

**DX11-B**

**DX-11B ADRSNG TST  
CZDXJCO**

**AH-A830C-MC  
FICHE 1 OF 1**

SEP 1982  
COPYRIGHT © 77-82  
MADE IN USA



The table contains multiple columns of data, including alphanumeric strings, numbers, and possibly dates, arranged in a grid format. The text is very faint and difficult to read, but appears to be organized into several columns. The data is presented in a structured, tabular layout typical of a technical or administrative document.



.REM %

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
45

IDENTIFICATION

PRODUCT CODE: AC-A829C-MC  
PRODUCT NAME: CZDXJCO DX11-B OFF-LINE ADDRESSING TEST  
DATE: MARCH 1982  
MAINTAINER: DIAGNOSTIC PROGRAMMING  
AUTHOR: R. MISNER

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

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED TO THE PURCHASER UNDER A LICENSE FOR USE ON A SINGLE COMPUTER SYSTEM AND CAN BE COPIED (WITH INCLUSION OF DIGITAL'S COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT AS MAY OTHERWISE BE PROVIDED IN WRITING BY DIGITAL.

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

COPYRIGHT (C) 1977,1982 BY DIGITAL EQUIPMENT CORPORATION

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 3

47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102

PROGRAM HISTORY  
-----

04-DEC-81 V. GUARINO

INSERTED CODE TO FORCE NO MORE THAN 124KW MEMORY ALLOCATION  
IN THE MEMORY SIZE (SIZE:) SUBROUTINE.

1.0 GENERAL PROGRAM INFORMATION

1.1 ABSTRACT  
-----

THIS PROGRAM IS A SUPPLEMENT TO DZDXA AND DZDXF DX11-B  
MAINTENANCE MODE DIAGNOSTICS AND IS A FUNCTIONAL CHECK THE  
ABILITY OF THE DX TO ACCESS ALL AREAS OF MEMORY USING DIFFERENT  
SOURCES FOR THE NPR DATA ADDRESS. IT IS ASSUMED THAT DZDXA AND  
DZDXF HAVE ALREADY BEEN RUN. THE THREE NPR ADDRESS SOURCES  
RESULT FROM DATA WRITES TO MEMORY, STATUS POINTER WORD (SPW)  
FETCHES, AND DEVICE STATUS TABLE (DST) FETCHES.  
DIAGNOSIS IS BASED ON THE ASSUMPTION THAT THE CPU CAN ACCESS  
MEMORY CORRECTLY WHILE THE DX'S ADDRESSING CAPABILITY  
IS SUSPECT.

ERROR MESSAGES DESCRIBE THE FAILING ADDRESS SOURCE AND THE  
EXPECTED LOCATION OF THE DX ACCESS. THE ACTUAL DX ADDRESS  
OF THE ACCESS CANNOT BE DETERMINED.  
THE OPERATOR INTERFACE IS DESIGNED TO BE COMPATIBLE WITH  
DZDXA AND DZDXF.

1.2 SYSTEM REQUIREMENTS  
-----

THIS PROGRAM REQUIRES ANY PDP11 PROCESSOR WITH AT LEAST  
8K OF MEMORY BUT NOT MORE THAN 128K; THE DX11-B AND A CONSOLE  
TERMINAL. THE PROGRAM WILL RUN UNDER XXDP,ACT,SLIDE, AND  
STAND ALONE. THIS PROGRAM HAS NOT BEEN TESTED ON CPU'S WITHOUT  
A SWITCH REGISTER.

1.3 RELATED DOCUMENTS AND STANDARDS  
-----

PROGRAM IS ASSEMBLED USING SYSMAC MD-11-DZQAC-C3 AND  
CONFORMS TO THE ACT INTERFACE AUTOCAT-11-QZAUB-B-D  
-----

1.4 DIAGNOSTIC HIERARCHY PREREQUISITES  
-----

IT IS ASSUMED THAT DZDXA, DZDXF, AND APPROPRIATE MEMORY  
DIAGNOSTICS ARE RUN ERROR FREE IN ORDER FOR THE ERROR  
MESSAGES IN THIS DIAGNOSTIC TO BE MEANINGFUL.

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 4

103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158

1.5 ASSUMPTIONS  
-----

THIS PROGRAM ASSUMES THAT THE CPU, UNIBUS ADDRESSING, AND PORTIONS OF THE DX11-B TESTED BY DZDXA AND DZDXF ARE OPERATING CORRECTLY IN ORDER FOR THE ERROR MESSAGES TO BE MEANINGFUL.

2.0 OPERATING INSTRUCTIONS  
-----

2.1 LOADING AND STARTING  
-----

USE STANDARD LOADING PROCEDURES FOR PDP11 UNDER THE ABSOLUTE LOADER, XXDP, SLIDE, OR ACT. START AT ADDRESS 200. THE PROGRAM WILL PRINT OUT THE MAINDEC NUMBER AND TITLE AND A PROMPTING MESSAGE AS FOLLOWS:

"TYPE: <D>, FOR DEFAULT PARAMETERS  
<P>, FOR PREVIOUS PARAMETERS  
<S>, FOR SELECT PARAMETERS  
<N>, FOR START WITH THIS TEST NO.

D,P,S,N?"

THE OPERATOR MUST RESPOND WITH A D OR S IF THIS IS THE FIRST START SINCE THE PROGRAM HAS BEEN LOADED INTO MEMORY. IF, IN RESPONSE TO THIS MESSAGE, THE OPERATOR TYPES AN S<CR> THE FOLLOWING DIAGLOG COULD TAKE PLACE:

"TEST NUMBER: 1<CR>  
BASE ADDRESS: 176200<CR>  
VECTOR ADDRESS: 300<CR>  
DX PRIORITY LEVEL: 4<CR>"

NOTE: THE ABOVE RESPONSES ARE THE DEFAULT PARAMETERS AND ARE EQUIVALENT TO TYPING A D<CR> IN RESPONSE TO D,P,S,N? A <CR> RESPONSE FOR A PARTICULAR ENTRY WILL SELECT THE DEFAULT VALUE. A CONTROL C AT ANY TIME WILL TAKE THE PROGRAM BACK TO THE PARAMETER ENTRY SECTION. RESPONSE TO THE CONTROL C MAY BE SLOW IF THE PROGRAM IS IN A PARTICULARLY LONG TEST. ALL PARAMETERS ARE CHECKED TO SEE IF THEY ARE IN A LOGICAL RANGE. IF NOT, THE ENTRY IS REQUESTED AGAIN.

"TEST NUMBER:"

THE PROGRAM IS ASKING FOR THE STARTING TEST NUMBER. 1 THRU 4 ARE VALID RESPONSES. THIS IS THE STARTING TEST NUMBER. TO LOOP ON A PARTICULAR TEST TYPE THE TEST NUMBER IN RESPONSE TO THIS MESSAGE AND SET SWITCH 14 ON THE CONSOLE.

'BASE ADDRESS:"

THIS IS A REQUEST FOR THE BASE ADDRESS OF THE DX IN THE

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 5

159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214

UNIBUS ADDRESS SPACE. IT IS ALSO THE ADDRESS OF THE DXDS. ALL OTHER DX CONTROL AND STATUS REGISTER ADDRESSES ARE GENERATED FROM THIS ENTRY.

'VECTOR ADDRESS:''

THIS IS A REQUEST FOR THE DX11-B INTERRUPT VECTOR ADDRESS.

'DX PRIORITY LEVEL''

THIS IS A REQUEST FOR THE DX'S INTERRUPT PRIORITY LEVEL. IT IS NOT USED BY THIS PARTICULAR PROGRAM.

'SET SWITCHES:''

THIS IS A REMINDER TO SET THE CONSOLE SWITCHES BEFORE STARTING THE PROGRAM. AFTER SETTING SWITCHES, RESPOND WITH A <CR>. CONSOLE SWITCHES HAVE THE FOLLOWING MEANING:

|        |                               |
|--------|-------------------------------|
| SW<15> | HALT ON ERROR                 |
| SW<14> | LOOP ON ERROR OR TEST         |
| SW<13> | INHIBIT ERROR PRINTOUT        |
| SW<12> | PRINT ONLY SHORT ERROR REPORT |
| SW<11> | INHIBIT ITERATIONS            |
| SW<10> | INHIBIT MEMORY MGT USE        |

TESTING WILL COMMENCE AS SOON AS A <CR> IS GIVEN IN RESPONSE TO THE "SET SWITCHES:'' MESSAGE.

THIS PROGRAM AUTO SIZES THE MEMORY. THE CONTROL UNIT ADDRESS AND DEVICES PER CONTROL UNIT ARE ALSO AUTO SIZED. THIS PARTICULAR PROGRAM DOES NOT NEED THE "LEGAL COMMANDS" FACILITY USED BY OTHER DX DIAGNOSTICS.

## 2.2 SPECIAL ENVIRONMENTS

-----

THIS PROGRAM WILL RUN IN UNATTENDED MODE UNDER ACT.

## 2.3 PROGRAM OPTIONS

-----

SEE SECTION 2.1 FOR PARAMETER SELECTION AND OPTIONS. NOTE THAT ALL CONSOLE SWITCH CHANGES ARE EFFECTIVE IMMEDIATELY EXCEPT SW<10> "INHIBIT MEMORY MANAGEMENT USES". IN ORDER TO CHANGE THE MEANING OF SW<10> SET THE SWITCH THEN RESTART THE PROGRAM VIA CONTROL C OR START AT ADDRESS 200.

## 2.4 EXECUTION TIMES

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 6

215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270

-----

SINGLE PASS EXECUTION TIME IS VARIABLE DEPENDING ON MEMORY SIZE. A PDP-11/45 WITH 65K WORDS OF MEMORY REQUIRES 50 SECONDS. EACH ROUTINE HAS A SINGLE ITERATION REGARDLESS OF MODE. END PASS IS SIGNALLED AFTER ALL TESTS HAVE BEEN RUN ONCE.

### 3.0 ERROR INFORMATION

-----

THE GENERAL ERROR REPORT FORMAT IS AS FOLLOWS:

LINE 1

-----

ERROR PC: XXXXXX ERROR NO. YYYYY

THIS GIVES THE PROGRAM COUNTER (XXXXXX) WHERE THE ERROR WAS DETECTED AND A UNIQUE ERROR NUMBER OR IDENTIFIER (YYYYY)

LINE 2

-----

THIS LINE GIVES A UNIQUE ERROR MESSAGE FOR EACH ERROR TYPE EXPLAINING THE SUSPECTED CAUSE OF THE FAILURE. THIS IS THE LAST LINE OF THE MESSAGE IF SWITCH 12 IS ON.

LINE 3

-----

TEST NO. XXXXXX

THIS IDENTIFIES THE TEST NUMBER (XXXXXX = 1-4) WHERE THE FAILURE WAS DETECTED.

LINE 4

-----

VIRT ADDR OF BUFFER BASE: XXXXXX

THIS GIVES THE VIRTUAL ADDRESS OF THE LOWEST LOCATION IN THE BUFFER WHICH THE DX WAS ACCESSING AT THE TIME OF ERROR.

LINE 5

-----

PHY ADDR OF BUFFER: XXXXXX

THIS GIVES THE PHYSICAL ADDRESS OF THE BUFFER BASE. ONLY OUTPUT IF MEMORY MANAGEMENT IS IN USE

LINE 6

-----

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 7

271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306

EA BITS IN OCT: XXXXXX

THE LOW TO BITS OF XXXXXX ARE THE EXTENDED ADDRESS BITS  
USED IN ACCESSING THE FAILING MEMORY LOCATION. ONLY OUTPUT IF  
MEMORY MANAGEMENT IN USE.

LINE 7

-----

VIRT ADDR OF ACCESS ERROR: XXXXXX

THIS MESSAGE IS ONLY OUTPUT FOR TEST 2. IT GIVES THE  
EXPECTED WORD LOCATION OF THE FAILING ACCESS.

LINE 7-8 ALTERNATE

ACTUAL: XXXXX EXPECTED: XXXXXX

IN SOME TESTS, THIS GIVES THE ACTUAL AND EXPECTED BYTE  
OF DATA. THIS WILL GIVE INSIGHT INTO FAILING ADDRESSES AS THERE  
IS A CORRESPONDENCE BETWEEN THIS DATA AND THE LOCATION OF THE  
FAILURE.

NOTE: AT THE END OF THE FIRST PASS EXECUTION THE FOLLOWING  
MESSAGE IS PRINTED:

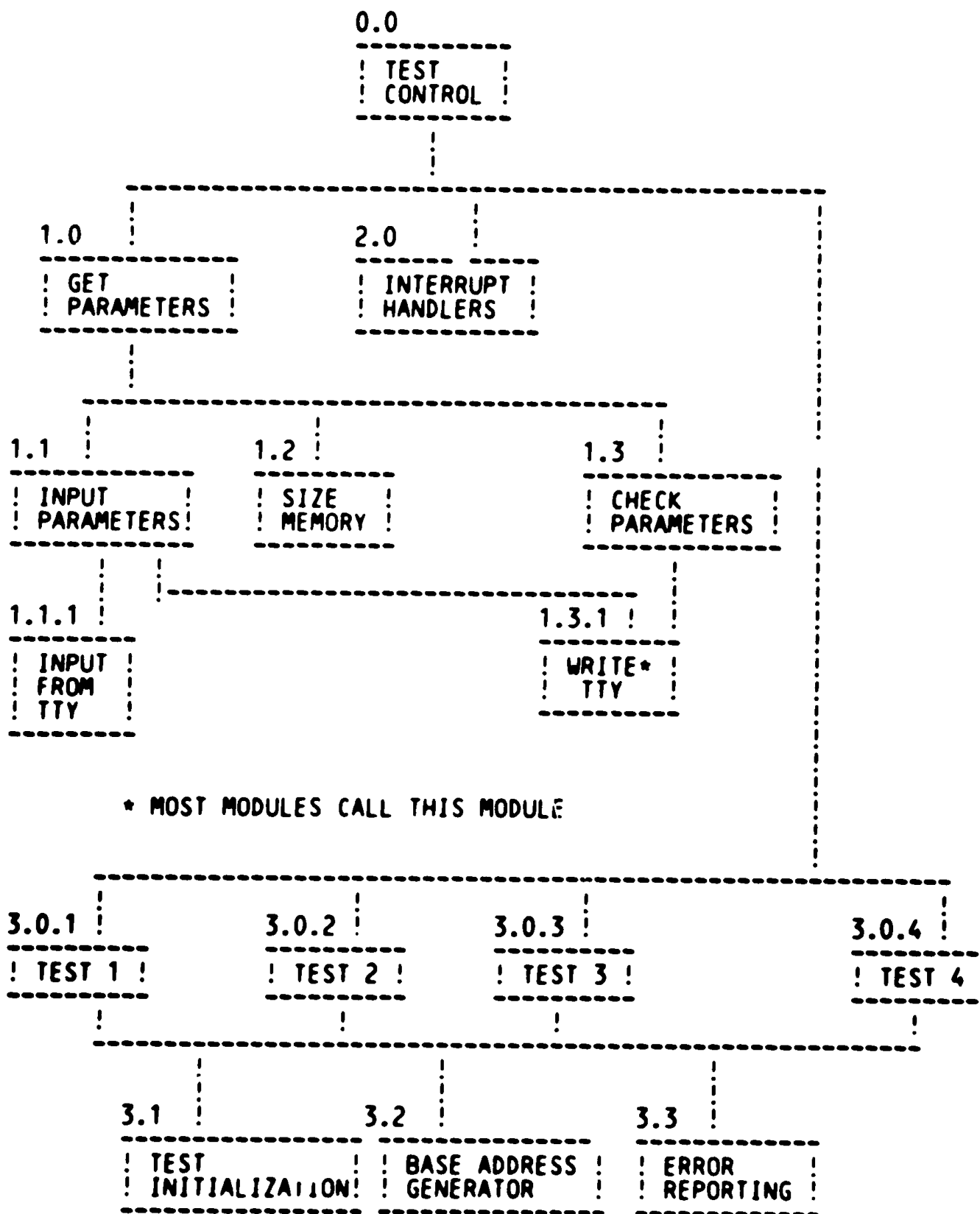
1ST IBM DEV RCGNZD. OCTAL: XXXXXX  
SIZE OF 1ST DEV GROUP: YYYYYY

THIS IS NOT AN ERROR MESSAGE BUT IS AN INDICATION OF  
HOW THE JUMPERS ARE SET UP ON MODULE A20. THIS INFOR-  
MATION IS PROVIDED ONLY AS A CONVENIENCE FOR THE  
OPERATOR. SEE MAINTENANCE MANUAL EK-DX11B-MM-002  
PARA. 2.4.2 FOR MORE INFORMATION.

4.0 PROGRAM ORGANIZATION

307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361

4.1 BLOCK DIAGRAM





MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 9

362  
363  
364  
365  
366  
367  
368

4.2 TEST DESCRIPTIONS

4.2.1 TST 1

369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424

MODULE 3.0.1 FAST NPR DATA TEST

THIS IS A FUNCTIONAL TEST TO CHECK THE ABILITY OF THE DX TO DO NPR'S OF DATA INTO MEMORY WITH LARGE BYTE COUNTS. THE DXBA IS CHECKED FOR ITS ABILITY TO COUNT AND ALL CARRIES IN THE COUNTER ARE EXERCISED. THE DATA TRANSFERS ACCESS ALL AVAILABLE MEMORY EXCEPT WHERE THE PROGRAM IS LOCATED, AND THE AREA WHICH MAY BE OCCUPIED BY A LOADER.

4.2.2 TST 2

MODULE 3.0.2 NPR DATA ADDR UNIQUENESS TEST

THIS MODULE TESTS THE DX11'S ABILITY TO UNIQUELY ADDRESS EACH WORD OF MEMORY. THIS TEST IS ACCOMPLISHED BY WRITING A BACKGROUND IN A BLOCK OF MEMORY. A WORD IS THEN TRANSFERRED FROM THE DX TO THE MEMORY BLOCK. THE CPU CHECKS THAT THE WORD ARRIVES AT ITS EXPECTED DESTINATION. IF YES THE WORD IS RESTORED TO MATCH THE BACKGROUND, AND THE NEXT WORD IS TESTED. THIS IS REPEATED FOR EACH WORD IN THE BLOCK AND FOR EACH BLOCK IN MEMORY.

NOTE: THIS TEST MAY NOT DETECT ADDRESSING PROBLEMS WHICH ARE COMMON TO THE CPU AND DX11. I.E. UNIBUS ADDRESSING PROBLEMS.

4.2.3 TST 3

MODULE 3.0.3 SPW ADDRESSING TEST

THIS MODULE TESTS THE ABILITY OF THE DX TO ACCESS THE STATUS POINTER WORD (SPW) TABLE IN ALL POSSIBLE LOCATIONS IN MEMORY. A BASE ADDRESS FOR THE SPW IS GENERATED.

THIS SPW TABLE IS FILLED WITH A PATTERN OF 00DEV WHERE DEV IS EQUAL TO THE DISPLACEMENT OF THE WORD FROM THE SPW BASE; I.E.  
SPW BASE =000000  
SPW BASE +1 =000001

..  
..  
..

SPW BASE+255=000377

GIVING A TOTAL OF 256 WORDS FOR THE SPW TABLE WITH IMMEDIATE STATUS EQUAL TO THE DEVICE NUMBER. AN ISS SEQUENCE IS INITIATED WITH A DEVICE # OF 0. IF ADRECC AND ADRECD DON'T COME ON FOR THIS DEV, NO TESTING IS DONE. THE DEV # IS INCREMENTED, AND ISS IS DONE AGAIN. IF ADRECC AND ADRECD COME ON IT MEANS THE DX JUMPERS ARE CUT TO RESPOND TO THIS DEV NUMBER. THE ISS SEQUENCE IS THEN CONTINUED UNTIL THE STATUS IS IN THE DXOS REG. THIS STATUS IS THEN CHECKED AND SHOULD BE EQUAL TO THE DEV # IF THE SPW WAS ACCESSED CORRECTLY.

THIS PROCEDURE IS REPEATED FOR ALL 256 POSSIBLE DEVICES AT ALL POSSIBLE LOCATIONS FOR THE SPW.

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 3GA(1052) 07-JUN-82 16:29 PAGE 11

425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480

THE FIRST RECOGNIZED DEV NO. AND COUNT OF THE FIRST GROUP OF RECOGNIZED DEVICES IS SAVED AND PRINTED AT END PASS TIME. IF NO DEV IS RECOGNIZED AND ERROR MESSAGE IS OUTPUT AT THE END OF THE TEST.

#### 4.2.4 TST 4

##### MODULE 3.0.4 DEVICE STATUS TABLE (DST) ACCESS TEST

THIS TEST CHECKS THE ABILITY TO ACCESS ALL BYTES OF THE DST IN ALL AREAS OF MEMORY. THE OBJECTIVE IS TO CHECK LOADING OF THE DXBA FROM THE SPW DST BASE, AND COMMAND (CUCR)

A VALID IBM DEVICE FROM TST3 IS CHECKED. THEN A BASE ADDR IS GENERATED FOR THE DST. EACH BYTE OF THE DST IS THEN LOADED WITH ITS OFFSET FROM THE DST BASE.

A SPW ENTRY FOR THE VALID IBM DEVICE IS GENERATED WITH APPROPRIATE DST BASE AND ZERO IMMEDIATE STATUS TO ASSURE A DST ACCESS. AN ISS SEQUENCE IS DONE WITH THE VALID DEV AND EVERY POSSIBLE IBM COMMAND THEREBY ACCESSING EVERY LOCATION IN THE DST. AFTER THE ISS, THE STATUS WILL EQUAL THE COMMAND OR AND ERROR IS FLAGGED.

```
%
.NLIST CND,MC,TOC
.LIST ME
.ENABL ABS,AMA
```

```
.TITLE MD-11-CZDXJ-C DX11-B ADDRESSING TEST
```

```
.*COPYRIGHT (C) -1977-
```

```
.*DIGITAL EQUIPMENT CORP.
```

```
.*MAYNARD, MASS. 01754
```

```
.*
.*PROGRAM BY R. MISNER
```

```
.*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
```

```
.*PACKAGE (MAINDEC-11-DZQAC-C5), JAN, 1981.
```

```
.*
$TN=1
```

```
$$SWR=160000 ;;HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYP0UT
```

```
.SBTTL BASIC DEFINITIONS
```

```
.*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
```

```
STACK= 1100
```

```
.EQUIV EMT,ERROR ;;BASIC DEFINITION OF ERROR CALL
```

```
.EQUIV IOT,SCOPE ;;BASIC DEFINITION OF SCOPE CALL
```

```
.*MISCELLANEOUS DEFINITIONS
```

```
HT= 11 ;;CODE FOR HORIZONTAL TAB
```

```
LF= 12 ;;CODE FOR LINE FEED
```

```
CR= 15 ;;CODE FOR CARRIAGE RETURN
```

```
CRLF= 200 ;;CODE FOR CARRIAGE RETURN-LINE FEED
```

```
PS= 177776 ;;PROCESSOR STATUS WORD
```

```
.EQUIV PS,PSW
```

```
STKLMT= 177774 ;;STACK LIMIT REGISTER
```

```
PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
```

000001  
160000

001100

000011  
000012  
000015  
000200  
177777  
177774  
177772



MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 12  
BASIC DEFINITIONS

```

481      177570      DSWR= 177570      ;;HARDWARE SWITCH REGISTER
482      177570      DDISP= 177570     ;;HARDWARE DISPLAY REGISTER
483
484      ;*GENERAL PURPOSE REGISTER DEFINITIONS
485      000000      R0=    %0          ;;GENERAL REGISTER
486      000001      R1=    %1          ;;GENERAL REGISTER
487      000002      R2=    %2          ;;GENERAL REGISTER
488      000003      R3=    %3          ;;GENERAL REGISTER
489      000004      R4=    %4          ;;GENERAL REGISTER
490      000005      R5=    %5          ;;GENERAL REGISTER
491      000006      R6=    %6          ;;GENERAL REGISTER
492      000007      R7=    %7          ;;GENERAL REGISTER
493      000006      SP=    %6          ;;STACK POINTER
494      000007      PC=    %7          ;;PROGRAM COUNTER
495
496      ;*PRIORITY LEVEL DEFINITIONS
497      000000      PR0=    0          ;;PRIORITY LEVEL 0
498      000040      PR1=    40         ;;PRIORITY LEVEL 1
499      000100      PR2=   100         ;;PRIORITY LEVEL 2
500      000140      PR3=   140         ;;PRIORITY LEVEL 3
501      000200      PR4=   200         ;;PRIORITY LEVEL 4
502      000240      PR5=   240         ;;PRIORITY LEVEL 5
503      000300      PR6=   300         ;;PRIORITY LEVEL 6
504      000340      PR7=   340         ;;PRIORITY LEVEL 7
505
506      ;*'SWITCH REGISTER' SWITCH DEFINITIONS
507      100000      SW15= 100000
508      040000      SW14= 40000
509      020000      SW13= 20000
510      010000      SW12= 10000
511      004000      SW11= 4000
512      002000      SW10= 2000
513      001000      SW09= 1000
514      000400      SW08= 400
515      000200      SW07= 200
516      000100      SW06= 100
517      000040      SW05= 40
518      000020      SW04= 20
519      000010      SW03= 10
520      000004      SW02= 4
521      000002      SW01= 2
522      000001      SW00= 1
523      .EQUIV      SW09,SW9
524      .EQUIV      SW08,SW8
525      .EQUIV      SW07,SW7
526      .EQUIV      SW06,SW6
527      .EQUIV      SW05,SW5
528      .EQUIV      SW04,SW4
529      .EQUIV      SW03,SW3
530      .EQUIV      SW02,SW2
531      .EQUIV      SW01,SW1
532      .EQUIV      SW00,SW0
533
534      ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
535      100000      BIT15= 100000
536      040000      BIT14= 40000

```

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 13  
BASIC DEFINITIONS

|     |        |  |
|-----|--------|--|
| 537 | 020000 | BIT13= 20000                                   |
| 538 | 010000 | BIT12= 10000                                   |
| 539 | 004000 | BIT11= 4000                                    |
| 540 | 002000 | BIT10= 2000                                    |
| 541 | 001000 | BIT09= 1000                                    |
| 542 | 000400 | BIT08= 400                                     |
| 543 | 000200 | BIT07= 200                                     |
| 544 | 000100 | BIT06= 100                                     |
| 545 | 000040 | BIT05= 40                                      |
| 546 | 000020 | BIT04= 20                                      |
| 547 | 000010 | BIT03= 10                                      |
| 548 | 000004 | BIT02= 4                                       |
| 549 | 000002 | BIT01= 2                                       |
| 550 | 000001 | BIT00= 1                                       |
| 551 |        | .EQUIV BIT09,BIT9                              |
| 552 |        | .EQUIV BIT08,BIT8                              |
| 553 |        | .EQUIV BIT07,BIT7                              |
| 554 |        | .EQUIV BIT06,BIT6                              |
| 555 |        | .EQUIV BIT05,BIT5                              |
| 556 |        | .EQUIV BIT04,BIT4                              |
| 557 |        | .EQUIV BIT03,BIT3                              |
| 558 |        | .EQUIV BIT02,BIT2                              |
| 559 |        | .EQUIV BIT01,BIT1                              |
| 560 |        | .EQUIV BIT00,BIT0                              |
| 561 |        |  |
| 562 |        | ;*BASIC "CPU" TRAP VECTOR ADDRESSES            |
| 563 | 000004 | ERRVEC= 4 ::TIME OUT AND OTHER ERRORS          |
| 564 | 000010 | RESVEC= 10 ::RESERVED AND ILLEGAL INSTRUCTIONS |
| 565 | 000014 | TBITVEC=14 ::'T' BIT                           |
| 566 | 000014 | TRIVEC= 14 ::TRACE TRAP                        |
| 567 | 000014 | BPTVEC= 14 ::BREAKPOINT TRAP (BPT)             |
| 568 | 000020 | IOTVEC= 20 ::INPUT/OUTPUT TRAP (IOT) **SCOPE** |
| 569 | 000024 | PWRVEC= 24 ::POWER FAIL                        |
| 570 | 000030 | EMTVEC= 30 ::EMULATOR TRAP (EMT) **ERROR**     |
| 571 | 000034 | TRAPVEC=34 ::'TRAP' TRAP                       |
| 572 | 000060 | TKVEC= 60 ::TTY KEYBOARD VECTOR                |
| 573 | 000064 | TPVEC= 64 ::TTY PRINTER VECTOR                 |
| 574 | 000240 | PIRQVEC=240 ::PROGRAM INTERRUPT REQUEST VECTOR |
| 575 |        | .SBTTL MEMORY MANAGEMENT DEFINITIONS           |
| 576 |        |  |
| 577 |        | ;*KT11 VECTOR ADDRESS                          |
| 578 |        |  |
| 579 | 000250 | MMVEC= 250                                     |
| 580 |        |  |
| 581 |        | ;*KT11 STATUS REGISTER ADDRESSES               |
| 582 |        |  |
| 583 | 177572 | SR0= 177572                                    |
| 584 | 177574 | SR1= 177574                                    |
| 585 | 177576 | SR2= 177576                                    |
| 586 | 172516 | SR3= 172516                                    |
| 587 |        |  |
| 588 |        | ;*KERNEL "I" PAGE DESCRIPTOR REGISTERS         |
| 589 |        |  |
| 590 | 172300 | KIPDR0= 172300                                 |
| 591 | 172302 | KIPDR1= 172302                                 |
| 592 | 172304 | KIPDR2= 172304                                 |

MD-11-C7DXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 14  
MEMORY MANAGEMENT DEFINITIONS

593 172306  
594 172310  
595 172312  
596 172314  
597 172316  
598  
599

KIPDR3= 172306  
KIPDR4= 172310  
KIPDR5= 172312  
KIPDR6= 172314  
KIPDR7= 172316

;\*KERNEL 'I' PAGE ADDRESS REGISTERS

601 172340  
602 172342  
603 172344  
604 172346  
605 172350  
606 172352  
607 172354  
608 172356  
609

KIPAR0= 172340  
KIPAR1= 172342  
KIPAR2= 172344  
KIPAR3= 172346  
KIPAR4= 172350  
KIPAR5= 172352  
KIPAR6= 172354  
KIPAR7= 172356

610 000200  
611 000004  
612 020000  
613 040000  
614 002000  
615 001000  
616 004000  
617  
618

DONE=200  
SOSIEN=4  
SELO=20000  
HLDO=40000  
CMDO=2000  
SRVO=1000  
ADRO=4000  
.SBTTL TRAP CATCHER

619 000000

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

623 000174  
624 000174 000000  
625 000176 000000  
626

.=174  
DISPREG: .WORD 0 ;;SOFTWARE DISPLAY REGISTER  
SWREG: .WORD 0 ;;SOFTWARE SWITCH REGISTER  
.SBTTL STARTING ADDRESS(ES)  
JMP @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM  
.SBTTL ACT11 HOOKS

627 000200 000137 001164  
628  
629

\*\*\*\*\*

631  
632 000204  
633 000046  
634 000046 001566  
635 000052  
636 000052 040000  
637 000204  
638 001000

HOOKS REQUIRED BY ACT11  
\$SVPC= . ;SAVE PC  
. =46  
\$ENDAD ;;1)SET LOC.46 TO ADDRESS OF \$ENDAD IN .SEOP  
. =52  
.WORD 40000 ;;2)SET LOC.52 TO 40000  
.\$SVPC ;; RESTORE PC  
. =1000



MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 15  
COMMON TAGS

.SBTTL COMMON TAGS

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

639  
640  
641  
642  
643  
644  
645 001100  
646 001100 001100  
647 001100 000000  
648 001102 000  
649 001103 000  
650 001104 000000  
651 001106 000000  
652 001110 000000  
653 001112 000000  
654 001114 000  
655 001115 001  
656 001116 000000  
657 001120 000000  
658 001122 000000  
659 001124 000000  
660 001126 000000  
661 001130 000000  
662 001132 000000  
663 001134 000  
664 001135 000  
665 001136 000000  
666 001140 177570  
667 001142 177570  
668 001144 177560  
669 001146 177562  
670 001150 177564  
671 001152 177566  
672 001154 000  
673 001155 002  
674 001156 012  
675 001157 000  
676 001160 077  
677 001161 015  
678 001162 000012  
679

.=1100

\$CMTAG: .WORD 0  
\$PASS: .WORD 0  
\$STNM: .BYTE 0  
\$ERFLG: .BYTE 0  
\$ICNT: .WORD 0  
\$LPADR: .WORD 0  
\$LPERR: .WORD 0  
\$ERTTL: .WORD 0  
\$ITEMB: .BYTE 0  
\$ERMAX: .BYTE 1  
\$ERRPC: .WORD 0  
\$GDADR: .WORD 0  
\$BDADR: .WORD 0  
\$GDDAT: .WORD 0  
\$BDDAT: .WORD 0  
\$AUTOB: .BYTE 0  
\$INTAG: .BYTE 0  
\$SWR: .WORD DSWR  
\$DISPLAY: .WORD DDISP  
\$TKS: 177560  
\$TKB: 177562  
\$TPS: 177564  
\$TPB: 177566  
\$NULL: .BYTE 0  
\$FILLS: .BYTE 2  
\$FILLC: .BYTE 12  
\$TPFLG: .BYTE 0  
\$QUES: .ASCII /?/  
\$CRLF: .ASCII <15>  
\$LF: .ASCII <12>

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

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 16  
ERROR POINTER TABLE

680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694 001164

.SBTTL ERROR POINTER TABLE  
;\*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.  
;\*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN  
;\*LOCATION \$ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.  
;\*NOTE1: IF \$ITEMB IS 0 THE ONLY PERTINENT DATA IS (\$ERRPC).  
;\*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:  
  
:\* EM ::POINTS TO THE ERROR MESSAGE  
:\* DH ::POINTS TO THE DATA HEADER  
:\* DT ::POINTS TO THE DATA  
:\* DF ::POINTS TO THE DATA FORMAT

\$ERRTB:

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 17  
MODULE 0.0 TEST CONTROL

```

695          .SBTTL MODULE 0.0      TEST CONTROL
696          MODULE 0.0      TEST CONTROL
697          :
698          :
699          :
700          :
701          :
702 001164   START:
703          .SBTTL INITIALIZE THE COMMON TAGS
704          ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
705 001164   012706 001100      MOV    #SCMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
706 001170   005026             CLR    (R6)+           ;;CLEAR MEMORY LOCATION
707 001172   022706 001140      CMP    #SWR,R6      ;;DONE?
708 001176   001374             BNE    -6             ;;LOOP BACK IF NO
709 001200   012706 001100      MOV    #STACK,SP     ;;SETUP THE STACK POINTER
710          ;;INITIALIZE A FEW VECTORS
711 001204   012737 006174 000034  MOV    #STRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
712 001212   012737 000340 000036  MOV    #340,@#TRAPVEC+2;LEVEL 7
713 001220   012737 006272 000024  MOV    #SPWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
714 001226   012737 000340 000026  MOV    #340,@#PWRVEC+2 ;;LEVEL 7
715          ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
716          ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
717 001234   013746 000004             MOV    @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
718 001240   012737 001274 000004  MOV    #64$,@#ERRVEC  ;;SET UP ERROR VECTOR
719 001246   012737 177570 001140  MOV    #DSWR,SWR      ;;SETUP FOR A HARDWARE SWICH REGISTER
720 001254   012737 177570 001142  MOV    #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
721 001262   022777 177777 177650  CMP    #-1,@SWR      ;;TRY TO REFERENCE HARDWARE SWR
722 001270   001012             BNE    66$           ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
723          ;;AND THE HARDWARE SWR IS NOT = -1
724 001272   000403             BR     65$           ;;BRANCH IF NO TIMEOUT
725 001274   012716 001302      64$:  MOV    #65$,(SP)     ;;SET UP FOR TRAP RETURN
726 001300   000002             RTI
727 001302   012737 000176 001140  65$:  MOV    #SWREG,SWR     ;;POINT TO SOFTWARE SWR
728 001310   012737 000174 001142  MOV    #DISPREG,DISPLAY
729 001316   012637 000004      66$:  MOV    (SP)+,@#ERRVEC ;;RESTORE ERROR VECTOR
730
731 001322   000005             RESET
732 001324   005037 012174             CLR    FIRSTP       ;;CLEAR 1ST PASS SW
733 001330   005037 012274             CLR    DEVMS        ;;SET UP TO PRINT DEVICE ADDR CONFIG FIRST TIME ONLY
734 001334   004737 001600             RESTART:JSR PC,GETPRM ;;TALK TO THE OPERATOR
735 001340   013737 012212 012214  MOV    COUNT,CNT     ;;SET UP ITERATION COUNT
736 001346   013737 012154 012206  GO:   MOV    ATESTN,CTESTN ;;GET FIRST TEST NO.
737 001354   005037 012220             CLR    LOCK         ;;CLEAR LOOP LOCK SW
738 001360   005037 012216             CLR    ERR          ;;RESET ERROR INDICATOR
739 001364   013702 012206      TST:  MOV    CTESTN,R2
740 001370   006372             ASL    R2           ;;MULTIPLY BY TWO FOR WORD BNDRY
741 001372   000172 001376      JMP    @TSTAB(R2)    ;;GO TO TEST THRU DISPATCH TABLE
742 001376   000000             TSTAB: HALT
743 001400   006454             TST1
744 001402   007176             TST2
745 001404   007444             TST3
746 001406   010156             TST4
747          ;;WHEN TEST IS DONE IT ALWAYS RETURNS HERE AFTER ONE ITERATION
748 001410      ENDTST:
749 001410   005046             CLR    -(SP)        ;;PUT NEW PS ON STACK
750 001412   012746 001420      MOV    #64$,-(SP)   ;;PUT NEW PC ON STACK

```



MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 18  
INITIALIZE THE COMMON TAGS

```

751 001416 000002          RTI          ;;POP NEW PC AND PS
752 001420          64$:
753
754 001420 032777 040000 177512      BIT      #40000,@SWR      ;IS LOOP SWITCH ON
755 001426 001356          BNE      TST             ;BR IF YES
756 001430 005237 012206          INC      CTESTN         ;INCREMENT TO NEXT TEST
757 001434 023737 012206 012136      CMP      CTESTN,HTESTN  ;HAVE ALL TEST BEEN RUN
758 001442 003750          BLE      TST             ;BR IF NO, DISPATCH TO NEXT ONE
759 001444 005337 012214          DEC      CNT            ;DECREMENT ITERATION COUNT
760 001450 001336          BNE      GO             ;START OVER IF NOT EXPIRED
761 001452 005737 012274          EOP:    TST      DEVMSS   ;HAVE WE PRINTED DEVICE CONFIG YET?
762 001456 001026          BNE      10$           ;BR IF YES
763 001460 005737 012272          TST      DEVCT         ;WERE DEVICES RECOGNIZED?
764 001464 001423          BEQ      10$           ;BR IF NO, NOTHING TO SAY
765 001466 004537 005372          JSR      R5,PRINT      ;PRINT DEV NO. ASCII
766 001472 013506          .WORD   DEVMS
767 001474 000000          .WORD   0
768 001476 004537 005372          JSR      R5,PRINT      ;PRINT DEV NO. OCTAL
769 001502 012270          .WORD   FRSTDV
770 001504 100001          .WORD   100001
771 001506 004537 005372          JSR      R5,PRINT      ;PRINT DEVICE COUNT ASCII
772 001512 013544          .WORD   DEVCMS
773 001514 000000          .WORD   0
774 001516 004537 005372          JSR      R5,PRINT      ;PRINT DEVICE COUNT OCT
775 001522 012272          .WORD   DEVCT
776 001524 100001          .WORD   100001
777 001526 012737 000001 012274      MOV      #1,DEVMSS     ;SET DEV CONFIG PRINTED SWITCH
778 001534 005737 012172          10$:    TST      KT             ;IS KT IN USE?
779 001540 001402          BEQ      15$           ;BR IF NO
780 001542 005037 177572          CLR      @#SRO         ;TURN OFF MEM MGT
781 001546 004537 005372          15$:    JSR      R5,PRINT
782 001552 013236          .WORD   BELL
783 001554 000000          .WORD   0
784 001556 013700 000042          MOV      @#42,R0       ;PRINT CONTROL PARAMETER
785 001562 001405          BEQ      END1          ;ARE WE UNDER ACT
786 001564 000005          RESET
787 001566 004710          $ENDAD: JSR      PC,@R0     ;GO TO MONITOR
788 001570 000240          NOP
789 001572 000240          NOP
790 001574 000240          NOP
791 001576 000656          END1:   BR       RESTART ;START OVER
792

```

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 19  
MODULE 1.0 GET PARAMETERS

```

793 .SBTTL MODULE 1.0 GET PARAMETERS
794 MODULE 1.0 GET PARAMETERS
795
796 CALLED AS FOLLOWS
797
798 JSR PC,GETPRM
799
800 THIS MODULE ASSURES THAT ALL PARAMETERS ARE ACQUIRED FOR
801 OPERATION OF THE DIAGNOSTIC TO DO THIS IT ESTABLISHES
802 A DIALOG WITH THE OPERATOR THROUGH THE INPUT PARAM MODULE.
803 MEMORY IS SIZED BY THE 'SIZE' MODULE, AND ALL PARMS ARE
804 CHECKED FOR VALIDITY BY THE CHECKP MODULE.
805
806 ^C AT ANY TIME CLEARS THE 1ST PASS SWITCH AND EVERYTHING
807 STARTS OVER. CR IN RESPONSE TO ANY QUESTION CAUSES THE DEFAULT
808 VALUE TO BE SUBSTITUTED.
809
810 THIS MODULE NEVER RETURNS TO CALLER UNLESS ALL PARAMETERS
811 HAVE BEEN ESTABLISHED WITH GOOD VALUES.
812
813
814 GETPRM: TST FIRSTP ;IS THIS THE FIRST PASS
815 BEQ 5$ ;BR IF YES
816 JMP ENDPRM ;DON'T GET PARM IF WE ALREADY HAVE THEM
817 5$: TST @42 ;ARE WE RUNNING IN AUTO MODE
818 BEQ 10$ ;BRANCH IF MACHINE IS ATTENDED
819 MOV DTESTN,ATESTN ;SETUP DEFAULTS
820 MOV DUBA,AUBA ;SETUP DEFAULTS
821 MOV DVECT,AVECT ;SETUP DEFAULTS
822 MOV DPRIOR,APRIOR ;SETUP DEFAULTS
823 BR 90$ ;GO SEE ABOUT MEMORY SIZING
824 10$: CMP #SENDAD,@#42 ;UNDER ACT11?
825 BEQ 90$ ;NO OPERATOR INTERACTION
826 TST PARMV ;ARE PARAMETERS GOOD
827 BNE 15$ ;BRANCH IF YES BECAUSE THIS
828 ;ALSO MEANS HEADER ISN'T NEEDED
829 JSR R5,PRINT ;CALL THE PRINTER
830 HEADER ;ADDRESS OF HEADER ASCIZ
831 .WORD 0 ;CONTROL WORD, CRLF + DON'T CONVERT
832 15$: JSR R5,INPRM ;GET D,P,S, OR N FROM OPER.
833 .WORD 1 ;THIS TELLS INPRM TO GET D,P,S,N
834 ;INPRM WILL RETURN IN R0 THE CHAR TYPED
835 CMPB #'D,R0 ;WAS DEFAULT PARMS REQUESTED?
836 BNE 20$ ;BR IF NO
837 MOV DTESTN,ATESTN ;DEFAULT SETUP
838 MOV DUBA,AUBA ;DEFAULT SETUP
839 MOV DVECT,AVECT ;DEFAULT SETUP
840 MOV DPRIOR,APRIOR ;DEFAULT SETUP
841 BR 90$ ;GO SEE ABOUT MEMORY SIZING
842 20$: CMPB #'P,R0 ;DOES OPERATOR WANT PREVIOUS
843 BNE 30$ ;BR IF NO
844 TST PARMV ;ARE PARMS SET UP
845 BEQ 25$ ;BR IF PARAMETERS NOT VALID
846 BR 90$ ;USE PREVIOUS PARMS, NO ACTION NEEDED
847 25$: JSR R5,PRINT ;TROUBLE, P TYPED AND PARMS
848 .WORD INVM ;NEVER SET UP PROPERLY

```

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 20  
MODULE 1.0 GET PARAMETERS

```

849 001772 000000      .WORD 0      ;CONTROL WORD,CRLF + DONT CONVERT
850 001774 000726      BR 10$      ;GO TO TRY AGAIN
851 001776 122700 000123 30$: CMPB #'S,RO ;ARE PARMS TO BE SELECTED?
852 002002 001062      BNE 40$     ;BR IF NO
853 002004 005037 012210 35$: CLR PARMV ;CLEAR PARM VALID SW
854 002010 004537 002272 JSR R5,INPRM ;ASK INPUT MODULE TO GET TEST NO:
855 002014 000002      .WORD 2      ;TEST NUMBER WILL BE RETURNED IN RO
856      ;IN OCTAL
857 002016 010037 012154 MOV R0,ATESTN ;SAVE TEST NUMBER IN ACTIVE PARM TABLE
858 002022 004537 005224 JSR R5,CHECKP ;CHECK IF TEST NO OK
859 002026 000002      .WORD 2      ;PARAMETER INDICATES VERIFY TEST #
860 002030 005700      TST R0      ;RO=0 IF CHECK OK
861 002032 001364      BNE 35$     ;STAY HERE UNTIL OPERATOR GETS IT RIGHT
862 002034 004537 002272 36$: JSR R5,INPRM ;ASK INPUT MODULE TO GET BASE ADDR
863 002040 000003      .WORD 3      ;PARAMETER INDICATING GET UBA BASE
864 002042 010037 012156 MOV R0,AUBA   ;SAVE UB BASE ADDR IN ACTIVE TABLE
865 002046 004537 005224 JSR R5,CHECKP ;CHECK FOR VALID ENTRY
866 002052 000003      .WORD 3      ;PARM TO INDICATE CHECK UBA
867 002054 005700      TST R0      ;RO=0 IF CHECK OK
868 002056 001366      BNE 36$     ;STAY HERE UNTIL HE GETS IT RIGHT
869 002060 004537 002272 37$: JSR R5,INPRM ;ASK INPUT MODULE TO GET VECTOR ADDR
870 002064 000004      .WORD 4      ;PARM INDICATING VECTOR NEEDED
871 002066 010037 012160 MOV R0,AVECT ;SAVE VECTOR IN ACTIVE PARM TABLE
872 002072 004537 005224 JSR R5,CHECKP ;CHECK FOR VALID VECTOR
873 002076 000004      .WORD 4      ;PARM INDICATING CHECK VECTOR
874 002100 005700      TST R0      ;RO=0 IF VECTOR OK
875 002102 001366      BNE 37$     ;STAY HERE UNTIL HE GETS IT RIGHT
876 002104 004537 002272 38$: JSR R5,INPRM ;GO GET DX PRIORITY
877 002110 000005      .WORD 5
878 002112 010037 012162 MOV R0,APRIOR ;SAVE PRIORITY
879 002116 004537 005224 JSR R5,CHECKP ;CHECK PRIORITY ENTRY
880 002122 000005      .WORD 5      ;PARM INDICATING PRIORITY CHECK
881 002124 005700      TST R0
882 002126 001366      BNE 38$     ;STAY HERE UNTIL HE GETS IT RIGHT
883 002130 004537 005372 JSR R5,PRINT ;PRINT SET SWITCHES MESSAGE
884 002134 013220      .WORD SETSW ;ADDR OF MESSAGE
885 002136 000000      .WORD 0
886 002140 004537 002532 JSR R5,INASC ;DUMMY INPUT REQUEST TO WAIT FOR
887 002144 000001      .WORD 1      ;OPERATOR TO SET SWITCHES
888 002146 000425      BR 90$     ;GO SEE ABOUT MEMORY SIZING
889 002150 122700 000116 40$: CMPB #'N,RO ;WAS AN 'N' INPUT
890 002154 001236      BNE 10$    ;BR IF NONE OF DPSN WERE INPUT
891 002156 004537 002272 45$: JSR R5,INPRM ;CALL FOR TEST NUMBER TO BE INPUT
892 002162 000002      .WORD 2      ;PARM SAYS GET TEST #.
893 002164 010037 012154 MOV R0,ATESTN ;SAVE STARTING TEST #
894 002170 004537 005224 JSR R5,CHECKP ;HAVE TEST # CHECKED FOR PROPER
895 002174 000002      .WORD 2      ;RANGE
896 002176 005700      TST R0      ;IS TEST # OK?
897 002200 001366      BNE 45$     ;STAY HERE UNTIL HE GETS IT RIGHT
898 002202 005737 012210 TST PARMV    ;WE HAVE STARTING TEST #, ARE OTHER PARMS OK?
899 002206 001005      BNE 90$     ;BR IF OK
900 002210 004537 005372 JSR R5,PRINT ;PRINT
901 002214 013062      INVPRM     ;POINTER TO 'NEED MORE PARMS'' MSG
902 002216 000000      .WORD 0
903 002220 000614      BR 10$     ;CONTROL WORD, CRLF + DON'T CONVERT
904      ;GO BACK AND DO IT ALL AGAIN

```

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 21  
MODULE 1.0 GET PARAMETERS

|     |        |        |        |        |  |
|-----|--------|--------|--------|--------|--|
| 905 |        |        |        |        |  |
| 906 | 002222 | 004737 | 004224 |        |  |
| 907 | 002226 | 012700 | 000015 |        |  |
| 908 | 002232 | 012701 | 012054 |        |  |
| 909 | 002236 | 013702 | 012156 |        |  |
| 910 | 002242 | 010221 |        |        |  |
| 911 | 002244 | 062702 | 000002 |        |  |
| 912 | 002250 | 005300 |        |        |  |
| 913 | 002252 | 001373 |        |        |  |
| 914 | 002254 | 012737 | 000001 | 012210 |  |
| 915 | 002262 | 012737 | 000001 | 012174 |  |
| 916 | 002270 | 000207 |        |        |  |

```

90$: JSR PC,SIZE ;CALL MEMORY SIZING MODULE
      MOV #15,R0 ;START OF LOOP TO SET UP DX ACCESS ADDRESSES
      MOV #DXDSA,R1 ;GET ADDR OF START OF TABLE
      MOV AUBA,R2 ;GET DX REG BASE ADDRESS
95$: MOV R2,(R1)+ ;MOVE UBA TO TABLE
      ADD #2,R2 ;STEP TO NEXT UBA
      DEC R0 ;DECREMENT COUNTER
      BNE 95$ ;FILL THE 13 LOCATIONS OF THE TABLE
      MOV #1,PARMV ;SET PARAMETER VALID SWITCH
      MOV #1,FIRSTP ;SET NOT FIRST PASS SWITCH
ENDPRM: RTS PC ;RETURN TO CALLER, END OF MODULE

```

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 22  
MODULE 1.0 GET PARAMETERS

917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941 002272 012501  
942  
943 002274 120127 000001  
944 002300 001016  
945 002302 004537 005372  
946 002306 013102  
947 002310 000000  
948 002312 004537 002532  
949 002316 000001  
950 002320 111100  
951  
952 002322 120027 000015  
953 002326 001100  
954 002330 113700 000120  
955 002334 000475  
956  
957 002336 120127 000002  
958 002342 001014  
959 002344 004537 005372  
960 002350 013114  
961 002352 000000  
962 002354 004537 002532  
963 002360 000002  
964 002362 011100  
965 002364 001061  
966 002366 013700 012124  
967 002372 000456  
968 002374 120127 000003  
969 002400 001014  
970 002402 004537 005372  
971 002406 013137  
972 002410 000000

```

.SBTTL MODULE 1.1 INPUT PARAMETERS
MODULE 1.1 INPUT PARAMETERS
CALLED AS FOLLOWS
JSR R5,INPRM
.WORD X

WHERE X DEFINES WHAT PARAMETER TO GET AS
FOLLOWS:
X=1 D,P,S,ORN PARAMETER SELECTION
X=2 STARTING TEST #
X=3 UBA BASE ADDR
X=4 DX VECTOR ADDR
X=5 DX PRIORITY

THE REQUESTED PARAMETER IS RETURNED IN R0 ALL CONVERTED
TO OCTAL AND DEFAULT SUBSTITUTED WHEN INDICATED. INPUTS
ARE CHECKED FOR EVERYTHING EXCEPT THAT THE NUMBERS ARE
IN THE CORRECT RANGE. CHECKING INCLUDES NUMERIC IF
REQUIRED, OCTAL IF REQUIRED, <CR> FOR DEFAULT, ETC.
THIS MODULE DESTROYS R1,R0
INPRM: MOV (R5)+,R1 ;GET THE PARAMETER FROM THE
;CALLING CODE.
CMPB R1,#1 ;IS PARM 1 - GET D,P,S, ORN?
BNE 10$ ;BR IF NO.
JSR R5,PRINT ;PRINT
.WORD PROMPT ;D,P,S,N?
.WORD 0 ;CONTROL WORD, CRLF + DON'T CONVERT
JSR R5,INASC ;READ THE INPUT -
.WORD 1 ;INASC RETURNS POINTER TO
MOV (R1),R0 ;INPUT IN R1
;1ST CHAR TO R0
CMPB R0,#CR ;WAS IT DEFAULTED?
BNE 90$ ;NO, ALL DONE
MOV 'P,R0 ;INSERT DEFAULT OF P=PREVIOUS PARMS.
BR 90$ ;ALL DONE

10$: CMPB R1,#2 ;IS PARM 2 - GET TEST #
BNE 20$ ;BR IF NO
JSR R5,PRINT ;PRINT
.WORD TESTN ;ADDR OF STARTING TESTN MSG.
.WORD 0 ;CONTROL WORD, CRLF + DON'T CONVERT
JSR R5,INASC ;GET OCTAL INPUT
.WORD 2 ;PARM TO GET OCTAL INPUT
MOV (R1),R0 ;GET INPUT #
BNE 90$ ;IF NON ZERO, WE'RE DONE
MOV DTESTN,R0 ;SUBSTITUTE THE DEFAULT
BR 90$ ;DONE
20$: CMPB R1,#3 ;WAS UBA REQUESTED
BNE 30$ ;BR IF NO
JSR R5,PRINT ;PRINT
.WORD BASEA ;ASK FOR BASE ADDR
.WORD 0 ;CONTROL WORD, CRLF + DON'T CONVERT

```



MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 23  
MODULE 1.1 INPUT PARAMETERS

|      |        |        |        |       |        |              |                                     |
|------|--------|--------|--------|-------|--------|--------------|-------------------------------------|
| 973  | 002412 | 004537 | 002532 |       | JSR    | R5,INASC     | :GET THE OCTAL                      |
| 974  | 002416 | 000002 |        |       | .WORD  | 2            | :INPUT                              |
| 975  | 002420 | 011100 |        |       | MOV    | (R1),R0      | :GET THE NUMBER INPUT               |
| 976  | 002422 | 001042 |        |       | BNE    | 90\$         | :BR IF NOT DEFAULTED                |
| 977  | 002424 | 013700 | 012126 |       | MOV    | DUBA,R0      | :SUBSTITUTE THE DEFAULT             |
| 978  | 002430 | 000437 |        |       | BR     | 90\$         | :DONE                               |
| 979  | 002432 | 120127 | 000004 | 30\$: | CMPB   | R1,#4        | :DOES CALLER WANT VECTOR?           |
| 980  | 002436 | 001014 |        |       | BNE    | 40\$         | :BR IF NO                           |
| 981  | 002440 | 004537 | 005372 |       | JSR    | R5,PRINT     | :PRINT                              |
| 982  | 002444 | 013155 |        |       | .WORD  | VECTA        | :ASK FOR VECTOR ADDR                |
| 983  | 002446 | 000000 |        |       | .WORD  | 0            | :CONTROL WORD, CRLF + DON'T CONVERT |
| 984  | 002450 | 004537 | 002532 |       | JSR    | R5,INASC     | :GET THE INPUT                      |
| 985  | 002454 | 000002 |        |       | .WORD  | 2            | :IN OCTAL                           |
| 986  | 002456 | 011100 |        |       | MOV    | (R1),R0      | :GET THE NUMBER INPUT               |
| 987  | 002460 | 001023 |        |       | BNE    | 90\$         | :BR IF NOT DEFAULTED                |
| 988  | 002462 | 013700 | 012130 |       | MOV    | DVECT,R0     | :SUBSTITUTE THE DEFAULT             |
| 989  | 002466 | 000420 |        |       | BR     | 90\$         | :DONE                               |
| 990  | 002470 | 120127 | 000005 | 40\$: | CMPB   | R1,#5        | :DOES CALLER WANT PRIORITY?         |
| 991  | 002474 | 001014 |        |       | BNE    | 80\$         | :BR IF INVALID CALL                 |
| 992  | 002476 | 004537 | 005372 |       | JSR    | R5,PRINT     | :PRINT                              |
| 993  | 002502 | 013175 |        |       | .WORD  | PRILV        | :ASK FOR PRIORITY LEVEL             |
| 994  | 002504 | 000000 |        |       | .WORD  | 0            | :CONTROL WORD, CRLF + DON'T CONVERT |
| 995  | 002506 | 004537 | 002532 |       | JSR    | R5,INASC     | :REQUEST INPUT                      |
| 996  | 002512 | 000002 |        |       | .WORD  | 2            | :IN OCTAL                           |
| 997  | 002514 | 011100 |        |       | MOV    | (R1),R0      | :GET THE INPUT                      |
| 998  | 002516 | 001004 |        |       | BNE    | 90\$         | :BR IF OK, NOT DEFAULTED            |
| 999  | 002520 | 013700 | 012132 |       | MOV    | DPRIOR,R0    | :SUBSTITUTE THE DEFAULT             |
| 1000 | 002524 | 000401 |        |       | BR     | 90\$         |                                     |
| 1001 |        |        |        |       |        |              |                                     |
| 1002 | 002526 | 000000 |        | 80\$: | HALT   |              | :HALT ON INVALID CALL TO THIS MOD   |
| 1003 | 002530 | 000205 |        | 90\$: | RTS    | R5           | :RETURN TO CALLER                   |
| 1004 |        |        |        |       | .SBTTL | MODULE 1.1.1 | INPUT FROM TTY                      |
| 1005 |        |        |        |       |        |              |                                     |

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 24  
MODULE 1.1.1 INPUT FROM TTY

|      |        |        |               |
|------|--------|--------|---------------|
| 1006 |        |        |               |
| 1007 |        |        |               |
| 1008 |        |        |               |
| 1009 |        |        |               |
| 1010 |        |        |               |
| 1011 |        |        |               |
| 1012 |        |        |               |
| 1013 |        |        |               |
| 1014 |        |        |               |
| 1015 |        |        |               |
| 1016 |        |        |               |
| 1017 |        |        |               |
| 1018 |        |        |               |
| 1019 |        |        |               |
| 1020 | 002532 | 004737 | 002620        |
| 1021 | 002536 | 012500 |               |
| 1022 | 002540 | 120027 | 000001        |
| 1023 | 002544 | 001004 |               |
| 1024 | 002546 | 104410 |               |
| 1025 | 002550 | 013637 | 013632        |
| 1026 | 002554 | 000406 |               |
| 1027 |        |        |               |
| 1028 | 002556 | 120027 | 000002        |
| 1029 | 002562 | 001006 |               |
| 1030 | 002564 | 104411 |               |
| 1031 | 002566 | 012637 | 013632        |
| 1032 | 002572 | 012701 | 013632        |
| 1033 | 002576 | 000205 |               |
| 1034 | 002600 | 000000 |               |
| 1035 |        |        |               |
| 1036 |        |        |               |
| 1037 |        |        |               |
| 1038 |        |        |               |
| 1039 |        |        |               |
| 1040 |        |        |               |
| 1041 | 002602 | 000000 |               |
| 1042 | 002604 | 000000 |               |
| 1043 | 002606 | 000000 |               |
| 1044 | 002610 | 000010 |               |
| 1045 |        | 002620 |               |
| 1046 |        |        |               |
| 1047 |        |        |               |
| 1048 |        |        |               |
| 1049 |        |        |               |
| 1050 |        |        |               |
| 1051 |        |        |               |
| 1052 |        |        |               |
| 1053 |        |        |               |
| 1054 |        |        |               |
| 1055 | 002620 | 005037 | 002602        |
| 1056 | 002624 | 012737 | 002610 002604 |
| 1057 | 002632 | 013737 | 002604 002606 |
| 1058 | 002640 | 012737 | 002670 000060 |
| 1059 | 002646 | 012737 | 000200 000062 |
| 1060 | 002654 | 005777 | 176266        |
| 1061 | 002660 | 012777 | 000100 176256 |

```

MODULE 1.1.1 INPUT FROM TTY
CALLED AS FOLLOWS
JSR R5,INASC
.WORD X
WHERE X=1 FOR ASCII INPUT
      X=2 FOR OCTAL INPUT
A POINTER TO THE INPUT MESSAGE IS RETURNED IN R1. THIS
ROUTINE DESTROYS R0,R1
INASC: JSR PC,$TKINT ;INITIALIZE THE TTY INPUT
        MOV (R5)+,R0 ;GET THE PASSED PARAMETER
        CMPB R0,#1 ;IS IT A REQUEST FOR ASCII?
        BNE 10$ ;BR IF NO.
        RDLIN ;TRAP TO $SRZAD MACRO
        MOV @(SP)+,INBUF ;GET THE INPUT DATA
        BR 20$ ;GO GET RESULT
10$: CMPB R0,#2 ;IS IT A REQUEST FOR OCTAL
     BNE 30$ ;BR IF NO
     RDOCT
     MOV (SP)+,INBUF ;POP STACK INTO INBUF
20$: MOV #INBUF,R1 ;SET POINTER TO INPUT
     RTS R5 ;RETURN TO CALLER
30$: HALT ;INVALID CALL TO MODULE
;
;SBTTL TTY INPUT ROUTINE
;*****
;ENABL LSB
$TKCNT: .WORD 0 ;:NUMBER OF ITEMS IN QUEUE
$TKQIN: .WORD 0 ;:INPUT POINTER
$TKQOUT: .WORD 0 ;:OUTPUT POINTER
$TKQSRT: .BLKB 10 ;:TTY KEYBOARD QUEUE
$TKQEND=.
;
;*TK INITIALIZE ROUTINE
;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
;
;*CALL:
;* JSR PC,$TKINT
;* RETURN
$TKINT: CLR $TKCNT ;:CLEAR COUNT OF ITEMS IN QUEUE
        MOV $TKQSRT,$TKQIN ;:MOVE THE STARTING ADDRESS OF THE
        MOV $TKQIN,$TKQOUT ;:QUEUE INTO THE INPUT & OUTPUT POINTERS.
        MOV $TKSRV,@TKVEC ;:INITIALIZE THE KEYBOARD VECTOR
        MOV #200,@TKVEC+2 ;:BR" LEVEL 4
        TST @TKKB ;:CLEAR DONE FLAG
        MOV #100,@TKKS ;:ENABLE TTY KEYBOARD INTERRUPT

```

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 25  
TTY INPUT ROUTINE

```

1062 002666 000207          RTS      PC          ;;RETURN TO CALLER
1063
1064
1065      ;*TK SERVICE ROUTINE
1066      ;*THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
1067      ;*BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
1068      ;*IT IN THE QUEUE.
1069      ;*IF THE CHARACTER IS A "CONTROL-C" (^C) $TKINT IS CALLED AND
1070      ;*UPON RETURN EXIT IS MADE TO THE "CONTROL-C" RESTART ADDRESS (START)
1071 002670 117746 176252  $TKSRV: MOVB   @STKB,-(SP)      ;;PICKUP THE CHARACTER
1072 002674 042716 177600      BIC     #^C177,(SP)      ;;STRIP THE JUNK
1073 002700 021627 000021      CMP     (SP),#$XON      ;;IS IT A RANDOM XON?
1074 002704 001002          BNE     30$             ;;BRANCH IF NO
1075 002706 005726          TST     (SP)+          ;;CLEAN RANDOM XON OFF STACK
1076 002710 000002          RTI     ;              ;;RETURN
1077 002712          30$:
1078 002712 021627 000003      CMP     (SP),#3        ;;IS IT A CONTROL C?
1079 002716 001007          BNE     1$             ;;BRANCH IF NO
1080 002720 104401 004022      TYPE   ,%CNTLC        ;;TYPE A CONTROL-C (^C)
1081 002724 004737 002620      JSR    PC,$TKINT      ;;INIT THE KEYBOARD
1082 002730 005726          TST     (SP)+          ;;CLEAN UP STACK
1083 002732 000137 001164      JMP    START          ;;CONTROL C RESTART
1084 002736 021627 000007      1$:  CMP     (SP),#7        ;;IS IT A CONTROL G?
1085 002742 001004          BNE     2$             ;;BRANCH IF NO
1086 002744 022737 000176 001140  CMP     #SWREG,SWR     ;;IS SOFT-SWR SELECTED?
1087 002752 001500          BEQ    6$             ;;GO TO SWR CHANGE
1088
1089 002754          2$:
1090 002754 022737 000010 002602  CMP     #10,$TKCNT    ;;IS THE QUEUE FULL?
1091 002762 001004          BNE     3$             ;;BRANCH IF NO
1092 002764 104401 004016      TYPE   ,%BELL        ;;RING THE TTY BELL
1093 002770 005726          TST     (SP)+          ;;CLEAN CHARACTER OFF OF STACK
1094 002772 000451          BR     5$             ;;EXIT
1095 002774 021627 000023      3$:  CMP     (SP),#23      ;;IS IT A CONTROL-S?
1096 003000 001021          BNE     32$           ;;BRANCH IF NO
1097 003002 005077 176136      CLR    @STKS          ;;DISABLE TTY KEYBOARD INTERRUPTS
1098 003006 005726          TST     (SP)+          ;;CLEAN CHAR OFF STACK
1099 003010 105777 176130      31$:  TSTB   @STKS          ;;WAIT FOR A CHAR
1100 003014 100375          BPL    31$           ;;LOOP UNTIL ITS THERE
1101 003016 117746 176124      MOVB   @STKB,-(SP)    ;;GET THE CHARACTER
1102 003022 042716 177600      BIC    #^C177,(SP)   ;;MAKE IT 7-BIT ASCII
1103 003026 022627 000021      CMP    (SP)+,#21     ;;IS IT A CONTROL-Q?
1104 003032 001366          BNE    31$           ;;BRANCH IF NO
1105 003034 012777 000100 176102  MOV    #100,@STKS    ;;REENABLE TTY KEYBOARD INTERRUPTS
1106 003042 000002          RTI     ;              ;;RETURN
1107 003044 005237 002602      32$:  INC    $TKCNT        ;;COUNT THIS CHARACTER
1108 003050 021627 000140      CMP    (SP),#140     ;;IS IT UPPER CASE?
1109 003054 002405          BLT    4$             ;;BRANCH IF YES
1110 003056 021627 000175      CMP    (SP),#175     ;;IS IT A SPECIAL CHAR?
1111 003062 003002          BGT    4$             ;;BRANCH IF YES
1112 003064 042716 000040      BIC    #40,(SP)      ;;MAKE IT UPPER CASE
1113 003070 112677 177510      4$:  MOVB   (SP)+,@STKQIN ;;AND PUT IT IN QUEUE
1114 003074 005237 002604      INC    $TKQIN        ;;UPDATE THE POINTER
1115 003100 023727 002604 002620  CMP    $TKQIN,$TKQEND ;;GO OFF THE END?
1116 003106 001003          BNE    5$             ;;BRANCH IF NO
1117 003110 012737 002610 002604  MOV    #STKQRT,$TKQIN ;;RESET THE POINTER

```

:RAN001  
:RAN001  
:RAN001  
:RAN001  
:RAN001

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 26  
TTY INPUT ROUTINE

```

1118 003116 000002 5$: RTI ;;RETURN
1119
1120
1121 ::*****
1122 ::*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
1123 ::*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
1124 ::*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP
1125 ::*CALL WHEN OPERATING IN TTY INTERRUPT MODE.
1126 003120 022737 000176 001140 $CKSWR: CMP #SWREG,SWR ;;IS THE SOFT-SWR SELECTED
1127 003126 001124 BNE 15$ ;;EXIT IF NOT
1128 003130 105777 176010 TSTB @STKS ;;IS A CHAR WAITING?
1129 003134 100121 BPL 15$ ;;IF NOT, EXIT
1130 003136 117746 176004 MOVB @STKB,-(SP) ;;YES
1131 003142 042716 177600 BIC #^C177,(SP) ;;MAKE IT 7-BIT ASCII
1132 003146 021627 000007 CMP (SP),#7 ;;IS IT A CONTROL-G?
1133 003152 001300 BNE 2$ ;;IF NOT, PUT IT IN THE TTY QUEUE
1134 ;;AND EXIT
1135
1136 ::*****
1137 ::*CONTROL IS PASSED TO THIS POINT FROM EIT 'R THE TTY INTERRUPT SERVICE
1138 ::*ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A
1139 ::*CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.
1140 003154 123727 001134 000001 6$: CMPB $AUTOB,#1 ;;ARE WE RUNNING IN AUTO-MODE?
1141 003162 001674 BEQ 2$ ;;BRANCH IF YES
1142 003164 005726 TST (SP)+ ;;CLEAR CONTROL-G OFF STACK
1143 003166 004737 002620 JSR PC,$TKINT ;;FLUSH THE TTY INPUT QUEUE
1144 003172 005077 175746 CLR @STKS ;;DISABLE TTY KEYBOARD INTERRUPTS
1145 003176 112737 000001 001135 MOVB #1,$INTAG ;;SET INTERRUPT MODE INDICATOR
1146 003204 104401 004034 $GTSWR: TYPE , $CNTLG ;;ECHO THE CONTROL-G (^G)
1147 003210 104401 004041 TYPE , $MSWR ;;TYPE CURRENT CONTENTS
1148 003214 013746 000176 MOV SWREG,-(SP) ;;SAVE SWREG FOR TYPEOUT
1149 003220 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1150 003222 104401 004052 TYPE , $MNEW ;;PROMPT FOR NEW SWR
1151 003226 005046 19$: CLR -(SP) ;;CLEAR COUNTER
1152 003230 005046 CLR -(SP) ;;THE NEW SWR
1153 003232 105777 175706 7$: TSTB @STKS ;;CHAR THERE?
1154 003236 100375 BPL 7$ ;;IF NOT TRY AGAIN
1155
1156 003240 117746 175702 MOVB @STKB,-(SP) ;;PICK UP CHAR
1157 003244 042716 177600 BIC #^C177,(SP) ;;MAKE IT 7-BIT ASCII
1158
1159 003250 021627 000003 CMP (SP),#3 ;;IS IT A CONTROL-C?
1160 003254 001015 BNE 9$ ;;BRANCH IF NOT
1161 003256 104401 004022 TYPE , $CNTLC ;;YES, ECHO CONTROL-C (^C)
1162 003262 062706 000006 ADD #6,SP ;;CLEAN UP STACK
1163 003266 123727 001135 000001 CMPB $INTAG,#1 ;;REENABLE TTY KEYBOARD INTERRUPTS?
1164 003274 001003 BNE 8$ ;;BRANCH IF NO
1165 003276 012777 000100 175640 MOV #100,@STKS ;;ALLOW TTY KEYBOARD INTERRUPTS
1166 003304 000137 001164 8$: JMP START ;;CONTROL-C RESTART
1167
1168
1169 003310 021627 000025 9$: CMP (SP),#25 ;;IS IT A CONTROL-U?
1170 003314 001005 BNE 10$ ;;BRANCH IF NOT
1171 003316 104401 004027 TYPE , $CNTLU ;;YES, ECHO CONTROL-U (^U)
1172 003322 062706 000006 20$: ADD #6,SP ;;IGNORE PREVIOUS INPUT
1173 003326 000737 BR 19$ ;;LET'S TRY IT AGAIN

```

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 27  
TTY INPUT ROUTINE

```

1174
1175
1176 003330 021627 000015 10$: CMP (SP),#15 ::IS IT A <CR>?
1177 003334 001022 BNE 16$ ::BRANCH IF NO
1178 003336 005766 000004 TST 4(SP) ::YES, IS IT THE FIRST CHAR?
1179 003342 001403 BEQ 11$ ::BRANCH IF YES
1180 003344 016677 000002 175566 MOV 2(SP),@SWR ::SAVE NEW SWR
1181 003352 062706 000006 11$: ADD #6,SP ::CLEAR UP STACK
1182 003356 104401 001161 14$: TYPE ,$CRLF ::ECHO <CR> AND <LF>
1183 003362 123727 001135 000001 CMPB $INTAG,#1 ::RE-ENABLE TTY KBD INTERRUPTS?
1184 003370 001003 BNE 15$ ::BRANCH IF NOT
1185 003372 012777 000100 175544 MOV #100,@$TKS ::RE-ENABLE TTY KBD INTERRUPTS
1186 003400 000002 15$: RTI ::RETURN
1187 003402 004737 005614 16$: JSR PC,$TYPEC ::ECHO CHAR
1188 003406 021627 000060 CMP (SP),#60 ::CHAR < 0?
1189 003412 002420 BLT 18$ ::BRANCH IF YES
1190 003414 021627 000067 CMP (SP),#67 ::CHAR > 7?
1191 003420 003015 BGT 18$ ::BRANCH IF YES
1192 003422 042726 000060 BIC #60,(SP)+ ::STRIP-OFF ASCII
1193 003426 005766 000002 TST 2(SP) ::IS THIS THE FIRST CHAR
1194 003432 001403 BEQ 17$ ::BRANCH IF YES
1195 003434 006316 ASL (SP) ::NO, SHIFT PRESENT
1196 003436 006316 ASL (SP) :: CHAR OVER TO MAKE
1197 003440 006316 ASL (SP) :: ROOM FOR NEW ONE.
1198 003442 005266 000002 17$: INC 2(SP) ::KEEP COUNT OF CHAR
1199 003446 056616 177776 BIS -2(SP),(SP) ::SET IN NEW CHAR
1200 003452 000667 BR 7$ ::GET THE NEXT ONE
1201 003454 104401 001160 18$: TYPE , $QUES ::TYPE ?<CR><LF>
1202 003460 000720 BR 20$ ::SIMULATE CONTROL-U
1203 .DSABL LSB
1204
1205
1206 *****
1207 *THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
1208 *CALL:
1209 * RDCHR ::GET A CHARACTER FROM THE QUEUE
1210 * RETURN HERE ::CHARACTER IS ON THE STACK
1211 * ::WITH PARITY BIT STRIPPED OFF
1212 *
1213
1214 003462 011646 $RDCHR: MOV (SP),-(SP) ::PUSH DOWN THE PC AND
1215 003464 016666 000004 000002 MOV 4(SP),2(SP) ::THE PS
1216 003472 005066 000004 CLR 4(SP) ::GET READY FOR A CHARACTER
1217 003476 005046 CLR -(SP) ::PUT NEW PS ON STACK
1218 003500 012746 003506 MOV #64$,-(SP) ::PUT NEW PC ON STACK
1219 003504 000002 RTI ::POP NEW PC AND PS
1220 003506 64$:
1221 003506 005737 002602 1$: TST $TKCNT ::WAIT ON A CHARACTER
1222 003512 001775 BEQ 1$
1223 003514 005337 002602 DEC $TKCNT ::DECREMENT THE COUNTER
1224 003520 117766 177062 000004 MOVB @TKQOUT,4(SP) ::GET ONE CHARACTER
1225 003526 005237 002606 INC $TKQOUT ::UPDATE THE POINTER
1226 003532 023727 002606 002620 CMP $TKQOUT,#$TKQEND ::DID IT GO OFF OF THE END?
1227 003540 001003 BNE 2$ ::BRANCH IF NO
1228 003542 012737 002610 002606 MOV #$TKQRT,$TKQOUT ::RESET THE POINTER
1229 003550 000002 2$: RTI ::RETURN

```



MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 28  
TTY INPUT ROUTINE

```

1230
1231
1232
1233
1234
1235
1236
1237 003552 010346
1238 003554 005046
1239 003556 012703 004006
1240 003562 022703 004016
1241 003566 101456
1242 003570 104407
1243 003572 112613
1244 003574 122713 000177
1245 003600 001022
1246 003602 005716
1247 003604 001007
1248 003606 112737 000134 004004
1249 003614 104401 004004
1250 003620 012716 177777
1251 003624 005303
1252 003626 020327 004006
1253 003632 103434
1254 003634 111337 004004
1255 003640 104401 004004
1256 003644 000746
1257 003646 005716
1258 003650 001466
1259 003652 112737 000134 004004
1260 003660 104401 004004
1261 003664 005016
1262 003666 122713 000025
1263 003672 001003
1264 003674 104401 004027
1265 003700 000726
1266 003702 122713 000022
1267 003706 001011
1268 003710 105013
1269 003712 104401 001161
1270 003716 104401 004006
1271 003722 000717
1272 003724 104401 001160
1273 003730 000712
1274 003732 111337 004004
1275 003736 104401 004004
1276 003742 122723 000015
1277 003746 001305
1278 003750 105063 177777
1279 003754 104401 001162
1280 003760 005726
1281 003762 012603
1282 003764 011646
1283 003766 016666 000004 000002
1284 003774 012766 004006 000004
1285 004002 000002

::*****
::THIS ROUTINE WILL INPUT A STRING FROM THE TTY
::CALL:
::* RDLIN          ::INPUT A STRING FROM THE TTY
::* RETURN HERE   ::ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
::*              ::TERMINATOR WILL BE A BYTE OF ALL 0'S

$RDLIN: MOV R3, -(SP)      ::SAVE R3
        CLR -(SP)        ::CLEAR THE RUBOUT KEY
1$:     MOV #TTYIN, R3    ::GET ADDRESS
2$:     CMP #TTYIN+10, R3 ::BUFFER FULL?
        BLOS 4$          ::BR IF YES
        RDCHR          ::GO READ ONE CHARACTER FROM THE TTY
10$:    MOVB (SP)+, (R3)  ::GET CHARACTER
        CMPB #177, (R3)  ::IS IT A RUBOUT
        BNE 5$          ::BR IF NO
        TST (SP)        ::IS THIS THE FIRST RUBOUT?
        BNE 6$          ::BR IF NO
1248:   MOVB #' \, 9$    ::TYPE A BACK SLASH
        TYPE ,9$
        MOV #-1, (SP)   ::SET THE RUBOUT KEY
6$:     DEC R3           ::BACKUP BY ONE
        CMP R3, #TTYIN  ::STACK EMPTY?
        BLO 4$          ::BR IF YES
        MOVB (R3), 9$   ::SETUP TO TYPEOUT THE DELETED CHAR.
        TYPE ,9$       ::GO TYPE
        BR 2$           ::GO READ ANOTHER CHAR.
5$:     TST (SP)        ::RUBOUT KEY SET?
        BEQ 7$          ::BR IF NO
1259:   MOVB #' \, 9$    ::TYPE A BACK SLASH
        TYPE ,9$
        CLR (SP)        ::CLEAR THE RUBOUT KEY
7$:     CMPB #25, (R3)  ::IS CHARACTER A CTRL U?
        BNE 8$          ::BR IF NO
        TYPE , $CNTLU  ::TYPE A CONTROL 'U'
        BR 1$           ::GO START OVER
8$:     CMPB #22, (R3)  ::IS CHARACTER A '^R'?
        BNE 3$          ::BRANCH IF NO
        CLRB (R3)       ::CLEAR THE CHARACTER
        TYPE , $CRLF    ::TYPE A 'CR' & 'LF'
        TYPE , TTYIN    ::TYPE THE INPUT STRING
        BR 2$           ::GO PICKUP ANOTHER CHACTER
4$:     TYPE $QUES      ::TYPE A '?'
        BR 1$           ::CLEAR THE BUFFER AND LOOP
3$:     MOVB (R3), 9$   ::ECHO THE CHARACTER
        TYPE ,9$
        CMPB #15, (R3)+ ::CHECK FOR RETURN
        BNE 2$          ::LOOP IF NOT RETURN
        CLRB -1(R3)    ::CLEAR RETURN (THE 15)
        TYPE , $LF     ::TYPE A LINE FEED
        TST (SP)+      ::CLEAN RUBOUT KEY FROM THE STACK
        MOV (SP)+, R3   ::RESTORE R3
        MOV (SP), -(SP) ::ADJUST THE STACK AND PUT ADDRESS OF THE
        MOV 4(SP), 2(SP) :: FIRST ASCII CHARACTER ON IT
        MOV #TTYIN, 4(SP)
RTI     ::RETURN

```

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 29  
TTY INPUT ROUTINE

|      |        |        |        |        |  |                                   |                  |   |
|------|--------|--------|--------|--------|--|-----------------------------------|------------------|---|
| 1286 | 004004 | 000    |        |        | 9\$:   | .BYTE                             | 0                | ::STORAGE FOR ASCII CHAR. TO TYPE         |
| 1287 | 004005 | 000    |        |        |  | .BYTE                             | 0                | ::TERMINATOR                              |
| 1288 | 004006 | 000010 |        |        | \$TTYIN:   | .BLKB                             | 10               | ::RESERVE 10 BYTES FOR TTY INPUT          |
| 1289 | 004016 | 177607 | 000377 |        | \$BELL:  | .ASCIZ                            | <207><377><377>  | ::CODE FOR BELL                           |
| 1290 | 004022 | 041536 | 005015 | 000    | \$CNTLC:   | .ASCIZ                            | /^C/<15><12>     | ::CONTROL 'C'                             |
| 1291 | 004027 | 136    | 006525 | 000012 | \$CNTLU:   | .ASCIZ                            | /^U/<15><12>     | ::CONTROL 'U'                             |
| 1292 | 004034 | 043536 | 005015 | 000    | \$CNTLG:   | .ASCIZ                            | /^G/<15><12>     | ::CONTROL 'G'                             |
| 1293 | 004041 | 015    | 051412 | 051127 | \$MSWR:  | .ASCIZ                            | <15><12>/SWR = / |   |
| 1294 | 004046 | 036440 | 000040 |        |  |                                   |                  |   |
| 1295 | 004052 | 020040 | 042516 | 020127 | \$MNEW:  | .ASCIZ                            | / NEW = /        |   |
| 1296 | 004060 | 020075 | 000    |        |  |                                   |                  |   |
| 1297 |        | 004064 |        |        |  |                                   |                  |   |
| 1298 |        |        |        |        | .EVEN  |                                   |                  |   |
| 1299 |        |        |        |        | .SBTTL   | READ AN OCTAL NUMBER FROM THE TTY |                  |   |
| 1300 |        |        |        |        |  |                                   |                  |   |
| 1301 |        |        |        |        | ::*****  |                                   |                  |   |
| 1302 |        |        |        |        | ::*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND       |                                   |                  |   |
| 1303 |        |        |        |        | ::*CHANGE IT TO BINARY.  |                                   |                  |   |
| 1304 |        |        |        |        | ::*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL        |                                   |                  |   |
| 1305 |        |        |        |        | ::*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A '?' WILL BE TYPED     |                                   |                  |   |
| 1306 |        |        |        |        | ::*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST     |                                   |                  |   |
| 1307 |        |        |        |        | ::*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN. |                                   |                  |   |
| 1308 |        |        |        |        | ::*CALL:   |                                   |                  |   |
| 1309 |        |        |        |        | ::*  | RDOCT                             |                  | :::READ AN OCTAL NUMBER                   |
| 1310 |        |        |        |        | ::*  | RETURN HERE                       |                  | :::LOW ORDER BITS ARE ON TOP OF THE STACK |
| 1311 |        |        |        |        | ::*  |                                   |                  | :::HIGH ORDER BITS ARE IN \$HIOCT         |
| 1312 | 004064 | 011646 |        |        | \$RDOCT:   | MOV                               | (SP),-(SP)       | :::PROVIDE SPACE FOR THE                  |
| 1313 | 004066 | 016666 | 000004 | 000002 |  | MOV                               | 4(SP),2(SP)      | :::INPUT NUMBER                           |
| 1314 | 004074 | 010046 |        |        |  | MOV                               | R0,-(SP)         | :::PUSH R0 ON STACK                       |
| 1315 | 004076 | 010146 |        |        |  | MOV                               | R1,-(SP)         | :::PUSH R1 ON STACK                       |
| 1316 | 004100 | 010246 |        |        |  | MOV                               | R2,-(SP)         | :::PUSH R2 ON STACK                       |
| 1317 | 004102 | 104410 |        |        | 1\$:   | RDLIN                             |                  | :::READ AN ASCII LINE                     |
| 1318 | 004104 | 012600 |        |        |  | MOV                               | (SP)+,R0         | :::GET ADDRESS OF 1ST CHARACTER           |
| 1319 | 004106 | 010037 | 004212 |        |  | MOV                               | R0,5\$           | :::AND SAVE IT                            |
| 1320 | 004112 | 005001 |        |        |  | CLR                               | R1               | :::CLEAR DATA WORD                        |
| 1321 | 004114 | 005002 |        |        |  | CLR                               | R2               |   |
| 1322 | 004116 | 112046 |        |        | 2\$:   | MOVB                              | (R0)+,-(SP)      | :::PICKUP THIS CHARACTER                  |
| 1323 | 004120 | 001420 |        |        |  | BEQ                               | 3\$              | :::IF ZERO GET OUT                        |
| 1324 | 004122 | 122716 | 000060 |        |  | CMPB                              | #'0,(SP)         | :::MAKE SURE THIS CHARACTER               |
| 1325 | 004126 | 003026 |        |        |  | BGT                               | 4\$              | :::IS AN OCTAL DIGIT                      |
| 1326 | 004130 | 122716 | 000067 |        |  | CMPB                              | #'7,(SP)         |   |
| 1327 | 004134 | 002423 |        |        |  | BLT                               | 4\$              |   |
| 1328 | 004136 | 006301 |        |        |  | ASL                               | R1               | :::*2                                     |
| 1329 | 004140 | 006102 |        |        |  | ROL                               | R2               |   |
| 1330 | 004142 | 006301 |        |        |  | ASL                               | R1               | :::*4                                     |
| 1331 | 004144 | 006102 |        |        |  | ROL                               | R2               |   |
| 1332 | 004146 | 006301 |        |        |  | ASL                               | R1               | :::*8                                     |
| 1333 | 004150 | 006102 |        |        |  | ROL                               | R2               |   |
| 1334 | 004152 | 042716 | 177770 |        |  | BIC                               | #^C7,(SP)        | :::STRIP THE ASCII JUNK                   |
| 1335 | 004156 | 062601 |        |        |  | ADD                               | (SP)+,R1         | :::ADD IN THIS DIGIT                      |
| 1336 | 004160 | 000756 |        |        |  | BR                                | 2\$              | :::LOOP                                   |
| 1337 | 004162 | 005726 |        |        | 3\$:   | TST                               | (SP)+            | :::CLEAN TERMINATOR FROM STACK            |
| 1338 | 004164 | 010166 | 000012 |        |  | MOV                               | R1,12(SP)        | :::SAVE THE RESULT                        |
| 1339 | 004170 | 010237 | 004222 |        |  | MOV                               | R2,\$HIOCT       |   |
| 1340 | 004174 | 012602 |        |        |  | MOV                               | (SP)+,R2         | :::POP STACK INTO R2                      |
| 1341 | 004176 | 012601 |        |        |  | MOV                               | (SP)+,R1         | :::POP STACK INTO R1                      |

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 30  
READ AN OCTAL NUMBER FROM THE TTY

|      |        |        |        |
|------|--------|--------|--------|
| 1342 | 004200 | 012600 |        |
| 1343 | 004202 | 000002 |        |
| 1344 | 004204 | 005726 |        |
| 1345 | 004206 | 105010 |        |
| 1346 | 004210 | 104401 |        |
| 1347 | 004212 | 000000 |        |
| 1348 | 004214 | 104401 | 001160 |
| 1349 | 004220 | 000730 |        |
| 1350 | 004222 | 000000 |        |
| 1351 |        |        |        |
| 1352 |        |        |        |

```

MOV      (SP)+,R0      ;;POP STACK INTO R0
RTI      ;;RETURN
4$:      TST      (SP)+  ;;CLEAN PARTIAL FROM STACK
          CLRB    (R0)   ;;SET A TERMINATOR
          TYPE    ;;TYPE UP THRU THE BAD CHAR.
5$:      .WORD   0
          TYPE    ,SQUES ;;'?'' 'CR'' & 'LF''
          BR     1$     ;;TRY AGAIN
$HIOCT:  .WORD   0     ;;HIGH ORDER BITS GO HERE
          .SBTTL  MODULE 1.2 SIZE MEMORY

```

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 31  
MODULE 1.2 SIZE MEMORY

1353  
1354  
1355  
1356  
1357  
1358  
1359  
1360  
1361  
1362  
1363  
1364 004224 005037 004574  
1365 004230 004737 004510  
1366 004234 005037 012172  
1367 004240 013737 005220 012166  
1368 004246 013737 005220 012164  
1369 004254 023727 005220 017776  
1370 004262 103415  
1371 004264 162737 000276 012164  
1372 004272 005737 000042  
1373 004276 001407  
1374 004300 023737 000042 000046  
1375 004306 001403  
1376 004310 162737 005500 012164  
1377 004316 005037 012170  
1378 004322 005037 012224  
1379 004326 037727 174606 002000  
1380 004334 001064  
1381 004336 012737 000200 004574  
1382 004344 004737 004510  
1383 004350 005737 004574  
1384 004354 100054  
1385 004356 012737 000001 012172  
1386 004364 022737 007540 005222  
1387 004372 003003  
1388 004374 012737 007540 005222  
1389 004402 013737 005222 012170  
1390 004410 062737 000040 012170  
1391 004416 013737 005222 012226  
1392 004424 006337 012226  
1393 004430 006337 012226  
1394 004434 006337 012226  
1395 004440 006337 012226  
1396 004444 006337 012226  
1397 004450 006337 012226  
1398 004454 052737 003776 012226  
1399 004462 013737 005222 012224  
1400 004470 042737 171777 012224  
1401 004476 000337 012224  
1402 004502 006337 012224  
1403 004506 000207  
1404  
1405  
1406  
1407  
1408

MODULE 1.2 SIZE MEMORY  
CALLED AS FOLLOWS  
JSR PC,SIZE  
THIS MODULE SETS UP MEMORY SIZE INCLUDING EXTENDED ADDRESSING BITS IN ACTIVE PARAMETER TABLE  
SIZE: CLR \$KT11 ;DON'T TRY FOR MEM MGT  
JSR PC,\$SIZE ;GET MEM SIZE WITH KT OFF  
CLR KT ;RESET KT AVAILABLE SWITCH  
MOV \$LSTAD,MMA ;SAVE END OF CORE WITH KT OFF  
MOV \$LSTAD,MSIZE ;GET END OF CORE TO WORK WITH  
CMP \$LSTAD,#17776 ;LESS THAN 8K?  
BLO 35\$ ;BRANCH IF YES, DON'T PRESERVE LOADERS  
SUB #276,MSIZE ;SAVE SPACE FOR ABS LOADER  
TST @#42 ;RUNNING UNDER ACT OR XXDP  
BEQ 35\$ ;BR IF NO  
CMP @#42,@#46 ;RUNNING UNDER XXDP CHAIN?  
BEQ 35\$ ;BR IF NO  
SUB #5500,MSIZE ;SAVE SPACE FOR LOADERS  
35\$: CLR EASIZE ;CLEAN UP EXTENDED ADDR  
CLR EABITS  
BIT @SWR,#SW10 ;CHECK IF MM TO BE USED  
BNE 30\$ ;BR IF NO  
10\$: MOV #200,\$KT11  
20\$: JSR PC,\$SIZE  
TST \$KT11 ;IS MM ACTIVE  
BPL 30\$ ;BR IF NO  
MOV #1,KT ;SET KT IN USE FLAG  
CMP #7540,\$LSTBK ;:124KW ? VRG  
BGT 1\$ ;:NO - BRANCH VRG  
1\$: MOV #7540,\$LSTBK ;:FORCE IT TO 124KW VRG  
MOV \$LSTBK,EASIZE  
ADD #40,EASIZE ;INCR TO FIRST INVALID SETTING  
MOV \$LSTBK,PHYMAX ;START GEN OF MAX PHYSICAL  
ASL PHYMAX ; ADDR  
ASL PHYMAX ; LOW  
ASL PHYMAX ; 16  
ASL PHYMAX ; BITS  
BIS #3776,PHYMAX ;SET UP TO USE LAST 1K TOO  
MOV \$LSTBK,EABITS  
BIC #171777,EABITS ;SAVE ONLY EXT ADDR BITS  
SWAB EABITS ;MOVE TO BIT POS. 3,2  
ASL EABITS ;MOVE TO BIT POS. 4,3 FOR DX11  
30\$: RTS PC ;RETURN  
;SBTTL ROUTINE TO SIZE MEMORY  
;\*\*\*\*\*  
;\*CALL:

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 32  
ROUTINE TO SIZE MEMORY

```

1409          ;*      JSR      PC,$SIZE
1410          ;*      RETURN
1411          ;*SLSTAD WILL CONTAIN:
1412          ;*      WITH KT11 OPTION      -- LAST VIRTUAL ADDRESS OF THE LAST BANK
1413          ;*      WITHOUT KT11 OPTION   -- LAST ABSOLUTE ADDRESS OF AVAILABLE MEMORY
1414          ;*SLSTBK WILL CONTAIN THE LAST BANK AS A SAF
1415          ;*$KT11 IS THE MEMORY MANAGEMENT KEY
1416          ;*BIT07 = 0 DON'T USE MEMORY MANAGEMENT
1417          ;*      MUST BE SETUP BEFORE THE CALL
1418          ;*BIT15 = 0 DON'T HAVE MEMORY MANAGEMENT OPTION
1419          ;*
1420          ;*      DETERMINED BY ROUTINE
1421 004510 010046 $SIZE: MOV      R0,-(SP)      ;;SAVE R0 ON THE STACK
1422 004512 010146      MOV      R1,-(SP)      ;;SAVE R1 ON THE STACK
1423 004514 010246      MOV      R2,-(SP)      ;;SAVE R2 ON THE STACK
1424 004516 010346      MOV      R3,-(SP)      ;;SAVE R3 ON THE STACK
1425 004520 010446      MOV      R4,-(SP)      ;;SAVE R4 ON THE STACK
1426 004522 013746 000114      MOV      @#114,-(SP)      ;;SAVE MEMORY ERROR VECTOR PS & PC      BK001
1427 004526 013746 000116      MOV      @#116,-(SP)
1428 004532 012737 000116 000114      MOV      #116,@#114      ;;IGNORE PARITY ERRORS WHILE SIZING
1429 004540 012737 000002 000116      MOV      #RTI,@#116
1430 004546 013746 000004      MOV      @#ERRVEC,-(SP)      ;;SAVE PRESENT ERROR VECTOR PS & PC
1431 004552 013746 000006      MOV      @#ERRVEC+2,-(SP)
1432 004556 010600      MOV      SP,R0      ;;SAVE THE STACK POINTER
1433          ;;SET THE ERRVEC PS TO THE PRESENT PS
1434 004560 104400      TRAP
1435 004562 012637 000006      MOV      (SP)+,@#ERRVEC+2      ;;PUSH OLD PSW AND PC ON STACK
1436 004566 012701 003776      MOV      #3776,R1      ;;SAVE THE PSW IN @#ERRVEC+2
1437 004572 105727      MOV      (PC)+      ;;SETUP ADDRESS
1438 004574 000200 $KT11: .WORD 200      ;;USE MEMORY MANAGEMENT?
1439 004576 100145      BPL      $SCORE      ;;SET TO USE MEMORY MANAGEMENT
1440 004600 012737 005104 000004      BPL      $SCORE      ;;BR IF NO
1441 004606 005737 177572      MOV      # $KTNEX,@#ERRVEC      ;;SET FOR TIMEOUT
1442 004612 052737 100000 004574      TST      @#SRO      ;;KT11 ARE YOU THERE?
1443 004620 012737 004650 000004      BIS      #100000,$KT11      ;;YES--SET KT11 KEY
1444 004626 005737 170200      MOV      #100$,@#ERRVEC      ;;SET FOR TIMEOUT
1445 004632 012737 176200 004670      TST      @#170200      ;;UNIBUS MAP ARE YOU THERE?      BK001
1446 004640 012737 000200 004666      MOV      #176200,@#$STOP      ;;YES-SET COMPARISON VALUE FOR 11/70      BK001
1447 004646 000411      MOV      #200,@#$MAP      ;;TURN ON MAP INDICATOR      BK001
1448 004650 012737 006200 004670 100$: BR      $MAPRG      ;;GO SET UP MAP REGISTERS      BK001
1449 004656 022626      MOV      #6200,@#$STOP      ;;COMPARISON VALUE FOR 18 BIT MAPPING      BK001
1450 004660 005037 004666      CMP      (SP)+,(SP)+      ;;CLEAN OFF STACK      BK001
1451 004664 000412      CLR      @#$MAP      ;;MAKE SURE MAP INDICATOR TURNED OFF      BK001
1452 004666 000000      BR      $NOMAP      ;;
1453 004670 000000 $MAP: .WORD 0      ;;=200 IF MAP PRESENT      BK001
1454 004672 012703 000037 $STOP: .WORD 0      ;;FILLED WITH APPROPRIATE COMPARISON VALUE      BK001
1455 004676 012702 170200 $MAPRG: MOV      #37,R3      ;;SET UP COUNTER      BK001
1456 004702 005022 100$: MOV      #170200,R2      ;;START WITH MAPLO      BK001
1457 004704 012722 000074      CLR      (R2)+      ;;LOAD ALL MAP REGISTERS      BK001
1458 004710 077304      MOV      #74,(R2)+      ;;WITH THE VALUE 17000000      BK001
1459 004712      SOB      R3,100$      ;;DO ALL 31 REGISTERS      BK001
1460 004712 005046      CLR      -(SP)      ;;INITIALIZE FOR 'PAR' LOADING
1461 004714 012702 172340      MOV      #KIPAR0,R2      ;;ADDRESS OF FIRST 'PAR'
1462 004720 012703 000010      MOV      #^D8,R3      ;;LOAD EIGHT 'PAR.'S' AND EIGHT 'PDR.'S'
1463 004724 012762 077406 177740 1$: MOV      #77406,-40(R2)      ;;PDR = 4K, UP, READ/WRITE
1464 004732 011622      MOV      (SP),(R2)+      ;;LOAD 'PAR'

```



MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 33  
ROUTINE TO SIZE MEMORY

```

1465 004734 062716 000200      ADD      #200,(SP)      ;;UPDATE FOR NEXT 'PAR'
1466 004740 077307      SOB      R3,1$        ;;LOOP UNTIL ALL EIGHT ARE LOADED
1467 004742 012742 177600      MOV      #177600,-(R2) ;;SETUP KIPAR7 FOR I/O
1468 004746 005042      CLR      -(R2)        ;;SETUP KIPAR6 FOR TESTING
1469 004750 012737 004766 000004      MOV      #2$,@#ERRVEC ;;CATCH TIMEOUT IF NO SR3
1470 004756 012737 000060 172516      MOV      #60,@#SR3    ;;ENABLE 22 BIT MODE AND UNIBUS MAP      BK001
1471 004764 000401      BR       3$          ;;THIS PDP-11 HAS A SR3 REGISTER
1472 004766 022626      2$: CMP      (SP)+,(SP)+ ;;CLEAN OFF THE STACK--NO SR3
1473 004770 005237 177572      3$: INC      @#SR0     ;;TURN ON MEMORY MANAGEMENT
1474 004774 012737 005042 000004      MOV      #SKTOUT,@#ERRVEC ;;SET FOR TIME OUT
1475 005002 105737 004666      TSTB    @#SMAP       ;;IS MAP THERE?      BK001
1476 005006 100006      BPL     4$          ;;NO-SKIP      BK001
1477 005010 012737 005064 000114      MOV      #SMMOUT,@#114 ;;SET UP MEMORY ERROR VECTOR      BK001
1478 005016 013737 000006 000116      MOV      @#ERRVEC+2,@#116 ;;LOCK OUT INTERRUPTS      BK001
1479 005024 005737 143776      4$: TST     @#143776   ;;TRAP ON NON-EX-MEM
1480 005030 062712 000040      ADD     #40,(R2)     ;;MAKE A 1K STEP
1481 005034 023712 004670      CMP     @#$STOP,(R2) ;;LAST ONE?
1482 005040 101371      BHI     4$          ;;NO--TRY IT
1483 005042 011202      SKTOUT: MOV    (R2),R2 ;;GET LAST BANK+1
1484 005044 005037 177572      CLR     @#SR0       ;;TURN OFF MEMORY MANAGEMENT
1485 005050 105737 004666      TSTB   @#SMAP       ;;IS MAP THERE?      BK001
1486 005054 100034      BPL     $SIZEX      ;;NO-SKIP      BK001
1487 005056 005037 172516      CLR     @#SR3       ;;TURN OFF MAP      BK001
1488 005062 000431      BR      $SIZEX
1489 005064 013704 177744      SMMOUT: MOV    @#177744,R4 ;;SAVE MEMORY ERROR REGISTER      BK001
1490 005070 010437 177744      MOV    R4,@#177744   ;;CLEAR BITS IN REGISTER      BK001
1491 005074 032704 000001      BIT    #1,R4         ;;MEMORY TIMEOUT?      BK001
1492 005100 001360      BNE     SKTOUT       ;;YES-EXIT      BK001
1493 005102 000002      RTI                    ;;MUST BE PARITY ERROR-IGNORE IT      BK001
1494 005104 042737 100000 004574      SKTNEX: BIC    #100000,$KT11 ;;KT11 NON-EXISTENT
1495 005112 012737 005142 000004      SCORE: MOV    #SCROUT,@#ERRVEC ;;SET FOR TIMEOUT
1496 005120 005002      CLR     R2          ;;SET UP BANK
1497 005122 062701 004000      1$: ADD    #4000,R1   ;;INCREMENT BY 1K
1498 005126 062702 000040      ADD    #40,R2       ;;1K STEP
1499 005132 005711      TST    (R1)         ;;TRAP ON TIME OUT
1500 005134 022701 177776      CMP    #177776,R1  ;;LAST ONE
1501 005140 001370      BNE     1$          ;;NO--TRY AGAIN
1502 005142 162701 004000      SCROUT: SUB   #4000,R1
1503 005146 162702 000040      $SIZEX: SUB   #40,R2 ;;DROP BACK
1504 005152 010006      MOV    R0,SP        ;;RESTORE THE STACK
1505 005154 012637 000006      MOV    (SP)+,@#ERRVEC+2 ;;RESTORE ERROR VECTOR
1506 005160 012637 000004      MOV    (SP)+,@#ERRVEC
1507 005164 012637 000116      MOV    (SP)+,@#116  ;;RESTORE MEMORY ERROR VECTOR
1508 005170 012637 000114      MOV    (SP)+,@#114
1509 005174 010137 005220      MOV    R1,$LSTAD    ;;LAST ADDRESS
1510 005200 010237 005222      MOV    R2,$LSTBK    ;;LAST BANK
1511 005204 012604      MOV    (SP)+,R4     ;;RESTORE R4      BK001
1512 005206 012603      MOV    (SP)+,R3     ;;RESTORE R3
1513 005210 012602      MOV    (SP)+,R2     ;;RESTORE R2
1514 005212 012601      MOV    (SP)+,R1     ;;RESTORE R1
1515 005214 012600      MOV    (SP)+,R0     ;;RESTORE R0
1516 005216 000207      RTS    PC
1517 005220 000000      $LSTAD: .WORD 0      ;;CONTAINS THE LAST ADDRESS
1518 005222 000000      $LSTBK: .WORD 0      ;;CONTAINS THE LAST BANK
1519
1520      .SBTTL  MODULE 1.3  CHECK PARAMETERS

```

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 34  
MODULE 1.3 CHECK PARAMETERS

```

1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542 005224 012501
1543 005226 120127 000002
1544 005232 001007
1545 005234 020037 012134
1546 005240 002443
1547 005242 020037 012136
1548 005246 003040
1549 005250 000446
1550 005252 120127 000003
1551 005256 001007
1552 005260 020037 012140
1553 005264 002431
1554 005266 020037 012142
1555 005272 003026
1556 005274 000434
1557 005276 120127 000004
1558 005302 001007
1559 005304 020037 012144
1560 005310 002417
1561 005312 020037 012146
1562 005316 003014
1563 005320 000422
1564 005322 120127 000005
1565 005326 001007
1566 005330 020037 012150
1567 005334 002405
1568 005336 020037 012152
1569 005342 003002
1570 005344 000410
1571 005346 000000
1572 005350 004537 005372
1573 005354 013250
1574 005356 000U00
1575 005360 012700 '77777
1576 005364 000401

```

```

:      MODULE 1.3      CHECK PARAMETERS
:
:      CALLED AS FOLLOWS
:
:      JSR      R5,CHECKP
:      .WORD   X
:      THIS MODULE HAS RESPONSIBILITY FOR VALIDATING PARAMETERS
:      INPUT BY THE OPERATOR.  IT COMPARES THE INPUT PARAMETER
:      AGAINST PARAMETER LIMITS DEFINED IN THE INPUT PARAMETER
:      LIMITS TABLE.  IF THE PARAMETER IS OUTSIDE THE SPECIFIED
:      LIMITS, AN ERROR MESSAGE IS OUTPUT AND ALL ONES IS RETURNED
:      IN R0.  IF THE PARAMETER IS OK, ALL ZEROS IS RETURNED IN
:      R0.
:
:      X DEFINES THE PARAMETER TO CHECK AS FOLLOWS:
:      X=2      STARTING TEST NUMBER
:      X=3      BASE ADDR OF DX CONTROL REGS
:      X=4      DX VECTOR ADDRESS
:      X=5      DX PRIORITY.
:      THE PARAMETER TO BE CHECKED IS PASSED IN R0.
:
CHECKP: MOV      (R5)+,R1      ;GET PARAMETER TO BE CHECKED
        CMPB    R1,#2      ;IS IT TEST NO?
        BNE    10$      ;BR IF NO
        CMP    RO,LTESTN   ;CHECK LOW LIMIT
        BLT    90$      ;BR IF ERROR
        CMP    RO,HTESTN   ;CHECK HIGH LIMIT
        BGT    90$      ;BR IF ERROR
        BR     80$      ;GO TO OK RETURN
10$:    CMPB    R1,#3      ;IS IT BASE ADDR OF DX
        BNE    20$      ;BR IF NO
        CMP    RO,LUBA     ;CHECK LOW LIMIT
        BLT    90$      ;BR IF BAD
        CMP    RO,HUBA     ;CHECK HIGH LIMIT
        BGT    90$      ;BR IF BAD
        BR     80$      ;OK, RETURN
20$:    CMPB    R1,#4      ;IS IT THE VECTOR?
        BNE    30$      ;BR IF NO
        CMP    RO,LVEC     ;CHECK THE LOW LIMIT
        BLT    90$      ;BR IF BAD
        CMP    RO,HVEC     ;CHECK THE HIGH LIMIT
        BGT    90$      ;BR IF BAD
        BR     80$      ;OK, RETURN
30$:    CMPB    R1,#5      ;IS PARM PRIORITY LEVEL
        BNE    40$      ;BR IF NO
        CMP    RO,LPRIOR   ;CHECK LOW LIMIT
        BLT    90$      ;BR IF BAD
        CMP    RO,HPRIOR   ;CHECK HIGH LIMIT
        BGT    90$      ;BR IF BAD
        BR     80$      ;OK, RETURN
40$:    HALT
90$:    JSR      R5,PRINT   ;ERROR, BAD MODULE CALL PARM
        .WORD   BADP      ;PRINT PARM LIMIT ERROR
        .WORD   0         ;ADDR OF MESSAGE
        MOV     #-1,R0    ;CONTROL WORD, CRLF + DON'T CONVERT
        BR     95$      ;SET ERROR RETURN CODE
:      ;RETURN

```

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 35  
MODULE 1.3 CHECK PARAMETERS

1577 005366 005000  
1578 005370 000205  
1579  
1580

80\$: CLR R0 ;SET GOOD RETURN CODE  
95\$: RTS R5 ;RETURN TO CALLER.

.SBTTL MODULE 1.3.1 WRITE TTY

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 36  
MODULE 1.3.1 WRITE TTY

1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1596 005372  
1597 005372 010046  
1598 005374 010146  
1599 005376 012500  
1600 005400 012501  
1601 005402 032701 000001  
1602 005406 001002  
1603 005410 104401  
1604 005412 013625  
1605 005414 005701  
1606 005416 100003  
1607 005420 011046  
1608 005422 104402  
1609 005424 000404  
1610 005426 010037 005434  
1611 005432 104401  
1612 005434 000000  
1613 005436  
1614 005436 012601  
1615 005440 012600  
1616 005442 000205  
1617  
1618  
1619  
1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1630  
1631  
1632  
1633  
1634  
1635 005444 105737 001157  
1636 005450 100002

```

MODULE 1.3.1 WRITE TTY
THIS MODULE HANDLES ALL OUTPUT TO THE OPERATOR
CALLED AS FOLLOWS
JSR R5 PRINT ;ADDRESS OF MESSAGE OR OCT #
.WORD MSG
.WORD CTL
RETURN
CTL= BIT0=1 ;DONT - CRLF BEFORE MESSAGE
BIT15=1 ;CONVERT OCTAL TO ASCII AND PRINT

PRINT:
MOV R0,-(SP) ;;PUSH R0 ON STACK
MOV R1,-(SP) ;;PUSH R1 ON STACK
MOV (R5)+,R0 ;GET ADDR OF MSG
MOV (R5)+,R1 ;GET PRINT CONTROL WORD
BIT #1,R1 ;DO WE NEED A CRLF?
BNE 10$ ;BR IF NO
TYPE ;TRAP TO TYPE ROUTINE
.WORD #CRLF1 ;POINTER TO CRLF ASCII
10$: TST R1 ;IS IT OCTAL?
BPL 20$ ;BR IF NO
MOV (R0),-(SP) ;PUT OCTAL # ON STACK
TYPOC ;TYPE 6 CHAR VIA TRAP
BR 40$ ;DONE
20$: MOV R0,25$ ;PUT MESSAGE ADDR INLINE
TYPE ;TRAP TO TYPE ASCII RTN
25$: .WORD 0 ;POINTER TO MSG GOES HERE
40$:
MOV (SP)+,R1 ;;POP STACK INTO R1
MOV (SP)+,R0 ;;POP STACK INTO R0
RTS R5 ;RETURN TO CALLER
;TYPE TRAP CALL COMES HERE
.SBTTL TYPE ROUTINE

*****
;ROUTINE TO TYPE ASCII 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 ASCII STRING
;OR
; TYPE
; MESADR
;
$TYPE: TSTB $TPFLG ;;IS THERE A TERMINAL?
BPL 1$ ;;BR IF YES

```

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 37  
TYPE ROUTINE

```

1637 005452 000000          HALT          ;;HALT HERE IF NO TERMINAL
1638 005454 000407          BR           3$          ;;LEAVE
1639 005456 010046          1$: MOV      R0,-(SP)    ;;SAVE R0
1640 005460 017600 000002    MOV      @2(SP),R0      ;;GET ADDRESS OF ASCIZ STRING
1641 005464 112046          2$: MOV     (R0)+,-(SP)  ;;PUSH CHARACTER TO BE TYPED ONTO STACK
1642 005466 001005          BNE      4$            ;;BR IF IT ISN'T THE TERMINATOR
1643 005470 005726          TST     (SP)+          ;;IF TERMINATOR POP IT OFF THE STACK
1644 005472 012600          60$: MOV   (SP)+,R0     ;;RESTORE R0
1645 005474 062716 000002    3$: ADD   #2,(SP)      ;;ADJUST RETURN PC
1646 005500 000002          RTI                    ;;RETURN
1647 005502 122716 000011    4$: CMPB  #HT,(SP)     ;;BRANCH IF <HT>
1648 005506 001430          BEQ     8$            ;;BRANCH IF NOT <CRLF>
1649 005510 122716 000200    CMPB  #CRLF,(SP)
1650 005514 001006          BNE     5$            ;;POP <CR><LF> EQUIV
1651 005516 005726          TST     (SP)+          ;;TYPE A CR AND LF
1652 005520 104401          TYPE
1653 005522 001161          $CRLF
1654 005524 105037 005742    CLRB   $CHARCNT       ;;CLEAR CHARACTER COUNT
1655 005530 000755          BR      2$            ;;GET NEXT CHARACTER
1656 005532 004737 005614    5$: JSR   PC,$TYPEC     ;;GO TYPE THIS CHARACTER
1657 005536 123726 001156    6$: CMPB  $FILLC,(SP)+  ;;IS IT TIME FOR FILLER CHARS.?
1658 005542 001350          BNE     2$            ;;IF NO GO GET NEXT CHAR.
1659 005544 013746 001154    MOV   $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
1660                                ;;AND THE NULL CHAR.
1661 005550 105366 000001    7$: DECB  1(SP)        ;;DOES A NULL NEED TO BE TYPED?
1662 005554 002770          BLT     6$            ;;BR IF NO--GO POP THE NULL OFF OF STACK
1663 005556 004737 005614    JSR   PC,$TYPEC     ;;GO TYPE A NULL
1664 005562 105337 005742    DECB   $CHARCNT       ;;DO NOT COUNT AS A COUNT
1665 005566 000770          BR      7$            ;;LOOP
1666
1667                                ;HORIZONTAL TAB PROCESSOR
1668
1669 005570 112716 000040    8$: MOV   #' ,(SP)     ;;REPLACE TAB WITH SPACE
1670 005574 004737 005614    9$: JSR   PC,$TYPEC     ;;TYPE A SPACE
1671 005600 132737 000007 005742  BITB   #7,$CHARCNT    ;;BRANCH IF NOT AT
1672 005606 001372          BNE     9$            ;;TAB STOP
1673 005610 005726          TST     (SP)+          ;;POP SPACE OFF STACK
1674 005612 000724          BR      2$            ;;GET NEXT CHARACTER
1675 005614
1676 005614 105777 173324    $TYPEC: TSTB   @STKS          ;;CHAR IN KYBD BUFFER?
1677 005620 100022          BPL     10$           ;;BR IF NOT
1678 005622 017746 173320    MOV   @STKB,-(SP)     ;;GET CHAR
1679 005626 042716 177600    BIC   #177600,(SP)    ;;STRIP EXTRANEIOUS BITS
1680 005632 122716 000023    CMPB  #'XOFF,(SP)     ;;WAS CHAR XOFF
1681 005636 001012          BNE     101$          ;;BR IF NOT
1682 005640
1683 005640 105777 173300    101$: TSTB   #':::         ;;WAIT FOR CHAR
1684 005644 100375          BPL     111$          ;;WAIT FOR CHAR
1685 005646 117716 173274    MOV   @STKB,(SP)     ;;GET CHAR
1686 005652 042716 177600    BIC   #177600,(SP)    ;;STRIP IT
1687 005656 122716 000021    CMPB  #'XON,(SP)     ;;WAS IT XON?
1688 005662 001366          BNE     101$          ;;BR IF NOT
1689 005664
1690 005664 005726          102$: TST     (SP)+          ;;FIX STACK
1691 005666
1692 005666 105777 173256    10$: TSTB   @STPS          ;;WAIT UNTIL PRINTER IS READY

```



MD-11-CZDXJ-C DX11-B ADDRESSING TST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 38  
TYPE ROUTINE

```

1693 005672 100375          BPL      10$
1694 005674 126627 000002 000021  CMPB    2(SP),#$XON      ;; IS CHARACTER A RANDOM XON?
1695 005702 001420          BEQ     $TYPEX          ;; BRANCH IF YES
1696 005704 116677 000002 173240  MOVB    2(SP),@$TPB     ;; LOAD CHAR TO BE TYPED INTO DATA REG.
1697 005712 122766 000015 000002  CMPB    #CR,2(SP)       ;; IS CHARACTER A CARRIAGE RETURN?
1698 005720 001003          BNE     1$              ;; BRANCH IF NO
1699 005722 105037 005742  CLRB    $CHARCNT       ;; YES--CLEAR CHARACTER COUNT
1700 005726 000406          BR      $TYPEX          ;; EXIT
1701 005730 122766 000012 000002 1$:  CMPB    #LF,2(SP)       ;; IS CHARACTER A LINE FEED?
1702 005736 001402          BEQ     $TYPEX          ;; BRANCH IF YES
1703 005740 105227          INCB    (PC)+          ;; COUNT THE CHARACTER
1704 005742 000000          $CHARCNT: .WORD      0  ;; CHARACTER COUNT STORAGE
1705 005744 000207          $TYPEX:  RTS          PC
1706
1707
1708          :TYPOC TRAP CALL COMES TO $TYPOC TAG BELOW
1709          .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
1710
1711          :*****
1712          :*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
1713          :*OCTAL (ASCII) NUMBER AND TYPE IT.
1714          :*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
1715          :*CALL:
1716          :*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
1717          :*      TYPOS    ;;CALL FOR TYPEOUT
1718          :*      .BYTE   N              ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
1719          :*      .BYTE   M              ;;M=1 OR 0
1720          :*                                  ;;1=TYPE LEADING ZEROS
1721          :*                                  ;;0=SUPPRESS LEADING ZEROS
1722          :*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
1723          :*$TYPOS OR $TYPOC
1724          :*CALL:
1725          :*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
1726          :*      TYPON   ;;CALL FOR TYPEOUT
1727          :*
1728          :*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
1729          :*CALL:
1730          :*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
1731          :*      TYPOC   ;;CALL FOR TYPEOUT
1732
1733 005746 017646 000000          $TYPOS: MOV    @2(SP),-(SP)      ;;PICKUP THE MODE
1734 005752 116637 000001 006171  MOVB    1(SP), $OFILL    ;;LOAD ZERO FILL SWITCH
1735 005760 112637 006173          MOVB    (SP)+, $OMODE+1  ;;NUMBER OF DIGITS TO TYPE
1736 005764 062716 000002          ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
1737 005770 000406          BR      $TYPON
1738 005772 112737 000001 006171  $TYPOC: MOVB    #1, $OFILL    ;;SET THE ZERO FILL SWITCH
1739 006000 112737 000006 006173  MOVB    #6, $OMODE+1    ;;SET FOR SIX(6) DIGITS
1740 006006 112737 000005 006170  $TYPON: MOVB    #5, $OCNT    ;;SET THE ITERATION COUNT
1741 006014 010346          MOV     R3,-(SP)        ;;SAVE R3
1742 006016 010446          MOV     R4,-(SP)        ;;SAVE R4
1743 006020 010546          MOV     R5,-(SP)        ;;SAVE R5
1744 006022 113704 006173          MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
1745 006026 005404          NEG     R4
1746 006030 062704 000006          ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
1747 006034 110437 006172          MOVB    R4, $OMODE     ;;SAVE IT FOR USE
1748 006040 113704 006171          MOVB    $OFILL,R4      ;;GET THE ZERO FILL SWITCH

```

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 39  
BINARY TO OCTAL (ASCII) AND TYPE

|      |        |        |               |          |        |                               |                                   |
|------|--------|--------|---------------|----------|--------|-------------------------------|-----------------------------------|
| 1749 | 006044 | 016605 | 000012        |          | MOV    | 12(SP),R5                     | ::PICKUP THE INPUT NUMBER         |
| 1750 | 006050 | 005003 |               |          | CLR    | R3                            | ::CLEAR THE OUTPUT WORD           |
| 1751 | 006052 | 006105 |               | 1\$:     | ROL    | R5                            | ::ROTATE MSB INTO 'C'             |
| 1752 | 006054 | 000404 |               |          | BR     | 3\$                           | ::GO DO MSB                       |
| 1753 | 006056 | 006105 |               | 2\$:     | ROL    | R5                            | ::FORM THIS DIGIT                 |
| 1754 | 006060 | 006105 |               |          | ROL    | R5                            |                                   |
| 1755 | 006062 | 006105 |               |          | ROL    | R5                            |                                   |
| 1756 | 006064 | 010503 |               |          | MOV    | R5,R3                         |                                   |
| 1757 | 006066 | 006103 |               | 3\$:     | ROL    | R3                            | ::GET LSB OF THIS DIGIT           |
| 1758 | 006070 | 105337 | 006172        |          | DECB   | \$OMODE                       | ::TYPE THIS DIGIT?                |
| 1759 | 006074 | 100016 |               |          | BPL    | 7\$                           | ::BR IF NO                        |
| 1760 | 006076 | 042703 | 177770        |          | BIC    | #177770,R3                    | ::GET RID OF JUNK                 |
| 1761 | 006102 | 001002 |               |          | BNE    | 4\$                           | ::TEST FOR 0                      |
| 1762 | 006104 | 005704 |               |          | TST    | R4                            | ::SUPPRESS THIS 0?                |
| 1763 | 006106 | 001403 |               |          | BEQ    | 5\$                           | ::BR IF YES                       |
| 1764 | 006110 | 005204 |               | 4\$:     | INC    | R4                            | ::DON'T SUPPRESS ANYMORE 0'S      |
| 1765 | 006112 | 052703 | 000060        |          | BIS    | #'0,R3                        | ::MAKE THIS DIGIT ASCII           |
| 1766 | 006116 | 052703 | 000040        | 5\$:     | BIS    | #',R3                         | ::MAKE ASCII IF NOT ALREADY       |
| 1767 | 006122 | 110337 | 006166        |          | MOVB   | R3,8\$                        | ::SAVE FOR TYPING                 |
| 1768 | 006126 | 104401 | 006166        |          | TYPE   | ,8\$                          | ::GO TYPE THIS DIGIT              |
| 1769 | 006132 | 105337 | 006170        | 7\$:     | DECB   | \$OCNT                        | ::COUNT BY 1                      |
| 1770 | 006136 | 003347 |               |          | BGT    | 2\$                           | ::BR IF MORE TO DO                |
| 1771 | 006140 | 002402 |               |          | BLT    | 6\$                           | ::BR IF DONE                      |
| 1772 | 006142 | 005204 |               |          | INC    | R4                            | ::INSURE LAST DIGIT ISN'T A BLANK |
| 1773 | 006144 | 000744 |               |          | BR     | 2\$                           | ::GO DO THE LAST DIGIT            |
| 1774 | 006146 | 012605 |               | 6\$:     | MOV    | (SP)+,R5                      | ::RESTORE R5                      |
| 1775 | 006150 | 012604 |               |          | MOV    | (SP)+,R4                      | ::RESTORE R4                      |
| 1776 | 006152 | 012603 |               |          | MOV    | (SP)+,R3                      | ::RESTORE R3                      |
| 1777 | 006154 | 016666 | 000002 000004 |          | MOV    | 2(SP),4(SP)                   | ::SET THE STACK FOR RETURNING     |
| 1778 | 006162 | 012616 |               |          | MOV    | (SP)+,(SP)                    |                                   |
| 1779 | 006164 | 000002 |               |          | RTI    |                               | ::RETURN                          |
| 1780 | 006166 | 000    |               | 8\$:     | .BYTE  | 0                             | ::STORAGE FOR ASCII DIGIT         |
| 1781 | 006167 | 000    |               |          | .BYTE  | 0                             | ::TERMINATOR FOR TYPE ROUTINE     |
| 1782 | 006170 | 000    |               | \$OCNT:  | .BYTE  | 0                             | ::OCTAL DIGIT COUNTER             |
| 1783 | 006171 | 000    |               | \$OFILL: | .BYTE  | 0                             | ::ZERO FILL SWITCH                |
| 1784 | 006172 | 000000 |               | \$OMODE: | .WORD  | 0                             | ::NUMBER OF DIGITS TO TYPE        |
| 1785 |        |        |               |          | .SBTTL | MODULE 2.0 INTERRUPT HANDLERS |                                   |

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 40  
MODULE 2.0 INTERRUPT HANDLERS

1786  
1787  
1788  
1789  
1790  
1791  
1792  
1793  
1794  
1795  
1796  
1797  
1798  
1799  
1800  
1801  
1802  
1803  
1804  
1805  
1806  
1807  
1808  
1809  
1810  
1811  
1812  
1813  
1814  
1815  
1816  
1817  
1818  
1819  
1820  
1821  
1822  
1823  
1824  
1825  
1826  
1827  
1828  
1829  
1830  
1831  
1832  
1833  
1834  
1835  
1836  
1837  
1838  
1839  
1840  
1841

MODULE 2.0

INTERRUPT HANDLERS

.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.

```
$TRAP:  MOV    2(SP),-(SP)    ;;ASSUME THE STATUS OF
        BIC    #20,(SP)     ;; THE CALLER--DO NOT ALLOW
        MOV    #1$,-(SP)    ;; T-BIT TRAPS
        RTI                    ;;SET THE NEW STATUS
1$:     MOV    R0,-(SP)     ;;SAVE R0
        MOV    2(SP),R0     ;;GET TRAP ADDRESS
        TST    -(R0)        ;;BACKUP BY 2
        MOVB   (R0),R0     ;;GET RIGHT BYTE OF TRAP
        ASL    R0           ;;POSITION FOR INDEXING
        MOV    $TRPAD(R0),R0 ;;INDEX TO TABLE
        RTS    R0          ;;GO TO ROUTINE
```

;;THIS IS USE TO HANDLE THE 'GETPRI' MACRO

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

.SBTTL TRAP TABLE

;;THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED  
;;BY THE 'TRAP' INSTRUCTION.

| ROUTINE        |              |                 |  |
|----------------|--------------|-----------------|--|
| -----          |              |                 |  |
| \$TRPAD: .WORD | \$TRAP2      |                 |  |
| \$TYPE         | ::CALL=TYPE  | TRAP+1(104401)  | TTY TYPEOUT ROUTINE                    |
| \$TYPOC        | ::CALL=TYPOC | TRAP+2(104402)  | TYPE OCTAL NUMBER (WITH LEADING ZEROS) |
| \$TYPOS        | ::CALL=TYPOS | TRAP+3(104403)  | TYPE OCTAL NUMBER (NO LEADING ZEROS)   |
| \$TYPON        | ::CALL=TYPON | TRAP+4(104404)  | TYPE OCTAL NUMBER (AS PER LAST CALL)   |
| \$GTSWR        | ::CALL=GTSWR | TRAP+5(104405)  | GET SOFT-SWR SETTING                   |
| \$CKSWR        | ::CALL=CKSWR | TRAP+6(104406)  | TEST FOR CHANGE IN SOFT-SWR            |
| \$RDCHR        | ::CALL=RDCHR | TRAP+7(104407)  | TTY TYPEIN CHARACTER ROUTINE           |
| \$RDLIN        | ::CALL=RDLIN | TRAP+10(104410) | TTY TYPEIN STRING ROUTINE              |
| \$RDOCT        | ::CALL=RDOCT | TRAP+11(104411) | READ AN OCTAL NUMBER FROM TTY          |

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 41  
TRAP TABLE

1842  
1843  
1844  
1845  
1846  
1847 006272 012737 006436 000024  
1848 006300 012737 000340 000026  
1849 006306 010046  
1850 006310 010146  
1851 006312 010246  
1852 006314 010346  
1853 006316 010446  
1854 006320 010546  
1855 006322 017746 172612  
1856 006326 010637 006442  
1857 006332 012737 006344 000024  
1858 006340 000000  
1859 006342 000776  
1860  
1861  
1862  
1863 006344 012737 006436 000024  
1864 006352 013706 006442  
1865 006356 005037 006442  
1866 006362 005237 006442  
1867 006366 001375  
1868 006370 012677 172544  
1869 006374 012605  
1870 006376 012604  
1871 006400 012603  
1872 006402 012602  
1873 006404 012601  
1874 006406 012600  
1875 006410 012737 006272 000024  
1876 006416 012737 000340 000026  
1877 006424 104401  
1878 006426 006444  
1879 006430 012716  
1880 006432 001334  
1881 006434 000002  
1882 006436 000000  
1883 006440 000776  
1884 006442 000000  
1885 006444 005015 047520 042527  
1886 006452 000122  
1887

.SBTTL POWER DOWN AND UP ROUTINES

```

*****
:POWER DOWN ROUTINE
$PWRDN: MOV    # $ILLUP,@#PWRVEC  ;;SET FOR FAST UP
        MOV    #340,@#PWRVEC+2  ;;PRIO:7
        MOV    R0,-(SP)          ;;PUSH R0 ON STACK
        MOV    R1,-(SP)          ;;PUSH R1 ON STACK
        MOV    R2,-(SP)          ;;PUSH R2 ON STACK
        MOV    R3,-(SP)          ;;PUSH R3 ON STACK
        MOV    R4,-(SP)          ;;PUSH R4 ON STACK
        MOV    R5,-(SP)          ;;PUSH R5 ON STACK
        MOV    @SWR,-(SP)        ;;PUSH @SWR ON STACK
        MOV    SP,$SAVR6         ;;SAVE SP
        MOV    # $PWRUP,@#PWRVEC ;;SET UP VECTOR
        HALT
        BR     .-2              ;;HANG UP

*****
:POWER UP ROUTINE
$PWRUP: MOV    # $ILLUP,@#PWRVEC  ;;SET FOR FAST DOWN
        MOV    $SAVR6,SP         ;;GET SP
        CLR    $SAVR6           ;;WAIT LOOP FOR THE TTY
1$:      INC    $SAVR6           ;;WAIT FOR THE INC
        BNE    1$               ;;OF WORD
        MOV    (SP)+,@SWR        ;;POP STACK INTO @SWR
        MOV    (SP)+,R5          ;;POP STACK INTO R5
        MOV    (SP)+,R4          ;;POP STACK INTO R4
        MOV    (SP)+,R3          ;;POP STACK INTO R3
        MOV    (SP)+,R2          ;;POP STACK INTO R2
        MOV    (SP)+,R1          ;;POP STACK INTO R1
        MOV    (SP)+,R0          ;;POP STACK INTO R0
        MOV    # $PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
        MOV    #340,@#PWRVEC+2  ;;PRIO:7
        TYPE
$PWRMG: .WORD $POWER           ;;REPORT THE POWER FAILURE
        MOV    (PC)+,(SP)       ;;POWER FAIL MESSAGE POINTER
$PWRAD: .WORD RESTART         ;;RESTART AT RESTART
        RTI                    ;;RESTART ADDRESS
$ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
        BR     .-2              ;; BEFORE THE POWER DOWN WAS COMPLETE
$SAVR6: 0
$POWER: .ASCIIZ <15><12>'POWER'
        .EVEN

```

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 42  
MODULE 3.0.1 TST1 FAST NPR DATA TEST

```

1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902 006454 004537 007050
1903 006460 004737 007100
1904 006464 004537 010620
1905 006470 100402
1906
1907 006472 013700 012160
1908 006476 012720 006550
1909 006502 012710 000340
1910 006506 012737 000402 012232 LOOP1:
1911
1912 006514 004737 007030
1913 006520 012737 100525 012120
1914 006526 053777 012236 003324
1915 006534 013777 012242 003322
1916 006542 004537 006706
1917 006546 177376
1918
1919
1920
1921 006550 005737 007026
1922 006554 001006
1923 006556 012716 006564
1924 006562 000002
1925 006564 012705 006572
1926 006570 000205
1927 006572 013700 012230
1928 006576 013701 012232
1929 006602 123720 012120
1930 006606 001404
1931 006610 004537 011272
1932 006614 000001
1933 006616 012306
1934 006620 005301
1935 006622 001367
1936 006624 042777 000200 003226
1937 006632 042777 000004 003240
1938 006640 042777 000030 003212
1939 006646 105737 012120
1940 006652 100404
1941 006654 012737 100652 012120
1942 006662 000721
1943

```

.SBTTL MODULE 3.0.1 TST1 FAST NPR DATA TEST

MODULE 3.0.1 FAST NPR DATA TEST

THIS IS A FUNCTIONAL TEST TO CHECK THE ABILITY OF THE DX TO DO NPR'S OF DATA INTO MEMORY WITH LARGE BYTE COUNTS. THE DXBA IS CHECKED FOR ITS ABILITY TO COUNT AND ALL CARRIES IN THE COUNTER ARE EXERCISED. THE DATA TRANSFERS ACCESS ALL AVAILABLE MEMORY EXCEPT WHERE THE PROGRAM IS LOCATED, AND THE AREA WHICH MAY BE OCCUPIED BY A LOADER.

```

TST1: JSR R5,INIT ;INITIALIZE THE SOFTWARE, PS, ETC
      JSR PC,DXRES ;PUT THE DX IN A GOOD KNOWN STATE
      JSR R5,BUFAD ;GET FIRST BUFFER BASE AND SET UP KT
      .WORD 100402 ;CTL BIT 15=INITIALIZE, BUFFER BASE DIVISIBLE
                        ;BY 400, SET UP BUFMAX
      MOV AVECT,R0 ;SET UP
      MOV #INTR1,(R0)+ ;DX INTERRUPT
      MOV #340,(R0) ;AND DX PSW
      MOV #402,BUF SIZE ;BUFFER SIZE A LITTLE BIGGER THAN OFFSET
                        ;TO CREATE A CARRY OUT OF HIGH ORDER BIT
10$: JSR PC,BUFCLR ;CLEAR THE BUFFER
      MOV #100525,DXMOO ;SET DATA PATTERN AND OPOUT
G01: BIS EACUR,@DXCSA ;SET UP DX EXT ADDR BITS
      MOV BFBAPH,@DXBAA ;SET UP DX BUS ADDR REG.
      JSR R5,DXGO ;START THE DX
      .WORD -402 ;BYTE COUNT

```

;INTERRUPT WILL RETURN HERE AFTER DATA TRANSFER IS COMPLETE

```

INTR1: TST INTERR ;DID WE GET HERE VIA INTER
      BNE 8$ ;BR IF NO, STACK IS OK
      MOV #5$,@SP ;SET UP RETURN ADDR
      RTI ;RELIEVE INTERRUPT STATUS
5$: MOV #8$,R5 ;SET UP JSR RETURN ADDR
      RTS ;RELIEVE SUBROUTINE STATUS
8$: MOV BUFBAS,R0 ;GET STARTING ADDR
      MOV BUFSIZE,R1 ;GET BUFFER SIZE
10$: CMPB DXMOO,(R0)+ ;CHECK A BYTE
      BEQ 15$ ;BRANCH IF NO ERROR
12$: JSR R5,ERRR ;REPORT AN ERROR
      .WORD 1 ;ERROR ID
      .WORD ERR1 ;ADDR OF ERROR MESSAGE
15$: DEC R1 ;DECREMENT BUFFER COUNT
      BNE 10$ ;LOOP UNTIL ALL OF BUFFER CHECKED
      BIC #DONE,@DXCSA ;TURN OFF FAST MODE
      BIC #SOSIEN,@DXESA ;TURN OFF FAST MODE
      BIC #30,@DXCSA ;CLEAR POSSIBLE CARRY IN EA BITS
13$: TSTB DXMOO ;CHECK IF FIRST OF TWO PASSES
      BMI 30$ ;BR IF NOT FIRST PASS
      MOV #100652,DXMOO ;SET UP SECOND DATA PATTERN
      BR G01 ;DO TRANSFER WITH THIS DATA

```

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 43  
MODULE 3.0.1 TST1 FAST NPR DATA TEST

```

1944 006664 004737 007030
1945 006670 004537 010620
1946 006674 000402
1947 006676 005701
1948 006700 001702
1949 006702 000137 001410
1950
1951
1952
1953 006706 013777 012120 003154
1954 006714 012577 003146
1955 006720 005046
1956 006722 012746 006730
1957 006726 000002
1958 006730
1959 006730 052777 000103 003122
1960
1961 006736 052777 060000 003124
1962 006744 042777 060000 003116
1963 006752 052777 002000 003110
1964 006760 042777 002000 003102
1965 006766 052777 000004 003104
1966
1967 006774 005037 007026
1968 007000 005001
1969 007002 005301
1970 007004 001376
1971 007006 004537 011272
1972 007012 000002
1973 007014 012332
1974 007016 012737 000001 007026
1975 007024 000205
1976 007026 000000
1977
1978
1979
1980
1981 007030 013700 012230
1982 007034 013701 012232
1983 007040 105020
1984 007042 005301
1985 007044 001375
1986 007046 000207
1987

```

```

30$: JSR PC, BUFCLR ;PUT BUFFER BACK TO ZEROS
ENDT1: JSR R5, BUFAD ;GET NEXT BUFFER ADDRESS
        .WORD 402 ;BASE TO BE DIVISIBLE BY 400
        TST R1 ;CHECK FOR OUT OF MEMORY
        BEQ LOOP1 ;BRANCH IF THERE IS MORE
        JMP ENDTST ;OTHERWISE WE'RE DONE WITH TEST
;
; SUBROUTINE TO START THE DX IN SOSIEN MODE
;
DXGO: MOV DXMOO, @DXMOA ;SEND DATA AND OP OUT TO MAINT REG
      MOV (R5)+, @DXBCA ;SET UP BYTE COUNT FROM PASSED PARM.
      CLR -(SP) ;PUT NEW PS ON STACK
      MOV #64$, -(SP) ;PUT NEW PC ON STACK
      RTI ;POP NEW PC AND PS
64$: BIS #103, @DXCSA ;SET FUNCTION TO IBM WRITE
      ;AND ENABLE INTERRUPTS
      BIS #SELO!HLDO, @DXMOA ;FORCE DX SELECTION
      BIC #SELO!HLDO, @DXMOA ;DROP SELECT LINES
      BIS #CMDO, @DXMOA ;RAISE COMMAND OUT
      BIC #CMDO, @DXMOA ;DROP COMMAND OUT
      BIS #SOSIEN, @DXESA ;ENABLE FAST SERVICE OUT/SERV IN
      ;TO UNLEASH NPR'S
      CLR INTERR ;CLEAR NO INTERRUPT OCCURED FLAG
      CLR R1
10$: DEC R1 ;WAIT HERE FOR INTERRUPT
      BNE 10$
      JSR R5, ERRR ;ERROR, INTERRUPT NEVER CAME
      .WORD 2 ;ERROR IDENTIFER
      .WORD ERR2 ;ERROR MESSAGE POINTER
      MOV #1, INTERR ;SET NO INTERRUPT OCCURED FLAG
      RTS R5 ;CONTINUE TESTING
INTERR: .WORD 0 ;SET TO 1 IF DX11 FAILED TO INTERRUPT
;
; SUBROUTINE TO CLEAR THE CURRENT BUFFER
;
BUFCLR: MOV BUFBAS, R0 ;GET BUFFER START
        MOV BUFSIZE, R1 ;GET BUFFER SIZE
10$: CLRB (R0)+ ;CLEAR A BYTE
      DEC R1 ;COUNT IT
      BNE 10$ ;LOOP TO CLEAR BUFFER
      RTS PC ;RETURN TO CALLER

```

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 44  
MODULE 3.1 TEST INITIALIZATION

```

1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999 007050
2000 007050 012746 000340
2001 007054 012746 007062
2002 007060 000002
2003 007062
2004 007062 005037 012216
2005 007066 010537 012222
2006 007072 005037 007026
2007 007076 000205
2008
2009
2010
2011
2012 007100 052777 000010 002772
2013 007106 042777 000200 002744
2014 007114 042777 000006 002736
2015 007122 052777 000001 002730
2016 007130 042777 001230 002722
2017
2018 007136 012777 014000 002716
2019 007144 005737 012304
2020 007150 001403
2021 007152 012777 016000 002702
2022 007160 032777 100000 002706
2023 007166 001402
2024 007170 000005
2025 007172 000742
2026 007174 000207
2027

```

```

.SBTTL MODULE 3.1 TEST INITIALIZATION
MODULE 3.1 TEST SOFTWARE INITIALIZATION

CALLED AS FOLLOWS:
JSR R5,INIT
RETURN COMES HERE

INIT:
MOV #340,-(SP) ;;PUT NEW PS ON STACK
MOV #64$,-(SP) ;;PUT NEW PC ON STACK
RTI ;;POP NEW PC AND PS

64$:
CLR ERR ;RESET ERROR OCCURRED INDICATION
MOV R5,LOCKAD ;SET UP LOCK ON ERROR RETURN ADDR
CLR INTERR ;CLEAR MISSED INTR FLAG
RTS R5

HARDWARE INITIALIZATION SUBROUTINE

DXRES: BIS #10,@DXESA ;DISABLE DX TIME OUT, DEBUG ONLY
BIC #DONE,@DXCSA ;CLEAR DONE AND LOCK0
BIC #6,@DXCSA ;SET FUNCTION TO RESET
BIS #1,@DXCSA ;ISSUE THE DX RESET
BIC #1230,@DXCSA ;CLEAR POTENTIAL DISASTER OF ONLINE OR INTEN
;CLEAR EA BITS ALSO
MOV #SPW,@DXOSA ;SET UP CU OFFSET TO SPW TABLE
TST SPWMVD ;IS SPW TABLE AT HOME BASE?
BEQ 5$ ;BR IF YES
MOV #SPW+2000,@DXOSA ;SET UP ALTERNATE SPW LOCATION
5$: BIT #100000,@DXCBA ;IS LOCK0 ON?
BEQ 10$ ;BR IF NO
RESET ;WE'RE IN TROUBLE, DX DIDN'T RESET
BR DXRES ;GO BACK AND MAKE SURE IT'S CLEARED
10$: RTS PC ;RETURN

```



MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 45  
MODULE 3.0.2 TST2 NPR ADDRESS UNIQUENESS

.SBTTL MODULE 3.0.2 TST2 NPR ADDRESS UNIQUENESS

MODULE 3.0.2 NPR DATA ADDR UNIQUENESS TEST

THIS MODULE TESTS THE DX11'S ABILITY TO UNIQUELY ADDRESS EACH WORD OF MEMORY. THIS TEST IS ACCOMPLISHED BY WRITING A BACKGROUND IN A BLOCK OF MEMORY. A WORD IS THEN TRANSFERRED FROM THE DX TO THE MEMORY BLOCK. THE CPU CHECKS THAT THE WORD ARRIVES AT ITS EXPECTED DESTINATION. IF YES THE WORD IS RESTORED TO MATCH THE BACKGROUND, AND THE NEXT WORD IS TESTED. THIS IS REPEATED FOR EACH WORD IN THE BLOCK AND FOR EACH BLOCK IN MEMORY.

NOTE: THIS TEST MAY NOT DETECT ADDRESSING PROBLEMS WHICH ARE COMMON TO THE CPU AND DX11. I.E. UNIBUS ADDRESSING PROBLEMS.

2028  
2029  
2030  
2031  
2032  
2033  
2034  
2035  
2036  
2037  
2038  
2039  
2040  
2041  
2042  
2043  
2044  
2045

2046 007176 004537 007050  
2047 007202 004737 007100  
2048 007206 004537 010620  
2049 007212 101000  
2050 007214 013700 012160  
2051 007220 012720 007306  
2052 007224 012710 000340  
2053 007230 012737 001000 012232  
2054 007236 004737 007030  
2055 007242 013737 012230 012240  
2056 007250 013737 012242 012244  
2057 007256 053777 012236 002574  
2058 007264 012737 100377 012120  
2059  
2060 007272 013777 012244 002564  
2061 007300 004537 006706  
2062 007304 177776  
2063  
2064  
2065 007306 005737 007026  
2066 007312 001006  
2067 007314 012716 007322  
2068 007320 000002  
2069 007322 012705 007330  
2070 007326 000205  
2071 007330 027727 002704 177777  
2072  
2073 007336 001404  
2074 007340 004537 011272  
2075 007344 000003  
2076 007346 012352  
2077 007350 005077 002664  
2078 007354 042777 000200 002476  
2079 007362 042777 000004 002510  
2080 007370 042777 000030 002462  
2081 007376 062737 000002 012244  
2082 007404 062737 000002 012240  
2083 007412 023737 012240 012234

```

TST2: JSR R5,INIT ;INITIALIZE THE SOFTWARE
      JSR PC,DXRES ;INITIALIZE THE HARDWARE
      JSR R5,BUFAD ;GET STARTING BUFFER ADDRESS
      .WORD 101000 ;PARM MEANS INIT AND SET UP1000 BYTE BUFFER
      MOV AVECT,RO ;GET THE DX VECTOR ADDR
      MOV #INTR2,(RO)+ ;SET UP RETURN ADDR
      MOV #340,(RO) ;SET INTR PSW
LOOP2: MOV #1000,BUFSEZ ;ESTABLISH BUFFER SIZE
      JSR PC,BUFCLR ;CLEAR THE BUFFER
      MOV BUFBAS,DATAD ;INITIALIZE DATA ADDR
      MOV BFBAPH,BFADPH ;GET PHYSICAL BASE ADDR
LOP2: BIS EACUR,@DXCSA ;SET EA BITS IN THE DX11
      MOV #100377,DXMOO ;DATA PATTERN TO BE WRITTEN BY DX

      MOV BFADPH,@DXBAA ;SET UP DX DATA ADDRESS
      JSR R5,DXGO ;START THE DX
      .WORD -2 ;BYTE COUNT = 2

;DX INTERRUPTS TO HERE AFTER NPR'ING A WORD
INTR2: TST INTERR ;DID WE COME HERE VIA DX INTR
      BNE B$ ;BR IS NO, STACK IS OK
      MOV #5$,@SP ;SET UP RETURN ADDR
      RTI ;RELIEVE INTERRUPT STATUS
5$: MOV #8$,R5 ;SET UP RETURN ADDR
      RTS R5 ;RELIEVE SUBROUTINE STATUS
8$: CMP @DATAD,#-1 ;DID DATA GET WHERE IT WAS SUPPOSED TO?
      ;ASSUME CPU CAN ADDRESS MEMORY CORRECTLY
      BEQ 10$ ;BRANCH IF OK
      JSR R5,ERRR ;REPORT THE ERROR
      .WORD 3 ;ERROR NUMBER THREE
      .WORD ERR3 ;ADDR OF ERROR MESSAGE
10$: CLR @DATAD ;RESTORE THE WORD UNDER TEST
      BIC #DONE,@DXCSA ;TURN OFF INTERRUPTS
      BIC #SOSIEN,@DXESA ;CLEAR FAST MODE
      BIC #30,@DXCSA ;CLEAR POSSIBLE CARRY IN EA BITS
      ADD #2,BFADPH ;BUMP THE PHYSICAL ADDRESS
      ADD #2,DATAD ;BUMP DATA POINTER
      CMP DATAD,BUFMAX ;DONE WITH BUFFER?
    
```

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 46  
MODULE 3.0.2 TST2 NPR ADDRESS UNIQUENESS

|      |        |        |        |       |            |                           |
|------|--------|--------|--------|-------|------------|---------------------------|
| 2084 | 007420 | 101716 |        | BLOS  | LOP2       | :BRANCH IF NO             |
| 2085 |        |        |        |       |            |                           |
| 2086 | 007422 | 004737 | 007030 | JSR   | PC, BUFCLR | :RESTORE BUFFER           |
| 2087 | 007426 | 004537 | 010620 | JSR   | R5, BUFAD  | :GET NEXT BUFFER          |
| 2088 | 007432 | 001000 |        | .WORD | 1000       | :SIZE =1000               |
| 2089 | 007434 | 005701 |        | TST   | R1         | :DID WE GET ONE?          |
| 2090 | 007436 | 001674 |        | BEQ   | LOOP2      | :BRANCH IF YES            |
| 2091 | 007440 | 000137 | 001410 | JMP   | ENDTST     | :OTHERWISE TERMINATE TEST |

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 47  
TST3 SPW ADDRESSING

2092  
2093  
2094  
2095  
2096  
2097  
2098  
2099  
2100  
2101  
2102  
2103  
2104  
2105  
2106  
2107  
2108  
2109  
2110  
2111  
2112  
2113  
2114  
2115  
2116  
2117  
2118  
2119  
2120  
2121  
2122  
2123  
2124  
2125  
2126  
2127  
2128  
2129  
2130  
2131  
2132

.SBTTL TST3 SPW ADDRESSING

MODULE 3.0.3 SPW ADDRESSING TEST

THIS MODULE TESTS THE ABILITY OF THE DX TO ACCESS THE STATUS  
POINTER WORD (SPW) TABLE IN ALL POSSIBLE LOCATIONS IN MEMORY.  
A BASE ADDRESS FOR THE SPW IS GENERATED.

THIS SPW TABLE IS FILLED WITH A PATTERN OF 000DEV  
WHERE DEV IS EQUAL TO THE DISPLACEMENT OF THE WORD FROM THE  
SPW BASE; I.E.

SPW BASE =000000  
SPW BASE +1 =000001

..  
..  
..

SPW BASE+255=000377

GIVING A TOTAL OF 256 WORDS FOR THE SPW TABLE WITH IMMEDIATE  
STATUS EQUAL TO THE DEVICE NUMBER.

AN ISS SEQUENCE IS INITIATED WITH A DEVICE # OF 0. IF  
ADRECC AND ADRECD DON'T COME ON FOR THIS DEV, NO TESTING IS DONE,  
THE DEV # IS INCREMENTED, AND ISS IS DONE AGAIN.

IF ADRECC AND ADRECD COME ON IT MEANS THE DX JUMPERS ARE CUT TO  
RESPOND TO THIS DEV NUMBER. THE ISS SEQUENCE IS THEN CONTINUED  
UNTIL THE STATUS IS IN THE DXOS REG. THIS STATUS IS THEN  
CHECKED AND SHOULD BE EQUAL TO THE DEV # IF THE SPW WAS  
ACCESSED CORRECTLY.

THIS PROCEDURE IS REPEATED FOR ALL 256 POSSIBLE DEVICES AT  
ALL POSSIBLE LOCATIONS FOR THE SPW.

THE FIRST RECOGNIZED DEV NO. AND COUNT OF THE FIRST  
GROUP OF RECOGNIZED DEVICES IS SAVED AND PRINTED AT END  
PASS TIME. IF NO DEV IS RECOGNIZED AND ERROR MESSAGE  
IS OUTPUT AT THE END OF THE TEST.

2133 007444 004537 007050  
2134 007450 005037 012264  
2135 007454 005037 012266  
2136 007460 005037 012272  
2137 007464 004737 007100  
2138 007470 012737 000400 012252  
2139 007476 004537 010620  
2140 007502 102000  
2141 007504 012737 002000 012232  
2142 007512 004737 007030  
2143 007516 005037 012250  
2144 007522 013700 012230  
2145 007526 013720 012250  
2146 007532 005237 012250  
2147 007536 022737 000377 012250

TST3: JSR R5,INIT ;INITIALIZE THE SOFTWARE  
CLR RECORD ;SET UP TO RECORD IBM DEV RECOG.  
CLR FRSTD ;SET TO RECORD FIRST DEV RECOGNIZED  
CLR DEVCT ;CLEAR DEVICE COUNTER  
JSR PC,DXRES ;RESET THE DX  
MOV #400,CMD ;INITIALIZE A COMMAND OF ZERO  
JSR R5,BUFAD ;GET THE FIRST BUFFER ADDR  
.WORD 102000 ;SPW TO BE ON 2000 BOUNDARY  
MOV #2000,BUFSE ;SET UP BUFFER SIZE  
JSR PC,BUFCLR ;CLEAR BUFFER  
LOOP3: CLR DEV ;START AT DEVICE 0  
MOV BUFAS,R0 ;GET SPW TABLE BASE  
10\$: MOV DEV,(R0)+ ;MOV IMMED. STATUS TO SPW EQUAL TO DEV  
INC DEV ;INC TO NEXT STATUS = NEXT DEV  
CMP #255.,DEV ;IS DST FULL?

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 48  
TST3 SPW ADDRESSING

```

2148 007544 103370          BHIS  10$      ;DO MORE IF IT IS NOT
2149 007546 005037 012250    CLR   DEV     ;START AT DEV 0
2150 007552 013777 012242 002302 LOP3: MOV   BFBAPH,@DXOSA ;SET UP OFFSET FOR SPW
2151 007560 053777 012236 002272 BIS   EACUR,@DXCSA ;SET UP EXT ADDR BITS
2152 007566 004537 010034 JSR   R5,PTYGEN ;ASSIGN PARITY TO THE DEVICE BYTE
2153 007572 012250          .WORD  DEV     ;ADDR OF DATA TO ASSIGN PARITY TO
2154 007574 113777 012250 002254 MOVB  DEV,@DXCAA ;SET DEV TO CUAR
2155 007602 053777 012250 002260 BIS   DEV,@DXMOA ;PUT DEV ADDR ON BUSOUT
2156 007610 052777 004000 002252 BIS   #ADRO,@DXMOA ;RAISE ADDR OUT
2157 007616 032777 000001 002250 BIT   #1,@DXCBA  ;IS ADRECD ON?
2158 007624 001452          BEQ   20$      ;BR IF ADDR NOT RECOGNIZED
2159 007626 032777 000002 002240 BIT   #2,@DXCBA  ;IS ADRECC ON?
2160 007634 001446          BEQ   20$      ;BR IF ADDR NOT RECOGNIZED
2161 007636 004537 010074 JSR   R5,SAVDEV  ;GO SAVE RECOGNIZED DEV DATA
2162 007642 052777 060000 002220 BIS   #HLDO!SELO,@DXMOA ;RAISE SEL OUT, HOLD OUT
2163 007650 042777 004000 002212 BIC   #ADRO,@DXMOA ;DROP ADRO
2164 007656 043777 012250 002204 BIC   DEV,@DXMOA ;REMOVE DEVICE FROM BUSO
2165 007664 053777 012252 002176 BIS   CMD,@DXMOA ;PUT CMD =0 ON BUSOUT
2166 007672 052777 002000 002170 BIS   #CMDO,@DXMOA ;RAISE CMD OUT
2167 007700 043777 012252 002162 BIC   CMD,@DXMOA ;REMOVE CMD FROM BUSO
2168 007706 042777 002000 002154 BIC   #CMDO,@DXMOA ;DROP CMD OUT
2169 007714 123777 012250 002140 CMPB  DEV,@DXOSA ;IS STATUS EQUAL TO DEV?
2170 007722 001404          BEQ   10$      ;BR IF OK
2171 007724 004537 011272 JSR   R5,ERRR   ;REPORT ERROR
2172 007730 000004          .WORD  4        ;ERROR ID
2173 007732 012411          .WORD  ERR4     ;ERROR MESSAGE ADDR
2174 007734 052777 001000 002126 10$: BIS   #SRVO,@DXMOA ;RAISE SERVICE OUT TO RELIEVE STATUS
2175 007742 042777 001000 002120 BIC   #SRVO,@DXMOA ;DROP SERVICE OUT
2176 007750 000402          BR    30$
2177 007752 004537 010140 20$: JSR   R5,DONTSV ;FC HANDLE RECOGNIZED DEV DATA
2178 007756 004737 007100 30$: JSR   PC,DXRES ;RESTORE DX
2179 007762 005237 012250          INC   DEV     ;GO TO NEXT DEV
2180 007766 105737 012250          TSTB  DEV     ;LAST DEV?
2181 007772 001267          BNE   LOP3    ;BR IF MORE DEVICES
2182 007774 004737 007030 JSR   PC,BUFCLR ;CLEAR OLD BUFFER BEFORE GETTING NEW ONE
2183 010000 004537 010620 JSR   R5,BUFAD  ;GET NEXT BUFFER BASE
2184 010004 002000          .WORD  2000   ;DIVISABLE BY 2000
2185 010006 005701          TST   R1     ;DID WE GET ANOTHER BUFFER?
2186 010010 001642          BEQ   LOOP3   ;BR IF YES AND DO NEXT ONE
2187 010012 005737 012272 TST   DEVCT   ;TEST IS DONE, DID WE EVER FIND A
2188                                ;RECOGNIZED DEVICE?
2189 010016 001004          BNE   40$    ;BR IF YES
2190 010020 004537 011272 JSR   R5,ERRR   ;ERROR, NO ADREDD AND ADRECD FOR ANY
2191                                ;IBM DEVICE NUMBER
2192 010024 000005          .WORD  5
2193 010026 012440          .WORD  ERR5   ;ADDR OF ERROR MSG
2194 010030 000137 001410 40$: JMP   ENDTST ;TERMINATE TEST
2195
2196
2197
2198
2199
2200          ; PARITY ASSIGNMENT SUBROUTINE
2201          ;
2202          ; CALLED AS FOLLOWS:
2203          ; JSR   R5,PTYGEN

```

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 49  
TST3 SPW ADDRESSING

```

2204 : .WORD ADDR ;ADDRESS OF DATA TO WHICH PARITY WILL BE
2205 : ;ASSIGNED
2206 :
2207 : RETURN HERE
2208 :
2209 :
2210 :
2211 : PTYGEN: MOV (R5)+,R3 ;GET ADDR OF DATA
2212 : MOV (R3),R4 ;GET THE DATA
2213 : CLR R2 ;INITIALIZE THE PARTIY SW
2214 : PTY2: ASLB R4 ;CHECK A BIT
2215 : BVC PTY3 ;BR IF ONES
2216 : COM R2 ;OTHERWISE FLIP THE PARITY SW
2217 : PTY3: ASLB R4 ;TWO BITS GET CHECKED AT ONCE
2218 : BNE PTY2 ;IF R4 = 0 WE'D BE DONE
2219 : TST R2 ;DID THE SWITCH END UP SET?
2220 : BMI PTY4 ;BR IF NO
2221 : BIS #400,(R3) ;SET THE BIT IN THE TARGET DATA
2222 : BR PTY5 ;EXIT
2223 : PTY4: BIC #400,(R3) ;CLEAR THE BIT IN THE TARGET DATA
2224 : PTY5: RTS R5 ;RETURN
2225
2226
2227
2228 : SAVDEV: TST RECORD ;ARE WE RECORDING DEVICES FOUND?
2229 : BNE 20$ ;BR IF NO AND EXIT
2230 : TST FRSTD ;IS THIS THE VERY FIRST DEV RECOGNIZED?
2231 : BNE 10$ ;BR IF NO
2232 : BIS #1,FRSTD ;SET FIRST DEV FOUND SW
2233 : MOVB DEV,FRSTDV ;SAVE FIRST DEVICE FOUND
2234 : MOV #1,DEVFND ;SET SWITCH = DEVICE FOUND
2235 : 10$: INC DEVCT ;COUNT THE DEV
2236 : 20$: RTS R5
2237
2238
2239
2240
2241 : DONTSV: TST DEVCT ;HAVE WE FOUND ANY DEV YET?
2242 : BEQ 30$ ;BR IF NO
2243 : BIS #1,RECORD ;TURN OFF DEV RECORDING
2244 : 30$: RTS R5
2245
2246
2247

```

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 50  
MODULE 3.0.4 TST4 DST ACCESS TEST

2248  
2249  
2250  
2251  
2252  
2253  
2254  
2255  
2256  
2257  
2258  
2259  
2260  
2261  
2262  
2263  
2264  
2265  
2266  
2267  
2268  
2269  
2270 010156 004537 007050  
2271 010162 004737 007100  
2272 010166 005737 012302  
2273 010172 001006  
2274 010174 004537 005372  
2275 010200 012535  
2276 010202 000000  
2277 010204 000137 010614  
2278 010210 005037 012250  
2279 010214 113737 012270 012250  
2280 010222 004537 010034  
2281 010226 012250  
2282 010230 004537 010620  
2283 010234 100400  
2284 010236 012737 000400 012232  
2285 010244 004737 007030  
2286 010250 005000  
2287 010252 013701 012230  
2288 010256 110021  
2289 010260 005200  
2290 010262 022700 000377  
2291 010266 103373  
2292 010270 005000  
2293 010272 013700 012250  
2294 010276 042700 177400  
2295 010302 006300  
2296 010304 017701 001552  
2297 010310 042701 001777  
2298 010314 050100  
2299 010316 005737 012172  
2300 010322 001412  
2301 010324 013701 012236  
2302 010330 000301  
2303 010332 006201

```

.SBTTL MODULE 3.0.4 TST4 DST ACCESS TEST

MODULE 3.0.4 DEVICE STATUS TABLE (DST) ACCESS TEST

THIS TEST CHECKS THE ABILITY TO ACCESS ALL BYTES OF THE
DST IN ALL AREAS OF MEMORY. THE OBJECTIVE IS TO CHECK LOADING
OF THE DXBA FROM THE SPW DST BASE, AND COMMAND (CUCR)

A VALID IBM DEVICE FROM TST3 IS CHECKED. THEN A BASE ADDR
IS GENERATED FOR THE DST. EACH BYTE OF THE DST IS THEN LOADED
WITH ITS OFFSET FROM THE DST BASE.

A SPW ENTRY FOR THE VALID IBM DEVICE IS GENERATED WITH AP-
PROPRIATE DST BASE AND ZERO IMMEDIATE STATUS TO ASSURE A DST
ACCESS. AN ISS SEQUENCE IS DONE WITH THE VALID DEV AND
EVERY POSSIBLE IBM COMMAND THEREBY ACCESSING EVERY LOCATION
IN THE DST. AFTER THE ISS, THE STATUS WILL EQUAL THE COMMAND
OR AND ERROR IS FLAGGED.

TST4: JSR R5,INIT ;INITIALIZE THE SOFTWARE
JSR PC,DXRES ;INITIALIZE THE HARDWARE
TST DEVFND ;DO WE HAVE A DEVICE?
BNE 10$ ;BR IF YES
JSR R5,PRINT ;TEST 3 MUST BE RUN SUCCESSFULLY FIRST
.WORD T4MSG ;TO FIND AND IBM DEVICE #
.WORD 0 ;AND ASSURE GOOD SPW ACCESS CAPABILITY
JMP END4 ;ABORT TEST
10$: CLR DEV ;CLEAN UP CURRENT DEV NO.
MOVB FRSTDV,DEV ;GET THE VALID DEV
JSR R5,PTYGEN ;ASSIGN PARITY TO IT
.WORD DEV
JSR R5,BUFAD ;GET A BUFFER OF 256 BYTES
.WORD 100400
MOV #400,BUFSIZE ;SET BUFFER SIZE
JSR PC,BUFCLR ;CLEAR THE BUFFER
LOOP4: CLR R0 ;START WITH CMD = 0
MOV BUFBAS,R1 ;GET THE DST BASE
10$: MOVB R0,(R1)+ ;SET CMD IN DST
INC R0 ;NEXT CMD
CMP #255,,R0 ;ARE ALL STATUS = TO CMD?
BHIS 10$ ;DO 256 BYTES
CLR R0
MOV DEV,R0 ;GET DEVICE
BIC #177400,R0
ASL R0 ;MAKE IT A WORD OFFSET
MOV @DXOSA,R1 ;GET THE SPW BASE
BIC #1777,R1 ;CLEAN IT UP
BIS R1,R0 ;ADD DEV TO OFFSET
TST KT ;MM IN USE?
BEQ 15$ ;BR IF NO
MOV EACUR,R1 ;GET EXT ADDR BITS
SWAB R1 ;POSITION FOR USE IN PAR
ASR R1
    
```

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 51  
MODULE 3.0.4 TST4 DST ACCESS TEST

|      |        |        |        |        |       |       |                   |                                      |
|------|--------|--------|--------|--------|-------|-------|-------------------|--------------------------------------|
| 2304 | 010334 | 010137 | 172352 |        |       | MOV   | R1,@#KIPAR5       | :USE PAR5 TO GET TO SPW TABLE        |
| 2305 | 010340 | 042700 | 040000 |        |       | BIC   | #40000,R0         | :SET SPW ADDR TO USE PAR5            |
| 2306 | 010344 | 052700 | 120000 |        |       | BIS   | #120000,R0        |                                      |
| 2307 | 010350 | 013701 | 012242 |        | 15\$: | MOV   | BFBAPH,R1         | :SET PHY BUFFER BASE                 |
| 2308 | 010354 | 105001 |        |        |       | CLRB  | R1                | :IMMED STATUS = 0                    |
| 2309 | 010356 | 010110 |        |        |       | MOV   | R1,(R0)           | :SET UP SPW                          |
| 2310 | 010360 | 005037 | 012252 |        |       | CLR   | CMD               | :START WITH CMD = 0                  |
| 2311 | 010364 | 004537 | 010034 |        | LOP4: | JSR   | R5,PTYGEN         | :GET COMMAND PARITY                  |
| 2312 | 010370 | 012252 |        |        |       | .WORD | CMD               |                                      |
| 2313 | 010372 | 053777 | 012236 | 001460 |       | BIS   | EACUR,@DXCSA      | :SET UP EXT ADDR BITS                |
| 2314 | 010400 | 113777 | 012250 | 001450 |       | MOVB  | DEV,@DXCAA        | :SET DEVICE TO CUAR                  |
| 2315 | 010406 | 053777 | 012250 | 001454 |       | BIS   | DEV,@DXMOA        | :PUT DEV ADDR ON BUS OUT             |
| 2316 | 010414 | 052777 | 004000 | 001446 |       | BIS   | #ADRO,@DXMOA      | :ADDR OUT                            |
| 2317 | 010422 | 052777 | 060000 | 001440 |       | BIS   | #HLDO!SELO,@DXMOA | :SELECT & HOLD OUT                   |
| 2318 | 010430 | 042777 | 004000 | 001432 |       | BIC   | #ADRO,@DXMOA      | :DROP ADDR OUT                       |
| 2319 | 010436 | 043777 | 012250 | 001424 |       | BIC   | DEV,@DXMOA        | :DROP DEV FROM BUS                   |
| 2320 | 010444 | 053777 | 012252 | 001416 |       | BIS   | CMD,@DXMOA        | :COMMAND TO BUS                      |
| 2321 | 010452 | 052777 | 002000 | 001410 |       | BIS   | #CMD0,@DXMOA      | :COMMAND OUT                         |
| 2322 | 010460 | 043777 | 012252 | 001402 |       | BIC   | CMD,@DXMOA        | :CLR CMD FROM BUS                    |
| 2323 | 010466 | 042777 | 002000 | 001374 |       | BIC   | #CMD0,@DXMOA      | :DROP COMMAND OUT                    |
| 2324 | 010474 | 122737 | 000004 | 012252 |       | CMPB  | #4,CMD            | :IS IT A SENSE COMMAND?              |
| 2325 | 010502 | 001004 |        |        |       | BNE   | 5\$               | :BR IF NO                            |
| 2326 | 010504 | 105777 | 001352 |        |       | TSTB  | @DXOSA            | :STATUS FORCED TO ZERO ON SENSE CMD  |
| 2327 | 010510 | 001411 |        |        |       | BEQ   | 10\$              | :BR IF GOOD                          |
| 2328 | 010512 | 000404 |        |        |       | BR    | 7\$               | :REPORT ERROR                        |
| 2329 | 010514 | 123777 | 012252 | 001340 | 5\$:  | CMPB  | CMD,@DXOSA        | :IS STATUS EQUAL DEVICE?             |
| 2330 | 010522 | 001404 |        |        |       | BEQ   | 10\$              | :BR IF YES                           |
| 2331 | 010524 | 004537 | 011272 |        | 7\$:  | JSR   | R5,ERRR           | :ERROR IN DST ACCESS                 |
| 2332 | 010530 | 000006 |        |        |       | .WORD | 6                 | :ERROR #                             |
| 2333 | 010532 | 012514 |        |        |       | .WORD | ERR6              | :MESSAGE ADDR                        |
| 2334 | 010534 | 052777 | 001000 | 001326 | 10\$: | BIS   | #SRVO,@DXMOA      | :RELIEVE STATUS                      |
| 2335 | 010542 | 042777 | 001000 | 001320 |       | BIC   | #SRVO,@DXMOA      |                                      |
| 2336 | 010550 | 004737 | 007100 |        |       | JSR   | PC,DXRES          | :CLEAR THE DX                        |
| 2337 | 010554 | 005237 | 012252 |        |       | INC   | CMD               | :TRY NEXT COMMAND                    |
| 2338 | 010560 | 105737 | 012252 |        |       | TSTB  | CMD               | :DONE IF BACK TO ZERO                |
| 2339 | 010564 | 001277 |        |        |       | BNE   | LOP4              | :BR TO DO NEXT CMD                   |
| 2340 | 010566 | 004737 | 007030 |        |       | JSR   | PC,BUFCLR         | :CLEAR OLD BUFFER                    |
| 2341 | 010572 | 004537 | 010620 |        | 20\$: | JSR   | R5,BUFAD          | :GET NEXT BUFFER BASE                |
| 2342 | 010576 | 000400 |        |        |       | .WORD | 400               |                                      |
| 2343 | 010600 | 005701 |        |        |       | TST   | R1                | :DID WE GET A BUFFER?                |
| 2344 | 010602 | 001004 |        |        |       | BNE   | END4              | :BR IF NO,END TEST                   |
| 2345 | 010604 | 005737 | 012242 |        |       | TST   | BFBAPH            | :IS BUFFER BASE 0'S?                 |
| 2346 | 010610 | 001770 |        |        |       | BEQ   | 20\$              | :BR IF YES, THE DX DOES NOT DO A     |
| 2347 |        |        |        |        |       |       |                   | :DST FETCH IF SPW ENTRY IS ALL ZEROS |
| 2348 | 010612 | 000616 |        |        |       | BR    | LOOP4             | :DO NEXT BUFFER                      |
| 2349 | 010614 | 000137 | 001410 |        | END4: | JMP   | ENDTST            | :END OF TEST 4                       |



MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 52  
BASE ADDRESS GENERATOR MODULE

```

2350 .SBTTL BASE ADDRESS GENERATOR MODULE
2351 MODULE 3.2 BASE ADDRESS GENERATOR
2352
2353 THIS MODULE SETS UP THE STARTING ADDRESS FOR DX BUFFERS
2354 AND HAS RESPONSIBILITY FOR ESTABLISHING EXTENDED ADDR
2355 BITS, BUFFER BASE, BUFFER SIZE, AND HAS CONTROL OF
2356 THE MEMORY MGT. REGISTERS. KIPAR6 IS ALWAYS USED
2357 TO ACCESS THE DXBUFFERS IF MEM MGT. IS ENABLED.
2358
2359 THIS MODULE IS CALLED AS FOLLOWS:
2360
2361 JSR R5, BUFAD
2362 .WORD PARM
2363 RETURN HERE WITH R1=0 MEANS BUFFER ALLOCATED
2364 R1=1 MEANS NO MEMORY LEFT
2365 PARM IS DEFINED AS FOLLOWS:
2366 BIT 15=1 INITIALIZE TO FIRST BUFFER LOCATION
2367 BIT 15=0 GET NEXT BUFFER LOCATION
2368 BITS 14:0 BASE ADDR OF BUFFER MUST BE DIVISIBLE BY
2369 THIS NUMBER
2370 BUFAD: MOV (R5)+, R0 ;GET THE PASSED PARAMETER
2371 CLR R1 ;RESET 'OUT OF MEMORY' INDICATOR
2372 TST R0 ;CHECK FOR INITIALIZE SW
2373 BPL NEXTB ;BRANCH IF NOT TO INITIALIZE
2374 CLR EACUR ;START INIT PROCESS BY CLEARING
2375 ;THE EA BITS
2376 MOV #ENDALL, R2 ;GET VIRTUAL END OF PROGRAM ADDR
2377 BIC #100000, R0 ;CLEAR HIGH BIT OF PARAM
2378 ADD R0, R2 ;OFFSET ABOVE PROGRAM
2379 SUB #1, R0 ;DEVELOP A MASK
2380 BIC R0, R2 ;TO MAKE BASE DIVISIBLE BY PROPER AMT.
2381 MOV R2, BUFBAS ;SAVE VIRTUAL BUFFER BASE
2382 MOV R2, BFBAPH ;SAVE PHYSICAL BUFFER BASE, SAME AS VIRT
2383 ;AT THIS POINT.
2384 TST KT ;IS KT IN USE?
2385 BEQ ENDB ;BRANCH IF NO TO END OF MODULE
2386 MOV #6, R3 ;INIT LOOP COUNTER
2387 CLR @KIPAR6 ;START TO INITIALIZE KIPAR6
2388 ASR R2, #6 ;SHIFT ADDRESS RIGHT 6 PLACES
2389 SOB R3, 10$ ;TO ALLIGN WITH PAR
2390 MOV R2, @KIPAR6 ;SET UP PAR6 TO REFERENCE FIRST BUFFER
2391 BIC #177700, BUFBAS ;CLEAR UP VIRTUAL BUFFER ADDR
2392 BIS #140000, BUFBAS ;BUFFER ACCESS IS ALWAYS THRU PAR6
2393 MOV #1, @SR0 ;TURN ON MM
2394 ADD BUFBAS, R0 ;FIND TOP OF CURRENT BUFFER
2395 INC R0 ;
2396 MOV R0, BUFMAX ;SAVE VIRT END OF BUFFER
2397 BR ENDB1 ;DONE WITH INITIALIZATION
2398
2399
2400 COME HERE IF NEXT BUFFER ADDR IS WANTED
2401
2402 NEXTB: ADD R0, BUFBAS ;INCR TO NEXT BUFFER SPACE
2403 BIC #77, BUFBAS ;CLEAR POSSIBLE LOW ORDER BITS
2404 ADD R0, BFBAPH ;STEP PHYSICAL ADDR
2405 BIC #77, BFBAPH ;CLEAR POSSIBLE LOW ORDER BITS

```

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 53  
BASE ADDRESS GENERATOR MODULE

|      |        |        |        |        |            |                  |  |   |
|------|--------|--------|--------|--------|------------|------------------|--|---|
| 2406 | 010774 | 103007 |        |        | BCC        | 2\$              |  | :BR IF NO CARRY                             |
| 2407 | 010776 | 062737 | 000010 | 012236 | ADD        | #10,EACUR        |  | :STEP EXTENDED ADDR BITS                    |
| 2408 | 011004 | 023737 | 012236 | 012224 | CMP        | EACUR,EABITS     |  | :HAVE THE EXT ADR BITS GONE TOO HIGH?       |
| 2409 | 011012 | 101125 |        |        | BHI        | 30\$             |  | :BR IF YES TO SET ALL DONE SW AND EXIT      |
| 2410 | 011014 | 005737 | 012172 |        | 2\$: TST   | KT               |  | :IS MEM MNG IN USE?                         |
| 2411 | 011020 | 001511 |        |        | BEQ        | 20\$             |  | :BRANCH IF NO                               |
| 2412 | 011022 | 032737 | 020000 | 012230 | BIT        | #20000,BUFBAS    |  | :WAS THERE A CARRY TO ACCESS NEXT PAR?      |
| 2413 | 011030 | 001417 |        |        | BEQ        | 10\$             |  | :BR IF NO CARRY                             |
| 2414 | 011032 | 062737 | 000200 | 172354 | ADD        | #200,@#KIPAR6    |  | :TAKE A 4K STEP                             |
| 2415 | 011040 | 023737 | 172354 | 012170 | CMP        | @#KIPAR6,EASIZE  |  | :CHECK FOR TOO HIGH                         |
| 2416 | 011046 | 103402 |        |        | BLO        | 5\$              |  | :BRANCH IF OK                               |
| 2417 | 011050 | 005201 |        |        | INC        | R1               |  | :SET THE "ALL DONE" SWITCH                  |
| 2418 | 011052 | 000506 |        |        | BR         | ENDB1            |  | :EXIT                                       |
| 2419 | 011054 | 042737 | 177700 | 012230 | 5\$: BIC   | #177700,BUFBAS   |  | :START BUFFER BASE OVER                     |
| 2420 | 011062 | 052737 | 140000 | 012230 | BIS        | #140000,BUFBAS   |  | :ACCESS THROUGH PAR6                        |
| 2421 | 011070 | 013703 | 012230 |        | 10\$: MOV  | BUFBAS,R3        |  | :GET THE BASE ADDR                          |
| 2422 | 011074 | 060003 |        |        | ADD        | R0,R3            |  | :CALCULATE TOP OF BUFFER                    |
| 2423 | 011076 | 032703 | 020000 |        | BIT        | #20000,R3        |  | :DIT IT CARRY ACROSS A PAR BOUNDARY         |
| 2424 | 011102 | 001322 |        |        | BNE        | NEXTB            |  | :BR IS YES. GO BACK AND GET A BETTER ADDR   |
| 2425 | 011104 | 010337 | 012234 |        | MOV        | R3,BUFMAX        |  | :SAVE TOP OF BUFFER ADDR                    |
| 2426 | 011110 | 013703 | 012242 |        | MOV        | BFBAPH,R3        |  | :GET PHYSICAL BUF BASE                      |
| 2427 | 011114 | 060003 |        |        | ADD        | R0,R3            |  | :ADD BUFFER SIZE TO GET BUF TOP             |
| 2428 | 011116 | 023737 | 012236 | 012224 | CMP        | EACUR,EABITS     |  | :ARE WE IN THE TOP PHYSICAL BANK?           |
| 2429 | 011124 | 001006 |        |        | BNE        | 12\$             |  | :BR IF NO                                   |
| 2430 | 011126 | 020337 | 012226 |        | CMP        | R3,PHYMAX        |  | :ARE WE OVER THE TOP OF MEM?                |
| 2431 | 011132 | 103055 |        |        | BHIS       | 30\$             |  | :BR IF YES, SET DONE SW                     |
| 2432 | 011134 | 020327 | 000004 |        | CMP        | R3,#4            |  | :WAS THERE A CARRY OUT OF TOP BANK?         |
| 2433 | 011140 | 101452 |        |        | BLOS       | 30\$             |  | :BR IF YES, WE MUST BE DONE                 |
| 2434 | 011142 | 013703 | 012242 |        | 12\$: MOV  | BFBAPH,R3        |  | :GET PHYSICAL BUFFER BASE                   |
| 2435 | 011146 | 060003 |        |        | ADD        | R0,R3            |  | :FIND TOP OF PHY BUFFER                     |
| 2436 | 011150 | 103677 |        |        | BCC        | NEXTB            |  | :BR IF CARRY OVER 32K BNDRY                 |
| 2437 |        |        |        |        |            |                  |  | :AND GO BACK AND GET A BUFFER WHICH DOESN'T |
| 2438 |        |        |        |        |            |                  |  | :STRADDEL A 32K BOUNDARY                    |
| 2439 | 011152 | 005737 | 012236 |        | TST        | EACUR            |  | :ARE WE IN THE FIRST BANK?                  |
| 2440 | 011156 | 001010 |        |        | BNE        | 3\$              |  | :BR IF NO                                   |
| 2441 | 011160 | 020337 | 012164 |        | CMP        | R3,MSIZE         |  | :ARE WE IN THE LOADER AREA?                 |
| 2442 | 011164 | 101405 |        |        | BLOS       | 3\$              |  | :BR IF OK                                   |
| 2443 | 011166 | 023737 | 012242 | 012166 | CMP        | BFBAPH,MMAX      |  | :IS BUFFER ABOVE LOADERS?                   |
| 2444 | 011174 | 103001 |        |        | BHIS       | 3\$              |  | :BRANCH IF WE'RE OK                         |
| 2445 | 011176 | 000664 |        |        | BR         | NEXTB            |  | :GO BACK AND GET A BETTER BUFFER ADDR       |
| 2446 | 011200 | 012777 | 014000 | 000654 | 3\$: MOV   | #SPW,@DXOSA      |  | :SET UP SPW ADDR                            |
| 2447 | 011206 | 005037 | 012304 |        | CLR        | SPWMVD           |  | :CLEAR SPW BASE ADDR MOVED SWITCH           |
| 2448 | 011212 | 020327 | 014000 |        | CMP        | R3,#SPW          |  | :IS TOP OF BUFFER IN SPW?                   |
| 2449 | 011216 | 101411 |        |        | BLOS       | 15\$             |  | :BR IF OK                                   |
| 2450 | 011220 | 023727 | 012242 | 016000 | CMP        | BFBAPH,#SPW+2000 |  | :IS BUFFER BASE IN TT?                      |
| 2451 | 011226 | 103005 |        |        | BHIS       | 15\$             |  | :BR IF NO                                   |
| 2452 | 011230 | 012777 | 016000 | 000624 | MOV        | #SPW+2000,@DXOSA |  | :MOV THE SPW                                |
| 2453 | 011236 | 005237 | 012304 |        | INC        | SPWMVD           |  | :SET SPW ADDR MOVED TO ALTERNATE SWITCH     |
| 2454 | 011242 | 000412 |        |        | 15\$: BR   | ENDB1            |  | :GO TO END OF MODULE                        |
| 2455 | 011244 | 013737 | 012230 | 012234 | 20\$: MOV  | BUFBAS,BUFMAX    |  | :START TO GET BUFMAXIMUM                    |
| 2456 | 011252 | 060037 | 012234 |        | ADD        | R0,BUFMAX        |  | :ADD IN BUFFER SIZE                         |
| 2457 | 011256 | 023737 | 012234 | 012164 | CMP        | BUFMAX,MSIZE     |  | :SEE IF WE'RE OUT OF MEMORY                 |
| 2458 | 011264 | 101401 |        |        | BLOS       | ENDB1            |  | :BR IF OK                                   |
| 2459 | 011266 | 005201 |        |        | 30\$: INC  | R1               |  | :SET OUT OF MEMORY SWITCH                   |
| 2460 | 011270 | 000205 |        |        | ENDB1: RTS | R5               |  | :RETURN FROM MODULE                         |

```

2461 .SBTTL MODULE 3.3 REPORT ERRORS
2462 MODULE 3.3 ERROR REPORTING
2463
2464 CALLED AS FOLLOWS:
2465
2466 JSR R5,ERRR
2467 .WORD X
2468 .WORD Y
2469 RETURN (UNLESS LOCK ON ERROR FUNCTION OVERRIDES RETURN)
2470 X EQUALS ERROR IDENTIFICATION NUMBER
2471 Y EQUALS ADDR OF UNIQUE ERROR MESSAGE
2472
2473 THIS MODULE FORMATS AND PRINTS ERROR MESSAGES AND HANDLES
2474 ERROR OPTIONS SUCH AS INHIBIT ERROR MESSAGES, SHORT ERROR
2475 REPORT, SCOPE LOOP, HALT ON ERROR, ETC.
2476
2477
2478 011272 032777 100000 167640 ERRR: BIT #SW15,@SWR ;HALT ON ERROR ON?
2479 011300 001401 BEQ 5$ ;BR IF NO
2480 011302 000000 HALT ;SW15 ON AND ERROR OCCURRED.
2481
2482 011304 013737 012206 012216 5$: MOV CTESTN,ERR ;SET ERROR DETECTED SW
2483 011312 005046 CLR -(SP) ;:PUT NEW PS ON STACK
2484 011314 012746 011322 MOV #64$,-(SP) ;:PUT NEW PC ON STACK
2485 011320 000002 RTI ;:POP NEW PC AND PS
2486 011322 64$:
2487 011322 005737 012220 TST LOCK ;ARE WE LOCKED ON ERROR?
2488 011326 001407 BEQ 10$ ;BR IF NO
2489 011330 032777 040000 167602 BIT #SW14,@SWR ;SCOPE LOOP SW ON?
2490 011336 001413 BEQ 15$ ;BR IF NO
2491 011340 013705 012222 MOV LOCKAD,R5 ;GET THE LOOP ADDR
2492 011344 000205 RTS R5 ;GO TO IT
2493 011346 032777 040000 167564 10$: BIT #SW14,@SWR ;SCOPE LOOP ON?
2494 011354 001404 BEQ 15$ ;BR IF NO
2495 011356 013737 012206 012220 MOV CTESTN,LOCK ;SET LOCKED ON ERR SW
2496 011364 000402 BR 20$
2497 011366 005037 012220 15$: CLR LOCK ;TURN OF LOCKED SW
2498 011372 010537 012254 20$: MOV R5,ERRPC ;SAVE ERR PC
2499 011376 012537 012256 MOV (R5)+,ERRID ;SAVE ERROR ID
2500 011402 012537 012260 MOV (R5)+,ERRMS ;SAVE ERROR MESSAGE ADDR
2501 011406 032777 020000 167524 BIT #SW13,@SWR ;INHIBIT PRINT ON?
2502 011414 001402 BEQ 25$ ;BR IF NO
2503 011416 000137 012040 JMP 97$ ;TERMINATE MODULE
2504 011422 004537 005372 25$: JSR R5,PRINT ;PRINT ERRPC
2505 011426 013307 .WORD ERPC ;ADDR OF MESSAGE
2506 011430 000000 .WORD 0 ;NO SPECIAL PRINT CONTROLS
2507 011432 162737 000004 012254 SUB #4,ERRPC ;BACKUP ADDR TO CORRECT VALUE
2508 011440 004537 005372 JSR R5,PRINT ;PRINT THE PC NUMBER ITSELF
2509 011444 012254 .WORD ERRPC
2510 011446 100001 .WORD 100001 ;NO CRLF, CONVERT TO ASCII
2511 011450 004537 005372 JSR R5,PRINT ;PRINT ERROR NUMBER ON SAME LINE
2512 011454 013322 .WORD ERNO
2513 011456 000001 .WORD 1 ;INHIBIT CRLF
2514 011460 004537 005372 JSR R5,PRINT ;PRINT THE OCT
2515 011464 012256 .WORD ERRID
2516 011466 100001 .WORD 100001

```

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 55  
MODULE 3.3 REPORT ERRORS

|      |        |        |        |        |       |       |             |  |
|------|--------|--------|--------|--------|-------|-------|-------------|--|
| 2517 | 011470 | 013737 | 012260 | 011502 |       | MOV   | ERRMS,30\$  | :GET MESSAGE POINTER                     |
| 2518 | 011476 | 004537 | 005372 |        |       | JSR   | R5,PRINT    | :PRINT THE UNIQUE MESSAGE                |
| 2519 | 011502 | 000000 |        |        | 30\$: | .WORD | 0           | :FILLED FROM TWO LINES UP                |
| 2520 | 011504 | 000000 |        |        |       | .WORD | 0           | :NO SPECIAL CONTROLS                     |
| 2521 | 011506 | 032777 | 010000 | 167424 |       | BIT   | #SW12,@SWR  | :SHORT ERROR REPORT ON?                  |
| 2522 | 011514 | 001145 |        |        |       | BNE   | 95\$        | :BR IF YES, WE'RE DONE                   |
| 2523 | 011516 | 004537 | 005372 |        |       | JSR   | R5,PRINT    | :PRINT TEST NO.                          |
| 2524 | 011522 | 013336 |        |        |       | .WORD | ERTN        |  |
| 2525 | 011524 | 000000 |        |        |       | .WORD | 0           |  |
| 2526 | 011526 | 004537 | 005372 |        |       | JSR   | R5,PRINT    | :PRINT THE OCTAL TEST NO.                |
| 2527 | 011532 | 012206 |        |        |       | .WORD | CTESTN      |  |
| 2528 | 011534 | 100001 |        |        |       | .WORD | 100001      |  |
| 2529 | 011536 | 022737 | 000005 | 012256 |       | CMP   | #5,ERRID    | :IS IT ERROR #5?                         |
| 2530 | 011544 | 001531 |        |        |       | BEQ   | 95\$        | :BR IF YES, REST OF STUFF IS MEANINGLESS |
| 2531 | 011546 | 004537 | 005372 |        |       | JSR   | R5,PRINT    | :PRINT VIRT ADDR                         |
| 2532 | 011552 | 013351 |        |        |       | .WORD | ERADR       |  |
| 2533 | 011554 | 000000 |        |        |       | .WORD | 0           |  |
| 2534 | 011556 | 004537 | 005372 |        |       | JSR   | R5,PRINT    | :PRINT VIRT ADDR OCTAL                   |
| 2535 | 011562 | 012230 |        |        |       | .WORD | BUFBAS      |  |
| 2536 | 011564 | 100001 |        |        |       | .WORD | 100001      |  |
| 2537 | 011566 | 005737 | 012172 |        |       | TST   | KT          | :IS MEM MGT IN USE?                      |
| 2538 | 011572 | 001431 |        |        |       | BEQ   | 90\$        | :BR IF NO WE'RE DONE                     |
| 2539 | 011574 | 004537 | 005372 |        |       | JSR   | R5,PRINT    | :PRINT PHYSICAL ADDR                     |
| 2540 | 011600 | 013404 |        |        |       | .WORD | ERADPH      |  |
| 2541 | 011602 | 000000 |        |        |       | .WORD | 0           |  |
| 2542 | 011604 | 004537 | 005372 |        |       | JSR   | R5,PRINT    | :PRINT PHYSICAL ADDR OCTAL               |
| 2543 | 011610 | 012242 |        |        |       | .WORD | BFBAPH      |  |
| 2544 | 011612 | 100001 |        |        |       | .WORD | 100001      |  |
| 2545 | 011614 | 004537 | 005372 |        |       | JSR   | R5,PRINT    | :PRINT EA BITS                           |
| 2546 | 011620 | 013431 |        |        |       | .WORD | EABIT       |  |
| 2547 | 011622 | 000000 |        |        |       | .WORD | 0           |  |
| 2548 | 011624 | 013737 | 012236 | 012262 |       | MOV   | EACUR,EAERR | :GET THE EA BITS                         |
| 2549 | 011632 | 006237 | 012262 |        |       | ASR   | EAERR       | :POSITION BITS                           |
| 2550 | 011636 | 006237 | 012262 |        |       | ASR   | EAERR       | : TO BITS 1:0                            |
| 2551 | 011642 | 006237 | 012262 |        |       | ASR   | EAERR       |  |
| 2552 | 011646 | 004537 | 005372 |        |       | JSR   | R5,PRINT    | :PRINT THE OCTAL EA BITS                 |
| 2553 | 011652 | 012262 |        |        |       | .WORD | EAERR       |  |
| 2554 | 011654 | 100001 |        |        |       | .WORD | 100001      |  |
| 2555 | 011656 | 023727 | 012256 | 000003 | 90\$: | CMP   | ERRID,#3    | :IS IT ERROR NO. 3?                      |
| 2556 | 011664 | 001010 |        |        |       | BNE   | 93\$        | :BR IF NO                                |
| 2557 | 011666 | 004537 | 005372 |        |       | JSR   | R5,PRINT    | :PRINT DATA ADDRESS ASCII                |
| 2558 | 011672 | 013452 |        |        |       | .WORD | DATAA       |  |
| 2559 | 011674 | 000000 |        |        |       | .WORD | 0           |  |
| 2560 | 011676 | 004537 | 005372 |        |       | JSR   | R5,PRINT    | :PRINT DATA ADDR OCT                     |
| 2561 | 011702 | 012240 |        |        |       | .WORD | DATAD       |  |
| 2562 | 011704 | 100001 |        |        |       | .WORD | 100001      |  |
| 2563 | 011706 | 023727 | 012256 | 000004 | 93\$: | CMP   | ERRID,#4    | :IS IT ERROR #4?                         |
| 2564 | 011714 | 001004 |        |        |       | BNE   | 94\$        | :BR IF NO                                |
| 2565 | 011716 | 113737 | 012250 | 012300 |       | MOVB  | DEV,EXP     | :GET DEV #                               |
| 2566 | 011724 | 000416 |        |        |       | BR    | 96\$        |  |
| 2567 | 011726 | 023727 | 012256 | 000006 | 94\$: | CMP   | ERRID,#6    | :IS IT ERROR #6?                         |
| 2568 | 011734 | 001035 |        |        |       | BNE   | 95\$        | :BR IF NO                                |
| 2569 | 011736 | 122737 | 000004 | 012252 |       | CMPB  | #4,CMD      | :WAS IT A SENSE CMD?                     |
| 2570 | 011744 | 001003 |        |        |       | BNE   | 98\$        | :BR IF NO                                |
| 2571 | 011746 | 005037 | 012300 |        |       | CLR   | EXP         | :EXPECTED STATUS IS ZERO FOR SENSE       |
| 2572 | 011752 | 000403 |        |        |       | BR    | 96\$        |  |

MD-11-CZDXJ-C DX11-B ADDRESSING TEST MACY11 30A(1052) 07-JUN-82 16:29 PAGE 56  
 CZDXJC.P11 07-JUN-82 16:26 MODULE 3.3 REPORT ERRORS

|      |        |        |        |        |       |       |              |                        |
|------|--------|--------|--------|--------|-------|-------|--------------|------------------------|
| 2573 | 011754 | 113737 | 012252 | 012300 | 98\$: | MOVB  | CMD,EXP      | :GET EXPECTED CMD      |
| 2574 | 011762 | 117737 | 000074 | 012276 | 96\$: | MOVB  | @DXOSA,ACTU  | :GET ACTUAL STATUS     |
| 2575 | 011770 | 004537 | 005372 |        |       | JSR   | R5,PRINT     | :PRINT ACTUAL MESSAGE  |
| 2576 | 011774 | 013575 |        |        |       | .WORD | ACTUMS       |                        |
| 2577 | 011776 | 000000 |        |        |       | .WORD | 0            |                        |
| 2578 | 012000 | 004537 | 005372 |        |       | JSR   | R5,PRINT     | :PRINT STATUS OCTAL    |
| 2579 | 012004 | 012276 |        |        |       | .WORD | ACTU         |                        |
| 2580 | 012006 | 100001 |        |        |       | .WORD | 100001       |                        |
| 2581 | 012010 | 004537 | 005372 |        |       | JSR   | R5,PRINT     | :PRINT EXPECTED STATUS |
| 2582 | 012014 | 013607 |        |        |       | .WORD | EXPMS        |                        |
| 2583 | 012016 | 000001 |        |        |       | .WORD | 1            |                        |
| 2584 | 012020 | 004537 | 005372 |        |       | JSR   | R5,PRINT     | :PRINT EXPECTED OCTAL  |
| 2585 | 012024 | 012300 |        |        |       | .WORD | EXP          |                        |
| 2586 | 012026 | 100001 |        |        |       | .WORD | 100001       |                        |
| 2587 | 012030 | 004537 | 005372 |        | 95\$: | JSR   | R5,PRINT     | :SPACE THE PAPER       |
| 2588 | 012034 | 013625 |        |        |       | .WORD | CRLF1        |                        |
| 2589 | 012036 | 000000 |        |        |       | .WORD | 0            |                        |
| 2590 | 012040 | 022737 | 001566 | 000042 | 97\$: | CMP   | #SENDAD,@#42 | :UNDER ACT?            |
| 2591 | 012046 | 001001 |        |        |       | BNE   | 99\$         | :BR IF NO              |
| 2592 | 012050 | 000000 |        |        |       | HALT  |              | :HALT ON ACT ERROR     |
| 2593 |        |        |        |        |       |       |              |                        |
| 2594 | 012052 | 000205 |        |        | 99\$: | RTS   | R5           | :RETURN FROM MODULE    |

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 57  
MODULE 3.3 REPORT ERRORS

2595  
 2596  
 2597  
 2598  
 2599  
 2600 012054 000000  
 2601 012056 000000  
 2602 012060 000000  
 2603 012062 000000  
 2604 012064 000000  
 2605 012066 000000  
 2606 012070 000000  
 2607 012072 000000  
 2608 012074 000000  
 2609 012076 000000  
 2610 012100 000000  
 2611 012102 000000  
 2612 012104 000000  
 2613  
 2614 012106 000000  
 2615 012110 000000  
 2616 012112 000000  
 2617 012114 000000  
 2618 012116 000000  
 2619 012120 000000  
 2620 012122 000000  
 2621

.....

DX ACCESS TABLES

DXDSA: .WORD 0 ;DEVICE STATUS REG ADDR.  
 DXCAA: .WORD 0 ;COMMAND AND ADDR REG ADDR.  
 DXCSA: .WORD 0 ;CONTROL UNIT STATUS REG ADDR  
 DXOSA: .WORD 0 ;OFFSET AND STATUS REG ADDR.  
 DXBAA: .WORD 0 ;BUS ADDRESS REG ADDR  
 DXBCA: .WORD 0 ;BYTE COUNT REG ADDR  
 DXMOA: .WORD 0 ;MAINTENANCE OUT REG ADDR.  
 DXMIA: .WORD 0 ;MAINTENANCE IN REG ADDR  
 DXCBA: .WORD 0 ;CONTROL BITS REG ADDR  
 DXNDA: .WORD 0 ;NPR DATA REG ADDR  
 DXESA: .WORD 0 ;EXTRA SIGNALS REG ADDR  
 DXMOBA: .WORD 0 ;MAINTENANCE OUT BUFFERED REG ADDR  
 DXES1A: .WORD 0 ;EXTRA EXTRA SIGNALS REG ADDR.  
 : REGS TO BE OUTPUT TO DX  
 DXDSO: .WORD 0 ;DEVICE STATUS REG OUT  
 DXCAO: .WORD 0 ;COMMAND AND ADDR REG OUT  
 DXCSO: .WORD 0 ;CONTROL UNIT STATUS REG OUT  
 DXOSO: .WORD 0 ;OFFSET AND STATUS REG OUT  
 DXBAO: .WORD 0 ;BUS ADDR REG. OUT  
 DXMOO: .WORD 0 ;MAINT OUT REG ADDR  
 DXBCO: .WORD 0

MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 58  
MODULE 3.3 REPORT ERRORS

2622  
2623  
2624  
2625 012124  
2626 012124 000001  
2627 012126 176200  
2628 012130 000300  
2629 012132 000004  
2630

:  
:  
:  
: DEFAULT PARAMETER TABLE  
DPAR:  
DTESTN: .WORD 1 ; STARTING TEST NUMBER  
DUBA: .WORD 176200 ; DEFAULT UNIBUS BASE ADDR  
DVECT: .WORD 300 ; DEFAULT VECTOR ADDR  
DPRIOR: .WORD 4 ; DEFAULT DX PRIORITY LEVEL



MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 59  
MODULE 3.3 REPORT ERRORS

2631  
2632  
2633  
2634  
2635  
2636 012134  
2637 012134 000001  
2638 012136 000004  
2639 012140 160000  
2640 012142 177756  
2641 012144 000040  
2642 012146 000776  
2643 012150 000000  
2644 012152 000006

INPUT PARAMETER LIMITS TABLE

LPAR: ;LOWEST POSSIBLE TEST NUMBER  
 LTESTN: 1 ;HIGHEST POSSIBLE TEST NUMBER  
 HTESTN: 4 ;LOWEST POSSIBLE UNIBUS ADDRESS  
 LUBA: 160000 ;HIGHEST POSSIBLE UNIBUS ADDRESS  
 HUBA: 177756 ;LOWEST POSSIBLE DX VECTOR ADDRESS  
 LVEC: 40 ;HIGHEST POSSIBLE DX VECTOR ADDRESS  
 HVEC: 776 ;LOWEST POSSIBLE FOR LEVEL  
 LPRIOR: 0 ;HIGHEST POSSIBLE FOR LEVEL  
 HPRIOR: 6

ACTIVE PARAMETER TABLE

THIS TABLE CONTAINS CONFIGURATION PARAMETER CURRENTLY  
IN USE.

|      |        |        |               |        |   |
|------|--------|--------|---------------|--------|---|
| 2645 |        |        |               |        |   |
| 2646 |        |        |               |        |   |
| 2647 |        |        |               |        |   |
| 2648 |        |        |               |        |   |
| 2649 |        |        |               |        |   |
| 2650 |        |        |               |        |   |
| 2651 | 012154 |        | APAR:         |        |   |
| 2652 | 012154 | 000000 | ATESTN: .WORD | 0      | : STARTING TEST NO.                                       |
| 2653 | 012156 | 000000 | AUBA: .WORD   | 0      | : DX UNIBUS BASE ADDR.                                    |
| 2654 | 012160 | 000000 | AVECT: .WORD  | 0      | : DX VECTOR ADDRESS                                       |
| 2655 | 012162 | 000000 | APRIOR: .WORD | 0      | : DX PRIORITY LEVEL                                       |
| 2656 | 012164 | 000000 | MSIZE: .WORD  | 0      | : ADDR OF LAST WORD OF MEM, NOT INCLUDING LOADERS, KT OFF |
| 2657 | 012166 | 000000 | MMAX: .WORD   | 0      | : LAST WORD OF MEM, KT OFF                                |
| 2658 | 012170 | 000000 | EASIZE: .WORD | 0      | : LOWEST INVALID SETTING OF PAR                           |
| 2659 | 012172 | 000000 | KT: .WORD     | 0      | : =0 NO KT IN USE =1 KT IN USE                            |
| 2660 | 012174 | 000000 | FIRSTP: .WORD | 0      | : 0 IF FIRST PASS, 1 IF NOT FIRST PASS                    |
| 2661 | 012176 | 177564 | TPS: .WORD    | 177564 | : TTY PRINTER STATUS REG ADDR                             |
| 2662 | 012200 | 177566 | TPB: .WORD    | 177566 | : TTY PRINTER BUFFER REG ADDR                             |
| 2663 | 012202 | 177560 | TKS: .WORD    | 177560 | : TTY KEYBOARD STATUS REG ADDR                            |
| 2664 | 012204 | 177562 | TDB: .WORD    | 177562 | : TTY KEYBOARD BUFFER.                                    |
| 2665 | 012206 | 000000 | CTESTN: .WORD | 0      | : CURRENT TEST NUMBER                                     |
| 2666 | 012210 | 000000 | PARMV: .WORD  | 0      | : 0 IF PARMS NOT VALID, 1 IF THEY ARE VALID               |
| 2667 | 012212 | 000001 | COUNT: .WORD  | 1      | : ITERATION COUNT CONSTANT                                |
| 2668 | 012214 | 000000 | CNT: .WORD    | 0      | : ITERATION VARIABLE                                      |
| 2669 | 012216 | 000000 | ERR: .WORD    | 0      | : IF ERROR DETECTED, CONTAINS TN                          |
| 2670 | 012220 | 000000 | LOCK: .WORD   | 0      | : SW=TN IF LOOP SW 14 ERR DETECTED                        |
| 2671 | 012222 | 000000 | LOCKAD: .WORD | 0      | : LOCK ON ERROR ADDR                                      |
| 2672 | 012224 | 000000 | EABITS: .WORD | 0      | : HIGHEST VALID SETTING OF ADDR 17 16                     |
| 2673 | 012226 | 000000 | PHYMAX: .WORD | 0      | : LOWEST INVALID PHYSICAL ADDR, LOW 16 BITS               |
| 2674 | 012230 | 000000 | BUFBAS: .WORD | 0      | : START OF CURRENT WORKING BUFFER, VIRTUAL                |
| 2675 | 012232 | 000000 | BUFSZE: .WORD | 0      | : SIZE OF CURRENT BUFFER IN OCT BYTES                     |
| 2676 | 012234 | 000000 | BUFMAX: .WORD | 0      | : MAX VIRTUAL ADDR FOR THIS KT SETTING                    |
| 2677 | 012236 | 000000 | EACUR: .WORD  | 0      | : CURRENT SETTING OF EA BITS                              |
| 2678 | 012240 | 000000 | DATAD: .WORD  | 0      | : CURRENT DATA ADDR                                       |
| 2679 | 012242 | 000000 | BFBAPH: .WORD | 0      | : CURRENT PHYSICAL BUFFER BASE ADDR                       |
| 2680 | 012244 | 000000 | BFADPH: .WORD | 0      | : CURRENT PHYSICAL DATA ADDR                              |
| 2681 | 012246 | 000000 | SPWPHY: .WORD | 0      | : PHYSICAL ADDR OF SPW START                              |
| 2682 | 012250 | 000000 | DEV: .WORD    | 0      | : CURRENT IBM DEVICE NO.                                  |
| 2683 | 012252 | 000400 | CMD: .WORD    | 400    | : CURRENT IBM COMMAND WITH PARITY BIT                     |
| 2684 | 012254 | 000000 | ERRPC: .WORD  | 0      | : PC WHERE ERROR WAS DETECTED                             |
| 2685 | 012256 | 000000 | ERRID: .WORD  | 0      | : ERROR NUMBER  |
| 2686 | 012260 | 000000 | ERRMS: .WORD  | 0      | : ADDR OF UNIQUE ERROR MESSAGE                            |
| 2687 | 012262 | 000000 | EAERR: .WORD  | 0      | : EABITS ON ERROR   |
| 2688 | 012264 | 000000 | RECORD: .WORD | 0      | : SWITCH TO CONTROL DEV REC RECORDING                     |
| 2689 | 012266 | 000000 | FRSTD: .WORD  | 0      | : FIRST DEVICE RECOGNIZED SW                              |
| 2690 | 012270 | 000000 | FRSTDV: .WORD | 0      | : FIRST DEVICE RECOGNIZED NUMBER                          |
| 2691 | 012272 | 000000 | DEVCT: .WORD  | 0      | : FIRST DEVICE GROUP COUNT                                |
| 2692 | 012274 | 000000 | DEVMSS: .WORD | 0      | : DEVICE MESSAGE PRINTED SW                               |
| 2693 | 012276 | 000000 | ACTU: .WORD   | 0      | : ACTUAL DATA   |
| 2694 | 012300 | 000000 | EXP: .WORD    | 0      | : EXPECTED DATA   |
| 2695 | 012302 | 000000 | DEVFND: .WORD | 0      | : = 1 IF DEVICE EVER RECOGNIZED                           |
| 2696 | 012304 | 000000 | SPWMVD: .WORD | 0      | : = 1 IF SPW TAPLE IN ALTERNATE LOCATION                  |

MD-11-CZDXJ-C DX11-B ADDRESSING TEST MACY11 30A(1052) 07-JUN-82 16:29 PAGE 61  
 CZDXJC.P11 07-JUN-82 16:26 MODULE 3.3 REPORT ERRORS

```

2697
2698 012306 ERR1: .NLIST BIN
2699 012314 .ASCIZ /FAST NPR DATA ERROR/
2700 012322
2701 012330
2702 012332 ERR2: .ASCIZ /NO DX INTERRUPT/
2703 012340
2704 012346
2705
2706 012352 ERR3: .ASCIZ /DX ADDRESSING ERROR, DATA XFER/
2707 012360
2708 012366
2709 012374
2710 012402
2711 012410
2712 012411 ERR4: .ASCIZ /SPW ACCESS, ADDR ERROR/
2713 012416
2714 012424
2715 012432
2716 012440 ERR5: .ASCIZ /NO IBM DEV ADR RECOGNIZED. CHECK A20 MODULE/
2717 012446
2718 012454
2719 012462
2720 012470
2721 012476
2722 012504
2723 012512
2724 012514 ERR6: .ASCIZ /DST ACCESS ERROR/
2725 012522
2726 012530
2727 012535 T4MSG: .ASCIZ /TST3 MUST BE RUN FIRST FOR TST 4 TO WORK/
2728 012542
2729 012550
2730 012556
2731 012564
2732 012572
2733 012600
2734 012606 HEADER: .ASCII 'MD-11-DZDXJC DX11B MEMORY ADDRESSING TEST'<15><12>
2735 012614
2736 012622
2737 012630
2738 012636
2739 012644
2740 012652
2741 012660
2742 012662 .ASCII 'TYPE: <D>, DEFAULT PARAMETERS'<15><12>
2743 012670
2744 012676
2745 012704
2746 012712
2747 012720
2748 012721 .ASCII '
2749 012726 <P>, PREVIOUS PARAMETERS'<15><12>
2750 012734
2751 012742
2752 012750

```

MD-11-CZDXJ-C DX11-B ADDRESSING TEST MACY11 30A(1052) 07-JUN-82 16:29 PAGE 62  
 CZDXJC.P11 07-JUN-82 16:26 MODULE 3.3 REPORT ERRORS

|      |        |         |        |                                      |
|------|--------|---------|--------|--------------------------------------|
| 2753 | 012754 | .ASCII  | '      | <S>, SELECT PARAMETERS'<15><12>      |
| 2754 | 012762 |         |        |                                      |
| 2755 | 012770 |         |        |                                      |
| 2756 | 012776 |         |        |                                      |
| 2757 | 013004 |         |        |                                      |
| 2758 | 013005 | .ASCIZ  | '      | <N>, START AT THIS TEST NO.'<15><12> |
| 2759 | 013012 |         |        |                                      |
| 2760 | 013020 |         |        |                                      |
| 2761 | 013026 |         |        |                                      |
| 2762 | 013034 |         |        |                                      |
| 2763 | 013042 |         |        |                                      |
| 2764 | 013044 | INVM:   | .ASCIZ | 'INVALID ENTRY'                      |
| 2765 | 013052 |         |        |                                      |
| 2766 | 013060 |         |        |                                      |
| 2767 | 013062 | INVPRM: | .ASCIZ | 'NEED MORE PARMS'                    |
| 2768 | 013070 |         |        |                                      |
| 2769 | 013076 |         |        |                                      |
| 2770 | 013102 | PROMPT: | .ASCIZ | 'D,P,S,N? '                          |
| 2771 | 013110 |         |        |                                      |
| 2772 | 013114 | TESTN:  | .ASCIZ | 'STARTING TEST NO. '                 |
| 2773 | 013122 |         |        |                                      |
| 2774 | 013130 |         |        |                                      |
| 2775 | 013136 |         |        |                                      |
| 2776 | 013137 | BASEA:  | .ASCIZ | 'BASE ADDRESS '                      |
| 2777 | 013144 |         |        |                                      |
| 2778 | 013152 |         |        |                                      |
| 2779 | 013155 | VECTA:  | .ASCIZ | 'VECTOR ADDRESS '                    |
| 2780 | 013162 |         |        |                                      |
| 2781 | 013170 |         |        |                                      |
| 2782 | 013175 | PRILV:  | .ASCIZ | 'DX PRIORITY LEVEL '                 |
| 2783 | 013202 |         |        |                                      |
| 2784 | 013210 |         |        |                                      |
| 2785 | 013216 |         |        |                                      |
| 2786 | 013220 | SETSW:  | .ASCIZ | 'SET SWITCHES '                      |
| 2787 | 013226 |         |        |                                      |
| 2788 | 013234 |         |        |                                      |
| 2789 | 013236 | BELL:   | .ASCIZ | <207> 'END PASS'                     |
| 2790 | 013244 |         |        |                                      |
| 2791 | 013250 | BADP:   | .ASCIZ | 'INPUT PARAMETER OUTSIDE LIMITS'     |
| 2792 | 013256 |         |        |                                      |
| 2793 | 013264 |         |        |                                      |
| 2794 | 013272 |         |        |                                      |
| 2795 | 013300 |         |        |                                      |
| 2796 | 013306 |         |        |                                      |
| 2797 | 013307 | ERPC:   | .ASCIZ | 'ERROR PC: '                         |
| 2798 | 013314 |         |        |                                      |
| 2799 | 013322 | ERNO:   | .ASCIZ | ' ERR NO.: '                         |
| 2800 | 013330 |         |        |                                      |
| 2801 | 013336 | ERTN:   | .ASCIZ | 'TEST NO.: '                         |
| 2802 | 013344 |         |        |                                      |
| 2803 | 013351 | ERADR:  | .ASCIZ | 'VIRT ADDR OF BUFFER BASE: '         |
| 2804 | 013356 |         |        |                                      |
| 2805 | 013364 |         |        |                                      |
| 2806 | 013372 |         |        |                                      |
| 2807 | 013400 |         |        |                                      |
| 2808 | 013404 | ERADPH: | .ASCIZ | 'PHY ADDR OF BUFFER: '               |

2809 013412  
 2810 013420  
 2811 013426  
 2812 013431 EABIT: .ASCIZ 'EA BITS IN OCT: '  
 2813 013436  
 2814 013444  
 2815 013452 DATAA: .ASCIZ 'VIRT ADDR OF ACCESS ERROR: '  
 2816 013460  
 2817 013466  
 2818 013474  
 2819 013502  
 2820 013506 DEVMS: .ASCIZ '1ST IBM DEV RCGNZD. OCTAL: '  
 2821 013514  
 2822 013522  
 2823 013530  
 2824 013536  
 2825 013544 DEVCMS: .ASCIZ 'SIZE OF 1ST DEV GROUP: '  
 2826 013552  
 2827 013560  
 2828 013566  
 2829 013574  
 2830 013575 ACTUMS: .ASCIZ 'ACTUAL: '  
 2831 013602  
 2832 013607 EXPMS: .ASCIZ ' EXPECTED: '  
 2833 013614  
 2834 013622  
 2835 013625 CRLF1: .ASCIZ / /<15><12>

2836  
 2837 013632 .LIST BIN  
 2838 .EVEN  
 2839 013632 012 ;INBUF MUST BE ON AN EVEN WORD BOUNDARY  
 2840 014000 INBUF: .BYTE 12 ;KEYBOARD INPUT BUFFER  
 2841 014000 SPW: .=14000  
 2842 ;START OF STATIONARY SPW TABLE USED ONLY TO  
 2843 014000 000002 .WORD 2 ;FIELD UNEXPECTED ERRORS.  
 2844 014002 000002 .WORD 2 ;STATUS OF UNIT CHECK WITH NO DST ACCESS  
 2845 014004 000002 .WORD 2 ;STATUS OF UNIT CHECK WITH NO DST ACCESS  
 2846 014006 000002 .WORD 2 ;STATUS OF UNIT CHECK WITH NO DST ACCESS  
 2847 014010 000002 .WORD 2 ;STATUS OF UNIT CHECK WITH NO DST ACCESS  
 2848 014012 000002 .WORD 2 ;STATUS OF UNIT CHECK WITH NO DST ACCESS  
 2849 014014 000002 .WORD 2 ;STATUS OF UNIT CHECK WITH NO DST ACCESS  
 2850 014016 000002 .WORD 2 ;STATUS OF UNIT CHECK WITH NO DST ACCESS  
 2851 014020 000002 .WORD 2 ;STATUS OF UNIT CHECK WITH NO DST ACCESS  
 2852 014022 000002 .WORD 2 ;STATUS OF UNIT CHECK WITH NO DST ACCESS  
 2853 014024 000002 .WORD 2 ;STATUS OF UNIT CHECK WITH NO DST ACCESS  
 2854 014026 000002 .WORD 2 ;STATUS OF UNIT CHECK WITH NO DST ACCESS  
 2855 014030 000002 .WORD 2 ;STATUS OF UNIT CHECK WITH NO DST ACCESS  
 2856 014032 000002 .WORD 2 ;STATUS OF UNIT CHECK WITH NO DST ACCESS  
 2857 014034 000002 .WORD 2 ;STATUS OF UNIT CHECK WITH NO DST ACCESS  
 2858 014036 000002 .WORD 2 ;STATUS OF UNIT CHECK WITH NO DST ACCESS  
 2859 014040 000002 .WORD 2 ;STATUS OF UNIT CHECK WITH NO DST ACCESS  
 2860 014042 000002 .WORD 2 ;STATUS OF UNIT CHECK WITH NO DST ACCESS  
 2861 014044 000002 .WORD 2 ;STATUS OF UNIT CHECK WITH NO DST ACCESS  
 2862 014046 000002 .WORD 2 ;STATUS OF UNIT CHECK WITH NO DST ACCESS  
 2863 014050 000002 .WORD 2 ;STATUS OF UNIT CHECK WITH NO DST ACCESS  
 2864 014052 000002 .WORD 2 ;STATUS OF UNIT CHECK WITH NO DST ACCESS











MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 68  
MODULE 3.3 REPORT ERRORS

|      |        |        |       |   |  |
|------|--------|--------|-------|---|--|
| 3089 | 014754 | 000002 | .WORD | 2 | ;STATUS OF UNIT CHECK WITH NO DST ACCESS |
| 3090 | 014756 | 000002 | .WORD | 2 | ;STATUS OF UNIT CHECK WITH NO DST ACCESS |
| 3091 | 014760 | 000002 | .WORD | 2 | ;STATUS OF UNIT CHECK WITH NO DST ACCESS |
| 3092 | 014762 | 000002 | .WORD | 2 | ;STATUS OF UNIT CHECK WITH NO DST ACCESS |
| 3093 | 014764 | 000002 | .WORD | 2 | ;STATUS OF UNIT CHECK WITH NO DST ACCESS |
| 3094 | 014766 | 000002 | .WORD | 2 | ;STATUS OF UNIT CHECK WITH NO DST ACCESS |
| 3095 | 014770 | 000002 | .WORD | 2 | ;STA US OF UNIT CHECK WITH NO DST ACCESS |
| 3096 | 014772 | 000002 | .WORD | 2 | ;STATUS OF UNIT CHECK WITH NO DST ACCESS |
| 3097 | 014774 | 000002 | .WORD | 2 | ;STATUS OF UNIT CHECK WITH NO DST ACCESS |
| 3098 | 014776 | 000002 | .WORD | 2 | ;STATUS OF UNIT CHECK WITH NO DST ACCESS |
| 3099 | 015000 |        |       |   | ;START OF STATIONARY TUMBLE TABLE        |
| 3100 | 016000 |        |       |   |  |
| 3101 |        | 000001 |       |   |  |

TT:  
ENDALL: .END







MD-11-CZDXJ-C DX11-B ADDRESSING TEST MACY11 30A(1052) 07-JUN-82 16:29 PAGE 73  
 CZDXJC.P11 07-JUN-82 16:26 CROSS REFERENCE TABLE -- USER SYMBOLS

|         |          |       |       |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
|---------|----------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|------|------|------|--|--|--|
| KIPAR5= | 172352   | 606#  | 2304* |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| KIPAR6= | 172354   | 607#  | 2387* | 2390* | 2414* | 2415  |       |       |       |       |       |      |      |      |  |  |  |
| KIPAR7= | 172356   | 608#  |       |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| KIPDR0= | 172300   | 590#  |       |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| KIPDR1= | 172302   | 591#  |       |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| KIPDR2= | 172304   | 592#  |       |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| KIPDR3= | 172306   | 593#  |       |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| KIPDR4= | 172310   | 594#  |       |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| KIPDR5= | 172312   | 595#  |       |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| KIPDR6= | 172314   | 596#  |       |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| KIPDR7= | 172316   | 597#  |       |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| KT      | 012172   | 778   | 1366* | 1385* | 2299  | 2384  | 2410  | 2537  | 2659# |       |       |      |      |      |  |  |  |
| LF      | = 000012 | 474#  | 1701  | 1707  |       |       |       |       |       |       |       |      |      |      |  |  |  |
| LOCK    | 012220   | 737*  | 2487  | 2495* | 2497* | 2670# |       |       |       |       |       |      |      |      |  |  |  |
| LOCKAD  | 012222   | 2005* | 2491  | 2671# |       |       |       |       |       |       |       |      |      |      |  |  |  |
| LOOP1   | 006506   | 1910# | 1948  |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| LOOP2   | 007230   | 2053# | 2090  |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| LOOP3   | 007516   | 2143# | 2186  |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| LOOP4   | 010250   | 2286# | 2348  |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| LOP2    | 007256   | 2057# | 2084  |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| LOP3    | 007552   | 2150# | 2181  |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| LOP4    | 010364   | 2311# | 2339  |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| LPAR    | 012134   | 2636# |       |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| LPRIOR  | 012150   | 1566  | 2643# |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| LTESTN  | 012134   | 1545  | 2637# |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| LUBA    | 012140   | 1552  | 2639# |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| LVEC    | 012144   | 1559  | 2641# |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| MMAX    | 012166   | 1367* | 2443  | 2657# |       |       |       |       |       |       |       |      |      |      |  |  |  |
| MMVEC   | = 000250 | 579#  |       |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| MSIZE   | 012164   | 1368* | 1371* | 1376* | 2441  | 2457  | 2656# |       |       |       |       |      |      |      |  |  |  |
| NEXTB   | 010750   | 2373  | 2402# | 2424  | 2436  | 2445  |       |       |       |       |       |      |      |      |  |  |  |
| PARMV   | 012210   | 826   | 844   | 853*  | 898   | 914*  | 2666# |       |       |       |       |      |      |      |  |  |  |
| PHYMAX  | 012226   | 1391* | 1392* | 1393* | 1394* | 1395* | 1396* | 1397* | 1398* | 2430  | 2673# |      |      |      |  |  |  |
| PIRQ    | = 177772 | 480#  |       |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| PIRQVE  | = 000240 | 574#  |       |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| PRILV   | 013175   | 993   | 2782# |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| PRINT   | 005372   | 765   | 768   | 771   | 774   | 781   | 829   | 847   | 883   | 900   | 945   | 959  | 970  | 981  |  |  |  |
|         |          | 992   | 1572  | 1596# | 2274  | 2504  | 2508  | 2511  | 2514  | 2518  | 2523  | 2526 | 2531 | 2534 |  |  |  |
|         |          | 2539  | 2542  | 2545  | 2552  | 2557  | 2560  | 2575  | 2578  | 2581  | 2584  | 2587 |      |      |  |  |  |
| PROMPT  | 013102   | 946   | 2770# |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| PRO     | = 000000 | 497#  |       |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| PR1     | = 000040 | 498#  |       |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| PR2     | = 000100 | 499#  |       |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| PR3     | = 000140 | 500#  |       |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| PR4     | = 000200 | 501#  |       |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| PR5     | = 000240 | 502#  |       |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| PR6     | = 000300 | 503#  |       |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| PR7     | = 000340 | 504#  |       |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| PS      | = 177776 | 477#  | 478   |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| PSW     | = 177776 | 478#  |       |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| PTYGEN  | 010034   | 2152  | 2211# | 2280  | 2311  |       |       |       |       |       |       |      |      |      |  |  |  |
| PTY2    | 010042   | 2214# | 2218  |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| PTY3    | 010050   | 2215  | 2217# |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| PTY4    | 010066   | 2220  | 2223# |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| PTY5    | 010072   | 2222  | 2224# |       |       |       |       |       |       |       |       |      |      |      |  |  |  |
| PWRVEC  | = 000024 | 569#  | 713*  | 714*  | 1847* | 1848* | 1857* | 1863* | 1875* | 1876* |       |      |      |      |  |  |  |









MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 77  
CROSS REFERENCE TABLE -- USER SYMBOLS

|          |          |       |       |      |       |       |      |      |      |      |       |       |       |      |  |  |  |  |  |
|----------|----------|-------|-------|------|-------|-------|------|------|------|------|-------|-------|-------|------|--|--|--|--|--|
| \$TYPE   | 005444   | 1635# | 1823  | 1831 |       |       |      |      |      |      |       |       |       |      |  |  |  |  |  |
| \$TYPEC  | 005614   | 1187  | 1656  | 1663 | 1670  | 1675# |      |      |      |      |       |       |       |      |  |  |  |  |  |
| \$TYPEX  | 005744   | 1695  | 1700  | 1702 | 1705# |       |      |      |      |      |       |       |       |      |  |  |  |  |  |
| \$TYPOC  | 005772   | 1738# | 1832  |      |       |       |      |      |      |      |       |       |       |      |  |  |  |  |  |
| \$TYPON  | 006006   | 1737  | 1740# | 1834 |       |       |      |      |      |      |       |       |       |      |  |  |  |  |  |
| \$TYPOS  | 005746   | 1733# | 1833  |      |       |       |      |      |      |      |       |       |       |      |  |  |  |  |  |
| \$XOFF = | 000023   | 1680  | 1707  |      |       |       |      |      |      |      |       |       |       |      |  |  |  |  |  |
| \$XON =  | 000021   | 1073  | 1687  | 1694 | 1707  |       |      |      |      |      |       |       |       |      |  |  |  |  |  |
| \$OFILL  | 006171   | 1734* | 1738* | 1748 | 1783# |       |      |      |      |      |       |       |       |      |  |  |  |  |  |
| .        | = 016000 | 619#  | 623#  | 632  | 633#  | 635#  | 637# | 638# | 645# | 679  | 708   | 1040  | 1044# | 1045 |  |  |  |  |  |
|          |          | 1046  | 1288# | 1289 | 1290  | 1297# | 1351 | 1707 | 1859 | 1883 | 2837# | 2840# |       |      |  |  |  |  |  |



MD-11-CZDXJ-C DX11-B ADDRESSING TEST  
 CZDXJC.P11 07-JUN-82 16:26

MACY11 30A(1052) 07-JUN-82 16:29 PAGE 80  
 CROSS REFERENCE TABLE -- MACRO NAMES

|         |    |      |      |
|---------|----|------|------|
| .SRAND  | 1# |      |      |
| .SRDDE  | 1# |      |      |
| .SRDOC  | 1# | 453# | 1298 |
| .SREAD  | 1# | 453# | 1037 |
| .SR2AZ  | 1# |      |      |
| .SSAVE  | 1# |      |      |
| .SSB2D  | 1# |      |      |
| .SSB2O  | 1# |      |      |
| .SSCOP  | 1# |      |      |
| .SSIZE  | 1# | 453# | 1405 |
| .SSUPR  | 1# |      |      |
| .STRAP  | 1# | 453# | 1796 |
| .STYPB  | 1# |      |      |
| .STYPD  | 1# |      |      |
| .STYPE  | 1# | 453# | 1618 |
| .STYPO  | 1# | 453# | 1708 |
| .\$4OCA | 1# |      |      |
| .1170   | 1# |      |      |

. ABS. 016000 000

ERRORS DETECTED: 0

CZDXJC,CZDXJC/SOL/CRF/NL:TOC=SYSMAC.C5,CZDXJC.P11  
 RUN-TIME: 14 12 .8 SECONDS  
 RUN-TIME RATIO: 54/28=1.9  
 CORE USED: 33K (65 PAGES)