	040520			
8020	036304	052124	051105	020116			
8021	036312	040516	042515	024040			
8022	036320	026130	026131	024532			
8023	036326	005015					
8024	036330	020040	020040	020040	.ASCII /		,DDDD...D(32 WORDS)/<15><12><12>
8025	036336	020040	0 0040	020040			
8026	036344	020040	0 0040	020040			
8027	036352	020040	0 0040	020040			
8028	036360	020040	0 0040	020040			
8029	036366	0 0040	0 0040	020040			
8030	036374	020040	042054	042104			
8031	036402	027104	027056	024104			
8032	036410	031063	053440	051117			
8033	036416	051504	006451	005012			
8034	036424	042105	052111	041040	.ASCII /EDIT BUFFER	EB	,PATTER NAME (X,Y,Z)/<15><12>
8035	036432	043125	042506	020122			
8036	036440	020040	020040	020040			
8037	036446	020040	020040	020040			
8038	036454	0 0040	042440	020102			
8039	036462	020040	020040	020040			
8040	036470	020040	050054	052101			
8041	036476	042524	020122	040516			
8042	036504	042515	024040	026130			
8043	036512	026131	024532	005015			
8044	036520	020040	020040	020040	.ASCII /		,STARTING WORD NUMBER/<15><12>
8045	036526	020040	020040	020040			
8046	036534	020040	020040	020040			
8047	036542	020040	020040	020040			
8048	036550	020040	020040	020040			
8049	036556	020040	020040	020040			
8050	036554	020040	051454	040524			
8051	036572	052122	047111	020107			
8052	036600	047527	042122	047040			
8053	036606	046525	042502	006522			
8054	036614	012					
8055	036615	040	020040	020040	.ASCII /		,DD..D(UP TO 32 WORDS)/<15><12>
8056	036622	020040	020040	020040			
8057	036630	020040	020040	020040			
8058	036636	020040	020040	020040			
8059	036644	020040	020040	020040			
8060	036652	020040	020040	020040			
8061	036660	020040	026040	042104			
8062	036666	027056	024104	050125			
8063	036674	052040	020117	031063			
8064	036702	053440	051117	051504			
8065	036710	006451	012				
8066	036713	103	046517	044520	.ASCII /COMPILE	CO	,NC (NO CHECK)/<15><12>
8067	036720	042514	020040	020040			

8068	036726	020040	020040	020040		
8069	036734	020040	020040	020040		
8070	036742	020040	020040	047503		
8071	036750	020040	020040	020040		
8072	036756	020040	026040	041516		
8073	036764	024040	047516	041440		
8074	036772	042510	045503	006451		
8075	037000	012				
8076	037001	040	020040	020040	.ASCII /	,BII (BUS INCREMENT INHIBIT)/<15>
8077	037006	020040	020040	020040		
8078	037014	020040	020040	020040		
8079	037022	020040	020040	020040		
8080	037030	020040	020040	020040		
8081	037036	020040	020040	020040		
8082	037044	020040	026040	044502		
8083	037052	020111	041050	051525		
8084	037060	044440	041516	042522		
8085	037066	042515	052116	044440		
8086	037074	044116	041111	052111		
8087	037102	006451	005012			
8088	037106	042105	052111	040440	.ASCII /EDIT ADD LINE	EA ,LINE NUMBER/<15><12>
8089	037114	042104	046040	047111		
8090	037122	020105	020040	020040		
8091	037130	000040	020040	020040		
8092	037136	000040	042440	020101		
8093	037144	000040	020040	020040		
8094	037152	000040	046054	047111		
8095	037160	00105	051516	041115		
8096	037166	051105	000015			
8097	037172	020040	020040	020040	.ASCII /	,NEW COMMAND/<15><12><12>
8098	037200	000040	020040	020040		
8099	037206	000040	020040	020040		
8100	037214	000040	020040	020040		
8101	037222	000040	020040	020040		
8102	037230	000040	020040	020040		
8103	037236	020040	047054	053505		
8104	037244	041440	046517	040515		
8105	037252	042116	005015	012		
8106	037257	0105	044504	020124	.ASCII /EDIT DELETE LINE	ED ,LINE NUMBER/<15><12><12>
8107	037264	042504	042514	042524		
8108	037272	046040	047111	020105		
8109	037300	020040	020040	020040		
8110	037306	020040	020040	042105		
8111	037314	020040	020040	020040		
8112	037322	020040	026040	044514		
8113	037330	042516	047040	046525		
8114	037336	042502	006522	005012		
8115	037344	051120	047111	020124	.ASCII /PRINT TEST	PT/<15><12><12>
8116	037352	042524	052123	020040		
8117	037360	020040	020040	020040		
8118	037366	020040	020040	020040		
8119	037374	020040	050040	006524		
8120	037402	005012				
8121	037404	051120	047111	020124	.ASCII /PRINT LINE	PL ,LINE NUMBER/<15><12><12>
8122	037412	044514	042516	020040		
8123	037420	020040	020040	020040		



8124	037426	020040	020040	020040		
8125	037434	020040	050040	020114		
8126	037442	020040	020040	020040		
8127	037450	020040	046054	047111		
8128	037456	020105	052516	041115		
8129	037464	051105	005015	012		
8130	037471	116	053505	052040	.ASCII	/NEW TEST NT/<15><12><12>
8131	037476	051505	020124	020040		
8132	037504	020040	020040	020040		
8133	037512	020040	020040	020040		
8134	037520	011040	020040	052116		
8135	037526	005015	012			
8136	037531	122	047125	020040	.ASCII	/RUN RU/<15><12><12>
8137	037536	020040	020040	020040		
8138	037544	020040	020040	020040		
8139	037552	020040	020040	020040		
8140	037560	020040	020040	052522		
8141	037566	005015	012			
8142	037571	120	044522	052116	.ASCII	/PRINT REGISTER PR ,REGISTER NAME OR NUMBER/<15><12>
8143	037576	051040	043505	051511		
8144	037604	042524	020122	020040		
8145	037612	020040	020040	020040		
8146	037620	020040	020040	051120		
8147	037626	020040	020040	020040		
8148	037634	020040	026040	042522		
8149	037642	044507	052123	051105		
8150	037650	047040	046501	020105		
8151	037656	051117	047040	046525		
8152	037664	042502	006522	005012		
8153	037672	042510	050114	020040	.ASCII	/HELP HP/<15><12><12>
8154	037700	020040	020040	020040		
8155	037706	020040	020040	020040		
8156	037714	020040	020040	020040		
8157	037722	020040	044040	006520		
8158	037730	005012				
8159	037732	044524	042515	047440	.ASCII	/TIME OUT CHANGE TO ,NNNNN (DECIMAL)/<15><12>
8160	037740	052125	041440	040510		
8161	037746	043516	020105	020040		
8162	037754	020040	020040	020040		
8163	037762	020040	052040	020117		
8164	037770	020040	020040	020040		
8165	037776	020040	047054	047116		
8166	040004	047116	024040	042504		
8167	040012	044503	040515	024514		
8168	040020	005015				
8169	040022	020040	020040	020040	.ASCII	/ ,? (PRINT CONSTANT)/<15><12><12>
8170	040030	020040	020040	020040		
8171	040036	020040	020040	020040		
8172	040044	020040	020040	020040		
8173	040052	020040	020040	020040		
8174	040060	020040	020040	020040		
8175	040066	020040	037454	024040		
8176	040074	051120	047111	020124		
8177	040102	047503	051516	040524		
8178	040110	052116	006451	005012		
8179	040116	046101	020114	042504	.ASCII	/ALL DECIMAL VALUES MUST BE 65535(10) OR LESS/<15><12><12><12>

8180	040124	044503	040515	020114
8181	040132	040526	052514	051505
8182	040140	046440	051525	020124
8183	040146	042502	033040	032465
8184	040154	032463	030450	024460
8185	040162	047440	020122	042514
8186	040170	051523	005015	005012
8187	040176	042504	042506	051122
8188	040204	042105	041440	046517
8189	040212	040515	042116	051440
8190	040220	046525	040515	054522
8191	040226	006472	005012	
8192	040232	047503	046515	047101
8193	040240	020104	020040	020040
8194	040246	020040	020040	020040
8195	040254	020040	020040	020040
8196	040262	020040	046516	047105
8197	040270	047117	041511	020040
8198	040276	020040	020040	040520
8199	040304	040522	042515	042524
8200	040312	006522	005012	
8201	040316	052523	051502	051531
8202	040324	042524	020115	052506
8203	040332	041516	044524	047117
8204	040340	020040	020040	020040
8205	040346	020040	020040	051440
8206	040354	020106	020040	020040
8207	040362	020040	020040	051454
8208	040370	041125	054523	052123
8209	040376	046505	041440	047115
8210	040404	006504	012	
8211	040407	040	020040	020040
8212	040414	020040	020040	020040
8213	040422	020040	020040	020040
8214	040430	020040	020040	020040
8215	040436	020040	020040	020040
8216	040444	020040	020040	020040
8217	040452	020040	020040	026040
8218	040460	051104	053111	020105
8219	040466	052516	006515	012
8220	040473	040	020040	020040
8221	040500	020040	020040	020040
8222	040506	020040	020040	020040
8223	040514	020040	020040	020040
8224	040522	020040	020040	020040
8225	040530	020040	020040	020040
8226	040536	020040	020040	026040
8227	040544	054503	044514	042116
8228	040552	051105	005015	
8229	040556	020040	020040	020040
8230	040564	020040	020040	020040
8231	040572	020040	020040	020040
8232	040600	020040	020040	020040
8233	040606	020040	020040	020040
8234	040614	020040	020040	020040
8235	040622	020040	020040	052054

.ASCII /DEFERRED COMMAND SUMMARY:/&lt;15&gt;&lt;12&gt;&lt;12&gt;

.ASCII /COMMAND NMENONIC PARAMETER/&lt;15&gt;&lt;12&gt;&lt;12&gt;

.ASCII /SUBSYSTEM FUNCTION SF ,SUBSYSTEM CMND/&lt;15&gt;&lt;12&gt;

.ASCII / ,DRIVE NUM/&lt;15&gt;&lt;12&gt;

.ASCII / ,CYLINDER/&lt;15&gt;&lt;12&gt;

.ASCII / ,TRACK/&lt;15&gt;&lt;12&gt;

8236	040630	040522	045503	005015				
8237	040636	020040	020040	020040	.ASCII /			,SECTOR/(15)(12)
8238	040644	020040	020040	020040				
8239	040652	020040	020040	020040				
8240	040660	020040	020040	020040				
8241	040666	020040	020040	020040				
8242	040674	020040	020040	020040				
8243	040702	020040	020040	051454				
8244	040710	041505	047524	006522				
8245	040716	012						
8246	040717	040	020040	020040	.ASCII /			,WORD COUNT/(15)(12)
8247	040724	020040	020040	020040				
8248	040732	020040	020040	020040				
8249	040740	020040	020040	020040				
8250	040746	020040	020040	020040				
8251	040754	020040	020040	020040				
8252	040762	020040	020040	026040				
8253	040770	047527	042122	041440				
8254	040776	052116	005015					
8255	041004	040	020040	020040	.ASCII /			,DATA PATTERN/(15)(12)
8256	041012	040	020040	020040				
8257	041020	040	020040	020040				
8258	041026	040	020040	020040				
8259	041034	040	020040	020040				
8260	041042	040	020040	020040				
8261	041050	040	042054					
8262	041056	052101	047520					
8263	041064	052124	051105	00516				
8264	041072	012						
8265	041073	012	043125	072506	.ASCII /BUFFER INITIALIZE	BI		,PATTERN SELECT/(15)(12)(12)
8266	041100	020122	047111	052111				
8267	041106	040511	044511	042532				
8268	041114	040	020040	020040				
8269	041122	040	020040	020040				
8270	041130	040	020040	020040				
8271	041136	040	026040					
8272	041144	052124	051105					
8273	041152	0116	042523	042514				
8274	041160	052103	005015	012				
8275	041165	104	052101	020101	.ASCII /DATA CC PARE	DC		,NNNNNN (OCTAL)/(15)(12)(12)
8276	041172	047503	050115	051101				
8277	041200	020105	020040	020040				
8278	041206	020040	020040	020040				
8279	041214	020040	020040	020040				
8280	041222	041504	020040	020040				
8281	041230	020040	020040	026040				
8282	041236	047116	047116	047116				
8283	041244	024040	041517	040524				
8284	041252	044514	005015	012				
8285	041257	123	04024	052524	.ASCII /STATUS COMPARE	SC		,STATUS WD SELECT/(15)(12)
8286	041264	020123	047503	050115				
8287	041272	051101	020105	020040				
8288	041300	020040	020040	020040				
8289	041306	020040	020040	020040				
8290	041314	041523	020040	020040				
8291	041322	020040	020040	026040				

8292	041330	052123	052101	051525			
8293	041336	053440	020104	042523			
8294	041344	042514	052103	005015			
8295	041352	020040	020040	020040	.ASCII /		,EXPECTED VALUE/<15><12>
8296	041360	020040	020040	020040			
8297	041366	020040	020040	020040			
8298	041374	020040	020040	020040			
8299	041402	020040	020040	020040			
8300	041410	020040	020040	020040			
8301	041416	020040	020040	042454			
8302	041424	050130	041505	042524			
8303	041432	020104	040526	052514			
8304	041440	006505	012				
8305	041443	040	020040	020040	.ASCII /		,MASK/<15><12><12>
8306	041450	020040	020040	020040			
8307	041456	020040	020040	020040			
8308	041464	020040	020040	020040			
8309	041472	020040	020040	020040			
8310	041500	020040	020040	020040			
8311	041506	020040	020040	026040			
8312	041514	040515	045523	005015			
8313	041522	012					
8314	041523	122	043505	051511	.ASCIIZ /REGISTER WRITE	RW	,REG NAME OR NUM/<15><12>
8315	041530	042524	020122	051127			
8316	041536	052111	020105	020040			
8317	041544	020040	020040	020040			
8318	041552	020040	020040	020040			
8319	041560	053522	020040	020040			
8320	041566	020040	020040	026040			
8321	041574	042522	020107	040516			
8322	041602	042515	047440	020122			
8323	041610	052516	006515	000012			
8324							
8325	041616	020040	020040	020040	ENDLOC: .EVEN		
8326	041624	020040	020040	020040	.ASCII /		,VALUE/<15><12><12>
8327	041632	020040	020040	020040			
8328	041640	020040	020040	020040			
8329	041646	020040	020040	020040			
8330	041654	020040	020040	020040			
8331	041662	020040	020040	053054			
8332	041670	046101	042525	005015			
8333	041676	012					
8334	041677	122	043505	051511	.ASCII /REGISTER COMPARE	RC	,REG NAME OR NUM/<15><12>
8335	041704	042524	020122	047503			
8336	041712	050115	051101	020105			
8337	041720	020040	020040	020040			
8338	041726	020040	020040	020040			
8339	041734	041522	020040	020040			
8340	041742	020040	020040	020040			
8341	041750	042522	020107	040516			
8342	041756	042515	047440	020122			
8343	041764	052516	006515	012			
8344	041771	040	020040	020040	.ASCII /		,EXPECTED VALUE/<15><12>
8345	041776	020040	020040	020040			
8346	042004	020040	020040	020040			
8347	042012	020040	020040	020040			

8348	042020	020040	020040	020040			
8349	042026	020040	020040	020040			
8350	042034	020040	020040	026040			
8351	042042	054105	042520	052103			
8352	042050	042105	053040	046101			
8353	042056	042525	005015				
8354	042062	020040	020040	020040	.ASCII /		,MASK/<15><12><12>
8355	042070	020040	020040	020040			
8356	042076	020040	020040	020040			
8357	042104	020040	020040	020040			
8358	042112	020040	020040	020040			
8359	042120	020040	020040	020040			
8360	042126	020040	020040	046454			
8361	042134	051501	006513	005012			
8362	042142	052123	046101	020114	.ASCII /STALL	ST	,NNNN (DECIMAL)/<15><12><12>
8363	042150	020040	020040	020040			
8364	042156	020040	020040	020040			
8365	042164	020040	020040	020040			
8366	042172	020040	020040	051440			
8367	042200	020124	020040	020040			
8368	042206	020040	020040	047054			
8369	042214	047116	047116	024040			
8370	042232	042504	044503	040515			
8371	042238	024514	005015	012			
8372	042235	120	044522	052116	.ASCII /PRINT MESSAGE	PM	,MESSAGE/<15><12><12>
8373	042242	046440	051505	040523			
8374	042250	042507	020040	020040			
8375	042256	020040	020040	020040			
8376	042264	020040	020040	020040			
8377	042272	046520	020040	020040			
8378	042300	020040	020040	026040			
8379	042306	042515	051523	043501			
8380	042314	006505	005012				
8381	042320	047125	041111	051525	.ASCII /UNIBUS INITIALIZE	UI/<15><12><12>	
8382	042326	044440	044516	044524			
8383	042334	046101	055111	020105			
8384	042342	020040	020040	020040			
8385	042350	020040	020040	052440			
8386	042356	006511	005012				
8387	042362	046101	020114	042504	.ASCII /ALL DECIMAL VALUES MUST BE LESS THAN 65535(10)/<15><12><12><12>		
8388	042370	044503	040515	020114			
8389	042376	040526	052514	051505			
8390	042404	046440	051525	020124			
8391	042412	042502	046040	051505			
8392	042420	020123	044124	047101			
8393	042426	033040	032465	032463			
8394	042434	037450	024460	005015			
8395	042442	005012					
8396	042444	052523	051502	051531	.ASCII /SUBSYSTEM COMMANDS:/<15><12><12>		
8397	042452	042524	020115	047503			
8398	042460	046515	047101	051504			
8399	042466	006472	005012				
8400	042472	047503	046515	047101	.ASCII /COMMAND	NMENONIC/<15><12><12>	
8401	042500	020104	020040	020040			
8402	042506	020040	020040	020040			
8403	042514	020040	046516	047105			

8404	042522	047117	041511	005015		
8405	042530	012				
8406	042531	012	041005	020104	.ASCII /READ DATA	RD/<15><12>
8407	042536	040504	040524	020040		
8408	042544	020040	020040	020040		
8409	042552	020040	020040	020040		
8410	042560	042122	005015			
8411	042564	051127	052111	020105	.ASCII /WRITE DATA	WD/<15><12>
8412	042572	040504	040524	020040		
8413	042600	020040	020040	020040		
8414	042606	020040	020040	053440		
8415	042614	006504	012			
8416	042617	012	044522	042524	.ASCII /WRITE CHECK	WC/<15><12>
8417	042624	041440	042510	045503		
8418	042632	020040	020040	020040		
8419	042640	020040	020040	020040		
8420	042646	041527	005015			
8421	042652	051127	052111	020105	.ASCII /WRITE HEADERS	WH/<15><12>
8422	042658	042510	042101	051105		
8423	042666	020123	020040	020040		
8424	042674	020040	020040	053440		
8425	042702	006510	012			
8426	042705	122	040505	020104	.ASCII /READ HEADER	RH/<15><12>
8427	042712	042510	042101	051105		
8428	042720	020040	020040	020040		
8429	042726	020040	020040	020040		
8430	042734	044122	005015			
8431	042740	042523	045505	020040	.ASCII /SEEK	SK/<15><12>
8432	042746	020040	020040	020040		
8433	042754	020040	020040	020040		
8434	042762	020040	020040	051440		
8435	042770	006513	012			
8436	042773	103	042514	051101	.ASCII /CLEAR SUBSYSTEM	CS/<15><12>
8437	043000	051440	041125	054523		
8438	043006	052123	046505	020040		
8439	043014	020040	020040	020040		
8440	043022	051503	005015			
8441	043026	047503	052116	047522	.ASCII /CONTROLLER CLEAR	CC/<15><12>
8442	043034	046114	051105	041440		
8443	043042	042514	051101	020040		
8444	043050	020040	020040	041440		
8445	043056	006503	012			
8446	043061	104	044522	042526	.ASCII /DRIVE CLEAR	DC/<15><12>
8447	043066	041440	042514	051101		
8448	043074	020040	020040	020040		
8449	043102	020040	020040	020040		
8450	043110	041504	005015			
8451	043114	042522	040503	044514	.ASCII /RECALIBRATE	RC/<15><12>
8452	043122	051102	052101	020105		
8453	043130	020040	020040	020040		
8454	043136	020040	020040	051040		
8455	043144	006503	012			
8456	043147	104	044522	042526	.ASCII /DRIVE SELECT	DS/<15><12>
8457	043154	051440	046105	041505		
8458	043162	020124	020040	020040		
8459	043170	020040	020040	020040		

8460	043176	051504	005015			
8461	043202	040520	045503	040440	.ASCII	/PACK ACK PA/<15><12>
8462	043210	045503	020040	020040		
8463	043216	020040	020040	020040		
8464	043224	020040	020040	050040		
8465	043232	006501	012			
8466	043235	125	046116	040517	.ASCII	/UNLOAD UL/<15><12>
8467	043242	020104	020040	020040		
8468	043250	020040	020040	020040		
8469	043256	020040	020040	020040		
8470	043264	046125	005015			
8471	043270	052123	051101	020124	.ASCII	/START SPINDLE SS/<15><12>
8472	043276	050123	047111	046104		
8473	043304	020105	020040	020040		
8474	043312	020040	020040	051440		
8475	043320	006523	012			
8476	043323	117	043106	042523	.ASCII	/OFFSET OF/<15><12>
8477	043330	020124	020040	020040		
8478	043336	020040	020040	020040		
8479	043344	020040	020040	020040		
8480	043352	043117	005015			
8481	043356	042522	042101	040440	.ASCII	/READ ALL HEADERS AH/<15><12><12><12>
8482	043354	046114	044040	040505		
8483	043372	042504	051522	020040		
8484	043400	020040	020040	040440		
8485	043406	006510	005012	012		
8486	043413	104	052101	020101	.ASCII	/DATA PATTERNS:/<15><12><12>
8487	043420	040520	052124	051105		
8488	043426	051516	006472	005012		
8489	043434	040520	052124	051105	.ASCII	/PATTERNS "A" THROUGH "I" ARE PROVIDED./<15><12>
8490	043442	051516	021040	021101		
8491	043450	052040	051110	052517		
8492	043456	044107	021040	021111		
8493	043464	040440	042522	050040		
8494	043472	047522	044526	042504		
8495	043500	027104	005015			
8496	043504	042522	042506	020122	.ASCIZ	/REFER TO THE FUNCTIONAL SPEC FOR DETAILS./<15><12><12>
8497	043512	047524	052040	042510		
8498	043520	043040	047125	052103		
8499	043526	047511	040516	020114		
8500	043534	050123	041505	043040		
8501	043542	051117	042040	052105		
8502	043550	044501	051514	006456		
8503	043556	005012	000			
8504		000001				

.END

[illegible]



[illegible]

C. SPEC	030156	6795	6891#						
DCK =	100000	1899#	6146	6312	6548				
DCLO =	000020	1908#							
DCMDPR	004510	2474	2505#						
DOISP =	177570	1553#	1687	2401					
DDT =	000400	1913#							
DECBIN	031314	2646	2749	2837	2976	3056	4005	7101#	
DESL =	000010	1863#							
DI =	040000	1848#	3533	6332	7004				
DISPLA	001142	1687#	2401*	2409*					
DISPRE	000174	1653#	2409						
DLT =	100000	1877#	6140	6270	6306	6546			
DMD =	000040	1924#							
DMPHOR	003465	2295#	3465						
DN RTE	006144	2323	3005#						
DONE	001667	1758#	4580*	4585	4600*	4773*	5024*		
DP RTE	006356	2341	3092#						
DP RTE1	006362	3091	3093#						
DRA =	000001	1903#	4993	6135	6918				
DRDY =	000200	1912#							
DRIVE	001176	1705#	3012	3021*	4260*	4281	5517*	5519	
DROT =	000040	1909#							
DRPAR =	000010	1886#							
DRPDRV=	002000	2064#	6780						
DRVCAL	013570	4580#	5141	5649					
DRVOSC=	000040	2056#	4791	6220	6386	6448			
DRVERR	015036	4766#	5002	5046	5344	5913			
DRVHRD=	000020	2055#	4788	6227	6422	6553			
DRVMSK=	000007	1862#	6129	6170	6321	6766			
DRVPOT=	000004	2053#	6328	6396					
DRVPOS=	000002	2052#	6328	6355	6817				
DRVSZO=	010000	2066#	4797	4800	6158	6920			
DRV TYP	001244	1728#	5529	5752	5766*	5767	5769	5777*	5779*
DRVUSE=	000001	2051#	6780						
DRY =	000200	1911#							
DR. CLR=	000005	1834#	6208	6371	6413	6437	6551		
DR. SEL=	000001	1833#	6278	6357	6672	6679	6686	6693	6897
DSC =	040000	1916#							
DSCNOT	014731	4752#	4796						
DSWR =	177570	1552#	1686	2400					
DTBAII=	100000	2069#	5524	6780	6879				
DTE =	010000	1896#	6548						
DT RTE	022450	2325	5745#						
DTYE =	000040	1888#	6548						
EARTE	004764	2345	2628#						
EB RTE	006346	2339	3090#						
ECCW =	020000	1932#							
ECH =	000100	1889#	6146	6312	6548				
ED RTE	005266	2347	2739#						
EMTVEC=	000030	1641#							
ENDLOC	041616	2417	2419	2423	3361	8325#			
EQSGN	002510	2200#	2861	2905	3699				
ERRFRE	013644	4600#	5912						
ERRVEC=	000004	1634#	2398	2399*	2410*	7769	7770	7774*	7775*
ETABL	023454	5040	5868#						
ESAH'	021634	4076							

[illegible]

CZR6RCO RK611/06 USR DEFINED  
CZR6RC.P11 02-DEC-77 11:07MACY11 30(1046) 02-DEC-77 11:59 PAGE 167  
CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0165

GDDAT	014467	4722#	5169	5228	5288									
GNS	= ***** U	1652	7863	7864	7865	7866	7867	7869	7871	7872	7873	7874	7875	7876
GO	= 000001	1838#												
GODRV	022142	5631	5640	5645	5648#									
GTSWR	= 104406	7869#												
HARDER	014646	4743#	4790											
HCRC	= 000400	1891#												
HOBUFF	001736	1781#	3448	5635										
HDR.AO	023606	5955#	6264	6268*	6958*									
HDR.CT	023610	5956#	6272*	6961*	6964*									
HELPO	002400	2186#	2440											
HPDATA	034636	3364	7878#											
HPFILE	002656	2219#	3360											
HPRTE	007366	2359	3358#											
HPVLD	001672	1761#	2438	2476*	3358									
HT	= 000011	1544#	7175	7216										
HVRC	= 000400	1892#	6146	6312	6548									
H.HEAD	023613	5959#												
IBUFPT	001710	1770#	2423*	2424*	2511	2512	2522	2635	2756	2844	2900	3389	3427	3644
		4347	4381	4480	5256	5608	5617							
ICDEC	004704	2463	2586#	2656	3656									
ICRTE	006254	2335	3046#											
ICTBL	003534	2320#	2591											
IDAE	= 002000	1894#												
IE	= 000100	1839#	5051	6492	6556	6844	6885	6951	7018					
ILC	= 000001	1881#	6548											
ILF	= 000004	1884#												
INTERR	003023	2238#	3693											
INTMSK	023617	5964#	6187	6211	6232	6257	6343	6375	6416	6427	6440	6711*	6715*	6719*
		6768#	6854											
INTR	= 000300	1829#												
IOBFSZ	003321	2276#	2433											
IOTVEC	= 000020	1639#												
IR	= 000100	1868#												
ISRTE	013020	2333	4439#											
ITCNT	001232	1719#	3051	3058*	3536	4549								
ITRTE	013026	2329	3565	4440#										
IVDADD	002554	2207#	2698											
IVDEDT	002532	2204#	2696	2791										
IVDLN	002575	2210#	2700	2795	2870									
IVDNC	003123	2251#	4234											
IVDPAR	002616	2213#	3188	3484	4226	4387								
IVDRUN	002756	2230#	3568	4389										
IVDWCT	003101	2247#	4232											
I.AERR	025646	6363	6369	6387	6396#									
I.ATTN	025176	6123	6167	6300#										
I.CCLR	026232	6201	6251	6482#	6554	6914	6979							
I.CSTS	026664	6194	6629#											
I.CST1	026706	6198	6250	6633#	6903									
I.DRV	023620	5968#	6768	6769										
I.ERR	024420	6195	6199#											
I.ERRA	024400	6194#	6262											
I.ERRC	024406	6132	6185	6197#	6330									
I.HDAL	024644	6178	6241#											
I.INTR	023764	2413	6102#											
I.ISRL	023612	5958#	6116	6119*	6122*	6493*	6790*	6944*	7017*					

SEQ 0166

6209	6279	6358	6372	6414	6438	6524#	6673	6680	6687	6694		
6120	6137	6164#										
6114	6152	6162	6190	6202	6207	6210	6216	6225	6234	6237	6252	6254
6280	6283	6292	6296	6318	6334	6341	6347	6359	6373	6380	6392	6412
6415	6420	6429	6432	6439	6444	6453#						
6206	6411	6670#	6924									
6534	6584#											
6353	6436#											
1774#	3652											
7765												
2290#	3554											
1545#	7210	7216										
1752#	3390#	3539*	3562*	4548*	4806	4924	5134*	5160	5200*	5219	5255*	5274
5318#	5334#	5362*	5508*	5544*								
1727#	2509#	2530*	2531	2649	2664*	2752	2782*	2840	3391*	3663		
3537	3541	3543#										
3557	3560#	3569										
1740#	4057*	4287										
1775#	3525*	3551*	4551	5081	5085							
1776#	3526*	3552*	5083									
2287#	5090											
3538#	3555	3559										
1766#	2428*	2510	2520	2636	3394	4249	5364	5365	5609			
1927#												
1871#	6110	6537	6912									
1928#												
1921#												
1929#												
1926#												
4734#	5175	5234										
1925#												
1997#												
1996#												
1993#												
2000#												
1995#												
1994#												
1999#												
1998#												
1874#	6125	6203	6546	6916								
1873#	6140	6306	6546									
2060#	4587	5527	6780	6842	6883	6910	6928	6949	6972	6981	7015	
1731#	3633	3638	5525									
2065#	4794	6368										
2196#	2525	2640										
2235#	2872	2916	3691	4391								
2353	3383#											
2553#	2556	2643	2654									
1885#												
1763#	3398	3645	3665									
1769#	2417*	3431	5257	5363	5366	5488	5624	5697				
2947	3019	3123	3173	3245	3460	3860	3967	4245	5764	7046#		
1756#	4173*	4180*	4193*	4263	4285	4						

[illegible]

**M13**

PUCODE	001174	1704#	4339*	4345*	4349	4379	4444	4475						
PUERR	003207	2261#	4393	5677										
PUFLSZ	001250	1736#	4348*	4382	4477	4484	4487							
PUNCH	012760	4351	4358	4364	4370	4376	4383	4409#						
PWRVEC=	000024	1640#	2394*	2395*	7793*	7794*	7803*	7809*	7821*	7822*				
P.ASOF=	000032	2082#	4903	6374*	6375	6408*	6639*	6827*						
P.ACUM=	000040	2085#	5146	6365*	6366	6362*	6383	6696*	6904*					
P.A01 =	000044	2087#	6675*											
P.A10 =	000050	2089#	6682*											
P.A11 =	000054	2091#	6689*											
P.BAHI=	000007	2044#	4846											
P.BALO=	000010	2045#	5617*	5624*	5635*	6861	6958							
P.BAR =	000024	2079#	6405*	6635*										
P.B00 =	000042	2086#	6360*	6361	6697*	6905*								
P.B01 =	000046	2088#	6676*											
P.B10 =	000052	2090#	6683*											
P.B11 =	000056	2092#	6281*	6690*										
P.CMND=	000001	2038#	5140*	5603*	5616*	5620*	5623*	5636*	5642*	5647*	6176	6793	6797	6809
		6819	6825	6831	6834	6840	6865	6872	6891	6938	6955	6977	6986	
P.CS1 =	000016	2076#	4885	4964	6197*	6249*	6287*	6288*	6290*	6291	6400*	6402*	6629*	6632*
		6783	6837*	6838*	6840*	6844*	6845	6869*	6872*	6874*	6875*	6885*	6886	6894*
		6895*	6897*	6898	6900*	6901	6906	6945*	6946*	6948*	6951*	6952	6968*	6969*
		6971*	6974	6993*	6994	7003*	7004							
P.CSIH=	000007	2043#	5528*	5529*	5530*	5533*	5693	6287	6528	6837	6874	6894	6945	6959
		6968												
P.CS2 =	000020	2077#	6203	6403*	6633*	6791*	6816	6881*	6882	6893	6912	6916	6942*	6943
		6967												
P.CYLN=	000002	2039#	5626*	5637*	6285	6821	6863	6965						
P.DCYL=	000030	2081#	6407*	6641*										
P.DRVN=	000000	2037#	5514*	5519*	6173	6350	6526	6765						
P.DS =	000036	2084#	6410*	6637*	6918									
P.DTS =	000026	2080#	4897	6406*	6636*									
P.EPAT=	000062	2094#	6199*	6785										
P.EPOS=	000060	2093#	6200*											
P.ER =	000034	2083#	4912	6409*	6638*									
P.OFST=	000006	2042#	5643*	6827										
P.PRST=	000014	2047#	4587	4784	4788	4791	4794	4797	4800	4817	4974*	4988	5138	5521*
		5524*	5527*	6045*	6158*	6192*	6220*	6227	6231*	6328*	6351	6355*	6368*	6386*
		6396*	6422	6426*	6436*	6448*	6553*	6705*	6780*	6817*				

[illegible]



ש ו ר

[illegible]

[illegible]



[illegible]



[illegible]

CZR6RCO RK611/06 USR DEFINED  
CZR6RC.P11 02-DEC-77 11:07

MACY11 30(1046) 02-DEC-77 11:59 PAGE 180  
CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0177

.HEADE	1510#	1511		
.SETUP	1510#	2383		
.SWRHI	1510#	1522		
.SWRLO	1510#	1532#	1533	1534
.SCATC	1510#	1646		
.SCMTA	1510#	1659		
.SDB2D	1510#	7616		
.SPOWE	1510#	7789		
.SRAND	1510#	7723		
.SREAD	1510#	7360		
.SSAVE	1510#	7678		
.SSIZE	1510#	7759		
.STRAP	1510#	7832		
.STYPO	1510#	7216		
.STYPE	1510#	7146		
.STYPO	1510#	7283		

. ABS. 043561 000

ERRORS DETECTED: 0

RM03:CZR6RC,RM03:CZR6RC.SEQ/SOL/CRF/NL:TOC/DOC/EQ:QNEWSW=RM03:DRIV10,RM03:CZR6RC.P11

RUN-TIME: 30 23 1 SECONDS

RUN-TIME RATIO: 604/55=10.8

CORE USED: 42K (83 PAGES)

DOCUMENT PAGES: 177

EOF1CZR6RCSEQ

00010000

780223

PDP10 411