

DR11

DR11 GEN NPR INTFC
CZDRLCO

AH-E780C-MC
FICHE 1 OF 1

FEB 1981
COPYRIGHT © 77-80
MADE IN USA



.REM @

IDENTIFICATION

PRODUCT CODE: AC-E779C-MC
PRODUCT NAME: CZDRLCO DR11 GEN NPR INTFC
DATE RELEASED: OCTOBER, 1980
MAINTAINER: DIAGNOSTIC ENGINEERING
AUTHOR: DAN P. MILLEVILLE

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

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

COPYRIGHT (C) 1977, 1980 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL	PDP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	

34			HISTORY	
35				
36		REV	DATE	NOTE
37				
38		A	1977	INITIAL RELEASE
39		B	1980	CORRECTION OF CODING ERRORS
40		C	1980	11/05 PROBLEM AND ENDLESS LOOP PROBLEM FIXED

TABLE OF CONTENTS

41		
42		
43		
44	1.0	ABSTRACT
45		
46	2.0	REQUIREMENTS
47		
48		2.1 EQUIPMENT
49		2.2 HARDWARE SWITCH SETTINGS
50		2.3 STORAGE
51		
52	3.0	TESTING MODES
53		3.1 DEFINITION
54		3.2 IMPLEMENTATION
55		
56	4.0	LOAD AND START PROCEDURE
57		
58	5.0	SWITCH REGISTER
59		
60		5.1 OPTIONS
61		5.2 SOFTWARE SWITCH REGISTER
62		5.3 LOADING OF THE SOFTWARE SWITCH REGISTER
63		5.4 PROGRAM AND/OR OPERATOR ACTION
64		
65	6.0	ERROR REPORTING
66		
67	7.0	OPERATING MODES
68		
69		7.1 MANUAL MODE
70		7.1.1 EDIT FUNCTION
71		7.1.2 LIST FUNCTION
72		7.1.3 BURST CALIBRATION FUNCTION
73		7.1.4 RUN FUNCTION
74		
75		7.2 AUTO MODE
76		7.3 RESTART AFTER PREVIOUS RUN
77		7.4 TESTING UNDER APT
78		
79		
80	8.0	MISCELLANEOUS
81		
82		8.1 POWER FAIL
83		8.2 END-OF-PASS MESSAGE SPECIAL FEATURE
84		
85	9.0	EXECUTION TIMES

86	10.0	SUBROUTINE DESCRIPTIONS
87		
88	10.1	READ
89	10.2	ERCAPT
90	10.3	FTCAPT
91	10.4	FIXTBL
92	10.5	LODBUF
93	10.6	CHKBFF
94	10.7	INTA
95	10.8	DATCHK
96	10.9	CLENUP
97	10.10	CHKCAB
98	10.11	DATOCK
99	10.12	ERRCHK
100	10.13	DEVADS
101	10.14	BPINIT
102	10.15	DRGET
103	10.16	TYP CNF
104	10.17	CHK4DR
105	10.18	ASIZE
106	10.19	VCTADS
107	10.20	CATCH
108	10.21	PSTATE
109	10.22	PNTPRI
110	10.23	SETTUP
111		
112	11.0	DATA STACKS
113		
114	11.1	PATRNS
115	11.2	EXPATO
116	11.3	EXPAT1

117
118
119
120
121
122
123
124
125
126
127

1.0 ABSTRACT

THIS DIAGNOSTIC PROGRAM IS CAPABLE OF TESTING THE DR11-W
NPR GENERAL INTERFACE IN DR11-W OR DR11-B MODE.

IT HAS THE FOLLOWING FEATURES:

1. APT11/XXDP COMPATIBLE
2. MULTIPLE BOARD TESTING USING TABLE CREATED BY USER
3. BURST DATA LATE CALIBRATION
4. INDEPENDENT 'LOGIC WRAP-AROUND' AND 'CABLE WRAP-AROUND' TESTING

2.0 REQUIREMENTS

2.1 EQUIPMENT

1. PDP11 STANDARD COMPUTER
2. I/O TYPE TERMINAL
3. 1-16 DR11-W MODULE(S)
4. LOOP BACK CABLE (NEEDED TO FULLY CHECK THE MODULE WITH THIS DIAGNOSTIC)

2.2 HARDWARE SWITCH SETTINGS

THE ADDRESS SELECTION SWITCH, E120, IS SET UP AS BELOW:

	1	2	3	4	5	6	7	8	9	10
ADDRESS BITS:	12	11	10	9	8	7	6	5	4	3

EXAMPLE: DEVICE ADDRESS 172410, SWITCHES 1, 3, 5 & 10 SHOULD BE OFF, AND ALL OTHERS SHOULD BE ON.

THE E105 SWITCHPACK: THIS SWITCHPACK MUST BE IN THE FOLLOWING POSITIONS TO RUN THIS DIAGNOSTIC:

- 1 - OFF
- 2 - ON
- 3 - OFF
- 4 - OFF
- 5 - ON FOR -W MODE, OFF FOR -B MODE

SINGLE SWITCH NEAR THE E105 SWITCHPACK:

- 2 CYCLE MODE - SWITCH HANDLE TOWARDS PACK E105
- N CYCLE MODE - SWITCH HANDLE TOWARDS E94

THE VECTOR SELECTION SWITCH, E15, IS SET UP AS BELOW:

	1	2	3	4	5	6	7	8
VECTOR BITS:	1	2	3	4	5	6	7	8

EXAMPLE: VECTOR ADDRESS 300, SWITCHES 6 & 7 SHOULD BE OFF, AND ALL OTHERS SHOULD BE ON.

2.3 STORAGE

THE PROGRAM USES APPROX. 52200 WORDS OF MEMORY

3.0 TESTING MODE

3.1 DEFINITION

THE DR11-W DIAGNOSTIC ACCOMPLISHES DEVICE REGISTER BIT TESTS, INTERNAL 'LOGIC' WRAP-AROUND TESTS, AND WITH THE BC06-R WRAP-AROUND CABLE IN J1 AND J2, PROVIDES EXTERNAL 'CABLE' WRAP-AROUND TESTS. IN ORDER TO FULLY CHECK THE MODULE, THE DIAGNOSTIC MUST BE RUN WITH AND WITHOUT THE WRAPAROUND CABLE IN PLACE, RESTARTING AT ADDRESS 200 EACH TIME, OR EDITING TO CHANGE THE CABLE MODE (SEE SECT. 7.1.1)

THERE ARE ONLY TWO LEGAL MODES OF OPERATION OF THIS DIAGNOSTIC:

1. DR11 WITH NO CABLE(S) IN USER SLOTS.
2. DR11 WITH WRAP-AROUND CABLE FROM J1 TO J2.

THIS DIAGNOSTIC IS NOT MEANT TO BE RUN IN THE FOLLOWING MODES:

1. DR11 CONNECTED TO ANOTHER DR11.
2. DR11 CONNECTED TO A USER DEVICE.

3.2 IMPLEMENTATION

DEVICE REGISTER BIT TESTS AND INTERNAL LOGIC WRAP-AROUND TESTS ARE EXECUTED UNCONDITIONALLY. CABLE WRAP-AROUND TESTS ARE EXECUTED ONLY IF THE BC06-R CABLE IS IN PLACE BETWEEN THE J1 AND J2 CONNECTORS ON THE DR11-W UNDER TEST. THE PRESENCE OF THIS CABLE IS 'SIZED' FOR AUTOMATICALLY FOR EACH BOARD WHEN THE DIAGNOSTIC IS STARTED AT ADDRESS 200. THE USER *MUST* VERIFY THAT THE 'SIZING' OCCURRED CORRECTLY BY OBSERVING THE OUTPUT OF THE PROGRAM WHEN STARTING AT 200. (REFER TO SECTION 5.1 FOR EXAMPLE) IF THIS SUMMARY IS NOT NEEDED, RAISE BIT 12 OF THE SWITCH REGISTER BEFORE PROGRAM EXECUTION.

IN MANUAL MODE (STARTING ADDRESS = 204), THE USER CAN FORCE UNIFORM TESTING PARAMETERS FOR ALL MODULES THROUGH USE OF THE EDIT FUNCTION (REFER TO SECTION 7.1.1).

215
216
217
218
219
220
221

4.0 LOAD AND START PROCEDURE

1. LOAD PROGRAM INTO MEMORY.
2. LOAD STARTING ADDRESS 200, 204 OR 210. (SEE SECTS. 7.1, 7.2, 7.3
RESPECTIVELY)
3. PRESS START.

222 5.0 SWITCH REGISTER
223
224 5.1 OPTIONS
225
226 SWITCH OCTAL FUNCTION
227
228 SW15=1 100000 HALT ON ERROR
229
230 THIS WILL CAUSE THE PROCESSOR TO HALT AT THE
231 NEXT ERROR.
232
233 SW14=1 040000 LOOP ON TEST
234
235 THIS WILL CAUSE THE PROCESSOR TO LOOP ON THE
236 TEST IT IS THEN EXECUTING.
237
238 SW13=1 020000 INHIBIT ERROR TYPEOUTS
239
240 THIS WILL CAUSE ERROR TYPEOUTS TO BE INHIBITED.
241
242 SW12=1 010000 DO NOT PRINT BOARD CONFIGURATION
243
244 THIS WILL CAUSE THE LIST OF ALL BOARDS AND
245 THEIR SETUP DATA THAT THE AUTOSIZE ROUTINE
246 FOUND TO NOT PRINT.
247
248 SW11=1 004000 TEST NUMBER TRACE ENABLING
249
250 THIS ENABLES THE PRINTING OF THE FOLLOWING AT
251 THE BEGINNING OF EACH TEST:
252
253 T # XX
254
255 THIS CAN BE USED WHEN AN UNEXPECTED TRAP OCCURS
256 IN A TEST, BUT LOOPING ON THAT TEST RESULTS IN
257 NO ERROR(S).
258
259 SW10=1 002000 BELL ON ERROR
260
261 THIS FUNCTION CAUSES THE TERMINAL BELL TO SOUND
262 WHEN AN ERROR OCCURS. THIS CAN BE USED IN CON-
263 JUNCTION WITH LOOP-ON-TEST AND INHIBIT-ERROR-
264 TYPEOUTS TO SEE IF A LOOSE CONNECTION MAY BE
265 CAUSING THE ERROR.
266
267 SW09=1 001000 LOOP ON ERROR
268
269 THIS FUNCTION WILL CAUSE LOOPING ON ERROR. IT
270 CAN BE USED IN CONJUNCTION WITH INHIBIT-ERROR-
271 TYPEOUTS WHEN USING A SCOPE TO FIND A FAULTY
272 COMPONENT.

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

SW08=1

000400

LOOP ON TEST IN SWR<6:0>

THIS FUNCTION CAUSES THE CPU TO JUMP TO THE TEST IN BITS <6:0> AND EXECUTE THAT TEST UNCONDITIONALLY. CHANGE THE SWITCH REGISTER TO EXIT. TO CREATE A TIGHTER LOOP ON THAT PARTICULAR TEST, SET LOOP-ON-TEST (40000) IN THE SWR ONCE THE TEST IS EXECUTING.

SW07=1

000200

INHIBIT MULTIPLE ERROR TYPEOUTS

ON ERROR CALLS IN LOOPS WHERE MULTIPLE ERRORS ARE POSSIBLE, THIS FUNCTION INHIBITS ANY ADDITIONAL DATA THAT MAY PRINT IN THAT LOOP.
EXAMPLE:

MULTIPLE TYPEOUTS ENABLED:

[ERROR MESSAGE]

[DATA HEADER]

XXXXXX	XXXXXX	XXXXXX	XXXXXX
XXXXXX	XXXXXX	XXXXXX	XXXXXX
XXXXXX	XXXXXX	XXXXXX	XXXXXX
XXXXXX	XXXXXX	XXXXXX	XXXXXX
XXXXXX	XXXXXX	XXXXXX	XXXXXX

>>>>>NOTE<<<<<<

A MAXIMUM OF 17 (OCTAL) DATA LINES WILL PRINT. IF THERE ARE MORE, A MESSAGE WILL PRINT AS FOLLOWS:

THERE ARE STILL MORE ERRORS, BUT WILL NOT BE PRINTED.
ERRORS WILL STILL BE COUNTED AND PRINTED AT THE EOP.

MULTIPLE TYPEOUTS DISABLED:

[ERROR MESSAGE]

[DATA HEADER]

XXXXXX XXXXXX XXXXXX XXXXXX

(NO MORE DATA WILL PRINT) THE TOTAL NUMBER OF ERRORS WILL STILL BE TOTALED AND PRINTED AT THE EOP OR EOD.

5.2 SOFTWARE SWITCH REGISTER

IF THE HARDWARE SWITCH REGISTER DOES NOT EXIST, OR IF ONE DOES AND IT CONTAINS '-1' (177777) THEN THE SOFTWARE SWITCH REGISTER (LOCATION 176) IS USED, WHICH ALLOWS THE USER THE SAME SWITCH OPTIONS AS THE HARDWARE SWITCH REGISTER.

5.3 LOADING THE SOFTWARE SWITCH REGISTER

THIS PROGRAM SUPPORTS THE DYNAMIC LOADING OF THE SOFTWARE SWITCH REGISTER (LOCATION 176) FROM THE TTY. THIS IS ACCOMPLISHED AS FOLLOWS:

1. TYPE CONTROL G <^G> REPEATEDLY, AS RESETS AND INITs DONE IN THE DIAGNOSTIC MAY CLEAR THE CHARACTER BEFORE THE CHARACTER IS RECOGNIZED. ONCE INPUT IS RECOGNIZED, THIS ALLOWS THE TTY TO ENTER DATA INTO LOCATION 176 AT THE END OF A TEST.
2. THE MACHINE WILL TYPE: SWR=XXXXXX NEW= (XXXXXX IS THE OCTAL CONTENTS OF THE SOFTWARE SWITCH REGISTER)
3. AFTER THE NEW=' THE OPERATOR CAN DO ONE OF THE FOLLOWING:
 - A. TYPE A NUMBER TO BE LOADED INTO LOCATION 176 FOLLOWED BY A <CR> (ONLY NUMBERS BETWEEN 0-7 WILL BE ACCEPTED AND ONLY 6 NUMBERS WILL BE ALLOWED). IF A <CR> IS THE FIRST ENTRY THE SOFTWARE SWITCH REGISTER WILL NOT BE CHANGED.
 - B. IF A CONTROL U <^U> IS DEPRESSED, THE PROGRAM WILL GO BACK TO STEP 2.

5.4 PROGRAM AND/OR OPERATOR ACTION

LOADING AND STARTING AT 200 WITH ALL SWITCHES DOWN IS NORMAL LOGIC TESTING. IF AN ERROR IS DETECTED, THERE WILL BE A PRINTOUT. WHEN AN ERROR IS DETECTED AND IT IS NECESSARY TO SCOPE ON IT, PLACE 100000 (BIT 15) IN THE SWITCH REGISTER TO HALT ON ERROR. AFTER HALTING AT THE ERROR TO BE LOOPED ON, ENTER 60000, LOOP-ON-ERROR AND INHIBIT PRINTOUTS. IF THERE IS MORE THAN ONE ERROR CALLED IN A TEST, AND YOU WISH TO LOOP ON OTHER THAN THE 1ST ERROR, YOU MUST CORRECT THE CONDITION CAUSING THE PREVIOUS ERROR(S) BEFORE YOU CAN LOOP ON THAT ERROR. NOP'ING THE PREVIOUS ERRORS WILL PRODUCE UNPREDICTABLE RESULTS FOR ANY SUBSEQUENT ERRORS IN THE TEST.

364
365
366
367
368
369
370
371

6.0 ERROR REPORTING

EACH TEST WILL CALL AN ERROR CONTAINING THE TEST NUMBER, ERROR PC
AND DATA THAT IS SIGNIFICANT TO THE PROBLEM THAT CAUSED THE ERROR.

IN THE CASE OF MULTIPLE BOARD TESTING, THE FAILING MODULE IS IDENTIFIED
BY THE DEVICE REGISTER ADDRESS, AND THE END-OF-DEVICE-TEST MESSAGE
FOLLOWING ALL ERRORS FOR THAT PARTICULAR MODULE.

7.0 OPERATING MODE

7.1 MANUAL MODE (STARTING ADDRESS = 204)

DEFINED AS NON-AUTOMATIC USE OF THE DIAGNOSTIC.

THIS MODE IS INTENDED FOR USE IN MANUFACTURING WHEN APT IS NOT AVAILABLE.

IN MANUAL MODE, ALL DR11-W HARDWARE MODULES *MUST*BE*CONFIGURED*
*AS*FOLLOWS*:

- > W/B, PRIORITY LEVEL, 2/N CYCLE AND CABLE STATES SET IDENTICAL
*IN*ALL*MODULES*.
- > ALL DEVICE ADDRESSES MUST BE SET IN A SERIES SPACED 10 LOCATIONS
APART, STARTING WITH THE ADDRESS INPUTED TO THE PROMPT 'STARTING
DEVICE ADDRESS XXXXXX :'. (ALL MODULES MUST BE ADDRESSED
WITHIN THE LEGAL ADDRESS RANGE OF 171000 TO 177000)
- > ALL VECTOR ADDRESSES MUST BE SET IN A SERIES SPACED 10 LOCATIONS
APART. (ALL MODULES MUST BE VECTORED WITHIN THE LEGAL VECTOR
RANGE OF 300 TO 770)
- > THE MODULE WITH THE LOWEST DEVICE ADDRESS MUST ALSO HAVE THE
LOWEST VECTOR ADDRESS, THE MODULE WITH THE NEXT TO THE LOWEST
DEVICE ADDRESS MUST ALSO HAVE THE NEXT TO THE LOWEST VECTOR
ADDRESS, ETC. FOR EXAMPLE:

BOARD #	DEVICE ADDRESS	VECTOR ADDRESS
0	172410	300
1	172420	310
2	172430	320
3	172440	330

ETC.

ONLY UNDER MANUAL MODE DOES THE DIAGNOSTIC OFFER 'BURST
DATA LATE' CALIBRATION. AFTER LOADING PROGRAM, DEPOSITING
SA 204, AND PRESSING START, THE PROGRAM TYPES THE FOLLOWING:

MULTIPLE BOARD DIALOGUE

ENTER COMMAND ([E]DIT, [L]IST, [B]URST CALIBRATION, [R]UN):

THE PROGRAM WILL ALLOW ONLY 1 CHARACTER INPUT, AUTOMATICALLY
PRINTING A <CRLF> WHEN THE CHARACTER IS INPUTED.

7.1.1 WHEN [E] IS ENTERED, THE PROGRAM ENTERS THE EDIT FUNCTION

NOTE: TO EXIT THIS ROUTINE AT ANY RESPONSE AND RETURN TO THE MBD PROMPT, ENTER CONTROL 'C' (^C). THIS DOES NOTHING BUT EXIT THE ROUTINE, AND DOES NOT CHANGE ANY VALUES PRESENT OR CHANGED. TO RETURN TO THE PREVIOUS PROMPT, TYPE <ESC>.

'EDIT' RESPONDS FIRST BY PRINTING:

OF BOARDS UNDER TEST X:

PROGRAM ACCEPTS A MAXIMUM OF 2 DECIMAL CHARACTERS. AN APPROPRIATE ERROR MESSAGE IS PRINTED IF THE NUMBER INPUTED IS OUT OF RANGE, OR AN ILLEGAL CHARACTER WAS INPUTED. ENTER <CR> IF PRESENT VALUE IS OK. NEXT:

STARTING DEVICE ADDRESS XXXXXX :

THE USER SHOULD RESPOND WITH THE LOWEST DEVICE ADDRESS IN THE SERIES. PROGRAM ACCEPTS A MAXIMUM OF 6 OCTAL DIGITS BETWEEN 171000 AND 177000. AN APPROPRIATE ERROR MESSAGE IS PRINTED IF THE NUMBER INPUTED IS OUT OF RANGE, OR AN ILLEGAL CHARACTER WAS INPUTED. ENTER <CR> IF PRESENT VALUE IS OK. NEXT:

STARTING VECTOR ADDRESS XXX :

THE USER SHOULD RESPOND WITH THE LOWEST VECTOR ADDRESS IN THE SERIES. PROGRAM ACCEPTS A MAXIMUM OF 3 OCTAL DIGITS BETWEEN 300 AND 777. AN APPROPRIATE ERROR MESSAGE IS PRINTED IF THE NUMBER INPUTED IS OUT OF RANGE, OR AN ILLEGAL CHARACTER WAS INPUTED. ENTER <CR> IF PRESENT VALUE IS OK. NEXT:

DR11-W OR B (W=0) CURRENT STATE = X :

PROGRAM ACCEPTS EITHER A 0 OR 1, REPEATING THE PROMPT IF ANY OTHER CHARACTER IS INPUTED. ENTER <CR> IF PRESENT VALUE IS OK. NEXT:

DEVICE PRIORITY PRESENT LEVEL = X :

PROGRAM ACCEPTS 1 CHARACTER BETWEEN 0 AND 7, REPEATING THE PROMPT IF ANOTHER CHARACTER IS INPUTED. ENTER <CR> IF PRESENT VALUE IS OK. NEXT:

2 OR N CYCLE BURST (2 CY=0) PRESENT STATE = X .

PROGRAM ACCEPTS A 0 OR 1, REPEATING THE PROMPT IF ANY OTHER CHARACTER IS INPUTED. ENTER <CR> IF PRESENT VALUE IS OK. NEXT:

DO CABLE TESTS (NO=0) PRESENT STATE = X :

PROGRAM ACCEPTS A 0 OR 1, REPEATING THE PROMPT IF ANY OTHER CHARACTER IS INPUTED. ENTER <CR> IF PRESENT VALUE IS OK. THEN THE COMMAND PROMPT IS REPRINTED.

7.1.2 WHEN [L] IS ENTERED, THE PROGRAM ENTERS THE LIST FUNCTION

THE DIAGNOSTIC THEN PRINTS THE FOLLOWING:

# OF BOARDS	START REGADR	VECADR	W-B	P-LEV	2-N CYCLE	CABLE TESTS
XX	XXXXXX	XXX	X	X	X	X

AS PREVIOUSLY MENTIONED, ALL BOARDS MUST BE SPACED 10 ADDRESS LOCATIONS APART STARTING WITH THE 'REGADR' VALUE ABOVE, AND VECTORS SPACED 10 ADDRESS LOCATIONS APART STARTING WITH THE 'VECADR' VALUE ABOVE. THE EXPECTED W-B, PRIORITY LEVEL, 2-N CYCLE AND CABLE TEST STATES WILL BE THE SAME FOR ALL MODULES.

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

7.1.3 WHEN [B] IS ENTERED, THE PROGRAM ENTERS THE BURST DATA LATE CALIBRATION ROUTINE, AND THE FOLLOWING IS TYPED:

BURST DATA LATE CALIBRATION IN PROGRESS..
ATTACH SCOPE PROBE..
TO CALIBRATE NEXT BOARD, TYPE ANY CHARACTER
DEVICE # 0 UNDER CALIBRATION

THIS ROUTINE WILL NOT EXECUTE IF YOU HAVE NOT USED EDIT TO DEPOSIT A LEGAL STARTING ADDRESS AND VECTOR ADDRESS, OR THE PROGRAM HAS ALREADY BEEN STARTED AT 200. THE MULTIPLE BOARD DIALOGUE (MBD) PROMPT WILL BE RETURNED IF THIS IS THE CASE. AS STATED IN THE DR11 ENGINEERING SPECIFICATION, THE 'BURST DLT' MULTIVIBRATOR TIME OUT MUST BE CALIBRATED SO AS TO BE COMPATIBLE WITH THE USER DEFINED TRANSFER RATE IN BURST MODE OPERATION. THE PROGRAM SOFTWARE ROUTINE SETS THE CYCLE BIT IN THE CSR OF THE DR11, A SHORT DELAY IS EXECUTED, AND THEN THE CYCLE BIT IS CLEARED. THE DIAGNOSTIC THEN TESTS FOR ANY CHARACTER WAITING, INDICATING THE USER WISHES TO GO ON TO THE NEXT BOARD. IF NONE, IT RE-EXECUTES THE SETTING AND CLEARING OF THE CYCLE BIT. IF A CHARACTER WAS INPUTED, IT CHECKS FOR THE NEXT BOARD, AND IF ANY, SETS UP THE ADDRESSES FOR THAT MODULE, THEN PRINTS THE FOLLOWING:

DEVICE # X UNDER CALIBRATION

'X' BEING THE DEVICE NUMBER. IT THEN REACCOMPLISHES THE SETTING AND CLEARING OF THE CYCLE BIT FOR THAT DEVICE. IF NO FURTHER MODULES ARE FOUND, THE MESSAGE:

BURST CALIBRATION COMPLETE

IS ISSUED, AND THE MBD PROMPT IS THEN RETURNED FOR ANOTHER COMMAND. TO ACCOMPLISH THE BURST DATA LATE CALIBRATION, ATTACH A SCOPE PROBE TO E83-7 ON THE DR11-W (REFER TO PRINT SET M8716-0-1). A POSITIVE PULSE WILL BE OBSERVED. THE PULSE SHOULD BE SET BETWEEN 3-30 US. BY ADJUSTING POT. R80.

7.1.4 WHEN [R] IS ENTERED, THE PROGRAM BEGINS DIAGNOSTIC TEST EXECUTION. THIS WILL BE BLOCKED IF LEGAL STARTING DEVICE ADDRESSES AND VECTOR ADDRESSES HAVE NOT BEEN SET UP. IF THEY ARE, THE REGISTER AND VECTOR TABLES ARE FILLED, AND NORMAL START IS EXECUTED.

7.2 AUTO-SIZE MODE (STARTING ADDRESS = 200)

THIS MODE IS THE NORMAL FIELD SERVICE MODE. IT SUPPORTS STANDALONE OPERATION AS WELL AS SCRIPT OPERATION UNDER ACT11 OR XXDP (CHAIN).

THE DR11 DIAGNOSTIC HAS THE FOLLOWING RUN CHARACTERISTICS WHEN OPERATING IN AUTO MODE:

- A. THE PROGRAM WILL TEST THE BOARDS RECOGNIZED BY THE AUTOSIZE ROUTINE. THE AUTOSIZE ROUTINE WILL LOOK AT ADDRESSES BETWEEN 172414 AND 172604 IN STEPS OF 10 (20 OCTAL LOCATIONS). IT WILL INITIALLY DETERMINE IF THE LOCATION IT FOUND TO EXIST IS A DR11 BY FORCING AN INTERRUPT. IF THE BOARD FAILS TO INTERRUPT, YOU MUST USE MANUAL MODE TO FORCE TEST EXECUTION OF THAT MODULE. THE PURPOSE OF THIS INITIAL TEST IS TO ELIMINATE TESTING A MODULE THAT IS NOT A DR11, AND DETERMINE THE INTERRUPT PRIORITY AND VECTOR OF THAT MODULE. THE ONLY LEGAL INTERRUPT VECTORS THE DR11 CAN BE SET UP FOR ARE AS FOLLOWS: 40, 50-174, AND 254-774, ALL IN STEPS OF 4. EACH BOARD CAN HAVE A VECTOR ANYWHERE IN THE STATED RANGES WITH NO RESTRICTIONS, ALLOWING COMPLETE FLEXIBILITY IN THE TEST SEQUENCE.

IN THE CASE OF MULTIPLE DR11-W'S ON THE SAME CPU, EACH DR11-W MUST HAVE ITS OWN UNIQUE DEVICE/VECTOR ADDRESSES. THERE ARE NO CONSTRAINTS THAT THE BOARDS MUST START WITH THE FIRST DEVICE ADDRESS 172410, OR THAT MULTIPLE BOARDS ARE ASSIGNED CONSECUTIVE DEVICE ADDRESSES. WHEN OPERATING IN AUTO-SIZE MODE, THE USER SHOULD VERIFY THE 'SIZED' CONFIGURATION BY KNOWING HOW THE BOARDS ARE SET UP AND COMPARING WITH THE AUTOSIZE OUTPUT WHEN STARTING AT 200.

AUTO-SIZING WILL DETERMINE THE INTERRUPT PRIORITY, INTERRUPT VECTOR, W/B, 2/N CYCLE, AND CABLE STATES OF EACH BOARD, INDEPENDENT OF THE STATES OF OTHER BOARDS.

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

B. THE FOLLOWING WILL NOT BE OFFERED TO THE USER IN AUTOSIZE MODE:

1. BURST DATA LATE CALIBRATION
2. MULTIPLE BOARD DIALOGUE

THE DIAGNOSTIC WILL PRINT THE FOLLOWING:

DIAGNOSTIC HAS DETERMINED THE FOLLOWING ABOUT THE DR11-W(S) IT HAS FOUND. USER *MUST* DETERMINE ACCURACY

BOARD#	REGADR	VECADR	W/B	P-LEV	2-N CY	CABLE
X	XXXXXX	XXX	X	X	X	X

DATA WILL CONTINUE TO PRINT UNTIL DATA FOR ALL MODULES HAS BEEN PRINTED.

(^X) INHIBITS EOP'S, (^Y) FOR ERROR SUMMARY
UNIBUS HANG? RESTART AT ADDRESS XXXXXX

CZDRLCO DR11 GEN NPR INTFC LOGIC TEST

THE CONTROL X (^X) FEATURE BYPASSES THE SECTIONS THAT PRINT THE END-OF-PASS AND END-OF-DEVICE MESSAGES. THIS IS TO IMPROVE THE NUMBER OF PASSES EXECUTED OVER ANY PERIOD OF TIME, AS WELL AS MAKE OVERNIGHT OR WEEKEND RUNS USE LESS PAPER. ERROR TYPEOUTS ARE NOT DISABLED IN THIS MODE. WHEN AN ERROR OCCURS, THE END-OF-PASS (EOP) WILL PRINT FOR THAT PASS, AND, IF MORE THAN ONE MODULE IS BEING TESTED, AN END-OF-DEVICE (EOD) AS WELL AS END-OF-PASS WILL PRINT SO YOU WILL KNOW WHICH DEVICE AND PASS WAS EXECUTING WHEN THE ERROR OCCURED. IN ORDER TO GET A PROGRESS REPORT, HIT ANY KEY REPEATEDLY, SINCE INITS AND RESETS DONE DURING THE EXECUTION OF THE DIAGNOSTIC MAY CLEAR THE CHARACTER WAITING FLAG BEFORE THE CHECK FOR THIS BIT. WHEN THE CHARACTER IS RECOGNIZED, AN EOP, AND IF MORE THAN ONE MODULE, AN EOD MESSAGE WILL PRINT GIVING THE USER A PROGRESS REPORT. TO DISABLE THIS FEATURE, REPEATEDLY ENTER (^X) AGAIN UNTIL THE CPU RECOGNIZES YOUR INPUT.

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

THE CONTROL Y (^Y) FUNCTION CALLS FOR A SUMMARY OF DEVICE(S) AND PASS(ES) THAT HAD ERRORS. IF NO ERRORS OCCURED SINCE THE BEGINNING OF THE DIAGNOSTIC, OR SINCE THE LAST ERROR REPORT, THE FOLLOWING IS PRINTED:

NO ERROR TOTALS TO REPORT

IF THERE WERE ERRORS, THE FOLLOWING IS PRINTED:

SUMMATION OF ERRORS SINCE BEGINNING OR LAST REPORT

BOARD #	PASS #	ERRITL
X	X	X
X	X	X
X	X	X (ETC.)

THE INFORMATION IS STORED ON A STACK THAT WILL HOLD UP TO 150 (DECIMAL) DEVICE-PASS ERROR DATA LINES ABOVE. IF THE LIMIT IS REACHED, DIAGNOSTIC WILL CONTINUE, BUT FURTHER DATA WILL NOT BE STORED. THE DATA ACCUMULATED IS NOT WRITTEN OVER, BUT WHEN (^Y) IS ENTERED, THE FOLLOWING IS PRINTED JUST BEFORE THE 'SUMMATION...' STATEMENT ABOVE:

STACK IS FULL - DATA MAY HAVE BEEN LOST

WHEN THE DATA IS PRINTED, THE STACK IS REINITIALIZED AND WILL START STORING UP TO ANOTHER 150 ERROR DATA LINES.

IN THE EVENT THE UNIBUS BECOMES HUNG, AND YOU HAVE NON-VOLATILE MEMORY OR BATTERY BACKUP, RESTART THE PROGRAM AT THE ADDRESS SPECIFIED BY THE 'UNIBUS HUNG...' PROMPT AT THE START OF THE DIAGNOSTIC. THE PRINTOUT WILL BE AS FOLLOWS:

DEVICE ADDRESS - XXXXXX, TEST NUMBER - XXXXXX, PASS NUMBER - XXXXXX

CPU WILL HALT. HITTING CONTINUE WILL CAUSE THE PROGRAM TO RESTART AS THOUGH YOU HAD STARTED AT 200.

646
647
648
649
650
651
652
653
654
655
656
657
658

7.3 RESTARTING PROGRAM IN MEMORY (STARTING ADDRESS = 210)

WHENEVER THE PROGRAM IS HALTED, ALL HISTORY OF PREVIOUS TESTING IS SAVED. IT WILL REMAIN INTACT UNTIL:

1. ANOTHER PROGRAM IS LOADED INTO MEMORY
2. THE USER RE-EDITS THE TABLE

TO RESTART THE PROGRAM, ENTER SA 210 AND START. THIS START PRECLUDES ANY SETUP AND NEGATES THE START MESSAGE OBTAINED WHEN STARTING AT 200. DO NOT START AT THIS LOCATION IF THE DIAGNOSTIC HAS NOT BEEN PREVIOUSLY "STARTED" AT EITHER 200 OR 204.

7.4 TESTING UNDER APT (AUTOMATED PRODUCT TESTING)

TO SET UP FOR MULTIPLE BOARDS FOR TESTING UNDER APT CONTROL,
THE APT SYSTEM MANAGER SHOULD ANSWER THE APT QUERIES TO THE
FOLLOWING ITEMS AS INDICATED BELOW:

SOFTWARE ENVIRONMENT: 000 - DUMP MODE
001 - SCRIPT MODE (APT MONITORS
DIAGNOSTIC)

ENVIRONMENT MODE (\$ENVN): 000 - LET DIAGNOSTIC AUTO-SIZE
CONFIGURATOR AND TEST
ACCORDINGLY

200 - DIAGNOSTIC MUST USE CONFIGURA-
TION SPECIFIED BY APT (\$VECT1,
\$VASE, \$DEVN, \$DDWX)

VECTOR ADDRESS (\$VFCT1): 300

DEVICE ADDRESS (\$BASE): 172410

DEVICE MAP (\$DEVN): XXXXXX - EACH SET BIT INDICATES THAT
BOARD IS PRESENT AND SHOULD
BE TESTED. EXAMPLES:

BIT 0 = BOARD #0 (DEVICE ADR = 172410, VEC ADR = 300)
BIT 1 = BOARD #1 (DEVICE ADR = 172420, VEC ADR = 310)
BIT 2 = BOARD #2 (DEVICE ADR = 172430, VEC ADR = 320)

BIT 15 = BOARD #15 (DEVICE ADR = 172600, VEC ADR = 470)

DEVICE DESCRIPTOR WORDS: XXXXXX - THERE IS 1 DESCRIPTOR WORD
FOR EACH DEVICE:

\$DDW0 IS FOR DEVICE 0
\$DDW1 IS FOR DEVICE 1, ETC.

EACH DESCRIPTOR WORD MUST BE
SET UP AS FOLLOWS:

BIT 0 - DR11-W OR -B MODE
(W=0, B=1)

BIT 1 - 2/N CYCLE
(0=2 CY, 1=N CY)

BIT 2 - CABLE TESTS
(0=NO, 1=YES)

BIT 5 \
BIT 6 > DEVICE PRIORITY
BIT 7 /

711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727

8.0 MISCELLANEOUS

8.1 POWER FAIL

IF A POWER FAILURE OCCURS AND BATTERY BACKUP MAINTAINS
THE PROGRAM IN MEMORY, OR A NON-VOLATILE MEMORY EXISTS,
THE PROGRAM WILL RESTART PRINTING THE FOLLOWING:

POWER FAILURE - RESTARTING PROGRAM

THE DIAGNOSIC WILL THEN RESTART AT ADDRESS 210.

IF CPU IS TURNED OFF WHILE RUNNING, THE ABOVE PROCEDURE
IS FOLLOWED. IF THE PROCESSOR IS HALTED FIRST, THEN
TURNED OFF, THE PROCESSOR WILL COME BACK UP HALTED. TO
RESTART THE PROGRAM, HIT CONTINUE, AND THE REMAINING
PROCEDURE IS THE SAME AS ABOVE.

728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746

8.2 END-OF-PASS MESSAGE

THE EOP WILL PRINT AS FOLLOWS WITH NO ERRORS ON THAT PASS:

END PASS # XXXXXX

THE EOP WILL PRINT AS FOLLOWS WITH SOME ERRORS WHEN TESTING 1
DEVICE:

END PASS # XXXXXX TOTAL ERRORS SINCE LAST REPORT XXXXXX

THE EOP WILL PRINT THE SAME AS WITH NO ERRORS ON ANY PARTICULAR
PASS WHEN TESTING MORE THAN ONE DEVICE AND ONE OR MORE DEVICES
HAS FAILED, SINCE 'TOTAL ERRORS' IS MEANINGLESS AND WILL MORE
THAN LIKELY BE INCORRECT.

THE PASS NUMBER IS CAPABLE OF GOING UP TO 99,999,999 DECIMAL,
OR ABOUT 3 MONTHS RUNNING WITH EOP DISABLED AND NO ERRORS. IN
OTHER WORDS, 32767 IS NOT THE LIMIT AS WITH OTHER DIAGNOSTICS.

747
748
749
750
751
752

9.0 EXECUTION TIME

ON A PDP11/44:

IN ALL MODES: APPROXIMATELY 8 PASSES PER SECOND WITH EOP MESSAGES
DISABLED AND NO ERRORS.

10.0 SUBROUTINE ABSTRACT

10.1 READ

THE READ SUBROUTINE IS USED IN THE EDIT ROUTINE TO INPUT UP TO 6 DIGITS IN OCTAL, 2 DIGITS IN DECIMAL, OR A SINGLE NON-NUMERIC CHARACTER. R4 IS USED AS THE LOCATION TO HOLD THE NUMBER IN OCTAL, AND IS CLEARED FOR THAT PURPOSE AT THE START OF THE SUBROUTINE. R3 IS TO BE PRELOADED WITH THE NUMBER OF DIGITS EXPECTED, SINCE A <CRLF> IS PRINTED WHEN THE LIMIT IS REACHED. ENTERING A <CRLF> BEFORE THE LIMIT IS ACCEPTABLE, AS IT WILL BE INTERPRETED AS A NON-NUMERIC CHARACTER AND EXIT. IN ANY CASE, THE LAST INPUTED ASCII CHARACTER IS LEFT IN LOCATION 'ANSWER'. IF A NUMERIC CHARACTER IS INPUTED, IT WILL CLEAR ALL BUT THE 1ST 4 BITS IN LOCATION 'ANSWER', EXPOSING THE VALUE OF THE DIGIT INPUTED, ROTATE R4 TO THE LEFT 3 PLACES TO MAKE ROOM FOR THE INPUTED DIGIT, AND ADD IT TO R4. LOCATION 'LRGSTC' IS TO BE LOADED WITH THE LARGEST ASCII NUMBER DIGIT ACCEPTABLE FOR THIS NUMBER, I.E. 7 OR 9 (FOR OCTAL OR DECIMAL INPUT RESPECTIVELY). ANY CHARACTER OUTSIDE ASCII '0' OR '7/9' IS TREATED AS A NON-NUMERIC, TRIGGERING AN AUTOMATIC <CRLF> AND EXIT.

10.2 ERCAPT

THIS SUBROUTINE SAVES THE UNIT NUMBER, PASS NUMBER AND TOTAL ERRORS FOR THAT DEVICE/PASS WHENEVER IT ENCOUNTERED ERRORS. THIS ROUTINE SAVES DATA FOR 150 (DECIMAL) PASSES. IF THE STACK SHOULD BECOME FULL, DATA STARTING WITH THE 151ST PASS CONTAINING ERRORS IS LOST.

10.3 PTCAPT

THIS SUBROUTINE PRINTS THE DATA STORED BY THE ERCAPT SUBROUTINE AND RESETS THE SPECIAL STACK POINTER. IF NO DATA WAS STORED, A MESSAGE STATING NO DATA WAS STORED IS PRINTED. IF THE STACK IS FOUND FULL, A MESSAGE ANNOUNCING THIS FINDING IS PRINTED, WARNING THAT DATA MAY HAVE BEEN LOST.

10.4 FIXTBL

THIS SUBROUTINE FILLS THE 17 OCTAL LOCATIONS STARTING AT 'REGADR' AND 'VECADR' FROM THE STARTING VALUES ALREADY LOADED IN THE FIRST LOCATIONS IN STEPS OF 10 FOR EACH TABLE.

10.5 LODBUF

THE INBUF BUFFER IS LOADED WITH AN INCREMENTING PATTERN (0,1,2,3,...) BEGINNING AT THE STARTING ADDRESS OF INBUF. THE NUMBER OF WORDS LOADED IS DETERMINED BY THE CONTENTS OF BUFLN.

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

10.6 CHKBFF

THE CHKBUFF BUFFER IS LOADED WITH A MODIFIED INCREMENTING PATTERN (0,0,2,2,4,4,6,6,...) BEGINNING AT THE STARTING ADDRESS OF CHKBUFF. THE NUMBER OF WORDS LOADED IS DETERMINED BY THE CONTENTS OF BUFLN. THIS BUFFER IS LOADED ONLY FOR TESTS WHICH USE THE MAINTENANCE MODE OF THE DR11-W WHICH HAS A SPECIAL ALTERNATING DATI-DATO SEQUENCE OF OPERATION.

10.7 INTA

THE IE BIT IS CLEARED IN THE CSR THEN THE CSR IS CHECKED FOR THE ABSENCE OF THE ERROR BIT AND THE PRESENCE OF READY. THE WCR IS CHECKED TO SEE THAT IT IS EQUAL TO ZERO. THE CORRECT CONTENTS OF THE BAR ARE CALCULATED AND CHECKED. THE PROGRAM WILL FAIL TO UPDATE THE PC RETURN ADDRESS BY 2 IF ERROR IS SET, READY IS CLEAR, READY AND ERROR ARE CLEAR OF THE CSR, WCR IS NOT ZERO OR THE BAR CONTENTS IS NOT ZERO. THIS WILL CALL THE ERROR THAT IS JUST AFTER THE JSR CALL IN THE TEST. IF ALL DATA IS ACCEPTABLE, THE PC IS UPDATED, AND THE RETURN FROM THE SUBROUTINE IS AFTER THE ERROR CALL.

10.8 DATCHK

THIS ROUTINE IS ENTERED TO CHECK INBUF AFTER A MAINTENANCE MODE OPERATION. THE CONTENTS OF INBUF AND THE CONTENTS OF CHKBUFF ARE CHECKED TO SEE THAT THEY ARE THE SAME. THE NUMBER OF COMPARISONS MADE IS DETERMINED BY THE CONTENTS OF BUFLN. ANY ERRORS RESULT IN AN RTS TO THE TEST TO CALL THE ERROR THERE. A JSR BACK TO THE SUBROUTINE IS EXECUTED TO RESUME ITS CHECKING. WHEN RETURNING, SP RETURN ADDRESS IS UPDATED BY 6 TO RETURN AFTER THE ERROR CALL AND JSR RETURN.

10.9 CLENUP

THE ROUTINE IS ENTERED AT THE END OF SEVERAL TESTS TO CLEAR ANY DATA THAT MAY HAVE BEEN LEFT IN ANY REGISTERS, AND TO RESTORE THE INTERRUPT VECTORS.

10.10 CHKCAB

THIS ROUTINE IS USED IN VARIOUS TESTS TO ALTER THE EXPECTED DATA IF THE WRAP-AROUND CABLE IS OUT.

10.11 DATOCK

AFTER A STRING OF DATO'S HAS BEEN COMPLETED THIS ROUTINE CHECKS THAT THE CORRECT DATA PATTERN WAS TRANSFERRED TO INBUF. THE NUMBER OF COMPARISONS MADE IS DETERMINED BY THE CONTENTS OF BUFLN. AN ERROR IN THE CHECK RESULTS IN AN RTS TO THE TEST TO CALL THE FIRST ERROR AFTER THE JSR CALL, WHERE A JSR RETURNS CONTROL BACK TO THE SUBROUTINE FOR FURTHER CHECKING. AN ADDITIONAL CHECK IS MADE ON BUFLN+2 TO INSURE THAT NOT TOO MANY WORDS WERE TRANSFERRED. IF THEY WERE, THE PC RETURN ADDRESS IS ALTERED SO THAT THE SECOND ERROR AFTER THE JSR IS CALLED. IF NO ERRORS, RETURN IS ALTERED TO JUST AFTER THE SECOND ERROR CALL.

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

10.12 ERRCHK
THIS ROUTINE CLEARS IE AND UPDATES THE PC FOR RETURN AFTER THE ERROR IN THE TEST IF ERROR IS CLEAR. IF SET, RETURN IS EXECUTED WITHOUT UPDATING THE PC RETURN SO THE ERROR CALL AFTER THE JSR CALL IN THE TEST WILL BE CALLED.

10.13 DEVADS
THIS ROUTINE GENERATES AN ADDRESS TABLE LOCATED AT REGADR STARTING WITH THE BASE DEVICE ADDRESS (CONTENTS OF \$BASE) IN STEPS OF 10.

10.14 BPINIT
THIS SUBROUTINE RELOADS THE ''.+2'' AND 'BPT' INTO THE UNUSED LOCATIONS BETWEEN 4 AND 776.

10.15 DRGET
THIS SUBROUTINE EXTRACTS INFORMATION ABOUT THE DR11 THAT INTERRUPTED AND LOADS THE ACCUMULATED DATA INTO THE DEVICE DESCRIPTOR WORD FOR THAT BOARD.

10.16 TYPCHF
THIS SUBROUTINE PRINTS THE BOARD CONFIGURATIONS THAT THE ASIZE SUBROUTINE FOUND ON THE UNIBUS.

10.17 CHK4DR
THIS SUBROUTINE CHECKS FOR A LOCATION AS BELONGING TO A DR11 BY TRYING TO FORCE AN INTERRUPT A TOTAL OF 4 TIMES WITH THE CPU AT PRIORITIES 6 THROUGH 3. IF THE ATTEMPT FAILS, ROUTINE CORRECTS THE STACK TO RETURN AFTER THE DR11 EXTRACTION ROUTINE. IF THE LOCATION DOES BELONG TO A DR11, THE RETURN ADDRESS OF THE SUBROUTINE ON THE STACK IS MOVED DOWN ONE LOCATION, AND THE ADDRESS+4 OF THE INTERRUPT VECTOR OF THE DR11 IS PUT IN THE RETURN ADDRESSES PREVIOUS LOCATION. THE STACK IS THEN POPPED 3 TIMES BY ADJUSTING THE POINTER UP 6, AND A NORMAL RETURN IS EXECUTED.

10.18 ASIZE
THIS ROUTINE AUTOSIZES THE BOARD CONFIGURATION AND PRINTS THE CONFIGURATION IF BIT 12 (10000) IS SET IN THE SWR.

10.19 VCTADS
THIS ROUTINE GENERATES THE VECTOR ADDRESS TABLE STARTING WITH THE ADDRESS IN LOCATION 'VECADR'.

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

10.20 CATCH

THIS ROUTINE REPORTS UNEXPECTED OR ERRONEOUS TRAPS OR INTERRUPTS THROUGH THE BREAK-POINT-TRAP LOADED IN LOCATIONS 4-776. THE STACK IS CLEANED 4 TIMES BEFORE THE ERROR CALL, AND RESTORED TWICE AFTER THE ERROR CALL FOR RETURNING TO THE SOURCE OF THE TRAP.

10.21 PSTATE

THIS ROUTINE PRINTS THE STATE OF THE BIT IN THE DDW THAT WAS PRELOADED IN LOCATION 'BITTST'.

10.22 PNTPRI

THIS ROUTINE PRINTS THE DEVICE PRIORITY IN THE DDW LOCATION.

10.23 SETUP

THIS SUBROUTINE INITIALIZES THE TRAP AND INTERRUPT VECTORS.

10.24 TSTMM

THIS SUBROUTINE CHECKS FOR EXISTENCE OF MEMORY MANAGEMENT AND IF IT EXISTS, CHECKS FOR THE ERROR CONDITION OF NO MEMORY LOCATION, BUT NO ERROR AND NEX BIT SETS. IF MEMORY MANAGEMENT IS NOT THERE, AN EXIT UPDATING THE RETURN ADDRESS BY 2 IS DONE. IF THERE, THE XBA16 AND XBA17 BITS OF THE EXPECTED DATA ARE CHECKED. IF BOTH ZERO, AN EXIT UPDATING THE RETURN ADDRESS BY 2 IS DONE. IF EITHER OR BOTH ARE SET, THE UPPER BYTE OF THE MEMORY MANAGEMENT LOCATION IS CHECKED FOR THE EXISTENCE OF UPPER MEMORY, INITIALIZED AT THE BEGINNING OF THE DIAGNOSTIC. IF NOT THERE (BITS 0, 1 OR 2 OF UPPER BYTE CLEAR), A NORMAL EXIT IS EXECUTED SO THE BRANCH IMMEDIATELY FOLLOWING THE JSR CALL WILL CAUSE A CHECK FOR THE ERROR BITS IN THE EXPECTED TO BE SET FOR ANOTHER CHECK.

947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966

11.0 DATA STACK

11.1 PATRNS

THIS SET OF 7 DATA WORDS IS USED TO CHECK ANY LOCATION FOR STUCK OR SHORTED BITS.

11.2 EXPATO

THIS SET OF DATA WORDS IS USED IN TEST 31 TO CHECK ALL POSSIBLE COMBINATIONS OF SET BITS IN THE CSR WITH THE MAINTENANCE BIT CLEAR. IT CONTAINS THE EXPECTED DATA THAT THE CSR SHOULD CONTAIN AFTER THE BIT COMBINATION IS WRITTEN TO THE CSR.

11.3 EXPAT1

THIS SET OF DATA WORDS IS USED IN TEST 3 TO CHECK ALL POSSIBLE COMBINATIONS OF SET BITS IN THE CSR WITH THE MAINTENANCE BIT SET. IT CONTAINS THE EXPECTED DATA THAT THE CSR SHOULD CONTAIN AFTER THE BIT COMBINATION IS WRITTEN TO THE CSR.

967
968
1006

2
*NLIST MC,MD,CND
*TITLE CZDRLCO-DR11 GEN NPR INTFC
*COPYRIGHT (C) 1980
*DIGITAL EQUIPMENT CORP.
*MAYNARD, MASS. 01754

*PROGRAM BY DAN MILLEVILLE

THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977, MODIFIED FOR THE CZDRLC DIAGNOSTIC. CHANGES ARE NOTED BY [;88] IN THE COMMENT FIELDS OF THE \$SCOPE, \$EOP, \$TYPE, \$ERROR, \$CKSWR \$PWDRN, \$PWUP (ELIMINATED), \$ERRTYP AND \$TYPDS ROUTINES. NEWTST WAS MODIFIED IN FUNCTION AS DESCRIBED BELOW. CHANGES WERE AS FOLLOWS:

\$SCOPE: END-OF-PASS MESSAGE DISABLING AND REENABLING CAPABILITY, PLUS UPDATING \$TESTN JUST AFTER \$TSTNM IS UPDATED, AS WELL AS ANY NEEDED MESSAGES.

\$EOP: CHANGED TO RECOGNIZE WHETHER OR NOT THE USER WISHES THE EOP MESSAGES PRINTED, AND PRINTS THE NUMBER OF ERRORS IN THAT PASS IF THERE WERE ANY, AND TYPES AN EXTRA <CRLF> IF THERE WERE ERRORS TO SPACE EOP FROM THE ERROR. CHANGED ALSO TO RECOGNIZE WHEN THE PASS COUNT GOES NEGATIVE, AND IF SO, CLEARS THE PASS COUNT, AND INCREMENTS THE NEXT LOCATION AFTER \$PASS TO COUNT A BLOCK OF 32768 PASSES HAS OCCURED. THIS LOCATION IS USED IN CONJUNCTION WITH \$PASS TO PRINT UP TO PASS # 99,999,999 DECIMAL.

\$TYPE: INSTEAD OF USING THE STACK TO LOAD A CHARACTER INPUTED WHILE PRINTING, LOOSING THE CHARACTER IN THE PROCESS, LOCATION 'CHARCT' IS USED TO SAVE IT FOR \$SCOPE TO USE.

\$ERROR: INCREMENT LOCATION 'ERRCNT' FOR POSSIBLE USE IN MULTIPLE ERROR PRINTOUTS.

\$CKSWR: TESTING AND PROCESSING OF THE CONTENTS OF LOCATION 'CHARCT' WAS ADDED TO INCREASE CHANCES OF DIAGNOSTIC CATCHING USER REQUEST FOR A SOFTWARE SWITCH REGISTER CHANGE.

\$ERRTYP: ADDED CAPABILITY TO PROCESS ERRORS WITHIN LOOPS (ERRORS WITH ERROR NUMBERS BETWEEN 201-377) SO THE MESSAGE AND DATA HEADER ARE PRINTED DURING THE 1ST ERROR ONLY, WITH DATA ONLY PRINTED FOR 2ND AND SUBSEQUENT ERRORS. IT CANCELS DATA PRINTING AFTER 20 (OCTAL) ERRORS HAVE BEEN DONE SO AS TO ELIMINATE MASSIVE ERROR TYPEOUTS, BUT CONTINUES TO TALLY THE ERRORS IN LOCATION '\$ERTTL' SO THE EOP MESSAGE WILL SHOW THE TOTAL NUMBER OF ERRORS IN THAT PASS. IF AN ERROR NUMBER BELOW 201 IS CALLED, 'ERRCNT' IS CLEARED SO IF THIS ERROR IS IN A LOOP, ANY SUBSEQUENT 201+ ERRORS WILL HAVE THERE HEADER REPRINTED.

NEWSTST: '.PAGE' WAS ADDED SO EACH NEW TEST WOULD BE ON A NEW PAGE. AT THE BEGINING OF EACH NEW TEST, THE TITLE AND TEST NUMBER ARE WRITTEN IN A SUBTITLE SO THAT EACH TEST WILL APPEAR IN THE TABLE OF CONTENTS AT THE BEGINING OF THE DIAGNOSTIC. AN ADDITIONAL TRACE FEATURE WAS ADDED AT THE BEGINNING OF EACH TEST TO CHECK BIT 11 OF THE SWITCH REGISTER. WITH THIS BIT SET, THE USER CAN SEE WHAT TEST WAS EXECUTING WHEN THE OPERATION CEASED TO BE SEQUENTIAL.

SPWRDN: THE OPERATIONS TO SAVE REGISTER CONTENTS WERE ELIMINATED DUE TO THE LACK OF THE NEED. A LOAD OF THE \$PWRUP ADDRESS, LOCATED JUST AFTER THE POWER DOWN HALT, IS LOADED INTO THE POWER TRAP VECTOR SO THAT ON POWER UP, THE PROGRAM WILL RESTART. IF PROCESSOR IS UNABLE TO GET TO THE POWER DOWN ROUTINE ON A POWER FAIL (CPU HALTED WHEN POWER FAILURE OCCURED), PROCESSOR WILL EXECUTE THE POWER DOWN ROUTINE AND HALT. HITTING CONTINUE WILL RESULT IN THE POWER UP ROUTINE EXECUTING AS IT WOULD IN THE EVENT OF A RESTORATION OF POWER AFTER A POWER FAILURE WITH THE CPU RUNNING.

\$TYPDS: THE ADDITION OF THE \$TYPDE FUNCTION WAS ADDED. THIS FUNCTION ALLOWS THE PRINTING OF NUMBERS LARGER THAN 32767 DECIMAL. THE LOCATION THAT CONTAINS THE COUNT IS TO BE TESTED AFTER BEING INCREMENTED. IF NEGATIVE, IT IS TO BE CLEARED AND A SECOND (OVERFLOW) LOCATION IS TO BE INCREMENTED. WHEN CALLING THE ROUTINE, THE OVERFLOW LOCATION IS TO BE PUT ON THE STACK, THEN THE NUMBER, THEN THE CALL. IF THE OVERFLOW LOCATION IS NON-ZERO, IT WILL ADD 32768 TO THE ASCII NUMBER FOR EACH COUNT IN THAT OVERFLOW LOCATION.

.SBTTL OPERATIONAL SWITCH SETTINGS

SWITCH	USE
15	HALT ON ERROR
14	LOOP ON TEST
13	INHIBIT ERROR TYPEOUTS
12	DO NOT PRINT BOARD CONFIGURATION
11	TEST NUMBER TRACE ENABLING
10	BELL ON ERROR
9	LOOP ON ERROR
8	LOOP ON TEST IN SWR<6:0>
7	INHIBIT MULTIPLE ERROR TYPEOUTS

1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021

>>>>>>>>>>NOTE<<<<<<<<<<<<
THE "TEST NUMBER TRACE ENABLING" ABOVE IS TO ENABLE PRINTING OF THE
TEST NUMBER AT THE BEGINNING OF EACH TEST, SHOULD YOU HAVE A PROBLEM
WITH THE DIAGNOSTIC JUMPING TO AN ILLEGAL LOCATION FROM ANYWHERE IN
THE TEST. BY ENABLING IT, YOU CAN SEE THE LAST TEST ENTERED BEFORE
THE OUT-OF-SEQUENCE JUMPING OCCURED. THIS IS USEFUL IF YOU ENCOUNTER
AN UNEXPECTED TRAP FROM A PARTICULAR TEST, BUT DISCOVER THAT THE TEST
RAN SUCCESSFULLY WHEN LOOPED.

1023

```
.SBTTL BASIC DEFINITIONS
;*INITIAL ADDRESS OF THE STACK POINTER *** 1300 ***
STACK= 1300
      ERROR=EMT
      SCOPE=10T
;*MISCELLANEOUS DEFINITIONS
HT= 11          ;;CODE FOR HORIZONTAL TAB
LF= 12          ;;CODE FOR LINE FEED
CR= 15          ;;CODE FOR CARRIAGE RETURN
CPLF= 200       ;;CODE FOR CARRIAGE RETURN-LINE FEED
PS= 177776     ;;PROCESSOR STATUS WORD
      PSW=PS
STKLM= 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= X0          ;;GENERAL REGISTER
R1= X1          ;;GENERAL REGISTER
R2= X2          ;;GENERAL REGISTER
R3= X3          ;;GENERAL REGISTER
R4= X4          ;;GENERAL REGISTER
R5= X5          ;;GENERAL REGISTER
R6= X6          ;;GENERAL REGISTER
R7= X7          ;;GENERAL REGISTER
SP= X6          ;;STACK POINTER
PC= X7          ;;PROGRAM COUNTER
;*PRIORITY LEVEL DEFINITIONS
PR0= 0          ;;PRIORITY LEVEL 0
PR1= 40         ;;PRIORITY LEVEL 1
PR2= 100        ;;PRIORITY LEVEL 2
PR3= 140        ;;PRIORITY LEVEL 3
PR4= 200        ;;PRIORITY LEVEL 4
PR5= 240        ;;PRIORITY LEVEL 5
PR6= 300        ;;PRIORITY LEVEL 6
PR7= 340        ;;PRIORITY LEVEL 7
;*SWITCH REGISTER SWITCH DEFINITIONS
SW15= 100000
SW14= 40000
SW13= 20000
SW12= 10000
SW11= 4000
SW10= 2000
SW09= 1000
SW08= 400
SW07= 200
SW06= 100
SW05= 40
SW04= 20
SW03= 10
SW02= 4
SW01= 2
SW00= 1
SW9=SW09
SW8=SW08
SW7=SW07
SW6=SW06
```

```
000040          SW5=SW05
000020          SW4=SW04
000010          SW3=SW03
000004          SW2=SW02
000002          SW1=SW01
000001          SW0=SW00

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

001000          BIT9=BIT09
000400          BIT8=BIT08
000200          BIT7=BIT07
000100          BIT6=BIT06
000040          BIT5=BIT05
000020          BIT4=BIT04
000010          BIT3=BIT03
000004          BIT2=BIT02
000002          BIT1=BIT01
000001          BIT0=BIT00

; *BASIC "CPU" TRAP VECTOR ADDRESSES
000004          ERRVEC= 4          ;; TIME OUT AND OTHER ERRORS
000010          RESVEC= 10         ;; RESERVED AND ILLEGAL INSTRUCTIONS
000014          TBITVEC=14         ;; "T" BIT
000014          TRTVEC= 14         ;; TRACE TRAP
000014          BPTVEC= 14         ;; BREAKPOINT TRAP (BPT)
000020          IOTVEC= 20         ;; INPUT/OUTPUT TRAP (IOT) **SCOPE**
000024          PWRVEC= 24         ;; POWER FAIL
000030          EMTVEC= 30         ;; EMULATOR TRAP (EMT) **ERROR**
000034          TRAPVEC=34         ;; "TRAP" TRAP
000060          TKVEC= 60          ;; TTY KEYBOARD VECTOR
000064          TPVEC= 64          ;; TTY PRINTER VECTOR
000240          PIRQVEC=240        ;; PROGRAM INTERRUPT REQUEST VECTOR

1024          000004          BUSERR =ERRVEC
1025          172410          ABASE  =172410          ;;BASE DEVICE ADDRESS
1026          000300          AVECT1  -300          ;;BASE VECTOR ADDRESS
```

1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051

000001
000002
000004
000010
000016
000020
000040
000100
000200
000400
000400
001000
002000
004000
006000
005000
003000
007000
010000
020000
040000
100000
100000

.SBTTL DEFINITIONS OF THE CSR BITS
;*****
GO =1 ;GO
F1 =2 ;FNCT1
F2 =4 ;FNCT2
F3 =10 ;FNCT3
FNC =16 ;FNCT1 & FNCT2 & FNCT3
X6 =20 ;XBA16
X7 =40 ;XBA17
IE =100 ;IE
RY =200 ;READY
CY =400 ;CYCLE
N2 =400 ;2/N BIT
DSC =1000 ;DSTAT C
DSB =2000 ;DSTAT B
DSA =4000 ;DSTAT A
DAB =6000 ;DSTAT A & B
DAC =5000 ;DSTAT A & C
DBC =3000 ;DSTAT B & C
DST =7000 ;DSTAT A & B & C
MA =10000 ;MAINT
AT =20000 ;ATTN
NX =40000 ;NEX
EIR =100000 ;EIR
FR =100000 ;ERROR

```

1052                                     .SBTTL CSR BIT COMPLIMENT DEFINITIONS
1053 ;*****
1054 CGO      =177776 ;COMPLIMENT OF GO
1055 CF1      =177775 ;COMPLIMENT OF FNCT1
1056 CF2      =177773 ;COMPLIMENT OF FNCT2
1057 CF3      =177767 ;COMPLIMENT OF FNCT3
1058 CFNC     =177761 ;COMPLIMENT OF FNCT1 & FNCT2 & FNCT3
1059 CX6      =177757 ;COMPLIMENT OF XBA16
1060 CX7      =177737 ;COMPLIMENT OF XBA17
1061 CIE      =177677 ;COMPLIMENT OF IE
1062 CRY      =177577 ;COMPLIMENT OF READY
1063 CCY      =177377 ;COMPLIMENT OF CYCLE
1064 CDSC     =176777 ;COMPLIMENT OF DSTAT C
1065 CDSB     =175777 ;COMPLIMENT OF DSTAT B
1066 CDSA     =173777 ;COMPLIMENT OF DSTAT A
1067 CDAB     =171777 ;COMPLIMENT OF DSTAT A & B
1068 CDAC     =172777 ;COMPLIMENT OF DSTAT A & C
1069 CDBC     =174777 ;COMPLIMENT OF DSTAT B & C
1070 CDST     =170777 ;COMPLIMENT OF DSTAT A & B & C
1071 CMA      =167777 ;COMPLIMENT OF MAINT
1072 CAT      =157777 ;COMPLIMENT OF ATTN
1073 CNX      =137777 ;COMPLIMENT OF NEX
1074 CEIR     =77777 ;COMPLIMENT OF EIR
1075 CER      =77777 ;COMPLIMENT OF ERROR

```

		.SBTTL	COMPLEMENTS OF BIT DEFINITIONS
1076		*****	*****
1077		*****	*****
1078	177776	CBIT0	=177776 ; COMPLIMENT OF BIT0
1079	177775	CBIT1	=177775 ; COMPLIMENT OF BIT1
1080	177773	CBIT2	=177773 ; COMPLIMENT OF BIT2
1081	177767	CBIT3	=177767 ; COMPLIMENT OF BIT3
1082	177757	CBIT4	=177757 ; COMPLIMENT OF BIT4
1083	177737	CBIT5	=177737 ; COMPLIMENT OF BIT5
1084	177677	CBIT6	=177677 ; COMPLIMENT OF BIT6
1085	177577	CBIT7	=177577 ; COMPLIMENT OF BIT7
1086	177377	CBIT8	=177377 ; COMPLIMENT OF BIT8
1087	176777	CBIT9	=176777 ; COMPLIMENT OF BIT9
1088	175777	CBIT10	=175777 ; COMPLIMENT OF BIT10
1089	173777	CBIT11	=173777 ; COMPLIMENT OF BIT11
1090	167777	CBIT12	=167777 ; COMPLIMENT OF BIT12
1091	157777	CBIT13	=157777 ; COMPLIMENT OF BIT13
1092	137777	CBIT14	=137777 ; COMPLIMENT OF BIT14
1093	077777	CBIT15	=77777 ; COMPLIMENT OF BIT15
1094	057777	CB1513	=57777 ; COMPLIMENT OF BIT15 & BIT 13

1095		.SBTTL	PRIORITY LEVELS AND OTHER DEFINITIONS
1096		*****	
1097	000140	LEVEL3	=140
1098	000200	LEVEL4	=200
1099	000240	LEVEL5	=240
1100	000300	LEVEL6	=300
1101	000340	LEVEL7	=340
1102	000033	ESC	=33
1103	000003	CNTLC	=3
1104	000015	CARETN	=15
1105	177572	MMRO	=177572
1106	172304	KIPDR2	=172304
1107	172324	KDPDR2	=172324
1108	172344	KIPAR2	=172344
1109	172364	KDPAR2	=172364
1110	000250	MMVECT	=250
1111	000252	MMPS	=252
1112	000004	TOVECT	=4
1113	000006	TMOPSW	=6
1114	000003	BPT	=3
1115	024360	TST40=ENDEV	;BRANCH TO TEST 40 = BRANCH TO ENDEV (THERE IS NO TEST 40)


```

1116 ;*****
1117 ;* ALL UNUSED LOCATIONS FROM 4-776 WILL CONTAIN A ".+2,BPT" SEQUENCE
1118 ;* TO CATCH ILLEGAL TRAPS & INTERRUPTS TO THE 'CATCH' LOCATION
1119 ;* 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
1120 ;*****
1121      000014      000014      ;THE BPT TRAP VECTOR POINTS TO THE
1122 000014 005536      BPTVCT: .WORD CATCH      ; ILLEGAL TRAP HANDLER 'CATCH'
1123 000016 000340      .WORD LEVEL7
1125      000042      .=42
1126 000042 000000      .WORD 0      ;CLEAR THIS LOCATION (FOR APT MONITOR STARTING ADDRESS)
1127      000174      .=174
1128 000174 000000      DISPRE: .WORD 0
1129 000176 000000      SWREG: .WORD 0
1130 ;*****
1131 ;PROGRAM STARTING LOCATIONS
1132 ;*****
1133
1134
1135 000200 000137 010266      JMP START1      ;NORMAL START
1136 000204 000137 032130      JMP MBD      ;ENTER MULTIPLE BOARD DIALOGUE
1137 000210 005037 001416      STAGIN: CLR $PASS      ;CLEAR $PASS
1138 000214 005037 002716      CLR $PASS2      ;CLEAR $PASS2
1139 000220 005037 001422      CLR $UNIT      ;CLEAR $UNIT
1140 000224 005037 001420      CLR $DEVCT      ;CLEAR $DEVCT
1141 000230 005037 001414      CLR $TESTN      ;CLEAR $TESTN
1142 000234 005037 002706      CLR EOPLOC      ;CLEAR EOPLOC
1143 000240 012737 042456 042454      MOV #CAPSTK,CAPNTR ;RESET THE CAPTURE POINTER
1144 000246 012706 010272      MOV #START,SP      ;RESET THE STACK POINTER
1145 000252 000137 011006      JMP BEGIN1      ;JUMP TO BEGIN1 FOR RESTART WITHOUT HEADER PRINTING
1146      001000      .=1000
  
```

1147

001000
000046 000046
000046 024750
000052 000052
000052 000000
001000

1148

001000
000024 000024
000024 000200
000044 000044
000044 001000
001000

001000
001000 000000
001002 001410
001004 000001
001006 000001
001010 000000
001012 000052

.SBTTL ACT11 HOOKS

:*****

:HOOKS REQUIRED BY ACT11

.\$VPC=.

;SAVE PC

.=46

.\$ENDAD

;;1)SET LOC.46 TO ADDRESS OF \$ENDAD IN .\$EOP

.=52

.WORD 0

;;2)SET LOC.52 TO ZERO

.=\$VPC

;; RESTORE PC

.SBTTL APT PARAMETER BLOCK

:*****

:SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT

:*****

.\$X=. ;;SAVE CURRENT LOCATION

.=24 ;;SET POWER FAIL TO POINT TO START OF PROGRAM

200 ;;FOR APT START UP

.=44 ;;POINT TO APT INDIRECT ADDRESS PNTR.

.\$APTHDR ;;POINT TO APT HEADER BLOCK

.=\$X ;;RESET LOCATION COUNTER

:*****

:SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC

:INTERFACE SPEC.

.\$APTHD:

.\$HIBTS: .WORD 0 ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.

.\$MBADR: .WORD \$MAIL ;;ADDRESS OF APT MAILBOX (BITS 0-15)

.\$TSTM: .WORD 1 ;;RUN TIM OF LONGEST TEST

.\$PASTM: .WORD 1 ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)

.\$UNITM: .WORD 0 ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT

.WORD \$ETEND-\$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)

1150

```
.SBTTL COMMON TAGS
*****
;THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
;USED IN THE PROGRAM.
.=1300

001300 001300
001300 000000
001302 000
001303 000
001304 000000
001306 000000
001310 000000
001312 000000
001314 000
001315 001
001316 000000
001320 000000
001322 000000
001324 000000
001326 000000
001330 000000
001332 000000
001334 000
001335 000
001336 000000
001340 177570
001342 177570
001344 177560
001346 177562
001350 177564
001352 177566
001354 000
001355 002
001356 012
001357 000
001360 000007
001362 000000
001364 000000
001366 000000
001370 000000
001372 000000
001374 000000
001376 000000
001400 207 377 377
001404 077
001405 015
001406 012 000

$CMTAG:
$STNM: .WORD 0
$ERFLG: .BYTE 0
$ICNT: .WORD 0
$LPADR: .WORD 0
$LPERR: .WORD 0
$ERTTL: .WORD 0
$ITEMB: .BYTE 0
$ERMAX: .BYTE 1
$EPRPC: .WORD 0
$GDADR: .WORD 0
$BDADR: .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
$TPFLG: .BYTE 0
$REPT 7
$TMP0: .WORD 0
$TMP1: .WORD 0
$TMP2: .WORD 0
$TMP3: .WORD 0
$TMP4: .WORD 0
$TMP5: .WORD 0
$TMP6: .WORD 0
$ESCAPE: 0
$BELL: .ASCIIZ <207><377><377>
$QUES: .ASCII /?/
$CRLF: .ASCII <15>
$LF: .ASCIIZ <12>

;;START OF COMMON TAGS
;;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)
;;USER DEFINED
;;USER DEFINED
;;USER DEFINED
;;USER DEFINED
;;USER DEFINED
;;USER DEFINED
;;USER DEFINED
;;ESCAPE ON ERROR ADDRESS
;;CODE FOR BELL
;;QUESTION MARK
;;CARRIAGE RETURN
;;LINE FEED
*****
```

```

.SBTTL  APT MAILBOX-ETABLE
*****
.EVEN
001410      $MAIL:
001410      $MSGTY: .WORD  AMSGTY  ;; APT MAILBOX
001412      $FATAL: .WORD  AFATAL  ;; MESSAGE TYPE CODE
001414      $TESTN: .WORD  ATESTN  ;; FATAL ERROR NUMBER
001416      $PASS:  .WORD  APASS   ;; TEST NUMBER
001420      $DEVCT: .WORD  ADEVCT  ;; PASS COUNT
001422      $UNIT:  .WORD  AUNIT   ;; DEVICE COUNT
001424      $MSGAD: .WORD  AMSGAD  ;; I/O UNIT NUMBER
001426      $MSGLG: .WORD  AMSGLG  ;; MESSAGE ADDRESS
001430      $ETABLE:
001430      $ENV:   .BYTE  AENV    ;; MESSAGE LENGTH
001431      $ENVM:  .BYTE  AENVM   ;; APT ENVIRONMENT TABLE
001432      $SWREG: .WORD  ASWREG  ;; ENVIRONMENT BYTE
001434      $USWR:  .WORD  AUSWR   ;; ENVIRONMENT MODE BITS
001436      $CPUOP: .WORD  ACPUOP  ;; APT SWITCH REGISTER
                                ;; USER SWITCHES
                                ;; CPU TYPE, OPTIONS
                                BITS 15-11=CPU TYPE
                                11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
                                11/70=06,PDQ=07,Q=10
                                BIT 10=REAL TIME CLOCK
                                BIT 9=FLOATING POINT PROCESSOR
                                BIT 8=MEMORY MANAGEMENT
                                ;; HIGH ADDRESS,M.S. BYTE
                                ;; MEM. TYPE,BLK#1
                                MEM.TYPE BYTE -- (HIGH BYTE)
                                900 NSEC CORE=001
                                300 NSEC BIPOLAR=002
                                500 NSEC MOS=003
001440      $MADR1: .WORD  AMADR1  ;; HIGH ADDRESS,BLK#1
001441      $MTYP1: .BYTE  AMTYP1  ;; MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
                                ;; HIGH ADDRESS,M.S. BYTE
                                ;; MEM.TYPE,BLK#2
                                ;; MEM.LAST ADDRESS,BLK#2
                                ;; HIGH ADDRESS,M.S.BYTE
                                ;; MEM.TYPE,BLK#3
                                ;; MEM.LAST ADDRESS,BLK#3
                                ;; HIGH ADDRESS,M.S.BYTE
                                ;; MEM.TYPE,BLK#4
                                ;; MEM.LAST ADDRESS,BLK#4
                                ;; INTERRUPT VECTOR#1,BUS PRIORITY#1
                                ;; INTERRUPT VECTOR#2BUS PRIORITY#2
                                ;; BASE ADDRESS OF EQUIPMENT UNDER TEST
                                ;; DEVICE MAP
                                ;; CONTROLLER DESCRIPTION WORD#1
                                ;; CONTROLLER DESCRIPTION WORD#2
                                ;; DEVICE DESCRIPTOR WORD#0
                                ;; DEVICE DESCRIPTOR WORD#1
                                ;; DEVICE DESCRIPTOR WORD#2
                                ;; DEVICE DESCRIPTOR WORD#3
                                ;; DEVICE DESCRIPTOR WORD#4
                                ;; DEVICE DESCRIPTOR WORD#5
                                ;; DEVICE DESCRIPTOR WORD#6
                                ;; DEVICE DESCRIPTOR WORD#7
                                ;; DEVICE DESCRIPTOR WORD#8
                                ;; DEVICE DESCRIPTOR WORD#9
001442      $MADR2: .WORD  AMADR2  ;; HIGH ADDRESS,BLK#1
001444      $MTYP2: .BYTE  AMTYP2  ;; MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
001445      $MADR3: .WORD  AMADR3  ;; HIGH ADDRESS,BLK#1
001446      $MTYP3: .BYTE  AMTYP3  ;; MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
001450      $MADR4: .WORD  AMADR4  ;; HIGH ADDRESS,BLK#1
001451      $MTYP4: .BYTE  AMTYP4  ;; MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
001452      $MADR5: .WORD  AMADR5  ;; HIGH ADDRESS,BLK#1
001454      $MTYP5: .BYTE  AMTYP5  ;; MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
001455      $MADR6: .WORD  AMADR6  ;; HIGH ADDRESS,BLK#1
001456      $MTYP6: .BYTE  AMTYP6  ;; MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
001460      $VECT1: .WORD  AVECT1  ;; INTERRUPT VECTOR#1,BUS PRIORITY#1
001462      $VECT2: .WORD  AVECT2  ;; INTERRUPT VECTOR#2BUS PRIORITY#2
001464      $BASE:  .WORD  ABASE   ;; BASE ADDRESS OF EQUIPMENT UNDER TEST
001466      $DEVN:  .WORD  ADEVN   ;; DEVICE MAP
001470      $CDW1:  .WORD  ACDW1   ;; CONTROLLER DESCRIPTION WORD#1
001472      $CDW2:  .WORD  ACDW2   ;; CONTROLLER DESCRIPTION WORD#2
001474      $DDW0:  .WORD  ADDW0   ;; DEVICE DESCRIPTOR WORD#0
001476      $DDW1:  .WORD  ADDW1   ;; DEVICE DESCRIPTOR WORD#1
001500      $DDW2:  .WORD  ADDW2   ;; DEVICE DESCRIPTOR WORD#2
001502      $DDW3:  .WORD  ADDW3   ;; DEVICE DESCRIPTOR WORD#3
001504      $DDW4:  .WORD  ADDW4   ;; DEVICE DESCRIPTOR WORD#4
001506      $DDW5:  .WORD  ADDW5   ;; DEVICE DESCRIPTOR WORD#5
001510      $DDW6:  .WORD  ADDW6   ;; DEVICE DESCRIPTOR WORD#6
001512      $DDW7:  .WORD  ADDW7   ;; DEVICE DESCRIPTOR WORD#7
001514      $DDW8:  .WORD  ADDW8   ;; DEVICE DESCRIPTOR WORD#8
001516      $DDW9:  .WORD  ADDW9   ;; DEVICE DESCRIPTOR WORD#9

```

CZDRLCO-DR11 GEN NPR INTFC
APT MAILBOX-ETABLE

MACRO M1113 27-OCT-80 15:53 PAGE ^{E 4}44-1

SEQ 0043

001520	000000	\$DDW10:	.WORD	ADDW10	::DEVICE	DESCRIPTOR	WORD#10
001522	000000	\$DDW11:	.WORD	ADDW11	::DEVICE	DESCRIPTOR	WORD#11
001524	000000	\$DDW12:	.WORD	ADDW12	::DEVICE	DESCRIPTOR	WORD#12
001526	000000	\$DDW13:	.WORD	ADDW13	::DEVICE	DESCRIPTOR	WORD#13
001530	000000	\$DDW14:	.WORD	ADDW14	::DEVICE	DESCRIPTOR	WORD#14
001532	000000	\$DDW15:	.WORD	ADDW15	::DEVICE	DESCRIPTOR	WORD#15
001534		\$ETEND:					
		.MEXIT					

```
.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 $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
;*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
;*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
$ERRTB:
;ITEM 1
      .WORD EM2      ;;CANNOT ACCESS DR11 REGISTER
      .WORD DH2      ;;TEST # ERR PC ABTPC REGISTER
      .WORD DT2      ;;$TESTN,$ERRPC,OLDPC1,DREG,0
      .WORD 0        ;;PRINT ALL DATA OCTAL
;ITEM 2
      .WORD EM3      ;;DR11-B OR W MODE INCORRECT (0=B, 1=W)
      .WORD DH3      ;;TEST # ERR PC EXPMOD ACTMOD CSRADR
      .WORD DT3      ;;$TESTN,$ERRPC,$TMP1,BORW,CSR,0
      .WORD 0        ;;PRINT ALL DATA OCTAL
;ITEM 3
      .WORD EM4      ;;INIT FAILED TO CLEAR WCR
      .WORD DH4      ;;TEST # ERR PC WCRADR WCRCONTENTS
      .WORD DT4      ;;$TESTN,$ERRPC,WCR,RWCR,0
      .WORD 0        ;;PRINT ALL DATA OCTAL
;ITEM 4
      .WORD EM5      ;;INIT FAILED TO CLEAR BAR
      .WORD DH5      ;;TEST # ERR PC BARADR BAREXP BARRCV
      .WORD DT5      ;;$TESTN,$ERRPC,BAR,EBAR,RBAR,0
      .WORD 0        ;;PRINT ALL DATA OCTAL
;ITEM 5
      .WORD EM6      ;;INIT FAILED TO CLEAR BDR
      .WORD DH6      ;;TEST # ERR PC BDRADR BDRCONTENTS
      .WORD DT6      ;;$TESTN,$ERRPC,BDR,RBDR,0
      .WORD 0        ;;PRINT ALL DATA OCTAL
;ITEM 6
      .WORD EM7      ;;INIT FAILED TO CLEAR ALL CSR R-W BITS
      .WORD DH7      ;;TEST # ERR PC CSRADR CSREXP CSRCONTENTS
      .WORD DT7      ;;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
      .WORD 0        ;;PRINT ALL DATA OCTAL
;ITEM 7
      .WORD EM10     ;;RESET FAILED TO CLEAR WCR
      .WORD DH4      ;;TEST # ERR PC WCRADR WCRCONTENTS
      .WORD DT4      ;;$TESTN,$ERRPC,WCR,RWCR,0
      .WORD 0        ;;PRINT ALL DATA OCTAL
```

```
001534
1151
1152 001534 044766
1153 001536 050307
1154 001540 053052
1155 001542 000000
1156
1157
1158 001544 045022
1159 001546 050350
1160 001550 053064
1161 001552 000000
1162
1163
1164 001554 045070
1165 001556 050417
1166 001560 053100
1167 001562 000000
1168
1169
1170 001564 045121
1171 001566 050463
1172 001570 053112
1173 001572 000000
1174
1175
1176 001574 045152
1177 001576 050532
1178 001600 053126
1179 001602 000000
1180
1181
1182 001604 045203
1183 001606 050576
1184 001610 053140
1185 001612 000000
1186
1187
1188 001614 045251
1189 001616 050417
1190 001620 053100
1191 001622 000000
```

```
1192
1193 001624 045303
1194 001626 050417
1195 001630 053100
1196 001632 000000
1197
1198
1199 001634 045346
1200 001636 050463
1201 001640 053112
1202 001642 000000
1203
1204
1205 001644 045400
1206 001646 050463
1207 001650 053112
1208 001652 000000
1209
1210
1211 001654 045450
1212 001656 050652
1213
1214 001660 053154
1215 001662 000000
1216
1217
1218 001664 045537
1219 001666 050652
1220
1221 001670 053154
1222 001672 000000
1223
1224
1225 001674 045563
1226 001676 050765
1227
1228 001700 053172
1229 001702 000000
1230
1231
1232 001704 045607
1233 001706 051100
1234 001710 053210
1235 001712 000000
1236
1237
1238 001714 045676
1239 001716 051100
1240 001720 053210
1241 001722 000000
1242
1243
1244 001724 045744
1245 001726 051100
1246 001730 053210
1247 001732 000000

;ITEM 10
      .WORD EM11 ;ATTEMPT TO SET ALL WCR BITS FAILED
      .WORD DH4  ;TEST # ERR PC WCRADR WCRCONTENTS
      .WORD DT4  ;$TESTN,$ERRPC,WCR,RWCR,0
      .WORD 0    ;PRINT ALL DATA OCTAL

;ITEM 11
      .WORD EM12 ;RESET FAILED TO CLEAR BAR
      .WORD DH5  ;TEST # ERR PC BARADR BAREXP BARRCV
      .WORD DT5  ;$TESTN,$ERRPC,BAR,EBAR,RBAR,0
      .WORD 0    ;PRINT ALL DATA OCTAL

;ITEM 12
      .WORD EM13 ;ATTEMPT TO SET ALL BAR BITS TO 1 FAILED
      .WORD DH5  ;TEST # ERR PC BARADR BAREXP BARRCV
      .WORD DT5  ;$TESTN,$ERRPC,BAR,EBAR,RBAR,0
      .WORD 0    ;PRINT ALL DATA OCTAL

;ITEM 13
      .WORD EM14 ;CSR BIT TEST FAILED (FATAL - DIAGNOSTIC NOT CONTINUED)
      .WORD DH14 ;BIT(S)
      .WORD 0    ;TEST # ERR PC TESTED CSRADR CSREXP CSRCONTENTS
      .WORD DT14 ;$TESTN,$ERRPC,BUT,CSR,ECSR,RCSR,0
      .WORD 0    ;PRINT ALL DATA OCTAL

;ITEM 14
      .WORD EM15 ;CSR BIT TEST FAILED
      .WORD DH14 ;BIT(S)
      .WORD 0    ;TEST # ERR PC TESTED CSRADR CSREXP CSRCONTENTS
      .WORD DT14 ;$TESTN,$ERRPC,BUT,CSR,ECSR,RCSR,0
      .WORD 0    ;PRINT ALL DATA OCTAL

;ITEM 15
      .WORD EM16 ;EIR BIT TEST FAILED
      .WORD DH16 ;BIT(S)
      .WORD 0    ;TEST # ERR PC TESTED EIRADR EIREXP EIRCONTENTS
      .WORD DT16 ;$TESTN,$ERRPC,BUT,CSR,EEIR,REIR,0
      .WORD 0    ;PRINT ALL DATA OCTAL

;ITEM 16
      .WORD EM17 ;READY AND MAINTENANCE ARE NOT THE ONLY BITS SET IN CSR
      .WORD DH17 ;TEST # ERR PC CSRADR CSREXP CSRCONTENTS
      .WORD DT17 ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
      .WORD 0    ;PRINT ALL DATA OCTAL

;ITEM 17
      .WORD EM20 ;ATTN AND ERROR FAILED TO SET PROPERLY
      .WORD DH17 ;TEST # ERR PC CSRADR CSREXP CSRCONTENTS
      .WORD DT17 ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
      .WORD 0    ;PRINT ALL DATA OCTAL

;ITEM 20
      .WORD EM21 ;ATTN AND ERROR FAILED TO CLEAR PROPERLY
      .WORD DH17 ;TEST # ERR PC CSRADR CSREXP CSRCONTENTS
      .WORD DT17 ;$TESTN,$ERRPC,CSR,ECSR,RCSR,0
      .WORD 0    ;PRINT ALL DATA OCTAL
```

1248					
1249	001734	046014	:ITEM 21	.WORD EM22	:ERROR BIT SHOULD HAVE BEEN CLEAR
1250	001736	051100		.WORD DH17	:TEST # ERR PC CSRADR CSREXP CSRCONTENTS
1251	001740	053210		.WORD DT17	:\$TESTN,\$ERRPC,CSR,ECSR,RCSR,0
1252	001742	000000		.WORD 0	:PRINT ALL DATA OCTAL
1253					
1254			:ITEM 22		
1255	001744	046124		.WORD EM24	:READY OF CSR WAS NOT SET
1256	001746	051100		.WORD DH17	:TEST # ERR PC CSRADR CSREXP CSRCONTENTS
1257	001750	053210		.WORD DT17	:\$TESTN,\$ERRPC,CSR,ECSR,RCSR,0
1258	001752	000000		.WORD 0	:PRINT ALL DATA OCTAL
1259					
1260			:ITEM 23		
1261	001754	046155		.WORD EM25	:BIT 0 OF THE BAR WAS SET
1262	001756	050463		.WORD DH5	:TEST # ERR PC BARADR BAREXP BARRCV
1263	001760	053112		.WORD DT5	:\$TESTN,\$ERRPC,BAR,EBAR,RBAR,0
1264	001762	000000		.WORD 0	:PRINT ALL DATA OCTAL
1265					
1266			:ITEM 24		
1267	001764	046302		.WORD EM30	:FUNCTION BIT(S) ARE NOT CLEAR
1268	001766	051100		.WORD DH17	:TEST # ERR PC CSRADR CSREXP CSRCONTENTS
1269	001770	053210		.WORD DT17	:\$TESTN,\$ERRPC,CSR,ECSR,RCSR,0
1270	001772	000000		.WORD 0	:PRINT ALL DATA OCTAL
1271					
1272			:ITEM 25		
1273	001774	046340		.WORD EM31	:DSTAT A, B OR C ARE NOT AS EXPECTED
1274	001776	051100		.WORD DH17	:TEST # ERR PC CSRADR CSREXP CSRCONTENTS
1275	002000	053210		.WORD DT17	:\$TESTN,\$ERRPC,CSR,ECSR,RCSR,0
1276	002002	000000		.WORD 0	:PRINT ALL DATA OCTAL
1277					
1278			:ITEM 26		
1279	002004	046404		.WORD EM32	:BDR IS NOT CLEAR
1280	002006	050532		.WORD DH6	:TEST # ERR PC BDRADR BDRCONTENTS
1281	002010	053126		.WORD DT6	:\$TESTN,\$ERRPC,BDR,RBDR,0
1282	002012	000000		.WORD 0	:PRINT ALL DATA OCTAL
1283					
1284			:ITEM 27		
1285	002014	046425		.WORD EM33	:ALL BDR BITS ARE NOT SET
1286	002016	050532		.WORD DH6	:TEST # ERR PC BDRADR BDRCONTENTS
1287	002020	053126		.WORD DT6	:\$TESTN,\$ERRPC,BDR,RBDR,0
1288	002022	000000		.WORD 0	:PRINT ALL DATA OCTAL
1289					
1290			:ITEM 30		
1291	002024	046516		.WORD EM35	:BDR SHOULD NOT HAVE BEEN LOADED WITH NEW PATTERN
1292	002026	051304		.WORD DH34	:TEST # ERR PC BDRADR BDREXP BDRCONTENTS
1293	002030	053254		.WORD DT34	:\$TESTN,\$ERRPC,BDR,EBDR,RBDR,0
1294	002032	000000		.WORD 0	:PRINT ALL DATA OCTAL
1295					
1296			:ITEM 31		
1297	002034	046577		.WORD EM36	:BDR PATTERN NOT CORRECT
1298	002036	051304		.WORD DH34	:TEST # ERR PC BDRADR BDREXP BDRCONTENTS
1299	002040	053254		.WORD DT34	:\$TESTN,\$ERRPC,BDR,EBDR,RBDR,0
1300	002042	000000		.WORD 0	:PRINT ALL DATA OCTAL

[illegible]

1354			;ITEM 43		
1355	002154	047374	.WORD	EM52	;EXPECTED INTERRUPT DID NOT OCCUR
1356	002156	051360	.WORD	DH43	;TEST # ERR PC CSRADR CSRCONTENTS
1357	002160	053270	.WORD	DT43	;\$TESTN,\$ERRPC,CSR,RCSR,0
1358	002162	000000	.WORD	0	;PRINT ALL DATA OCTAL
1359					
1360			;ITEM 44		
1361	002164	047463	.WORD	EM54	;BAR IS WRONG
1362	002166	051230	.WORD	DH26	;TEST # ERR PC BARADR BAREXP BARCONTENTS
1363	002170	053240	.WORD	DT26	;\$TESTN,\$ERRPC,BAR,EBAR,RBAR,
1364	002172	000000	.WORD	0	;PRINT ALL DATA OCTAL
1365					
1366			;ITEM 45		
1367	002174	047500	.WORD	EM55	;BAD DATA IN BDR
1368	002176	051304	.WORD	DH34	;TEST # ERR PC BDRADR BDREXP BDRCONTENTS
1369	002200	053254	.WORD	DT34	;\$TESTN,\$ERRPC,BDR,EBDR,RBDR,
1370	002202	000000	.WORD	0	;PRINT ALL DATA OCTAL
1371					
1372			;ITEM 46		
1373	002204	047556	.WORD	EM57	;BUFFER DATA NOT CORRECT
1374	002206	051602	.WORD	DH57	;CHECK CHECK INPUT INPUT
1375					;TEST # ERR PC BUFADR BUFDAT BUFADR BUFDAT CSRADR
1376	002210	053342	.WORD	DT57	;\$TESTN,\$ERRPC,\$TMP4,\$TMP2,\$TMP5,\$TMP3,CSR,0
1377	002212	000000	.WORD	0	;PRINT ALL DATA OCTAL
1378					
1379			;ITEM 47		
1380	002214	047606	.WORD	EM60	;TOO MANY WORDS WERE TRANSFERED
1381	002216	051747	.WORD	DH60	;DIDNOT
1382					;TEST # ERR PC EXPECT ADRESS CSRADR
1383	002220	053362	.WORD	DT60	;\$TESTN,\$ERRPC,\$TMP2,\$TMP3,CSR,0
1384	002222	000000	.WORD	0	;PRINT ALL DATA OCTAL
1385					
1386			;ITEM 50		
1387	002224	047645	.WORD	EM61	;UNEXPECTED TRAP OR INTERRUPT TO TRAP ADDRESS BELOW
1388	002226	052045	.WORD	DH61	;TEST # ERR PC WCRADR OLDPC TRAP ADR
1389	002230	053376	.WORD	DT61	;\$TESTN,\$ERRPC,WCR,OLDPC2,BDVECT,0
1390	002232	000000	.WORD	0	;PRINT ALL DATA OCTAL
1391					
1392			;ITEM 51		
1393	002234	047730	.WORD	EM62	;CSR AND-OR WCR AND-OR BAR ARE INCORECT
1394	002236	052116	.WORD	DH62	;TEST # ERR PC WCRADR WCREXP WCRRCV CSREXP CSRRCV BAREXP BAR
1395	002240	053412	.WORD	DT62	;\$TESTN,\$ERRPC,WCR,EWCR,RWCR,ELSR,RCSR,EBAR,RBAR,0
1396	002242	000000	.WORD	0	;PRINT ALL DATA OCTAL
1397					
1398			;ITEM 52		
1399	002244	047777	.WORD	EM63	;DR11 INTERRUPTED AT WRONG LEVEL
1400	002246	052225	.WORD	DH63	;TEST # ERR PC EXPLVL RCVLVL CSRADR
1401	002250	053436	.WORD	DT63	;\$TESTN,\$ERRPC,DRLEV,LEVEL,CSR,0
1402	002252	000000	.WORD	0	;PRINT ALL DATA OCTAL
1403					
1404			;ITEM 53		
1405	002254	047017	.WORD	EM43	;DR11 FAILED TO INTERRUPT
1406	002256	052274	.WORD	DH64	;TEST # ERR PC EXPLVL CSRADR
1407	002260	053452	.WORD	DT64	;\$TESTN,\$ERRPC,\$TMP1,CSR,0
1408	002262	000000	.WORD	0	;PRINT ALL DATA OCTAL

1409

1410 002264 050037
1411 002266 052333
1412 002270 053464
1413 002272 000000

;ITEM 54

.WORD EM65
.WORD DH65
.WORD DT65
.WORD 0

;2-N CYCLE BURST SWITCH IN WRONG POSITION
;TEST # ERR PC CSRADR EIREXP EIRRCV
;\$TESTN,\$ERRPC,CSR,EEIR,REIR,0
;PRINT ALL DATA OCTAL

1414 002274	ER200:		;THIS IS THE STARTING POINT FOR ERROR MESSAGES 201
1415			;THROUGH 277. THEY ARE USED FOR MULTIPLE ERROR MESSAGES.
1416	:ITEM 201		
1417 002274 047556	.WORD	EM57	;BUFFER DATA NOT CORRECT
1418 002276 051602	.WORD	DH57	;CHECK CHECK INPUT INPUT
1419			;TEST # ERR PC BUFADR BUFDAT BUFADR BUFDAT CSRADR
1420 002300 053342	.WORD	DT57	;\$TESTN,\$ERRPC,\$TMP4,\$TMP2,\$TMP5,\$TMP3,CSR,0
1421 002302 000000	.WORD	0	;PRINT ALL DATA OCTAL
1422			
1423	:ITEM 202		
1424 002304 050153	.WORD	EM202	;CSR PATTERN NOT CORRECT
1425 002306 052451	.WORD	DH202	;TEST # ERR PC CSRADR PATLDD CSREXP CSRRCV
1426 002310 053514	.WORD	DT202	;\$TESTN,\$ERRPC,CSR,BUT,ECSR,RCSR,0
1427 002312 000000	.WORD	0	;PRINT ALL DATA OCTAL
1428			
1429	:ITEM 203		
1430 002314 046206	.WORD	EM26	;BIT PATTERN TEST FAILED IN BAR
1431 002316 051230	.WORD	DH26	;TEST # ERR PC BARADR BAREXP BARCONTENTS
1432 002320 053240	.WORD	DT26	;\$TESTN,\$ERRPC,BAR,EBAR,RBAR,
1433 002322 000000	.WORD	0	;PRINT ALL DATA OCTAL
1434			
1435	:ITEM 204		
1436 002324 046245	.WORD	EM27	;WCR DATA PATTERN NOT CORRECT
1437 002326 051154	.WORD	DH23	;TEST # ERR PC WCRADR WCREXP WCRCONTENTS
1438 002330 053224	.WORD	DT23	;\$TESTN,\$ERRPC,WCR,EWCR,RWCR,0
1439 002332 000000	.WORD	0	;PRINT ALL DATA OCTAL
1440			
1441	:ITEM 205		
1442 002334 046577	.WORD	EM36	;BDR PATTERN NOT CORRECT
1443 002336 051304	.WORD	DH34	;TEST # ERR PC BDRADR BDREXP BDRCONTENTS
1444 002340 053254	.WORD	DT34	;\$TESTN,\$ERRPC,BDR,EBDR,RBDR,0
1445 002342 000000	.WORD	0	;PRINT ALL DATA OCTAL
1446			
1447	:ITEM 206		
1448 002344 047435	.WORD	EM53	;WCR NOT EQUAL TO ZERO
1449 002346 052670	.WORD	DH210	;TEST # ERR PC WCRADR WCRCONTENTS
1450 002350 053564	.WORD	DT210	;\$TESTN,\$ERRPC,WCR,RWCR,0
1451 002352 000000	.WORD	0	;PRINT ALL DATA OCTAL
1452			
1453	:ITEM 207		
1454 002354 047463	.WORD	EM54	;BAR IS WRONG
1455 002356 051230	.WORD	DH26	;TEST # ERR PC BARADR BAREXP BARCONTENTS
1456 002360 053240	.WORD	DT26	;\$TESTN,\$ERRPC,BAR,EBAR,RBAR,
1457 002362 000000	.WORD	0	;PRINT ALL DATA OCTAL
1458			
1459	:ITEM 210		
1460 002364 047520	.WORD	EM56	;DATA NOT TRANSFERRED CORRECTLY
1461 002366 051523	.WORD	DH56	;TEST # ERR PC NPR1AD NPR1EX NPR1RC CSRADR
1462 002370 053324	.WORD	DT56	;\$TESTN,\$ERRPC,ANPR1,ENPR1,NPR1,CSR,0
1463 002372 000000	.WORD	0	;PRINT ALL DATA OCTAL
1464			
1465	:ITEM 211		
1466 002374 050203	.WORD	EM211	;BDR AND-OR WCR AND-OR BAR ARE INCORRECT
1467 002376 052734	.WORD	DH211	;TEST # ERR PC WCRADR WCREXP WCRRCV BDREXP BDRRCV BAREXP BAR
1468 002400 053576	.WORD	DT211	;\$TESTN,\$ERRPC,WCR,EWCR,RWCR,ECSR,RCSR,EB,RBAR,0
1469 002402 000000	.WORD	0	;PRINT ALL DATA OCTAL

1470

1471 002404 047156

1472 002406 051100

1473 002410 053210

1474 002412 000000

;ITEM 212

.WORD EM46

.WORD DH17

.WORD DT17

.WORD 0

;FUNCTION BITS DIDN'T INCREMENT IN MAINT MODE

;TEST # ERR PC CSRADR CSREXP CSRCONTENTS

;\$TESTN,\$ERRPC,CSR,ECSR,RCSR,0

;PRINT ALL DATA OCTAL

1475
1476
1477
1478
1479
1480
1481
1482 002414 000001
1483
1484 002416
1485 002456
1486
1487
1488
1489
1490
1491
1492 002516 000000
1493 002520 000000
1494 002522 000000
1495 002524 000000
1496 002526 000000
1497 002530 000000
1498 002532 000000
1499 002534 000000
1500
1501
1502
1503 002536 000000
1504 002540 000000
1505 002542 000000
1506 002544 000000
1507 002546 000000
1508 002550 000000
1509 002552 000000
1510 002554 000000
1511 002556 000000
1512
1513 002560 000000
1514 002562 000000
1515 002564 000000
1516 002566 000000
1517 002570 000000
1518
1519 002572 000000
1520 002574 000000
1521 002576 000000
1522 002600 000000
1523 002602 000000
1524 002604 000000
1525 002606 002612
1526 002610 000000
1527 002612 052525
1528 002614 173000
1529 002616 040450
1530 002620 041452
1531 002622 000000

```
.SBTTL  STORAGE LOCATIONS
*****
:
:      STORAGE LOCATIONS
:
*****
QTYBRD: .WORD  1      ;TOTAL # DR11 BOARDS BEING TESTED (DEFAULT = 1)
REGADR: .BLKW  16.    ;TOTAL: 16 LOCATIONS FOR BOARD ADDRESSES
VECADR: .BLKW  16.    ;TOTAL: 16 LOCATIONS FOR VECTOR ADDRESSES

;REGISTER AND VECTOR ADDRESS STORAGE LOCATIONS FOR THE DR11 UNDER TEST

;*****
;DO NOT INSERT ANY ITEMS BETWEEN ANY OF THE LOCATIONS BELOW
;*****
WCR:    .WORD  0
BAR:    .WORD  0
CSR:    .WORD  0
BDR:    .WORD  0
DRINV:  .WORD  0
DRVS:   .WORD  0
SDRINV: .WORD  0
SDRVS:  .WORD  0
;*****
;DO NOT INSERT ANY ITEMS BETWEEN ANY OF THE LOCATIONS ABOVE
;*****
BUT:    .WORD  0      ;BIT(S) UNDER TEST LOCATION
LEVEL:  .WORD  0      ;BR LEVEL LOCATION
BDVECT: .WORD  0
DEVMSK: .WORD  0
TABINX: .WORD  0
DREG:   .WORD  0
DRLEV:  .WORD  0
NXTTST: .WORD  0
PASCNT: .WORD  0

RCSR:   .WORD  0      ;CSR ACTUALLY READ FROM DEVICE UNDER TEST
REIR:   .WORD  0      ;EIR ACTUALLY READ FROM DEVICE UNDER TEST
RBDR:   .WORD  0      ;BDR ACTUALLY READ FROM DEVICE UNDER TEST
RBAR:   .WORD  0      ;BAR ACTUALLY READ FROM DEVICE UNDER TEST
RWCR:   .WORD  0      ;WCR ACTUALLY READ FROM DEVICE UNDER TEST

ECSR:   .WORD  0      ;CSR EXPECTED
EEIR:   .WORD  0      ;EIR EXPECTED
EBDR:   .WORD  0      ;BDR EXPECTED
EBAR:   .WORD  0      ;BAR EXPECTED
EWCR:   .WORD  0      ;WCR EXPECTED
ENPR1:  .WORD  0      ;EXPECTED OF NPR1
ANPR1:  .WORD  NPR1   ;ADDRESS OF NPR1
BORW:   .WORD  0
NPR1:   .WORD  52525
DIOMEM: .WORD  173000
INBUF:  .WORD  XINBUF
CHKBUF: .WORD  XCHKBU
BUFLN:  .WORD  0
```

1532	002624	000000	LENCHK: .WORD	0	
1533	002626	000000	BRWAIT: .WORD	0	
1534	002630	000000	WCLEN: .WORD	0	
1535	002632	000000	RDYCHK: .WORD	0	
1536	002634	177560	TKS: .WORD	177560	
1537	002636	177562	TKB: .WORD	177562	
1538	002640	177564	TPS: .WORD	177564	
1539	002642	177566	TPB: .WORD	177566	
1540	002644	000000	MSG: .WORD	0	
1541	002646	000000	ADDR: .WORD	0	
1542	002650	000000	MESSAG: .WORD	0	
1543	002652	000000	FLAG: .WORD	0	
1544	002654	000000	FNCNT: .WORD	0	
1545	002656	000000	INBUF1: .WORD	0	
1546	002660	000000	TIME: .WORD	0	;GENERAL PURPOSE TIMER
1547	002662	000000	LOOP: .WORD	0	;GENERAL PURPOSE LOOP COUNTER
1548	002664	000000	ANSWER: .WORD	0	
1549	002666	000000	BDFAIL: .WORD	0	
1550	002670	000000	MANSIZ: .WORD	0	
1551	002672	000000	OLDPC1: .WORD	0	;LOCATION TO STORE RETURN PC IN SUBROUTINES WITH ERROR CALLS
1552	002674	000000	OLDPS1: .WORD	0	;LOCATION TO STORE PS
1553	002676	000000	OLDPC2: .WORD	0	;LOCATION TO STORE RETURN PC IN SUBROUTINES WITH ERROR CALLS
1554	002700	000000	OLDPS2: .WORD	0	;LOCATION TO STORE PS
1555	002702	000000	OFL: .WORD	0	;FIRST CHAR FLAG
1556	002704	000000	IRGSTC: .WORD	0	;LOCATION FOR LARGEST NUMBER CHARACTER FOR THE READ SUBROUTINE
1557	002706	000000	EOPLOC: .WORD	0	;LOCATION TO HOLD FLAG DECIDING IF EOP MSGS ARE TO BE PRINTED
1558	002710	000000	BITTST: .WORD	0	;LOCATION TO PUT THE BIT STATE TO PRINT - USED BY SUBROUTINE PSTATE
1559	002712	000000	MEMGMT: .WORD	0	;LOCATION TO HOLD FLAG SAYING MEMORY MANAGEMENT IS AVAILABLE
1560	002714	000000	ERRCNT: .WORD	0	;LOCATION TO HOLD TOTAL ERRORS FOR A PARTICULAR TEST
1561	002716	000000	\$PASS2: .WORD	0	;LOCATION TO HOLD OVERFLOW PASS NUMBER

```
1562 .SBTTL DEVICE DESCRIPTOR WORD BIT DESCRIPTION
1563 :*****
1564 : DESCRIPTION OF BITS IN THE DDW (DEVICE DESCRIPTOR WORD):
1565 :
1566 : BIT 0 DR11-W=0, DR11-B=1
1567 : BIT 1 2 CYCLE=0, N CYCLE=1
1568 : BIT 2 CABLE DOESN'T EXIST=0, CABLE DOES EXIST=1
1569 : BIT 5 \
1570 : BIT 6 > BR PRIORITY
1571 : BIT 7 /
1572
1573 002720 000000 DDW: .WORD 0 ;LOCATION FOR STORAGE OF THE DEVICE DESCRIPTOR WORD
```



```

1574 .C3TTL SUBROUTINE TO INPUT A CHARACTER OR UP TO A 6 DIGIT NUMBER
1575 *****
1576 *
1577 * THIS SUBROUTINE IS USED IN THE EDIT ROUTINE TO INPUT NUMBERS AND A
1578 * SINGLE CHARACTER. R3 IS TO BE LOADED WITH THE NUMBER OF DIGITS
1579 * EXPECTED. THIS SUBROUTINE WILL EXIT IF A NON-NUMERIC CHARACTER IS
1580 * INPUTED, LEAVING THE CHARACTER IN LOCATION 'ANSWER'. IF A NUMERIC
1581 * CHARACTER IS INPUTED, IT WILL CLEAR ALL BUT THE 1ST 4 BITS EXPOSING
1582 * THE VALUE OF THE DIGIT INPUTED, AND ADD IT TO R4, WHICH WAS CLEARED
1583 * AT THE BEGINING OF THIS SUBROUTINE. LOCATION 'LRGSTC' IS TO BE LOADED
1584 * WITH THE LARGEST ASCII CHARACTER ACCEPTABLE FOR THIS NUMBER, I.E. 7
1585 * OR 9 (FOR OCTAL OR DECIMAL INPUT RESPECTIVELY). IT WILL ONLY ACCEPT
1586 * THE NUMBER DIGIT EQUAL TO OR LESS THAN THIS DIGIT. IT WILL ONLY ACCEPT
1587 * THE MAXIMUM NUMBER OF DIGITS SPECIFIED BY R3, PRINTING A <CRLF> WHEN
1588 * THAT LIMIT IS REACHED. IF A <CR> IS INPUTED BEFORE THE MAXIMUM IS
1589 * REACHED, ROUTINE EXITS LEAVING THE INPUTED NUMBER IN R4.
1590 *****
1591 READ: CLR R4 ;CLEAR THE CHARACTER RECEIVER
1592 002722 005004 TSTB CHARCT ;SEE IF A CHARACTER WAS INPUTED DURING PRINTING
1593 002724 105737 030707 BEQ 1$ ;BRANCH TO INPUT A CHARACTER IF NOT
1594 002730 001406 MOV B CHARCT,ANSWER ;MOVE THE CHARACTER TO THE ANSWER LOCATION
1595 002732 113737 030707 002664 CLRB CHARCT ;CLEAR THAT SUCKER
1596 002740 105037 030707 BR 2$ ;GO CHECK IT OUT, YOU DUMMY
1597 002744 000403 1$: RDCHR ;GET A CHARACTER
1598 002746 104411 MOV (SP)+,ANSWER ;POP INPUTED CHARACTER OFF STACK
1599 002750 012637 002664 2$: CMP #CNTLC,ANSWER ;SEE IF A ^C WAS INPUTED
1600 002754 022737 000003 002664 BNE 3$ ;BRANCH AROUND ITS PRINTING IF NOT
1601 002762 001003 TYPE ,CNTRLC ;TYPE: '^C'
1602 002764 104401 035073 BR 6$ ;KICK OUT OF THIS ROUTINE
1603 002770 000453 CMP #ESC,ANSWER ;SEE IF AN <ESC> WAS INPUTED
1604 002772 022737 000033 002664 3$: BNE 4$ ;BRANCH AROUND ITS PRINTING IF NOT
1605 003000 001003 TYPE ,ESCAPE ;TYPE: '<ESC>'
1606 003002 104401 034762 BR 6$ ;KICK OUT OF THIS ROUTINE
1607 003006 000444 MOV B ANSWER,LETNCR ;MOVE CHARACTER FOR PRINTING
1608 003010 113737 002664 035071 4$: TYPE ,LETNCR ;GO TYPE THE INPUTED CHARACTER
1609 003016 104401 035071 CMP #'/,ANSWER ;SEE IF A NON-NUMERIC/ALPH CHARACTER WAS INPUTED
1610 003022 022737 000057 002664 BPL 5$ ;BRANCH TO R4 TEST IF SO
1611 003030 100016 CMPB LRGSTC,ANSWER ;SEE IF A NON-OCTAL/NUMERIC CHARACTER WAS INPUTED
1612 003032 123737 002704 002664 BMI 6$ ;BRANCH TO EXIT IF SO
1613 003040 100427 MOV ANSWER,-(SP) ;MOVE ASCII TO STACK FOR PREPARATION
1614 003042 013746 002664 BIC #60,(SP) ;CLEAR ALL BUT THE NUMBER INPUTED
1615 003046 042716 000060 ASL R4 ;SHIFT R4 THREE PLACES
1616 003052 006304 ASL R4 ;TO MAKE ROOM FOR
1617 003054 006304 ASL R4 ;THIS CHARACTER
1618 003056 006304 ADD (SP)+,R4 ;ADD THE OCTAL NUMBER TO R4
1619 003060 062604 DEC R3 ;SUBTRACT 1 FROM THE LOOP COUNTER AND
1620 003062 005303 BNE 1$ ;BRANCH BACK IF NOT ALL CHARACTERS INPUTED
1621 003064 001330 CMPB #'9,LRGSTC ;SEE IF NUMBER IS TO BE DECIMAL
1622 003066 122737 000071 002704 5$: BNE 6$ ;BRANCH IF NOT
1623 003074 001011 CMP #7,R4 ;SEE IF INPUTED NUMBER IS 7 OR LESS
1624 003076 022704 000007 BPL 6$ ;BRANCH IF SO
1625 003102 100006 CMP #CARETN,ANSWER ;SEE IF CARRIAGE RETURN WAS INPUTED
1626 003104 022737 000015 002664 BEQ 6$ ;BRANCH IF SO - R4 IS CORRECT
1627 003112 001402 ADD #2,R4 ;ADD 2 TO R4 TO MAKE OCTAL NUMBER THE DECIMAL EQUIVALENT
1628 003114 062704 000002 6$: TYPE ,<CRLF> ;PRINT A <CRLF>
1629 003120 104401 001405 RTS ;EXIT
1630 003124 000207 PC

```



```

1648 .SBITL SUBROUTINE TO PRINT THE DATA STORED BY SUBROUTINE ERCAPT
1649 *****
1650 *
1651 * THIS SUBROUTINE IS INVOKED BY ENTERING (^Y) DURING THE EXECUTION OF THE
1652 * TEST. IT PRINTS ALL DATA STORED IN THE 'CAPSTK' STACK STORED BY THE
1653 * 'ERCAPT' SUBROUTINE, REINITIALIZES THE POINTER, AND RETURNS. *DATA*
1654 *PRINTED*IS*LOST* BECAUSE OF THE REINITIALIZATION OF THE COUNTER.
1655 *
1656 *****
1657 003170 022737 042456 042454 PTCAPT: CMP #CAPSTK,CAPNTR ;SEE IF THERE IS ANY DATA TO PRINT
1658 003176 001003 BNE 1$ ;BRANCH TO PRINT IT IF THERE IS
1659 003200 104401 034211 TYPE ,NODATA ;TYPE: 'NO ERROR TOTALS TO REPORT'
1660 003204 000462 BR 7$ ;KICK OUT
1661 003206 022737 044736 042454 1$: CMP #ENDSTK,CAPNTR ;SEE IF STACK IS FULL OF BULL
1662 003214 001002 BNE 2$ ;BRANCH AROUND STACK IS FULL MESSAGE IF NOT
1663 003216 104401 034020 TYPE ,STKIFL ;TYPE: 'STACK IS FULL - DATA MAY HAVE BEEN LOST'
1664 003222 104401 034073 2$: TYPE ,ERCHDR ;TYPE: 'SUMMATION OF ERRORS SINCE BEGINNING OR LAST REPORT'
1665 ; 'BOARD # PASS # ERRITL'
1666 003226 012700 042456 MOV #CAPSTK,RO ;MOVE ADDRESS OF STACK TO PRINT TO RO
1667 003232 012701 000020 MOV #16,R1 ;MOVE 16 BOARDS TO SEARCH TO RO
1668 003236 005037 001362 CLR $TMP1 ;CLEAR $TMP1, DEVICE POINTER
1669 003242 021037 001362 3$: CMP (RO),$TMP1 ;SEE IF UNIT NUMBER IS TO PRINT
1670 003246 001403 BEQ 4$ ;BRANCH TO PRINT DATA IF SO
1671 003250 062700 000010 ADD #10,RO ;GO TO NEXT SET OF DATA AND
1672 003254 000420 BR 6$ ;GO SEE IF ANY MORE TO CHECK
1673 003256 012046 4$: MOV (RO)+,-(SP) ;MOVE UNIT NUMBER TO STACK FOR PRINTING
1674 003260 104405 TYPDS ;GO TYPE UNIT NUMBER IN DECIMAL
1675 003262 104401 035250 TYPE ,SPACES ;TYPE 2 SPACES
1676 003266 012046 MOV (RO)+,-(SP) ;MOVE OVERFLOW PASS NUMBER TO STACK FOR PRINTING
1677 003270 001002 BNE 5$ ;BRANCH IF NON-ZERO
1678 003272 104401 035250 TYPE ,SPACES ;TYPE AN EXTRA 2 SPACES - NUMBER WILL NOT BE EXTENDED
1679 003276 012046 5$: MOV (RO)+,-(SP) ;MOVE PASS NUMBER TO STACK FOR PRINTING
1680 003300 104406 TYPDE ;GO TYPE PASS NUMBER IN EXTENDED DECIMAL
1681 003302 104401 035250 TYPE ,SPACES ;TYPE 2 SPACES
1682 003306 012046 MOV (RO)+,-(SP) ;MOVE ERROR TOTAL TO STACK FOR PRINTING
1683 003310 104405 TYPDS ;GO TYPE ERROR TOTAL IN DECIMAL
1684 003312 104401 001405 TYPE ,$CRLF ;TYPE <CRLF>
1685 003316 020037 042454 6$: CMP RO,CAPNTR ;SEE IF ALL DATA HAS BEEN PRINTED
1686 003322 001347 BNE 3$ ;BRANCH BACK IF NOT
1687 003324 005237 001362 INC $TMP1 ;INCREMENT DEVICE POINTER
1688 003330 012700 042456 MOV #CAPSTK,RO ;INITIALIZE TO BEGINNING FOR ANOTHER POSSIBLE PASS
1689 003334 005301 DEC R1 ;DECREMENT THE LOOP COUNTER AND
1690 003336 001341 BNE 3$ ;BRANCH UNTIL ALL DEVICE DATA PRINTED
1691 003340 012737 042456 042454 MOV #CAPSTK,CAPNTR ;REINITIALIZE CAPNTR
1692 003346 104401 001405 TYPE ,$CRLF ;TYPE ANOTHER <CRLF>
1693 003352 105037 030707 7$: CLRB CHARCT ;CLEAR ANY CHARACTER ENTERED DURING PRINTING
1694 003356 000207 RTS PC ;EXIT

```

```

1695
1696
1697
1698
1699
1700
1701
1702
1703 003360 005037 001466
1704 003364 000261
1705 003366 006137 001466
1706 003372 022737 000001 002414
1707 003400 001433
1708 003402 013701 002414
1709 003406 005301
1710 003410 005002
1711 003412 012703 000002
1712 003416 000261
1713 003420 006137 001466
1714 002416
1715 003424 016263 002416 002416
1716 003432 062763 000010 002416
1717 002456
1718 003440 016263 002456 002456
1719 003446 062763 000010 002456
1720 003454 013763 001474 001474
1721 022223
1722 003464 005301
1723 003466 001353
1724 003470 000207

.SBTTL SUBROUTINE TO FILL ALL TABLE BOARD ENTRIES
*****
;
; THIS SUBROUTINE FILLS ALL TABLE BOARD ENTRIES FOR THE ADDRESSES AND
; VECTORS FROM THE VALUES IN 'REGADR' AND 'VECADR', AND SHOULD BE SET
; UP BEFORE ENTERING THIS SUBROUTINE.
;
*****
FIXTBL: CLR $DEVM ;CLEAR THE DEVICE MASK
        SEC ;SET THE CARRY BIT AND
        ROL $DEVM ;ROTATE IT INTO $DEVM FOR 1 BOARD
        CMP #1,QTYBRD ;SEE IF ONLY 1 BOARD PRESENT
        BEQ 2$ ;KICK OUT IF SO - TABLE DOESN'T NEED FILLING
        MOV QTYBRD,R1 ;FILL ALL TABLE BOARD ENTRIES FROM FIRST
        DEC R1 ;DECREMENT SINCE 1ST POSITION IS ALREADY FILLED
        CLR R2 ;CLEAR INDEX TO SEND POINTER
        MOV #2,R3 ;PUT 2 IN INDEX TO RECEIVE POINTER
1$: SEC ;SET THE CARRY BIT AND
   ROL $DEVM ;ROTATE IT INTO $DEVM
   RA=REGADR ;REDEFINE REGADR AS RA FOR SPACE REASONS
   MOV RA(R2),RA(R3) ;TRANSFER ADDRESS TO NEXT POSITION AND
   ADD #10,RA(R3) ;ADD 10 FOR NEXT POSITION
   VA=VECADR ;REDEFINE VECADR AS VA FOR SPACE REASONS
   MOV VA(R2),VA(R3) ;TRANSFER VECTOR TO NEXT POSITION AND
   ADD #10,VA(R3) ;ADD 10 FOR NEXT POSITION
   MOV $DDWO,$DDWO(R3) ;MOVE DEVICE DESCRIPTOR WORD TO NEXT POSITION
   CMP (R2)+,(R3)+ ;UPDATE INDEX POINTERS
   DEC R1 ;DECREMENT THE LOOP COUNTER
   BNE 1$ ;BRANCH BACK IF NOT DONE
2$: RTS PC ;EXIT

```

1725				.SBTTL	SUBROUTINE TO LOAD INBUF WITH AN INCREMENTING PATTERN	
1726				*****		
1727				;		
1728				;	THIS SUBROUTINE CLEARS THE FIRST LOCATION OF THE BUFFER AND LOADS	
1729				;	NUMBERS STARTING WITH 1 INTO THE BUFFER.	
1730				;		
1731				*****		
1732	003472	013702	002616	LODBUF: MOV	INBUF,R2	;MOVE STARTING ADDRESS OF INBUF TO R2
1733	003476	013703	002622		MOV BUFLN,R3	;MOVE LOOP COUNTER TO R3
1734	003502	005203			INC R3	;CORRECT COUNTER
1735	003504	005001			CLR R1	;CLEAR THE LOADING COUNTER
1736	003506	010122		1\$: MOV	R1,(R2)+	;LOAD NEXT BUFFER WORD
1737	003510	005201			INC R1	;INCREMENT THE LOADING COUNTER
1738	003512	005303			DEC R3	;DECREMENT THE LOOP COUNTER AND
1739	003514	001374			BNE 1\$	;BRANCH BACK IF NOT DONE
1740	003516	000207			RTS PC	;EXIT

```

1741                                     .SBTTL  SUBROUTINE TO LOAD THE CHKBUFF WITH EVEN #'S STARTING WITH 0
1742                                     ;*****
1743                                     ;*
1744                                     ;*   THIS SUBROUTINE CLEARS THE FIRST LOCATION OF THE BUFFER AND LOADS
1745                                     ;*   EVEN NUMBERS STARTING WITH 0 INTO THE BUFFER.
1746                                     ;*
1747                                     ;*****
1748 003520 013702 002620 CHKBFF: MOV      CHKBUFF,R2      ; STARTING ADDRESS OF CHECK-BUFFER TO R2
1749 003524 013701 002622      MOV      BUFLN,R1      ; MOVE LOOP COUNTER TO R1
1750 003530 005003      CLR      R3      ; WIPE OUT R3
1751 003532 010322 1$:      MOV      R3,(R2)+      ; MOVE R3 TO CHKBUFF ADDRESS AND INC BY 2
1752 003534 010322      MOV      R3,(R2)+      ; MOVE R3 TO NEXT CHKBUFF ADDRESS AND INC BY 2
1753 003536 022341      CMP      (R3)+,-(R1)      ; ADD 2 TO NUMBER FOR BUFFER & SUBTRACT 2 FROM LOOP COUNTER
1754 003540 005701      TST      R1      ; SEE IF R1 HAS REACHED ZERO YET
1755 003542 001373      BNE      1$      ; BRANCH BACK IF NOT DONE
1756 003544 000207 2$:      RTS      PC      ; EXIT

```

```

1757 .SBTTL SUBROUTINE TO CLEAR IE, CHECK ERROR, READY, WCR=0, AND BAR
1758 *****
1759 :
1760 : THIS SUBROUTINE HAS THE NEED TO CALL AN ERROR IN THE TEST. THE ERROR
1761 : IS TO BE LOCATED IN THE TEST JUST AFTER THE JSR CALL. FUTURE USE OF
1762 : THIS SUBROUTINE MUST BE HANDLED AS FOLLOWS:
1763 :
1764 : JSR PC,INTA ;SUBROUTINE CALL
1765 : ERROR +51 ;ERROR CALL
1766 : CONTINUE ;SUBROUTINE RETURNS HERE IF NO ERROR
1767 :
1768 *****
1769 003546 042777 000100 176746 INTA: BIC #IE,@CSR ;CLEAR IE
1770 003554 013702 002622 MOV BUFLN,R2 ;BUFFER LENGTH TO R2
1771 003560 063702 002622 ADD BUFLN,R2 ;NUMBER OF XFERS TIMES 2
1772 003564 063702 002616 ADD INBUF,R2 ;CORRECT BAR
1773 003570 017737 176726 002560 MOV @CSR,RCSR ;MOVE RECEIVED DATA TO RCSR
1774 003576 032737 010000 002560 BIT #MA,RCSR ;SEE IF WE ARE IN MAINTENANCE MODE
1775 003604 001005 BNE 1$ ;BRANCH AROUND CABLE TEST IF WE ARE, BIT 0 WILL BE CLEAR
1776 003606 032737 000004 002720 BIT #BIT2,DDW ;SEE IF THERE IS A CABLE
1777 003614 001401 BEQ 1$ ;BRANCH IF NO CABLE
1778 003616 005202 INC R2 ;CABLE MODE TESTING LEAVES BIT 0 OF BAR SET. CHECK ODD ADRS
1779 003620 017737 176674 002566 1$: MOV @BAR,RBAR ;MOVE RECEIVED DATA TO RBAR
1780 003626 005037 002602 CLR EWR ;CLEAR EXPECTED LOCATION
1781 003632 010237 002600 MOV R2,EBAR ;MOVE EXPECTED DATA TO EBAR
1782 003636 013737 002560 002572 MOV RCSR,ECSR ;MOVE EXPECTED DATA TO ECSR
1783 003644 042737 100000 002572 BIC #ER,ECSR ;MAKE SURE ERROR BIT IS CLEAR
1784 003652 052737 000200 002572 BIS #RY,ECSR ;MAKE SURE READY BIT IS SET
1785 003660 017737 176632 002570 MOV @WCR,RWCR ;MOVE RECEIVED DATA TO RWCR
1786 003666 001012 BNE 2$ ;BRANCH TO RETURN WITHOUT UPDATING PC SO ERROR WILL CALL
1787 003670 023737 002560 002572 CMP RCSR,ECSR ;DOES CSR CONTAIN WHAT IT SHOULD
1788 003676 001006 BNE 2$ ;BRANCH TO RETURN WITHOUT UPDATING PC SO ERROR WILL CALL
1789 003700 023737 00256 002600 CMP RBAR,EBAR ;DOES BAR CONTAIN WHAT IT SHOULD
1790 003706 001002 BNE 2$ ;BRANCH TO RETURN WITHOUT UPDATING PC SO ERROR WILL CALL
1791 003710 062716 000002 ADD #2,(SP) ;CORRECT PC RETURN TO AFTER THE ERROR CALL
1792 003714 000207 2$: RTS PC ;EXIT

```

```

1793 .SBTTL SUBROUTINE TO CHECK INBUF AFTER A MAINTENANCE MODE OPERATION
1794 *****
1795 :
1796 : THIS SUBROUTINE HAS THE NEED TO CALL AN ERROR IN THE TEST AND RETURN
1797 : TO THE SUBROUTINE. THE ERROR AND RETURN JSR ARE TO BE LOCATED IN THE
1798 : TEST JUST AFTER THE JSR CALL. YOU *MUST* CLEAR LOCATION 'ERRCNT'
1799 : BEFORE EXECUTION OF THIS SUBROUTINE, OTHERWISE YOU MAY NOT GET ANY
1800 : ERRORS PRINTED, OR IF SO, JUST THE DATA WITHOUT THE HEADER. FUTURE
1801 : USE OF THIS SUBROUTINE MUST BE HANDLED AS FOLLOWS:
1802 :
1803 : JSR PC,DATCHK ;SUBROUTINE CALL
1804 : ERROR +201 ;ERROR CALL
1805 : JSR PC,DATCH2 ;RETURN TO SUBROUTINE AFTER ERROR RTS
1806 : CONTINUE ;SUBROUTINE RETURNS HERE IF NO ERROR(S) OR WHEN DONE
1807 :
1808 *****
1809 003716 011637 001362 DATCHK: MOV (SP), $TMP1 ;SAVE PC RETURN
1810 003722 013702 002620 MOV CHKBUFF, R2 ;STARTING ADDRESS OF CHECK BUFFER TO R2
1811 003726 013703 002616 MOV INBUF, R3 ;STARTING ADDRESS OF IN BUFFER TO R3
1812 003732 005037 002624 CLR LENCHK ;CLEAR LENGTH CHECK
1813 003736 005237 002624 DATCH1: INC LENCHK ;MAKE A COMPARISON
1814 003742 022223 CMP (R2)+, (R3)+ ;IS THE DATA CORRECT?
1815 003744 001423 BEQ DATCH2 ;BRANCH IF OK
1816 003746 013716 001362 MOV $TMP1, (SP) ;RESTORE ORIGINAL PC RETURN
1817 003752 016237 177776 001364 MOV -2(R2), $TMP2 ;MOVE CHECK BUFFER CONTENTS TO $TMP2
1818 003760 010237 001370 MOV R2, $TMP4 ;MOVE ADDRESS +2 TO $TMP4
1819 003764 162737 000002 001370 SUB #2, $TMP4 ;CORRECT SO IT POINTS TO ADDRESS CAUSING ERROR
1820 003772 016337 177776 001366 MOV -2(R3), $TMP3 ;MOVE INPUT BUFFER CONTENTS TO $TMP3
1821 004000 010337 001372 MOV R3, $TMP5 ;MOVE ADDRESS +2 TO $TMP5
1822 004004 162737 000002 001372 SUB #2, $TMP5 ;CORRECT SO IT POINTS TO ADDRESS CAUSING ERROR
1823 004012 000207 RTS PC ;RETURN TO ERROR CALL - PC ON STACK ALREADY POINTS THERE
1824 004014 023737 002624 002622 DATCH2: CMP LENCHK, BUFLN ;SEE IF THE BUFFER HAS BEEN CHECKED
1825 004022 001345 BNE DATCH1 ;GO BACK FOR ANOTHER TRY IF NOT
1826 004024 013716 001362 MOV $TMP1, (SP) ;RESTORE PC RETURN
1827 004030 062716 000006 ADD #6, (SP) ;CORRECT IT SO RETURN IS AFTER THE ERROR CALL
1828 004034 000207 RTS PC ;RETURN

```



```
1829                                     .SBTTL SUBROUTINE TO RESTORE DR11 INT VECT & SET CPU PRIORITY TO 7.
1830 :*****
1831 :*
1832 :* THIS ROUTINE IS USED IN VARIOUS TESTS TO CLEAR ANY DATA THAT
1833 :* MAY BE LEFT IN ANY REGISTERS, AND RESTORE CPU PRIORITY TO 7.
1834 :*
1835 :*****
1836 004036 012737 000340 177776 CLENU: MOV    #LEVEL7,PSW    ;RESTORE CPU TO PRIORITY LEVEL 7
1837 004044 012777 010000 176450      MOV    #MA,@CSR      ;DO AN INIT CLEARING WCR, BAR & BDR BY SETTING
1838 004052 005077 176444          CLR    @CSR          ;AND CLEARING THE MAINT BIT AND CLEAR THE CSR
1839 004056 000207          RTS    PC          ;EXIT
```

```

1840                                     .SBTTL SUBROUTINE TO CHECK FOR CABLE MODE AND ALTER EXPECTED DATA
1841                                     :*****
1842                                     :
1843                                     : THIS SUBROUTINE CHECKS THE DDW (DEVICE DESCRIPTOR WORD) FOR THE CABLE
1844                                     : BEING IN OR OUT AND SETS BITS 12, 10, 8 AND 6 IN THE EXPECTED DATA
1845                                     : LOCATION IF THE CABLE IS OUT.
1846                                     :
1847                                     :*****
1848 004060 032737 000004 002720 CHKAB: BIT    #BIT2,DDW    ;CHECK FOR CABLE STATUS
1849 004066 001006                BNE      1$             ;EXIT IF CABLE DOES EXIST
1850 004070 052737 127000 002572                BIS      #127000,ECSR ;SET BITS 12, 10, 8, 7 & 6 - CABLE DOESN'T EXIST
1851 004076 052737 127000 002536                BIS      #127000,BUT  ;SET BITS 12, 10, 8, 7 & 6 IN BITS UNDER TEST LOCATION TOO
1852 004104 000207                1$:      RTS      PC      ;RETURN
  
```



```

1853 .SBTTL SUBROUTINE TO CHECK CORRECT DATA PATTERN WAS MOVED TO INBUF
1854 *****
1855 :
1856 : THIS SUBROUTINE HAS THE NEED TO CALL 2 ERRORS. THE ERRORS ARE TO BE
1857 : LOCATED IN THE TEST JUST AFTER THE JSR CALL. FUTURE USE OF THIS
1858 : SUBROUTINE MUST BE HANDLED AS FOLLOWS:
1859 :
1860 : JSR PC,DATOCK ;SUBROUTINE CALL
1861 : ERROR +46 ;ERROR CALL
1862 : JSR PC,DATOC2 ;RETURN TO SUBROUTINE AFTER ERROR RTS
1863 : ERROR +47 ;2ND ERROR CALL
1864 : CONTINUE ;SUBROUTINE RETURNS HERE WHEN DONE
1865 :
1866 *****
1867 004106 011637 001362 DATOCK: MOV (SP), $TMP1 ;SAVE PC RETURN
1868 004112 012702 052525 MOV #52525, R2 ;DATO NUMBER TO R2
1869 004116 013703 002616 MOV INBUF, R3 ;STARTING ADDRESS OF IN BUFFER TO R3
1870 004122 005037 002624 CLR LENCHK ;CLEAR LENGTH CHECK
1871 004126 005237 002624 DATOC1: INC LENCHK ;MAKE A COMPARISON
1872 004132 020223 CMP R2, (R3)+ ;IS THE DATA CORRECT?
1873 004134 001415 BEQ DATOC2 ;BRANCH IF OK
1874 004136 013716 001362 MOV $TMP1, (SP) ;MOVE OLD PC RETURN TO STACK
1875 004142 010237 001364 MOV R2, $TMP2 ;MOVE EXPECTED DATA TO $TMP2
1876 004146 016337 177776 001366 MOV -2(R3), $TMP3 ;MOVE RECEIVED DATA TO $TMP3
1877 004154 010337 001370 MOV R3, $TMP4 ;MOVE ADDRESS +2 TO $TMP4
1878 004160 162737 000002 001370 SUB #2, $TMP4 ;CORRECT ADDRESS SO IT POINTS TO ADDRESS CAJSING ERROR
1879 004166 000207 RTS PC ;RETURN TO ERROR CALL
1880 004170 023737 002624 002622 DATOC2: CMP LENCHK, BUFLN ;SEE IF THE BUFFER HAS BEEN CHECKED
1881 004176 001353 BNE DATOC1 ;BUFFER CHECKED?
1882 004200 020223 CMP R2, (R3)+ ;CHECK END OF BUFFER + 1
1883 004202 001017 BNE 1$ ;BRANCH IF NOT LOADED
1884 004204 010237 001364 MOV R2, $TMP2 ;MOVE EXPECTED DATA TO $TMP2
1885 004210 016337 177776 001366 MOV -2(R3), $TMP3 ;MOVE RECEIVED DATA TO $TMP3
1886 004216 010337 001370 MOV R3, $TMP4 ;MOVE ADDRESS +2 TO $TMP4
1887 004222 162737 000002 001370 SUB #2, $TMP4 ;CORRECT ADDRESS SO IT POINTS TO ADDRESS CAUSING ERROR
1888 004230 013716 001362 MOV $TMP1, (SP) ;CORRECT PC RETURN
1889 004234 062716 000006 ADD #6, (SP) ;POINT TO 2ND ERROR CALL AFTER JSR PC,DATOCK
1890 004240 000207 RTS PC ;RETURN TO ERROR CALL
1891 004242 013716 001362 1$: MOV $TMP1, (SP) ;RESTORE RETURN ADDRESS
1892 004246 062716 000010 ADD #10, (SP) ;POINT TO PROPER RETURN AFTER THE ERROR CALLS
1893 004252 000207 RTS PC ;EXIT

```

```
1894                                     .SBTTL SUBROUTINE TO CLEAR IE AND HALT IF ERROR IS SET
1895                                     *****
1896                                     :
1897                                     : THIS SUBROUTINE HAS THE NEED TO CALL AN ERROR IN THE TEST. THE ERROR
1898                                     : IS TO BE LOCATED IN THE TEST JUST AFTER THE JSR CALL. FUTURE USE OF
1899                                     : THIS SUBROUTINE MUST BE HANDLED AS FOLLOWS:
1900                                     : JSR PC,ERRCHK ;SUBROUTINE CALL
1901                                     : ERROR +21 ;ERROR CALL
1902                                     : CONTINUE ;SUBROUTINE RETURNS HERE IF NO ERROR
1903                                     :
1904                                     *****
1905 004254 042777 000100 176240 ERRCHK: BIC #IE,@CSR ;CLEAR IE
1906 004262 017737 176234 002560 MOV @CSR,RCSR ;MOVE RECEIVED DATA TO RCSR
1907 004270 013737 002560 002572 MOV RCSR,ECSR ;MOVE EXPECTED DATA TO ECSR
1908 004276 042737 100000 002572 BIC #ER,ECSR ;CLEAR THE ERROR BIT
1909 004304 052737 000200 002572 BIS #RY,ECSR ;SET THE READY BIT
1910 004312 013701 002560 MOV RCSR,R1 ;MOVE DATA TO R1 FOR CHECKING
1911 004316 012737 000200 001360 MOV #RY,$TMP0 ;MOVE EXPECTED DATA TO $TMP0
1912 004324 042701 077577 BIC #77577,R1 ;CLEAR ALL BUT THE ERROR AND READY BITS
1913 004330 022701 000200 CMP #RY,R1 ;SEE IF ERROR BIT IS CLEAR AND READY IS SET
1914 004334 001002 BNE 1$ ;BRANCH AROUND PC CORRECTION SO ERROR WILL CALL
1915 004336 062716 000002 ADD #2,(SP) ;CORRECT PC RETURN - DATA OK
1916 004342 000207 1$: RTS PC ;EXIT
```

```

1917
1918
1919
1920
1921
1922
1923
1924 004344 012702 002416
1925 004350 013700 001464
1926 004354 012701 000020
1927 004360 010022
1928 004362 062700 000010
1929 004366 005301
1930 004370 001373
1931 004372 000207
  
```

```

.SBTTL SUBROUTINE TO GENERATE DEVICE ADDRESS TABLE
*****
:
:
:   THIS SUBROUTINE GENERATES AN ADDRESS TABLE LOCATED AT REGADR STARTING
:   WITH THE BASE DEVICE ADDRESS (CONTENTS OF $BASE) IN STEPS OF 10.
:
*****
DEVADS: MOV    #REGADR,R2      ;POINT R2 TO THE DEVICE ADDRESS TABLE
        MOV    $BASE,R0      ;LOAD BASE DEVICE ADDRESS IN R0
        MOV    #16.,R1       ;MOVE LOOP COUNTER TO R1
1$:     MOV    R0,(R2)+       ;MOVE DEVICE ADDRESS TO TABLE
        ADD    #10,R0        ;POINT R0 TO NEXT DEVICE ADDRESS
        DEC    R1            ;DECREMENT THE LOOP COUNTER AND
        BNE    1$           ;BRANCH IF NOT ALL DONE YET
        RTS    PC           ;EXIT
  
```

```

1932                                     .SBTTL  SUBROUTINE TO RESET THE ''.+2'' AND 'BPT' LOCATIONS
1933                                     ;*****
1934                                     ;*
1935                                     ;*   THIS SUBROUTINE LOADS ''.+2'' AND 'BPT' INTO ALL UNUSED LOCATIONS
1936                                     ;*   BETWEEN 4-776.
1937                                     ;*
1938                                     ;*****
1939 004374 012700 004436  BPINIT: MOV      #BPTINT,R0      ;POINT R0 TO TABLE OF BPT INIT LOCATIONS
1940 004400 012701 000003      MOV      #3,R1          ;DO 3 SETS OF ''.+2'' AND 'BPT' SETUPS
1941 004404 012002      1$:  MOV      (R0)+,R2          ;MOVE START ADDRESS TO R2
1942 004406 012003      MOV      (R0)+,R3          ;MOVE END ADDRESS TO R3
1943 004410 010204      MOV      R2,R4            ;MOVE ADDRESS TO R4
1944 004412 005724      TST      (R4)+          ;INCREMENT R4 TO PRODUCE THE ''.+2'' NUMBER
1945 004414 010422      2$:  MOV      R4,(R2)+      ;MOVE THE NUMBER TO THE LOCATION
1946 004416 012722 000003      MOV      #BPT,(R2)+   ;MOVE 'BPT' TO THE NEXT LOCATION
1947 004422 022424      CMP      (R4)+,(R4)+      ;ADD 4 TO R4
1948 004424 020203      CMP      R2,R3          ;SEE IF WE HAVE DONE ALL FOR THIS LOCATION
1949 004426 001372      BNE      2$              ;BRANCH BACK FOR ANOTHER TRANSFER IF NOT
1950 004430 005301      DEC      R1              ;DECREMENT R1
1951 004432 001364      BNE      1$              ;BRANCH BACK IF 3 GROUPS NOT DONE
1952 004434 000207      RTS      PC              ;EXIT
1953
1954 004436 000004 000014 000050  BPTINT: .WORD 4,14,50      ;ADDRESSES USED TO PUT ''.+2'' & 'BPT' BACK
1955 004444 000174 000254 001000      .WORD 174,254,1000

```

Year	Address	Hex	Dec	Label	Instruction	Comment
1956					.SBTTL	SUBROUTINE TO EXTRACT INFORMATION ABOUT THE DR11
1957					*****	
1958					;	
1959					;	
1960					;	THIS SUBROUTINE EXTRACTS INFORMATION ABOUT THE DR11 THAT INTERRUPTED
1961					;	AND LOADS THE DATA FOUND INTO THE DEVICE DESCRIPTOR WORD FOR THAT
1962					;	BOARD.
1963					;	
1964	004452	012710	010000	DRGET:	MOV #MA,(R0)	;SET THE MAINTENANCE BIT AND
1965	004456	005010			CLR (R0)	;CLEAR THE CSR TO DO AN INIT
1966	004460	016612	000002		MOV 2(SP),(R2)	;MOVE VECTOR+4 FOR THIS MODULE TO VECTOR LOCATION
1967	004464	012616			MOV (SP)+,(SP)	;MOVE RETURN OF THIS SUBROUTINE TO ITS PROPER POSITION
1968	004466	162712	000004		SUB #4,(R2)	;DUMB VECTOR IS WRONG - CORRECT IT
1969	004472	013711	002540		MOV LEVEL,(R1)	;PUT THE PRIORITY LEVEL INTO THE \$DDWXX LOCATION
1970	004476	005237	001466		INC \$DEVN	;INDICATE DEVICE EXISTENCE IN DEVICE MAP
1971	004502	005237	002414		INC QTYBRD	;INCREMENT DEVICE COUNT
1972	004506	052710	100000		BIS #EIR,(R0)	;GO TO EIR TO GET B/W STATE
1973	004512	011037	002562		MOV (R0),REIR	;MOVE EIR TO REIR
1974	004516	005010			CLR (R0)	;GO BACK TO CSR
1975	004520	012710	010000		MOV #MA,(R0)	;SET THE MAINT BIT
1976	004524	005010			CLR (R0)	;DO AN INIT
1977	004526	032737	000001	002562	BIT #BIT0,REIR	;TEST FOR B/W STATE
1978	004534	001003			BNE 1\$	;BRANCH IF A W
1979	004536	052711	000001		BIS #BIT0,(R1)	;SET STATE IN DEVICE DESCRIPTOR WORD
1980	004542	000406			BR 2\$	;GO TO CABLE STATUS TEST
1981	004544	032737	000400	002562	1\$: BIT #N2,REIR	;CHECK 2/N CYCLE STATE
1982	004552	001402			BEQ 2\$	;BRANCH IF 2 CYCLE
1983	004554	052711	000002		BIS #BIT1,(R1)	;N CYCLE - SET BIT IN DEVICE DESC
1984	004560	011037	002560		2\$: MOV (R0),RCSR	;MOVE RECEIVED DATA TO RCSR TO GET CABLE STATUS
1985	004564	032737	127000	002560	BIT #127000,RCSR	;CHECK IF ANY BITS ARE SET - THEY ARE IF NO CABLE
1986	004572	001015			BNE 3\$	;BRANCH IF NO CABLE
1987	004574	112710	000004		MOVB #F2,(R0)	;CABLE IS POSSIBLY IN - SET FNCT2
1988	004600	011037	002560		MOV (R0),RCSR	;MOVE RECEIVED DATA TO RCSR
1989	004604	052710	010000		BIS #MA,(R0)	;SET THE MAINTENANCE BIT
1990	004610	005010			CLR (R0)	;CLEAR THE CSR TO DO AN INIT
1991	004612	032737	020000	002560	BIT #AT,RCSR	;TEST THE ATTN BIT
1992	004620	001402			BEQ 3\$	;BRANCH IF NOT SET - NO CABLE
1993	004622	052711	000004		BIS #BIT2,(R1)	;SET CABLE BIT IN DEVICE DESC
1994	004626	000207		3\$:	RTS PC	;EXIT

1995						.SBTTL SUBROUTINE TO PRINT THE AUTOSIZED BOARD CONFIGURATIONS
1996						*****
1997						THIS SUBROUTINE PRINTS THE BOARD CONFIGURATIONS FOUND BY ASIZE
1998						*****
1999						TYP CNF: CLR LOOP ;CLEAR THE BOARD NUMBER COUNTER
2000						MOV \$DEVN,R5 ;GET DEVICE MAP
2001	004630	005037	002662			MOV #REGADR,R1 ;MOVE THE ADDRESS OF THE REGISTER ADDRESS TABLE TO R1
2002	004634	013705	001466			MOV #VECADR,R2 ;MOVE THE ADDRESS OF THE VECTOR ADDRESS TABLE TO R2
2003	004640	012701	002416			CLR R3 ;CLEAR THE DEVICE DESCRIPTOR ADDRESS POINTER
2004	004644	012702	002456			TYPE ,NOBUT ;TYPE: 'NO. BOARDS UNDER TEST: '
2005	004650	005003				MOV QTYBRD,-(SP) ;MOVE THE QUANTITY TO THE STACK FOR TYPEOUT
2006	004652	104401	035433			TYPDS ;TYPE THE NUMBER
2007	004656	013746	002414			TYPE ,BRVWPC ;TYPE THE HEADER
2008	004662	104405				MOV #16,R0 ;SET UP LOOP COUNTER FOR 16 BOARDS
2009	004664	104401	035472			1\$: BIT #BIT0,R5 ;DEVICE UNDER TEST?
2010	004670	012700	000020			BEQ 8\$;BRANCH IF NO
2011	004674	032705	000001			MOV LOOP,-(SP) ;PUT BOARD # ON STACK FOR TYPEOUT
2012	004700	001466	002662			TYPDS ;PRINT BOARD # (0 TO 16)
2013	004702	013746				TYPE ,SPACE6 ;TYPE 6 SPACE CHARACTERS
2014	004706	104405				MOV (R1),-(SP) ;SAVE REGISTER ADDRESS FOR TYPEOUT
2015	004710	104401	035257			TYPDC ;PRINT DEVICE REGISTER ADDRESS
2016	004714	011146				TYPE ,SPACE6 ;TYPE 6 SPACE CHARACTERS
2017	004716	104402				MOV (R2),-(SP) ;SAVE VECTOR ADDRESS FOR TYPEOUT
2018	004720	104401	035257			TYPDS ;PRINT VECTOR ADDRESS
2019	004724	011246				TYPOS ;PRINT 3 DIGITS, LEADING ZEROS SUPPRESSED
2020	004726	104403				.BYTE 3,0 ;TYPE 7 SPACE CHARACTERS
2021	004730	003	000			TYPE ,SPACE7 ;TYPE 7 SPACE CHARACTERS
2022	004732	104401	035266			MOV \$DDW0(R3),DDW ;MOVE DEVICE DESCRIPTOR WORD TO DDW
2023	004736	016337	001474	002720		BIT #BIT0,DDW ;TEST WHICH STATE, B OR W, FOR THIS BOARD
2024	004744	032737	000001	002720		BEQ 2\$;GO PRINT W STATE IF W
2025	004752	001403				TYPE ,B ;TYPE A 'B'
2026	004754	104401	035303			BR 3\$;GO TO NEXT CHECK
2027	004760	000402				2\$: TYPE ,W ;TYPE A 'W'
2028	004762	104401	035305			3\$: TYPE ,SPACE7 ;TYPE 7 SPACE CHARACTERS
2029	004766	104401	035266			JSR PC,PNTPRI ;PRINT DEVICE PRIORITY
2030	004772	004737	005630			TYPE ,SPACE7 ;TYPE 7 SPACE CHARACTERS
2031	004776	104401	035266			BIT #BIT1,DDW ;TEST 2/N CYCLE STATE
2032	005002	032737	000002	002720		BEQ 4\$;GO PRINT 2 STATE IF 2
2033	005010	001403				TYPE ,N ;TYPE AN 'N'
2034	005012	104401	035320			BR 5\$;GO TO NEXT CHECK
2035	005016	000402				4\$: TYPE ,TWO ;TYPE A '2'
2036	005020	104401	035316			5\$: TYPE ,SPACE7 ;TYPE 7 SPACE CHARACTERS
2037	005024	104401	035266			BIT #BIT2,DDW ;TEST CABLE STATE
2038	005030	032737	000004	002720		BEQ 6\$;GO PRINT 'NO' IF NO CABLE
2039	005036	001403				TYPE ,YES ;TYPE 'YES'
2040	005040	104401	035312			BR 7\$;GO TO LOOP CHECK
2041	005044	000402				6\$: TYPE ,NO ;TYPE 'NO'
2042	005046	104401	035307			7\$: TYPE ,CRLF ;TYPE A <CRLF>
2043	005052	104401	001405			8\$: INC LOOP ;INCREMENT BOARD COUNT FOR POSSIBLE NEXT PASS
2044	005056	005237	002662			CMP (R1)+,(R2)+ ;INCREMENT COUNTERS
2045	005062	022122				ASR R5 ;SHIFT R5 TO THE RIGHT TO MOVE BOARD BIT INTO BIT 0
2046	005064	006205				ADD #2,R3 ;ADD 2 TO THE DEVICE DESCRIPTOR WORD POINTER
2047	005066	062703	000002			DEC R0 ;DECREMENT THE LOOP COUNTER AND
2048	005072	005300				1\$;BRANCH BACK FOR CHECK IF 16 BOARDS NOT DONE
2049	005074	001277				TYPE ,CRLF ;TYPE ANOTHER <CRLF>
2050	005076	104401	001405			PC ;EXIT
2051	005102	000207				


```

2052 .SBTTL SUBROUTINE TO CHECK FOR LOCATION BELONGING TO A DR11
2053 *****
2054
2055 THIS SUBROUTINE CHECKS FOR THE LOCATION BELONGING TO A DR11.
2056 *****
2057
2058 005104 012737 000340 002540 CHK4DR: MOV #LEVEL7,LEVEL ;MOVE PRIORITY 7 TO LEVEL
2059 005112 012703 005264 MOV #LEVELS,R3 ;MOVE ADDRESS OF PRIORITY LEVELS TO R3
2060 005116 012704 000004 MOV #4,R4 ;DO 4 PRIORITY CHECKS
2061 005122 012737 000400 002660 1$: MOV #400,TIME ;SET UP WAIT LOOP COUNTER
2062 005130 012710 010000 MOV #MA,(R0) ;SET THE MAINTENANCE BIT AND
2063 005134 005010 CLR (R0) ;CLEAR TO POSSIBLY DO AN INIT
2064 005136 013737 000014 001362 MOV BPTVCT,$TMP1 ;SAVE BPT TRAP VECTOR
2065 005144 012737 005222 000014 MOV #3$,BPTVCT ;INTERRUPTS TO 3$
2066 005152 012337 177776 MOV (R3)+,PSW ;SET CPU PRIORITY TO NEXT LEVEL
2067 005156 000240 NOP ;KILL A LITTLE TIME
2068 005160 012710 000105 MOV #IE+FNCT2+GO,(R0) ;SET IE, FNCT2 AND GO ATTEMPTING ANOTHER INTERRUPT
2069 005164 005337 002660 2$: DEC TIME ;DECREMENT TIME
2070 005170 001375 BNE 2$ ;BRANCH BACK UNTIL ZERO
2071 005172 012737 000340 177776 MOV #LEVEL7,PSW ;SET CPU PRIORITY BACK TO 7
2072 005200 013737 001362 000014 MOV $TMP1,BPTVCT ;RESTORE BPT TRAP VECTOR
2073 005206 162737 000040 002540 SUB #40,LEVEL ;PUT LOCATION 'LEVEL' AT NEXT PRIORITY - INTERRUPT FAILED
2074 005214 005304 DEC R4 ;DECREMENT LOOP COUNTER
2075 005216 001341 BNE 1$ ;BRANCH BACK IF NOT ALL PRIORITY LEVELS CHECKED YET
2076 005220 000416 BR 4$ ;EXIT - THIS LOCATION DOESN'T BELONG TO A DR11
2077 005222 012737 000340 177776 3$: MOV #LEVEL7,PSW ;AHAAH - THIS *IS* A DR11 - SET CPU PRIORITY BACK TO 7
2078 005230 013737 001362 000014 MOV $TMP1,BPTVCT ;RESTORE BPT TRAP VECTOR
2079 005236 016666 000010 000006 MOV 10(SP),6(SP) ;MOVE THIS SUBROUTINE'S RETURN UP ONE SPOT ON STACK
2080 005244 011666 000010 MOV (SP),10(SP) ;MOVE TRAP ADDRESS TO RETURN'S OLD LOCATION
2081 005250 062706 000006 ADD #6,SP ;KICK GARBAGE OFF STACK - GOT TO KEEP IT CLEAN
2082 005254 000402 BR 5$ ;BRANCH TO KICK OUT
2083 005256 062716 000012 4$: ADD #12,(SP) ;CORRECT RETURN TO NOT DO DR11 SETUP
2084 005262 000207 5$: RTS PC ;KICK OUT
2085
2086 005264 000300 000240 LEVELS: .WORD LEVEL6,LEVEL5 ;PRIORITY LEVELS TO LOAD INTO THE PSW
2087 005270 000200 000140 .WORD LEVEL4,LEVEL3
  
```

```
2088 .SBTTL SUBROUTINE TO AUTO SIZE DR11 BOARD CONFIGURATION
2089 *****
2090 *
2091 * THIS SUBROUTINE AUTOSIZES THE BOARD CONFIGURATION AND CALLS FOR THE
2092 * PRINTING OF THE CONFIGURATION IF BIT 12 OF THE SWITCH REGISTER IS
2093 * CLEAR.
2094 *
2095 *****
2096 ASIZE: MOV #SDDW0,R0 ;MOVE ADDRESS OF FIRST DEVICE DESCRIPTOR WORD TO R0
2097 MOV #16.,R1 ;CLEAR 16 WORDS OUT OF THE DICTIONARY
2098 1$: CLR (R0)+ ;CLEAR THE WORD - SORRY CAN'T LOOK IT UP ANY MORE
2099 DEC R1 ;DECREMENT THE LOOP COUNTER AND
2100 BNE 1$ ;BRANCH BACK IF NOT DONE YET
2101 JSR PC,BPINIT ;GO RESET THE '.+2' AND 'BPT' LOCATIONS
2102 JSR PC,DEVADS ;GENERATE DEVICE ADDRESS TABLE
2103 CLR QTYBRD ;CLEAR DEVICE COUNT
2104 CLR $DEVM ;CLEAR DEVICE MAP
2105 MOV TOVECT,OLDPC1 ;SAVE TIMEOUT VECTOR
2106 MOV #3$,TOVECT ;SET TIMEOUT POINTER TO 3$
2107 MOV TMOPSW,OLDPS1 ;SAVE TIMEOUT PS
2108 MOV #LEVEL7,TMOPSW ;CPU PRIORITY TO 7
2109 MOV #172604,R0 ;POINT R0 TO UNIT #16 CSR ADDRESS LOCATION
2110 MOV #SDDW15,R1 ;LOAD DEVICE DESC ADDRESS
2111 MOV #VECADR+36,R2 ;POINT R2 TO UNIT #16 VECTOR ADDRESS LOCATION
2112 MOV #16.,R5 ;DO 16 BOARDS
2113 2$: CLR (R0) ;CHECK FOR REGISTER EXISTENCE
2114 JSR PC,CHK4DR ;GO CHECK FOR A DR11 AT THIS LOCATION IF IT DOES
2115 JSR PC,DRGET ;GO EXTRACT INFO FROM THE DR11
2116 BR 4$ ;BRANCH OVER THE STACK CORRECTION
2117 3$: ADD #4,SP ;CORRECT STACK AFTER TIMEOUT
2118 4$: SUB #10,R0 ;POINT R0 TO NEXT DEVICE ADDRESS
2119 CMP -(R1),-(R2) ;POINT R1 AND R2 TO NEXT DEVICE & VECTOR LOCATIONS
2120 DEC R5 ;DECREMENT LOOP COUNTER
2121 BEQ 5$ ;EXIT IF ALL DONE WITH 16 BOARDS
2122 ASL $DEVM ;ADJUST DEVICE MAP FOR NEXT UNIT CHECK
2123 BR 2$ ;GO CHECK NEXT LOCATION
2124 5$: MOV OLDPS1,TMOPSW ;RESTORE TIMEOUT PS
2125 MOV OLDPC1,TOVECT ;RESTORE TIMEOUT VECTOR
2126 BIT #BIT12,@SWR ;CHECK FOR CONFIGURATION PRINT
2127 BNE 6$ ;BRANCH IF PRINT NOT REQUESTED
2128 TST QTYBRD ;SEE IF ANY BOARDS WERE FOUND
2129 BEQ 6$ ;BRANCH TO RETURN IF NOT - NO BOARD INFO TO PRINT
2130 JSR PC,TYPCNF ;GO TYPE THE BOARD CONFIGURATIONS
2131 6$: RTS PC ;EXIT
```



```

2132
2133
2134
2135
2136
2137
2138 005502 012702 002456
2139 005506 013700 001460
2140 005512 042700 177400
2141 005516 012701 000020
2142 005522 010022
2143 005524 062700 000010
2144 005530 005301
2145 005532 001373
2146 005534 000207
  
```

```

      .SBTTL  SUBROUTINE TO GENERATE VECTOR ADDRESS TABLE
      *****
      *
      *      THIS SUBROUTINE GENERATES THE VECTOR ADDRESS TABLE.
      *
      *****
VCTADS: MOV      #VECADR,R2      ;GET LOCATION OF VECTOR TABLE
          MOV      $VECT1,R0      ;COPY BASE VECTOR
          BIC      #177400,R0     ;CLEAR UPPER BYTE
          MOV      #16.,R1        ;DO 16 VECTORS
1$:      MOV      R0,(R2)+        ;PUT VECTOR ADDRESS IN TABLE
          ADD      #10,R0         ;POINT R0 TO NEXT VECTOR ADDRESS
          DEC      R1             ;DECREMENT LOOP COUNTER
          BNE      1$            ;BRANCH IF NOT ALL DONE YET
          RTS      PC            ;EXIT
  
```

```

2147 .SBTTL ROUTINE TO REPORT UNEXPECTED OR ERRONEOUS TRAPS OR INTERRUPTS
2148 *****
2149 :
2150 : THIS IS THE ROUTINE TO REPORT UNEXPECTED OR ERRONEOUS TRAPS OR INTERRUPTS
2151 :
2152 :*****
2153 005536 012737 000340 177776 CATCH: MOV #LEVEL7,PSW ;REESTABLISH CPU PRIORITY AT 7
2154 005544 012637 002542 MOV (SP)+,BDVECT ;GET ADDRESS OF TRAP VECTOR + 4
2155 005550 012637 002674 MOV (SP)+,OLDPS1 ;SAVE PS
2156 005554 012637 002676 MOV (SP)+,OLDPC2 ;SAVE PC OF ADDRESS OF INSTRUCTION CAUSING TRAP
2157 005560 012637 002700 MOV (SP)+,OLDPS2 ;SAVE 2ND PS
2158 005564 162737 000004 002542 SUB #4,BDVECT ;ADJUST TO POINT TO TRAP ADDRESS
2159 005572 104050 ERROR +50 ;UNEXPECTED TRAP OR INTERRUPT TO TRAP ADDRESS BELOW
2160 005574 013746 002700 MOV OLDPS2,-(SP) ;RESTORE PS RETURN ON STACK
2161 005600 013746 002676 MOV OLDPC2,-(SP) ;RESTORE PC RETURN ON STACK
2162 005604 000002 RTI ;RETURN
  
```

2163						.SBTTL SUBROUTINE TO PRINT STATE OF A DDW BIT
2164						*****
2165						;
2166						THIS SUBROUTINE PRINTS THE STATE OF THE BIT IN THE DDW THAT WAS
2167						PRELOADED INTO BITTST.
2168						;
2169						*****
2170	005606	005046				PSTATE: CLR -(SP) ;SHOW STATE AS ZERO INITIALLY
2171	005610	033737	002710	002720		BIT BITTST,DDW ;CHECK STATE OF BIT IN DDW USING BIT SET IN BITTST
2172	005616	001401				BEQ 1\$;BRANCH IF NOT SET
2173	005620	005216				INC (SP) ;SHOW A '1' STATE FOR THAT BIT
2174	005622	104403			1\$:	TYPOS ;TYPE THE STATE, LEADING ZEROS SUPPRESSED
2175	005624	001	000			.BYTE 1,0 ;TYPE 1 CHARACTER, SUPPRESS LEADING ZEROS
2176	005626	000207				RTS PC ;EXIT

2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192

005630 013746 002720
005634 006216
005636 006216
005640 006216
005642 006216
005644 006216
005646 042716 177770
005652 104403
005654 001 000
005656 000207

```
.SBTTL SUBROUTINE TO PRINT DEVICE PRIORITY
*****
;
; THIS SUBROUTINE PRINTS THE DEVICE PRIORITY
;
*****
PNTPRI: MOV DDW, -(SP) ;PUT DEVICE DESCRIPTOR WORD ON STACK
        ASR (SP) ;SHIFT RIGHT STACK LOCATION 5 PLACES
        ASR (SP)
        ASR (SP)
        ASR (SP)
        ASR (SP)
        BIC #177770, (SP) ;MASK TO GET PRIORITY
        TYPOS ;TYPE THE DEVICE PRIORITY
        .BYTE 1,0 ;TYPE 1 CHARACTER, SUPPRESS LEADING ZEROS
        RTS PC ;EXIT
```

```
2193 .SBTTL INITIALIZE THE COMMON TAGS SUBROUTINE
2194 *****
2195 *
2196 * THIS SUBROUTINE INITIALIZES THE INTERRUPT VECTORS. USE IS AS FOLLOWS:
2197 * MOV #STACK,SP ;INITIALIZE THE STACK
2198 * JSR PC,SETUP ;CALL THE SUBROUTINE
2199 *
2200 *****
2201 005660 011637 001360 SETUP: MOV (SP), $TMP0 ;SAVE RETURN ADDRESS
2202 005664 012706 001300 ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
005670 005026 001300 MOV # $CMTAG, R6 ;;FIRST LOCATION TO BE CLEARED
005672 022706 001340 100$: CLR (R6)+ ;;CLEAR MEMORY LOCATION
005676 001374 001340 CMP #SWR, R6 ;;DONE?
005700 012706 001300 BNE 100$ ;;LOOP BACK IF NO
MOV #STACK, SP ;;SETUP THE STACK POINTER
;;INITIALIZE A FEW VECTORS
MOV # $SCOPE, @IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
MOV #340, @IOTVEC+2 ;;LEVEL 7
MOV # $ERROR, @EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
MOV #340, @EMTVEC+2 ;;LEVEL 7
MOV # $TRAP, @TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
MOV #340, @TRAPVEC+2 ;;LEVEL 7
MOV # $PWRDN, @PWRVEC ;;POWER FAILURE VECTOR
MOV #340, @PWRVEC+2 ;;LEVEL 7
MOV $ENDCT, $EOPCT ;;SETUP END-OF-PROGRAM COUNTER
CLR $ESCAPE ;;CLEAR THE ESCAPE ON ERROR ADDRESS
MOVB #1, $ERMAX ;;ALLOW ONE ERROR PER TEST
MOV #TST1+2, $LPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
MOV #TST1+2, $LPERR ;;SETUP THE ERROR LOOP ADDRESS
;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
006020 013746 000004 MOV @ERRVEC, -(SP) ;;SAVE ERROR VECTOR
006024 012737 006060 000004 MOV #64, @ERRVEC ;;SET UP ERROR VECTOR
006032 012737 177570 001340 MOV #DSWR, SWR ;;SETUP FOR A HARDWARE SWICH REGISTER
006040 012737 177570 001342 MOV #DDISP, DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
006046 022777 177777 173264 CMP #-1, @SWR ;;TRY TO REFERENCE HARDWARE SWR
006054 001011 BNE 66$ ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
;;AND THE HARDWARE SWR IS NOT = -1
006056 000402 BR 65$ ;;BRANCH IF NO TIMEOUT
006060 062706 000004 64$: ADD #4, SP ;;CLEAN UP STACK AFTER INTERRUPT
006064 012737 000176 001340 65$: MOV #SWREG, SWR ;;POINT TO SOFTWARE SWR
006072 012737 000174 001342 MOV #DISPREG, DISPLAY
006100 012637 000004 66$: MOV (SP)+, @ERRVEC ;;RESTORE ERROR VECTOR
006104 005037 001416 CLR $PASS ;;CLEAR PASS COUNT
006110 132737 000200 001431 BITB #APTSIZE, $ENVN ;;TEST USER SIZE UNDER APT
006116 001403 BEQ 67$ ;;YES, USE NON-APT SWITCH
006120 012737 001432 001340 MOV #SWREG, SWR ;;NO, USE APT SWITCH REGISTER
006126 67$:
2203 006126 013746 001360 MOV $TMP0, -(SP) ;PUT RETURN ADDRESS ON STACK AND
2204 006132 000207 RTS PC ;RETURN TO THE CALLING ROUTINE
```

```

2205                                     .SBTTL MEMORY MANAGEMENT AND LOCATION CHECK SUBROUTINE
2206                                     *****
2207                                     *
2208                                     * THIS SUBROUTINE CHECKS FOR MEMORY MANAGEMENT EXISTENCE AND WHETHER OR
2209                                     * NOT A LOCATION IN UPPER MEMORY EXISTS.
2210                                     *
2211                                     *****
2212 006134 005737 002712 TSTMM: TST MEMGMT ;TEST TO SEE IF MEMORY MANAGEMENT EXISTS
2213 006140 001440 BEQ 4$ ;BRANCH IF NOT
2214 006142 012737 000401 006152 MOV #CY+GO,1$ ;SET UP BIT TEST DATA
2215 006150 040227 BIC R2,(PC)+ ;TEST TO SEE IF BOTH THE CYCLE AND GO BITS ARE SET
2216 006152 000401 1$: .WORD CY+GO ;LOCATION TO STORE THE CYCLE AND GO BITS
2217 006154 001032 BNE 4$ ;KICK OUT IF CYCLE AND/OR GO ARE CLEAR
2218 006156 032737 000060 002572 BIT #X6+X7,ECSR ;SEE IF XBA16 OR XBA17 WERE SET IN EXPECTED DATA
2219 006164 001426 BEQ 4$ ;BRANCH OUT IF NOT
2220 006166 032737 000040 002572 BIT #X7,ECSR ;SEE IF XBA17 IS SET
2221 006174 001005 BNE 2$ ;GO CHECK STATUS OF XBA16 IF SET
2222 006176 132737 000001 002713 BITB #B10,MEMGMT+1 ;SEE IF 200000+NOCARE WAS FOUND TO EXIST - IF NOT,
2223 006204 001420 BEQ 5$ ;GO SET EXPECTED ERROR AND NEX BITS AND CHECK FOR ERROR
2224 006206 000415 BR 4$ ;BRANCH OUT IF LOCATION EXISTS
2225 006210 032737 000020 002572 2$: BIT #X6,ECSR ;SEE IF XBA16 IS SET
2226 006216 001005 BNE 3$ ;BRANCH TO CHECK 600000+NOCARE IF SET
2227 006220 132737 000002 002713 BITB #BIT1,MEMGMT+1 ;SEE IF 400000+NOCARE WAS FOUND TO EXIST - IF NOT,
2228 006226 001407 BEQ 5$ ;GO SET EXPECTED ERROR AND NEX BITS AND CHECK FOR ERROR
2229 006230 000404 BR 4$ ;BRANCH OUT IF LOCATION EXISTS
2230 006232 132737 000004 002713 3$: BITB #BIT2,MEMGMT+1 ;SEE IF 600000+NOCARE WAS FOUND TO EXIST - IF NOT,
2231 006240 001402 BEQ 5$ ;GO SET EXPECTED ERROR AND NEX BITS AND CHECK FOR ERROR
2232 006242 062716 000002 4$: ADD #2,(SP) ;CORRECT PC RETURN
2233 006246 000207 5$: RTS PC ;KICK OUT
    
```


2234
2235
2236
2237
2238
2239
2240
2241
2242 000002
2243 006250 177777
2244 006252 000000
2245 006254 052525
2246 006256 125252
2247 006260 031463
2248 006262 007417
2249 006264 000377
2250 000010

```
.SBTTL BIT PATTERN
*****
;
; THIS IS A BIT PATTERN TABLE THAT CAN BE USED TO CHECK ANY LOCATION FOR
; ALL COMBINATIONS OF STUCK AND/OR SHORTED BITS.
;
*****

PATRNS: .RADIX 2 ;THIS ENABLES YOU TO SEE THE PATTERNS IN BINARY
        .WORD 1111111111111111 ;ALL SET BITS
        .WORD 0000000000000000 ;ALL CLEAR BITS
        .WORD 0101010101010101 ;EVEN BITS SET, ODD BITS CLEAR
        .WORD 1010101010101010 ;ODD BITS SET, EVEN BITS CLEAR
        .WORD 0011001100110011 ;PAIRS OF BITS SET
        .WORD 0000111100001111 ;GROUPS OF 4 BITS SET
        .WORD 0000000011111111 ;UPPER BYTE CLEAR, LOWER BYTE SET
        .RADIX 8 ;THIS RETURNS MODE BACK TO OCTAL
```

2251
 2252
 2253
 2254
 2255
 2256
 2257
 2258
 2259
 2260
 2261
 2262
 2263
 2264
 2265
 2266
 2267
 2268
 2269
 2270
 2271
 2272
 2273
 2274
 2275
 2276
 2277
 2278
 2279
 2280
 2281
 2282
 2283
 2284
 2285
 2286
 2287
 2288
 2289
 2290
 2291

006266
 006266
 006306
 006326
 006346
 006366
 006406
 006426
 006446
 006466
 006506
 006526
 006546
 006566
 006606
 006626
 006646
 006666
 006706
 006726
 006746
 006766
 007006
 007026
 007046
 007066
 007106
 007126
 007146
 007166
 007206
 007226
 007246

.SBTTL EXPECTED DATA TABLE FOR CSR CHECK TEST 30												

* THE 'EXPAT' TABLE IS USED TO CHECK THE CONTENTS OF THE CSR AFTER SETTING												
* THE BITS IN THE CSR.												

MAICLR: ; X=					0	1	2	3	4	5	6	7
.WORD	0200	0000	1202	1002	122204	122204	123206	123206	; CSR 00000X EXPECTED			
.WORD	4210	4010	5212	5012	126214	126214	127216	127216	; CSR 00001X EXPECTED			
.WORD	0220	0020	1222	1022	122224	122224	123226	123226	; CSR 00002X EXPECTED			
.WORD	4230	4030	5232	5032	126234	126234	127236	127236	; CSR 00003X EXPECTED			
.WORD	0240	0040	1242	1042	122244	122244	123246	123246	; CSR 00004X EXPECTED			
.WORD	4250	4050	5252	5052	126254	126254	127256	127256	; CSR 00005X EXPECTED			
.WORD	0260	0060	1262	1062	122264	122264	123266	123266	; CSR 00006X EXPECTED			
.WORD	4270	4070	5272	5072	126274	126274	127276	127276	; CSR 00007X EXPECTED			
.WORD	0300	0100	1302	1102	122304	022104	123306	023106	; CSR 00010X EXPECTED			
.WORD	4310	4110	5312	5112	126314	026114	127316	027116	; CSR 00011X EXPECTED			
.WORD	0320	0120	1322	1122	122324	022124	123326	023126	; CSR 00012X EXPECTED			
.WORD	4330	4130	5332	5132	126334	026134	127336	027136	; CSR 00013X EXPECTED			
.WORD	0340	0140	1342	1142	122344	022144	123346	023146	; CSR 00014X EXPECTED			
.WORD	4350	4150	5352	5152	126354	026154	127356	027156	; CSR 00015X EXPECTED			
.WORD	0360	0160	1362	1162	122364	022164	123366	023166	; CSR 00016X EXPECTED			
.WORD	4370	4170	5372	5172	126374	026174	127376	027176	; CSR 00017X EXPECTED			
.WORD	0600	0200	1602	1202	122604	122604	123606	123606	; CSR 00040X EXPECTED			
.WORD	4610	4210	5612	5212	126614	126614	127616	127616	; CSR 00041X EXPECTED			
.WORD	0620	0220	1622	1222	122624	122624	123626	123626	; CSR 00042X EXPECTED			
.WORD	4630	4230	5632	5232	126634	126634	127636	127636	; CSR 00043X EXPECTED			
.WORD	0640	0240	1642	1242	122644	122644	123646	123646	; CSR 00044X EXPECTED			
.WORD	4650	4250	5652	5252	126654	126654	127656	127656	; CSR 00045X EXPECTED			
.WORD	0660	0260	1662	1262	122664	122664	123666	123666	; CSR 00046X EXPECTED			
.WORD	4670	4270	5672	5272	126674	126674	127676	127676	; CSR 00047X EXPECTED			
.WORD	0700	0300	1702	1302	122704	022304	123706	023306	; CSR 00050X EXPECTED			
.WORD	4710	4310	5712	5312	126714	026314	127716	027316	; CSR 00051X EXPECTED			
.WORD	0720	0320	1722	1322	122724	022324	123726	023326	; CSR 00052X EXPECTED			
.WORD	4730	4330	5732	5332	126734	026334	127736	027336	; CSR 00053X EXPECTED			
.WORD	0740	0340	1742	1342	122744	022344	123746	023346	; CSR 00054X EXPECTED			
.WORD	4750	4350	5752	5352	126754	026354	127756	027356	; CSR 00055X EXPECTED			
.WORD	0760	0360	1762	1362	122764	022364	123766	023366	; CSR 00056X EXPECTED			
.WORD	4770	4370	5772	5372	126774	026374	127776	027376	; CSR 00057X EXPECTED			

					.SBTTL	EXPECTED DATA TABLE FOR CSR CHECK TEST 3											
2292																	
2293																	
2294																	
2295	007266	010200	010000	011202	MAISET:	: X=	0	1	2	3	4	5	6	7			
2296	007306	014210	014010	015212		.WORD	10200	10000	11202	11002	12204	12004	13206	13006	:CSR	01000X EXPECTED	
2297	007326	010220	010020	011222		.WORD	14210	14010	15212	15012	16214	16014	17216	17016	:CSR	01001X EXPECTED	
2298	007346	014230	014030	015232		.WORD	10220	10020	11222	11022	12224	12024	13226	13026	:CSR	01002X EXPECTED	
2299	007366	010240	010040	011242		.WORD	14230	14030	15232	15032	16234	16034	17236	17036	:CSR	01003X EXPECTED	
2300	007406	014250	014050	015252		.WORD	10240	10040	11242	11042	12244	12044	13246	13046	:CSR	01004X EXPECTED	
2301	007426	010260	010060	011262		.WORD	14250	14050	15252	15052	16254	16054	17256	17056	:CSR	01005X EXPECTED	
2302	007446	014270	014070	015272		.WORD	10260	10060	11262	11062	12264	12064	13266	13066	:CSR	01006X EXPECTED	
2303	007466	010300	010100	011302		.WORD	14270	14070	15272	15072	16274	16074	17276	17076	:CSR	01007X EXPECTED	
2304	007506	014310	014110	015312		.WORD	10300	10100	11302	11102	12304	12104	13306	13106	:CSR	01010X EXPECTED	
2305	007526	010320	010120	011322		.WORD	14310	14110	15312	15112	16314	16114	17316	17116	:CSR	01011X EXPECTED	
2306	007546	014330	014130	015332		.WORD	10320	10120	11322	11122	12324	12124	13326	13126	:CSR	01012X EXPECTED	
2307	007566	010340	010140	011342		.WORD	14330	14130	15332	15132	16334	16134	17336	17136	:CSR	01013X EXPECTED	
2308	007606	014350	014150	015352		.WORD	10340	10140	11342	11142	12344	12144	13346	13146	:CSR	01014X EXPECTED	
2309	007626	010360	010160	011362		.WORD	14350	14150	15352	15152	16354	16154	17356	17156	:CSR	01015X EXPECTED	
2310	007646	014370	014170	015372		.WORD	10360	10160	11362	11162	12364	12164	13366	13166	:CSR	01016X EXPECTED	
2311	007666	010600	011202	011602		.WORD	14370	14170	15372	15172	16374	16174	17376	17176	:CSR	01017X EXPECTED	
2312	007706	014610	015212	015612		.WORD	10600	11202	11602	12204	12604	13206	13606	14210	:CSR	01040X EXPECTED	
2313	007726	010620	011222	011622		.WORD	14610	15212	15612	16214	16614	17216	17616	10200	:CSR	01041X EXPECTED	
2314	007746	014630	015232	015632		.WORD	10620	11222	11622	12224	12624	13226	13626	14230	:CSR	01042X EXPECTED	
2315	007766	010640	011242	011642		.WORD	14630	15232	15632	16234	16634	17236	17636	10220	:CSR	01043X EXPECTED	
2316	010006	014650	015252	015652		.WORD	10640	11242	11642	12244	12644	13246	13646	14250	:CSR	01044X EXPECTED	
2317	010026	010660	011262	011662		.WORD	14650	15252	15652	16254	16654	17256	17656	10240	:CSR	01045X EXPECTED	
2318	010046	014670	015272	015672		.WORD	10660	11262	11662	12264	12664	13266	13666	14270	:CSR	01046X EXPECTED	
2319	010066	010700	011302	011702		.WORD	14670	15272	15672	16274	16674	17276	17676	10260	:CSR	01047X EXPECTED	
2320	010106	014710	015312	015712		.WORD	10700	11302	11702	12304	12704	13306	13706	14310	:CSR	01050X EXPECTED	
2321	010126	010720	011322	011722		.WORD	14710	15312	15712	16314	16714	17316	17716	10300	:CSR	01051X EXPECTED	
2322	010146	014730	015332	015732		.WORD	10720	11322	11722	12324	12724	13326	13726	14330	:CSR	01052X EXPECTED	
2323	010166	010740	011342	011742		.WORD	14730	15332	15732	16334	16734	17336	17736	10320	:CSR	01053X EXPECTED	
2324	010206	014750	015352	015752		.WORD	10740	11342	11742	12344	12744	13346	13746	14350	:CSR	01054X EXPECTED	
2325	010226	010760	011362	011762		.WORD	14750	15352	15752	16354	16754	17356	17756	10340	:CSR	01055X EXPECTED	
2326	010246	014770	015372	015772		.WORD	10760	11362	11762	12364	12764	13366	13766	14370	:CSR	01056X EXPECTED	
						.WORD	14770	15372	15772	16374	16774	17376	17776	10360	:CSR	01057X EXPECTED	

```

2327      .SBTTL  MAIN PROGRAM - INITIALIZATION ROUTINES
2328      :*****
2329      :
2330      :MAIN PROGRAM - INITIALIZATION ROUTINES
2331      :
2332      :*****
2333
2334 010266 005037 002670      START1: CLR      MANSIZE      ;CLEAR THE MANSIZE SO WE WILL AUTOSIZE
2335 010272 005037 001416      START:  CLR      $PASS      ;CLEAR $PASS
2336 010276 005037 002716      CLR      $PASS2     ;CLEAR $PASS2
2337 010302 005037 001412      CLR      $FATAL     ;CLEAR ERROR NO.
2338 010306 005037 001410      CLR      $MSGTYP    ;CLEAR MESSAGE TYPE
2339 010312 005037 001414      CLR      $TESTN     ;CLEAR TEST NO.
2340 010316 005037 001420      CLR      $DEVCT     ;CLEAR DEVICE COUNT
2341 010322 005037 001422      CLR      $UNIT      ;CLEAR UNIT NUMBER
2342 010326 012737 000006 000004  MOV      #TOVECT+2,TOVECT ;INITIALIZE TIMEOUT VECTORS TO 6
2343 010334 012737 000003 000006  MOV      #BPT,TMOPSW ;CATCHER ROUTINE
2344 010342 012706 001300      MOV      #STACK,SP   ;INITIALIZE THE STACK
2345 010346 004737 005660      JSR      PC,SETUP    ;GO TO THE SETUP ROUTINE TO INITIALIZE VECTORS
2346 010352 005037 001422      CLR      $UNIT      ;CLEAR $UNIT
2347 010356 005037 001420      CLR      $DEVCT     ;CLEAR $DEVCT
2348 010362 005037 001414      CLR      $TESTN     ;CLEAR $TESTN
2349 010366 005037 002706      CLR      EOPLOC     ;CLEAR EOPLOC
2350 010372 132737 000001 001430  BITB      #BIT0,$ENV ;CHECK IF ON APT
2351 010400 001404      BEQ      1$              ;BR IF NOT APT
2352 010402 132737 000200 001431  BITB      #BIT7,$ENVM ;DID APT SIZE
2353 010410 001007      BNE      2$              ;BR, IF APT SIZED
2354 010412 022737 177777 002670 1$:  CMP      #-1,MANSIZE ;WAS CONFIGURATION SET UP IN MULT. BOARD ROUTINE?
2355 010420 001422      BEQ      BEGIN            ;IF YES, SKIP SELF-SIZING
2356 010422 004737 005274      JSR      PC,ASIZE   ;AUTOMATICALLY SIZE FOR BOARD CONFIGURATION
2357 010426 000417      BR       BEGIN            ;BRANCH
2358 010430 005037 002414      2$:  CLR      QTYBRD   ;CLEAR DEVICE CNT
2359 010434 013702 001466      MOV      $DEVM,R2   ;MOVE DEVICE MAP TO R2
2360 010440 005702      3$:  TST      R2           ;TEST MSB OF DEVICE MAP
2361 010442 100002      BPL      4$              ;BR, IF MSB IS ZERO
2362 010444 005237 002414      INC      QTYBRD     ;INCREMENT DEVICE COUNT, IF MSB=1
2363 010450 000241      4$:  CLC                   ;CLEAR THE CARRY BIT FOR THE ROL
2364 010452 006102      ROL      R2              ;SHIFT NEXT BIT INTO MSB POSITION
2365 010454 001371      BNE      3$              ;CONTINUE CHECKING $DEVM, IF MORE BITS SET
2366 010456 004737 004344      JSR      PC,DEVADS  ;GENERATE DEVICE ADDRESS TABLE
2367 010462 004737 005502      JSR      PC,VCTADS  ;GENERATE VECTOR ADDRESS TABLE

```

2368					.SBTTL	DETERMINE MEM MGMT AND UPPER MEMORY EXISTENCE
2369	010466	005737	001416		BEGIN: TST	\$PASS ;SEE IF THIS IS THE FIRST PASS
2370	010472	001145			BNE	BEGIN1 ;BRANCH IF NOT
2371	010474	005737	002716		TST	\$PASS2 ;SEE IF UPPER LOCATION HAS BEEN SET
2372	010500	001142			BNE	BEGIN1 ;BRANCH IF NOT
2373	010502	005037	002712		CLR	MEMGMT ;CLEAR THE MEMORY MANAGEMENT FLAG
2374	010506	013737	000004	002672	MOV	TOVECT,OLDPC1 ;SAVE TIMEOUT VECTOR
2375	010514	012737	010750	000004	MOV	#3\$,TOVECT ;TIMEOUT VECTOR TO 3\$
2376	010522	013737	000006	002674	MOV	TMOPSW,OLDPS1 ;SAVE TIMEOUT PS
2377	010530	012737	000340	000006	MOV	#LEVEL7,TMOPSW ;PS TIMEOUT TO PRIORITY 7
2378	010536	005737	177572		TST	MMRO ;TEST FOR THE PRESENCE OF MEMORY MANAGEMENT
2379	010542	105237	002712		INCB	MEMGMT ;INCREMENT FLAG SHOWING MEMORY MANAGEMENT EXISTS
2380	010546	012737	077406	172304	MOV	#77406,KIPDR2 ;MAKE KIPDR2 RESIDENT
2381	010554	012737	077406	172324	MOV	#77406,KDPDR2 ;MAKE KDPDR2 RESIDENT
2382	010562	013737	000250	002676	MOV	MMVECT,OLDPC2 ;SAVE MEMORY MANAGEMENT VECTOR
2383	010570	012737	010722	000250	MOV	#1\$,MMVECT ;MEMORY MANAGEMENT VECTOR TO 1\$
2384	010576	013737	000252	002700	MOV	MMPS,OLDPS2 ;SAVE MEMORY MANAGEMENT PS
2385	010604	012737	000340	000252	MOV	#LEVEL7,MMPS ;MEMORY MANAGEMENT PS TO PRIORITY 7
2386	010612	005237	177572		INC	MMRO ;TURN ON MEMORY MANAGEMENT
2387	010616	012737	002400	172344	MOV	#2400,KIPAR2 ;SET UP KIPAR2 TO ACCESS LOCATION 240000+BITS 12-0 OF NOCARE
2388	010624	012737	002400	172364	MOV	#2400,KDPAR2 ;SET UP KDPAR2 TO ACCESS LOCATION 240000+BITS 12-0 OF NOCARE
2389	010632	005737	053700		TST	NOCARE ;SEE IF BITS 12-0 OF NOCARE ADRS +240000 EXISTS
2390	010636	152737	000001	002713	BISB	#BIT0,MEMGMT+1 ;SET BIT 0 OF UPPER BYTE OF MEMGMT IF IT DOES
2391	010644	012737	004400	172344	MOV	#4400,KIPAR2 ;SET UP KIPAR2 TO ACCESS LOCATION 440000+BITS 12-0 OF NOCARE
2392	010652	012737	004400	172364	MOV	#4400,KDPAR2 ;SET UP KDPAR2 TO ACCESS LOCATION 440000+BITS 12-0 OF NOCARE
2393	010660	005737	053700		TST	NOCARE ;SEE IF BITS 12-0 OF NOCARE ADRS +440000 EXISTS
2394	010664	152737	000002	002713	BISB	#BIT1,MEMGMT+1 ;SET BIT 1 OF UPPER BYTE OF MEMGMT IF IT DOES
2395	010672	012737	006400	172344	MOV	#6400,KIPAR2 ;SET UP KIPAR2 TO ACCESS LOCATION 640000+BITS 12-0 OF NOCARE
2396	010700	012737	006400	172364	MOV	#6400,KDPAR2 ;SET UP KDPAR2 TO ACCESS LOCATION 640000+BITS 12-0 OF NOCARE
2397	010706	005737	053700		TST	NOCARE ;SEE IF BITS 12-0 OF NOCARE ADRS +640000 EXISTS
2398	010712	152737	000004	002713	BISB	#BIT2,MEMGMT+1 ;SET BIT 2 OF UPPER BYTE OF MEMGMT IF IT DOES
2399	010720	000402			BR	2\$;BRANCH OVER STACK CORRECTION
2400	010722	062706	000004		ADD	#4,SP ;CORRECT STACK AFTER MM TRAP
2401	010726	005037	177572		CLR	MMRO ;TURN OFF MEMORY MANAGEMENT
2402	010732	013737	002700	000252	MOV	OLDPS2,MMPS ;RESTORE MEMORY MANAGEMENT PS
2403	010740	013737	002676	000250	MOV	OLDPC2,MMVECT ;RESTORE MEMORY MANAGEMENT VECTOR
2404	010746	000402			BR	4\$;BRANCH OVER STACK CORRECTION
2405	010750	062706	000004		ADD	#4,SP ;CORRECT STACK AFTER TIMEOUT
2406	010754	013737	002674	000006	MOV	OLDPS1,TMOPSW ;RESTORE TIMEOUT PS
2407	010762	013737	002672	000004	MOV	OLDPC1,TOVECT ;RESTORE TIMEOUT VECTOR
2408	010770	104401	037170		TYPE	,M1 ;TYPE: '(*X) INHIBITS EOP'S, (*Y) FOR ERROR SUMMARY'
2409						;'UNIBUS HANG? RESTART AT ADDRESS '
2410	010774	012746	053622		MOV	#UBHANG,-(SP) ;MOVE ADDRESS OF HANG ROUTINE TO STACK
2411	011000	104402			TYPOC	;'GO TYPE THE ADDRESS IN OCTAL
2412	011002	104401	037307		TYPE	,M1A ;'CZDRLCO DR11 GEN NPR INTFC LOGIC TEST'

```
2413 .SBTTL PREPARE ADDRESSES AND VECTORS FOR UUT
2414 011006 012737 000001 002544 BEGIN1: MOV #BIT0,DEVMSK ;SET UP BIT MASK TO TEST $DEVMSK FOR DEVICES
2415 011014 005037 002546 CLR TABINX ;CLEAR LOCATION TO STORE TABLE OFFSETS
2416 011020 033737 002544 001466 TSTDEV: BIT DEVMSK,$DEVMSK ;CHECK TO SEE IF DEVICE IS TO BE TESTED
2417 011026 001026 BNE 2$ ;BR, IF YES
2418 011030 005737 002544 TST DEVMSK ;SEE IF BIT 15 IS SET
2419 011034 100013 BPL 1$ ;BRANCH TO CONTINUE IF NOT SET
2420 011036 005737 001416 TST $PASS ;SEE IF THIS IS THE FIRST PASS
2421 011042 001361 BNE BEGIN1 ;BRANCH TO REINITIALIZE THE DEVMSK LOCATION IF NOT
2422 011044 005737 002716 TST $PASS2 ;SEE IF THIS IS THE FIRST PASS
2423 011050 001356 BNE BEGIN1 ;BRANCH TO REINITIALIZE THE DEVMSK LOCATION IF NOT
2424 011052 104401 037037 TYPE ,NODVPR ;TYPE: 'NO DEVICES RECOGNIZED - DIAGNOSTIC CANNOT BE RUN'
2425 ; 'RESTART AT 204 IF A DEVICE IS PRESENT'
2426 011056 000000 HALT ;FATAL ERROR - HALT HERE
2427 011060 000137 010266 JMP START1 ;JUMP TO START1 TO CHECK AGAIN FOR A MODULE
2428 011064 006337 002544 1$: ASL DEVMSK ;SHIFT MASK TO CHECK NEXT $DEVMSK BIT
2429 011070 062737 000002 002546 ADD #2,TABINX ;INCREMENT TABLE INDEX
2430 011076 005237 001422 INC $UNIT ;INCREMENT UNIT NUMBER
2431 011102 000746 BR TSTDEV ;GO TEST NEXT BIT OF DEVICE MAP
2432
2433 011104 006337 002544 2$: ASL DEVMSK ;UPDATE DEVICE MAP TEST MASK
2434 011110 013702 002546 MOV TABINX,R2 ;MOVE TABLE OFFSET TO R2
2435 011114 062737 000002 002546 ADD #2,TABINX ;UPDATE TABLE OFFSET FOR NEXT DEVICE
2436 011122 016200 002416 MOV REGADR(R2),R0 ;PUT UUT ADDRESS INTO R0
2437 011126 012701 002516 MOV #WCR,R1 ;POINT R1 TO STORAGE AREA FOR UUT ADDRESSES
2438 011132 012703 000004 MOV #4,R3 ;MOVE 4 ADDRESSES
2439 011136 010021 3$: MOV R0,(R1)+ ;TRANSFER UUT ADDRESS
2440 011140 062700 000002 ADD #2,R0 ;POINT TO NEXT UUT REGISTER
2441 011144 005303 DEC R3 ;DECREMENT THE LOOP COUNTER AND
2442 011146 001373 BNE 3$ ;BRANCH IF NOT DONE TRANSFERRING
2443 011150 016200 002456 MOV VECADR(R2),R0 ;PUT UUT VECTOR INTO R0
2444 011154 010021 MOV R0,(R1)+ ;TRANSFER UUT VECTORS TO ACTIVE TABLE AREA
2445 011156 062700 000002 ADD #2,R0 ;POINT TO NEXT VECTOR
2446 011162 010011 MOV R0,(R1) ;TRANSFER VECTOR TO TABLE AREA
2447 011164 016237 001474 002720 MOV $DDW0(R2),DDW ;SET UP DDW TO PROPER DEVICE DESCRIPTOR WORD
2448 011172 013737 002720 002552 MOV DDW,DRLEV ;MOVE THE WORD TO THE DRLEV LOCATION
2449 011200 042737 177437 002552 BIC #177437,DRLEV ;STRIP ALL BITS EXCEPT BR LEVEL
2450 011206 105037 030707 REINIT: CLRB CHARCT ;CLEAR THE CHARACTER LOCATION OF ANY CHARACTER
2451 011212 004737 004374 JSR PC,BPINIT ;GO RESET THE '+2' AND 'BPT' LOCATIONS
2452 011216 004737 004036 JSR PC,CLENUP ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
2453 011222 105737 002706 TSTB EOPLOC ;SEE IF ^X IS ENABLED (IS THE PRINTER DISABLED)
2454 011226 001006 BNE TST1 ;GO DO TEST IF NOT
2455 ;*****
2456 ;*DO*NOT*REMOVE*THE*WAIT*LOOP*ROUTINE*BELOW*. BECAUSE OF THE SPEED OF THIS DIAG-
2457 ;* NOSTIC (.125 SECOND FOR 1 PASS, NO ERRORS), SOME VIDEO TERMINALS PRINT ERRONEOUS
2458 ;* CHARACTER(S) WITH THE EOP MESSAGE DUE TO THE RESET EXECUTED IN TEST 4. THIS WAIT
2459 ;* LOOP ENABLES THOSE TERMINALS TO 'CATCH UP' BEFORE ITS EXECUTION.
2460 ;*****
2461 011230 012737 010000 011240 MOV #10000,2$ ;REESTABLISH THE WAIT LOOP COUNTER
2462 011236 005327 1$: DEC (PC)+ ;DECREMENT THE LOCATION TO KILL TIME
2463 011240 010000 2$: .WORD 10000 ;LOCATION TO BE DECREMENTED
2464 011242 001375 BNE 1$ ;BRANCH BACK UNTIL ZERO
2465 ;*****
2466 ;MAIN PROGRAM - DEVICE TESTS
2467 ;*****
2468
2469
```

2477

```
.SBTTL TEST #1 - CAN ALL DR11 REG BE ADDRESSED WITHOUT ERROR?
*****
*TEST 1 CAN ALL DR11 REG BE ADDRESSED WITHOUT ERROR?
*
* THIS TEST INSURES THAT THE CSR, BAR, BDR AND WCR REGISTERS CAN BE
* ACCESSED FOR THIS DEVICE. IF NOT, THE REST OF THE DIAGNOSTIC CANNOT
* BE RUN.
*****
TST1: BIT #BIT11,@SWR ;22 TEST TO SEE IF TEST NUMBER TRACER ENABLED
      BEQ 1001$ ;22 BRANCH IF NOT
      TYPE ,1000$ ;22 TYPE: 'T # 1'
      BR 1001$ ;22 GET OVER ASCII
      .ASCIIZ /T # 1/<CRLF> ;22 THE ASCII MESSAGE
      .EVEN
1001$:
SCOPE ;PROCESS LOOPING AND TEST NUMBER INCREMENT
MOV #999$,$LPERR ;SET LOOP ON ERROR TO 999$
999$: CLR $ERTTL ;CLEAR THE ERROR TOTAL - NEW PASS
      MOV #1$,BUSERR ;CPU ERROR VECTOR TO 1$
      MOV WCR,DREG ;SAVE ADDRESS FOR POSSIBLE ERROR TYPING
      TST @WCR ;ACCESS REGISTER
      MOV BAR,DREG ;SAVE ADDRESS FOR POSSIBLE ERROR TYPING
      TST @BAR ;ACCESS REGISTER
      MOV CSR,DREG ;SAVE ADDRESS FOR POSSIBLE ERROR TYPING
      TST @CSR ;ACCESS REGISTER
      MOV BDR,DREG ;SAVE ADDRESS FOR POSSIBLE ERROR TYPING
      TST @BDR ;ACCESS REGISTER
      MOV DRINV,DREG ;SAVE ADDRESS FOR POSSIBLE ERROR TYPING
      TST @DRINV ;ACCESS REGISTER
      TST $ERTTL ;SEE IF THERE WERE ANY ERRORS
      BEQ 2$ ;BRANCH TO CONTINUE IF NONE
      JMP ENDEV ;GO TO END OF DEVICE ROUTINE - FATAL ERRORS
1$: MOV (SP)+,OLDPC1 ;SAVE PC OF TRAP FOR ERROR PRINTOUT
      MOV (SP)+,OLDPS1 ;SAVE PS FOR RESTORATION AFTER ERROR CALL
      ERROR +1 ;CANNOT ACCESS DR11 REGISTER
      MOV OLDPS1,-(SP) ;PUT PS BACK ON STACK
      MOV OLDPC1,-(SP) ;PUT PC BACK ON STACK
      RTI ;RETURN TO PROGRAM
2$: MOV #6,BUSERR ;RESTORE #6 TO BUS ERROR
```

011244	032777	004000	170066
011252	001407		
011254	104401	011262	
011260	000404		
011262	124	040	043
011272			
011272	000004		
011274	012737	011302	001310
2478	011302	005037	001312
2479	011306	012737	011410
2480	011314	013737	002516
2481	011322	005777	171170
2482	011326	013737	002520
2483	011334	005777	171160
2484	011340	013737	002522
2485	011346	005777	171150
2486	011352	013737	002524
2487	011360	005777	171140
2488	011364	013737	002526
2489	011372	005777	171130
2490	011376	005737	001312
2491	011402	001414	
2492	011404	000137	024360
2493	011410	012637	002672
2494	011414	012637	002674
2495	011420	104001	
2496	011422	013746	002674
2497	011426	013746	002672
2498	011432	000002	
2499	011434	012737	000006

2506

```
.SBTTL TEST #2 - CHECK B OR W STATUS IS AS EXPECTED
*****
*TEST 2 CHECK B OR W STATUS IS AS EXPECTED
*
* THIS TEST INSURES THAT THE B OR W STATUS IN THE DEVICE DESCRIPTOR
* WORD MATCHES WHAT THE EIR SAYS THE MODULE IS.
*
*****
TST2: BIT #BIT11,@SWR ;## TEST TO SEE IF TEST NUMBER TRACER ENABLED
      BEQ 1001$ ;## BRANCH IF NOT
      TYPE ,1000$ ;## TYPE: 'T # 2'
      BR 1001$ ;## GET OVER ASCII
      .ASCIIZ /T # 2/<CRLF> ;## THE ASCII MESSAGE
      .EVEN
1001$:
SCOPE
MOV #999$,$LPERR ;PROCESS LOOPING AND TEST NUMBER INCREMENT
CLR @CSR ;SET LOOP ON ERROR TO 999$
MOV #EIR,@CSR ;GO TO CSR
MOV @CSR,BORW ;FORCE TO BE EIR
BIC #CBIT0,BORW ;ATTEMPT EIR READ
MOV DDW,$TMP1 ;MASK OFF ALL BITS EXCEPT BIT 0
BIC #CBIT0,$TMP1 ;GET DEVICE DESCRIPTOR WORD
BEQ 1$ ;MASK OFF ALL BUT B OR W BIT
CLR $TMP1 ;BRANCH IF IT IS CLEAR
BR 2$ ;CLEAR THE BIT
MOV #1,$TMP1 ;GO TEST THE BIT
CMP BORW,$TMP1 ;SET THE BIT
BEQ TST3 ;B OR W STATE AS EXPECTED?
ERROR +2 ;BRANCH IF OK
;DR11-B OR W MODE INCORRECT (0=B, 1=W)
```

011442	032777	004000	167670
011450	001407		
011452	104401	011460	
011456	000404		
011460	124	040	043
011470			
011470	000004		
011472	012737	011500	001310
2507 011500	005077	171016	
2508 011504	012777	100000	171010
2509 011512	017737	171004	002610
2510 011520	042737	177776	002610
2511 011526	013737	002720	001362
2512 011534	042737	177776	001362
2513 011542	001403		
2514 011544	005037	001362	
2515 011550	000403		
2516 011552	012737	000001	001362
2517 011560	023737	002610	001362
2518 011566	001401		
2519 011570	104002		

2527

.SBTTL TEST #3 - CHECK CSR BIT PATTERNS WITH MAINT BIT SET

*TEST 3 CHECK CSR BIT PATTERNS WITH MAINT BIT SET

THIS TEST SETS ALL POSSIBLE COMBINATIONS OF SET BITS IN THE CSR WITH
 THE MAINTENANCE BIT SET, AND COMPARES THE RECEIVED CSR CONTENTS WITH
 THAT OF THE EXPECTED PATTERNS IN THE 'MAISET' TABLE.

011572	032777	004000	167540	TST3:	BIT	#BIT11,@SWR	;22 TEST TO SEE IF TEST NUMBER TRACER ENABLED
011600	001407				BEQ	1001\$	;22 BRANCH IF NOT
011602	104401	011610			TYPE	,1000\$	;22 TYPE: 'T # 3'
011606	000404				BR	1001\$	;22 GET OVER ASCII
011610	124	040	043	1000\$:	.ASCIIZ	/T # 3/<CRLF>	;22 THE ASCII MESSAGE
					.EVEN		
011620				1001\$:			
011620	000004				SCOPE		;PROCESS LOOPING AND TEST NUMBER INCREMENT
011622	012737	011704	001310		MOV	#999\$,\$LPERR	;SET LOOP ON ERROR TO 999\$
2528 011630	004737	004036			JSR	PC,CLEUP	;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
2529 011634	012737	000200	001362		MOV	#RY,\$TMP1	;MOVE READY BIT TO \$TMP1
2530 011642	032737	000004	002720		BIT	#BIT2,DDW	;TEST TO SEE IF CABLE IS IN
2531 011650	001003				BNE	1\$	;BRANCH AROUND NON-CABLE SETUP IF IN
2532 011652	052737	127000	001362		BIS	#127000,\$TMP1	;SET THE BITS TO BE EXPECTED IN \$TMP1
2533 011660	012701	007266		1\$:	MOV	#MAISET,R'	;MOVE ADDRESS OF EXPECTED PATTERNS TO R1
2534 011664	012702	010000			MOV	#MA,R2	;START WITH JUST THE MAINTENANCE BIT
2535 011670	005037	002714			CLR	ERRCNT	;CLEAR THE ERRCNT LOCATION FOR USE OF ERROR +202
2536 011674	012700	000002			MOV	#2,R0	;DO 2 SETS OF 200 PATTERNS
2537 011700	012703	000200		2\$:	MOV	#200,R3	;MOVE 200 TO THE LOOP COUNTER
2538 011704	052777	010000	170610	999\$:	BIS	#MA,@CSR	;SET MAINTENANCE AND
2539 011712	005077	170604			CLR	@CSR	;CLEAR TO DO AN INIT
2540 011716	017737	170600	002560		MOV	@CSR,RCSR	;MOVE RECEIVED DATA TO RCSR
2541 011724	023737	001362	002560		CMP	\$TMP1,RCSR	;MAKE SURE EXPECTED DATA CAME UP
2542 011732	001404				BEQ	3\$	;BRANCH IF SO
2543 011734	013737	001362	002572		MOV	\$TMP1,ECSR	;MOVE EXPECTED DATA TO ECSR
2544 011742	104032				ERROR	+32	;CSR IS WRONG
2545 011744	012777	177777	170544	3\$:	MOV	#-1,@WCR	;MOVE 1 WORD COUNT TO WCR IN CASE OF IE ENABLED
2546 011752	012777	053700	170540		MOV	#NOCARE,@BAR	;MOVE A NOT-CARE ADDRESS TO BAR FOR SAME REASON
2547 011760	010277	170536			MOV	R2,@CSR	;SET THE PARTICULAR FUNCTION BITS IN CSR
2548 011764	017737	170532	002560		MOV	@CSR,RCSR	;MOVE RECEIVED DATA TO RCSR
2549 011772	011137	002572			MOV	(R1),ECSR	;MOVE EXPECTED DATA TO ECSR
2550 011776	023737	002572	002560		CMP	ECSR,RCSR	;COMPARE EXPECTED WITH RECEIVED
2551 012004	001425				BEQ	7\$	;BRANCH IF OK
2552 012006	012737	000401	012016		MOV	#CY+GO,4\$	;REESTABLISH TEST PATTERN
2553 012014	040227				BIC	R2,(PC)+	;SEE IF BOTH CYCLE AND GO WERE SET
2554 012016	000401			4\$:	.WORD	CY+GO	;LOCATION TO HOLD BOTH CYCLE AND GO BITS
2555 012020	001013				BNE	6\$	;BRANCH TO ERROR ONLY IF CYCLE AND GO WERE SET
2556 012022	032737	000060	002572		BIT	#X6+X7,ECSR	;SEE IF EITHER XBA16 OR XBA17 ARE SET
2557 012030	001407				BEQ	6\$	;BRANCH TO ERROR IF BOTH ARE CLEAR
2558 012032	052737	140000	002572	5\$:	BIS	#ER+NX,ECSR	;SET THE ERROR AND NEX BITS - EXPECT THEM TO SET
2559 012040	023737	002560	002572		CMP	RCSR,ECSR	;NOW SEE IF DATA MATCHES
2560 012046	001415				BEQ	10\$	;BRANCH AROUND ERROR IF IT DOES
2561 012050	010237	002536		6\$:	MOV	R2,BUT	;MOVE THE BITS SET INTO CSR TO THE BUT LOCATION
2562 012054	104202				ERROR	+202	;CSR PATTERN NOT CORRECT
2563 012056	000411				BR	10\$	;BRANCH AROUND MM TESTS
2564 012060	012737	000401	012070	7\$:	MOV	#CY+GO,8\$	;REESTABLISH TEST PATTERN
2565 012066	040227				BIC	R2,(PC)+	;SEE IF BOTH CYCLE AND GO WERE SET
2566 012070	000401			8\$:	.WORD	CY+GO	;LOCATION TO HOLD BOTH CYCLE AND GO BITS

2567	012072	001003			BNE	10\$	·BRANCH AROUND MEM MGMT TEST IF EITHER OR BOTH WERE CLEAR
2568	012074	004737	006134	9\$:	JSR	PC,TSTMM	;GO CHECK FOR MEMORY MANAGEMENT EXISTENCE
2569	012100	000754			BR	5\$	;IF RETURN IS HERE, GO BACK TO SET EXPECTED DATA
2570	012102	062701	000002	10\$:	ADD	#2,R1	;INCREMENT R1 TO NEXT EXPECTED PATTERN
2571	012106	005202			INC	R2	;INCREMENT THE PATTERN
2572	012110	005303			DEC	R3	;DECREMENT THE LOOP COUNTER
2573	012112	001274			BNE	999\$	;BRANCH BACK IF NOT DONE
2574	012114	062702	000200		ADD	#200,R2	;ADD 200 TO PATTERN LOCATION
2575	012120	005300			DEC	R0	;DECREMENT THE LOOP COUNTER AND
2576	012122	001266			BNE	2\$	;BRANCH BACK IF 2ND OCTAL GROUP NOT DONE

2583

.SBTTL TEST #4 - CHECK WCR, BAR & BDR, & RESET CLRS 4 DEV REGS

 *TEST 4 CHECK WCR, BAR & BDR, & RESET CLRS 4 DEV REGS
 *

* THIS TEST INSURES THAT THE WCR, BAR AND BDR REGISTER BITS CAN ALL BE
 * SET, AND THAT A RESET CLEARS ALL 3 PLUS THE CSR REGISTER.
 *

012124	032777	004000	167206	TST4:	BIT	#BIT11,@SWR	:88 TEST TO SEE IF TEST NUMBER TRACER ENABLED
012132	001407				BEQ	1001\$	:88 BRANCH IF NOT
012134	104401	012142			TYPE	,1000\$	:88 TYPE: 'T # 4'
012140	000404				BR	1001\$	:88 GET OVER ASCII
012142	124	040	043	1000\$:	.ASCIIZ	/T # 4/<CRLF>	:88 THE ASCII MESSAGE
					.EVEN		
012152				1001\$:			
012152	000004				SCOPE		:PROCESS LOOPING AND TEST NUMBER INCREMENT
012154	012737	012162	001310		MOV	#999\$,\$LPERR	:SET LOOP ON ERROR TO 999\$
2584 012162	012777	010000	170332	999\$:	MOV	#MA,@CSR	:SET THE MAINTENANCE BIT AND
2585 012170	005077	170326			CLR	@CSR	:CLEAR TO DO AN INIT
2586 012174	012777	010000	170320		MOV	#MA,@CSR	:SET THE MAINTENANCE BIT TO DO THIS TEST IN MAINT MODE
2587 012202	012777	177777	170306		MOV	#-1,@WCR	:ALL ONES TO WCR
2588 012210	017737	170302	002570		MOV	@WCR,RWCR	:MOVE RECEIVED DATA TO RWCR
2589 012216	022737	177777	002570		CMP	#-1,RWCR	:SEE IF DATA WAS LOADED PROPERLY
2590 012224	001423				BEQ	4\$	:BRANCH IF OK
2591 012226	012737	012236	001310		MOV	#1\$,\$LPERR	:MOVE NEW LOOP ON ERROR LOCATION TO \$LPERR
2592 012234	000412				BR	2\$	:BRANCH OVER LOOP SETUP
2593 012236	012777	177777	170252	1\$:	MOV	#-1,@WCR	:ALL ONES TO WCR
2594 012244	017737	170246	002570		MOV	@WCR,RWCR	:MOVE RECEIVED DATA TO RWCR
2595 012252	022737	177777	002570		CMP	#-1,RWCR	:SEE IF DATA WAS LOADED PROPERLY
2596 012260	001401				BEQ	3\$	:BRANCH IF OK
2597 012262	104010			2\$:	ERROR	+10	:ATTEMPT TO SET ALL WCR BITS FAILED
2598 012264	032777	001000	167046	3\$:	BIT	#BIT9,@SWR	:SEE IF WE SHOULD LOOP BACK
2599 012272	001361				BNE	1\$	:BRANCH BACK IF SO
2600 012274	012777	177777	170216	4\$:	MOV	#-1,@BAR	:ALL ONES TO BAR
2601 012302	017737	170212	002566		MOV	@BAR,RBAR	:MOVE RECEIVED DATA TO RBAR
2602 012310	022737	177776	002566		CMP	#-2,RBAR	:SEE IF ALL BITS WERE SET (DON'T EXPECT BIT 0 TO SET)
2603 012316	001426				BEQ	8\$	:BRANCH IF OK
2604 012320	012737	012336	001310		MOV	#5\$,\$LPERR	:MOVE NEW LOOP ON ERROR LOCATION TO \$LPERR
2605 012326	012737	177776	002600		MOV	#-2,EBAR	:MOVE EXPECTED DATA TO EBAR
2606 012334	000412				BR	6\$	:BRANCH OVER LOOP SETUP
2607 012336	012777	177777	170154	5\$:	MOV	#-1,@BAR	:ALL ONES TO BAR
2608 012344	017737	170150	002566		MOV	@BAR,RBAR	:MOVE RECEIVED DATA TO RBAR
2609 012352	022737	177776	002566		CMP	#-2,RBAR	:SEE IF ALL BITS WERE SET (DON'T EXPECT BIT 0 TO SET)
2610 012360	001401				BEQ	7\$	:BRANCH IF OK
2611 012362	104012			6\$:	ERROR	+12	:ATTEMPT TO SET ALL BAR BITS TO 1 FAILED
2612 012364	032777	001000	166746	7\$:	BIT	#BIT9,@SWR	:SEE IF WE SHOULD LOOP BACK
2613 012372	001361				BNE	5\$	:BRANCH BACK IF SO
2614 012374	052777	000001	170120	8\$:	BIS	#GO,@CSR	:SET GO TO CLEAR READY TO PREPARE TO SET BAR BIT 0
2615 012402	017737	170114	002560		MOV	@CSR,RCSR	:READ CSR TO SET BIT 0 OF BAR
2616 012410	012777	177777	170102		MOV	#-1,@BAR	:ALL ONES TO BAR
2617 012416	017737	170076	002566		MOV	@BAR,RBAR	:MOVE RECEIVED DATA TO RBAR
2618 012424	022737	177777	002566		CMP	#-1,RBAR	:SEE IF ALL BITS WERE SET (*DO* EXPECT BIT 0 TO SET)
2619 012432	001431				BEQ	12\$	:BRANCH IF OK
2620 012434	012737	012452	001310		MOV	#9\$,\$LPERR	:MOVE NEW LOOP ON ERROR LOCATION TO \$LPERR
2621 012442	012737	177777	002600		MOV	#-1,EBAR	:MOVE EXPECTED DATA TO EBAR
2622 012450	000415				BR	10\$	:BRANCH OVER LOOP SETUP
2623 012452	017737	170044	002560	9\$:	MOV	@CSR,RCSR	:ACCESS CSR TO SET BIT 0 OF BAR

2624	012460	012777	177777	170032		MOV	#-1,@BAR	;ALL ONES TO BAR
2625	012466	017737	170026	002566		MOV	@BAR,RBAR	;MOVE RECEIVED DATA TO RBAR
2626	012474	022737	177777	002566		CMP	#-1,RBAR	;SEE IF ALL BITS WERE SET (*DO* EXPECT BIT 0 TO SET)
2627	012502	001401				BEQ	11\$	;BRANCH IF OK
2628	012504	104012			10\$:	ERROR	+12	;ATTEMPT TO SET ALL BAR BITS TO 1 FAILED
2629	012506	032777	001000	166624	11\$:	BIT	#BIT9,@SWR	;SEE IF WE SHOULD LOOP BACK
2630	012514	001356				BNE	9\$	;BRANCH BACK IF SO
2631	012516	012777	177777	170000	12\$:	MOV	#-1,@BDR	;ALL ONES TO BDR
2632	012524	017737	167774	002564		MOV	@BDR,RBDR	;MOVE RECEIVED DATA TO RBDR
2633	012532	022737	177777	002564		CMP	#-1,RBDR	;SEE IF DATA WAS LOADED PROPERLY
2634	012540	001423				BEQ	16\$	;BRANCH IF OK
2635	012542	012737	012552	001310		MOV	#13\$,\$LPERR	;MOVE NEW LOOP ON ERROR LOCATION TO \$LPERR
2636	012550	000412				BR	14\$	;BRANCH OVER LOOP SETUP
2637	012552	012777	177777	167744	13\$:	MOV	#-1,@BDR	;ALL ONES TO BDR
2638	012560	017737	167740	002564		MOV	@BDR,RBDR	;MOVE RECEIVED DATA TO RBDR
2639	012566	022737	177777	002564		CMP	#-1,RBDR	;SEE IF DATA WAS LOADED PROPERLY
2640	012574	001401				BEQ	15\$	;BRANCH IF OK
2641	012576	104027			14\$:	ERROR	+27	;ALL BDR BITS ARE NOT SET
2642	012600	032777	001000	166532	15\$:	BIT	#BIT9,@SWR	;SEE IF WE SHOULD LOOP BACK
2643	012606	001361				BNE	13\$	;BRANCH BACK IF SO
2644	012610	012777	010576	167704	16\$:	MOV	#10576,@CSR	;SET ALL CSR WRITEABLE BITS
2645	012616	000005				RESET		;RESET THE WORLD OF ITS TROUBLES - HOPEFULLY
2646	012620	012737	012162	001310		MOV	#999\$,\$LPERR	;RESET THE LOOP ON ERROR LOCATION
2647	012626	017737	167664	002570		MOV	@WCR,RWCR	;WAS WCR CLEARED?
2648	012634	001401				BEQ	17\$	;BRANCH IF WCR WAS CLEARED
2649	012636	104007				ERROR	+7	;RESET FAILED TO CLEAR WCR
2650	012640	017737	167654	002566	17\$:	MOV	@BAR,RBAR	;MOVE RECEIVED DATA TO RBAR
2651	012646	001403				BEQ	18\$	;BRANCH IF BAR WAS CLEARED
2652	012650	005037	002600			CLR	EBAR	;CLEAR EXPECTED LOCATION
2653	012654	104011				ERROR	+11	;RESET FAILED TO CLEAR BAR
2654	012656	017737	167640	002560	18\$:	MOV	@CSR,RCSR	;MOVE RECEIVED DATA TO RCSR
2655	012664	012737	000200	002572		MOV	#RY,ECSR	;MOVE EXPECTED DATA TO ECSR
2656	012672	004737	004060			JSR	PC,CHKCAB	;GO CHECK CABLE STATUS AND ALTER EXPECTED IF NECESSARY
2657	012676	023737	002572	002560		CMP	ECSR,RCSR	;SEE IF EXPECTED DATA WAS RECEIVED
2658	012704	001401				BEQ	19\$	;BRANCH IF IT WAS
2659	012706	104032				ERROR	+32	;READY IS NOT THE ONLY BIT SET
2660	012710	052777	010000	167604	19\$:	BIS	#MA,@CSR	;MAINT MODE (SO THAT IDR GETS ODR CONTENTS)
2661	012716	017737	167602	002564		MOV	@BDR,RBDR	;MOVE CONTENTS OF BDR TO RBDR
2662	012724	001401				BEQ	TST5	;BRANCH IF IT CORRECTLY REMAINS ZERO
2663	012726	104026				ERROR	+26	;BDR IS NOT CLEAR

2669

.SBTTL TEST #5 - DEVICE INIT CLEARS CSR, WCR, BDR AND BAR

*TEST 5 DEVICE INIT CLEARS CSR, WCR, BDR AND BAR

THIS TEST INSURES THAT DEVICE INIT CLEARS THE CSR, WCR, BDR AND BAR.

012730	032777	004000	166402	TST5:	BIT	#BIT11,@SWR	:00	TEST TO SEE IF TEST NUMBER TRACER ENABLED	
012736	001407				BEQ	1001\$	:00	BRANCH IF NOT	
012740	104401	012746			TYPE	1000\$	:00	TYPE: 'T # 5'	
012744	000404				BR	1001\$	:00	GET OVER ASCII	
012746	124	040	043	1000\$:	.ASCIIZ	/T # 5/<CRLF>	:00	THE ASCII MESSAGE	
					.EVEN				
012756	000004			1001\$:	SCOPE				
2670	012760	005077	167536	999\$:	CLR	@CSR	:	FORCE ACCESS TO CSR	
2671	012764	012777	167524		MOV	#-1,@WCR	:	ALL ONES TO WCR	
2672	012772	012777	167524		MOV	#-1,@BDR	:	ALL ONES TO BDR	
2673	013000	012777	167512		MOV	#-1,@BAR	:	ALL ONES TO BAR	
2674	013006	012777	167506		MOV	#10576,@CSR	:	SET ALL WRITEABLE BITS IN THE CSR	
2675	013014	042777	167500		BIC	#MA,@CSR	:	CLEAR THE MAINT BIT TO DO AN INIT	
2676	013022	017737	167470		MOV	@WCR,RWCR	:	MOVE RECEIVED CONTENTS TO RWCR	
2677	013030	001401			BEQ	1\$	:	BRANCH IF WCR WAS CLEARED	
2678	013032	104003			ERROR	+3	:	INIT FAILED TO CLEAR WCR	
2679	013034	017737	167460	1\$:	MOV	@BAR,RBAR	:	MOVE RECEIVED CONTENTS TO RBAR	
2680	013042	001403			BEQ	2\$	:	BRANCH IF BAR WAS CLEARED	
2681	013044	005037	002600		CLR	EBAR	:	CLEAR EXPECTED LOCATION	
2682	013050	104004			ERROR	+4	:	INIT FAILED TO CLEAR BAR	
2683	013052	017737	167444	2\$:	MOV	@CSR,RCSR	:	MOVE RECEIVED DATA TO RCSR	
2684	013060	012737	000200	002572	MOV	#RY,ECSR	:	EXPECT READY BIT ONLY TO BE SET	
2685	013066	004737	004060		JSR	PC,CHKCAB	:	GO CHECK CABLE STATUS AND ALTER EXPECTED IF NECESSARY	
2686	013072	023737	002572	002560	CMP	ECSR,RCSR	:	SEE IF EXPECTED DATA WAS RECEIVED	
2687	013100	001401			BEQ	3\$	:	BRANCH IF THEY WERE ALL CLEAR	
2688	013102	104006			ERROR	+6	:	INIT FAILED TO CLEAR ALL CSR R-W BITS	
2689	013104	012777	010000	167410	3\$:	MOV	#MA,@CSR	:	GO BACK INTO MAINT MODE (SO THAT IDR GETS ODR CONTENTS)
2690	013112	017737	167406	002564	MOV	@BDR,RBDR	:	MOVE RECEIVED CONTENTS TO RBDR	
2691	013120	001401			BEQ	TST6	:	BRANCH IF IT WAS CLEARED	
2692	013122	104005			ERROR	+5	:	INIT FAILED TO CLEAR BDR	

2702

.SBTTL TEST #6 - BIT PATTERN TEST OF WCR, BDR AND BAR REGISTERS

*TEST 6 BIT PATTERN TEST OF WCR, BDR AND BAR REGISTERS

*

THIS TEST RUNS 7 BIT PATTERNS THROUGH THE WCR, BDR AND BAR TO CHECK FOR ANY STUCK OR SHORTED PINS. LOCATION \$LPERR IS NOT SET UP AT THE START SINCE A DIFFERENT METHOD OF ERROR LOOPING IS DONE. WHEN AN ERROR IS DETERMINED TO EXIST, THE \$LPERR IS INITIALIZED TO A ROUTINE SPECIFICALLY WRITTEN FOR THAT PARTICULAR ERROR TO CREATE A VERY TIGHT LOOP.

013124	032777	004000	166206	TST6:	BIT	#BIT11,@SWR	;; TEST TO SEE IF TEST NUMBER TRACER ENABLED
013132	001407				BEQ	1001\$	;; BRANCH IF NOT
013134	104401	013142			TYPE	,1000\$	;; TYPE: 'T # 6'
013140	000404				BR	1001\$	;; GET OVER ASCII
013142	124	040	043	1000\$:	.ASCIIZ	/T # 6/<CRLF>	;; THE ASCII MESSAGE
					.EVEN		
				1001\$:	SCOPE		
2703	013152	000004			JSR	PC,CLENUP	;; SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
2704	013154	004737	004036		CLR	ERRCNT	;; CLEAR THE ERRCNT LOCATION FOR USE OF ERROR +204
2705	013160	005037	002714		MOV	#PATRNS,R1	;; MOVE ADDRESS OF BIT PATTERNS TO R1
2706	013164	012701	006250		MOV	#7,R2	;; DO 7 PATTERNS
2707	013170	012702	000007		MOV	#MA,@CSR	;; GO TO MAINTENANCE MODE & SET GO TO DISABLE BAR BIT 0
2708	013174	012777	010000	167320	1\$:	MOV	(R1),@BDR
2709	013202	011177	167316		MOV	@BDR,RBDR	;; MOVE RECEIVED DATA TO RBDR
2710	013206	017737	167312	002564	MOV	@BDR,RBDR	;; MOVE RECEIVED DATA TO RBDR
2711	013214	021137	002564		CMP	(R1),RBDR	;; SEE IF EXPECTED DATA WAS RECEIVED
2712	013220	001423			BEQ	5\$	;; BRANCH IF SO
2713	013222	012737	013236	001310	MOV	#2\$,\$LPERR	;; SET UP LOOP ON ERROR LOCATION
2714	013224	011137	002576		MOV	(R1),EBDR	;; MOVE EXPECTED DATA TO EBDR
2715	013234	000410			BR	3\$	;; SKIP OVER LOOP ON ERROR SETUP
2716	013236	011177	167262		2\$:	MOV	(R1),@BDR
2717	013242	017737	167256	002564	MOV	@BDR,RBDR	;; MOVE RECEIVED DATA TO RBDR
2718	013250	021137	002564		CMP	(R1),RBDR	;; SEE IF DATA IS OK NOW
2719	013254	001401			BEQ	4\$	;; BRANCH OUT IF SO - OK NOW
2720	013256	104205			ERROR	+205	;; BDR PATTERN NOT CORRECT
2721	013260	032777	001000	166052	4\$:	BIT	#BIT9,@SWR
2722	013266	001363			BNE	2\$	;; SEE IF WE SHOULD LOOP BACK
2723	013270	005077	167226		5\$:	CLR	@CSR
2724	013274	011177	167216		MOV	(R1),@WCR	;; CLEAR CSR TO DO AN INIT - WCR & BAR DON'T NEED MAINT MODE
2725	013300	017737	167212	002570	MOV	@WCR,RWCR	;; MOVE THE DATA TO WCR
2726	013312	001423			MOV	@WCR,RWCR	;; MOVE RECEIVED DATA TO RWCR
2727	013314	012737	013330	001310	CMP	(R1),RWCR	;; SEE IF EXPECTED DATA WAS RECEIVED
2728	013322	011137	002602		BEQ	9\$	;; BRANCH IF SO
2729	013326	000410			MOV	#6\$,\$LPERR	;; SET UP LOOP ON ERROR LOCATION
2730	013330	011177	167162		MOV	(R1),EWCR	;; MOVE EXPECTED DATA TO EWCR
2731	013334	017737	167156	002570	BR	7\$	;; SKIP OVER LOOP ON ERROR SETUP
2732	013342	021137	002570		6\$:	MOV	(R1),@WCR
2733	013346	001401			MOV	@WCR,RWCR	;; LOAD BIT PATTERN TO WCR
2734	013350	104204			CMP	(R1),RWCR	;; MOVE RECEIVED DATA TO RWCR
2735	013352	032777	001000	165760	8\$:	BEQ	8\$
2736	013360	001363			ERROR	+204	;; SEE IF DATA IS OK NOW
2737	013362	011177	167132		BIT	#BIT9,@SWR	;; BRANCH OUT IF SO - OK NOW
2738	013366	017737	167126	002566	BNE	6\$	;; WCR DATA PATTERN NOT CORRECT
2739	013374	011137	002600		9\$:	MOV	(R1),@BAR
2740	013400	042737	000001	002600	MOV	@BAR,RBAR	;; SEE IF WE SHOULD LOOP BACK
2741	013406	023737	002600	002566	MOV	(R1),EBAR	;; BRANCH BACK IF SO
					BIC	#BIT0,EBAR	;; MOVE THE DATA TO BAR
					CMP	EBAR,RBAR	;; MOVE RECEIVED DATA TO RBAR
							;; MOVE EXPECTED DATA TO EBAR
							;; DO NOT EXPECT BIT 0 TO BE READ
							;; SEE IF EXPECTED DATA WAS RECEIVED

2742	013414	001421				BEQ	13\$		:BRANCH IF SO
2743	013416	012737	013426	001310		MOV	#10\$,\$LPERR		:SET UP LOOP ON ERROR LOCATION
2744	013424	000410				BR	11\$		:SKIP OVER LOOP ON ERROR SETUP
2745	013426	011177	167066		10\$:	MOV	(R1),@BAR		:LOAD BIT PATTERN TO BAR
2746	013432	017737	167062	002566		MOV	@BAR,RBAR		:MOVE RECEIVED DATA TO RBAR
2747	013440	021137	002566			CMP	(R1),RBAR		:SEE IF DATA IS OK NOW
2748	013444	001401				BEQ	12\$		:BRANCH OUT IF SO - OK NOW
2749	013446	104203			11\$:	ERROR	+203		:BAR DATA PATTERN NOT CORRECT
2750	013450	032777	001000	165662	12\$:	BIT	#BIT9,@SWR		:SEE IF WE SHOULD LOOP BACK
2751	013456	001363				BNE	10\$		:BRANCH BACK IF SO
2752	013460	005721			13\$:	TST	(R1)+		:GO TO NEXT PATTERN
2753	013462	005302				DEC	R2		:DECREMENT THE LOOP COUNTER AND
2754	013464	001243				BNE	1\$		:BRANCH BACK IF NOT DONE
2755	013466	004737	004036			JSR	PC,CLENUP		:SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7

2762

.SBTTL TEST #7 - TEST CSR AND EIR BIT0

TEST 7 TEST CSR AND EIR BIT0
* THIS TEST INSURES THAT BIT 0 OF THE CSR IS CLEAR WHEN IN CSR MODE (BIT
* 15 CLEAR), AND SET WHEN IN EIR MODE (BIT 15 SET).
*

013472	032777	004000	165640	TST7:	BIT	#BIT11,@SWR	;## TEST TO SEE IF TEST NUMBER TRACER ENABLED
013500	001407				BEQ	1001\$	;## BRANCH IF NOT
013502	104401	013510			TYPE	,1000\$	;## TYPE: 'T # 7'
013506	000404				BR	1001\$	;## GET OVER ASCII
013510	124	040	043	1000\$:	.ASCIIZ	/T # 7/<CRLF>	;## THE ASCII MESSAGE
					.EVEN		
013520				1001\$:			
013520	000004				SCOPE		;PROCESS LOOPING AND TEST NUMBER INCREMENT
013522	012737	013530	001310		MOV	#999\$,\$LPERR	;SET LOOP ON ERROR TO 999\$
2763 013530	032737	000001	002610	999\$:	BIT	#BIT0,BORW	;TEST TO SEE IF WE ARE TESTING A DR11-W
2764 013536	001444				BEQ	TST10	;BRANCH TO NEXT TEST IF A DR11-B
2765 013540	005077	166756			CLR	@CSR	;FORCE ACCESS TO CSR
2766 013544	012737	000001	002536		MOV	#BIT0,BUT	;MOVE BIT 0 INDICATOR TO BIT UNDER TEST LOCATION
2767 013552	017737	166744	002560		MOV	@CSR,RCSR	;MOVE CSR CONTENTS TO RCSR
2768 013560	032737	000001	002560		BIT	#BIT0,RCSR	;CLEAR ALL BUT BIT 0
2769 013566	001407				BEQ	1\$	;BRANCH IF A ZERO
2770 013570	013737	002560	002572		MOV	RCSR,ECSR	;MOVE CSR TO EXPECTED DATA, ECSR AND
2771 013576	042737	000001	002572		BIC	#BIT0,ECSR	;CLEAR THE BIT THAT SHOULD HAVE BEEN CLEAR
2772 013604	104014				ERROR	+14	;CSR BIT TEST FAILED
2773 013606	012777	100000	166706	1\$:	MOV	#EIR,@CSR	;GO TO EIR MODE
2774 013614	017737	166702	002562		MOV	@CSR,REIR	;MOVE CSR CONTENTS TO RCSR
2775 013622	032737	000001	002562		BIT	#BIT0,REIR	;CLEAR ALL BUT BIT 0
2776 013630	001007				BNE	TST10	;BRANCH IF NOT A ZERO
2777 013632	013737	002562	002574		MOV	REIR,EEIR	;MOVE CONTENTS TO ECSR ALSO AND
2778 013640	052737	000001	002574		BIS	#BIT0,EEIR	;SET THE 0 BIT - EXPECTED IT TO BE 1
2779 013646	104015				ERROR	+15	;EIR BIT TEST FAILED

.SBTTL TEST #10 - ATTN CAN BE SET VIA FNCT2 & ERROR BIT SETS

; *TEST 10 ATTN CAN BE SET VIA FNCT2 & ERROR BIT SETS

```

: * THIS TEST INSURES THAT THE ATTN BIT (BIT 13) SETS VIA FNCT2 AND ERROR
: * BIT SETS.

```

[illegible]

```

1ST10:  BIT      #BIT11, @SWR      ;22 TEST TO SEE IF TEST NUMBER TRACER ENABLED
        BEQ      1001$             ;22 BRANCH IF NOT
        TYPE     ,1000$            ;22 TYPE: 'T # 10'
        BR       1001$             ;22 GET OVER ASCII
1000$:  .ASCIIZ  /T # 10/<CRLF>    ;22 THE ASCII MESSAGE

```

```

1001$: SEVEN SCOPE ;PROCESS LOOPING AND TEST NUMBER INCREMENT
MOV #999$,SLPERR ;SET LOOP ON ERROR TO 999$
BIT #BIT0,DDW ;TEST TO SEE IF WE ARE TESTING A DR11-W
BNE TST11 ;BRANCH TO NEXT TEST IF A DR11-B
999$: CLR @CSR ;FORCE ACCESS TO CSR
MOV #MA,@CSR ;MAINT
MOV @CSR,RCSR ;MOVE RECEIVED DATA TO RCSR
CMP #MA+RY,RCSR ;SEE IF EXPECTED DATA WAS RECEIVED
BEQ 1$ ;BRANCH IF THEY ARE
MOV #MA+RY,ECSR ;MOVE EXPECTED DATA TO ECSR
ERROR +16 ;READY AND MAINTENANCE ARE NOT THE ONLY BITS SET IN CSR
1$: MOVB #F2,@CSR ;SET FNCT2
MOV @CSR,RCSR ;MOVE THE CONTENTS TO RCSR
MOV RCSR,R1 ;MOVE CONTENTS TO R1 FOR BIT TEST
BIC #CB1513,R1 ;CLEAR ALL BUT BITS ERROR & ATTN FOR TEST
CMP #ER+AT,R1 ;TEST TO SEE IF ERROR AND ATTN ARE SET
BEQ 2$ ;BRANCH IF IT IS PROPERLY SET
MOV RCSR,ECSR ;MOVE EXPECTED DATA TO ECSR
JSR PC,CHKCAB ;GO CHECK CABLE STATUS AND ALTER EXPECTED IF NECESSARY
BIS #ER+AT,ECSR ;SET THE BITS THAT SHOULD HAVE BEEN SET
ERROR +17 ;ATTN AND ERROR FAILED TO SET PROPERLY
2$: BIC #AT+F2,@CSR ;CLEAR ATTN & FNCT2
MOV @CSR,RCSR ;MOVE CSR DATA TO RCSR
BIT #ER+AT,RCSR ;BIT TEST ATTN AND ERROR BITS TO SEE IF THEY ARE CLEAR
BEQ 3$ ;BRANCH IF ATTN IS CLEAR
MOV RCSR,ECSR ;MOVE EXPECTED DATA TO ECSR
BIC #ER+AT,ECSR ;CLEAR THE BITS THAT WERE SUPPOSED TO BE CLEAR
JSR PC,CHKCAB ;GO CHECK CABLE STATUS AND ALTER EXPECTED IF NECESSARY
ERROR +20 ;ATTN AND ERROR FAILED TO CLEAR PROPERLY
3$: CLR @CSR ;RETURN TO CSR

```

2820

.SBTTL TEST #11 - FNCT BIT 1 CONTROLS DSTAT BIT 9

*TEST 11 FNCT BIT 1 CONTROLS DSTAT BIT 9

* THIS TEST INSURES THAT FNCT BIT 1 CONTROLS DSTAT BIT 9.

014104	032777	004000	165226	TST11:	BIT	#BIT11,@SWR	:00	TEST TO SEE IF TEST NUMBER TRACER ENABLED
014112	001407				BEQ	1001\$	:00	BRANCH IF NOT
014114	104401	014122			TYPE	,1000\$	:00	TYPE: 'T # 11'
014120	000404				BR	1001\$	:00	GET OVER ASCII
014122	124	040	043	1000\$:	.ASCIIZ	/T # 11/<CRLF>	:00	THE ASCII MESSAGE
					.EVEN			
014132				1001\$:				
014132	000004				SCOPE			:PROCESS LOOPING AND TEST NUMBER INCREMENT
014134	012737	014142	001310		MOV	#999\$,\$LPERR		:SET LOOP ON ERROR TO 999\$
2821	014142	005077	166354	999\$:	CLR	@CSR		:CLR FUNCT BITS AND FORCE ACCESS TO CSR
2822	014146	012777	010000		MOV	#MA,@CSR		:MAINT MODE
2823	014154	017737	166342		MOV	@CSR,RCSR		:MOVE CONTENTS OF CSR TO RCSR
2824	014162	013737	002560		MOV	RCSR,ECSR		:MOVE EXPECTED TO ECSR
2825	014170	013701	002572		MOV	ECSR,R1		:MOVE CONTENTS TO R1 FOR TESTING
2826	014174	042701	177761		BIC	#CFNC,R1		:CLEAR ALL BUT THE FNCT BITS
2827	014200	001404			BEQ	1\$		:BRANCH IF THE FUNCTION BITS ARE CLEAR
2828	014202	042737	000016		BIC	#FNC,ECSR		:CLEAR THE BITS THAT SHOULD HAVE BEEN CLEAR
2829	014210	104024			ERROR	+24		:FUNCTION BIT(S) ARE NOT CLEAR
2830	014212	052777	000002	1\$:	BIS	#F1,@CSR		:SET FNCT1
2831	014220	017737	166276		MOV	@CSR,RCSR		:MOVE CONTENTS OF CSR TO RCSR
2832	014226	013737	002560		MOV	RCSR,ECSR		:MOVE EXPECTED DATA TO ECSR
2833	014234	013701	002572		MOV	ECSR,R1		:MOVE CONTENTS TO R1 FOR TEST
2834	014240	042701	170777		BIC	#CDST,R1		:CLEAR ALL BUT BITS 9, 10 & 11
2835	014244	022701	001000		CMP	#DSC,R1		:SEE IF DSTAT A AND B ARE CLEAR & C IS SET
2836	014250	001407			BEQ	TST12		:BRANCH TO NEXT TEST IF ALL CLEAR
2837	014252	042737	006000		BIC	#DAB,ECSR		:CLEAR THE DSTAT A & B BITS THAT SHOULD HAVE BEEN CLEAR
2838	014260	052737	001000		BIS	#DSC,ECSR		:SET THE DSTAT C BIT THAT SHOULD HAVE BEEN SET
2839	014266	104025			ERROR	+25		:DSTAT A, B OR C ARE NOT AS EXPECTED

2845

```
.SBTTL TEST #12 - FNCT BIT 2 CONTROLS DSTAT BIT 10
*****
*TEST 12      FNCT BIT 2 CONTROLS DSTAT BIT 10
*
*      THIS TEST INSURES THAT FNCT BIT 2 CONTROLS DSTAT BIT 10.
*
*****
014270 032777 004000 165042 TST12: BIT      #BIT11,@SWR      ;## TEST TO SEE IF TEST NUMBER TRACER ENABLED
014276 001407                                BEQ      1001$      ;## BRANCH IF NOT
014300 104401 014306                                TYPE    ,1000$      ;## TYPE: 'T # 12'
014304 000404                                BR       1001$      ;## GET OVER ASCII
014306      124      040      043 1000$: .ASCIIZ /T # 12/<CRLF> ;## THE ASCII MESSAGE
                                           .EVEN
                                           1001$:
014316                                SCOPE
014316 000004                                MOV      #999$,$LPERR ;PROCESS LOOPING AND TEST NUMBER INCREMENT
2846 014320 012737 014326 001310 999$: CLR      @CSR      ;SET LOOP ON ERROR TO 999$
2847 014332 012777 010000 166162      MOV      #MA,@CSR ;LR FUNCT BITS AND FORCE ACCESS TO CSR
2848 014340 017737 166156 002560      MOV      @CSR,RCSR ;MAINT MODE
2849 014346 013737 002560 002572      MOV      RCSR,ECSR ;MOVE CONTENTS OF CSR TO RCSR
2850 014354 013701 002572      MOV      ECSR,R1 ;MOVE EXPECTED TO ECSR
2851 014360 042701 177761      MOV      ECSR,R1 ;MOVE CONTENTS TO R1 FOR TESTING
2852 014364 001404      BIC      #CFNC,R1 ;CLEAR ALL BUT THE FNCT BITS
2853 014366 042737 000016 002572      BEQ      1$ ;BRANCH IF THE FUNCTION BITS ARE CLEAR
2854 014374 104024      BIC      #FNC,ECSR ;CLEAR THE BITS THAT SHOULD HAVE BEEN CLEAR
2855 014376 052777 000004 166116 1$: BIS      #F2,@CSR ;FUNCTION BIT(S) ARE NOT CLEAR
2856 014404 017737 166112 002560      MOV      @CSR,RCSR ;SET FNCT2
2857 014412 013737 002560 002572      MOV      RCSR,ECSR ;MOVE CONTENTS OF CSR TO RCSR
2858 014420 013701 002572      MOV      ECSR,R1 ;MOVE EXPECTED DATA TO ECSR
2859 014424 042701 170777      MOV      ECSR,R1 ;MOVE CONTENTS TO R1 FOR TEST
2860 014430 022701 002000      BIC      #CDST,R1 ;CLEAR ALL BUT THE DSTAT BITS
2861 014434 001407      CMP      #DSB,R1 ;IF DSTAT A AND C ARE CLEAR & B IS SET
2862 014436 042737 005000 002572      BEQ      TST13 ;BRANCH TO NEXT TEST IF AS EXPECTED
2863 014444 052737 002000 002572      BIC      #DAC,ECSR ;CLEAR THE BITS THAT SHOULD HAVE BEEN CLEAR
2864 014452 104025      BIS      #DSB,ECSR ;SET THE BIT THAT SHOULD HAVE BEEN SET
                                           ERROR +25 ;DSTAT A, B OR C ARE NOT AS EXPECTED
```

2870

.SBTTL TEST #13 - FNCT BIT 3 CONTROLS DSTAT BIT 11

 *TEST 13 FNCT BIT 3 CONTROLS DSTAT BIT 11
 *

* THIS TEST INSURES THAT FNCT BIT 3 CONTROLS DSTAT BIT 11.
 *

014454	032777	004000	164656	TST13:	BIT	#BIT11,ASWR	;## TEST TO SEE IF TEST NUMBER TRACER ENABLED
014462	001407				BEQ	1001\$	;## BRANCH IF NOT
014464	104401	014472			TYPE	,1000\$	;## TYPE: 'T # 13'
014470	000404				BR	1001\$	;## GET OVER ASCII
014472	124	040	043	1000\$:	.ASCIIZ	/T # 13/<CRLF>	;## THE ASCII MESSAGE
					.EVEN		
014502				1001\$:			
014502	000004				SCOPE		;PROCESS LOOPING AND TEST NUMBER INCREMENT
014504	012737	014512	001310		MOV	#999\$,\$LPERP	;SET LOOP ON ERROR TO 999\$
2871	014512	005077	166004	999\$:	CLR	@CSR	;CLR FNCT BITS AND FORCE ACCESS TO CSR
2872	014516	012777	010000		MOV	#MA,@CSR	;MAINT MODE
2873	014524	017737	165772		MOV	@CSR,RCSR	;MOVE CONTENTS OF CSR TO RCSR
2874	014532	013737	002560		MOV	RCSR,ECSR	;MOVE EXPECTED TO ECSR
2875	014540	013701	002572		MOV	ECSR,R1	;MOVE CONTENTS TO R1 FOR TESTING
2876	014544	042701	177761		BIC	#CFNC,R1	;CLEAR ALL BUT THE FNCT BITS
2877	014550	001404			BEQ	1\$	;BRANCH IF THE FUNCTION BITS ARE CLEAR
2878	014552	042737	000016		BIC	#FNC,ECSR	;CLEAR THE BITS THAT SHOULD HAVE BEEN CLEAR
2879	014560	104024			ERROR	+24	;FUNCTION BIT(S) ARE NOT CLEAR
2880	014562	052777	000010	1\$:	BIS	#F3,@CSR	;SET FNCT3
2881	014570	017737	165726		MOV	@CSR,RCSR	;MOVE CONTENTS OF CSR TO RCSR
2882	014576	013737	002560		MOV	RCSR,ECSR	;MOVE EXPECTED DATA TO ECSR
2883	014604	013701	002572		MOV	ECSR,R1	;MOVE CONTENTS TO R1 FOR TEST
2884	014610	042701	170777		BIC	#CDST,R1	;CLEAR ALL BUT DSTAT BITS
2885	014614	022701	004000		CMP	#DSA,R1	;SEE IF DSTAT B AND C ARE CLEAR & A IS SET
2886	014620	001407			BEQ	TST14	;BRANCH TO NEXT TEST IF DATA OK
2887	014622	042737	003000		BIC	#DBC,ECSR	;CLEAR THE BITS THAT SHOULD HAVE BEEN CLEAR
2888	014630	052737	004000		BIS	#DSA,ECSR	;SET THE BIT THAT SHOULD HAVE BEEN SET
2889	014636	104025			ERROR	+25	;DSTAT A, B OR C ARE NOT AS EXPECTED

2897

.SBTTL TEST #14 - EIR BLOCKS DATA XFERS FROM ODR TO IDR

*TEST 14 EIR BLOCKS DATA XFERS FROM ODR TO IDR

*

THIS TEST INSURES THAT GOING TO EIR MODE BLOCKS DATA TRANSFERS FROM
 ODR TO IDR (ODR RECEIVES DATA WHEN WRITING TO THE BDR, AND WHEN READING
 THE BDR, THE IDR IS READ).

*

014640	032777	004000	164472	TST14:	BIT	#BIT11,@SWR	;22 TEST TO SEE IF TEST NUMBER TRACER ENABLED
014646	001407				BEQ	1001\$	;22 BRANCH IF NOT
014650	104401	014656			TYPE	,1000\$	;22 TYPE: 'T # 14'
014654	000404				BR	1001\$	;22 GET OVER ASCII
014656	124	040	043	1000\$:	.ASCIIZ	/T # 14/<CRLF>	;22 THE ASCII MESSAGE
					.EVEN		
014666				1001\$:			
014666	000004				SCOPE		;PROCESS LOOPING AND TEST NUMBER INCREMENT
014670	012737	014706	001310		MOV	#999\$,\$LPERR	;SET LOOP ON ERROR TO 999\$
2898 014676	032737	000001	002610		BIT	#BIT0,BCRW	;TEST TO SEE IF WE ARE TESTING A DR11-W
2899 014704	001451				BEQ	TST15	;BRANCH TO NEXT TEST IF A DR11-B
2900 014706	005077	165610		999\$:	CLR	@CSR	;FORCE ACCESS TO CSR
2901 014712	012777	010000	165602		MOV	#MA,@CSR	;SET MAINT MODE (SO THAT IDR GETS ODR CONTENTS)
2902 014720	012777	052525	165576		MOV	#52525,@BDR	;SET ALT 0'S AND 1'S TO BDR
2903 014726	017737	165572	002564		MOV	@BDR,RBDR	;MOVE RECEIVED DATA TO RBDR
2904 014734	022737	052525	002564		CMP	#52525,RBDR	;SEE IF DATA WAS LOADED PROPERLY
2905 014742	001404				BEQ	1\$	;BRANCH IF IT WAS
2906 014744	012737	052525	002576		MOV	#52525,EBDR	;MOVE EXPECTED DATA TO EBDR
2907 014752	104031				ERROR	+31	;BDR PATTERN NOT CORRECT
2908 014754	052777	100000	165540	1\$:	BIS	#EIR,@CSR	;GO TO EIR
2909 014762	012737	052525	002576		MOV	#52525,EBDR	;MOVE EXPECTED DATA TO EBDR
2910 014770	012777	125252	165526		MOV	#125252,@BDR	;SET ALT 1'S AND 0'S TO BDR
2911 014776	017737	165522	002564		MOV	@BDR,RBDR	;MOVE RECEIVED DATA TO RBDR
2912 015004	022737	052525	002564		CMP	#52525,RBDR	;TEST FOR OLD PATTERN
2913 015012	001404				BEQ	2\$	;BRANCH IF ORIGINAL PATTERN STILL THERE
2914 015014	012737	052525	002576		MOV	#5,25,EBDR	;MOVE EXPECTED DATA TO EBDR
2915 015022	104030				ERROR	+30	;BDR SHOULD NOT HAVE BEEN LOADED WITH NEW PATTERN
2916 015024	004737	004036		2\$:	JSR	PC,CLENUP	;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7

2923

.SBTTL TEST #15 - DR11 INTERRUPTS WITH CPU AT LEVEL 3

*TEST 15 DR11 INTERRUPTS WITH CPU AT LEVEL 3

* THIS TEST INSURES THAT THE DR11 WILL INTERRUPT WITH THE CPU PRIORITY
 * AT LEVEL 3.
 * *****

015030	032777	004000	164302	TST15:	BIT	#BIT11,@SWR	;28 TEST TO SEE IF TEST NUMBER TRACER ENABLED
015036	001407				BEQ	1001\$	;28 BRANCH IF NOT
015040	104401	015046			TYPE	,1000\$	;28 TYPE: 'T # 15'
015044	000404				BR	1001\$	;28 GET OVER ASCII
015046	124	040	043	1000\$:	.ASCIIZ	/T # 15/<CRLF>	;28 THE ASCII MESSAGE
					.EVEN		
015056				1001\$:			
015056	000004				SCOPE		;PROCESS LOOPING AND TEST NUMBER INCREMENT
015060	012737	015066	001310		MOV	#999\$,\$LPERR	;SET LOOP ON ERROR TO 999\$
2924	015066	012777	010000	999\$:	MOV	#MA,@CSR	;SET MAINTENANCE BIT AND
2925	015074	005077	165422		CLR	@CSR	;CLEAR CSR TO DO AN INIT
2926	015100	012737	000140		MOV	#LEVEL3,PSW	;STATUS AT LEVEL 3
2927	015106	017737	165410		MOV	@CSR,RCSR	;MOVE CSR CONTENTS TO RCSR
2928	015114	105737	002560		TSTB	RCSR	;SEE IF READY BIT (BIT 7) IS SET
2929	015120	100406			BMI	1\$	;BRANCH IF IT IS
2930	015122	012737	000200		MOV	#RY,ECSR	;SET THE BIT THAT SHOULD HAVE BEEN SET
2931	015130	004737	004060		JSR	PC,CHKCAB	;GO CHECK CABLE STATUS AND ALTER EXPECTED IF NECESSARY
2932	015134	104022			ERROR	+22	;READY OF CSR WAS NOT SET
2933	015136	017737	165364	1\$:	MOV	@DRINV,SDRINV	;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
2934	015144	017737	165360		MOV	@DRVS,SDRVS	;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
2935	015152	012777	015242		MOV	#3\$,@DRINV	;SET UP INTERRUPT VECTOR
2936	015160	012777	010000		MOV	#MA,@CSR	;MAINT MODE
2937	015166	012737	001000		MOV	#1000,TIME	;SET THE TIME COUNTER
2938	015174	052777	000105		BIS	#IE+F2+GO,@CSR	;IE, FNCT2 AND GO
2939	015202	005337	002660	2\$:	DEC	TIME	;DECREMENT DOWN TO ZERO
2940	015206	001375			BNE	2\$	;BRANCH IF NOT THERE YET
2941	015210	017737	165306		MOV	@CSR,RCSR	;MOVE RECEIVED DATA TO RCSR
2942	015216	013777	002532		MOV	SDRINV,@DRINV	;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2943	015224	013777	002534		MOV	SDRVS,@DRVS	;RESTORE LOCATION USED AS THE INTERRUPT PS
2944	015232	104035			ERROR	+35	;DR11 FAILED TO INTERPUPT
2945	015234	005077	165262		CLR	@CSR	;CLEAR THE CSR TO DO AN INIT
2946	015240	000410			BR	TST16	;BRANCH TO THE NEXT TEST
2947	015242	062706	000004	3\$:	ADD	#4,SP	;CLEAN THE STACK AFTER THE INTERRUPT
2948	015246	013777	002532		MOV	SDRINV,@DRINV	;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2949	015254	013777	002534		MOV	SDRVS,@DRVS	;RESTORE LOCATION USED AS THE INTERRUPT PS

2956

.SBTTL TEST #16 - DR11 FAILS TO INTERRUPT WITH CPU AT LEVEL 7

*TEST 16 DR11 FAILS TO INTERRUPT WITH CPU AT LEVEL 7

* THIS TEST INSURES THAT THE DR11 FAILS TO INTERRUPT WITH THE CPU PRIORITY
 * AT LEVEL 7.

015262	032777	004000	164050	TST16:	BIT	#BIT11,@SWR	;00 TEST TO SEE IF TEST NUMBER TRACER ENABLED
015270	001407				BEQ	1001\$	;00 BRANCH IF NOT
015272	104401	015300			TYPE	1000\$	;00 TYPE: 'T # 16'
015276	000404				BR	1001\$	;00 GET OVER ASCII
015300	124	040	043	1000\$:	.ASCIIZ	/T # 16/<CRLF>	;00 THE ASCII MESSAGE
					.EVEN		
015310				1001\$:			
015310	000004				SCOPE		;PROCESS LOOPING AND TEST NUMBER INCREMENT
015312	012737	015320	001310		MOV	#999\$,\$LPERR	;SET LOOP ON ERROR TO 999\$
2957	015320	004737	004036	999\$:	JSR	PC,CLENUP	;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
2958	015324	017737	165172		MOV	@CSR,RCSR	;MOVE CSR DATA TO RCSR
2959	015332	105737	002560		TSTB	RCSR	;CLEAR ALL BUT THE READY BIT (BIT 7)
2960	015336	100411			BMI	1\$	;BRANCH IF IT IS SET
2961	015340	012737	000200		MOV	#RY,ECSR	;MOVE EXPECTED DATA TO ECSR
2962	015346	004737	004060		JSR	PC,CHKCAB	;GO CHECK CABLE STATUS AND ALTER EXPECTED IF NECESSARY
2963	015352	052737	000200		BIS	#RY,ECSR	;SET THE BIT THAT SHOULD HAVE BEEN SET
2964	015360	104022			ERROR	+22	;READY OF CSR WAS NOT SET
2965	015362	017737	165140	1\$:	MOV	@DRINV,SDRINV	;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
2966	015370	017737	165134		MOV	@DRVS,SDRVS	;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
2967	015376	012777	015464		MOV	#3\$,@DRINV	;SET UP INT VECTOR
2968	015404	012777	000340		MOV	#LEVEL7,@DRVS	
2969	015412	012737	001000		MOV	#1000,TIME	;SET TIME DELAY COUNTER
2970	015420	012777	010000		MOV	#MA,@CSR	;MAINT MODE
2971	015426	052777	000105		BIS	#IE+F2+GO,@CSR	;IE, FNCT2 AND GO
2972	015434	005337	002660	2\$:	DEC	TIME	;DECREMENT UNTIL WE GET TO ZERO
2973	015440	001375			BNE	2\$	;BRANCH BACK IF NOT ZERO YET
2974	015442	005077	165054		CLR	@CSR	;CLEAR THE CSR TO DO AN INIT
2975	015446	013777	002532		MOV	SDRINV,@DRINV	;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2976	015454	013777	002534		MOV	SDRVS,@DRVS	;RESTORE LOCATION USED AS THE INTERRUPT PS
2977	015462	000416			BR	TST17	;BRANCH TO THE NEXT TEST
2978	015464	062706	000004	3\$:	ADD	#4,SP	;RESTORE STACK
2979	015470	017737	165026		MOV	@CSR,RCSR	;MOVE RECEIVED DATA TO RCSR
2980	015476	005077	165020		CLR	@CSR	;CLEAR IE
2981	015502	013777	002532		MOV	SDRINV,@DRINV	;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
2982	015510	013777	002534		MOV	SDRVS,@DRVS	;RESTORE LOCATION USED AS THE INTERRUPT PS
2983	015516	104036			ERROR	+36	;DR11 INTERRUPTED, BUT IT SHOULDN'T HAVE

2990

.SBTTL TEST #17 - DR11 INTERRUPTS AT CORRECT BR LEVEL

*TEST 17 DR11 INTERRUPTS AT CORRECT BR LEVEL

*

* THIS TEST INSURES THAT THE DR11 WILL INTERRUPT AT THE CORRECT LEVEL AS
* DEFINED IN THE DEVICE DESCRIPTOR WORD.

*

015520	032777	004000	163612	TST17:	BIT	#BIT11, @SWR	;BB TEST TO SEE IF TEST NUMBER TRACER ENABLED
015526	001407				BEQ	1001\$	;BB BRANCH IF NOT
015530	104401	015536			TYPE	,1000\$	;BB TYPE: 'T # 17'
015534	000404				BR	1001\$	;BB GET OVER ASCII
015536	124	040	043	1000\$:	.ASCIIZ	/T # 17/<CRLF>	;BB THE ASCII MESSAGE
					.EVEN		
015546				1001\$:			
015546	000004				SCOPE		;PROCESS LOOPING AND TEST NUMBER INCREMENT
015550	012737	015632	001310		MOV	#999\$, \$LPERR	;SET LOOP ON ERROR TO 999\$
2991 015556	004737	004036			JSR	PC, CLENUP	;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
2992 015562	013737	002720	001362		MOV	DDW, \$TMP1	;MOVE DEVICE DESCRIPTOR WORD TO \$TMP1
2993 015570	006237	001362			ASR	\$TMP1	;SHIFT THE LEVEL TO THE RIGHT 5 PLACES
2994 015574	006237	001362			ASR	\$TMP1	
2995 015600	006237	001362			ASR	\$TMP1	
2996 015604	006237	001362			ASR	\$TMP1	
2997 015610	006237	001362			ASR	\$TMP1	
2998 015614	042737	177770	001362		BIC	#177770, \$TMP1	;CLEAR ALL BUT THE PRIORITY
2999 015622	012700	000003		1\$:	MOV	#3, R0	;DO 3 PRIORITY LEVELS
3000 015626	012701	005264			MOV	#LEVELS, R1	;MOVE ADDRESS OF CPU PRIORITIES TO R1
3001 015632	012777	010000	164662	999\$:	MOV	#MA, @CSR	;SET THE MAINTENANCE BIT AND
3002 015640	005077	164656			CLR	@CSR	;CLEAR TO DO AN INIT
3003 015644	011137	177776			MOV	(R1), PSW	;PUT PRIORITY INTO PSW
3004 015650	017737	164646	002560		MOV	@CSR, RCSR	;MOVE RECEIVED DATA TO RCSR
3005 015656	012737	000200	002572		MOV	#RY, ECSR	;MOVE READY BIT TO ECSR
3006 015664	004737	004060			JSP	PC, CHKCAR	;GO CHECK CABLE STATUS AND ALTER EXPECTED IF NECESSARY
3007 015670	023737	002560	002572		CMP	RCSR, ECSR	;SEE IF RECEIVED DATA MATCHES EXPECTED
3008 015676	001412				BEQ	2\$	;BRANCH IF OK
3009 015700	012737	015622	001310		MOV	#1\$, \$LPERR	;SET UP FOR POSSIBLE LOOP ON ERROR FOR THIS ERROR ONLY
3010 015706	012737	000200	002572		MOV	#RY, ECSR	;MOVE EXPECTED DATA TO ECSR
3011 015714	104022			ERROR	+22		;READY OF CSR WAS NOT SET
3012 015716	012737	015632	001310		MOV	#999\$, \$LPERR	;RETURN ORIGINAL LOOP ON ERROR ADDRESS - DID NOT LOOP
3013 015724	017737	164576	002532	2\$:	MOV	@DRINV, SDRINV	;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
3014 015732	017737	164572	002534		MOV	@DRVS, SDRVS	;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
3015 015740	012777	016052	164560		MOV	#4\$, @DRINV	;SET UP INTERRUPT VECTOR
3016 015746	012777	000340	164554		MOV	#LEVEL7, @DRVS	;SET UP INTERRUPT PS
3017 015754	012777	010000	164540		MOV	#MA, @CSR	;MAINT MODE
3018 015762	012737	000400	002660		MOV	#400, TIME	;SET DELAY COUNTER
3019 015770	052777	000105	164524		BIS	#IE+F2+GO, @CSR	;IE, FNCT2 AND GO
3020 015776	005337	002660		3\$:	DEC	TIME	;DECREMENT UNTIL WE GET TO ZERO
3021 016002	001375				BNE	3\$	;BRANCH BACK IF NOT ZERO YET
3022 016004	005077	164512			CLR	@CSR	;CLEAR CSR TO DO AN INIT
3023 016010	013777	002532	164510		MOV	SDRINV, @DRINV	;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3024 016016	013777	002534	164504		MOV	SDRVS, @DRVS	;RESTORE LOCATION USED AS THE INTERRUPT PS
3025 016024	013737	177776	002540		MOV	PSW, LEVEL	;SAVE OLD STATUS LEVEL
3026 016032	005721				TST	(R1)+	;INCREMENT R1 TO POINT TO NEXT PRIORITY LEVEL
3027 016034	005300				DEC	R0	;DECREMENT LOOP COUNTER AND
3028 016036	001275				BNE	999\$	;BRANCH BACK FOR ANOTHER TRY IF NOT DONE
3029 016040	104053			ERROR	+53		;DR11 FAILED TO INTERRUPT
3030 016042	013737	002552	002540		MOV	DRLEV, LEVEL	;SET LEVEL TO CONTAIN THE ANTICIPATED LEVEL

3031	016050	000422			BR	TST20	:BRANCH TO THE NEXT TEST
3032	016052	062706	000004	48:	ADD	#4,SP	:RESTORE STACK
3033	016056	005077	164440		CLR	@CSR	:CLEAR IE
3034	016062	013777	002532	164436	MOV	SDRINV,@DRINV	:RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3035	016070	013777	002534	164432	MOV	SDRVS,@DRVS	:RESTORE LOCATION USED AS THE INTERRUPT PS
3036	016076	042737	177437	002540	BIC	#177437,LEVEL	:CLEAR ALL BITS BUT THE BR LEVEL
3037	016104	023737	002540	002552	CMP	LEVEL,DRLEV	:SEE IF LEVEL INTERRUPTED MATCHES EXPECTED
3038	016112	001401			BEQ	TST20	:BRANCH AROUND ERROR CALL IF IT IS AS EXPECTED
3039	016114	104052			ERROR	+52	:DR11 INTERRUPTED AT WRONG LEVEL

3046

.SBTTL TEST #20 - A GO WITHOUT CLEARING ERROR CAUSES INTRPT

*TEST 20 A GO WITHOUT CLEARING ERROR CAUSES INTRPT

* THIS TEST INSURES THAT SETTING THE GO BIT WITHOUT CLEARING THE ERROR
 * BIT CAUSES AN INTERRUPT.
 * *****

016116	032777	004000	163214	TST20:	BIT	#BIT11,@SWR	:88 TEST TO SEE IF TEST NUMBER TRACER ENABLED
016124	001407				BEQ	1001\$	:88 BRANCH IF NOT
016126	104401	016134			TYPE	,1000\$	:88 TYPE: 'T # 20'
016132	000404				BR	1001\$	:88 GET OVER ASCII
016134	124	040	043	1000\$:	.ASCIZ	/T # 20/<CRLF>	:88 THE ASCII MESSAGE
					.EVEN		
016144				1001\$:			
016144	000004				SCOPE		:PROCESS LOOPING AND TEST NUMBER INCREMENT
016146	012737	016154	001310		MOV	#999\$,SLPERR	:SET LOOP ON ERROR TO 999\$
3047 016154	004737	004036		999\$:	JSR	PC,CLEUP	:SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
3048 016160	017737	164342	002532		MOV	@DRINV,SDRINV	:SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
3049 016166	017737	164336	002534		MOV	@DRVS,SDRVS	:SAVE LOCATION TO BE USED AS THE INTERRUPT PS
3050 016174	012777	164276	164324		MOV	#2\$,@DRINV	:INTERRUPT VECTOR TO 3\$
3051 016202	012777	000140	164320		MOV	#LEVEL3,@DRVS	:INTERRUPT STATUS TO LEVEL 3
3052 016210	005037	177776			CLR	PSW	:LET THE DR11 INTERRUPT
3053 016214	012737	001000	002660		MOV	#1000,TIME	:MOVE DELAY COUNTER TO LOCATION
3054 016222	012777	010101	164272		MOV	#MA+IE+GO,@CSR	:SET MAINT, IE AND GO
3055 016230	052777	020004	164264		BIS	#AT+F2,@CSR	:SET ATTN AND FNCT2
3056 016236	005337	002660		1\$:	DEC	TIME	:DECREMENT UNTIL WE REACH ZERO
3057 016242	001375				BNE	1\$	:BRANCH IF NOT ZERO YET
3058 016244	013777	002532	164254		MOV	SDRINV,@DRINV	:RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3059 016252	013777	002534	164250		MOV	SDRVS,@DRVS	:RESTORE LOCATION USED AS THE INTERRUPT PS
3060 016260	017737	164236	002560		MOV	@CSR,RCSR	:MOVE RECEIVED DATA TO RCSR
3061 016266	104035				ERROR	+35	:DR11 FAILED TO INTERRUPT
3062 016270	005077	164226			CLR	@CSR	:CLEAR THE CSR TO DO AN INIT
3063 016274	000512				BR	TST21	:BRANCH TO THE NEXT TEST
3064 016276	062706	000004		2\$:	ADD	#4,SP	:READJUST STACK AFTER THE INTERRUPT
3065 016302	013777	002532	164216		MOV	SDRINV,@DRINV	:RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3066 016310	013777	002534	164212		MOV	SDRVS,@DRVS	:RESTORE LOCATION USED AS THE INTERRUPT PS
3067 016316	017737	164200	002560		MOV	@CSR,RCSR	:MOVE RECEIVED DATA TO RCSR
3068 016324	100407				BMI	3\$	:BRANCH IF ERROR IS SET
3069 016326	013737	002560	002572		MOV	RCSR,ECSR	:MOVE EXPECTED DATA TO ECSR
3070 016334	052737	100000	002572		BIS	#ER,ECSR	:SET THE BIT THAT SHOULD HAVE BEEN SET
3071 016342	104037				ERROR	+37	:ERROR BIT SHOULD NOT BE CLEAR
3072 016344	017737	164156	002532	3\$:	MOV	@DRINV,SDRINV	:SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
3073 016352	017737	164152	002534		MOV	@DRVS,SDRVS	:SAVE LOCATION TO BE USED AS THE INTERRUPT PS
3074 016360	012777	016454	164140		MOV	#5\$,@DRINV	:INTERRUPT VECTOR TO 6\$
3075 016366	005077	164126			CLR	@BAR	:PREVENT CAUSING ANOTHER ERROR
3076 016372	012777	177777	164116		MOV	#-1,@WCR	:SET-UP WCR
3077 016400	012737	001000	002660		MOV	#1000,TIME	:LOAD 1000 IN LOCATION TIME FOR WAIT LOOP
3078 016406	052777	000001	164106		BIS	#GO,@CSR	:SET 'GO' IN CSR
3079 016414	005237	002660		4\$:	INC	TIME	:DELAY - WAIT FOR INTERRUPT
3080 016420	001375				BNE	4\$	:BRANCH BACK IF NOT ZERO
3081 016422	013777	002532	164076		MOV	SDRINV,@DRINV	:RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3082 016430	013777	002534	164072		MOV	SDRVS,@DRVS	:RESTORE LOCATION USED AS THE INTERRUPT PS
3083 016436	017737	164060	002560		MOV	@CSR,RCSR	:MOVE RECEIVED DATA TO RCSR
3084 016444	104035				ERROR	+35	:DR11 FAILED TO INTERRUPT
3085 016446	005077	164050			CLR	@CSR	:CLEAR CSR TO DO AN INIT
3086 016452	000423				BR	TST21	:BRANCH TO THE NEXT TEST

3087	016454	062706	000004	5\$:	ADD	#4,SP	;CLEAN UP STACK AFTER INTERRUPT
3088	016460	013777	002532	164040	MOV	SDRINV,@DRINV	;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3089	016466	013777	002534	164034	MOV	SDRVS,@DRVS	;RESTORE LOCATION USED AS THE INTERRUPT PS
3090	016474	017737	164022	002560	MOV	@CSR,RCSR	;MOVE RECEIVED DATA TO RCSR - IS ERROR CLEAR
3091	016502	100007			BPL	TST21	;BRANCH TO NEXT TEST IF IT IS
3092	016504	013737	002560	002572	MOV	RCSR,ECSR	;MOVE EXPECTED DATA TO ECSR
3093	016512	042737	100000	002572	BIC	#ER,ECSR	;CLEAR THE BIT THAT SHOULD HAVE BEEN CLEAR
3094	016520	104021			ERROR	+21	;ERROR BIT SHOULD HAVE BEEN CLEAR

3101

.SBTTL TEST #21 - FUNCTION BITS INC WITH MAINT MODE TRANSFERS

 *TEST 21 FUNCTION BITS INC WITH MAINT MODE TRANSFERS

*
 * THIS TEST INSURES THAT THE FUNCTION BITS INCREMENT WITH MAINTENANCE
 * MODE TRANSFERS.
 *

016522	032777	004000	162610	TST21:	BIT	#BIT11,@SWR	;00 TEST TO SEE IF TEST NUMBER TRACER ENABLED
016530	001407				BEQ	1001\$	;00 BRANCH IF NOT
016532	104401	016540			TYPE	1000\$	;00 TYPE: 'T # 21'
016536	000404				BR	1001\$	;00 GET OVER ASCII
016540	124	040	043	1000\$:	.ASCIIZ	/T # 21/<CRLF>	;00 THE ASCII MESSAGE
					.EVEN		
016550				1001\$:			
016550	000004				SCOPE		;PROCESS LOOPING AND TEST NUMBER INCREMENT
016552	012737	016574	001310		MOV	#999\$,\$LPERR	;SET LOOP ON ERROR TO 999\$
3102 016560	012737	000016	001364		MOV	#16,\$TMP2	;SET UP FUNCTION COUNT COMPARE
3103 016566	012737	177771	001360		MOV	#-7,\$TMP0	;SET UP WCR LOAD VARIABLE
3104 016574	004737	004036		999\$:	JSR	PC,CLEUP	;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
3105 016600	013777	002616	163712		MOV	INBUF,@BAR	;SET-UP BAR
3106 016606	017737	163714	002532	1\$:	MOV	@DRINV,SDRINV	;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
3107 016614	017737	163710	002534		MOV	@DRVS,SDRVS	;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
3108 016622	012777	016732	163676		MOV	#3\$,@DRINV	;INTERRUPT VECTOR
3109 016630	013777	002540	163672		MOV	LEVEL,@DRVS	;INTERRUPT VECTOR PRIORITY TO LEVEL OF DEVICE
3110 016636	013777	001360	163652		MOV	\$TMP0,@WCR	;SET UP FOR NUMBER OF TRANSFERS IN \$TMP0
3111 016644	005037	177776			CLR	PSW	;LET THE DR11 INTERRUPT
3112 016650	012777	010000	163644		MOV	#MA,@CSR	;MAINT MODE
3113 016656	012737	001000	002660		MOV	#1000,TIME	;MOVE WAIT COUNTER TO LOCATION TIME
3114 016664	052777	000501	163630		BIS	#IE+CY+GO,@CSR	;IE, CYCLE & GO
3115 016672	005337	002660		2\$:	DEC	TIME	;DECREMENT UNTIL ZERO
3116 016676	001375				BNE	2\$	;BRANCH BACK IF NOT
3117 016700	017737	163616	002560		MOV	@CSR,RCSR	;MOVE RECEIVED DATA TO RCSR
3118 016706	013777	002532	163612		MOV	SDRINV,@DRINV	;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3119 016714	013777	002534	163606		MOV	SDRVS,@DRVS	;RESTORE LOCATION USED AS THE INTERRUPT PS
3120 016722	104035				ERROR	+35	;DR11 FAILED TO INTERRUPT
3121 016724	005077	163572			CLR	@CSR	;CLEAR THE CSR TO DO AN INIT
3122 016730	000442				BR	TST22	;BRANCH TO NEXT TEST
3123 016732	062706	000004		3\$:	ADD	#4,\$P	;CLEAN UP STACK AFTER INTERRUPT
3124 016736	013777	002532	163562		MOV	SDRINV,@DRINV	;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3125 016744	013777	002534	163556		MOV	SDRVS,@DRVS	;RESTORE LOCATION USED AS THE INTERRUPT PS
3126 016752	017737	163544	002560		MOV	@CSR,RCSR	;MOVE RECEIVED DATA TO RCSR
3127 016760	013701	002560			MOV	RCSR,R1	;MOVE RECEIVED DATA TO R1 ALSO AND
3128 016764	042701	177761			BIC	#CFNC,R1	;CLEAR ALL BUT THE FUNCTION BITS
3129 016770	020137	001364			CMP	R1,\$TMP2	;SEE IF FUNCTION BIT(S) HAD INCREMENTED PROPERLY
3130 016774	001412				BEQ	4\$	;BRANCH IF THEY HAD
3131 016776	013737	002560	002572		MOV	RCSR,ECSR	;MOVE RECEIVED DATA TO EXPECTED LOCATION
3132 017004	042737	000016	002572		BIC	#FNC,ECSR	;CLEAR THE FUNCTION BIT(S) THAT WERE THERE AND
3133 017012	053737	001364	002572		BIS	\$TMP2,ECSR	;PUT FUNCTION BIT(S) EXPECTED IN THEIR PLACE
3134 017020	104212				ERROR	+212	;FUNCTION BITS DIDN'T INCREMENT IN MAINT MODE
3135 017022	005237	001360		4\$:	INC	\$TMP0	;ADJUST WCR LOAD LOCATION
3136 017026	162737	000002	001364		SUB	#2,\$TMP2	;SUBTRACT 2 FROM FUNCTION COUNT TEST LOCATION
3137 017034	001264				BNE	1\$	;BRANCH BACK FOR ANOTHER TRY

3143

.SBTTL TEST #22 - TEST FOR 10 MAINT MODE TRANSFERS

*TEST 22 TEST FOR 10 MAINT MODE TRANSFERS

*

THIS TEST CHECKS IF 10 MAINTENANCE MODE TRANSFERS CAN BE DONE.

*

017036	032777	004000	162274	TST22:	BIT	#BIT11, @SWR	;## TEST TO SEE IF TEST NUMBER TRACER ENABLED
017044	001407				BEQ	1001\$	;## BRANCH IF NOT
017046	104401	017054			TYPE	,1000\$	;## TYPE: 'T # 22'
017052	000404				BR	1001\$	;## GET OVER ASCII
017054	124	040	043	1000\$:	.ASCII2	/T # 22/<CRLF>	;## THE ASCII MESSAGE
					.EVEN		
017064				1001\$:			
017064	000004				SCOPE		;PROCESS LOOPING AND TEST NUMBER INCREMENT
017066	012737	017074	001310		MOV	#999\$, \$LPERR	;SET LOOP ON ERROR TO 999\$
3144	017074	005077	163422	999\$:	CLR	@CSR	;FORCE ACCESS TO CSR
3145	017100	012737	000010		MOV	#10, BUFLN	;BUFLN=10
3146	017106	013737	002622		MOV	BUFLN, WCLN	;PREPARE NUMBER FOR WCR
3147	017114	005437	002630		NEG	WCLN	;2'S COMPLEMENT OF BUFLN
3148	017120	004737	003472		JSR	PC, LODBUF	;LOAD IN BUFFER WITH INCREMENTING PATTERN
3149	017124	004737	003520		JSR	PC, CHKBFF	;LOAD CHECK BUFFER WITH MODIFIED INCREMENTING PATTERN
3150	017130	013777	002630	163360	MOV	WCLN, @WCR	;SET UP WCR
3151	017136	013777	002616	163354	MOV	INBUF, @BAR	;SET UP BAR
3152	017144	012777	177777	163352	MOV	#-1, @BDR	;MAINT AIDE
3153	017152	017737	163350	002532	MOV	@DRINV, SDRINV	;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
3154	017160	017737	163344	002534	MOV	@DRVS, SDRVS	;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
3155	017166	012777	017270	163332	MOV	#2\$, @DRINV	;INTERRUPT VECTOR
3156	017174	013777	002540	163326	MOV	LEVEL, @DRVS	;INTERRUPT STATUS AT PRIORITY LEVEL OF DEVICE
3157	017202	005037	177776		CLR	PSW	;LET DR11 INTERRUPT
3158	017206	012777	010000	163306	MOV	#MA, @CSR	;MAINT MODE
3159	017214	052777	000501	163300	BIS	#IE+CY+GO, @CSR	;IE, CYCLE & GO
3160	017222	012737	001000	002660	MOV	#1000, TIME	;SET LOOP COUNTER FOR WAIT
3161	017230	005337	002660	1\$:	DEC	TIME	;DECREMENT UNTIL WE GET TO ZERO
3162	017234	001375			BNE	1\$	;BRANCH BACK IF NOT ZERO
3163	017236	013777	002532	163262	MOV	SDRINV, @DRINV	;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3164	017244	013777	002534	163256	MOV	SDRVS, @DRVS	;RESTORE LOCATION USED AS THE INTERRUPT PS
3165	017252	017737	163244	002560	MOV	@CSR, RCSR	;MOVE RECEIVED DATA TO RCSR
3166	017260	104035			ERROR	+35	;DR11 FAILED TO INTERRUPT
3167	017262	005077	163234		CLR	@CSR	;CLEAR THE CSR TO DO AN INIT
3168	017266	000402			BR	3\$	;BRANCH AROUND THE STACK CLEANUP
3169	017270	062706	000004		ADD	#4, SP	;CLEAN UP STACK AFTER THE INTERRUPT
3170	017274	013777	002532	163224	MOV	SDRINV, @DRINV	;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3171	017302	013777	002534	163220	MOV	SDRVS, @DRVS	;RESTORE LOCATION USED AS THE INTERRUPT PS
3172	017310	004737	003546		JSR	PC, INTA	;GO CLEAR IE, CHECK ERROR, READY, WCR=0 AND BAR
3173	017314	104051			ERROR	+51	;CSR AND/OR WCR AND/OR BAR ARE INCORRECT
3174	017316	005037	002714		CLR	ERRCNT	;CLEAR THE ERRCNT LOCATION FOR USE OF ERPR +201
3175	017322	004737	003716		JSR	PC, DATCHK	;CHECK INBUF AFTER A MAINTENANCE MODE OPERATION
3176	017326	104201			ERROR	+201	;BUFFER DATA NOT CORRECT
3177	017330	004737	004014		JSR	PC, DATCH2	;GO BACK TO SUBROUTINE AFTER ERROR RTS IN DATCHK
3178							

3185

.SBTTL TEST #23 - TEST 10 MAINTENANCE MODE XFERS

*TEST 23 TEST 10 MAINTENANCE MODE XFERS

*

* THIS TEST CHECKS THAT 10 MAINTENANCE MODE TRANSFERS, ATTEMPTED BEFORE
 * SERVICING A PENDING INTERRUPT OF A PREVIOUS TRANSFER, ARE UNSUCCESSFUL.

*

017334 032777 004000 161776
 017342 001407
 017344 104401 017352
 017350 000404
 017352 124 040 043

TST23: BIT #BIT11,ASWR ;## TEST TO SEE IF TEST NUMBER TRACER ENABLED
 BEQ 1001\$;## BRANCH IF NOT
 TYPE ,1000\$;## TYPE: 'T # 23'
 BR 1001\$;## GET OVER ASCII
 1000\$: .ASCIIZ /T # 23/<CRLF> ;## THE ASCII MESSAGE
 .EVEN

1001\$:

017362
 017362 000004
 017364 012737 017372 001310
 3186 017372 004737 004036
 3187 017376 012737 001000 002660
 3188 017404 012737 000010 002622
 3189 017412 013737 002622 002630
 3190 017420 005437 002630
 3191 017424 004737 003472
 3192 017430 004737 003520
 3193 017434 013777 002630 163054
 3194 017442 013777 002616 163050
 3195 017450 012777 177777 163046
 3196 017456 017737 163044 002532
 3197 017464 017737 163040 002534
 3198 017472 012777 017572 163026
 3199 017500 013777 002540 163022
 3200 017506 012777 010000 163006
 3201 017514 052777 000501 163000
 3202 017522 005337 002660
 3203 017526 001375
 3204 017530 013777 002532 162770
 3205 017536 013777 002534 162764
 3206 017544 004737 003546
 3207 017550 104051
 3208 017552 005037 002714
 3209 017556 004737 003716
 3210 017562 104201
 3211 017564 004737 004014
 3212 017570 000415
 3213 017572 062706 000004
 3214 017576 013777 002532 162722
 3215 017604 013777 002534 162716
 3216 017612 017737 162704 002560
 3217 017620 104036
 3218 017622 000523
 3219 017624 012777 010000 162670
 3220 017632 012737 001000 002660
 3221 017640 013777 002630 162650
 3222 017646 013777 002616 162644
 3223 017654 017737 162646 002532
 3224 017662 017737 162642 002534
 3225 017670 012777 020042 162630

SCOPE ;PROCESS LOOPING AND TEST NUMBER INCREMENT
 MOV #999\$,SLPERR ;SET LOOP ON ERROR TO 999\$
 999\$: JSR PC,CLEUP ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
 MOV #1000,TIME ;SET DELAY
 MOV #10,BUFLEN ;BUFLEN=10
 MOV BUFLN,WCLN ;PREPARE NUMBER FOR WCR
 NEG WCLN ;2'S COMPLEMENT OF BUFLN
 JSR PC,LODBUF ;LOAD IN BUFFER WITH INCREMENTING PATTERN
 JSR PC,CHKBUF ;LOAD CHECK BUFFER WITH MODIFIED INCREMENTING PATTERN
 MOV WCLN,WCR ;SET UP WCR
 MOV INBUF,BAR ;SET UP BAR
 MOV #-1,ABDR ;MAINT AIDE
 MOV @DRINV,SDRINV ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
 MOV @DRVS,SDRVS ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
 MOV #2\$,@DRINV ;INTERRUPT VECTOR
 MOV LEVEL,@DRVS ;INTERRUPT STATUS AT PRIORITY LEVEL OF DEVICE
 MOV #MA,@CSR ;MAINT MODE
 BIS #IE+CY+GO,@CSR ;IE, CYCLE & GO
 1\$: DEC TIME ;WAIT FOR TRANSFERS TO COMPLETE
 BNE 1\$;BRANCH BACK IF WE ARE STILL WAITING
 MOV SDRINV,@DRINV ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
 MOV SDRVS,@DRVS ;RESTORE LOCATION USED AS THE INTERRUPT PS
 JSR PC,INTA ;GO CLEAR IE, CHECK ERROR, READY, WCR=0 AND BAR
 ERROR +51 ;CSR AND-OR WCR AND-OR BAR ARE INCORECT
 CLR ERRCNT ;CLEAR THE ERRCNT LOCATION FOR USE OF ERROR +201
 JSR PC,DATCHK ;CHECK INBUF AFTER A MAINTENANCE MODE OPERATION
 ERROR +201 ;BUFFER DATA NOT CORRECT
 JSR PC,DATCH2 ;GO BACK TO SUBROUTINE AFTER ERROR RTS IN DATCHK
 BR 3\$;BRANCH TO CONTINUE
 2\$: ADD #4,SP ;CLEAN UP THE STACK FROM THIS INTERRUPT
 MOV SDRINV,@DRINV ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
 MOV SDRVS,@DRVS ;RESTORE LOCATION USED AS THE INTERRUPT PS
 MOV @CSR,RCSR ;MOVE RECEIVED DATA TO RCSR
 ERROR +36 ;DR11 INTERRUPTED, BUT IT SHOULDN'T HAVE
 BR TST24 ;BRANCH TO NEXT TEST
 3\$: MOV #MA,@CSR ;MAINT MODE
 MOV #1000,TIME ;SET TIME LOOP COUNTER
 MOV WCLN,WCR ;MOVE WCLN TO WCR
 MOV INBUF,BAR ;MOVE INBUF TO BAR
 MOV @DRINV,SDRINV ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
 MOV @DRVS,SDRVS ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
 MOV #8\$,@DRINV ;SET UP INTERRUPT VECTOR

3226	017676	012777	010000	162616	MOV	#MA,@CSR	;MAINT MODE
3227	017704	052777	000501	162610	BIS	#IE+CY+GO,@CSR	;IE, CYCLE & GO
3228	017712	005337	002660		DEC	TIME	;DECREMENT TO ZERO WHILE WAITING
3229	017716	001375			BNE	4\$	;BRANCH BACK IF NOT ZERO
3230	017720	013777	002532	162600	MOV	SDRINV,@DRINV	;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3231	017726	013777	002534	162574	MOV	SDRVS,@DRVS	;RESTORE LOCATION USED AS THE INTERRUPT PS
3232	017734	017737	162562	002560	MOV	@CSR,RCSR	;MOVE RECEIVED DATA TO RCSR
3233	017742	022737	010700	002560	CMP	#10700,RCSR	;SEE IF ONLY READY, MAINT, IE & CYCLE ARE SET
3234	017750	001404			BEQ	5\$	;BRANCH IF THEY ARE
3235	017752	012737	010700	002572	MOV	#10700,ECSR	;MOVE EXPECTED DATA TO ECSR
3236	017760	104040			ERROR	+40	;CSR IS WRONG
3237	017762	017737	162532	002566	MOV	@BAR,RBAR	;MOVE RECEIVED DATA TO RBAR
3238	017770	022777	177770	162520	CMP	#-10,@WCR	;CHECK THAT NO TRANSFERS WERE MADE
3239	017776	001004			BNE	6\$	;BRANCH TO ERROR IF THERE WERE
3240	020000	023737	002616	002566	CMP	INBUF,RBAR	;CHECK THAT NO TRANSFERS WERE MADE
3241	020006	001412			BEQ	7\$	;BRANCH AROUND ERROR IF NONE
3242	020010	017737	162502	002570	MOV	@WCR,RWCR	;MOVE RECEIVED DATA TO RWCR
3243	020016	012737	177770	002602	MOV	#-10,EWCR	;MOVE EXPECTED DATA TO EWCR
3244	020024	013737	002616	002600	MOV	INBUF,EBAR	;MOVE EXPECTED DATA TO EBAR
3245	020032	104041			ERROR	+41	;TRANSFERS SHOULD HAVE BEEN INHIBITED
3246	020034	005077	162462		CLR	@CSR	;CLEAR THE CSR TO DO AN INIT
3247	020040	000414			BR	TST24	;BRANCH TO NEXT TEST
3248	020042	062706	000004		ADD	#4,SP	;CLEAN UP STACK AFTER INTERRUPT
3249	020046	013777	002532	162452	MOV	SDRINV,@DRINV	;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3250	020054	013777	002534	162446	MOV	SDRVS,@DRVS	;RESTORE LOCATION USED AS THE INTERRUPT PS
3251	020062	017737	162434	002560	MOV	@CSR,RCSR	;MOVE RECEIVED DATA TO RCSR
3252	020070	104042			ERROR	+42	;DR11 SHOULD NOT HAVE INTERRUPTED A 2ND TIME

3258

.SBTTL TEST #24 - TEST FOR 200 NPR TRANSFERS IN MAINT MODE

*TEST 24 TEST FOR 200 NPR TRANSFERS IN MAINT MODE

* THIS TEST CHECKS FOR 200 NPR TRANSFERS IN MAINTENANCE MODE.

020072	032777	004000	161240	TST24:	BIT	#BIT11,@SWR	:00 TEST TO SEE IF TEST NUMBER TRACER ENABLED
020100	001407				BEQ	1001\$	:00 BRANCH IF NOT
020102	104401	020110			TYPE	1000\$	:00 TYPE: 'T # 24'
020106	000404				BR	1001\$	:00 GET OVER ASCII
020110	124	040	043	1000\$:	.ASCIIZ	/T # 24/<CRLF>	:00 THE ASCII MESSAGE
					.EVEN		
020120				1001\$:			
020120	000004				SCOPE		:PROCESS LOOPING AND TEST NUMBER INCREMENT
020122	012737	020130	001310		MOV	#999\$,\$LPERR	:SET LOOP ON ERROR TO 999\$
3259 020130	004737	004036		999\$:	JSR	PC,CLEUP	:SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
3260 020134	005077	162362			CLR	@CSR	:FORCE ACCESS TO CSR
3261 020140	012737	000200	002622		MOV	#200,BUFLEN	:LENGTH OF BUFFER = 200
3262 020146	013737	002622	002630		MOV	BUFLEN,WCLN	:PREPARE NUMBER FOR WCR
3263 020154	005437	002630			NEG	WCLN	:2'S COMPLEMENT OF BUFLN
3264 020160	004737	003472			JSR	PC,LODBUF	:LOAD INBUF WITH INCREMENTING PATTERN
3265 020164	004737	003520			JSR	PC,CHKBUF	:LOAD CHKBUFF WITH MODIFIED INCREMENTED PATTERN
3266 020170	013777	002630	162320		MOV	WCLN,@WCR	:SET UP WCR
3267 020176	013777	002616	162314		MOV	INBUF,@BAR	:SET UP BAR
3268 020204	012777	177777	162312		MOV	#-1,@BCR	:MAINT AIDE
3269 020212	017737	162310	002532		MOV	@DRINV,SDRINV	:SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
3270 020220	017737	162304	002534		MOV	@DRVS,SDRVS	:SAVE LOCATION TO BE USED AS THE INTERRUPT PS
3271 020226	012777	020330	162272		MOV	#2\$,@DRINV	:INT VECTOR
3272 020234	013777	002540	162266		MOV	LEVEL,@DRVS	:INTERRUPT STATUS AT PRIORITY LEVEL OF DEVICE
3273 020242	005037	177776			CLR	PSW	:LET THE DR11 INTERRUPT
3274 020246	012777	010000	162246		MOV	#MA,@CSR	:MAINT MODE
3275 020254	012737	001000	002660		MOV	#1000,TIME	:SET WAIT LOOP COUNTER
3276 020262	052777	000501	162232		BIS	#IE+CY+GO,@CSR	:IE, CYCLE & GO
3277 020270	005337	002660		1\$:	DEC	TIME	:DECREMENT UNTIL WE GET TO ZERO
3278 020274	001375				BNE	1\$	:BRANCH BACK IF NOT ZERO
3279 020276	017737	162220	002560		MOV	@CSR,RCSR	:MOVE RECEIVED DATA TO RCSR
3280 020304	013777	002532	162214		MOV	SDRINV,@DRINV	:RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3281 020312	013777	002534	162210		MOV	SDRVS,@DRVS	:RESTORE LOCATION USED AS THE INTERRUPT PS
3282 020320	104043				ERROR	+43	:EXPECTED INTERRUPT DID NOT OCCUR
3283 020322	005077	162174			CLR	@CSR	:CLEAR THE CSR TO DO AN INIT
3284 020326	000402				BR	3\$	:BRANCH AROUND THE STACK CLEANUP
3285 020330	062706	000004		2\$:	ADD	#4,SP	:CLEAN UP THE STACK AFTER INTERRUPT
3286 020334	013777	002532	162164	3\$:	MOV	SDRINV,@DRINV	:RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3287 020342	013777	002534	162160		MOV	SDRVS,@DRVS	:RESTORE LOCATION USED AS THE INTERRUPT PS
3288 020350	004737	003546			JSR	PC,INTA	:GO CLEAR IE, CHECK ERROR, READY, WCR=0 AND BAR
3289 020354	104051				ERROR	+51	:CSR AND-OR WCR AND-OR BAR ARE INCORRECT
3290 020356	005037	002714			CLR	ERRCNT	:CLEAR THE ERRCNT LOCATION FOR USE OF ERROR +201
3291 020362	004737	003716			JSR	PC,DATCHK	:CHECK INBUF AFTER A MAINTENANCE MODE OPERATION
3292 020366	104201				ERROR	+201	:BUFFER DATA NOT CORRECT
3293 020370	004737	004014			JSR	PC,DATCH2	:GO BACK TO SUBROUTINE AFTER ERROR RTS IN DATCHK

3300

.SBTTL TEST #25 - DOING DATO TO DIODE MEMORY CAUSES NEX

*TEST 25 DOING DATO TO DIODE MEMORY CAUSES NEX

* THIS TEST INSURES THAT DOING A DATO TO DIODE MEMORY CAUSES THE NEX BIT
 *(BIT 14) TO SET.
 *

020374	032777	004000	160736	TST25:	BIT	#BIT11,@SWR	:88 TEST TO SEE IF TEST NUMBER TRACER ENABLED
020402	001407				BEQ	1001\$	:88 BRANCH IF NOT
020404	104401	020412			TYPE	1000\$	:88 TYPE: 'T # 25'
020410	000404				BR	1001\$	:88 GET OVER ASCII
020412	124	040	043	1000\$:	.ASCIIZ	/T # 25/<CRLF>	:88 THE ASCII MESSAGE
					.EVEN		
020422				1001\$:			
020422	000004				SCOPE		:PROCESS LOOPING AND TEST NUMBER INCREMENT
020424	012737	020432	001310		MOV	#999\$,\$LPERR	:SET LOOP ON ERROR TO 999\$
3301 020432	004737	004036		999\$:	JSR	PC,CLEUP	:SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
3302 020436	005077	162060			CLR	@CSR	:FORCE ACCESS TO CSR
3303 020442	012777	177776	162046		MOV	#-2,@WCR	:SET UP WCR
3304 020450	013777	002614	162042		MOV	DIOMEM,@BAR	:SET UP BAR
3305 020456	017737	162044	002532		MOV	@DRINV,SDRINV	:SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
3306 020464	017737	162040	002534		MOV	@DRVS,SDRVS	:SAVE LOCATION TO BE USED AS THE INTERRUPT PS
3307 020472	012777	020602	162026		MOV	#2\$,@DRINV	:INTERRUPT VECTOR TO 3\$
3308 020500	013777	002540	162022		MOV	LEVEL,@DRVS	:INTERRUPT STATUS AT PRIORITY LEVEL OF DEVICE
3309 020506	005037	177776			CLR	PSW	:LET THE DR11 INTERRUPT
3310 020512	012777	010000	162002		MOV	#MA,@CSR	:MAINT MODE
3311 020520	052777	000062	161774		BIS	#F1+X6+X7,@CSR	:SET FNCT1, XBA16, AND XBA17
3312 020526	052777	000501	161766		BIS	#IE+CY+GO,@CSR	:SET IE, CYCLE, AND GO
3313 020534	012737	001000	002660		MOV	#1000,TIME	:SET DELAY COUNTER
3314 020542	005337	002660		1\$:	DEC	TIME	:DECREMENT UNTIL ZERO
3315 020546	001375				BNE	1\$	:BRANCH BACK IF NOT
3316 020550	013777	002532	161750		MOV	SDRINV,@DRINV	:RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3317 020556	013777	002534	161744		MOV	SDRVS,@DRVS	:RESTORE LOCATION USED AS THE INTERRUPT PS
3318 020564	017737	161737	002560		MOV	@CSR,RCSR	:MOVE RECEIVED DATA TO RCSR
3319 020572	104035				ERROR	+35	:DR11 FAILED TO INTERRUPT
3320 020574	005077	161722			CLR	@CSR	:CLEAR THE CSR TO DO AN INIT
3321 020600	000431				BR	TST26	:BRANCH TO THE NEXT TEST
3322 020602	062706	000004		2\$:	ADD	#4,SP	:RESTORE THE STACK
3323 020606	013777	002532	161712		MOV	SDRINV,@DRINV	:RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3324 020614	013777	002534	161706		MOV	SDRVS,@DRVS	:RESTORE LOCATION USED AS THE INTERRUPT PS
3325 020622	017737	161374	002560		MOV	@CSR,RCSR	:MOVE CSR DATA TO RCSR
3326 020630	013701	002560			MOV	RCSR,R1	:MOVE DATA TO R1 FOR CHECKING
3327 020634	042701	037577			BIC	#37577,R1	:CLEAR ALL BUT ERROR, NEX AND READY BITS
3328 020640	022701	140200			CMP	#ER+NX+RY,R1	:SEE IF ALL THESE BITS ARE SET
3329 020644	001407				BEQ	TST26	:BRANCH TO THE NEXT TEST IF THEY ARE ALL SET
3330 020646	013737	002560	002572		MOV	RCSR,ECSR	:MOVE EXPECTED DATA TO ECSR
3331 020654	052737	140200	002572		BIS	#ER+NX+RY,ECSR	:SET THE BIT THAT SHOULD HAVE BEEN SET
3332 020662	104040				ERROR	+40	:CSR IS WRONG

3339

.SBTTL TEST #26 - CROSSING 32K DOESN'T CAUSE BAOF OR FORCE ERROR

*TEST 26 CROSSING 32K DOESN'T CAUSE BAOF OR FORCE ERROR

* *

* THIS TEST INSURES THAT CROSSING THE 32K BOUNDARY DOES NOT CAUSE A BAOF
* OR FORCE ERROR.
* *

020664	032777	004000	160446	TST26:	BIT	#BIT11,@SWR	:00 TEST TO SEE IF TEST NUMBER TRACER ENABLED
020672	001407				BEQ	1001\$	:00 BRANCH IF NOT
020674	104401	020702			TYPE	1000\$	:00 TYPE: 'T # 26'
020700	000404				BR	1001\$	:00 GET OVER ASCII
020702	124	040	043	1000\$:	.ASCIIZ	/T # 26/<CRLF>	:00 THE ASCII MESSAGE
					.EVEN		
020712				1001\$:			
020712	000004				SCOPE		:PROCESS LOOPING AND TEST NUMBER INCREMENT
020714	012737	020722	001310		MOV	#999\$,\$LPERR	:SET LOOP ON ERROR TO 999\$
3340 020722	004737	004036		999\$:	JSR	PC,CLEUP	:SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
3341 020726	012777	177760	161562		MOV	#-20,@WCR	:SET UP WCR
3342 020734	012777	177776	161556		MOV	#-2,@BAR	:SET UP BAR FOR PROC STATUS ADDRESS
3343 020742	017737	161560	002532		MOV	@DRINV,SDRINV	:SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
3344 020750	017737	161554	002534		MOV	@DRVS,SDRVS	:SAVE LOCATION TO BE USED AS THE INTERRUPT PS
3345 020756	012777	021054	161542		MOV	#2\$,@DRINV	:INTERRUPT VECTOR TO 3\$
3346 020764	013777	002540	161536		MOV	LEVEL,@DRVS	:INTERRUPT STATUS AT PRIORITY LEVEL OF DEVICE
3347 020772	012737	001000	002660		MOV	#1000,TIME	:SET WAIT LOOP COUNTER
3348 021000	005037	177776			CLR	PSW	:LET THE DR11 INTERRUPT
3349 021004	012777	010000	161510		MOV	#MA,@CSR	:MAINT MODE
3350 021012	052777	000563	161502		BIS	#563,@CSR	:CYCLE, IE, FNCT1, XBA17, XBA16, AND GO TO CSR
3351 021020	005337	002660		1\$:	DEC	TIME	:DECREMENT UNTIL WE GET TO ZERO
3352 021024	001375				BNE	1\$	:BRANCH BACK IF NOT ZERO
3353 021026	013777	002532	161472		MOV	SDRINV,@DRINV	:RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3354 021034	013777	002534	161466		MOV	SDRVS,@DRVS	:RESTORE LOCATION USED AS THE INTERRUPT PS
3355 021042	017737	161454	002560		MOV	@CSR,RCSR	:MOVE CSR CONTENTS TO RCSR
3356 021050	104035				ERROR	+35	:DR11 FAILED TO INTERRUPT
3357 021052	000433				BR	TST27	:BRANCH TO NEXT TEST
3358 021054	062706	000004		2\$:	ADD	#4,SP	:CLEAN UP STACK AFTER INTERRUPT
3359 021060	013777	002532	161440		MOV	SDRINV,@DRINV	:RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3360 021066	013777	002534	161434		MOV	SDRVS,@DRVS	:RESTORE LOCATION USED AS THE INTERRUPT PS
3361 021074	017737	161422	002560		MOV	@CSR,RCSR	:MOVE CSR CONTENTS TO RCSR
3362 021102	013701	002560			MOV	RCSR,R1	:MOVE CONTENTS TO R1 FOR TESTING
3363 021106	042701	037577			BIC	#37577,R1	:CLEAR ALL BUT THE ERROR AND READY BITS
3364 021112	022701	140200			CMP	#ER+RY+NX,R1	:SEE IF ERROR, READY AND NEX ARE SET
3365 021116	001411				BEQ	TST27	:BRANCH TO NEXT TEST IF THEY ARE
3366 021120	013737	002560	002572		MOV	RCSR,ECSR	:MOVE EXPECTED DATA TO ECSR
3367 021126	052737	140200	002572		BIS	#ER+RY+NX,ECSR	:SET THE BITS THAT SHOULD HAVE BEEN SET
3368 021134	104040				ERROR	+40	:CSR IS WRONG
3369 021136	005077	161360			CLR	@CSR	:CLEAR THE CSR TO DO AN INIT

3376

.SBTTL TEST #27 - CHECK ACTUAL POSITION OF 2-N BURST SWITCH

*TEST 27 CHECK ACTUAL POSITION OF 2-N BURST SWITCH

*

* THIS TEST INSURES THAT THE 2-N BURST SWITCH IS IN THE POSITION THAT
 * THE DEVICE DESCRIPTOR WORD SAYS IT SHOULD BE.

*

021142	032777	004000	160170	TST27:	BIT	#BIT11,@SWR	;22 TEST TO SEE IF TEST NUMBER TRACER ENABLED
021150	001407				BEQ	1001\$	;22 BRANCH IF NOT
021152	104401	021160			TYPE	,1000\$	;22 TYPE: 'T # 27'
021156	000404				BR	1001\$	;22 GET OVER ASCII
021160	124	040	043	1000\$:	.ASCIZ	/T # 27/<CRLF>	;22 THE ASCII MESSAGE
					.EVEN		

1001\$:

SCOPE ;PROCESS LOOPING AND TEST NUMBER INCREMENT

021170	000004				MOV	#999\$,\$LPERR	;SET LOOP ON ERROR TO 999\$
021172	012737	021210	001310		MOV	#BIT0,BORW	;TESTING A 'B' OR A 'W'?
3377 021200	032737	000001	002610		BIT	INOUT	;BRANCH TO INOUT IF DR11-B
3378 021206	001456				BEQ	INOUT	;BRANCH TO INOUT IF DR11-B
3379 021210	004737	004036		999\$:	JSR	PC,CLENUP	;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
3380 021214	012777	100000	161300		MOV	#EIR,@CSR	;GO TO EIR MODE
3381 021222	017737	161274	002562		MOV	@CSR,REIR	;MOVE EIR DATA TO REIR
3382 021230	005077	161266			CLR	@CSR	;GO BACK TO CSR
3383 021234	013701	002562			MOV	REIR,R1	;MOVE DATA TO R1 ALSO
3384 021240	000301				SWAB	R1	;GET BIT 8 INTO BIT 0 BY SWAPPING BYTES
3385 021242	006301				ASL	R1	;MOVE BIT 0 INTO BIT 1
3386 021244	042701	177775			BIC	#CBIT1,R1	;CLEAR ALL BUT BIT 1
3387 021250	013702	002720			MOV	DDW,R2	;PUT DEVICE DESCRIPTOR WORD IN R2
3388 021254	042702	177775			BIC	#CBIT1,R2	;CLEAR ALL BUT BIT 1
3389 021260	001402				BEQ	1\$	;BRANCH IF IT IS CLEAR
3390 021262	005002				CLR	R2	;CLEAR THE BIT
3391 021264	000402				BR	2\$	;GO TEST THE BIT
3392 021266	012702	000002		1\$:	MOV	#BIT1,R2	;SET THE BIT
3393 021272	020102			2\$:	CMP	R1,R2	;SEE IF RECEIVED MATCHES EXPECTED
3394 021274	001017				BNE	5\$	;BRANCH TO CHECK FOR LOOP ON TEST
3395 021276	013737	002562	002574		MOV	REIR,EEIR	;MOVE EXPECTED DATA TO EEIR
3396 021304	032737	000400	002574		BIT	#BIT8,EEIR	;TEST STATE OF BIT 8
3397 021312	001404				BEQ	3\$	;BRANCH IF IT IS CLEAR
3398 021314	042737	000400	002574		BIC	#BIT8,EEIR	;REVERSE STATE - EXPECTED CLEAR
3399 021322	000403				BR	4\$	;GO CALL ERROR
3400 021324	052737	000400	002574	3\$:	BIS	#BIT8,EEIR	;REVERSE STATE - EXPECTED SET
3401 021332	104054			4\$:	ERROR	+54	;2-N CYCLE BURST SWITCH IN WRONG POSITION
3402 021334	032777	040400	157776	5\$:	BIT	#BIT14+BIT8,@SWR	;SEE IF WE SHOULD LOOP ON THIS TEST
3403 021342	001322				BNE	999\$	;BRANCH BACK IF SO

```
3404                                .SBTTL   CODE TO CHECK CABLE STATUS FOR EXECUTION OF CABLE TESTS
3405                                ;           CABLE MODE TESTING (WRAP-AROUND CABLE IN USER SLOTS)
3406                                ;
3407                                ;        TESTS 30 THRU 37 ARE PERFORMED IF BIT 2 OF DEVICE DESCRIPTOR WORD IS
3408                                ;        SET, INDICATING CABLE IS IN.
3409                                ;
3410 021344   032737   000004   002720   INOUT: BIT        #BIT2,DDW        ;SEE IF CABLE IS IN
3411 021352   001002                        BNE        TST30        ;BRANCH TO NEXT TEST IF CABLE IS IN
3412 021354   000137   024360                JMP        ENDEV        ;JUMP TO ENDEV - TESTS ARE NOT TO BE DONE
```

3420

.SBTTL TEST #30 - CHECK CSR BIT PATTERNS WITH MAINT BIT CLEAR

*TEST 30 CHECK CSR BIT PATTERNS WITH MAINT BIT CLEAR

*

* THIS TEST SETS ALL POSSIBLE COMBINATIONS OF SET BITS IN THE CSR WITH
 * THE MAINTENANCE BIT CLEAR, AND COMPARES THE RECEIVED CSR CONTENTS WITH
 * THAT OF THE EXPECTED PATTERNS IN 'MAICLR'.
 *

*

021360	032777	004000	157752	TST30:	BIT	#BIT11,@SWR	:88	TEST TO SEE IF TEST NUMBER TRACER ENABLED
021366	001407				BEQ	1001\$	:88	BRANCH IF NOT
021370	104401	021376			TYPE	,1000\$	:88	TYPE: 'T # 30'
021374	000404				BR	1001\$	:88	GET OVER ASCII
021376	124	040	043	1000\$:	.ASCIIZ	/T # 30/<CRLF>	:88	THE ASCII MESSAGE
					.EVEN			
021406				1001\$:				
021406	000004				SCOPE			:PROCESS LOOPING AND TEST NUMBER INCREMENT
021410	012737	021444	001310		MOV	#999\$,\$LPERR		:SET LOOP ON ERROR TO 999\$
3421	021416	004737	004036		JSR	PC,CLEUP		:SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
3422	021422	005037	002714		CLR	ERRCNT		:CLEAR THE ERRCNT LOCATION FOR USE OF ERROR +202
3423	021426	012700	000002		MOV	#2,R0		:DO 2 SETS OF 200 PATTERNS
3424	021432	012701	006266		MOV	#MAICLR,R1		:MOVE ADDRESS OF EXPECTED PATTERNS TO R1
3425	021436	005002			CLR	R2		:START WITH PATTERN ZERO
3426	021440	012703	000200	1\$:	MOV	#200,R3		:MOVE 200 TO THE LOOP COUNTER
3427	021444	052777	010000	161050	999\$:	BIS	#MA,@CSR	:SET MAINTENANCE AND
3428	021452	005077	161044		CLR	@CSR		:CLEAR TO DO AN INIT
3429	021456	017737	161040	002560	MOV	@CSR,RCSR		:MOVE RECEIVED DATA TO RCSR
3430	021464	022737	000200	002560	CMP	#RY,RCSR		:MAKE SURE READY BIT IS THE ONLY BIT SET
3431	021472	001404			BEQ	2\$		:BRANCH IF SO
3432	021474	012737	000200	002572	MOV	#RY,ECSR		:MOVE EXPECTED DATA TO ECSR
3433	021502	104032			ERROR	+32		:READY IS NOT THE ONLY BIT SET
3434	021504	012777	177777	161004	2\$:	MOV	#-1,@WCR	:MOVE 1 WORD COUNT TO WCR IN CASE OF IE ENABLED
3435	021512	012777	053700	161000	MOV	#NOCARE,@BAR		:MOVE A NOT-CARE ADDRESS TO BAR FOR SAME REASON
3436	021520	010277	160776		MOV	R2,@CSR		:SET THE PARTICULAR FUNCTION BITS IN CSR
3437	021524	017737	160772	002560	MOV	@CSR,RCSR		:MOVE RECEIVED DATA TO RCSR
3438	021532	011137	002572		MOV	(R1),ECSR		:MOVE EXPECTED DATA TO ECSR
3439	021536	023737	002572	002560	CMP	ECSR,RCSR		:COMPARE EXPECTED WITH RECEIVED
3440	021544	001427			BEQ	6\$		:BRANCH IF OK
3441	021546	012737	000401	021556	MOV	#CY+GO,3\$		:REESTABLISH BIT PATTERN
3442	021554	040227			BIC	R2,(PC)+		:SEE IF BOTH CYCLE AND GO WERE SET
3443	021556	000401		3\$:	.WORD	CY+GO		:LOCATION TO HOLD BOTH CYCLE AND GO BITS
3444	021560	001016			BNE	5\$		:BRANCH TO ERROR ONLY IF EITHER OR BOTH BITS WERE CLEAR
3445	021562	005737	002712		TST	MEMGMT		:SEE IF MEMORY MANAGEMENT IS OUT THERE
3446	021566	001404			BEQ	4\$		:BRANCH IF NOT
3447	021570	032737	000060	002572	BIT	#X6+X7,ECSR		:SEE IF EITHER XBA16 OR XBA17 ARE SET
3448	021576	001407			BEQ	5\$		:BRANCH TO ERROR IF BOTH ARE CLEAR
3449	021600	052737	140000	002572	4\$:	BIS	#ER+NX,ECSR	:SET THE ERROR AND NEX BITS - EXPECT THEM TO SET
3450	021606	023737	002572	002560	CMP	ECSR,RCSR		:NOW SEE IF DATA MATCHES
3451	021614	001403			BEQ	6\$		:BRANCH AROUND ERROR IF IT DOES
3452	021616	010237	002536	5\$:	MOV	R2,BUT		:MOVE THE BITS SET INTO CSR TO THE BUT LOCATION
3453	021622	104202			ERROR	+202		:CSR PATTERN NOT CORRECT
3454	021624	062701	000002	6\$:	ADD	#2,R1		:INCREMENT R1 TO NEXT EXPECTED PATTERN
3455	021630	005202			INC	R2		:INCREMENT THE PATTERN
3456	021632	005303			DEC	R3		:DECREMENT THE LOOP COUNTER
3457	021634	001303			BNE	999\$		:BRANCH BACK IF NOT DONE
3458	021636	062702	000200		ADD	#200,R2		:ADD 200 TO PATTERN LOCATION
3459	021642	005300			DEC	R0		:DECREMENT THE LOOP COUNTER AND

CZDRL:O-DR11 GEN NPR INTFC MACRO M1113 27-OCT-80 15:53 M 9
TEST #30 - CHECK CSR BIT PATTERNS WITH MAINT BIT CLEAR PAGE 109-1

SEQ 0116

3460 021644 001275

BNE 1\$

;BRANCH BACK IF 2ND OCTAL GROUP NOT DONE

3466

.SBTTL TEST #31 - CHECK BAR WITH CSR CLEAR

 *TEST 31 CHECK BAR WITH CSR CLEAR
 *
 * THIS TEST CHECKS THAT BAR BIT 0 IS CLEAR WITH CSR CLEAR (CSR=0).
 *

021646	032777	004000	157464	TST31:	BIT	#BIT11,@SWR	;88 TEST TO SEE IF TEST NUMBER TRACER ENABLED
021654	001407				BEQ	1001\$	;88 BRANCH IF NOT
021656	104401	021664			TYPE	,1000\$	;88 TYPE: 'T # 31'
021662	000404				BR	1001\$	;88 GET OVER ASCII
021664	124	040	043	1000\$:	.ASCIIZ	/T # 31/<CRLF>	;88 THE ASCII MESSAGE
					.EVEN		
021674				1001\$:			
021674	000004				SCOPE		;PROCESS LOOPING AND TEST NUMBER INCREMENT
021676	012737	021704	001310		MOV	#999\$,\$LPERR	;SET LOOP ON ERROR TO 999\$
3467	021704	004737	004036	999\$:	JSR	PC,CLEUP	;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
3468	021710	012777	001360		MOV	#\$TMP0,@BAR	;PUT AN ADDRESS IN THE BAR
3469	021716	012777	000001		MOV	#GO,@CSR	;SET JUST THE GO BIT TO CLEAR THE READY BIT
3470	021724	017737	160572		MOV	@CSR,RCSR	;MOVE RECEIVED DATA TO RCSR
3471	021732	001403			BEQ	1\$	;BRANCH AROUND ERROR IF EQUAL TO ZERO
3472	021734	005037	002572		CLR	ECSR	;MOVE EXPECTED DATA TO ECSR
3473	021740	104040			ERROR	+40	;CSR IS WRONG
3474	021742	017737	160552	1\$:	MOV	@BAR,RBAR	;MOVE RECEIVED DATA TO RBAR
3475	021750	032737	000001		BIT	#BIT0,RBAR	;SEE IF THIS BIT IS CLEAR
3476	021756	001407			BEQ	TST32	;BRANCH TO NEXT TEST IF IT WAS
3477	021760	013737	002566		MOV	RBAR,EBAR	;MOVE EXPECTED DATA TO EBAR
3478	021766	042737	000001		BIC	#BIT0,EBAR	;CLEAR THE BIT THAT SHOULD HAVE BEEN CLEAR
3479	021774	104044			ERROR	+44	;BAR IS WRONG

3486

.SBTTL TEST #32 - TEST 7 SINGLE DATI NON BURST MODE TRANSFERS

 *TEST 32 TEST 7 SINGLE DATI NON BURST MODE TRANSFERS

THIS TEST DOES 7 BIT PATTERNS OF SINGLE DATI NON BURST MODE TRANSFERS,
 AND THAT THEY ARE DONE PROPERLY.

021776	032777	004000	157334	TST32:	BIT	#BIT11,@SWR	;88 TEST TO SEE IF TEST NUMBER TRACER ENABLED
022004	001407				BEQ	1001\$	;88 BRANCH IF NOT
022006	104401	022014			TYPE	,1000\$	;88 TYPE: 'T # 32'
022012	000404				BR	1001\$	;88 GET OVER ASCII
022014	124	040	043	1000\$:	.ASCIIZ	/T # 32/<CRLF>	;88 THE ASCII MESSAGE
					.EVEN		
				1001\$:			
022024					SCOPE		;PROCESS LOOPING AND TEST NUMBER INCREMENT
022024	000004				MOV	#999\$,\$LPERR	;SET LOOP ON ERROR TO 999\$
022026	012737	022050	001310	1\$:	MOV	#7,R2	;SET UP LOOP COUNTER - DO 7 BIT PATTERNS
3487 022034	012702	000007			CLR	ERRCNT	;CLEAR THE ERRCNT LOCATION FOR USE OF ERROR +201
3488 022040	005037	002714			MOV	#PATRNS,R3	;MOVE ADDRESS OF PATTERNS TO R3
3489 022044	012703	006250		999\$:	MOV	@DRINV,SDRINV	;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
3490 022050	017737	160452	002532		MOV	@DRVS,SDRVS	;SAVE LOCATION TO BE USED AS THE INTERRUPT PS
3491 022056	017737	160446	002534		MOV	#3\$,@DRINV	;INTERRUPT VECTOR TO 4\$
3492 022064	012777	022226	160434		MOV	LEVEL,@DRVS	;INTERRUPT STATUS AT PRIORITY LEVEL OF DEVICE
3493 022072	013777	002540	160430		CLR	PSW	;LET THE DR11 INTERRUPT
3494 022100	005037	177776			MOV	#MA,@CSR	;DO AN INIT BY SETTING AND
3495 022104	012777	010000	160410		CLR	@CSR	;CLEARING THE CSR MAINTENANCE BIT
3496 022112	005077	160404			MOV	#-1,@WCR	;SET UP FOR 1 TRANSFER
3497 022116	012777	177777	160372		MOV	#NPR1,@BAR	;TRANSFER FROM BUS ADDRESS IN NPR1
3498 022124	012777	002612	160366		CLR	@BDR	;GET READY TO RECEIVE DATA
3499 022132	005077	160366			MOV	(R3),NPR1	;SET UP TRANSFER DATA
3500 022136	011337	002612			MOV	#F3+IE,@CSR	;SET THE NON BURST (FNCT3) AND IE
3501 022142	012777	000110	160352		BIS	#CY+GO,@CSR	;SET THE CYCLE AND GO BITS
3502 022150	052777	000401	160344		CLR	TIME	;CLEAR THE TIME LOCATION FOR WAIT LOOP
3503 022156	005037	002660		2\$:	INC	TIME	;INCREMENT UNTIL WE GET TO ZERO AGAIN
3504 022162	005237	002660			BNE	2\$	;BRANCH BACK IF WE AREN'T THERE YET
3505 022166	001375				MOV	SDRINV,@DRINV	;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3506 022170	013777	002532	160330		MOV	SDRVS,@DRVS	;RESTORE LOCATION USED AS THE INTERRUPT PS
3507 022176	013777	002534	160324		MOV	@CSR,RCSR	;MOVE RECEIVED DATA TO RCSR
3508 022204	017737	160312	002560		MOV	(R3),ENPR1	;MOVE PATTERN TO ENPR1
3509 022212	011337	002604			ERROR	+35	;DR11 FAILED TO INTERRUPT
3510 022216	104035				CLR	@CSR	;CLEAR THE CSR TO DO AN INIT
3511 022220	005077	160276			BR	5\$	;BRANCH TO SEE IF THERE ARE ANY MORE PATTERNS TO CHECK
3512 022224	000450			3\$:	ADD	#4,SP	;CLEAN STACK AFTER INTERRUPT
3513 022226	062706	000004			MOV	SDRINV,@DRINV	;RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3514 022232	013777	002532	160266		MOV	SDRVS,@DRVS	;RESTORE LOCATION USED AS THE INTERRUPT PS
3515 022240	013777	002534	160262		JSR	PC,ERRCHK	;CLEAR IE, CHECK FOR ERROR
3516 022246	004737	004254			ERROR	+21	;ERROR BIT SHOULD HAVE BEEN CLEAR
3517 022252	104021				MOV	@BAR,RBAR	;MOVE RECEIVED DATA TO RBAR
3518 022254	017737	160240	002566		MOV	#NPR1+3,EBAR	;MOVE EXPECTED DATA TO EBAR
3519 022262	012737	002615	002600		MOV	@BDR,RBDR	;MOVE RECEIVED DATA TO RBDR
3520 022270	017737	160230	002564		MOV	(R3),EBDR	;MOVE EXPECTED DATA TO EBDR
3521 022276	011337	002576			MOV	@WCR,RWCR	;MOVE RECEIVED DATA TO RWCR
3522 022302	017737	160210	002570		CLR	EWCR	;MOVE EXPECTED DATA TO EWCR
3523 022310	005037	002602			CMP	RBAR,EBAR	;COMPARE RECEIVED WITH EXPECTED
3524 022314	023737	002566	002600		BNE	4\$	;BRANCH IF WRONG
3525 022322	001010				CMP	RBDR,EBDR	;COMPARE RECEIVED WITH EXPECTED
3526 022324	023737	002564	002576				

3527	022332	001004			BNE	4\$	;BRANCH IF WRONG
3528	022334	023737	002570	002602	CMP	RWCR,EWCR	;COMPARE RECEIVED WITH EXPECTED
3529	022342	001401			BEQ	5\$	;BRANCH IF OK
3530	022344	104211			ERROR	+211	;CSR AND-OR WCR AND-OR BAR ARE INCORRECT
3531	022346	062703	000002		ADD	#2,R3	;INCREMENT TO NEXT PATTERN
3532	022352	005302			DEC	R2	;DECREMENT THE LOOP COUNTER
3533	022354	001235			BNE	999\$	;BRANCH BACK IF NOT ZERO YET

3539

.SBTTL TEST #33 - TEST STRING OF 200 DATIS BURST MODE XFERS

 :*TEST 33 TEST STRING OF 200 DATIS BURST MODE XFERS
 :*

:* THIS TEST DOES 200 DATI TRANSFERS IN BURST MODE.
 :*

022356	032777	004000	156754	TST33:	BIT	#BIT11,@SWR	:88 TEST TO SEE IF TEST NUMBER TRACER ENABLED
022364	001407				BEQ	1001\$	:88 BRANCH IF NOT
022366	104401	022374			TYPE	,1000\$	:88 TYPE: 'T # 33'
022372	000404				BR	1001\$	:88 GET OVER ASCII
022374	124	040	043	1000\$:	.ASCIIZ	/T # 33/<CRLF>	:88 THE ASCII MESSAGE
					.EVEN		

1001\$:

022404					SCOPE		:PROCESS LOOPING AND TEST NUMBER INCREMENT
022404	000004				MOV	#999\$,\$LPERR	:SET LOOP ON ERROR TO 999\$
022406	012737	022414	001310	999\$:	JSR	PC,CLENUP	:SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
3540 022414	004737	004036			MOV	#200,BUFLEN	:LENGTH OF BUFFER=200
3541 022420	012737	000200	002622		JSR	PC,LODBUF	:LOAD THE BUFFER WITH INCREMENTING PATTERN
3542 022426	004737	003472			MOV	#-200,WCLEN	:PREPARE NUMBER FOR WCR
3543 022432	012737	177600	002630		MOV	WCLEN,@WCR	:SET UP WCR
3544 022440	013777	002630	160050		MOV	INBUF,@BAR	:SET UP BAR
3545 022446	013777	002616	160044		MOV	#-1,@BDR	:MAINT AIDE
3546 022454	012777	177777	160042		MOV	@DRINV,SDRINV	:SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
3547 022462	017737	160040	002532		MOV	@DRVS,SDRVS	:SAVE LOCATION TO BE USED AS THE INTERRUPT PS
3548 022470	017737	160034	002534		MOV	#2\$,@DRINV	:INT VECTOR
3549 022476	012777	022606	160022		MOV	LEVEL,@DRVS	:INTERRUPT STATUS TO LEVEL OF DEVICE
3550 022504	013777	002540	160016		CLR	PSW	:LET THE DR11 INTERRUPT
3551 022512	005037	177776			MOV	#IE,@CSR	:SET INTERRUPT ENABLE
3552 022516	012777	000100	157776		BIS	#CY+GO,@CSR	:CYCLE, GO
3553 022524	052777	000401	157770		MOV	#1000,TIME	:WAIT FOR INTERRUPT
3554 022532	012737	001000	002660	1\$:	DEC	TIME	:DECREMENT UNTIL WE REACH ZERO
3555 022540	005337	002660			BNE	1\$	:BRANCH BACK IF NOT ZERO
3556 022544	001375				MOV	SDRINV,@DRINV	:RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3557 022546	013777	002532	157752		MOV	SDRVS,@DRVS	:RESTORE LOCATION USED AS THE INTERRUPT PS
3558 022554	013777	002534	157746		MOV	@CSR,RCSR	:MOVE RECEIVED DATA TO RCSR
3559 022562	017737	157734	002560		ERROR	+35	:DR11 FAILED TO INTERRUPT
3560 022570	104035				MOV	#MA,@CSR	:SET THE MAINTENANCE BIT AND
3561 022572	012777	010000	157722		CLR	@CSR	:CLEAR THE CSR TO DO AN INIT
3562 022600	005077	157716			BR	TST34	:BRANCH TO NEXT TEST
3563 022604	000426			2\$:	ADD	#4,SP	:CLEAN UP STACK AFTER INTERRUPT
3564 022606	062706	000004			MOV	SDRINV,@DRINV	:RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3565 022612	013777	002532	157706		MOV	SDRVS,@DRVS	:RESTORE LOCATION USED AS THE INTERRUPT PS
3566 022620	013777	002534	157702		JSR	PC,INTA	:GO CLEAR IE, CHECK ERROR, READY, WCR=0 AND BAR
3567 022626	004737	003546			ERROR	+51	:CSR AND-OR WCR AND-OR BAR ARE INCORRECT
3568 022632	104051				MOV	@BDR,RBDR	:MOVE RECEIVED DATA TO RBDR
3569 022634	017737	157664	002564		CMP	#177,RBDR	:CHECK THAT WORD #200 OF INBUF IS IN BDR
3570 022642	022737	000177	002564		BEQ	TST34	:BRANCH TO NEXT TEST IF IT IS
3571 022650	001404				MOV	#177,EBDR	:MOVE EXPECTED DATA TO EBDR
3572 022652	012737	000177	002576		ERROR	+45	:BAD DATA IN BDR
3573 022660	104045						

3580

.SBTTL TEST #34 - TEST 7 SINGLE DATO NON BURST MODE TRANSFERS

*TEST 34 TEST 7 SINGLE DATO NON BURST MODE TRANSFERS

*

* THIS TEST DOES 7 PATTERNS OF SINGLE DATO NON BURST MODE TRANSFERS, AND
 * THAT THEY ARE DONE PROPERLY.

*

022662	032777	004000	156450	TST34:	BIT	#BIT11,@SWR	:00 TEST TO SEE IF TEST NUMBER TRACER ENABLED
022670	001407				BEQ	1001\$	:00 BRANCH IF NOT
022672	104401	022700			TYPE	,1000\$	:00 TYPE: 'T # 34'
022676	000404				BR	1001\$	:00 GET OVER ASCII
022700	124	040	043	1000\$:	.ASCIIZ	/T # 34/<CRLF>	:00 THE ASCII MESSAGE
					.EVEN		
022710				1001\$:			
022710	000004				SCOPE		:PROCESS LOOPING AND TEST NUMBER INCREMENT
022712	012737	022734	001310		MOV	#999\$,\$LPERR	:SET LOOP ON ERROR TO 999\$
3581	022720	012702	000007	1\$:	MOV	#7,R2	:DO 7 BIT PATTERNS
3582	022724	005037	002714		CLR	ERRCNT	:CLEAR THE ERRCNT LOCATION FOR USE OF ERROR +201
3583	022730	012703	006250		MOV	#PATRNS,R3	:MOVE ADDRESS OF PATTERNS TO R3
3584	022734	017737	157566	002532	999\$:	MOV	@DRINV,\$DRINV
3585	022742	017737	157562	002534		MOV	@DRVS,\$DRVS
3586	022750	012777	023122	157550		MOV	#3\$,@DRINV
3587	022756	013777	002540	157544		MOV	LEVEL,@DRVS
3588	022764	005037	177776		CLR	PSW	:LET DR11 INTERRUPT
3589	022770	012777	010000	157524		MOV	#MA,@CSR
3590	022776	005077	157520		CLR	@CSR	:DO AN INIT
3591	023002	012777	177777	157506		MOV	#-1,@WCR
3592	023010	012777	002612	157502		MOV	#NPR1,@BAR
3593	023016	005037	002612		CLR	NPR1	:TRANSFER TO BUS ADDRESS IN NPR1
3594	023022	011377	157476		MOV	(R3),@BDR	:GET READY TO RECEIVE DATA
3595	023026	012777	000112	157466		MOV	#F1+F3+IE,@CSR
3596	023034	052777	000401	157460		BIS	#CY+GO,@CSR
3597	023042	012737	001000	002660		MOV	#1000,TIME
3598	023050	005337	002660		2\$:	DEC	TIME
3599	023054	001375				BNE	2\$
3600	023056	013777	002532	157442		MOV	\$DRINV,@DRINV
3601	023064	013777	002534	157436		MOV	\$DRVS,@DRVS
3602	023072	017737	157424	002560		MOV	@CSR,RCSR
3603	023100	011337	001360			MOV	(R3),\$TMP0
3604	023104	104035				ERROR	+35
3605	023106	012777	010000	157406		MOV	#MA,@CSR
3606	023114	005077	157402			CLR	@CSR
3607	023120	000445				BR	6\$
3608	023122	062706	000004		3\$:	ADD	#4,SP
3609	023126	013777	002532	157372		MOV	\$DRINV,@DRINV
3610	023134	013777	002534	157366		MOV	\$DRVS,@DRVS
3611	023142	004737	004254			JSR	PC,ERRCHK
3612	023146	104021				ERROR	+21
3613	023150	017737	157342	002570		MOV	@WCR,RWCR
3614	023156	001403				BEQ	4\$
3615	023160	011337	001362			MOV	(R3),\$TMP1
3616	023164	104206				ERROR	+206
3617	023166	017737	157326	002566	4\$:	MOV	@BAR,RBAR
3618	023174	022737	002615	002566		CMP	#NPR1+3,RBAR
3619							
3620	023202	001406				BEQ	5\$

:MOVE RECEIVED ADDRESS TO RBAR
 :COMPARE CORRECT BAR WITH BAR CABLE MODE TESTING LEAVES
 :BIT 0 OF BAR SET. THEREFORE MUST CHECK FOR ODD ADDRESS
 :BRANCH IF IT IS OK

3621	023204	012737	002615	002600	MOV	#NPR1+3,EBAR	;MOVE EXPECTED DATA TO EBAR
3622	023212	011337	001362		MOV	(R3), \$TMP1	;MOVE PATTERN TO \$TMP0
3623	023216	104207			ERROR	+207	;BAR IS WRONG
3624	023220	021337	002612	5\$:	CMP	(R3), NPR1	;CHECK FOR CORRECT DATA
3625	023224	001403			BEQ	6\$	;BRANCH IF CORRECT DATA WAS TRANSFERRED
3626	023226	011337	002604		MOV	(R3), ENPR1	;MOVE EXPECTED DATA TO ENPR1
3627	023232	104210			ERROR	+210	;DATA NOT TRANSFERED CORRECTLY
3628	023234	062703	000002	6\$:	ADD	#2, R3	;POINT TO NEXT BIT PATTERN
3629	023240	005302			DEC	R2	;COUNT 1 PATTERN DONE
3630	023242	001234			BNE	999\$	;BRANCH BACK IF NOT DONE

3636

.SBTTL TEST #35 - TEST STRING OR 200 DATOS BURST MODE XFERS

*TEST 35 TEST STRING OR 200 DATOS BURST MODE XFERS

*

THIS TEST CHECKS 200 DATO TRANSFERS IN BURST MODE.

*

023244	032777	004000	156066	TST35:	BIT	#BIT11,@SWR	;## TEST TO SEE IF TEST NUMBER TRACER ENABLED
023252	001407				BEQ	1001\$	;## BRANCH IF NOT
023254	104401	023262			TYPE	,1000\$	;## TYPE: 'T # 35'
023260	000404				BR	1001\$	;## GET OVER ASCII
023262	124	040	043	1000\$:	.ASCIIZ	/T # 35/<CRLF>	;## THE ASCII MESSAGE
					.EVEN		

1001\$:

SCOPE ;PROCESS LOOPING AND TEST NUMBER INCREMENT

MOV #999\$,\$LPERR ;SET LOOP ON ERROR TO 999\$

999\$:

CLR @CSR ;FORCE ACCESS TO CSR

MOV #200,BUFLEN ;LENGTH OF BUFFER=200

JSR PC,LODBUF ;LOAD THE BUFFER WITH INCREMENTING PATTERN

MOV BUFLEN,WCLLEN ;PREPARE NUMBER FOR WCR

NEG WCLLEN ;2'S COMPLEMENT OF BUFLEN

MOV WCLLEN,@WCR ;SET UP WCR

MOV INBUF,@BAR ;SET UP BAR

MOV #52525,@BDR ;SET UP BDR

MOV @DRINV,SDRINV ;SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR

MOV @DRVS,SDRVS ;SAVE LOCATION TO BE USED AS THE INTERRUPT PS

MOV #2\$,@DRINV ;INTERRUPT VECTOR

MOV LEVEL,@DRVS ;INTERRUPT STATUS TO LEVEL OF DEVICE

CLR PSW ;LET THE DR11 INTERRUPT

MOV #IE+F1,@CSR ;IE, FNCT1

BIS #CY+GO,@CSR ;CYCLE, GO

MOV #1000,TIME ;MOVE WAIT LOOP VALUE TO TIME LOCATION

1\$:

DEC TIME ;DECREMENT UNTIL WE REACH ZERO

BNE 1\$;BRANCH BACK IF NOT ZERO

MOV SDRINV,@DRINV ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR

MOV SDRVS,@DRVS ;RESTORE LOCATION USED AS THE INTERRUPT PS

MOV @CSR,RCSR ;MOVE RECEIVED DATA TO RCSR

ERROR +35 ;DR11 FAILED TO INTERRUPT

MOV #MA,@CSR ;SET THE MAINTENANCE BIT AND

CLR @CSR ;CLEAR THE CSR TO DO AN INIT

BR TST36 ;BRANCH TO NEXT TEST

2\$:

ADD #4,SP ;CLEAN STACK AFTER INTERRUPT

MOV SDRINV,@DRINV ;RESTORE LOCATION USED AS THE INTERRUPT VECTOR

MOV SDRVS,@DRVS ;RESTORE LOCATION USED AS THE INTERRUPT PS

JSR PC,INTA ;GO CLEAR IE, CHECK ERROR, READY, WCR=0 AND BAR

ERROR +51 ;CSR AND-OR WCR AND-OR BAR ARE INCORRECT

JSR PC,DATOCK ;CHECK INBUF

ERROR +46 ;BUFFER DATA NOT CORRECT

JSR PC,DATOC2 ;GO BACK TO SUBROUTINE AFTER ERROR RTS IN DATOCK

ERROR +47 ;TOO MANY WORDS WERE TRANSFERED

	023272			
	023272	000004		
	023274	012737	023302	001310
3637	023302	005077	157214	
3638	023306	012737	000200	002622
3639	023314	004737	003472	
3640	023320	013737	002622	002630
3641	023326	005437	002630	
3642	023332	013777	002630	157156
3643	023340	013777	002616	157152
3644	023346	012777	052525	157150
3645	023354	017737	157146	002532
3646	023362	017737	157142	002534
3647	023370	012777	023500	157130
3648	023376	013777	002540	157124
3649	023404	005037	177776	
3650	023410	012777	000102	157104
3651	023416	052777	000401	157076
3652	023424	012737	001000	002660
3653	023432	005337	002660	
3654	023436	001375		
3655	023440	013777	002532	157060
3656	023446	013777	002534	157054
3657	023454	017737	157042	002560
3658	023462	104035		
3659	023464	012777	010000	157030
3660	023472	005077	157024	
3661	023476	000421		
3662	023500	062706	000004	
3663	023504	013777	002532	157014
3664	023512	013777	002534	157010
3665	023520	004737	003546	
3666	023524	104051		
3667	023526	004737	004106	
3668	023532	104046		
3669	023534	004737	004170	
3670	023540	104047		

3676

.SBTTL TEST #36 - TEST STRING OF 200 DATIS NON-BURST MODE

*TEST 36 TEST STRING OF 200 DATIS NON-BURST MODE

*

THIS TEST DOES 200 DATI TRANSFERS IN NON BURST MODE.

*

023542	032777	004000	155570	TST36:	BIT	#BIT11, @SWR	:88 TEST TO SEE IF TEST NUMBER TRACER ENABLED
023550	001407				BEQ	1001\$	:88 BRANCH IF NOT
023552	104401	023560			TYPE	,1000\$	:88 TYPE: 'T # 36'
023556	000404				BR	1001\$	:88 GET OVER ASCII
023560	124	040	043	1000\$:	.ASCIIZ	/T # 36/<CRLF>	:88 THE ASCII MESSAGE
					.EVEN		

023570				1001\$:			
023570	000004				SCOPE		:PROCESS LOOPING AND TEST NUMBER INCREMENT
023572	012737	023600	001310		MOV	#999\$, \$LPERR	:SET LOOP ON ERROR TO 999\$

3677	023600	004737	004036	999\$:	JSR	PC, CLEUP	:SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
3678	023604	005077	156712		CLR	@CSR	:FORCE ACCESS TO CSR
3679	023610	012737	000200		MOV	#200, BUFLN	:LENGTH OF BUFFER=200
3680	023616	004737	003472		JSR	PC, LODBUF	:LOAD THE BUFFER WITH INCREMENTING PATTERN
3681	023622	013737	002622		MOV	BUFLN, WCLN	:PREPARE NUMBER FOR WCR
3682	023630	005437	002630		NEG	WCLN	:2'S COMPLEMENT OF BUFLN
3683	023634	013777	002630		MOV	WCLN, @WCR	:SET-UP WCR
3684	023642	013777	002616		MOV	INBUF, @BAR	:SET-UP BAR
3685	023650	012777	177777		MOV	#-1, @BDR	:MAINT AIDE
3686	023656	017737	156644		MOV	@DRINV, SDRINV	:SAVE LOCATION TO BE USED AS THE INTERRUPT VECTOR
3687	023664	017737	156640		MOV	@DRVS, SDRVS	:SAVE LOCATION TO BE USED AS THE INTERRUPT PS
3688	023672	012777	024002		MOV	#2\$, @DRINV	:INT VECTOR
3689	023700	013777	002540		MOV	LEVEL, @DRVS	:INTERRUPT STATUS TO LEVEL OF DEVICE
3690	023706	005037	177776		CLR	PSW	:LET THE DR11 INTERRUPT
3691	023712	012777	000110		MOV	#F3+IE, @CSR	:FNCT3, IE
3692	023720	052777	000401		BIS	#CY+GO, @CSR	:CYCLE, GO
3693	023726	012737	001000		MOV	#1000, TIME	:SET WAIT LOOP COUNTER
3694	023734	005337	002660	1\$:	DEC	TIME	:DECREMENT UNTIL WE REACH ZERO
3695	023740	001375			BNE	1\$	:BRANCH BACK IF NOT ZERO
3696	023742	013777	002532		MOV	SDRINV, @DRINV	:RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3697	023750	013777	002534		MOV	SDRVS, @DRVS	:RESTORE LOCATION USED AS THE INTERRUPT PS
3698	023756	017737	156540		MOV	@CSR, RCSR	:MOVE RECEIVED DATA TO RCSR
3699	023764	104035			ERROR	+35	:DR11 FAILED TO INTERRUPT
3700	023766	012777	010000		MOV	#MA, @CSR	:SET THE MAINTENANCE BIT AND
3701	023774	005077	156522		CLR	@CSR	:CLEAR THE CSR TO DO AN INIT
3702	024000	000426			BR	TST37	:BRANCH TO NEXT TEST
3703	024002	062706	000004	2\$:	ADD	#4, SP	:CLEAN UP STACK AFTER INTERRUPT
3704	024006	013777	002532		MOV	SDRINV, @DRINV	:RESTORE LOCATION USED AS THE INTERRUPT VECTOR
3705	024014	013777	002534		MOV	SDRVS, @DRVS	:RESTORE LOCATION USED AS THE INTERRUPT PS
3706	024022	004737	003546		JSR	PC, INTA	:GO CLEAR IE, CHECK ERROR, READY, WCR=0 AND BAR
3707	024026	104051			ERROR	+51	:CSR AND-OR WCR AND-OR BAR ARE INCORRECT
3708	024030	017737	156470		MOV	@BDR, RBDR	:MOVE RECEIVED DATA TO RBDR
3709	024036	022737	000177		CMP	#177, RBDR	:CHECK THAT WORD #200 OF INBUF IS IN BDR
3710	024044	001404			BEQ	TST37	:BRANCH TO NEXT TEST IF OK
3711	024046	012737	000177		MOV	#177, EBDR	:MOVE EXPECTED DATA TO EBDR
3712	024054	104045			ERROR	+45	:BAD DATA IN BDR

3718

.SBTTL TEST #37 - TEST STRING OF 200 DATOS NON-BURST MODE

*TEST 37 TEST STRING OF 200 DATOS NON-BURST MODE

*

THIS TEST DOES 200 DATOS IN NON BURST MODE.

*

024056	032777	004000	155254	TST37:	BIT	#BIT11,@SWR	:88 TEST TO SEE IF TEST NUMBER TRACER ENABLED
024064	001407				BEQ	1001\$	:88 BRANCH IF NOT
024066	104401	024074			TYPE	,1000\$	:88 TYPE: 'T # 37'
024072	000404				BR	1001\$	:88 GET OVER ASCII
024074	124	040	043	1000\$:	.ASCIIZ	/T # 37/<CRLF>	:88 THE ASCII MESSAGE
					.EVEN		

1001\$:

	024104				SCOPE		:PROCESS LOOPING AND TEST NUMBER INCREMENT
	024104	000004			MOV	#999\$,\$LPERR	:SET LOOP ON ERROR TO 999\$
3719	024106	012737	024114	001310	999\$:	JSR	PC,CLNUP
3720	024114	004737	004036			CLR	@CSR
3721	024120	005077	156376				:SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
3722	024124	012737	000200	002622		MOV	#200,BUFLEN
3723	024132	004737	003472				:FORCE ACCESS TO CSR
3724	024136	013737	002622	002630		JSR	PC,LODBUF
3725	024144	005437	002630			MOV	BUFLEN,WCLN
3726	024150	013777	002630	156340		NEG	WCLN
3727	024156	013777	002616	156334		MOV	WCLN,@WCR
3728	024164	012777	052525	156332		MOV	INBUF,@BAR
3729	024172	017737	156330	002532		MOV	#52525,@BDR
3730	024200	017737	156324	002534		MOV	@DRINV,SDRINV
3731	024206	012777	024316	156312		MOV	@DRVS,SDRVS
3732	024214	013777	002540	156306		MOV	#2\$,@DRINV
3733	024222	005037	177776			MOV	LEVEL,@DRVS
3734	024226	012777	000112	156266		CLR	PSW
3735	024234	052777	000401	156260		MOV	#F1+F3+IE,@CSR
3736	024242	012737	001000	002660		BIS	#CY+GO,@CSR
3737	024250	005337	002660		1\$:	MOV	#1000,TIME
3738	024254	001375				DEC	TIME
3739	024256	013777	002532	156242		BNE	1\$
3740	024264	013777	002534	156236		MOV	SDRINV,@DRINV
3741	024272	017737	156224	002560		MOV	SDRVS,@DRVS
3742	024300	104035				MOV	@CSR,RCSR
3743	024302	012777	010000	156212		ERROR	+35
3744	024310	005077	156206			MOV	#MA,@CSR
3745	024314	000421				CLR	@CSR
3746	024316	062706	000004		2\$:	BR	TST40
3747	024322	013777	002532	156176		ADD	#4,SP
3748	024330	013777	002534	156172		MOV	SDRINV,@DRINV
3749	024336	004737	003546			MOV	SDRVS,@DRVS
3750	024342	104051				JSR	PC,INTA
3751	024344	004737	004106			ERROR	+51
3752	024350	104046				JSR	PC,DATOCK
3753	024352	004737	004170			ERROR	+46
	024356	104047				JSR	PC,DATOC2
						ERROR	+47

:GO BACK TO SUBROUTINE AFTER ERROR RTS IN DATOCK
 :TOO MANY WORDS WERE TRANSFERED

```
3754 .SBTTL END OF DEVICE PASS ROUTINE
3755 *****
3756 024360 000004 ENDEV: SCOPE ;FOR POSSIBLE LOOP ON TEST
3757 024362 005037 001302 CLR $TSTNM ;CLEAR TEST NO. COUNT FOR SCOPE ROUTINE
3758 024366 022737 000001 002414 CMP #1,QTYBRD ;IS THERE MORE THAN 1 BOARD UNDER TEST?
3759 024374 001444 BEQ $$ ;BRANCH IF NO
3760 024376 005737 001312 TST $ERTTL ;SEE IF THERE WERE ANY ERRORS
3761 024402 001014 BNE 2$ ;BRANCH AROUND EOP TEST IF SO
3762 024404 105737 002706 TSTB EOPLOC ;SEE IF EOP MESSAGES ARE TO PRINT
3763 024410 001413 BEQ 3$ ;BRANCH IF SO
3764 024412 105737 002707 TSTB EOPLOC+1 ;SEE IF EOP WAS REQUESTED BEFORE
3765 024416 001003 BNE 1$ ;BRANCH TO PRINT MESSAGE IF SO
3766 024420 105737 030707 TSTB CHARCT ;SEE IF EOD REQUESTED
3767 024424 001430 BEQ $$ ;BRANCH AROUND MESSAGE PRINT IF NOT
3768 024426 105237 002707 1$: INCB EOPLOC+1 ;INCREMENT UPPER BYTE OF EOPLOC TO CALL EOP MESSAGE
3769 024432 000402 BR 3$ ;BRANCH IF NOT - NO ERRORS TO SPACE FROM
3770 024434 104401 001405 2$: TYPE , $CRLF ;TYPE A <CRLF>
3771 024440 104401 035322 3$: TYPE , BOARD ;TYPE: 'BOARD # '
3772 024444 013746 001422 MOV $UNIT,-(SP) ;SAVE $UNIT FOR TYPEOUT
3773 024450 104405 TYPDS ;GO TYPE--DECIMAL ASCII WITH SIGN
3774 024452 104401 036237 TYPE ,TSTCOM ;TYPE: ' TESTING COMPLETE'
3775 024456 005737 001312 TST $ERT'L ;SEE IF ANY ERRORS THIS DEVICE
3776 024462 001407 BEQ 4$ ;BRANCH AROUND TOTAL ERRORS MESSAGE IF NONE
3777 024464 104401 034250 TYPE ,ETDEV ;TYPE: ' - TOTAL ERRORS THIS DEVICE = '
3778 024470 013746 001312 MOV $ERTTL,-(SP) ;MOVE NUMBER OF ERRORS TO THE STACK
3779 024474 104405 TYPDS ;GO TYPE THE NUMBER
3780 024476 004737 003126 JSR PC,ERCAPT ;GO LOG THE UNIT #, PASS # & # OF ERRORS THIS DEVICE
3781 024502 104401 001405 4$: TYPE , $CRLF ;TYPE A <CRLF>
3782 024506 005237 001420 5$: INC $DEVCT ;INCREMENT DEVICE COUNTER
3783 024512 023737 002414 001420 CMP QTYBRD,$DEVCT ;ALL DEVICES TESTED?
3784 024520 001404 BEQ $EOP ;GO TO END OF PASS ROUTINE IF SO
3785 024522 005237 001422 INC $UNIT ;INCREMENT THE UNIT NUMBER
3785 024526 000137 011020 JMP TSTDEV ;GO TEST NEXT DEVICE
```


3786

```
.SBTTL  END OF PASS ROUTINE
*****
;*INCREMENT THE PASS NUMBER ($PASS)
;*TYPE 'END PASS #XXXXX' (WHERE XXXXX IS A DECIMAL NUMBER)
;*IF THERES A MONITOR GO TO IT
;*IF THERE ISN'T JUMP TO GOAGIN
$EOP:
SCOPE
CLR      $STNM      ;;ZERO THE TEST NUMBER
INC      $PASS      ;;INCREMENT THE PASS NUMBER
BPL      10$        ;;BRANCH IF STILL POSITIVE
CLR      $PASS      ;;CLEAR THE PASS LOCATION AND
INC      $PASS2     ;;INCREMENT $PASS2 TO SHOW 1 OVERFLOW
10$: DEC  (PC)+      ;;LOOP?
$EOPCT: .WCRD      1
BGT      $DOAGN     ;;YES
MOV      (PC)+,a(PC)+ ;;RESTORE COUNTER
$ENDCT: .WORD      1
TSTB     EOPLOC     ;;SEE IF EOP MESSAGES ARE TO PRINT
BEQ      1$         ;;BRANCH IF THEY ARE
TST      $ERTTL     ;;SEE IF ANY ERRORS THIS PASS
BNE      2$         ;;BRANCH IF THERE ARE TO PRINT EOP
TSTB     EOPLOC+1   ;;SEE IF ERROR OCCURED IN ANOTHER DEVICE
BNE      1$         ;;BRANCH IF ANY TO PRINT EOP
TSTB     CHARCT     ;;SEE IF ANY CHARACTER WAS INPUTED
BEQ      $GET42     ;;BRANCH IF NOT
CMPB     #7,CHARCT  ;;SEE IF OTHER THAT (^G) TYPED, REQUESTING EOP
BEQ      $GET42     ;;BRANCH IF A (^G) - THIS NOT MEANT FOR EOP ROUTINE
CLRB     CHARCT     ;;CLEAR THE LOCATION
BR       3$         ;;GET OVER $ERTTL TEST AND <CRLF> PRINT
1$: TST  $ERTTL     ;;SEE IF AN EXTRA <CRLF> NEEDS PRINTING
BEQ      3$         ;;BRANCH IF NOT
2$: TYPE , $CRLF    ;;TYPE A <CRLF> TO SPACE EOP FROM ERROR
3$: CLRB  EOPLOC+1  ;;CLEAR THE UPPER BYTE OF EOPLOC
    TYPE 'END PASS #'
    MOV  $PASS2,-(SP) ;;MOVE OVERFLOW TO STACK FOR PRINTING
    MOV  $PASS,-(SP) ;;MOVE $PASS TO THE STACK FOR PRINTING
    TYPE THE NUMBER IN EXTENDED DECIMAL
    TYPE A NULL CHARACTER
    TST  $ERTTL     ;;SEE IF ANY ERRORS THIS PASS
    BEQ  4$         ;;BRANCH AROUND NUMBER OF ERRORS MESSAGE IF NONE
    CMP  #1, QTYBRD ;;SEE IF ONLY 1 BOARD IS BEING TESTED
    BNE  4$         ;;BRANCH IF NOT - 'TOTAL ERRORS' IS MEANINGLESS
    TYPE ,TESLR     ;;TYPE: ' TOTAL ERRORS SINCE LAST REPORT '
    MOV  $ERTTL,-(SP) ;;PUT $ERTTL ON STACK FOR PRINTING
    TYPE THE NUMBER OF ERRORS IN DECIMAL
    JSR  PC,ERCAPT  ;;GO CAPTURE THE DEVICE, PASS & # OF ERRORS
    4$: TYPE , $CRLF ;;TYPE A <CRLF>
    $GET42: MOV a#42,R0 ;;GET MONITOR ADDRESS
    BEQ  $DOAGN     ;;BRANCH IF NO MONITOR
    RESET
    $ENDAD: JSR PC,(R0) ;;CLEAR THE WORLD
    NOP      ;;GO TO MONITOR
    NOP      ;;SAVE ROOM
    NOP      ;;FOR
    $DOAGN:  ;;ACT11
    NOP
```

024760	000137			JMP	@(PC)+	::RETURN
024762	025002			\$RTNAD:	.WORD	GOAGIN
024764	377	377	000	\$ENULL:	.BYTE	-1,-1,0
024767	105	116	104	\$ENDMG:	.ASCIZ	/END PASS #/
3787	025002	005037	001420	GOAGIN:	CLR	\$DEVCT
3788	025006	022737	000001		CMP	#1,QTYBRD
3789	025014	001002			BNE	RSTRT
3790	025016	000137	011206		JMP	REINIT
3791	025022	005037	001422	RSTRT:	CLR	\$UNIT
3792	025026	000137	011006		JMP	BEGIN1

;CLEAR DEVICE COUNT
 ;IS THERE ONLY ONE DEVICE UNDER TEST?
 ;BR, IF NOT
 ;GO DO ANOTHER PASS
 ;CLEAR UNIT NUMBER
 ;GO BEGIN TEST OF NEXT DEVICE

3793

.SBTTL TYPE ROUTINE

*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
*NOTE1: \$NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
*NOTE2: \$FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
*NOTE3: \$FILLC CONTAINS THE CHARACTER TO FILL AFTER.
*

*CALL:

*1) USING A TRAP INSTRUCTION

* TYPE ,MESADR ;:MESADR IS FIRST ADDRESS OF AN ASCIZ STRING

*OR

* TYPE
* MESADR

025032	105737	001357	\$TYPE:	TSTB	\$TPFLG	::IS THERE A TERMINAL?
025036	100002			BPL	1\$	::BR IF YES
025040	000000			HALT		::HALT HERE IF NO TERMINAL
025042	000430			BR	3\$	::LEAVE
025044	010046		1\$:	MOV	RO,-(SP)	::SAVE RO
025046	017600	000002		MOV	@2(SP),RO	::GET ADDRESS OF ASCIZ STRING
025052	122737	000001	001430	CMPB	#APTENV,\$ENV	::RUNNING IN APT MODE
025060	001011			BNE	62\$	::NO,GO CHECK FOR APT CONSOLE
025062	132737	000100	001431	BITB	#APTSPool,\$ENV	::SPOOL MESSAGE TO APT
025070	001405			BEQ	62\$	::NO,GO CHECK FOR CONSOLE
025072	010037	025102		MOV	RO,61\$	::SETUP MESSAGE ADDRESS FOR APT
025076	004737	031534		JSR	PC,\$ATY3	::SPOOL MESSAGE TO APT
025102	000000		61\$:	.WORD	0	::MESSAGE ADDRESS
025104	132737	000040	001431	62\$:	BITB	#APTCsup,\$ENV
025112	001003			BNE	60\$	::APT CONSOLE SUPPRESSED
025114	112046		2\$:	MOVB	(RO)+,-(SP)	::YES,SKIP TYPE OUT
025116	001005			BNE	4\$	::PUSH CHARACTER TO BE TYPED ONTO STACK
025120	005726			TST	(SP)+	::BR IF IT ISN'T THE TERMINATOR
025122	012600			MOV	(SP)+,RO	::IF TERMINATOR POP IT OFF THE STACK
025124	062716	000002	60\$:	ADD	#2,(SP)	::RESTORE RO
025130	000002		3\$:	RTI		::ADJUST RETURN PC
025132	122716	000011	4\$:	CMPB	#HT,(SP)	::RETURN
025136	001430			BEQ	8\$	::BRANCH IF <HT>
025140	122716	000200		CMPB	#CRLF,(SP)	::BRANCH IF NOT <CRLF>
025144	001006			BNE	5\$	
025146	005726			TST	(SP)+	::POP <CR><LF> EQUIV
025150	104401			TYPE		::TYPE A CR AND LF
025152	001405			\$CRLF		
025154	105037	025374		CLRB	\$CHARCNT	::CLEAR CHARACTER COUNT
025160	000755			BR	2\$	::GET NEXT CHARACTER
025162	004737	025244	5\$:	JSR	PC,\$TYPEC	::GO TYPE THIS CHARACTER
025166	123726	001356	6\$:	CMPB	\$FILLC,(SP)+	::IS IT TIME FOR FILLER CHARS.?
025172	001350			BNE	2\$	::IF NO GO GET NEXT CHAR.
025174	013746	001354		MOV	\$NULL,-(SP)	::GET # OF FILLER CHARS. NEEDED
						::AND THE NULL CHAR.
025200	105366	000001	7\$:	DECB	1(SP)	::DOES A NULL NEED TO BE TYPED?
025204	002770			BLT	6\$	::BR IF NO--GO POP THE NULL OFF OF STACK
025206	004737	025244		JSR	PC,\$TYPEC	::GO TYPE A NULL
025212	105337	025374		DECB	\$CHARCNT	::DO NOT COUNT AS A COUNT
025216	000770			BR	7\$	::LOOP
				HORIZONTAL TAB	PROCESSOR	
025220	112716	000040	8\$:	MOVB	#' ,(SP)	::REPLACE TAB WITH SPACE

```
025224 004737 025244 9$: JSR PC,$TYPEC ;;TYPE A SPACE
025230 132737 000007 025374 BITB #7,$CHARCNT ;;BRANCH IF NOT AT
025236 001372 BNE 9$ ;;TAB STOP
025240 005726 TST (SP)+ ;;POP SPACE OFF STACK
025242 000724 BR 2$ ;;GET NEXT CHARACTER
025244 105777 154100 $TYPEC: TSTB @STPS ;;WAIT UNTIL PRINTER IS READY
025250 100375 BPL $TYPEC
025252 116677 000002 154072 MOVB 2(SP),@STPB ;;LOAD CHAR TO BE TYPED INTO DATA REG.
025260 105777 154060 TSTB @STKS ;;SEE IF KEYBOARD IS TALKING.
025264 100027 BPL 2$ ;;BRANCH IF IT ISN'T.
025266 117737 154054 030707 MOVB @STKB,CHARCT ;;PUT CHARACTER IN CHARCT
025274 142737 000200 030707 BICB #200,CHARCT ;;BIT CLEAR PARITY BIT.
025302 122737 000023 030707 CMPB #23,CHARCT ;;SEE IF THIS IS A ^S.
025310 001015 BNE 2$ ;;BRANCH TO CONTINUE IF IT ISN'T.
025312 105777 154026 3$: TSTB @STKS ;;WAIT FOR ANOTHER INPUT.
025316 100375 BPL 3$ ;;BRANCH BACK IF NOT READY.
025320 117737 154022 030707 MOVB @STKB,CHARCT ;;PUT CHARACTER IN CHARCT
025326 142737 000200 030707 BICB #200,CHARCT ;;BIT CLEAR PARITY BIT.
025334 122737 000021 030707 CMPB #21,CHARCT ;;SEE IF THIS IS A ^Q.
025342 001363 BNE 3$ ;;BRANCH BACK FOR MORE WAIT IF NOT.
025344 122766 000015 000002 2$: CMPB #CR,2(SP) ;;IS CHARACTER A CARRIAGE RETURN?
025352 001003 BNE 1$ ;;BRANCH IF NO
025354 105037 025374 CLRB $CHARCNT ;;YES--CLEAR CHARACTER COUNT
025360 000406 BR $TYPEX ;;EXIT
025362 122766 000012 000002 1$: CMPB #LF,2(SP) ;;IS CHARACTER A LINE FEED?
025370 001402 BEQ $TYPEX ;;BRANCH IF YES
025372 105227 INCB (PC)+ ;;COUNT THE CHARACTER
025374 000000 $CHARCNT: .WORD 0 ;;CHARACTER COUNT STORAGE
025376 000207 $TYPEX: RTS PC
```

3794

```
.SBTTL BINARY TO OCTAL (ASCII) AND TYPE
*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
*OCTAL (ASCII) NUMBER AND TYPE IT.
*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
*CALL:
*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*      TYPOS      ;;CALL FOR TYPEOUT
*      .BYTE      N      ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
*      .BYTE      M      ;;M=1 OR 0
*                               ;;1=TYPE LEADING ZEROS
*                               ;;0=SUPPRESS LEADING ZEROS
*
*$STYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
*$TYPOS OR $TYPOC
*CALL:
*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*      TYPON      ;;CALL FOR TYPEOUT
*
*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
*CALL:
*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
*      TYPOC      ;;CALL FOR TYPEOUT
*$TYPOS: MOV      @ (SP),-(SP)      ;;PICKUP THE MODE
        MOVB      1(SP), $OFILL      ;;LOAD ZERO FILL SWITCH
        MOVB      (SP)+, $OMODE+1      ;;NUMBER OF DIGITS TO TYPE
        ADD      #2, (SP)      ;;ADJUST RETURN ADDRESS
        BR      $TYPON
*$TYPOC: MOVB      #1, $OFILL      ;;SET THE ZERO FILL SWITCH
        MOVB      #6, $OMODE+1      ;;SET FOR SIX(6) DIGITS
*$TYPON: MOVB      #5, $OCNT      ;;SET THE ITERATION COUNT
        MOV      R3, -(SP)      ;;SAVE R3
        MOV      R4, -(SP)      ;;SAVE R4
        MOV      R5, -(SP)      ;;SAVE R5
        MOVB      $OMODE+1, R4      ;;GET THE NUMBER OF DIGITS TO TYPE
        NEG      R4
        ADD      #6, R4      ;;SUBTRACT IT FOR MAX. ALLOWED
        MOVB      R4, $OMODE      ;;SAVE IT FOR USE
        MOVB      $OFILL, R4      ;;GET THE ZERO FILL SWITCH
        MOV      12(SP), R5      ;;PICKUP THE INPUT NUMBER
        CLR      R3      ;;CLEAR THE OUTPUT WORD
        ROL      R5      ;;ROTATE MSB INTO 'C'
        BR      3$      ;;GO DO MSB
2$: ROL      R5      ;;FORM THIS DIGIT
    ROL      R5
    ROL      R5
    MOV      R5, R3
3$: ROL      R3      ;;GET LSB OF THIS DIGIT
    DECB      $OMODE      ;;TYPE THIS DIGIT?
    BPL      7$      ;;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
4$: INC      R4      ;;DON'T SUPPRESS ANYMORE 0'S
    BIS      #'0, R3      ;;MAKE THIS DIGIT ASCII
5$: BIS      #' , R3      ;;MAKE ASCII IF NOT ALREADY

025400 017646 000000
025404 116637 000001 025623
025412 112637 025625
025416 062716 000002
025422 000406
025424 112737 000001 025623
025432 112737 000006 025625
025440 112737 000005 025622
025446 010346
025450 010446
025452 010546
025454 113704 025625
025460 005404
025462 062704 000006
025466 110437 025624
025472 113704 025623
025476 016605 000012
025502 005003
025504 006105 1$:
025506 000404
025510 006105 2$:
025512 006105
025514 006105
025516 010503
025520 006103 3$:
025522 105337 025624
025526 100016
025530 042703 177770
025534 001002
025536 005704
025540 001403
025542 005204 4$:
025544 052703 000060
025550 052703 000040 5$:
```

025554	110337	025620		MOVB	R3,8\$	::SAVE FOR TYPING
025560	104401	025620		TYPE	8\$	::GO TYPE THIS DIGIT
025564	105337	025622	7\$:	DECB	\$OCNT	::COUNT BY 1
025570	003347			BGT	2\$	::BR IF MORE TO DO
025572	002402			BLT	6\$	::BR IF DONE
025574	005204			INC	R4	::INSURE LAST DIGIT ISN'T A BLANK
025576	000744			BR	2\$	::GO DO THE LAST DIGIT
025600	012605		6\$:	MOV	(SP)+,R5	::RESTORE R5
025602	012604			MOV	(SP)+,R4	::RESTORE R4
025604	012603			MOV	(SP)+,R3	::RESTORE R3
025606	016666	000002 000004		MOV	2(SP),4(SP)	::SET THE STACK FOR RETURNING
025614	012616			MOV	(SP)+,(SP)	
025616	000002			RTI		::RETURN
025620	000		8\$:	.BYTE	0	::STORAGE FOR ASCII DIGIT
025621	000			.BYTE	0	::TERMINATOR FOR TYPE ROUTINE
025622	000		\$OCNT:	.BYTE	0	::OCTAL DIGIT COUNTER
025623	000		\$OFILL:	.BYTE	0	::ZERO FILL SWITCH
025624	000000		\$OMODE:	.WORD	0	::NUMBER OF DIGITS TO TYPE

3795

```

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
*REPLACED WITH SPACES.
*CALL:
*      MOV      NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
*      TYPDS      ;;GO TO THE ROUTINE
*
;; IF YOU SHOULD HAVE A NUMBER GREATER THAN 32767 TO PRINT, LOAD THE
;; MULTIPLE OF 32768 ON THE STACK, THEN THE REMAINDER AS YOU WOULD
;; ABOVE. FOR INSTANCE, WHEN INCREMENTING A COUNTER TO BE PRINTED,
;; TEST FOR NEGATIVE. IF NEGATIVE, CLEAR THE LOCATION AND INCREMENT
;; A SPECIAL OVERFLOW LOCATION.
;; CALL:
;;      MOV      OVFNUM,-(SP)  ;;PUT OVERFLOW NUMBER ON STACK
;;      MOV      NUM,-(SP)    ;;PUT REMAINING NUMBER ON STACK
;;      TYPDE      ;;GO TO THE ROUTINE
;;      SHOW ENTRY AT $TYPDE
;;      GO TO ROUTINE
;;      SHOW ENTRY AT $TYPDS
025626 012737 000001 001364 $TYPDE: MOV      #1,$TMP2
025634 000402                BR      $TYPD
025636 005037 001364      $TYPDS: CLR      $TMP2
025642                $TYPD:
025642 010046                MOV      R0,-(SP)      ;;PUSH R0 ON STACK
025644 010146                MOV      R1,-(SP)      ;;PUSH R1 ON STACK
025646 010246                MOV      R2,-(SP)      ;;PUSH R2 ON STACK
025650 010346                MOV      R3,-(SP)      ;;PUSH R3 ON STACK
025652 010546                MOV      R5,-(SP)      ;;PUSH R5 ON STACK
025654 012746 020200        MOV      #20200,-(SP)   ;;SET BLANK SWITCH AND SIGN
025660 016605 000020        MOV      20(SP),R5      ;;GET THE INPUT NUMBER
025664 100004                BPL      1$           ;;BR IF INPUT IS POS.
025666 005405                NEG      R5           ;;MAKE THE BINARY NUMBER POS.
025670 112766 000055 000001 1$: MOVB      #'-,1(SP)  ;;MAKE THE ASCII NUMBER NEG.
025676 005000                CLR      R0           ;;ZERO THE CONSTANTS INDEX
025700 012703 026142        MOV      #$DBLK,R3      ;;SETUP THE OUTPUT POINTER
025704 112723 000040        MOVB      #' ,(R3)+    ;;SET THE FIRST CHARACTER TO A BLANK
025710 005002                CLR      R2           ;;CLEAR THE BCD NUMBER
025712 016001 026132        MOV      $DTBL(R0),R1  ;;GET THE CONSTANT
025716 160105                SUB      R1,R5        ;;FORM THIS BCD DIGIT
025720 002402                BLT      4$           ;;BR IF DONE
025722 005202                INC      R2           ;;INCREASE THE BCD DIGIT BY 1
025724 000774                BR      3$
025726 060105                4$: ADD      R1,R5      ;;ADD BACK THE CONSTANT
025730 005702                TST      R2           ;;CHECK IF BCD DIGIT=0
025732 001002                BNE      5$           ;;FALL THROUGH IF 0
025734 105716                TSTB     (SP)         ;;STILL DOING LEADING 0'S?
025736 100407                BMI      7$           ;;BR IF YES
025740 106316                5$: ASLB      (SP)     ;;MSD?
025742 103003                BCC      6$           ;;BR IF NO
025744 116663 000001 177777 6$: MOVB      1(SP),-1(R3)  ;;YES--SET THE SIGN
025752 052702 000060        BIS      #'0,R2      ;;MAKE THE BCD DIGIT ASCII
025756 052702 000040        7$: BIS      #' ,R2    ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
025762 110223                MOVB      R2,(R3)+    ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
025764 005720                TST      (R0)+       ;;JUST INCREMENTING
025766 020027 000010        CMP      R0,#10      ;;CHECK THE TABLE INDEX
025772 002746                BLT      2$           ;;GO DO THE NEXT DIGIT

```

025774	003002			BGT	8\$	::GO TO EXIT
025776	010502			MOV	R5,R2	::GET THE LSD
026000	000764			BR	6\$	::GO CHANGE TO ASCII
026002	105726		8\$:	TSTB	(SP)+	::WAS THE LSD THE FIRST NON-ZERO?
026004	100003			BPL	9\$	::BR IF NO
026006	116663	177777	177776	MOVB	-1(SP),-2(R3)	::YES--SET THE SIGN FOR TYPING
026014	105013		9\$:	CLRB	(R3)	::SET THE TERMINATOR
026016	005737	001364		TST	\$TMP2	::WAS ENTRY AT \$TYPDS?
026022	001405			BEQ	10\$	::BRANCH IF SO
026024	005766	000020		TST	20(SP)	::TEST THE OVERFLOW LOCATION
026030	001402			BEQ	10\$	::BRANCH IF NON-ZERO
026032	004737	026154		JSR	PC,EXPAND	::GO EXPAND THE DECIMAL NUMBER
026036			10\$:			
026036	012605			MOV	(SP)+,R5	::POP STACK INTO R5
026040	012603			MOV	(SP)+,R3	::POP STACK INTO R3
026042	012602			MOV	(SP)+,R2	::POP STACK INTO R2
026044	012601			MOV	(SP)+,R1	::POP STACK INTO R1
026046	012600			MOV	(SP)+,R0	::POP STACK INTO R0
026050	104401	026142		TYPE	\$DBLK	::NOW TYPE THE NUMBER
026054	005737	001364		TST	\$TMP2	::SEE IF ENTRY WAS AT \$TYPDS
026060	001417			BEQ	11\$	::BRANCH IF SO
026062	016666	000002	000006	MOV	2(SP),6(SP)	::ADJUST THE STACK
026070	012666	000002		MOV	(SP)+,2(SP)	::
026074	005726			TST	(SP)+	::
026076	105037	026151		CLRB	\$DBLK+7	::REPLACE ORIGINAL TERMINATOR
026102	112737	000040	026150	MOVB	' ', \$DBLK+6	::REPLACE ORIGINAL SPACE CHARACTER
026110	112737	000040	026142	MOVB	' ', \$DBLK	::REPLACE ORIGINAL SPACE CHARACTER
026116	000002			RTI		::RETURN TO USER
026120	016666	000002	000004	MOV	2(SP),4(SP)	::ADJUST THE STACK
026126	012616			MOV	(SP)+,(SP)	::
026130	000002			RTI		::RETURN TO USER
026132	023420	001750	000144	\$DTBL:	.WORD	10000.,1000.,100.,10.
026142				\$DBLK:	.BLKW	5


```

.SBTTL SUBROUTINE TO EXPAND DECIMAL TO LARGER THAN 32767
*****
EXPAND: MOV #SDBLK+6,R2 ;88 MOVE LOCATION OF LCD+1 TO R2
MOV #SDBLK+12,R3 ;88 MOVE NEW LOCATION OF LCD+2 TO R3
CLRB -(R3) ;88 MAKE SURE TERMINATOR IS THERE
MOVB -(R2),-(R3) ;88 MOVE THE 5 ASCII'S TO THEIR NEW LOCATIONS
MOVB -(R2),-(R3) ;88
MOVB -(R2),-(R3) ;88
MOVB -(R2),-(R3) ;88
MOVB -(R2),-(R3) ;88
MOVB #',-(R3) ;88 MOVE 4 SPACE CHARACTERS TO THE 4 NEW LOCATIONS
MOVB #',-(R3) ;88
MOVB #',-(R3) ;88
MOVB #',-(R3) ;88
MOV $TMP0,-(SP) ;88 SAVE $TMP0
CLR $TMP0 ;88 CLEAR LOCATION TO USE AS ACCUMULATOR
MOV $TMP1,-(SP) ;88 SAVE $TMP1
CLR $TMP1 ;88 CLEAR LOCATION TO USE AS 2ND ACCUMULATOR
MOV #SDBLK+7,R1 ;88 MOVE ADDRESS OF LCD TO R2
MOV #SNUMS+10,R2 ;88 MOVE ADDRESS+10 OF WORD STREAM OF 8*6 TO R2
1$: MOV 26(SP),R0 ;88 MOVE OVERFLOW LOCATION CONTENTS TO R0
MOVB (R1),$TMP0 ;88 MOVE ASCII TO THE TEMPORARY LOCATION
BIS #'0,$TMP0 ;88 MAKE LOCATION AN ASCII IF NOT ALREADY
2$: ADD (R2),$TMP0 ;88 ADD THE NUMBER TO THE ASCII
CMP #'9,$TMP0 ;88 HAVE WE SURPASSED ASCII '9'?
BGE 4$ ;88 BRANCH IF NOT
SUB #10,$TMP0 ;88 SUBTRACT 10 FROM THE ASCII AND
MOV R1,R3 ;88 MOVE PRESENT ASCII ADDRESS TO R3
3$: DEC R3 ;88 DECREMENT TO NEXT SIGNIFICANT ASCII DIGIT
INCB (R3) ;88 ADD THE CARRY TO THE ASCII
BISB #'0,(R3) ;88 SET BITS TO MAKE IT A NUMBER ASCII
CMPB #'9,(R3) ;88 SEE IF CARRY NEEDS TO BE TRANSFERED
BGE 4$ ;88 BRANCH IF NOT
MOVB (R3),$TMP1 ;88 MOVE BYTE TO LOCATION $TMP1
SUB #10,$TMP1 ;88 SUBTRACT 10 DECIMAL AND
MOVB $TMP1,(R3) ;88 MOVE IT BACK
CMP #SDBLK,R3 ;88 SEE IF ALL POSITIONS HAVE BEEN DONE
BNE 3$ ;88 BRANCH BACK IF NOT
4$: DEC R0 ;88 DECREMENT THE OVERFLOW LOCATION R0
BNE 2$ ;88 BRANCH IF NOT ALL ADDED
MOVB $TMP0,(R1) ;88 MOVE NEW NUMBER TO LOCATION
DEC R1 ;88 POINT R1 TO THE NEXT ASCII LOCATION
TST -(R2) ;88 POINT R2 TO NEXT DIGIT TO ADD
CMP #SDBLK+3,R1 ;88 SEE IF ALL DIGITS HAVE BEEN ADDED
BNE 1$ ;88 BRANCH IF NOT DONE
MOV (SP)+,$TMP1 ;88 RESTORE $TMP1
MOV (SP)+,$TMP0 ;88 RESTORE $TMP0
RTS PC ;88 RETURN
026154 012702 026150
026160 012703 026154
026164 105043
026166 114243
026170 114243
026172 114243
026174 114243
026176 114243
026200 112743 000040
026204 112743 000040
026210 112743 000040
026214 112743 000040
026220 013746 001360
026224 005037 001360
026230 013746 001362
026234 005037 001362
026240 012701 026151
026244 012702 026420
026250 016600 000026
026254 111137 001360
026260 052737 000060 001360
026266 061237 001360
026272 022737 000071 001360
026300 002025
026302 162737 000012 001360
026310 010103
026312 005303
026314 105213
026316 152713 000060
026322 122713 000071
026326 002012
026330 111337 001362
026334 162737 000012 001362
026342 113713 001362
026346 022703 026142
026352 001357
026354 005300
026356 001343
026360 113711 001360
026364 005301
026366 005742
026370 022701 026145
026374 001325
026376 012637 001362
026402 012637 001360
026406 000207
026410 000003 000002 000007 $NUMS: .WORD 3,2,7,6,8.

```

3796

```
.SBTTL TTY INPUT ROUTINE
*****
.ENABL LSB
*****
;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
;*WHEN OPERATING IN TTY FLAG MODE.

026422 022737 000176 001340 $CKSWR: CMP    #SWREG,SWR    ;;IS THE SOFT-SWR SELECTED?
026430 001112                BNE     15$           ;;BRANCH IF NO
026432 105737 030707        TSTB   CHARCT        ;;SEE IF CHARACTER WAS INPUTED DURING PRINT
026436 001405                BEQ     1$           ;;BRANCH IF NOT
026440 113746 030707        MOVB   CHARCT,-(SP)    ;;MOVE CHARACTER TO THE STACK
026444 105037 030707        CLRB   CHARCT        ;;CLEAR THE CHARACTER FROM CHARCT
026450 000405                BR     2$           ;;GO CHECK IT OUT
026452 105777 152666        1$: TSTB   @STKS      ;;CHAR THERE?
026456 100077                BPL     15$         ;;IF NO, DON'T WAIT AROUND
026460 117746 152662        MOVB   @STKB,-(SP)    ;;SAVE THE CHAR
026464 042716 177600        2$: BIC     #^C177,(SP) ;;STRIP-OFF THE ASCII
026470 022726 000007        CMP     #7,(SP)+      ;;IS IT A CONTROL G?
026474 001404                BEQ     3$           ;;YES, BRANCH AROUND RETURN TO USER SETUP
026476 116637 177776 030707        MOVB   -2(SP),CHARCT ;;MOVE CHARACTER BACK TO CHARCT
026504 000464                BR     15$         ;;RETURN TO USER
026506 123727 001334 000001 3$: CMPB   $AUTOB,#1  ;;ARE WE RUNNING IN AUTO-MODE?
026514 001460                BEQ     15$         ;;BRANCH IF YES
026516 104401 027202        TYPE    ,SCNTLG      ;;ECHO THE CONTROL-G (^G)
026522 104401 027207        $GTSWR: TYPE    ,SMSWR ;;TYPE CURRENT CONTENTS
026526 013746 000176        MOV     SWREG,-(SP)   ;;SAVE SWREG FOR TYPEOUT
026532 104402                TYPOC              ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
026534 104401 027220        TYPE    ,SMNEW      ;;PROMPT FOR NEW SWR
026540 105037 030707        19$: CLRB   CHARCT    ;;CLEAR ANY CHARACTER THAT WAS INPUTED DURING PRINT
026544 005046                CLR     -(SP)       ;;CLEAR COUNTER
026546 005046                CLR     -(SP)       ;;THE NEW SWR
026550 105777 152570        7$: TSTB   @STKS      ;;CHAR THERE?
026554 100375                BPL     7$          ;;IF NOT TRY AGAIN
026556 117746 152564        MOVB   @STKB,-(SP)   ;;PICK UP CHAR
026562 042716 177600        BIC     #^C177,(SP)  ;;MAKE IT 7-BIT ASCII
026566 021627 000025        9$: CMP     (SP),#25  ;;IS IT A CONTROL-U?
026572 001005                BNE     10$         ;;BRANCH IF NOT
026574 104401 027175        TYPE    ,SCNTLU     ;;YES, ECHO CONTROL-U (^U)
026600 062706 000006        20$: ADD     #6,SP    ;;IGNORE PREVIOUS INPUT
026604 000746                BR     $GTSWR      ;;LET'S TRY IT AGAIN
026606 021627 000015        10$: CMP     (SP),#15 ;;IS IT A <CR>?
026612 001022                BNE     16$         ;;BRANCH IF NO
026614 005766 000004        TST     4(SP)       ;;YES, IS IT THE FIRST CHAR?
026620 001403                BEQ     11$         ;;BRANCH IF YES
026622 016677 000002 152510        MOV     2(SP),@SWR ;;SAVE NEW SWR
026630 062706 000006        11$: ADD     #6,SP    ;;CLEAR UP STACK
026634 104401 001405        14$: TYPE    ,SCRLF   ;;ECHO <CR> AND <LF>
026640 123727 001335 000001        CMPB   $INTAG,#1 ;;RE-ENABLE TTY KBD INTERRUPTS?
026646 001003                BNE     15$         ;;BRANCH IF NOT
026650 012777 000100 152466        MOV     #100,@STKS ;;RE-ENABLE TTY KBD INTERRUPTS
026656 000002                15$: RTI              ;;RETURN
026660 004737 025244        16$: JSR     PC,$TYPEC ;;ECHO CHAR
026664 021627 000060        CMP     (SP),#60    ;;CHAR < 0?
026670 002420                BLT     18$         ;;BRANCH IF YES
026672 021627 000067        CMP     (SP),#67    ;;CHAR > 7?
```


026676	003015		BGT	18\$	::BRANCH IF YES
026700	042726	000060	BIC	#60,(SP)+	::STRIP-OFF ASCII
026704	005766	000002	TST	2(SP)	::IS THIS THE FIRST CHAR
026710	001403		BEQ	17\$	::BRANCH IF YES
026712	006316		ASL	(SP)	::NO, SHIFT PRESENT
026714	006316		ASL	(SP)	::CHAR OVER TO MAKE
026716	006316		ASL	(SP)	::ROOM FOR NEW ONE.
026720	005266	000002	17\$: INC	2(SP)	::KEEP COUNT OF CHAR
026724	056616	177776	BIS	-2(SP),(SP)	::SET IN NEW CHAR
026730	000707		BR	7\$	::GET THE NEXT ONE
026732	104401	001404	18\$: TYPE	\$QUES	::TYPE ?<CR><LF>
026736	000720		BR	20\$	::SIMULATE CONTROL-U
			.DSABL	LSB	
			;*****		

```

.SBTTL ROUTINE TO INPUT A SINGLE CHARACTER FROM TTY
;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
;*CALL:
;*      RDCHR
;*      RETURN HERE
;*      INPUT A SINGLE CHARACTER FROM THE TTY
;*      CHARACTER IS ON THE STACK
;*      WITH PARITY BIT STRIPPED OFF

026740 011646 $RDCHR: MOV      (SP),-(SP)      ;;PUSH DOWN THE ^C
026742 016666      MOV      4(SP),2(SP)      ;;SAVE THE PS
026750 105777 000004 000002 1$: TSTB      @STKS      ;;WAIT FOR
026754 100375      BPL      1$      ;;A CHARACTER
026756 117766 152364 000004      MOVB      @STKB,4(SP)      ;;READ THE TTY
026764 042766 177600 000004      BIC      #^C<177>,4(SP)      ;;GET RID OF JUNK IF ANY
026772 026627 000004 000023      CMP      4(SP),#2$      ;;IS IT A CONTROL-S?
027000 001013      BNE      3$      ;;BRANCH IF NO
027002 105777 152336      2$: TSTB      @STKS      ;;WAIT FOR A CHARACTER
027006 100375      BPL      2$      ;;LOOP UNTIL ITS THERE
027010 117746 152332      MOVB      @STKB,-(SP)      ;;GET CHARACTER
027014 042716 177600      BIC      #^C<177>,(SP)      ;;MAKE IT 7-BIT ASCII
027020 022627 000021      CMP      (SP)+,#21      ;;IS IT A CONTROL-Q?
027024 001366      BNE      2$      ;;IF NOT DISCARD IT
027026 000750      BR      1$      ;;YES, RESUME
027030 026627 000004 000140 3$: CMP      4(SP),#140      ;;IS IT UPPER CASE?
027036 002407      BLT      4$      ;;BRANCH IF YES
027040 026627 000004 000175      CMP      4(SP),#175      ;;IS IT A SPECIAL CHAR?
027046 003003      BGT      4$      ;;BRANCH IF YES
027050 042766 000040 000004      BIC      #40,4(SP)      ;;MAKE IT UPPER CASE
027056 000002      4$: RTI      ;;GO BACK TO USER

```

```

                                .SBTTL ROUTINE TO INPUT A STRING FROM TTY
                                *****
                                *THIS ROUTINE WILL INPUT A STRING FROM THE TTY
                                *CALL:
                                *
                                *      RDLIN          ;; INPUT A STRING FROM THE TTY
                                *      RETURN HERE      ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
                                *                      ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
                                *
                                $RDLIN: MOV      R3, -(SP)      ;; SAVE R3
                                1$:  MOV      #TTYIN, R3      ;; GET ADDRESS
                                2$:  CMP      #TTYIN+7., R3    ;; BUFFER FULL?
                                BLOS      4$                ;; BR IF YES
                                RDCHR      (SP)+, (R3)        ;; GO READ ONE CHARACTER FROM THE TTY
                                MOV      (R3), 9$            ;; GET CHARACTER
                                10$: CMP      #177, (R3)      ;; IS IT A RUBOUT
                                BNE      3$                  ;; SKIP IF NOT
                                4$:  TYPE      , $QUES        ;; TYPE A '?'
                                BR        1$                ;; CLEAR THE BUFFER AND LOOP
                                3$:  MOV      (R3), 9$        ;; ECHO THE CHARACTER
                                TYPE      , 9$
                                CMP      #15, (R3)+          ;; CHECK FOR RETURN
                                BNE      2$                  ;; LOOP IF NOT RETURN
                                CLRB      -1(R3)             ;; CLEAR RETURN (THE 15)
                                TYPE      , $LF              ;; TYPE A LINE FEED
                                MOV      (SP)+, R3          ;; RESTORE R3
                                MOV      (SP), -(SP)         ;; ADJUST THE STACK AND PUT ADDRESS OF THE
                                MOV      4(SP), 2(SP)        ;; FIRST ASCII CHARACTER ON IT
                                MOV      #TTYIN, 4(SP)
                                RTI
                                9$:  .BYTE      0            ;; RETURN
                                .BYTE      0            ;; STORAGE FOR ASCII CHAR. TO TYPE
                                .BYTE      0            ;; TERMINATOR
                                $TTYIN: .BLKB      7.        ;; RESERVE 7. BYTES FOR TTY INPUT
                                $CNTLU: .ASCIZ  /^U/<15><12>  ;; CONTROL 'U'
                                $CNTLG: .ASCIZ  /^G/<15><12>  ;; CONTROL 'G'
                                $MSWR:  .ASCIZ  <15><12>/SWR = /
                                $MNEW:  .ASCIZ  / NEW = /
                                .EVEN
  027060 010346
  027062 012703 027166
  027066 022703 027175
  027072 101405
  027074 104411
  027076 112613
  027100 122713 000177
  027104 001003
  027106 104401 001404
  027112 000763
  027114 111337 027164
  027120 104401 027164
  027124 122723 000015
  027130 001356
  027132 105063 177777
  027136 104401 001406
  027142 012603
  027144 011646
  027146 016666 000004 000002
  027154 012766 027166 000004
  027162 000002
  027164 000
  027165 000
  027166
  027175 136 125 015
  027202 136 107 015
  027207 015 012 123
  027220 040 040 116

```

3797

```

.SBTTL READ A DECIMAL NUMBER FROM THE TTY
*****
*THIS ROUTINE WILL READ A DECIMAL (ASCII) NUMBER FROM THE TTY AND
*CHANGE IT TO BINARY. IF TOO MANY CHARACTERS OR ANY ILLEGAL CHARACTERS
*ARE READ A '?' FOLLOWED BY A CARRIAGE RETURN-LINE FEED WILL BE TYPED.
*THE COMPLETE NUMBER MUST BE RETYPED. THE INPUT IS TERMINATED BY THE
*USER TYPING A CARRIAGE RETURN. THE RANGE OF THE INPUT NUMBER IS
*POSITIVE 32767 TO NEGATIVE 32768.
*CALL:
*
RDDEC
RETURN HERE
;;READ A DECIMAL NUMBER
;;NUMBER IS ON TOP OF THE STACK

027232 011646 000004 000002 $RDDEC: MOV (SP),-(SP) ;;PROVIDE SPACE FOR
027234 016666 000004 000002 MOV 4(SP),2(SP) ;;THE INPUT NUMBER
027242 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
027244 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
027246 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK
027250 104412 1$: RDLIN ;;READ AN ASCII LINE
027252 012600 MOV (SP)+,R0 ;;ADDRESS OF 1ST CHAR.
027254 010037 027400 MOV R0,6$ ;;SAVE INCASE OF BAD INPUT
027260 005046 CLR -(SP) ;;CLEAR DATA WORD
027262 005002 CLR R2 ;;SIGN SET POSITIVE
027264 122710 000055 CMPEB #'-(R0) ;;SEE IF A MINUS SIGN WAS TYPED
027270 001001 BNE 2$ ;;BR IF NO MINUS SIGN
027272 112002 MOVB (R0)+,R2 ;;SAVE FOR LATER USE
027274 112001 2$: MOVB (R0)+,R1 ;;PICKUP THIS CHARACTER
027276 001424 BEQ 3$ ;;GET OUT IF ZERO
027300 122701 000060 CMPEB #'0,R1 ;;MAKE SURE THIS CHARACTER
027304 003032 000071 BGT 5$ ;;IS A DIGIT BETWEEN 0 & 9
027306 122701 000071 CMPEB #'9,R1
027312 002427 BLT 5$
027314 032716 170000 BIT #^C7777,(SP) ;;DON'T LET NUMBER GET TO BIG
027320 001024 BNE 5$ ;;BR IF NUMBER WOULD OVERFLOW
027322 006316 ASL (SP) ;;*2
027324 011646 MOV (SP),-(SP) ;;SAVE FOR LATER
027326 006316 ASL (SP) ;;*4
027330 006316 ASL (SP) ;;*8
027332 062616 ADD (SP)+,(SP) ;;*10
027334 102416 BVS 5$ ;;OVERFLOW ISN'T ALLOWED
027336 162701 000060 SUB #'0,R1 ;;STRIP AWAY THE ASCII JUNK
027342 060116 ADD R1,(SP) ;;ADD IN THIS DIGIT
027344 102412 BVS 5$ ;;OVERFLOW ISN'T ALLOWED
027346 000752 BR 2$ ;;LOOP
027350 005702 3$: TST R2 ;;CHECK IF NUMBER IS NEG
027352 001401 BEQ 4$ ;;BR IF NO
027354 005416 NEG (SP) ;;YES--NEGATE THE NUMBER
027356 012666 000012 4$: MOV (SP)+,12(SP) ;;SAVE THE RESULT
027362 012602 MOV (SP)+,R2 ;;POP STACK INTO R2
027364 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
027366 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
027370 000002 RTI ;;RETURN
027372 005726 5$: TST (SP)+ ;;CLEAN PARTIAL NUMBER FROM STACK
027374 105010 CLRB (R0) ;;SET A TERMINATOR
027376 104401 TYPE ;;TYPE THE INPUT UP TO BAD CHAR.
027400 000000 6$: .WORD 0 ;;POINTER GOES HERE
027402 104401 001404 TYPE $QUES ;;'?' 'CR' & 'LF'
027406 000720 BR 1$ ;;TRY AGAIN

```

3798

```

.SBTTL READ AN OCTAL NUMBER FROM THE TTY
*****
*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
*CHANGE IT TO BINARY.
*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A '?' WILL BE TYPED
*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
*CALL:
*
*      RDOCT                      ;;READ AN OCTAL NUMBER
*      RETURN HERE                ;;LOW ORDER BITS ARE ON TOP OF THE STACK
*                                  ;;HIGH ORDER BITS ARE IN $HIOCT
$RDOCT: MOV      (SP),-(SP)        ;;PROVIDE SPACE FOR THE
MOV      4(SP),2(SP)              ;;INPUT NUMBER
MOV      R0,-(SP)                 ;;PUSH R0 ON STACK
MOV      R1,-(SP)                 ;;PUSH R1 ON STACK
MOV      R2,-(SP)                 ;;PUSH R2 ON STACK
1$: RDLIN                      ;;READ AN ASCII LINE
MOV      (SP)+,R0                 ;;GET ADDRESS OF 1ST CHARACTER
MOV      R0,5$                   ;;AND SAVE IT
CLR      R1                       ;;CLEAR DATA WORD
CLR      R2
2$: MOVB      (R0)+,-(SP)          ;;PICKUP THIS CHARACTER
BEQ      3$                      ;;IF ZERO GET OUT
CMPB     #'0,(SP)                 ;;MAKE SURE THIS CHARACTER
BGT      4$                      ;;IS AN OCTAL DIGIT
CMPB     #'7,(SP)
BLT      4$
ASL      R1                      ;;*2
ROL      R2                      ;;*4
ASL      R1                      ;;*8
ROL      R2
BIC      #'C7,(SP)               ;;STRIP THE ASCII JUNK
ADD      (SP)+,R1                ;;ADD IN THIS DIGIT
BR       2$                      ;;LOOP
3$: TST      (SP)+                ;;CLEAN TERMINATOR FROM STACK
MOV      R1,12(SP)               ;;SAVE THE RESULT
MOV      R2,$HIOCT
MOV      (SP)+,R2                ;;POP STACK INTO R2
MOV      (SP)+,R1                ;;POP STACK INTO R1
MOV      (SP)+,R0                ;;POP STACK INTO R0
RTI                               ;;RETURN
4$: TST      (SP)+                ;;CLEAN PARTIAL FROM STACK
CLRB     (R0)                   ;;SET A TERMINATOR
TYPE     ;;TYPE UP THRU THE BAD CHAR.
5$: .WORD    0
TYPE     $QUES                  ;; '?' 'CR' & 'LF'
BR       1$                      ;;TRY AGAIN
$HIOCT: .WORD    0                ;;HIGH ORDER BITS GO HERE
  
```

027410	011646		
027412	016666	000004	000002
027420	010046		
027422	010146		
027424	010246		
027426	104412		
027430	012600		
027432	010037	027536	
027436	005001		
027440	005002		
027442	112046		
027444	001420		
027446	122716	000060	
027452	003026		
027454	122716	000067	
027460	002423		
027462	006301		
027464	006102		
027466	006301		
027470	006102		
027472	006301		
027474	006102		
027476	042716	177770	
027502	062601		
027504	000756		
027506	005726		
027510	010166	000012	
027514	010237	027546	
027520	012602		
027522	012601		
027524	012600		
027526	000002		
027530	005726		
027532	105010		
027534	104401		
027536	000000		
027540	104401	001404	
027544	000730		
027546	000000		

3799

```
.SBTTL TRAP DECODER
*****
;THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE 'TRAP' INSTRUCTION
;AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
;OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
;GO TO THAT ROUTINE.
027550 010046          $TRAP: MOV      RO,-(SP)          ;;SAVE RO
027552 016600 000002   MOV      2(SP),RO          ;;GET TRAP ADDRESS
027556 005740          TST      -(RO)             ;;BACKUP BY 2
027560 111000          MOVB     (RO),RO           ;;GET RIGHT BYTE OF TRAP
027562 006300          ASL      RO                ;;POSITION FOR INDEXING
027564 016000 027604   MOV      $TRPAD(RO),RO     ;;INDEX TO TABLE
027570 000200          RTS      RO                ;;GO TO ROUTINE

;;THIS IS USE TO HANDLE THE 'GETPRI' MACRO
027572 011646          $TRAP2: MOV     (SP),-(SP)    ;;MOVE THE PC DOWN
027574 016666 000004 000002 MOV     4(SP),2(SP)    ;;MOVE THE PSW DOWN
027602 000002          RTI                       ;;RESTORE THE PSW

.SBTTL TRAP TABLE
;THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
;BY THE 'TRAP' INSTRUCTION.
;ROUTINE
;-----
027604 027572          $TRPAD: .WORD  $TRAP2
027606 025032          $TYPE  ;;CALL=TYPE          TRAP+1(104401) TTY TYPEOUT ROUTINE
027610 025424          $TYPOC ;;CALL=TYPOC         TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
027612 025400          $TYPOS ;;CALL=TYPOS         TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
027614 025440          $TYPON ;;CALL=TYPON         TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
027616 025636          $TYPDS ;;CALL=TYPDS         TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
027620 025626          $TYPDE ;;CALL=TYPDE         TRAP+6(104406) TYPE DECIMAL NUMBER GREATER THAN 32767
027622 026522          $GTSWR ;;CALL=GTSWR         TRAP+7(104407) GET SOFT-SWR SETTING
027624 026422          $CKSWR ;;CALL=CKSWR         TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
027626 026740          $RDCHR ;;CALL=RDCHR         TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
027630 027060          $RDLIN ;;CALL=RDLIN         TRAP+12(104412) TTY TYPEIN STRING ROUTINE
027632 027410          $RDOCT ;;CALL=RDOCT         TRAP+13(104413) READ AN OCTAL NUMBER FROM TTY
027634 027232          $RDDEC ;;CALL=RDDEC         TRAP+14(104414) READ A DECIMAL NUMBER FROM TTY
```


3800

.SBTTL SCOPE HANDLER ROUTINE

*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT

*AND LOAD THE TEST NUMBER(\$TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)

*AND LOAD THE ERROR FLAG (\$ERFLG) INTO DISPLAY<15:08>

*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:

*SW14=1 LOOP ON TEST

*SW09=1 LOOP ON ERROR

*SW08=1 LOOP ON TEST IN SWR<6:0>

*CALL

* SCOPE ::SCOPE=IOT

027636	032777	00100C	151474	\$SCOPE:	BIT	#BIT09,@SWR	::LOOP ON ERROR?
027644	001406				BEQ	1\$	::BR IF NO
027646	005737	001312			TST	\$ERTTL	::SEE IF THERE WERE ANY ERRORS YET
027652	001403				BEQ	1\$	::BR IF NOT YET
027654	013716	001310			MOV	\$LPERR,(SP)	::FUDGE RETURN
027660	000002				RTI		::RETURN
027662	032777	040000	151450	1\$:	BIT	#BIT14,@SWR	::LOOP ON TEST?
027670	001403				BEQ	2\$	::BR IF NO
027672	013716	001306			MOV	\$LPADR,(SP)	::FUDGE RETURN
027676	000002				RTI		::RETURN
027700	105737	002706		2\$:	TSTB	EOPLOC	::TEST TO SEE IF EOP'S ARE TO PRINT
027704	001043				BNE	6\$	::BRANCH IF NOT
027706	105737	030707			TSTB	CHARCT	::SEE IF A CHARACTER WAS INPUTED IN A PRINT ROUTINE
027712	001406				BEQ	3\$	::BRANCH TO CHECK KEYBOARD IF NOT
027714	113737	030707	002706		MOVB	CHARCT,EOPLOC	::MOVE THE CHARACTER TO EOPLOC
027722	105037	030707			CLRB	CHARCT	::CLEAR THE LOCATION
027726	000411				BR	4\$	::BRANCH TO CHECK THE CHARACTER
027730	105777	151410		3\$:	TSTB	@\$TKS	::SEE IF EOP REQUESTED
027734	100055				BPL	8\$	::BRANCH IF NOT
027736	117737	151404	002706		MOVB	@\$TKB,EOPLOC	::GET CHARACTER
027744	142737	000200	002706		BICB	#200,EOPLOC	::CLEAR PARITY BIT
027752	122737	000030	002706	4\$:	CMPB	#30,EOPLOC	::SEE IF A (^X)
027760	001406				BEQ	5\$	::BRANCH IF IT IS
027762	113737	002706	030707		MOVB	EOPLOC,CHARCT	::MOVE CHARACTER BACK TO CHARCT FOR POSSIBLE FUTURE USE
027770	105037	002706			CLRB	EOPLOC	::CLEAR THE LOCATION
027774	000435				BR	8\$	::BRANCH TO START SCOPE ROUTINE
027776	104401	030326		5\$:	TYPE	,HAKTPM	::TYPE: 'HIT ANY KEY TO OBTAIN A PROGRESS REPORT,'
							::'ENTER (^X) TO RESUME EOP'S AND EOD'S'
							::'ENTER THE KEY REPEATEDLY, AS RESETS DONE IN THE D
							::'MAY CLEAR THE CHARACTER BEFORE THE TESTS FOR THE
030002	105337	002706			DECB	EOPLOC	::MAKE ^X ANOTHER CHARACTER AND
030006	105037	030707			CLRB	CHARCT	::CLEAR ANY CHARACTER THAT MAY BE THERE
030012	000426				BR	8\$	::BRANCH TO START SCOPE ROUTINE
030014	105737	030707		6\$:	TSTB	CHARCT	::SEE IF A CHARACTER WAS INPUTED IN THE TYPE ROUTINE
030020	001011				BNE	7\$	::GO TEST THE CHARACTER IF SO
030022	105777	151316			TSTB	@\$TKS	::SEE IF CHARACTER WAITING
030026	100020				BPL	8\$	::BRANCH IF NOT
030030	117737	151312	030707		MOVB	@\$TKB,CHARCT	::MOVE THE CHARACTER FOR TESTING
030036	142737	000200	030707		BICB	#200,CHARCT	::CLEAR THE PARITY BIT
030044	122737	000030	030707	7\$:	CMPB	#30,CHARCT	::SEE IF ANOTHER (^X) WAS INPUTED
030052	001006				BNE	8\$	::BRANCH IF NOT
030054	105037	002706			CLRB	EOPLOC	::CLEAR EOPLOC TO RESUME EOP MESSAGES
030060	104401	030636			TYPE	,EOPRSM	::TYPE: 'EOP'S AND EOD'S WILL RESUME PRINTING'
030064	105037	030707			CLRB	CHARCT	::MAKE CHARACTER SOMETHING ELSE
030070	105737	030707		8\$:	TSTB	CHARCT	::SEE IF A CHARACTER IS WAITING
030074	001410				BEQ	9\$	::BRANCH AROUND (^Y) TEST IF NOT

```
030076 122737 000031 030707      CMPB    #31,CHARCT      ;## IS THIS A (^Y) (REQUEST FOR RUN SUMMARY)
030104 001004                      BNE      9$                ;## BRANCH IF NOT
030106 004737 003170              JSR      PC,PTCAPT          ;## GO TO SUBROUTINE TO PRINT SUMMARY
030112 105037 030707              CLRB    CHARCT            ;## CLEAR THE CHARCT LOCATION SO SUMMARY NOT REPEATED
030116 104410                      CFSW      CKSWR          ;## TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER
                                9$:      CFSW      CKSWR          ;## TESTER####
                                ;####START OF CODE FOR THE XOR TESTER####
030120 000416                      $XTSTR: BR      6$          ;## IF RUNNING ON THE 'XOR' TESTER CHANGE
                                ;THIS INSTRUCTION TO A 'NOP' (NOP=240)
030122 013746 000004                      MOV     @#ERRVEC,-(SP) ;## SAVE THE CONTENTS OF THE ERROR VECTOR
030126 012737 030146 000004          MOV     #5$,@#ERRVEC      ;## SET FOR TIMEOUT
030134 005737 177060                      TST     @#177060      ;## TIME OUT ON XOR?
030140 012637 000004                      MOV     (SP)+,@#ERRVEC ;## RESTORE THE ERROR VECTOR
030144 000444                      BR      $SVLAD              ;## GO TO THE NEXT TEST
030146 022626                      5$:    CMP     (SP)+,(SP)+    ;## CLEAR THE STACK AFTER A TIME OUT
030150 012637 000004                      MOV     (SP)+,@#ERRVEC ;## RESTORE THE ERROR VECTOR
030154 000432                      BR      7$                ;## LOOP ON THE PRESENT TEST
030156                      6$:;####END OF CODE FOR THE XOR TESTER####
030156 032777 000400 151154          BIT     #BIT08,@SWR        ;## LOOP ON SPEC. TEST?
030164 001432                      BEQ      4$                ;## BR IF NO
030166 005046                      CLR      -(SP)              ;## CLEAR A TEMP. LOCATION
030170 117716 151144                      MOVB   @SWR,(SP)      ;## PICKUP THE DESIRED TEST NUMBER
030174 042716 000200                      BIC     #5$SWRMK,(SP)    ;## MASK OUT UNDESIRED BITS
030200 001416                      BEQ      8$                ;## BRANCH IF BAD TEST NUMBER IN SWR
030202 022716 000037                      CMP     #37,(SP)      ;## CHECK THE NUMBER IN THE SWR
030206 002413                      BLT      8$                ;## BRANCH IF TEST NUMBER IS OUT OF RANGE
030210 011637 001302                      MOV     (SP), $TSTNM    ;## UPDATE THE TEST NUMBER IN $TSTNM
030214 011637 001414                      MOV     (SP), $TESTN    ;## UPDATE THE TEST NUMBER IN $TESTN
030220 005316                      DEC      (SP)              ;## BACKUP BY ONE
030222 006316                      ASL      (SP)              ;## SCALE THE TEST NUMBER AS AN INDEX
030224 062716 030752                      ADD     #5$SW08TBL,(SP) ;## FORM THE ADDRESS OF TEST POINTER
030230 013637 001306                      MOV     @ (SP)+,$LPADR ;## SET LOOP ADDRESS TO DESIRED TEST
030234 000426                      BR      $OVER              ;## GO LOOP ON THE TEST
030236 005726                      8$:    TST     (SP)+          ;## CLEAN THE BAD TEST NUMBER OFF OF THE STACK
030240                      2$:    ;TSTB   $ERFLG              ;## HAS AN ERROR OCCURRED? ;## ELIMINATED FOR CZDRLC
030240 001406                      BEQ      $SVLAD              ;## BR IF NO
030242 013737 001310 001306          7$:    MOV     $LPERR,$LPADR ;## SET LOOP ADDRESS TO LAST SCOPE
030250 000420                      BR      $OVER              ;##
030252 105037 001303                      4$:    CLRB    $ERFLG      ;## ZERO THE ERROR FLAG
030256 105237 001302                      $SVLAD: INCB    $TSTNM    ;## COUNT TEST NUMBERS
030262 113737 001302 001414          MOVB   $TSTNM,$TESTN    ;## SET TEST NUMBER IN APT MAILBOX
030270 011637 001306                      MOV     (SP), $LPADR    ;## SAVE SCOPE LOOP ADDRESS
030274 011637 001310                      MOV     (SP), $LPERR    ;## SAVE ERROR LOOP ADDRESS
030300 005037 001376                      CLR     $ESCAPE      ;## CLEAR THE ESCAPE FROM ERROR ADDRESS
030304 112737 000001 001315          MOVB   #1,$ERMAX      ;## ONLY ALLOW ONE(1) ERROR ON NEXT TEST
030312 013777 001302 151022          $OVER: MOV     $TSTNM,@DISPLAY ;## DISPLAY TEST NUMBER
030320 013716 001306                      MOV     $LPADR,(SP)    ;## FUDGE RETURN ADDRESS
030324 000002                      RTI                          ;## RETURN
030326 136 130 200 HAKTPM: .ASCII /^X/<CRLF>/HIT ANY KEY TO OBTAIN A PROGRESS REPORT,/<CRLF> ;##
030402 105 116 124 .ASCII /ENTER (^X) TO RESUME EOP'S AND EOD'S/<CRLF> ;##
030447 105 116 124 .ASCII /ENTER THE KEY REPEATEDLY, AS RESETS DONE IN THE DIAGNOSTIC/<CRLF> ;##
030542 115 101 131 .ASCII /MAY CLEAR THE CHARACTER BEFORE THE TESTS FOR THE CHARACTER/<CRLF> ;##
030636 136 130 200 EOPRSM: .ASCII /^X/<CRLF>/EOP'S AND EOD'S WILL RESUME PRINTING/<CRLF> ;##
030707 000 CHARCT: .BYTE 0 ;## LOCATION TO HOLD INPUTED CHARACTER
030710 040 040 124 TESLR: .ASCII / TOTAL ERRORS SINCE LAST REPORT / ;## ERROR MESSAGE FOR $EOP
                                .EVEN
```


030752		\$SW08TBL:		
030752	011274	.WORD	TST1+30	; STARTING ADDRESS+30 OF TEST 1
030754	011472	.WORD	TST2+30	; STARTING ADDRESS+30 OF TEST 2
030756	011622	.WORD	TST3+30	; STARTING ADDRESS+30 OF TEST 3
030760	012154	.WORD	TST4+30	; STARTING ADDRESS+30 OF TEST 4
030762	012760	.WORD	TST5+30	; STARTING ADDRESS+30 OF TEST 5
030764	013154	.WORD	TST6+30	; STARTING ADDRESS+30 OF TEST 6
030766	013522	.WORD	TST7+30	; STARTING ADDRESS+30 OF TEST 7
030770	013700	.WORD	TST10+30	; STARTING ADDRESS+30 OF TEST 10
030772	014134	.WORD	TST11+30	; STARTING ADDRESS+30 OF TEST 11
030774	014320	.WORD	TST12+30	; STARTING ADDRESS+30 OF TEST 12
030776	014504	.WORD	TST13+30	; STARTING ADDRESS+30 OF TEST 13
031000	014670	.WORD	TST14+30	; STARTING ADDRESS+30 OF TEST 14
031002	015060	.WORD	TST15+30	; STARTING ADDRESS+30 OF TEST 15
031004	015312	.WORD	TST16+30	; STARTING ADDRESS+30 OF TEST 16
031006	015550	.WORD	TST17+30	; STARTING ADDRESS+30 OF TEST 17
031010	016146	.WORD	TST20+30	; STARTING ADDRESS+30 OF TEST 20
031012	016552	.WORD	TST21+30	; STARTING ADDRESS+30 OF TEST 21
031014	017066	.WORD	TST22+30	; STARTING ADDRESS+30 OF TEST 22
031016	017364	.WORD	TST23+30	; STARTING ADDRESS+30 OF TEST 23
031020	020122	.WORD	TST24+30	; STARTING ADDRESS+30 OF TEST 24
031022	020424	.WORD	TST25+30	; STARTING ADDRESS+30 OF TEST 25
031024	020714	.WORD	TST26+30	; STARTING ADDRESS+30 OF TEST 26
031026	021172	.WORD	TST27+30	; STARTING ADDRESS+30 OF TEST 27
031030	021410	.WORD	TST30+30	; STARTING ADDRESS+30 OF TEST 30
031032	021676	.WORD	TST31+30	; STARTING ADDRESS+30 OF TEST 31
031034	022026	.WORD	TST32+30	; STARTING ADDRESS+30 OF TEST 32
031036	022406	.WORD	TST33+30	; STARTING ADDRESS+30 OF TEST 33
031040	022712	.WORD	TST34+30	; STARTING ADDRESS+30 OF TEST 34
031042	023274	.WORD	TST35+30	; STARTING ADDRESS+30 OF TEST 35
031044	023572	.WORD	TST36+30	; STARTING ADDRESS+30 OF TEST 36
031046	024106	.WORD	TST37+30	; STARTING ADDRESS+30 OF TEST 37

3801

```
.SBTTL ERROR HANDLER ROUTINE
*****
*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
*AND GO TO $ERRTYP ON ERROR
*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
*SW15=1      HALT ON ERROR
*SW13=1      INHIBIT ERROR TYPEOUTS
*SW10=1      BELL ON ERROR
*SW09=1      LOOP ON ERROR
*CALL
*
*      ERROR      N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER
$ERROR:
031050      104410      001303      7$:      CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR
031050      105237      001303      7$:      INCB      $ERFLG      ;;SET THE ERROR FLAG
031056      001775      001302      150254      BEQ      7$      ;;DON'T LET THE FLAG GO TO ZERO
031060      013777      002000      150244      MOV      $TSTNM,@DISPLAY      ;;DISPLAY TEST NUMBER AND ERROR FLAG
031066      032777      001402      150244      BIT      #BIT10,@SWR      ;;BELL ON ERROR?
031074      001402      001400      1$      BEQ      1$      ;;NO - SKIP
031076      104401      001312      1$      TYPE      ,SBELL      ;;RING BELL
031102      005237      002714      1$      INC      $ERTTL      ;;COUNT THE NUMBER OF ERRORS
031106      005237      001316      1$      INC      ERRCNT      ;;INCREMENT THE ERROR COUNT
031112      011637      000002      001316      MOV      (SP),$ERRPC      ;;GET ADDRESS OF ERROR INSTRUCTION
031116      162737      150166      001314      SUB      #2,$ERRPC
031124      117737      020000      150200      MOVB     @ERRPC,$ITEMB      ;;STRIP AND SAVE THE ERROR ITEM CODE
031132      032777      031250      20$:      BIT      #BIT13,@SWR      ;;SKIP TYPEOUT IF SET
031140      001002      031250      20$:      BNE      20$      ;;SKIP TYPEOUTS
031142      004737      031250      20$:      JSR      PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
031146      122737      000001      001430      CMPB     #APTENV,$ENV      ;;RUNNING IN APT MODE
031154      001007      001314      031170      BNE      2$      ;;NO, SKIP APT ERROR REPORT
031156      113737      031544      21$:      MOVB     $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
031164      004737      031544      21$:      JSR      PC,$ATY4      ;;REPORT FATAL ERROR TO APT
031170      000      21$:      .BYTE      0
031171      000      21$:      .BYTE      0
031172      000777      150140      22$:      BR      22$      ;;APT ERROR LOOP
031174      005777      150140      2$:      TST      @SWR      ;;HALT ON ERROR
031200      100002      3$:      BPL      3$      ;;SKIP IF CONTINUE
031202      000000      3$:      HALT      ;;HALT ON ERROR!
031204      104410      3$:      CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR
031206      005737      001376      3$:      TST      $ESCAPE      ;;CHECK FOR AN ESCAPE ADDRESS
031212      001402      001376      4$:      BEQ      4$      ;;BR IF NONE
031214      013716      001000      150112      MOV      $ESCAPE,(SP)      ;;FUDGE RETURN ADDRESS FOR ESCAPE
031220      032777      001310      4$:      BIT      #BIT9,@SWR      ;;SEE IF WE ARE TO LOOP ON ERROR
031226      001402      5$:      BEQ      5$      ;;BRANCH OUT IF NOT
031230      013716      000042      5$:      MOV      $LPERR,(SP)      ;;FUDGE RETURN
031234      022737      024750      000042      CMP      #SENDAD,@#42      ;;ACT-11 AUTO-ACCEPT?
031242      001001      6$:      BNE      6$      ;;BRANCH IF NO
031244      000000      6$:      HALT      ;;YES
031246      000002      RTI      ;;RETURN
```

3802

.SBTTL ERROR MESSAGE TYPEOUT ROUTINE

*THIS ROUTINE USES THE 'ITEM CONTROL BYTE' (\$ITEMB) TO DETERMINE WHICH
*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE 'ERROR TABLE' (\$ERRTB),
*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
\$ERRTYP:

031250					MOV	RO,-(SP)	::SAVE RO
031250	010046				CLR	RO	::CLEAR RO TO RECEIVE ITEM INDEX
031252	005000				MOVB	@\$ITEMB,RO	::PICKUP THE ITEM INDEX
031254	113700	001314			BNE	1\$	::IF ITEM NUMBER IS ZERO, TYPE THE PC OF THE ERROR
031260	001004				MOV	\$ERRPC,-(SP)	::SAVE \$ERRPC FOR TYPEOUT
031262	013746	001316					::ERROR ADDRESS
							::GO TYPE--OCTAL ASCII(ALL DIGITS)
031266	104402				TYPOC		::GET OUT
031270	000513				BR	14\$	::
031272	010037	001360		1\$:	MOV	RO,\$TMP0	::MOVE RO TO \$TMP0 FOR 200 TEST
031276	042700	000200			BIC	#200,RO	::CLEAR BIT 7 IF PRESENT
031302	005300				DEC	RO	::MAKE POINTER AN INDEX
031304	006300				ASL	RO	::SHIFT TO MULTIPLY BY 10 (OCTAL)
031306	006300				ASL	RO	::
031310	006300				ASL	RO	::
031312	105737	001360			TSTB	\$TMP0	::SEE IF ITEM NUMBER IS OVER 200
031316	100041				BPL	4\$	::BRANCH IF ITEM NUMBER IS LESS THAN 200
031320	023727	002714	000020		CMP	ERRCNT,#20	::SEE IF 20 (OCTAL) ERRORS HAVE PRINTED
031326	002404				BLT	2\$	::BRANCH TO PRINT THE ERROR IF LESS
031330	003073				BGT	14\$	::BRANCH TO RETURN IF GREATER - NO MORE DATA IS TO PRINT
031332	104401	036663			TYPE	,NOMORE	::TYPE MESSAGE ANNOUNCING NO MORE PRINTING OF ERRORS
031336	000470				BR	14\$	::BRANCH TO RETURN
031340	022737	000001	002714	2\$:	CMP	#1,ERRCNT	::SEE IF THIS IS THE FIRST ERROR
031346	001415				BEQ	3\$	::BRANCH IF IT WAS AND GO TYPE ERROR MESSAGE
031350	123737	001360	031524		CMPB	\$TMP0,MEPITM	::SEE IF ITEM MATCHES LAST MULTIPLE ERROR
031356	001011				BNE	3\$	::BRANCH IF NOT - NEW HEADER NEEDED
031360	032777	000200	147752		BIT	#BIT7,@SWR	::SEE IF SWITCH REGISTER BIT 7 IS SET
031366	001054				BNE	14\$	::BRANCH TO RETURN IF SWITCH SET
031370	042700	177400			BIC	#177400,RO	::CLEAR UPPER BYTE OF RO EXPOSING ITEM BYTE
031374	062700	002300			ADD	#ER200+4,RO	::POINT TO DATA TABLE ENTRY
031400	000434				BR	9\$	::BRANCH TO PRINT DATA
031402	113737	001360	031524	3\$:	MOVB	\$TMP0,MEPITM	::MOVE ITEM NUMBER TO MEPITM FOR POSSIBLE FUTURE USE
031410	042700	177000			BIC	#177000,RO	::CLEAR UPPER BYTE OF RO
031414	062700	000540			ADD	#ER200-\$ERRTB,RO	::ADD 200 BASE POINTER TO RO AND
031420	000402				BR	5\$	::BRANCH AROUND ERRCNT CLEAR
031422	005037	002714		4\$:	CLR	ERRCNT	::CLEAR ERRCNT SO MULTIPLE ERRORS GET NEW HEADER
031426	104401	001405		5\$:	TYPE	,SCRLF	::TYPE <CRLF>
031432	062700	001534			ADD	#\$ERRTB,RO	::FORM TABLE POINTER
031436	012037	031446			MOV	(RO)+,6\$	::PICKUP 'ERROR MESSAGE' POINTER
031442	001404				BEQ	7\$	::SKIP TYPEOUT IF NO POINTER
031444	104401				TYPE		::TYPE THE 'ERROR MESSAGE'
031446	000000			6\$:	.WORD	0	::'ERROR MESSAGE' POINTER GOES HERE
031450	104401	001405			TYPE	,SCRLF	::'CARRIAGE RETURN' & 'LINE FEED'
031454	012037	031464		7\$:	MOV	(RO)+,8\$	::PICKUP 'DATA HEADER' POINTER
031460	001404				BEQ	9\$	::SKIP TYPEOUT IF 0
031462	104401				TYPE		::TYPE THE 'DATA HEADER'
031464	000000			8\$:	.WORD	0	::'DATA HEADER' POINTER GOES HERE
031466	104401	001405			TYPE	,SCRLF	::'CARRIAGE RETURN' & 'LINE FEED'
031472	011000			9\$:	MOV	(RO),RO	::PICKUP 'DATA TABLE' POINTER
031474	001407				BEQ	13\$	::GO AROUND ROUTINE TO TYPE THE DATA IF NONE
031476	013046			10\$:	MOV	@(RO)+,-(SP)	::PUT OCTAL DATA ON STACK FOR TYPING
031500	104402				TYPOC		::TYPE AN OCTAL NUMBER

CZDRLCO-DR11 GEN NPR INTFC
ERROR MESSAGE TYPEOUT ROUTINE

MACRO M1113 27-OCT-80 15:53 PAGE 133-1 F 12

SEQ 0148

031502 005710
031504 001403
031506 104401 035250
031512 000771
031514 104401 001405
031520 012600
031522 000207
031524 000000

TST (R0)
BEQ 13\$
TYPE ,SPACES
BR 10\$
13\$: TYPE ,\$CRLF
14\$: MOV (SP)+,R0
RTS PC
MEPITM: .WORD 0

::: IS THERE ANOTHER NUMBER?
::: BR IF NO
::: TYPE TWO(2) SPACES
::: GO BACK TO PRINT THE OCTAL NUMBER
::: 'CARRIAGE RETURN' & 'LINE FEED'
::: RESTORE R0
::: RETURN
::: && LOCATION TO STORE 200+ ERROR ITEM NUMBER

3803

```
.SBTTL APT COMMUNICATIONS ROUTINE
*****
031526 112737 000001 031772 $ATY1: MOVB #1,$FFLG ;;TO REPORT FATAL ERROR
031534 112737 000001 031770 $ATY3: MOVB #1,$MFLG ;;TO TYPE A MESSAGE
031542 000403          BR $ATYC
031544 112737 000001 031772 $ATY4: MOVB #1,$FFLG ;;TO ONLY REPORT FATAL ERROR
031552 $ATYC:
031552 010046          MOV R0,-(SP) ;;PUSH R0 ON STACK
031554 010146          MOV R1,-(SP) ;;PUSH R1 ON STACK
031556 105737 031770          TSTB $MFLG ;;SHOULD TYPE A MESSAGE?
031562 001450          BEQ 5$ ;;IF NOT: BR
031564 122737 000001 001430          CMPB #APTENV,$ENV ;;OPERATING UNDER APT?
031572 001031          BNE 3$ ;;IF NOT: BR
031574 132737 000100 001431          BITB #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
031602 001425          BEQ 3$ ;;IF NOT: BR
031604 017600 000004          MOV @4(SP),R0 ;;GET MESSAGE ADDR.
031610 062766 000002 000004          ADD #2,4(SP) ;;BUMP RETURN ADDR.
031616 005737 001410          1$: TST $MSGTYPE ;;SEE IF DONE W/ LAST XMISSION?
031622 001375          BNE 1$ ;;IF NOT: WAIT
031624 010037 001424          MOV R0,$MSGAD ;;PUT ADDR IN MAILBOX
031630 105720          2$: TSTB (R0)+ ;;FIND END OF MESSAGE
031632 001376          BNE 2$
031634 163700 001424          SUB $MSGAD,R0 ;;SUB START OF MESSAGE
031640 006200          ASR R0 ;;GET MESSAGE LGTH IN WORDS
031642 010037 001426          MOV R0,$MSGGLT ;;PUT LENGTH IN MAILBOX
031646 012737 000004 001410          MOV #4,$MSGTYPE ;;TELL APT TO TAKE MSG.
031654 000413          BR 5$
031656 017637 000004 031702          3$: MOV @4(SP),4$ ;;PUT MSG ADDR IN JSR LINKAGE
031664 062766 000002 000004          ADD #2,4(SP) ;;BUMP RETURN ADDRESS
031672 013746 177776          MOV 177776,-(SP) ;;PUSH 177776 ON STACK
031676 004737 025032          JSR PC,$TYPE ;;CALL TYPE MACRO
031702 000000          4$: .WORD 0
031704          5$:
031704 105737 031772          10$: TSTB $FFLG ;;SHOULD REPORT FATAL ERROR?
031710 001416          BEQ 12$ ;;IF NOT: BR
031712 005737 001430          TST $ENV ;;RUNNING UNDER APT?
031716 001413          BEQ 12$ ;;IF NOT: BR
031720 005737 001410          11$: TST $MSGTYPE ;;FINISHED LAST MESSAGE?
031724 001375          BNE 11$ ;;IF NOT: WAIT
031726 017637 000004 001412          MOV @4(SP),$FATAL ;;GET ERROR #
031734 062766 000002 000004          ADD #2,4(SP) ;;BUMP RETURN ADDR.
031742 005237 001410          INC $MSGTYPE ;;TELL APT TO TAKE ERROR
031746 105037 031772          12$: CLRB $FFLG ;;CLEAR FATAL FLAG
031752 105037 031771          CLRB $LFLG ;;CLEAR LOG FLAG
031756 105037 031770          CLRB $MFLG ;;CLEAR MESSAGE FLAG
031762 012601          MOV (SP)+,R1 ;;POP STACK INTO R1
031764 012600          MOV (SP)+,R0 ;;POP STACK INTO R0
031766 000207          RTS PC ;;RETURN
031770          000          $MFLG: .BYTE 0 ;;MESSG. FLAG
031771          000          $LFLG: .BYTE 0 ;;LOG FLAG
031772          000          $FFLG: .BYTE 0 ;;FATAL FLAG
                                .EVEN
                                APTSIZE=200
                                APTENV=001
                                APTSPOOL=100
                                APTCSUP=040
.SBTTL POWER DOWN AND UP ROUTINE
```

3804

```
*****
:POWER DOWN AND UP ROUTINE
031774 012737 032012 000024 $PWRDN: MOV #SPWRUP,PWRVEC ;## SET UP VECTOR TO RETURN TO THE HALT BELOW
032002 012737 000340 000026      MOV #LEVEL7,PWRVEC+2;## RETURN PRIORITY TO 7
032010 000000      HALT ;:HALT PROCESSOR
032012 012737 031774 000024 $PWRUP: MOV #SPWRDN,PWRVEC ;## RESET PWRVEC TO PWRDN ROUTINE AND
032020 012737 000340 000026      MOV #LEVEL7,PWRVEC+2;## PRIORITY TO 7
032026 012706 001300      MOV #STACK,SP ;## REINITIALIZE THE STACK,
032032 012746 000340      MOV #LEVEL7,-(SP) ;## SET UP RETURN PRIORITY TO 7 AND
032036 012746 000210      MOV #STAGIN,-(SP) ;## MOVE STAGIN ADDRESS TO STACK AND
032042 005037 001360      CLR $TMP0 ;## CLEAR WAIT LOOP COUNTER
032046 005237 001360      1$: INC $TMP0 ;## GIVE TTY TIME TO RECOVER FROM POWER FAILURE
032052 001375      BNE 1$ ;## BRANCH BACK UNTIL ZERO AGAIN
032054 104401 032062      TYPE , $POWER ;## TYPE THE POWER FAILURE MESSAGE ASCIZED BELOW
032060 000002      RTI ;## RETURN TO PROGRAM
032062 200 120 117 $POWER: .ASCIZ <CRLF>/POWER FAILURE - RESTARTING PROGRAM/<CRLF>
      .EVEN
```



```
3806 .SBTTL MULTIPLE BOARD DIALOGUE ROUTINE
3807 :*****
3808 :>>>>>>MULTIPLE BOARD DIALOGUE ROUTINE<<<<<<<
3809 :*****
3810 MBD: MOV #STACK,SP ;INITIALIZE THE STACK
3811 JSR PC,SETUP ;GO INITIALIZE THE COMMON TAGS
3812 TYPE ,MBDIAL ;TYPE: 'MULTIPLE BOARD DIALOGUE'
3813 PROMPT: CLRB CHARCT ;CLEAR LOCATION FOR POSSIBLE INPUT DURING PRINT
3814 TYPE ,ECLR ;TYPE: 'ENTER COMMAND ([E]DIT, [L]IST, [B]URST CALIBRATION,
3815 MOV #1,R3 ;EXPECT 1 CHARACTER
3816 JSR PC,READ ;GO READ 1 CHARACTER
3817 CMP #'L,ANSWER ;LIST PRESENT TABLE?
3818 BNE 1$ ;BRANCH IF NO
3819 TYPE ,HEADER ;TYPE: '# OF START
3820 ; 'BOARDS REGADR VECADR W-B P-LEV CYCLE 2-N T
3821 MOV $DDW0,DDW ;SET UP THE DEVICE DESCRIPTOR WORD FOR PRINTING
3822 MOV QTYBRD,-(SP) ;MOVE NUMBER OF DEVICES TO STACK FOR TYPING
3823 TYPDS ;TYPE THE NUMBER OF DEVICES
3824 TYPE ,SPACE3 ;TYPE 3 SPACE CHARACTERS
3825 MOV REGADR,-(SP) ;MOVE THE DEVICE REGISTER ADDRESS TO THE STACK
3826 TYPOC ;TYPE THE DEVICE REGISTER ADDRESS
3827 TYPE ,SPACE6 ;TYPE 6 SPACE CHARACTERS
3828 MOV VECADR,-(SP) ;MOVE THE DEVICE VECTOR ADDRESS TO THE STACK
3829 TYPOS ;TYPE THE DEVICE VECTOR ADDRESS
3830 .BYTE 3,0 ;TYPE 3 CHARACTERS, LEADING ZEROS SUPPRESSED
3831 TYPE ,SPACE7 ;TYPE 7 SPACE CHARACTERS
3832 BIT #BIT0,DDW ;SEE WHICH W/B STATE FOR BOARDS
3833 BEQ 10$ ;GO PRINT W STATE IF W
3834 TYPE ,B ;TYPE A 'B'
3835 BR 11$ ;GO TO NEXT CHECK
3836 TYPE ,W ;TYPE A 'W'
3837 TYPE ,SPACE7 ;TYPE 7 SPACE CHARACTERS
3838 JSR PC,PNTPRI ;PRINT DEVICE PRIORITY
3839 TYPE ,SPACE7 ;TYPE 7 SPACE CHARACTERS
3840 BIT #BIT1,DDW ;SEE WHICH 2/N STATE FOR BOARDS
3841 BNE 12$ ;GO PRINT N STATE IF N
3842 TYPE ,TWO ;TYPE A '2'
3843 BR 13$ ;GO TO NEXT CHECK
3844 TYPE ,N ;TYPE AN 'N'
3845 TYPE ,SPACE7 ;TYPE 7 SPACE CHARACTERS
3846 BIT #BIT2,DDW ;SEE WHICH CABLE STATE FOR BOARDS
3847 BEQ 14$ ;GO PRINT NO CABLE IF NONE
3848 TYPE ,YES ;TYPE 'YES'
3849 BR 15$ ;BRANCH TO CONTINUE
3850 TYPE ,NO ;TYPE 'NO'
3851 TYPE ,CRLF2 ;TYPE 2 <CRLF>'S
3852 BR PROMPT ;BRANCH TO PROMPT ANOTHER COMMAND
3853 CMP #'R,ANSWER ;RUN PROGRAM?
3854 BNE 4$ ;BRANCH IF NOT
3855 TST REGADR ;SEE IF REGADR HAS BEEN LOADED
3856 BNE 3$ ;BRANCH TO CHECK VECADR IF SO
3857 TYPE ,MUSTED ;TYPE: 'DEVICE ADDRESS AND/OR VECTOR TABLE NOT SET UP - MUS
3858 BR PROMPT ;BRANCH BACK FOR PROMPT MESSAGE
3859 TST VECADR ;SEE IF VECADR HAS BEEN LOADED
3860 BEQ 2$ ;BRANCH BACK TO PRINT ERROR MESSAGE IF NOT
3861 JSR PC,FIXTBL ;FILL TABLE
3862 MOV #-1,MANSIZE ;MOVE -1 TO MANSIZE TO INDICATE WE HAVE MANUALLY SIZED
```

3863	032426	000137	010272		JMP	START	;JUMP TO START
3864	032432	022737	000105	002664 4\$:	CMP	#'E,ANSWER	;EDIT TABLE?
3865	032440	001414			BEQ	EDIT	;BRANCH TO EDIT IF SO
3866	032442	022737	000102	002664	CMP	#'B,ANSWER	;ENTER BURST DATA LATE CALIBRATION?
3867	032450	001235			BNE	PROMPT	;BRANCH TO PROMPT IF COMMAND NOT RECOGNIZED
3868	032452	005737	002416		TST	REGADR	;SEE IF REGADR HAS BEEN LOADED
3869	032456	001750			BEQ	2\$	;BRANCH TO ERROR MESSAGE IF NOT
3870	032460	005737	002456		TST	VECADR	;SEE IF VECADR HAS BEEN LOADED
3871	032464	001745			BEQ	2\$	;BRANCH TO ERROR MESSAGE IF NOT
3872	032466	000137	033612		JMP	BDLCR	;JUMP TO BURST DATA LATE CALIBRATION ROUTINE

3873					.SBTTL	TABLE EDIT ROUTINE		
3874	032472	104401	035433		TYPE	,NOBUT	:TYPE: 'NUMBER OF BOARDS UNDER TEST: '	
3875	032476	013746	002414		MOV	QTYBRD,-(SP)	:PUT QUANTITY OF BOARDS ON STACK FOR PRINTING	
3876	032502	104403			TYPDS		:GO PRINT THE QUANTITY OF BOARDS	
3877	032504	104401	035276		TYPE	,SPACEC	:TYPE: ' : '	
3878	032510	012703	000002		MOV	#2,R3	:EXPECT MAX OF 2 CHARACTERS	
3879	032514	112737	000071	002704	MOVB	#'9,LRGSTC	:MOVE ASCII '9' TO THE LARGEST CHARACTER LOCATION	
3880	032522	004737	002722		JSR	PC,READ	:GO READ 2 CHARACTERS	
3881	032526	022703	000002		CMP	#2,R3	:SEE IF NON-NUMERIC WAS THE ONLY INPUT	
3882	032532	001017			BNE	2\$	:BRANCH IF NOT	
3883	032534	022737	000033	002664	CMP	#ESC,ANSWER	:SEE IF USER WANTS TO GO BACK TO PREVIOUS PROMPT	
3884	032542	001453			BEQ	5\$	:BRANCH TO PROMPT JUMP IF SO	
3885	032544	022737	000003	002664	CMP	#CNTLC,ANSWER	:SEE IF USER WANTS TO EXIT (^C)	
3886	032552	001447			BEQ	5\$	:BRANCH TO PROMPT JUMP IF EXIT REQUESTED	
3887	032554	022737	000015	002664	CMP	#CARETN,ANSWER	:SEE IF A <CR> WAS INPUTED	
3888	032562	001412			BEQ	4\$	:IF <CR> USE EXISTING NUMBER	
3889	032564	104401	034307	1\$:	TYPE	,BDNERR	:TYPE: 'ILLEGAL NUMBER (# OTHER THAN 1-16) OR CHARACTER INP	
3890	032570	000740			BR	EDIT	:BRANCH BACK FOR NEW INPUT	
3891	032572	005704		2\$:	TST	R4	:CHECK FOR ZERO MODULES INPUT	
3892	032574	001773			BEQ	1\$	:BRANCH TO PRINT ERROR MESSAGE IF SO	
3893	032576	022704	000020		CMP	#20,R4	:SEE IF BOARD NUMBER IS ILLEGAL	
3894	032602	100770			BMI	1\$	:BRANCH TO PRINT ERROR MESSAGE IF SO	
3895	032604	010437	002414	3\$:	MOV	R4,QTYBRD	:MOVE INPUTED NUMBER TO QTYBRD	
3896	032610	104401	036204	4\$:	TYPE	,SDADRS	:TYPE: ' STARTING DEVICE ADDRESS: '	
3897	032614	013746	002416		MOV	REGADR,-(SP)	:MOVE THE PRESENT ADDRESS TO THE STACK FOR PRINTING	
3898	032620	104402			TYPOC		:PRINT THE ADDRESS	
3899	032622	104401	035276		TYPE	,SPACEC	:TYPE: ' : '	
3900	032626	012703	000006		MOV	#6,R3	:EXPECT MAXIMUM 6 CHARACTERS	
3901	032632	112737	000067	002704	MOVB	#'7,LRGSTC	:MOVE ASCII '7' TO THE LARGEST CHARACTER LOCATION	
3902	032640	004737	002722		JSR	PC,READ	:GO READ 6 CHARACTERS	
3903	032644	022703	000006		CMP	#6,R3	:SEE IF NON-NUMERIC CHARACTER INPUTED	
3904	032650	001022			BNE	8\$	:BRANCH IF NOT	
3905	032652	022737	000033	002664	CMP	#ESC,ANSWER	:SEE IF USER WANTS TO GO BACK TO PREVIOUS PROMPT	
3906	032660	001704			BEQ	EDIT	:BRANCH AROUND ESC PRINT AND PREVIOUS PROMPT BRANCH IF SO	
3907	032662	022737	000003	002664	CMP	#CNTLC,ANSWER	:SEE IF USER WANTS TO EXIT (^C)	
3908	032670	001005			BNE	7\$	:BRANCH AROUND PROMPT JUMP IF NOT	
3909	032672	000137	032144	5\$:	JMP	PROMPT	:JUMP TO PROMPT A NEW COMMAND	
3910	032676	104401	034432	6\$:	TYPE	,ADRERR	:TYPE: 'ADDRESS INPUTED IS NOT IN THE RANGE 171000 TO 17700	
3911	032702	000742			BR	4\$	:BRANCH BACK FOR REINPUT	
3912	032704	022737	000015	002664	7\$:	CMP	#CARETN,ANSWER	:SEE IF <CR> WAS ONLY CHARACTER INPUTED
3913	032712	001417			BEQ	10\$	:IF <CR> USE EXISTING REG ADDRESS	
3914	032714	000735			BR	4\$	:BRANCH BACK - INPUT NOT LEGAL	
3915	032716	022704	171000	8\$:	CMP	#171000,R4	:IS ANSWER BELOW 171000	
3916	032722	101365			BHI	6\$	:BRANCH TO PRINT ERROR MESSAGE IF IT IS	
3917	032724	022704	177000		CMP	#177000,R4	:IS ANSWER ABOVE 177000	
3918	032730	103762			BLO	6\$	:BRANCH TO PRINT ERROR MESSAGE IF NOT	
3919	032732	032704	000007		BIT	#7,R4	:TEST TO MAKE SURE A '0' IS PRESENT IN LOWEST OCTAL DIGIT	
3920	032736	001403			BEQ	9\$	:BRANCH AROUND ERROR MESSAGE TYPE IF SO	
3921	032740	104401	034600		TYPE	,ADLCHR	:TYPE: 'ADDRESS INPUTED HAS OTHER THAN 0 FOR LEAST'	
3922							: 'SIGNIFICANT OCTAL DIGIT'	
3923	032744	000721			BR	4\$	:BRANCH BACK FOR REINPUT	
3924	032746	010437	002416	9\$:	MOV	R4,REGADR		

3930	032772	012703	000003		MOV	#3,R3	:EXPECT ONLY 3 CHARACTERS
3931	032776	004737	002722		JSR	PC,READ	:GO READ 3 CHARACTERS
3932	033002	022703	000003		CMP	#3,R3	:SEE IF NON-NUMERIC WAS THE ONLY INPUT
3933	033006	001015			BNE	11\$	:BRANCH IF NOT
3934	033010	022737	000033	002664	CMP	#ESC,ANSWER	:SEE IF USER WANTS TO GO BACK TO PREVIOUS PROMPT
3935	033016	001674			BEQ	4\$	:BRANCH TO PREVIOUS PROMPT IF SO
3936	033020	022737	000003	002664	CMP	#CNTLC,ANSWER	:SEE IF USER WANTS TO EXIT (^C)
3937	033026	001721			BEQ	5\$	:BRANCH TO PROMPT JUMP IF SO
3938	033030	022737	000015	002664	CMP	#CARETN,ANSWER	:SEE IF <CR> WAS INPUTED
3939	033036	001417			BEQ	15\$	:BRANCH IF NO CHANGE WANTED
3940	033040	000744			BR	10\$	:BRANCH BACK - INPUT WAS ILLEGAL
3941	033042	022704	000123		11\$: CMP	#123,R4	:SEE IF ANSWER IS BELOW 124
3942	033046	100403			BMI	13\$	:BRANCH AROUND ERROR MESSAGE IF NOT
3943	033050	104401	034517		TYPE	,VECERR	:TYPE: 'VECTOR INPUTED IS NOT IN THE RANGE OF 124 TO 777'
3944	033054	000736			BR	10\$	:BRANCH BACK FOR REINPUT
3945	033056	032704	000003		13\$: BIT	#3,R4	:MAKE SURE LEAST SIGNIFICANT OCTAL DIGIT IS '0' OR '4'
3946	033062	001403			BEQ	14\$	:BRANCH OVER ERROR PRINTING IF NOT
3947	033064	104401	034703		TYPE	,VCLCHR	:TYPE: 'VECTOR INPUTED SHOULD HAVE ZERO AS LEAST DIGIT'
3948	033070	000730			BR	10\$	:BRANCH BACK FOR REINPUT
3949	033072	010437	002456		14\$: MOV	R4,VECADR	:INSTALL NEW VECTOR ADDRESS IN TABLE
3950	033076	104401	036077		15\$: TYPE	,DR1WOB	:TYPE: 'DR11-W OR B (W=0, B=1) CURRENT STATE = '
3951	033102	013737	001474	002720	MOV	\$DDWO,DDW	:MOVE DEVICE DESCRIPTOR WORD TO DDW
3952	033110	012737	000001	002710	MOV	#BIT0,BITTST	:MOVE BIT STATE TO PRINT TO BITTST
3953	033116	004737	005606		JSR	PC,PSTATE	:PRINT CURRENT W/B STATE
3954	033122	104401	035276		TYPE	,SPACEC	:TYPE: ' '
3955	033126	012703	000001		MOV	#1,R3	:ONLY INPUT 1 CHARACTER
3956	033132	004737	002722		JSR	PC,READ	:GO READ 1 CHARACTER
3957	033136	005703			TST	R3	:SEE IF NON-NUMERIC WAS THE ONLY INPUT
3958	033140	001415			BEQ	16\$	:BRANCH AROUND NON-NUMERIC TESTS IF SO
3959	033142	022737	000033	002664	CMP	#ESC,ANSWER	:SEE IF USER WANTS TO GO BACK TO PREVIOUS PROMPT
3960	033150	001700			BEQ	10\$	:BRANCH TO PREVIOUS PROMPT IF SO
3961	033152	022737	000015	002664	CMP	#CARETN,ANSWER	:SEE IF USER WANTS NO CHANGE
3962	033160	001417			BEQ	18\$	:BRANCH IF SO
3963	033162	022737	000003	002664	CMP	#CNTLC,ANSWER	:SEE IF USER WANTS TO EXIT (^C)
3964	033170	001640			BEQ	5\$	:BRANCH TO PROMPT JUMP IF SO
3965	033172	000741			BR	15\$	:BRANCH BACK - INPUT NOT LEGAL
3966	033174	005704			16\$: TST	R4	:CHECK FOR LEGAL INPUT
3967	033176	001403			BEQ	17\$	:BRANCH IF OK
3968	033200	022704	000001		CMP	#1,R4	:CHECK FOR ILLEGAL INPUT
3969	033204	001334			BNE	15\$	:BRANCH BACK IF ILLEGAL STATE INPUTED
3970	033206	042737	000001	001474	17\$: BIC	#BIT0,\$DDWO	:CLEAR THE BIT TO BE ALTERED
3971	033214	050437	001474		BIS	R4,\$DDWO	:PUT USER INPUT INTO \$DDWO
3972	033220	104401	036035		18\$: TYPE	,DEVPRI	:TYPE: 'DEVICE PRIORITY PRESENT LEVEL = '
3973	033224	013737	001474	002720	MOV	\$DDWO,DDW	:MOVE DEVICE DESCRIPTOR WORD TO DDW
3974	033232	004737	005630		JSR	PC,PNTPRI	:PRINT DEVICE PRIORITY
3975	033236	104401	035276		TYPE	,SPACEC	:TYPE: ' '
3976	033242	012703	000001		MOV	#1,R3	:ONLY INPUT 1 CHARACTER
3977	033246	004737	002722		JSR	PC,READ	:GO READ 1 CHARACTER
3978	033252	005703			TST	R3	:SEE IF NON-NUMERIC WAS THE ONLY INPUT
3979	033254	001415			BEQ	19\$	:BRANCH AROUND NON-NUMERIC TESTS IF NOT
3980	033256	022737	000033	002664	CMP	#ESC,ANSWER	:SEE IF USER WANTS TO GO BACK TO PREVIOUS PROMPT
3981	033264	001704			BEQ	15\$	:BRANCH TO PREVIOUS PROMPT IF SO
3982	033266	022737	000003	002664	CMP	#CNTLC,ANSWER	:SEE IF USER WANTS TO EXIT (^C)
3983	033274	001544			BEQ	26\$	:BRANCH IF EXIT WANTED
3984	033276	022737	000015	002664	CMP	#CARETN,ANSWER	:SEE IF <CR> INPUTED FOR NO CHANGE WANTED
3985	033304	001413			BEQ	20\$	:BRANCH IF NO CHANGE WANTED
3986	033306	000744			BR	18\$	:BRANCH BACK - INPUT NOT LEGAL

Address	Hex	Dec	Label	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op41
---------	-----	-----	-------	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	------

```

4042      .SBTTL BURST DATA LATE CALIBRATION ROUTINE
4043      :*****
4044      :>>>>>>BURST DATA LATE CALIBRATION ROUTINE<<<<<<<
4045      :*****
4046
4047 033612 012737 177777 002670 BDLCR: MOV    #-1,MANSIZE ;MOVE -1 TO MANSIZE
4048 033620 004737 003360      JSR    PC,FIXTBL  ;GO FILL TABLE
4049 033624 104401 035110      TYPE   ,BDLCRM  ;TYPE: 'BURST DATA LATE CALIBRATION'
4050      :                                           ;TYPE: 'ATTACH SCOPE PROBE...'
4051      :                                           ;TO CALIBRATE NEXT BOARD, TYPE ANY CHARACTER'
4052 033630 012737 000001 002544      MOV    #BIT0,DEVMSK ;SET UP BIT MASK TO TEST $DEVN FOR DEVICES
4053 033636 012700 002456      MOV    #VECADR,R0 ;MOVE VECADR TO R0
4054 033642 012701 002416      MOV    #REGADR,R1 ;MOVE REGADR TO R1
4055 033646 005037 001422      CLR    $UNIT ;CLEAR $UNIT
4056 033652 033737 002544 001466 2$: BIT    DEVMSK,$DEVN ;CHECK TO SEE IF DEVICE IS TO BE TESTED
4057 033660 001015      BNE    $S ;BRANCH IF SO
4058 033662 005737 002544      TST    DEVMSK ;SEE IF BIT 15 IS SET
4059 033666 100004      BPL    4$ ;BRANCH TO CONTINUE IF NOT SET
4060 033670 104401 034377      TYPE   ,BCDONE ;TYPE: 'BURST CALIBRATION COMPLETE'
4061 033674 000137 032144      JMP    PROMPT ;JUMP TO PROMPT A NEW COMMAND
4062 033700 022021      4$: CMP    (R0)+,(R1)+ ;INCREMENT THE TWO POINTERS
4063 033702 006337 002544      ASL    DEVMSK ;UPDATE DEVICE MAP TEST MASK
4064 033706 005237 001422      INC    $UNIT ;INCREMENT UNIT NUMBER
4065 033712 000757      BR     2$ ;GO TEST NEXT BIT OF DEVICE MASK
4066 033714 011137 002522 5$: MOV    (R1),CSR ;PUT UUT CSR ADDRESS INTO DEVICE CSR LOCATION
4067 033720 062737 000004 002522 ADD    #4,CSR ;POINT CSR TO CSR ADDRESS
4068 033726 011037 002526      MOV    (R0),DRINV ;PUT UUT VECTOR ADDRESS INTO DEVICE DRINV
4069 033732 104401 035076      TYPE   ,DEVICE ;TYPE: 'DEVICE #'
4070 033736 013746 001422      MOV    $UNIT,-(SP) ;PUT UNIT NUMBER ON STACK FOR TYPEOUT
4071 033742 104405      TYPDS ;GO TYPE THE UNIT NUMBER IN DECIMAL
4072 033744 104401 035407      TYPE   ,UCAL ;TYPE: ' UNDER CALIBRATION'
4073 033750 004737 004036      JSR    PC,CLENUP ;SUBROUTINE TO CLEAR DEVICE REGISTERS & SET CPU PRI TO 7
4074 033754 005077 146542 6$: CLR    @CSR ;CLR CYCLE BIT
4075 033760 012737 000077 002660 MOV    #77,TIME ;MOVE WAIT LOOP COUNTER TO TIME
4076 033766 052777 000400 146526 BIS    #CY,@CSR ;SET CYCLE BIT
4077 033774 005337 002660 7$: DEC    TIME ;SUBTRACT 1 FROM TIME UNTIL ZERO
4078 034000 001375      BNE    7$ ;BRANCH BACK IF NOT ZERO YET
4079 034002 105777 145336      TSTB   @$TKS ;IS A CHARACTER WAITING INDICATING USER WANTS TO GO ON?
4080 034006 100362      BPL    6$ ;BRANCH IF NOT
4081 034010 017737 146622 002660 MOV    @TKB,TIME ;WASTE THE CHARACTER, CLEARING THE CHARACTER FLAG
4082 034016 000730      BR     4$ ;GO ON TO NEXT BOARD

```

4083					.SBTTL	ASCII AND ASCII2 MESSAGES AND LOCATIONS
4084	034020	200	123	124	STKIFL: .ASCII2	<CRLF>/STACK IS FULL - DATA MAY HAVE BEEN LOST/<CRLF><CRLF>
4085	034073	136	131	200	ERCHDR: .ASCII	/^Y/<CRLF>/SUMMATION OF ERRORS SINCE START OR LAST REPORT/
4086	034154	200	200	102	.ASCII2	<CRLF><CRLF>/BOARD # PASS # ERRITL/<CRLF>
4087	034211	136	131	200	NODATA: .ASCII2	/^Y/<CRLF>/NO ERROR TOTALS TO REPORT/<CRLF><CRLF>
4088	034250	040	055	040	ETDEV: .ASCII2	/ - TOTAL ERRORS THIS DEVICE = /
4089	034307	111	114	114	BDNERR: .ASCII2	/ILLEGAL NUMBER (# OTHER THAN 1-16) OR CHARACTER INPUTED/
4090	034377	102	125	122	BCDONE: .ASCII2	/BURST CALIBRATION COMPLETE/
4091	034432	101	104	104	ADRERR: .ASCII2	/ADDRESS INPUTED IS NOT IN THE RANGE 171000 TO 177000/
4092	034517	126	105	103	VECERR: .ASCII2	/VECTOR INPUTED IS NOT IN THE RANGE OF 124 TO 777/
4093	034600	101	104	104	ADLCHR: .ASCII2	/ADDRESS INPUTED HAS OTHER THAN 0 FOR LEAST SIGNIFICANT OCTAL DIGIT/
4094	034703	126	105	103	VCLCHR: .ASCII2	/VECTOR INPUTED SHOULD HAVE ZERO AS LEAST DIGIT/
4095	034762	074	105	123	ESCAPE: .ASCII2	/<ESC>/
4096	034770	200	104	105	MUSTED: .ASCII	<CRLF>/DEVICE ADDRESS AND-OR VECTOR TABLE NOT SET UP - /
4097	035051	115	125	123	.ASCII2	/MUST EDIT FIRST/
4098	035071	040	000		LETNCR: .ASCII2	/ /
4099	035073	136	103	000	CNTRLC: .ASCII2	/^C/
4100	035076	104	105	126	DEVICE: .ASCII2	/DEVICE # /
4101	035110	200	102	125	BDLCRM: .ASCII	<CRLF>/BURST DATA LATE CALIBRATION/
4102	035144	200	101	124	.ASCII	<CRLF>/ATTACH SCOPE PROBE.../
4103	035172	200	124	117	.ASCII2	<CRLF>/TO CALIBRATE NEXT BOARD, TYPE ANY CHARACTER/<CRLF>
4104	035250	040	040	000	SPACES: .ASCII2	/ /
4105	035253	040	040	040	SPACE3: .ASCII2	/ /
4106	035257	040	040	040	SPACE6: .ASCII2	/ /
4107	035266	040	040	040	SPACE7: .ASCII2	/ /
4108	035276	040	040	072	SPACEC: .ASCII2	/ : /
4109	035303	102	000		B: .ASCII2	/B/
4110	035305	127	000		W: .ASCII2	/W/
4111	035307	116	117	000	NO: .ASCII2	/NO/
4112	035312	131	105	123	YES: .ASCII2	/YES/
4113	035316	062	000		TWO: .ASCII2	/2/
4114	035320	116	000		N: .ASCII2	/N/
4115	035322	102	117	101	BOARD: .ASCII2	/BOARD # /
4116	035333	200	200	000	CRLF2: .ASCII2	<CRLF><CRLF>
4117	035336	200	101	114	BCFIN: .ASCII2	<CRLF>/ALL BOARDS CALIBRATED - BEGINNING TEST/<CRLF>
4118	035407	040	125	116	UCAL: .ASCII2	/ UNDER CALIBRATION/<CRLF>
4119	035433	200	116	125	NOBUT: .ASCII2	<CRLF>/NUMBER OF BOARDS UNDER TEST: /
4120	035472	200	200	104	BRVWPC: .ASCII	<CRLF><CRLF>/DIAGNOSTIC HAS DETERMINED THE FOLLOWING ABOUT THE/<CRLF>
4121	035556	104	122	061	.ASCII	/DR11-W(S) IT HAS FOUND. USER *MUST* DETERMINE ACCURACY/<CRLF><CRLF>
4122	035647	040	040	040	.ASCII2	/ BOARD# REGADR VECADR W-B P-LEV 2-N CY CABLE/<CRLF>
4123	035747	200	062	040	TORNCB: .ASCII2	<CRLF>/2 OR N CYCLE BURST (2 CY=0, N CY=1) PRESENT STATE = /
4124	036035	200	104	105	DEVPR1: .ASCII2	<CRLF>/DEVICE PRIORITY PRESENT LEVEL = /
4125	036077	200	104	122	DR1WOB: .ASCII2	<CRLF>/DR11-W OR B (W=0, B=1) CURRENT STATE = /
4126	036150	200	123	124	SVADRS: .ASCII2	<CRLF>/STARTING VECTOR ADDRESS: /
4127	036204	200	123	124	SDADRS: .ASCII2	<CRLF>/STARTING DEVICE ADDRESS: /
4128	036237	040	124	105	TSTCOM: .ASCII2	/ TESTING COMPLETE/
4129	036261	200	104	117	DOCTS: .ASCII2	<CRLF>/DO CABLE TESTS (NO=0, YES=1) PRESENT STATE = /
4130	036340	200	200	043	HEADER: .ASCII	<CRLF><CRLF>/# OF START 2-N CABLE/
4131	036434	200	102	117	.ASCII2	<CRLF>/BOARDS REGADR VECADR W-B P-LEV CYCLE TESTS/<CRLF>
4132	036531	200	200	115	MBDIAL: .ASCII2	<CRLF><CRLF>/MULTIPLE BOARD DIALOGUE/<CRLF>
4133	036564	200	105	116	ECLR: .ASCII2	<CRLF>/ENTER COMMAND ([E]DIT, [L]IST, [B]URST CALIBRATION, [R]UN): /
4134	036663	124	110	105	NOMORE: .ASCII	/THERE ARE STILL MORE ERRORS, BUT WILL NOT BE PRINTED./<CRLF>
4135	036751	105	122	122	.ASCII2	/ERRORS WILL STILL BE COUNTED AND PRINTED AT THE EOP./<CRLF>
4136	037037	200	116	117	NODVPR: .ASCII	<CRLF>/NO DEVICES RECOGNIZED - DIAGNOSTIC CANNOT BE RUN/<CRLF>
4137	037121	122	105	123	.ASCII2	/RESTART AT 204 IF A DEVICE IS PRESENT/<CRLF>
4138	037170	200	050	136	M1: .ASCII	<CRLF>/(^X) INHIBITS EOP'S, (^Y) FOR ERROR SUMMARY/<CRLF>
4139	037245	125	116	111	.ASCII2	/UNIBUS HANG? RESTART AT ADDRESS /

4140	037307	200	200	103	M1A:	.ASCII2	<CRLF><CRLF>/CZDRLCO DR11 GEN NPR INTFC LOGIC TEST/<CRLF>	
4141	037360	104	105	126	BARADR:	.ASCII2	/DEVICE ADDRESS - /	
4142	037402	054	040	124	TSNUMB:	.ASCII2	/, TEST NUMBER - /	
4143	037423	054	040	120	PASNUM:	.ASCII2	/, PASS NUMBER - /	
4144						.EVEN		
4145	037444	000000			.SAV:	.WORD	0	
4146	037446					.BLKW	400	
4147	040446	000000			BUFF:	.WORD	0	
4148	040450	040450			XINBUF:			
4149	040452					.BLKW	400	
4150	041452	041452			XCHKBU:			
4151	041454					.BLKW	400	
4152	042454	042456			CAPNTR:	.WORD	CAPSTK	;LOCATION TO HOLD POINTER FOR CAPTURE STACK
4153	042456				CAPSTK:	.BLKW	600.	;LOCATIONS TO STORE UP TO 150 DECIMAL PASSES AND THEIR ERROR
4154	044736	000000			ENDSTK:	.WORD	0	;FLAG SIGNALING END OF THE STACK

4155				.SBTTL	ERROR MESSAGES
4156 044740	124	105	123	EM1:	.ASCIZ /TEST SEQUENCING ERROR/
4157 044766	103	101	116	EM2:	.ASCIZ /CANNOT ACCESS DR11 REGISTER/
4158 045022	104	122	061	EM3:	.ASCIZ /DR11-B OR W MODE INCORRECT (0=B, 1=W)/
4159 045070	111	116	111	EM4:	.ASCIZ /INIT FAILED TO CLEAR WCR/
4160 045121	111	116	111	EM5:	.ASCIZ /INIT FAILED TO CLEAR BAR/
4161 045152	111	116	111	EM6:	.ASCIZ /INIT FAILED TO CLEAR BDR/
4162 045203	111	116	111	EM7:	.ASCIZ /INIT FAILED TO CLEAR ALL CSR R-W BITS/
4163 045251	122	105	123	EM10:	.ASCIZ /RESET FAILED TO CLEAR WCR/
4164 045303	101	124	124	EM11:	.ASCIZ /ATTEMPT TO SET ALL WCR BITS FAILED/
4165 045346	122	105	123	EM12:	.ASCIZ /RESET FAILED TO CLEAR BAR/
4166 045400	101	124	124	EM13:	.ASCIZ /ATTEMPT TO SET ALL BAR BITS TO 1 FAILED/
4167 045450	103	123	122	EM14:	.ASCIZ /CSR BIT TEST FAILED (FATAL - DIAGNOSTIC NOT CONTINUED)/
4168 045537	103	123	122	EM15:	.ASCIZ /CSR BIT TEST FAILED/
4169 045563	105	111	122	EM16:	.ASCIZ /EIR BIT TEST FAILED/
4170 045607	122	105	101	EM17:	.ASCIZ /READY AND MAINTENANCE ARE NOT THE ONLY BITS SET IN CSR/
4171 045676	101	124	124	EM20:	.ASCIZ /ATTN AND ERROR FAILED TO SET PROPERLY/
4172 045744	101	124	124	EM21:	.ASCIZ /ATTN AND ERROR FAILED TO CLEAR PROPERLY/
4173 046014	105	122	122	EM22:	.ASCIZ /ERROR BIT SHOULD HAVE BEEN CLEAR/
4174 046055	102	111	124	EM23:	.ASCIZ /BIT PATTERN NOT LOADED PROPERLY IN WCR/
4175 046124	122	105	101	EM24:	.ASCIZ /READY OF CSR WAS NOT SET/
4176 046155	102	111	124	EM25:	.ASCIZ /BIT 0 OF THE BAR WAS SET/
4177 046206	102	111	124	EM26:	.ASCIZ /BIT PATTERN TEST FAILED IN BAR/
4178 046245	127	103	122	EM27:	.ASCIZ /WCR DATA PATTERN NOT CORRECT/
4179 046302	106	125	116	FM30:	.ASCIZ /FUNCTION BIT(S) ARE NOT CLEAR/
4180 046340	104	123	124	EM31:	.ASCIZ /DSTAT A, B OR C ARE NOT AS EXPECTED/
4181 046404	102	104	122	EM32:	.ASCIZ /BDR IS NOT CLEAR/
4182 046425	101	114	114	EM33:	.ASCIZ /ALL BDR BITS ARE NOT SET/
4183 046456	102	104	122	EM34:	.ASCIZ /BDR FAILS TO HOLD A BIT PATTERN/
4184 046516	102	104	122	EM35:	.ASCIZ /BDR SHOULD NOT HAVE BEEN LOADED WITH NEW PATTERN/
4185 046577	102	104	122	EM36:	.ASCIZ /BDR PATTERN NOT CORRECT/
4186 046627	122	105	101	EM37:	.ASCIZ /READY IS NOT THE ONLY BIT SET/
4187 046665	122	105	101	EM40:	.ASCIZ /READY SHOULD NOT BE SET/
4188 046715	122	105	101	EM41:	.ASCIZ /READY WAS CLEARED BUT NEVER SET AGAIN/
4189 046763	122	105	101	EM42:	.ASCIZ /READY CANNOT BE SET BY INIT/
4190 047017	104	122	061	EM43:	.ASCIZ /DR11 FAILED TO INTERRUPT/
4191 047050	104	122	061	EM44:	.ASCIZ /DR11 INTERRUPTED, BUT IT SHOULDN'T HAVE/
4192 047120	105	122	122	EM45:	.ASCIZ /ERROR BIT SHOULD NOT BE CLEAR/
4193 047156	106	125	116	EM46:	.ASCIZ /FUNCTION BITS DIDN'T INCREMENT IN MAINT MODE/
4194 047233	103	123	122	EM47:	.ASCIZ /CSR IS WRONG/
4195 047250	124	122	101	EM50:	.ASCIZ /TRANSFERS SHOULD HAVE BEEN INHIBITED/
4196 047315	104	122	061	EM51:	.ASCIZ /DR11 SHOULD NOT HAVE INTERRUPTED A SECOND TIME/
4197 047374	105	130	120	EM52:	.ASCIZ /EXPECTED INTERRUPT DID NOT OCCUR/
4198 047435	127	103	122	EM53:	.ASCIZ /WCR NOT EQUAL TO ZERO/
4199 047463	102	101	122	EM54:	.ASCIZ /BAR IS WRONG/
4200 047500	102	101	104	EM55:	.ASCIZ /BAD DATA IN BDR/
4201 047520	104	101	124	EM56:	.ASCIZ /DATA NOT TRANSFERED CORRECTLY/
4202 047556	102	125	106	EM57:	.ASCIZ /BUFFER DATA NOT CORRECT/
4203 047606	124	117	117	EM60:	.ASCIZ /TOO MANY WORDS WERE TRANSFERED/
4204 047645	125	116	105	EM61:	.ASCIZ /UNEXPECTED TRAP OR INTERRUPT TO TRAP ADDRESS BELOW/
4205 047730	103	123	122	EM62:	.ASCIZ /CSR AND-OR WCR AND-OR BAR ARE INCORRECT/
4206 047777	104	122	061	EM63:	.ASCIZ /DR11 INTERRUPTED AT WRONG LEVEL/
4207 050037	062	055	116	EM65:	.ASCIZ /2-N CYCLE BURST SWITCH IN WRONG POSITION/
4208 050110	115	125	114	EM66:	.ASCIZ /MULTICYCLE BIT IN THE EIR IS WRONG/
4209 050153	103	123	122	EM202:	.ASCIZ /CSR PATTERN NOT CORRECT/
4210 050203	102	104	122	EM211:	.ASCIZ /BDR AND-OR WCR AND-OR BAR ARE INCORRECT/

```
4211
4212 050252      105      130      120 DH1: .SBTTL DATA HEADERS
4213 050270      124      105      123 .ASCII /EXPCTD RECVD/<CRLF>
4214 050307      124      105      123 .ASCIIZ /TEST # TEST #/
4215 050350      124      105      123 DH2: .ASCIIZ /TEST # ERR PC ABRTPC REGISTER/
4216 050417      124      105      123 DH3: .ASCIIZ /TEST # ERR PC EXPMOD ACTMOD CSRADR/
4217 050463      124      105      123 DH4: .ASCIIZ /TEST # ERR PC WCRADR WCRCONTENTS/
4218 050532      124      105      123 DH5: .ASCIIZ /TEST # ERR PC BARADR BAREXP BARRCV/
4219 050576      124      105      123 DH6: .ASCIIZ /TEST # ERR PC BDRADR BDRCONTENTS/
4220 050652      040      040      040 DH7: .ASCIIZ /TEST # ERR PC CSRADR CSREXP CSRCONTENTS/
4221 050701      124      105      123 DH14: .ASCII / BIT(S)/<CRLF>
4222 050765      040      040      040 .ASCIIZ /TEST # ERR PC TESTED CSRADR CSREXP CSRCONTENTS/
4223 051014      124      105      123 DH16: .ASCII / BIT(S)/<CRLF>
4224 051100      124      105      123 .ASCIIZ /TEST # ERR PC TESTED EIRADR EIREXP EIRCONTENTS/
4225 051154      124      105      123 DH17: .ASCIIZ /TEST # ERR PC CSRADR CSREXP CSRCONTENTS/
4226 051230      124      105      123 DH23: .ASCIIZ /TEST # ERR PC WCRADR WCREXP WCRCONTENTS/
4227 051304      124      105      123 DH26: .ASCIIZ /TEST # ERR PC BARADR BAREXP BARCONTENTS/
4228 051360      124      105      123 DH34: .ASCIIZ /TEST # ERR PC BDRADR BDREXP BDRCONTENTS/
4229 051424      124      105      123 DH43: .ASCIIZ /TEST # ERR PC CSRADR CSRCONTENTS/
4230 051523      124      105      123 DH50: .ASCIIZ /TEST # ERR PC WCRADR WCREXP WCRRCV BARADR BAREXP BARRCV/
4231 051602      040      040      040 DH56: .ASCIIZ /TEST # ERR PC NPR1AD NPR1EX NPR1RC CSRADR/
4232 051660      124      105      123 DH57: .ASCII / CHECK CHECK INPUT INPUT/<CRLF>
4233 051747      040      040      040 .ASCIIZ /TEST # ERR PC BUFADR BUFDAT BUFADR BUFDAT CSRADR/
4234 051776      124      105      123 DH60: .ASCII / DIDNOT/<CRLF>
4235 052045      124      105      123 .ASCIIZ /TEST # ERR PC EXPECT ADDRESS CSRADR/
4236 052116      124      105      123 DH61: .ASCIIZ /TEST # ERR PC WCRADR OLDPC TRAP ADR/
4237 052176      103      123      122 DH62: .ASCIIZ /TEST # ERR PC WCRADR WCREXP WCRRCV CSREXP /
4238 052225      124      105      123 .ASCIIZ /CSRRCV BAREXP BARRCV/
4239 052274      124      105      123 DH63: .ASCIIZ /TEST # ERR PC EXPLVL RCVLVL CSRADR/
4240 052333      124      105      123 DH64: .ASCIIZ /TEST # ERR PC EXPLVL CSRADR/
4241 052402      124      105      123 DH65: .ASCIIZ /TEST # ERR PC CSRADR EIREXP EIRRCV/
4242 052451      124      105      123 DH66: .ASCIIZ /TEST # ERR PC PATERN CSRADR CSRRCV/
4243 052530      124      105      123 DH202: .ASCIIZ /TEST # ERR PC CSRADR PATLDD CSREXP CSRRCV/
4244 052614      124      105      123 DH203: .ASCIIZ /TEST # ERR PC CSRADR PATERN CSREXP CSRCONTENTS/
4245 052670      124      105      123 DH207: .ASCIIZ /TEST # ERR PC PATERN CSRADR CSRCONTENTS/
4246 052734      124      105      123 DH210: .ASCIIZ /TEST # ERR PC WCRADR WCRCONTENTS/
4247 .EVEN DH211: .ASCIIZ /TEST # ERR PC WCRADR WCREXP WCRRCV BDREXP BDRRCV BAREXP BARRCV/
```


4248						.SBTTL	DATA TABLES
4249	053044	001362	001414	000000	DT1:	.WORD	\$TMP1,\$TESTN,0
4250	053052	001414	001316	002672	DT2:	.WORD	\$TESTN,\$ERRPC,OLDPC1,DREG,0
4251	053064	001414	001316	001362	DT3:	.WORD	\$TESTN,\$ERRPC,\$TMP1,BORW,CSR,0
4252	053100	001414	001316	002516	DT4:	.WORD	\$TESTN,\$ERRPC,WCR,RWCR,0
4253	053112	001414	001316	002520	DT5:	.WORD	\$TESTN,\$ERRPC,BAR,EBAR,RBAR,0
4254	053126	001414	001316	002524	DT6:	.WORD	\$TESTN,\$ERRPC,BDR,RBDR,0
4255	053140	001414	001316	002522	DT7:	.WORD	\$TESTN,\$ERRPC,CSR,ECSR,RCSR,0
4256	053154	001414	001316	002536	DT14:	.WORD	\$TESTN,\$ERRPC,BUT,CSR,ECSR,RCSR,0
4257	053172	001414	001316	002536	DT16:	.WORD	\$TESTN,\$ERRPC,BUT,CSR,EEIR,REIR,0
4258	053210	001414	001316	002522	DT17:	.WORD	\$TESTN,\$ERRPC,CSR,ECSR,RCSR,0
4259	053224	001414	001316	002516	DT23:	.WORD	\$TESTN,\$ERRPC,WCR,EWCR,RWCR,0
4260	053240	001414	001316	002520	DT26:	.WORD	\$TESTN,\$ERRPC,BAR,EBAR,RBAR,0
4261	053254	001414	001316	002524	DT34:	.WORD	\$TESTN,\$ERRPC,BDR,EBDR,RBDR,0
4262	053270	001414	001316	002522	DT43:	.WORD	\$TESTN,\$ERRPC,CSR,RCSR,0
4263	053302	001414	001316	002516	DT50:	.WORD	\$TESTN,\$ERRPC,WCR,EWCR,RWCR,BAR,EBAR,RBAR,0
4264	053324	001414	001316	002606	DT56:	.WORD	\$TESTN,\$ERRPC,ANPR1,ENPR1,NPR1,CSR,0
4265	053342	001414	001316	001370	DT57:	.WORD	\$TESTN,\$ERRPC,\$TMP4,\$TMP2,\$TMP5,\$TMP3,CSR,0
4266	053362	001414	001316	001364	DT60:	.WORD	\$TESTN,\$ERRPC,\$TMP2,\$TMP3,CSR,0
4267	053376	001414	001316	002516	DT61:	.WORD	\$TESTN,\$ERRPC,WCR,OLDPC2,BDVECT,0
4268	053412	001414	001316	002516	DT62:	.WORD	\$TESTN,\$ERRPC,WCR,EWCR,RWCR,ECSR,RCSR,EBAR,RBAR,0
4269	053436	001414	001316	002552	DT63:	.WORD	\$TESTN,\$ERRPC,DRLEV,LEVEL,CSR,0
4270	053452	001414	001316	001362	DT64:	.WORD	\$TESTN,\$ERRPC,\$TMP1,CSR,0
4271	053464	001414	001316	002522	DT65:	.WORD	\$TESTN,\$ERRPC,CSR,EEIR,REIR,0
4272	053500	001414	001316	002604	DT66:	.WORD	\$TESTN,\$ERRPC,ENPR1,CSR,RCSR,0
4273	053514	001414	001316	002522	DT202:	.WORD	\$TESTN,\$ERRPC,CSR,BUT,ECSR,RCSR,0
4274	053532	001414	001316	002522	DT203:	.WORD	\$TESTN,\$ERRPC,CSR,\$TMP0,ECSR,RCSR,0
4275	053550	001414	001316	001362	DT207:	.WORD	\$TESTN,\$ERRPC,\$TMP1,CSR,RCSR,0
4276	053564	001414	001316	002516	DT210:	.WORD	\$TESTN,\$ERRPC,WCR,RWCR,0
4277	053576	001414	001316	002516	DT211:	.WORD	\$TESTN,\$ERRPC,WCR,EWCR,RWCR,EBDR,RBDR,EBAR,RBAR,0

4278
4279 053622 104401 037360
4280 053626 013746 002516
4281 053632 104402
4282 053634 104401 037402
4283 053640 013746 001414
4284 053644 104403
4285 053646 002 000
4286 053650 104401 037423
4287 053654 013746 002716
4288 053660 013746 001416
4289 053664 104406
4290 053666 104401 001405
4291 053672 000000
4292 053674 000137 010266
4293 053700 000000
4294
4295 000001

```
.SBTTL BUS HANG ROUTINE
UBHANG. TYPE ,BARADR ;TYPE: 'DEVICE ADDRESS - '
MOV WCR,-(SP) ;PUT DEVICE ADDRESS ON STACK
TYPDC ;GO TYPE IT IN OCTAL
TYPE ,TSNUMB ;TYPE: ', TEST NUMBER - '
MOV $TESTN,-(SP) ;PUT TEST NUMBER ON STACK
TYPDS ;GO TYPE IT IN OCTAL
.BYTE 2,0 ;TYPE 2 DIGITS, LEADING ZEROS SUPRESSED
TYPE ,PASNUM ;TYPE: ', PASS NUMBER - '
MOV $PASS2,-(SP) ;MOVE OVERFLOW NUMBER TO THE STACK
MOV $PASS,-(SP) ;PUT PASS NUMBER ON STACK
TYPDE ;GO TYPE IT IN EXTENDED DECIMAL
TYPE ,$CRLF ;TYPE A <CRLF>
HALT ;WHOA - YOU GOTTA SERIOUS PROBLEMA, BUDDY!
JMP START1 ;JUMP TO RESTART PROGRAM
NOCARE: .WORD 0 ;LOCATION FOR USE WHENEVER CYCLE BIT OF CSR IS USED. THIS
;SHOULD *ALWAYS* BE THE LAST WORD LOCATION IN THIS DIAGNOSTIC
.END
```

ABASE = 172410	AVECT1= 000300	CARETN= 000015	DATCHK 003716	DT1 053044
ACDW1 = 000000	AVECT2= 000000	CAT = 157777	DATCH1 003736	DT14 053154
ACDW2 = 000000	B 035303	CATCH 005536	DATCH2 004014	DT16 053172
ACPUOP= 000000	BAR 002520	CBIT0 = 177776	DATOCK 004106	DT17 053210
ADDR 002646	BARADR 037360	CBIT1 = 177775	DATOC1 004126	DT2 053052
ADDW0 = 000000	BCDONE 034377	CBIT10= 175777	DATOC2 004170	DT202 053514
ADDW1 = 000000	BCFIN 035336	CBIT11= 173777	DBC = 003000	DT203 053532
ADDW10= 000000	BDFAIL 002666	CBIT12= 167777	DDISP = 177570	DT207 053550
ADDW11= 000000	BDLCR 033612	CBIT13= 157777	DDW 002720	DT210 053564
ADDW12= 000000	BDLCRM 035110	CBIT14= 137777	DEVADS 004344	DT211 053576
ADDW13= 000000	BDNERR 034307	CBIT15= 077777	DEVICE 035076	DT23 053224
ADDW14= 000000	BDR 002524	CBIT2 = 177773	DEVMSK 002544	DT26 053240
ADDW15= 000000	BDVECT 002542	CBIT3 = 177767	DEVPRI 036035	DT3 053064
ADDW2 = 000000	BEGIN 010466	CBIT4 = 177757	DH1 050252	DT34 053254
ADDW3 = 000000	BEGIN1 011006	CBIT5 = 177737	DH14 050652	DT4 053100
ADDW4 = 000000	BITTST 002710	CBIT6 = 177677	DH16 050765	DT43 053270
ADDW5 = 000000	BIT0 = 000001	CBIT7 = 177577	DH17 051100	DT5 053112
ADDW6 = 000000	BIT00 = 000001	CBIT8 = 177377	DH2 050307	DT50 053302
ADDW7 = 000000	BIT01 = 000002	CBIT9 = 176777	DH202 052451	DT56 053324
ADDW8 = 000000	BIT02 = 000004	CB1513= 057777	DH203 052530	DT57 053342
ADDW9 = 000000	BIT03 = 000010	CCY = 177377	DH207 052614	DT6 053126
ADEVCT= 000000	BIT04 = 000020	CDAB = 171777	DH210 052670	DT60 053362
ADEVN = 000000	BIT05 = 000040	CDAC = 172777	DH211 052734	DT61 053376
ADLCHR 034600	BIT06 = 000100	CDBC = 174777	DH23 051154	DT62 053412
ADRERR 034432	BIT07 = 000200	CDSA = 173777	DH26 051230	DT63 053436
AENV = 000000	BIT08 = 000400	CDSB = 175777	DH3 050350	DT64 053452
AENVN = 000000	BIT09 = 001000	CDSC = 176777	DH34 051304	DT65 053464
AFATAL= 000000	BIT1 = 000002	CDST = 170777	DH4 050417	DT66 053500
AMADR1= 000000	BIT10 = 002000	CEIR = 077777	DH43 051360	DT7 053140
AMADR2= 000000	BIT11 = 004000	CER = 077777	DH5 050463	EBAR 002600
AMADR3= 000000	BIT12 = 010000	CFNC = 177761	DH50 051424	EBDR 002576
AMADR4= 000000	BIT13 = 020000	CF1 = 177775	DH56 051523	ECELR 036564
AMAMS1= 000000	BIT14 = 040000	CF2 = 177773	DH57 051602	ECSR 002572
AMAMS2= 000000	BIT15 = 100000	CF3 = 177767	DH6 050532	EDIT 032472
AMAMS3= 000000	BIT2 = 000004	CGO = 177776	DH60 051747	EEIR 002574
AMAMS4= 000000	BIT3 = 000010	CHARCT 030707	DH61 052045	EIR = 100000
AMSGAD= 000000	BIT4 = 000020	CHKBFF 003520	DH62 052116	EMTVEC= 000030
AMSGLG= 000000	BIT5 = 000040	CHKBUFF 002620	DH63 052225	EM1 044740
AMSGTY= 000000	BIT6 = 000100	CHKCAB 004060	DH64 052274	EM10 045251
AMTYP1= 000000	BIT7 = 000200	CHK4DR 005104	DH65 052333	EM11 045303
AMTYP2= 000000	BIT8 = 000400	CIE = 177677	DH66 052402	EM12 045346
AMTYP3= 000000	BIT9 = 001000	CKSWR = 104410	DH7 050576	EM13 045400
AMTYP4= 000000	BOARD 035322	CLENUP 004036	DIOMEM 002614	EM14 045450
ANPR1 002606	BORW 002610	CMA = 167777	DISPLA 001342	EM15 045557
ANSWER 002664	BPINIT 004374	CNTLC = 000003	DISPRE 000174	EM16 045563
APASS = 000000	BPT = 000003	CNTRLC 035073	DOCTS 036261	EM17 045607
APRIOR= 000000	BPTINT 004436	CNX = 137777	DREG 002550	EM2 044766
APTCU= 000040	BPTVCT 000014	CR = 000015	DRGET 004452	EM20 045676
APTENV= 000001	BPTVEC= 000014	CRLF = 000200	DRINV 002526	EM202 050153
APTSIZ= 000200	BRVWPC 035472	CRLF2 035333	DRLEV 002552	EM21 045744
APTSPD= 000100	BRWAIT 002626	CRY = 177577	DRVS 002530	EM211 050203
ASIZE 005274	BUFF 040446	CSR 002522	DR1WOB 036077	EM22 046014
ASWREG= 000000	BUFLEN 002622	CX6 = 177757	DSA = 004000	EM23 046055
AT = 020000	BUSERR= 000004	CX7 = 177737	DSB = 002000	EM24 046124
ATESTN= 000000	BUT 002536	CY = 000400	DSC = 001000	EM25 046155
AUNIT = 000000	CAPNTR 042454	DAB = 006000	DST = 007000	EM26 046206
AUSWR = 000000	CAPSTK 042456	DAC = 005000	DSWR = 177570	EM27 046245

EM3	045022	F1	= 000002	NXTTST	002554	STACK	= 001300	TST12	014270
EM30	046302	F2	= 000004	N2	= 000400	STAGIN	000210	TST13	014454
EM31	046340	F3	= 000010	OFL	002702	START	010272	TST14	014640
EM32	046404	GO	= 000001	OLDPC1	002672	START1	010266	TST15	015030
EM33	046425	GOAGIN	025002	OLDPC2	002676	STKIFL	034020	TST16	015262
EM34	046456	GTSWR	= 104407	OLDPS1	002674	STKLMT	= 177774	TST17	015520
EM35	046516	HAKTPM	030326	OLDPS2	002700	SVADRS	036150	TST2	011442
EM36	046577	HEADER	036340	PASCNT	002556	SWR	001340	TST20	016116
EM37	046627	HT	= 000011	PASNUM	037423	SWREG	000176	TST21	016522
EM4	045070	IE	= 000100	PATRNS	006250	SW0	= 000001	TST22	017036
EM40	046665	INBUF	002616	PIRQ	= 177772	SW00	= 000001	TST23	017334
EM41	046715	INBUF1	002656	PIRQVE	= 000240	SW01	= 000002	TST24	020072
EM42	046763	INOUT	021344	PNTPRI	005630	SW02	= 000004	TST25	020374
EM43	047017	INTA	003546	PROMPT	032144	SW03	= 000010	TST26	020664
EM44	047050	IOTVEC	= 000020	PRO	= 000000	SW04	= 000020	TST27	021142
EM45	047120	KDPA2	= 172364	PR1	= 000040	SW05	= 000040	TST3	011572
EM46	047156	KDPDR2	= 172324	PR2	= 000100	SW06	= 000100	TST30	021360
EM47	047233	KIPAR2	= 172344	PR3	= 000140	SW07	= 000200	TST31	021646
EM5	045121	KIPDR2	= 172304	PR4	= 000200	SW08	= 000400	TST32	021776
EM50	047250	LENCHK	002624	PR5	= 000240	SW09	= 001000	TST33	022356
EM51	047315	LETNCR	035071	PR6	= 000300	SW1	= 000002	TST34	022662
EM52	047374	LEVEL	002540	PR7	= 000340	SW10	= 002000	TST35	023244
EM53	047435	LEVELS	005264	PS	= 177776	SW11	= 004000	TST36	023542
EM54	047463	LEVEL3	= 000140	PSTATE	005606	SW12	= 010000	TST37	024056
EM55	047500	LEVEL4	= 000200	PSW	= 177776	SW13	= 020000	TST4	012124
EM56	047520	LEVEL5	= 000240	PTCAPT	003170	SW14	= 040000	TST40	= 024360
EM57	047556	LEVEL6	= 000300	PWRVEC	= 000024	SW15	= 100000	TST5	012730
EM6	045152	LEVEL7	= 000340	QTYBRD	002414	SW2	= 000004	TST6	013124
EM60	047606	LF	= 000012	RA	= 002416	SW3	= 000010	TST7	013472
EM61	047645	LODBUF	003472	RBAR	002566	SW4	= 000020	TWO	035316
EM62	047730	LOOP	002662	RBDR	002564	SW5	= 000040	TYP CNF	004630
EM63	047777	LRGSTC	002704	RCSR	002560	SW6	= 000100	TYPDE	= 104406
EM65	050037	MA	= 010000	RDCHR	= 104411	SW7	= 000200	TYPDS	= 104405
EM66	050110	MAICLR	006266	RDDEC	= 104414	SW8	= 000400	TYPE	= 104401
EM7	045203	MAISET	007266	RDLIN	= 104412	SW9	= 001000	TYPOC	= 104402
ENDEV	024360	MANSIZ	002670	RDOCT	= 104413	TABINX	002546	TYPON	= 104404
ENDSTK	044736	MBD	032130	RDYCHK	002632	TBITVE	= 000014	TYPOS	= 104403
ENPR1	002604	MBDIAL	036531	READ	002722	TESLR	030710	UBHANG	053622
EOPLOC	002706	MEMGMT	002712	REGADR	002416	TIME	002660	UCAL	035407
EOPRSM	030636	MEPITH	031524	REINIT	011206	TKB	002636	VA	= 002456
ER	= 100000	MESSAG	002650	REIR	002562	TKS	002634	VCLCHR	034703
ERCAPT	003126	MMPS	= 000252	RESVEC	= 000010	TKVEC	= 000060	VCTADS	005502
ERCHDR	034073	MMRO	= 177572	RSTRT	025022	TMOPSW	= 000006	VECADR	002456
ERRCHK	004254	MMVECT	= 000250	RWCR	002570	TORNCB	035747	VECERR	034517
ERRCNT	002714	MSG	002644	RY	= 000200	TOVECT	= 000004	W	035305
ERROR	= 104000	MUSTED	034770	R6	= 0000006	TPB	002642	WCLEN	002630
ERRVEC	= 000004	M1	037170	R7	= 0000007	TPS	002640	WCR	002516
ER200	002274	M1A	037307	SCOPE	= 000004	TPVEC	= 000064	XCHKBU	041452
ESC	= 000033	N	035320	SDADRS	036204	TRAPVE	= 000034	XINBUF	040450
ESCAPE	034762	NO	035307	SDRINV	002532	TRTVEC	= 000014	X6	= 000020
ETDEV	034250	NOBUT	035433	SDRVS	002534	TSNUMB	037402	X7	= 000040
EWCR	002602	NOCARE	053700	SETTUP	005660	TSTCOM	036237	YES	035312
EXPAND	026154	NODATA	034211	SPACEC	035276	TSTDEV	011020	\$APTHD	001000
FIXTBL	003360	NODVPR	037037	SPACE3	035250	TSTM	006134	\$ATYC	031552
FLAG	002652	NOMORE	036663	SPACE6	035257	TST1	011244	\$ATY1	031526
FNC	= 000016	NPR1	002612	SPACE7	035266	TST10	013650	\$ATY3	031534
FNCNT	002654	NX	= 040000			TST11	014104	\$ATY4	031544

\$AUTOB	001334	\$DEVCT	001420	\$ICNT	001304	\$OVER	030312	\$TMP6	001374
\$BASE	001464	\$DEVN	001466	\$INTAG	001335	\$PASS	001416	\$TN	= 000040
\$BDADR	001322	\$DOAGN	024760	\$ITEMB	001314	\$PASS2	002716	\$TPB	001352
\$BDDAT	001326	\$DTBL	026132	\$LF	001406	\$PASTM	001006	\$TPFLG	001357
\$BELL	001400	\$ENDAD	024750	\$LFLG	031771	\$POWER	032062	\$TPS	001350
\$CDW1	001470	\$ENDCT	024566	\$LPADR	001306	\$PWRDN	031774	\$TRAP	027550
\$CDW2	001472	\$ENDMG	024767	\$LPERR	001310	\$PWRUP	032012	\$TRAP2	027572
\$CHARC	025374	\$ENULL	024764	\$MADR1	001442	\$QUES	001404	\$TRP	= 000015
\$CKSWR	026422	\$ENV	001430	\$MADR2	001446	\$RDCHR	026740	\$TRPAD	027604
\$CMTAG	001300	\$ENVN	001431	\$MADR3	001452	\$RDDEC	027232	\$TSTM	001004
\$CM3	= 000000	\$EOP	024532	\$MADR4	001456	\$RDLIN	027060	\$TSTNM	001302
\$CM4	= 000007	\$EOPCT	024560	\$MAIL	001410	\$RDOCT	027410	\$TTYIN	027166
\$CNTLG	027202	\$ERFLG	001303	\$MAMS1	001440	\$RDSZ	= 000007	\$TYPD	025642
\$CNTLU	027175	\$ERMAX	001315	\$MAMS2	001444	\$RTNAD	024762	\$TYPDE	025626
\$CPUOP	001436	\$ERROR	031050	\$MAMS3	001450	\$SCOPE	027636	\$TYPDS	025636
\$CRLF	001405	\$ERRPC	001316	\$MAMS4	001454	\$SETUP	= 000137	\$TYPE	025032
\$DBLK	026142	\$ERRTB	001534	\$MBADR	001002	\$STUP	= 177777	\$TYPEC	025244
\$DDW0	001474	\$ERRTY	031250	\$MFLG	031770	\$SVLAD	030256	\$TYPEX	025376
\$DDW1	001476	\$ERTTL	001312	\$MNEW	027220	\$SVPC	= 001000	\$TYPOC	025424
\$DDW10	001520	\$ESCAP	001376	\$MSGAD	001424	\$SWR	= 163400	\$TYPON	025440
\$DDW11	001522	\$ETABL	001430	\$MSGLG	001426	\$SWREG	001432	\$TYPDS	025400
\$DDW12	001524	\$ETEND	001534	\$MSGTY	001410	\$SWRMK	= 000200	\$UNIT	001422
\$DDW13	001526	\$FATAL	001412	\$MSWR	027207	\$SW08T	030752	\$UNITM	001010
\$DDW14	001530	\$FFLG	031772	\$MTYP1	001441	\$TESTN	001414	\$USWR	001434
\$DDW15	001532	\$FILLC	001356	\$MTYP2	001445	\$TKB	001346	\$VECT1	001460
\$DDW2	001500	\$FILLS	001355	\$MTYP3	001451	\$TKS	001344	\$VECT2	001462
\$DDW3	001502	\$GDADR	001320	\$MTYP4	001455	\$TMP0	001360	\$XTSTR	030120
\$DDW4	001504	\$GDDAT	001324	\$NULL	001354	\$TMP1	001362	\$GET4	= 000000
\$DDW5	001506	\$GET42	024740	\$NUMS	026410	\$TMP2	001364	\$SW08	= 000040
\$DDW6	001510	\$GTSWR	026522	\$NWTST	= 000001	\$TMP3	001366	\$OFILL	025623
\$DDW7	001512	\$HD	= 000000	\$OCNT	025622	\$TMP4	001370	\$.SAV	037444
\$DDW8	001514	\$HIBTS	001000	\$OMODE	025624	\$TMP5	001372	\$.X	= 001000
\$DDW9	001516	\$HIOCT	027546						

. ABS. 053702 000
000000 001
ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 56272 WORDS (220 PAGES)
DYNAMIC MEMORY: 20346 WORDS (78 PAGES)
ELAPSED TIME: 00:12:58
CZDRLC.BIN,CZDRLC.SEQ/-SP/NL:TOC=CZDRLC.MLB/ML,CZDRLC.P11