

NC11-A

DIAGNOSTIC
MD-11-DZNCA-C

EP DZNCA-C-DL-A

NOV 1976

COPYRIGHT 1976

digital

FICHE 1 OF 1

MADE IN USA

Frame 1: Diagnostic title and initial parameters.	Frame 2: Fuel system status table.	Frame 3: Fuel system status table.	Frame 4: Fuel system status table.	Frame 5: Fuel system status table.	Frame 6: Fuel system status table.
Frame 7: Fuel system status table.	Frame 8: Fuel system status table.	Frame 9: Fuel system status table.	Frame 10: Fuel system status table.	Frame 11: Fuel system status table.	Frame 12: Fuel system status table.
Frame 13: Fuel system status table.	Frame 14: Fuel system status table.	Frame 15: Fuel system status table.	Frame 16: Fuel system status table.	Frame 17: Fuel system status table.	Frame 18: Fuel system status table.
Frame 19: Fuel system status table.	Frame 20: Fuel system status table.	Frame 21: Fuel system status table.	Frame 22: Fuel system status table.	Frame 23: Fuel system status table.	Frame 24: Fuel system status table.
Frame 25: Fuel system status table.	Frame 26: Fuel system status table.	Frame 27: Fuel system status table.	Frame 28: Fuel system status table.	Frame 29: Fuel system status table.	Frame 30: Fuel system status table.
Frame 31: Fuel system status table.	Frame 32: Fuel system status table.	Frame 33: Fuel system status table.	Frame 34: Fuel system status table.	Frame 35: Fuel system status table.	Frame 36: Fuel system status table.
Frame 37: Fuel system status table.	Frame 38: Fuel system status table.	Frame 39: Fuel system status table.	Frame 40: Fuel system status table.	Frame 41: Fuel system status table.	Frame 42: Fuel system status table.
Frame 43: Fuel system status table.	Frame 44: Fuel system status table.	Frame 45: Fuel system status table.	Frame 46: Fuel system status table.	Frame 47: Fuel system status table.	Frame 48: Fuel system status table.

B01

.REM %

IDENTIFICATION

PRODUCT CODE:	MAINDEC-11-DZPCA-C-D
PRODUCT NAME:	NC11-A LOGIC TEST (GAMMA-CAMERA INTERFACE)
DATE:	MAY 1976
MAINTAINER:	DIAGNOSTIC ENGINEERING
AUTHOR:	R. MOORE

COPYRIGHT (C) 1973, 1974 & 1976
DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.

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

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

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

MAINDEC-11-DZPCA-C NC11-A LOGIC TEST MACY11 27(732) 21-SEP-76 14:14 PAGE 1
DZPCA.P11

107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160

1.0 ABSTRACT

THE NC11 (GAMMA-CAMERA INTERFACE) DIAGNOSTIC PROGRAM IS A SERIES OF TESTS DESIGNED TO TEST ALL LOGIC FUNCTIONS AND DATA PATHS. THE GAMMA-CAMERA IS NOT REQUIRED. TIMING MARK & JOYSTICK/ADC TESTING IS OPTIONAL (START 204) WHICH REQUIRES MANUAL INTERVENTION.

2.0 REQUIREMENTS

2.1 EQUIPMENT

1. PDP-11 COMPUTER WITH AT LEAST 16K OF MEMORY
2. I/O TERMINAL (I.E. ASR33 TTY),
3. NC11 INTERFACE
4. H306 JOYSTICK - OPTIONAL - SEE SECTION 9.8 & 9.9 THIS DOCUMENT

2.2 DOCUMENTATION - OPTIONAL

1. NC11-A CAMERA FRONT-END INSTRUCTION MANUAL
2. DRAWING SET NC11-A-0 (CONTAINS TEST PROCEDURES)

2.3 PRELIMINARY OPERATIONS

1. PART 1 OF THE NC11-A ADJUSTMENT PROCEDURE (A-SP-NC11-A-23) SHOULD HAVE BEEN COMPLETED BEFORE RUNNING THIS DIAGNOSTIC.

2.4 STORAGE

THIS PROGRAM USES ALL OF LOWER 16K OF MEMORY.

3.0 LOADING PROCEDURE

NORMAL PROCEDURE FOR LOADING A BINARY PROGRAM INTO MEMORY SHOULD BE FOLLOWED.

4.0 STARTING PROCEDURE

4.1 OPERATOR ACTION

INSURE THAT THE NC11-A 'TEST MODE' SWITCH (ON THE BACK OF THE INTERFACE) IS IN THE OFF POSITION.

4.2 PROGRAM STARTS

1. LOADING ADDRESS 200 AND STARTING WILL IDENTIFY THE PROGRAM, INITIALIZE THE SYSTEM, AND BEGIN LOGIC TESTING.
2. LOADING ADDRESS 204 AND STARTING WILL INITIALIZE THE SYSTEM, AND GO DIRECTLY TO THE TIMING MARK & JOYSTICK LOGIC TESTS. NOTE THAT THIS REQUIRES USER INTERACTION (SEE SECTION 9.8 & 9.9).

161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
2165.0. SWITCH REGISTER

5.1. OPTIONS

SWITCH	OCTAL	FUNCTION
SW15=1	100000	HALT ON ERROR
SW14=1	040000	LOOP ON TEST
SW13=1	020000	INHIBIT TYPEOUTS
SW11=1	004000	INHIBIT ITERATIONS
SW10=1	002000	BELL ON ERROR
SW09=1	001000	LOOP ON ERROR
SW08=1	0004XX	LOOP ON TEST IN SWR<7-0>

5.2 SOFTWARE CONTROL

1. COMPUTERS WITHOUT A HARDWARE SWITCH REGISTER HAVE A SOFTWARE SWITCH REGISTER LOCATION IN MEMORY CALLED 'SWREG' (LOC 176) WHICH CAN BE CHANGED BY THE CONSOLE FACILITIES.
2. THE SOFTWARE SWITCH REGISTER CAN ALSO BE CHANGED UNDER PROGRAM CONTROL BY TYPING THE 'CNTRL & G' * KEYS FOLLOWED BY ENTERING NEW SWITCH REGISTER OCTAL DATA TERMINATED BY A CARRIAGE RETURN.
3. THE PROGRAM WILL DEFAULT TO THE SOFTWARE SWITCH REGISTER IF THE USER STARTS THE PROGRAM WITH ALL SWITCHES SET. CONTROL OF THE SWITCH REGISTER IS THEN ACCOMPLISHED AS DEFINED IN STEP #2 ABOVE.

* IF THE PROGRAM IS PERFORMING RESET INSTRUCTIONS, SEVERAL 'CONTROL & G' COMMANDS MAY BE NECESSARY TO BE ACKNOWLEDGED BY THE PROGRAM.

6.0 ERROR REPORTING

6.1 ERROR COMMENT

ALL ERRORS ARE ACCOMPANIED WITH AN ENGLISH LANGUAGE DESCRIPTIVE COMMENT AS TO THE TYPE OF FAILURE. FURTHER QUALIFICATION OF THE ERROR CAN BE OBTAINED IF NEEDED FROM THE COMMENT AT THE ERROR PC OR FROM THE TEST ITSELF. SEE SECTION 9.9 FOR JOYSTICK/ADC ERROR AND DATA REPORTING.

6.2 ERROR DATA

*ERRPC	LISTING ADDRESS WHERE THE ERROR WAS DETECTED
*TSTNUM	TEST NUMBER WHERE THE ERROR OCCURRED
BUSADR	NC11 BUS ADDRESS WHERE 'RCVD' WAS READ
EXPCT	REGISTER DATA THAT WAS EXPECTED
RCVD	ACTUAL REGISTER DATA READ
XYHOLD	DATA WRITTEN TO XYHOLD REG. ON ADDRESS MAKE, OR DATA WRITTEN IN LIST MODE
GDADR	EXPECTED MAKE ADDRESS
BDADR	HARDWARE ADDRESS MADE
ADRS	ADDRESS OF CURRENT NPR FOR ALL RESOLUTIONS
GDDAT	EXPECTED MEMORY CONTENTS AFTER NPR

F01

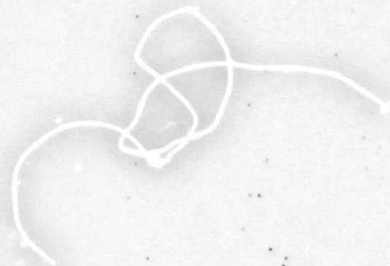
MAINDEC-11-DZNC-A-C NC11-A LOGIC TEST MACY11 27(732) 21-SEP-76 14:14 PAGE 5
DZNC.A.P11

217
218
219

BDDAT

ACTUAL MEMORY CONTENTS AFTER NPR

*ALWAYS REPORTED



20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72

7.0 MISCELLANEOUS

7.1 NC11 BUS/VECTOR/PRIORITY ADDRESS MODIFICATION

MODIFY LOCATION 'SBASE' IF BASE BUS ADDRESS IS NOT 164000.
 MODIFY LOCATION 'SVECT1' IS BASE VECTOR ADDRESS IS NOT 270.*
 MODIFY LOCATION 'SVECT1' IF PRIORITY LEVEL IS NOT 7.*

* CAUTION, LOCATION 'SVECT1' CONTAINS BOTH THE VECTOR ADDRESS AND THE PRIORITY LEVEL. THE TOP ORDER 3 BITS (15-13) REPRESENT THE PRIORITY LEVEL (1-7) AND THE LOWER ORDER 13 BITS REPRESENT THE VECTOR ADDRESS. WHEN CHANGING EITHER ONE, MAKE SURE YOU REPLACE THE OTHER.

7.2 XXDP/APT NOTES

THIS DIAGNOSTIC IS CHAINABLE (START 200) UNDER XXDP.
 THIS DIAGNOSTIC SUPPORTS 'APT', HOWEVER, IT HAS NOT BEEN TESTED.

7.3 POWER FAIL

A POWER FAILURE WILL CAUSE A RESTART MESSAGE ON POWER UP AT WHICH TIME THE PROGRAM IS RESTARTED.

8.0 EXECUTION TIME

EXECUTION TIME RANGES FROM ABOUT 30 SECONDS WITH NO ITERATIONS TO ABOUT 3 MINUTES WITH ITERATIONS ENABLED. IF THE USER SELECTS THE TIMING MARK & JOYSTICK/ADC TESTS AT PROGRAM START 204, THE PROGRAM WILL REMAIN IN THIS SECTION CONTINUALLY OUTPUTTING JOYSTICK DATA TO THE PRINTER DEPENDING ON SW13.

9.0 PROGRAM TEST DESCRIPTIONS

9.1 GENERAL

THIS DIAGNOSTIC CONTAINS A SERIES OF INDEPENDENT TESTS DESIGNED TO TEST LOGIC FUNCTIONS OF THE NC11-A GAMMA-CAMERA INTERFACE. EACH TEST IS LISTED IN THE TABLE OF CONTENTS AT THE BEGINNING OF THE LISTING. THE COMMENT FIELD WITHIN EACH TEST CAN BE HELPFUL IN UNDERSTANDING EACH TEST.

9.2 REGISTER TESTS

THE FOLLOWING REGISTERS ARE READ/WRITE & RESET TESTED:

1. COMMAND AND STATUS
2. OFFSET
3. HIGH & LOW ORDER Z (WC & CA)
4. DATA INCREMENT (I)

273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323

9.3 FLAG OR INTERRUPT TESTS

- 1. Z OVERFLOW FLAG & INTERRUPT
- 2. WORD COUNT OVERFLOW FLAG (WCO) & INTERRUPT
- 3. TIME OUT FLAG
- 4. INCREMENT OVERFLOW FLAG & INTERRUPT

9.4 LIST MODE TESTS

- 1. ONE WORD TRANSFER OF ALL DATA PATTERNS TO ADDRESS 20000.
- 2. A 8192 WORD TRANSFER OF COMPLEMENTING DATA PATTERNS STARTING AT ADDRESS 20000 WITH WC & CA & WCO CHECKS.

9.5 PROPER ADDRESS GENERATION TESTS

THE ADDRESS MAKER OF THE INTERFACE IS TESTED BY A RESOLUTION (MATRIX) RUNNING THRU ALL POSSIBLE SOFTWARE ADC VALUES AND ALL ADDRESS OFFSETS. AFTER A SUCCESSFUL PASS OF ADDRESS GENERATION OF A RESOLUTION THE TEST IS REPEATED WITH THE NPR BIT IN THE CSR ENABLED. THIS WILL ALLOW CHECKING OF THE ADDRESS PATH TO THE BUS ALONG WITH THE NPR INCREMENT LOGIC (NPR'S ARE LIMITED TO 16K OF MEM).

- 1. JOYSTICK MODE
- 2. JOYSTICK MODE FOR OVERRIDE
- 3. RESOLUTION 0
- 4. RESOLUTION 1
- 5. RESOLUTION 2
- 6. RESOLUTION 3
- 7. RESOLUTION 4
- 8. RESOLUTION 5
- 9. RESOLUTION 6
- 10. RESOLUTION 7

9.6 PROPER ADDRESS GENERATION TESTS WITH B GAMMA SET

THE ADDRESS MAKER IS TESTED IN THE SAME MANNER AS ABOVE EXCEPT THAT THE NPR PART IS NOT EXERCISED. DUAL ISOTOPE OPERATION APPLIES TO THE FOLLOWING RESOLUTIONS:

- 1. RESOLUTION 2
- 2. RESOLUTION 3
- 3. RESOLUTION 6
- 4. RESOLUTION 7

9.7 MEMORY INCREMENT OPERATION TESTS

THE FOLLOWING ARE TESTED IN THE MEMORY INCREMENT TESTS:

- 1. WORD
- 2. EVEN BYTE
- 3. ODD BYTE

324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379

9.8 TIMING MARK TEST (START 204)

TO EXECUTE THIS TEST, TWO SHORTING 'BNC' CONNECTIONS MUST BE INSERTED IN BOTH THE 'AZ AND BZ 'BNC' CONNECTORS LOCATED AT THE BACK OF THE INTERFACE. ALSO THE JOYSTICK MUST BE APPROXIMATELY CENTERED BEFORE SELECTING THIS TEST. NOTE THAT THE JOYSTICK POTS MUST BE IN RANGE OF THE NAD01'S OR A POSSIBLE NO CONVERT SITUATION WILL CAUSE AN ERROR. (REQUIREMENTS IN SP-NC11-A-23 MUST HAVE BEEN VERIFIED BEFORE SELECTING THIS TEST). THE TEST IS LOOKING FOR THE TIME MARK FLOP TO SET BIT 15 IN AN ADDRESS MAKE IN LIST MODE. WHEN THIS TEST HAS COMPLETED THE JOYSTICK/ADC LOGIC TEST IS ENTERED.

9.9 JOYSTICK ADC CONVERT, NO CONVERT & JOYSTICK LOGIC TEST (START 204).

THIS TEST IS EXECUTED CONTINUALLY AFTER COMPLETING THE TIMING MARK TEST. THE PURPOSE OF THIS TEST IS TO DETERMINE THAT THE NC11 JOYSTICK LOGIC AND ADC'S ARE FUNCTIONING (REQUIREMENTS IN SP-NC11-A-23 MUST BE VERIFIED BEFORE SELECTING THIS TEST). THERE WILL BE A CONSTANT PRINTER OUTPUT OF X & Y ADC CONVERTED VALUES UPON SUCCESSFUL CONVERSIONS, OR A 'NO CONVRT' MESSAGE IF THE ADC'S FAIL TO CONVERT (OUT OF RANGE), OR A 'BAR DN' MESSAGE IF THE INTERRUPT BAR IS DEPRESSED. SW13 WILL INHIBIT ALL TYPEOUTS MENTIONED. THE TEST IS DESIGNED TO INDICATE THAT ALL X & Y DATA BITS ARE GOOD, THAT THE NO CONVERT BIT WILL SET WITH THE JOYSTICK OUT OF RANGE, AND THAT THE INTERRUPT BAR ON THE H306 JOYSTICK IS SENSED BY THE LOGIC.

10.0 LISTING

```

%
.TITLE MAINDEC-11-DZNCA-C NC11-A LOGIC TEST
.*COPYRIGHT (C) 1976
.*DIGITAL EQUIPMENT CORP.
.*MAYNARD, MASS. 01754
.*
.*PROGRAM BY R. MOORE
.*
.*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
.*PACKAGE (MAINDEC-11-DZQAC-CO),MAR 21, 1976.
.*
$TN=1
$SWR=160000 ;;HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYP0UT
$SWR=167400
$TN=1
.SBTTL OPERATIONAL SWITCH SETTINGS
.*
.* SWITCH USE
.* -----
.* 15 HALT ON ERROR
.* 14 LOOP ON TEST
.* 13 INHIBIT ERROR TYPEOUTS
.* 11 INHIBIT ITERATIONS
.* 10 BELL ON ERROR
.* 9 LOOP ON ERROR
    
```

000001
160000
167400
000001

```

380          ;*          8          LOOP ON TEST IN SWR<7:0>
381          .SBTTL  BASIC DEFINITIONS
382
383          ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
384          001100  STACK= 1100
385          .EQUIV  EMT,ERROR          ;;BASIC DEFINITION OF ERROR CALL
386          .EQUIV  IOT,SCOPE          ;;BASIC DEFINITION OF SCOPE CALL
387
388          ;*MISCELLANEOUS DEFINITIONS
389          000011  HT= 11          ;;CODE FOR HORIZONTAL TAB
390          000012  LF= 12          ;;CODE FOR LINE FEED
391          000015  CR= 15          ;;CODE FOR CARRIAGE RETURN
392          000200  CRLF= 200        ;;CODE FOR CARRIAGE RETURN-LINE FEED
393          177776  PS= 177776      ;;PROCESSOR STATUS WORD
394          .EQUIV  PS,PSW
395          177774  STKLMT= 177774   ;;STACK LIMIT REGISTER
396          177772  PIRQ= 177772    ;;PROGRAM INTERRUPT REQUEST REGISTER
397          177570  DSWR= 177570    ;;HARDWARE SWITCH REGISTER
398          177570  DDISP= 177570   ;;HARDWARE DISPLAY REGISTER
399
400          ;*GENERAL PURPOSE REGISTER DEFINITIONS
401          000000  R0= %0          ;;GENERAL REGISTER
402          000001  R1= %1          ;;GENERAL REGISTER
403          000002  R2= %2          ;;GENERAL REGISTER
404          000003  R3= %3          ;;GENERAL REGISTER
405          000004  R4= %4          ;;GENERAL REGISTER
406          000005  R5= %5          ;;GENERAL REGISTER
407          000006  R6= %6          ;;GENERAL REGISTER
408          000007  R7= %7          ;;GENERAL REGISTER
409          .EQUIV  R6,SP          ;;STACK POINTER
410          .EQUIV  R7,PC          ;;PROGRAM COUNTER
411
412          ;*PRIORITY LEVEL DEFINITIONS
413          000000  PR0= 0          ;;PRIORITY LEVEL 0
414          000040  PR1= 40         ;;PRIORITY LEVEL 1
415          000100  PR2= 100        ;;PRIORITY LEVEL 2
416          000140  PR3= 140        ;;PRIORITY LEVEL 3
417          000200  PR4= 200        ;;PRIORITY LEVEL 4
418          000240  PR5= 240        ;;PRIORITY LEVEL 5
419          000300  PR6= 300        ;;PRIORITY LEVEL 6
420          000340  PR7= 340        ;;PRIORITY LEVEL 7
421
422          ;*"SWITCH REGISTER" SWITCH DEFINITIONS
423          100000  SW15= 100000
424          040000  SW14= 40000
425          020000  SW13= 20000
426          010000  SW12= 10000
427          004000  SW11= 4000
428          002000  SW10= 2000
429          001000  SW09= 1000
430          000400  SW08= 400
431          000200  SW07= 200
432          000100  SW06= 100
433          000040  SW05= 40
434          000020  SW04= 20
435          000010  SW03= 10
  
```

436 000004
 437 000002
 438 000001
 439
 440
 441
 442
 443
 444
 445
 446
 447
 448
 449
 450
 451 100000
 452 040000
 453 020000
 454 010000
 455 004000
 456 002000
 457 001000
 458 000400
 459 000200
 460 000100
 461 000040
 462 000020
 463 000010
 464 000004
 465 000002
 466 000001
 467
 468
 469
 470
 471
 472
 473
 474
 475
 476
 477
 478
 479 000004
 480 000010
 481 000014
 482 000014
 483 000014
 484 000020
 485 000024
 486 000030
 487 000034
 488 000060
 489 000064
 490 000240
 491 164000

SW02= 4
 SW01= 2
 SW00= 1
 .EQUIV SW09,SW9
 .EQUIV SW08,SW8
 .EQUIV SW07,SW7
 .EQUIV SW06,SW6
 .EQUIV SW05,SW5
 .EQUIV SW04,SW4
 .EQUIV SW03,SW3
 .EQUIV SW02,SW2
 .EQUIV SW01,SW1
 .EQUIV SW00,SW0

.*DATA BIT DEFINITIONS (BIT00 TO BIT15)

BIT15= 100000
 BIT14= 40000
 BIT13= 20000
 BIT12= 10000
 BIT11= 4000
 BIT10= 2000
 BIT09= 1000
 BIT08= 400
 BIT07= 200
 BIT06= 100
 BIT05= 40
 BIT04= 20
 BIT03= 10
 BIT02= 4
 BIT01= 2
 BIT00= 1
 .EQUIV BIT09,BIT9
 .EQUIV BIT08,BIT8
 .EQUIV BIT07,BIT7
 .EQUIV BIT06,BIT6
 .EQUIV BIT05,BIT5
 .EQUIV BIT04,BIT4
 .EQUIV BIT03,BIT3
 .EQUIV BIT02,BIT2
 .EQUIV BIT01,BIT1
 .EQUIV BIT00,BIT0

.*BASIC "CPU" TRAP VECTOR ADDRESSES

ERRVEC= 4
 RESVEC= 10
 TBITVEC= 14
 TRTVEC= 14
 BPTVEC= 14
 IOTVEC= 20
 PWRVEC= 24
 EMTVEC= 30
 TRAPVEC= 34
 TKVEC= 60
 TPVEC= 64
 PIRQVEC= 240
 ABASE= 164000

;; TIME OUT AND OTHER ERRORS
 ;; RESERVED AND ILLEGAL INSTRUCTIONS
 ;; "T" BIT
 ;; TRACE TRAP
 ;; BREAKPOINT TRAP (BPT)
 ;; INPUT/OUTPUT TRAP (IOT) **SCOPE**
 ;; POWER FAIL
 ;; EMULATOR TRAP (EMT) **ERROR**
 ;; "TRAP" TRAP
 ;; TTY KEYBOARD VECTOR
 ;; TTY PRINTER VECTOR
 ;; PROGRAM INTERRUPT REQUEST VECTOR
 ;; BASE NC11 BUS ADRS EQUATE

492	160270			AVECT1= 160270	;BASE NC11 VECTOR ADRS EQUATE - TOP 3 BITS=PRIORITY
493	000340			APRIOR= 340	;NC11 PRIORITY LEVEL EQUATE
494	000001			ADEVM= 1	;DEFAULT TO ONE NC11
495				.SBTTL TRAP CATCHER	
496					
497	000000			.=0	
498				;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"	
499				;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS	
500				;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS	
501	000174			.=174	
502	000174	000000		DISPREG: .WORD 0	::SOFTWARE DISPLAY REGISTER
503	000176	000000		SWREG: .WORD 0	::SOFTWARE SWITCH REGISTER
504				.SBTTL STARTING ADDRESS(ES)	
505	000200	000137	001614	JMP @#START2	::JUMP TO STARTING ADDRESS OF PROGRAM
506		000204		.=204	
507	000204	000137	001604	JMP @#START1	;WILL SELECT TIMING MARK & JOYSTICK TESTS

508
509
510
511
512 000210
513 000046
514 000046 012646
515 000052 000052
516 000052 000000
517 000210
518 001000
519
520
521
522
523
524 001000
525 000024 000024
526 000024 000200
527 000044 000044
528 000044 001000
529 001000
530
531
532
533
534 001000
535 001000 000000
536 001002 001174
537 001004 000012
538 001006 000036
539 001010 000000
540 001012 000031

.SBTTL ACT11 HOOKS

HOOKS REQUIRED BY ACT11

```

$SVPC=      ;SAVE PC
.=46
$ENDAD      ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
.=52
.WORD 0     ;;2)SET LOC.52 TO ZERO
.= $SVPC    ;; RESTORE PC
.=1000

```

.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)
$STMT: .WORD 10. ;;RUN TIM OF LONGEST TEST
$PASTM: .WORD 30. ;;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)

```

541
542
543
544
545
546
547 001100
548 001100
549 001100 000000
550 001102 000
551 001103 000
552 001104 000000
553 001106 000000
554 001110 000000
555 001112 000000
556 001114 000
557 001115 001
558 001116 000000
559 001120 000000
560 001122 000000
561 001124 000000
562 001126 000000
563 001130 000000
564 001132 000000
565 001134 000
566 001135 000
567 001136 000000
568 001140 177570
569 001142 177570
570 001144 177560
571 001146 177562
572 001150 177564
573 001152 177566
574 001154 000
575 001155 002
576 001156 012
577 001157 000
578 001160 000000
579 001162 000000
580 001164 177607 000377
581 001170 077
582 001171 015
583 001172 000012
584
585
586
587
588
589 001174
590 001174 000000
591 001176 000000
592 001200 000000
593 001202 000000
594 001204 000000
595 001206 000000
596 001210 000000

.SBTTL COMMON TAGS

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

SCM TAG: =1100

;; START OF COMMON TAGS
\$CM TAG: .WORD 0
\$STNM: .BYTE 00
\$ERFLG: .BYTE 00
\$ICNT: .WORD 00
\$LPADR: .WORD 00
\$LPERR: .WORD 00
\$ERTTL: .WORD 00
\$ITEMB: .BYTE 00
\$ERMAX: .BYTE 1
\$ERRPC: .WORD 00
\$GDADR: .WORD 00
\$BDADR: .WORD 00
\$GDDAT: .WORD 00
\$BDDAT: .WORD 00
\$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
\$TIMES: 0
\$ESCAPE: 0
\$BELL: .ASCIZ <207><377><377>
\$QUES: .ASCII /?/
\$CRLF: .ASCII <15>
\$LF: .ASCIZ <12>

;; CONTAINS THE TEST NUMBER
;; CONTAINS ERROR FLAG
;; CONTAINS SUBTEST ITERATION COUNT
;; CONTAINS SCOPE LOOP ADDRESS
;; CONTAINS SCOPE RETURN FOR ERRORS
;; CONTAINS TOTAL ERRORS DETECTED
;; CONTAINS ITEM CONTROL BYTE
;; CONTAINS MAX. ERRORS PER TEST
;; CONTAINS PC OF LAST ERROR INSTRUCTION
;; CONTAINS ADDRESS OF 'GOOD' DATA
;; CONTAINS ADDRESS OF 'BAD' DATA
;; CONTAINS 'GOOD' DATA
;; CONTAINS 'BAD' DATA
;; RESERVED--NOT TO BE USED

;; AUTOMATIC MODE INDICATOR
;; INTERRUPT MODE INDICATOR

;; ADDRESS OF SWITCH REGISTER
;; ADDRESS OF DISPLAY REGISTER
;; TTY KBD STATUS
;; TTY KBD BUFFER
;; TTY PRINTER STATUS REG. ADDRESS
;; TTY PRINTER BUFFER REG. ADDRESS
;; CONTAINS NULL CHARACTER FOR FILLS
;; CONTAINS # OF FILLER CHARACTERS REQUIRED
;; INSERT FILL CHARS. AFTER A "LINE FEED"
;; "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
;; MAX. NUMBER OF ITERATIONS
;; ESCAPE ON ERROR ADDRESS
;; CODE FOR BELL
;; QUESTION MARK
;; CARRIAGE RETURN
;; LINE FEED

::*****
: .SBTTL APT MAILBOX-ETABLE

::*****
: .EVEN
\$MAIL: .WORD
\$MSGTY: .WORD AMSGTY
\$FATAL: .WORD AFATAL
\$TESTN: .WORD ATESTN
\$PASS: .WORD APASS
\$DEVCT: .WORD ADEVCT
\$UNIT: .WORD AUNIT
\$MSGAD: .WORD AMSGAD

;; APT MAILBOX
;; MESSAGE TYPE CODE
;; FATAL ERROR NUMBER
;; TEST NUMBER
;; PASS COUNT
;; DEVICE COUNT
;; I/O UNIT NUMBER
;; MESSAGE ADDRESS

```

6097 001212 000000
6098 001214
6099 001214 000
6100 001215 000
6101 001216 000000
6102 001220 000000
6103 001222 000000
6104
6105
6106
6107
6108
6109
6110 001224 000
6111 001225 000
6112
6113
6114
6115
6116 001226 000000
6117
6118 001230 000
6119 001231 000
6120 001232 000000
6121 001234 000
6122 001235 000
6123 001236 000000
6124 001240 000
6125 001241 000
6126 001242 000000
6127 001244 160270
6128 001246 000000
6129 001250 164000
6130 001252 000001
6131 001254 000000
6132 001256

```

```

$MSGLG: .WORD  AMSGLG  :: MESSAGE LENGTH
$ETABLE:      :: APT ENVIRONMENT TABLE
$ENV: .BYTE   AENV     :: ENVIRONMENT BYTE
$ENVM: .BYTE  AENVM    :: ENVIRONMENT MODE BITS
$SWREG: .WORD ASWREG   :: APT SWITCH REGISTER
$USWR: .WORD  AUSWR    :: USER SWITCHES
$CPUOP: .WORD  ACPUOP  :: 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
$MAMS1: .BYTE  AMAMS1  :: HIGH ADDRESS, M.S. BYTE
$MTYP1: .BYTE  AMTYP1  :: MEM. TYPE, BLK#1
::
:: MEM. TYPE BYTE -- (HIGH BYTE)
::           900 NSEC CORE=001
::           300 NSEC BIPOLAR=002
::           500 NSEC MOS=003
$MADR1: .WORD  AMADR1  :: HIGH ADDRESS, BLK#1
::
:: MEM. LAST ADDR.=3 BYTES, THIS WORD AND LOW OF "TYPE" ABOVE
$MAMS2: .BYTE  AMAMS2  :: HIGH ADDRESS, M.S. BYTE
$MTYP2: .BYTE  AMTYP2  :: MEM. TYPE, BLK#2
$MADR2: .WORD  AMADR2  :: MEM. LAST ADDRESS, BLK#2
$MAMS3: .BYTE  AMAMS3  :: HIGH ADDRESS, M.S. BYTE
$MTYP3: .BYTE  AMTYP3  :: MEM. TYPE, BLK#3
$MADR3: .WORD  AMADR3  :: MEM. LAST ADDRESS, BLK#3
$MAMS4: .BYTE  AMAMS4  :: HIGH ADDRESS, M.S. BYTE
$MTYP4: .BYTE  AMTYP4  :: MEM. TYPE, BLK#4
$MADR4: .WORD  AMADR4  :: MEM. LAST ADDRESS, BLK#4
$VECT1: .WORD  AVECT1  :: INTERRUPT VECTOR#1, BUS PRIORITY#1
$VECT2: .WORD  AVECT2  :: INTERRUPT VECTOR#2, BUS PRIORITY#2
$BASE: .WORD  ABASE    :: BASE ADDRESS OF EQUIPMENT UNDER TEST
$DEVN: .WORD  ADEVN    :: DEVICE MAP
$CDW1: .WORD  ACDW1    :: CONTROLLER DESCRIPTION WORD#1
$ETEND:
.MEXIT

```



```

787
788
789 001546 164000
790 001550 164002
791 001552
792 001552 164004
793 001554 164006
794 001556 164010
795 001560 164012
796 001562 164014
797
798
799
800 001564 000270
801 001566 000272
802 001570 000274
803 001572 000276
804
805
806
807 001574 000340
808 001576 000300
809
810
811
812 001600 000000
813 001602 000000

;NC11 BUS REGISTER ADDRESS POINTERS
NCCSR: 164000 ;COMMAND/STATUS
NCOFF: 164002 ;OFFSET ADRS
NCADR: ;READ ADRS MAKE OR WRITE LIST MD NPR DATA
NCXYH: 164004 ;READ X/Y ADC OR WRITE X/Y ADRS
NCZHWC: 164006 ;HIGH ORDER Z OR WORD COUNT
NCZLCA: 164010 ;LOW ORDER Z OR CURRENT ADRS
NCIREG: 164012 ;INCREMENT
NCSFR: 164014 ;SPECIAL FUNCTION (SF)

;NC11 VECTOR ADDRESS POINTERS
NCVCT0: 270 ;CELL OVERFLOW INTR VECTOR PTR
NCVCT2: 272
NCVCT4: 274 ;Z OVERFLOW OR WC OVERFLOW VECTOR PTR
NCVCT6: 276

;NC11 DEVICE LEVEL(S)
NCBRL: 340 ;DEVICE LEVEL
NCBRL1: 300 ;DEVICE LEVEL -1

;COMMON PROGRAM LOCATION(S)
TSTNUM: 0 ;CONTAINS TEST NUMBER ON ERROR
STST54: 0 ;0 SAYS DO LOGIC TESTS, -1 SAYS DO TIMING MARK & JOYSTICK TESTS

```

```

814 .SBTTL PROGRAM START
815 001604 012737 177777 001602 START1: MOV #-1,STST54 ;SELECT TIMING MARK & JOYSTICK TESTS
816 001612 000402 BR START ;START
817 001614 005037 001602 START2: CLR STST54 ;SELECT LOGIC TESTS
818 001620
819 .SBTTL INITIALIZE THE COMMON TAGS
820 ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
821 001620 012706 001100 MOV $CMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
822 001624 005026 CLR (R6)+ ;;CLEAR MEMORY LOCATION
823 001626 022706 001140 CMP $SWR,R6 ;;DONE?
824 001632 001374 BNE -6 ;;LOOP BACK IF NO
825 001634 012706 001100 MOV $STACK,SP ;;SETUP THE STACK POINTER
826 ;;INITIALIZE A FEW VECTORS
827 001640 012737 016150 000020 MOV $$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
828 001646 012737 000340 000022 MOV #340,@#IOTVEC+2 ;;LEVEL 7
829 001654 012737 015606 000030 MOV $ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
830 001662 012737 000340 000032 MOV #340,@#EMTVEC+2 ;;LEVEL 7
831 001670 012737 017412 000034 MOV $TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
832 001676 012737 000340 000036 MOV #340,@#TRAPVEC+2 ;;LEVEL 7
833 001704 012737 017206 000024 MOV $SPWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
834 001712 012737 000340 000026 MOV #340,@#PWRVEC+2 ;;LEVEL 7
835 001720 005037 001160 CLR $TIMES ;;INITIALIZE NUMBER OF ITERATIONS
836 001724 005037 001162 CLR $ESCAPE ;;CLEAR THE ESCAPE ON ERROR ADDRESS
837 001730 112737 000001 001115 MOVB #1,$ERMAX ;;ALLOW ONE ERROR PER TEST
838 001736 012737 001736 001106 MOV #,$SLPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
839 001744 012737 001744 001110 MOV #,$SLPERR ;;SETUP THE ERROR LOOP ADDRESS
840 ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
841 ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
842 001752 013746 000004 MOV @#ERRVEC,-(SP) ;;SAVE ERROR VECTOR
843 001756 012737 002012 000004 MOV #64,$@#ERRVEC ;;SET UP ERROR VECTOR
844 001764 012737 177570 001140 MOV $DSWR,$SWR ;;SETUP FOR A HARDWARE SWICH REGISTER
845 001772 012737 177570 001142 MOV $DDISP,$DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
846 002000 022777 177777 177132 CMP #-1,$SWR ;;TRY TO REFERENCE HARDWARE SWR
847 002006 001012 BNE 66$ ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
848 ;;AND THE HARDWARE SWR IS NOT = -1
849 002010 000403 BR 65$ ;;BRANCH IF NO TIMEOUT
850 002012 012716 002020 64$: MOV #65$,(SP) ;;SET UP FOR TRAP RETURN
851 002016 000002 RTI
852 002020 012737 000176 001140 65$: MOV $SWREG,$SWR ;;POINT TO SOFTWARE SWR
853 002026 012737 000174 001142 MOV $DISPREG,$DISPLAY
854 002034 012637 000004 66$: MOV (SP)+,@#ERRVEC ;;RESTORE ERROR VECTOR
855
856 002040 005037 001202 CLR $PASS ;;CLEAR PASS COUNT
857 002044 132737 000200 001215 BITB $APTSIZE,$ENVM ;;TEST USER SIZE UNDER APT
858 002052 001403 BEQ 67$ ;;YES,USE NON-APT SWITCH
859 002054 012737 001216 001140 MOV $SSWREG,$SWR ;;NO,USE APT SWITCH REGISTER
860 002062
861 002062 013737 001244 001574 67$: SETUP1: MOV $VECT1,$NCBRL ;;GET PRIORITY LEVEL
862 002070 042737 017777 001574 BIC #17777,$NCBRL ;;RID VECTOR ADRS
863 002076 000337 001574 SWAB $NCBRL ;;ADJUST FOR PSW
864 002102 013737 001574 001576 MOV $NCBRL,$NCBRL1 ;;PREPARE DEVICE LEVEL -1
865 002110 162737 000040 001576 SUB #40,$NCBRL1 ;;DO IT
866 002116 012700 001546 MOV $NCCSR,$R0 ;;SET UP REG ADRS POINTERS
867 002122 013701 001250 MOV $BASE,$R1 ;;GET BASE ADRS
868 002126 010120 SETUP2: MOV R1,(R0)+
869 002130 062701 000002 ADD #2,R1

```

```

870 002134 022700 001564      CMP      #NCSFR+2,R0      ;ALL DONE?
871 002140 001372              BNE      SETUP2         ;BR IF NOT
872 002142 012700 001564      MOV      #NCVCT0,R0     ;SET UP NC11 VECTOR ADRS POINTERS
873 002146 013701 001244      MOV      $VECT1,R1     ;GET BASE VECTOR ADRS
874 002152 042701 160000      BIC      #160000,R1    ;GET RID OF PRIORITY LEVEL
875 002156 010120              SETUP3: MOV     R1,(R0)+  ;
876 002160 062701 000002      ADD      #2,R1         ;
877 002164 022700 001574      CMP      #NCVCT6+2,R0  ;ALL DONE?
878 002170 001372              BNE      SETUP3         ;BR IF NOT
879
880                               .SBTTL  TYPE PROGRAM NAME
881 002172 005227 177777      ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
882 002176 001044              INC      #-1           ;FIRST TIME?
883 002200 104400 002246      BNE      64$           ;BRANCH IF NO
884                               .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
885 002204 005737 000042      TYPE    ,65$         ;TYPE ASCIZ STRING
886 002210 001012              TST     @#42         ;ARE WE RUNNING UNDER XXDP/ACT?
887 002212 123727 001214 000001 BNE      66$           ;BRANCH IF YES
888 002220 001406              CMPB   $ENV,#1       ;ARE WE RUNNING UNDER APT?
889 002222 023727 001140 000176 BEQ      66$           ;BRANCH IF YES
890 002230 001005              CMP    SWR,#SWREG    ;SOFTWARE SWITCH REG SELECTED?
891 002232 104405              BNE    67$           ;BRANCH IF NO
892 002234 000403              GTSWR ;GET SOFT-SWR SETTINGS
893 002236 112737 000001 001134 66$:  BR      67$         ;
894 002244              MOVB   #1,$AUTOB   ;SET AUTO-MODE INDICATOR
895 002244 000421              BR      64$         ;GET OVER THE ASCIZ
896                               ;;65$: .ASCIZ <CRLF>#MD11-DZNC-C NC11-A LOGIC TEST#<CRLF>
897 002310              64$:
898 002310 012737 000340 177776 RESTRT: MOV   #PR7,PSW   ;SET PRIORITY TO HIGHEST LEVEL
899 002316 012706 001100              MOV   #STACK,SP    ;ALWAYS RESET STACK PTR
900 002322 000005              RESET ;INITIALIZE NC11 BEFORE TESTING
901 002324 005737 001602              TST   STST54       ;GO TO THE MANUAL INTERVENTION TESTS?
902 002330 001405              BEQ   1$           ;BR IF NOT
903 002332 112737 000054 001102      MOVB  #54,$STSTNM   ;SET UP STARTING TEST # 54 FOR THESE TSTS
904 002340 000137 012054              JMP   @#TEST54     ;GO TO MANUAL TESTS - NC11 JOYSTICK REQUIRED
905 002344              1$:

```

```

906
907
908
909 002344 000240
910 002346 012737 002362 001106
911 002354 012737 000001 001200
912 002362 112737 000001 001102
913 002370 012737 002424 001110
914 002376 005037 001124
915 002402 005037 001126
916 002406 012737 002442 000004
917 002414 013700 001546
918 002420 012701 000007
919 002424 010037 001122
920 002430 005710
921 002432 005720
922 002434 005301
923 002436 001403
924 002440 000771
925 002442 022626
926 002444 104001
927 002446 012737 000006 000004
928
929
930
931
932 002454 000004
933 002456 012737 000004 001160
934 002464 013737 001546 001122
935 002472 012737 002504 001110
936 002500 012700 170000
937 002504 010037 001124
938 002510 042737 170200 001124
939 002516 010077 177024
940 002522 017737 177020 001126
941 002530 042737 170200 001126
942 002536 023737 001124 001126
943 002544 001401
944 002546 104002
945 002550 005200
946 002552 001354
947
948
949
950
951 002554 000004
952 002556 012737 000010 001160
953 002564 012777 000102 176770
954 002572 013737 001546 001122
955 002600 005037 001124
956 002604 012777 007577 176734
957 002612 000005
958 002614 017737 176726 001126
959 002622 001401
960 002624 104003
961

;*****
;*TEST 1 TEST THAT ALL NC11 REGS ARE ACCESSIBLE
;*****
TST1: <NOP>
MOV #10$, $LPADR ;; SET SCOPE LOOP ADDRESS
MOV #1, $TESTN ;; SET TEST NUMBER IN APT MAIL BOX
10$: MOVB #1, $STSTNM ;; SET TEST # TO 1
MOV #1$, $LPERR ;; SET UP SCOPE LOOP ADRS
CLR $GDDAT ;; NO DATA COMPARE
CLR $BDDAT ;; NO DATA COMPARE
MOV #2$, @#ERRVEC ;; SET UP TIMEOUT RETURN ADRS
MOV NCCSR, RO ;; SET UP 1ST NC11 BUS ADRS
1$: MOV #7, R1 ;; SET UP REG COUNT
MOV RO, $BDADR ;; SET UP CURRENT NC BUS ADRS
TST (RO) ;; SEE IF THERE
TST (RO)+ ;; BUMP TO NEXT
DEC R1 ;; COUNT 7 OF THEM
BEQ 3$ ;; BR IF ALL DONE
BR 1$ ;; TRY NEXT
2$: CMP (SP)+, (SP)+ ;; FIX STACK SINCE NO RTI
ERROR 1 ;; BUS ADRS INDICATED DID NOT RESPOND
3$: MOV #ERRVEC+2, @#ERRVEC ;; RESTORE LOC 4

;*****
;*TEST 2 TEST THAT COMMAND/STATUS REG IS WRITE/READABLE (COUNT PATTERN)
;*****
TST2: SCOPE
MOV #4, $TIMES ;; DO 4 ITERATIONS
MOV NCCSR, $BDADR ;; SET UP CSR ADRS
MOV #1$, $LPERR ;; SET UP SCOPE LOOP ADRS
1$: MOV #170000, RO ;; START AT 0 - HI BITS FOR NOISE
MOV RO, $GDDAT ;; LD EXPECTED
BIC #170200, $GDDAT ;; MASK TO WRITEABLE BITS
MOV RO, @NCCSR ;; SEND PATTERN OUT
MOV @NCCSR, $BDDAT ;; READ IT BACK
BIC #170200, $BDDAT ;; MASK OUT UNKNOWNNS
CMP $GDDAT, $BDDAT ;; CORRECT?
BEQ 2$ ;; BR IF SO
2$: ERROR 2 ;; NCCSR WRITE (READ) FAILURE
INC RO ;; ADVANCE PATTERN
BNE 1$ ;; REPEAT NX PATTERN IF NOT ALL TESTED

;*****
;*TEST 3 TEST THAT "RESET" WILL CLEAR THE COMMAND/STATUS REG
;*****
TST3: SCOPE
MOV #10, $TIMES ;; DO 10 ITERATIONS
MOV #102, @NCSFR ;; CLR ALL
MOV NCCSR, $BDADR ;; SETUP CSR ADRS
CLR $GDDAT ;; LD EXPECTED
MOV #7577, @NCCSR ;; LOAD ALL WRITEABLE BITS
RESET ;; CLR CSR
MOV @NCCSR, $BDDAT ;; READ IT
BEQ TST4 ;; NEXT TEST IF CLEARED
3: ERROR 3 ;; RESET FAILED TO CLR CSR REG

```

K02

962
963
964
965 002626 000004
966 002630 012737 000004 001160
967 002636 013737 001550 001122
968 002644 012737 002654 001110
969 002652 005000
970 002654 010037 001124
971 002660 042737 001774 001124
972 002666 010077 176656
973 002672 017737 176652 001126
974 002700 023737 001124 001126
975 002706 001401
976 002710 104002
977 002712 062700 002001
978 002716 103356
979
980
981
982
983 002720 000004
984 002722 012737 000010 001160
985 002730 013737 001550 001122
986 002736 005037 001124
987 002742 012777 174003 176600
988 002750 000005
989 002752 017737 176572 001126
990 002760 001401
991 002762 104003
992
993
994
995
996 002764 000004
997 002766 012737 000004 001160
998 002774 013737 001554 001122
999 003002 012737 003016 001110
1000 003010 012737 177777 001124
1001 003016 013777 001124 176530
1002 003024 017737 176524 001126
1003 003032 023737 001124 001126
1004 003040 001401
1005 003042 104002
1006 003044 005337 001124
1007 003050 102362
1008
1009
1010
1011
1012 003052 000004
1013 003054 012737 000004 001160
1014 003062 013737 001556 001122
1015 003070 012737 003104 001110
1016 003076 012737 177777 001124
1017 003104 013777 001124 176444

```
*****
*TEST 4 TEST THAT THE OFFSET REG IS WRITE/READABLE (COUNT PATTERN)
*****
TST4: SCOPE
MOV #4,STIMES ;DO 4 ITERATIONS
MOV NCOFF,$BDADR ;SET UP OFFSET REG ADRS
MOV #1,$SLPERR ;SET UP SCOPE LOOP ADRS
CLR RO ;START WITH ALL BITS ZERO
1$: MOV RO,$GDDAT ;LD EXPECTED
BIC #1774,$GDDAT ;MASK TO WRITEABLE BITS
MOV RO,$NCOFF ;SEND PATRN OUT
MOV $NCOFF,$BDDAT ;READ IT BACK
CMP $GDDAT,$BDDAT ;CORRECT?
BEQ 2$ ;BR IF SO
ERROR 2 ;OFFSET REG WRITE/READ FAILURE
2$: ADD #2001,RO ;ADVANCE PATTERN
BCC 1$ ;REPEAT NX PATTERN IF NOT ALL TESTED

*****
*TEST 5 TEST THAT "RESET" WILL CLEAR THE OFFSET REG
*****
TST5: SCOPE
MOV #10,STIMES ;DO 10 ITERATIONS
MOV NCOFF,$BDADR ;SET LD OFFSET REG ADRS
CLR $GDDAT ;LD EXPECTED
MOV #174003,$NCOFF ;LOAD ALL WRITEABLE BITS
RESET ;CLR OFFSET REG
MOV $NCOFF,$BDDAT ;READ IT
BEQ TST6 ;NEXT TEST IF CLEARED
ERROR 3 ;RESET FAILED TO CLR HI OFFSET REG

*****
*TEST 6 TEST THAT THE HIGH ORDER Z REG IS WRITE/READABLE (COUNT PATTERN)
*****
TST6: SCOPE
MOV #4,STIMES ;DO 4 ITERATIONS
MOV NCZHC,$BDADR ;SET UP Z HI REG ADRS
MOV #1,$SLPERR ;SET UP SCOPE LOOP ADRS
MOV #-1,$GDDAT ;LD EXPECTED - START WITH ALL ONES
1$: MOV $GDDAT,$NCZHC ;LD Z REG HI
MOV $NCZHC,$BDDAT ;READ IT BACK
CMP $GDDAT,$BDDAT ;CORRECT?
BEQ 2$ ;BR IF SO
ERROR 2 ;Z HI REG WRITE/READ FAILURE
2$: DEC $GDDAT ;DECREASE COUNT BY ONE
BVC 1$ ;AGAIN TILL ALL PATTERNS DONE

*****
*TEST 7 TEST THAT THE LOW ORDER Z REG IS WRITE/READABLE (COUNT PATTERN)
*****
TST7: SCOPE
MOV #4,STIMES ;DO 4 ITERATIONS
MOV NCZLCA,$BDADR ;SET UP Z HI REG ADRS
MOV #1,$SLPERR ;SET UP SCOPE LOOP ADRS
MOV #-1,$GDDAT ;LD EXPECTED - START WITH ALL ONES
1$: MOV $GDDAT,$NCZLCA ;LD Z REG LOW
```



```

1018 003112 017737 176440 001126      MOV      QNCZLCA,$BDDAT  ;READ IT BACK
1019 003120 023737 001124 001126      CMP      $GDDAT,$BDDAT  ;CORRECT?
1020 003126 001401                BEQ      2$              ;BR IF SO
1021 003130 104002                ERROR    2              ;Z LOW REG WRITE/READ FAILURE
1022 003132 005337 001124      2$:     DEC      $GDDAT    ;DECREASE COUNT BY ONE
1023 003136 102362                BVC      1$              ;AGAIN TILL ALL PATTERNS DCNE
1024
1025
1026      ;*****
1027      ;*TEST 10      TEST THAT "RESET" WILL CLEAR LOW/HIGH ORDER Z REGS
1028      ;*****
1028 003140 000004      TST10:  SCOPE
1029 003142 012737 000010 001160      MOV      #10,$TIMES    ;;DO 10 ITERATIONS
1030 003150 005037 001124                CLR      $GDDAT        ;LD EXPECTED
1031 003154 012777 177777 176374      MOV      #-1,QNCZLCA   ;SET ALL BITS - Z LOW
1032 003162 012777 177777 176364      MOV      #-1,QNCZHC    ;SET ALL BITS - Z HIGH
1033 003170 000005                RESET                ;DO A BUS RESET
1034 003172 017737 176360 001126      MOV      QNCZLCA,$BDDAT ;READ Z LOW
1035 003200 001404                BEQ      1$              ;BR IF CLRED
1036 003202 013737 001556 001122      MOV      NCZLCA,$BDADR  ;SET UP Z LOW REG ADRS
1037 003210 104003                ERROR    3              ;RESET FAILED TO CLEAR Z LOW REG
1038 003212 017737 176336 001126      1$:     MOV      QNCZHC,$BDDAT ;READ Z HIGH
1039 003220 001404                BEQ      TST11           ;NEXT TEST IF CLRED
1040 003222 013737 001554 001122      MOV      NCZHC,$BDADR   ;SET UP Z HIGH REG ADRS
1041 003230 104003                ERROR    3              ;RESET FAILED TO CLEAR Z HIGH REG
1042
1043
1044      ;*****
1045      ;*TEST 11      TEST THAT THE I REG IS WRITE/READABLE (COUNT PATTERN)
1046      ;*****
1046 003232 000004      TST11:  SCOPE
1047 003234 012737 000004 001160      MOV      #4,$TIMES    ;;DO 4 ITERATIONS
1048 003242 013737 001560 001122      MOV      NCIREG,$BDADR ;SET UP I REG ADRS
1049 003250 012737 003264 001110      MOV      #1$,SLPERR    ;SET UP SCOPE LOOP ADRS
1050 003256 012737 177777 001124      MOV      #-1,$GDDAT    ;LD EXPECTED - START WITH ALL ONES
1051 003264 013777 001124 176266      1$:     MOV      $GDDAT,QNCIREG ;LD I REG
1052 003272 017737 176262 001126      MOV      QNCIREG,$BDDAT ;READ IT BACK
1053 003300 023737 001124 001126      CMP      $GDDAT,$BDDAT ;CORRECT?
1054 003306 001401                BEQ      2$              ;BR IF SO
1055 003310 104002                ERROR    2              ;I REG WRITE/READ FAILURE
1056 003312 005337 001124      2$:     DEC      $GDDAT    ;DECREASE COUNT BY ONE
1057 003316 103762                BCS      1$              ;AGAIN TILL ALL PATTERNS DONE
1058
1059
1060      ;*****
1061      ;*TEST 12      TEST THAT "RESET" WILL CLEAR I REG
1062      ;*****
1062 003320 000004      TST12:  SCOPE
1063 003322 012737 000010 001160      MOV      #10,$TIMES   ;;DO 10 ITERATIONS
1064 003330 013737 001560 001122      MOV      NCIREG,$BDADR ;SET UP I REG ADRS
1065 003336 005037 001124                CLR      $GDDAT        ;LD EXPECTED
1066 003342 012777 177777 176210      MOV      #-1,QNCIREG   ;SET ALL BITS
1067 003350 000005                RESET                ;CLR I REG
1068 003352 017737 176202 001126      MOV      QNCIREG,$BDDAT ;READ I REG
1069 003360 001401                BEQ      TST13           ;NEXT TEST IF CLEARED
1070 003362 104003                ERROR    3              ;RESET FAILED TO CLR I REG
1071
1072
1073      ;*****
1073      ;*TEST 13      TEST SPECIAL FUNCTION "CLEAR Z/WC & Z/CA"

```

```

1074
1075 003364 000004
1076 003366 005037 001124
1077 003372 012777 177777 176154
1078 003400 012777 177777 176150
1079 003406 012777 000020 176146
1080 003414 017737 176134 001126
1081 003422 001404
1082 003424 013737 001554 001122
1083 003432 104004
1084 003434 017737 176116 001126 1$:
1085 003442 001404
1086 003444 013737 001556 001122
1087 003452 104004
1088
1089
1090
1091
1092 003454 000004
1093 003456 013737 001560 001122
1094 003464 005037 001124
1095 003470 012777 177777 176062
1096 003476 012777 000100 176056
1097 003504 017737 176050 001126
1098 003512 001401
1099 003514 104005
1100
1101
1102
1103
1104 003516 000004
1105 003520 012737 000004 001160
1106 003526 012737 003546 001110
1107 003534 012700 000001
1108 003540 012777 000020 176014
1109 003546 012777 000004 176006 1$:
1110 003554 013737 001556 001122
1111 003562 010037 001124
1112 003566 017737 175764 001126
1113 003574 023737 001124 001126
1114 003602 001401
1115 003604 104006
1116 003606 013737 001554 001122 2$:
1117 003614 005037 001124
1118 003620 017737 175730 001126
1119 003626 001401
1120 003630 104006
1121 003632 005200
1122 003634 001344
1123
1124
1125
1126
1127 003636 000004
1128 003640 012737 000004 001160
1129 003646 012737 003666 001110

```

```

*****
TST13: SCOPE
CLR $GDDAT ;LD EXPECTED
MOV #-1, JNCZHC ;SET ALL BITS - Z/WC
MOV #-1, JNCZLCA ;SET ALL BITS - Z/CA
MOV #20, JNCSFR ;SEND SPECIAL FUNCT CLR
MOV JNCZHC, $BDDAT ;READ Z/WC
BEQ 1$ ;BR IF CLRED
MOV NCZHC, $BDADR ;SET UP Z/WC BUS ADRS
ERROR 4 ;SPECIAL FUNCTION CLR Z/WC FAILED
1$: MOV JNCZLCA, $BDDAT ;READ Z/CA
BEQ TST14 ;NEXT TEST IF CLEARED
MOV NCZLCA, $BDADR ;SET UP Z/CA BUS ADRS
ERROR 4 ;SPECIAL FUNCTION CLR Z/CA FAILED
*****
;TEST 14 TEST SPECIAL FUNCTION "CLEAR ALL" (USE I REG)
*****
TST14: SCOPE
MOV NCIREG, $BDADR ;SET UP I REG ADRS
CLR $GDDAT ;LD EXPECTED
MOV #-1, JNCIREG ;SET ALL BITS
MOV #100, JNCSFR ;SEND SPECIAL FUNCTION "CLR ALL"
MOV JNCIREG, $BDDAT ;READ I REG
BEQ TST15 ;NEXT TEST IF CLEARED
ERROR 5 ;SPECIAL FUNCTION FAILED TO "CLEAR ALL" I REG
*****
;TEST 15 TEST THAT Z LOW INCREMENTS ON SPECIAL FUNCT CONVERT ADC'S
*****
TST15: SCOPE
MOV #4, $TIMES ;DO 4 ITERATIONS
MOV #1$, $LPERR ;SET UP SCOPE LOOP ADRS
MOV #1, R0 ;LD EXPECTED IN R0
MOV #20, JNCSFR ;SPECIAL FUNCTION CLR Z
1$: MOV #4, JNCSFR ;START ADC'S
MOV NCZLCA, $BDADR ;SET UP Z/CA REG ADRS
MOV R0, $GDDAT ;LD EXPECTED
MOV JNCZLCA, $BDDAT ;READ Z REG LO
CMP $GDDAT, $BDDAT ;CORRECT?
BEQ 2$ ;BR IF SO
ERROR 6 ;Z LOW FAILED TO INCREMENT ON ADC START
2$: MOV NCZHC, $BDADR ;SET UP Z/WC REG ADRS
CLR $GDDAT ;LD EXPECTED
MOV JNCZHC, $BDDAT ;SEE THAT HI ORDER Z REG IS 0
BEQ 3$ ;BR IF SO
ERROR 6 ;HI ORDER Z SHOULD NOT HAVE INCREMENTED
3$: INC R0 ;SET UP NEXT EXPECTED COUNT
BNE 1$ ;BR IF LOW Z NOT EQ TO 177777
*****
;TEST 16 TEST THAT Z HIGH INCREMENTS ON SPECIAL FUNCT. CONVERT ADC'S
*****
TST16: SCOPE
MOV #4, $TIMES ;DO 4 ITERATIONS
MOV #1$, $LPERR ;SET UP SCOPE LOOP ADRS

```

```

1130 003654 012700 000001          MOV      #1,RO          ;LD EXPECTED IN RO
1131 003660 012777 000020 175674    MOV      #20,ANCSFR     ;SPECIAL FUNCTION CLR Z
1132 003666 012777 177777 175662 1$:    MOV      #-1,ANZLCA     ;ALWAYS SET LO Z FOR CARRY TO HI Z
1133 003674 012777 000004 175660    MOV      #4,ANCSFR     ;START ADC'S
1134 003702 013737 001554 001122    MOV      NCZHW,SBADR    ;SET UP Z/WC REG ADRS
1135 003710 010037 001124          MOV      RO,$GDDAT      ;LD EXPECTED
1136 003714 017737 175634 001126    MOV      ANZHW,SBDDAT   ;READ Z REG HI
1137 003722 023737 001124 001126    CMP      $GDDAT,$BDDAT ;CORRECT?
1138 003730 001401          BEQ      2$            ;BR IF SO
1139 003732 104006          ERROR    6            ;LO ORDER Z FAILED TO INCREMENT TO HI Z
1140 003734 013737 001556 001122 2$:    MOV      NCZLCA,$BADR   ;SET UP Z/CA REG ADRS
1141 003742 005037 001124          CLR      $GDDAT        ;LD EXPECTED
1142 003746 017737 175604 001126    MOV      ANZLCA,$BDDAT ;SEE THAT LO ORDER Z REG WENT TO 0
1143 003754 001401          BEQ      3$            ;BR IF SO
1144 003756 104006          ERROR    6            ;LO ORDER Z SHOULD HAVE INCREMENTED TO 0
1145 003760 005200 3$:    INC      RO            ;SET UP NEXT EXPECTED COUNT
1146 003762 001341          BNE     1$            ;BR IF HI Z NOT EQ TO 177777

```

```

*****
;TEST 17 TEST THAT Z OVFL0 FLAG WILL SET THEN CLR WITH SPECIAL FUNCT CLR Z
*****

```

```

1150
1151 003764 000004          †TST17: SCOPE
1152
1153 003766 012777 000020 175566    MOV      #20,ANCSFR     ;SPECIAL FUNCTION CLR Z/FLG
1154 003774 012777 177777 175554    MOV      #-1,ANZLCA     ;LOAD ALL BITS LO Z
1155 004002 012777 177777 175544    MOV      #-1,ANZHW     ;LOAD ALL BITS H Z
1156 004010 012777 000004 175544    MOV      #4,ANCSFR     ;START ADC'S
1157 004016 013737 001546 001122    MOV      NCCSR,$BADR    ;SET UP CSR REG ADRS
1158 004024 012737 010000 001124    MOV      #BIT12,$GDDAT ;LD EXPECTED
1159 004032 032777 010000 175506    BIT      #BIT12,ANCCSR ;LOOK FOR Z OVERFLOW FLAG
1160 004040 001003          BNE     1$            ;BR IF SET
1161 004042 005037 001126          CLR      $BDDAT        ;SHOW IT WAS NOT SET
1162
1163 004046 104007          ERROR    7            ;Z OVERFLOW FLAG FAILED TO SET
1164 004050 005037 001124          1$:    CLR      $GDDAT        ;LD EXPECTED
1165 004054 012777 000020 175500    MOV      #20,ANCSFR     ;SPECIAL FUNCTION CLR Z FLAG
1166 004062 032777 010000 175456    BIT      #BIT12,ANCCSR ;DID Z OVERFLOW FLAG CLR?
1167 004070 001404          BEQ      TST20         ;NEXT TEST IF SO
1168 004072 012737 010000 001126    MOV      #BIT12,$BDDAT ;SHOW IT IS STILL SET
1169 004100 104007          ERROR    7            ;SPECIAL FUNCTION CLR Z FLG FAILURE

```

```

*****
;TEST 20 TEST THAT RESET WILL CLEAR Z OVERFLOW FLAG
*****

```

```

1170
1171
1172
1173
1174 004102 000004          †TST20: SCOPE
1175 004104 012737 000010 001160    MOV      #10,$TIMES    ;DO 10 ITERATIONS
1176 004112 012777 000020 175442    MOV      #20,ANCSFR     ;SPECIAL FUNCTION CLR Z/FLAG
1177 004120 012777 177777 175430    MOV      #-1,ANZLCA     ;LOAD ALL BITS LO Z
1178 004126 012777 000004 175426    MOV      #4,ANCSFR     ;START ADC'S
1179 004134 013737 001546 001122    MOV      NCCSR,$BADR    ;SET UP CSR REG ADRS
1180 004142 005037 001124          CLR      $GDDAT        ;LD EXPECTED
1181 004146 000005          RESET                    ;I/O CLR
1182 004150 032777 010000 175370    BIT      #BIT12,ANCCSR ;DID Z OVERFLOW FLAG CLR?
1183 004156 001404          BEQ      TST21         ;NEXT TEST IF SO
1184 004160 012737 010000 001126    MOV      #BIT12,$BDDAT ;SHOW IT IS STILL SET
1185 004166 104003          ERROR    3            ;RESET FAILED TO CLEAR Z OVERFLOW FLAG

```

```

1186
1187
1188
1189
1190 004170 000004
1191 004172 012737 000340 177776
1192 004200 012777 000100 175354
1193 004206 012777 004306 175354
1194 004214 013737 001546 001122
1195 004222 012777 000340 175342
1196 004230 012777 177777 175320
1197 004236 012777 177777 175310
1198 004244 012777 004000 175274
1199 004252 012777 000004 175302
1200 004260 013737 001576 177776
1201 004266 017737 175254 001126
1202 004274 042737 167777 001126
1203 004302 104010
1204 004304 000410
1205 004306 022626
1206 004310 032777 010000 175230
1207 004316 001003
1208 004320 005037 001126
1209 004324 104010
1210 004326 013777 001572 175234
1211 004334 005077 175232
1212
1213
1214
1215
1216 004340 000004
1217 004342 012737 000340 177776
1218 004350 012777 004464 175212
1219 004356 013737 001546 001122
1220 004364 012777 000340 175200
1221 004372 012700 000340
1222 004376 012777 000100 175156
1223 004404 012777 177777 175144
1224 004412 012777 177777 175134
1225 004420 012777 004000 175120
1226 004426 012737 010000 001124
1227 004434 012777 000004 175120
1228 004442 010037 177776
1229 004446 021616
1230 004450 162700 000040
1231 004454 023700 001576
1232 004460 001406
1233 004462 000767
1234 004464 022626
1235 004466 017737 175054 001126
1236 004474 104011
1237 004476 013777 001572 175064
1238 004504 005077 175062
1239
1240
1241

```

```

*****
*TEST 21 TEST THAT Z OVERFLOW WILL CAUSE AN INTERRUPT AT CHOSEN BR LEVEL -1
*****

```

```

†ST21: SCOPE
MOV #PR7,PSW ;SET CPU PRIORITY TO HIGHEST
MOV #100,ANCSFR ;CLR ALL
MOV #15,ANCVCT4 ;SET UP INTR RETURN ADRS
MOV NCCSR,$BDADR ;SET UP CSR ADRS
MOV #PR7,ANCVCT6 ;SET UP INTR PRIORITY
MOV #-1,ANCVCT6 ;SET ALL BITS Z LO
MOV #-1,ANCVCT6 ;SET ALL BITS Z HI
MOV #BIT11,ANCCSR ;ENABLE Z OVERFLOW INTR
MOV #4,ANCSFR ;START ADC'S
MOV NCBRL1,PSW ;ALLOW INTR
MOV ANCCSR,$BDDAT ;READ CSR
BIC #167777,$BDDAT ;SAVE ONLY Z OVERFLOW FLAG
ERROR 10 ;Z OVERFLOW FAILED TO INTERRUPT
BR 25 ;GO RESTORE INTR VECTOR
15: CMP (SP)+,(SP)+ ;FIX STACK SINCE NO RTI
BIT #BIT12,ANCCSR ;MAKE SURE Z OVERFLOW FLAG IS SET
BNE 25 ;BR IF SO
CLR $BDDAT ;SHOW IT WAS NOT SET
ERROR 10 ;Z OVERFLOW INTERRUPTED BUT NO FLAG
25: MOV ANCVCT6,ANCVCT4 ;RESTORE VECTOR PTR TO HALT
CLR ANCVCT6 ;LOAD HALT

```

```

*****
*TEST 22 TEST THAT NC11 DOES NOT INTR AT HIGHER BR LEVELS (USE Z OVFL0)
*****

```

```

†ST22: SCOPE
MOV #PR7,PSW ;SET CPU PRIORITY TO HIGHEST
MOV #35,ANCVCT4 ;SET UP INTR RETURN ADRS
MOV NCCSR,$BDADR ;SET UP CSR ADRS
MOV #PR7,ANCVCT6 ;SET UP INTR PRIORITY
MOV #PR7,RO ;GET TOP BR LEVEL
15: MOV #100,ANCSFR ;CLR ALL
MOV #-1,ANCVCT6 ;SET ALL BITS Z LO
MOV #-1,ANCVCT6 ;SET ALL BITS Z HI
MOV #BIT11,ANCCSR ;ENABLE Z OVERFLOW INTR
MOV #BIT12,$BDDAT ;LD EXPECTED
MOV #4,ANCSFR ;START ADC'S
25: MOV RO,PSW ;SET UP PRIORITY
CMP (SP),(SP) ;WASTE TIME
SUB #40,RO ;LOWER LEVEL
CMP NCBRL1,RO ;TO DEVICE LEVEL -1?
BR 25 ;BR IF SO
35: CMP (SP)+,(SP)+ ;TRY THIS LEVEL
MOV ANCCSR,$BDDAT ;FIX STACK SINCE NO RTI - SHOULDN'T HAVE INTERRUPTED
ERROR 11 ;READ CSR
45: MOV ANCVCT6,ANCVCT4 ;NC11 INTERRUPTED AT DEVICE LEVEL OR HIGHER
CLR ANCVCT6 ;RESTORE VECTOR PTR TO HALT
;LOAD HALT

```

```

*****
*TEST 23 TEST FOR PREMATURE Z OVERFLOW INTERRUPT FROM HIGH Z COUNT
*****

```

```

1242
1243 004510 000004
1244 004512 012737 000004 001160
1245 004520 013737 001546 001122
1246 004526 012737 000340 177776
1247 004534 012700 000001
1248 004540 012777 000100 175014
1249 004546 012777 004622 175014
1250 004554 012777 000340 175010
1251 004562 012777 004000 174756
1252 004570 012777 177777 174760 1$:
1253 004576 012777 000004 174756
1254 004604 013737 001576 177776
1255 004612 021616
1256 004614 005200
1257 004616 001364
1258 004620 000407
1259 004622 022626 2$:
1260 004624 005037 001124
1261 004630 017737 174712 001126
1262 004636 104010
1263 004640 013777 001572 174722 3$:
1264 004646 005077 174720
1265
1266
1267
1268
1269 004652 000004
1270 004654 012737 000004 001160
1271 004662 013737 001546 001122
1272 004670 012737 000340 177776
1273 004676 012700 000001
1274 004702 012777 000100 174652
1275 004710 012777 004764 174652
1276 004716 012777 000340 174646
1277 004724 012777 004000 174614
1278 004732 012777 177777 174614 1$:
1279 004740 012777 000004 174614
1280 004746 013737 001576 177776
1281 004754 021616
1282 004756 005200
1283 004760 001364
1284 004762 000405
1285 004764 022626 2$:
1286 004766 017737 174554 001126
1287 004774 104010
1288 004776 013777 001572 174564 3$:
1289 005004 005077 174562
1290
1291
1292
1293
1294 005010 000004
1295 005012 012737 000004 001160
1296 005020 012737 000340 177776
1297 005026 012737 177777 001124

```

```

*****
*TEST 23: SCOPE
MOV #4,$TIMES ;:DO 4 ITERATIONS
MOV NCCSR,$BDADR ;:SET UP CSR ADRS
MOV #PR7,PSW ;:SET CPU PRIORITY TO HIGHEST
MOV #1,R0 ;:R0 WILL COUNT HI Z
MOV #100,$NCSFR ;:CLR ALL
MOV #2,$$,$NCVCT4 ;:SET UP INTR RETURN ADRS
MOV #PR7,$NCVCT6 ;:SET UP INTR PRIORITY
MOV #BIT11,$NCCSR ;:ENABLE Z OVERFLOW INTR
1$: MOV #-1,$NCZLCA ;:ALWAYS SET LOW ORDER Z BITS
MOV #4,$NCSFR ;:START ADC'S
MOV NCBRL1,PSW ;:ALLOW A POTENTIAL INTR
CMP (SP),(SP) ;:STALL
INC R0 ;:COUNT ALONG WITH HI Z REG
BNE 1$ ;:BR IF NOT A FULL 16 BIT COUNT YET
BR 3$ ;:GO RESTORE INTR VECTOR
2$: CMP (SP)+,(SP)+ ;:FIX STACK SINCE NO RTI - SHOULDN'T HAVE INTR'ED
CLR $GDDAT ;:LD EXPECTED
MOV $NCCSR,$BDDAT ;:READ CSR
ERROR 10 ;:Z OVERFLOW INTERRUPTED ON LESS THAN FULL COUNT
3$: MOV $NCVCT6,$NCVCT4 ;:RESTORE VECTOR PTR TO HALT
CLR $NCVCT6 ;:LOAD HALT

```

```

*****
*TEST 24 TEST FOR PREMATURE Z OVERFLOW INTERRUPT FROM LOW Z COUNT
*****
*TEST 24: SCOPE
MOV #4,$TIMES ;:DO 4 ITERATIONS
MOV NCCSR,$BDADR ;:SET UP CSR ADRS
MOV #PR7,PSW ;:SET CPU PRIORITY TO HIGHEST
MOV #1,R0 ;:R0 WILL COUNT LO Z
MOV #100,$NCSFR ;:CLR ALL
MOV #2,$$,$NCVCT4 ;:SET UP INTR RETURN ADRS
MOV #PR7,$NCVCT6 ;:SET UP INTR PRIORITY
MOV #BIT11,$NCCSR ;:ENABLE Z OVERFLOW INTR
1$: MOV #-1,$NCZHWC ;:ALWAYS SET HIGH ORDER Z BITS
MOV #4,$NCSFR ;:START ADC'S
MOV NCBRL1,PSW ;:ALLOW A POTENTIAL INTR
CMP (SP),(SP) ;:STALL
INC R0 ;:COUNT ALONG WITH LO Z REG
BNE 1$ ;:BR IF NOT A FULL 16 BIT COUNT YET
BR 3$ ;:GO RESTORE INTR VECTOR
2$: CMP (SP)+,(SP)+ ;:FIX STACK SINCE NO RTI - SHOULDN'T HAVE INTR'ED
MOV $NCCSR,$BDDAT ;:READ CSR
ERROR 10 ;:Z OVERFLOW INTERRUPTED ON LESS THAN FULL COUNT
3$: MOV $NCVCT6,$NCVCT4 ;:RESTORE VECTOR PTR TO HALT
CLR $NCVCT6 ;:LOAD HALT

```

```

*****
*TEST 25 TEST A LIST MODE NPR TO ADRS 20000 ALL DATA PATTERNS
*****
*TEST 25: SCOPE
MOV #4,$TIMES ;:DO 4 ITERATIONS
MOV #PR7,PSW ;:DON'T WANT ANY INTR'S
MOV #-1,$GDDAT ;:LD EXPECTED

```

```

1298 005034 012700 020000      MOV      #20000,R0      ;SET UP WORKING ADRS IN R0
1299 005040 010037 001122      MOV      R0,$B0ADR    ;SET UP LIST ADRS
1300 005044 012777 000100 174510 1$:  MOV      #100,$NCSFR  ;CLR ALL
1301 005052 012777 000405 174466  MOV      #405,$NCCSR  ;SELECT LIST MD + NPR ENABLE
1302 005060 012777 177777 174466  MOV      #-1,$NCZHWC  ;SET UP WC IN CASE IT GETS AWAY
1303 005066 010077 174464      MOV      R0,$NCZLCA  ;SET UP CA
1304 005072 005010      CLR      (R0)        ;MAKE WORKING ADRS ZERO
1305 005074 013737 001124 013226  MOV      $GDDAT,XYHOLD ;DATA WRITTEN IN LIST MD TO XYHOLD REG
1306 005102 013777 001124 174442  MOV      $GDDAT,$NCXYH ;CAUSE A LIST TO MEMORY
1307 005110 012737 005044 001110  MOV      #1,$SLPERR   ;SET UP SCOPE LOOP ADRS
1308 005116 011037 001126      MOV      (R0),$BDDAT ;READ DATA WRITTEN TO CORE
1309 005122 023737 001124 001126  CMP      $GDDAT,$BDDAT ;CORRECT?
1310 005130 001401      BEQ     2$          ;BR IF SO
1311 005132 104012      ERROR   12         ;LIST MODE NPR FAILURE, CHECK DATA
1312
1313
1314 005134 005337 001124      2$:  DEC     $GDDAT      ;AND CA PATHS - IF ADRS BITS BAD, A
1315 005140 102341      BVC     1$         ;PROGRAM LOC COULD BE DESTROYED.
1316
1317
1318
1319
1320 005142 000004      ;*****
1321 005144 012737 000010 001160  ;*TEST 26 TEST LIST MODE FOR PROPER CA & WC INCREMENTING WITH DATA & WCO CHECKS
1322 005152 012737 000340 177776  ;*****
1323 005160 012700 020000      ;ST26: SCOPE
1324 005164 012702 125252      MOV      #10,$TIMES   ;DO 10 ITERATIONS
1325 005170 012701 160000      MOV      #PR7,PSW     ;DON'T WANT ANY INTR'S
1326 005174 012777 000100 174360 1$:  MOV      #20000,R0    ;START AT LOC 20000
1327 005202 010077 174350      MOV      #125252,R2   ;INITIALIZE DATA PATTERN
1328 005206 010177 174342      MOV      #-8192,R1    ;DO 40000 XFRS (8)
1329 005212 012777 000405 174326  MOV      #100,$NCSFR  ;CLR ALL
1330 005220 012777 000001 174334 2$:  MOV      R0,$NCZLCA  ;SET UP CA AT 20000
1331 005226 005010      MOV      R1,$NCZHWC  ;SET UP WC
1332 005230 010277 174316      MOV      #405,$NCCSR ;SELECT LIST MD + NPR ENABLE
1333 005234 010237 001124      CLR     (R0)        ;CLR HOLD REG
1334 005240 010237 013226      MOV      R2,$GDDAT   ;ZERO CA LOC
1335 005244 010037 001122      MOV      R2,XYHOLD   ;CAUSE A LIST NPR TO MEM
1336 005250 011037 001126      MOV      R0,$B0ADR  ;LD EXPECTED DATA
1337 005254 023737 001124 001126  MOV      (R0),$BDDAT ;DATA WRITTEN IN LIST MD TO XYHOLD REG
1338 005262 001401      CMP      $GDDAT,$BDDAT ;SET UP CURRENT LIST ADRS
1339 005264 104012      BEQ     3$         ;GET DATA WRITTEN TO CORE
1340
1341 005266 010037 001124      3$:  MOV      R0,$GDDAT   ;CORRECT?
1342 005272 062737 000002 001124  ADD     #2,$GDDAT    ;BR IF SO
1343 005300 017737 174252 001126  MOV     $NCZLCA,$BDDAT ;LIST MODE NPR DATA FAILURE, CHECK DATA
1344 005306 023737 001124 001126  CMP     $GDDAT,$BDDAT ;AND CA PATHS
1345 005314 001404      BEQ     4$         ;LD EXPECTED CA
1346 005316 013737 001556 001122  MOV     NCZLCA,$B0ADR ;SHOULD BE TWO GREATER THAN BEFORE LAST NPR LIST
1347 005324 104013      ERROR   13         ;GET CA
1348 005326 010137 001124      4$:  MOV      R1,$GDDAT   ;CORRECT?
1349 005332 005237 001124      INC     $GDDAT      ;BR IF SO
1350 005336 017737 174212 001126  MOV     $NCZHWC,$BDDAT ;SET UP CA REG ADRS
1351 005344 023737 001124 001126  CMP     $GDDAT,$BDDAT ;LIST MODE CA INCREMENT FAILURE
1352 005352 001404      BEQ     5$         ;LD EXPECTED WC
1353 005354 013737 001554 001122  MOV     NCZHWC,$B0ADR ;SHOULD BE ONE GREATER THAN BEFORE LAST NPR LIST
1354
1355

```

E03

MAINDEC-11-DZPCA-C NC11-A LOGIC TEST MACY11 27(732) 21-SEP-76 14:14 PAGE 30
DZPCA.P11 T26 TEST LIST MODE FOR PROPER CA & WC INCREMENTING WITH DATA & WCO CHECKS

```

1354 005362 104014          ERROR 14          ;LIST MODE WC INCREMENT FAILURE
1355 005364 013737 001546 001122 5$:  MOV    NCCSR,$BDADR ;SET UP CSR REG ADRS
1356 005372 005737 001124          TST    $GDDAT      ;WCO EXPECTED?
1357 005376 001413          BEQ    6$          ;BR IF SO
1358 005400 032777 010000 174140  BIT    #BIT12,$NCCSR ;WCO FLAG SET WHEN NOT EXPECTED?
1359 005406 001422          BEQ    7$          ;BR IF NOT
1360 005410 005037 001124          CLR    $GDDAT      ;LD EXPECTED
1361 005414 012737 010000 001126  MOV    #BIT12,$BDDAT ;SHOW IT WAS SET
1362 005422 104015          ERROR 15          ;WC OVERFLOW FLAG SET PREMATURELY
1363 005424 000413          BR     7$          ;CONTINUE NEXT TO NEXT LIST NPR
1364 005426 032777 010000 174112 6$:  BIT    #BIT12,$NCCSR ;WCO BIT SET WHEN EXPECTED?
1365 005434 001014          BNE    8$          ;BR IF SO
1366 005436 012737 010000 001124  MOV    #BIT12,$GDDAT ;LD EXPECTED WCO FLAG
1367 005444 005037 001126          CLR    $BDDAT      ;SHOW IT WAS NOT SET
1368 005450 104015          ERROR 15          ;WC OVERFLOW FLAG FAILED TO SET ON WCO
1369 005452 000420          BR     TST27       ;NEXT TEST IF NO LOOP ON ER
1370 005454 005102          7$:  COM    R2          ;COMPLEMENT NPR DATA
1371 005456 005201          INC    R1          ;ADVANCE WC
1372 005460 062700 000002          ADD    #2,R0       ;BUMP CA
1373 005464 000655          BR     2$          ;DO LIST TO NEXT ADRS
1374 005466 000005          8$:  RESET          ;CLR WCO
1375 005470 032777 010000 174050  BIT    #BIT12,$NCCSR ;DID RESET CLR WCO FLAG?
1376 005476 001406          BEQ    TST27       ;NEXT TEST IF CLRED
1377 005500 005037 001124          CLR    $GDDAT      ;LD EXPECTED
1378 005504 012737 010000 001126  MOV    #BIT12,$BDDAT ;INDICATE IT WAS SET
1379 005512 104015          ERROR 15          ;WCO FLAG FAILED TO CLR WITH RESET

```

```

1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399

```

```

*****
;TEST 27 TEST SPECIAL FUNCTION CLEAR WCO FLAG
*****

```

```

1384 005514 000004          TST27: SCOPE
1385 005516 012737 000340 177776  MOV    #PR7,PSW    ;DON'T WANT WCO INTR
1386 005524 012777 000100 174030  MOV    #100,$NCSFR ;CLR ALL
1387 005532 012777 020000 174016  MOV    #20000,$NCZLCA ;SET UP CA AT 20000
1388 005540 012777 177777 174006  MOV    #-1,$NCZHWC  ;SET UP WC FOR ONE XFER
1389 005546 012777 000405 173772  MOV    #405,$NCCSR  ;SELECT LIST MD + NPR ENABLE
1390 005554 005077 173772          CLR    $NCXYH      ;CAUSE A LIST TO MEM
1391 005560 013737 001546 001122  MOV    NCCSR,$BDADR ;SET UP CSR ADRS
1392 005566 012777 000020 173766  MOV    #20,$NCSFR  ;USE SPECIAL FUNCTION CLR WCO FLAG
1393 005574 032777 010000 173760  BIT    #BIT12,$NCSFR ;DID IT CLR?
1394 005602 001401          BEQ    TST30       ;NEXT TEST IF SO
1395 005604 104015          ERROR 15          ;WCO FLAG FAILED TO CLR WITH SPECIAL FUNCTION CLR.

```

```

1396
1397
1398
1399

```

```

*****
;TEST 30 TEST THAT WCO WILL CAUSE AN INTERRUPT
*****

```

```

1400 005606 000004          TST30: SCOPE
1401 005610 012737 000340 177776  MOV    #PR7,PSW    ;SET PRIORITY TO HIGHER
1402 005616 012777 000100 173736  MOV    #100,$NCSFR ;CLR ALL
1403 005624 012777 005752 173736  MOV    #25,$NCVCT4  ;SET UP INTR SERVICE ADRS
1404 005632 012777 000340 173732  MOV    #PR7,$NCVCT6 ;SET UP INTR PRIORITY
1405 005640 012777 004405 173700  MOV    #4405,$NCCSR ;ENABLE WCO, LIST MD + NPR ENABLE
1406 005646 012777 177776 173700  MOV    #-2,$NCZHWC  ;SET UP WC FOR 2 XFERS
1407 005654 012777 020000 173674  MOV    #20000,$NCZLCA ;SET UP CA AT ADRS 20000
1408 005662 012700 000002          MOV    #2,R0       ;R0 WILL COUNT NPR'S
1409 005666 005077 173660 15:  CLR    $NCXYH      ;CAUSE A LIST TO MEM

```

```

1410 005672 013737 001546 001122      MOV      NCCSR,$BDADR      ;SET UP CSR ADRS
1411 005700 013737 001576 177776      MOV      NCBRL1,PSW      ;ALLOW INTR ANYTIME
1412 005706 021616                    CMP      (SP),(SP)        ;STALL
1413 005710 021616                    CMP      (SP),(SP)        ;STALL SOMEMORE
1414 005712 012777 000001 173642      MOV      #1,$NCSFR        ;SPECIAL FUNCTION CLR X Y HOLD REG
1415 005720 005300                    DEC      R0                ;DID WE EXPECT AN INTR?
1416 005722 001361                    BNE     1$                ;BR IF NOT
1417 005724 012737 010000 001124      MOV      #BIT12,$GDDAT    ;LD EXPECTED
1418 005732 017737 173610 001126      MOV      $NCCSR,$BDDAT    ;READ CSR
1419 005740 042737 167777 001126      BIC     #167777,$BDDAT    ;SAVE ONLY WCO FLAG
1420 005746 104015                    ERROR   1$                ;WCO FAILED TO INTERRUPT
1421 005750 000433                    BR      4$                ;SET UP EXIT TEST
1422 005752 022626                    2$:    CMP      (SP)+,(SP)+  ;FIX STACK SINCE NO RTI
1423 005754 005300                    DEC      R0                ;CHECK THAT THE INTR WAS EXPECTED
1424 005756 001412                    BEQ     3$                ;BR IF SO
1425 005760 005037 001124                    CLR     $GDDAT            ;DID NOT WANT WCO INTR UNTIL 2ND XFER
1426 005764 017737 173556 001126      MOV      $NCCSR,$BDDAT    ;READ CSR
1427 005772 042737 167777 001126      BIC     #167777,$BDDAT    ;SAVE ONLY FLAG
1428 006000 104015                    ERROR   1$                ;WCO INTERRUPTED BEFORE LAST XFER
1429 006002 000416                    BR      4$                ;WCO INTERRUPTED BEFORE LAST XFER
1430 006004 012737 010000 001124      3$:    MOV      #BIT12,$GDDAT    ;LD EXPECTED
1431 006012 017737 173530 001126      MOV      $NCCSR,$BDDAT    ;READ CSR
1432 006020 042737 167777 001126      BIC     #167777,$BDDAT    ;SAVE ONLY WCO FLAG
1433 006026 023737 001124 001126      CMP     $GDDAT,$BDDAT     ;CORRECT?
1434 006034 001401                    BEQ     4$                ;BR IF SO
1435 006036 104015                    ERROR   1$                ;WCO FLAG FAILED TO SET ON WCO ON INTR
1436 006040 013777 001572 173522      4$:    MOV      NCVCT6,$NCVCT4  ;RESTORE VECTOR PTR TO HALT
1437 006046 005077 173520                    CLR     $NCVCT6          ;LOAD HALT
1438
1439
1440
1441
1442 006052 000004                    ;*****
1443 006054 012737 006122 001110      ;*TEST 31      TEST JOY STICK MODE FOR CORRECT ADDRESS MAKE
1444 006062 005037 013232                    ;*****
1445 006066 005037 013212                    TST31:  SCOPE
1446 006072 012777 000100 173462      MOV      #2$,$SLPERR      ;SET UP SCOPE LOOP ADRS
1447 006100 005037 013234                    CLR     TAM                ;SELECT NO NPR
1448 006104 012737 000010 013236      CLR     DUAL              ;NO B GAMMA
1449 006112 004737 012714                    MOV      #100,$NCSFR      ;CLR ALL
1450 006116 004737 012746                    CLR     OFFSET            ;NO OFFSET IN JOY STICK MD
1451 006122 004737 013050                    1$:    MOV      #10,$OFCR      ;SELECT JOY STICK MODE
1452 006126 000405                    2$:    JSR     PC,REGLD      ;GO LOAD CSR
1453 006130 023737 001120 001122      JSR     PC,MAKE           ;GO MAKE UP ADRS
1454 006136 001767                    JSR     PC,MKLOOP         ;GO MAKE UP SAME ADRS (SCOPE LOOP)
1455 006140 104017                    BR      TST32            ;NEXT TEST - ALL ADRS MADE
1456
1457
1458
1459
1460 006142 000004                    ;*****
1461 006144 012737 006204 001110      ;*TEST 32      TEST JOYSTICK MODE FOR ABILITY TO OVERRIDE OTHER FUNCTIONS
1462 006152 012777 000100 173402                    ;*****
1463 006160 012737 177777 013234      TST32:  SCOPE
1464 006166 012737 003415 013236      MOV      #2$,$SLPERR      ;SET UP SCOPE LOOP ADRS
1465 006174 004737 012714                    MOV      #100,$NCSFR      ;CLR ALL
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499

```


1466	006200	004737	012746	1\$:	JSR	PC,MAKE	:GO MAKE UP ADRS
1467	006204	004737	013050	2\$:	JSR	PC,MKLOOP	:GO MAKE UP SAME ADRS (SCOPE LOOP)
1468	006210	000405			BR	TST33	:NEXT TEST - ALL ADRS MADE
1469	006212	023737	001120 001122		CMP	\$GDADR,\$BDADR	:ADRS CORRECT?
1470	006220	001767			BEQ	1\$:TRY NEXT ADRS IF SO
1471	006222	104020			ERROR	20	:JOYSTICK MD FAILED TO OVERRIDE LIST MD,
1472							: OFFSET REG, OR RESOLUTION LOGIC

 : *TEST 33 TEST RESOLUTION 0 TO YIELD 0
 : *****

1477	006224	000004		TST33:	SCOPE		
1478	006226	005037	013232		CLR	TAM	:NO NPR ENABLE THIS TEST
1479	006232	012777	000100 173322		MOV	#100,\$NCSFR	:CLR ALL
1480	006240	012737	006266 001110		MOV	#3\$,\$LPERR	:SET UP SCOPE LOOP ADRS
1481	006246	005037	013236		CLR	SOFCSR	:SELECT NO MODE OR RESOLUTION
1482	006252	005037	013234		CLR	OFFSET	:SELECT 0 OFFSET INITIALLY
1483	006256	004737	012714	1\$:	JSR	PC,REGLD	:GO LOAD CSR & OFFSET REGS
1484	006262	004737	012746	2\$:	JSR	PC,MAKE	:GO MAKE UP ADRS
1485	006266	004737	013050	3\$:	JSR	PC,MKLOOP	:GO MAKE UP SAME ADRS (SCOPE LOOP)
1486	006272	000405			BR	4\$:GO UPDATE OFFSET
1487	006274	023737	001120 001122		CMP	\$GDADR,\$BDADR	:ADRS CORRECT?
1488	006302	001767			BEQ	2\$:TRY NEXT ADRS IF SO
1489	006304	104017			ERROR	17	:RESOLUTION 0 SHOULD HAVE MADE ADRS 0
1490	006306	062737	170000 013234	4\$:	ADD	#170000,OFFSET	:SET ALL OFFSET BITS SHOULD HAVE NO EFFECT
1491	006314	103360			BCC	1\$:NOW REPEAT WITH OFFSET BITS SET

 : *TEST 34 TEST RESOLUTION 1 - LIST MATRIX
 : *****

1496	006316	000004		TST34:	SCOPE		
1497	006320	005037	013232		CLR	TAM	:NO NPR ENABLE THIS TEST
1498	006324	012777	000100 173230		MOV	#100,\$NCSFR	:CLR ALL
1499	006332	012737	006362 001110		MOV	#3\$,\$LPERR	:SET UP SCOPE LOOP ADRS
1500	006340	005037	013234		CLR	OFFSET	:SELECT 0 OFFSET INITIALLY
1501	006344	012737	000400 013236		MOV	#400,SOFCSR	:SELECT LIST MATRIX
1502	006352	004737	012714	1\$:	JSR	PC,REGLD	:GO LOAD CSR & OFFSET
1503	006356	004737	012746	2\$:	JSR	PC,MAKE	:GO MAKE UP ADRS
1504	006362	004737	013050	3\$:	JSR	PC,MKLOOP	:GO MAKE UP SAME ADRS (SCOPE LOOP)
1505	006366	000405			BR	4\$:GO UPDATE OFFSET
1506	006370	023737	001120 001122		CMP	\$GDADR,\$BDADR	:ADRS CORRECT?
1507	006376	001767			BEQ	2\$:TRY NEXT ADRS IF SO
1508	006400	104017			ERROR	17	:LIST MATRIX ADRS MAKE ER
1509	006402	062737	170000 013234	4\$:	ADD	#170000,OFFSET	:SET ALL OFFSET BITS - SHOULD HAVE NO EFFECT
1510	006410	103360			BCC	1\$:NOW REPEAT WITH OFFSET BITS SET

 : *TEST 35 TEST RESOLUTION 2 - 64X64X16
 : *****

1515	006412	000004		TST35:	SCOPE		
1516	006414	012737	000100 001160		MOV	#100,\$TIMES	:DO 100 ITERATIONS
1517	006422	012737	006466 001110		MOV	#4\$,\$LPERR	:SET UP SCOPE LOOP ADRS
1518	006430	005037	013232		CLR	TAM	:SELECT NO NPR ENABLE INITIALLY
1519	006434	012777	000100 173120	1\$:	MOV	#100,\$NCSFR	:CLR ALL
1520	006442	012737	020000 013234		MOV	#20000,OFFSET	:SET UP BASE
1521	006450	012737	001000 013236		MOV	#1000,SOFCSR	:SELECT 64X64X16

H03

1522	006456	004737	012714	2\$:	JSR	PC,REGLD	;GO LOAD CSR + OFFSET
1523	006462	004737	012746	3\$:	JSR	PC,MAKE	;GO DO ADRS MAKE
1524	006466	004737	013050	4\$:	JSR	PC,MKLOOP	;ADRS MAKE SCOPE LOOP
1525	006472	000417			BR	6\$;RETURN HERE IF ALL ADRS MADE
1526	006474	005737	013232		TST	TAM	;WAS NPR ENABLED?
1527	006500	001006			BNE	5\$;BR IF SO
1528	006502	023737	001120 001122		CMP	\$GDADR,\$BDADR	;ADRS CORRECT?
1529	006510	001764			BEQ	3\$;TRY NEXT ADRS IF SO
1530	006512	104017			ERROR	17	;ADRS MAKE ER 64X64X16
1531	006514	000422			BR	TST36	;TO NEXT TEST
1532	006516			5\$:			
1533	006516	023737	001124 001126		CMP	\$GDDAT,\$BDDAT	;CORRECT?
1534	006524	001756			BEQ	3\$;TRY NEXT ADRS IF SO
1535	006526	104021			ERROR	21	;NPR INCREMENT ER AT ADRS INDICATED
1536	006530	000414			BR	TST36	;TO NEXT TEST
1537	006532	062737	020000 013234	6\$:	ADD	#20000,OFFSET	;ADVANCE BASE ADRS
1538	006540	103405			BCS	7\$;BR IF 15 BITS TESTED(NON-NPR)
1539	006542	100345			BPL	2\$;BR IF 14 BITS NOT TESTED
1540	006544	005737	013232		TST	TAM	;ALLOW ONLY 14 BITS IF NPR
1541	006550	001004			BNE	TST36	;NEXT TEST IF NPR TESTED TO 16K
1542	006552	000741			BR	2\$;GO TO 15 BITS SINCE NON-NPR
1543	006554	005237	013232	7\$:	INC	TAM	;SELECT NPR
1544	006560	000725			BR	1\$;REPEAT TEST(NPR ENABLED)

1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577

```
*****  
*TEST 36 TEST RESOLUTION 3 - 32X32X16  
*****  
TST36: SCOPE  
MOV #20,$TIMES ;DO 20 ITERATIONS  
MOV #4,$SLPERR ;SET UP SCOPE LOOP ADRS  
CLR TAM ;SELECT NO NPR ENABLE INITIALLY  
1$: MOV #100,$NCSFR ;CLR ALL  
MOV #20000,OFFSET ;SET UP BASE  
MOV #1400,$OFCR ;SELECT 32X32X16  
2$: JSR PC,REGLD ;GO LOAD CSR + OFFSET  
3$: JSR PC,MAKE ;GO DO ADRS MAKE  
4$: JSR PC,MKLOOP ;ADRS MAKE SCOPE LOOP  
BR 6$ ;RETURN HERE IF ALL ADRS MADE  
TST TAM ;WAS NPR ENABLED?  
BNE 5$ ;BR IF SO  
CMP $GDADR,$BDADR ;ADRS CORRECT?  
BEQ 3$ ;TRY NEXT ADRS IF SO  
ERROR 17 ;ADRS MAKE ER 32X32X16  
BR TST37 ;TO NEXT TEST  
5$:  
CMP $GDDAT,$BDDAT ;CORRECT?  
BEQ 3$ ;TRY NEXT ADRS IF SO  
ERROR 21 ;NPR INCREMENT ER AT ADRS INDICATED  
BR TST37 ;TO NEXT TEST  
6$:  
ADD #4000,OFFSET ;ADVANCE BASE ADRS  
BCS 7$ ;BR IF 15 BITS TESTED(NON-NPR)  
BPL 2$ ;BR IF 14 BITS NOT TESTED  
TST TAM ;ALLOW ONLY 14 BITS IF NPR  
BNE TST37 ;NEXT TEST IF NPR TESTED TO 16K  
BR 2$ ;GO TO 15 BITS SINCE NON-NPR  
7$:  
INC TAM ;SELECT NPR
```

```

1578 006730 000725          BR      1$          ;REPEAT TEST(NPR ENABLED)
1579
1580
1581          ;*****
1582          ;*TEST 37      TEST RESOLUTION 4 - 128X128X8
1583          ;*****
1583 006732 000004          TST37:  SCOPE
1584 006734 012737 000200 001160          MOV      #200,$TIMES          ;;DO 200 ITERATIONS
1585 006742 012737 007006 001110          MOV      #4$,$LPERR          ;;SET UP SCOPE LOOP ADRS
1586 006750 005037 013232          CLR      TAM                  ;;NO NPR ENABLE THIS TEST
1587 006754 012777 000100 172600 1$:  MOV      #100,$NCSFR          ;;CLR ALL
1588 006762 012737 040000 013234          MOV      #40000,$OFFSET      ;;SET UP BASE ADRS
1589 006770 012737 002000 013236          MOV      #2000,$OFCSR         ;;SELECT 128X128X8
1590 006776 004737 012714          2$:  JSR      PC,REGLD          ;;GO LOAD CSR + OFFSET
1591 007002 004737 012746          3$:  JSR      PC,MAKE           ;;GO DO ADRS MAKE
1592 007006 004737 013050          4$:  JSR      PC,MKLOOP        ;;ADRS MAKE SCOPE LOOP
1593 007012 000417          BR      6$                    ;;RETURN HERE IF ALL ADRS MADE
1594 007014 005737 013232          TST      TAM                  ;;WAS NPR ENABLED?
1595 007020 001006          BNE     5$                    ;;BR IF SO
1596 007022 023737 001120 001122          CMP      $GDADR,$BDADR        ;;ADRS CORRECT?
1597 007030 001764          BEQ     3$                    ;;TRY NEXT ADRS IF SO
1598 007032 104017          ERROR   17                    ;;ADRS MAKE ER 128X128X8
1599 007034 000422          BR      TST40                 ;;TO NEXT TEST
1600 007036
1601 007036 023737 001124 001126 5$:  CMP      $GDDAT,$BDDAT        ;;CORRECT?
1602 007044 001756          BEQ     3$                    ;;TRY NEXT ADRS IF SO
1603 007046 104021          ERROR   21                    ;;NPR INCREMENT ER AT ADRS INDICATED
1604 007050 000414          BR      TST40                 ;;TO NEXT TEST
1605 007052 062737 040000 013234 6$:  ADD      #40000,$OFFSET        ;;ADVANCE BASE ADRS
1606 007060 103405          BCS     7$                    ;;BR IF 15 BITS TESTED(NON-NPR)
1607 007062 100345          BPL     2$                    ;;BR IF 14 BITS NOT TESTED
1608 007064 005737 013232          TST      TAM                  ;;ALLOW ONLY 14 BITS IF NPR
1609 007070 001004          BNE     TST40                 ;;NEXT TEST IF NPR TESTED TO 16K
1610 007072 000741          BR      2$                    ;;GO TO 15 BITS SINCE NON-NPR
1611 007074 005237 013232 7$:  INC      TAM                  ;;SELECT NPR
1612 007100 000725          BR      1$                    ;;REPEAT TEST(NPR ENABLED)
1613
1614          ;*****
1615          ;*TEST 40      TEST RESOLUTION 5 - 128X128X8 (4K BASE)
1616          ;*****
1617 007102 000004          TST40:  SCOPE
1618 007104 012737 000200 001160          MOV      #200,$TIMES          ;;DO 200 ITERATIONS
1619 007112 012737 007156 001110          MOV      #4$,$LPERR          ;;SET UP SCOPE LOOP ADRS
1620 007120 005037 013232          CLR      TAM                  ;;SELECT NO NPR ENABLE INITIALLY
1621 007124 012777 000100 172430 1$:  MOV      #100,$NCSFR          ;;CLR ALL
1622 007132 012737 000000 013234          MOV      #0,$OFFSET          ;;SET UP BASE
1623 007140 012737 002400 013236          MOV      #2400,$OFCSR         ;;SELECT 128X128X8 (4K BASE)
1624 007146 004737 012714          2$:  JSR      PC,REGLD          ;;GO LOAD CSR + OFFSET
1625 007152 004737 012746          3$:  JSR      PC,MAKE           ;;GO DO ADRS MAKE
1626 007156 004737 013050          4$:  JSR      PC,MKLOOP        ;;ADRS MAKE SCOPE LOOP
1627 007162 000417          BR      6$                    ;;RETURN HERE IF ALL ADRS MADE
1628 007164 005737 013232          TST      TAM                  ;;WAS NPR ENABLED?
1629 007170 001006          BNE     5$                    ;;BR IF SO
1630 007172 023737 001120 001122          CMP      $GDADR,$BDADR        ;;ADRS CORRECT?
1631 007200 001764          BEQ     3$                    ;;TRY NEXT ADRS IF SO
1632 007202 104017          ERROR   17                    ;;ADRS MAKE ER 128X128X8 (4K BASE)
1633 007204 000422          BR      TST41                 ;;TO NEXT TEST

```

```

1634 007206          5$:
1635 007206 023737 001124 001126  CMP      $GDDAT,$BDDAT  ;CORRECT?
1636 007214 001756          BEQ      3$           ;TRY NEXT ADRS IF SO
1637 007216 104021          ERROR    21           ;NPR INCREMENT ER AT ADRS INDICATED
1638 007220 000414          BR       TST41        ;TO NEXT TEST
1639 007222 062737 100000 013234 6$:  ADD      #100000,OFFSET ;ADVANCE BASE ADRS
1640 007230 103405          BCS     7$           ;BR IF 15 BITS TESTED(NON-NPR)
1641 007232 100345          BPL     2$           ;BR IF 14 BITS NOT TESTED
1642 007234 005737 013232  TST     TAM          ;ALLOW ONLY 14 BITS IF NPR
1643 007240 001004          BNE     TST41        ;NEXT TEST IF NPR TESTED TO 16K
1644 007242 000741          BR      2$           ;GO TO 15 BITS SINCE NON-NPR
1645 007244 005237 013232  7$:  INC     TAM          ;SELECT NPR
1646 007250 000725          BR      1$           ;REPEAT TEST(NPR ENABLED)

```

```

*****
;TEST 41      TEST RESOLUTION 6 - 64X64X8
*****

```

```

1651 007252 000004          †TST41: SCOPE
1652 007254 012737 000040 001160  MOV     #40,$TIMES    ;DO 40 ITERATIONS
1653 007262 012737 007326 001110  MOV     #4$,SLPERR    ;SET UP SCOPE LOOP ADRS
1654 007270 005037 013232          CLR     TAM          ;SELECT NO NPR ENABLE INITIALLY
1655 007274 012777 000100 172260  1$:  MOV     #100,$NCSFR   ;CLR ALL
1656 007302 012737 020000 013234  MOV     #20000,OFFSET ;SET UP BASE
1657 007310 012737 003000 013236  MOV     #3000,$OFCR   ;SELECT 64X64X8
1658 007316 004737 012714          JSR    PC,REGLD      ;GO LOAD CSR + OFFSET
1659 007322 004737 012746          JSR    PC,MAKE       ;GO DO ADRS MAKE
1660 007326 004737 013050          JSR    PC,MKLOOP     ;ADRS MAKE SCOPE LOOP
1661 007332 000417          BR     6$           ;RETURN HERE IF ALL ADRS MADE
1662 007334 005737 013232  TST     TAM          ;WAS NPR ENABLED?
1663 007340 001006          BNE     5$           ;BR IF SO
1664 007342 023737 001120 001122  CMP     $GDADR,$BDADR ;ADRS CORRECT?
1665 007350 001764          BEQ     3$           ;TRY NEXT ADRS IF SO
1666 007352 104017          ERROR  17           ;ADRS MAKE ER 64X64X8
1667 007354 000422          BR     TST42        ;TO NEXT TEST

```

```

1668 007356          5$:
1669 007356 023737 001124 001126  CMP     $GDDAT,$BDDAT ;CORRECT?
1670 007364 001756          BEQ     3$           ;TRY NEXT ADRS IF SO
1671 007366 104021          ERROR  21           ;NPR INCREMENT ER AT ADRS INDICATED
1672 007370 000414          BR     TST42        ;TO NEXT TEST
1673 007372 062737 010000 013234 6$:  ADD     #10000,OFFSET ;ADVANCE BASE ADRS
1674 007400 103405          BCS    7$           ;BR IF 15 BITS TESTED(NON-NPR)
1675 007402 100345          BPL    2$           ;BR IF 14 BITS NOT TESTED
1676 007404 005737 013232  TST    TAM          ;ALLOW ONLY 14 BITS IF NPR
1677 007410 001004          BNE    TST42        ;NEXT TEST IF NPR TESTED TO 16K
1678 007412 000741          BR     2$           ;GO TO 15 BITS SINCE NON-NPR
1679 007414 005237 013232  7$:  INC    TAM          ;SELECT NPR
1680 007420 000725          BR     1$           ;REPEAT TEST(NPR ENABLED)

```

```

*****
;TEST 42      TEST RESOLUTION 7 - 32X32X8
*****

```

```

1685 007422 000004          †TST42: SCOPE
1686 007424 012737 000020 001160  MOV     #20,$TIMES    ;DO 20 ITERATIONS
1687 007432 012737 007476 001110  MOV     #4$,SLPERR    ;SET UP SCOPE LOOP ADRS
1688 007440 005037 013232          CLR     TAM          ;SELECT NO NPR ENABLE INITIALLY
1689 007444 012777 000100 172110  1$:  MOV     #100,$NCSFR   ;CLR ALL

```

```

1690 007452 012737 020000 013234      MOV      #20000,OFFSET      ;SET UP BASE
1691 007460 012737 003400 013236      MOV      #3400,$OFCSR     ;SELECT 32X32X8
1692 007466 004737 012714          2$:     JSR      PC,REGLD     ;GO LOAD CSR + OFFSET
1693 007472 004737 012746          3$:     JSR      PC,MAKE     ;GO DO ADRS MAKE
1694 007476 004737 013050          4$:     JSR      PC,MKLOOP   ;ADRS MAKE SCOPE LOOP
1695 007502 000417          BR       6$              ;RETURN HERE IF ALL ADRS MADE
1696 007504 005737 013232          TST     TAM              ;WAS NPR ENABLED?
1697 007510 001006          BNE     5$              ;BR IF 50
1698 007512 023737 001120 001122      CMP     $GDADR,$BDADR    ;ADRS CORRECT?
1699 007520 001764          BEQ     3$              ;TRY NEXT ADRS IF 50
1700 007522 104017          ERROR   17             ;ADRS MAKE ER 32X32X8
1701 007524 000422          BR      TST43           ;; TO NEXT TEST
1702 007526          5$:
1703 007526 023737 001124 001126      CMP     $GDADR,$BDADR    ;CORRECT?
1704 007534 001756          BEQ     3$              ;TRY NEXT ADRS IF 50
1705 007536 104021          ERROR   21             ;NPR INCREMENT ER AT ADRS INDICATED
1706 007540 000414          BR      TST43           ;; TO NEXT TEST
1707 007542 062737 002000 013234      6$:     ADD     #2000,OFFSET    ;ADVANCE BASE ADRS
1708 007550 103405          BCS     7$              ;BR IF 15 BITS TESTED(NON-NPR)
1709 007552 100345          BPL     2$              ;BR IF 14 BITS NOT TESTED
1710 007554 005737 013232          TST     TAM              ;ALLOW ONLY 14 BITS IF NPR
1711 007560 001004          BNE     TST43           ;;NEXT TEST IF NPR TESTED TO 16K
1712 007562 000741          BR      2$              ;GO TO 15 BITS SINCE NON-NPR
1713 007564 005237 013232      7$:     INC     TAM              ;SELECT NPR
1714 007570 000725          BR      1$              ;REPEAT TEST(NPR ENABLED)

```

```

*****
;TEST 43      TEST RESOLUTION 2 WITH B-GAMMA SET
*****

```

```

1718          TST43: SCOPE
1719 007572 000004          MOV     #200,$TIMES      ;DO 200 ITERATIONS
1720 007574 012737 000200 001160      MOV     #3,$LPERR       ;SET UP SCOPE LOOP ADRS
1721 007602 012737 007654 001110      CLR     TAM              ;NO NPR ENABLE THIS TEST
1722 007610 005037 013232          MOV     #1,DUAL         ;DUAL SAYS B-GAMA ADRS MAKE
1723 007614 012737 000001 013212      MOV     #100,$NCSFR     ;CLR ALL
1724 007622 012777 000100 171732      MOV     #20000,OFFSET   ;SET UP BASE
1725 007630 012737 020000 013234      MOV     #1000,$OFCSR    ;SELECT 64X64X16
1726 007636 012737 001000 013236      1$:     JSR      PC,REGLD     ;GO LOAD CSR + OFFSET
1727 007644 004737 012714          2$:     JSR      PC,MAKE     ;GO DO ADRS MAKE
1728 007650 004737 012746          3$:     JSR      PC,MKLOOP   ;ADRS MAKE SCOPE LOOP
1729 007654 004737 013050          BR      4$              ;RETURN HERE IF ALL ADRS MADE
1730 007660 000406          CMP     $GDADR,$BDADR    ;ADRS CORRECT?
1731 007662 023737 001120 001122      BEQ     2$              ;TRY NEXT ADRS IF 50
1732 007670 001767          ERROR   17             ;ADRS MAKE ER 64X64X16 WITH B-GAMMA
1733 007672 104017          BR      TST44           ;; TO NEXT TEST
1734 007674 000404          4$:     ADD     #20000,OFFSET ;ADVANCE BASE ADRS
1735 007676 062737 020000 013234      BCC     1$              ;BR IF NOT THRU 15 BITS
1736 007704 103357

```

```

*****
;TEST 44      TEST RESOLUTION 3 WITH B-GAMMA SET
*****

```

```

1737
1738          TST44: SCOPE
1739 007706 000004          MOV     #100,$TIMES     ;DO 100 ITERATIONS
1740 007710 012737 000100 001160      MOV     #3,$LPERR       ;SET UP SCOPE LOOP ADRS
1741 007716 012737 007770 001110      CLR     TAM              ;NO NPR ENABLE THIS TEST
1742 007724 005037 013232          MOV     #1,DUAL         ;DUAL SAYS B-GAMMA ADRS MAKE
1743 007730 012737 000001 013212

```

```

1746 007736 012777 000100 171616      MOV      #100, JNCSFR      ; CLR ALL
1747 007744 012737 020000 013234      MOV      #20000, OFFSET  ; SET UP BASE
1748 007752 012737 001400 013236      MOV      #1400, SOFCSR   ; SELECT 32X32X16
1749 007760 004737 012714          1$: JSR      PC, REGLD      ; GO LOAD CSR + OFFSET
1750 007764 004737 012746          2$: JSR      PC, MAKE      ; GO DO ADRS MAKE
1751 007770 004737 013050          3$: JSR      PC, MKLOOP    ; ADRS MAKE SCOPE LOOP
1752 007774 000406          BR      4$              ; RETURN HERE IF ALL ADRS MADE
1753 007776 023737 001120 001122      CMP      $GDADR, $BDADR  ; ADRS CORRECT?
1754 010004 001767          BEQ      2$              ; TRY NEXT ADRS IF SO
1755 010006 104017          ERROR   17              ; ADRS MAKE ER 32X32X16 WITH B-GAMMA
1756 010010 000404          BR      TST45           ; TO NEXT TEST
1757 010012 062737 010000 013234 4$: ADD      #10000, OFFSET  ; ADVANCE BASE ADRS
1758 010020 103357          BCC     1$              ; BR IF NOT THRU 15 BITS
1759
1760

```

```

*****
;*TEST 45      TEST RESOLUTION 6 WITH B-GAMMA SET
*****

```

```

1761
1762
1763 010022 000004      TST45: SCOPE
1764 010024 012737 000200 001160      MOV      #200, $TIMES    ; DO 200 ITERATIONS
1765 010032 012737 010104 001110      MOV      #3$, $LPERR    ; SET UP SCOPE LOOP ADRS
1766 010040 005037 013232          CLR      TAM            ; NO NPR ENABLE THIS TEST
1767 010044 012737 000001 013212      MOV      #1, DUAL       ; 'DUAL' SAYS B-GAMMA ADRS MAKE
1768 010052 012777 000100 171502      MOV      #100, JNCSFR   ; CLR ALL
1769 010060 012737 020000 013234      MOV      #20000, OFFSET ; SET UP BASE
1770 010066 012737 003000 013236      MOV      #3000, SOFCSR  ; SELECT 64X64X8
1771 010074 004737 012714          1$: JSR      PC, REGLD    ; GO LOAD CSR + OFFSET
1772 010100 004737 012746          2$: JSR      PC, MAKE    ; GO DO ADRS MAKE
1773 010104 004737 013050          3$: JSR      PC, MKLOOP  ; ADRS MAKE SCOPE LOOP
1774 010110 000406          BR      4$              ; RETURN HERE IF ALL ADRS MADE
1775 010112 023737 001120 001122      CMP      $GDADR, $BDADR ; ADRS CORRECT?
1776 010120 001767          BEQ      2$              ; TRY NEXT ADRS IF SO
1777 010122 104017          ERROR   17              ; ADRS MAKE ER 64X64X8 WITH B-GAMMA
1778 010124 000404          BR      TST46           ; TO NEXT TEST
1779 010126 062737 020000 013234 4$: ADD      #20000, OFFSET ; ADVANCE BASE ADRS
1780 010134 103357          BCC     1$              ; BR IF NOT THRU 15 BITS
1781
1782

```

```

*****
;*TEST 46      TEST RESOLUTION 7 WITH B-GAMMA SET
*****

```

```

1783
1784
1785 010136 000004      TST46: SCOPE
1786 010140 012737 000040 001160      MOV      #40, $TIMES    ; DO 40 ITERATIONS
1787 010146 012737 010220 001110      MOV      #3$, $LPERR    ; SET UP SCOPE LOOP ADRS
1788 010154 005037 013232          CLR      TAM            ; NO NPR ENABLE THIS TEST
1789 010160 012737 000001 013212      MOV      #1, DUAL       ; 'DUAL' SAYS B-GAMMA ADRS MAKE
1790 010166 012777 000100 171366      MOV      #100, JNCSFR   ; CLR ALL
1791 010174 012737 020000 013234      MOV      #20000, OFFSET ; SET UP BASE
1792 010202 012737 003400 013236      MOV      #3400, SOFCSR  ; SELECT 32X32X8
1793 010210 004737 012714          1$: JSR      PC, REGLD    ; GO LOAD CSR + OFFSET
1794 010214 004737 012746          2$: JSR      PC, MAKE    ; GO DO ADRS MAKE
1795 010220 004737 013050          3$: JSR      PC, MKLOOP  ; ADRS MAKE SCOPE LOOP
1796 010224 000406          BR      4$              ; RETURN HERE IF ALL ADRS MADE
1797 010226 023737 001120 001122      CMP      $GDADR, $BDADR ; ADRS CORRECT?
1798 010234 001767          BEQ      2$              ; TRY NEXT ADRS IF SO
1799 010236 104017          ERROR   17              ; ADRS MAKE ER 32X32X8 WITH B-GAMMA
1800 010240 000404          BR      TST47           ; TO NEXT TEST
1801 010242 062737 004000 013234 4$: ADD      #4000, OFFSET  ; ADVANCE BASE ADRS

```

```

1802 010250 103357          BCC      1$          ;BR IF NOT THRU 15 BITS
1803
1804
1805          ;*****
1806          ;*TEST 47      TEST WORD INCREMENT AND INCREMENT OVERFLOW FLAG
1807          ;*****
1807 010252 000004          TST47:  SCOPE
1808 010254 012737 000004 001160          MOV      #4,$TIMES          ;:DO 4 ITERATIONS
1809 010262 012737 010344 001110          MOV      #2,$SLPERR        ;:SET UP SCOPE LOOP ADRS
1810 010270 012700 000100          MOV      #100,$RO          ;:SET UP OVERFLOW FLAG CLR COUNTER
1811 010274 012777 000100 171260          MOV      #100,$NCSFR       ;:CLR ALL
1812 010302 012777 020000 171240          MOV      #20000,$NCOFF     ;:SET UP OFFSET REG TO 20000
1813 010310 012777 001001 171230          MOV      #1001,$NCCSR     ;:SELECT RESOLUTION 2 WITH NPR ENABLE
1814 010316 005037 020000          CLR      @#20000          ;:START LOC 20000 AT ZERO
1815 010322 005001          CLR      R1                ;:R1 CONTAINS EXPECTED NPR DATA
1816 010324 005002          CLR      R2                ;:R2 WHEN NON-ZERO SAYS INCREMENT OVFLG EXPECTED
1817
1818 010326 005201          1$:   INC      R1          ;:INCREMENT EXPECTED
1819 010330 001005          BNE      2$                ;:BR IF NO OVERFLOW
1820 010332 005101          COM      R1                ;:CAN'T EXCEED 177777
1821 010334 012702 000001          MOV      #1,$R2           ;:OVERFLOW FLAG SHOULD SET WHEN CELL 177777
1822 010340 005300          DEC      RO                ;:COUNT CLR FLAG ATTEMPTS, 32 SOFT, 32 HARD

```

```

1823 010342 001521          BEQ      TST50      ;: NEXT TEST IF ALL CLR FLAG ATTEMPTS COMPLETED
1824 010344 012777 000001 171210 2$: MOV      #1, ANCSFR ;: CLR XY HOLD REG
1825 010352 012777 000000 171172   MOV      #0, ANCXH ;: DO AN WORD INC NPR TO 20000
1826 010360 000240          NOP          ;: STALL
1827 010362 005702          TST      R2      ;: EXPECT OVERFLOW?
1828 010364 001452          BEQ      5$      ;: BR IF NOT - BUT OK IT DID NOT SET
1829 010366 032777 020000 171152   BIT      #20000, ANCCSR ;: LOOK FOR OVERFLOW FLAG
1830 010374 001012          BNE      3$      ;: BR IS SET
1831 010376 013737 001546 001122   MOV      NCCSR, $BDADR ;: SET UP CSR REG ADRS
1832 010404 012737 070000 001126   MOV      #70000, $BDDAT ;: EXPECTED INCREMENT OVERFLOW REG
1833 010412 005037 001126          CLR      $BDDAT ;: SHOW IT WAS NOT SET
1834 010416 104022          ERROR    22     ;: INCREMENT OVERFLOW FLAG FAILED TO SET
1835 010420 000472          BR       TST50   ;: NEXT TEST ON ER
1836 010422 020427 000040 3$: CMP      R4, #40 ;: CLR INCR OVFL0 BY SOFTWARE?
1837 010426 100026          BPL      4$      ;: BR IF SO
1838 010430 005037 020002          CLR      @#20002 ;: INSURE LOC 20002=0
1839 010434 012777 000001 171120   MOV      #1, ANCSFR ;: CLR XY HOLD REG
1840 010442 012777 000002 171102   MOV      #2, ANCXH ;: DO A NPR TO LOC 20002 TO CLR OVFL0 FLAG
1841 010450 032777 020000 171070   BIT      #20000, ANCCSR ;: DID FLAG GET CLRED BY HARDWARE?
1842 010456 001433          BEQ      6$      ;: BR IF SO
1843 010460 013737 001546 001122   MOV      NCCSR, $BDADR ;: SET UP NC CSR REG ADRS
1844 010466 005037 001124          CLR      $GDDAT ;: EXPECTED NO OVERFLOW FLAG
1845 010472 012737 020000 001126   MOV      #20000, $BDDAT ;: SHOW IT DID NOT CLR ON ADRS MAKE
1846 010500 104022          ERROR    22     ;: INCREMENT OVERFLOW FLAG FAILED TO CLR ON
1847                                     ;: HARDWARE MAKE ADRS
1848                                     ;: NEXT TEST ON ER
1849 010502 000441          BR       TST50   ;: DO A SPECIAL FUNCTION CLR ON INCR OVFL0 FLG
1850 010504 012777 000002 171050 4$: MOV      #2, ANCSFR ;: DID FLAG SET IN ER OR FAIL TO CLR?
1851 010512 032777 020000 171026 5$: BIT      #20000, ANCCSR ;: BR IF OK (CLRED) - CK DATA INC
1852 010520 001412          BEQ      6$      ;: SET UP REG CSR ADRS
1853 010522 013737 001546 001122   MOV      NCCSR, $BDADR ;: EXPECTED NO FLAG
1854 010530 005037 001124          CLR      $GDDAT ;: SHOW IT WAS SET
1855 010534 012737 020000 001126   MOV      #20000, $BDDAT ;: INCREMENT OVERFLOW FLAG SET PREMATURELY
1856 010542 104022          ERROR    22     ;: OR FAILED TO CLR BY SPECIAL FUNCTION CLR
1857                                     ;: NEXT TEST ON ER
1858 010544 000420          BR       TST50   ;: SET UP XY HOLD LOC FOR TYPE
1859 010546 012737 000000 013226 6$: MOV      #0, XYHOLD ;: SET UP ADRS OF NPR
1860 010554 012737 020000 001122   MOV      #20000, $BDADR ;: LD EXPECTED
1861 010562 010137 001124          MOV      R1, $GDDAT ;: GET LOCATION AFTER INCREMENT
1862 010566 013737 020000 001126   MOV      @#20000, $BDDAT ;: CORRECT?
1863 010574 023737 001124 001126   CMP      $GDDAT, $BDDAT ;: BR IF SO - DO ANOTHER ADRS MAKE
1864 010602 001651          BEQ      1$      ;: WORD INCREMENT FAILURE
1865 010604 104023          ERROR    23
1866                                     ;: *****
1867                                     ;: *TEST 50      TEST EVEN BYTE INCREMENT AND INCREMENT OVERFLOW FLAG
1868                                     ;: *****
1869 010606 000004          TST50: SCOPE
1870 010610 012737 000100 001160   MOV      #100, $TIMES ;: DO 100 ITERATIONS
1871 010616 012737 010700 001110   MOV      #2$, $LPERR ;: SET UP SCOPE LOOP ADRS
1872 010624 012700 000100          MOV      #100, R0 ;: SET UP OVERFLOW FLAG CLR COUNTER
1873 010630 012777 000100 170724   MOV      #100, ANCSFR ;: CLR ALL
1874 010636 012777 036000 170704   MOV      #36000, ANCOFF ;: OFFSET BITS 10-13 SHOULD HAVE NO AFFECT AT RES 4
1875 010644 012777 002001 170674   MOV      #2001, ANCCSR ;: SELECT RESOLUTION 4 WITH NPR ENABLE
1876 010652 005037 020000          CLR      @#20000 ;: START LOC 20000 AT ZERO
1877 010656 005001          CLR      R1      ;: R1 CONTAINS EXPECTED NPR DATA
1878 010660 005002          CLR      R2      ;: R2 WHEN NON ZERO SAYS INCREMENT OVFL FLG EXPECTED

```



```

1879
1880 010662 105201 1S: INCB R1 ; INCREMENT EXPECTED
1881 010664 001005 BNE 2S ; BR IF NO OVERFLOW
1882 010666 105101 COMB R1 ; CAN'T EXCEED 377
1883 010670 012702 000001 MOV #1,R2 ; OVERFLOW FLAG SHULD SET WHEN CELL 377
1884 010674 005300 DEC R0 ; COUNT CLR FLAG ATTEMPTS, 32 SOFT, 32 HARD
1885 010676 001516 BEQ TST51 ; NEXT TEST IF ALL CLR FLAG ATTEMPTS COMPLETED
1886 010700 012777 000001 170654 2S: MOV #1,ANCSFR ; CLR XY HOLD REG
1887 010706 012777 040000 170636 MOV #40000,ANCSYH ; DO AN WORD INC NPR TO 20000
1888 010714 000240 NOP ; STALL
1889 010716 005702 TST R2 ; EXPECT OVERFLOW?
1890 010720 001447 BEQ 5S ; BR IF NOT - BUT CHECK IF DID NOT SET
1891 010722 032777 020000 170616 BIT #20000,ANCCSR ; LOOK FOR OVERFLOW FLAG
1892 010730 001007 BNE 3S ; BR IF SET
1893 010732 013737 001546 001122 MOV NCCSR,$BDADR ; SET UP CSR REG ADRS
1894 010740 005037 001126 CLR $BDDAT ; SHOW IT WAS NOT SET
1895 010744 104022 ERROR 22 ; INCREMENT OVERFLOW FLAG FAILED TO SET
1896 010746 000472 BR TST51 ; NEXT TEST ON ER
1897 010750 020027 000040 3S: CMP R0,#40 ; CLR INCR OVFL0 BY SOFTWARE?
1898 010754 100026 BPL 4S ; BR IF SO
1899 010756 005037 020002 CLR #20002 ; INSURE LOC 20002=0
1900 010762 012777 000001 170572 MOV #1,ANCSFR ; CLR XY HOLD REG
1901 010770 012777 040002 170554 MOV #40002,ANCSYH ; DO A NPR TO LOC 20002 TO CLR OVFL0 FLAG
1902 010776 032777 020000 170542 BIT #20000,ANCCSR ; DID FLAG GET CLRED BY HARDWARE?
1903 011004 001433 BEQ 6S ; BR IF SO
1904 011006 013737 001546 001122 MOV NCCSR,$BDADR ; SET NPR NC CSR REG - ADRS
1905 011014 005037 001124 CLR $GDDAT ; EXPECTED NO OVERFLOW FLAG
1906 011020 012737 020000 001126 MOV #20000,$BDDAT ; SHOW IF DID NOT CLR ON ADRS MAKE
1907 011026 104022 ERROR 22 ; INCREMENT OVERFLOW FLAG FAILED TO CLR ON
1908 ; HARDWARE MAKE ADRS
1909 ; NEXT TEST ON ER
1910 011030 000441 BR TST51
1911 011032 012777 000002 170522 4S: MOV #2,ANCSFR ; DO A SPECIAL FUNCTION CLR ON INCR OVFL0 FLAG
1912 011040 032777 020000 170500 5S: BIT #20000,ANCCSR ; DID FLAG SET IN ER OR FAIL TO CLR?
1913 011046 001412 BEQ 6S ; BR IF OK (CLRED) - CK DATA INC
1914 011050 013737 001546 001122 MOV NCCSR,$BDADR ; SET UP REG CSR ADRS
1915 011056 005037 001124 CLR $GDDAT ; EXPECTED NO FLAG
1916 011062 012737 020000 001126 MOV #20000,$BDDAT ; SHOW IT WAS SET
1917 011070 104022 ERROR 22 ; INCREMENT OVERFLOW FLAG SET PREMATURELY
1918 ; OR FAILED TO CLR BY SPECIAL FUNCTION ON
1919 ; NEXT TEST ON ER
1920 011072 000420 BR TST51
1921 011074 012737 040000 013226 6S: MOV #40000,XYHOLD ; SET UP XY HOLD LOC FOR TYPE
1922 011102 012737 020000 001120 MOV #20000,$GDADR ; SET UP ADRS OF NPR
1923 011110 010137 001124 MOV R1,$GDDAT ; LD EXPECTED
1924 011114 013737 020000 001126 MOV #20000,$BDDAT ; GET LOCATION AFTER INCREMENT
1925 011122 023737 001124 001126 CMP $GDDAT,$BDDAT ; CORRECT?
1926 011130 001654 BEQ 1S ; BR IF SO - DO ANOTHER ADRS MAKE
1927 011132 104024 ERROR 24 ; EVEN BYTE INCREMENT FAILURE
1928
1929 ; *****
1930 ; *TEST 51 TEST ODD BYTE INCREMENT AND INCREMENT OVERFLOW FLAG
1931 ; *****
1932 TST51: SCOPE
1933 MOV #100,$TIMES ; DO 100 ITERATIONS
1934 MOV #25,$LPERR ; SET UP SCOPE LOOP ADRS
MOV #100,R0 ; SET UP OVERFLOW FLAG CLR COUNTER
MOV #100,ANCSFR ; CLR ALL

```

```

1935 011164 012777 030000 170356      MOV      #30000,ANCOFF      ;OFFSET BITS 10-13 SHOULD HAVE NO AFFECT AT RES 4
1936 011172 012777 002001 170346      MOV      #2001,ANCCSR     ;SELECT RESOLUTION 2 WITH NPR ENABLE
1937 011200 005037 020000                CLR      AN20000         ;START LOC 20000 AT ZERO
1938 011204 005037 013230                CLR      GOOD            ;GOOD CONTAINS EXPECTED NPR DATA
1939 011210 005002                CLR      R2              ;R2 WHEN NON-ZERO SAYS INCREMENT OVFL0 FLAG EXPECTED
1940
1941 011212 105237 013231      1S:      INCB      GOOD+1      ;INCREMENT EXPECTED
1942 011216 001006                BNE      2S              ;BR IF NO OVERFLOW
1943 011220 105137 013231      COMB     GOOD+1         ;CAN'T EXCEED 177400
1944 011224 012702 000001      MOV      #1,R2          ;OVERFLOW FLAG SHOULD SET WHEN CELL 177400
1945 011230 005300                DEC      R0              ;COUNT CLR FLAG ATTEMPTS, 32 SOFT, 32 HARD
1946 011232 001522                BEQ      TST52           ;NEXT TEST IF ALL CLR FLAG ATTEMPTS COMPLETED
1947 011234 012777 000001 170320      2S:      MOV      #1,ANCSFR   ;CLR XY HOLD REG
1948 011242 012777 040001 170302      MOV      #40001,ANCXH   ;DO AN WORD IN NPR TO 20001
1949 011250 000240                NOP                       ;STALL
1950 011252 005702                TST     R2              ;EXPECT OVERFLOW?
1951 011254 001452                BEQ      5S              ;BR IF NOT - BUT CHECK IT DID NOT SET
1952 011256 032777 020000 170262      BIT     #20000,ANCCSR   ;LOOK FOR OVERFLOW FLAG
1953 011264 001012                BNE     3S              ;BR IF SET
1954 011266 013737 001546 001122      MOV     NCCSR,$BDADR    ;SET UP CSR REG ADRS
1955 011274 012737 020000 001124      MOV     #20000,$GDDAT   ;EXPECT INCREMENT OVERFLOW FLAG
1956 011302 005037 001126                CLR     $BDAT           ;SHOW IT WAS NOT SET
1957 011306 104022                ERROR   22             ;INCREMENT OVERFLOW FLAG FAILED TO SET
1958 011310 000473                BR      TST52           ;NEXT TEST ON ER
1959 011312 020027 000040      3S:      CMP     R0,#40          ;CLR OVERFLOW BY SOFTWARE?
1960 011316 100026                BPL     4S              ;BR IF SO
1961 011320 005037 020002                CLR     AN20002        ;INSURE LOC=0
1962 011324 012777 000001 170230      MOV     #1,ANCSFR      ;CLR XY HOLD REG
1963 011332 012777 040002 170212      MOV     #40002,ANCXH   ;DO A NPR TO LOC 20002 TO CLR OVFL0 FLG
1964 011340 032777 020000 170200      BIT     #20000,ANCCSR   ;DID FLAG GET CLEARED BY HARDWARE?
1965 011346 001433                BEQ     6S              ;BR IF SO
1966 011350 013737 001546 001122      MOV     NCCSR,$BDADR    ;SET NPR NC CSR REG ADRS
1967 011356 005037 001124                CLR     $GDDAT         ;EXPECTED NO OVERFLOW FLAG
1968 011362 012737 020000 001126      MOV     #20000,$BDAT   ;SHOW IT DID NOT CLR ON ADRS MAKE
1969 011370 104022                ERROR   22             ;INCREMENT OVERFLOW FLAG FAILED TO CLR ON
1970
1971 011372 000442                BR      TST52           ;NEXT TEST ON ER
1972 011374 012777 000002 170160      4S:      MOV     #2,ANCSFR      ;DO A SPECIAL FUNCTION CLR ON INCR OVFL0 FLAG
1973 011402 032777 020000 170136      5S:      BIT     #20000,ANCCSR   ;DID FLAG SET IN ER OR FAIL TO CLR?
1974 011410 001412                BEQ     6S              ;BR IF OK (CLRED) - CK DATA INC
1975 011412 013737 001546 001122      MOV     NCCSR,$BDADR    ;SET UP REG CSR ADRS
1976 011420 005037 001124                CLR     $GDDAT         ;EXPECTED NO FLAG
1977 011424 012737 020000 001126      MOV     #20000,$BDAT   ;SHOW IT WAS SET
1978 011432 104022                ERROR   22             ;INCREMENT OVERFLOW FLAG SET PREMATURELY
1979
1980 011434 000421                BR      TST52           ;NEXT TEST ON ER
1981 011436 012737 040001 013226      6S:      MOV     #40001,XYHOLD   ;SET UP XY HOLD LOC FOR TYPE
1982 011444 012737 020001 001122      MOV     #20001,$BDADR   ;SET UP ADRS OF NPR
1983 011452 013737 013230 001124      MOV     GOOD,$GDDAT     ;LD EXPECTED
1984 011460 013737 020000 001126      MOV     AN20000,$BDAT   ;GET LOCATION AFTER INCREMENT
1985 011466 023737 001124 001126      CMP     $GDDAT,$BDAT   ;CORRECT?
1986 011474 001646                BEQ     1S              ;BR IF SO - DO ANOTHER ADRS MAKE
1987 011476 104024                ERROR   24             ;ODD BYTE INCREMENT FAILURE
1988
1989
1990

```

```

;*****
;*TEST 52 TEST THAT INCREMENT OVERFLOW FLAG WILL INTERRUPT

```

```

1991
1992 011500 000004
1993 011502 012737 000004 001160
1994 011510 012737 000340 177776
1995 011516 012777 011644 170040
1996 011524 012777 000340 170034
1997 011532 005000
1998 011534 012777 000103 170020
1999 011542 012777 020000 170000
2000 011550 012777 001101 167770
2001 011556 005037 020000
2002
2003 011562 012777 000001 167772 1S: MOV #1, JNCSFR ; CLR XY HOLD REG
2004 011570 012777 000000 167754 MOV #0, JNCXYH ; DO AN ADRS MAKE TO LOC 20000
2005 011576 012737 000300 177776 MOV #PR6, PSW ; ALLOW AN INTR ON INCREMENT OVFL0
2006 011604 013737 001546 001122 MOV NCCSR, $BDADR ; SET UP CSR REG ADRS
2007 011612 005200 INC RO ; COUNT ALONG WITH LOC 20000
2008 011614 001362 BNE 1S ; OVERFLOW OCCURS INCREMENTING AT 177777
2009 011616 012737 020000 001124 MOV #20000, $GDDAT ; SHOULD NEVER GET HERE - EXPECT OVFL0 FLG
2010 011624 017737 167716 001126 MOV JNCCSR, $BDDAT ; READ CSR
2011 011632 042737 157777 001126 BIC #157777, $BDDAT ; SAVE INCREMENT OVERFLOW FLAG
2012 011640 104022 ERROR 22 ; INCREMENT OVERFLOW FLAG FAILED TO INTERRUPT
2013 011642 000430 BR 4S ; RESTORE VECTOR LOCS
2014 011644 022626 2S: CMP (SP)+, (SP)+ ; FIX STACK SINCE NO RTI
2015 011646 020027 177777 CMP RO, #-1 ; EXPECT INTR ON COUNT OF 177777
2016 011652 001412 BEQ 3S ; BR IF EXPECTED
2017 011654 005037 001124 CLR $GDDAT ; LD EXPECTED
2018 011660 017737 167662 001126 MOV JNCCSR, $BDDAT ; READ CSR
2019 011666 042737 157777 001126 BIC #157777, $BDDAT ; SAVE ONLY INCREMENT OVFL0 FLAG
2020 011674 104022 ERROR 22 ; PREMATURE INCREMENT OVERFLOW INTR
2021 011676 000412 BR 4S ; RESTORE VECTOR LOCS
2022 011700 012737 020000 001124 3S: MOV #20000, $GDDAT ; LD EXPECTED
2023 011706 032777 020000 167632 BIT #20000, JNCCSR ; LOOK FOR INCREMENT OVERFLOW FLAG
2024 011714 001003 BNE 4S ; BR IF SET
2025 011716 005037 001126 CLR $BDDAT ; SHOW IT WAS NOT SET
2026 011722 104022 ERROR 22 ; INCREMENT OVFL0 FLAG NOT SET ON INTR
2027 011724 013777 001566 167632 4S: MOV JNCVCT2, JNCVCT0 ; RESTORE VECTOR PTR
2028 011732 005077 167630 CLR JNCVCT2 ; RESTORE HALT
2029
2030
2031
2032

```

```

*****
; TEST 53 TEST THAT TIMEOUT WILL SET THEN CLR WITH SPECIAL FUNCTION CLR
*****

```

```

2033 011736 000004
2034 011740 012777 000100 167614
2035 011746 012777 140003 167574
2036 011754 012777 002001 167564
2037 011762 012777 040000 167562
2038 011770 013737 001546 001122
2039 011776 012737 100000 001124
2040 012004 005777 167536
2041 012010 100403
2042 012012 005037 001126
2043 012016 104016
2044 012020 012777 000100 167534 1S: MOV #100, JNCSFR ; CLR ALL
2045 012026 005777 167514 TST JNCCSR ; SET UPPER ADRS BITS 14, 15, 16, 17 (740000)
2046 012032 100006 BPL 2S ; SELECT RES4 WITH NPR ENABLE
; TRY A BYTE INC TO LOC (760000) RESERVED TIMEOUT ADRS
; SET UP CSR ADRS
; EXPECT THE TIMEOUT BIT
; SEE IF SET
; BR IF SO
; SHOW IT DID NOT SET
; TIMEOUT FAILED TO SET AT ADRS 760000
; NOW USE SPECIAL FUNCTION CLR ON TIMEOUT FLAG
; LOOK FOR IT CLEARED
; BR IF SO

```

E04

```

2047 012034 005037 001124          CLR      $GDDAT      ;EXPECTED ZERO
2048 012040 012737 100000 001126  MOV      #BIT15,$BDDAT ;SHOW IT WAS STILL SET
2049 012046 104016          ERROR    16          ;TIMEOUT FLAG FAILED TO CLR WITH SPECIAL FUNCTION
2050 012050 000137 012560          2$:     JMP      $EOP        ;GO REPORT 'ENDPASS'
2051
2052 012054          TEST54:
2053          ;*****
2054          ;*TEST 54      TEST THAT TIMING MARK SETS IN LIST MODE - START 204
2055          ;*****
2056 012054 000240          TST54:  <NOP>
2057 012056 012737 012064 001106  MOV      #10,$SLPADR  ;:SET SCOPE LOOP ADDRESS
2058 012064 012737 012064 001110  10$:    MOV      #10,$SLPERR ;:SET UP SCOPE LOOP ADRS
2059 012072 012700 002000          MOV      #2000,R0     ;:DO CHECK 2000 TIMES (8)
2060 012076 013737 001552 001122  1$:     MOV      NCADR,$BDADR ;:SET UP ADRS REG ADRS
2061 012104 012777 000102 167450  MOV      #102,$NCSFR ;:CLEAR INTERFACE
2062 012112 012777 000400 167426  MOV      #400,$NCCSR ;:SELECT LIST MD
2063 012120 012777 000000 167424  MOV      #0,$NCXYH   ;:CLR ADRS REG
2064 012126 012777 000102 167426  MOV      #102,$NCSFR ;:CLR ALL
2065 012134 012777 000040 167420  MOV      #40,$NCSFR  ;:SET TIME MARK
2066 012142 012777 000422 167376  MOV      #422,$NCCSR ;:LIST MD, EXT INPUT, ENABLE ADC
2067 012150 052777 000004 167404  2$:     BIS      #4,$NCSFR   ;:START ADC'S
2068 012156 004737 012702          JSR      PC,DELAY     ;:GO STALL FOR A/D DONE
2069 012162 017737 167364 001126  MOV      $NCADR,$BDDAT ;:GET ADRS MADE
2070 012170 001002          BNE     3$
2071 012172 104025          ERROR    25
2072
2073 012174 000435          BR      TST55
2074 012176 042737 077777 001126  3$:     BIC      #77777,$BDDAT ;:SAVE ONLY TIME MARK ADRS BIT
2075 012204 100405          BMI     4$
2076 012206 012737 100000 001124  MOV      #100000,$GDDAT ;:SHOW IT WAS EXPECTED
2077 012214 104026          ERROR    26
2078
2079 012216 000424          BR      TST55
2080 012220 052777 000004 167334  4$:     BIS      #4,$NCSFR   ;:START ADC'S
2081 012226 004737 012702          JSR      PC,DELAY     ;:GO STALL FOR A/D DONE
2082 012232 017737 167314 001126  MOV      $NCADR,$BDDAT ;:GET ADRS MADE
2083 012240 001002          BNE     5$
2084 012242 104025          ERROR    25
2085
2086 012244 000411          BR      TST55
2087 012246 042737 077777 001126  5$:     BIC      #77777,$BDDAT ;:SAVE TIME MARK ADRS BIT ONLY
2088 012254 100003          BPL     6$
2089 012256 005037 001124          CLR      $GDDAT      ;:EXPECTED NO TIME MARK
2090 012262 104026          ERROR    26
2091 012264 005300          6$:     DEC      R0
2092 012266 001303          BNE     1$
2093
2094          ;*****
2095          ;*TEST 55      TEST JOYSTICK ADC CONVERT, NO CONVERT & JOYSTICK LOGIC
2096          ;*****
2097 012270 000004          TST55:  SCOPE
2098 012272 012700 000001  1$:     MOV      #1,R0        ;:R0 COUNTS NO. OF ITEMS PER LINE
2099 012276 012777 000102 167256  MOV      #102,$NCSFR  ;:CLR INTERFACE
2100 012304 012777 000032 167234  MOV      #32,$NCCSR  ;:SELECT EXTERNAL JOYSTICK, ENAB ADC
2101 012312 005300          2$:     DEC      R0
2102 012314 001010          BNE     3$
2103

```

F04

MAINDEC-11-DZNC-A-C NC11-A LOGIC TEST MACY11 27(732) 21-SEP-76 14:14 PAGE 44
 DZNC.A.P11 T55 TEST JOYSTICK ADC CONVERT, NO CONVERT & JOYSTICK LOGIC

2103	012316	012700	000006			MOV	#6,RO	;RESET TYPE COUNTER
2104	012322	032777	020000	166610	10\$:	BIT	#BIT13,ASWR	;ARE WE TYPING?
2105	012330	001002				BNE	3\$;BR IF NOT
2106	012332	104400	070000			TYPE	,MSG2	;GO TYPE 'LF+CR'
2107	012336	104406			3\$:	CKSWR		;GO LOOK FOR 'CONTROL-G'
2108	012340	012777	000004	167214		MOV	#4,ANCSFR	;DO A CONVERSION
2109	012346	004737	012702			JSR	PC,DELAY	;GO STALL FOR A/D DONE
2110	012352	105777	167170			TSTB	ANCCSR	;SHOULD NOT BE BUSY
2111	012356	100016				BPL	4\$;BR IF DONE
2112	012360	032777	040000	167160		BIT	#40000,ANCCSR	;NO CONVERT SHOULD BE SET IF STILL BUSY
2113	012366	001032				BNE	5\$;BR IF "NO CONVERT" FLAG IS SET
2114	012370	013737	001546	001122		MOV	NCCSR,\$BDADR	;SET UP REG CSR ADRS
2115	012376	012737	040000	001124		MOV	#40000,\$GDDAT	;EXPECTED NO CONVERT WHEN BUSY STILL SET
2116	012404	005037	001126			CLR	\$BDDAT	;SHOW IT WAS NOT SET
2117	012410	104027				ERROR	27	;NO CONVERT FLAG NOT SET WHEN BUSY SET
2118	012412	000727				BR	1\$;GO CONVERT AGAIN
2119								
2120	012414	017737	167132	013222	4\$:	MOV	ANCXH,XYAD	;READ CONVERTED VALUES
2121	012422	012777	000010	167132		MOV	#10,ANCSFR	;CLR JOYSTICK DEPRESS BUTTON
2122	012430	005777	167112			TST	ANCCSR	;IS BUTTON DEPRESSED?
2123	012434	100016				BPL	6\$;BR IF NOT
2124	012436	032777	020000	166474		BIT	#BIT13,ASWR	;TYPE "BAR DN"?
2125	012444	001326				BNE	10\$;BR IF NO TYPE
2126	012446	104400	070003			TYPE	,MSG3	;TYPE "BAR DN"
2127	012452	000717				BR	2\$;GO CONVERT AGAIN
2128								
2129	012454	032777	020000	166456	5\$:	BIT	#BIT13,ASWR	;TYPE "NO CNVRT" MESSAGE?
2130	012462	001317				BNE	10\$;BR IF NOT
2131	012464	104400	070013			TYPE	,MSG4	;TYPE "NO CNVRT"
2132	012470	000710				BR	2\$;GO CONVERT AGAIN
2133								
2134	012472	005037	013216		6\$:	CLR	XADC	;CLR X LOC
2135	012476	005037	013220			CLR	YADC	;CLR Y LOC
2136	012502	113737	013222	013216		MOVB	XYAD,XADC	;GET X CONVERSION
2137	012510	113737	013223	013220		MOVB	XYAD+1,YADC	;GET Y CONVERSION
2138	012516	032777	020000	166414		BIT	#BIT13,ASWR	;TYPE X+Y DATA?
2139	012524	001276				BNE	10\$;BR IF NOT
2140	012526	104400	070025			TYPE	,MSG5	;TYPE X=
2141	012532	013746	013216			MOV	XADC,-(SP)	;SAVE XADC FOR TYPEOUT
2142								;TYPE X VALUE
2143	012536	104402				TYPOS		;GO TYPE--OCTAL ASCII
2144	012540	003				.BYTE	3	;TYPE 3 DIGIT(S)
2145	012541	001				.BYTE	1	;TYPE LEADING ZEROS
2146	012542	104400	070031			TYPE	,MSG6	;TYPE Y=
2147	012546	013746	013220			MOV	YADC,-(SP)	;SAVE YADC FOR TYPEOUT
2148								;TYPE Y VALUE
2149	012552	104402				TYPOS		;GO TYPE--OCTAL ASCII
2150	012554	003				.BYTE	3	;TYPE 3 DIGIT(S)
2151	012555	001				.BYTE	1	;TYPE LEADING ZEROS
2152	012556	000655				BR	2\$;GO CONVERT AGAIN

```

2153
2154
2155
2156
2157
2158
2159
2160
2161 012560
2162 012560 000004
2163 012562 005037 001102
2164 012566 005037 001160
2165 012572 005237 001202
2166 012576 042737 100000 001202
2167 012604 005327
2168 012606 000001
2169 012610 003022
2170 012612 012737
2171 012614 000001
2172 012616 012606
2173 012620 104400 012665
2174 012624 013746 001202
2175 012630 104404
2176 012632 104400 012662
2177 012636 013700 000042
2178 012642 001405
2179 012644 000005
2180 012646 004710
2181 012650 000240
2182 012652 000240
2183 012654 000240
2184 012656
2185 012656 000137
2186 012660 002310
2187 012662 377 377 000
2188 012665 015 042412 042116
2189 012672 050040 051501 020123
2190 012700 000043

```

.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 RESTRT

```

```

SEOP:
SCOPE
CLR $STNM ;;ZERO THE TEST NUMBER
CLR $TIMES ;;ZERO THE NUMBER OF ITERATIONS
INC $PASS ;;INCREMENT THE PASS NUMBER
BIC #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER
DEC (PC)+ ;;LOOP?
SEOPCT: .WORD 1
BGT $DOAGN ;;YES
MOV (PC)+,a(PC)+ ;;RESTORE COUNTER
SENDCT: .WORD 1
SEOPCT
TYPE $SENDMG ;;TYPE "END PASS #"
MOV $PASS,-(SP) ;;SAVE $PASS FOR TYPEOUT
TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN
TYPE $SENULL ;;TYPE A NULL CHARACTER
$GET42: MOV a#42,RO ;;GET MONITOR ADDRESS
BEQ $DOAGN ;;BRANCH IF NO MONITOR
RESET ;;CLEAR THE WORLD
SENDAD: JSR PC,(RO) ;;GO TO MONITOR
NOP ;;SAVE ROOM
NOP ;;FOR
NOP ;;ACT11
$DOAGN: JMP a(PC)+ ;;RETURN
SRTNAD: .WORD RESTRT
$ENULL: .BYTE -1,-1,0 ;;NULL CHARACTER STRING
$SENDMG: .ASCIZ <15><12>/END PASS #/

```

```

2191
2192
2193
2194 012702 012705 000010
2195 012706 005305
2196 012710 001376
2197 012712 000207
2198
2199
2200
2201
2202
2203 012714 013777 013234 166626
2204 012722 053737 013232 013236
2205 012730 013777 013236 166610
2206 012736 012737 000200 013214
2207 012744 000207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226 012746 062716 000004
2227 012752 006237 013214
2228 012756 001003
2229 012760 012737 177677 013214
2230 012766 022737 177777 013214
2231 012774 001505
2232 012776 013737 013214 013216
2233 013004 042737 177600 013216
2234 013012 013737 013216 013220
2235 013020 113737 013216 013226
2236 013026 005737 013212
2237 013032 001403
2238 013034 052737 000200 013226
2239
2240 013042 113737 013220 013227
2241 013050 062716 000002
2242
2243 013054 012777 000001 166500
2244 013062 004737 013242
2245 013066 013737 013230 001120
2246 013074 005737 013232

```

```

*****
: THIS IS A COMMON DELAY ROUTINE OF A COUNT OF 10
*****
DELAY: MOV #10,R5 ;SET UP COUNT IN R5
IS: DEC R5 ;COUNT AWAY
BNE IS ;TILL DONE
RTS PC ;RETURN WHEN DONE
*****
: THIS SUBROUTINE LOADS INTERFACE REGISTERS FROM SOFTWARE REGS
: IT ALSO INIT'S THE SOFTWARE ADC'S
*****
REGLD: MOV OFFSET,ANCOFF ;LOAD OFFSET REGISTER
BIS TAM,SOFCSR ;SET STAGE II ENABLE IF WARRANTED
MOV SOFCSR,ANCCSR ;SET UP COMMAND REGISTER
MOV #200,XYPTRN ;START ADC'S WITH FLOATING ONE
RTS PC ;RETURN TO ADRS MAKE TEST
*****
: THIS IS A ROUTINE TO SIMULATE 2 ADC'S
: ON EACH CALL THE ONE OR ZERO PATTERN IS SHIFTED
: RIGHT AND LOADED INTO THE SOFTWARE XADC & YADC -
: FIRST A ONE IS FLOATED RIGHT AND THEN A ZERO - WHEN THE
: ZERO HAS PASSED THRU BIT 0 OF THE ADC'S THE RETURN IS TO CALL
: +4 - ALL OTHER RETURNS ARE TO CALL+6.
: THE XYHOLD LOC CONTAINS THE X/Y ADC VALUES AS WRITTEN
: TO THE XYHOLD NC11 REG.
: THE ADDRESS IS THEN MADE IN SOFTWARE AND THEN IN HARDWARE
: SOFTWARE ADDRESS IS RETURNED IN $GDADR
: HARDWARE ADDRESS RETURNED IN $BDADR
: WHEN NPR (TAM=1) IS ENABLED THE CONTENTS OF THE NPR'ED LOC
: ARE SAVED IN 'LOCSAV' AND RESTORED AFTER THE NPR.
: THE MEMORY LOC AFTER THE NPR IS PUT IN $BDDAT
: THE EXPECTED NPR DATA IS PUT IN $GDDAT
*****
MAKE: ADD #4,(SP) ;RETURN AFTER THE SCOPE LOOP JSR
ASR XYPTRN ;SHIFT ONE/ZERO RIGHT
BNE IS ;BR IF NOT DONE WITH ZERO YET
MOV #177677,XYPTRN ;NOW SET UP FLOATING ZERO PATTERN
IS: CMP #-1,XYPTRN ;ALL DONE FLOATING ZERO?
BEQ RNGDN ;BR IF SO
MOV XYPTRN,XADC ;SET UP X VALUE
BIC #177600,XADC ;LOW ORDER 7 BITS ONLY
MOV XADC,YADC ;GOES TO Y VALUE ALSO
SAMDN: MOVB XADC,XYHOLD ;FORM THE XYHOLD WORD
TST DUAL ;B GAMMA SET?
BEQ NODUAL ;BR IF NOT
BIS #200,XYHOLD ;SET B GAMMA IN XYHOLD WORD
NODUAL: MOVB YADC,XYHOLD+1 ;SET UP YADC IN XYHOLD WORD
MKLOOP: ADD #2,(SP) ;RETURN TO CALL +6 TO CK DATA
;OR +2 IF ENTERED HERE FROM SCOPE LOOP
MOV #1,ANCSFR ;CLEAR XYHOLD
JSR PC,SAMAK ;DO A SOFTWARE ADDRESS MAKE
MOV GOOD,$GDADR ;GET MAKE ADRS
TST TAM ;ARE WE DOING NPR'S

```

```

2247 013100 001423          BEQ      2$          ;BR IF NOT
2248 013102 042737 000001 013230      BIC      #1,GOOD    ;MAKE INTO WORD ADRS
2249 013110 017737 000114 013240      MOV      @GOOD,LOCSAV ;SAVE LOCATION OF NPR
2250 013116 005077 000106          CLR      @GOOD      ;ZERO LOCATION
2251 013122 032737 000001 001120      BIT      #1,$GDADR   ;LOOK FOR BYTE INCREMENT
2252 013130 001404          BEQ      1$          ;BR IF LOW BYTE
2253 013132 012737 000400 001124      MOV      #400,$GDDAT ;HI BYTE INCREMENT
2254 013140 000403          BR       2$          ;SKIP
2255 013142 012737 000001 001124 1$: MOV      #1,$GDDAT   ;LOW BYTE INCREMENT
2256 013150 013777 013226 166374 2$: MOV      XYHOLD,@NCXYH ;DO A HARDWARE ADRS MAKE
2257 013156 021616          CMP      (SP),(SP)   ;STALL
2258 013160 017737 166366 001122      MOV      @NCADR,$BDADR ;READ THE HARDWARE ADRS MADE
2259 013166 005737 013232          TST      TAM        ;DOING NPR'S?
2260 013172 001406          BEQ      RNGDN      ;BR IF NOT
2261 013174 017737 000030 001126      MOV      @GOOD,$BDDAT ;READ THE CONTENTS OF THE NPR'ED ADRS
2262 013202 013777 013240 000020      MOV      LOCSAV,@GOOD ;RESTORE THE NPR LOC WITH PREVIOUS CONTENTS
2263 013210 000207          RTS      PC         ;NOW RETURN TO CK MAKE ADRS OR NPR DATA
2264 013212 000000          DUAL:    0          ;NON-ZERO INDICATES B GAMMA BEING TESTED
2265 013214 000200          XYPTRN: 200        ;CONTAINS PATTERN FOR X/Y SOFT ADC'S
2266 013216 000000          XADC:    0          ;SOFTWARE X ADC
2267 013220 000000          YADC:    0          ;SOFTWARE Y ADC
2268 013222 000000          XYAD:    0          ;HARDWARE XY DATA READ
2269 013224 000000          XTEMP:   0          ;USED IN BUILDING SOFTWARE ADDRESS MAKE
2270 013226 000000          XYHOLD: 0          ;SOFTWARE XY ADC VALUE WRITTEN TO XYHOLD REG
2271 013230 000000          GOOD:   0          ;MEMORY ADRS OF ADRS MAKE NPR
2272 013232 000000          TAM:    0          ;NON-ZERO INDICATES NPR ENABLED TO ADRS MADE
2273 013234 000000          OFFSET: 0         ;OFFSET ADRS BITS LOADED TO OFFSET REG
2274 013236 000000          SOFCSR: 0         ;CSR BITS WRITTEN TO NC CSR
2275 013240 000000          LOCSAV: 0         ;ALL SINGLE LOCS ARE SAVED & RESTORED FROM THIS LOC (NPR
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285 013242 005037 013230      SAMAK:  CLR GOOD    ;CLEAR FINAL ADDRESS
2286 013246 032737 000010 013236      BIT      #10,SOFCSR ;JOYSTICK MODE?
2287 013254 001042          BNE      JSM        ;YES
2288 013256 113737 013216 013230  SDCM:  MOV      XADC,GOOD ;SET UP FOR FORMATION
2289 013264 013700 013236          MOV      SOFCSR,R0  ;STORE SOFCSR IN R0 FOR SPEED
2290 013270 032700 003400          BIT      #3400,R0   ;CHECK FOR RES0
2291 013274 001444          BEQ      RES00      ;
2292 013276 032700 003000          BIT      #3000,R0   ;CHECK FOR RES1
2293 013302 001444          BEQ      RES01      ;
2294 013304 032700 002400          BIT      #2400,R0   ;CHECK FOR RES2
2295 013310 001445          BEQ      RES02      ;
2296 013312 032700 002000          BIT      #2000,R0   ;CHECK FOR RES3
2297 013316 001524          BEQ      RES03      ;
2298 013320 032700 001400          BIT      #1400,R0   ;CHECK FOR RES4
2299 013324 001410          BEQ      RES04X     ;
2300 013326 032700 001000          BIT      #1000,R0   ;CHECK FOR RES5
2301 013332 001407          BEQ      RES05X     ;
2302 013334 032700 000400          BIT      #400,R0    ;CKECK FOR RES6

```

```

:*****
:THIS CODE DETERMINES WHAT RESOLUTION IS SELECTED AND
:THEN DIRECTS THE PROGRAM TO THE SOFTWARE ADDRESS MAKER
:ROUTINE FOR THAT RESOLUTION - EACH ROUTINE ALIGNS THE
:X & Y ADC VALUES TO THERE PROPER ADDRESS BIT ASSIGNMENTS
:AND THEN ADDS IN THE OFFSET ADDRESS BITS ACCORDING TO THE
:LOGIC OF THAT RESOLUTION - THE ADDRESS MADE IS RETURNED IN 'GOOD'
:*****

```



```

2303 013340 001406          BEQ RES06X
2304 013342 000137 014260      JMP RES07          ;MUST BE RES7
2305 013346 000137 013716      RES04X: JMP RES04  ;JUMP TO SOFTWARE ADRS MAKER RES4
2306 013352 000137 014014      RES05X: JMP RES05  ;JUMP TO SOFTWARE ADRS MAKER RES5
2307 013356 000137 014136      RES06X: JMP RES06  ;JUMP TO SOFTWARE ADRS MAKER RES6
2308
2309
2310
2311
2312
2313 013362 113737 013216 013230 JSM:  MOVB XADC,GOOD  ;GET X VALUE
2314 013370 113737 013220 013231      MOVB YADC,GOOD+1    ;GET Y VALUE
2315 013376 042737 100200 013230      BIC #100200,GOOD    ;CLEAR BITS 15 AND 7
2316 013404 000207
2317
2318
2319
2320
2321
2322 013406 005037 013230      RES00: CLR GOOD     ;NULL - ALWAYS EXPECT ZERO
2323 013412 000207
2324
2325
2326
2327
2328
2329 013414 113737 013220 013231 RES01: MOVB YADC,GOOD+1 ;LOAD X&Y DIRECTLY WITH NO OFFSET
2330 013422 000207
2331
2332
2333
2334
2335
2336
2337
2338 013424 004737 013560          ;MAKE ADDRESS A LA RESOLUTION 2
2339 013430 013737 013230 013224      ;64X64X16
2340 013436 113737 013220 013230      0 0 0 Y Y Y Y Y X X X X X 0
2341 013444 004737 013560          ;0 15 14 13 6 5 4 3 2 1 6 5 4 3 2 1
2342 013450 000337 013230          RES02: JSR PC,M64X16 ;DO ROTATION AND STRIP
2343 013454 006237 013230          MOV GOOD,XTEMP    ;SAVE X
2344 013460 006237 013230          MOVB YADC,GOOD    ;GET Y
2345 013464 063737 013224 013230      JSR PC,M64X16     ;GET RID OF UNUSED BITS
2346 013472 013737 013234 013224      SWAB GOOD         ;POSITION Y
2347 013500 042737 017777 013224      ASR GOOD          ;TO EXACT POSITION
2348 013506 005737 013212          ASR GOOD
2349 013512 001416          ADD XTEMP,GOOD    ;GET SAVED X
2350 013514 032737 020000 013224      MOV OFFSET,XTEMP  ;ACCOUNT FOR OFFSET
2351 013522 001407          BIC #17777,XTEMP  ;ONLY BITS 15 14 13
2352 013524 052737 040000 013224      TST DUAL         ;G GAM SET?
2353 013532 042737 020000 013224      BEQ ND2          ;BR IF NOT
2354 013540 000403          BIT #BIT13,XTEMP  ;OFFSET 13 SET?
2355 013542 052737 020000 013224      BEQ IS          ;BR IF NOT
2356 013550 063737 013224 013230      BIS #BIT14,XTEMP  ;OFFSET 13 & B GAM SETS A14
2357 013556 000207          BIC #BIT13,XTEMP  ;OFFSET 13 & B GAM CLRS A13
2358 013560 042737 177601 013230      BR ND2          ;GO ADD OFFSET IN
                          IS:  BIS #BIT13,XTEMP  ;OFFSET 13 CLRED & B GAM SETS A13
                          ND2: ADD XTEMP,GOOD  ;ADD IN OFFSET
                          RTS PC
                          M64X16: BIC #177601,GOOD ;CLEAN GARBAGE

```

```

2359 013566 000207
2360
2361
2362
2363
2364
2365
2366
2367 013570 004737 013702
2368 013574 013737 013230 013224
2369 013602 113737 013220 013230
2370 013610 004737 013702
2371 013614 000337 013230
2372 013620 006237 013230
2373 013624 006237 013230
2374 013630 006237 013230
2375 013634 063737 013224 013230
2376 013642 013737 013234 013224
2377 013650 042737 003777 013224
2378 013656 005737 013212
2379 013662 001403
2380 013664 052737 004000 013224
2381 013672 063737 013224 013230
2382 013700 000207
2383 013702 006237 013230
2384 013706 042737 177701 013230
2385 013714 000207
2386
2387
2388
2389
2390
2391
2392
2393 013716 004737 014004
2394 013722 013737 013230 013224
2395 013730 113737 013220 013230
2396 013736 004737 014004
2397 013742 000337 013230
2398 013746 006237 013230
2399 013752 063737 013224 013230
2400 013760 013737 013234 013224
2401 013766 042737 037777 013224
2402 013774 063737 013224 013230
2403 014002 000207
2404 014004 042737 177600 013230
2405 014012 000207
2406
2407
2408
2409
2410
2411
2412
2413 014014 004737 014004
2414 014020 013737 013230 013224

```

```

RTS PC ;
;*****
;MAKE ADDRESS A LA RESOLUTION 3
;32X32X16
;0 0 0 0 0 Y Y Y Y Y X X X X X 0
;15 14 13 12 11 6 5 4 3 2 6 5 4 3 2
;*****
RES03: JSR PC,M32X16 ;SHIFT AND STRIP
MOV GOOD,XTEMP ;SAVE X
MOV B YADC,GOOD ;GET Y
JSR PC,M32X16 ;GO ALIGN + DELETE UNUSED BITS
SWAB GOOD ;POSITION Y
ASR GOOD ;TO EXACT POSITION
ASR GOOD
ASR GOOD
ADD XTEMP,GOOD ;ADD X VALUE
MOV OFFSET,XTEMP ;GET OFFSET
BIC #3777,XTEMP ;USE BITS 15 14 13 12 11 OF OFFSET
TST DUAL ;B GAMA SET?
BEQ ND3 ;BR IF NOT
BIS #4000,XTEMP ;B GAMA SETS BIT 11
ND3: ADD XTEMP,GOOD ;NOW ADD IN OFFSET
RTS PC
M32X16: ASR GOOD ;ALIGN
BIC #177701,GOOD ;RID UNUSED BITS
RTS PC
;*****
;MAKE ADDRESS A LA RESOLUTION 4
;128X128X8 BASE 12K
;0 0 Y Y Y Y Y Y X X X X X X X
;15 14 6 5 4 3 2 1 0 6 5 4 3 2 1 0
;*****
RES04: JSR PC,M128X8 ;GO DELETE UNUSED BITS
MOV GOOD,XTEMP ;SAVE X
MOV B YADC,GOOD ;GET Y
JSR PC,M128X8 ;GO DELETE UNUSED
SWAB GOOD ;POSITION Y
ASR GOOD ;TO EXACT POSITION
ADD XTEMP,GOOD ;ADD IN X VALUE
MOV OFFSET,XTEMP ;GET OFFSET
BIC #3777,XTEMP ;OFFSET BIT 15,14 ONLY
ADD XTEMP,GOOD ;ADD IN OFFSET
RTS PC
M128X8: BIC #177600,GOOD ;RID UNUSED BITS
RTS PC
;*****
;MAKE ADDRESS A LA RESOLUTION 5
;128X128X8 BASE 4K
;0 Y Y Y Y Y Y X X X X X X X
;15 6 (6) 5 4 3 2 1 0 6 5 4 3 2 1 0
;*****
RES05: JSR PC,M128X8 ;GO DELETE UNUSED BITS
MOV GOOD,XTEMP ;SAVE X

```

```

2415 014026 113737 013220 013230
2416 014034 004737 014004
2417 014040 000337 013230
2418 014044 006337 013230
2419 014050 006237 013230
2420 014054 006237 013230
2421 014060 005737 013230
2422 014064 100004
2423 014066 042737 120000 013230
2424 014074 000403
2425 014076 052737 020000 013230
2426 014104 063737 013224 013230
2427 014112 013737 013234 013224
2428 014120 042737 077777 013224
2429 014126 063737 013224 013230
2430 014134 000207
2431
2432
2433
2434
2435
2436
2437
2438 014136 004737 014244
2439 014142 013737 013230 013224
2440 014150 113737 013220 013230
2441 014156 004737 014244
2442 014162 000337 013230
2443 014166 006237 013230
2444 014172 006237 013230
2445 014176 063737 013224 013230
2446 014204 013737 013234 013224
2447 014212 042737 007777 013224
2448 014220 005737 013212
2449 014224 001403
2450 014226 052737 010000 013224
2451 014234 063737 013224 013230
2452 014242 000207
2453 014244 006037 013230
2454 014250 042737 177700 013230
2455 014256 000207
2456
2457
2458
2459
2460
2461
2462
2463 014260 004737 014372
2464 014264 013737 013230 013224
2465 014272 113737 013220 013230
2466 014300 004737 014372
2467 014304 000337 013230
2468 014310 006237 013230
2469 014314 006237 013230
2470 014320 006237 013230

```

```

MOV B YADC,GOOD ;GET Y
JSR PC,M128X8 ;GO DELETE UNUSED BITS
SWAB GOOD ;POSITION Y
ASL GOOD ;TAKE A LOOK AT Y6
ASR GOOD ;NOW POSITION
ASR GOOD
TST GOOD ;WAS Y6 SET?
BPL SET13 ;BR IF NOT
BIC #120000,GOOD ;YES, ONLY LEAVE 14 SET (Y6=1)
BR DN05 ;SKIP NEXT INSTR
SET13: BIS #20000,GOOD ;Y6=0 SETS 13
DN05: ADD XTEMP,GOOD ;ADD IN X VALUE
MOV OFFSET,XTEMP ;GET OFFSET
BIC #77777,XTEMP ;OFFSET BIT 15 ONLY
ADD XTEMP,GOOD ;ADD IN OFFSET
RTS PC ;

```

```

*****
MAKE ADDRESS A LA RESOLUTION 6
64X64XB
0 0 0 0 Y Y Y Y Y Y X X X X X X
15 14 13 12 6 5 4 3 2 1 6 5 4 3 2 1
*****

```

```

RES06: JSR PC,M64XB ;GO ALIGN & STRIP UNUSED BITS
MOV GOOD,XTEMP ;SAVE X
MOV B YADC,GOOD ;GET Y
JSR PC,M64XB ;GO ALIGN & STRIP UNUSED BITS
SWAB GOOD ;POSITION Y
ASR GOOD ;TO EXACT POSITION
ASR GOOD
ADD XTEMP,GOOD ;ADD IN X VALUE
MOV OFFSET,XTEMP ;GET OFFSET
BIC #7777,XTEMP ;ONLY BITS 15,14,13,12
TST DUAL ;B GAM SET?
BEQ ND6 ;BR IF NOT
ND6: BIS #10000,XTEMP ;YES, B GAM SETS 12
ADD XTEMP,GOOD ;ADD IN OFFSET
RTS PC ;
M64XB: ROR GOOD ;ALIGN
BIC #177700,GOOD ;RID UNUSED BITS
RTS PC ;

```

```

*****
MAKE ADDRESS A LA RESOLUTION 7
32X32XB
0 0 0 0 0 0 Y Y Y Y Y X X X X X
15 14 13 12 11 10 6 5 4 3 2 6 5 4 3 2
*****

```

```

RES07: JSR PC,M32XB ;GO ALIGN & STRIP UNUSED BITS
MOV GOOD,XTEMP ;SAVE X
MOV B YADC,GOOD ;GET Y
JSR PC,M32XB ;GO ALIGN & SAVE UNUSED BITS
SWAB GOOD ;POSITION Y
ASR GOOD ;TO EXACT POSITION
ASR GOOD
ASR GOOD

```

2471	014324	063737	013224	013230					
2472	014332	013737	013234	013224					
2473	014340	042737	001777	013224					
2474	014346	005737	013212						
2475	014352	001403							
2476	014354	052737	002000	013224					
2477	014362	063737	013224	013230	ND7:				
2478	014370	000207							
2479	014372	004737	014244		M32X8:				
2480	014376	006237	013230						
2481	014402	000207							

AUD XTEMP,GOOD	;ADD IN X
MOV OFFSET,XTEMP	;GET OFFSET
BIC #1777,XTEMP	;ONLY BITS 15,14,13,12,11,10
TST DUAL	;B GAM SET?
BEQ ND7	;BR IF NOT
BIS #2000,XTEMP	;YES, B GAM SETS 10
ADD XTEMP,GOOD	;ADD IN OFFSET
RTS PC	
JSR PC,M64X8	;GO ALIGN & STRIP UNUSED BITS
ASR GOOD	;ALIGN
RTS PC	

2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537

.SBTTL SYSMAC ROUTINES

.NLIST MC,MD,CND
.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
*
*

```

014404 105737 001157 $TYPE: TSTB $STPFLG ;; IS THERE A TERMINAL?
014410 100002 BPL 1$ ;; BR IF YES
014412 000000 HALT ;; HALT HERE IF NO TERMINAL
014414 000430 BR 3$ ;; LEAVE
014416 010046 1$: MOV RO,-(SP) ;; SAVE RO
014420 017600 000002 MOV 32(SP),RO ;; GET ADDRESS OF ASCIZ STRING
014424 122737 000001 001214 CMPB #APTENV,$ENV ;; RUNNING IN APT MODE
014432 001011 BNE 62$ ;; NO, GO CHECK FOR APT CONSOLE
014434 132737 000100 001215 BITB #APTPOOL,$ENVM ;; SPOOL MESSAGE TO APT
014442 001405 BEQ 62$ ;; NO, GO CHECK FOR CONSOLE
014444 010037 014454 MOV RO,61$ ;; SETUP MESSAGE ADDRESS FOR APT
014450 004737 014674 JSR PC,$ATY3 ;; SPOOL MESSAGE TO APT
014454 000000 61$: .WORD 0 ;; MESSAGE ADDRESS
014456 132737 000040 001215 62$: BITB #APTCSUP,$ENVM ;; APT CONSOLE SUPPRESSED
014464 001003 BNE 60$ ;; YES, SKIP TYPE OUT
014466 112046 2$: MOVB (RO)+,-(SP) ;; PUSH CHARACTER TO BE TYPED ONTO STACK
014470 001005 BNE 4$ ;; BR IF IT ISN'T THE TERMINATOR
014472 005726 TST (SP)+ ;; IF TERMINATOR POP IT OFF THE STACK
014474 012600 60$: MOV (SP)+,RO ;; RESTORE RO
014476 062716 000002 3$: ADD #2,(SP) ;; ADJUST RETURN PC
014502 000002 RTI ;; RETURN
014504 122716 000011 4$: CMPB #HT,(SP) ;; BRANCH IF <HT>
014510 001430 BEQ 8$
014512 122716 000200 CMPB #CRLF,(SP) ;; BRANCH IF NOT <CRLF>
014516 001006 BNE 5$
014520 005726 TST (SP)+ ;; POP <CR><LF> EQUIV
014522 104400 TYPE ;; TYPE A CR AND LF
014524 001171 $CRLF
014526 105037 014662 CLRB $CHARCNT ;; CLEAR CHARACTER COUNT
014532 000755 BR 2$ ;; GET NEXT CHARACTER
014534 004737 014616 5$: JSR PC,$TYPEC ;; GO TYPE THIS CHARACTER
014540 123726 001156 6$: CMPB $FILLC,(SP)+ ;; IS IT TIME FOR FILLER CHARS.?
014544 001350 BNE 2$ ;; IF NO GO GET NEXT CHAR.
014546 013746 001154 MOV $NULL,-(SP) ;; GET # OF FILLER CHARS. NEEDED
2536 014552 105366 000001 7$: DECB 1(SP) ;; AND THE NULL CHAR.
;; DOES A NULL NEED TO BE TYPED?

```

2538	014556	002770		BLT	6\$:: BR IF NO--GO POP THE NULL OFF OF STACK
2539	014560	004737	014616	JSR	PC,\$TYPEC		:: GO TYPE A NULL
2540	014564	105337	014662	DECB	\$CHARCNT		:: DO NOT COUNT AS A COUNT
2541	014570	000770		BR	7\$:: LOOP

; HORIZONTAL TAB PROCESSOR

2542							
2543							
2544							
2545	014572	112716	000040	8\$:	MOVB	#' (SP)	:: REPLACE TAB WITH SPACE
2546	014576	004737	014616	9\$:	JSR	PC,\$TYPEC	:: TYPE A SPACE
2547	014602	132737	000007		BITB	#7,\$CHARCNT	:: BRANCH IF NOT AT
2548	014610	001372			BNE	9\$:: TAB STOP
2549	014612	005726			TST	(SP)+	:: POP SPACE OFF STACK
2550	014614	000724			BR	2\$:: GET NEXT CHARACTER
2551	014616	105777	164326	\$TYPEC:	TSTB	\$STPB	:: WAIT UNTIL PRINTER IS READY
2552	014622	100375			BPL	\$TYPEC	
2553	014624	116677	000002		MOVB	2(SP), \$STPB	:: LOAD CHAR TO BE TYPED INTO DATA REG.
2554	014632	122766	000015		CMPB	#CR, 2(SP)	:: IS CHARACTER A CARRIAGE RETURN?
2555	014640	001003			BNE	1\$:: BRANCH IF NO
2556	014642	105037	014662		CLRB	\$CHARCNT	:: YES--CLEAR CHARACTER COUNT
2557	014646	000406			BR	\$TYPEX	:: EXIT
2558	014650	122766	000012	1\$:	CMPB	#LF, 2(SP)	:: IS CHARACTER A LINE FEED?
2559	014656	001402			BEQ	\$TYPEX	:: BRANCH IF YES
2560	014660	105227			INCB	(PC)+	:: COUNT THE CHARACTER
2561	014662	000000		\$CHARCNT:	WORD	0	:: CHARACTER COUNT STORAGE
2562	014664	000207		\$TYPEX:	RTS	PC	

.SBTTL APT COMMUNICATIONS ROUTINE

2563								
2564								
2565								
2566								
2567	014666	112737	000001	015132	\$SATY1:	MOVB	#1,\$FFLG	:: TO REPORT FATAL ERROR
2568	014674	112737	000001	015130	\$SATY3:	MOVB	#1,\$MFLG	:: TO TYPE A MESSAGE
2569	014702	000403				BR	\$ATYC	
2570	014704	112737	000001	015132	\$SATY4:	MOVB	#1,\$FFLG	:: TO ONLY REPORT FATAL ERROR
2571	014712				\$SATYC:			
2572	014712	010046				MOV	RO, -(SP)	:: PUSH RO ON STACK
2573	014714	010146				MOV	R1, -(SP)	:: PUSH R1 ON STACK
2574	014716	105737	015130			TSTB	\$MFLG	:: SHOULD TYPE A MESSAGE?
2575	014722	001450				BEQ	5\$:: IF NOT: BR
2576	014724	122737	000001	001214		CMPB	#APTENV, \$ENV	:: OPERATING UNDER APT?
2577	014732	001031				BNE	3\$:: IF NOT: BR
2578	014734	132737	000100	001215		BITB	#APTPOOL, \$ENVM	:: SHOULD SPOOL MESSAGES?
2579	014742	001425				BEQ	3\$:: IF NOT: BR
2580	014744	017600	000004			MOV	24(SP), RO	:: GET MESSAGE ADDR.
2581	014750	062766	000002	000004		ADD	#2, 4(SP)	:: BUMP RETURN ADDR.
2582	014756	005737	001174		1\$:	TST	\$MSGTYPE	:: SEE IF DONE W/ LAST XMISSION?
2583	014762	001375				BNE	1\$:: IF NOT: WAIT
2584	014764	010037	001210			MOV	RO, \$MSGAD	:: PUT ADDR IN MAILBOX
2585	014770	105720			2\$:	TSTB	(RO)+	:: FIND END OF MESSAGE
2586	014772	001376				BNE	2\$	
2587	014774	163700	001210			SUB	\$MSGAD, RO	:: SUB START OF MESSAGE
2588	015000	006200				ASR	RO	:: GET MESSAGE LNTH IN WORDS
2589	015002	010037	001212			MOV	RO, \$MSGGLT	:: PUT LENGTH IN MAILBOX
2590	015006	012737	000004	001174		MOV	#4, \$MSGTYPE	:: TELL APT TO TAKE MSG.
2591	015014	000413				BR	5\$	
2592	015016	017637	000004	015042	3\$:	MOV	24(SP), 4\$:: PUT MSG ADDR IN JSR LINKAGE
2593	015024	062766	000002	000004		ADD	#2, 4(SP)	:: BUMP RETURN ADDRESS

```

2594 015032 013746 177776      MOV      177776, -(SP)      ;; PUSH 177776 ON STACK
2595 015036 004737 014404      JSR      PC, $TYPE        ;; CALL TYPE MACRO
2596 015042 000000                4$:      .WORD      0
2597 015044                5$:
2598 015044 105737 015132      10$:    TSTB     $FFLG        ;; SHOULD REPORT FATAL ERROR?
2599 015050 001416                BEQ     12$              ;; IF NOT: BR
2600 015052 005737 001214      TST     $ENV             ;; RUNNING UNDER APT?
2601 015056 001413                BEQ     12$              ;; IF NOT: BR
2602 015060 005737 001174      11$:    TST     $MSGTYPE     ;; FINISHED LAST MESSAGE?
2603 015064 001375                BNE     11$              ;; IF NOT: WAIT
2604 015066 017637 000004 001176  MOV     24(SP), $FATAL    ;; GET ERROR #
2605 015074 062766 000002 000004  ADD     #2, 4(SP)        ;; BUMP RETURN ADDR.
2606 015102 005237 001174      INC     $MSGTYPE        ;; TELL APT TO TAKE ERROR
2607 015106 105037 015132      12$:    CLRB     $FFLG        ;; CLEAR FATAL FLAG
2608 015112 105037 015131      CLRB     $LFLG         ;; CLEAR LOG FLAG
2609 015116 105037 015130      CLRB     $MFLG         ;; CLEAR MESSAGE FLAG
2610 015122 012601                MOV     (SP)+, R1        ;; POP STACK INTO R1
2611 015124 012600                MOV     (SP)+, R0        ;; POP STACK INTO R0
2612 015126 000207                RTS     PC                ;; RETURN
2613 015130 000                $MFLG: .BYTE      0        ;; MESSG. FLAG
2614 015131 000                $LFLG: .BYTE      0        ;; LOG FLAG
2615 015132 000                $FFLG: .BYTE      0        ;; FATAL FLAG

```

```

APTSIZE=200
APTENV=001
APTSPool=100
APTCSUP=040
.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

```

```

*$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
*$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

```

```

2646 015134 017646 000000 015357  $TYPOS: MOV     2(SP), -(SP)      ;; PICKUP THE MODE
2647 015140 116637 000001  MOVB    1(SP), $OFILL     ;; LOAD ZERO FILL SWITCH
2648 015146 112637 015361  MOVB    (SP)+, $OMODE+1  ;; NUMBER OF DIGITS TO TYPE
2649 015152 062716 000002  ADD     #2, (SP)         ;; ADJUST RETURN ADDRESS

```

```

2650 015156 000406 BR STYPON
2651 015160 112737 000001 015357 STYPOC: MOVB #1,$OFILL ;; SET THE ZERO FILL SWITCH
2652 015166 112737 000006 015361 MOVB #6,$OMODE+1 ;; SET FOR SIX(6) DIGITS
2653 015174 112737 000005 015356 STYPON: MOVB #5,$OCNT ;; SET THE ITERATION COUNT
2654 015202 010346 MOV R3,-(SP) ;; SAVE R3
2655 015204 010446 MOV R4,-(SP) ;; SAVE R4
2656 015206 010546 MOV R5,-(SP) ;; SAVE R5
2657 015210 113704 015361 MOVB $OMODE+1,R4 ;; GET THE NUMBER OF DIGITS TO TYPE
2658 015214 005404 NEG R4
2659 015216 062704 000006 ADD #6,R4 ;; SUBTRACT IT FOR MAX. ALLOWED
2660 015222 110437 015360 MOVB R4,$OMODE ;; SAVE IT FOR USE
2661 015226 113704 015357 MOVB $OFILL,R4 ;; GET THE ZERO FILL SWITCH
2662 015232 016605 000012 MOV 12(SP),R5 ;; PICKUP THE INPUT NUMBER
2663 015236 005003 CLR R3 ;; CLEAR THE OUTPUT WORD
2664 015240 006105 1$: ROL R5 ;; ROTATE MSB INTO "C"
2665 015242 000404 BR 3$ ;; GO DO MSB
2666 015244 006105 2$: ROL R5 ;; FORM THIS DIGIT
2667 015246 006105 ROL R5
2668 015250 006105 ROL R5
2669 015252 010503 MOV R5,R3
2670 015254 006103 3$: ROL R3 ;; GET LSB OF THIS DIGIT
2671 015256 105337 015360 DECB $OMODE ;; TYPE THIS DIGIT?
2672 015262 100016 BPL 7$ ;; BR IF NO
2673 015264 042703 177770 BIC #177770,R3 ;; GET RID OF JUNK
2674 015270 001002 BNE 4$ ;; TEST FOR 0
2675 015272 005704 TST R4 ;; SUPPRESS THIS 0?
2676 015274 001403 BEQ 5$ ;; BR IF YES
2677 015276 005204 4$: INC R4 ;; DON'T SUPPRESS ANYMORE 0'S
2678 015300 052703 000060 BIS #'0,R3 ;; MAKE THIS DIGIT ASCII
2679 015304 052703 000040 5$: BIS #' ,R3 ;; MAKE ASCII IF NOT ALREADY
2680 015310 110337 015354 MOVB R3,$$ ;; SAVE FOR TYPING
2681 015314 104400 015354 TYPE $$ ;; GO TYPE THIS DIGIT
2682 015320 105337 015356 7$: DECB $OCNT ;; COUNT BY 1
2683 015324 003347 BGT 2$ ;; BR IF MORE TO DO
2684 015326 002402 BLT 6$ ;; BR IF DONE
2685 015330 005204 INC R4 ;; INSURE LAST DIGIT ISN'T A BLANK
2686 015332 000744 BR 2$ ;; GO DO THE LAST DIGIT
2687 015334 012605 6$: MOV (SP)+,R5 ;; RESTORE R5
2688 015336 012604 MOV (SP)+,R4 ;; RESTORE R4
2689 015340 012603 MOV (SP)+,R3 ;; RESTORE R3
2690 015342 016666 000002 000004 MOV 2(SP),4(SP) ;; SET THE STACK FOR RETURNING
2691 015350 012616 MOV (SP)+,(SP)
2692 015352 000002 RTI ;; RETURN
2693 015354 000 8$: .BYTE 0 ;; STORAGE FOR ASCII DIGIT
2694 015355 000 .BYTE 0 ;; TERMINATOR FOR TYPE ROUTINE
2695 015356 000 $OCNT: .BYTE 0 ;; OCTAL DIGIT COUNTER
2696 015357 000 $OFILL: .BYTE 0 ;; ZERO FILL SWITCH
2697 015360 000000 $OMODE: .WORD 0 ;; NUMBER OF DIGITS TO TYPE
2698
2699
2700
2701
2702
2703
2704
2705

```

```

*****
*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.

```



```

2706          : *CALL:
2707          : *      MOV      NUM, -(SP)          ;; PUT THE BINARY NUMBER ON THE STACK
2708          : *      TYPDS          ;; GO TO THE ROUTINE
2709
2710          $TYPDS:
2711          015362 010046          MOV      R0, -(SP)          ;; PUSH R0 ON STACK
2712          015364 010146          MOV      R1, -(SP)          ;; PUSH R1 ON STACK
2713          015366 010246          MOV      R2, -(SP)          ;; PUSH R2 ON STACK
2714          015370 010346          MOV      R3, -(SP)          ;; PUSH R3 ON STACK
2715          015372 010546          MOV      R5, -(SP)          ;; PUSH R5 ON STACK
2716          015374 012746 020200  MOV      #20200, -(SP)      ;; SET BLANK SWITCH AND SIGN
2717          015400 016605 000020  MOV      20(SP), R5        ;; GET THE INPUT NUMBER
2718          015404 100004          SPL          ;; BR IF INPUT IS POS.
2719          015406 005405          NEG      R5                ;; MAKE THE BINARY NUMBER POS.
2720          015410 112766 000055 000001  MOVB     #'-, 1(SP)        ;; MAKE THE ASCII NUMBER NEG.
2721          015416 005000          CLR      R0                ;; ZERO THE CONSTANTS INDEX
2722          015420 012703 015576  MOV      #SDBLK, R3        ;; SETUP THE OUTPUT POINTER
2723          015424 112723 000040  MOVB     #' , (R3)+        ;; SET THE FIRST CHARACTER TO A BLANK
2724          015430 005002          CLR      R2                ;; CLEAR THE BCD NUMBER
2725          015432 016001 015566  MOV      $DTBL(R0), R1     ;; GET THE CONSTANT
2726          015436 160105          SUB      R1, R5            ;; FORM THIS BCD DIGIT
2727          015440 002402          BLT     4$                ;; BR IF DONE
2728          015442 005202          INC     R2                ;; INCREASE THE BCD DIGIT BY 1
2729          015444 000774          BR      3$
2730          015446 060105          4$: ADD     R1, R5            ;; ADD BACK THE CONSTANT
2731          015450 005702          TST     R2                ;; CHECK IF BCD DIGIT=0
2732          015452 001002          BNE     5$                ;; FALL THROUGH IF 0
2733          015454 105716          TSTB   (SP)               ;; STILL DOING LEADING 0'S?
2734          015456 100407          BMI     7$                ;; BR IF YES
2735          015460 106316          5$: ASLB   (SP)            ;; MSD?
2736          015462 103003          BCC     6$                ;; BR IF NO
2737          015464 116663 000001 177777  MOVB     1(SP), -1(R3)     ;; YES--SET THE SIGN
2738          015472 052702 000060 6$: BIS   #'0, R2          ;; MAKE THE BCD DIGIT ASCII
2739          015476 052702 000040 7$: BIS   #' , R2          ;; MAKE IT A SPACE IF NOT ALREADY A DIGIT
2740          015502 110223          MOVB     R2, (R3)+        ;; PUT THIS CHARACTER IN THE OUTPUT BUFFER
2741          015504 005720          TST     (R0)+            ;; JUST INCREMENTING
2742          015506 020027 000010  CMP     R0, #10          ;; CHECK THE TABLE INDEX
2743          015512 002746          BLT     2$                ;; GO DO THE NEXT DIGIT
2744          015514 003002          BGT     8$                ;; GO TO EXIT
2745          015516 010502          MOV     R5, R2            ;; GET THE LSD
2746          015520 000764          BR      6$                ;; GO CHANGE TO ASCII
2747          015522 105726          8$: TSTB   (SP)+          ;; WAS THE LSD THE FIRST NON-ZERO?
2748          015524 100003          BPL     9$                ;; BR IF NO
2749          015526 116663 177777 177776  MOVB     -1(SP), -2(R3)   ;; YES--SET THE SIGN FOR TYPING
2750          015534 105013          9$: CLRB   (R3)            ;; SET THE TERMINATOR
2751          015536 012605          MOV     (SP)+, R5        ;; POP STACK INTO R5
2752          015540 012603          MOV     (SP)+, R3        ;; POP STACK INTO R3
2753          015542 012602          MOV     (SP)+, R2        ;; POP STACK INTO R2
2754          015544 012601          MOV     (SP)+, R1        ;; POP STACK INTO R1
2755          015546 012600          MOV     (SP)+, R0        ;; POP STACK INTO R0
2756          015550 104400 015576  TYPE     $SDBLK          ;; NOW TYPE THE NUMBER
2757          015554 016666 000002 000004  MOV     2(SP), 4(SP)      ;; ADJUST THE STACK
2758          015562 012616          MOV     (SP)+, (SP)
2759          015564 000002          RTI
2760          015566 023420          $DTBL: 10000.
2761          015570 001750          1000.

```

2762 015572 000144
2763 015574 000012
2764 015576 000004
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779 015606
2780 015606 104406
2781 015610 104406
2782 015612 105237 001103
2783 015616 001775
2784 015620 013777 001102 163314
2785 015626 032777 002000 163304
2786 015634 001402
2787 015636 104400 001164
2788 015642 005237 001112
2789 015646 011637 001116
2790 015652 162737 000002 001116
2791 015660 117737 163232 001114
2792 015666 032777 020000 163244
2793 015674 001004
2794 015676 004737 015776
2795 015702 104400 001171
2796 015706
2797 015706 122737 000001 001214
2798 015714 001007
2799 015716 113737 001114 015730
2800 015724 004737 014704
2801 015730 000
2802 015731 000
2803 015732 000777
2804 015734 005777 163200
2805 015740 100002
2806 015742 000000
2807 015744 104406
2808 015746 032777 001000 163164
2809 015754 001402
2810 015756 013716 001110
2811 015762 005737 001162
2812 015766 001402
2813 015770 013716 001162
2814 015774
2815 015774 000002
2816
2817

```
100.  
10.  
$DBLK: .BLKW 4  
.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 SWRCK 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  
$ERROR:  
CKSWR ;;TEST FOR CHANGE IN SOFT-SWR  
CKSWR  
7$: INCB $ERFLG ;;SET THE ERROR FLAG  
BEQ 7$ ;;DON'T LET THE FLAG GO TO ZERO  
MOV $STNM,$DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG  
BIT #BIT10,$SWR ;;BELL ON ERROR?  
BEQ 1$ ;;NO - SKIP  
TYPE $BELL ;;RING BELL  
1$: INC $ERTTL ;;COUNT THE NUMBER OF ERRORS  
MOV (SP),$ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION  
SUB #2,$ERRPC  
MOVB $ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE  
BIT #BIT13,$SWR ;;SKIP TYPEOUT IF SET  
BNE 20$ ;;SKIP TYPEOUTS  
JSR PC,$SWRCK ;;GO TO USER ERROR ROUTINE  
TYPE $CRLF  
20$: CMPB #APTENV,$ENV ;;RUNNING IN APT MODE  
BNE 2$ ;;NO SKIP APT ERROR REPORT  
MOVB $ITEMB,21$ ;;SET ITEM NUMBER AS ERROR NUMBER  
JSR PC,$ATY4 ;;REPORT FATAL ERROR TO APT  
21$: .BYTE 0  
.BYTE 0  
22$: BR 22$ ;;APT ERROR LOOP  
2$: TST $SWR ;;HALT ON ERROR  
BPL 3$ ;;SKIP IF CONTINUE  
HALT ;;HALT ON ERROR!  
CKSWR ;;TEST FOR CHANGE IN SOFT-SWR  
3$: BIT #BIT09,$SWR ;;LOOP ON ERROR SWITCH SET?  
BEQ 4$ ;;BR IF NO  
MOV $LPERR,(SP) ;;FUDGE RETURN FOR LOOPING  
4$: TST $ESCAPE ;;CHECK FOR AN ESCAPE ADDRESS  
BEQ 5$ ;;BR IF NONE  
MOV $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE  
5$: RTI ;;RETURN  
*****  
;GO TYPE ERROR
```

```

2818
2819
2820 015776 113737 001102 001600
2821 016004 004737 016014
2822 016010 104406
2823 016012 000207
2824
2825
2826
2827
2828
2829
2830
2831 016014
2832 016014 104400 001171
2833 016020 010046
2834 016022 005000
2835 016024 153700 001114
2836 016030 001004
2837
2838 016032 013746 001116
2839
2840 016036 104401
2841 016040 000426
2842 016042 005300
2843 016044 006300
2844 016046 006300
2845 016050 006300
2846 016052 062700 001256
2847 016056 012037 016066
2848 016062 001404
2849 016064 104400
2850 016066 000000
2851 016070 104400 001171
2852 016074 012037 016104
2853 016100 001404
2854 016102 104400
2855 016104 000000
2856 016106 104400 001171
2857 016112 011000
2858 016114 001004
2859 016116 012600
2860 016120 104400 001171
2861 016124 000207
2862 016126
2863 016126 013046
2864 016130 104401
2865 016132 005710
2866 016134 001770
2867 016136 104400 016144
2868 016142 000771
2869 016144 020040 000
2870 016150
2871
2872
2873

```

```

;GO UPDATE SOFTWARE SWR IF 'CNTRL/G'
;*****
SWRCK:  MOV  $STNM,TSTNUM  ;SET UP TEST # ON ER
        JSR  PC,SERRTYP   ;GO TYPE ERROR
        CKSWR              ;LOOK AT KEYBOARD
        RTS  PC           ;RETURN TO ERROR HANDLER
.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.

SERRTYP:
        TYPE  $SCLF        ;;"CARRIAGE RETURN" & "LINE FEED"
        MOV  RO,-(SP)     ;;SAVE RO
        CLR  RO           ;;PICKUP THE ITEM INDEX
        BISB 2($ITEMB,RO
        BNE  1$          ;;IF ITEM NUMBER IS ZERO, JUST
                        ;;TYPE THE PC OF THE ERROR
        MOV  $ERRPC,-(SP) ;;SAVE $ERRPC FOR TYPEOUT
                        ;;ERROR ADDRESS
        TYPOC            ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
        BR   6$          ;;GET OUT
1$:     DEC  RO           ;;ADJUST THE INDEX SO THAT IT WILL
        ASL  RO           ;;WORK FOR THE ERROR TABLE
        ASL  RO
        ASL  RO
        ADD  # $ERRTB,RO  ;;FORM TABLE POINTER
        MOV  (RO)+,2$    ;;PICKUP "ERROR MESSAGE" POINTER
        BEQ  3$          ;;SKIP TYPEOUT IF NO POINTER
        TYPE "ERROR MESSAGE"
                        ;;"ERROR MESSAGE" POINTER GOES HERE
        TYPE $SCLF      ;;"CARRIAGE RETURN" & "LINE FEED"
        BEQ  3$          ;;PICKUP "DATA HEADER" POINTER
        TYPE "DATA HEADER"
                        ;;"DATA HEADER" POINTER GOES HERE
2$:     .WORD 0          ;;"CARRIAGE RETURN" & "LINE FEED"
        TYPE $SCLF      ;;"CARRIAGE RETURN" & "LINE FEED"
        BEQ  5$          ;;PICKUP "DATA TABLE" POINTER
        MOV  (RO),RO     ;;GO TYPE THE DATA
        BNE  7$          ;;RESTORE RO
        BEQ  5$          ;;"CARRIAGE RETURN" & "LINE FEED"
        TYPE "DATA TABLE"
        RTS  PC         ;;RETURN
7$:     MOV  2(RO)+,-(SP) ;;SAVE 2(RO)+ FOR TYPEOUT
        TYPOC            ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
        TST  (RO)        ;;IS THERE ANOTHER NUMBER?
        BEQ  6$          ;;BR IF NO
        TYPE " "         ;;TYPE TWO(2) SPACES
        BR   7$          ;;LOOP
8$:     .ASCIZ "/ /"    ;;TWO(2) SPACES
        .EVEN
.SBTTL  SCOPE HANDLER ROUTINE

;*****

```

```

2874 ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
2875 ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
2876 ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
2877 ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
2878 ;*SW14=1 LOOP ON TEST
2879 ;*SW11=1 INHIBIT ITERATIONS
2880 ;*SW09=1 LOOP ON ERROR
2881 ;*SW08=1 LOOP ON TEST IN SWR<7:0>
2882 ;*CALL
2883 ;* SCOPE ;;SCOPE=IOT
2884
2885 $SCOPE:
2886 016150 104406 CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
2887 016152 104406 CKSWR
2888 016154 032777 040000 162756 1$: BIT #BIT14,$SWR ;;LOOP ON PRESENT TEST?
2889 016162 001114 BNE $OVER ;;YES IF SW14=1
2890 ;*START OF CODE FOR THE XOR TESTER*****
2891 016164 000416 $XTSTR: BR 6$ ;;IF RUNNING ON THE "XOR" TESTER CHANGE
2892 ;*THIS INSTRUCTION TO A "NOP" (NOP=240)
2893 016166 013746 000004 MOV @#ERRVEC,-(SP) ;;SAVE THE CONTENTS OF THE ERROR VECTOR
2894 016172 012737 016212 000004 MOV #5$,@#ERRVEC ;;SET FOR TIMEOUT
2895 016200 005737 177060 TST @#177060 ;;TIME OUT ON XOR?
2896 016204 012637 000004 MOV (SP)+,@#ERRVEC ;;RESTORE THE ERROR VECTOR
2897 016210 000463 BR $SVLAD ;;GO TO THE NEXT TEST
2898 016212 022626 5$: CMP (SP)+,(SP)+ ;;CLEAR THE STACK AFTER A TIME OUT
2899 016214 012637 000004 MOV (SP)+,@#ERRVEC ;;RESTORE THE ERROR VECTOR
2900 016220 000423 BR 7$ ;;LOOP ON THE PRESENT TEST
2901 016222 6$;*****END OF CODE FOR THE XOR TESTER*****
2902 016222 032777 000400 162710 BIT #BIT08,$SWR ;;LOOP ON SPEC. TEST?
2903 016230 001404 BEQ 2$ ;;BR IF NO
2904 016232 127737 162702 001102 CMPB @SWR,$TSTNM ;;ON THE RIGHT TEST? SWR<7:0>
2905 016240 001465 BEQ $OVER ;;BR IF YES
2906 016242 105737 001103 2$: TSTB $ERFLG ;;HAS AN ERROR OCCURRED?
2907 016246 001421 BEQ 3$ ;;BR IF NO
2908 016250 123737 001115 001103 CMPB $ERMAX,$ERFLG ;;MAX. ERRORS FOR THIS TEST OCCURRED?
2909 016256 101015 BHI 3$ ;;BR IF NO
2910 016260 032777 001000 162652 BIT #BIT09,$SWR ;;LOOP ON ERROR?
2911 016266 001404 BEQ 4$ ;;BR IF NO
2912 016270 013737 001110 001106 7$: MOV $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
2913 016276 000446 BR $OVER
2914 016300 105037 001103 4$: CLRB $ERFLG ;;ZERO THE ERROR FLAG
2915 016304 005037 001160 CLR $TIMES ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
2916 016310 000415 BR 1$ ;;ESCAPE TO THE NEXT TEST
2917 016312 032777 004000 162620 3$: BIT #BIT11,$SWR ;;INHIBIT ITERATIONS?
2918 016320 001011 BNE 1$ ;;BR IF YES
2919 016322 005737 001202 TST $PASS ;;IF FIRST PASS OF PROGRAM
2920 016326 001406 BEQ 1$ ;;INHIBIT ITERATIONS
2921 016330 005237 001104 INC $ICNT ;;INCREMENT ITERATION COUNT
2922 016334 023737 001160 001104 CMP $TIMES,$ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
2923 016342 002024 BGE $OVER ;;BR IF MORE ITERATION REQUIRED
2924 016344 012737 000001 001104 1$: MOV #1,$ICNT ;;REINITIALIZE THE ITERATION COUNTER
2925 016352 013737 016430 001160 MOV $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
2926 016360 105237 001102 $SVLAD: INCB $TSTNM ;;COUNT TEST NUMBERS
2927 016364 113737 001102 001200 MOVB $TSTNM,$STESTN ;;SET TEST NUMBER IN APT MAILBOX
2928 016372 011637 001106 MOV (SP),$LPADR ;;SAVE SCOPE LOOP ADDRESS
2929 016376 011637 001110 MOV (SP),$LPERR ;;SAVE ERROR LOOP ADDRESS

```

```

2930 016402 005037 001162          CLR      $ESCAPE          ;; CLEAR THE ESCAPE FROM ERROR ADDRESS
2931 016406 112737 000001 001115    MOV      #1,$ERMAX        ;; ONLY ALLOW ONE(1) ERROR ON NEXT TEST
2932 016414 013777 001102 162520 $OVER:  MOV      $STNM,$DISPLAY ;; DISPLAY TEST NUMBER
2933 016422 013716 001106          MOV      $LPADR,(SP)     ;; FUDGE RETURN ADDRESS
2934 016426 000002          RTI                          ;; FIXES PS
2935 016430 003720          $MXCNT: 2000             ;; MAX. NUMBER OF ITERATIONS
2936          .SBTTL  TTY INPUT ROUTINE
2937
2938          ;;*****
2939          .ENABL  LSB
2940
2941          ;;*****
2942          ;;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
2943          ;;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
2944          ;;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
2945          ;;*WHEN OPERATING IN TTY FLAG MODE.
2946 016432 022737 000176 001140 $CKSWR: CMP      #SWREG,SWR      ;; IS THE SOFT-SWR SELECTED?
2947 016440 001074          BNE      15$             ;; BRANCH IF NO
2948 016442 105777 162476          TSTB    $STKS           ;; CHAR THERE?
2949 016446 100071          BPL      15$             ;; IF NO, DON'T WAIT AROUND
2950 016450 117746 162472          MOV      $STKB,-(SP)    ;; SAVE THE CHAR
2951 016454 042716 177600          BIC     #17,(SP)        ;; STRIP-OFF THE ASCII
2952 016460 022726 000007          CMP     #7,(SP)         ;; IS IT A CONTROL G?
2953 016464 001062          BNE      15$             ;; NO, RETURN TO USER
2954 016466 123727 001134 000001  CMP      $AUTOB,#1      ;; ARE WE RUNNING IN AUTO-MODE?
2955 016474 001456          BEQ     15$             ;; BRANCH IF YES
2956
2957 016476 104400 017157          $GTSWR: TYPE    ,SCNTLG    ;; ECHO THE CONTROL-G (↑G)
2958 016502 104400 017164          TYPE    ,SMSWR         ;; TYPE CURRENT CONTENTS
2959 016506 013746 000176          MOV     SWREG,-(SP)    ;; SAVE SWREG FOR TYPEOUT
2960 016512 104401          TYPOC   ,MNEW          ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
2961 016514 104400 017175          TYPE    ,MNEW          ;; PROMPT FOR NEW SWR
2962 016520 005046          19$:   CLR     -(SP)    ;; CLEAR COUNTER
2963 016522 005046          CLR     -(SP)          ;; THE NEW SWR
2964 016524 105777 162414          7$:   TSTB    $STKS      ;; CHAR THERE?
2965 016530 100375          BPL     7$             ;; IF NOT TRY AGAIN
2966
2967 016532 117746 162410          MOV     $STKB,-(SP)    ;; PICK UP CHAR
2968 016536 042716 177600          BIC     #17,(SP)      ;; MAKE IT 7-BIT ASCII
2969
2970
2971
2972 016542 021627 000025          9$:   CMP     (SP),#25    ;; IS IT A CONTROL-U?
2973 016546 001005          BNE     10$            ;; BRANCH IF NOT
2974 016550 104400 017152          TYPE    ,SCNTLU        ;; YES, ECHO CONTROL-U (↑U)
2975 016554 062706 000006          20$:  ADD     #6,SP        ;; IGNORE PREVIOUS INPUT
2976 016560 000757          BR      19$            ;; LET'S TRY IT AGAIN
2977
2978
2979 016562 021627 000015          10$:  CMP     (SP),#15      ;; IS IT A <CR>?
2980 016566 001022          BNE     16$            ;; BRANCH IF NO
2981 016570 005766 000004          TST     4(SP)          ;; YES, IS IT THE FIRST CHAR?
2982 016574 001403          BEQ     11$            ;; BRANCH IF YES
2983 016576 016677 000002 162334  MOV     2(SP), $SWR    ;; SAVE NEW SWR
2984 016604 062706 000006          11$:  ADD     #6,SP        ;; CLEAR UP STACK
2985 016610 104400 001171          14$:  TYPE    ,SCRLF        ;; ECHO <CR> AND <LF>

```

```

2986 016614 123727 001135 000001      CMPB   $INTAG,#1      ;;RE-ENABLE TTY KBD INTERRUPTS?
2987 016622 001003                    BNE    15$           ;;BRANCH IF NOT
2988 016624 012777 000100 162312      MOV    #100,$STKS    ;;RE-ENABLE TTY KBD INTERRUPTS
2989 016632 000002                    RTI                      ;;RETURN
2990 016634 004737 014616                    JSR    PC,$TYPEC     ;;ECHO CHAR
2991 016640 021627 000060                    CMP    (SP),#60      ;;CHAR < 0?
2992 016644 002420                    BLT    18$           ;;BRANCH IF YES
2993 016646 021627 000067                    CMP    (SP),#67      ;;CHAR > ??
2994 016652 003015                    BGT    18$           ;;BRANCH IF YES
2995 016654 042726 000060                    BIC    #60,(SP)+     ;;STRIP-OFF ASCII
2996 016660 005766 000002                    TST    2(SP)         ;;IS THIS THE FIRST CHAR
2997 016664 001403                    BEQ    17$           ;;BRANCH IF YES
2998 016666 006316                    ASL    (SP)          ;;NO, SHIFT PRESENT
2999 016670 006316                    ASL    (SP)          ;;CHAR OVER TO MAKE
3000 016672 006316                    ASL    (SP)          ;;ROOM FOR NEW ONE.
3001 016674 005266 000002                    17$: INC    2(SP)     ;;KEEP COUNT OF CHAR
3002 016700 056616 177776                    BIS    -2(SP),(SP)   ;;SET IN NEW CHAR
3003 016704 000707                    BR     7$            ;;GET THE NEXT ONE
3004 016706 104400 001170                    18$: TYPE  $QUES     ;;TYPE ?<CR><LF>
3005 016712 000720                    BR     20$          ;;SIMULATE CONTROL-U
3006 .DSABL  LSB

```

3009 *****
3010 *THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY

3011 *CALL:
3012 * RDCHR ;; INPUT A SINGLE CHARACTER FROM THE TTY
3013 * RETURN HERE ;; CHARACTER IS ON THE STACK
3014 * ;; WITH PARITY BIT STRIPPED OFF
3015 *

```

3017 016714 011646                    $RDCHR: MOV    (SP),-(SP)   ;;PUSH DOWN THE PC
3018 016716 016666 000004 000002      MOV    4(SP),2(SP)  ;;SAVE THE PS
3019 016724 105777 162214                    1$: TSTB  $STKS      ;;WAIT FOR
3020 016730 100375                    BPL    1$           ;;A CHARACTER
3021 016732 117766 162210 000004      MOVB  $STKB,4(SP)   ;;READ THE TTY
3022 016740 042766 177600 000004      BIC   #177,4(SP)    ;;GET RID OF JUNK IF ANY
3023 016746 026627 000004 000023      CMP   4(SP),#23     ;;IS IT A CONTROL-S?
3024 016754 001013                    BNE    3$           ;;BRANCH IF NO
3025 016756 105777 162162                    2$: TSTB  $STKS      ;;WAIT FOR A CHARACTER
3026 016762 100375                    BPL    2$           ;;LOOP UNTIL ITS THERE
3027 016764 117746 162156                    MOVB  $STKB,-(SP)   ;;GET CHARACTER
3028 016770 042716 177600                    BIC   #177,(SP)    ;;MAKE IT 7-BIT ASCII
3029 016774 022627 000021                    CMP   (SP)+,#21     ;;IS IT A CONTROL-Q?
3030 017000 001366                    BNE    2$           ;;IF NOT DISCARD IT
3031 017002 000750                    BR     1$           ;;YES, RESUME
3032 017004 026627 000004 000140      3$: CMP   4(SP),#140  ;;IS IT UPPER CASE?
3033 017012 002407                    BLT    4$           ;;BRANCH IF YES
3034 017014 026627 000004 000175      CMP   4(SP),#175    ;;IS IT A SPECIAL CHAR?
3035 017022 003003                    BGT    4$           ;;BRANCH IF YES
3036 017024 042766 000040 000004      BIC   #40,4(SP)     ;;MAKE IT UPPER CASE
3037 017032 000002                    4$: RTI              ;;GO BACK TO USER

```

3038 *****
3039 *THIS ROUTINE WILL INPUT A STRING FROM THE TTY

3040 *CALL:
3041 * RDLIN ;; INPUT A STRING FROM THE TTY

```

3042          ;;          RETURN HERE          ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
3043          ;;          ;;          ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
3044
3045 017034 010346          SRDLIN: MOV      R3, -(SP)          ;; SAVE R3
3046 017036 012703 017142 1$: MOV      #STTYIN, R3          ;; GET ADDRESS
3047 017042 022703 017152 2$: CMP      #STTYIN+8., R3          ;; BUFFER FULL?
3048 017046 101405          BLOS     4$          ;; BR IF YES
3049 017050 104407          RDCHR          ;; GO READ ONE CHARACTER FROM THE TTY
3050 017052 112613          MOVB   (SP)+, (R3)          ;; GET CHARACTER
3051 017054 122713 000177 10$: CMPB   #177, (R3)          ;; IS IT A RUBOUT
3052 017060 001003          BNE     3$          ;; SKIP IF NOT
3053 017062 104400 001170 4$: TYPE   $QUES          ;; TYPE A '?'
3054 017066 000763          BR      1$          ;; CLEAR THE BUFFER AND LOOP
3055 017070 111337 017140 3$: MOVB   (R3), 9$          ;; ECHO THE CHARACTER
3056 017074 104400 017140          TYPE   9$
3057 017100 122723 000015          CMPB   #15, (R3)+          ;; CHECK FOR RETURN
3058 017104 001356          BNE     2$          ;; LOOP IF NOT RETURN
3059 017106 105063 177777          CLRB   -1(R3)          ;; CLEAR RETURN (THE 15)
3060 017112 104400 001172          TYPE   $LF          ;; TYPE A LINE FEED
3061 017116 012603          MOV    (SP)+, R3          ;; RESTORE R3
3062 017120 011646          MOV    (SP), -(SP)          ;; ADJUST THE STACK AND PUT ADDRESS OF THE
3063 017122 016666 000004 000002          MOV    4(SP), 2(SP)          ;; FIRST ASCII CHARACTER ON IT
3064 017130 012766 017142 000004          MOV    #STTYIN, 4(SP)
3065 017136 000002          RTI
3066 017140          000          9$: .BYTE   0          ;; RETURN
3067 017141          000          ;; STORAGE FOR ASCII CHAR. TO TYPE
3068 017142 000010          $TTYIN: .BLKB 8.          ;; TERMINATOR
3069 017152 052536 005015          $CNTLU: .ASCIZ /#U/<15><12>          ;; RESERVE 8 BYTES FOR TTY INPUT
3070 017157          136 006507 000012          $CNTLG: .ASCIZ /#G/<15><12>          ;; CONTROL "U"
3071 017164 005015 053523 020122          $MSWR: .ASCIZ <15><12>/SWR = /          ;; CONTROL "G"
3072 017172 020075          000
3073 017175          040 047040 053505          $MNEW: .ASCIZ / NEW = /
3074 017202 036440 000040
3075          .SBTTL POWER DOWN AND UP ROUTINES
3076
3077          ;; *****
3078          ;; POWER DOWN ROUTINE          ;; *****
3079 017206 012737 017352 000024          $PWRDN: MOV    #SILLUP, @#PWRVEC          ;; SET FOR FAST UP
3080 017214 012737 000340 000026          MOV    #340, @#PWRVEC+2          ;; PRIO:7
3081 017222 010046          MOV    R0, -(SP)          ;; PUSH R0 ON STACK
3082 017224 010146          MOV    R1, -(SP)          ;; PUSH R1 ON STACK
3083 017226 010246          MOV    R2, -(SP)          ;; PUSH R2 ON STACK
3084 017230 010346          MOV    R3, -(SP)          ;; PUSH R3 ON STACK
3085 017232 010446          MOV    R4, -(SP)          ;; PUSH R4 ON STACK
3086 017234 010546          MOV    R5, -(SP)          ;; PUSH R5 ON STACK
3087 017236 017746 161676          MOV    @SWR, -(SP)          ;; PUSH @SWR ON STACK
3088 017242 010637 017356          MOV    SP, $SAVR6          ;; SAVE SP
3089 017246 012737 017260 000024          MOV    #SPWRUP, @#PWRVEC          ;; SET UP VECTOR
3090 017254 000000          HALT
3091 017256 000776          BR      -2          ;; HANG UP
3092
3093          ;; *****
3094          ;; POWER UP ROUTINE          ;; *****
3095 017260 012737 017352 000024          $PWRUP: MOV    #SILLUP, @#PWRVEC          ;; SET FOR FAST DOWN
3096 017266 013706 017356          MOV    $SAVR6, SP          ;; GET SP
3097 017272 005037 017356          CLR    $SAVR6          ;; WAIT LOOP FOR THE TTY

```

```

3098 017276 005237 017356      1$:  INC  $SAVR6      ;;WAIT FOR THE INC
3099 017302 001375              BNE  1$          ;;OF WORD
3100 017304 012677 161630      MOV  (SP)+, @SWR  ;;POP STACK INTO @SWR
3101 017310 012605              MOV  (SP)+, R5    ;;POP STACK INTO R5
3102 017312 012604              MOV  (SP)+, R4    ;;POP STACK INTO R4
3103 017314 012603              MOV  (SP)+, R3    ;;POP STACK INTO R3
3104 017316 012602              MOV  (SP)+, R2    ;;POP STACK INTO R2
3105 017320 012601              MOV  (SP)+, R1    ;;POP STACK INTO R1
3106 017322 012600              MOV  (SP)+, R0    ;;POP STACK INTO R0
3107 017324 012737 017206 000024  MOV  #SPWRDN, @PWRVEC ;;SET UP THE POWER DOWN VECTOR
3108 017332 012737 000340 000026  MOV  #340, @PWRVEC+2 ;;PRIO:7
3109 017340 104400              TYPE  ;;REPORT THE POWER FAILURE
3110 017342 017360      SPWRMG: .WORD  PWRMSG  ;;POWER FAIL MESSAGE POINTER
3111 017344 012716              MOV  (PC)+, (SP)  ;;RESTART AT RESTR
3112 017346 002310      SPWRAD: .WORD  RESTR  ;;RESTART ADDRESS
3113 017350 000002              RTI
3114 017352 000000      $ILLUP: HALT      ;;THE POWER UP SEQUENCE WAS STARTED
3115 017354 000776              BR   -2          ;;BEFORE THE POWER DOWN WAS COMPLETE
3116 017356 000000      $SAVR6: 0        ;;PUT THE SP HERE
3117 017360 005015 042522 052123  PWRMSG: .ASCIZ <15><12>/RESTARTED FROM PWR FAIL/
3118 017366 051101 042524 020104
3119 017374 051106 046517 050040
3120 017402 051127 043040 044501
3121 017410 000114

```

.EVEN
.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.

```

3131 017412 010046      $TRAP: MOV  R0, -(SP)  ;;SAVE R0
3132 017414 016600 000002  MOV  2(SP), R0  ;;GET TRAP ADDRESS
3133 017420 005740      TST  -(R0)      ;;BACKUP BY 2
3134 017422 111000      MOVB (R0), R0   ;;GET RIGHT BYTE OF TRAP
3135 017424 006300      ASL  R0          ;;POSITION FOR INDEXING
3136 017426 016000 017434  MOV  $TRPAD(R0), R0 ;;INDEX TO TABLE
3137 017432 000200      RTS  R0         ;;GO TO ROUTINE

```

.SBTTL TRAP TABLE

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

ROUTINE

```

3146 017434      $TRPAD:
3147 017434 014404      $TYPE  ;;CALL=TYPE      TRAP+0(104400)  TTY TYPEOUT ROUTINE
3148 017436 015160      $TYPOC ;;CALL=TYPOC     TRAP+1(104401)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
3149 017440 015134      $TYPOS ;;CALL=TYPOS     TRAP+2(104402)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
3150 017442 015174      $TYPON ;;CALL=TYPON     TRAP+3(104403)  TYPE OCTAL NUMBER (AS PER LAST CALL)
3151 017444 015362      $TYPDS ;;CALL=TYPDS     TRAP+4(104404)  TYPE DECIMAL NUMBER (WITH SIGN)
3152
3153 017446 016502      $GTSWR ;;CALL=GTSWR     TRAP+5(104405)  GET SOFT-SWR SETTING

```


M05

3154										
3155	017450	016432				SCKSWR	::CALL=CKSWR		TRAP+6(104406)	TEST FOR CHANGE IN SOFT-SWR
3156	017452	016714				SRDCHR	::CALL=RDCHR		TRAP+7(104407)	TTY TYPEIN CHARACTER ROUTINE
3157	017454	017034				SRDLIN	::CALL=RDLIN		TRAP+10(104410)	TTY TYPEIN STRING ROUTINE
3158										
3159		070000								
3160										
3161										
3162										
3163	070000	005015	000			.SBTTL	ASCII MESSAGES			
3164	070003	040	040502	020122		MSG2:	.ASCIZ <15><12>			
3165	070010	047104	000			MSG3:	.ASCIZ / BAR DN/			
3166	070013	040	047516	041440		MSG4:	.ASCIZ / NO CNVRT/			
3167	070020	053116	052122	000						
3168	070025	040	036530	000		MSG5:	.ASCIZ / X=/			
3169	070031	040	036531	000		MSG6:	.ASCIZ / Y=/			
3170										
3171	070035	102	051525	052040		EM1:	.ASCIZ /BUS TIMEOUT ER/			
3172	070042	046511	047505	052125						
3173	070050	042440	000122							
3174	070054	042522	020107	042522		EM2:	.ASCIZ 'REG READ/WRITE ER'			
3175	070062	042101	053457	044522						
3176	070070	042524	042440	000122						
3177	070076	052502	020123	042522		EM3:	.ASCIZ /BUS RESET CLR ER/			
3178	070104	042523	020124	046103						
3179	070112	020122	051105	000						
3180	070117	123	020106	046103		EM4:	.ASCIZ /SF CLR Z ER/			
3181	070124	020122	020132	051105						
3182	070132	000								
3183	070133	123	020106	046103		EM5:	.ASCIZ /SF CLR ALL ER/			
3184	070140	020122	046101	020114						
3185	070146	051105	000							
3186	070151	132	044440	041516		EM6:	.ASCIZ /Z INC ER ON ADC START/			
3187	070156	042440	020122	047117						
3188	070164	040440	041504	051440						
3189	070172	040524	052122	000						
3190	070177	132	047440	043126		EM7:	.ASCIZ /Z OVFL0 FLAG ER/			
3191	070204	047514	043040	040514						
3192	070212	020107	051105	000						
3193	070217	132	047440	043126		EM10:	.ASCIZ /Z OVFL0 INTR ER/			
3194	070224	047514	044440	052116						
3195	070232	020122	051105	000						
3196	070237	111	052116	020122		EM11:	.ASCIZ /INTR AT WRONG BR LEVEL/			
3197	070244	052101	053440	047522						
3198	070252	043516	041040	020122						
3199	070260	042514	042526	000114						
3200	070266	044514	052123	046440		EM12:	.ASCIZ /LIST MD NPR DATA ER/			
3201	070274	020104	050116	020122						
3202	070302	040504	040524	042440						
3203	070310	000122								
3204	070312	044514	052123	046440		EM13:	.ASCIZ /LIST MD CA INC ER/			
3205	070320	020104	040503	044440						
3206	070326	041516	042440	000122						
3207	070334	044514	052123	046440		EM14:	.ASCIZ /LIST MD WC INC ER/			
3208	070342	020104	041527	044440						
3209	070350	041516	042440	000122						

3210	070356	041527	020117	046106	EM15:	.ASCIZ	/WCO FLAG OR INTR ER/
3211	070364	043501	047440	020122			
3212	070372	047111	051124	042440			
3213	070400	000122					
3214	070402	044524	042515	052517	EM16:	.ASCIZ	/TIMEOUT FLAG ER/
3215	070410	020124	046106	043501			
3216	070416	042440	000122				
3217	070422	042101	051522	046440	EM17:	.ASCIZ	/ADRS MAKE ER/
3218	070430	045501	020105	051105			
3219	070436	000					
3220	070437	112	054517	052123	EM20:	.ASCIZ	/JOYSTICK MD FAILED TO OVERRIDE LIST MD/
3221	070444	041511	020113	042115			
3222	070452	043040	044501	042514			
3223	070460	020104	047524	047440			
3224	070466	042526	051122	042111			
3225	070474	020105	044514	052123			
3226	070502	046440	000104				
3227	070506	050116	020122	047111	EM21:	.ASCIZ	/NPR INC ER/
3228	070514	020103	051105	000			
3229	070521	111	041516	047411	EM22:	.ASCIZ	/INC OVFL0 FLAG OR INTR ER/
3230	070526	043126	047514	043040			
3231	070534	040514	020107	051117			
3232	070542	044440	052116	020122			
3233	070550	051105	000				
3234	070553	127	051117	020104	EM23:	.ASCIZ	/WORD INC ER/
3235	070560	047111	020103	051105			
3236	070566	000					
3237	070567	102	052131	020105	EM24:	.ASCIZ	/BYTE INC ER/
3238	070574	047111	020103	051105			
3239	070602	000					
3240	070603	101	041504	047440	EM25:	.ASCIZ	/ADC OR JOYSTICK LOGIC ER/
3241	070610	020122	047512	051531			
3242	070616	044524	045503	046040			
3243	070624	043517	041511	042440			
3244	070632	000122					
3245	070634	044524	044515	043516	EM26:	.ASCIZ	/TIMING MARK FAILED ON ADRS MAKE/
3246	070642	046440	051101	020113			
3247	070650	040506	046111	042105			
3248	070656	047440	020116	042101			
3249	070664	051522	046440	045501			
3250	070672	000105					
3251	070674	047516	041440	053116	EM27:	.ASCIZ	/NO CNVRT FAILED/
3252	070702	052122	043040	044501			
3253	070710	042514	000104				
3254	070714	051105	050122	020103	DH1:	.ASCIZ	/ERRPC TSTNUM BUSADR EXPCT RCVD/
3255	070722	020040	051524	047124			
3256	070730	046525	020040	052502			
3257	070736	040523	051104	020040			
3258	070744	054105	041520	020124			
3259	070752	020040	041522	042126			
3260	070760	000					
3261	070761	105	051122	041520	DH2:	.ASCIZ	/ERRPC TSTNUM XYHOLD GDADR BDADR/
3262	070766	020040	052040	052123			
3263	070774	052516	020115	054040			
3264	071002	044131	046117	020104			
3265	071010	043440	040504	051104			

3266	071016	020040	041040	040504								
3267	071024	051104	000									
3268	071027	105	051122	041520	DH3:	.ASCIZ	/ERRPC	TSTNUM	XYHOLD	ADRS	GDDAT	BDDAT/
3269	071034	020040	052040	052123								
3270	071042	052516	020115	054040								
3271	071050	044131	046117	020104								
3272	071056	040440	051104	020123								
3273	071064	020040	043440	042104								
3274	071072	052101	020040	041040								
3275	071100	042104	052101	000								
3276	071105	105	051122	041520	DH4:	.ASCIZ	/ERRPC	TSTNUM/				
3277	071112	020040	052040	052123								
3278	071120	052516	000115									
3279												
3280												
3281	071124	001116	001600	001122	DT1:	.EVEN						
3282	071132	001124	001126	000000		\$ERRPC, TSTNUM, \$BDADR, \$GDDAT, \$BDDAT, 0						
3283	071140	001116	001600	013226	DT2:	\$ERRPC, TSTNUM, XYHOLD, \$GDADR, \$BDADR, 0						
3284	071146	001120	001122	000000								
3285	071154	001116	001600	013226	DT3:	\$ERRPC, TSTNUM, XYHOLD, \$BDADR, \$GDDAT, \$BDDAT, 0						
3286	071162	001122	001124	001126								
3287	071170	000000										
3288	071172	001116	001600	000000	DT4:	\$ERRPC, TSTNUM, 0						
3289		000001				.END						

ABASE = 164000	491#	588	629	
ACDW1 = 000000	588	631		
ACDW2 = 000000	588			
ACPUOP = 000000	588	603		
ADDW0 = 000000	588			
ADDW1 = 000000	588			
ADDW10 = 000000	588			
ADDW11 = 000000	588			
ADDW12 = 000000	588			
ADDW13 = 000000	588			
ADDW14 = 000000	588			
ADDW15 = 000000	588			
ADDW2 = 000000	588			
ADDW3 = 000000	588			
ADDW4 = 000000	588			
ADDW5 = 000000	588			
ADDW6 = 000000	588			
ADDW7 = 000000	588			
ADDW8 = 000000	588			
ADDW9 = 000000	588			
ADEVCT = 000000	588	594		
ADEVN = 000001	494#	588	630	
RENV = 000000	588	599		
RENVN = 000000	588	600		
AFATAL = 000000	588	591		
AMADR1 = 000000	588	616		
AMADR2 = 000000	588	620		
AMADR3 = 000000	588	623		
AMADR4 = 000000	588	626		
AMAMS1 = 000000	588	610		
AMAMS2 = 000000	588	618		
AMAMS3 = 000000	588	621		
AMAMS4 = 000000	588	624		
AMSGAD = 000000	588	596		
AMSGLG = 000000	588	597		
AMSGTY = 000000	588	590		
AMTYP1 = 000000	588	611		
AMTYP2 = 000000	588	619		
AMTYP3 = 000000	588	622		
AMTYP4 = 000000	588	625		
APASS = 000000	588	595		
APRIOR = 000340	493#	588		
APTCSU = 000040	2515	2515		
APTENV = 000001	2508	2508	2618#	2797
APTSIZ = 000200	857	2508		
APTSPO = 000100	2510	2508	2619#	
ASWREG = 000000	588	611		
ATESTN = 000000	588	592		
AUNIT = 000000	588	595		
AUSWR = 000000	588	602		
AVECT1 = 160270	492#	588	627	
AVECT2 = 000000	588	628		
BIT0 = 000001	476#			
BIT00 = 000001	466#	476		
BIT01 = 000002	465#	475		
BIT02 = 000004	464#	474		

ADD	869	876	977	1342	1372	1490	1509	1537	1571	1605	1639	1673	1707	1735	1757
	1779	1801	2226	2241	2345	2356	2375	2381	2399	2402	2426	2429	2445	2451	2471
ASL	2477	2521	2581	2593	2605	2649	2659	2730	2846	2975	2984				
ASLB	2418	2843	2844	2845	2998	2999	3000	3135							
ASRB	2227	2343	2344	2372	2373	2374	2383	2398	2419	2420	2443	2444	2468	2469	2470
BCC	2480	2589													
BCCS	978	1491	1510	1736	1758	1780	1802	2736							
BFC	1057	1538	1572	1606	1640	1674	1708								
	858	888	902	923	943	959	975	990	1004	1020	1035	1039	1054	1069	1081
	1085	1098	1114	1119	1138	1143	1167	1183	1232	1310	1338	1345	1352	1357	1359
	1376	1394	1424	1434	1454	1470	1488	1507	1529	1534	1563	1568	1597	1602	1631
	1636	1665	1670	1699	1704	1732	1754	1776	1798	1823	1828	1842	1851	1863	1885
	1890	1903	1912	1924	1946	1951	1965	1974	1986	2016	2178	2231	2237	2247	2252
	2260	2291	2293	2295	2297	2299	2301	2303	2349	2351	2379	2449	2475	2511	2524
	2559	2575	2579	2599	2601	2676	2783	2786	2809	2812	2848	2853	2866	2903	2905
	2907	2911	2920	2955	2982	2997									
BGE	2923														
BGT	2169	2683	2744	2994	3035										
BHI	2909														
BIC	862	874	938	941	971	1202	1419	1427	1432	2011	2019	2074	2087	2166	2233
	2248	2315	2347	2353	2358	2377	2384	2401	2404	2423	2428	2447	2454	2473	2673
	2951	2968	2995	3022	3028	3036									
BIS	2067	2080	2204	2238	2352	2355	2380	2425	2450	2476	2678	2679	2738	2739	3002
BISB	2835														
BIT	1159	1166	1182	1206	1358	1364	1375	1393	1829	1841	1850	1891	1902	1911	1952
	1964	1973	2023	2104	2112	2124	2129	2138	2251	2286	2290	2292	2294	2296	2298
	2300	2302	2350	2785	2792	2808	2898	2902	2910	2917					
BITB	857	2510	2515	2547	2578										
BLOS	3048														
BLT	2538	2684	2727	2743	2992	3033									
BMI	2041	2075	2734												
BNE	824	847	871	878	882	886	890	946	1122	1146	1160	1207	1257	1283	1365
	1416	1527	1541	1561	1575	1595	1609	1629	1643	1663	1677	1697	1711	1819	1830
	1881	1892	1942	1953	2008	2024	2070	2083	2092	2102	2105	2113	2125	2130	2139
	2196	2228	2287	2509	2516	2518	2526	2534	2548	2555	2577	2583	2586	2603	2674
	2732	2793	2798	2836	2858	2889	2918	2947	2953	2973	2980	2987	3024	3030	3052
	3058	3099													
BPL	1529	1573	1607	1641	1675	1709	1837	1898	1960	2046	2089	2111	2123	2422	2503
	2552	2672	2718	2748	2805	2949	2965	3020	3026						
BR	816	849	892	895	924	1204	1233	1258	1284	1363	1369	1373	1421	1429	1452
	1468	1486	1505	1525	1531	1536	1542	1544	1559	1565	1570	1576	1578	1593	1599
	1604	1610	1612	1627	1633	1638	1644	1646	1661	1667	1672	1678	1690	1695	1701
	1706	1712	1714	1730	1734	1752	1756	1774	1778	1796	1800	1835	1848	1857	1896
	1909	1918	1958	1971	1980	2013	2021	2073	2079	2086	2118	2127	2132	2152	2254
	2354	2424	2505	2531	2541	2550	2557	2569	2591	2650	2665	2686	2729	2746	2803
	2841	2868	2891	2897	2900	2913	2916	2976	3003	3005	3031	3054	3091	3115	
BVC	1007	1023	1315												
CLR	817	822	835	836	856	914	915	955	969	986	1030	1065	1076	1094	1117
	1141	1161	1164	1180	1208	1211	1238	1260	1264	1289	1304	1331	1360	1367	1377
	1390	1409	1425	1437	1444	1445	1447	1478	1481	1482	1497	1500	1518	1552	1586
	1620	1654	1688	1722	1744	1766	1788	1814	1815	1816	1833	1838	1844	1853	1876
	1877	1878	1894	1899	1905	1914	1937	1938	1939	1956	1961	1967	1976	1997	2001
	2017	2025	2028	2042	2047	2089	2116	2134	2135	2163	2164	2250	2285	2322	2663
	2721	2724	2834	2915	2930	2962	2963	3097							
CLRB	2530	2556	2607	2608	2609	2750	2914	3059							

CMP	823	846	870	877	889	925	942	974	1003	1019	1053	1113	1137	1205	1229
	1231	1234	1255	1259	1291	1285	1309	1337	1344	1351	1412	1413	1422	1433	1453
	1469	1487	1506	1528	1533	1562	1567	1596	1601	1630	1635	1664	1669	1698	1703
	1731	1753	1775	1797	1836	1862	1897	1923	1959	1985	2014	2015	2230	2257	2742
	2898	2922	2946	2952	2972	2979	2991	2993	3023	3029	3032	3034	3047	3051	3057
CMPB	897	2508	2523	2525	2533	2554	2558	2576	2797	2904	2908	2554	2986		
COM	1370	1820													
COMB	1882	1943													
DEC	922	1006	1022	1056	1314	1415	1423	1822	1884	1945	2091	2101	2167	2195	2842
DECB	2537	2540	2671	2682											
EMT	385														
HALT	501	2504	2806	3090	3114										
INC	881	945	1121	1145	1256	1282	1349	1371	1543	1577	1611	1645	1679	1713	1818
INCB	2007	2165	2606	2677	2685	2728	2788	2921	3001	3098					
IOI	1880	1941	2560	2782	2926										
JMP	386														
JSR	505	507	904	2050	2185	2304	2305	2306	2307						
	1449	1450	1451	1465	1466	1467	1483	1484	1485	1502	1503	1504	1522	1523	1524
	1556	1557	1558	1590	1591	1592	1624	1625	1626	1658	1659	1660	1692	1693	1694
	1727	1728	1729	1749	1750	1751	1771	1772	1773	1793	1794	1795	2068	2091	2109
	2180	2244	2338	2341	2367	2370	2393	2396	2413	2416	2438	2441	2463	2466	2479
	2513	2532	2539	2546	2595	2794	2800	2821	2990						
MOV	815	821	825	827	828	829	830	831	832	833	834	838	839	842	843
	844	845	850	852	853	854	859	861	864	866	867	868	872	873	875
	998	899	910	911	913	916	917	919	919	927	933	934	935	936	937
	939	940	952	953	954	956	958	966	967	968	970	972	973	984	985
	987	989	997	998	999	1000	1001	1002	1013	1014	1015	1016	1017	1018	1029
	1031	1032	1034	1036	1038	1040	1047	1048	1049	1050	1051	1052	1063	1064	1066
	1068	1077	1078	1079	1080	1082	1084	1086	1093	1095	1096	1097	1105	1106	1107
	1108	1109	1110	1111	1112	1116	1118	1128	1129	1130	1131	1132	1133	1134	1135
	1136	1140	1142	1153	1154	1155	1156	1157	1158	1165	1168	1175	1176	1177	1178
	1179	1184	1191	1192	1193	1194	1195	1196	1197	1198	1199	1200	1201	1210	1217
	1218	1219	1220	1221	1222	1223	1224	1225	1226	1227	1228	1235	1237	1244	1245
	1246	1247	1248	1249	1250	1251	1252	1253	1254	1261	1263	1270	1271	1272	1273
	1274	1275	1276	1277	1278	1279	1280	1286	1288	1295	1296	1297	1298	1299	1300
	1301	1302	1303	1305	1306	1307	1308	1321	1322	1323	1324	1325	1326	1327	1328
	1329	1330	1332	1333	1334	1335	1336	1341	1343	1346	1348	1350	1353	1355	1361
	1366	1378	1385	1386	1387	1388	1389	1391	1392	1401	1402	1403	1404	1405	1406
	1407	1408	1410	1411	1414	1417	1418	1426	1430	1431	1436	1443	1446	1448	1461
	1462	1463	1464	1479	1480	1498	1499	1501	1516	1517	1519	1520	1521	1550	1551
	1553	1554	1555	1584	1585	1587	1588	1589	1618	1619	1621	1622	1623	1652	1653
	1655	1656	1657	1686	1687	1689	1690	1691	1720	1721	1723	1724	1725	1726	1742
	1743	1745	1746	1747	1748	1764	1765	1767	1768	1769	1770	1786	1787	1789	1790
	1791	1792	1808	1809	1810	1811	1812	1813	1821	1824	1825	1831	1832	1839	1840
	1843	1845	1849	1852	1854	1858	1859	1860	1861	1870	1871	1872	1873	1874	1875
	1883	1886	1887	1893	1900	1901	1904	1906	1910	1913	1915	1919	1920	1921	1922
	1931	1932	1933	1934	1935	1936	1944	1947	1948	1954	1955	1962	1963	1966	1968
	1972	1975	1977	1981	1982	1983	1984	1993	1994	1995	1996	1998	1999	2000	2003
	2004	2005	2006	2009	2010	2018	2022	2027	2034	2035	2036	2037	2038	2039	2044
	2048	2057	2058	2059	2060	2061	2062	2063	2064	2065	2066	2069	2076	2082	2098
	2099	2100	2103	2108	2114	2115	2120	2121	2141	2147	2170	2174	2177	2194	2203
	2205	2206	2229	2232	2234	2243	2245	2249	2253	2255	2256	2258	2261	2262	2289
	2339	2346	2368	2376	2394	2400	2414	2427	2439	2446	2464	2472	2506	2507	2512
	2520	2535	2572	2573	2580	2584	2589	2590	2592	2594	2604	2610	2611	2646	2654
	2655	2656	2662	2669	2687	2688	2689	2690	2691	2711	2712	2713	2714	2715	2716
	2717	2722	2725	2745	2751	2752	2753	2754	2755	2757	2758	2784	2789	2810	2813

	2833	2838	2847	2852	2857	2859	2863	2893	2894	2896	2899	2912	2924	2925	2928
	2929	2932	2933	2959	2983	2988	3017	3018	3045	3046	3061	3062	3063	3064	3079
	3080	3081	3082	3083	3084	3085	3086	3087	3088	3089	3095	3096	3100	3101	3102
MOV8	3103	3104	3105	3106	3107	3108	3111	3131	3132	3136					
	837	893	903	912	2136	2137	2235	2240	2288	2313	2314	2329	2340	2369	2395
	2415	2440	2465	2517	2545	2553	2567	2568	2570	2647	2648	2651	2652	2653	2657
	2660	2661	2680	2720	2723	2737	2740	2749	2791	2799	2820	2927	2931	2950	2967
	3021	3027	3050	3055	3134										
NEG	2658	2719													
NOP	909	1826	1888	1949	2056	2181	2182	2183							
RESET	900	957	988	1033	1067	1181	1374	2179							
ROL	2664	2666	2667	2668	2670										
ROR	2453														
RTI	851	2522	2692	2759	2815	2934	2989	3037	3065	3113					
RTS	2197	2207	2263	2316	2323	2330	2357	2359	2382	2385	2403	2405	2430	2452	2455
	2478	2481	2562	2612	2823	2861	3137								
SUB	865	1230	2587	2726	2790										
SWAB	863	2342	2371	2397	2417	2442	2467								
TRAP	3139	3148	3149	3150	3151	3153	3155	3156	3157						
TST	885	901	920	921	1356	1526	1540	1560	1574	1594	1608	1628	1642	1662	1676
	1696	1710	1827	1889	1950	2040	2045	2122	2236	2246	2259	2348	2378	2421	2448
	2474	2519	2527	2549	2582	2600	2602	2675	2731	2741	2804	2811	2865	2895	2919
	2981	2996	3133												
TST8	2110	2502	2551	2574	2585	2598	2733	2747	2906	2948	2964	3019	3025		
.ASCII	581	582													
.ASCIZ	580	583	897	2188	2869	3069	3070	3071	3073	3117	3163	3164	3166	3168	3169
	3171	3174	3177	3180	3183	3186	3190	3193	3196	3200	3204	3207	3210	3214	3217
	3220	3227	3229	3234	3237	3240	3245	3251	3254	3261	3268	3276			
.BLKB	3068														
.BLKW	2764														
.BYTE	550	551	556	557	565	566	574	575	576	577	599	600	610	611	618
	619	621	622	624	625	2144	2145	2150	2151	2187	2613	2614	2615	2693	2694
	2695	2696	2801	2802	3066	3067									
.DSABL	3006														
.ENABL	356	2939													
.END	3289														
.ENDC	361	377	379	380	381	385	477	491	506	511	515	517	522	524	531
	544	548	550	578	579	580	581	585	588	610	618	621	624	627	628
	629	630	631	634	814	825	826	829	831	833	835	836	838	840	861
	883	889	895	897	907	908	909	910	911	912	930	931	932	933	934
	949	950	951	952	953	960	963	964	965	966	967	981	982	983	984
	985	991	994	995	996	997	998	1010	1011	1012	1013	1014	1026	1027	1028
	1029	1030	1040	1044	1045	1046	1047	1048	1060	1061	1062	1063	1064	1070	1073
	1074	1075	1076	1086	1090	1091	1092	1093	1099	1102	1103	1104	1105	1106	1125
	1126	1127	1128	1129	1149	1150	1151	1152	1168	1172	1173	1174	1175	1176	1184
	1188	1189	1190	1191	1214	1215	1216	1217	1241	1242	1243	1244	1245	1267	1268
	1269	1270	1271	1292	1293	1294	1295	1296	1318	1319	1320	1321	1322	1370	1377
	1382	1383	1384	1385	1395	1398	1399	1400	1401	1440	1441	1442	1443	1453	1458
	1459	1460	1461	1469	1475	1476	1477	1478	1494	1495	1496	1497	1513	1514	1515
	1516	1517	1532	1537	1542	1547	1548	1549	1550	1551	1566	1571	1576	1581	1582
	1583	1584	1585	1600	1605	1610	1615	1616	1617	1618	1619	1634	1639	1644	1649
	1650	1651	1652	1653	1668	1673	1678	1683	1684	1685	1686	1687	1702	1707	1712
	1717	1718	1719	1720	1721	1735	1739	1740	1741	1742	1743	1757	1761	1762	1763
	1764	1765	1779	1783	1784	1785	1786	1787	1801	1805	1806	1807	1808	1809	1824
	1836	1849	1858	1867	1868	1869	1870	1871	1886	1897	1910	1919	1928	1929	1930
	1931	1932	1947	1959	1972	1981	1990	1991	1992	1993	1994	2031	2032	2033	2034

	2054	2055	2056	2057	2058	2074	2080	2087	2095	2096	2097	2098	2145	2146	2151
	2152	2156	2157	2158	2160	2163	2169	2172	2173	2177	2179	2185	2187	2188	2191
	2192	2194	2200	2203	2210	2226	2278	2285	2310	2313	2319	2322	2326	2329	2333
	2338	2362	2367	2388	2393	2408	2413	2433	2438	2458	2463	2488	2517	2567	2568
	2571	2598	2613	2624	2701	2768	2771	2782	2789	2794	2795	2796	2804	2815	2816
	2817	2820	2827	2842	2871	2874	2877	2882	2888	2890	2901	2904	2905	2906	2908
	2910	2917	2921	2926	2928	2932	2935	2936	2939	2940	2942	2970	3006	3010	3038
	3039	3046	3048	3051	3053	3069	3075	3078	3087	3088	3094	3100	3101	3111	3113
	3117	3126	3132	3135	3147	3148	3149	3150	3151	3152	3153	3154	3155	3156	3157
	3158														
.EQUIV	385	386	394	409	410	439	440	441	442	443	444	445	446	447	448
	467	468	469	470	471	472	473	474	475	476					
.EVEN	588	897	2616	2870	3122	3280									
.IF	357	377	378	379	380	381	383	449	477	504	510	513	515	521	523
	530	543	547	549	578	579	580	584	585	587	610	618	621	624	627
	628	629	630	631	632	634	814	820	825	827	829	831	833	835	836
	838	856	882	883	884	887	896	906	908	910	911	929	931	933	934
	948	950	952	953	959	962	964	966	967	980	982	984	985	990	993
	995	997	998	1009	1011	1013	1014	1025	1027	1029	1030	1039	1043	1045	1047
	1048	1059	1061	1063	1064	1069	1072	1074	1076	1085	1089	1091	1093	1098	1101
	1103	1105	1106	1124	1126	1128	1129	1148	1150	1152	1167	1171	1173	1175	1176
	1183	1187	1189	1191	1213	1215	1217	1240	1242	1244	1245	1266	1268	1270	1271
	1291	1293	1295	1296	1317	1319	1321	1322	1369	1376	1381	1383	1385	1394	1397
	1399	1401	1439	1441	1443	1452	1457	1459	1461	1468	1474	1476	1478	1493	1495
	1497	1512	1514	1516	1517	1531	1536	1541	1546	1548	1550	1551	1565	1570	1575
	1580	1582	1584	1585	1599	1604	1609	1614	1616	1618	1619	1633	1638	1643	1648
	1650	1652	1653	1667	1672	1677	1682	1684	1686	1687	1701	1706	1711	1716	1718
	1720	1721	1734	1738	1740	1742	1743	1756	1760	1762	1764	1765	1778	1782	1784
	1786	1787	1800	1804	1806	1808	1809	1823	1835	1848	1857	1866	1868	1870	1871
	1885	1896	1909	1918	1927	1929	1931	1932	1946	1958	1971	1980	1989	1991	1993
	1994	2030	2032	2034	2053	2055	2057	2058	2073	2079	2086	2094	2096	2098	2144
	2145	2150	2151	2155	2156	2157	2158	2159	2160	2162	2168	2171	2173	2177	2179
	2185	2187	2188	2191	2193	2199	2202	2209	2225	2277	2284	2309	2312	2318	2321
	2325	2328	2332	2337	2361	2366	2387	2392	2407	2412	2432	2437	2457	2462	2487
	2508	2566	2568	2571	2598	2613	2623	2700	2767	2770	2781	2785	2792	2794	2795
	2797	2804	2808	2815	2816	2819	2826	2841	2857	2873	2876	2881	2887	2888	2900
	2902	2903	2904	2906	2907	2908	2917	2919	2927	2929	2934	2935	2936	2938	2940
	2941	2942	2970	3009	3010	3038	3046	3047	3051	3052	3068	3069	3075	3077	3087
	3088	3093	3100	3101	3109	3111	3113	3117	3125	3131	3135	3139	3148	3149	3150
	3151	3152	3153	3155	3156	3157	3158								
.IFF	377	379	380	381	383	511	515	517	522	524	531	544	547	550	578
	585	588	825	882	883	906	907	908	909	910	930	931	932	933	949
	950	951	952	960	963	964	965	966	981	982	983	984	991	994	995
	996	997	1010	1011	1012	1013	1026	1027	1028	1029	1040	1044	1045	1046	1047
	1060	1061	1062	1063	1070	1073	1074	1075	1076	1086	1090	1091	1092	1093	1099
	1102	1103	1104	1105	1125	1126	1127	1128	1149	1150	1151	1152	1168	1172	1173
	1174	1175	1184	1188	1189	1190	1191	1214	1215	1216	1217	1241	1242	1243	1244
	1267	1268	1269	1270	1292	1293	1294	1295	1318	1319	1320	1321	1370	1377	1382
	1383	1384	1385	1395	1398	1399	1400	1401	1440	1441	1442	1443	1453	1458	1459
	1460	1461	1469	1475	1476	1477	1478	1494	1495	1496	1497	1513	1514	1515	1516
	1532	1537	1542	1547	1548	1549	1550	1566	1571	1576	1581	1582	1583	1584	1600
	1605	1610	1615	1616	1617	1618	1634	1639	1644	1649	1650	1651	1652	1668	1673
	1678	1683	1684	1685	1686	1702	1707	1712	1717	1718	1719	1720	1735	1739	1740
	1741	1742	1757	1761	1762	1763	1764	1779	1783	1784	1785	1786	1801	1805	1806
	1807	1808	1824	1836	1849	1858	1867	1868	1869	1870	1886	1897	1910	1919	1928
	1929	1930	1931	1947	1959	1972	1981	1990	1991	1992	1993	2031	2032	2033	2034

	2053	2054	2055	2056	2057	2074	2080	2087	2095	2096	2097	2098	2145	2146	2151
	2152	2156	2159	2163	2169	2172	2187	2192	2194	2200	2203	2210	2226	2278	2285
	2310	2313	2319	2322	2326	2329	2333	2338	2362	2367	2388	2393	2408	2413	2433
	2438	2458	2463	2489	2567	2624	2701	2768	2770	2785	2815	2816	2817	2820	2827
	2842	2871	2974	2901	2904	2905	2908	2935	2936	2939	2942	3010	3012	3017	3038
	3039	3048	3052	3069	3078	3094	3109	3126	3132						
.IFT	897	2795	2916	3012	3017										
.IFTF	897	2794	2914	2957	3010	3013									
.IIF	356	361	366	367	374	375	376	377	380	381	501	584	588	826	829
	835	836	838	839	883	2142	2148	2157	2163	2164	2175	2187	2191	2564	2771
	2772	2773	2774	2775	2780	2807	2815	2816	2839	2864	2877	2878	2879	2880	2881
	2882	2886	2915	2916	2932	2935	2936	2939	2960	3061	3069	3075	3147	3148	3149
.IRP	3150	3151	3153	3155	3156	3157									
	814	906	911	929	948	962	980	993	1009	1025	1043	1059	1072	1089	1101
	1124	1148	1171	1187	1213	1240	1266	1291	1317	1381	1397	1439	1457	1474	1493
	1512	1546	1580	1614	1648	1682	1716	1738	1760	1782	1804	1866	1927	1989	2030
.LIST	2053	2094	2572	2573	2594	2610	2611	2711	2751	2781	2887	3081	3087	3100	3101
	356	380	491	501	578	585	588	814	840	883	884	897	906	910	929
	933	948	952	962	966	980	984	993	997	1009	1013	1025	1029	1043	1047
	1059	1063	1072	1076	1089	1093	1101	1105	1124	1128	1148	1152	1171	1175	1187
	1191	1213	1217	1240	1244	1266	1270	1291	1295	1317	1321	1381	1385	1397	1401
	1439	1443	1457	1461	1474	1478	1493	1497	1512	1516	1546	1550	1580	1584	1614
	1618	1648	1652	1682	1686	1716	1720	1738	1742	1760	1764	1782	1785	1804	1808
	1866	1870	1927	1931	1989	1993	2030	2034	2053	2057	2094	2098	2143	2179	2815
.MACRO	2981	3038	3139	3147	3148	3149	3150	3151	3152	3153	3154	3155	3156	3157	3158
.MCALL	381	541	787	856	3139										
.MEXIT	356	491	585	840	884										
.NLIST	356	380	491	501	578	585	588	814	840	883	884	897	906	910	929
	933	948	952	962	966	980	984	993	997	1009	1013	1025	1029	1043	1047
	1059	1063	1072	1076	1089	1093	1101	1105	1124	1128	1148	1152	1171	1175	1187
	1191	1213	1217	1240	1244	1266	1270	1291	1295	1317	1321	1381	1385	1397	1401
	1439	1443	1457	1461	1474	1478	1493	1497	1512	1516	1546	1550	1580	1584	1614
	1618	1648	1652	1682	1686	1716	1720	1738	1742	1760	1764	1782	1786	1804	1808
	1866	1870	1927	1931	1989	1993	2030	2034	2053	2057	2094	2098	2163	2179	2484
	2815	2881	3038	3139	3147	3148	3149	3150	3151	3152	3153	3154	3155	3156	3157
.PAGE	508	541	634	649	697	745	797	814	906	2153	2191	2482			
.REM	1														
.REPT	501														
.SBTTL	370	381	495	504	508	519	541	585	634	814	819	879	884	906	929
	948	962	980	993	1009	1025	1043	1059	1072	1089	1101	1124	1148	1171	1187
	1213	1240	1266	1291	1317	1381	1397	1439	1457	1474	1493	1512	1546	1580	1614
	1648	1682	1716	1738	1760	1782	1804	1866	1927	1989	2030	2053	2094	2153	2482
	2485	2564	2621	2698	2765	2824	2871	2936	3075	3123	3139	3162			
.TITLE	356														
.WORD	501	502	503	516	535	536	537	538	539	540	549	552	553	554	555
	558	559	560	561	562	563	564	567	568	569	590	591	592	593	594
	595	596	597	601	602	603	616	620	623	626	627	628	629	630	631
	2168	2171	2186	2514	2561	2596	2697	2850	2855	3110	3112				

H07

MAINDEC-11-DZNCA-C NC11-A LOGIC TEST MACY11 27(732) 21-SEP-76 14:14 PAGE 88
DZNCA.P11 CROSS REFERENCE TABLE -- PERMANENT SYMBOLS

*.DZNCA.SEG/SOL/CRF=DZNCA.P11
RUN-TIME: 64 40 8 SECONDS
RUN-TIME RATIO: 333/112=2.9
CORE USED: 26K (51 PAGES)

